Introduction to Operating Systems

COMP755 Advanced OS

What does an Operating System do?

- An Operating System provides a logical environment for using the computer.
- If you are using a system with Linux *(or Windows or Unix or Solaris or Android or whatever)*, it doesn't matter what hardware you are using, it still runs the same.
- Users work with logical concepts instead of physical hardware.

Mapping of Physical to Virtual

Virtual	Physical
file	blocks on the disk
large address space	limited RAM
single user	many users
named devices	hardware addresses
display windows	bit mapped display
printer	printer (any kind)

Purpose of an OS

- A program that controls the execution of application programs
- An interface between the user or applications and the hardware
- Masks the details of the hardware
- Allocates resources to programs

Functions of an OS

- **Compatibility**
  The OS provides a virtual machine allowing programs to run on a variety of machines.
- **Convenience**
  Provides an easy to use interface.
- **Efficiency**
  Allows resources to be used efficiently.
- **Security**

  Restrict access to resources

Services of an Operating System

- Sharing
- Program execution
- Controlled access to files and I/O devices
- System access and security
- Error detection and recovery
- Hardware control

OS Interface

- The user interface can be a simple command line or a highly functional GUI
- There is a programmatic interface to the OS
- The API provides many functions to perform just about everything a program does except computation
- Programming language libraries call the OS API to perform much of their functionality

Just a Program

- The operating system is a computer program.
- Most operating systems are large (sometimes very, very large) programs.
- Operating systems can execute privileged instructions to control the hardware.
- Operating system relinquishes control of the processor to execute other programs

History of Operating Systems

Who Cares?

This is **not** a history class.

- Operating systems are shaped by the path taken to get here.

"I think there is a world market for about five computers."

Thomas J. Watson (1945)

president of IBM

Hardware Leads the OS

- Because of the better performance of modern computers, the OS can provide more services.
- Older computers didn’t have the speed or storage to run the OS of today.

The Evolution of Operating Systems

- No Operating Systems
- Monitors
- Simple Batch Systems
- Multiprogrammed Batch Systems
- Time Sharing Systems

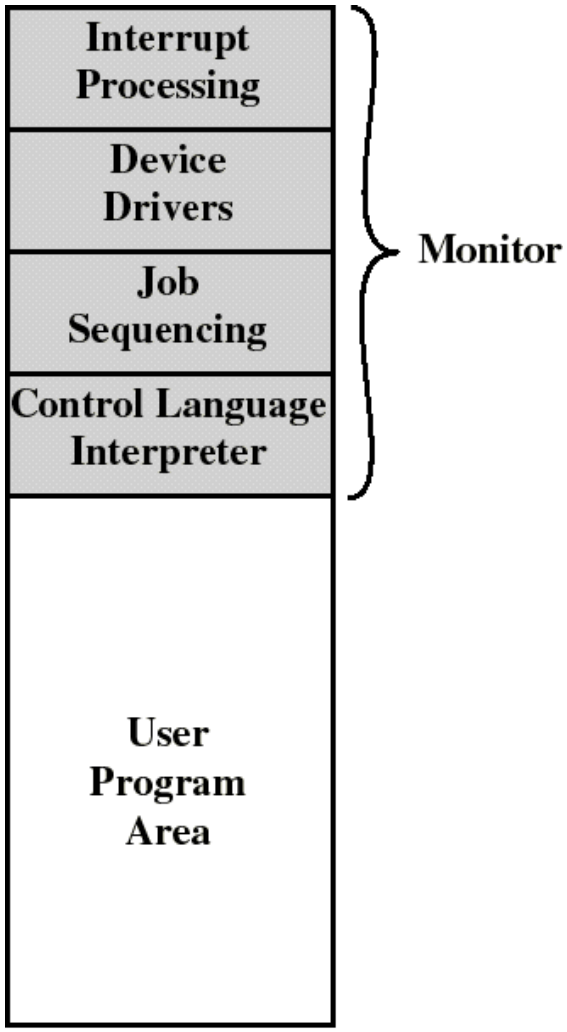
No Operating Systems

- Each program directly interfaced with the hardware.
- One person used the computer at a time.
- “Job Scheduling” was done with a clipboard
- Libraries of commonly used procedures were the first start of operating systems.

Monitor Programs

- The monitor was a program that loaded application programs into RAM.
- The monitor, or a small portion of it, remained in RAM while the application program ran.
- The monitor contained device drivers to simplify access to peripherals.
- When the program terminated, it would jump back to the monitor *(or not)*.

- The monitor loads a job into the User Program Area
- A monitor instruction branches to the start of the user program
- When the user program is finished, the CPU fetches instructions from the Monitor



Memory Layout of Resident Monitor

Simple Batch Systems

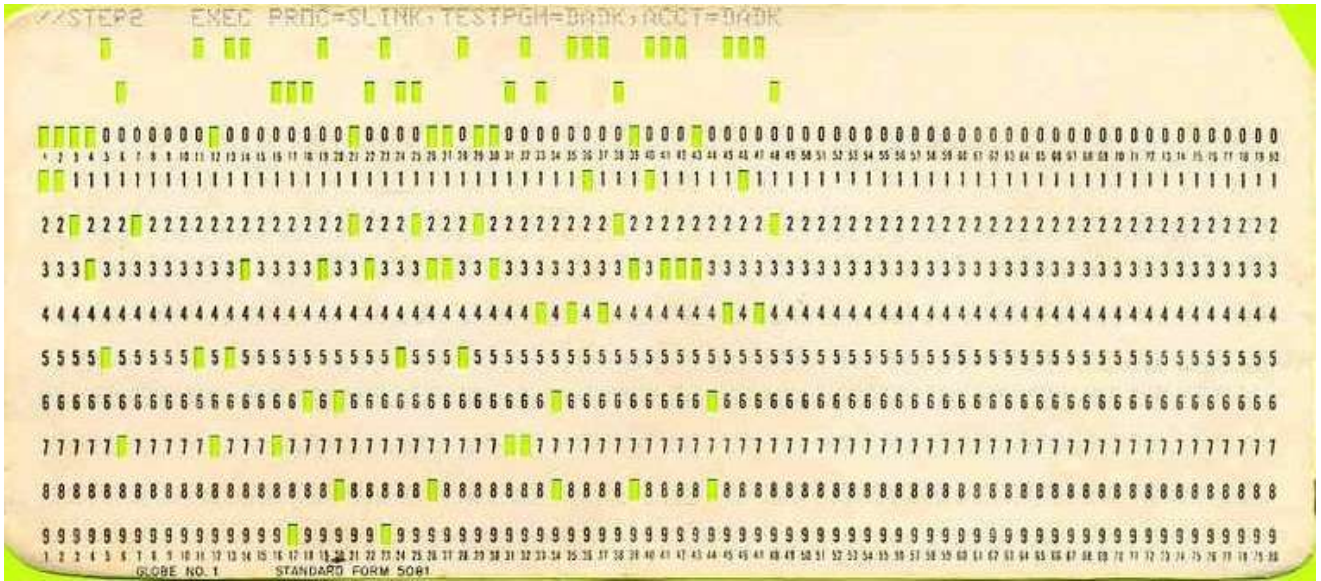
- The user submits a job (written on punched cards) to the computer.
- The operating system would copy the input data to a disk.
- When a job completed, the OS would select one of the jobs from the disk and run it.
- Printed output from the job was written to the disk. When the program terminated, the output file was copied to the printer.

Job Control Language (JCL)

- JCL was used to specify commands to the operating system
- The first characters of the input card identified it as JCL or data
- The JCL specified what program was to be run or what data file was to be used

JCL Example

```
//KENJOB   RUN ACCOUNT=COMP755
//    DD   DDNAME=*
data cards
//    EXEC   PROC=WILLIAMS.PROG
//
```



Multiprogrammed Batch Systems

- Several programs had to be kept in RAM at the same time, each protected from the other.
- The OS had to be able to switch from one user environment to another.
- Relied on hardware that supports I/O interrupts and DMA

OS Security

- If there is only one user of the computer and it has no network connection, security is a very minor concern
- Multiple concurrent programs require the OS to keep each program separate
- Different users requires access permissions

Software Engineering

- OS/360 was one of the first large (over 1000 programmers) software projects
- Fred Brooks claims in The Mythical Man Month that he made a multi-million dollar mistake of not developing a coherent architecture before starting development

Wikipedia, History of SE

Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor’s time is shared among multiple users
- Multiple users simultaneously access the system through terminals
- A new control language was required for interactive work

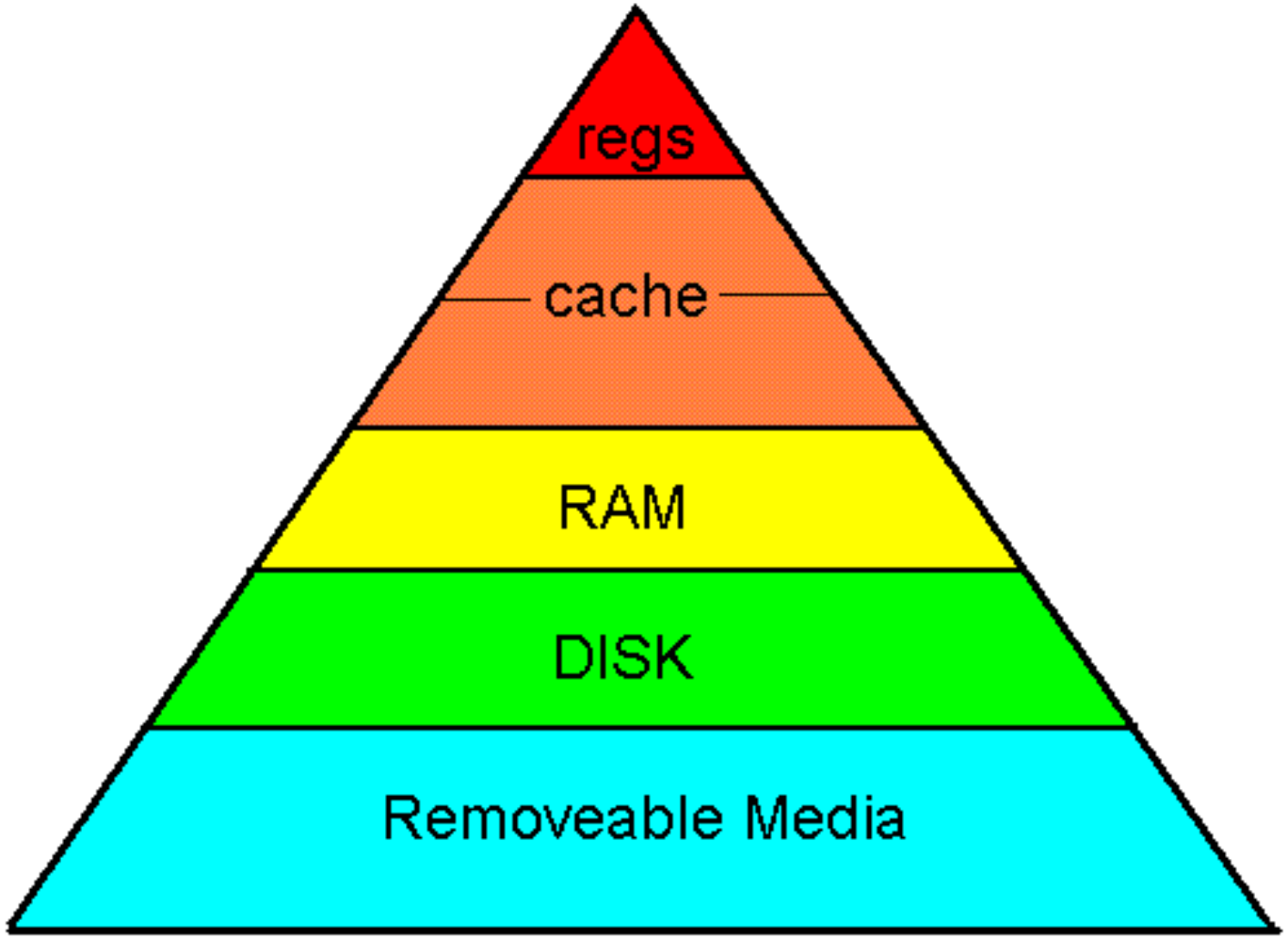
Memory Organization

- Early computers did not have cache or Virtual Memory. Cache has little impact on the OS.
- Some early machines had two types of RAM. The OS moved jobs between the fast and slow RAM.

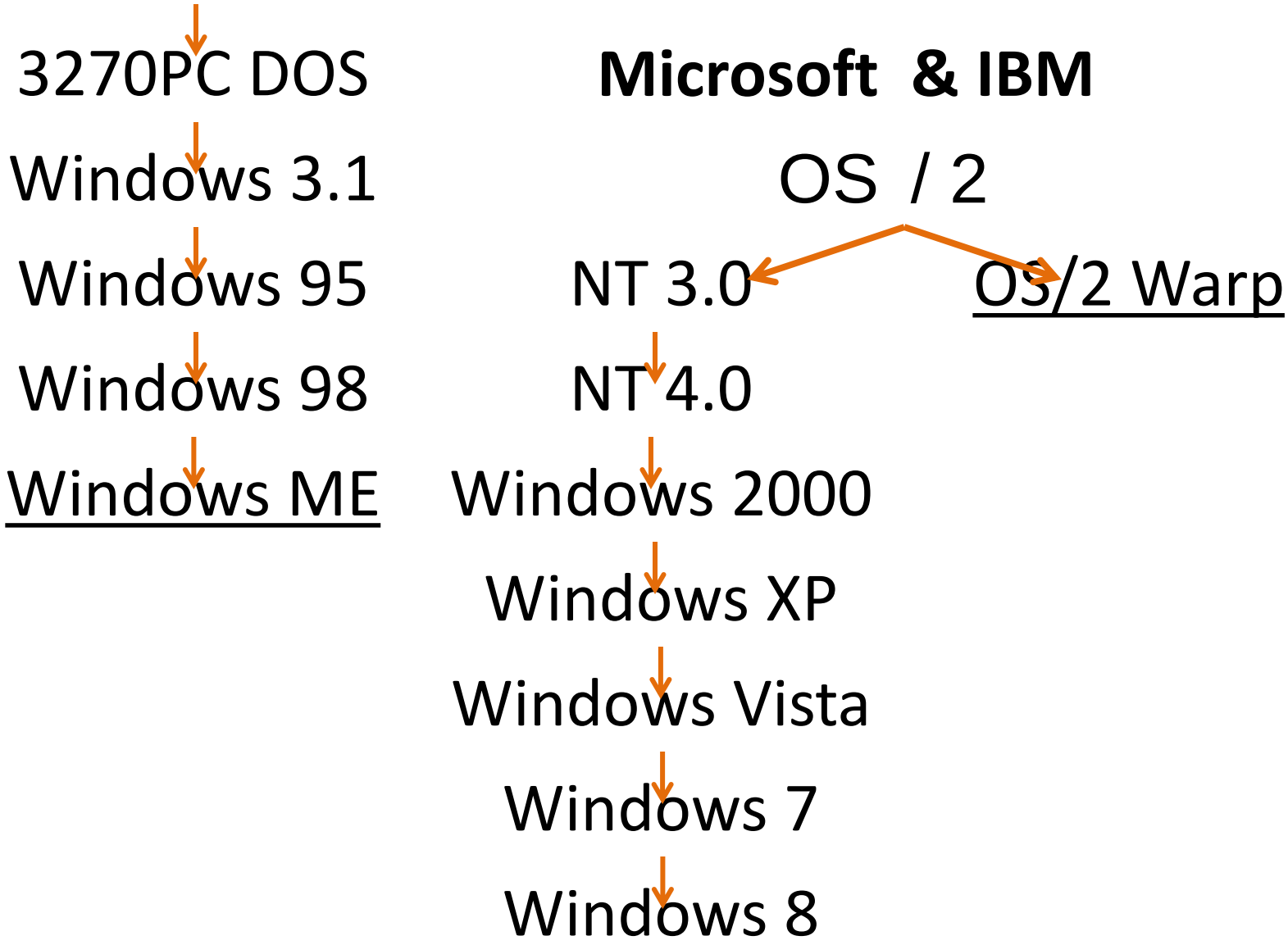
Virtual Memory

- The IBM/370 introduced virtual memory
- The Intel 386 provided virtual memory support
- The operating system has to move pages between RAM and disk
- The OS has to maintain the page tables and addressing environment

Memory Hierarchy

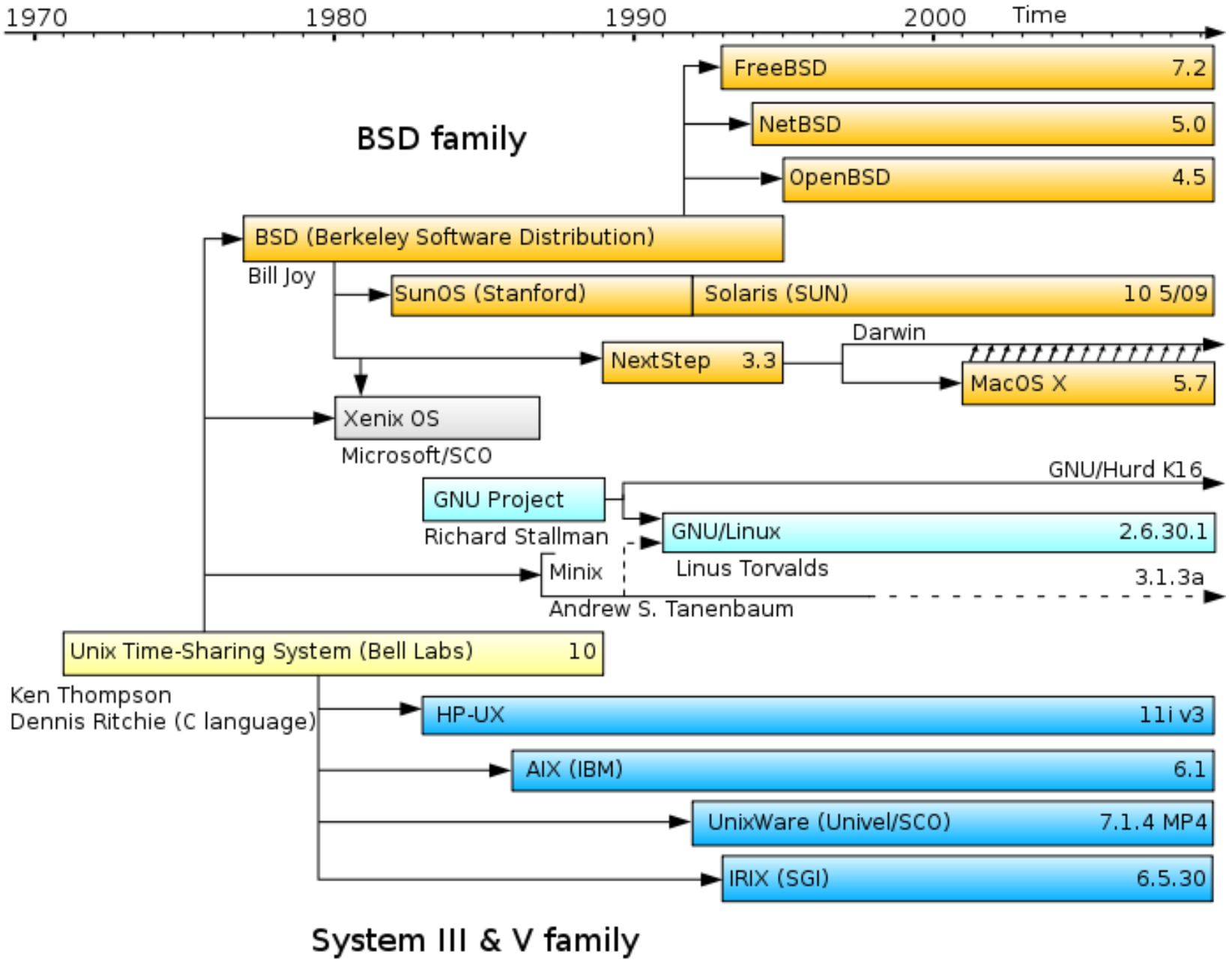


History of Microsoft Windows



History of Unix

- Originally developed for a PDP-7 in 1970 by Brian Kernighan and Dennis Richie.
- Written in C in 1973
- OS with source code was available free.
- Andrew Tanenbaum created Minix
- Linus Torvalds extended Minix to Linux


1970
1980
1990
2000
Time
BSD family
FreeBSD 7.2
NetBSD 5.0
OpenBSD 4.5
BSD (Berkeley Software Distribution)
Bill Joy
SunOS (Stanford)
Solaris (SUN) 10 5/09
Darwin
NextStep 3.3
MacOS X 5.7
Xenix OS
Microsoft/SCO
GNU/Hurd K16
GNU Project
Richard Stallman
GNU/Linux 2.6.30.1
Linus Torvalds
Minix 3.1.3a
Andrew S. Tanenbaum
Unix Time-Sharing System (Bell Labs) 10
Ken Thompson
Dennis Ritchie (C language)
HP-UX 11i v3
AIX (IBM) 6.1
UnixWare (Univel/SCO) 7.1.4 MP4
IRIX (SGI) 6.5.30
System III & V family

Marketing

- Unix was free
  - I used an IBM system whose OS whose list price was $12,000 a month
- Android is free
  - 79% of all smart phones use Android