Role of Operating System

- Operating systems are designed to provide uniform abstraction across multiple applications: fair sharing of resources
- General purpose OS like Solaris in wizard.cse.nd.edu
  - Mail, web, samba server, telnet, emacs …
  - Memory fs, afs, ufs …
  - Fibre channel devices, floppy disks …
- What about applications/services such as video games, data base servers, mail servers
  - OS gets in the way of these applications in the name of fairness (MSDOS is the ideal OS!!)

What is the role of OS?

- Create multiple virtual machines that each user can control all to themselves
  - IBM 360/370 …
- Monolithic kernel: Linux
  - One kernel provides all services.
  - New paradigms are harder to implement
  - May not be optimal for any one application
- Microkernel: Mach
  - Microkernel provides minimal service
  - Application servers provide OS functionality
- Nanokernel: OS is implemented as application level libraries

Case study: Multics

- Goal: Develop a convenient, interactive, useable time shared computer system that could support many users.
  - Bell Labs and GE in 1965 joined an effort underway at MIT (CTSS) on Multics (Multiplexed Information and Computing Service) mainframe timesharing system.

- Multics was designed to the swiss army knife of OS
- Multics achieved most of the these goals, but it took a long time
  - One of the negative contribution was the development of simple yet powerful abstractions (UNIX)

Multics: Designed to be the ultimate OS

- "One of the overall design goals is to create a computing system which is capable of meeting almost all of the present and near-future requirements of a large computer utility. Such systems must run continuously and reliably 7 days a week, 24 hours a day in a way similar to telephone or power systems, and must be capable of meeting wide service demands: from multiple man-machine interaction to the sequential processing of absentee-user jobs; from the use of the system with dedicated languages and subsystems to the programming of the system itself; and from centralized bulk card, tape, and printer facilities to remotely located terminals. Such information processing and communication systems are believed to be essential for the future growth of computer use in business, in industry, in government and in scientific laboratories as well as stimulating applications which would be otherwise undone."

Contributions

- Segmented memory
  - The Multics memory architecture divides memory into segments. Each segment has addresses from 0 to 256K words (1 MB). The file system is integrated with the memory access system so that programs access files by making memory references.
- Virtual memory
  - Multics uses paged memory in the manner pioneered by the Atlas system. Addresses generated by the CPU are translated by hardware from a virtual address to a real address. A hierarchical three-level scheme, using main storage, paging device, and disk, provides transparent access to the virtual memory.
- High-level language implementation
  - Multics was written in the PL/I language, which was, in 1965, a new proposal by IBM. Only a small part of the operating system was implemented in assembly language. Writing an OS in a high-level language was a radical idea at the time.

Contributions (cont)

- Shared memory multiprocessor
  - The Multics hardware architecture supports multiple CPUs sharing the same physical memory. All processors are equivalent.
- Multi-language support
  - In addition to PL/I, Multics supports BCPL, BASIC, APL, FORTRAN, LISP, C, COBOL, ALGOL 68 and Pascal. Routines in these languages can call each other.
- Relational database
  - Multics provided the first commercial relational database product, the Multics Relational Data Store (MRDS), in 1978.
- Security
  - Multics was designed to be secure from the beginning. In the 1980s, the system was awarded the B2 security rating by the US government NCSC, the first (and for years only) system to get a B2 rating.

Contributions (cont.)

- On-line reconfiguration
  - As part of the computer utility orientation, Multics was designed to be able to run 7 days a week, 24 hours a day. CPUs, memory, I/O controllers, and disk drives can be added to and removed from the system configuration while the system is running.
- Software Engineering
  - The development team spent a lot of effort finding ways to build the system in a disciplined way. The Multics System Programmer's Manual (MSPM) was written before implementation started: it was 3000 or so pages and filled about 4 feet of shelf space in looseleaf binders. (Clingen and Corbató mention that we couldn't have built the system without the invention of the photocopier.) High level language, design and code review, structured programming, modularization and layering were all employed extensively to manage the complexity of the system, which was one of the largest software development efforts of its day.

Legacy - Positive and negative

- UNIX:
  - Ken Thompson and Dennis Ritchie, the inventors of UNIX, worked on Multics until Bell Labs dropped out of the Multics development effort in 1969. The UNIX system's name is a pun on Multics attributed to Brian Kernighan. Some ideas in Multics were developed further in UNIX.
- GCOS 6
  - Honeywell's GCOS 6 operating system for the Level 6 minicomputers was strongly influenced by Multics.
- Primos
  - Prime's Primos operating system shows a strong Multics influence. Bill Poduska worked on Multics at MIT before founding Prime, and several other senior Multicians worked at Prime. Poduska referred to Primos as "Multics in a shoebox."

Legacy

- VOS
  - Stratus's VOS operating system shows a strong Multics influence. Bob Freiburghouse, former Multics languages manager, was one of the founders of Stratus; many Multicians are still Stratus employees.
- Apollo Domain
  - Bill Poduska went on from Prime to help found Apollo, and Domain was known as "Multics in a Matchbox." Apollo's OS shows strong Multics influence. For instance, the basic access to stuff on disk is via a single-level store directly based on Multics. Supposedly some of the motivation for the object-store style of file system came from Multics too. (Info from Frederick Roeber) [Jerry Saltzer adds:] In addition, it uses a shared memory model, despite being distributed across a network
- NTT DIPS
  - NTT undertook a massive effort to clone Multics, which led to their DIPS (Denden Information Processing System) series of mainframes. DIPS machines are still in widespread use in Japan today by NTT, but everyone agrees that they are going away. I believe that Intermetrics developed the DIPS PL/I compiler for NTT.

Legacy

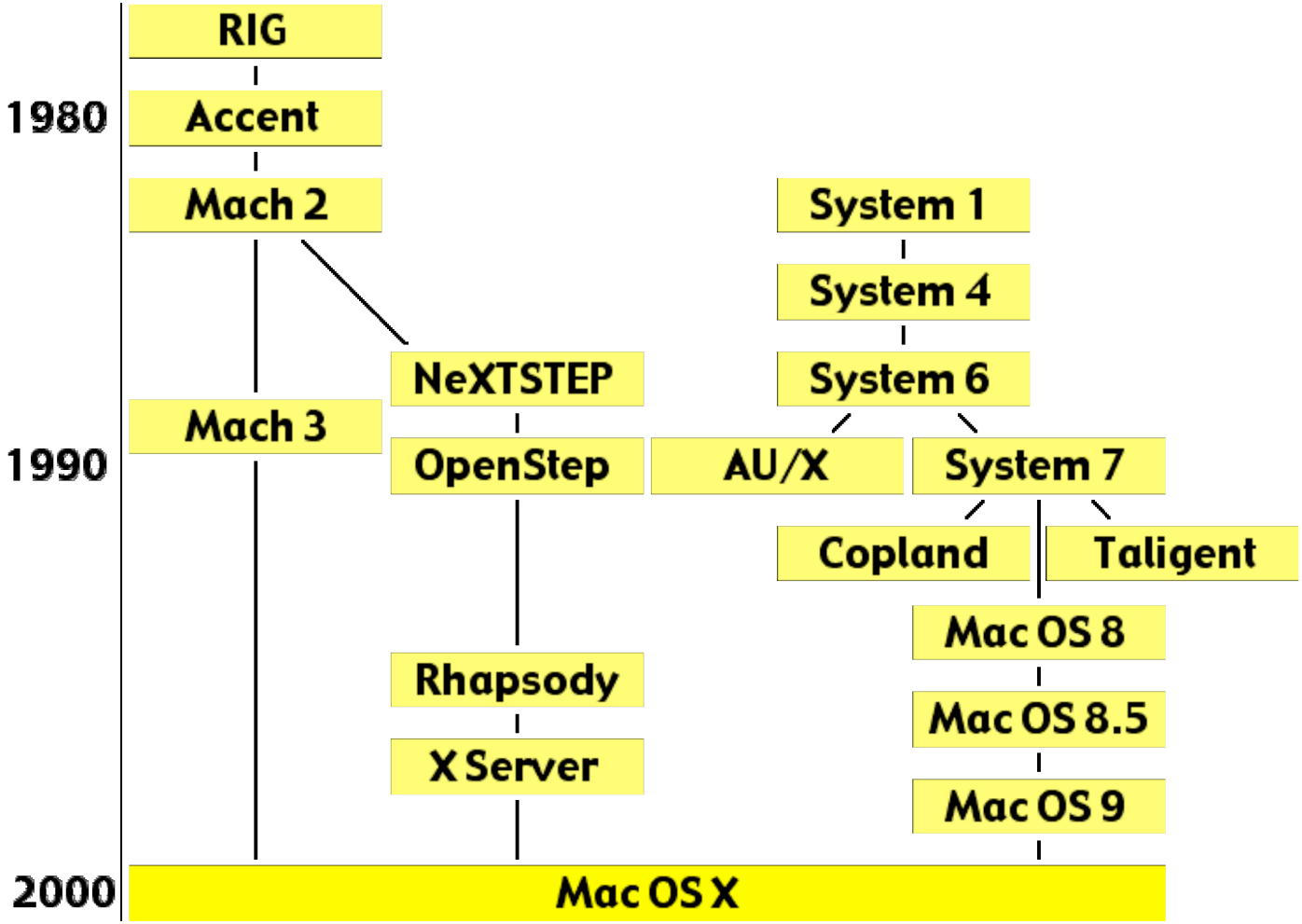
- Amber
- IBM System 360
- Tenex, TOPS 20, GCOS etc etc

- Most of the the project members were influential in shaping the computer industry and they took Multics ideas with them

- Monolithic vs microkernel debate

Mach

- Microkernel



- Courtesy: J*onathan 'Wolf' Rentzsch*

Mach abstractions

- Microkernel only provide code mechanisms. Policies implemented in OS servers (BSD, and others)
- Tasks
  - Resource ownership
- Threads
  - Scheduling and preemption
- Virtual Memory (Memory Objects)
    - Memory protection and management•
- Task-to-Task Communication (Ports)
  - Controlled interaction

Exokernel

- If you use the right library (basically customize the OS for each application) then you can get good performance

- What about:
  - Version control (which library should I use?)
  - General purpose usage setup: do they benefit?
  - Effort duplication among applications?

- What is the right approach for next generation OS?