The IBM System/360 Line

Document Number 0867-5309

August 8, 2025

Evie Cooper

Foreword

Some amount of the information in this guide originated in Dave Morton's excellent guide named "IBM Mainframe Operating Systems: Timeline and Brief Explanation For the IBM System/360 and Beyond." Most of my knowledge lies in VM and VSE, whereas Dave's seems to be mostly in MVS. I hope this guide is useful to the mainframe hobbyist community, and thanks to Dave for producing the original this booklet was inspired by! In addition, I sourced knowledge from the following sources:

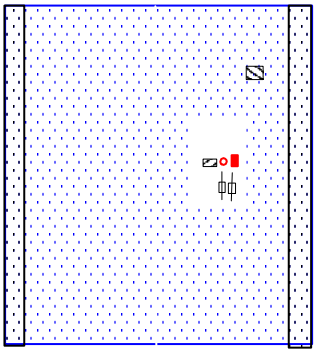
- Enterprise Systems Media z/Journal
- Lynn Wheeler's IBMVM Listserv post archive
- Various manuals on Bitsavers
- The CBT Tape archive
- "IBM Mainframe Public Domain Software Collection"
- Paul Pierce's System/360 website
- Melinda Varian's *VM and the VM Community*
- IBM program product announcements
- linas.org's Linux ESA/390 page
- My real-life experiences running these systems
- Marist Linux/390 webpage
- Argecy Corp's website

I will say that while I have tried to ensure the information presented in this guide is as accurate as humanly possible, there will inevitably be some errors. I'm a bit uncertain on some details surrounding the very early S/360 OSes, and I have placed disclaimers where applicable. Note that I was reading quite a bit of conflicting information on specifically BOS/360, so read with caution and please alert me at once if you have a plausible correction (with evidence, please!).

This document was formatted on SCRIPT/VS 1.4.0 with BookMaster Release 4, on z/VM 4.4, on a Multiprise 3000 System/390, powered half by solar panels sitting on my balcony.

To contact me, email me at ***wec@bam.moe***

AI statement: I did not use any generative AI, deep research tools, chatbots, or anything; I only used my hands and the sources listed to produce this document.



The System/360 Mainframe: Hardware and Software History

Contents

Figures

Mainframe Hardware

Origins

The road to commercial computing was fraught with confusion, failures, and false starts. By the mid-1960s, computers as a machine that a business could purchase were starting to develop serious customer bases. The so-called “third generation” of computers featured integrated circuits. In the mid-1960s, you could get a computer from a variety of now-forgotten names:

- IBM, who sold the 7090 (a large system), 1401 (a small system), and 1130 (targeted for low-cost markets); all of these systems were replaced with the System/360
- Burroughs, whose B5000 (and later B5500/B5700) led the way in computers that did not require heavy assembly language usage to program; the OS (MCP) still lives on today
- UNIVAC, whose 1107 and 1108 gained popularity as batch systems, lives on today as OS 2200 from Unisys
- General Electric, whose 600 series (specifically the 645) was the birthplace of the highly-influential Multics system; otherwise, GECOS was the OS used for batch loads -- GCOS lives on today (version 7 and 8, who are somewhat different) as emulated systems
- RCA (Radio Corporation of America), who diversified into computing and had reasonable success with their 501 series (their later Spectra 70 was compatible with the System/360)
- CDC (Control Data Corporation), whose coming 6000 series would define supercomputing in the mid-1960s
- NCR (National Cash Register, though this acronym is never expanded), whose computers in the early 1960s saw some success in smaller customers
- DEC (Digital Equipment Corporation), whose PDP (Programmable Data Processors) were popular in academia

What is a mainframe?: Though usually a colloquial term that refers to some kind of IBM-brand computer (at least today), a mainframe generally refers to a large computer system with several distinct characteristics:

- A distributed processing nature. The CPU need not directly drive peripherals, as it is attached to a variety of coprocessors whose purpose is to driver peripherals for the host computer CPU(s).
- Extensive teleprocessing capabilities. Mainframes can support many users (sometimes in the high hundreds range), and, historically, this necessitated the creation of some kind of network to connect terminals to terminal controllers. These networks were often interlinked with other networks (in the past, these were things like X.25 networks; later, it was the Internet)
- Large storage arrays. In the early days, this often meant entire floors of buildings being filled with disk drives; today, it means several racks of disks on larger machines, ranking in large capacities.
- Purpose-built fault recovery. Mainframes have historically sought high uptime, and this is still true for today. Paths to peripherals are often multipathed, and devices may be duplicated for extra reliability. The CPUs of most mainframes are sometimes duplicated too.
- An operations team. Being a large and complex system (often running a suitably large and complex OS), these large computers are often kept by a number of operators.

Often times, people take computing today for granted. You can go to the computer store, and you can buy a CPU from either AMD or Intel -- and any programs you write will work on either CPU, provided the OS ran on both is the same (or is compatible). However, this was not always the case! If you bought a computer in the *1960s*, this were quite a bit different. You found yourself in a landscape of "one-model-machines" -- machines that, while certainly rather capable, presented a hefty challenge when it came to porting software. There was usually only one (or a few) model(s) of a computer in each CPU architecture, and this presented a rather difficult issue.

In that era, totally different computers were *everywhere*, as seen on the list above. If you were, for example, buying a computer to print payroll checks and do other accounting tasks, an IBM 1401 would be fine. If you were doing research or engineering tasks, that 1401 would be useless -- you'd instead find yourself juxtaposed between a choice between an IBM 7040, DEC PDP-1, and a bunch of other computers. There was no reason to buy IBM, and IBM knew this.

Because every CPU architecture in those days was different, you'd need to rewrite all of your programs when you upgraded computers (as you might be upgrading from an IBM 7090 to a UNIVAC 1107)! IBM saw this void, and wondered if they could build a line of machines (rather than just one machine) with a common CPU architecture. That way, if someone bought a basic System/360, they could upgrade it later on and not have to rewrite any code.

While this certainly sounds silly today, this was seen as somewhat revolutionary back then. Likewise, the OSes found on it had to be portable across machines -- this will be the primary focus of this booklet. With that out of the way, let's begin!

The System/360

The 360 was an early computer, and it was not uncommon in that era to find systems engineers that were the direct trained descendants of the original generation of computing pioneers. One of Howard Aiken's students was Dr. Frederick Brooks, who would later become the director of IBM's System Architecture. When the 360 project came up, Dr. Brooks was selected to be the lead of the whole project -- a tall order and a difficult task for anyone in that era. He would later go on to win over 20 awards/honors, and even won the National Medal of Technology in 1985. The principal architect would be Dr. Gene Amdahl, already known for his work on earlier (and smaller) IBM computers.

The 360 project was also unbelievably expensive in that era -- it cost IBM 5 billion dollars; this put the project on the map with giants of the era like the Apollo Project (which was NASA's project to put men on the Moon, of which the salaries of the people involved was $5 billion too). Actually, the 360 project completely dwarfed Project Gemini, the direct precursor to the Apollo Project, by 4 billion dollars. Needless to say, the 360 project needed to deliver.

Competitors to IBM were rapidly catching up to IBM with similar machines to the 1401, at a lower cost. IBM knew this, and development of the 360 coalesced around the early 60s and the hardware was officially announced in April 1964. In 1965, the first 360 machines were shipped to customers. This was unbelievably fast, and for good reason -- 60,000 people were employed to make this possible. A variety of models were launched then, ranging from the tiny Model 20 (which itself was a subset of all other 360s), the entry-level Model 30, and the big powerful Model 75. The Model 91 would come later in January 1966, and it screamed at 16.6 MIPS (million instructions per second, a common measurement of mainframe CPU speed). All the while, the same software and I/O devices could be used on all of these machines!

Possibly the most influential model of the System/360 line was the Model 67, with its optional virtual memory hardware, that would set the stage for the online timesharing (and not batch-oriented) model of the successors to the 360. The Model 67 found itself embraced by the academic world, with many universities and research labs (like Bell Labs) adopting it for their computing needs (and modifying it to suit their specific needs, too). The so-called "DAT Box" (unrelated to the feature found on the 370/Advanced Functions machines that launched later) was highly sought-after for these machines.

The System/370

Original S/370 and 370/AF: While the 360 was certainly successful, a successor was definitely in order. The System/370 was announced in June 1970, with the first machines showing up at customer shops in 1971. By 1972, the unusual lack of virtual memory hardware on the original 370s caused IBM to launch the 370/Advanced Function line -- these contained an infamous "DAT box" (dynamic address translation, the name for the memory man-

agement unit on the 360; note that the 370's was different in design). These introduced new generations of the 360 OSes (the original 370 announcement just ran the 360 OSes, nothing new for the 370 itself), and upgrades for earlier 370s that did not have the DAT box were available (the Model 155 and Model 165).

3033: The 370, being derived from the 360, had a rather famous 16 megabyte storage limit (storage being the IBM term for what other computer platforms called "RAM"). As the 370 evolved throughout its life, the first major gain was something called ECPS (Extended Control Program Support), which was hardware acceleration to certain OS functions. Seeing that the address space limit was starting to become an issue. The 3033 (the 303x series was the successor to the second-generation 370/Advanced Function machines) introduced a function called "Dual Address Space" -- the MVS operating system (which will be discussed later) could use this to effectively double the system's storage capacity (with plenty of caveats, of course).

3081/3083, 370/XA: Needing an actual address space increase that wasn't a bank-switching technique, the 3033 and 3081 (launched in October 1981) added a feature called "Extended Real Addressing" -- for the first time, it was now possible to use 64 MB of real storage (the physical address bus now had 26 bits, rather than the infamous 24-bit limit). When the 3081 and 3083 launched, they also contained a totally new machine operating mode: 370/Extended Architecture (commonly called XA). This expanded the address bus up to 31 bits (both real and virtual; the 26-bit 370 EAS systems were still limited to 24-bit virtual storage spaces), completely overhauled the I/O architecture, and changed how the system booted.

4300: During the early 80s, there were a variety of smaller air-cooled System/370s made. Here they are:

- 4321, announced 1979 and withdrawn in 1981, 8 MB max storage
- 4331, announced 1979 and withdrawn in 1981: 8 MB max storage
- 4341, announced 1979 and withdrawn in 1986: 16 MB max storage
- 4361, announced 1983 and withdrawn in 1987: 12 MB max storage

Paradoxically, the 4341 was not only more performant than the 4361, but had more storage! This was because the 4331/4361 were developed in Germany (by the VSE developers; as such, these machines had many microcode features that greatly increased VSE performance through a mechanism called ECPS:VSE), and the 4341/4381 were developed in New York (and targeted totally different workloads). All of these models utilized a 3270 as a console in lieu of the earlier 3215s (see the chapter below for more information on these). The 4331 and 4361 had an integrated DASD controller that drove a string of either 3310s or 3370s; these were FBA DASDs and no CKD DASDs were available without an external channel-attached controller on this machine.

3090/4381: After the 370/XA launched, IBM announced Enterprise Systems Architecture/370 on the 3090E and 4381 machines in February 1988. This added additional addressing modes, 16 access registers (32 bits wide), and a means for a program to use several virtual address spaces; this would later evolve in 1990 with the ESA/390 architecture.

9370: *Note: the 9371 is not discussed here as it is different from the 9373/9375/9377; see the PC Mainframes section for information on it*

While IBM was doing great in the high-end systems department with the high-performance (and high-dollar) System/370s and 370/XAs, the midrange department was still cornered by machines like DEC's VAX line. IBM saw this, and too wanted to break through in that market. Designed by IBM Endicott in 1985 and released in 1986, the IBM 9370 line was the first major computer that consisted of a CPU emulating a different CPU (in this case, an IBM 801 RISC CPU emulating a low-end System/370). This machine was intended to be a "superminicomputer", suitable for engineering, scientific, and office use. These machines came in 3 different model classes, each progressively gaining more support (with the high-end 9377s being able to run MVS/SP). Sadly, these machines did not sell nearly as well as IBM thought they would, and have been relegated to "failed products" lists.

The System/390

ES/9000: The IBM mainframe line continued to grow consistently over time! When it was time to unveil the next upgrade to the 370 in September 1990, IBM opted to launch a totally new architecture and line of machines. The System/390, much like the System/370, would be a machine for the 1990s (like the 370 was a machine for the 1970s). The ESA/390 architecture would succeed the ESA/370 architecture, and the first new machine to bear it was the Enterprise System/9000 (ES/9000) line. Gone were the old massive thick device connection channel cables -- the System/390 introduced a fiber upgrade called ESCON (Enterprise Systems Connection). The ES/9000 line featured rack-mountable machines, smaller air-cooled mainframes, and massive water-cooled mainframes. While the ESA OSes (VM/ESA, MVS/ESA, VSE/ESA) could run on the ESA/370 3090 or 4381, the ES/9000 line was faster, smaller, and cheaper.

The ES/9000 series featured 18 different models, as mentioned. The original 9021s were essentially converted 3090J processors, but expanded with optional ESCON, sysplex coupling facility support, and the addition of a new cryptographic coprocessor (which accelerated DES). Later models could have a Vector Facility array processor, the cryptographic accelerator (the Integrated Cryptographic Feature), and a data compression accelerator. In addition, all of the ES/9000s could be logically partitioned to subdivide the machine's hardware resources to run multiple OSes simultaneously -- this also meant that there was a different method of device assignment (see the section at the back for SA IOCP). The models fell into three main groups:

- 9021, a large watercooled machine that would replace the 3090
- 9121, an aircooled machine that would replace the 4381
- 9221, a rackmounted aircooled machine that replaced the 9370

It should be noted that the 9221 was originally a System/370-only machine (on models 120/130/150) and was later upgraded with ESA/390 capability with the ESA Option, shipped July 1991. The same team that designed the CPUs in the 9221 used their experience later to design the CMOS 9672s.

9672: The ES/9000 would later be replaced by a machine using a more advanced digital logic technology (CMOS, in lieu of ECL, used by the ES/9000s). These machines bore the model number *9672*. There were six generations of these machines, and a new one came out every year from 1994 to 1999. Each one got progressively faster, and gained more storage. The last to support 370-mode guests was the G3. The G5 added support for IEEE 754 floating-point, and the G6 featured one of the fastest CPUs on the market.

The history of the 9672 was marked originally by two machines (both launched in 1994): the *Parallel Transaction Server*, and the *Parallel Query Server* (model number 9673). Later in 1994, these two machines would be remarked as the *Parallel Enterprise Server*

The final 9672s added something called the Open Systems Adapter; this was essentially a directly-attached card that emulated Ethernet and Token Ring interfaces on the mainframe. This would later evolve with the successor machine line (the System Z).

To facilitate intercommunication and form a sysplex, the 9672s were often seen with the 9674 Coupling Facility.

The Multiprise Line

Multiprise 2000: Launched in September 1996, the Multiprise 2000 was essentially a cut-down 9672 G3. It (alongside the G3) was the last machine to support a 370-mode IPL, with all newer machines requiring a native ESA/370 or ESA/390 OS. Being a cut-down 9672 G3, it had half the cores and slower cache; alas, it used nearly the same physical form as the machine it was based from. Disks and tapes were connected via standard ESCON or

parallel (bus and tag) channels. The line was upgraded on October 1997 with an 11%-faster MP2000. The MP2000 had CPU model number 2003.

Application StarterPak 3000: Essentially a Multiprise 2000 in a half-height case, the AS3000 was a short-lived product that paved the way for the later Multiprise 3000 (one of the best-selling System/390s made). The AS3000 did not contain an integrated Support Element; this was provided by a modified IBM Aptiva workstation cabled to the S/390 CPU card cage with a thick cable (not unlike the one seen on the MP3000 later). These had no integrated I/O features outside of a disk array (see the Mainframe DASD chapter below for more information on this), an integrated 4mm DAT tape drive, an OSA, and 3270 consoles on the Support Element. The front of the machine that held the power switch contained the CPU and its support cards, and the back contained a disk array plus 4mm tape drive (all attached via standard parallel SCSI). The AS3000 had CPU serial 3000.

Multiprise 3000: In September 1999, the Multiprise 3000 was launched -- this half-height compact machine featured partially-emulated I/O and a PCI bus. The partially-emulated I/O consisted of the P/390 device emulators, but the system also contained a real disk array (or two, if the user chose such a configuration) that was driven by the S/390 I/O processor: it was considered a "direct system device", wherein the S/390 CPU drives a card present in the machine that emulates CKD DASD with an SSA RAID array. FBA emulation was not possible; the user would have to use the (much slower) emulated I/O devices to achieve this. The CPU itself was derived from either the System/390 9672-R16 (the 7060-H30 and 7060-H50) or the 9672-R26 (the 7060-H70).

There were actually two Multiprise 3000 models that were made but barely sold outside of IBM: the 7060-H55 and 7060-H75 (built on the 9672-R16 and 9672-R26 platforms, respectively). These were different from the public-sold 7060s as they actually had Open Systems Adapters. There were several of these machines sold outside of IBM (one notably to the City of Los Angeles's Board of Water and Power).

The System/390 Integrated Server was launched before the MP3000 in 1998, but it was essentially a P/390 (discussed next). The MP3000 had CPU model number 7060.

PC Mainframes

Throughout the years, IBM has always had a fascination with producing PC-based mainframe-compatibles. These allowed a desktop-sized machine to run mainframe-native software, albeit at *significantly* reduced performance. Here are those machines:

Personal Computer XT/370: Based on the IBM 3270 PC (a PC that could emulate an IBM 3270 terminal), the XT/370 contained a dual-board processor card (featuring two Motorola 68000s) with one of those two boards containing 512 KB of mainframe storage. In addition, a 3270 emulator card was provided for talking to a host mainframe (as this was not entirely a free-standing machine, but more of a "smart terminal" that just so happened to run 370 code on it). The card stack ran VM/PC Control Program, which itself was a cut-down version of VM/SP (which we will definitely discuss later). The XT/370 was extremely slow, registering 0.1 MIPS (thanks in no small part to the fact that two 68000s emulated the 370 CPU, while a modified Intel 8087 provided floating-point support). Nonetheless, it was quite a fancy machine considering the design constraints the engineers faced in October 1983.

Personal Computer AT/370: Launched in 1984, the AT/370 was based on the Personal Computer AT, and was somewhat upgraded. The PC/AT featured 16-bit expansion card slots (the ISA bus), and ran 60% faster than the XT/370.

IBM 7437 VM/SP Technical Workstation: Launched in April 1988, this machine was a PS/2 Model 60/70/80 modified with a Microchannel card bearing a 370 CPU. This machine ran full VM/SP (again, to be discussed later), and was intended to hook up to a 5080 Graphics System (which connected to the machine with another Microchannel card). These machines would run CATIA or CADAM, and it is said that they had the same performance as a single-terminal 9370.

9371: The 9371 was an important stepping stone for what would later evolve into the machines show below; based on the earlier 7437 technology, the 9371 leveraged the 801 RISC CPU (like the other 9370s) but did not have any real I/O devices -- those were emulated by a 386 CPU running OS/2. Models 10 and 12 were MCA cabinets that used that 386 running OS/2 for device emulation (and, as such, the DASD consisted of a MCA SCSI card hooked up to some SCSI disks), whereas Model 14 had more of a PC emphasis and unlocked the OS/2 side for running nearly any application (both DOS and OS/2).

Personal/370 (P/370): Based on the earlier 7437 CPU card, the P/370 (which launched in November 1989) was intended to run any 370 OS (so long as it supported FBA DASD, which meant no MVS) on either a PS/2 (running OS/2) or an RS/6000 (running AIX). Like the 7437, all the I/O devices were emulated, but the user could choose to also add a 370 channel card to connect real I/O devices. The P/370 was much faster than the 7437 and it actually rivalled a low-end 4381 -- it scored about 4 MIPS.

S/390 Processor Card and the P/390: Launched in 1995, the S/390 Processor Card provided an ESA/390 CPU and 32 MB of main storage. This would be inserted into a MCA-slot-bearing PC or RS/6000, but a PCI version came later. There was a third version called the P/390E that featured 256 MB of storage, and a 1 GB P/390 (these are extremely rare) was also made.

S/390 Integrated Server: In November 1998, IBM launched a machine that was a Pentium II server (uniprocessor) with a P/390 card. This so-called Integrated Server used the same chassis as the MP3000, but had less performance; it did, however, have the ESCON and parallel channel capability. Alas, this machine was not very performant and did not sell well.

System Z

Seeing that the ESA/390 architecture was aging in the late 90s and was still handicapped by the 2 GB storage limit (bought about by the 31-bit address space introduced with the 370/XA line), IBM sought a upgrade. This was developed in 1999 as "ESAME" (ESA Modal Extensions), but was eventually renamed to the z/Architecture. The machines from this line are known as System Z, and the first one was the zSeries z900. Based on the earlier 9672 G6, this machine launched in December 2000 with either 12 or 20 processors (16 of which could be defined as central processors, with the leftovers being reserved for I/O duties). The z800 (lower-end than the z900) would launch in 2002 to provide a lower-end model; all of the original zSeries machines before the z14 could still operate in ESA/390 mode. Even then, a superscalar Z CPU did not exist until October 2003 -- the machine that introduced this (the z990) was a complete redesign of the Z CPU (that featured new features like out-of-order execution, NUMA, etc) that was not based off of the earlier 9672 architecture.

The System z9 was released in July 2005, bearing the earlier new Z CP but with many upgrades. The zAAP acceleration engine was added, which accelerated Java applications. There were two lines launched: the z9 Business Class and the z9 Enterprise Class (with the z9EC being the bigger of the two). This would also be the case for the z10, which launched in February 2008 (EC) and October 2008 (BC); the z10 was codenamed the “eclipZ” as its clock speed was supposed to *eclipse* the commonly-available PC servers at the time.

The successor to the z9 and z10 was the zEnterprise line, of which there were 4 submodels. Launched in July 2010, the z196 was the enterprise-class machine (clocked at 5.2 GHz) whereas the z114 was the business-class machine (clocked at 3.8 GHz). The z114 came in two physical varieties: the book-style machines had the same physical configuration as the late-generation 9672 machines, whereas the non-book configuration was native to the z114.

As hardware development rolled on, the z13 launched in 2015 to great avail. Though the CPU clock speed did not meaningfully increase, the cache and memory controller were improved. This generation also added a new variant of the Vector Facility, modelled somewhat after SSE on Intel/AMD x86 CPUs. The steady architectural improvements made around this time started to solidify the System Z line as a rather fast computer, as opposed to the earlier view of “lots of I/O, but a weirdly slow CPU.” The processor elements drove DDR3 RAM, in an array somewhat like a RAID disk array.

Scoring a 30% total-chip (with a 10% per-core) speedup, IBM unveiled the z14 in July 2017. Of note was the massive die shrinkage; the z13 was fabricated on a 22nm die process, and the z14 was fabricated on the then-new 14nm process -- this meant better power efficiency! Each processor package could have up to 10 active cores, and each core supported two-way simultaneous multithreading (readers may be familiar with Intel's HyperThreading scheme, this is similar). Rather than doing what IBM had been doing for years and having dedicated Crypto Express cards, the z14 processor elements now had their own cryptographic processors (called CPACFs)... for each core!

The new vector facility introduced one generation before gained packed-decimal SIMD support (which boosted the performance of COBOL and PL/I applications that were somewhat vectorized), and the memory controller was replaced with a DDR4 one. Around this time, IBM added support for System Z to upstream LLVM/Clang, and this included native support for the z14.

Gaining more and more performance (and also more and more modularity), the z15 was announced in September 2019. Doubling the cache sizes and gaining a new data compression coprocessor (similar to the earlier crypto coprocessors), the number of processing elements per package grew to 12. These sold somewhat well, and were noted to be rather performant compared to their predecessors.

In 2021, when AI was starting to become a big thing that caused every company to shift their marketing focus to it, IBM announced the z16 (with the CPU being called the “Telum”) and made it available for purchase in 2022. This was the first CPU to contain a crypto coprocessor, vector coprocessor, compression coprocessor, and now a neural network coprocessor! These were programmed with instructionns called NNPAs (Neural Network Processing Assists) and were implemented at a very low level. Though nobody rushed out to buy mainframes to train and run AI models on, the reliability of the processors (and their newfound ability to run ML inference during normal z/OS operations) led many Z customers to consider (and often follow through with) adding ML features to various applications.

Seemingly not much more performant than the z16, the z17 (launched in April 2025) gained more support for ML features. In the mid-2020s, AI chatbots and assistants were all the rage; IBM grew their earlier AI support but targeted it for true business workloads (analyzing medical images, computing the risk of lending out a loan, things like that). As if the CPU was not already accelerated enough, the Spyre ML accelerator was released alongside the z16, and was a PCIe card that worked tightly with the host CPU to accelerate ML workloads.

Emulation

Throughout the years, there have been several attempts to emulate the z/Architecture as a logical successor to the P/390. Here are some of the extant emulators:

FLEX-ES: Fundamental Software launched an emulator in 2005 that was tightly-controlled and tightly-licensed, and is the only emulator that IBM has licensed for production workloads. In the mid-2000s, an Intel PC-based server could easily outperform an old System/390 (especially an old low-end ES/9000), so IBM ensured FLEX-ES was somewhat difficult to acquire.

zPDT The zPDT is actually a family of emulators:

- The zPDT proper
- ZD&T Personal Edition
- ZD&T Enterprise Edition

The original zPDT is a PartnerWorld ISV offering that allows customers to run z/OS 1.6, z/VSE 4.1, and z/TPF (all listed versions are "and future versions"). This is intended only for application development, and is not licensed for production workloads; strangely enough, it is against the license terms to publish benchmarks of the zPDTs.

The ZD&T (of which there is also an Enterprise and Personal Edition) is intended as a, as its name expands out to, as a "Z Development and Test environment. Many hobbyists petitioned IBM to produce ZD&T Personal Edition, the result of the infamous “IBM Corporation mainframe hobbyist license” quest of the early 2020s. Like the zPDT, this runs on x86_64 Linux. There is, however, a ZD&T Enterprise Edition, as well as a System Z version that utilities KVM on Linux -- this version allows native speed execution of System Z OSes while emulating storage and network devices entirely on a Z host. The original zPDT has the CPU model number of 1090, and the ZD&Ts have a CPU model number of 1091 (that is, the CPU model number returned to the guest OS).

In 2025, IBM (to much controversy) discontinued the zPDT products in favor of a new cloud-hosted platform. Many people found this to be a tone-deaf move on the part of IBM, but few have an underlying understanding of the new software stack. In reality, the new zPDT replacement is actually a System Z version of the zPDT that runs its workloads under KVM (on Linux). All of the I/O devices are emulated, and, when the “zPDT Enterprise Edition” was released, people discovered that it provided a way to run z/OS on systems without CKD DASD (which, surprisingly, is a major issue in the mainframe community for some people -- some people have mainframes that use the built-in zFCP fibre channel adapter and create emulated FBA DASD drives under z/VM). This is because, as mentioned, all I/O devices are emulated on this System Z zPDT.

Hercules: Seeing a need for a good open-source emulator for the 360 line, Roger Bowler started the Hercules project in 1999. Many other people would contribute to it over the years, and the current most-advanced version is "SDL 4.x Hyperion." It should be noted that Hercules is famously “not legal”
for running production workloads on, and its surprisingly good performance on modern machines has caused IBM to fear it since its inception; there are many stories floating around the community of IBM cracking down heavily on Hercules installs on IBM corporate computers, as well as fearmongering in the community that running hobbyist/personal installs of IBM OSes on Hercules is sure to result in trouble; there are, of course, companies in the wild that have been known to (at least temporarily) run a production workload on Hercules (at least as a failover option). Since the P/390, zPDT/ZD&T, and Hercules all use the same DASD image file form (which is sometimes even found on real DASD arrays, the likes of which can be found in the S/390 IS, the MP3000, and the Bustech DASD emulation products).

Hercules has had an interesting history. Over the years, IBMers have constantly derided it as “piracy,” but many mainframe hobbyists have found it an indispensable tool (as running an early System/370 in your house is simply not a feasible thing to do, and P/370s are very rare). In the early 2000s, many mainframers (in the so-called real world, i.e. real mainframe customers) started to evaluate using the emulator as a disaster recovery platform. Not before long, some companies realized that it outsped low-end System/390 machines (like the Integrated Server 3006) on reasonably-fast computers... and promptly realized they could save a decent bit of money by running their workload on an emulator. Since it used the same DASD image format as all of IBM's emulated DASD systems (like the P/390 and similar), migration was not difficult -- you could also use real SCSI tape drives with Hercules and restore dumps of something like a period-accurate DASD array to the emulated disks.

TurboHercules Controversy: 2009 marked the peak of IBM's efforts to crush Hercules when one of the key Hercules developers (Roger Bowler) founded a company in France called TurboHercules. This enhanced version of Hercules was not only much faster (scoring apparently over 3200 MIPS on an 8-processor Intel Nehalem system clocking in at no more than 3 GHz), but unbelievably cheap. As such, Bowler asked IBM to start licensing z/OS to run on TurboHercules in July 2009. IBM did not take kindly to this; they became to be rather scared, and, in March 2010, TurboHercules began to file a complaint with the EU regulators. Bowler's company argued that IBM was in violation of EU antitrust laws, as IBM was limiting their OSes to only run on IBM hardware.

Surprisingly, Microsoft actually invested in TurboHercules in November 2010. Unfortunately, the EU government's probe into the antitrust allegations was closed in September 2011... with no action whatsoever. The company would end up fizzling out (certainly under major threat from IBM), but TurboHercules's motives live on. Most small VSE systems could likely be migrated to run on Hercules if 21st Century Software (the current owners of the VSE line) would allow customers to do so.

Mainframe Peripherals

DASD Disk Drives

Mainframes were (and still are) well-known for their rather large amounts of disk storage. On the IBM System/360 family, the DASDs (that is, Direct Access Storage Device) are of a CKD (count-key-data) architecture. Unlike the HDDs and SSDs you are likely used to, CKD DASDs used a format wherein each data record stored on the disk could vary in length. This meant efficient utilization of disk surface area and good feature support for the record-oriented storage interfaces present on the mainframe OSes.

S/360 Generation: The first hard disks introduced with the System/360 were adapted from the earlier 7090 peripheral set; these included:

2302, an adapted IBM 1302 disk modified to work with the 2841 disk control unit, and was introduced in 1964. *112 MB.*

2311, a similar device that used a removable disk pack. Announced alongside the 2302, it hooked up to the same control unit and several manufactures built compatible units. *7.5 MB.*

2314, a 9-drive array introduced on 22 April 1965. Looking somewhat like pizza ovens, the removable-back disk design quickly became a success for IBM (and other manufacturers who wished to make competitive products). This disk storage system quickly became the default disk array of the System/360, and its large size combined with high reliability (for the day) set the stage for IBM's future dominance in hard disk technology. IBM later released the 2319, which was a 2314 with one drive assembly missing that was announced in 1970. The purpose of this product is unclear; it had a much lower rental price than the 2314, and could directly attach to the integrated storage controller on the System/370 135 or 145. *29 MB per module.*

S/370 Generation: By the time the System/370 was available, hard disk technology had come a long way. These new disks included:

3330, the new disk drive array for the S/370. Announced in June 1970, the product line also launched with a new controller (the 3830); each controller could operate up to 8 disks. The second model of this product (launched in August 1972) introduced the 3333, which included a disk string control unit (a “string” being a rack of up to 8 drives in a row sharing a controller); this meant the string itself did not have to directly connect to the 3830 control unit. The 3330 line also introduced a real-time error-correcting code, making the product even more reliable than the 2314. IBM launched a double-density version of the 3330 (3330 Model 11) in 1973. *100 MB (single density), 200 MB (double density).*

2319, which was a 2314 with one drive assembly missing that was announced in 1970. The purpose of this product is unclear; it had a much lower rental price than the 2314, and could directly attach to the integrated storage controller on the System/370 135 or 145. *29 MB per module.*

2305, a fixed-head storage drive that was often (incorrectly) called a "drum", introduced in 1970. Unlike other DASDs we will discuss, this used its own model of control unit (the 2835) and was constructed such that there was a fixed head above every track on the disk. The disk assembly looked rather odd, and this device was not replaced by any other similar device. These were very sought-after on the System/370 Advanced Functions systems, where the virtual storage features allowed for paging out to disk; the faster the disk, the faster the machine. The seek times on these are surprisingly low: 2.5 milliseconds on a single-density model, and 5 milliseconds on a double-density model. *5.4 MB (single-density), 11.2 MB (double-density)*

3340, a fixed-disk (i.e. non-removable) disk subsystem famously called the Winchester (with a 30 MB capacity and an average 30 millisecond seek time, people joked that it shared a number with a Winchester rifle) introduced in

March 1973, any of the disks can function as the head of a disk drive string (but the 3344 that was launched alongside the 3340 could not serve as a string head, requiring attachment to a 3340). The disk modules were removable, but the disks were not; that is, when you took the disk apart, you could remove the storage mechanism but not extract the disks from the head assembly. The hardware design this disk pioneered became the standard for later disks (especially those of smaller design): since the heads were always over the disk, the disk could spin down and stop while the heads were parked as close as possible to the center hub of the disk platter stack. This would be replicated by hard disks in PCs later, hence the colloquial term for a PC hard drive that was common in the 80s: a "Winchester disk." Introduced in March 1973, and was compatible with the 3830 Model 2/3 from the earlier 3330. *34.9 MB (single-density), 68.9 MB (double-density).*

3350, an evolution of the 3340 introduced in 1975 -- they even looked the same. Each disk was actually two disk units side-by-side, identifiable because every other unit in a string will have a power switch. There were several models of this disk (all with the same capacity): A2, A2F, B2, B2F, C2, and C2F. The A2 and A2F disks had an additional circuit board mounted in them that allowed them to function as the head of a disk string; this functionality could be enabled or disabled with a switch. All strings would start with an A2/A2F disk, and then could be followed by up to 3 B2/B2F disks ***or*** 2 B2/B2Fs with one C2/C2F. The C2/C2F disks were special: they contained an extra controller that functioned as a hot-spare (the disk units had no such hot-spare functionality; this would only be seen with later disks backed by RAID arrays) if the controller circuitry in the A2/A2F failed. The "F" models featured a fixed head that remained stationary over the first 5 cylinders of storage; the clever systems programmer would position a JES2 checkpoint dataset, TSO swap dataset index region, or MVS paging dataset index region and get a cheap performance boost. *317.5 MB.*

3375, a CKD version of the 3370 (discussed below). The controller circuitry was slightly different, but otherwise unchanged. *409.8 MB.*

S/370 Generation FBA: Something new was introduced for the IBM 4331/4341 in January 1979: a vastly simpler architecture of disk that used fixed-byte sectors (hence the acronym Fixed Block Architecture). These are of the same design as standard PC HDDs (and their emulatory counterpart SSDs). MVS required CKD DASD, but VM, VSE, MUSIC, Linux, and AIX/ESA could use FBA disks instead. For some customers, this saved a lot of money; for others, it was a headache, as it meant they were stuck probably running VSE and unable to easily upgrade to MVS.

3370, the first FBA disk introduced alongside the 4331/4341 (and also the System/38). Interestingly enough, it was the first hard drive to ever use the thin-film-head technology that is now standard on HDDs. These connected to an integrated DASD controller within the 4331/4341. *285.6 MB (model A1/B1), 364.9 MB (model A2/B2)*

3310, a FBA DASD for the 4331. Launched in January 1979 alongside the 3370, these also attached to an integrated controller on the 4331; not many of these were sold. *64.3 MB.*

9332/9335/9336/0671, some FBA disks seen alongside the 9370. These were SCSI disks announced in October 1986, and were attached to the host computer using IPI. The 9332 used an IBM 0667 SCSI disk; the 9335 used its own disk; the 9336 used an IBM 0681 SCSI disk. The 0671 was a bare SCSI disk, attached to an IPI protocol translator (much like the rest of these disks), and then attached to the host computer. *9332 = 184 MB (single-density) or 284 MB (double-density), 9335 = 412 MB, 9336 = 471 MB (single-density) or 856 MB (double-density), 0671 = 319 MB*

S/370 XA Generation: While 3350s were often still used on the 370/XA mainframes, the most dominant model used during this hardware generation was the...

3380, a large disk introduced in June 1980. The disks were constructed in a rather interesting way: each disk unit itself held two actuator mechanisms, and those could access up to 630 MB; put together, both provided a capacity of 2.52 GB per disk unit (for the confused, this resulted in each physical disk being split into two volumes, so, the OS would see two DASDs at two sequential addresses of 630 MB each). There were manufacturing issues (specifically with regards to lubrication), so the first disks shipped in October 1981. The string configuration was similar

to the 3350, wherein a model A disk could start a string and attach to up to 4 model Bs. This product was also introduced along with a new storage controller: the 3880. When properly configured, a 3380/3380 pair could process two I/O operations simultaneously! A double-density model was introduced in February 1985, and a triple-capacity model was introduced in August 1987. If connected to the later 3990 control unit, it is possible to get a quad-path string (which can handle 4 operations simultaneously). There were three types of models of 3380: the model As could start strings and be string members, the model Bs are string members, and the model Cs are special model As that have in-built controllers and can therefore attach directly to a parallel block channel (i.e. a bus-and-tag cable). *630 MB (A/B/C), 1.26 GB (E), 1.89 GB (K)*

System/390 Generation: The S/390 saw the last physical mainframe disks that implemented low-level CKD (humorously called a SLED, a Single Large Expensive Disk). There were a few landmark products seen here:

3390, a series of disks that used pressurized sealed disk assembles that were driven with brushless DC motors (as opposed to AC motors, seen on earlier models); introduced in November 1989. Strings were also different: a model A was placed in the middle, and up to two model Bs can be attached to the left and right of it. A model A can have 4 or 8 disk drives within it, and a model B can have up to 12; one could then attach up to 2 strings to a 3990 (the new controller introduced with the 3390) and have a disk subsystem of 64 disks. *946 MB (model 1), 1.82 GB (model 2), 2.83 GB (model 3)*

9345, an unusual and rare CKD DASD array comprised of specially-designed 5.25" drives in November 1990. These were part of IBM's Supercomputing Systems Extensions project, and these were the first hard drives to use magnetoresistive heads (which is the current standard today). These attached to the 9340 control unit, and it differed from a 3390 in the sense that it completely lacked the nonvolatile cache. These were often seen on ES/9000 systems. *1 GB (model 1), 1.5 GB (model 2)*

RAMAC and RAMAC II, two poorly-named products (no relation to the IBM 305 RAMAC that was the first hard drive ever), were RAID-5 arrays for System/390s. There were two versions of both: one model attached to a 3990 Model 6 controller, and another model had an integrated controller. Each unit consisted of a rack with 16 drawers, and each drawer held 4 drives. These were IBM Ultrastar XP type SCSI drives, using magnetoresistive heads. There was an interesting design fluke with these; each 3990-6 controller could only connect to one RAMAC/RAMAC II array, and could only address up to 180 GB. If you chose to get the integrated controller model, you'd save about $75,000, but you would lose a bit of performance and caching provided by the 3990-6. Since most mainframe shops wished to exceed 180 GB of disk capacity, and the only option was to buy two controllers, IBM sold two 3990-6es for the price of one! The RAMAC II launched in July 1995.

These machines held an integrated disk array controller that assembled 3390 volumes from the RAID disk arrays. The controller assembled a massive array from all of the disks, and there were several SCSI controllers within it to spread the load. The choice of a RAID-5 array (with the antonym to RAID being SLED) lowered costs *significantly*, and the increased performance meant that ESCON channels could finally be saturated with throughput with disk operations. Since these could handle up to 4 ESCON channels, performance was impressive for the time. This technology would become the foundation for future RAID arrays IBM provided, and rumors swirled about a RAMAC replacement that used SSA disks. This would materialize as smaller products found in the Multiprise line first, and would get its release 5 years in the future.

Multiprise 2000 Disk Array, and you can probably guess which System/390 it was found in. Remember, the Multiprise 2000 was a cut-down System/390 Parallel Enterprise Server, and the integrated disk array consisted of two rackmount parallel SCSI disk enclosures that held a total of 16 disks. Two I/O cards would be inserted into the S/390 CPU's I/O card cage -- one with 68-pin SCSI connectors, and another with control logic in it. These were essentially cut-down derivatives of the RAMAC II, and functioned in the same way. The way the S/390 CPU accessed the disks was also interesting: when the channel path was defined, it would be defined as an IBM CorporationSD" (a Internal System Device). When the CPU was initialized, it would begin to directly communicate with the disk controller cards without any intermediary ESCON channels.

Multiprise 3000 Disk Array, the successor to the Multiprise 2000, also had a revamped disk array. As mentioned in the section on the MP3000, this was done by having an IBM SSA (which was big at the time, SSA stands for Serial Storage Architecture and was the engineering precursor to technologies like SATA and SAS) controller present on the PCI bus within the cabinet. When the mainframe CPU is initialized, it grabs control of the card through a card on the PCI bus called a "camel card" (for the confused, the S/390 CPU would talk to the camel card, and that would allow it to talk to both the emulated devices on the Support Element and the SSA RAID card). Disk I/O operations were done by the S/390 CPU itself, and this reduced the hardware demands. SSA disks were much faster than the older parallel SCSI disks, and each 18.2 GB disk within an array (the MP3000 cabinet could accomodate two arrays, with more attachable externally) summed to about 73 GB per array. While the S/390 CPU was running, the OS/2 Support Element would have to communicate with the S/390 CPU (through the camel card) to configure the array; the array could be configured with the OS/2-native SSA configurator program while the S/390 CPU is stopped, however. The lessons learned from the RAMAC II, MP2000, and MP3000 disk arrays would all be orchestrated together in the successor product.

System Z Generation: When the System Z came out, the RAMAC II was seen as obsolete. Improvements had been made, and the most impressive (performance-wise) of those improvements came from, of all things, the Multiprise 3000's array. Despite it not sounding like much, it was a truly impressively performant solution for the time. There were two models of disk server that followed this:

The ***Versatile Storage Server***, released in 1998, was a transitional product that bridged the engineering gap between the RAMAC and what came next. This was developed concurrently with the MP3000 disk array, and was an adapted/enlarged version of that technology (minus the bus-mastering PCI SSA RAID card). The 2105-B09 VSS held up to 64 SSA disks, and these were attached to some VSS storage processors that provided an ESCON interface. If 64 disks wasn't enough, the customer could also purchase the 2105-100 expansion rack: this held 112 disks. While a commercial failure, it would see its primetime in the subsequent product.

The ***Enterprise Storage Server***, often called the *Shark*, was truly an impressive product. For one, this was not just a mainframe storage array! The ESS model F10 and F20 could be equipped with a variety of host adapters (that is, a peripheral interface circuit card that provided an emulation of a disk; for example, it could be a "SCSI target", and show up as a number of disks to the attached computer). You could use:

- UltraSCSI (i.e. SCSI Ultra-3/SCSI-160)
- Fibre Channel (original 1.25 gigabit Fibre Channel)
- FICON, the new System Z peripheral interface
- ESCON, since many System/390s were still in use when this product launched

The architecture of this was a modular evolution of all of the technology IBM had sold to that point. The ESS consisted of two clusters, able to communicate with each other through a shared cache. Each cluster attached to up to 4 device adapters, and these attached to the SSA disk loops in a rather odd fashion. Since an SSA loop had a ring-in and ring-out port, cluster A device adapter 1 connected to ring-in, and cluster B device adapter 1 connected to ring-out. The clusters connected to a Common Parts Interconnect bus that attached to the host adapters seen in the above list.

The controller could assemble RAID arrays from the disks and export volumes, or the disks can be directly exported to the host adapter (just not in CKD ESCON mode, since that required DASD emulation, only in FBA mode). AIX and TPF both had measures to mirror disks/recover from failures, so the slight performance boost afforded by this technique was worth it for some customers.

The ESS was a hit, not just for mainframes, but also broke through to other storage demand markets. People used these on AS/400s (as they supported the sector sizes required for such a thing), high-end UNIX machines from Digital/Sun/HP/Data General/Silicon Graphics, PC-based servers running Windows NT or NetWare, and even those Sequent SMP NUMA-Q minicomputers. Multiple computers could even be placed on the same SCSI bus, simply by changing the SCSI ID of each host adapter; this let users get even more out of the systems.

Providing a path forward from the ESS line was the ***DS6800, DS8100, and DS8800 series***, with one of those being significantly better than the other. There were many hardware revisions of the DS8800 line (10 at the time of writing this), and these consisted of POWER servers fitted with HBAs driving a rack of disk drives, and forking those connections out over Fibre Channel, FICON, or (on older models) ESCON.

The DS6800 (called the "Baby Shark") was a questionably-designed 3U disk array that could be attached to a System Z. These were intended to be low-cost, and featured a PowerPC 750GX-based Linux-powered control computer on-board. These units took direct Fibre Channel disks (despite their name, they did not have SFP connectors; in fact, they were essentially SCSI320 drives that spoke the FC protocol directly instead of parallel SCSI). However, these units were notorious for their failures, because the control CPU was not properly cooled. Sometimes, these would end badly, with the CPU becoming physically desoldered from its mount!

Tape Drives

While disks were certainly heavily used on the 360 line, tapes were where the real mass storage was at. Predating disks, the history of mainframe tape drives is as old as computing itself! However, the tapes we are interested in are *half-inch plastic-base dense-oxide* reels seen starting in the early 1950s. IBM pioneered the early ***7-track format*** in 1952, with the model 726 tape drive (seen on the 704 mainframe.

These drives were called "vacuum-column" drives, since there were two long vertical columns the tape would unroll into (with the head being in the middle). In other words, the tape would snake off the supply reel (the one you removed when you took the tape out), down column 1, up the other side of column 1, past the tape head and capstan mechanism, down column 2, up the other side of column 2, and finally onto the take-up reel. The "vacuum" comes forum the trigger mechanism used to advance the tape: the most pertinent part to move was the capstan (which moved the tape left and right on the tape head) and the movement of the tapes within the columns would trigger the motors on both reels. When the tape level in the vacuum raises above a certain threshold (marked by a hole in the back of the column), the vacuum drawn on the tape will be exposed to that hole. A vacuum switch (not unlike the force-multiplying pneumatic mechanism seen on player pianos) triggers the motors, and the tape advances either directions (depending on which of the two columns triggered).

Needless to say, the 726/727/728/729/7330 7-track tape drives were a big hit before the System/360. Once the 360 came out, we got the following tapes:

2400 Series: Though the 7-track format was good, IBM launched a newer 9-track format in 1965 with the S/360. The new 2400 series included several submodels, and several submodels of controllers. I shall list the controllers first:

- 2803 Model 1/2: can control and power up to 8 drives
- 2803 Model 3: can control and power up to 8 2401-8 drives
- 2804 Model 1/2: can control and power up to 8 drives, with one reading and one writing at the same time
- 2804 Model 3: can control and power up to 8 2401-8 drives, with simultaneous read/write support

The following 2400 drives were made:

Table 1. IBM 2404 Drive Models			
Model	***Tracks***	***Modulation***	***Density/Speed***
1	9	NRZI	800, 30kbps
2	9	NRZI	800, 60kbps
3	9	NRZI	800, 90kbps
4	9	PE	1600, 60kbps
5	9	PE	1600, 120kbps
6	9	PE	1600, 160kbps
1	7	NRZI	200/556/800, various
2	7	NRZI	200/556/800, various
3	7	NRZI	200/556/800, various
2401-8	7	NRZI	200/556/800, various

For the confused, NRZI is a non-return-to-zero-inverted encoding, and PE is a phase encoding (these are modulation modes used to record and read the data on the tape as an analog signal).

There was an alternate controller available that offered much better performance with newer tape drive: the ***2415***. There were 6 submodels:

- Model 1: NRZI, 1 control, 2 tapes
- Model 2: NRZI, 1 control, 4 tapes
- Model 3: NRZI, 1 control, 6 tapes
- Model 4: PE, 1 control, 2 tapes
- Model 5: PE, 1 control, 4 tapes
- Model 6: PE, 1 control, 6 tapes

2420 Model 7: Evolving from the earlier 2400 series, the 2420 (specifically the Model 7) introduced a very clever feature: an automatically-loading tape mechanism! The operator placed a tape on the right spool (as was customary, note that this is conventionally backwards to reel-to-reel audio tape recorders, reel-to-reel VTRs, and such seen in the broadcasting world), pushed a button, and the machine would release a belt that spanned the circumference of the tape reel to access the tape. To thread the tape, the machine would pull the tape off the reel pneumatically and run it through the machine (of course, it would drop it down through the vacuum columns too). This technology would live on in the successor, and was the envy of other tape drives from DEC and CDC that used the same format!

3400 Series: When the System/370 was announced in 1970, it included a new tape drive line. The 3400 series would be that successor: the 2420 autoloading scheme would be adopted for the 3400 series, and a new recording mode would be added. The new drives ran at a whopping 1.25 megabit transfer rate! As before, there were a variety of models:

- 3420: floor-standing vacuum-column units
- 3410: half-height-standing units
- 3422: follow-up to the 3420
- 3430: follow-up to the 3422

Now, let us examine the model characteristics:

Table 2. IBM 3400 Drive Models

Model	***Modulation***	***Density (bpi)***	***Speed***
3420-3	NRZI or PE	800 or 1600	75 ips
3420-5	NRZI or PE	800 or 1600	125 ips
3420-7	NRZI or PE	800 or 1600	200 ips
3420-4	GCR	6250	1.25 mbps
3420-6	PE or GCR	1600 or 6250	1.25 mbps
3420-8	PE or GCR	1600 or 6250	1.25 mbps
3410-1	PE	1600	20 kbps
3410-2	PE or NRZI	1600 or 800	40 kbps
3410-3	PE or NRZI	1600 or 800	80 kbps
3422	GCR	6250	1.25 mbps
3430-B01	GCR or PE	6250 or 1600	312 or 80 kilobytes/second

There were a variety of controllers produced for this line of tapes:

- 3803-1: 3420 Model 1/3/5
- 3803-2: 3420 Model 4/6/8
- 3411: 3410 (drives up to 4 drives)
- 3430-A01: 3430-B01 (drives up to 3 drives)

8809 and 9347: The ***8809*** was released in the mid-80s, and it was a 9-track manual-thread drive that did not utilize a vacuum column. These units were very compact, and decently fast; alas, they were low-cost and only used the 1600 bpi density mode. This drive was originally for the System/36, but could be connected to low-end models of the 4300 series.

Somewhat of a technical miracle, the ***9347*** was an auto-loading, flat-mount, rack-mount, thin 9-track reel-to-reel drive. You inserted the tape, and a complex loading mechanism pneumatically sucked the tape off the reel, ran it through a predefined path, and spun it onto the take-up reel. These drives did not use vacuum columns, and instead used a springloaded armature mechanism to alleviate tension. These were SCSI-interfacing, and were seen with the 9370 series. These were 1600 or 6250 bpi. *Note: the AS/400 line had the 9348, a very similar drive.*

3480: While reel-to-reel drives were great, they were not particularly compact, especially in the 1980s when everything else was beginning to shrink. IBM's answer to this was the 3480: a cartridge tape drive. By 1984 when the 3480 launched, cartridge tapes were standard in all other tape formats: Stereo-8 (i.e. 8-track) tape players were seen in cars, cassette tapes were all the rage, VHS/Betamax and U-Matic held video, so on and so forth.

The original 3480 tapes were half-inch square tapes, rolled entirely on a supply reel within the cartridge. The tape formulation changed to a chromium dioxide powder (the same used for video tapes of that era and “high bias” cassette tapes, marked CrO2). When the square cartridge was inserted into the drive, an armature loading mechanism would pull the tape out of the cartridge, past the head assembly (which was a new magnetoresistive design, sharing technology with hard disk drives of the era, like the 3380 DASD), and onto the internal take-up reel.

The original 3480 drive launched in 1984, and was attached to a System/370 with a parallel channel interface. The drive had a transfer speed of 3 megabytes per second, recorded with a 38000 BPI density onto 36 tracks, and the tape passed the heads at 78.6 inches per second. To avoid buffer-fill issues, a 512KB buffer was provided. The controller box drove up to 4 drive units, which each contained 2 tape drives (so, a total of 8 drives). The cartridges held 200 MB.

Successor drives to the 3480 series added ESCON support (though these were 3490Es, see below); other manufacturers (namely Fujitsu) made fully-compatible SCSI implementations of the 3480/3490E drives.

In 1986, a new model of 3480 came out: the ***3480 IDRC***, for Improved Data Recording Capacity. While this may sound like an increase in density, it was actually an improvement in controller electronics: it was a data compression scheme that allowed up to 400 megabytes to be stored on a tape assuming 2:1 compression.

3490E: The successor to the 3480 was the 3490E, launched in 1991. These drives almost universally had autoloaders that fed cartridges into the drives from a stack, and connected via channel, SCSI, or ESCON. The original model was the ***3490-B40*** with the ***3490-A20*** external controller box. Each tape held 800 MB.

There was also an IDRC capability for the 3490E; these upped cartridge capacity to 2400 MB (and these drives were almost universally attached using ESCON).

3590: In 1995, the IBM Magstar product hit the market. This was a big upgrade over the 3490E drives, albeit being much more expensive. There are many models of 3590 drive, and many controllers; the original controller, the ***3590-A50***, was actually an RS/6000 that adapted SCSI 3590 drives to ESCON. Here are the models:

Table 3. IBM 3590 Drive Models

Model	***Tracks***	***Length/Capacity***	***Speed***
3590-B11/B1A	128	320m/634m, 10GB/20GGB	9 megabytes/second
3590-E11/E1A	256	320m/634m, 20GB/40GB	14 megabytes/second
3590-H11/H1A	384	320m/634m, 30GB/60GB	14 megabytes/second

3592: The successor to the 3590 was the 3592, originally launched in 2003. These drives were attached via ESCON or FICON (via a converter, like the 3590-A50), SCSI, SAS, or USB (depending on the generation), The following drive models were produced:

Table 4. IBM 3592 Drive Models			
Model	***Year***	***Capacity***	***Speed***
3592 (3592-J1A)	2003	300 GB	40 MB/s
TS1120 (3592-E05)	2005	700 MB	100 MB/s
TS1130 (3592-E05)	2008	1 TB	160 MB/s
TS1140 (3592-E07)	2011	4 TB	250 MB/s
TS1150 (3592-E08)	2014	10 TB	360 MB/s
TS1155 (3592-55F/55E)	2017	15 TB	360 MB/s
TS1160 (3592-60F)	2018	20 TB	400 MB/s
TS1170 (3592-70F/70S)	2023	50 TB	400 MB/s

The F/E/S notation on some models listed indicates the attachment type:

- F: Fibre Channel
- E: Ethernet
- S: SAS

Line Printers

1403: The *1403* was the first printer introduced with the System/360 in its original run. This was a chain-type printer, where the font would revolve around on a chain and be struck onto the paper (with an ink ribbon between the font train and the paper) by hammers. On the 1403, the paper was pushed onto the font from behind, with the ribbon remaining stationary on rollers. The 1403 was the line printer that was introduced for the 1401 computer; the ones available with the System/360 were of several models:

- Model 2: 132 column, 600 lines per minute (upgradable to 1285 if letters were not required to be printed)
- Model 3: 132 columns, 1100 lines per minute (1400 if the character set was swapped out)
- Model 7: 120 columns, 600 lines per minute
- Model N1: 132 columns, 1100 lines per minute (1400 when swapped for the Universal Character Set); featured a motor-driven lid cover that reached down to the floor to deaden the noise

The 1403 had a feature called “carriage control,” wherein a loop of punched tape (positioned to the right of the print train mechanism) would have a series of pre-punched holes that described the layout of the “form,” or a map of fixed locations on the paper. For example, the printer could be given a datastream of a few lines of text seperated by form feed characters, and it would read the carriage control tape to know where to skip to when a form feed character was seen. The position of the tape was mechanically interlocked to the paper train, so its position always matched the paper's.

There was a similar model, the 1404, that had the ability to print onto punched cards. The mechanism was rather clever, in the sense that it could interpret cards (i.e. read what's on the cards and print the data on the top).

2821: The control unit that the 1403 was attached to was the *2821*, which also serviced other devices (which are described in greater detail below). The following models were available:

- Model 1: one 2540 and one 1403
- Model 2: one 1403

- Model 3: two or three 1403s
- Model 4: one 2540 and one 1404 (only for the System/360 model 25, 30, 40, and 50)
- Model 5: one 2540 and two or three 1403s
- Model 6: one 2540

The 2821 contained enough buffering logic to buffer one card or one line of text. A switch was present on the cabinet that switched which of the two channel connectors on the back panel it would use to connect to the host computer, so you could use it on two different computers. Strangely enough, the early models of the 2821 were not implemented using the SLT integrated circuits of the rest of the System/360 line, but were instead implemented with Standard Modular System cards (if you've seen the inside of some of the DEC PDP computers with their flip-chip cards, IBM SMS is not far off). Newer revisions had some SLT logic, but was not made entirely of SLT. Finally, in 1985, the product was withdrawn.

3211: The *3211* was the immediate successor to the 1403 series. Launched on June 30, 1970, the original 3211 could print at 2000 lines per minute (which was double the speed of the extant 1403 models). Unlike the 1403, the 3211 was a combination of 3 different distinct devices:

- The 3211 itself, the main printer; this was referred to as a "front printer" because the print hammers would bang the ribbon into the paper, rather than banging the paper into the ribbon (like on the 1403). The 3211 boasted the following features:
  - Interchangable type cartridges for a variety of fonts and configurations.
  - 90 inch-per-second paper movement during form feeds.
  - 132 columns, changeable to 150 with a replaced type cartridge
  - A power stacker that processed the printed output
  - A control panel that did not require the host computer to be operational for diagnostics and testing
  - Programmable forms control buffers (FCBs) in lieu of carriage control tapes
- The 3216, the print cartridge; these held a total of 432 letters, of which were organized such that there were 4 letters per type element (so, the revolving print train had 108 pieces attached to it). These type slugs were mounted onto the 108 carriers, such that they could be replaced. The 3216 included an oil-dispensing system.
- The 3811, the control unit that attached the printer to the channel of the mainframe.

3203: Another option that replaced the 1403 was the *3203*, and was nearly the same mechanism. The carriage control tape mechanism was removed, and connected to the mainframe in a rather strange way. Models 1 and 4 did not require an external control unit, and instead attached directly to the System/370; the caveat was that the S/370 in question had to have an integrated 3203 controller. The Model 3 attached to a 3770 batch terminal, and the Model 5 had a parallel channel adapter.

The printers printed at 1200 LPM except for the model 1, which printed at half that (600 LPM). The 3203 was special for the time as it could actually print Braille, and this was done by putting a rubber strip over a special print train font set that could print Braille dots without punching holes clean through the paper.

In the early 70s, IBM started to explore printer architectures that did not use a revolving print train of heavy print elements, and instead used a flexible band. Other manufacturers at the time had drum printers (like the DEC line printers, like the LP25, though most were made by Dataproducts), but IBM thought band printers would yield much better print quality -- drum printers were fast, but the text often looked somewhat jerky and was not vertically-lined usually.

3618: The first was the *3618*, a so-called"Administrative Line Printer." This printer was 80 columns (upgradable to 132), but was not a 370 peripheral. Alas, it was announced in 1973.

3262: The *3262* was the first band printer for 370 systems; there were a variety of models:

Table 5. IBM 3262 Models			
Model	***Interface***	***Speed***	***Date***
1	370 channel	650	Jan 30, 1979
2	8310 loop	650	Oct 2, 1979
3	3270 coax	650	Oct 31, 1979
5	370 channel	650	Nov 12, 1982
11	370 channel	325	Oct 2, 1979
12	8130 loop	325	Oct 2, 1979
13	3270 coax	325	Oct 31, 1979
B1	System/36	650	May 17, 1983
C1	System/36	650	May 17. 1983

5262: The successor was the *562*, launched on October 2, 1984. These used a different print band from the earlier 3262, and had a print speed of 650 LPM. Model 1 was twinax-attached, to a System/36.

4245: The high-speed option of the era was the *4245*, launched in 1983. The following models were available:

- Model 1 (May 3, 1983), 2000 LPM
- Model 12/20 (April 16, 1985), 1200/2000 LPM; features included:
  - Improved reliability over the Model 1
  - 64 dB volume during operation
  - Capable of printing MICR

Here is a convenient table of the models and their attachments:

Table 6. IBM 4245 Models			
Model	***Attachment***	***Speed***	***Date***
1	370 channel	2000 LPM	May 3, 1983
12	370 channel	1200 LPM	April 16, 1985
20	370 channel	2000 LPM	April 16, 1985
D12	3270 coax	1200 LPM	May 1, 1985
D20	3270 coax	2000 LPM	May 1, 1985
T12	twinax	1200 LPM	June 16, 1986
T20	twinax	2000 LPM	June 16, 1986

4248: The *4248* was the successor, and was much faster -- it ran at 2000, 3000, or 3600 LPM! Launched on February 7, 1984, the 132 column (or 168 columns, if you wish to print 2 documents side-by-side at the same time) printer had a band that travelled at 45 miles per hour, with hammers that flew through the air for only 30 microseconds. There were two models:

- Model 1, the original described above
- Model 2, launched in February 1987, could print at 4000 LPM.

The 4248 had a program called Automatic Flight Time Compensation that would measure how long it took the hammers to strike the page, and store a table of delays on the microcode diskette; it would then use that to calibrate the page advancement speed to prevent drifty and unaligned text issues.

6262: The grand successor to all of these was the *6262*, and the following models existed:

Table 7. IBM 6262 Models

Model	***Attachment***	***Speed***	***Date***
A12	parallel/serial	1200	Aug 28, 1990
D12	3270 coax	1200	Feb 2, 1988
P12	Dataproducts	1200	Oct 24, 1989
T12	twinax	1200	Feb 2, 1988
014	370 channel	1400	Feb 2, 1988
A14	parallel/serial	1400	Aug 28, 1990
D14	3270 coax	1400	Feb 2, 1988
P14	Dataproducts	1400	Oct 24, 1989
022	370 channel	2200	Oct 24, 1989
A22	serial/parallel	2200	Aug 28, 1990
D22	3270 coax	2200	Oct 24, 1989
P22	Dataproducts	2200	Oct 24, 1989
T22	twinax	2200	Oct 24, 1989

Console Printer/Keyboards

1052: The first typewriter console for the System/360 was the *1052-7* (though other models were occasionally used), which was derived from the 1050 Data Communications System. There were many subparts to the 1050 system (including 1052 typewriter terminals, 1053 printers, 1057 punches, 1056 readers, 1054/1055 paper tape readers/punches, 1092/1093 programmed keyboards, 1058 printing punches, 1051 control units,and something called the “IBM 2064 FBI Program”). These attached directly to the System/360 host computer, and were used as OS system consoles.

3210/3215: The successors to the 1052 were the *3210/3215*, launched in late 1970. There were 3 models:

- 3210 Model 1: used a Selectric printer with a Card Punch keyboard, capable of directly modifying data in the host computer's storage; sat upon a table. Powered by the host processor power circuitry.
- 3210 Model 2: used the same keyboard/printer elements, was not capable of modifying data in the host computer's storage; sat on a pedestal and was connected to the host processor via a long remote cable. Powered by an internal power supply.
- 3215: used a 7x7 dot-matrix printhead with an Elastic-Diaphragm keyboard and printed at 80 characters per second; capable of modifying the host computer's storage, was also powered by the host computer's power circuitry.

Card Readers/Punches

The System/360 was a batch machine, so the availability of card readers and punches was a requirement. There were several available:

1442: An extant (at the time of the 360's release) card reader/punch machine seen on the 1130, 1800, and 1440 computers. The holes in the cards were illuminated by fiber optics. The punch famously did not print a textual representation of the cards to the top! A dedicated card "interpreter" would be used to print those letters that correspond to every column on the card. The following models were made:

- Model 1: reads at 80 cards/minute, punches at 50-270 cards/minute; comes with one stacker (with a second one being optional).
- Model 2: reads at 400 cards/minute, punches at 91-360 cards/minute
- Model 6: reads 300 cards/minute, connects to a System/3 or 1130
- Model 7: reads 400 cards/minute, connects to a System/3 or 1130
- *Model N1*: reads 400 cards/minute, connects to a System/360 or System/370 by way of a parallel channel
- Model 3: read-only at 300 cards/minute, connects to a 1410 or 7010
- Model 4: read-only at 400 cards/minute, comes with two stackers instead of one, connects to a 1440
- Model 5: punch-only at 180 columns/second, connects to a 2922 Programmable Terminal or System/360 Model 20
- Model N2: punch-only at 180 columns/second, connects to a System/360 (not the Model 20) or System/370

2540: An alternative was the 2540, launched in 1965. There was only really one model, and these were very often seen on System/360 installations. These attached to the host through a 2821 control unit *(see the section above for information on this, underneath the 1403 section in the Printers segment)*. These were much faster than the fastest model of 1442; these read at 1000 cards/minute, and punched at 300 cards/minute.

There were two different devices that would show up to the host, the 2540R (reader) and 2540P (punch). The reader had one input hopper and three output stackers (with the third one being sharable between the punch or reader, but not at the same time). The punch, likewise, held three output hoppers (with the third one being shared).

The 2540 could both read and punch column binary mode, allowing for any data to be stored on cards.

3501/3521: Between the 2540 and its more successful successor, the 3501 reader and 3521 punch seemed to be smaller models of the related 3505/3525 with only one stacker. Little information on this product exists; it did not seem to be very popular.

3505/3525: The successor was the 3505/3525, launched in 1971 for the System/370. It read at 1200 cards/minute (or, on the Model B2, 800 cards/minute) and could read cards that were optically marked (like, if you were to use a pen to color in the spots on the card by hand). There were 3 submodels of the 3525: the P1, P2, and P3; these punched at 100, 200, and 300 cards/minute respectively.

Networking Devices

The mainframe, being a natural system to experience a connection to a network, found itself amongst a sea of networking peripherals. Here are some of those:

Channel-to-Channel Adapters: While this may seem foreign to those familiar with PC hardware, this is not uncommon in mainframes! A CTCA (as it is often abbreviated) lets two mainframes communicate directly with each other using a channel path. This is roughly equivalent to connecting two PCs together with PCIe ribbon cables, albeit through a small box to facilitate the connection. There were two CTCAs made:

3088 CTCAs were the first, and this was originally a box that would be connected to a spare channel on a System/360. These were available up until the end of parallel block channels, and was eventually replaced when ESCON became the standard interface. Note that by the mid-80s, these devices were emulated; the channel controller itself gained this ability directly.

ESCON and FICON CTCAs followed shortly thereafter; these were configured in the IOCS configuration file as nothing more than a built-in device and were directly supported by the channel processor (there were no external boxes to be seen with these).

8232 (LAN Channel Station): The 8232 LCS was the first attempt to create an Ethernet, Token Ring, PC Network (an IBM product, an old version of Token Ring), and MAP (Manufacturing Automation Protocol, a General Electric product that became IEEE 802.4 Token Bus) interface box for a System/370 in 1987. The device in question was an IBM 7532 Industrial Computer, and it ran PC-DOS. There were actually two models of 8232, with Model 1 consisting of one 7532 (and could therefore connect to up to 2 networks), and a Model 2 that was just two 7532s. The LCS program, the so-called "IBM LAN Channel Support Program," executed under the control of PC-DOS, and the 7532(s) were mounted in a waist-high 19-inch rack (which had a cutout for the 7532(s)'s power switch(es)). 4 disks were required during installation:

- PC-DOS, which would eventually be partially copied to the target disk
- The LCS Support Program, which would be customized and written to the target
- The Local Area Network Support Program, for configuring network boards
- The target disk, which would eventually be bootable and be left in the drive

The device was internally referred to as a "PCCA" (which I am guessing means PC Channel Adapter), and ran the following programs in a modular fashion:

- 3 modules to allow the LCS to connect to 8 different kinds of mainframe and to the 2 different types of networks:
  - MTCMMAIN.EXE (the main task that executed the others)
  - ETHTASK.TSK (for Ethernet support)
  - TOKTASK.TSK (for Token Ring support)
- 3 modules to support the LCS:
  - MONTASK.TSK (presumably to monitor the function of the device)
  - PCCATSK1.TSK (to drive the host channel adapter)
  - SCRN8232.TSK (presumably to drive the display status screen)
- 1 program to configure everything:
  - MTCMCONF.EXE

To actually configure the device, the user had to be aware of the following parameters:

- Model 1 or 2, since the Model 2 meant twice the work
- Which network cards were installed in slot 5 and 6 of the 7532, which could be Ethernet, Token Ring, or IBM PC Network

- The device address on the host that this would be attached as
- 32 or 64 control unit blocks for block transfers
- 1mbps or 3mbps channel speed
- The memory address, IRQ, and type of the LAN card(s)

Once the configuration and installation was done (which would be done by producing a DOS disk onto the target disk, running the LAN Support Program to configure and load the network card drivers at boot, and then configuring and copying the LCS software), the box was rebooted, and everything would be ready for the host program.

One can tell that this box is based on earlier work done during the development of WISCNET for VM (search this document for WISCNET to read about the developmental history of this box).

Some other manufacturers built machines that were compatible with the LCS. These included:

- The Cisco 7200 series ISR, with a parallel channel or ESCON interface board
- Bustech Netshuttle
- Polaris StarGate
- Interlink 3762

3172 (Nways Iterconnect Controller): In November 1990, IBM launched the successor to the earlier (and much slower) LCS. This was 1990, and this was the era of ESCON -- the new 3172 came with an ESCON adapter, but were often used with parallel channel adapters. In this era, LAN networks rarely went past 10 megabits (except for FDDI, which was truly fast at 100 or sometimes 200 megabits), so a parallel channel adapter would suffice for most customers.

The 3172 consisted of a rack-mount cabinet with a control panel (the so-called “operations panel”), a 486 CPU with some RAM, a floppy drive and hard drive, a channel adapter, a LAN card, and a power supply. The following LAN options were available, and you could have up to 4 cards (except for MAP boards, since those required 2 card slots):

- Ethernet (10 megabit)
- Token Ring (16/4 megabit)
- FDDI (100 megabit)
- MAP 3.0 Broadband Mode (for the unaware, Manufacturing Automation Protocol is is an IEEE 802.4 10 megabit token-bus network)
- MAP 3.0 Carrierband Mode (same thing, but 5 megabits)

When the machine was ran in normal mode, OS/2 would load up and run similar programs to those found on the old 8232 LCS. This device showed up to the host as a 3088 CTCA, just like the LCS; since this technically did provide CTCA-style functions, IBM did provide a special mode that allowed up to 4 T1 lines to be bonded together as a real CTCA. The 3172 supported the following LAN protocols:

- SNA; this was proper 802.3 encapsulated SNA and this would be ran under the control of VTAM
- TCP/IP; this was either 802.3 encapsulated or DIX (Ethernet II) encapsulated (the standard in modern times) TCP/IP and this would be ran under the control of a TCP/IP stack
- OSI CLNS; this was the infamous TCP/IP competitor that was ran under the control of OSI Communication Services (a currently lost program)

WAN cards were also supported (these were IBM Microchannel WAC boards which held a V.24 or V.35 serial interface), and this in-turn supported the following protocols:

- A point-to-point SDLC line (this would talk to downstream 3274s and 3174s, or an emulation thereof)
- A point-to-multipoint (i.e. multidropped) SDLC line
- An X.25 line (which can carry SNA, TCP/IP, and presumably OSI traffic)
- Frame Relay

In a LAN mode, there were two distinct operating modes that the 3172 could present to the host. These were:

- Standard LCS mode; this was the standard but did not offer as good of performance as it could be
- 3172 Offload mode, wherein the TCP/IP stack ran on the 3172, called CLAW (Combined Link Access to Workstation) mode

CLAW mode was rather interesting. It required the installation of additional software, and ran alongside the ICP (the Interconnect Program, the program that ran the system).

2216 (Nways Multiaccess Controller): In 1999, IBM launched a true competitor to the Cisco Integrated Service Router (in those days, the Cisco 7200 was looking like the be-all-and-end-all of multiprotocol routing). Though the 2216 was primarily a router, it had mainframe support (then again, so did the Cisco 7200). The 2216 supported the following port interfaces:

- Parallel channels
- ESCON channels
- Token-Ring
- 10/100 Ethernet
- EIA-232/V.24
- V.35
- X.21
- ISDN PRI (T1/J1)
- ISDN PRI (E1)
- ATM multimode
- ATM singlemode
- FDDI
- HSSI

Pretty impressive, right? Well, it also supported the following network protocols:

- IPv4, with IPsec
- IPX
- AppleTalk Phase II
- Banyan VINES
- DECnet Phase IV
- DECnet Phase V (sometimes called DECnet/OSI)
- SNA (APPN, DLSw, Frame Relay, and Enterprise Extender)

- NetBIOS framing

Wow. Now, how was this useful for mainframes? Well, recall that it had channel and ESCON interfaces. These could be attached to the host, where they would emulate a 3172 with downstream network nodes; this meant it could directly support TN3270E by also emulating a 3174 as a downstream PU of the channel. Since these also supported APPN and HPR, these were far advanced over an old 3174. To recap, these supported 2 operating modes: IP and SNA (just like the 3172).

These routers had a rather fast CPU: a 233 MHz PowerPC 604; this was significantly faster than the 486 found in the 3172s, and this provided much faster network throughput (until it was later replaced with the OSA).

Open Systems Adapter: In the late 90s, it was starting to become obvious that the extant method of doing networking on mainframes (with a 3172 or 2216) was too high-latency. The result of the effort to drop that latency was something called an Open Systems Adapter, seen on 9672 Gen 3 machines and newer. These operated in a radically different mode called ***QDIO*** (Queued Direct I/O), wherein the host CPU would drive the network card with a memory-mapped-I/O technique (rather than having something process a CCW program). The new OSA cards supported a variety of networks:

- 10/100 Ethernet
- Gigabit Ethernet (OSA-Express)
- 10 Gigabit Ethernet (OSA-Express2)
- 155 ATM
- 4/16/100 Token Ring

Over the years, there were several hardware revisions of the OSA. Here is a brief summary of each model:

- The original OSA and OSA-2: introducing support for QDIO, these supported Fast Ethernet, 4/16/100 Token Ring, FDDI (optional), and 155 ATM (optional). The IP QDIO mode (see below for something called OSD mode) was a layer-3 solution only; you could not pass arbitrary Ethernet datagrams through the adapter (much in the same way as the LCS). Up to 80 IP stacks could be ran with one channel path.
- OSA-Express: adding Gigabit Ethernet support as well as the Integrated Console Controller (see below), the OSA-Express cards found on the z890 and newer also supported true layer-2 Ethernet. The number of IP stacks supported was raised to 160.
- OSA-Express2: new for the z9, 10 Gigabit Ethernet support was added. You could now have up to 640 IP stacks, allowing many (probably Linux) VM guests to share the same card and still have good performance.

There were several operating modes that the card could be put in:

- The OSE operating mode simulates a 3172 LAN Channel Station (which was once a modified Microchannel-based PC running an OS/2 program called ICP that provided TCP/IP and SNA access for Ethernet, FDDI, and Token Ring networks)
- OSD mode (which was added on the OSA-Express cards) is the standard operating mode in which a Gigabit/Fast Ethernet or 155-megabit ATM controller is driven by the CPU using DMA (this mechanism is called QDIO, i.e. Queued Direct I/O).
- OSC mode (the Integrated Console Controller mentioned above)
- OSN mode, introduced with the OSA-Express2, was intended to provide an Ethernet interface for Communications Controller for Linux to use.

Something the z890 and z990 added was the Integrated Console Controller. In the "olden days", you had very few options for providing a 3270 console to a mainframe. If you had a Multiprise 3000, great! If not, you would need either a 3174 with coax terminals or an IBM 2074 console controller that would emulate a local terminal controller

with TN3270 sockets. The ICC was a special operating mode of the OSA card, running in the OSC mode. This would claim a LAN port for a TN3270 server, and it would simulate local 3270 terminals. One could then connect a TN3270 emulator and get a console on the mainframe.

Non-Programmable Communications Controllers

In the early days of the System/360, there became some kind of need for terminals attached to mainframes. Before the aforementioned 3270 series was made available, the dominant way to connect terminals to a S/360 was through one of these. These devices did continue to exist into the 80s and 90s in an emulated capacity.

2701: The simplest of the ***270x series*** was the 2701, announced in 1964 and delivered in 1967. This device was a four-port "Data Adapter Unit" that provided either synchronous transmit-receive (STR) or bisync interfaces at up to 40.8 kbps. These officially supported the following devices listed in the product announcement, providing an interesting view of terminal devices available in the mid-1960s:

- Devices that connected to the IBM 7710/7711:
  - IBM 7710/7711, the original STR interface that connected IBM 1401 computers together (or to an S/360 via a 270x)
  - IBM 7701/7702 Magnetic Tape Transmission Terminal
  - IBM 1009 Data Transmission Unit
  - IBM 1013 Card Transmission Terminal
- The IBM 7740 Communication Control System, the successor to the 7710/7711 (often seen alongside the IBM 1410 computer):
  - IBM 7750 Programmed Transmission Control
- IBM 1030 Data Collection System:
  - 1031 Input Station (card reader, badge reader, or manual input terminal; attaches to the remote computer via the 1031A)
  - 1032 Digital Time Unit (wrote timestamps onto data)
  - 1033 Printer (shares the 1031A)
  - 1034 Card Punch
  - 1035 Badge Reader
- IBM 1050 Data Communication System (see the section on the *1052* for more information on this)
- IBM 1070 Process Communication System:
  - 1071 Terminal Control Unit
  - 1072 Terminal Multiplexer
  - 1073 Latching Contact/Counter Terminal/Digital Pulse Converter
  - 1074 Binary Display
  - 1075 Decimal Display
  - 1076 Manual Binary Input
  - 1077 Manual Decimal Input
  - 1078 Pulse Counter
- AT&T 83B2 Selcall Terminal

- Western Union Plan 115A Outstation
- Common Carrier TWX Station
- IBM 2740/2741
- IBM 2260/2848

Out of the devices listed here, only the last two (which weren't even originally included on the supported list in the product announcement) were “timesharing” devices; the rest of these were devices driven by a computer in an era before embedded controllers and microprocessors.

2702: If you needed more communications lines, one could get a 2702, released alongside the 2701. The trade-off was that these 31 lines were STR-line-coding only, and ran at a lower speed than what was possible on the 2701. This box supported the same devices as those listed above for the 2701, minus anything that communicated using bisync (so, anything starting with a 7 in that model list).

2703: Announced in 1965 and made available shortly after the 2701/2702, the 2703 was the most popular of the three models. Gone was the 4 or 31 line count limit, these had 176 (half-duplex) lines that ran in either STR mode, bisync mode, or TTY mode (i.e. a normal async serial port). These attached to the host computer via a selector channel (similar to the 2701/2702), and each device following the control unit address represented a configured port: the control unit could be at address 300 and port 5 could be at address 304. While this box certainly had many communications lines (and was perfect for timesharing), the line speed was limited to 2400 baud (increased to 4800 baud by 1970), one could connect an IBM 2712 to multiplex 14 extra devices onto one single high-speed line to the 2703 (the host computer would need to be made aware that a multiplexer was present on that port).

These devices were quite popular with TSS/360 and later CP/CMS and TSO users, and were electrically simple (which allowed for quick cloning by various other manufacturers). Clones included the Memorex 1270, and three devices made by NCR-Comten.

Since the 2703 was so popular, an emulator program was written for the 3705 that replaced it (see below).

AWS2703: *Note: this device ONLY emulates ports on a 2703 running in TTY mode*

A P/370 and P/390 device manager program that emulated the 2703 and 2540 punch was also available. This was intended to connect to a modem attached to an asynchronous port, but OS/390 that existed during the period this device manager was available famously could not actually utilize a 2703 for TSO! VM, however, could; this was a documented use. Alas, one could use this to drive all manner of unusual devices from the P/390 operating system.

This emulator also emulated the 2540 punch, which was explicitly made available such that RSCS on VM would control the port, and a user could punch a plotter output file onto a real serial port to drive a plotter using this technique. While one could drive a plotter in an unattended fashion, the IBM manuals for the P/390 directly state that this is a bad idea!

9370 ASCII Subsystem Controller: This device was a logically-similar device to the 2703, and somewhat functioned like one. This was a card installed into a 9370 that provided up to 16 serial ports that ran in three different modes:

- Native ASCII support mode
- ASCII/3270 conversion mode
- ASCII/3270 transparent mode

The following devices could be connected:

- 3101 Display Terminal

- 3161/3163 Display Terminals
- IBM Personal Computer
- DEC VTxxx terminal
- ASCII printer
- ASCII plotter

4331/4341/9370 Telecommunications Subsystem: While the 37x5 series was considered a "programmable" communications controller, a non-programmable analogue was the Telecommunications Adapter. This connected to:

- Up to 3 Multi-Protocol Two-Line adapters (providing synchronous serial)
- Up to 3 Asynchronous Four-line Communications adapters

These lines could be 64 kbps SDLC lines, or 19.2 kbps HDLC (X.25) lines. This device existed in an analogous form on the IBM 43xx processors, and these devices are sometimes colloquially called an INTEGRATED COMMUNICATIONS ADAPTER.

AWSICA: *Note: this device is similar to the AWS2703 manager, except this device operates in BSC mode in lieu of TTY mode.*

Though OS/390 VTAM could not use a 2703 in TTY mode for a TSO terminal, OS/390 JES2 (as well as VM/ESA RSCS) could use an emulated (by this AWSICA adapter) 2703 BSC line for NJE communication. This adapter used WAC ports (Wide Area Connector ISA card, the manager could also use the older Multiprotocol adapter cards) in synchronous RS232 mode, X.21 mode, V.35 mode, or RS422 mode. Depending on the card type used, the user had to load either AWSMPADD.SYS (for the Multiprotocol cards) or AWSWACDD.SYS (for WAC cards).

2260 Display System

The 2260 series was the logical precursor to the 3270 series, and was *much* more primitive, both from a design standpoint and a functionality standpoint. Introduced in 1964, this was the first real successful teleprocessing/timesharing terminal (that wasn't a specialized graphics device for CAD) seen on the S/360; through the success of the S/360 and this line, this became the foundation for the first successful CRT video text terminal.

There were three models of 2260 display (listed as columns by rows):

- Model 1: 40x6
- Model 2: 40x12
- Model 3: 80x12

These devices included optional keyboards, which could be either a numeric-only keyboard or a full typewriter keyboard -- this setup was referred to in the product announcement as a "man-to-machine communication" system implementing a "visual I/O concept." When keys were pressed, the letters typed would be echoed on the CRT screen via way of the terminal controller (described below). The display could show uppercase letters, numbers, and 25 special symbols (which included the space, newline, and other such characters).

The displays were rather odd in the sense that the image was drawn 90 degrees rotated to how a normal CRT television would draw the image: the scanlines on the CRT screen started at the left-most edge, and then scanned down (so, top-to-bottom scanning, incrementing rightward as the image draws). Remember, this was an era before cheap semiconductor RAM chips existed, so the engineers found a very clever solution for storing the image. Rather than using some kind of odd storage tube, the engineers found that they could actually fashion up an acoustic delay line!

2848 Controller: The screen image was stored in a delay line whose length and speed-of-sound characteristic meant that one image could be stored in a big spiral-wound wire loop retained by rubber holders. A “transmitter” would impart a vibration into the wire as a twist of sorts, and a “receiver” on the other side was a device that could detect the torsion (this was a device built somewhat like a phonograph pickup). The image could be updated by waiting for the timeslot on the ring (kept up by a counter) and then retransmitting the modified signal into the ring; since the image will dampen out after one revolution around the acoustic ring, the image must constantly be re-imparted into the ring by the transmitting transducer. Of course, this inevitably introduced some issues; if you walked heavily by the controller (or installed the controller by an elevator, door, printer, or something else that produced vibrations), the image would be scrambled and gradually fade away on all of the screens!

The 2848 could attach to the host computer by either a channel (operating locally), or through a communications set (an old term for what we might call a modem today, either dial-type or leased-line) via a 2701 Data Adapter.

2265: This was a single-unit terminal that attached directly to either a channel or communications line, and performed the functions of a single 2260 and 2848. This is analogous to, for instance, the future IBM 3275 controller-terminal. This was announced alongside the 2770, and is discussed there.

NCP Communications Controllers

3705/3704: In 1972, IBM launched a programmable communications controller that would implement SNA networking and drive downstream 3274s/3174s. These came in the form of the 3705, which itself ran several programs:

- The EP (Emulation Program): this program emulated the earlier 2702/2703 "dumb" communications controller (which mainly provided ASCII, Bisync, or STR lines).
- The NCP (Network Control Program, no relation to ARPAnet NCP or DECnet NCP): this implemented the Systems Network Architecture network protocol and attached to cluster controllers (i.e. 3274/3174) over synchronous serial lines or T1 circuits.
- The PEP (Partitioned Emulation Program): provides a subset of the EP and a subset of the NCP, used when both functions were needed.

The 3705 drove up to 352 communications lines, and the smaller 3704 (introduced in 1973) supported up to 32 communication lines.

3710: This was the successor to the 3705, and was introduced in 1984. This was a much smaller network controller that was about as large as three 8-inch floppy drives set vertically, and was analogous to a shrunk 3705. These used a 3101 ASCII terminal for the control terminal, unlike an operator panel seen on the earlier 3705s. These had RS-232, V.35, and X.21 communications adapters, and introduced the ability to connect to another 3725 (see below) through an X.25 network.

3725: In 1983, IBM decided to overhaul the 3705 architecture by reimplementing the formerly-hardware line processors with ones using microcoded processors. This was a hefty design change, and there were several models in the series:

- 3725: The original communications controller in the series
- 3720: A shrunk version of the 3725 that replaced the 3710
- 3726: An expansion box for the 3725 (to add more line cards)

The 3725 was configured with 512 KB to 2 MB of main storage (installed in 256 KB increments), a single type of channel adapter (which could talk to up to 6 host processors, or 4 multiprocessor host processors), a single type of communications scanner card (which implemented the SDLC, BSC, and start/stop protocols), five different types of line interface cards (for a total of 256 full-duplex or half-duplex communications lines), and a terminal for the

operators console (a 3727). Each frame added more and more line cards, with 96 with one frame or 256 with two frames (which added 160). The lines ran at a maximum of 256 kbps.

3745: The final NCPs were introduced from 1988 to 1992. The 3745, introduced in 1988, was directly equipped with T1 line cards to match the era. These were large machines, spanning three cabinets; the 3476 Nways Controller was originally an expansion cabinet for the 3745 that added Token Ring and ESCON connections. In 1995, the 3746-950 was launched, shrinking and combining the featureset into the final NCP made.

The following models were made:

- 130
- 150
- 160
- 170
- 210
- 310
- 410
- 610

Communications Controller for Linux: CCL is a z/Linux (and Linux/390) based emulator for the 3745, capable of running unmodified 3745/3746 NCP programs. These would be able to communicate with the host via channel-to-channel adapters (usually under VM), or talk on a network using a special operating mode of the OSA cards (OSN mode).

3270 Display Controllers

The 3270 Display System was a relatively advanced (for its time, arguably still even today) terminal system that replaced the earlier 2260 system in 1971. These functioned as parts of a distributed network, attempting to alleviate as much of the host CPU load as possible during terminal control tasks. We will discuss the controllers first, then the terminals next.

The 3270 terminals (see below for more information on them) attached to a communications controller through a coax connection; many sites used "baluns" (just like in radioelectronics) to connect 3270 terminals to their associated control units over phone lines (in lieu of "expensive" 93 ohm RG-62 coax cables). The controllers included:

3272: The first channel-attached 3270 controller for S/360 and S/370 machines was the so-called “3272 Local Controller”; this was a nondescript gray box with a big wire-wrapped backplane inside it. The circuitry was somewhat similar to the 3271 -- it was almost entirely comprised of 7400-series logic ICs. The terminals attached to coax jacks at the back, and the 3272 connected to the host mainframe with a bus-and-tag cable.

3271: The remote version of the 3272 is the 3271, and it had a V.24 serial port on the back that either connected it directly (via synchronous serial SDLC) to a 2703 or 3705 communications controller, or could be attached to a synchronous modem (the 3863/3864/3865/3868/3872/5811/5865/5866/5868). These would be intended for attachment at perhaps a remote office, and the modem could be a normal dial-type modem or a leased-line modem. Alas, one could also directly connect it to a 2703/3705 if desired.

3274: The second "local non-SNA" terminal controller (that could also do "local SNA" and just plain "SNA") was the 3274, and it replaced the earlier 3271 and 3272. There were several models of 3274, and they were equipped with 8/12/16/32 ports -- these came in two types:

- Category A, the newer (despite it coming first) coax interface with a slightly different coax protocol introduced with the 3274. These worked with new terminals introduced in the 3274 product generation, like the 3278 and 3279. The first 8 ports were always category A; category A ports were seen in the first cluster on a multi-cluster (i.e. two boards of BNC ports) 3274.
- Category B, the older coax interface, intended to hook up to devices like the 3277 terminal and 3284 printer. These were always found in a cluster that came after the first one.

Conversely, there were 3 different types of 3274 for each model. These were:

- ***A-unit***, a local (i.e. parallel channel) control unit that provided "local SNA" terminals; this showed up to the host computer as one device. VTAM would have to drive this device as a proper SNA control unit, and terminals would be addressed as proper SNA devices. The 3274-xA was intended to replace the 3271.
- ***B-unit***, also a local channel control unit that provided "local non-SNA" terminals. This provided many devices (32 if you had 32 ports) to the host, and these devices were either defined as either displays or printers. Since these were local devices, you would, instance, find yourself logged into VM with a 3 or 4 digit device address for the terminal, as opposed to an LU name for a 3274-xA. The 3274-xB was intended to replace the 3272.
- ***C-unit***, a remote SDLC/BSC or local SNA control unit.
- ***D-unit***, the same as the 3274-xB except these could not communicate with 30xx processor.

The 3274s loaded from an 8-inch floppy disk, and held a rather advanced integrated computer. By the end of its life, it supported quite an interesting array of software features:

- 3270 Extended Data Stream and highlighting
- Programmed Symbol Sets, for custom fonts
- A V.24 synchronous serial interface (works up to 14.4kbps)
- A V.35 synchronous serial interface (works up to 56kbps)
- SNA or X.25 network attachment on the synchronous serial port
- DownStream Load for the 3290 and 3179G graphics terminals
- Distributed Function Terminal (DFT) mode
- Control Unit Terminal (CUT) mode
- Support for the 9901 and 3299 multiplexers
- Entry Assist
- Dual Logic, allowing 2 sessions to be used on a CUT terminal

There were a number of models, and, while I will not give the exact details on every one of these, I will provide a table of the models:

Table 8. IBM 3274 Models			
Model	***Attachment***	***Supported Processors***	***Max Ports***
3274-A01	local SNA	S/370, 30xx, or 4300	32
3274-B01	local non-SNA	S/360, S/370, 30xx, or 4300	32
3274-C01	local SNA or remote SNA	S/360, S/370, 30xx, or 4300	32
3274-D01	local non-SNA	S/370 or 4300	32
3274-21A	local SNA	S/370, 30xx, or 4300	32
3274-21B	local non-SNA	S/360, S/370, 30xx, or 4300	32
3274-21C	remote SNA (ASCII or EBCDIC)	S/3, S/360, S/370, 30xx, 303x, 3081, 308x, 3090, 4300, 8100, or 9370	32
3274-21D	local non-SNA	S/370 or 4300	32
3274-31A	local SNA	S/370, 30xx, or 4300	32
3274-31C	remote SNA (ASCII or EBCDIC)	S/3, S/360, S/370, 30xx, 303x, 3081, 308x, 3090, 4300, 8100, or 9370	32
3274-31D	local non-SNA	S/370 or 4300	32
3274-41A	local SNA	S/370, 30xx, or 4300	32
3274-41C	remote SNA (ASCII or EBCDIC)	S/3, S/360, S/370, 30xx, 303x, 3081, 308x, 3090, 4300, 8100, or 9370	32
3274-41D	local non-SNA	S/370 or 4300	32
3274-51C	remote SNA	S/3, S/360, S/370, 30xx, 303x, 3081, 308x, 3090, 4300, 8100, or 9370	12
3274-61C	remote SNA	S/3, S/360, S/370, 30xx, 303x, 3081, 308x, 3090, 4300, 8100, or 9370	16

3174: The 3174 (introduced in 1986) was a vastly-improved and simplified iteration of the 3274. Gone was the 8-inch floppy, the new 3174 booted off of 5.25" floppies. Likewise, the design was adapted to be more generic: the largest floor-standing models had 10 slots, which could be populated with either channel adapters or serial cards (depending on whether or not the unit was a local or remote configuration), or coax adapters. The local models were bimodal -- that is, they could be either local SNA or local non-SNA.

Over the course of the lifetime of the 3174, several releases of the "Configuration Support" firmware were made available:

Configuration Support A, the first firmware released alongside the 3174, supported everything that existed when it was released (so, synchronous serial, parallel channels, Token Ring). Software features included:

- Country Extended Code Page (for foreign languages)
- Multiple Logical Terminals (terminal multiplexing)
- Response Time Monitor (which incremented every second during an X SYSTEM wait period)
- Intelligent Printer Data Stream

Configuration Support S, which followed A, allowed either a local or remote controller to function as a Token Ring DSPU Gateway system, wherein up to 80 downstream physical terminal units could be attached over a network.

Configuration Support B, which expanded the TR DSPU to 250 downstream PUs and introduced a feature that alleviated VTAM's need to perform polled I/O on the channel: Group Polling.

Configuration Support C, the final version(s), which added neat features like:

- ISDN support (answering; PCs would usually originate the call)
- APPN SNA support
- 3174 Peer Communication (allowed PCs attached via coax to access a Token Ring LAN controlled by the controller)
- 5250 terminal emulation
- TCP/IP TN3270 and normal Telnet support
- Ethernet support

The following models existed:

Table 9. IBM 3174 Models

Model (Year)	*Attachment*	*Coax (ASCII) Ports*	*Form Factor*
3174-01L (1986)	Channel	4/32, 0/24	Floor
3174-01R (1986)	V.21/TR	4/32, 0/24	Floor
3174-02R (1986)	X.21	4/32, 0/24	Floor
3174-03R (1986)	TR	4/32, 0/24	Floor
3174-51R (1986)	Synch Serial/TR	9/16, 0/0	Desktop
3174-52R (1986)	X.21	9/16, 0/0	Desktop
3174-53R (1986)	TR	9/16, 0/0	Desktop
3174-81R (1986)	Synch Serial	4/8, 0/0	Desktop
3174-82R (1986)	X.21	4/8, 0/0	Desktop
3174-11L (1989)	Channel	4/64, 0/24	Floor
3174-11R (1989)	Synch Serial	4/64, 0/24	Floor
3174-12R (1989)	X.21	4/64, 0/24	Floor
3174-13R (1989)	TR/Ethernet	4/64, 0/24	Floor
3174-61R (1989)	Serial/TR/Ethernet	9/16, 8/8	Desktop
3174-62R (1989)	X.21	9/16, 8/8	Desktop
3174-63R (1989)	TR	9/16, 8/8	Desktop
3174-91R (1989)	Synch Serial	4/8, 0/0	Desktop
3174-92R (1989)	X.21	4/86, 0/0	Desktop
3174-21R (1991)	Serial/TR/Ethernet	4/64, 0/0	Rack
3174-21L (1991)	Channel	4/64, 0/0	Rack
3174-22R (1991)	ESCON/TR/Ethernet	4/64, 0/0	Rack
3174-22R (1991)	X.21	4/64, 0/0	Rack
3174-23R (1991)	TR	4/64, 0/0	Rack
3174-12L (1991)	ESCON/TR/Ethernet	4/64, 0/24	Floor
3174-90R (1991)	TR	1/1, 0/0	Desktop
3174-91R (1991)	Synch Serial	4/8, 0/0	Desktop
3174-92R (1991)	X.21	4/8, 0/0	Desktop
3174-14R (1993)	Ethernet	4/64, 0/24	Floor
3174-24R (1993)	Ethernet	4/64, 0/24	Rack
3174-64R (1993)	Ethernet	4/16, 0/8	Desktop

Combination Controller/Displays: There were two of these made, and these combined the functions of a controller and a terminal in some fashion:

- 3275, a combination remote SNA controller and terminal, sometimes used for a system console with one extra optional printer attached(1971)
- 3276, a combination remote SNA controller and a terminal, attached directly to a 2703/3705 or to a synchronous serial modem (just like the 3275) but could attach up to 6 terminals or printers (by way of modules that could be installed in the 3276 controller box that sat underneath the terminal)

3299 Multiplexer: This external box worked with (originally) a 3274 equipped with the 9901 Multiplexer Feature. This box allowed a single coax port on the 3274 to provide 8 coax ports to downstream devices, providing a great increase in density.

3270 Display Terminals

Replacing the earlier 2260 (which we will discuss next), the 3270 is a block-oriented terminal (rather different from the VT100-clone terminals you're probably used to emulating with programs like xterm) and printer system. While real 3270 terminals are long gone and the protocol lives on as Telnet 3270 (or, more commonly, TN3270), it is interesting to study where all of those classic mainframe green screens came from.

Being introduced in 1971 (as mentioned in the Controllers section), the 3270 line was intended to provide the highest number of terminals on the mainframe possible. That meant the controllers had to do a lot of work, but so did the terminals! The effect of the extremely efficient design was that a single mainframe with 16 MB of main storage could supposedly support 17,500 terminals on CICS with this product line.

As partially discussed earlier, the 3270 terminals interface with a host controller in one of two methods:

- Coax connection with a 93 ohm RG-62 cable fitted with BNC connectors; one terminal per cable (alternatively, with a 150 ohm balun that converts the coax BNC plugs to something that can take a phone line) -- this interface can transfer 2.3587 megabits per second.
- Direct attachment to a 37x5 or 4331 communications controller's serial port (this is the case for the 3275 and 3276).

There were two operating modes for these terminals:

- ***Control Unit Terminal (CUT)*** mode, wherein the terminal data stream would be consumed by the controller itself; the controller would then instruct the terminal to move the cursor to a position on screen, write a character with attributes, read keys from the keyboard via polling, so on and so forth. The characters transferring downstream to or from the terminal are not in the EBCDIC character set, but instead a special “3270 data stream.”
- ***Distributed Function Terminal (DFT)*** mode, wherein most of the data stream is forwarded off to the terminal -- this, naturally, requires a more sophisticated terminal, but means the display will interpret the 3270 data stream protocol itself. This allowed for extended attributes, more colors, and, most interestingly, full vector graphics and custom programmed-symbol fonts!

Some terminals were CUT terminals, and others were DFT terminals. Alas, there were many models of 3270 made. Here are some of them in quick succession:

3277: Essentially the first 3270 terminal, the 3277 was a rather interesting product! Not only could you get one of three kinds of keyboards (a basic keyboard which could only type in uppercase, a text keyboard that could type in mixed-case or APL, and special keyboard which only really had a numeric keypad), but you could also get a selector pen (a type of light pen that allowed for point-and-click behavior in some applications), or even ASCII character set function. The other models in the series were:

- Model 1: 40x12
- Model 2: 80x24 (a successful product)
- Model GA (with a RS232 serial port intended to drive either a Tektronix 4013 or 4015 graphics display)

3278: Improving on the 3277 in 1979, the 3278 had a much-improved keyboard and new graphics/datastream features. The 5 models were as follows:

- Model 1: 80x12
- Model 2: 80x24
- Model 2A: 80x24, but with the bottom 4 lines reserved (these were special console terminals)
- Model 3: 80x32 or 80x24 (changable with a switch)
- Model 4: 80x43 or 80x24
- Model 5: 132x27 or 80x24

Among other design improvements, the 3278 also introduced an extended highlighting system. Fields could blink, be inverted, be underscored, or have a different character set. There was also support for programmed symbols, a means of loading custom fonts (but could also be used to display monochrome bitmaps with some cleverness). There was an interesting design flaw that, during a programmed symbol set load, some models of 3278 or 3279 would have green streaks flash across the display... the so-called “green lightning.” This looked like a fuzz of green lines, and occasional blue lines would flash in the status area.

3279: The third generation of 3270s came in 1979. These featured color, and the following models were made:

- Model 2A: 80x24, base color
- Model 2B: 80x24, extended color
- Model 2C: 80x24, base color, 4 lines reserved (this was a console terminal)
- Model 3A: 80x32, base color
- Model 3B: 80x32, extended color
- Model S3G: 80x32 extended color, with programmed symbol set capability

Base Color meant there were 4 colors. This resulted in a color pattern like this:

Table 10. Base Colors

Protection	***Intensity***	***Color***
Unprotected	Normal	Green
Unprotected	Intensified	Red
Protected	Normal	Blue
Protected	Intensified	White

Extended Color allows for a expanded data stream, and the following colors were possible:

- Neutral/white
- Red
- Green

- Blue
- Green
- Pink
- Yellow
- Turquoise

In order to allow for the graphics features to be sufficiently utilized, IBM wrote a program package called Graphical Data Display Manager (or, more commonly, GDDM) at the Hursley Development Laboratory (outside of Winchester, England; the machines it was developed on all bore names starting with WINVM).

3290: The 3290 is a rather interesting terminal, or, as per its actual name, an Information Panel! Launched on March 8, 1983, the 3290 used a beautiful orange gas plasma display and could run in one of two modes:

- Four distinct 3278 Model 2 terminals
- A single terminal, with a massive 162x62 size

You could also use programmed symbol graphics on it, but the only caveat was that the controller it was connected to had to support the Downstream Load feature (since the 3290 had to load its microcode from the controller).

3178/3179: In the era of simplified electronics design, IBM launched the 3178 in April 1983. This consisted of a box that sat underneath a monitor, and a PC-derived keyboard. These were mildly successful, but were replaced a year later.

On March 20, 1984, IBM announced the replacement to the 3279: the 3179. These were lower-cost and had simpler electronics. There were 3 models:

- 3279 Model 1: 80x24, no graphics
- 3279 Model G1: 80x24, graphics
- 3279 Model G2: 80x43, graphics

The graphics support on the 3179G models was quite desirable, as it had what IBM referred to as All Points Addressable (APA) graphics -- that is, you could deposit pixels on the display anywhere with a resolution of 720x384. This terminal also supported vector graphics, and the vector graphic rasterization was performed on the terminal itself. These were terminals that required a Downstream Load from the controller.

3180: Launched on March 20, 1984, the IBM 3180 is one of the most interesting and complex terminals. There were several submodels:

- Model 2 and Model 2+: 80x24, basic or extended mode
- Model 3 and Model 3+: 80x32, basic or extended mode
- Model 4 and Model 4+: 80x43, basic or extended mode
- Model 5 and Model 5+: 132x27, basic or extended mode

Basic mode provided a simple terminal, and extended mode allowed for the screen to be resized, rotated, or a small partition could be created by the application. These were also intended to connect to a System/36 (the precursor to the AS/400 for those not familiar with this platform), and were monochrome green displays.

3191: Introduced in June 1986, the 3191 was a monochrome green or amber terminal that could connect to a System/370 or an 8100. There were two model groups:

- Model A/B: 80x24

- Model D/E/L: 80x24 or 80x43

3192: Essentially a (sometimes) color version of the 3191, the 3192 launched in January 1987. As usual, there were several models:

- Model C: 80x24 or 80x32
- Model D: Monochrome green, 80x24/80x32/80x44/132x27
- Model F: 80x24/80x32/80x44/132x27
- Model G: 80x24 or 80x32, graphics support
- Model L: Monochrome green, 80x24/80x32/80x44/132x27, but with a light pen
- Model W: Monochrome white, 80x24/80x32/80x44/132x27

The 3192G graphics model supported 16 colors and could also attach to an IBM Proprinter for hardcopy output.

3193: In 1983, IBM launched something that could be seen as a bit of a graphics workstation. Intended to connect to a scanner and a printer, the 3193 was actually a high-resolution portrait-orientation monochrome display. A US Letter or A4 document could be displayed on the CRT, and the attached scanner would transmit a compressed datastream to the terminal. There were 2 submodels, with Model 1 having a 122-key keyboard, and Model 2 having a 102-key Enhanced PC keyboard.

The screen was a whopping 880x1152 pixels, with a line resolution of 80x48. Two logical terminals could be used, and the image data stream was supported (which GDDM supported).

3194: More of a "smart workstation," the 3194 was a desktop-computer look-alike that had a 1.44MB PC floppy drive (where data could be transferred to and from the disk by using the IND$FILE transfer program on the host). There were 3 models:

- Model C: 80x24 or 80x32, color (12-inch monitor)
- Model D: 80x24/80x32/80x44/132x27, monochrome (15-inch monitor)
- Model H: 80x24/80x32/80x44/132x27, color (14-inch monitor)

3104: A variant of the 3178, the 3104 was intended to only connect to an 8100 through an R-loop. Little seems to be known about this product.

3472: The IBM InfoWindow was introduced in 1989, and was the final dedicated 3270 terminal before PCs running emulators took over. These terminals supported graphics fully, and could run 5 concurrent sessions! There were a variety of devices that could be attached without any difficulty:

- Mouse
- Color plotter
- Graphics tablet
- Barcode scanner

Printers: One of the other features provided by the 3270 line was support for printers. There were several printers made over the years:

- 3284 matrix printer
- 3286 matrix printer
- 3287 matrix printer (there was a color model available)

- 3288 line printer
- 3268-1 (an R-loop printer for an 8100)
- 4224 matrix printer

RJE Terminals

In the batch era (which IBM certainly prolonged), having cheap remote batch terminals with a card reader/card punch/line printer was a cheap way to expand the geographical footprint of a computer. These came from IBM in the form of various Remote Job Entry workstations, which attached to the host computer through a variety of communications processors (described below). These connections were made in the form of bisync connections or STR (Synchronous Transmit/Receive, an early form of synchronous serial from IBM) connections, either directly to the computer or through a dial modem, leased-line modem, packet-switched X.25 network, or some other such similar connection.

RJE was supported by most batch spooling programs on the System/360 line; this included HASP (JES2, OS/360), ASP (JES3, OS/360), RES (JES1, OS/VS1), POWER (VSE), RSCS and (VM). Some places took RJE and ran it over other networks, like UCLA's NETRJS that ran on the pre-TCP/IP ARPANET (and later on the TCP/IP ARPANET). RJE was also ran over SNA. The RJE terminals themselves, much in the same way as the 3270 terminals, were often emulated by minicomputers/microcomputers.

In the pre-emulation era, IBM made the following batch terminals:

2780 Data Transmission Terminal: This was released in 1967, and was itself comprised of four major sub-units:

- A bisync controller box, these would provide the minimum amount of electronics necessary to talk to the host computer
- A card reader and punch, derived from the IBM 1442 described earlier; this particular iteration read 400 cards per minute and punched 355 cards per minute
- A line printer, derived from the IBM 1403, that printed 240 lines per minute (though the chain could be shortened to a limited character set to increase print speed to 300 lines per minute)
- A line transmit/receive buffering box that buffered up to one print line or one read/punched card

There were in turn four different models made:

- Model 1: capable of reading cards and printing output from the host computer
- Model 2: capable of reading cards, punching cards, and printing output from the host
- Model 3: capable of only printing data from the host
- Model 4: capable of reading and punching cards, but not print

2770 Data Communication System: Though not strictly a successor to the 2780 and more of an alternative, the 2770 was announced in 1969 and supposedly surpassed every other IBM terminal made to that point in input and output capabilities. The 2770 came with an included CRT terminal and keyboard, and could drive several downstream devices:

- 2265 Display Station (see the previous section on the 2260 for more on this)
- 2502 Card Reader (model A1 or A2)
- 5496 Data Recorder (punched the 96-column cards for the System/3)
- 1053 Printer (model 1)

- 1017 Paper Tape Reader (5/6/7/8 column, 120 chars/second)
- 1018 Paper Tape Punch
- 545 Card Punch (model 3, non-printing; model 4, printing)
- 50 Magnetic Data Inscriber
- 1255 Magnetic Character Reader (model 1/2/3)

3780: The successor to the 2780 was the 3780, launched in 1972. Unlike the predecessor, there was a single model of this (but one could get an optional card punch with it). The built-in card reader read cards at 600 cards/minute, the punch (the 3781) punched 91 cards/minute maximum, and the printer printed 300 lines/minute minimum.

The 3780 was extensively emulated by various other computers, like the DEC DN60 in the late 70s. Software implementations were also relatively common.

3770: This was the ultimate batch terminal, released in 1974. Succeeding the 3780, these supported a variety of communications protocols:

- SDLC
- BSC
- Multileaved BSC
- SNA

This was a series of desk-console terminals that existed in the following models:

- 3771, with a wire matrix printer:
  - Model 1: 40 cps printer
  - Model 2: 80 cps printer
  - Model 3: 120 cps printer
  - *All models could feature an optional reader and punch*
- 3773, with a diskette and wire matrix printer:
  - Model 1: 40 cps printer
  - Model 2: 80 cps printer
  - Model 3: 120 cps printer
- 3774, with a wire matrix printer (and optionally a belt printer):
  - Model 1: 80 cps printer
  - Model 2: 12 cps printer
- 3775, with a belt printer (and optionally a diskette):
  - Model 1: 120 lpm printer
  - Model P1: included a display terminal
- 3776, with a belt printer
  - Model 1: 300 lpm printer
  - Model 2: 400 lpm printer

- 3777, with a train printer
  - Model 1: 1000 lpm printer
  - Model 2: adds a card punch
  - Model 3: adds a tape drive
  - Model 4: replaces the printer with a slower 3262

Early OSes - BPS/360, BOS/360, and TOS/360

BPS/360

In the early days of computing, you had to write nearly all of your software. You might have a vendor-provided compiler for a high-level language, an assembler, or nothing at all! When IBM launched the 360, everyone had decided that computers being sold in that era should come with some kind of operating system. IBM filled that void with, initially, two OS offerings: for the smaller machines (that is, the Model 30 and machines configured without much main storage), BPS/360; for the larger systems, the customer could select BOS/360 or later systems like TOS/360 and DOS/360.

BPS/360 (Basic Programming Support) was a combination of two OS releases, both launching with the S/360 in mid-1965:

- A punched-card edition, requiring no permanent storage devices
- A tape edition, loading a small resident supervisor and using tapes

The card version of BPS/360 was essentially many decks of standalone utility programs. Disks and tapes could be initialized from certain decks, an assembler could be loaded from one deck, and high-level languages were available too: RPG (Report Program Generator) and a subset of FORTRAN IV were on offer -- the tape version of the FORTRAN compiler required 16 KB of storage, and the tape version of the assembler likewise had more features (setting up the trend of the card versions of the utilities being "worse" than the tape versions). Alas, BPS/360 was a bit of a failure. By the time BPS/360 was released, disks were starting to become rather cheap -- this is definitely in direct contrast to the computing landscape of the IBM 1401 and IBM 7090 precursor errors, were disks were seen as a rare luxury.

BOS/360

Seeing the need for a real proper OS, IBM launched BOS/360 alongside the System/360. There were initially three versions of BOS/360, which would later end up being renamed:

- BOS 8K for disk systems --> BOS
- BOS 16K for disk systems --> DOS
- BOS 16K for tape systems --> TOS

At the time of writing this, there exists a copy of the third system on Bitsavers; it is labelled BOS, but is actually an early version of TOS. The renaming of the three products occurred in May 1966, and manuals from the time period can be somewhat obtuse if the reader does not have a keen eye for the tape size being used. Alas, the 16K disk system would go on to live a long and prosperous life (and still remains alive today). Please do note that I (your author, Evie) am not 100% sure that this information is correct -- there appears to be conflicting information online and in manuals.

BOS/360 provided the following facilities:

- A supervisor, a punched-card job loader, and an IPL loader (necessary to control and operate the system)
- A linkage editor, a librarian, and a system generation utility named the Load System Program (all necessary for maintaining the system)
- An assembler, an RPG compiler, and FORTRAN, COBOL, and PL/I compilers (for application development)
- Sort/merge, a debugging aid called Autotest, various file utility programs, and a remote job entry (RJE) client

Alas, BOS/360 proper (i.e. the 8K disk system that would run off a 2311 DASD disk drive), is merely a simplistic version of the later DOS/360 system that would be released shortly thereafter. For users without disk drives and only tape drives (provided they had enough storage), TOS/360 was a choice. Both DOS/360 and TOS/360 shared many similarities; there were at least 26 releases of TOS before its discontinuation, but DOS continued for at least 26 releases. *Note: I am not entirely sure on the accuracy of these version numbers -- I kept running into conflicting information in the composition of this work.*

DOS/360, VSE, and z/VSE

DOS/360 and TOS/360

DOS/360 and TOS/360 (even though TOS/360 died off) featured a rather odd memory model -- since the BOS derivatives were considered to be a "stop-gap OS" until OS/360 came out (which we will discuss next), the architecture of the system can best be described as "antiquated." Rather than having a memory-management model reminiscent of other timesharing OSes like VMS where a program is free to grow its virtual address space, DOS/360 and TOS/360 presented a variety of fixed memory partitions within the non-virtual address space. DOS/360 did not have a relocating loader, so all of the address offsets within programs had to be made absolute when the program was linked -- this meant that a program that was link-edited (the "link editor" being the old term for what we now call just a "linker") for partition 0 would not run in partition 3. Since there was no virtual memory, a malfunctioning program in one partition could spell the doom of another partition; the multiple partitions were merely for timeslicing at the scheduler level.

Strangely enough, early versions of DOS/360 did not have a batch spooler; a program would have to be started by hand (but multiple could be ran from the console). Needless to say, this was rather antiquated even for DOS/360 standards; IBM introduced POWER (an acronym for Priority Output Writers, Execution processors and input Readers), and Software Design, Inc sold a spooler program called GRASP. *Side note: I have read conflicting dates as to when POWER was introduced, and I cannot confirm that its existence predates GRASP.*

Alas, DOS/360 (while TOS/360 languished and eventually died) featured quite a number of successful third-party applications, but the eventual release of OS/360 made DOS look more and more obsolete. Alas, many customers continued to use DOS and it kept being upgraded alongside the hardware. With the three memory partitions the system offered (background, foreground 1, and foreground 2), users found the simplistic system to be quite good for what it was.

Interesting fact I cannot verify: Dave's guide states that BOS/360 had a spooler, while DOS/360 and TOS/360 did not!

DOS/VS

As the System/370 matured in 1973 with the addition of the DAT box on all models (therefore giving the machines virtual storage, i.e. memory, support), DOS sought an upgrade. While other teams elsewhere at IBM were working on upgrading other OSes, DOS remained a strange fixture as it was still rather antiquated in its design. When DOS/VS was introduced in 1972 (the first version being Release 28; Release 27 was the last release of DOS/360), users now could run up to 5 memory partitions (BG, then F1 through F4; F for Foreground, later colloquially known as Fixed)! Alas, the primitive OS still had the strange non-relocating-loader limit; though the system was technically running within a virtual storage address space, it was more of a case of "there is a single linear virtual address space containing all the partitions (allowing for paging on and off of DASD), but each partition still is lined up back-to-back and therefore programs must be link-edited for the partition they will run in." This "issue" would be fixed by VSE/SP, but I cannot find a conclusive date when it was.

DOS/VSE

In 1979, IBM introduced extensions to DOS/VS that allowed it to gain support for the then-new 4300 series; these machines contained a new feature called ECPS that would speed up I/O performance by sharing the storage address space the processor used with the I/O channels -- non-4300 systems could, of course, run DOS/VSE anyways. By then, compilers were getting more sophisticated, as were the programs being ran on the system; CICS/DOS/VS was the primary workload of choice on these systems, even though CICS/DOS/VS ran on DOS/VS. Alas, DOS/VS did not really contain advanced networking or online development facilities; programming CICS consisted of submitting

batch jobs (likely originating from another user under a VM system, which will be discussed later) and submitting another batch job to make the transaction available to online terminals. When DOS/VSE became available, it began to gain a number of solutions to these problems starting in the early 80s.

SSX/VSE

During the life of DOS/VSE, IBM sought it necessary to produce a relatively "turnkey" package of the system. Small Systems Executive/VSE contained the following products, giving us an interesting list of the "who's-who" of DOS/VSE program products in 1982:

- VSE/AF (see below), an upgraded version of DOS/VSE
- VSE/POWER, the job scheduler
- ACF/VTAM, for SNA networking and terminal control (which was finally made available to DOS/VSE systems)
- VSE/VSAM, providing OS VSAM to VSE
- CICS/DOS/VS, an online transaction processing system
- VSE/ICCF, an online program development/test/compile/edit system that ran under CICS
- VSE/OCCF, a program that oversaw the system and restarted components if they failed
- VSE/IPCS, a debugger
- DOS COBOL, a COBOL compiler
- VSE Back Up/Restore, does what it says
- VSE/Space Management, used for maintaining DASD storage
- VSE/DITTO, used to transfer data and files from one device to another

VSE/AF

Launched in 1983, VSE/AF was seen as a much-needed upgrade to DOS/VSE. It should be noted how programs were stored and ran on DOS: a so-called "core image library" would be a dataset that contained multiple members, of which your programs were stored in. When a program was called, VSE would fetch the MYPROG core image (which had been already link-edited and was ready to run) from the library. Maintaining these libraries was necessary only for program-related tasks. VSE/AF Version 2 changed this. Rather than limiting libraries to just core images (i.e. programs), IBM suggested they could be used for any purpose -- source files, data files, anything. While this certainly streamlined storage management on VSE, it was also seen as somewhat of a radical change; CMS/DOS, a component of VM/CMS that provided DOS/VSE compatibility, could not use VSE/AF Version 2 and newer libraries.

VSE/SP

When the IBM 9370 launched, IBM saw it fit to produce a successor to SSX/VSE (which, by then, had remained stale; it was now 1986 and SSX/VSE was commonplace in 1983). VSE/SP combined all of the features of SSX/VSE, but updated them to the current era. VSE/SP systems often found themselves running under VM (which was somewhat commonplace at that time), and it saw extensive transaction-processing usage on smaller systems that did not run OS (or MVS in that era). VSE/SP was also a 24-bit system; there was no VSE/XA.

VSE/ESA

As VSE aged and the shops that ran VSE strove to push the system harder and harder, VSE was due for a major architectural upgrade around 1990 -- this came in the form of VSE/ESA, which assumed the role of a 370/XA upgrade for VSE/SP. In that era, there was still a desire to run what would have then been considered "old and obsolete" 370-only machines (such as 9370s or old 4300 series machines), so the new VSE/ESA had both a 31-bit and 24-bit nucleus (31-bit being the colloquial term for *370/XA native*). VSE/ESA Version 1 was bimodal, whereas Version 2 required ESA/370 hardware. When Version 2 was released, it included a scheduler called the Turbo Dispatcher that finally allowed a multiprocessor system to meaningfully be utilized with a single VSE image (multiple partitions could run simultaneously across multiple processors in the machine); up to 4 processors could be adequately used, but the VSE/ESA nucleus could accommodate up to 10.

VSE/ESA also overhauled the partitioning scheme. While later versions of DOS/VSE (and therefore VSE/AF and VSE/SP) added a feature to the link editor to produce a relocatable module, the addresses within each partition were still technically shared -- this meant that the BG partition could run from X'0000' to X'01FF' and the F1 partition would run from X'0200' to X'05FF' (these are not real numbers, merely an example) and it was up to the DAT box to stop a program in BG from clobbering the memory of F1. VSE/ESA addressed this, and also allowed for up to 384 MB of storage to be used.

First, each partition (of which there were 12 so-called static partitions, sometimes called fixed partitions) now had their own private virtual address spaces -- every partition's memory address scheme started at 0 and it was no longer a strict requirement to make every program position-independent and relocatable. In addition, VSE gained a new type of partition known as a dynamic partition, wherein programs that would spill out of a static partition's address range could run (but also physically reside in memory past the 16 MB bar). VSE/ESA Version 1 could run up to 150 dynamic partitions.

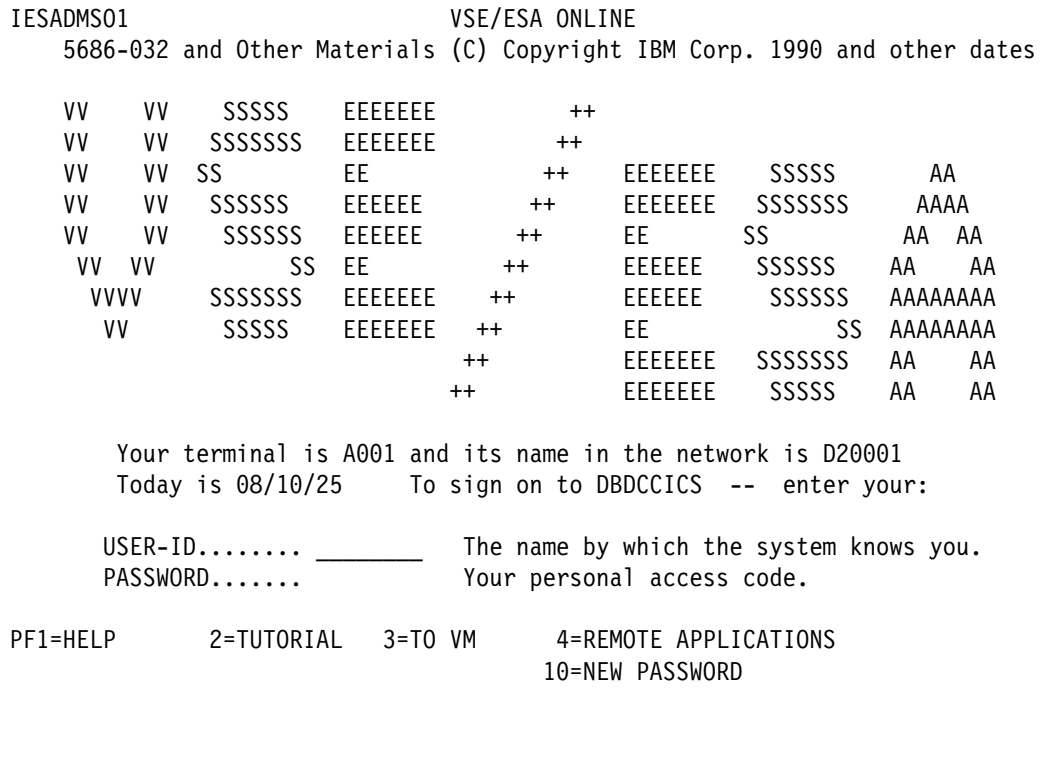


Figure 1. VSE/ESA 1.1.0 logon screen

```
IESADMSL.IESEPROG         VSE/ESA FUNCTION SELECTION
                                                        APPLID: DBDCCICS
  Enter the number of your selection and press the ENTER key:

        1  Program Development Library
        2  Manage Batch Queues
        3  Create Application Jobstream
        4  Problem Handling
        5  Operations
        6  File Management
        7  Command Mode
        8  CICS-Supplied Transactions









PF1=HELP                  3=SIGN OFF                    6=ESCAPE(U)
                          9=Escape(m)

==>
```

Figure 2. VSE/ESA 2.7.0 programmer-type menu

Something of particular interest is a little-known controversy surrounding TCP/IP on VSE. For most of its life, VSE was notably anemic in its networking facilities outside of mainframe-native network protocols -- VSE had support for SNA and NJE (the two stalwarts of mainframe networking for most of its existence), but it struggled to "get with the times" when TCP/IP came around. Furthermore, no version of VSE ever supported the OSI protocol, something that both MVS and VM did support (in this context, the "OSI protocol" refers to the OSI protocol family with CLNS/CLNP, TP0-TP4, etc; this protocol is completely unrelated to the TCP/IP stack). Throughout its lifetime, there were at least 2 VSE-native TCP/IP stacks:

- Connectivity Systems International's TCP/IP for VSE, providing IPv4 support only
- Barnard Software's IPv6/VSE, providing both IPv4 and IPv6

IBM would bundle the installers for both stacks after VSE/ESA had been replaced by its successor, but the customer still had to separately license the stack.

```
PING 1.1.1.1
TCP200I Client -- Startup --
TCP207I Copyright (c) 1995-1999 Connectivity Systems Incorporated
TCP202I Attempting to Establish Connection
TCP204I Connection has been Established
Client manager connected. Generated on 04/08/02 at 02.00
Selected client: PING
001.001.001.001
PING 1 was successful, milliseconds: 00009.
PING 2 was successful, milliseconds: 00009.
PING 3 was successful, milliseconds: 00008.
PING 4 was successful, milliseconds: 00008.
PING 5 was successful, milliseconds: 00008.
Client completed: PING
TCP201I Client -- Shutdown --
TCP205I Connection Complete -- Already Closed
```

Figure 3. CSI TCP/IP stack running a ping from a blank CICS screen

z/VSE

When the System Z launched in 2000, there was a notable lack of a new VSE system until 2005. z/VSE 3.1 was released, but it contained no new features over VSE/ESA 2.7 that came just before it -- it did not run in 64-bit mode, and it also did not require Z hardware. In 2007, IBM did release a Z-native version of the OS: z/VSE 4.1. It actually did support the 64-bit operating mode, and the user could use up to 8 GB of storage with it. Though user programs can use 64-bit registers, no user program will ever actually see a 64-bit address in any partition -- the supervisor only allowed user programs to see 31-bit virtual addresses within the partitions. A new set of compilers was released for the new release, though they were really just rebadges of the earlier ESA generation of compilers.

```
IESADMSO1                          z/VSE ONLINE
    5609-ZV4 and Other Materials (C) Copyright IBM Corp. 2005 and other dates


                                  ++
                                 ++    VV    VV    SSSSS    EEEEEEE
                                ++     VV    VV   SSSSSSS   EEEEEEE
                  zzzzzz       ++      VV    VV  SS         EE
                  zzzzz       ++       VV    VV   SSSSSS    EEEEEE
                    zz       ++        VV    VV    SSSSSS   EEEEEE
                   zz       ++          VV  VV          SS  EE
                  zzzzzz   ++            VVVV     SSSSSSS   EEEEEEE
                 zzzzzzz  ++              VV       SSSSS    EEEEEEE

        Your terminal is CL20 and its name in the network is Z62LCL20
        Today is 08/10/2025   To sign on to DBDCCICS  --  enter your:

       USER-ID........ ________   The name by which the system knows you.
       PASSWORD.......            Your personal access code.

 PF1=HELP       2=TUTORIAL               4=REMOTE APPLICATIONS
                                        10=NEW PASSWORD
```

Figure 4. z/VSE 4.1.0 logon screen

```
 SYSTEM:  ZVSE41              z/VSE 4.1         TURBO (01)       USER:  SYS
 VM USER ID:VSE                                                 TIME:  1 8:53:34
 F1 0001 1Q14I  NO MATCHING PUB FOR 030
 F1 0001 1QH3I   146 OF 742 DBLK GROUPS LOST
 d q
 AR 0015 1C39I COMMAND PASSED TO VSE/POWER
 F1 0001 1R49I  QUEUE FILE 003% FULL - 1266 FREE QUEUE RECORDS
 F1 0001 1R49I  USED QUEUE RECORDS: 44, CRE-Q: 3, DEL-Q: 0
 F1 0001 1R49I  RDR-Q: 25, LST-Q: 16, PUN-Q: 0, XMT-Q: 0
 F1 0001 1R49I  QUEUE FILE EXTENT ON FBA-250, SYS001, 59904, 1024
 F1 0001 1R49I  DATA FILE 029% FULL - 530 FREE DBLK GROUPS
 F1 0001 1R49I  CURRENT DBLK SIZE=07680, DBLK GROUP SIZE=00008
 F1 0001 1R49I  DATA FILE EXTENT 1 ON FBA-251, SYS002, 376320, 89088
 F1 0001 1R49I  ACCOUNT FILE 37 % FULL
 F1 0001 1R49I  ACCOUNT FILE EXTENT ON FBA-251, SYS000, 465408, 2048




 ==>

 1=HLP 2=CPY 3=END 4=RTN 5=DEL 6=DELS 7=RED 8=CONT 9=EXPL 10=HLD         12=RTRV

 ACT_MSG: HOLDRUN           PAUSE: 01  SCROLL: 1           MODE:  CONSOLE
```

Figure 5. z/VSE 4.1 system console

VSEn

In 2022, a non-IBM company took over the development of z/VSE. This company, known as 21st Century Software, released VSE^{n}. Since 21CW longer could use IBM trademarks, every program product on the system was promptly renamed to an alternate name; DITTO became SMDMU, VTAM became VCDD, so on and so forth. Otherwise, VSE^{n} was essentially a "maintenance release;" the days of new VSE features are long since gone, with most of the development on VSE happening in the realm of the "connectors" -- workstation utilities that make administering, programming, and operating VSE systems a bit easier for a novice user not familiar with the panel-driven ICCF user interface.

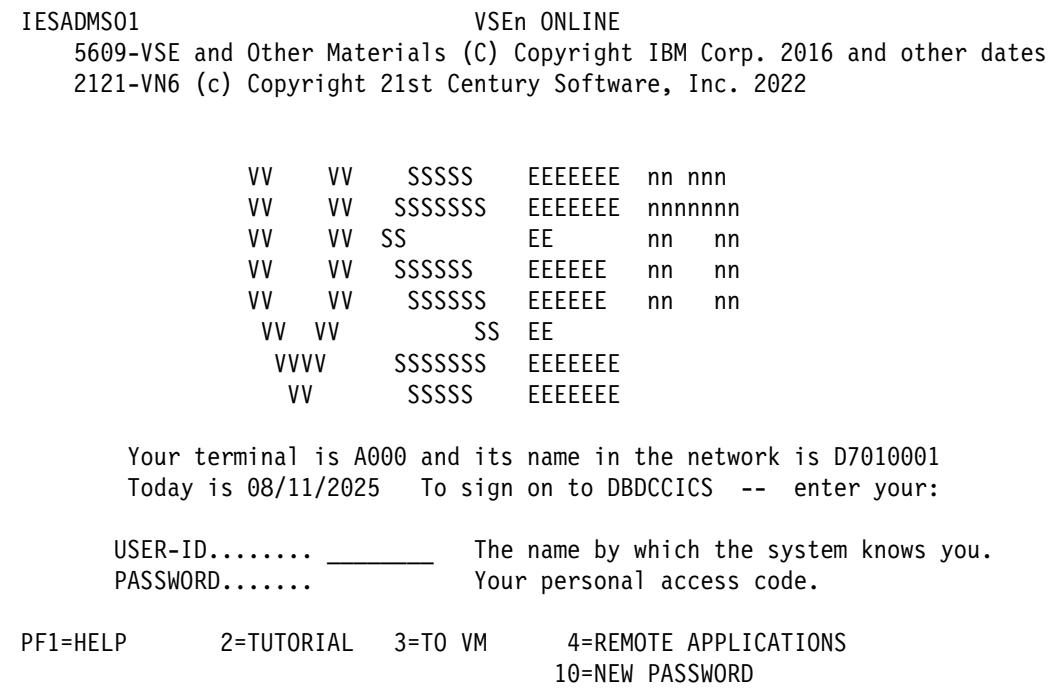


Figure 6. VSEn 6.3.0 logon screen

OS/360, MVS, and z/OS

Initially, before the System/360 was shipped, IBM hoped that there would be a single batch OS and a single timesharing OS (TSS/360) for users to select between. Instead, this was not the case; during the development run of the System/360, IBM had not yet finished the "ultimate batch OS" -- OS/360. As a matter of fact, its development was dragging so far behind that they were forced to release the aforementioned BOS/360 and BPS/360 initially. In other words, the BOS/360 derivatives (that is, DOS/360 and TOS/360) were intended to be *temporary* operating systems; stopgap measures to hold over customers until OS/360 released. Sadly, due to major ABI and architectural differences, users found it rather difficult to port programs from DOS/360 to OS/360 when it eventually did come out.

OS/360's development was so laggy that its release came in three major waves:

OS/360 PCP

Because OS/360 had logged behind so much in its development, IBM was desperate to get something shipped. This came in the form of the OS/360 Primary Control Program, bearing the Single Sequential Scheduler (SSS). In 1966, customers could finally get a tape for what was essentially a single-tasking OS. PCP was a sequential-schedule system where a job would be submitted from the reader, and rudimentary spooling facilities meant that the reader would have to be pre-stacked with every job that the system would run. Needless to say, this was seen as a rather useless system by nearly every 360 customer.It did have at least one redeeming quality, though: if your mainframe had 48 KB of storage, you could run PCP and have the facilities of OS/360 without the bloat of a big multi-partition scheduler. For customers that had CP-40 or CP-67 (which will be discussed later), PCP provided a storage-light single-user development-capable OS system.

OS/360 MFT

For users that had a more advanced ("midrange 360") system, they could choose to run OS/360 MFT -- Multi-tasking with a Fixed number of Tasks. This option was called "Multiple Sequential Schedulers" (MSS), and was also a temporary holdover measure until the third scheduler option was finished. MFT, being a more advanced version of PCP, provided you with 4 memory partitions (similar to how DOS worked). It was notoriously difficult to manage multiple partitions by hand from the console (as the operator would have to manually direct jobs from the reader into a partition such that the OS/360 nucleus could parse the JCL and run the job), so nearly every site chose to install a program called HASP -- Houston Automatic Spool Priority (as in, "Houston, we have a problem") alleviated this, and it would live on in the MVS line as JES2 (which we will cover later).

When MFT was replaced with MVT (see below), MFT received a major upgrade called MFT-II: gone was the 4-partition limit, you now had 52 at your disposal! The final releases of MFT-II actually eliminated the task limit, so stalled jobs could wait in a queue and eventually be dispatched to an available partition.

OS/360 MVT and TSO

After much aforementioned delay, OS/360 MVT came out -- Multitasking with a After much aforementioned delay, OS/360 MVT came out -- Multitasking with a Variable number of Tasks! The scheduler was said to be of the "multiple priority schedulers" configuration, and could even run on a multiprocessor system if the customer had the expensive Model 65 Multiprocessing (often called the "Model 65MP"). Since MVT was not available until 1967, the OS had gained substantial complexity; users needed at least 512 KB of storage to run this now-released OS/360. Since MVT provided a non-fixed-partition memory management model, it was now possible to have various programs running at the same time with different memory usages. However, this introduced an interesting problem! Because the System/360s that MVT ran on did not have a DAT box to perform virtual memory mapping, users found that their MVT systems would have vastly-underutilized main storage after running many programs.

This was simply due to the fact that there was no virtual memory; the scheduler would employ a best-fit algorithm to try to find an available "memory hole" between two (or more) running programs in main storage, and the program could use as much storage as it wanted to until it hit the bound of the next program in the address space. This, of course, resulted in heavy memory fragmentation, particularly on timesharing systems.

Time Sharing Option: Speaking of timesharing, the ultimate failure of TSS/360 (which will be discussed later) resulted in a vacuum-void of a quality timesharing system (from IBM's perspective; mind you, CP-67 was well on its way to becoming the dominant timesharing system by this time). IBM's answer to this void came in 1971 with the release of the Time Sharing Option on OS/360 Release 20.1. Since IBM had many communications controllers and terminals at the time, there needed to be some kind of unified way to drive all of these devices; throughout the life of OS/360 (and DOS/360 too), there were a variety of "telecommunications access methods" (what we might nowadays might call "an API"):

- Basic Telecommunications Access Method (BTAM), providing little more than basic typewriter terminal control
- Queued Telecommunications Access Method (QTAM), allowing terminal devices to be operated like files
- Telecommunications Access Method (TCAM), an evolved QTAM with added message-passing and queuing support
- Virtual Telecommunications Access Method (VTAM), a true network interface and cross-system-network

When TSO was unveiled, it used the then-advanced TCAM to support a variety of terminals -- typewriter terminals, early glass terminals that used acoustic delay-line memory to store the screen (the 2260), and the later famous 3270 series. TSO was primarily seen as a massive aid to programmers, who could now use an online editor, job result output viewer, debugger, interactive compiler, and more in an online fashion. While the TSO command line could best be described as "hand-keyed JCL" and the TSO editor could best be described as "an exercise in getting lost on a computer," it provided a framework for better applications further down the road that would turn TSO into a proper timesharing experience (these being mainly the alphabet soup of ISPF, SDSF, SDF, and other such things).

Something of particular interest is the modularity of the system. Because PCP/MFT/MFT-II/MVT were merely just different "kernels" running the same underlying operating system, absolutely no changes were necessary to move programs from a machine running one to a machine running the other; even the batch job JCL that ran the programs would work. Programs running on the newest release of MFT and MVT could both spawn multiple subtasks, allowing a form of multithreading; both scheduler options provided some form of spooling (though HASP and ASP filled that void).

HASP and ASP: OS/360 was intended to be as modular as possible. One of the results of that was the creation of a universal interface for a spooler to hook into; the OS/360 nucleus itself would not be responsible for much spooling, instead deferring to a dedicated subsystem that would perform that and other tasks. There were two major programs that resulted from this: HASP and ASP.

HASP, or the aforementioned Houston Automatic Spooling Priority, was the first mature job scheduler for OS/360 MFT/MVT. It would live on as JES2 on the descendants of OS/360, and was written for the Lyndon B. Johnson Space Center in Houston, Texas. It was intended to run on a single machine with one or more processors, and newer versions of it supported remote workstations (something very important in that era, as computers were rare and a RJE workstation in a remote office was quite a useful deal in the 1960s. Eventually, HASP gained multi-access-spool support, allowing multiple machines connected to the same spooling disk to both execute queued jobs.

ASP (Attached Support Processor), on the other hand, was a much more complex beast. Living on today as JES3, ASP is a real loosely-coupled multiprocessor system that released first in March 1967; two S/360s would be connected via a CTCA and they did not have to be the same model. ASP also provided a level of failover should a system crash or malfunction. Perhaps the coolest feature in the product was the fact that it actually supported scheduling and dispatch of 7090/7094 jobs; the 360 was backwards-compatible with 709x thanks to an emulator program. The OS could choose to engage it, and ASP would: if it encountered a 709x job, it would invoke the emulator and run the job accordingly.

OS/VS1

When the System/370 was announced, a new OS was in order. IBM naturally sought to provide some kind of new OS. Learning much from the slow and difficult development of OS/360 and applying what they had already been doing with DOS/VS, they chose to produce OS/VS1 in 1972. This was, by all accounts, OS/360 MFT-II running within a single virtual address space. Much like DOS/VS was DOS/360 running within a virtual address space, OS/VS1 crutched on OS/360's relocating loader but provided virtual storage paging (what modern computer users might know as "swapping") to the entire address space.

OS/VS1 had no TSO, but there was a version of CICS available for it and every compiler that ran under the later OS/VS2 would run on OS/VS1. VS1 did have its own job scheduler called JES1 (Job Entry Subsystem 1) that was derived from the basic scheduler found on OS/360 MFT, but it was ultimately a failure (as many people had modified HASP and ASP in the OS/360 days, and wished to continue to use the software they had spent so much effort tailoring). JES1 was relatively novel in the sense that, unlike on MFT where every reader/writer (writer being a printer or punch) task took up its own partition, VS1 users could reclaim those partitions and only temporarily use a free partition when a job started or finished. MFT's scheduler also used multiple spooling files on the system DASD, but JES1 used a singular common spooling dataset -- this was a feature inspired by the spool/checkpoint model of HASP.

OS/VS1 BPE

Plain OS/VS1 had 7 releases, with 4 releases of an add-on product called Basic Programming Extensions. These added support for more hardware common on early-1980s System/370 machines (like 3380 DASD and newer printers), and allowed VM to handshake VS1's console with VCNA (the precursor to ACF/VTAM on VM). Release 4 of BPE was announced in March 1984, but VS1 continued to feature IBM service until the 28th of February, 1990 -- a stunningly-long time for a stopgap OS.

```
IPL 14A
IEA760A SPECIFY VIRTUAL STORAGE SIZE

IEA788I NON-PAGING MODE
IEE116A TOD CLOCK INVALID
IEA101A SPECIFY SYSTEM AND/OR SET PARAMETERS FOR OS/VS1 70.D  BPE 01.03.0
r 00,'date=80.180,clock=10.10.10,gmt'
IEE055A SPECIFY LOCAL DATE AND/OR CLOCK
r 00,'clock=10.10.10'
 IEF597I  SYSTEM RESTART IN PROGRESS
 IEF450I VMWTR   .VM00D   .         ABEND S2F3       TIME=03.41.01
 IEF421I INIT=VMWTR.VM00D (2) NO RESTART
 IEF450I VMWTR2  .VM00E   .         ABEND S2F3       TIME=03.41.01
 IEF421I INIT=VMWTR2.VM00E (2) NO RESTART
 IEF450I WTR     .S7F     .         ABEND S2F3       TIME=03.41.01
 IEF421I INIT=WTR.S7F (2) NO RESTART
 IEF450I WTRZ    .WTRZ    .         ABEND S2F3       TIME=03.41.01
 IEF421I INIT=WTRZ.WTRZ (2) NO RESTART
 IEF450I INIT    .P1      .         ABEND S2F3       TIME=03.41.01
 IEF421I INIT=INIT.P1 (2) NO RESTART
 IEF450I INIT    .P2      .         ABEND S2F3       TIME=03.41.01
 IEF421I INIT=INIT.P2 (2) NO RESTART

                                                      MORE...   VMHPO
```

Figure 7. OS/VS1 Release 7 BPE Release 3 starting under VM/HPO.

OS/VS2 SVS

Much like OS/VS1, a stopgap port of OS/360 MVT to the System/370 was released in 1972. Working in the same way as VS1 (where the system ran within a 16 MB virtual memory address space to allow for paging), OS/VS2 Release 1 was a major development milestone in what would later become MVS. Release 1 was often called SVS (Single Virtual Storage), as that accurately reflected the memory-management model of the OS.

SVS, being an enhancement of MVT, changed a few core scheduler functions to properly take advantage of the new hardware. One such example was the switchover from using less-accurate timing mechanisms on the 360 to using the 370's interval timer. Likewise, the way that the OS loads SVC (SuperVisor Call, a system call if you will) routines is quite different. On MVT, the nucleus would load all of its SVC routines from SYS1.SVCLIB into various transient area memory blocks. SVS, instead, would load these from SYS1.LPALIB into the Pagable Link Pack Area (PLPA) during a cold IPL (that is, starting the system with the CLPA option) -- this was then pagable, unlike on MVT where paging was not an option.

Thanks to a lawsuit with Applied Data Research, SVS (and VS1, for that fact) did not include many applications that OS/360 once came with; this included compilers, assemblers, a sort/merge program, and some timesharing features. This caused a price headache for some users, who were forced to pay more for what was once free software.

SVS was intended to be a bit more reliable than MVT, and a new feature that was added to aid this was the Authorized Program Facility; certain programs could use SVCs that may cause system damage if and only if they were link-edited with a certain option and were placed into the system-wide link-list (think of this like the system-wide PATH on a UNIX system). SVS also permitted users to append to the PLPA on subsequent warm IPLs (that is, without the CLPA start option); the Modified LPA would follow the Fixed LPA specified from the SVCLIB dataset. As primitive as this sounds now, this was seen as an advanced feature then!

SVS was also a uniprocessor-only system; customers with a multiprocessor 370 configuration would have to wait for MVS. Likewise, it introduced no new batch spooler; it retained MVT's primitive built-in spooler, but most customers opted to just use HASP or ASP instead *(side note: there is an image of OS/360 MVT 21.8 retrofitted with SVS HASP that you can download and run on Hercules)*.

SVS did include TSO, and it did include both TCAM and VTAM to drive terminals. For customers that had 2250 vector-graphics terminals, they could no longer be used with TSO as the access method library once provided by OS/360 to drive them was no longer present. As a matter of fact, SVS didn't even include VSAM at first; VSAM had not yet proven itself as a major storage access method at the time of SVS's release, but it certainly did by the time of MVS's release (for the confused, VSAM was used as the underlying storage engine that many mainframe database applications use, even to this day; it is a record-indexed or key-indexed file seeking/storage/retrieval method).

OS/VS2 MVS

Note: also called "MVS/370"

In 1974, after SVS had ran its course two years prior, OS/VS2 was primed for another upgrade. This came in the form of OS/VS2 Release 2, famously called Multiple Virtual Storage (or MVS). Unlike SVS, MVS actually provided multiple virtual address spaces to programs; wherein each program would only be able to access its own private address space (in the same way that, for example, processes found themselves on a virtual memory UNIX system). This provided a massive boom to system reliability, also aided by the natural inclusion of all of the big new features added in SVS. Since MVS provided uniform address spaces within each job address space, it was now possible to implement commonplace virtual memory features like shared libraries, shared communications segments, and memory-mapped files (though MVS famously did not provide this).

MVS also provided symmetric multiprocessing support for the first time on an OS/VS system; SVS was famously lacking in this regard. Alas, ASP users could still use their loosely-coupled multiprocessing configurations; they could also choose to upgrade to the new JES3 -- HASP users would find themselves installing JES2 (as both programs had been renamed when MVS released, and IBM started to properly include them in the base OS). Though the second-to-last release of MVT (21.8F) came in 1974 too, MVS was rapidly adopted.

MVS/SE: IBM, seeing that they could charge extra for extensions to base OS/VS2 MVS, released MVS/System Extensions in 1978. This was mainly a performance upgrade to existing OS/VS sites, as some repetitive OS kernel function segments were relocated into the machine's microcode *(side note: I do not know how this was done).* TSO logical-swapping was introduced too, wherein a user could remain in virtual memory but be swapped out to the DASD paging file; this definitely caused the user's online sequence to take a performance hit, but high-memory-usage conditions could be alleviated thanks to this. There was another release of MVS/SE, Release 2, that further microcodified the OS.

MVS/SP 1.1-1.3

The logical upgrade to MVS/SE was MVS/System Product, which bundled many useful products. These included:

- ACF/VTAM (SNA networking)
- DFP (Data Facility Product)
- TSO Command Package
- TSO/E (TSO Extensions, replacing the TSO Command Package)
- JES2 or JES3 Version 1

Other optional products like ISPF and SDSF were frequently bundled, and this saw hefty adoption. At this point, the version number of MVS was **3.8**, which would be reflected in all subsequent versions of MVS. During IPL, the system would display the SP release level:

```
IEA101A SPECIFY SYSTEM PARAMETERS FOR RELEASE 03.8 .VS2,VER=SP1.3.3 JBB1329
```

Figure 8. MVS/SP 1.3.4 IPL top message

```
 ----------------------  ISPF/PDF PRIMARY OPTION MENU  -----------------------
 OPTION  ===>
                                                            USERID   - WEC
    0  ISPF PARMS  - Specify terminal and user parameters    TIME     - 14:50
    1  BROWSE      - Display source data or output listings  TERMINAL - 3278
    2  EDIT        - Create or change source data            PF KEYS  - 12
    3  UTILITIES   - Perform utility functions
    4  FOREGROUND  - Invoke language processors in foreground
    5  BATCH       - Submit job for language processing
    6  COMMAND     - Enter TSO command or CLIST
    7  DIALOG TEST - Perform dialog testing
    8  LM UTILITIES- Perform library administrator utility functions
    9  IBM PRODUCTS- Additional IBM program development products
    C  CHANGES     - Display summary of changes for this release
    T  TUTORIAL    - Display information about ISPF/PDF
    X  EXIT        - Terminate ISPF using log and list defaults

 Enter END command to terminate ISPF.
```

Figure 9. ISPF/PDF user interface on MVS/SP 1.3.4

One of the biggest "upgrades" bought with MVS/SP Version 1 was the addition of support for the aforementioned Dual Address Space feature of the 303x System/370 CPUs. Newer versions of MVS/SP gained support for the Extended Real Addressing (the 26-bit real address scheme) introduced on the 3033 and 3081; information processing workloads were starting to get memory-intensive enough by the late 1970s that having more than 16 MB of real storage began to look more and more attractive.

ISPF and SDSF: As MVS/SP was gaining more and more traction, IBM produced a rather useful user interface for MVS. As TSO was certainly obtuse to use, many sought some kind of simplified interface. IBM launched this in the form of the Structured Programming Facility (i.e. SPF) in 1978. This program used the 3270 terminals to provide a proper dialog-driven interface, editor, and file manager for TSO. This would later evolve into the Interactive System Productivity Facility (ISPF), and it was even ported to VM and VSE!

A similar program was SDSF, the Spool Display and Search Facility, launched in the mid-1980s. This was a replacement to the cumbersome TSO way of retrieving spooled batch job output; many sites had both ISPF and SDSF installed. Both programs still exist on z/OS as staples of the environment.

```
V1R2----------------------  SDSF PRIMARY OPTION MENU  --------------------------
COMMAND INPUT ===>                                            SCROLL ===> PAGE

 Type an option or command and press Enter.

    LOG       - Display the system log
    DA        - Display active users of the system
    I         - Display jobs in the JES2 input queue
    O         - Display jobs in the JES2 output queue
    H         - Display jobs in the JES2 held output queue
    ST        - Display status of jobs in the JES2 queues
    PR        - Display JES2 printers on this system
    INIT      - Display JES2 initiators on this system

    TUTOR     - Short course on SDSF (ISPF only)
    END       - Exit SDSF

    Use Help key for more information.

    5665-488 (C) COPYRIGHT IBM CORP. 1981, 1989. ALL RIGHTS RESERVED
```

Figure 10. SDSF V1R2 on MVS/SP 1.3.4

MVS 3.8J

In 1981, the final base (i.e. free) release of MVS would release before all development effort went into successor systems: MVS 3.8J. This version is most famous to the mainframe hobbyist community that emerged in the early 2000s, as it was easily attained from a set of tapes found at Princeton University. The mainframe hobbyist community has supported MVS 3.8J with updates over the years, making it into quite the entertaining system for a mainframe newcomer.

MVS/XA

When the System/370 XA (and later ESA/370) machines had hit the mainstream, it was time for MVS to be alleviated of its 64 MB real storage limit. At the time MVS/XA was being released, application data needs had grown past 64 MB (the 64 MB limit being the 26-bit address limit of MVS/SP). MVS Version 2 (called MVS/XA) was released in 1983, famously developed under a version of VM/CMS that did hefty hardware simulation before the hardware it was destined to run on (much in the same way System/370 software was developed).

XA versions of program products quickly followed, including CICS/VS, compilers, IMS/VS, etc -- the new MVS/XA machines could now sustain many more logged-on users with improved swapping. ISPF/PDF had now become nearly universal, and TSO became (relatively) easy to use thanks to it. MVS/XA found itself ran for about a decade, and the 2 GB real storage limit was decided to be plenty for any computer in the 80s. 24-bit programs could still be ran (as clever S/360 programmers realized that the upper 8 bits of a 24-bit address could be used for other purposes), and conversion of programs to the new 31-bit addressing scheme would be done with the help of an updated XA-capable assembler (so-called Assembler H). Some 370/XA machines available then found themselves bearing more than 2 processors, and MVS/XA could properly utilize these. Over the course of its life, MVS/XA collected many important changes to the MVS line, which would gradually drag it towards its next major release -- MVS/ESA.

```
  IEA097I I/O CONFIGURATION 00 SELECTED
  IEA091I NUCLEUS 1 SELECTED
| IEA101A SPECIFY SYSTEM PARAMETERS FOR RELEASE 03.8 .VS2,VER=SP2.2.3
|  JBB2223

| IEA347A SPECIFY MASTER CATALOG PARAMETER

  IEA940I THE FOLLOWING PAGE DATA SETS ARE IN USE:
   PLPA .......- PAGE.VX23CAT.PLPA
   COMMON .....- PAGE.VX23CAT.COMMON
   LOCAL ......- PAGE.VX23CAT.LOCAL1
  IEA301I IEFUJV NOT FOUND IN SYS1.LINKLIB
  IEA301I IEFUTL NOT FOUND IN SYS1.LINKLIB
  IEA829I SVC 66 FOR BTAM NOT USABLE, MODULE IGC0006F NOT FOUND IN LPA .
  IEA826I IEASVC00: SVC244: TYPE 4 ROUTINE IGC0024D NOT FOUND.
  IEA191I CONSOLE 010 DEFINED AS MASTER CONSOLE.
  IEA958I EXCP APPENDAGE NAME TABLE NOT BUILT
  IEF338I DEFAULT EDT ID VALUE 00 USED.
  IEA598I TIME ZONE = W.05.00.00


  #
```

Figure 11. Console of an IPLing MVS/XA 2.2.3 system.

In addition to the changes listed, I/O device channels could now be dynamically selected and reconnected should hardware failures occur; the 370/XA architecture featured a totally different I/O channel control model (using subchannels in lieu of plain channels found on normal S/370 systems), and this design allowed this facility to develop. Cached DASD controllers were starting to develop in the mid-80s, and MVS/XA natively supported these (some models of the 3880 had cache, and the 3990 had cache natively). To expand the performance of systems armed with slower but still relatively quick auxiliary storage, the Auxiliary Storage Manager was overhauled to allow paging of jobs more quickly (to get them out of main storage if a constraint is hit). Expanding VSAM disk storage, the new Data-In-Virtual feature provided a UNIX-style memory-mapped-file access method (this would be expanded in MVS/ESA with a feature called data spaces and later hiperspaces). The new 370/XA Vector Facility

was also supported -- by this time, mainframes were starting to run a lot of CAD workloads thanks to advanced vector graphics terminals (like te 5080) gaining popularity. A vector facility (somewhat akin in function to a modern GPU 8minus the video output) provided good acceleration for such applications, and it was trivial to use from FORTRAN programs. The last release of MVS/XA came in 1989, with the above MVS/XA 2.2.3.

MVS/ESA

Version 3: MVS/ESA was a relatively quick product, development-wise. Building on MVS/XA Release 2 (which itself was MVS/SP Version 2, so MVS/XA R2 = 2.2.x), MVS/ESA SP Version 3 (so, MVS/SP 3.x.x) released in February 1988. Running on the new ESA/370 machines, MVS/ESA overhauled a few key components of MVS. Rather than allowing a process to only have a single address space, programs could now use so-called data spaces (thanks in no small part to having a set of new access registers on the ESA/370 CPUs). By the time MVS/ESA 3.1.3 was commonplace, Hiperspaces were also present -- these were a bit different than data spaces, accessed using two macros and functioning somewhat like DMA from the view of the program. A new facility called data lookaside (DLF) provided a shared hiperspace facility that multiple jobs could use called hiperbatch; DFSMS made storage management much easier by providing an IBM-native tape, DASD, and backup management product that came with the OS (yet another third-party program you didn't have to buy or write). Communications were becoming more and more advanced, too: in 1988, IBM released a TCP/IP stack for MVS/SP, but its performance was limited by a number of factors. On MVS/ESA, the TCP/IP stack code saw some substantial performance improvements and feature additions.

Version 4: *MVS/SP V4* released in 1990, adding support for the ESA/390 hardware that was then relatively new. A very major feature of MVS (at least on newer MVS systems) was the parallel sysplex -- specialized coupling hardware that provided reasonably tight-coupling (well, certainly moreso than a JES3 cluster) of multiple mainframes together; to facilitate this, an XCF (cross-coupling facility) box had to be used. Rather than using assembler macros to add and remove I/O device definitions (and then having to do a nucleus generation), once could now use Hardware Configuration Definition dialog system -- this was a menu-driven application that ran from ISPF, and thanks to Dynamic Reconfiguration Management, I/O devices could be hot-added and hot-removed without IPLing. Systems that possessed PR/SM (so, System/390s) would work with DRM and HCD to do this hardware redefinition on-the-fly. The MVS console interface was upgraded too, with much more functionality to aid multiple-console'd mainframes that were starting to be seen. As networking continued to evolve and IBM pushed APPC (over SNA), MVS gained native support for APPC applications through APPC/MVS (meaning that APPC was no longer the realm of VM/CMS; DOS, OS/2, OS/400, AIX, and any third-party platform had yet to gain APPC support and would not do so until about 1993 or so). MVS JCL was also expanded -- you could now use various statements like IF (already seen in VSE JCL) and INCLUDE to trigger the manual inclusion of a JCL procedure. Version 4.1.0 and 4.2.0 was said to be rather unstable and crashy, but 4.3.0 (the last release in the MVS V4 line) was very stable.

OpenEdition: Announced in February 1993 as an expansion to existing MVS/ESA 4.3.0 systems, OE provided full UNIX emulation and POSIX compliance to MVS. To facilitate this, a UNIX-style hierarchical file system (which itself was provided by DFSMS) was integrated into the system, and the OpenEdition (OMVS) kernel would mount various HFS datasets to various paths within the UNIX-style VFS (virtual filesystem for those not familiar with UNIX terminology). Most of the actual command line tooling was not from IBM's own UNIX, AIX, but instead from Mortice Kern Systems's InterOpen package (which was famously seen in other UNIX emulator packages for other OSes, including the famous POSIX subsystem on Windows NT). Strangely enough, even with OMVS being an emulation of a UNIX environment, it was actually more POSIX-compliant than most UNIX systems on the market then (with some UNIX specialists in that era claiming that it was more than 80% compliant with the POSIX, X/Open, Single UNIX Specification, and UNIX Certification requirements -- moreso than its native UNIX competition)! OMVS gained the valuable FIPS 151 certification for MVS, which was necessary if the platform was to remain a purchasable interest of the US federal government.

Version 5: *MVS/SP V5* was announced in April 1994, and bought big expansions to the sysplex and Coupling Facility features: the new Parallel Sysplex feature was an even-further-evolved incarnation of the coupling scheme unveiled in MVS/ESA Version 4. The System Resource Manager gained a new Workload Manager expansion, called Goal Mode, wherein the WLM will work with things like CICS to try to pre-allocate resources (rather than lazily allocate resources when the CICS transaction asked for them). Version 5 represented the peak of the MVS/ESA craft before it was ultimately replaced, and major work was done on this period in streamlining the OS. A slow P/390 could run it reasonably well, even! Alas, the many complex licensing configurations and installation methods left IBM yearning for a grand unification, and that would come with OS/390.

OS/390

Version 1: In late 1995, IBM began to wonder if it was a good idea to continue to offer the separate packaging/licensing scheme for MVS/ESA. While that OS version had definitely gained quite a bit of new features over its life, this left IBM and customers with a massive number of product configurations (and ordering MVS/ESA was apparently becoming more and more difficult). IBM's answer in late 1995 was to attempt to unify the packaging and distribution for MVS/ESA, and this resulted in OS/390. While OS/390 Version 1 and Version 2 (the latest release of which is *OS/390 V2R10*) ran up until the year 2000, OS/390 itself did not really introduce many new products.

The compilers for MVS/ESA were renamed to things like "XYZ for OS/390 and VM" (such as "COBOL for OS/390 and VM"), and OpenEdition was integrated into the base. VTAM and TCP/IP were combined together under the Communications Server for OS/390 name, and CICS/ESA became CICS Transaction Server. Db2 started to become used in large distributed systems, and System/390s seemed to be adopted for big database server loads. By then, it was the year 2000, and MVS was looking rather old with its 2 GB addressing limit.

Version 2: Alas, OS/390 Version 2 (bearing an MVS/SP number of 6.x.x, though it is no longer displayed on the IPL screen) included a virtual tape facility through DFSMSrmm and a parallel batch queuing system called Smart Batch. The last release, V2R10, was released on September 23, 2000; this version was ran on many systems that were licensed for it but not newer MVS releases (like z/OS) for about a decade. V2R10 was the direct precursor to the successor to OS/390 -- systems that released after this version found themselves even often bearing a DASD with a VOLSER of "OS39M1" -- this is the case for the P/390 development pre-packaged ADCD systems.

z/OS

The road to z/OS was certainly rather interesting. While the 64-bit System Z hardware was essentially an expanded System/390 CMOS 9672 machine (see the earlier information on the System/390 hardware), MVS would need some overhauls to become 64-bit. IBM used (as usual) an enhanced version of VM/CMS to develop the "64-bit OS/390" -- when it was done, it was decided to rename the line to the System Z (and OS/390 to z/OS).

The new 64-bit z/OS had some rather strange limitations. For instance, it was not possible to place executable code above the 2 GB bar (note: THE LINE is 16 MB, THE BAR is 2 GB); it had to remain within the first 2 GB of storage. However, data could be placed above the bar; it is not uncommon to see programs like CICS and Db2 eat up storage past the bar for its own data. z/OS V2R3 alleviated this, but hardly any supervisor calls work when the code for the running program resides above the bar. The z/OS compilers could produce 64-bit-laden operations (as in, the programs used 64-bit registers), but they themselves did not ever generate code that resided above the bar.

z/OS V1R5 was the last version that could IPL in 31-bit mode -- future versions of the OS required a 64-bit system as many core system modules only shipped in AMODE 64 varieties.

As z/OS proceeded through its life, it gained a variety of third-party software. IBM produced ports of Node.JS, Golang, and other such programs; other ISVs (independent software vendors) provided other similar ports of languages like Python (though Rocket Software's offerings were notably lambasted by their users).

```
  Menu  Utilities  Compilers  Options  Status  Help
 ------------------------------------------------------------------------------
                           ISPF Primary Option Menu

 0  Settings      Terminal and user parameters           User ID . : WEC
 1  View          Display source data or listings        Time. . . : 14:59
 2  Edit          Create or change source data           Terminal. : 3278
 3  Utilities     Perform utility functions              Screen. . : 1
 4  Foreground    Interactive language processing        Language. : ENGLISH
 5  Batch         Submit job for language processing     Appl ID . : ISR
 6  Command       Enter TSO or Workstation commands      TSO logon : ISPFPROC
 7  Dialog Test   Perform dialog testing                 TSO prefix: WEC
 9  IBM Products  IBM program development products       System ID : EVIEMVS
 10 SCLM          SW Configuration Library Manager       MVS acct. : ACCT#
 11 Workplace     ISPF Object/Action Workplace           Release . : ISPF 5.5
 M  More          Additional IBM Products


      Enter X to Terminate using log/list defaults


 Option ===>
  F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F10=Actions  F12=Cancel
```

Figure 12. ISPF on z/OS V1R5

```
  Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY WEC      TSU00050  DSID     2 LINE 0       COLUMNS 02- 81
 COMMAND INPUT ===>                                          SCROLL ===> PAGE
******************************** TOP OF DATA ********************************
                   J E S 2  J O B  L O G  --  S Y S T E M  S Y S 1  --  N O D E

18.32.43 TSU00050 ---- MONDAY,    06 OCT 2025 ----
18.32.43 TSU00050  $HASP373 WEC      STARTED
18.32.43 TSU00050  IEF125I WEC - LOGGED ON - TIME=18.32.43
19.26.48 TSU00050  IEF450I WEC ISPFPROC DBSPROC - ABEND=S622 U0000 REASON=000000
   507                     TIME=19.26.48
19.26.48 TSU00050  $HASP395 WEC      ENDED
------ JES2 JOB STATISTICS ------
  06 OCT 2025 JOB EXECUTION DATE
            2 CARDS READ
          699 SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
           39 SYSOUT SPOOL KBYTES
        54.09 MINUTES EXECUTION TIME
        1 //WEC     JOB 'ACCT#',REGION=8192K
        2 //DBSPROC  EXEC DBSPROC
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 13. SDSF displaying job output on z/OS V1R5

```
                    RMF V1R5   Processor Delays                Line 1 of 13
Command ===>                                                 Scroll ===> CSR

Samples: 100     System: SYS1  Date: 10/07/25  Time: 10.03.20  Range: 100   Sec

            Service  DLY USG  Appl EAppl ----------- Holding Job(s) -----------
Jobname  CX Class     %   %      %     %  %  Name      %  Name      %  Name

TCPIP    SO SYSSTC     1   2   0.2   0.2   1 JES2MON
RMF      S  SYSSTC     1   0   0.1   0.1   1 JES2MON
HTTPD1   SO STCLOM     1   0   0.1   0.1   1 JES2MON
VMCF     S  SYSSTC     1   0   0.0   0.0   1 JES2MON
VTAM     S  SYSSTC     1   0   0.0   0.0   1 JES2MON
IBMUSER  T  TSO01      0   2   0.6   0.6
JES2MON  S  SYSTEM     0   1   0.1   0.1
JES2     S  SYSSTC     0   1   0.1   0.1
WEC      TO TSO01      0   1   0.9   0.9
*MASTER* S  SYSTEM     0   0   1.0   1.0
WLM      S  SYSTEM     0   0   0.2   0.2
CATALOG  S  SYSTEM     0   0   0.1   0.1
RMFGAT   S  SYSSTC     0   0   0.2   0.2
```

Figure 14. z/OS V1R5 Resource Management Facility performance reporting

TSS/360 and TSS/370

Origins

In the mid-1960s, there was a huge emphasis on a new computing technology called "timesharing" -- that is, "have a share of time" on the computer (we would nowadays call this a "multiuser computer"). Timesharing systems were also multitasking and interactive systems, and would frequently be driven via teleprocessing (that is, terminals located physically far from the computer). Granted, OS/360 had TSO, but it was a far cry from a true timesharing system (TSO was commonly viewed as nothing more than a hand-keyed batch job at a terminal, with TSO commands that were functionally equivalent to JCL batch job commands); as such, IBM sought a true timesharing system for the S/360 to try to get a leg up on the competition.

TSS/360

In 1967, IBM made TSS/360 available (it was not an officially supported product, and never would be; read on to see why this was the case) for customers that had the System/360 Model 67 (the 67 had the "DAT box", which was the 360's memory management unit that would later find its way into every System/370 on the S/370 AF line). Despite having a very rough launch, TSS pioneered implementations of a lot of how critical concepts in computing:

- Very early implementation of symmetric multiprocessing, using a single "resident supervisor" (kernel) that communicated with other CPUs in the system through interrupts)
- Position-independent code, something that alleviated the link-editing issues of the earlier DOS/360's memory management scheme
- Combined virtual-memory/virtual-machine architecture for every running user (this would later influence CP-67, see the section on VM for more information on this)
- Dynamic linker (very new for 1967, TSS is perhaps the first real implementation of it)
- Table-weighted thread priority scheduler

In that list, there is an entry for a dynamic linker; for those that are unaware, dynamic linking allows a program to load and get linked with the language runtime when it loads (as opposed to doing all of that ahead of time by the linker, which means every routine in the language runtime library that is used in the program must be linked into the program itself, increasing program size drastically). To save memory/storage on the memory-constrained 360 systems, TSS would employ a dynamic linker that would share a language's runtime library between several programs running it at once (or keep it in memory for another program to use later).

TSS's Dynamic Linker: The TSS dynamic linker is relatively similar to how early UNIX systems implemented dynamic linking (like UNIX System V release 4.1). For those that are familiar with writing 360 assembler programs, you will be familiar with the concept of the CSECT. Every program will have a CSECT (control section), some DSECTs (dummy sections, somewhat like structs in C), and a TSS-specific area called a PSECT (a prototype section). There is also a program-writable register save area (remember, the 360 didn't have a stack). When a program is linked dynamically, the TSS loader will load the shared library (or, well, map it) into the program's address space (read-only, of course). In order to access the functions, the program will load the address of the functions in the runtime library from that PSECT. When a dynamically-linked routine is called, the program will save its registers into the save area, load the address of the symbol from the PSECT table, and jump to the routine. As such, every program will have a unique PSECT table (as the functions will be placed into the virtual memory address space of the program at any location, so long as it does not overlap something that's stored with the program).

Commands: Being a time-sharing system, TSS had *commands!* These consisted of 7 different categories (as seen in the manual):

- "Task Management" (LOGON, LOGOFF, etc)
- "Data Management" (DELETE, CATALOG, DDEF, etc)
- "Program Management" (LOAD, DISPLAY, DUMP, etc)
- "Command Creation" (BUILTIN, PROCDEF, etc)
- "Message Handling" (LIMEN, BREVITY, etc)
- "User Profile" (PROFILE, DEFAULT, SYNONYM, etc)
- "Program Product Language Interface"
  - ASM (Assembler F)
  - HASM (Assembler H)
  - COBOL (COBOL/360)
  - PLI (PL/I 360)
  - PLIOPT (PL/I Optimizer)
  - FTNH (FORTRAN H)

TSS/370

To little fanfare, IBM did provide a 370-native version of TSS as a PRPQ (a "Program Request Price Quotation") intended only as a migration path (hopefully off TSS and onto MVS/TSO) for 370 customers. As mentioned in the section on the 370's initial launch, the original 370s did not have a DAT box; when TSS/370 launched in 1971, users would have to adapt their earlier 360/67 DAT box; because TSS/370 relied on intrinsics only found on the DAT box, it is rather difficult to run TSS/370 under VM.

A fun fact that seems to be poorly recorded: the announcement for TSS/370 occurred at the same time as the official decommital (and therefore soft-cancellation) of TSS/360; this was done under heavy pressure of various execs that, though they tried to kill the CP/CMS project, realized that it was a far superior timesharing system to TSS. An interesting note about TSS is that it actually *uses 31-bit addressing*, and the sneak-peak teaser for the 370/XA machines (wherein all 370 machines would gain a 31-bit address space) was released at the same time as this TSS debacle. For the curious, this May 1971 meeting was held at the Westchester Country Club.

```
?
ls
ENTER PHY ADDRESS OF CARD RDR OR DEFAULT = EOB.

ENTER ADDR OF PAGING DISK
251
PRINT MAPS? 0 1 2 3...NONE IVM RESSUP BOTH
3
ENTER CODE FOR FUNCTIONS NOT TO BE LOADED.ALL FUNCTIONS WANTED = EOB
99
QUICK START REQUESTED?  IF Y ENTER ADDR  OF PACK FOR QUICK START DATA SET--N = EOB

DELTA DATA SETS?  Y OR N
n
CURRENT CLOCK VALUE IS 12/05/25 10.27.59 IS CLOCK CORRECT?
no
ENTER DATE AND TIME AS
MM/DD/YY HH.MM.SS
10/07/85 10.28.30
DEPRESS TOD SWITCH

CURRENT CLOCK VALUE IS 10/07/85 10.28.31 IS CLOCK CORRECT?
yes
$ run
 BULKIO REQUIRED?
y
   Y ACCEPTED
10:28:45        SYSOPER0 NO DRUM PATH AVAILABLE
10:28:45        TSS00001 BATCH MONITOR HAS BEEN INITIALIZED
10:28:45 R=0001 SYSOPER0 VALIDATE THE SYSTEM HARDWARE CONFIGURATION. REPLY OK WHEN DONE.
reply 1,ok
TSS370
 STARTUP COMPLETE,USERS MAY LOGON
10:28:56        SYSOPER0 *CZAWS*   ASNBD'S DONE: A20E-01.
10:28:56        SYSOPER0 *CZAWU*   BSN=0316 STARTED  ON 020E, SYSOPER0.SYSLOG.G0030V00
10:28:56 ->->-> SYSOPER0 CZCMD   004I     ON PTR 020E MOUNT FORM    PAPER
MOUNT CARRIAGE TAPE SAME
MOUNT CHAIN/TRAIN 'PN'
SET DENSITY 6
THEN NR/R
10:29:15 ->->-> SYSOPER0 CZCMD   002I
VERIFY THAT THE FOLLOWING 20-CHARACTER MESSAGE APPEARS ALSO ON PTR 020E,
- ***   BUFFER LOADED ABOVE.    *
sard
                  SYSTEM ACTIVITY AND RESOURCES AT  10/07/85  10:29:37
                                                      REPLYS:     0
USERS - CONV:    1  BACK:    1  REM :    0  RJEA:    2
QUEUES- EXEC:    0  PRNT:    1  PNCH:    0  TAPE:    0  RJE :    0
AVAIL - PTRS:    1  RDRS:    2  PUNS:    2  TAPS:   48  DISK:  150
PAGES - PUB :    18108 TEM :        0 AUXDR :       0   AUXDK : 22499
TSS370
```

Figure 15. TSS/370 Release 3 starting up

```
?
logon tss
 TSS/370 RELEASE 3.01
 ENTER PASSWORD
XXXXXXXXEEEEEEEEMMMMMMMMibm
  TASKID=0012 LOGON AT 10:35 ON 10/07/85
_logoff
 LOGOFF AT 10:59 ON 10/07/85

TASK ABENDING. ANY BUFFERED OUTPUT CONTINUES.
CZAFJ - NON-ZERO RC FROM RELEAS (CZCAD1)
CZAFJ - NON-ZERO RC FROM RELEAS (CZCAD1)
 TASK DELETED

?
```

Figure 16. TSS/370 Release 3 user logon and logoff

RAX and MUSIC

In the void of TSS, several systems emerged. While VM (to be discussed later) would become the dominant timesharing system on the 370 line, several one-off systems were written. One of those was called RAX, and it was introduced in 1966.

RAX

IBM worked with Lockheed to produce *Remote Access Computing*, a system that evolved out of RACS (an earlier system, seemingly rather poorly documented). The original target of RAX (which had the program number 360A-CX-17X) was a System/360 Model 30 or better, bearing at least 64 K of storage. There were two choices for terminals: the 1050 remote typewriter terminal, or the 2260 display terminals (of interest as they were the precursor to the legendary 3270 series, and famously used acoustic delay lines to store the screen image).

RAX had online compilers for BASIC (which was derived from CALL/360 BASIC), COBOL, FORTRAN IV, and Assembler F. Several sites adopted this because of the online nature of the system:

- US Department of Agriculture
- McGill University
- Boston University
- University of St. Andrews
- University of Rhode Island
- Lockheed
- Bell Aircraft Corp

RAX, being relatively commonplace in an academic environment, led to the development of further systems, including...

MUSIC

When McGill installed RAX, they found it rather impressive and began to modify it. In 1972, McGill's stack of RAX modifications was added to IBM's catalog of user-written programs as the "McGill University System for Interactive Computing" (i.e. MUSIC). It provided a primitive interactive timesharing environment that rapidly gained more and more features; when version 4.0 came out (in 1978), true filesystem permissions and other such features were introduced. When the IBM 4300 series System/370 machines became available, MUSIC 5.0 gained support for those but also the "new" FBA DASDs.

In 1985, IBM saw MUSIC and chose to adopt it as a System Product, and so MUSIC/SP was born. Version 1.0 added real virtual storage support (which was surprisingly lacking that whole time); version 2.2 in 1990 streamlined performance; version 2.3 introduced the current MUSIC/SP filesystem and the TCP/IP support the system became so well-known for all in 1991. Rather than having a "real" TCP/IP stack, MUSIC/SP (when ran under VM) would be able to talk to the VM/CMS TCP/IP stack via IUCV and use that for networking services (this will be described in more detail later).

The Filesystem: Gaining ideas from other systems like TOPS, the MUSIC/SP filesystem is assembled as a single file index that can span multiple DASD volumes. Files could be located in the file index in a single operation by performing a hash function operation on the username. The filesystem was modelled after that found on MS-DOS, with directories and flat files throughout. The DOS "drive letter" was the user ID, and directories func-

tioned and were used just like on DOS. There were two kinds of files: public and private. To save on disk space in an era where disk space was a rare and expensive thing, all files were compressible.

The Emulator In an era of mainframe OS compatibility, MUSIC joined the fray with its own MVS emulator. Many of the compilers and language interpreters were ran on the MVS emulator, to great success! The emulator implemented a small subset of MVS supervisor calls (i.e. system services), and it was remarked that MVS programs could run on MUSIC faster than they could run on real MVS (though I have yet to see a citation for this). Full-screen TSO services were also emulated, allowing for program packages like GDDM and DisplayWrite/370 to work flawlessly on MUSIC. Since MVS DASD storage is accessed through dataset mechanisms and not (generally) a DOS-style filesystem, the MVS emulator contains a facility to map a MUSIC user file to a MVS dataset name.

The Networking: Even though it showed up a few years late to the TCP/IP stack race, MUSIC/SP's TCP/IP facilities (by way of the VM TCP/IP stack) were rather famous. Generally, sites would run two copies of the VM TCP/IP stack, with one being dedicated to MUSIC and the other being present for CMS applications/users (the two would interlink using an IUCV socket; see the section on VM for more information on both this and the VM TCP/IP stack). MUSIC also had an HTTP server and browser, a Telnet client, and a visual FTP client to accompany the FTP server program.

The Emails MUSIC was somewhat known for its email handling system, and several mailers had been written over the years. The most commonly-used one was called RDMAILER, and it would work with both SMTP sites and RSCS/NJE (i.e. BITNET) sites through a modified RSCS running on the VM host (as nearly all MUSIC systems were ran under VM if they wanted networking). Pieces of WISCNET (itself a TCP/IP stack for VM, which would later go on to become the famous "FAL" TCP/IP stack IBM provides) were ported to MUSIC, and MUSIC functioned as a "client" to a WISCNET stack running elsewhere on VM. Though MUSIC never gained support for protocols like NFS or the Sun ONC/RPC system (even though MUSIC was often found in UNIX environments at universities), what was available was of notoriously good quality.

The Job Language: For the most part, MUSIC's JCL was intended to be a simplified imitation of the JCL found on MVS systems. Programs could be ran with ease, as essentially every key program would be found in the LPA (just like on MVS). For example, the C compiler could be ran like this:

```
/SYS REGION=16384
/FILE SYSLIN NAME(CTEST.TEXT) NEW(REPLACE)
/FILE SYSLIB PDS(*.H,$IBC:UPK.*.HDR)
/LOAD C370
/OPT MARGINS(1,80) NOSEQUENCE
/JOB NOGO
#include <stdio.h>

int main() {
 puts("MUSIC C compiler demo\n");
 return 0;
}
```

Within reason, MUSIC was seen as a rather "easy" system, but its demise in the early 2000s as it was starting to be seen as obsolete meant it was over. Also, note that PDS argument to /FILE -- this assembles a PDS based on those MUSIC disk files.

```
MM      MM  UU     UU   SSSSSSS   IIIIII   CCCCCCC        /  SSSSSSS   PPPPPPP
MMM    MMM  UU     UU  SSSSSSSSS  IIIIII  CCCCCCCCC      /  SSSSSSSSS  PPPPPPPP
MMMM  MMMM  UU     UU  SSS          II    CC            /   SSS        PP    PP
MM MMMM MM  UU     UU   SSSSSSS     II    CC           /     SSSSSSS   PPPPPPPP
MM  MM  MM  UU     UU        SSS    II    CC          /           SSS  PPPPPPP
MM      MM  UUUUUUUUU  SSSSSSSSS  IIIIII  CCCCCCCCC  /      SSSSSSSSS  PP
MM      MM   UUUUUUU    SSSSSSS   IIIIII   CCCCCCC  /        SSSSSSS   PP



         Multi-User System for Interactive Computing / System Product




Press the ENTER key to view next page when you see this message --->  More...
```

Figure 17. MUSIC/SP ESA 5.3 logon screen

```
 --------------------Full Screen Interface for MUSIC------------------ Page 1/1
 Command ===>
  * NO NEW MAIL *
  Place the cursor on an item and press ENTER or RETURN.

  MUSIC tools:
     Mail          Electronic mail facility
     Programming   Compilers, processors, tutorials, etc
     CI            Course Information
     Internet      Internet access, news reader, gopher, etc
     More          Other general MUSIC tools
  MUSIC files:
     FLIB *        Full Library Screen for current directory
     FLIB          Filespec=> WE00:*              < pattern
     FUTIL         Other file related utilities
  MUSIC environment:
     Help          General help and online documentation
     New Password  Change your password
     Defaults      FSI customization
     Profile       Profile utility and options
     Off           Terminate your session and disconnect from MUSIC
     Suggest       Make a suggestion or send a comment to support staff
 F1=Help      F2=Suggest    F3=End       F9=Find       F12=Retrieve
```

Figure 18. MUSIC's Full Screen Interface

MTS

The *Michigan Terminal System* was an interesting OS, born of the 1960s, for the System/360 Model 67. Remember earlier the discussion about the 370-67's DAT box -- this is what was necessary to make MTS work. As you can probably guess, MTS was a timesharing system, developed by and for the University of Michigan Computing Center.

MTS itself is comprised of several components, and it evolved to remain current with other System/360-family OSes over time. Parts include:

- The University of Michigan Multiprogramming Supervisor (***UMMPS***): the core of the OS; this provided a multitasking, multiprocessor, timesharing, virtual memory supervisor program that allowed for the execution of several simultaneous reentrant programs started by users
- Command Language Subsystems and Job Programs: these handled user interaction
- Various user-written applications and IBM-written applications, like compilers and runtime libraries

Now, let us examine the history of this fine system.

History

Before MTS, the University of Michigan had and IBM 7090 mainframe, and wished to run it for academic purposes. Their solution to this problem was to take the General Motors Executive System (itself hailing from the IBM 701) and modify it into the University of Michigan Executive system (called UMES). This was a batch system, and, by the time it was adopted, it was obvious that timesharing was the future. While UMES was certainly tailored to run reasonably well under a load of many small/short jobs that students and staff would submit, it was clear that this was a dying form of computing in the academic space. Remember, before the invention of virtual memory, computer execution capacities were primarily limited by the availability of real memory, and that exhausted quickly in that era too.

IBM was famously hesitant to support a timesharing workload on the System/360 -- this is why many often remark that the 360 itself was a *step back* in computing technology, as many other computers (like those mentioned in the introductory chapter of this book) already had virtual memory support (and timesharing was readily adopted on these). Several people who were at the University of Michigan published a rather influential paper on the topic of timesharing in January 1966 in the Journal of the ACM entitled *Program and Addressing Structure in a Time-Sharing Environment.*

While CTSS (devised at MIT) did not employ a virtual memory system in the modern sense we understand today, this paper described a timesharing system that worked with a modern virtual memory system (and described what would become the System/360-67's DAT box). IBM was, as discussed elsewhere in this book (particularly in the section on VM/CMS), hesitant to provide such a thing.

After hefty discussion with the University of Michigan folks that wrote the paper, IBM finally agreed to design a special variant of the System/360 Model 65 explicitly intended to support this timesharing system -- this was called the System/360 Model 65M (the M for Michigan). Since IBM initially did not provide a timesharing operating system with the System/360 announcement, this new development at the University of Michigan started to make headwinds around the computing industry rather quickly.

Several other System/360 customers caught wind of this: MIT's Lincoln Laboratory (the one that would later become the wellspring for CP/CMS), Princeton University (a very influential IBM shop), Carnegie Mellon University, and even General Motors (who always had a fascination with timesharing even in the early days of computing)! The System/360 was already a hit with these customers, but they saw it as a batch-oriented system -- when

they heard that someone was planning to do timesharing on one, there was a bit of a demand for these special System/360 Model 65M machines!

IBM would repackage and slightly modify the S/360-65M into the System/360 Model 67, and it became an officially-supported IBM product. As soon as that was made available, over 100 customers tried to acquire one; after the S/360's launch, IBM learned the error of their ways and relented; as a response, TSS/360 was born (and quickly died to its customer-written competitors, you can read more about that in the TSS chapter). This new OS was supposed to be delivered approximately at the same time as the S/360-67, but people got impatient.

IBM was somewhat slow to deliver that S/360-65M to the University. As a response, the University Computing Center people cleverly designed a simple timesharing system for a System/360 Model 50 that they had (this was funded by an ARPA program called CONCOMP, the Conversational Usage of Computers). The original codebase for this system consisted of two halves: a "half-page of code written out on a kitchen table" and a modified program from MIT called the Lincoln Laboratory Multiprogramming Supervisor -- if that name sounds familiar, that's because the UMMPS core of MTS is actually derived from MIT's work! The MTS job program would run under the control of LMMPS, and was quite interesting. Starting in 1965, the University was well on the way to an OS of their own.

The CONCOMP research project was intended to "anticipate the availability" of the S/360-67, and the machine was delivered on time with no delays. However, what the staff at the Computing Center had written was, really, a stopgap measure to "warm up" their minds to write a full timesharing system. They fully expected to discard this system as soon as TSS/360 became available, and, they did. The paper on CONCOMP notes that, while the engineers were happy that the S/360-67 arrived on time, they were quite unhappy that TSS/360 did not. Furthermore, TSS was plagued by performance and reliability issues -- something that came to define TSS as a massive failure in IBM's history. The paper also lambasted TSS for not even supporting all of the terminals that IBM had on the market at the time; note that IBM did have graphical terminals, and various soon-to-be-timesharing sites had these! To make matters worse, IBM did not sell a single terminal controller that could uniformily control all these varieties of terminals.

Fed up with TSS lacking support and quality for just about anything that the Computing Center required, the Computing Center staff agreed that the only way forward was to keep working on that stopgap OS they threw away a few months ago. Transitioning from the S/360-50 to the S/360-67 took a while, so the staff continued development on the S/360-50. The team had several goals they wished to achieve:

- Design a conversational system that could "support any type of effective man-made interaction"
- Develop a hardware attachment that could support a myriad of remote terminals
- Be able to support 50 to 60 interactive users with 4 to 6 batch jobs, concurrently
- Be able to support high-level language compilers, graphics applications, and more
- Participate on a network, those being the early ARPANET and MERIT network

TSS was doing so bad by then that IBM literally cancelled the project, but would later reinstate it; all the while, MTS continued to grow. LLMPS was morphed into UMMPS, and development took off. Being an academic system first and foremost, a consortium was founded to maintain and develop MTS. The members were:

- University of Michigan (1967-1999)
- University of British Columbia (1968-1998)
- University of Newcastle upon Tyne (1969-1992)
- University of Alberta (1971-1994)
- Wayne State University (1971-1998)
- Rensselaer Polytechnic Institute (1976-1992)

- Simon Fraser University (1977-1992)
- University of Durham (1982-1992)

The sites that did run MTS often ran multiple MTS systems, and for good reason: MTS systems could network together!

Computers Running MTS

The first machine that ran MTS was the single-processor S/360-67 that the Computing Center acquired in early 1967 (which bore the serial number 2, the first to leave IBM). This initial implementation did not yet support virtual memory, but it could sustain 5 online users and 1 background batch job. In November 1967, the engineers added virtual memory support; this upped the number of users to 50 and the number of batch jobs to 5. It should be noted that IBM actually sold a duplex (i.e. two processor) variant of the S/360-67; the Computing Center acquired one of these in August 1968, and it replaced the earlier simplex (i.e. one processor) system -- by then, the system could sustain 70 users and 8 batch jobs.

The University of Michigan continued to upgrade, but not always to IBM hardware! They received the second Amdahl 470V/6 ever made in 1975, and MTS worked great on it. Having great success with this IBM competitor, they continued purchasing their computers -- they received the first shipped Amdahl 5860 in 1982. They chose to return to IBM to receive the first factory-shipped 3090-400 in 1986, to great success.

MTS's open consortium nature meant that MTS sites often contributed new code back to the original project (much in the same way that the development of CP/CMS went). Rensselaer modified MTS to support 32-way multi-processor S/370-XA processors in 1984 (a big boom over the original 4 supported by 370-non-XA MTS), but the 6-processor system at RPI was likely the largest system MTS was ever ran on. In 1991, the Computing Center upgraded again to an IBM ES/9000-720, and could support over 600 users! It is believed that the last MTS site to exist was actually Rensselaer's MTS system, which was retired in June 1999.

Front-End Processors: One of the requirements associated with the CONCOMP project was terminal and networking support. Initially, the Computing Center used terminal controllers like the IBM 2703 or Memorex 1270 to support asynchronous serial lines (many of which were attached to modems), but this was a terrible solution. For one, the 2703 (unlike the later 3705) was a fixed-function and non-programmable system; feeling the limitations of this, the CC engineers came up with a clever solution: attach a PDP-8 (later a PDP-11) to a S/370 via a parallel channel, and use that to handle networking and terminal control!

The U of M was the first to attempt this (after all, they had the idea), and they called it the Data Concentrator. This was a modified PDP-8 born out of the requirements of the CONCOMP project, and was actually the first non-IBM device that ever attached to an IBM mainframe channel interface. This evolved into a two-tier system and was converted over to a PDP-11 architecture; the second tier was called the Remote Data Concentrator, and it networked with the Data Concentrator at the Computing Center over synchronous serial (as was common in the day).

The University of British Columbia produced an upgrade to this system; they developed two PDP-11 boxes: the Host Interface Machine (HIM) and Network Interface Machine (NIM) were true network controllers for MTS.

Networking and MERIT: Though the DC/RDC and HIM/HIM systems were great, they were initially not the best solution for networking: the University of Michigan and Wayne State University to design hardware to support the MERIT Network. This came in the form of a dedicated PDP-11 that they called a Communication Controller; these attached to a parallel channel on one end, and talked to remote CCs on the other end. These first permitted host-to-host remote logon connections, and later supported terminal-to-host connections (with the terminals being driven directly by the CC); since batch jobs were still frequently used, batch-to-host support was also added to facilitate remote job submission.

The original host-to-host system was explicitly developed to allow remote access to and from non-MTS computing systems. This first came in the form of the CDC SCOPE/HUSTLER system that Michigan State University had. Eventually, the MERIT people renamed the CC to be a Primary Communications Controller, and then developed a Secondary Communications Controller. The PCPs were the main network nodes, and could communicate with each other via either Ethernet or synchronous serial. SCPs were in turn connected to PCPs via synchronous serial links. The PCP and SCP would be able to attach to GTE Telenet, Tymnet, ADP Autonet, and more -- this was then updated to add support for TCP/IP connections (and the 370 channel devices emulated on the PDP-11 to do this are now emulated by the Hercules emulator).

The MERIT Network PCP/SCP project was huge: at the peak of MERIT's development, there were more than 300 PCPs and SCPs with a combined total of about 10,000 terminal ports.

Job Spooling: Though it was not originally written for MTS (and was instead written for OS/360), MTS included HASP (which would later become JES2 on the MVS line). There were other spoolers available, but nearly every site ran HASP; it was quite well-integrated with the system, and could handle both card-submitted batch jobs and jobs submitted by online timesharing users (as one would expect). Of course, spooled output devices (printers and punches) were under the control of HASP.

Versions

Over the years, the following versions of MTS existed:

D1.0

There are two extant versions of MTS 1.0, only one of which has been archived. The first was produced for the University of British Columbia's Computer Center in October 1968, and the second was produced for Newcastle University (i.e. NUMAC). The second one was produced by copying the tapes from the first version and appending any changes/upgrades onto the end of the tape volume. This version was intended to run only on a System/360 Model 67.

The base MTS system was not very feature-rich on this version, but the system did have some interesting components, including (but certainly not limited to):

- Lisp
- Assembler G
- GPSS-360
- PL/360
- FORTRAN G (FORTRAN II compiler)
- Support for the 2250 display terminal
- Support for the 2703 communications processor
- SNOBOL4
- WATFOR (Waterloo FORTRAN)
- Sort/Merge
- An accounting system

The NUMAC updates included a PDP-8 assembler, updates to all the compilers and assemblers, fixes for the 2780 line driver, and a PL/I compiler/library.

D1.0 Mod 1

This was a tape that had some revisions to the D1.0 distribution; these were intended to stack on top of NUMAC's version of D1.0. This version was believed to be released sometime in 1969, but there are no certain dates. Update included:

- WATFOR (Version 0 Level 7, August 6 1968)
- Updated device tables
- Updated system editor

D2.0-D2.3

This version was released in February 1970, and was a mostly incremental update from D1.0 Mod 1. The D2.0 tapes that do exist mainly contained documentation updates, but some interesting things were added:

- XCOM
- A new dump/restore program
- Revised RJE support
- PL (a dialect of PL/1 for teaching)
- GPSS/360
- WATFOR (same version as D1.0 Mod 1)
- Assembler simulator

D2.1 (from June 1970) was an incremental update that updated device tables; D2.2 (from 1971) included a format driver to automate system installation; D2.3 (1972) included a version of BASIC.

D3.0

Released in August 1973, this version included a variety of new tools. Being a new major release, this was distributed on quite a number of tapes (coming on 6 1600 BPI tapes, or 9 800 BPI tapes). The last tape in the tape set (whichever density the site asked for) is a self-restoring dump/restore tape that would install the MTS starter system. The D2.2 driver file system provided its driver file on the last tape. The format of that last tape is as follows:

- The TSS DASDI program, used to initialize a DASD for the system to be restored onto
- The TSS Dump/Restore program, used to read the starter system further down on the tape
- The TABLES deck generator program, which would be IPLed to produce a TABLES card deck used for installation later
- The OS-derived ISSDASDI program, used to format a DASD pack that would hold HASP spooling space
- Data for the TABLES deck generator program
- The dump/restore data
- The TABLES program's source
- An assembly listing of TABLES

By this release, HASP was fully configured to accept remote workstations (using RJE; see the sections earlier in this book on it, or flip to the index at the back to find it quickly). The version of HASP found on this MTS release explicilty supported the 360/20, other 360s, an 1130, an 1800, or a System/3 as an RJE workstation.

The following programs were included on this distribution:

- Assembler G
- PL/I
- PLC
- GASP
- BASIC
- UMLOAD
- APL
- PL/I
- Accounting system
- Billing system
- Virtual 67
- DITTO
- ASMTIDY
- TSS assembler
- Polygraphics
- SNOBOL4

D3.1-D3.2

Rather than a rather manual update method, these revisions (released in March 1974 and March 1975, respectively) were much easier to install (thanks to the power of the installation driver system). These versions contained differential updates to the various programs and compilers found on base D3.0, and carried the OS to its next major release.

D4.0

In August 1977, the next release hit. The tapes for this distribution came either in 1600 BPI or 6250 BPI density variants, and was split across at least 3 backup tapes with 1 dump/restore tape containing the starter system (which installed like the D3.0 release). This release was twice as big as the last release, and was in a slightly different format. The tapes were in the format of a dump produced by the MTS *FS program, and the installer could use a program called *RAMROD* to create the new nucleus. The release was a massive program update of the previous release, and did not really add anything new.

D4.1-D4.3

Coming on one 6250 BPI reel in May 1978, D4.1 introduced a better way of communicating what changes were being made. A clever SNOBOL4 program generated a “change form” that described what changed. This release was an overhaul of system internals, and not so much programs.

March 1979 bought D4.2, which came with a dump/restore tape (which meant the system could be installed straight away for a new install site); alas, that dump/restore tape has seemingly been lost. This version contained updates to most of the programs and compilers present on the system; since it came on 3 tapes and was a pretty large update, the University of Michigan saw it fit to just include a restorable system.

D4.2A came in July 1979, and was rebadged as Redistribution 16 (RD16) -- this may seem confusing as first, but U of M asked for all of the MTS sites to send in tapes containing their system to be redistributed to every other MTS

site. This was done in a rather haphazard method, and there was no attempt made to produce anything that was consistent, simple, or easy to install. Alas, a driver file was made, so installation was at least possible.

D4.2B (RD17) came in October 1979, and was a very small supplement intended to overlay on D4.2A.

D4.3 released in May 1980, and added support for more hardware. Growing on the 1977 original 4.0 release, this release included updates to:

- PDP-8 cross assembler
- Linear algebra software
- Waterloo FORTRAN updates
- Extended XPL
- Support for the Memorex 1270 (a 3274 clone)
- ALGOL compiler updates
- Additional terminals (TI Model 733, DECwriter)

D5.0

Around this time, MTS releases started to slow down. Most sites had started to explore other interactive timesharing systems (including early UNIX, VMS, and even VM/CMS), but D5.0 was released in August 1981. A full 360 MB distribution on 4 tape reels at 6250 BPI density, the following new components were included:

- PL360
- Extended XPL
- GASP (mentioned before, this is the General Motors Associative Programming Language)
- WIREWRAP
- REDUCE 2
- PDP-9 remote graphics terminal driver
- Statistical computation programs
- Linear algebra programs
- Fourier transform routines
- PDP-8 assembler and linker
- SPIRES
- MTS APL
- Z80 and 8080 assembler

D5.1

This release was mostly the same as D5.0, and was released in August 1983. Note the programs that were added on D5.0 -- MTS was a key platform for the burgeoning microcomputer race (and this would end up being its undoing!). This distribution consisted of a one-tape update to the base release, and could be installed with the driver program.

D6.0.

At long last, the last version of MTS was upon the world. Many years after the D5.1 release, U of M released D6.0 in April 1988. By then, TCP/IP support had been added using the aforementioned two host interface processor methods, but development had mostly stalled after D5.1's released -- the shutdown dates of various MTS sites were mostly in the early 90s (as mentioned earlier). This final version was the “gold master” of MTS, and consisted of 9 tapes.

Alas, this version was the first to support 370/XA machines, and TONS of new features were added. These include:

- 370/XA support, but not ESA/370 support. This would be rectified in the future, though.
- Named Address Spaces, something that functioned like Discontinuous Saved Segments on VM. These were used to allow programs to share libraries and data, and loaded at IPL time into the virtual address space.
- Expanded Storage support, which MTS used as a store-through-mode disk cache. Writes were written immediately to DASD, but reads were backed by the cache.
- Vector Facility support. This was used by having an program that used the opcodes and registers for the VF; MTS would save and restore the contents of the VF across task-switching timeslices.
- More message support. This was an enhancement to the *MESSAGES facility to allow more than 2.1 million messages.
- More programs: $MAKE, $FTP, $DUPLICATE, $FSM (Full Screen Message), $HELP, and an improved editor
- Resource Manager. This was a program, written by UBC, that could be used to drive an NJE network (it apparently also had printing support, but this was not used in a distributed-system MTS cluster).
- Network support. This added support to the base distribution for UBCNet and Merit network interface hardware.
- HIM support. This is the aforementioned TCP/IP host processor for MTS, developed at UBC. This supported UDP, TCP, and TLNT (telnet) devices on the logical channel the HIM exposed to the host, and could run an FTP server.
- Updated DASDI program. This added support for 3380s and CMS minidisks.
- Rewritten FBA support. This new FBA driver code could use the integrated 9370 FBA DASD disks (the 9332 and 9335, as well as clones, were supported).
- Network Server support. A Server is a different type of connection to MTS (the other two being Terminal or Batch) wherein a remote system would connect to a system to perform a login; once that was validated, files and jobs could be transferred.

Needless to say, MTS got very advanced on this release; this release was certainly the largest of all of the predecessors by a significant amount!

D6.0A

Years after, in January 2012, three hobbyists (who are Mike Alexander, Jeff Odgen, and Gavin Eadie) dusted off the MTS 6.0 distribution and tried to get it running on Hercules. To their surprise, it mostly worked... but the installation process was painful and not very turnkey. As such, the MTS D6.0A distribution was made available in the form of a single 3380 disk (alongside a Hercules configuration file).

Author's note: I tried to get this to run on a real S/390 mainframe, but I couldn't get it to IPL; apparently, it will not work with either a Bustech or IBM Shark 3380 controller!

```
 University of Michigan Computing Center - Device: DS02   Task:    40   ge 1/1
%


MM        MM  TTTTTTTTTTTT   SSSSSSSSSS
MMM      MMM  TTTTTTTTTTTT  SSSSSSSSSSSS
MMMM    MMMM       TT       SS        SS
MM MM  MM MM       TT       SS
MM  MMMM  MM       TT       SSS
MM   MM   MM       TT        SSSSSSSSS
MM        MM       TT         SSSSSSSSS
MM        MM       TT                SSS
MM        MM       TT                 SS
MM        MM       TT       SS        SS
MM        MM       TT       SSSSSSSSSSSS
MM        MM       TT        SSSSSSSSSS
```

Figure 19. MTS 6.0A user terminal screen

CP/CMS, VM/CMS, and z/VM

In the wake of TSS's commercial failure as a software platform, a true successor would emerge. This came in the form of VM/CMS, but its history was long and convoluted. Despite having a rough start, it found itself cemented in a rather interesting position in the world of computing: many saw it as a parallel universe alternative to UNIX, and several people remarked that CMS had better UNIX features than UNIX itself (a big example of that being the infamous "CMS Pipelines" program).

Origins: before VM

The first timesharing system ever demonstrated in a meaningful capacity came out of MIT in 1961. The so-called "Compatible Timesharing System" (or CTSS for short) found itself as both an academic miracle and a commercial marvel. While CTSS was originally somewhat primitive, it would later go on to influence every other timesharing system (including VM, but also Multics). Keep in mind, computing was radically different during CTSS's heydey; computing then was nearly entirely based on batch processing, and interactive work was almost completely ignored. While CTSS was used in a meaningful capacity from 1964 to 1971, a second-generation system was quickly wanted. MIT (and others) saw the massive success that academic and scientific users had with timesharing, and wanted to spearhead the second "greatest hit in computing" of the time: Project MAC.

While IBM was trying to get commercial computing companies to improve their timesharing offerings, it quickly became apparent that they would have to do most of the work themselves. Project MAC would later go on to produce Multics on the GE (and later Honeywell) mainframe line, but the "spirit of the problem" still remained. One of the hardest problems was finding a computer that could actually sustain many logged-on users. While the Project MAC team settled on the aforementioned platform, many industry vendors provided their own bids for hardware. One of those was the repeat-supplier IBM (who had already given MIT plenty of computing hardware and support before), and they offered a System/360. However, the MIT guys hated the 360! The machine didn't include the address translation hardware they so desperately needed for timesharing (the infamous "DAT box" that would form the virtual storage features of the System/370 machines later), and this was a major blow to the project.

Nonetheless, they would eventually have their wish and get a DAT box (which they called a "Blaauw Box" after Gerrit Blaauw, its designer); sadly, even after all of this, IBM would end up losing the Project MAC contract (they also lost important connections with Bell Labs, who famously became a GE customer thanks to their partnership with MIT and the development of Multics). The MIT guys, armed with their successful fulfilled proposal of what was the first System/360 Model 67 (with that direly-needed Blaauw Box) -- this machine would be destined to run TSS/360... but the keen reader will know the blazing failure that this OS was.

Part of the reason IBM never included timesharing as one of the original goals of the System/360 project was simply because of the target market: the System/360 was intended to be a business batch processing machine, and not an interactive timesharing scientific/academic machine. In that era, there were some hot criticisms of virtual memory systems that most people have forgotten: the first machine to have virtual memory paging was the Ferranti Atlas (which was one of the first supercomputers, and was located in the UK). Many people remarked that the virtual memory features on the Atlas didn't work (or worked very poorly), and nobody understood why; the engineers would later attribute it to the machine having a rather low amount of real memory, and the memory-consuming loads were making the system thrash heavily.

While IBM was originally devastated by the loss of the Project MAC contract, the Lincoln Laboratory eventually did get their aforementioned System/360 Model 67. IBM also got their ducks in the row and followed GE's lead: they began to take timesharing seriously, rather than trying to "get rid of" the scientific/academic demands of the universities with customized PRPQ-type operating systems (like RAX).

However, the shipment of the Model 67 got pushed back further and further, and the engineers began to get impatient. A System/360 Model 40 was available to the engineers, and they quickly got to work designing some hard-

ware upgrades and microcode updates that would implement a reduced-function implementation of the DAT Box on the 360/40. In 1965, they got to work.

CP-40: In the middle of 1965, the engineers could finally start their work on CP-40's virtual machine control program. They had already been dismayed with TSS/360 and essentially began to ignore it; they noted that TSS/360 was comprised primarily of lofty goals and forgotten hopes. Meanwhile, CP-40 and CMS were coming along nicely!

First, CMS was written. The so-called Cambridge Monitor System would function as a single-user single-tasking interactive OS that would run on the 360/40 without the need for the virtual memory features. When they were ready to try to run several CMS sessions, they hardcoded a variety of virtual machines into the CP-40 nucleus and regenerated it. Growing on the lessons learned from the M44/44X project (the first machine that actually implemented virtual machines, on a modified IBM 7044), the so-called pseudomachines (of which there were 14) with 256K of storage quickly sprang to life. The virtual CMS disks were presented to the user's VM by divvying up the computer's real disk storage into a variety of extents (called "minidisks"), unit-record devices (printers, punches, and card readers) were emulated using a spooling technique (still used today by z/VM), and the typewriter terminals provided CP console functions that would allow the user to start or stop their VM, manage devices within it.

CMS: On the heels of CP-40's development (technically before), CMS was finding itself maturing. The engineers that implemented the original CMS used a variety of System/360s around the Boston/Cambridge area (including the famous modified 360-40 while it was still on the factory floor in Poughkeepsie). The OS of choice that was used for CMS's development was none other than BPS/360 (if you could even call that an "operating system" in the traditional sense). To load the early versions, they used BPS's three-card loader; this was placed before the CMS nucleus in the deck and the machine was IPLed from the reader.

Eventually, CMS was becoming mature enough that the programmers could use CMS to develop CMS (the OS was then said to be "self-hosting"); when CP-40 matured enough to be able to run multiple copies of CMS, the programmers quickly jumped at the opportunity to do so and timesharing CP/CMS was born.

CP-67: Eventually, in September 1966, the engineers working on CP-40 at the Cambridge Scientific Center realized that they would be able to migrate CP-40 to the System/360 Model 67 they had finally gotten. However, the migration was not so smooth: the engineers ended up rewriting a substantial amount of CP to be not only more flexible, but also more performant. Exciting new features included:

- Dynamically-allocated control blocks
- Variable number of virtual machines
- Larger guest virtual storage
- Most CP modules made reentrant

As CP-67 development rolled on, the engineers at the MIT Lincoln Lab started to become extremely frustrated with the lack of quality of TSS/360; they looked at CP-67, saw that it was not only passingly similar to the old CTSS, but was actually stable and usable (a tall order for TSS in its era, which often did not stay alive for long and required constant re-IPLs). Seeing this, the Lincoln Lab computing crew started asking IBM for a tape of this "miracle" software product. Keep in mind, IBM was *heavily* invested in TSS, and a major customer like the MIT Lincoln Lab ditching TSS would be a major blow to IBM.

Indeed, it was, and the Lincoln Lab put CP-67 into production in April 1967. Both Lincoln and Cambridge worked hand-in-hand on CP-67 by then, and other customers started to join the CP-67 fray. One of the most influential ones was Union Carbide: they had been considering running TSS, but its massive lack of quality led them to seek a suitable alternative. TSO on OS/360 had not yet matured in any releasable capacity, so they chose to run CP-67. Because CP-67 was still very much so an academic software project, they sent some of their computer staff to Cambridge to work on CP-67; they were directly shaping the future of the software they were running, something IBM was strangely scared of engaging in with TSS.

TSS was not completely dead in the water, however. Melinda Varian remarked at SHARE 30 (in February 1968) that there were at least 18 sites with System/360 Model 67s trying to run TSS, but none of them were satisfied with the results. During SHARE 30, IBM famously announced that they would be de-committing from TSS (this was essentially the final nail in the coffin for TSS, which had nearly no sites running it as CP-67 was gaining massive popularity over it).

CP/CMS: In May 1968, CP-67 had matured enough that IBM was ready to start providing it to installations in a meaningful capacity. IBM released it in the so-called Type III Library (i.e. a collection of user-written source code libraries) the next month in June. Within 2 months, two timesharing companies were founded to sell CP/CMS timesharing (those being Computer Software Systems -- later renamed to National CSS -- and Interactive Data Corporation, eventually renamed to ICE Data Services) -- these companies poached a variety of key CP/CMS engineers from Cambridge, Lincoln, and the existing CP/CMS cites, but these companies' efforts provided a major publicity boost to CP/CMS, virtual machines, and the cumbersome engineering project that was the System/360 Model 67.

By April 1969, there were at least 15 360/67 sites that were running CP/CMS. In June of the following year, Version 2 was done; Version 3.1 (released later in November 1971) could run up to 60 CMS users, stably, with minimum downtime. By the first quarter of 1972, there were at least 44 sites running CP/CMS; Version 3.2 was released around that time as a maintenance release... by then, a quarter of those 44 CP/CMS sites were within IBM. At this point, TSS was essentially dead.

In May 1970, on the heels of the System/370 announcement, the CP/CMS programmers and architects sought to upgrade CP/CMS to run on the new System/370 processors (though with caveats; see the section in the System/370 hardware history at the start of this book on the controversy surrounding the availability of the DAT box on the launch models of the 370). To achieve this, the CP/CMS programmers actually emulated a S/370 on a S/360-67; this emulation technique had already been employed during the S/360-40 to S/360-67 transition, and it would be pivotal for future generations of the IBM System/370 descendants.

This emulation technique proved to be a major boom for what would become CP/370's popularity; the OS/VS programmers had no real S/370s to try their OS on, and being able to emulate the machine they were targeting meant they no longer had to "fly blind" with development -- this was further helped by a lack of prototype working S/370s during the OS/VS development cycle. IBM had been trying desperately to get rid of CP/CMS, but they had completely failed at this point: the CP-370 developers actually won an IBM award for their efforts| Eventually, with the S/370 Advanced Function announcement, CP/CMS was officially discontinued.

VM/370

After CP/CMS's discontinuation, IBM unveiled the S/370 AF systems that came with stock virtual memory management hardware -- because of this, IBM announced four new OSes:

- OS/VS1
- OS/VS2
- DOS/VS
- VM/370

The new VM/370 was a reimplementation of CP/CMS's control program, which certainly benefitted from the "second time run-through" design improvement principle. 110 people within IBM were now officially working on VM/370. It goes without saying that the development of VM/370 was certainly rather cumbersome. For one, there was the aforementioned massive shortage of prototype System/370s; the result of this was that CP-370 had to be developed under CP-67 emulating a 370. This was necessary because the virtual memory relocation hardware on the 370 was different than the 360: the 370 allowed for two different page sizes (2K and 4K), plus two segment sizes (64K and 1M) -- the 360 only supported 4K pages with 1M segments.

CP-67 was modified to emulate the 64K segments and to trap/emulate the 370-specific instructions. This modified CP-67 was actually ran as a guest of the production CP-67 first-level system at Cambridge; in order to evaluate the functionality of CP-67's emulation of a 370, CP-370 was ran under that modified CP-67 second-level system. Next, the 370 CMS could be ran under that third-level CP-370 as fourth-level virtual machines; CP-370 could also be ran fourth-level with the 370 CMSes running at the fifth level to ensure CP-370 could run itself under itself.

In January 1971, a prototype System/370-145 (one of the few that had a DAT box, which was distinctly different in design from the earlier 360/37 one) was made available to the programmers at Endicott. They travelled up there to test it, and the CP-370 prototype worked on the first try! The Cambridge people finally got a non-prototype 370-145 with a DAT box in the fall of 1971 amidst security precautions: there was quite a bit of concern in the center that the people who worked in the adjacent buildings would notice that the 370 being delivered to Cambridge had a DAT box, and this would go against what IBM had told its customers. In February 1972, the first fully-working VM/370 CP nucleus was IPLed; this resulted in a very high development velocity moving forward into July 1972, wherein an internal-use-only distribution of VM/370 was finished on the 5th of the month.

This was on the head of System/370 AF announcement (which would come on the 2nd of August, 1972); all of the new 370s would have DAT hardware and such, plus the aforementioned operating systems earlier in this section were announced. Note that VM/370 was up and running in the IBM Field Support Centers the day the 370 AF announcement hit (unlike OS/VS2, which was still far off from being finished).

In order to have a "sane" reference platform, a 370-145 with 512K of storage was the target for VM/370 Release 1. Note that when Release 1 was available, the largest machine that you could buy was a 370-168 with 8M of storage; the VM/370 marketing team assumed that only about 1 customer would be running VM/370 on a 370-168, but the first one sold and delivered was destined to only run VM/370 timesharing. A decade after this announcement, about 10% of the high-end machines built in Poughkeepsie would have VM installed; 15 years later, VM would outpace MVS in terms of number of licenses sold.

The VM developers at IBM were notorious for producing an unbelievable amount of code per each programmer; each programmer produced well over a thousand lines of assembler code for VM per month. The best part is the VM programmers were actually having fun! There is an interesting anecdote documented in *VM and the VM Community* wherein one of the systems assurance testers was tested with a challenge consisting of a bunch of junk code: these things would do things like wish the developers happy birthday if the system had been IPLed on their birthday, but other things (like something that printed "BONG BONG BONG BONG" on every terminal at midnight) were left in surprsingly long. The man who did that testing, Dick Newson, remarks that his team wanted to modify the VM DIAL command (enterable on the login screen/prompt) to say "Aren't you glad you use DIAL?" but this fell through due to nonexistent legal threats they were scared of (DIAL is a brand of soap sold in America).

VM/370 Update Process: By the time Release 1 had solidified, most of the development on VM was happening within IBM (which included the Cambridge Scientific Center). Starting in January 1973, the development crew was moved to Burlington (adjacent to Boston). Once they were there, they began to provide service tapes for VM Release 1. These so-called Program Level Change (or PLC) tapes contained more and more features that would stack on top of the base. Release 1 PLC 09 added the CMSBATCH service machine mechanism (which still lives on in z/VM), the ability to spool a console to a file for further review or printing, support for the aforementioned 370-168, support for the 2305 paging drum memory unit, a rewritten scheduler that would replace the basic scheduler (this was called the "biased scheduler"), among other things.

VM/370 Release 2 and Growth: When Release 2 came out, VM was already growing somewhat rapidly (in April 1974). The old typewriter terminals were starting to be replaced by a relatively new (and now famous) peripheral: the 3270 series terminals. There is an anecdote wherein Dick Newson walked around MIT and observed the users of early CRT monitors; he noticed that the scrolling was too extreme and came up with the screen hold feature.

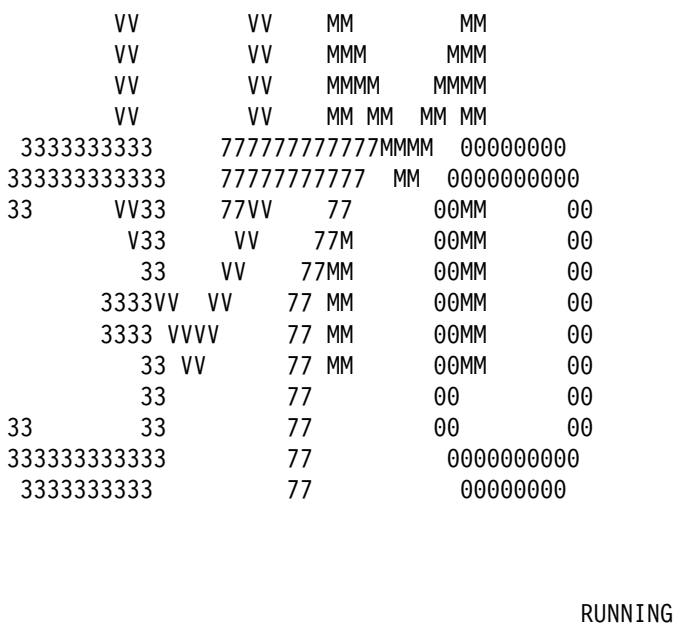


Figure 20. VM/370 Release 4 login screen

PLC 05 (on Release 2) gained support for the new 3705 communications processor, which necessitated the creation of commands to load and IPL the communications processor (as it was a computer of its own). The 3705 generally replaced the earlier 2703, and provided all manner of telecommunications line capability. To grow on this new support, PLC 11 January 1975) added a now famous subsystem to VM: RSCS (see the notes on this below). In March 1975 (with PLC 13), OS/VS1-CP paging handshaking was added as well as the CP monitor system (which was necessary for future programs like VM/RTM). PLC 23 (February 1976) added IPCS (indispensable for debugging), and VM/370 Release 3 was released shortly after.

Release 3 added both support for 3350 DASD and also VMCF (which allowed VMs to intercommunicate, and would later live on as IUCV, aka APPC/VM).

CPREMOTE, RSCS, and VNET: Around the time Release 3 was gaining an install base, various IBMers sought a native inter-VM network system that could allow for file sharing between systems. This had already somewhat been written starting in 1969 by two Cambridge people in a package known as CPREMOTE -- this was essentially the first example of what we would now call a service virtual machine (which lives on today, just check out any z/VM system). This program provided a communications method in which two CP-67 machines shared a common spool interface. Since it was distributed alongside CP-67, many places within IBM adopted it quickly, but its shortcomings were obvious. Alas, a fork of it was produced (called CP2780) that controlled terminals for RJE (Remote Job Entry, a common feature found on mainframes at the time; a 2780 was an RJE workstation).

CPREMOTE was certainly rather popular, but it ended up falling into a situation where it needed to be replaced, badly. This manifested in 1971, wherein the CPREMOTE authors decided it was time to start over, and try to combine the features of CPREMOTE and CP2780. The resultant program from this effort was called RSCS, or the Remote Spooling and Communication Service. The original RSCS was released, as mentioned, in 1975; it was, however, mostly incomplete. This bothered the authors, who had literally left in dummy function calls to functions that would facilitate remote communication over a protocol that would eventually become known as NJE (Network Job Entry). To alleviate this, a modified RSCS was produced (which was a PRPQ) called VNET. Now that it was

available, VM/370 sites quickly repurposed some spare BSC lines on their 2703s and 3705s to form a network: this happened within IBM naturally, but was initially hampered.

Initially, VNET used asynchronous modems. This meant that an operator in the computer room had to physically dial a phone number on a telephone, place the phone receiver into the modem acoustic coupler, then start VNET processing on both sides. Since VNET was mostly used within IBM, someone had a clever idea: IBM had tons of leased lines between sites, and some of them were unused. The VNET users, realizing this, quickly found unused phone lines that ran from one site (or building) to another, attach it to a synchronous modem, and start the network links.

The first multi-site network was, as documented in *VM and the VM Community*, was called Sun (or the Subsystem Unified Network, no relation to the Stanford University Network of UNIX fame). Apparently, this was two disjoint networks that were bridged together by a phone cable ran across a parking lot at a California IBM site. The protocol spoken over the line used a modified RSCS (which itself would find its code merged into mainline RSCS, made available as the NJI line driver). An engineer reverse-engineered HASP/JES2's NJI line driver protocol (as it allowed for critical things like routing between/through multiple nodes), which at that point was completely undocumented. When the two sides negotiated the link, the engineer (Tim Hartmann) ran a banner-printing program on a machine located in Poughkeepsie and had the output print on a printer attached to a machine in San Jose... which was quickly noticed by the author of the HASP/JES2 NJI line driver code (Ken Field). Upon noticing this, Field requested several more copies and taped them on the walls all over the building. The message? ***Machines of the world unite! Rise to the SUN!***

By Feburary 1976, there were 50 systems connected to the burgeoning VNET network (as it had come to be known by that point). By March 1979, there were at least 239 systems spanning 11 different countries. Because VNET grew so quickly, essentially every VMer at IBM was networked in some form or fashion. Due to this, someone named Peter Capek hada a bright idea: Make a *VM Newsletter*, for VM, on VM, sent through VM!

The newsletter was replaced in 1983 by a more purpose-built conferencing system called IBMVM, managed by a program called TOOLSRUN. IBMVM was a "conference" comprised of many files (which for some reason are called "fora", hence TOOLS-adjacent support programs with FORA in their name). The conference disk was mirrored on systems attached to the network, and this information-sharing scheme produced massive gains in productivity for the IBMers (as TOOLS conferences were starting to be used by everyone, not just VMers).

Of course, IBM's dependence on a network meant it would not be long before someone would try to exploit said dependence for either humorous or malicious means. This came in the form of an interesting script that hit in December 1987. The so-called CHRISTMA EXEC made its way to VNET through BITNET, which itself was attached to the European college NJE network named EARN.

VM/370 BSEPP and SEPP

The road to a VM/370 program product was long and arduous. In 1976, IBM forked VM/370 Release 3 PLC 06 to emulate 370/XA VMs on a standard System/370. All the while, IBM was hot on the trail of VM/370 Release 5 in 1977 (which introduced VMAP as an optional program product). There were also made available two system-extension versions of Release 5:

- VM/370 BSEPP (Basic System Extensions Program Product), which introduced support for FBA DASDs, EDGAR (sometimes called the Display Editing System, the first real fullscreen 3270 editor), and CP improvements.
- VM/370 SEPP (System Extensions Program Product) introduced the Wheeler Scheduler (offering greatly improved CP dispatch performance), a variety of user modifications people had written (many of which were sourced from the Waterloo VM Library), the ability to send VM accounting records to disk, swap table migration, page migration, and shadow table maintenance enhancements (the last three are copied verbatim from *VM and the VM Community*).

Much of the "controversy" around VM/370 in the 1970s (the so-called Doubtful Decade (as it was named in a SHARE presentation given in 1978) revolved around a series of questionable release choices that resulted in rather bad user satisfaction during that era. Initially, users were actually rather disappointed when the original VM/370 release was made available; Release 3 was another low point, but users began to like VM/370 quite a bit (i.e. were impressed with its stability) around Release 5.

VM/370 Release 6 and BSEPP/SEPP Release 2: In 1979, VM/370 Release 6 hit the market. Users quickly installed it, and it was often noted to be one of the most stable versions of pre-VM/SP VM made. Release 2 of BSEPP and SEPP introduced a feature that became critical for networking later: logical device support; this release also revamped the CMS filesystem with its new version: EDF (Extended Disk Filesystem). The new version of RSCS, the Program Product version, was also released on the same day; it totally overhauled line driver support and combined all of the modifications made to RSCS over the years. Though it was lacking, a CMS HELP facility was also introduced; the version found in this release was nothing more than a text file displayer and was a far cry from the panel-and-menu-based HELP system seen on z/VM today.

BSEPP and SEPP were essentially stopgap releases, and were essentially warm-up measures for a truly program-product version of VM. This would come shortly after in the form of:

VM/SP

In late 1980, VM/System Product Release 1 hit the market. Gone were the days of having the complete source code to VM; most of VM/SP was now copyright IBM and written in a language people did not have compilers for (PL/S and later PL/X). Alas, new features were added:

- EXEC 2, a new CMS scripting language
- IUCV, the evolution of VMCF
- A MIH (missing interrupt handler) for device reliability
- Multiprocessor system support
- Subsystem communications
- XEDIT, Xavier's vastly improved EDITor

Though about 100,000 new lines of code were thrown into the system, there had been a massive lack in testing. Many people realized that VM/SP Release 1 was thrown out the door and onto the market well before it should have been, and it was obvious! CMS would randomly corrupt minidisks, CP would randomly crash, and IBM revamped the PUT process that made it even harder and slower to get fixes shipped to customers. VM/SP took all of the improvements of VM/(B)SEPP and slammed them into a bunch of new code; it was a mad sight to behold.

It seems that IBM did not learn from their mistakes with OS/360, and, as mentioned, they threw tons of programmers at VM/SP R1's development crew. Since none of these people were VM experts (or even passingly good from historical reports), the fixes that this crew did produce were of terrible quality.

One of the more questionable things added to VM/SP R1 was something called VTAM -- the VTAM Communications Network Application. This was a primitive implementation of SNA for VM, and it functioned by having a dedicated MVS guest system that communicated with CP via VMCF. Needless to say, this was not a good solution; many VM systems programmers quickly became fed up with VCNA as it required them to learn MVS, learn SMP/E, figure out MVS JCL, and other such things foreign to them. It would be a while before true SNA networking came to VM with ACF/VTAM V3 on VM/SP Release 4.

PROFS: In late 1981, IBM made available a program that would end up becoming absolutely famous: the Professional Office System, or PROFS. IBM had worked with a company named AMOCO (a federal credit union in the United States) to develop PROFS, and IBM released it as a PRPQ. This PRPQ version was a massive success, and

IBM released it later as a fully supported program product in the mid-80s. IBM became somewhat dependent on PROFS, and, by 1987, there were a million PROFS users outside IBM. There is an expression that the PROFS main menu welcome screen was the most displayed thing on any CRT terminal from any manufacturer, but this may just be a clever marketing ploy by someone steeped in cleverness.

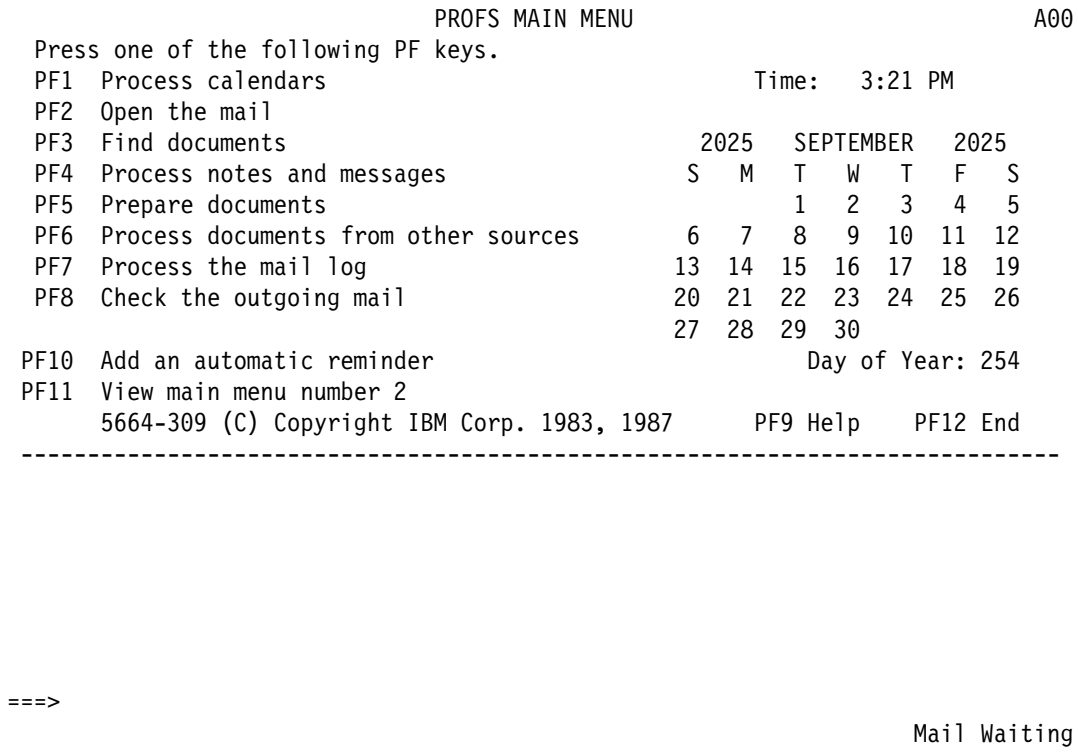


Figure 21. PROFS V2 R1.1 Main Menu

VM/SP R2 and R3: As the 1980s rolled on, IBM decided it was high time to ditch their heavily patched SP 1 system (the one in Poughkeepsie is the VM system in question), and the results were staggering. Now that the VMers had a taste of their own medicine, they had a change of heart; SP Release 2 contained a massive number of fixes that SP 1 should have had, ranging to a whole host of bugfixes to a command retrieve key. One of the interesting things added were the Productivity Tools, the now-familiar FILELIST, RDRLIST, and other such menu-driven things.

VM/SP Release 3 added REXX, and it quickly filled a void of the poor EXEC 1 and EXEC 2 programming environment. Mike Colishaw had been working dutifully on REXX, trying hard to get it into VM; some people saw it as BASIC, but many saw it as the true potential-bringer it was. Not before too long, the entire system was using REXX *everywhere*.

VM/Passthrough: Going by several names, one of which was PVM, was an interesting cross-system networking product for VM made in the late 70s. PVM emerged as a program named *V6* in 1974. Its slated purpose was to allow remote access to the IBM RETAIN (a database system that service personell can use to catalog problems and note solutions) system in Raleigh. By 1980, VM/Passthrough was launched as a real product.

Strangely enough, when PVM was made available as a product, it had already had years of battle-testing; it was so well-used within IBM that it was launched with essentially no bugs. PVM started to evolve quickly, and began to gain many new link types. By its final release (2.1.1, 1998), it had support for a wide variety of links:

- CTCA
- Bisync

- APPC/VM
- TCP/IP
- 3274 emulation
- IUCV (local system only)

```
                          VM/Pass-Through Facility

            You can select a node with the cursor and press ENTER

  L VMESA      N ZVMA       N ZVMB       N ZVMC       N VMHPO      N VMSP
  N ZVMD       N VMESA370   N ZVME       N VMESA21    N VMESA12    N VMHPO42
  N VMIS       N VMESAG     N VSEESA     N VISIONS    N EVIEVM     N EVIEVSE
  N EVIEVMG



Destination ----------------->          Port ----------------------->
Route ----------------------->          Language --------------------> AMENG
End Session ----------------> ####      Verify ----------------------> OFF
Your Identification ---------> GRAF020  Nickname --------------------->
Session Roll ----------------> ?SELECT  Menu Select -----------------> ?MENU
PF8= Scroll               CLEAR key = Top Screen                   PA1= Exit



****>                                                                  <****
****>                                                                  <****
```

Figure 22. VM/Passthrough menu

VM/SP HPO: March 27, 1982 introduced a new version of VM: the VM/SP High Performance Option. The first version was Release 2 (for VM/SP Release 2). Later versions, namely HPO Release 3.2 and 3.4, were massive performance booms. One touted feature was an increase in how much storage could be used; Release 3 added support for the Extended Real Addressing scheme described in the hardware section, allowing for 64 MB of real storage to be used on the system. Other features included:

- Improved storage management and simplified tuning
- Users can create up to 9900 spool files, rather than having the entire system limited to 9900 files
- Scheduler enhancements, mainly with regards to storage over-commitment
- Expanded storage can now be used for paging, allowing for relatively fast access to more than 64 MB of real storage
- Extended hardware support (4381s, 3090s)
- Sped-up TSAF (will be discussed later)

There were many VM/SP HPO releases, here they are:

- R2M5
- R4M0
- R4M2

- R5M0
- R6M0

VM/XA MA: I shall discuss in more detail later the details of VM/XA SP (which came later), but the VM/XA Migration Aid was a primitive 31-bit XA-mode version of VM derived from The Tool (as it was known); it was the program product version of the forked VM/370 CP that could create XA VMs but modified to run XA VMs on an XA host. Just like the first run of CP-370, VM/XA MA IPLed first try on a prototype 3081 when it became available. The Migration Aid was intended as a tool to help users migrate from OS/VS1 and OS/VS2 to the new MVS/XA, while running their "old" 370-mode OS/VS images as they moved.

The OCO Announcement: February 8, 1983 was a sad day for VMers. During the runtime of SP 2 and SP 3, IBM announced (as mentioned earlier) that the complete source would no longer come with the product. It was now Object Code Only! VM customers fought tooth and nail against this, even going as far as the time when SHARE published a white paper; IBM never responded to it. There was a particular good example of the OCO policy gone wrong with:

VM/PC: If you read the section earlier on the XT/370, you will recall that the XT/370 was provided with a version of VM called VM/PC. This was the first real OCO version of VM, and it was a colossal failure. Many features were implemented incorrectly (or not implemented at all), and the customers of it were powerless to correct these issues without the source and associated compilers! As such, VM/PC was dead as soon as it was released.

VM/SP Release 4: Late in 1985, IBM released SP 4. People adopted it, but... it was clear IBM had not learnt their lesson. SP 4 had just as many bugs as earlier versions, but some new features were actually derided for being bad replacements to earlier features or were of poor implementation quality. The major feature that was added was CMS HELP (as mentioned earlier): sure, it was now a menu-driven system, but the help menu pages were essentially copy-paste versions of the hardcopy manuals made to be read on CMS.

```
 VM/370 ONLINE--PRESS ENTER OR CLEAR KEY TO BEGIN SESSION


         VV        VV    MM        MM
         VV        VV    MMM      MMM
         VV        VV    MMMM    MMMM
         VV        VV    MM MM  MM MM
  3333333333     777777777777MMMM  00000000
 333333333333    77777777777  MM  0000000000
 33      VV33    77VV    77      00MM      00
          V33     VV    77M      00MM      00
           33    VV    77MM      00MM      00
        3333VV  VV    77 MM      00MM      00
        3333 VVVV     77 MM      00MM      00
           33 VV      77 MM      00MM      00
           33         77         00        00
 33        33         77         00        00
 333333333333         77          0000000000
  3333333333          77            00000000


                                                RUNNING   VMSP4
```

Figure 23. VM/SP Release 4 logon screen

SP 4 also introduced GCS, the Group Control System. Its slated purpose was to run a true real port of VTAM on VM, but customers derided it too for being a closed system with no source. Alas, the new VTAM V3 port to VM was reasonably well-received otherwise. GCS, however, almost completely stagnated; with no source and little documentation, only IBM provided programs that ran under it. For reference, GCS provided a simplified implementation of a MVS scheduler, some supervisor calls, and console emulation that allowed for porting MVS programs over to VM; one would think that many customers would move off of MVS and onto VM because of this (since GCS could, in theory, run applications either in one VM or in several VMs), but nobody did. When it was all said and done, there were only a few products that used it: VTAM, RSCS, NetView, and not much else.

CMS gained a new feature that almost nobody wanted: CMS Windows(properly called CMS Session Services). It was a cool novelty, but many people asked *what's the point?* Release 4 also added multiple language support, but it was cumbersome to use, difficult to implement new languages for, and very awkward. Users that found themselves as large RSCS routing nodes found their performance degraded, and it wouldn't get any better with Release 5.

WISCNET and VM TCP/IP: In the early 80s, people at the University of Wisconsin had a bright idea: what if we produced a TCP/IP stack for VM? The result of this was a project called WISCNET, which provided the aforementioned Internet support for VM. This project was of particular interest because IBM worked with the university; in the age of OCO software coming out of IBM, a product that came with its source was a breath of fresh air.

By 1984, IBM was selling WISCNET as a PRPQ; this was later adopted by a team within IBM as a true program product. The result of that effort was the famous "FAL", so named after its program number (5798-FAL). While some people saw it as a Pascal monster, others saw it as a solid product. WISCNET (and later VM TCP/IP) both contained code to build NJE-to-Internet mail gateways. For sites that had both WISCNET and VM TCP/IP tapes installed, they would find themselves possessing a variety of interesting programs:

- A TN3270 program for UNIX
- X Windows for VM (later)
- Both Pascal and C API support, including support for the standard BSD socket API on CMS

WISCNET's software requirements were rather slim, only requiring VMCF support for the intercommunication between the TCP/IP stack virtual machine and server/client VMs. As for the hardware side, things were much more complicated. In the "modern era" of System/390s, mainframes possess "open systems adapter" cards (as discussed in more detail in the earlier hardware sections). These OSAs emulate something called an IBM LAN Channel Station, essentially a DOS PC fitted with at least two cards: a bus-and-tag channel card, and at least one Token Ring or Ethernet card. A program would run on that DOS machine that provided the service of what became to be known as a "LAN Channel Station." The 8232 LCS became available during the time IBM was selling VM TCP/IP, so things were much different in the era of WISCNET.

If you wished to attach your WISCNET install to an Ethernet network, you needed a box called a DACU (device access control unit). This box contained a shortened DD11-CK UNIBUS backplane that held a bus-and-tag card and an Interlan Np100 10 megabit Ethernet UNIBUS board. There was an IBM PC that ran the control software; this device was the direct inspiration for the later 3172 LCS.

During the time of WISCNET, there were some other competing solutions: one of which was *Spartacus TCP/IP for VM*, originally written by people that had written a TCP/IP stack for MVS called KNET. Much like WISCNET and the DACU, KNET and Spartacus used a similar box to provide an Ethernet port on the mainframe (theirs was called the K200). Of course, WISCNET had IBM's backing and ended up winning out; the DACU was transformed into the 8232 LCS, and the LCS also gained support for SNA over Ethernet and Token Ring (though it provided a slightly different operating mode to do this). When the P/370, P/390, IS3006, and MP3000 became available, the system provided device emulators that emulated both the Ethernet and SNA operating modes of an 8232 LCS. For more information on the 8232, see the earlier section in this document that describes it!

SQL/Data System: In an era of uncertain futures of databases, IBM produced System R -- the first SQL database. While it ran on MVS and was barely popular, an offshoot of it became a massively popular product in the 1980s. This product was SQL/DS, and it ran on VM and VSE; MVS customers got Db2 (which was also SQL). SQL/DS and the VM Shared File System actually are both derivatives of System R! If you've used unixODBC or Microsoft SQL Server, you may have noticed a program named something like ISQL. Believe it or not, this is actually derived from SQL/DS, which provided commands just like that that did the same thing (that is, provide Interactive SQL where you can run SQL statements directly).

```
isql
ARI0659I Line-edit symbols reset:
         LINEND=# LINEDEL=OFF CHARDEL=OFF
         ESCAPE=OFF TABCHAR=OFF
ARI0320I The default server name is SQLDBA.
ARI7716I User WEC connected to server SQLDBA.
ARI7399I The ISQL default profile values are in effect.
ARI7079I ISQL initialization complete.
ARI7080A Please enter an ISQL command or an SQL statement.
select * from wec.testtab
ARI7960I The query cost estimate for this SELECT statement is 1.
         COL1  COL2           COL3
 ------------  -----  ------------
            1  HI                2
 * End of Result *** 1 Rows Displayed ***Cost Estimate is 1*********************
exit
ARI7601I ISQL ended normally on your request.
ARI0660I Line-edit symbols restored:
         LINEND=# LINEDEL=¢ CHARDEL=@
         ESCAPE=" TABCHAR=“
Ready; T=0.03/0.04 16:43:13



                                                                  RUNNING   EVIEVM
```

Figure 24. SQL/DS on VM ISQL

Eventually, by the 1990s, SQL/DS was renamed to Db2 for VM and Db2 for VSE; they have little to no relation to Db2 for MVS or Db2 LUW ("Linux Unix Windows", which also ran on OS/2).

VM/SP Release 5 and 6: SP 5 was intended to add more features to SP 4, and several key bugfixes were included. Released in December 1986, it still contained many new derided features. While later versions of SP 5 were remarkably stable, the criticism of SP 5 pales in comparison to SP 6. Hitting the market on October 20, 1987, SP 6 introduced TSAF, a shared CMS filesystem, and a very unstable version of CMS. The new CMS SFS was certainly a great idea: you could now have directories... but if the SFS server VMs crashed, your data was as good as dead (this has happened to your author's VM systems)! TSAF was a cool feature as it allowed for speedy cross-system collections of a singular APPC/VM domain to be created, but it was not truly useful until the final releases of VM/ESA a decade later. CMS SP 5 and 6 were also quite derided: Cornell famously never put CMS SP 5 into production on account of major bugs; most sites stayed with CMS SP 4. Sadly, some customers tried to move to VM/XA SP, but it always ended in a disaster; VM/XA SP, while it had been made available, had hardly any features (and the ones it did have were either of sub-par quality in their implementation, or just missing altogether). VM/SP 6 did add VMSES/E, which made service much easier, but many diehard VM systems programmers derided it for various reasons.

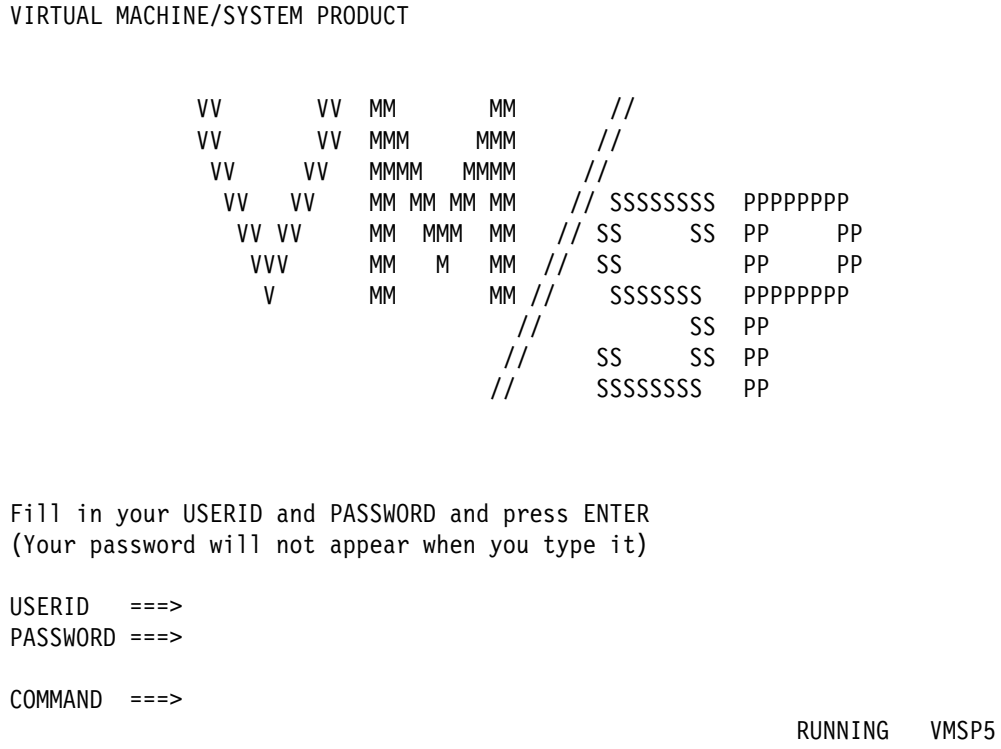


Figure 25. VM/SP Release 5 logon screen

VM/IS: Originally intended to be a turnkey package of VM/SP Release 5 or 6 and installed on an IBM 9370, VM/Integrated System was the "sysprogless VM." This was intended to actually be a massive success, but, in true IBM fashion in that era, it certainly wasn't. Instead, VM/IS was a bit of a flop; alas, it included the following products:

- SQL/DS
- DisplayWrite/370
- PROFS
- QMF
- VS BASIC
- GDDM
- RSCS
- ISPF
- FORTRAN
- TCP/IP
- VTAM

Alas, VM/IS was certainly a wonderful system. See the following screencaps for some examples of what the system could do:

```
 VIRTUAL MACHINE/SYSTEM PRODUCT


               ************************************************
               ***  ******  **   *   *******  *    ***     *****
               ***  ******  *  *   *  *****  ***  ***   **  ****
               ****  ****  **  ** **  ****  ****  ****   *******
               *****  **  ***  *****  ***  *****  ******   *****
               ******    ****  *****  **  ******  ***  **   ****
               *******  *****  *****  *  ******    ***     *****
               ************************************************

                     Virtual Machine  /  Integrated System



 Fill in your USERID and PASSWORD and press ENTER
 (Your password will not appear when you type it)

 USERID   ===>
 PASSWORD ===>

 COMMAND  ===>
                                                          RUNNING   VMIS
```

Figure 26. VM/IS logon screen (system built on VM/SP Release 5)

```
IBM BASIC/VM VERSION 1 RELEASE 2.0  1925/09/21  16:14
   (c) Copyright IBM Corporation 1982, 1985
 * load eliza
255 LINE(S) LOADED.  'ELIZA' IS THE WORKSPACE NAME.
 * run
HI!  I'M ELIZA.  WHAT'S YOUR PROBLEM?
 ? nothing much
ARE YOU SAYING NO JUST TO BE NEGATIVE?
 ? not really, I'm just being silly
YOU ARE BEING A BIT NEGATIVE.
 ? you are right...
SAY, DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?
 ? probably?
WHAT DOES THAT SUGGEST TO YOU?
 ? that you are fun to talk to!
I SEE.
 ? thank you for helping me demonstrate VM/IS!
I'M NOT SURE I UNDERSTAND YOU FULLY.
 ? that's fine!
COME COME ELUCIDATE YOUR THOUGHTS.
 ? okie!
CAN YOU ELABORATE ON THAT?
 ?
```

Figure 27. The original BASIC ELIZA on VS BASIC for VM

```
DWSAMPLE RFT      J1                                                    Page 1
                                                                        Line 0
===>
<---+_---1_---+_---2_---+_---3_--V+_---4_---+_---5_---+_---6_---+_>--7_---+_---
------------------------------------------------------------------------ Page 1 ----
To:  All Staff

From:  Planning Office

Subject:  Quarterly Business Meeting

Date:  September 23, 1985


This year the Corporet Division has scheduled the Autumn Quarterly
Meeting for Friday, October 23, in Wolverhampton, U.K.  So that we
might all prepare for this important meeting, the Planning Office
has compiled the following "mini-report" on one area considered
vital to the company's growth:  international development . spell
ing

We hope the report will help prepare you for some of the major
PF 1=HELP       2=Insert     3=END        4=Instr.     5=RFind      6=Aid
PF 7=Backward   8=Forward    9=Block     10=Command   11=Next      12=Cmdline
```

Figure 28. DisplayWrite/370 for VM

```
----------------------  ISPF/PDF PRIMARY OPTION PANEL  ------------------------
OPTION  ===>
                                                            USERID   - WEC
   0  ISPF PARMS  - Specify terminal and user parameters    TIME     - 21:20
   1  BROWSE      - Display source data or output listings  TERMINAL - 3278
   2  EDIT        - Create or change source data            PF KEYS  - 12
   3  UTILITIES   - Perform utility functions
   4  FOREGROUND  - Invoke language processors in foreground
   5  BATCH       - Submit to batch for language processing
   6  COMMAND     - Enter CMS command or EXEC
   7  DIALOG TEST - Perform dialog testing
   8  LM UTILITIES- Perform library management utility functions
   9  IBM PRODUCTS- Additional IBM program development products
   C  CHANGES     - Display summary of changes for this release
   T  TUTORIAL    - Display information about ISPF/PDF
   X  EXIT        - Terminate using console, log, and list defaults


Enter END command to terminate ISPF.

5684-123 (C) COPYRIGHT IBM CORP 1980, 1990.

 F1=HELP     F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
 F7=UP       F8=DOWN     F9=SWAP    F10=LEFT    F11=RIGHT   F12=CURSOR
```

Figure 29. ISPF on VM/IS

```
Licensed Materials - Property of IBM
5645-DB2 5648-A70 (C) Copyright IBM Corp. 1982, 1998
All Rights Reserved.
IBM is a registered trademark of International Business Machines

QMF HOME PANEL                          Query       Management     Facility
Version 6 Release 1
                                       ******     **    **       ********
Authorization ID                      **     **   ***   ***      **
 WEC                                 **     **   ****  ****     *******
                                    **     **   ** ** ** **    **
Connected to                       **   * **   **  ****  **   **
 SQLDBA                             ******    **   ***   **  **
                                         **

Enter a command on the command line or press a function key.
For help, press the Help function key or enter the command HELP.

1=Help        2=List        3=End         4=Show       5=Chart       6=Query
7=Retrieve    8=Edit Table  9=Form       10=Proc      11=Profile     12=Report
OK, you may enter a command.
COMMAND ===>
```

Figure 30. QMF, a UI for SQL/DS, on VM

VM/XA SP

As mentioned before, there were three different versions of "VM/XA" that were made available:

- VM/XA MA (Migration Aid): a stopgap measure to migrate MVS/SP systems to MVS/XA; announced 21 October 1981.
- VM/XA SF (System Facility): announced 12 February 1985, SF was an expanded version of the Migration Aid that gained support for new things, like:
  - Support for the 3090
  - Support for the 3800 laser printer
  - "Dedicated" support for the 3370 (FBA) and then-new 3380 (CKD) DASD
  - A subset of support for VM/SP's IUCV (which would be necessary for TCP/IP later)
  - Multiprocessor guest support
  - Support for Named Saved Systems, in which saved systems are stored in spool files (as opposed to hardcoded into DMKSNT at fixed locations)
  - CMS 3.1 (from VM/SP Release 3), though somewhat buggy, and only for "installation and maintenance"
- VM/XA SP (System Product), of which there were two versions: a version of VM/XA intended to match the functionality of VM/SP; announced 11 June 1987. Features included:
  - A bimodal CMS that supports 31-bit addressing modes
  - A port of VM/Real Time Monitor
  - National Language Support for CMS (a feature found in VM/SP)
  - Support for PROFS (which required a good and usable version of CMS)

– Support for a port of RACF, providing better security over just DIRMAINT by itself

Most of the "meat and potatoes" of VM/XA came from VM/XA SP, wherein IBM was trying to supplant the usage of VM/SP HPO on large machines (like the high-end 308x or 438x processors). Sadly, users found it to be rather lacking in many ways! For one, users derided the bimodal CMS (which hit the scene in March 1988) for being horribly unstable and lacking in support for a large amount of applications. Depending on the virtual machine's mode, you would get either a 370-mode or XA-mode CMS. It works like this:

```
set machine 370
System reset.
System = 370
ipl cms
DMSACP723I Y (19E) R/O
DMSINS327I The multitasking saved segment VMMTLIB could not be loaded
VM/ESA REL. 2.2 01/21/23 10:22

DMSDCS3993E PIPES saved segment can not be loaded beyond 16M
DMSDCS3993E VMLIB saved segment can not be loaded beyond 16M
DMSWSP100W Shared S-STAT not available
DASD 0222 LINKED R/O; R/W BY OPERATOR
DMSACP723I C (222) R/O
Ready; T=0.10/0.12 10:03:53
set machine xa
System reset.
System = XA
IPL CMS
DMSACP723I Y (19E) R/O
VM/ESA REL. 2.2 01/21/23 10:22

DMSWSP100W Shared S-STAT not available
DASD 0222 LINKED R/O; R/W BY OPERATOR
DMSACP723I C (222) R/O
Ready; T=0.05/0.06 10:04:53
```

Figure 31. PROFS V2 R1.1 Main Menu

Okay, I know this is a VM/ESA 1.2 system, but it is analogous to VM/XA Version 2.

VM/XA ultimately was somewhat of a flop; it never had the substantial amount of effort being poured into it in the same way that VM/SP and VM/SP HPO had. VM/XA never had the same amount of supported features, users, or documentation as its internal competition. One of the biggest issues with it was simply how different its CMS was; some users of VM/XA SP actually built and ran old versions of CMS from VM/SP (like Release 3 or Release 4) on VM/XA as 370-mode virtual machines. While this also meant that there was no way to take advantage of the 31-bit CMS/XA address space, one must also ask if any such applications ever even existed. For timesharing installations (like universities) running VM/XA, this solution was perfectly fine. Also, since VM/XA's CMS was so riddled with OCO modules, any third-party programs that used CMS nucleus extensions (which large VM programs did) would be useless on VM/XA. Ultimately, the major flop that was VM/XA led IBM to seek a suitable replacement... and it came in the form of...

VM/ESA

VM/ESA was seen as the grand unification of the VM/SP and VM/XA line. When VM/ESA was announced on September 5, 1990, they had a tall order: replace VM/SP, VM/SPA HPO, and VM/XA SP. Now, note that VM/SP and VM/XA were, in no uncertain term, two different systems; the VM/XA version of CP had diverged so much, its module prefix was changed from DMK (for the 370 CP) to HCP. VM/ESA would have to encompass two different types of processors: the System/370 machines (of which there were still certainly some operating in that era) and the System/390 machines (with VM/ESA being the new premier version of VM that would run on those). As such, there were several versions that would end up being announced:

- VM/ESA V1R1 370 Feature (September 28, 1990): the successor to VM/SP Release 6
- VM/ESA V1R1 ESA Feature (March 29, 1991): the successor to VM/XA SP Version 2
- VM/ESA V1R1.1 ESA Feature (December 27, 1991)

Parallel Worlds: The goals were clear: unify the flailing VM line, and make VM a system for the 1990s. Of course, this too would be a tall order -- VM had, at that point, come off of a rather bad decade of questionable decision after questionable decision, and this version would also introduce new features. One of those was VM Data Spaces, which you may recall as being a new feature of MVS/ESA systems (if you are reading this paper in linear order). VM Data Spaces were essentially the same thing, except a new virtual machine operating mode (to complement 370 mode, XA mode, and ESA mode) was added: XC mode. A certain DIAGNOSE function could be used (X'248') to copy data to and from a Data Space into that VM's main address space. This would end up being a useful feature for filepool servers. Alas, the 370 Feature wasn't totally ignored: all of the VM/SP HPO modifications were added onto 370 Feature (the main ones being the 64 MB real storage limit and the spool file constraint removal).

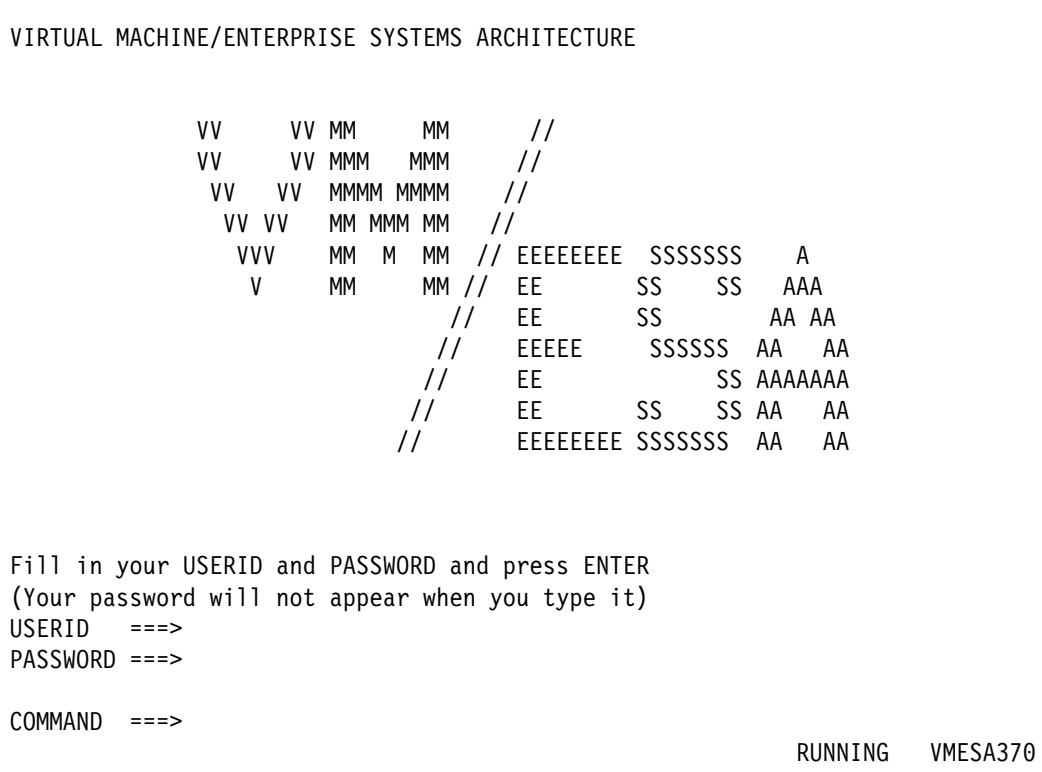


Figure 32. VM/ESA V1R1 370 Feature logon screen

Because there were now two concurrent versions of VM riding together at the same time, much effort had to be put into preserving parity between the 370 Feature and the ESA Feature. One of those was the retroactive reintroduction of a feature similar to VM/XA that would lower the amount of times the system programmer would have to gen-

erate and IPL a new CP nucleus on account of configuration changes. The following tasks would often necessitate an IPL on older versions of VM/SP to change:

- The list of saved segments (DMKSNT)
- The logon screen logo and printer separator logo (DMKBOX)
- The system ID (DMKSYS)
- 3705 network control programs (when not under the control of VTAM)
- 3800 printer image libraries
- National language support libraries

IBM's solution to all of this was something called an override file. This was essentially a cut-down version of those original assembler source files, and they mirrored the functionality of the parm disk mechanism of VM/XA (wherein system configuration files were stored as plain CMS files within special extents of the system DASD space called parm disks). Sadly, it was often seen as a rather bad solution to a problem that could be alleviated by implementing a parm disk style mechanism for the aging VM/SP nucleus... alas, this never happened.

The ESA Feature also introduced other things that are seemingly forgotten, such as:

- *Support for FBA DASD.* Believe it or not, VM/XA did not support FBA DASDs! VM/ESA 1.1 introduced support for these (note that VM/ESA 1.0 370 Feature had support from the start).
- *Guest encryption support.* MVS/ESA SP 4, when running under ESA Feature, could now use the cryptographic processors on the host in a guest.
- *DFSMS/VM.* Also called System Managed Storage, DFSMS/VM was intended to be a port of the MVS feature of the same name, but it was ultimately a flop.
- *A bimodal CMS* This would be the replacement to VM/XA's terrible bimodal CMS; the version found on the ESA Feature was actually reasonable.

VM/ESA Version 2: Eventually, Version 1 would come to an end. The 370 Feature would be discontinued after Version 1.5, and Version 2 would take over with ESA support only. This version represented IBM essentially flailing around in an effort to try to find something that would make the System/390 mainframe line attractive to new customers, but, by the mid-1990s, nobody was actively becoming a new mainframe customer (much in the same way as today). Alas, a lot of new features came out of the dark days of VM in the mid-90s; VM/ESA V2R1 was announced on June 12, 1995. New features included:

- *OpenEdition for VM/ESA*, more about this below
- *16MB line constraint relief* for CMS and GCS virtual machines
- *GMS GUI*, an API for CMS that allowed the creation of workstation-based GUIs (very similar to the ISPF Workstation Agent in the MVS world)
- *RSCS with TCP/IP support*, where RSCS could now natively speak NJE over TCP/IP
- *VSE/VSAM Compression Support*, allowing for improved VSAM support on CMS
- *Distributed Computing Environment for VM*, a UNIX-world communications and RPC system that was popular in the era
- *Dynamic CP Exit Facility*, wherein CP can gain new features without a re-IPL
- *LAN File Services/ESA*, a combined NFS and SMB server for VM that used an OS/2 server attached to a mainframe as a communications processor

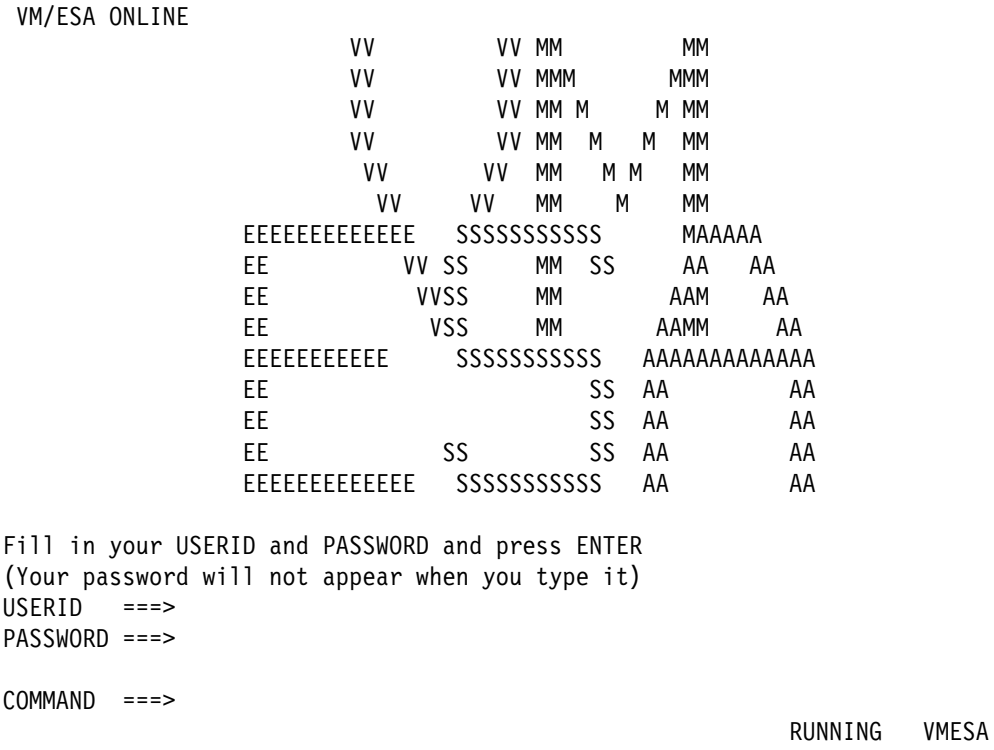


Figure 33. VM/ESA Version 2 Release 1 logon screen

I shall go in backwards order on this, starting with LFS/ESA. Sadly, LFS was essentially obsolete the moment it hit; the then-current System/390 Parallel Enterprise Servers of that year had shelved their old IBM 3172 LAN Channel Station boxes for the Open Systems Adapter cards, and those absolutely blew the 3172 away in LAN performance. LFS, requiring a dedicated communications processor, was almost completely forgotten about.

Likewise, the CMS GUI Feature was also something else that was questioned with regards to its necessity. Sure, the GUI client applet ran on OS/2, Windows, and AIX... but who needed this? This was part of the "mainframe GUI craze" of the mid-90s (see LANRES for more examples of this), but that didn't mean that anyone needed it or wanted it. Eventually, in the days of z/VM, it was finally discontinued.

OpenEdition, however, was a very interesting feature. Providing a reasonably POSIX-compliant environment for VM, clever people (mainly Neale Ferguson) ported large amounts of open-source software to VM. OpenEdition actually provided a port of the Java JVM and JDK; while this was not often used, it was certainly an impressive feat. Likewise, thanks to OpenVM, you could now run things like Apache, Samba, LDAP services, InterNetNews (an NNTP/Usenet server package), an IRC server, and more... all from CMS! OpenEdition even provided a form of POSIX threads, but the fork/exec method ultimately worked better for most usecases.

The last version of VM/ESA was in 1999 with VM/ESA V2R4; this famous version represented a big comeback on IBM's behalf. VM/ESA Version 2 had started out rough, but it had ironed out by the turn of the century. With the new System Z machines, it was time for...

z/VM

z/VM 3.x: The first version of z/VM was V3R1 (i.e. 3.1), intended to continue the version numbering scheme set by VM/ESA Version 1. z/VM 3.1 was intended to be a bit of a stopgap port, but it certainly added some key new things:

- 64-bit support for hosts and guests (with retaining what VM called '32-bit' support for System/390 machines)

- ESS (Shark) DASD array native Fast Copy support
- TCP/IP upgrades:
  - Initial version of SSL for z/VM
  - OpenVM NFS client
  - FTP server upgrades to make the server browsable from web browsers
  - IP multicasting support
  - QDIO OSA-Express2 support (adding support for Gigabit Ethernet and 155 megabit ATM)
- DFSMS/MVS compatibility

z/VM 3.1 would later get re-released on February 20, 2001, with more features (or, as the IBM program announcement calls it, "enhancements"). This version is when z/VM lost the old ROUTED server and gained the replacement MPROUTE server (that would handle OSPF and RIP routing, as opposed to ROUTED, which only did RIP).

z/VM 4.x: z/VM 4.1 was announced on May 29, 2001, and z/VM 4.4 (the last of the 4.x lineage) was announced on May 13, 2003. Of interest is the 32-bit nucleus; all versions of z/VM up till 4.4 had a 32-bit nucleus that you could use (a fun fact for the clever reader: I actually composed this paper on a z/VM 4.4 system running on a 32-bit System/390). z/VM 4.2 introduced something called a Guest LAN, wherein a feature that allowed you to create a virtual network within CP was added (this would be later expounded upon with z/VM version 4.4, where it was transfigured into a true virtual Ethernet switch feature that no longer required a router served by the TCP/IP service virtual machine).

Something interested added to z/VM around this time was something called an *EDEVICE*. When the System Z machines gained FICON interfaces, something they quickly gained was support for a FICON controller running in native Fibre Channel mode. These so-called zFCP cards allowed native SCSI devices to be mounted on the mainframe... albeit at the requirement of having OS support for this; z/VM can emulate FBA DASDs on these zFCP interfaces.

```
z/VM ONLINE


                 / VV           VVV MM       MM
                /  VV         VVV  MMM      MMM
 ZZZZZZ        /   VV       VVV   MMMM    MMMM
    ZZ        /    VV     VVV     MM MM MM MM
   ZZ        /     VV  VVV       MM  MMM  MM
  ZZ        /      VVVVV        MM   M   MM
 ZZ        /       VVV         MM       MM
ZZZZZZ    /        V           MM       MM

  built on IBM Virtualization Technology


Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)
USERID   ===>
PASSWORD ===>

COMMAND  ===>
                                                      RUNNING   EVIEVM
```

Figure 34. z/VM Version 4 Release 4 logon screen

z/VM 5.x: An interesting feature most mainframe hobbyists have forgotten about is something called the Integrated Console Controller. In the "olden days", you had very few options for providing a 3270 console to a mainframe. If you had a Multiprise 3000, great! If not, you would need either a 3174 with coax terminals or an IBM 2074 console controller that would emulate a local terminal controller with TN3270 sockets. The z890/z990 (as mentioned above) gained this, and z/VM 5.1 gained support for controlling those.

z/VM 6.x: On July 22, 2010, IBM announced z/VM 6.1. The 6.x line was mainly intended to be a stabilization feature update set, as z/VM was beginning to find itself in large deployments running many Linux guests. These fixes would continue until its replacement (7.x) came out. The next version after this (7.x) was the last real release of z/VM; subsequent releases were made into a "rolling release" model.

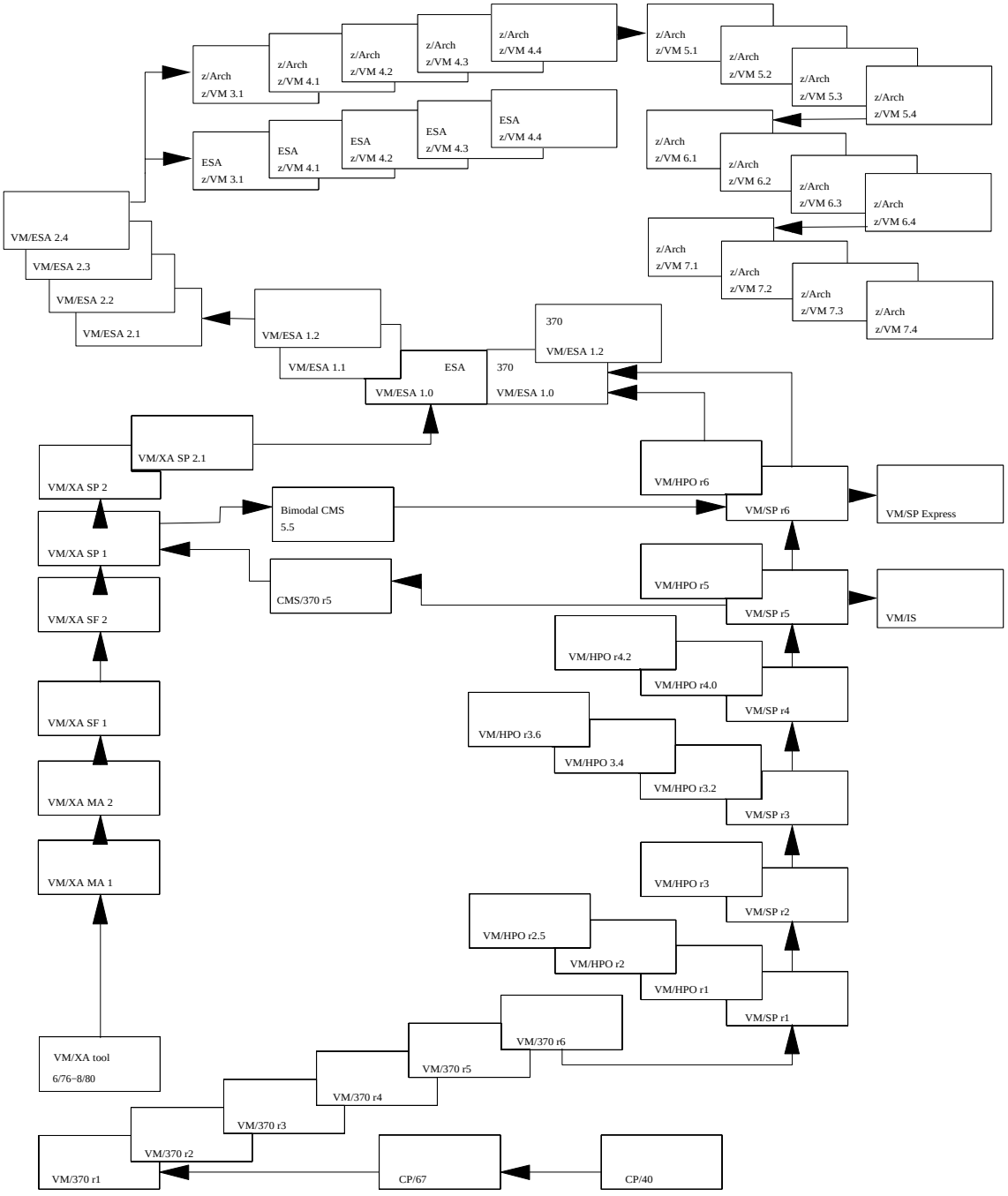


Figure 35. z/VM History -- zoom in!

Mainframe Linux

In the late 1990s, there was a strange gap of a good UNIX system for mainframes. Sure, there was AIX/ESA and UTS, but they were expensive, hard to run, and ultimately were commercial failures (compared to the MVS UNIX environment). In 1998, Linas Vepstas sought to change that. His project became to be known as the Bigfoot project, and IBM started a competing project around that time that would be known as "Linux S/390".

Bigfoot Linux/i370

Linas's port was based on a number of things. First, he had to either produce of acquire a modified version of EGCS (recall the famous "forking" of GCC into EGCS, and later re-merging); at this point, there existed a modified version of GCC and Binutils (the GNU assembler and linker) maintained by David Pitts that could assemble HLASM-syntax assembler files. This compiler toolchain ran on MVS's UNIX emulator (in this era, it was OS/390 UNIX System Services), but it did not find itself being used for Linas's project.

Linas took a *different*version of Binutils and GCC. Going off Binutils 2.9.5 and GCC 2.8.1 (by the end of the project; mind you, he started with Binutils 2.9.1 and EGCS 1.1.1). With this, he was able to start a hasty port of Linux to the i370 CPU architecture (as that compiler toolchain, and David Pitts's, knew it as). This port would target 370/XA machines, and would load directly from memory.

Now, it is necessary to understand a new feature of the System/390s around the time this project was being worked on. The Support Element gained a new feature to "Load from CD-ROM or Server", wherein a specific file would contain a memory map and a list of files to populate the system storage (at the locations specified in the memory map) with. These files bore the .INS extension, and the Bigfoot/i370 INS file looked like this:

```
* INS file to load Linux kernel
* Format: <file to load> [address where to load]
* The elf-stripped Linux kernel
vmlinux.bin  0x00000000

* Boot command line
cmd_line  0x000d9000

* Load RAMDISK at the same address as given in command line
disk-image  0x00800000
```

Figure 36. Linux/i370 .INS load file

A common feature seen on Linux kernels is the concept of the kernel command line. This provides parameters for the kernel that it needs either during or after booting, such as the root filesystem to use. The kernel command line for this system would have been placed immediately after the kernel's end in the memory address space, and it looks like this:

```
root=/dev/ram init=/bin/sh -i i370_initrd=0x800000,4096
```

Figure 37. Linux/i370 parameter/command line file

The ramdisk would contain the root filesystem, as i370 Linux never gained drivers for anything (except for a 3215 console). This would be an exactly 4-megabyte (the 4096 at the end of the command line specifies this) disk image formatted with the Linux ext2 filesystem. Upon loading, this file would be mounted by the kernel's virtual filesystem layer and ext2 driver as the root; it would also contain the initial program to run, which in the case of the example I shall demonstrate, is the shell.

When you wished to load the system, you would instruct the Support Element to do an IPL from a CD-ROM (which could also be a path on the SE's filesystem) or a "server" (an FTP server). Since there were no drivers for anything (including network devices), the system is as good as useless. Nonetheless, here is a sample load and run of i370 Linux:

```
Linux version 2.2.1 (root@linux-arm64vm) (gcc version 3.4.6) #1
Sun Sep 14 02:09:53 PM CDT 2025
Boot command line: root=/dev/ram init=/bin/sh
-i i370_initrd=0x800000,4096

Device 0009 CU ID 3215  Model 00  ready
Device 0009 mapped to unix /dev/console (227, 1)
Device 000D found CU ID 3505  Model 01  ready
Device 000E found CU ID 0000  Model 00  ready
Device 0200 found CU ID 3215  Model 00  not ready
Device 0200 mapped to unix /dev/3270/raw0 (227, 128)
Device 0201 found CU ID 3215  Model 00  not ready
Device 0201 mapped to unix /dev/3270/raw1 (227, 129)
Device 0202 found CU ID 3215  Model 00  not ready
Device 0202 mapped to unix /dev/3270/raw2 (227, 130)
Device 0203 found CU ID 3215  Model 00  not ready
Device 0203 mapped to unix /dev/3270/raw3 (227, 131)
Calibrating delay loop... 465.31 BogoMIPS
Memory: 59640k available (748k code, 940k data, 48k init)
Init Ramdisk: 4096k  [00800000,00c00000]
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.2
Based upon Swansea University Computer Society NET3.039
NET4: Unix domain sockets 1.0 for Linux NET4.0.
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
Starting kswapd v 1.1.1.1
pty: 256 Unix98 ptys configured
RAM disk driver initialized:  16 RAM disks of 262144K size
loop: registered device at major 7
i370_setup_devices /dev/console (c 227 1)
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 4096 blocks [1 disk] into ram disk...
VFS: Mounted root (ext2 filesystem).
freed initmem from 0010e000 to 0011a000  (12 pages total)
raw3215_open of /dev/console (c 227 1)
can't access tty; job control turned off

/ # uname -a
Linux (none) 2.2.1 #1 Sun Sep 14 02:09:53 PM CDT 2025 i370 GNU/Linux
i370_do_signal sh/1 signr=17 handler=8187ca0c
i370_sys_sigreturn: regs=7f800768
/ # mount -a
i370_do_signal sh/1 signr=17 handler=8187ca0c
i370_sys_sigreturn: regs=7f800768
/ # mount
/dev/root on / type ext2 (rw)
/proc on /proc type proc (rw)
i370_do_signal sh/1 signr=17 handler=8187ca0c
i370_sys_sigreturn: regs=7f800768
/
```

Figure 38. A rare view of a running i370 Linux system

Ultimately, there would be IBM's own Linux-for-S/390 port that would end up totally replacing Bigfoot/i370; Linas's crew was essentially powerless to beat IBM's crew on their own front... and the i370 Linux port dies out.

Well, mostly. In September 2024, Linas actually dusted off the old source code for the i370 port, made it work on Hercules (which was now mature, as opposed to how it was in the original i370 era), and published the source code on his website; this effort is how the earlier listing of a i370 kernel starting was taken.

IBM Linux/390

In the late 90s, IBM partnered with Marist College to attempt to do the same thing (in parallel) as Linas was spearheading. IBM's effort focused on newer machines (those being the 9672 G2-and-newer), whereas i370 focused on older machines. Actually, there were several differences between s390 (as it became to be known, "s390" being the name of the directory under the Linux kernel source tree that held the machine-specific code) and i370:

- The i370 assembler could process HLASM-style code, whereas the s390 assembler was only capable of understanding the UNIX assembler syntax.
- The s390 port required a 9672-G2 or newer, whereas the i370 port could run on any 370/XA-or-newer machine.
- The s390 project was spearheaded entirely by IBM, whereas i370 was a "vendor-neutral" project.

The Marist folks worked on Linux/390 intensely, and by 2000, a usable distribution had been produced. This came in the form of a modified version of Red Hat that became to be known as "Marist Linux". This OS could be started in three ways:

- From a tape, which could be produced from MVS, VM, or VSE
- From the VM punched card reader (this method is still used today)
- From an INS file when installing without VM, just like the i370 port (this method is also still used today)

Installation was relatively simple; you would load the kernel (which itself was followed by the kernel command line text and then the ramdisk, just like i370), configure a network device, format a DASD, then restore the root filesystem. After that, re-IPL and you were off to the races!

By 2001, the System Z was in full swing and s390 Linux was quickly ported over to it; this came in the form of a modified Red Hat (as usual) called Think Blue Linux. This ended up being discontinued in favor of other "mainline" Linux images around that time; Debian quickly gained a s390 distribution on their list, as did CentOS, SUSE Enterprise Linux, and more. Today, s390x (the "x" being the 64-bit port of s390 Linux to the System Z) Linux's install base is nearly entirely Ubuntu Server, SUSE Enterprise Linux, Red Hat Enterprise Linux, and Debian. More and more drivers are on the system today now than ever, and one can even attach PCIe cards to the System Z PCIe card cages (via adapters, of course)!

Mainframe UNIXes

In the early 80s, there were two competing efforts to get UNIX ported to the System/370. These were:

- Princeton University UNIX/370: a native port of UNIX v6 that evolved into Amdahl UTS
- AT&T Bell Labs UNIX/370: a hacked version of TSS/370, postdating Princeton but being intended for much larger systems

Let us examine these two interesting systems, then examine their successors. I have taken the liberty of discussing these in reverse order, as the fruits of Princeton's labor came only after Bell Labs's endeavor.

Bell Labs UNIX/370

Bell Labs's attempt was certainly more interesting than Princeton's. For one, they did not directly port UNIX to run on the S/370 CPU; instead, they took TSS/370 as a base, and replaced the user process supervisor with one that functioned as a UNIX system. While this may seem rather odd and inane, this was actually done for good reasons: first, the I/O model of the S/370 is radically different from the PDP-11 system UNIX was originally written to run on. Because of this, the Bell Labs engineers instead chose to do the TSS hack; this also provided an unexpected benefit: the IBM field service personnel could run their native problem analysis tools to determine what could be going wrong with a S/370, as any hardware error reports (from the EREP program) would be in a standard format that did not necessitate the writing of UNIX-native code to emulate this format. Put simply, they took the easiest way out.

The Purpose: One must also understand why this project was started in the first place: Bell Labs was the progenitor of some of the telephone switches used throughout North America. One of those was the 5ESS (ESS meaning Electronic Switching System), a combination analog and ISDN phone switch. In 1978, Bell Labs started to seriously consider bringing UNIX over to a larger computer than what they were using; the 5ESS development needs were so large relative to the comparatively small machines the engineers had to work with that their patience eventually expired and the project began. The decision to use UNIX for the 5ESS software development actually predated the formal start of the project in 1980. By late 1980, the programmers writing the 5ESS software were using a whopping 9 PDP-11/70s. UNIX was selected in the first place because the programmers that would be working on the 5ESS project were already familiar with tools found on PWB/UNIX, which we now today know as:

- make, the program generation and maintenance utility
- m4, the macro processor
- awk, the text programming language
- sed, the text stream editor
- yacc, Yet Another Compiler Compiler
- lex, a lexer generator
- an expanded editor, predating vi
- SCCS, the UNIX source code control management system

The development crew having to use PDP-11s found themselves in a rather terrible position. Their 9 PDP-11s meant fragmented development, but the machines were connected together "using a commercially available high-speed network with drivers written for the UNIX operating system." Alas, they were not using a centralized source code management system (like Git today); when they were ready to build a firmware image for the prototype 5ESS they had, they would have to send all of the objects over to one of the PDP-11s for linking (that same machine would coalesce all of the object files from the other systems). Since the 5ESS project began to grow, it was

starting to no longer become feasible to continue to use the approach they were using; around this time, UNIX/370 caught the eye of the switch engineers.

Since the PDP-11s were obsolete performance-wise and cumbersome to conduct as an entire network, the search for a replacement began. The engineers looked at a VAX 11/780, but it was only twice as fast as the PDP-11. This meant they could halve their PDP-11s, but the VAX systems were also a lot larger and more power-hungry. As such, someone considered getting a real mainframe. An IBM 3033 AP System/370 was 15 times faster than the PDP-11s, and, though it was larger than a room full of VAX-11s, it was much easier to work with (as it was a single system, not a disjoint cluster in an era before UNIX had good network filesystems and remote management tooling). The engineers working on UNIX/370 dusted off their terminals and got back to work.

As mentioned before, the radically foreign I/O architecture of the S/370 initially scared some of the engineers. UNIX had several roadblocks in its original implementation that made the project even more difficult on top of that:

- The System/370s were multiprocessing machines (if equipped), UNIX did not support multiprocessor machines at all.
- The System/370 memory model was that of paging, not swapping like it was on PDP-11 UNIX.
- The customer engineers from IBM expected the aforementioned EREP-format error logs, UNIX would have to gain code for this.
- The I/O devices on most System/370 machines could be multipathed, UNIX didn't even support channelized I/O.

The Hacked TSS: Needless to say, this would be daunting task. As mentioned, the engineers realized they could skip having to write a lot of new code by adapting TSS/370. Primary sources mention that choosing TSS/370 as the system to base UNIX/370 off of was, at best, quite controversial. Nobody knew much about the system as it was rather poorly documented outside of IBM, and its install base (especially after the announcement of VM/370 being a fully supported system) consisted of so few systems you could count them on your fingers. In 1979, Bell Labs talked to IBM about modifying TSS to support the structure the rest of the UNIX/370 system expected. IBM followed through with this, and, under a program license agreement, provided Bell Labs with the modified TSS/370 system in 1980.

This new TSS was surprisingly structured. There were 3 levels of programs that the OS would run, consisting of userland programs (like shells, editors, compilers, and other such tools), the UNIX System Supervisor (adapting TSS to a C-style UNIX kernel interface, this was essentially a userland port of the UNIX kernel), and the Resident Supervisor (the TSS core that provided multitasking, hardware management, system control functions, and such). Every UNIX process ran within its own 16-megabyte virtual memory address space (for the first time in UNIX history), and the Resident Supervisor provided services to the virtual address spaces (like scheduling the execution of the address spaces, dispatching the execution queue, managing real storage, etc).

Memory Model: Strangely enough, this memory management does not work like you think it does if you are used to the VAX virtual memory UNIX model. Each UNIX system process ran in its own containerized address space *along with a copy of the UNIX System Supervisor*. A UNIX system process is essentially a virtual UNIX machine; multiple user programs could run within the address space. The UNIX System Supervisor was mapped into the high 8 megabytes of the virtual memory address space, and user programs resided in the lower 8 megabytes. The bottom page was reserved for interrupt vectors (in true S/370 fashion). Since the S/370 memory management unit allowed for pages to be shared between processes (which, in this context, a process running under TSS would be an entire virtual UNIX machine); this allowed several identical running copies of various UNIX programs across several UNIX system processes to be shared. Furthermore, the UNIX System Supervisor could be shared amongst the system processes, alleviating excess memory usage.

Due to this design, there were 2 layers of system calls present in this strange OS. User programs could make UNIX system calls, which would be trapped by the UNIX System Supervisor. If the system call was not related to anything that required hardware control, the UNIX System Supervisor could dispatch and execute that system call on

its own. If it needed to call the Resident Supervisor (remember, this is the modified TSS/370 nucleus), it could do so. Interestingly, all of the UNIX System Supervisor to Resident Supervisor system calls were processed fully asynchronously; when the system call execution was done and it needed to return data to the caller, it would actually trigger an interrupt to the virtual address space. As you can probably tell by now, the "UNIX address spaces" were absolutely "UNIX virtual machines", and there was a strong CP/CMS influence on this system! Likewise, if the UNIX System Supervisor needed to interrupt a user process, it would do so through a signal (in true UNIX fashion). The system could even utilize the S/370 Virtual Machine Assist feature, boosting performance even further by alleviating an unnecessary layer of system call return indirection.

The S/370, as mentioned, also featured a paging memory architecture. This meant the UNIX system address spaces could be fully preempted and paged out of memory whenever there was a storage constraint (or just inactivity, like an idling shell that's been logged in for a day with no commands being ran). The Resident Supervisor handled all of the virtual memory operations, and the UNIX System Supervisors running under it would not have to do any of their own memory management outside of handling the brk() system call that grew a user process's address space. To alleviate memory pressure even more, the Resident Supervisor could also asynchronously perform page migration -- this was a feature wherein paged-out processes stored on fast non-RAM storage (devices such as fixed-head disks, drum memory units, or "solid-state memory") could be moved over to slower storage (like normal moving-head DASD disks).

Device Model: The disk and I/O subsystem was also rather different from PDP-11 UNIX. For one, the engineers chose to use the normal UNIX filesystem of the era... albeit with an enlarged block size (up to 4096, which matches the S/370 page size). The I/O subsystem also cached as much as it could; on a 16-megabyte S/370, 4 megabytes of that address space was allocated for disk I/O buffering. Note that this caching and filesystem I/O processing is done by the UNIX System Supervisor and not by the Resident Supervisor; this improved performance quite a bit, and especially moreso if multiple paging layers were used. Speaking of which, the Resident Supervisor provides underlying system calls that permit disk access, albeit in a rather unusual way. Rather than executing disk I/O operations synchronously, the block I/O layer (in the RS, not the USS) functioned through memory-mapped file buffers. For example, to read a file, the USS will allocate a buffer, then signal to the RS to read in the data and deposit it into that buffer. Likewise, for writing, there is outbound caching; whenever a write occurs, the RS will claim that the write was successful, but it really just waits in a flush queue (this is the same as the modern-day disk caching layers found on modern OS kernels).

Process management within the USS regions was also rather clever. In this era, UNIX did not have threads; spawning new processes with fork and exec was essentially the only option. In the modern day, the UNIX fork call employs a copy-on-write technique; that is, only copy the memory portions that you absolutely need (and do it one page at a time to save on space and time). However, UNIX/370 didn't do partial CoW like one might find on a modern kernel; instead, it would spawn off an asynchronous copy job and the program would be usable during the copy through page sharing (of which the shared pages had a rather clever name: multiplexed pages). UNIX/370 also had design issues related to process synchronization that needed to be worked out. Consider the case of being ran under a VM: at any point, while your process is running within a VM, your execution may be stalled or preempted at any time. This is also true with UNIX/370: the USS address spaces could be stalled at any time by the RS for various reasons (page faults, storage exhaustion, disk I/O). Since the S/370 machine UNIX/370 targeted was also a multiprocessor system, one would also have to figure out how to synchronize user programs across several physical CPUs. To address this, the engineers used the Dijkstra Semaphore mechanism in which resource accesses were counted and serialized between multiple CPUs.

Terminal handling was also somewhat difficult. On the S/370, the usage of plain ASCII terminals was rare; this is in direct contrast to what UNIX expected, wherein full-duplex immediate data transfer was often necessary to support the usage of nearly all programs (from the shell to the editor). This may not be an issue on small systems, wherein the CPU is interrupted with every keypress since it needs to echo the character back to the terminal, but this becomes very interrupt-heavy with many logged-on users. The AT&T 3B20 machines (which, in the era in which UNIX/370 was written, the largest system that ran UNIX) had front-end I/O processors that alleviated the interrupt load on the host CPU, but those were tailored to machines foreign to the S/370. IBM, of course, manufactured 3705s (replacing 2703s) and 3274s; neither of these were suitable for large amounts of directly-controlled

ASCII terminals. Note that ASCII terminals could be connected to a 2703/3705, but the OS on the host S/370 processor did not have direct character control over them (a good example to see this would be noticing the distinct lack of fullscreen programs usable on ASCII terminals connected to a VM/CMS system).

To alleviate this terminal issue, the engineers found a rather clever solution to replace the terminal controllers that would be of no use to UNIX: use an IBM Series/1 as a front-end processor. Bell Labs asked IBM for help once again to rig up a Series/1 with a S/370 channel interface as a terminal controller; IBM delivered this code and hardware in late 1980 to great success. Later, the engineers realized they could use a full AT&T 3B20S as a front-end processor too; this used conventional UNIX as the OS on it (rather than some strange, to the engineers, Series/1 OS). Seeing that they had figured out how to run multiple different computers as terminal controller front-end processors, some postulated that it would be possible to have frequently-used typing-heavy programs (like fullscreen editors and shells) run directly on the front-end processor; this never matured, but this idea was seen implemented on other non-IBM computers.

Deployment: In order to deploy this great system (the original 3033AP), Bell Labs realized an ideal migration would consist of a gradual turnover to the mainframe. The networking code the 9 5ESS development PDP-11s used was ported over to the 370, and the user migration was done with 10% of the userbase moved over to the 370 every other weekend. The initial deployment had an 80% reliability, but this was upped to 95% and eventually 98% within 6 months of the original deployment. Once Bell Labs realized they could now run *multiple* mainframes in a UNIX/370 network, they did so; they had a tri-system network of a 3033AP, 3033UP, and 3081K system (in 1983). Near the end of their project, they added a 4341 into the fray; this machine was just as functional as the other systems. Eventually, UNIX/370 would come to an end, but the important lessons learned from it would go on to shape UNIX for the better years down the line.

Princeton UNIX/370

Computing at Princeton has a very colorful history, and they played pivotal roles in the community aspect of IBM mainframes. However, our story takes us to the mid-1970s, when two mainframes were installed in Princeton's campus. There was a System/370 Model 158 that ran APL timesharing for the State of New Jersey's educational system; there was also a System/360 Model 91 monster that only ran batch jobs. Since so few people had access to the APL system, some departments sought computers of their own. One of those was the Electrical Engineering and Computer Science department, who chose to purchase a PDP-11/45. It originally ran what I believe was RSTS, but it was converted over to UNIX by 1975. Many EECS students and staff took to the system to great success, but trouble was on the horizon. In late 1975, New Jersey decided they would end the relationship in which the 370/158 would be taken away. The EECS UNIX users then heavily petitioned the Computing Center staff to consider buying a small PDP-11/70 to timeshare with UNIX. The original plan that Princeton had was to buy a System/370 Model 145 (because that was all that the university could afford); a PDP-11/70 would be even cheaper, and save the money quite a large chunk of money.

IBM, never to be outdone, decided they would try to undercut the DEC competition and sold Princeton a tremendously discounted 370/158... which just so happened to be the 370/158 that was already sitting in the building. In that era, a lot of mainframes were actually *rented* from IBM, and the State of New Jersey was doing exactly that. When they stopped paying for the APL timesharing, Princeton quickly installed VM/370 on the system, and so began Princeton's famous love of VM. Sadly, the EECS UNIX users were rather disappointed by the decision, but not all hope was lost. Tom Lyon jokingly proposed that, well, since the System/370 was running a virtual machine monitor, someone could port UNIX (which was written in a high-level language, only requiring the porting of a compiler) to run under it!

His older classmates told him it was a terrible idea, and the idea was gone as soon as it left his mind. Surprisingly, not all hope was lost, though; in the summer of 1975, Eric Schmidt (the former Google CEO and philanthropist) took an internship at Bell Labs. There, he helped write the UNIX lex program. He discovered that Bell Labs had a C compiler for the System/360 that cross-generated object code from a PDP-11, but it was unclear who wrote it;

Eric figured that Mike Lesk (who he was working with to write lex) was responsible for its creation. Seeing that the tools were sitting there, Eric remembered Tom's seemingly off-the-cuff non-serious remark.

The Initial Port: Saying yes to Eric's request to assemble a Princeton team to port UNIX to the 370, the process began. However, there were many issues to overcome beforehand. Of course, the EECS folks had their PDP-11/45 (complete with the UNIX source code too, as one would frequently have in that era)... but it was far from the 370/158 and totally disconnected. Even without a network link, issues were all-too-present; the PDP-11 did not have a 9-track tape drive, and the 370 of course did not have a DECtape drive. To overcome this, Tom writes that a rather crazy workaround involving many hops was found:

- The EECS PDP-11/45 was connected to a PDP-8 in the same EECS building, but it did not have very good reliability
- An excessively long serial cable that ran from the PDP-8 in the B wing of the E-Quad to a terminal room in a totally different wing (the E wing) of the E-Quad; this cable was so long and ran through two too many buildings, and was plagued by lots of line noise
- A multiplexed line-of-sight infrared optical link that connected the terminal room to the computer center located across the street at 87 Prospect Ave; this link provided a 9600 baud connection, but it broke during rain or foggy weather (this device was manufactured by Tran Telecommunications)
- The 370/158 mainframe in the basement of the Computing Center, wherein the multiplexed optical link was attached to an IBM 2703 communications processor
- VM/370's RSCS program, which would receive files over one of the serial links (which was ultimately tied to the serial port on the PDP-8)
- A test virtual machine, which would receive the virtual card decks from RSCS

Needless to say, this complex scheme was less-than-ideal. Nonetheless, the C compiler (which ran on the development PDP-11) did not produce object code, only Assembler XF input listings. This meant that the C compiler's assembler output would have to be transmitted to VM, assembled on VM, linked, and finally could be tested. Apparently, the linker was quite a headache to the UNIX guys, furthering the difficulty. By 1977, Tom had been working rather solo (since two of the other guys he had on his UNIX team had graduated); he got the kernel to work reasonably well, ported over the filesystem (which required disk access to be figured out), got a shell to work, and got some userland programs ported over. Sadly, the terrible communications scheme began to degrade, ultimately ending the project in the fall of 1977. However, change would come when a certain company reached out.

Amdahl: Amdahl is a company that, throughout the 80s, was IBM's biggest competitor. It had been founded by one of the lead designers of the System/360, and offered System/370-compatible (and later System/390-compatible) machines that sold reasonably well. John Hiles, who worked for Amdahl, managed to get Tom in one of those "right place, right time" moments: Tom was doing a summer internship at Bell Labs in 1977, when Ken Thompson answers the phone: it's John, who was trying to get information about what was seemingly just a rumored port of UNIX to the System/370. Tom, being close by to Ken, has a telephone receiver handed to him, and he begins to explain Princeton's UNIX/370.

UTS: Seeing that UNIX/370 might be a serious future for Amdahl, John went and acquired a UNIX license from Bell Labs. On spring break in 1978, Tom made it out to Amdahl to do some consulting for a week. Armed with 4 DECtapes containing the UNIX/370 sources (just to be clear, this UNIX/370 has absolutely nothing to do with the later Bell Labs TSS-based UNIX port), he and John rented some time on a PDP-11 held at a local DEC field office. There, he loaded up UNIX on the rented PDP at that office, and copied the DECtapes over to 9-track tapes. In a rare feat of preservation, images of these original tapes not only exist, but are available on GitHub; there were 4 DECtapes:

- the full UNIX/370 source
- the UNIX/370 "virgin source"

- the modified 370 C compiler
- the modified 370 cross-assembler

Shortly thereafter, in June, Tom began his job at Amdahl. By early 1979, the team had managed to bring over the rest of UNIX v6 to the 370 CPU architecture, and they were running a full UNIX on an Amdahl 470V/6 (of which Amdahl had partnered with Fujitsu to build) under VM/370. This port of UNIX was named "Au" (like "gold", on the periodic table) and quickly took success to its users. Tom had written a 3270 terminal driver for the OS, which also allowed for full-screen applications to be written (such as an advanced editor). In late 1979, Amdahl acquired a UNIX v7 tape from Bell Labs, and Amdahl UTS was properly announced in 1980.

Tapes for UTS were eventually sent back to Princeton, owing to the fact that a lot of the UTS developers were Princeton graduates. These tapes found themselves discarded on the side of the road outside of the Computing Center, until Dave Jones at Sine Nomine Associates discovered their untimely demise. These tapes were recovered, dumped, and in that stack were tapes for the original 1980 release of Amdahl UTS (complete with the PWB/UNIX tooling). Alas, it can be loaded and ran:

```
IPL 220
Enter system name:
autouts
UTS Version 1.0
Storage = 6144K
1405 free pages
/dev/dsk220:
/dev/dsk110:
/dev/dsk330:
/dev/dsk550:
/dev/dsk660:
no space on dev 4/0
no space on dev 4/0
no space on dev 4/0
no space on dev 4/0
no space on dev 4/0
no space on dev 4/0
no space on dev 4/0
DISCONNECT AT 16:37:29 CDT MONDAY 10/06/25

Press enter or clear key to continue


                                                    RUNNING   VMSP
```

Figure 39. Amdahl UTS 1.0 running under VM/SP R5

IX/370

Little is known about IX/370 in this day and age, solely because of the fact that there are no known extant tapes of it available in this era. Nonetheless, it was designed to run under VM/SP, and was supposedly popular on the 9370.

What information is available for IX/370 comes from, of all places, the 9370 product announcements and the associated program product announcements that followed it. IX/370 boasted that it had virtual memory storage and paging, which IBM claimed many other UNIX System V implementations lacked (this is true; note that this product was produced before the newer UNIX System V Release 4, the version most commonly seen on UNIX workstations and servers through the 90s).

It also remarked some interesting notes about the filesystem implementation. The announcements note that IX/370 used a filesystem block size of 4096 (does this sound familiar?), whereas other UNIX implementations used 512 or 1024. It also boasts the addition of a system call that facilitated a filesystem locking mechanism (one might recognize it from newer UNIX systems: lockf).

Most interestingly, it notes that several IX/370 subsystems could run under one master IX/370 system. Each subsystem operated independently of the others, with fully isolated users, programs, and more. This is notably similar to the famous Bell Labs UNIX/370 system! Since it had to run under VM, it supported the VM-emulated line printers (and could also associate tag data with the print jobs, allowing for file distribution on an RSCS network). In order to make the system passingly attractive to an office customer base, IBM licensed INmail and INnet from Interactive Systems Corporation (of note for their PC UNIX offerings in this era; PC/IX and 386/ix were reasonably popular until Sun acquired ISC and released Interactive UNIX). To connect to ASCII terminals, both the UNIX/370 Series/1 terminal attachment as well as the 9370's ASCII Subsystem Controller.

Needless to say, all signs point to IX/370 as being derived from the Bell Labs UNIX/370 project. Alas, no tapes of it exist, so further study is impossible.

AIX/370

In 1988, IBM partnered with Locus Computing Corporation to produce a (supposed) port of AIX to the PS/2 (i.e. Intel 386) platform. In reality, it was a port of the OSF/1 codebase to the 386, and it was surprisingly expensive for the customers that did buy it. Nonetheless, the people that did buy it found a reasonably useful system, but it was not a commercial success.

More interestingly, however, was something else that resulted from the IBM/Locus partnership (also released that same year, on March 15, 1988): AIX/370. This was a true and real port of UNIX (again, OSF/1) to the 370 and 370/XA CPU. In a flash of brilliance, it also supported Token Ring and Ethernet networking through the IBM 8232 LCS boxes (predating the later 3172); to date, no other version of UNIX on the mainframe save for Amdahl UTS supported this. It had to be ran under some version of VM (supported was VM/SP, VM/SP HPO, and, surprisingly, VM/XA SP), and was fully bimodal; it even supported the 3090's Vector Facility! It was fully POSIX-compliant, and had feature-parity matching with both UNIX System V Release 2 *and* 4.3 BSD (at that point, those were the only two versions of UNIX that had significant usage share).

This port was no joke. It had, as evidenced by the announcement letter:

- System/370, XA, and 3090 Vector Facility support; quite a tall order for a totally new OS
- full TCP/IP support
- a "DOS Server" feature that may have been related to an earlier Locus product
- The Transparent Computing Facility, allowing access to distributed computing systems
- full X11 support (albeit through a network; no display devices attached to the mainframe would be able to show X graphics)
- an optimizing C compiler
- a command to invoke a program on CMS (named oncms)

The addressing size limits were rather interesting. When the OS was running in 370 mode, user programs could access up to 8 MB of virtual memory; in XA mode, the ceiling was raised to 770 MB (a rather odd limit). The Vector Facility was also fully supported, and IBM had planned to release a port of VS FORTRAN (both the compiler and the Engineering/Scientific Subroutine Library) to AIX/370, but no surviving copies of this exist.

The TCP/IP stack was interesting, as it was the UNIX System V TCP/IP stack ported straight to the 370. It had support for SMTP, FTP, Telnet, the BSD socket API, the ability to be a router/gateway (which supported Ethernet,

Token Ring, and channel-to-channel adapters, presumably for talking to the VM TCP/IP stack directly), and support for the BSD "r commands" (rsh, rcp, rlogin).

The DOS Server feature was a host-based program that ran on an AIX system (which could be AIX/6000 or AIX/370) wherein connections from DOS (3.3) PCs were accepted. The DOS PC would run the IBM AIX Access for DOS Users program, which provided access to AIX files, printers, and included a VT100 emulator for remote connection to AIX (this program also included an absolutely famous unused error message, which was, and I quote, "Sex feels so fucking good, I just can't stop.", I am dead serious).

In addition to that DOS Server package, AIX/370 could also talk on an NJE network (with the aid of RSCS, presumably running on the same VM system under which AIX/370 executed). Full NETDATA file support was present, allowing users to easily transmit files, datasets, and emails between AIX/370 and other OSes.

Since X.25 was quite popular in this era, AIX/370 also had support for X.25 networks. However, it would require a PS/2 Model 80 (though it would possibly work with other systems) that had an X.25 synchronous serial adapter installed, and was running AIX PS/2. The AIX/370 host would, by way of being on the same LAN as the AIX PS/2 box, be able to utilize an X.25 network (this almost certainly seemed to be a solution wherein users would log into the AIX PS/2 system from the AIX/370 system, then be able to make X.25 PAD connections from there or vice-versa).

The Transparent Computing Facility was another AIX PS/2 overlap product, and it allowed an AIX/370 system or an AIX PS/2 system to become a member of a true cluster system. Programs could run on any host, files could be stored on any host, and you did not have to switch which node you were logged into in order to run a program on a different machine. This also incurred a shared file system, and there were redundancy and locking measures to ensure that a system failure on one node would not result in other programs automatically being reverted to older versions of files. Because of the distributed nature of this clustering scheme, processes could also hot-migrate from one system to another in a bit of a load-balancing scheme. Really, this was a surprisingly advanced program, and the functionality of it has not been seen since (as no copies of AIX/370 are known to exist, and it does not seem to be archived for AIX PS/2 either).

IBM, having partnered with Interactive Systems Corporation to provide first-class ISV application support for both AIX PS/2 and AIX/370, made available a number of ISC's own programs to both of these systems. INnet was a UUCP-style networking package that ISC produced, and it worked hand in hand with INmail to transfer emails to remote sites. Since it was fully compatible with UUCP, users were not locking themselves into some proprietary network system that would stop being useful when support ended. ISC also provided an improved FTP client (wherein the command that was ran was all-caps too, "FTP") and an improved editor (INed, a program that is often seen on Interactive UNIX systems of the same vintage) that was a heavily improved offering over the built-in UNIX vi program; it supported multiple edit sessions, several windows, and better copy-paste functions. Alas, it was eventually replaced with...

AIX/ESA

Seeing that it was high time for an upgrade to AIX/370 (only if there were only ever a few customers), IBM announced AIX/ESA in 1990. This was overtly based on the OSF/1 codebase, even though earlier versions were based on it. The most important change was the ability for it to run without the aid of VM, directly on the real hardware; it also gained multiprocessor support (though it started with support for 3 CPUs; support for 6 CPUs was added later by September 30, 1992).

Support for HIPPI (High Performance Parallel Interface) adapters was also added; this was an old interface intended to connect peripherals to supercomputers, and had an effective throughput of 800 megabits. AIX/ESA also gained for the new-and-improved 3172 Interconnect Controller (though frequently just called the *3172 LCS*), but you could now use a channel-attached (a parallel channel, mind you; the ESCON interfaces did not yet function with this scheme) RS/6000 as a front-end network processor on the mainframe. This supposedly ran faster than the 3172 LCS

boxes, which was accessed like a 3088 CTC adapter; this was aided by the much faster POWER CPU of the RS/6000 (which certainly dwarfed the CPU performance of the 8283s and 3172s, which were essentially just PCs).

Since this was based off of OSF/1, there were more features than meets the eye! For one, you could now dynamically load kernel modules (a feature now standard on UNIX, but was almost entirely unheard of in 1992), use an advanced UNIX accounting system (not banking, but taking account of processes), and more.

Sadly, AIX/ESA barely had any installations. It was a massive failure because it suffered the dreaded fate of competing against a product from the same vendor. This competition came from MVS/ESA V4R3's OpenEdition feature, wherein MVS, the stalwart of mainframe operating systems, had its own UNIX implementation. This system saw much support, and since it came with MVS, sites that were merely trying to stay with the latest version of MVS found themselves in possession of a rather good UNIX environment for free.

TPF and ACP

The history of TPF is almost synchronous with OS/360, and the System/360 in general. In the early days of commercial computing, there was a software system that IBM and American Airlines developed called SABRE (Semi-Automated Business Research Environment). This was intended to be a computerized replacement to the earlier way of manually handling airline reservations: before the dawn of computing, airline ticket salespeople would sell seats on a flight, and would then report to the departing city's airport how many seats were being sold. If the plane was close to full, the head office would send a message to the salespeople to stop selling any more seats. Seeing that this was a problem, IBM approached American Airlines in 1953 and asked for help. By the early 60s, the future was clear -- computers were the answer.

SABRE: While IBM was working with American Airlines, they chose to start software development contracts with Pan- Am and Delta Airlines too. As such, there were three airline reservation systems that IBM was making in the early 1960s (before the S/360):

- DELTAMATIC (ran on an IBM 7070), Delta Airlines
- PANAMAC (ran on an IBM 7080), Pan-Am Airlines
- SABRE (ran on an IBM 7090), American Airlines

In 1962, nobody had really figured out or perfected the craft of software engineering and software development project management. As such, the development of SABRE ended up being plagued by constant delays, not to mention that IBM was (at the same time) working on similar systems for two other radically-diffferent computers.

When SABRE was finished in 1964, it was the first online tranaction processing system and the first large-scale data processing system that wasn't owned by the US government. Raytheon built terminals that attached to the host mainframe via telecommunications lines, and these were placed in all of the serviced airports that American Airlines had a presence in. In 1965, IBM finished DELTAMATIC and PANAMAC.

PARS: Seeing that IBM had made a poor decision to write three radically-different airline reservation systems, IBM sought to unify the customers' computing needs on the up-and-coming System/360 line -- this replacement program needed to run on a variety of models (with the specifified range being the Model 40 through the Model 75; the Model 65 was the most popular). This replacement software package would be known as PARS: the *Programmed Airline Reservation System.*

PARS was originally written to target midsized airlines, rather than the giants discussed above. The development turnaround time on PARS was amazing compared to that which came before it: in 1965, shortly after the annoucement of the System/360 and on the tails of the shipments of the first System/360s, PARS was made available to customers. The first customer to adopt it was Eastern Air Lines, wherein they named their install “System One” Later that year, the British Overseas Airways Corporation chose to build a PARS installation (which they named BOADICEA, as in, the ancient queen of England). Seeing that England was a smaller country and would absolutely be making many international flights, IBM updated PARS to support such features and BOAC gained their IPARS system (the I was for International).

During that same timeframe from 1971 to 1973, American Airlines was undergoing their migration off of SABRE and onto a PARS system (which ran on several System/360 mainframes). Seeing that American Airlines was having such good success with PARS, all but one of the ten major US airline carrier companiess followed suit and adopted PARS.

Seeing that the hardware was evolving to the System/370 and different-sized airlines were having good success adopting it, IBM split the operating system component from the transaction processing component; the OS became known as ACP (Airline Control Program) and PARS ran on top of that.

ACP: In the early 1970s, the PARS systems were seen providing great success to the companies that installed them. Fewer cancelled flights, fewer delays, and more -- everyone was taking notice. Unlike the precursor SABRE system, the design of ACP/PARS was *very* forward-thinking. Rather than a centralized system, PARS/ACP was **distributed!**

However, not everyone was content. With more money moving around in that era, the big US banks looked upon the airline industry with envy. They were getting more customers through the door than ever, and the banks were a decade behind in computing software. In the early 70s, the banks had adopted CICS and IMS for most of their transaction-processing workload, but the performance left much to be desired.

The banks had a whole suite of online transaction-processing systems, but their slow CICS/IMS systems were far from performance-competitive with the airline computer systems. As such, they begged IBM to give them a version of the PARS/IPARS transaction processing system, and they delivered in 1975!

The “new” TPF system was full of interesting software innovations -- for example, the SabreTalk programming language (sometimes called PL/TPF)! This language wsa produced in a three-way partnership between IBM, American Airlines, and Eastern Air Lines. By 1973, Eastern Air Lines was selling the SabreTalk compiler (for $95,000).

Eventually, that program would come to pass; British Airways was the last major user of SabreTalk programs comprising their Flight Operations Computer System (FICO) under the ALCS system... except the programs are compiiled by first transpiling to C (with a commercially-available compiler) and then compiling that C program. Delta Air Lines also used SabreTalk programs, but has been transitioning to C++ on TPF.

TPF and TPF/ESA: In 1979, IBM continued to genericify ACP into the Transaction Processing Facility (i.e. TPF). There were various versions of TPF, with TPF/ESA 4.1 being the last version that ran on ESA/390 machines. Programs for TPF were predominately written in assembler language.

TPF could run in both a tightly-coupled multiprocessing mode (i.e. standard multiprocessor mainframes like the 3083), or loosely-coupled mode across a network. For some reason, a single CPU within a TPF LPAR is called an *instruction stream* or i-stream, and a machine with more than one i-stream will be running tightly-coupled TPF. Several of those systems could then be joined together loosely.

Of course, such computer systems require some kind of serialized access to disks and other such devices. This is done by the Record Hold Table, which implements spinlocks for synchronization within the TPF OS nucleus. Of course, the DASD controllers themselves had a record-locking facility, but one would have to get the LLF (Limited Locking Facility) PRPQ or the ELLF (Extended Limited Locking Facility); these two were eventually replaced by MPLF (Multipathing Lock Facility) which could use the Coupling Facility.

z/TPF: z/TPF 1.1 was released in September 2005; the new 64-bit support was nice, but the development model changed drastically. Before z/TPF, one would cross-develop programs from MVS in assembly language (or a specialized language like SabreTalk); once z/TPF was released, this development model changed significantly. Instead of the old development model in assembler, IBM provided a C-based development model derived from the GNU development toolchain. IBM had spent much effort improving the quality of the s390x port of GCC (for z/Linux); with these improvements available and TPF's development model decades obsolecent, IBM chose to make *Linux* the development platform!

Programs that would run on z/TPF would actually be s390x-linux-gnu ELF files, either compiled on z/Linux or later on PC Linux. If the programmer wishes to debug a TPF program, it is done so in a client-server fashion. TPF itself doesn't come with a debugger, and the programmer has to go buy a third-party debugger (like CMSTPF, TPF/GI, or zTPFGI; alternatively, Step by Step Trace). IBM now provides a debugger, running on a PC workstation, called TPF Toolkit -- this is a modified version of Eclipse (as IBM is known for doing).

TPF provides some basic operating system constructs for tasks like disk I/O and networking. TPF, naturally, supports both SNA and TCP/IP networking; likewise, CKD DASD disks and tapes are supported. The TPF operator's

console is a 3270 terminal called the *prime CRAS* (a strange acronym, standing for Computer Room Agent Set); user terminals are local 3270 or local SNA 3270 terminals (both operating modes of a channel-attached 3x74).

Though I would love to write more, I do not have access to any TPF systems or install media for it, but I hope this was an informative writeup on this strange OS.

IBM 9370: DPPX/370

DPPX has an interesting history; the Distributed Processing Executive was first found on the IBM 8100 Information System in 1978. In 1986, IBM eventually discontinued the decade-old 8100, and the last version was DPPX/SP Release 4. Seeing the void, IBM ported DPPX to the System/370 in a rather interesting successful project: in 1988, launching with the IBM 9370, IBM unveiled the 8100's replacement.

DPPX is, as having an interesting history, also has a rather odd architecture. The OS is written in a high-level-language called PL/DS (PL/Distributed Systems), derived from PL/I. When DPPX was being ported to the 370, PL/DS 2 also resulted; this was a necessary creation, since the PL/DS source for the 8100 was filled with inline assembly and other such primitives.

Since DPPX was intended to be a distributed processing node on a network of more systems of its kind, the OS was fully operable remotely (and could be ran without operator intervention). Likewise, the DPPX commands and dialogs were fitted with extensive online help; the OS was certainly, for its time, extremely easy to use (compared to something like VSE or MVS). Of course, these manuals were also available in print.

The original DPPX featured a key-lookup database, wherein a program could look up a record by providing a key. At that point, the program can traverse through the database forwards... but not backwards. Since it took more CPU time to run the query over then seek backwards to the right record given that key, application performance could decrease quite a bit. DPPX/370 fixed this, but it was done in a rather odd fashion: since the DPPX DBMS provided alternate keys, it would be possible for every key to have an alternate key that pointed to the previous record. The program would then select the alternate key, which was just the binary 1's complement of the primary key. Reading the next record in the returned data upon selecting that alternate key would read the previous record associated with the primary key.

DPPX was a surprisingly complex system. The shell was the Host Command Facility (HCF), and the Distributed HCF application allowed a user to remotely log into a DPPX/370 system through a teleprocessing network (inevitably SNA; DPPX/370 never gained TCP/IP support). Much like CICS on MVS and VSE, DPPX/370 provided the Distributed Transaction Management System (DTMS). The Distributed Systems Network Executive (DSNX) allowed cross-system file sharing, something flashy and new in that era.

DPPX/370 also provided language compilers for languages besides PL/DS, those of which were COBOL and FORTRAN. Applications written in any language that wished to provide full-screen user interfaces could use the Command Facilities Extensions system, and users that wished to log into remote mainframes (of the 370 variety) could use Data Stream Capability (which functioned somewhat like PVM's 3271/3274 emulation). Finally, there was a Performance Tool to monitor the system's resource utilization.

Solaris on System Z

Coming soon!

Popular Mainframe Programs

In this section, let us explore some of the more famous mainframe software products!

Customer Information Control System (CICS)

When it comes to most popular mainframe programs, CICS is easily a top 5. CICS is a transaction processing system, wherein small programs are ran over and over (either under the control of a terminal or a network service request); CICS also does file handling, terminal control, database manager access, and (later) web support.

History: CICS was written shortly after OS/360 became mature enough to host user-written (or, more accurately, IBM-written) programs. It replaced an earlier program called the Minimum Teleprocessing Communications System (MTCS) that was written to run under OS/MFT, OS/VS1, and DOS/VS. MTCS was not intended to be used for high-volume transaction processing, and, as such, was entirely singlethreaded; an unofficial multithreaded version was written by IBM Littlewoods, but this was made well after the heyday of MTCS (as a matter of fact, it was intended to facilitate a MTCS-CICS bridge -- this was intended to be a migration path off of MTCS). MTCS transactions were identified by 4-letter transaction codes, and the transaction code would be entered on a terminal (originally a 2741, later 3270 displays near the end of MTCS's life) to initiate the program.

CICS was developed in conjunction with the Michigan Bell company starting in 1966 at IBM Des Plaines; the original target market was the public utility industry. In 1968, IBM announced the first version of CICS: the Public Utility Customer Information Control System, or PU-CICS. When it became apparent that this program would have a userbase outside of public utility companies, the PU was dropped. This slightly-updated version was a program product, and was released in conjunction with Information Management System (IMS) by July 8, 1969.

The basic gist of CICS was finished by 1969, but it didn't support much (for instance, it only supported 2741 typewriter terminals attached via a 2703). The first big adopter of CICS was Amoco (Standard Oil of Indiana), who would eventually undertake rewrites of major components of CICS. Amoco had tons of Teletype 33 ASR terminals attached to 2703 ASCII ports, but CICS didn't support non-2741 terminals initially; as such, programmers at IBM Des Plaines tried to add support for popular third-party terminals of the day, but the team was handicapped by a rather interesting limitation: they were so poorly-funded (they weren't funded at all, actually) that they could not afford $100 a month to rent terminals to test. This was compounded by IBM management, constantly falsely stating that batch processing was the future of computing and not timesharing/interactive systems/virtual memory (see the aforementioned discussion around the failures of IBM there)!

Seeing that IBM was less than helpful, the Amoco programmers opted to make the support improvements happen themselves. They found out that after starting CICS with IBM's *untested* non-2741 terminal support, OS/360 would just crash. As a response, Amoco programmers had to rewrite the CICS Terminal Control Program!

CICS was eventually handed off to IBM Palo Alto, but it was seen as a smaller and less-important product than IMS; by then, IMS had a transaction monitor, database manager, and things of that nature -- it was certainly more technically-advanced than CICS, but not everyone wished to adopt it (and many stuck with CICS, even to this day). IBM chose to actually end CICS development in 1974 and tried to get everyone to move to IMS, the IBM Hursley site (over in England) took over development. They had just finished developing the PL/I compiler (only to kick it off to some other IBM office) and were therefore familiar with many of the CICS customers (due to most CICS customers writing transactions in PL/I and COBOL).

By 1972, there were three versions of CICS available:

- DOS-ENTRY: intended to run on DOS/360 machines that did not have much memory
- DOS-STANDARD: intended to run on DOS/360 machines that had more memory (and could support application development)

- OS-STANDARD: intended to run on the large OS/360 machines

CICS was updated gradually over time with several other versions; the DOS/360 versions of CICS ultimately evolved into two major releases of CICS:

- CICS/VSE, the traditional CICS version for VSE/SP and newer versions
- CICS Transaction Server for VSE, used to support CICS/ICCF on versions of VSE starting with VSE/ESA V2

The MVS CICS versions progressed through various versions, and were updated to support virtual storage on OS/VS2 (CICS/VS), updated for MVS/XA (CICS/MVS), and enhanced for MVS/ESA (CICS/ESA). When OS/390 was released, CICS/ESA became CICS Transaction Server, a name that sticks to this day with z/OS. When z/OS did come out, CICS was updated to support 64-bit programs.

Other CICS Versions: There were a number of spin-off versions of CICS, including:

- CICS/VM: released in 1988, this was intended to provide a CICS analogue for VM/CMS systems. This was not a real CICS subsystem, and was instead a series of libraries that would run alongside normal CMS program modules. This was intended to be a simplified environment for converting CICS transactions over to interactive CMS programs.
- CICS for Windows NT and OS/2: intended for application development and small customers, these programs were remarkably similar and supported a client-server communication model (wherein CICS clients provided 3270 displays to the CICS system; users could also use TN3270 clients). VSAM file support was provided by Btrieve, and it fully worked with Db2 connections.
- CICS for AIX, HP-UX, Solaris, and DIGITAL UNIX: a UNIX conversion of the CICS/NT and CICS/2 products, these products were intended to be a step-up from the PC-based editions. The same client-server and Db2 support was present.
- TXseries, the updated name for the above two programs
- CICS/400, a full CICS subsystem for OS/400 (IBM i) that interoperated with the inbuilt OS/400 database (sometimes called Db2/400, even though OS/400 itself was a database manager). Users would open CICS sessions by running the STRCICS CL command.

```
***DFH2312   WELCOME TO CICS/VS  *** 19:24:44


         CCCCCC    IIIII    CCCCCC     SSSSSS         VVVV  VVVV    SSS
       CCCCCCCC   IIIII   CCCCCCCC   SSSSSSSS         VVV   VVV   SSSSS
      CCCC  CC    III    CCCC  CC   SSSS  SS          VVV  VVV   SSSS
     CCC         III    CCC         SSSS       ***    VVV VVV    SSSS
    CCC         III    CCC           SSSS     ***     VVVVVV      SSSS
   CCCC  CC    III    CCCC  CC   SS  SSSS             VVVVV   SS  SSSS
  CCCCCCCC   IIIII   CCCCCCCC   SSSSSSSS              VVVV   SSSSSSSS
  CCCCCC    IIIII    CCCCCC     SSSSSS                VVV    SSSSSS
```

Figure 40. CICS/VS 1.7.0 welcome screen (MVS/SP 1.3.4)

```
WELCOME TO CICS 18:26:37


          ******\  ******\  ******\   ******\(R)
         ********\ ******\ ********\ ********\
         **\\\\**\   **\\\ **\\\\**\ **\\\\**\
         **\    \\   **\   **\    \\ **\    \\
         **\         **\   **\       *******\
         **\         **\   **\        *******\
         **\         **\   **\         \\\\**\
         **\   **\   **\   **\   **\ **\   **\
         ********\ ******\ ********\ ********\
          ******\\ ******\  ******\\  ******\\
           \\\\\\   \\\\\\   \\\\\\    \\\\\\
```

Figure 41. CICS TS 3.2.0 welcome screen (z/OS 1.5)

```
VSE/ESA CICSOLD 00:25:59


      CCC    IIII     CCC     SSS            VVV   VVV   SSS    EEEEEEE
     CCCCC    II     CCCCC   SSSSS            VV   VV   SSSSS   EE   EE
    CC  CC    II    CC  CC   SS  SS           VV   VV   SS  SS  EE
    CC        II    CC        SS      ***     VV   VV    SS     EEEEE
    CC        II    CC          SS    ***     VV   VV      SS   EEEEE
    CC  CC    II    CC  CC   SS  SS            VV VV    SS  SS  EE
     CCCCC    II     CCCCC   SSSSS              VVV     SSSSS   EE   EE
      CCC    IIII     CCC     SSS                V       SSS    EEEEEEE
```

Figure 42. CICS/VSE 1.2 welcome screen (VSE/ESA 2.1.0)

```
VSE/ESA ONLINE 00:27:45


      ******\  ******\  ******\   ******\
     ********\ ******\ ********\ ********\
     **\\\\**\   **\\\ **\\\\**\ **\\\\**\
     **\    \\   **\   **\    \\ **\    \\
     **\         **\   **\       *******\
     **\         **\   **\        *******\
     **\         **\   **\         \\\\**\
     **\   **\   **\   **\   **\ **\   **\
     ********\ ******\ ********\ ********\
      ******\\ ******\  ******\\  ******\\
       \\\\\\   \\\\\\   \\\\\\    \\\\\\ TM
```

Figure 43. CICS TS 2.3 welcome screen (VSE/ESA 2.4.0)

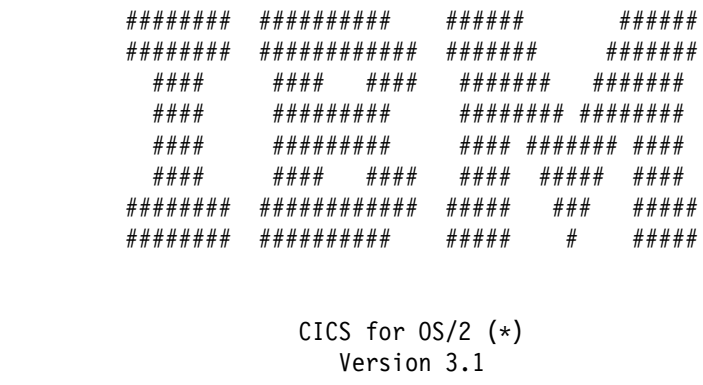


Figure 44. CICS for OS/2 3.1 welcome screen (OS/2 Warp 4)

Application Programming: CICS introduced a rather different method of application development. Initially, CICS used a library of assembler macros for calling CICS services. This was called ***Macro-Level Programming.***These looked like this sample assembler program that read a string from a terminal and then echoed it:

```
         COPY   DFHCSADS          COPY CSA AND TCA SYMBOLIC STORAGE DEFS
         COPY   DFHTCADS
LENGTH   DS     H
MESSAGE  DS     CL32
         COPY   DFHTCTTE          COPY SYMBOLIC STORAGE DEFINITIONS
         COPY   DFHTIOA
MYSTR    DS     CL32
CICSTEST CSECT
         BALR   2,0
         USING  *,2
         L      11,TCAFCAAA       SET UP TCTTE AND TIOA ADDRESSABILITY
         L      10,TCTTEDA
         MVC    MYSTR,=C'ENTER TEXT TO ECHO'
         MVC    TIOATDL,=H'18'
         DFHTC  TYPE=(WRITE,READ,WAIT,ERASE)
         L      10,TCTTEDA
         MVC    LENGTH,TIOATDL    SAVE INPUTTED LENGTH
         MVC    MESSAGE,MYSTR     SAVE INPUTTED MESSAGE
         DFHSC  TYPE=GETMAIN,CLASS=TERMINAL,NUMBYTE=32     ALLOC MEMORY
         L      10,TCASCSA
         ST     10,TCTTEDA
         MVC    MYSTR,MESSAGE     MOVE INPUTTED STRING TO OUTPUT
         MVC    TIOATDL,LENGTH    MOVE LENGTH
         DFHTC  TYPE=WRITE        WRITE STRING
         DFHPC  TYPE=RETURN       EXIT PROGRAM
         END
```

Figure 45. CICS macro-level sample program (assembler)

As you can see in the above example, there are a variety of macros for various functions, such as:

- DFHFC: file control
- DFHTC: terminal control
- DFHDI: batch data interchange
- DFHIC: interval control
- DFHKC: task control
- DFHPC: program control

In addition, there are a variety of canned storage definition macros. Now, one might ask, *what about COBOL and PL/I?* Well, these were supported, but it was extremely primitive. Note that COBOL does not have pointers like PL/I does, so a COBOL CICS macro-level program would have to leverage the linkage section to actually do anything. Note that the linkage section is normally used for communication between programs (like passing parameters to subprograms), but this was crufty and unergonomic.

Examine the following COBOL program that does the same thing as the assembler program above:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CICSMCRO.
ENVIRONMENT DIVISION.
DATA DIVISION.
LINKAGE SECTION.
01  DFHBLLDS COPY DFHBLLDS.
    02  TCTTEAR PIC S9(8) COMP.
    02  TIOABAR PIC S9(8) COMP.
01  DFHCSADS COPY DFHCSADS.
01  DFHTCADS COPY DFHTCADS.
    02  SAVE-LENGTH PIC S9(8) COMP.
    02  SAVE-MESSAGE PIC X(36).
01  DFHTCTTE COPY DFHTCTTE.
01  DFHTIOA COPY DFHTIOA.
    02 TIOAMSG PIC X(36).
PROCEDURE DIVISION.
    MOVE CSACDTA TO TCASBAR.
    MOVE CSAOPFLA TO CSAOPBAR.
    MOVE TCAFCAAA TO TCTEAR.
    MOVE TCTEDA TO TIOABAR.
    MOVE 'ENTER MESSAGE TO BE ECHOED' TO TIOAMSG.
    MOVE 26 TO TIOATDL.
DFHTC TYPE=(WRITE,READ,WAIT)
    MOVE TCTTEDA TO TIOABAR.
    MOVE TIOATDL TO SAVE-LENGTH.
    MOVE TIOAMSG TO SAVE-MESSAGE.
DFHSC TYPE=GETMAIN,NUMBYTE=36,CLASS=TERMINAL
    MOVE TCASCSA TO TIOABAR.
    MOVE TIOABAR TO TCTTEDA.
    MOVE SAVE-MESSAGE TO TIOAMSG.
    MOVE SAVE-LENGTH TO TIOATDL.
DFHTC TYPE=WRITE
DFHPC TYPE=RETURN
    GOBACK.
```

Figure 46. CICS macro-level sample program (COBOL)

This is not very easy to use, but it was the price the programmers in the late 1970s paid. A PL/I program looks similar:

```
CICSMCRO: PROC OPTIONS(MAIN,REENTRANT);
  %INCLUDE DFHCSADS;
  %INCLUDE DFHTCADS;
    2 SAVE_LENGTH BIN FIXED(15);
    2 SAVE_MSG CHAR(36);
  %INCLUDE (DFHTCTTE);
  %INCLUDE (DFHTIOA);
    2 TIOAMSG CHAR(36);
  TIOAMSG = 'ENTER MSG TO BE ECHOED';
  TIOATDL = 26;
  DFHTC TYPE=(WRITE,READ,WAIT) TIOABAR=TCTTEDA;
  SAVE_LENGTH = TIOATDL;
  SAVE_MSG = TIOAMSG;
  DFHSC TYPE=GETMAIN,NUMBYTE=36,CLASS=TERMINAL
  TIOABAR = TCASCSA;
  TCTTEDA = TIOABAR;
  TIOAMSG = SAVE_MSG;
  TIOATDL = SAVE_LENGTH;
  DFHTC TYPE=WRITE;
  DFHPC TYPE=RETURN;
END;
```

Figure 47. CICS macro-level sample program (PL/I)

This is certainly more readable than the COBOL version, but this is still extremely cumbersome to write. As one might imagine, this would eventually be replaced: this came with an update produced in the early 80s, and was known as ***Command-Level Programming.***

Rather than manually calling macros from high-level languages or writing the program entirely in assembler, command-level programs would use a variety of EXEC CICS statements -- these would, in turn, be processed by some kind of precompiler (which one was invoked depended on the language being used). The result of this was that application development time fell significantly, as did user complaints.

Examine the following sample COBOL program:

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. CICSHI.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  WS-MESSAGE PIC X(40).
       01  WS-LENGTH PIC S9(4) COMP.
       PROCEDURE DIVISION.
       BEGIN.
           MOVE 'HELLO MVS CICS!' TO WS-MESSAGE
           MOVE 15 TO WS-LENGTH
           EXEC CICS SEND TEXT
               FROM(WS-MESSAGE)
               LENGTH(WS-LENGTH)
           END-EXEC
           EXEC CICS RETURN
           END-EXEC
           STOP RUN.
```

Figure 48. CICS command-level sample program (COBOL)

To translate this program, you would pass it through the DFHECP1$ program. The other translators are:

- Assembler: DFHEAP1$
- C: DFHEDP1$
- COBOL: DFHECP1$
- PL/I: DFHEPP1$

To run these procedures, one could do it by hand or by using a stored procedure that came with CICS. There were many of these, and they were as follows:

Table 11.

Language	**Non-LE**	**LE**
Assembler	DFHEITAL (online), DFHEXTAL (batch)	(none)
C	DFHEITDL (online), DFHEXTDL (batch)	DFHYITDL (online), DFHYXTDL (batch)
C++	(none)	DFHYITEL (online), DFHYXTEL (batch)
COBOL	DFHEITVL (online), DFHEXTVL (batch)	DFHYITVL (online), DFHYXTVL (batch)
PL/I	DFHEITPL (online), DFHEXTPL (batch)	DFHYITPL (online), DFHYXTPL (batch)

After the program was translated, compiled, and linked, it would need to be defined to the system. This was conventionally done in the past by updating the DFHPCT (program control table, which defines transactions) and DFHPPT (processing program table) and re-assembling it. By 1985, online resource definition was now possible; one could run the CEDA transaction and define the transaction.

Another useful feature for application development was a function known as ***Basic Mapping Support*** (i.e. BMS). This was an assembler-macro way of defining panels. CICS BMS maps were always cumbersome, obtuse, and best left generated by some kind of generator; the most common generator was IBM Screen Definition Facility II (SDF II), popular on MVS and VSE.

Here is an example CICS BMS map source:

```
         PRINT ON,NOGEN
LOGPNL   DFHMSD TYPE=MAP,LANG=C,FOLD=UPPER,MODE=INOUT,STORAGE=AUTO,    *
               SUFFIX=
LOGMENU  DFHMDI SIZE=(24,80),MAPATTS=(COLOR,HILIGHT),COLUMN=1,LINE=1,  *
               DATA=FIELD,TIOAPFX=YES,OBFMT=NO
         DFHMDF POS=(1,1),LENGTH=1,ATTRB=(PROT,BRT)
         DFHMDF POS=(1,76),LENGTH=4,INITIAL='LOGM',ATTRB=(PROT,BRT),   *
               COLOR=PINK
         DFHMDF POS=(2,30),LENGTH=23,INITIAL='LogManager370 Main Menu',*
               ATTRB=(PROT,BRT),COLOR=YELLOW
         DFHMDF POS=(6,3),LENGTH=1,INITIAL='1',ATTRB=(PROT,BRT),       *
               COLOR=NEUTRAL
         DFHMDF POS=(6,9),LENGTH=22,INITIAL='Add entries to logbook',  *
               ATTRB=(PROT,BRT),COLOR=TURQUOISE
         DFHMDF POS=(8,3),LENGTH=1,INITIAL='2',ATTRB=(PROT,BRT),       *
               COLOR=NEUTRAL
         DFHMDF POS=(8,9),LENGTH=25,                                   *
               INITIAL='Update entries in logbook',ATTRB=(PROT,BRT),   *
               COLOR=TURQUOISE
         DFHMDF POS=(10,3),LENGTH=1,INITIAL='3',ATTRB=(PROT,BRT),      *
               COLOR=NEUTRAL
         DFHMDF POS=(10,9),LENGTH=27,                                  *
               INITIAL='Remove entries from logbook',ATTRB=(PROT,BRT), *
               COLOR=TURQUOISE
         DFHMDF POS=(12,3),LENGTH=1,INITIAL='4',ATTRB=(PROT,BRT),      *
               COLOR=NEUTRAL
         DFHMDF POS=(12,9),LENGTH=24,                                  *
               INITIAL='Query logbook statistics',ATTRB=(PROT,BRT),    *
               COLOR=TURQUOISE
         DFHMDF POS=(14,3),LENGTH=1,INITIAL='5',ATTRB=(PROT,BRT),      *
               COLOR=NEUTRAL
         DFHMDF POS=(14,9),LENGTH=34,                                  *
               INITIAL='Generate ADIF and store to dataset',           *
               ATTRB=(PROT,BRT),COLOR=TURQUOISE
         DFHMDF POS=(16,3),LENGTH=1,INITIAL='6',ATTRB=(PROT,BRT),      *
               COLOR=NEUTRAL
         DFHMDF POS=(16,9),LENGTH=4,INITIAL='Help',ATTRB=(PROT,BRT),   *
               COLOR=TURQUOISE
         DFHMDF POS=(18,3),LENGTH=1,INITIAL='7',ATTRB=(PROT,BRT),      *
               COLOR=NEUTRAL
         DFHMDF POS=(18,9),LENGTH=4,INITIAL='Quit',ATTRB=(PROT,BRT),   *
               COLOR=TURQUOISE
         DFHMDF POS=(22,1),LENGTH=11,INITIAL='Option ===>',ATTRB=(PROT,*
               BRT),COLOR=GREEN
* CHOICE                          CHOICE
CHOICE   DFHMDF POS=(22,14),LENGTH=1,ATTRB=(UNPROT,NORM),COLOR=BLUE,   *
               HILIGHT=UNDERLINE
         DFHMDF POS=(22,16),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(24,2),LENGTH=6,INITIAL='F3=End',ATTRB=(PROT,BRT), *
               COLOR=BLUE
```

Figure 49. CICS BMS map sample

When assembled and displayed by a program using the EXEC CICS SEND MAP command, it looked like this:

```
                        LogManager370 Main Menu


   1     Add entries to logbook

   2     Update entries in logbook

   3     Remove entries from logbook

   4     Query logbook statistics

   5     Generate ADIF and store to dataset

   6     Help

   7     Quit


 Option ===>

  F3=End
```

Figure 50. Sample CICS map displayed on a 3270 terminal

Of course, CICS maps were dependent on the device. CICS supported many terminals in its day, some of which weren't really traditional terminals (like supermarket barcode scanners, RJE batch terminals, an audio response phone communication device, and other such strange devices).

Modern Features: Over the course of the 90s and 2000s, CICS gained many features that modernized it. These included the ability to call CICS services from Java applications (all the rage in the late 90s), REST API support (critical for the modern age), and a full CICS Web subsystem that permitted a CICS region to function as a web server or web client.

The web service functions are achieved through a tapestry of configuration options, and is certainly more than meets the eye at first. For example, one can first prepare CICS to receive web client connections by defininining a TCPIPSERVICE, then give it a URIMAP to build a list of URLs, then either allow it to serve static content from z/OS UNIX files *or* generate dynamic content programmatically. If that seems too basic, CICS can also dynamically generate HTML forms from BMS maps (like those described above).

CICS Applications: While there are more CICS applications that exist than I could ever readily compose a writeup for, here are some influential early ones:

- Source Program Maintenance Online II: SPMOL-II (with the worst verbal name ever, "*spimoli*") was a circa-1970s online editor program that ran within CICS. Since OS/VS1 did not run TSO, SPMOL-II provided the only good interactive editing facility that OS had; the slated purpose of this program was not for general text editing but explicitly for the development and maintenance of source programs (as the name might imply; put simply, this was written to be an online editor for COBOL, PL/I, and facilitate the submission of JCL directly into the JES2 INTRDR in lieu of a card deck).
- STAIRS
- ATMS III: an online administrative document processing subsystem, ATMS evolved from the earlier ATS/360 (Administrative Terminal System) and was designed as a successor to it that ran under CICS (instead of doing everything itself: ATS/360 did all disk I/O by directly executing channel programs). To facilitate the formatting of documents, IBM SCRIPT/VS ran under ATMS III and could save to files and/or spool to printers.

- GDDM: while not strictly a CICS program, the Graphical Data Display Manager provided graphics support for 3270 terminals that supported graphics as well as a whole barrage of AFP and similar (like PostScript) printers. The base GDDM applications minus GDDM-OPS (the Online Presentation System), as well as most user-written programs, could run under CICS.

GDDM

The blurb directly above this describes GDDM in passing, but the Graphical Data Display Manager provided a true graphics subsystem for the mainframe. Gone were the days of text-only graphics and printouts, this new program from 1988 made everything much more colorful and artistic. There were a number of programs shipped with a full installation of GDDM:

Base GDDM components: The basic GDDM package contained a series of library routines, copyfiles/headers/macros, and examples for various programming languages. One of those was the GDDM-REXX interface, which expedited application development. GDDM applications could be written pretty easily by using simple commands not unlike the commands for graphics seen on old 8-bit home computers that ran BASIC from a ROM; this example C program would draw a basic screen and capture mouse input from the light pen (or mouse, if displayed on a 3270 emulator like PCOMM):

```
#include <stdio.h>
#include <stdlib.h>
#include <gddm.h>


int main() {
    int dtype, devid, segid, symbid;
    fsinit();
    gssati(1,1);
    gsseg(5);
    gscm(3);
    gscb(10.0, 10.0);
    gsmove(1.0, 90.0);
    gstag(1);
    gschap(1, "#");
    gstag(2);
    gschap(2, "#");
    gsenab(3, 1, 1);
    gsread(1, &dtype, &devid);
    gsenab(3, 1, 0);
    gsqpik(&segid, &symbid);
    fsterm();
    printf("picked segment id: %d, symbol id: %d\n", segid, symbid);
    return 0;
}
```

Figure 51. GDDM C program that draws a demo menu

(thanks Vinatron for the program)

As one might imagine, GDDM found itself used for many programs that required any and all graphics!

GDDM-ICU: The ICU, or the Interactive Chart Utility, provided a simplistic way of generating charts on a mainframe. Data can easily be imported and exported in a simple tabular format (and, of course, edited on the spot). All manner of standard charts were supported, and this program was often used in conjunction with a page printer (loaded with transparencies) to produce presentation slides.

Charts would often look something like this:

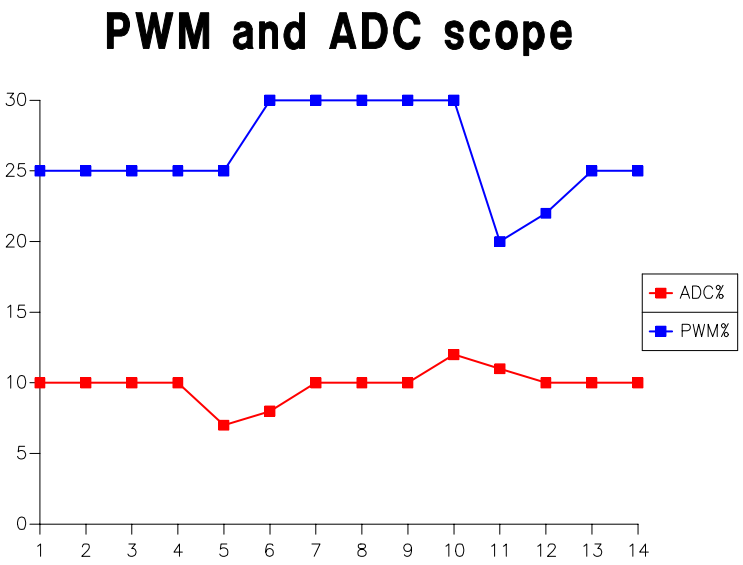


Figure 52. GDDM-ICU chart

GDDM-IVU: Intended to work with the 3193 portrait display and scanner, the Image Viewer Utility provided a scanner and image display utility for ADMIMG-format pictures. Since no printer natively supported ADMIMG files and one might wish to preview monochrome graphics for inclusion in DCF/BookMaster documents later, this program can load and save AFP page segment files.

GDDM-OPS: In the past, Microsoft PowerPoint was not really a thing; as such, producing slides for a presentation was much more difficult. Originally, presentation slides were hand-drawn or hand-typeset directly onto transparency slides, but these were not the same as the presentations we would later come to see dominate every business meeting ever -- producing printed text was a rather cumbersome ordeal, and usually involved expensive typesetting equipment and the like. Certain people realized that you could use graphics systems from television to produce slides, but they were of poor resolution (since television was limited to about 480 lines in this era); Genigraphics was the first to produce a widely-adopted computer whose job was graphic design (and these were quickly used to produce presentation slides onto transparencies).

In the mid-80s, IBM decided they would try to take their hand at producing a competitor to the popular Genigraphics computers. IBM had already been working on GDDM (as seen above), and some 3270-compatible terminals (and, of course, PCs running 3270 emulators) could drive “video projectors” to illuminate a projection screen.[1]

GDDM Online Presentation System leveraged the then-new advanced graphics features of the 3270 line (see the section in the first half of this book on the 3179-G) to render beautiful presentation slides for display on a terminal, printing with a printer (this output could then in-turn be photographically transferred onto slides), or on several terminals (provided that OPS was being ran on VM). OPS supported MVS and VM, and used a rather odd command language:

[1] A video projector is what we would nowadays refer to as just a “projector”; this is in contrast with a film projector or slide projector, which were more common in the mid-to-late 80s

```
)ops prefix !; reset; trc off
)ops dark; aspect 100 67; reset
! * draw the red line
!line 2 65.4 83 w 1.2 dr r
!mix on
!rays 88 62.5 7 dr r
!rays 92 62.5 7 dr g
!rays 96 62.5 7 dr b
!text 'z/OS' cc 92 62.5 ss admuukso h 5 sh 15 col w
!mix off
! * The next line is the chart heading. Uses admuuhi as the symbol set
! * is drawn in red, with a height of 7.8.
!text 'Objectives' 1 57.0 ss admuuhi  col r h 7.8
! * set the symbol set, color and height for the bullet text
!ss admuuh; col y; h 5
! * start the bullets at x=6, y=47
!start 6 47
! * ensure text wraps at edge
!tf xmax 95
! * start the 'unordered list'
!tful psp .4l
! * and now the bullet text. add/delete bullets as necessary
!tfli Figure out how to run SQL statements
!tfli Write a program
!tfli Precompile/compile/link/bind/run it
!tfli Show interactive and batch program prep
!tfli Memorize JCL
!tfli Write CICS and Db2 application
!tfeul
```

Figure 53. Example GDDM-OPS presentation

When processed, OPS presentations could look like this:

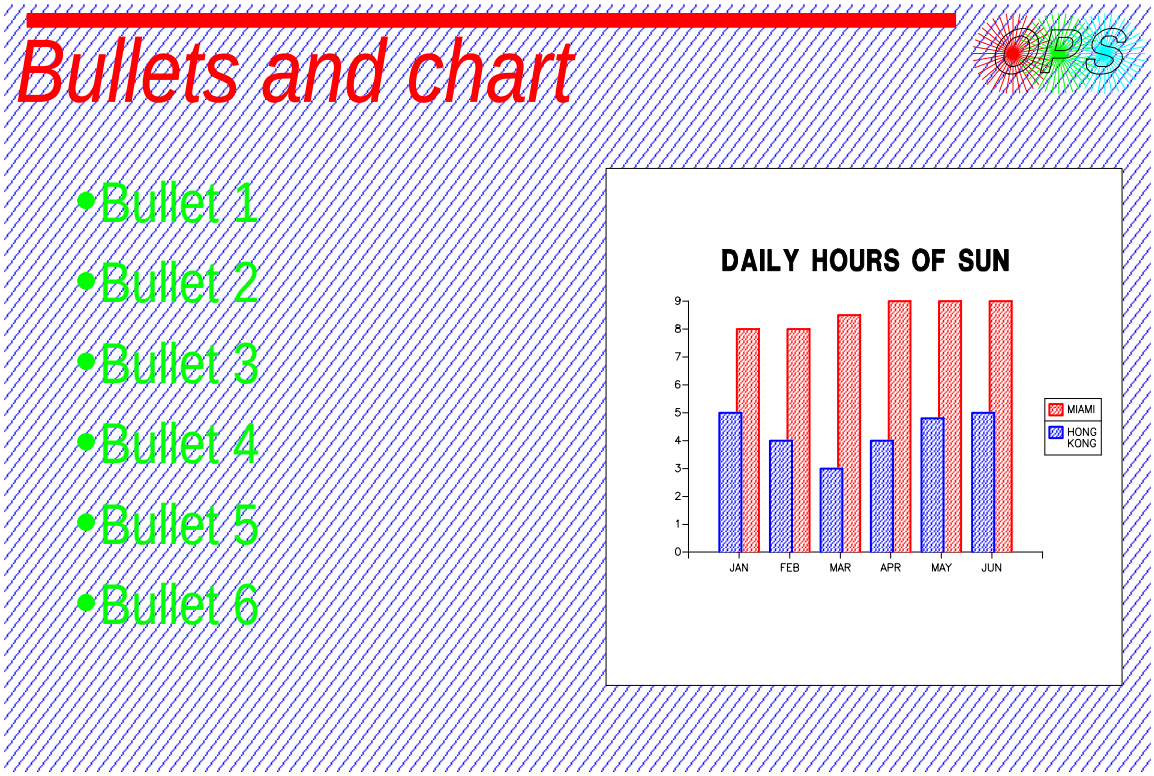


Figure 54. GDDM-OPS presentation (extracted using GDFSAVE)

OPS stopped being updated in 1993 as Microsoft PowerPoint displaced it (and other earlier presentation packages, like Harvard Graphics on MS-DOS), but it continued to be available in most GDDM installs (and is even still available to this day).

Usage requires either a real 3270 terminal with vector graphics support, or a 3270 terminal emulator with graphics (out of the scope of this document, but IBM Personal Communications i.e. PCOMM is ideal).

GDDM-ISE and GDDM-VSE: Not to be confused with the operating system of the same name, the Vector Symbol Editor is used to create custom vector symbols (i.e. vector fonts) and the Image Symbol Editor is used for creating bitmapped fonts. These can be used by user-written GDDM programs, GDDM-OPS, or GDDM-ICU.

SCRIPT/VS

SCRIPT is a program derived from one of the earliest programs ever written for a computer: *RUNOFF*. This was a program that ran on, of all platforms, CTSS; it provided a basic text formatting function that pagination and justification. The original RUNOFF was written in MAD in 1964; by 1967, it had been rewritten in BCPL and was bought over to Multics.

In 1968, at the peak of CP-67's development at Lincoln Labs, IBM contacted Stuart Madnick (at MIT) to produce a simple document formatter that would run under CP-67's CMS. This program was modelled by its author after the CTSS RUNOFF program (seeing that CTSS and its popular programs were a memory still hot on the mind of the

early CP-40 and CP-67 users/developers), and was compatible. Since this was the late 60s and early 70s, this version of SCRIPT was not terrifically useful; it supported typewriter terminal and line printer output only.[2]

This program was ported over to OS/VS TSO; this version was ported by William Dwyer (Yale University) in 1974. Shortly thereafter, several students at the University of Waterloo became disappointed by the lack of features afforded to them by this NSCRIPT program, and rewrote it as Waterloo SCRIPT (i.e. **WSCRIPT**).

WSCRIPT would continue to be updated, and worked on MVS, CMS, and MUSIC -- the MUSIC version was used to format the manuals for that operating system into PDFs (by way of outputting PostScript).

What is more interesting is IBM's development of SCRIPT. By 1978, their own rewrite of SCRIPT had been updated so substantially that it began to gain support for more and more line printers -- it had gained the ability to output a FORTRAN carriage control listing, which permitted the printing of various paper sizes, bold and underlined text, and strikeout text. This program was renamed to the *Document Composition Facility*, and it ran under the big three mainframe OSes (MVS, VM, VSE).

DCF supported the then-new IBM 3800 Printing Subsystem (which introduced a greatly-advanced printer data stream); in the mid-80s, DCF 3.0 gained support for the newly-introduced Advanced Function Printers (the most popular of which was the IBM 3820) which supported full graphics printing!

SCRIPT was used with several custom macro packages, but IBM put out two major ones:

- GML (Generalized Markup Language) Starter Set

  GML is an interesting language, because it is the direct source for SGML (Standardized GML) and, by extension, XML and HTML! The GML package that came with DCF was called the GML Starter Set, and it included a large library of macros for automating the production of various document elements (title pages, tables of contents, footnote lists, etc).

- BookMaster

  BookMaster, *of which this document is actually composed in*, is a superset of the GML Starter Set macros that may or may not be used in conjunction with **BookManager**, a program used to produce online books that could either be printed (of course, by running the input source straight through DCF/SCRIPT with BookMaster installed) or displayed online with a 3270 terminal! BookManager and BookMaster are technically two different programs, but are heavily related on account of this.

Over the years, IBM DCF/SCRIPT gained support for quite a number of output devices, including:

- Terminals: 3270 and 2741
- Line printers: 1403, 3800 (a laser printer, but treated like a line device in this mode)
- AFP page printers: 3800-3, 3820, 4250, 3812, 4224, 4028
- PostScript printers

All of these output devices had several variations for several paper sizes.

OfficeVision/VM

Note: before reading this, read the PROFS section in the VM chapter!

---

[2] Typewriter terminal output would have been done likely on an IBM 2741-style terminal; you would eject your cheap paper used for your login session, load some nicer paper, hit enter at the prompt after invoking SCRIPT that prompted you to load that paper, and wait on the printout to grow.

OV/VM 1.1 and 1.2: OfficeVision/VM was an evolution of PROFS, but intended to function alongside PCs. When OV/VM 1.1 was announced on May 16 of 1989, PCs running DOS (and sometimes OS/2) were in plentiful supply -- these usually ran some kind of host connection program like Communications Manager/2 or Personal Communications, but could just as easily run a front-end to OV/VM. The "OS/2 Office Feature" provided this feature on OS/2, and the DOS Office Direct Connect Feature provided it for DOS. OV/VM 1.1 included extended DisplayWrite/370 support, since it was presumed that this would be the dominant word processing system; it ended up indeed being the dominant document system for many businesses for quite a while!

PC users that needed to send a fax could use the Office Facsimile Application, but this program required an external fax server in order to send and receive the documents.[3]

OV/VM 1.1 released in December 1989, and OV/VM 1.2 released in March 1990. Throughout its life, OV/VM supported a number of extra programs:

- DisplayWrite/370, a full-screen word processor that used RFT-format files[4]
- AS (Application System), an application development and exeution engine sometimes used to write business applications in the era OV/VM was popular
- DCF (Document Composition Facility, i.e. SCRIPT), for formatting typeset documents without the usage of DisplayWrite/370
- ISPF/VM, used to display the OV/VM administration dialogs, some user panels (like the one used to generate author profiles), the DOS Office Direct Connect dialogs, the Control File Aids (settings editors), and the PROFS database split/merge/transfer functions.
- RSCS, used for networking OV/VM systems together
- Host-Displaywriter Document Interchange (HDDI), a program that converted DCF documents to and from RFT format.
- GDDM, if users wanted to store images in documents

OV/VM supported extensive inter-system communications with both RSCS support for email and remote calendars, as well as full APPC/VM support for remote calendars, mailboxes, and distribution managers.

---

[3] In this era (the late 80s), modems that could send and receive faxes were extremely rare and expensive; it would not be till the late 90s when every modem ever could do so

[4] RFT is Revisable Form Text, and is unrelated to Microsoft's Rich Text Format (RTF); many PC word processors (including WordPerfect, Microsoft Word, Ami Pro/Lotus Word Pro, etc) could load and save RFT files.

```
                        OfficeVision/VM Main Menu                          A0
 Press one of the following PF keys.
 PF1  Process calendars                                Time:   4:16 PM
 PF2  Open the mail
 PF3  Find documents                                 1926    JANUARY    1926
 PF4  Process notes and messages                    S   M   T   W   T   F   S
 PF5  Prepare documents                                                 1   2
 PF6  Process documents from other sources          3   4   5   6   7   8   9
 PF7  Process the mail log                         10  11  12  13  14  15  16
 PF8  Check the status of outgoing mail            17  18  19  20  21  22  23
                                                   24  25  26  27  28  29  30
PF10  Add an automatic reminder                    31        Day of Year: 011
PF11  View main menu number 2
      5684-084 (C) Copyright IBM Corp. 1983, 1989      PF9 Help    PF12 End
 ------------------------------------------------------------------------------




===>
```

Figure 55. OfficeVision/VM 1.1 Main Menu

```
                              PROCESS CALENDARS

 Calendar for:   System Administrator
                 ------------------------------------
 Calendar date:  01/11/26
                 ------------------                      Time:  9:09 PM

 Press one of the following PF keys.               2026    JANUARY    20
                                                  S   M   T   W   T   F
  PF1  Work with the day's schedule                                   1   2
  PF2  View  7 days of the calendar               4   5   6   7   8   9
           --                                    11  12  13  14  15  16
  PF3  View the conference room schedules        18  19  20  21  22  23
  PF4  Work with the next day's schedule         25  26  27  28  29  30
  PF5  Work with the previous day's schedule               Day of Year:
  PF6  View the month
  PF7  Schedule a meeting
  PF8  Print  7 days of the calendar
             --
 PF10  View calendar main menu number 2



 PF9 Help    PF12 Return
```

Figure 56. OV/VM 1.1 Calendar

```
                        LIST OF AVAILABLE DOCUMENT STYLES

  Press the PF key for the document style you want.

  PF1  STANDARD This is an RFT interoffice memo.

  PF2  DISTLIST This is an RFT memo beginning with "To:  Distribution".

  PF3  BUSINESS This is an RFT formal letter to send outside your company

  PF4  STANDDCF This is a DCF interoffice memo using the memo prompter.

  PF5  DISTDCF  This is a DCF memo beginning with "To:  Distribution".

  PF6  BUSDCF   This is a DCF formal letter to send outside your company.

  PF7  BLANK    This is an empty RFT format file.

  PF8  MEETING  This is a DCF interoffice meeting notice.


                                                              Screen   1 o
 PF9 Help    PF10 Next Screen    PF11 Previous Screen    PF12 Return
===>
```

Figure 57. OV/VM Document Style Choice Menu

```
 9        RFTD     A5                          PROMPT
  ** Prompt: Type addressee:
 ===>
 <---+_---1_---+_---2_---+_---3_---+V---4_---+_---5_---+_---6_---+_---7_>
 ------------------------------------------------------------------- Page
  January 11, 1926                          Draft   9
  P. R. ADMINISTRATOR , 222-222-2222
  LARGE CO.
  TECHNICAL
  BUILDING 55
  WASHINGTON, DC 222222
  SYSADMIN/VMESA12


 Memo to:

 Subject:

 Reference:



 PF 1=Block      2=Insert     3=Cmdline    4=Instr.     5=Tspell     6=Ai
 PF 7=Next       8=Command    9=HELP      10=Forward   11=Backward  12=EN
```

Figure 58. OV/VM DisplayWrite/370 Memo Prompter

OV/VM 1.2 (April 24, 1992) added some pretty serious improvements; the most meaningful improvement was the introduction of the new OVMAIL facility, a totally-rewritten user interface that greatly increased user productivity with emails.

```
ML00                          OfficeVision/VM Mail            Items 1 to

Type / or an action next to an item and press Prompt or ENTER.
      --Name-- --Type-- M- --Blocks-- --Date-- ---------Description------
_____ In-basket

_____ Non-OV/VM Mail

_____ Document Log




Personal Storage is 01% full (8 of 900 blocks).
Command ===>
F1=Help  F2=Keys  F3=Exit  F4=Prompt  F5=Refresh  F6=TopBot  F7=Backward
F8=Forward  F9=Retrieve  F10=Sort  F11=Find  F12=Cancel
```

Figure 59. OV/VM 1.2 OVMAIL Menu

ESA Calendar Feature: Before OfficeVision/VM Release 3 was announced and made available, an interesting enhancement was made available in October 1993: the *ESA Calendar Feature*. This was a massive upgrade to the earlier and more primitive OV/VM calendar (which had not meaningfully evolved since PROFS 2.11)! The product announcement provides a rather quaint list of enhancements this new calendar had:

- Improved calendar performance and capacity, especially related to the calendar search function; it was now handled client-side (on the user virtual machine) in lieu of deferring the search operation to the calendar manager service machine
- Exploitation of VM Data Spaces, allowing for faster calendar operation if all of the calendar servers in a distributed arrangement (under one VM system) run with XC-mode CMS
- Usage of 31-bit CMS addressing, permitting for calendars that may spill over 15 MB
- Ability to use SFS for calendar storage, previously not possible with the earlier calendar
- Allows for distributed calendar processing across several calendar servers
- The protocol between the client and the server is now fully documented, allowing for all manner of custom applications to be written
- Supports a laundry list of network protocols:
  - APPC/VM
  - IUCV
  - RSCS
  - CPI Communications
- Backwards-compatible with earlier user-written (in assembler) OV/VM calendar exits

One might ask why my description of this ESA Calendar Feature is so lengthy, and it is due to the obvious amount of work that was placed into it: it is clear that a skunkworks group made this highly-advanced program!

```
 CW00                          Process Calendars

Calendar for  W. E. C_______________________________
Date. . . . . 01/11/26  Sunday

Select one of the following.  Then press Enter.         Time:  8:58 PM EST

 1_  1. Work with the day's schedule               2026      January     2026
     2. View 7_  days of the calendar               S   M   T   W   T   F   S
     3. View the conference room schedules                          1   2   3
     4. Work with the next day's schedule           4   5   6   7   8   9  10
     5. Work with the previous day's schedule      11  12  13  14  15  16  17
     6. View the month                             18  19  20  21  22  23  24
     7. Schedule a meeting                         25  26  27  28  29  30  31
     8. Print 7_  days of the calendar                        Day of Year: 11
     9. View the business week
    10. View six month calendar
    11. Manage list of authorized users
    12. Search facility
    13. Add company holidays

CCL000I (C) Copyright IBM Corporation 1993. All Rights Reserved
Command ===>
F1=Help  F2=Options  F3=Exit  F10=Next month  F11=Previous month  F12=Cancel
```

Figure 60. OfficeVision/VM 1.3 ESA Calendar Feature

CallUp: An aptly-named program, CallUp was a phone directory program that was very often seen alongside OV/VM. Some of the very large OV/VM systems made heavy use of CallUp, and IBM themselves used it for far longer than most people would consider its useful life to be!

```
  File  Distribute  Inquire  Options  Help
_____________________________________________________________________________
                            Search Request

To request a search, type the search data and press Enter.
                                                          Lines  1 to 14
               Corporate and Personal Directories
                 Name  . . . . . .
                 Search Location(s)  . .   =SAMPLE
                 Department  . . . . . .
                 User ID . . . . . .
                 Node ID . . . . . .

                 Manager . . . . . .     1. Yes
                                         2. No

                 Telephone Extension . .
                 Additional Data . . . .

               Services Directory
(C) Copyright IBM Corporation 1988, 1992.  All rights reserved.
Command ===>
F1=Help  F2=Set 2  F3=Exit  F4=Profile  F5=Refresh  F6=Fuzzy search
F7=Backward  F8=Forward  F9=Command  F10=Actions  F11=Dist list  F12=Can
```

Figure 61. CallUp Main Menu

Release 3, 4, and the end: OV/VM 1.3 was announced on June 11, 1996, and included an updated ESA Calendar Feature. This release was not very complex, but the following things were added:

- Usage of VMSES/E
- Retention Management System (old documents can now be auto-purged)
- Support for remote in-baskets and remote conference room scheduling added
- Added SMTP support
- Proofreading added[5]
- Time zone support added to the ESA Calendar Feature
- OVMAIL facility now integrated into the base

OV/VM Release 4 (announced November 18, 1997) was the final version of the PROFS line, and the stated purpose of this release was full Y2K compliance; in addition, numerable changes and enhancements were added:

- The OV/VM user code saved segment is now relocatable above 16 MB, allowing for the base OV/VM program (and not just the ESA Calendar Feature) to feature full 31-bit addressing
- The user A-disks can now be in SFS (somehow, this was not permitted with older OV/VM releases); filesystem permissions are fully integrated too
- The database/calendar/mailbox/distribution managers can now run arbitrary CMS commands from their consoles
- MIME emails are now supported when received through SMTP
- Users can now postpone (and save) a note, to work on it later

[5] The OV/VM proofreader is derived from an IBM program that was never officially released called PROOF

- The NOTIFY command has been improved
- The RECALL command has been improved (this command allows you to un-send mail)
- The OVMAIL note editor has been rewritten and substantially improved

Standalone Utility Programs

Throughout the history of mainframes, there were several notable standalone utilities used for maintaining the system. Here is a selection of some of them:

Device Support Facilities (ICKDSF): Device Support Facilities (ICK is just the prefix) is a program that is used to format and initialize DASDs. This program ran under MVS, VM, VSE, AIX/370/ESA, and possibly others; Linux, Solaris, MUSIC, MTS, and TPF users would have to run the standalone version described here. This was normally loaded from a tape, and it would wait for an interrupt from some kind of console/terminal device after IPLing. You would do that by pushing an attention key on a 3270, 3215, or possibly others. Once the messages appeared, you could use it.

ICKDSF was sometimes wrapped by a program called CPFMTXA on VM; in the early days, this program was called Format/Allocate.

```
ICK005E DEFINE INPUT DEVICE, REPLY 'DDDD,CUU' OR 'CONSOLE'
ENTER INPUT/COMMAND:


ICK006E DEFINE OUTPUT DEVICE, REPLY 'DDDD,CUU' OR 'CONSOLE'
ENTER INPUT/COMMAND:


ICKDSF - SA      DEVICE SUPPORT FACILITIES  11.0B                TIME: 1
        11/20/25     PAGE   1

ENTER INPUT/COMMAND:
end
 END
```

```
                                                        VM READ   VMSP5
```

Figure 62. Standalone ICKDSF

VM Format/Allocate (DMKFMT): This program was used in the VM/370 era to, after initializing DASDs with ICKDSF, would lay out the disks for VM CP. You would allocate the following kinds of space with it:

- PERM: used to store minidisks
- DRCT: used to store the system directory
- PAGE: paging space
- SPOL: spooling space

- PARM: used to hold parm disks (VM/XA only, CPFMTXA)

This program was eventually replaced by a successor seen on VM/XA: CPFMTXA. This program was ran from CMS, and did not run standalone; if you needed to perform these functions standalone, you would actually use ICKDSF's CPVOL command!

In the VM/370 era, you would run this program (most likely) by IPLing it from a virtual machine's card reader with the DASD you wished to format attached to said VM.

VM Standalone Directory Creation Utility (DMKDIR): This program does exactly what the name implies: it loads from a card reader, and is followed by a deck containing a VM directory. This would be used during a VM/370 or VM/SP tailored installation procedure or cross-system generation, which would involve and necessitate the creation of a system directory. The filename was “IPL DIR”

Standalone I/O Configuration Program (SA IOCP): While not strictly a program one might would load from a reader or something similar, the SA IOCP program is critical to any mainframe. Since mainframes since the System/370 era (this list would include the 370/XA machines) no longer use a hardwired logic method of I/O configuration and channel assignment, it became necessary to define some kind of I/O configuration file. Naturally, these looked something like this:

```
CHPID PATH=04,TYPE=CNC
CTLUNIT CUNUMBR=400,PATH=(04),UNIT=3990,UNITADD=((00,256),CUADD=0
IODEVICE ADDRESS=(400,16),CUNUMBR=400,UNIT=3390,UNITADD=0,STADET=Y
```

Figure 63. Sample IOCP definitions for a 3390

This snippet here does the following things per each line:

1. Defines an ESCON (CNC) channel on channel path ID 4.
2. Defines a 3990 (CKD DASD) control unit on CHPID 4 with 256 maximum devices attached to it. This CU is at address 400.
3. Defines 16 3390 DASDs, starting at device address 400 (technically unrelated to the CTLUNIT's CU number), attached to the controller with CU number 400 (that's the DEVICE -> CTLUNIT linkage).

In order to "cook" this into a working IOCDS dataset that the mainframe CPU loads during the power-on-reset phase of the IPL, one must either run the IOCP programs on the various OSes (IOCP programs are known to have existed for MVS/ESA, VM/ESA, VSE/ESA, and AIX/ESA, as well as their more modern counterparts), *or* run the standalone IOCP!

The SA IOCP program is not loaded like a normal standalone program might be (like from a reader or a tape), but instead loaded directly from the SE. Under CPC Customization, there is an option for configuring I/O devices -- dragging the group icon onto that icon will open a dialog showing several options of known IOCDS datasets and IOCS sources (an IOCDS is a “processed” version of an IOCS source file) and the option to edit an IOCS is provided. After an edit, you would choose the option from the menu at the top left to build an IOCDS dataset. Once that is done, the mainframe can be re-powered-on with the new IOCDS (since you are prompted which one you want to load during the power-on procedure). The SA IOCP program itself does run on the S/390 or System Z CPU, but is stored as a memory image.

DASD Dump/Restore (DMKDDR): DDR is one of the classic programs seen from VM, and is often used by systems programmers to migrate DASDs between system by way of first dumping them on tape. DDR is a simple program that asks for an input and output device -- these devices can be disks or tapes. It can also copy DASDs by specifying two DASDs for the input and output devices. Otherwise, it will dump to tape or restore from tape.

The tape format can be one of the following options, all similarly-named:

- Mode Compacted (MODE COMP), wherein compression is done by the tape drive itself; this is the 3480 IDRC or 3490E IDRC enablement option, but works for other tapes. This is specified on the output device like "OUTPUT 590 3490 (MODE COMP."
- Compacted, wherein DDR does software compression, specified before any MODE options with the COMP.
- LZMA compaction, wherein DDR uses the LZMA compression instruction of an S/390 G6 or newer -- this is specified with the LZCOMP option.

```
ddr
VM/ENTERPRISE SYSTEMS ARCHITECTURE DASD DUMP/RESTORE PROGRAM
ENTER:
input 100 dasd
ENTER:
output 200 dasd
ENTER:
copy all
HCPDDR716D NO VOL1 LABEL FOUND
DO YOU WISH TO CONTINUE?  RESPOND YES, NO OR REREAD:
yes
HCPDDR716D NO VOL1 LABEL FOUND
DO YOU WISH TO CONTINUE?  RESPOND YES, NO OR REREAD:
yes
COPYING
END OF COPY
ENTER:

END OF JOB
PRT FILE 0220 SENT FROM WEC      PRT WAS 0220 RECS 0006 CPY  001 A NOHOLD
Ready; T=0.01/0.10 10:50:51


                                                          VM READ   EVIEVM
```

Figure 64. Example DDR run (copying a DASD)

OS DASD Initialization (IBCDASDI): Hailing from the OS/360 era, IBCDASDI was a much more cumbersome way of doing tasks that would later be performed by ICKDSF. The input jobstreams were rather convoluted:

```
DINIT1 JOB 'INITIALIZE AN EXAMPLE DASD UNDER VM'
       MSG TODEV=1052,TOADDR=009
       DADEF TODEV=3380,TOADDR=100,IPL=NO,VOLID=TEST01,BYPASS=YES
       VLD NEWVOLID=TEST01,OWNER=DEMO
       VTOCD STRTADR=1,EXTENT=5
       END
```

Figure 65. Example IBCDASDI jobstream

Naturally, one would need to IPL this program and tell it which input device to use with a terminal input like "input=1442,00c." After that, the program will read the above jobstream from the reader, and the initialization will occur (hopefully).

OS Dump/Restore (IBCDMPRS): This program is very similar to the above IBCDASDI program,DMPRS was always ran by MVS systems programmers to restore a starter system from tape to a temporary DASD such that they could build the system. The input jobstreams looked like this:

```
DASD1  JOB 'RESTORE MVS STARTER SYSTEM SYSRES PACK'
       MSG TODEV=1052,TOADDR=009
       RESTORE FROMDEV=3400,FROMADDR=580,TODEV=3330,TOADDR=150,
               VOLID=MVSRES
       END
```

Figure 66. Example IBCDMPRS jobstream

Standalone Utilities (ZZSA): ZZSA is a program written by Jan Jaeger that provides an emergency rescue environment and standalone fullscreen editor -- the only other real option for shops that broke their MVS install is to be running it under VM, wherein they can use CMS's OS simulation to edit a probably-erroring PARMLIB member. This program could IPL from a DASD, tape, reader, or VM saved segment (which is how the below screenshots were captured), wherein it would present a 3270 user interface and an editor that is not unlike that of ISPF's. With this, users could hopefully un-brick their mainframes.

```
 ZZSAPRIM                    Stand Alone Utilities

 Option ===>


     0 ListDev  - List all devices

     1 Browse   - Browse dataset or member               Console     000

     2 Edit     - Edit dataset or member                 IPL Device  ???

     3 ListVTOC - List Volume Table Of Contents          IPL CPU     000

     4 ListPDS  - List PDS directory                     CPU Version FF

     5 DispVol  - Display DASD volume label              CPU Serial  010

     6 Dump     - Dump DASD record by CCHHR              CPU Model   706

     7 Zap      - Alter DASD record by CCHHR             Date (TOD)  22/

     X Exit     - Terminate program                      Time (TOD)  16:

                                          Jan Jaeger - Version 12/07/9
```

Figure 67. ZZSA main menu

```
 ZZSABROW Device List
 Command ===>                                                 Line 0000 Co
******************************** Top of Data ***************************
SCH=0000 DEV=0009 CHP=00                      C/T=3274-1D
SCH=0001 DEV=000C CHP=00
SCH=0002 DEV=000D CHP=00
SCH=0003 DEV=000E CHP=00
SCH=0004 DEV=0190 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=0005 DEV=0191 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=0006 DEV=0193 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=0007 DEV=019D CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=0008 DEV=019E CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=0009 DEV=0370 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=000A DEV=0CF1 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=000B DEV=0CF2 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=000C DEV=0123 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=000D DEV=0124 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=000E DEV=0125 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=000F DEV=0126 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=0010 DEV=0127 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=0011 DEV=0128 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=0012 DEV=0129 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
SCH=0013 DEV=0194 CHP=FD                      C/T=3990-E2 D/T=3390-0A VOL
           F3=End     F5=RFind         F7=Up F8=Down  F10=Left F11=Right
```

Figure 68. ZZSA device list

```
 ZZSABROW VTOC listing VSAM01(0100)
 Command ===>                                                 Line 0000 Co
******************************** Top of Data ***************************
Z9999992.VSAMDSPC.TE1F9DC4.T9B02555          DSORG=VS   RECFM=???? BLKSIZ
Z9999996.VSAMDSPC.TE1F9DBB.TED3F3A5          DSORG=VS   RECFM=???? BLKSIZ
******************************** Bottom of Data ************************
```

```
           F3=End     F5=RFind         F7=Up F8=Down  F10=Left F11=Right
```

Figure 69. ZZSA dataset list

Alas, no mainframe is without it!

Afterword

I have composed this to, the best of my ability, to be as accurate as I can make it. I have composed this since the 2nd of August to be the ultimate reference or the mainframe hobbyist community; I hope that this is sufficient and brings people some fun and insightful knowledge!

Index

Numerics

B

C

D

E

G

I

L

M

O

P

R

S

T

U

V

Z

Glossary

Linux/i370. See *Bigfoot Linux*.

OSA. Open Systems Adapter, a network interface card for System/390 and System Z systems.

ICA. The Integrated Communications Adapter found on the ES/9000 and emulated on the P/390, IS3006, and MP3000 systems that provides WAN interfaces on SDLC or BSC ports.

SDLC. Synchronous Data Link Control, a synchronous-serial WAN interface similar to X.25's HDLC.

X.25. An obsolecent WAN network protocol family common in the era of mainframes.

Token Ring. An obsolecent networking technology pushed by IBM that used a ring network topology to form a LAN, as opposed to Ethernet (which forms a bus topology)

FDDI. Fiber Distributed Data Interface, a mixed WAN/LAN high-speed ring network that was popular in the 1990s in large systems. Requiring two fiber pairs and nothing more, some FDDI rings spanned entire continents.

ATM. Asynchronous Transfer Mode, a WAN network protocol stack that was inspired by the virtual-circuit-switched protocols that came before it (like X.25).

ATM LANE. ATM LAN Emulation, used to simulate an Ethernet network over an ATM virtual circuit. Supported on the OSA cards.

LEXX. The Live Parsing Editor, originally written to edit GML documents, but later enhanced with other languages. The first syntax-highlighting editor, its influence on modern IDEs cannot be forgotten.

DAT (tape). A 4mm helical-scan tape format intended for audio storage, but later reused by computers under the name DDS (Digital Data Storage). The Multiprise 3000 and S/390 Integrated Server (as well as being possible on a P/390) had a DAT drive that showed up to the host as a 3480 cartridge tape drive.

OMA. Optical Media Attach, a scheme in which a virtual tape is assembled from files on a CD. IBM provided a program called Optical Media Attach/2 that ran on OS/2, and by means of a channel adapter card (either parallel or ESCON), a fake tape drive appeared to the host. Used to distribute OS install media, programs, and documentation.

SSA. Serial Storage Architecture, the architectural precursor to SATA and SAS. IBM pushed this technology in the late 90s and early 2000s to decent success, and it featured good performance. Though the SSA RAID cards had a rather obtuse user interface, they were often better than the competition. The IS/3006 and MP3000 both used SSA.

IS/3006. System/390 Integrated Server model 3006.

MP3000. Multiprise 3000.

Shark. IBM Enterprise System Storage (ESS) DASD array.

VTL. A Virtual Tape Library, often seen attached to modern mainframes, emulating tape drives with disk files; used for backup.

Bustech. A DASD emulation product that was available in the early 2000s; acquired by Dell and then later killed.

RJE. Remote Job Entry, a remote batch job and output retrieval system that used batch terminals armed with printers, card readers, and sometimes card punches

NJE. Network Job Entry, an upgraded RJE that supported file transfer and command execution

ALCS. AirLine Control System, a transaction monitor that runs on MVS. Versions include 2.3.1 (running on OS/390 2.10 and z/OS 1.1) and 2.4.1 (running on z/OS 1.10). Distinct from CICS.

Personal Communications (PCOMM). IBM's 3270 and 5250 terminal emulator for PCs, also supports VT emulation (as in, the DEC VT100 and VT220) as a side-effect of needing to be able to do so to properly emulate 3270 NVT mode.

```
*** .EDF#INIT      500  13 <snap @zero@ls>
***
*** NAME         (INDEX) LCL AREA SIZE <VALUE>
***
*** &@zero@ls                 48     1 <0>
***
STARTING PASS 2 OF 3.
STARTING PASS 3 OF 3.
```