OPERATING SYSTEMS

Chapter 1 Part 2

Contents

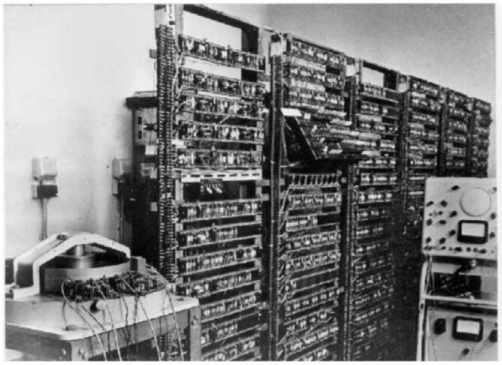
- History of OS
- OS zoo
- OS Architectures

1.3 History of Operating Systems

Generations:

- (1945–55) Vacuum Tubes
- (1955–65) Transistors and Batch Systems
- (1965–1980) ICs and Multiprogramming
- (1980–Present) Personal Computers
- (1990 – Present) – Mobile computers

Generation Of Computers 1st To 5th



First Generation 1946-1959

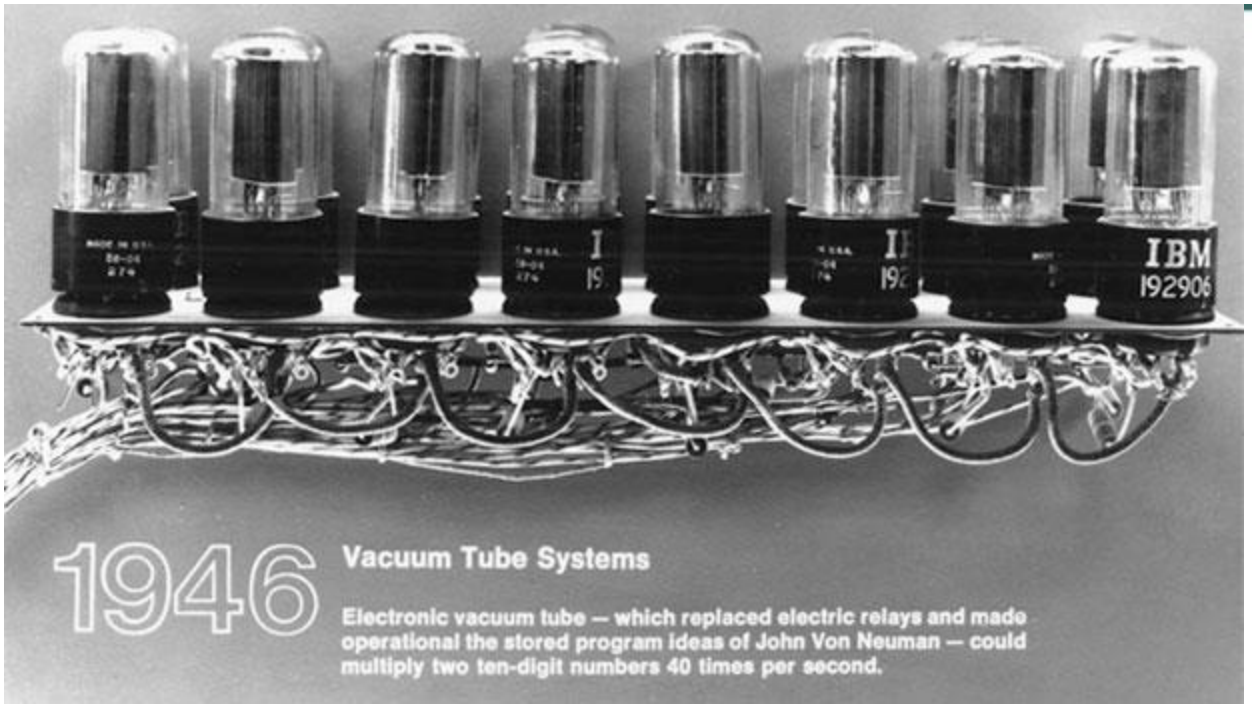
Second Generation
1959-1965

Third Generation
1965-1971

Fourth Generation
1971-1980

Fifth Generation 1980- Present

1st: vacuum tubes (1945–55)



Professor John Atanasoff and his graduate student Clifford Berry built what is now regarded as the first functioning digital computer at Iowa State University. It used 300 vacuum tubes.

1st: vacuum tubes (1945–55)

- Large and slow
- No concept of OS
- No notion of programmers
  - All work (design, build, operate and maintain the computer) done by engineers
- All programming was done with machine language, or even by wiring circuits using cables

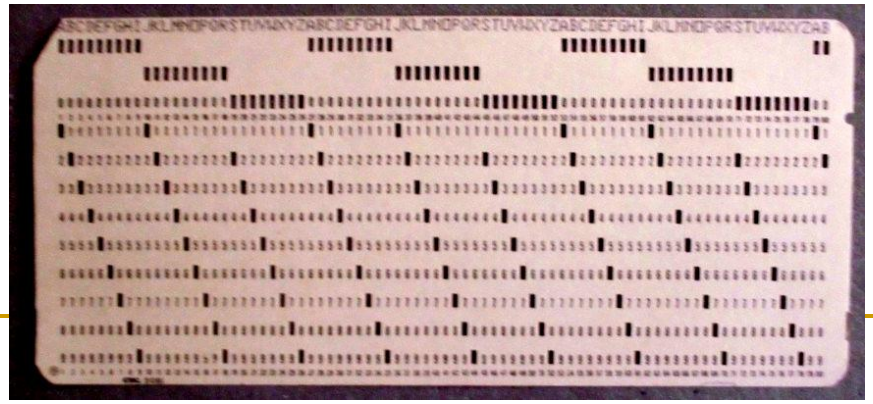
1st: vacuum tubes (1945–55)

- Very time consuming:
  - The usual mode of operation was for the engineer to sign up for a block of time using the signup sheet on the wall, then come down to the machine room, and spend the next few hours hoping that none of the 20,000 or so vacuum tubes would burn out during the run.

- For simple calculations:
  - All the problems were simple straightforward mathematical and numerical calculations, such as calculations with sines, cosines, and logarithms, or computing artillery trajectories.

2nd: Transistors and batch systems (1955–65)

- Computers are managed by professional operators.
- For the first time, there was a clear separation between designers, builders, operators, programmers, and maintenance personnel.
- Programmers use punch card to run programs; operators operate (load compiler, etc.) and collect output to the user



2nd: transistors and batch systems (1955–65)

- Used mostly for scientific and engineering calculations (e.g. partial differential equations).
- They were largely programmed in FORTRAN and assembly language.

Run a job (program):

1. Write the code on a paper (in FORTRAN or assembler)
2. Punch it on the cards
3. Bring the card to the input room and hand it to the operator
4. Go for coffee until the output is ready.
5. Operator collects the output from the printer and hand it over to the output room so that the programmer can collect it later.

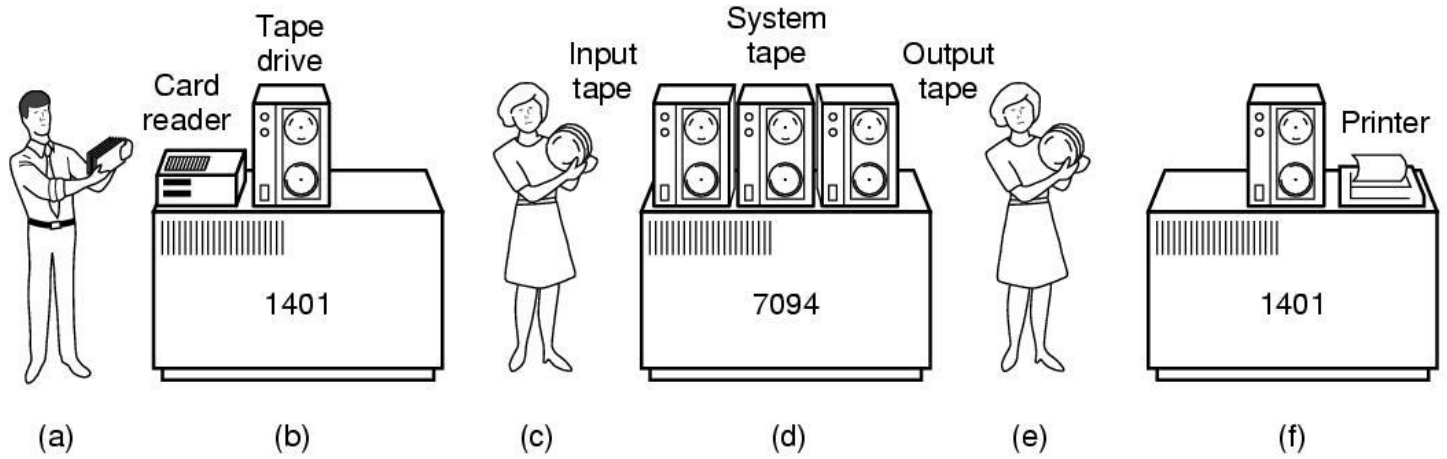
2nd: transistors and batch systems

Much computer time was wasted while operators were walking around the machine room. → batch systems proposed.

Batch system: Collect a batch of jobs from the input room, then read them into a magnetic tape; the same for output

2nd: transistors and batch systems

Use inexpensive computer to handle the punch cards, then use an expensive computer for computation

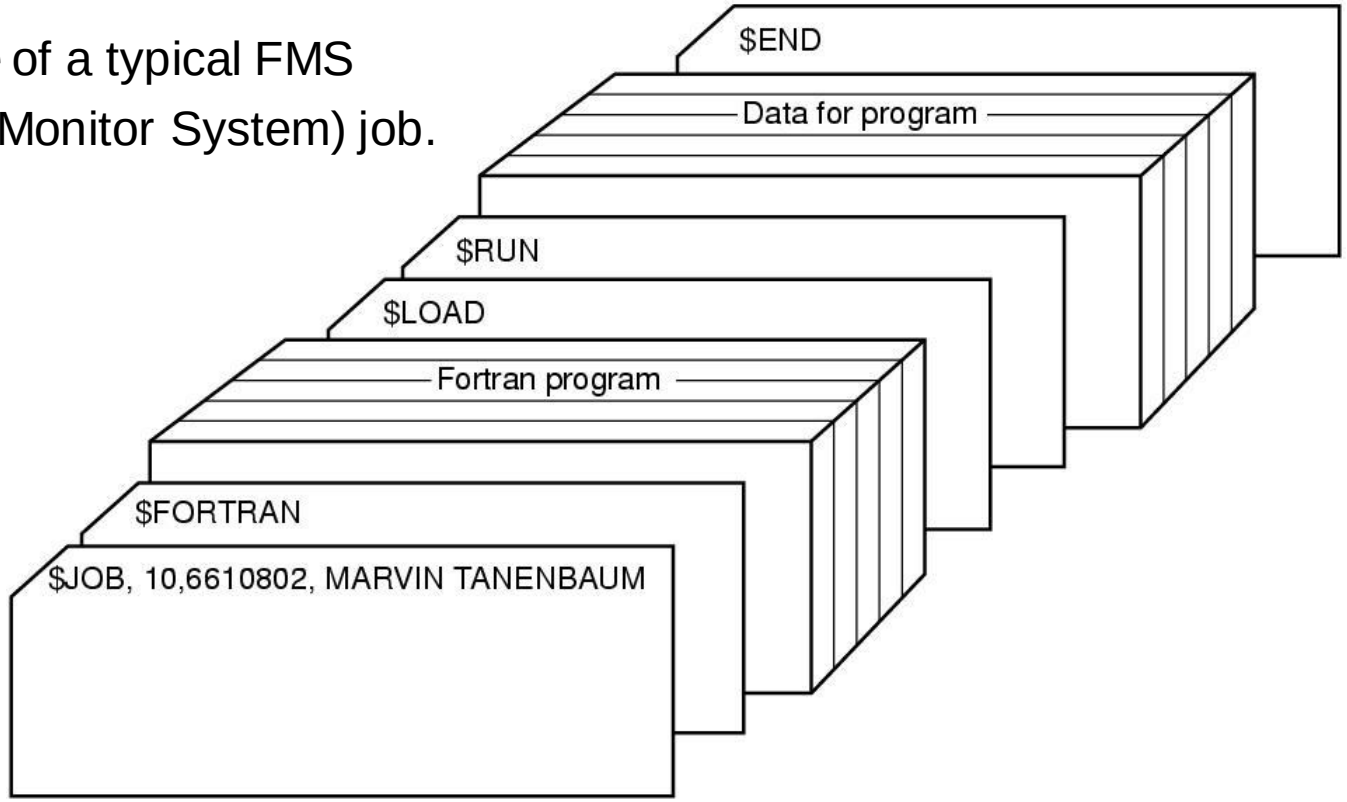


(a) Programmers bring cards to 1401.
(b) 1401 reads batch of jobs onto tape.
(c) Operator carries input tape to 7094.
(d) 7094 does computing.
(e) Operator carries output tape to 1401.
(f) 1401 prints output.

- 1401 - Less expensive machine to perform I/O
- 7094 - Expensive one for execution

2nd: transistors and batch systems

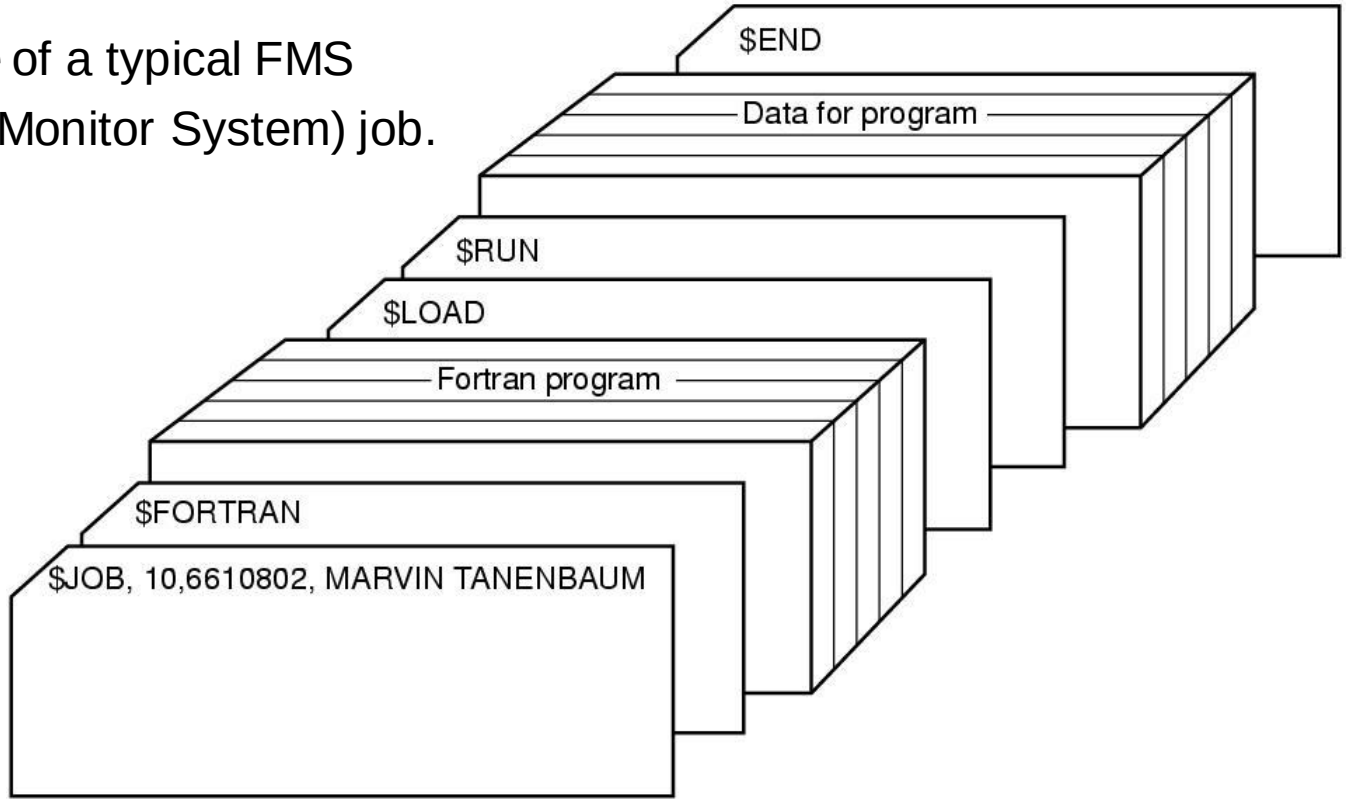
Structure of a typical FMS (FORTRAN Monitor System) job.



- **$JOB** card - specifies the maximum run time in minutes, account number to be charged, and programmer's name.
- **$FORTRAN** card - tells the OS to load the FORTRAN compiler from the system tape.
- It was directly followed by the program to be compiled

2nd: transistors and batch systems

Structure of a typical FMS (FORTRAN Monitor System) job.

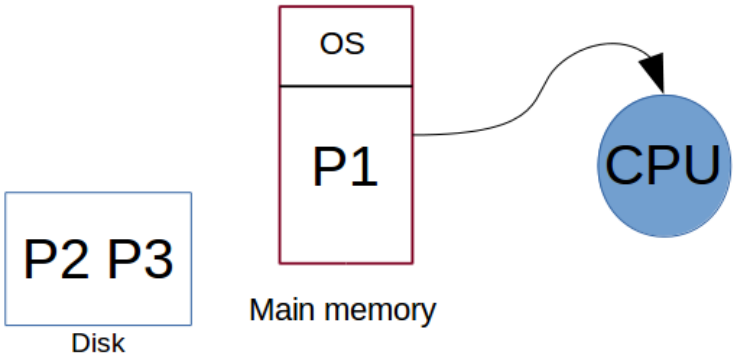


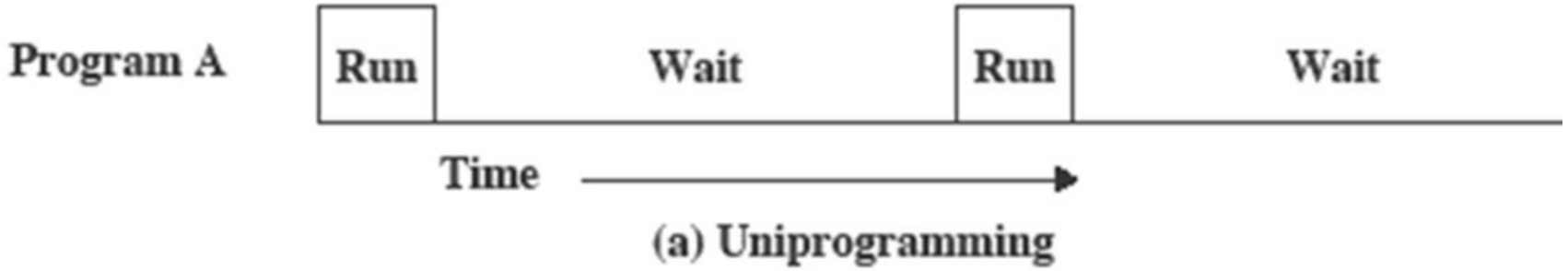
- **$LOAD** card - directing the OS load the object program just compiled. (Compiled programs were often written on scratch tapes and had to be loaded explicitly.)
- Next came the **$RUN** card, telling the OS to run the program with the data following it.
- Finally, the **$END** card marked the end of the job.

2nd: transistors and batch systems

Uniprogramming

- Executing one job at a time until finished.
- Processor must wait for I/O instruction to complete before proceeding to next instruction.





3rd: IC and Multiprogramming (1965–1980)

- (2nd generation) By the early 1960s, most computer manufacturers had two distinct, incompatible, product lines.
  - 7094 for numerical calculations in science and engineering
  - 1401 for printing by banks and insurance companies

- IBM 360 (3rd generation)
  - Aims to adapts 1401/7094, covers all trades of life (scientific and commercial)
  - Millions of lines of assembly language written by thousands of programmers
  - First major computer line to use (small-scale) ICs (Integrated Circuits), thus providing a major price/ performance advantage over the second-generation machines, which were built up from individual transistors.

3rd: IC and Multiprogramming (1965–1980)

Rescue?

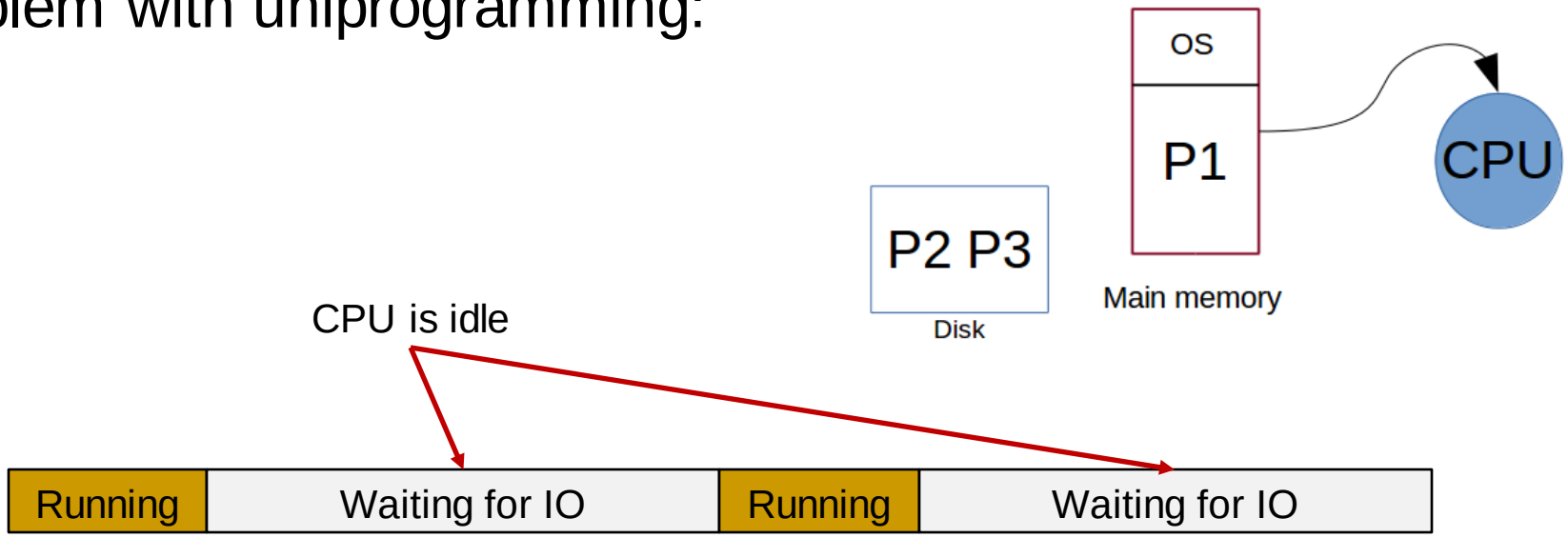
- OS/360: a dinosaur stuck in a tar pit
  - It consisted of millions of lines of assembly language written by thousands of programmers, and
  - Contained thousands upon thousands of bugs, which necessitated a continuous stream of new releases in an attempt to correct them.
  - Each new release fixed some bugs and introduced new ones, so the number of bugs probably remained constant over time

3rd: IC and Multiprogramming (1965–1980)

- However, OS/360 introduces several key techniques
  - Multi-programming: solve the problem of CPU idling
  - Spooling: Simultaneous Peripheral Operation On Line

3rd: IC and Multiprogramming

Problem with uniprogramming:

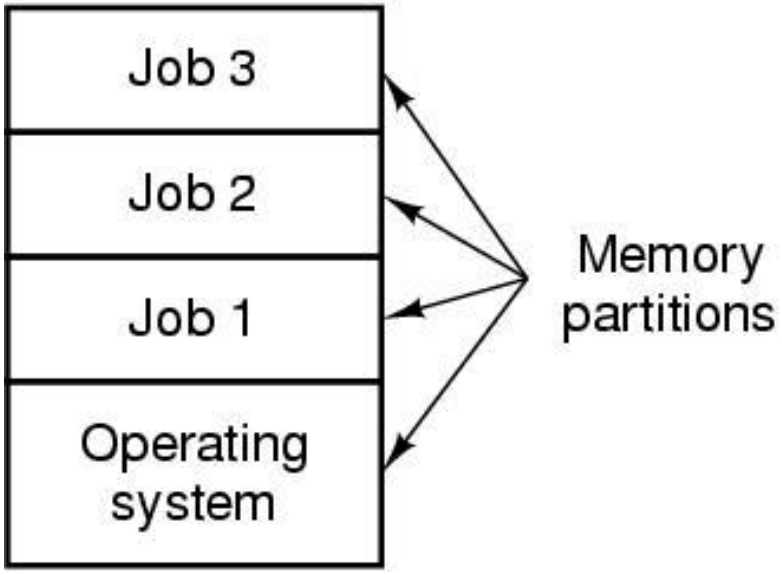


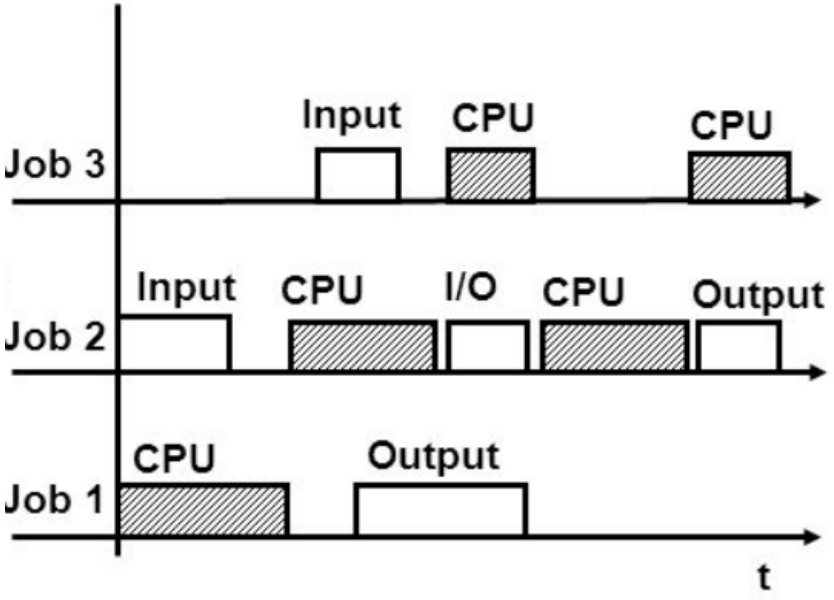
- When the current job paused to wait for a tape or other I/O operation to complete, the CPU simply sat idle until the I/O finished.
- With commercial data processing, the I/O wait time can often be 80% of the total time, so something had to be done to avoid having the (expensive) CPU be idle so much.

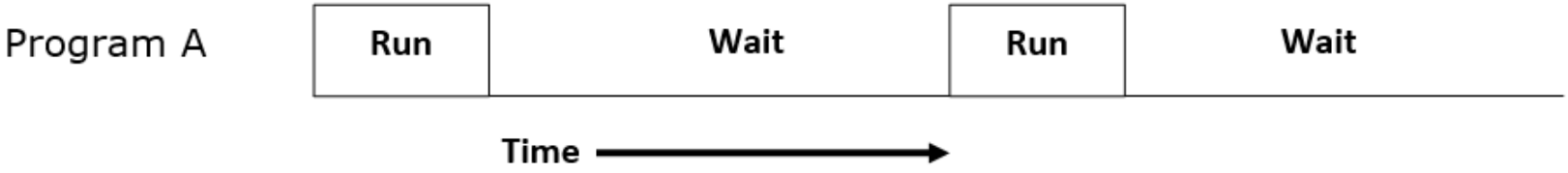
3rd: IC and Multiprogramming

Solution: Multiprogramming:

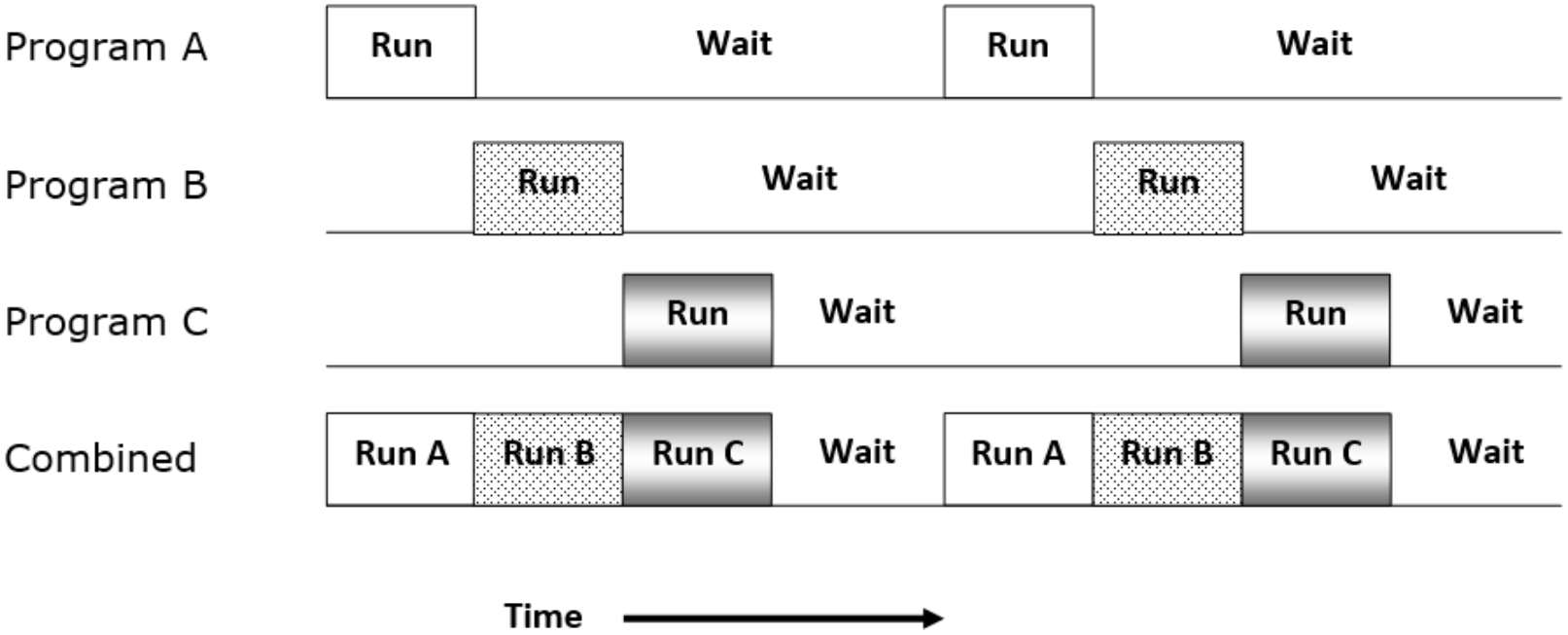
- Partition memory into several pieces, with a different job in each partition.
- While one job was waiting for I/O to complete, another job could be using the CPU.







(a)



Maximized CPU utilization!!!

(b)

Figure 1.3: (a) Uni-programming (b) Multiprogramming with three programs.

3rd: IC and Multiprogramming

Spooling: **S**imultaneous **P**eripheral **O**peration **O**n **L**ine

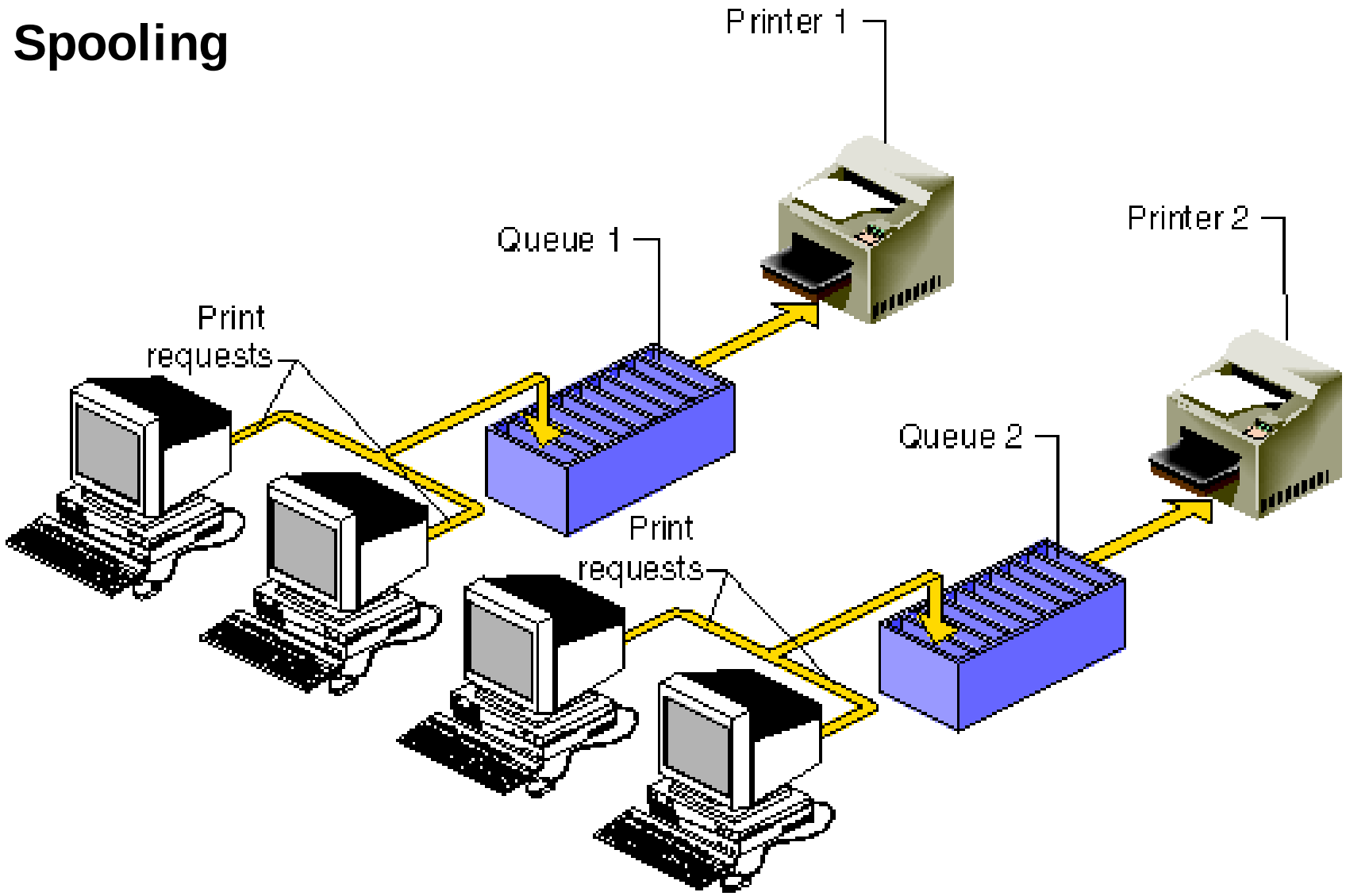
SPOOL stands for **S**imultaneous **P**eripheral **O**perations **O**n**L**ine. It just means storing jobs in a queue so the process can do other things.

It is most often used today for printing. Imagine if you couldn't use Word until your 200 page document had printed.

Real world example: Correcting students' books

3rd: IC and Multiprogramming

Spooling



3rd: IC and Multiprogramming

- 3rd generation OS was well suited for big scientific calculations and massive data processing
- Problem: Difficult to debug quickly
  - The time between submitting a job and getting back the output was often several hours, so a single misplaced comma could cause a compilation to fail, and the programmer to waste half a day.

4th: personal computers (1980 – present)

- Development of LSIC (Large Scale Integrated Circuits) computers became more powerful, compact, reliable, and affordable.
- OS – multi-programming, interactive, multi-user
- In 1980 IBM designed IBM PC
  - OSs used - BASIC Interpreter, DOS, MS-DOS
- GUI – Lisa
  - Steve Jobs's first attempt to make GUI called Lisa –too expensive and failed commercially.
- MS-DOS with GUI– Win95/98/me - WinNT/xp…

5th: mobile computers

- “The brick” - first true handheld phone appeared in the 1970s and, at roughly one kilogram of weight

NBC News

First cell phone a true 'brick'

Historical Timeline

Generation	Period	Operating System Genre
First Generation	1945 - 1955	Serial Processing (No Operating System)
Second Generation	1955 - 1965	Simple Batch OS
Third Generation	1965 - 1980	Multiprogramming OS
Fourth Generation	1980 - present	Time-Sharing OS

- **Multiprogramming** is the effective utilization of CPU time, by allowing several programs to use the CPU at the same time
- **Time sharing** is the sharing of a computing facility by several users that want to use the same facility at the same time.

Computer Generations

Generation	Hardware Components		Characteristics	Computers
First Generation (1942-1959)		⊙ Vaccum Tubes	⊙ Machine Language ⊙ Huge Size ⊙ Highly Expensive ⊙ High Consumption of Electricity	⊙ ENIAC ⊙ UNIVAC ⊙ EDVAC ⊙ EDSAC ⊙ IBM-701
Second Generation (1959-1965)		⊙ Transistors ⊙ Magnetic Tapes	⊙ Batch processing, Multiprogramming OS ⊙ Expensive ⊙ FORTRAN, COBOL	⊙ IBM 7000 ⊙ CDC 1604 ⊙ ATLAS ⊙ NCR 304 ⊙ Honeywell 400
Third Generation (1965-1975)		⊙ Integrated Circuits	⊙ Remote processing, time-sharing, Multiprogramming OS ⊙ Faster, Compact & Cheaper ⊙ PASCAL PL/1, BASIC, ALGOL-68	⊙ IBM 360/370 ⊙ PDP 8/11 ⊙ CDC 6600
Fourth Generation (1975-1988)		⊙ VLSI Microprocessor circuits	⊙ Time-sharing, real-time networks, distributed, GUI OS ⊙ Faster, Compact & Affordable ⊙ C, C++, DBASE	⊙ DEC 10 ⊙ STAR 1000 ⊙ CRAY-1/11 ⊙ Apple II ⊙ VAX 9000
Fifth Generation (1988-Present)		⊙ ULSI Microprocessor circuits	⊙ Parallel Processing & Artificial Intelligence technology ⊙ C and C++, Java, .Net	⊙ IBM ⊙ Pentium ⊙ Param

Quiz time

1. _____ is a technique in which an Operating System collects the programs and data together in a batch before processing starts.
   A. Batch processing
   B. Interactivity processing
   C. Real Time processing
   D. Distributed processing

2. Which of the following is an advantage of Batch processing?
   A. Job could not enter an infinite loop.
   B. Easy to debug program.
   C. Increased performance
   D. All of the above

3. _____ is when multiple jobs are executed by the CPU simultaneously by switching between them

A. Multiprogramming
B. Distributed Environment
C. Spooling

4. Multiprogramming generally?

A. Decreases CPU utilization
B. Share the processor
C. Both A and B
D. None of the above

5. Which of the following are the advantage of Multiprogramming?

A. High and efficient CPU utilization.
B. CPU scheduling is not required.
C. memory management is good.
D. All of the above

1.4 The Operating System Zoo

- Mainframe operating systems
- Server operating systems
- Multiprocessor operating systems
- Personal computer operating systems
- Handheld operating systems
- Embedded operating systems
- Sensor node operating systems
- Real-time operating systems
- Smart card operating systems

Mainframe OS

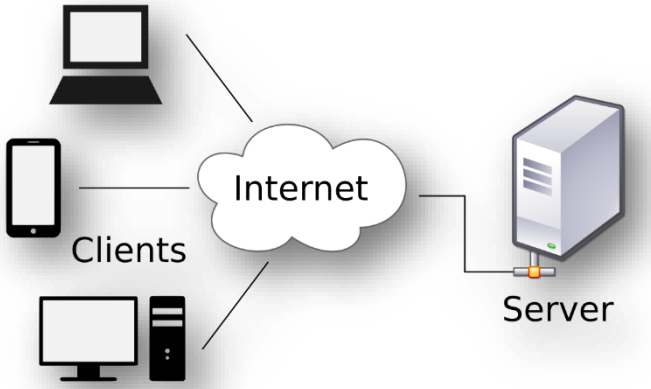
For large-scale processing - expensive

- Mainframe computers or mainframes are computers used primarily by large organizations for critical applications; bulk data processing, such as census, industry and consumer statistics, enterprise resource planning; and transaction processing.
- Mainframes typically cost several hundred thousand dollars.

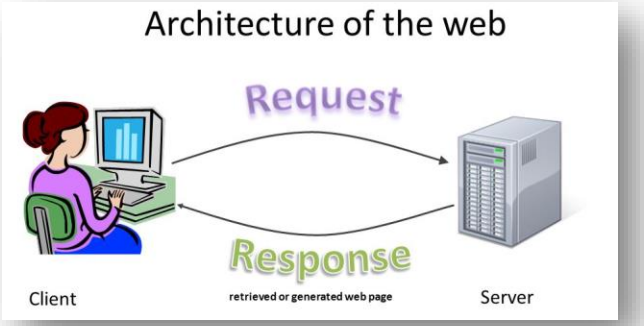
Mainframe OS

- Most mainframes computers are sold by IBM, and the OS are also provided by IBM.
- z/OS, is IBM's foremost mainframe OS
- The primary focus of a **supercomputer** is speed, whereas for a **mainframe** it is to deal with extensive amount of data processing (I/O). **Mainframe** computers are not as powerful as **supercomputers**.

Server OS

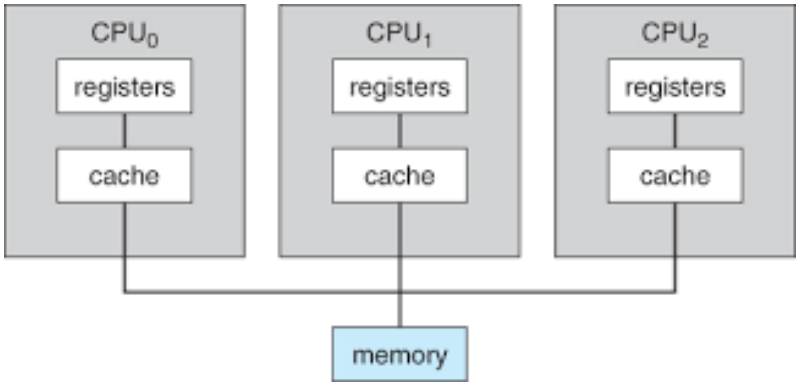


- Servers are to handle multiple user requests at the same time.
  - Servers “serve” client computers data that users request.
  - Servers are either very large personal computers, workstations, or even mainframes.
  - E.g. web servers, file servers.



- Server OS runs on servers
  - They serve multiple users at once over a network and allow the users to share hardware and software resources.
  - Internet providers run many server machines to support their customers to store the web sites and handle the incoming requests.
  - Typical server OS are UNIX and Windows 2000.

Multi-processor OS



- Parallel computing – A single, large job can be distributed over multiple processors. Some special software must coordinate their work and assemble the final result.
- Multiple CPUs into a single system, or a system connects multiple systems
- They need special OS, but often these are variations on the server OSs, with special features for communication and connectivity.
- Example: Windows, Mac, Linux

Personal Computer OS

- Their job is to provide a good interface to a single user.
- Common examples are Windows 98, Windows 10, Macintosh operating system, and Linux.

Embedded OS

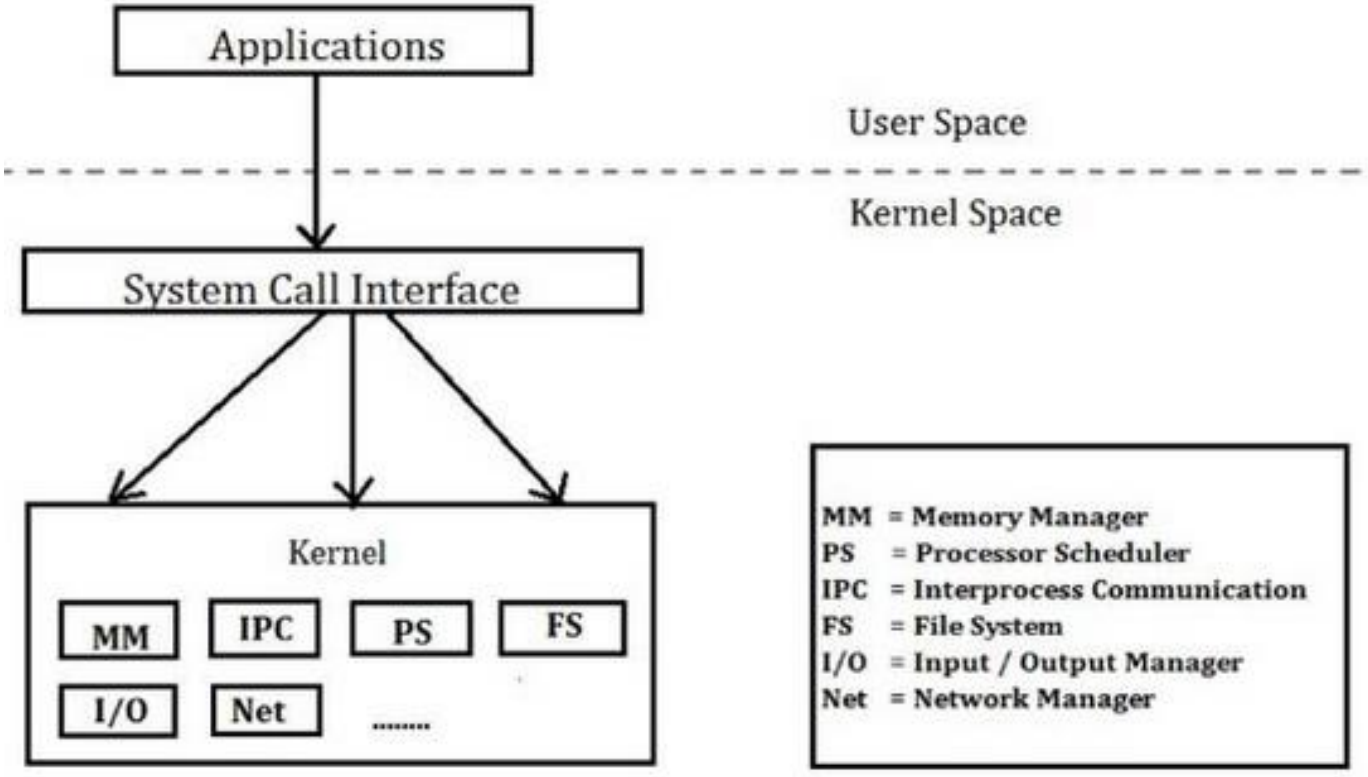
- Examples:
  - Ovens
  - Refrigerators
  - Automobiles
  - Washing machines
  - Exercise bike

1.5 Operating System Architectures

- Today's OS are
  - Complex
  - Provide many services
  - Support variety of hardware and software
- OS architectures help manage this complexity
- Different architectures
  - Monolithic OS
  - Layered OS
  - Microkernel OS
  - Hybrid Kernel
  - Client Server

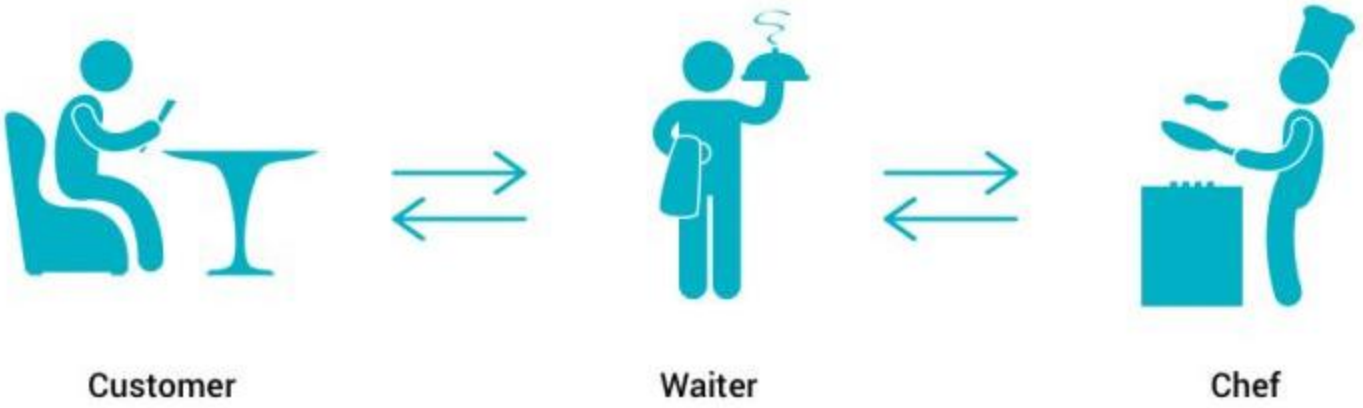
Monolithic OS

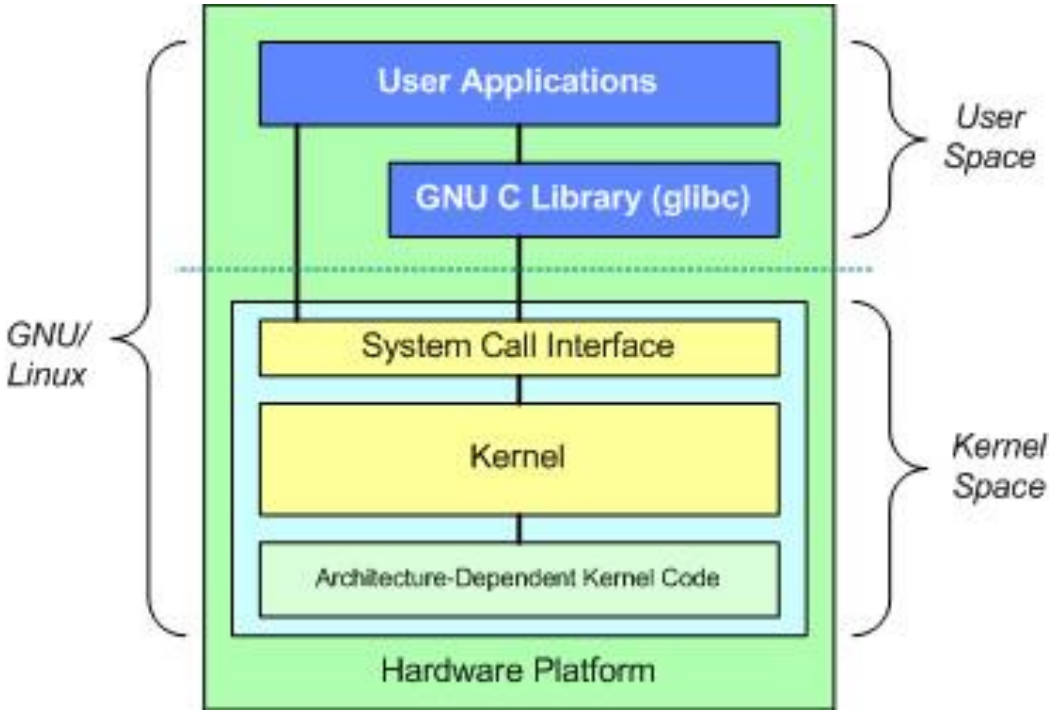
- The entire OS runs as a single program in kernel mode
- E.g. OS/360, LINUX



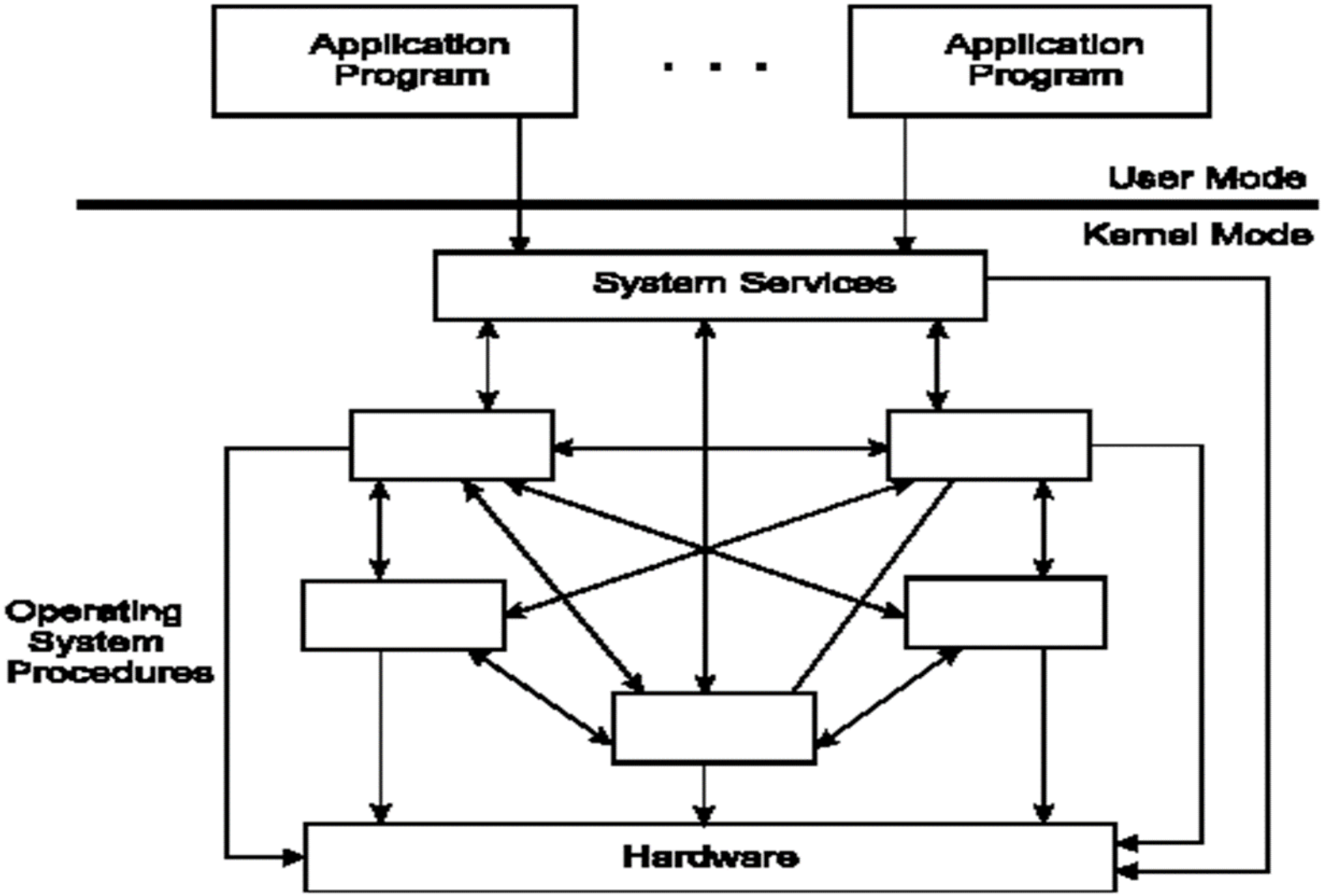
Monolithic Operating System Kernel Architecture

Application programs System Call Interface OS





Monolithic OS

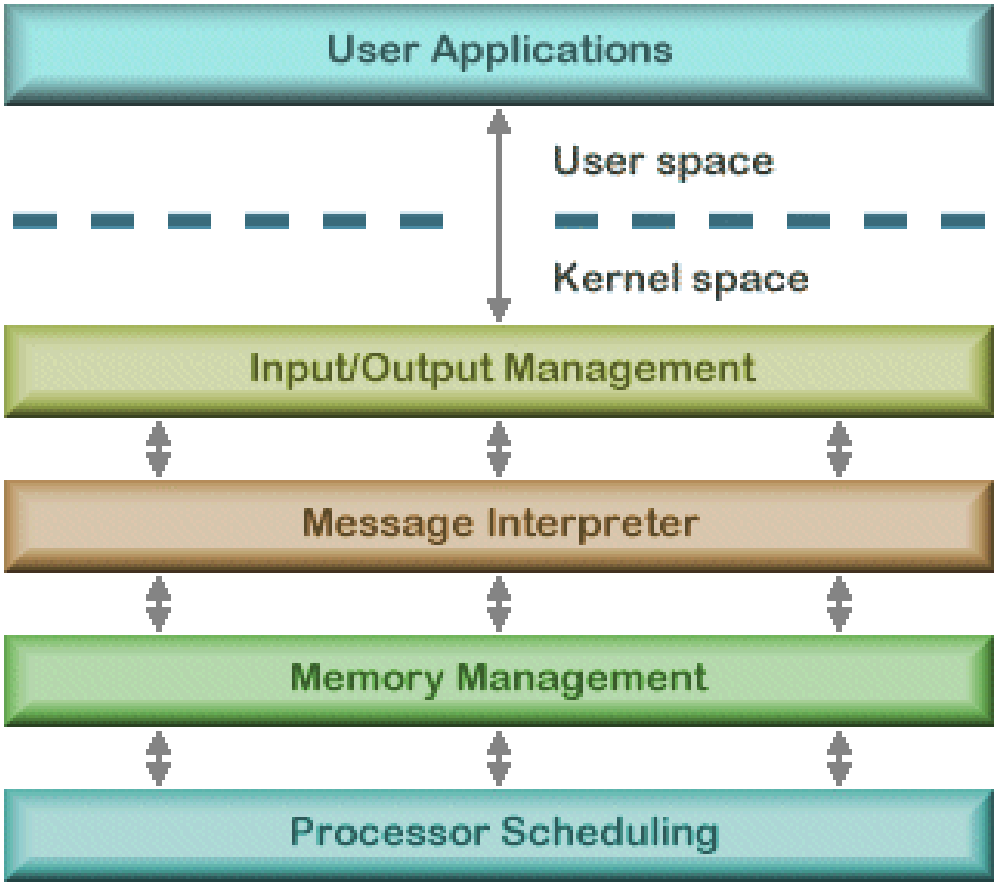


Monolithic OS

- The entire OS runs as a single program in kernel mode
- The OS is written as a collection of procedures, linked together into a single large executable binary program.
- Each procedure in the system is free to call any other one, if the latter provides some useful computation that the former needs.
- Being able to call any procedure you want is very efficient.
- But having thousands of procedures that can call each other without restriction may also lead to a system that is difficult to understand and manage.
- Also, a crash in any of these procedures will take down the entire OS.

Layered OS

- Tries to improve on monolithic kernel designs
- Groups components that perform similar functions into layers
- E.g.: "THE system" built by E. W. Dijkstra and his students in 1968.



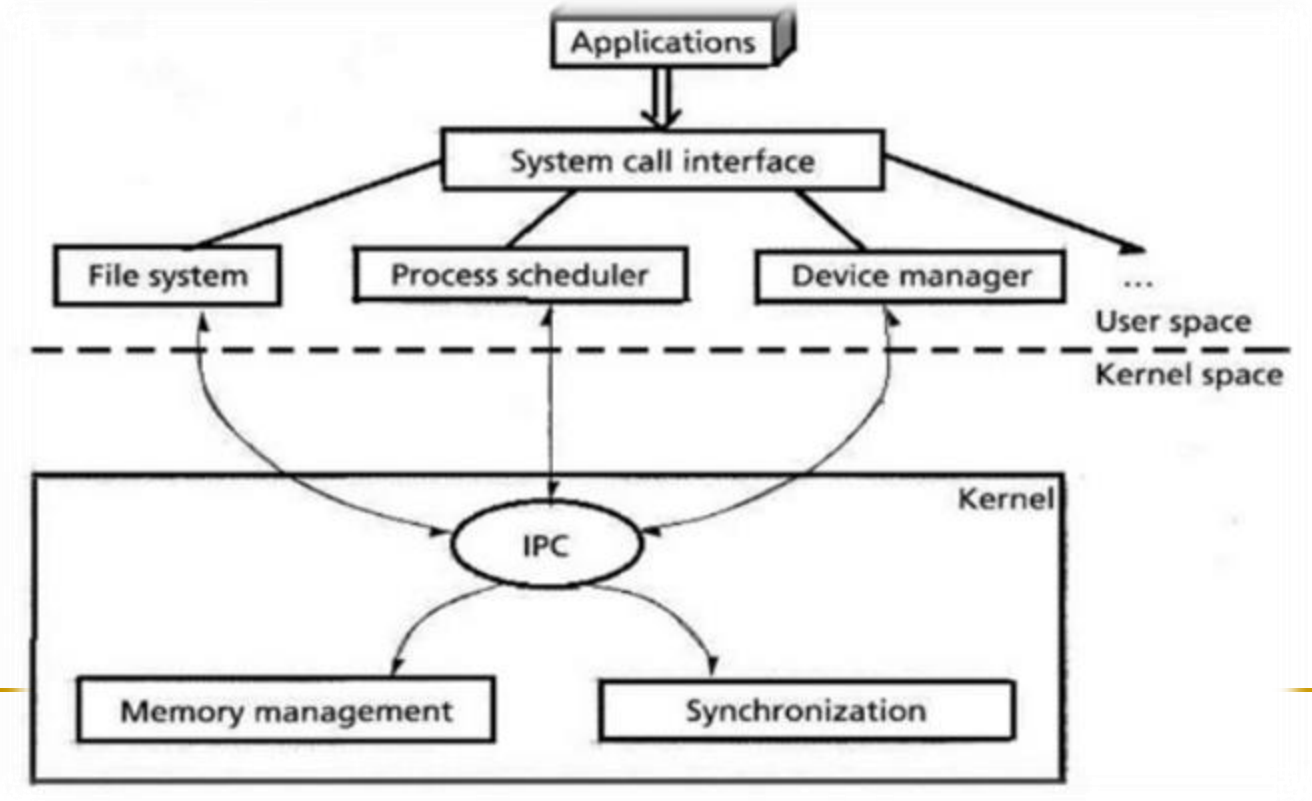
Layered OS

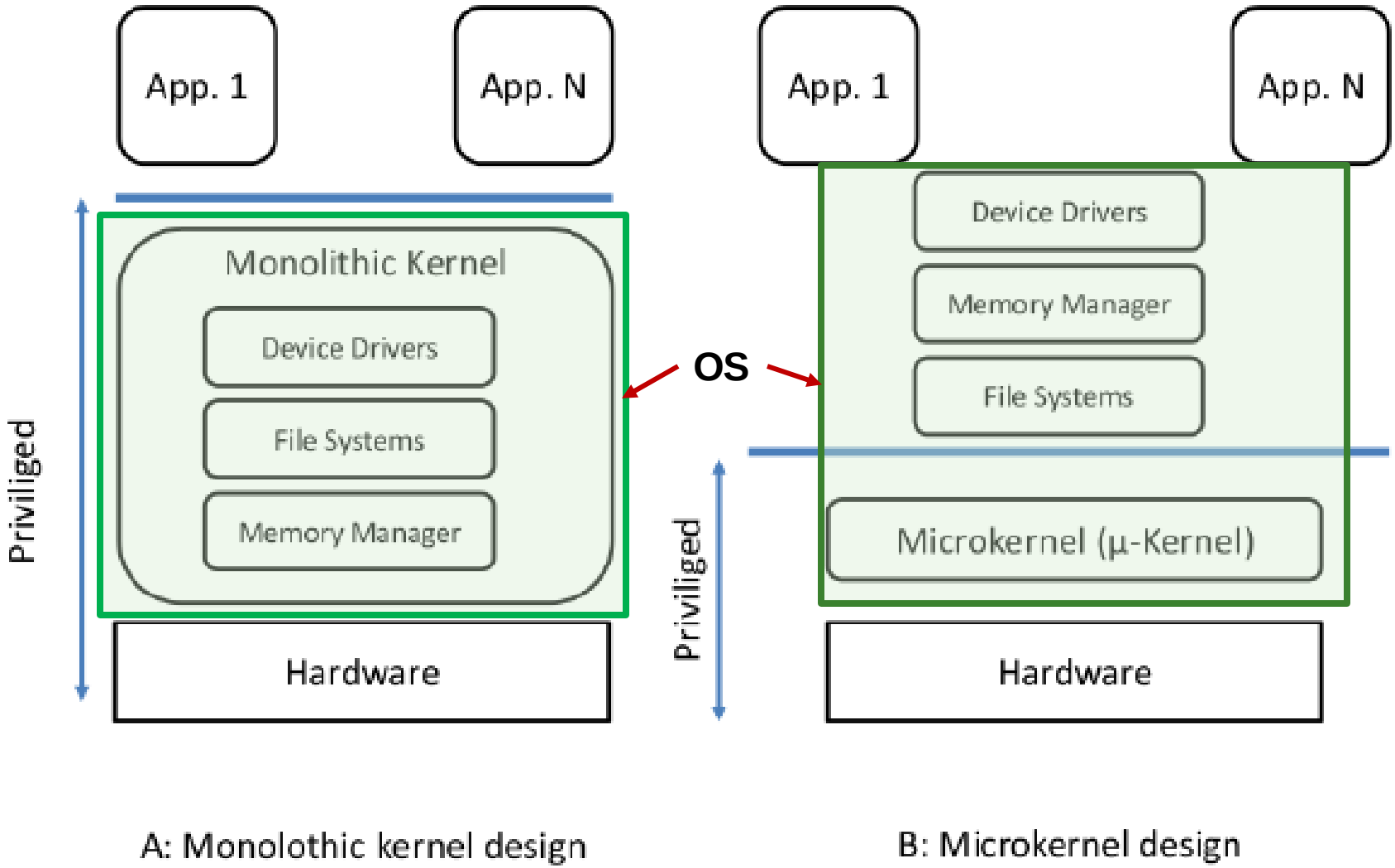
- Each layer communicates only with layers immediately above and below it.
- Not always easy to have a truly layered architecture as some functionalities are mutually dependent.
- In addition, a layered architecture is often inefficient since it requires a high number of traversals of interfaces.
- Disadvantages:
  - Processes' requests might pass through many layers before completion
  - System throughput can be less than monolithic kernels

Microkernel Architecture

Split the OS into small, well-defined modules

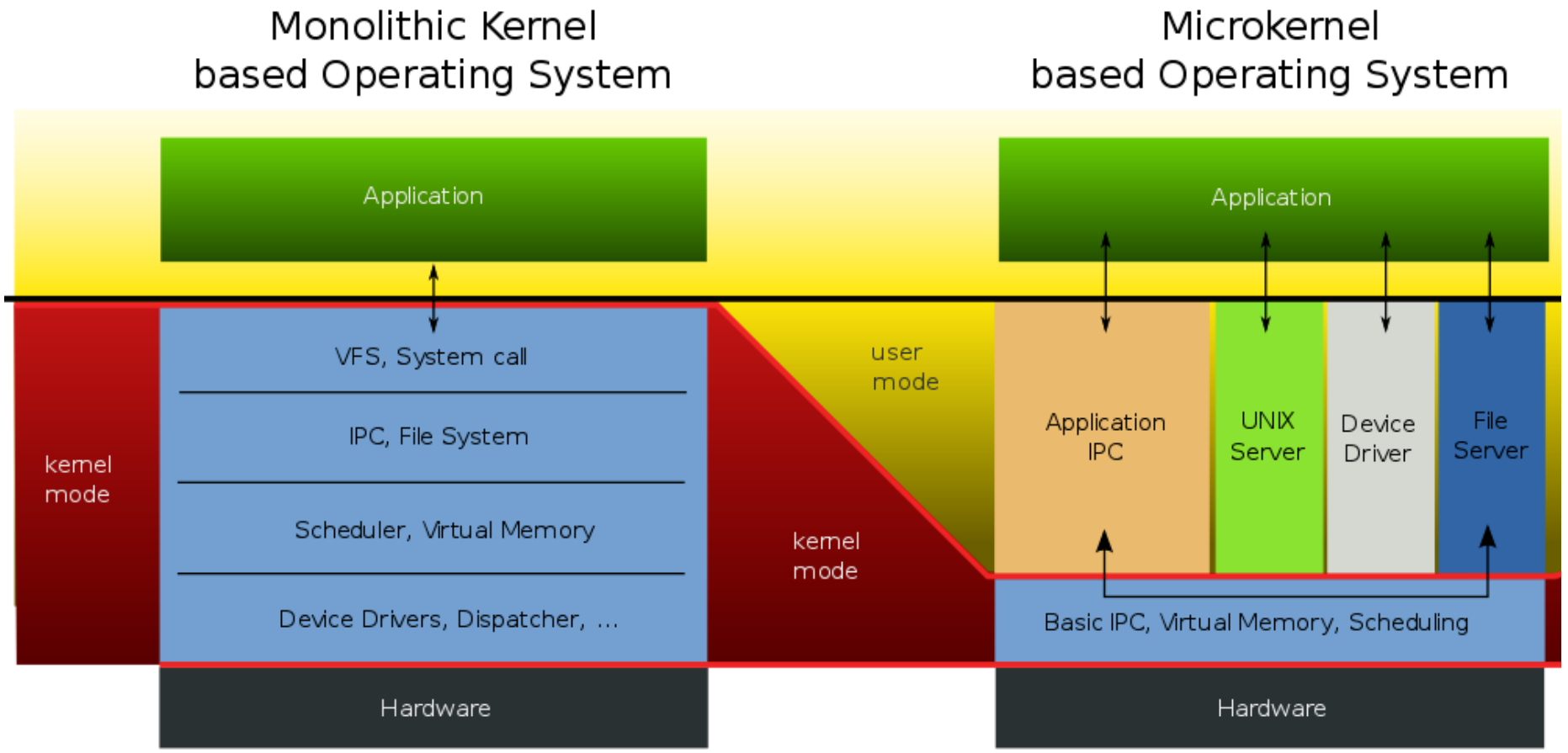
- Only one of the module -the microkernel- runs in kernel mode
- All the others (process management, networking, file system interaction and device management) execute outside the kernel with a lower privilege level.





A: Monolothic kernel design

B: Microkernel design


Monolithic Kernel
based Operating System
Application
VFS, System call
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
kernel
mode
Microkernel
based Operating System
Application
user
mode
Application
IPC
UNIX
Server
Device
Driver
File
Server
Basic IPC, Virtual Memory, Scheduling
kernel
mode
Hardware

Microkernel Architecture

- Microkernel exhibit a high degree of modularity, making them extensible, portable and scalable.
- Because the microkernel does not rely on each component to execute, one or more components can fail, without causing the OS to fail.
- In particular, by running each device driver and file system as a separate user process, a bug in one of these can crash that component, but cannot crash the entire system.
  - Thus a bug in the audio driver will cause the sound to be garbled or stop, but will not crash the computer.
  - In contrast, in a monolithic system with all the drivers in the kernel, a buggy audio driver can easily reference an invalid memory address and bring the system to a grinding halt instantly.

Microkernel Architecture

- A few of the better-known microkernels include Integrity, K42, L4, PikeOS, QNX, Symbian, and MINIX 3.

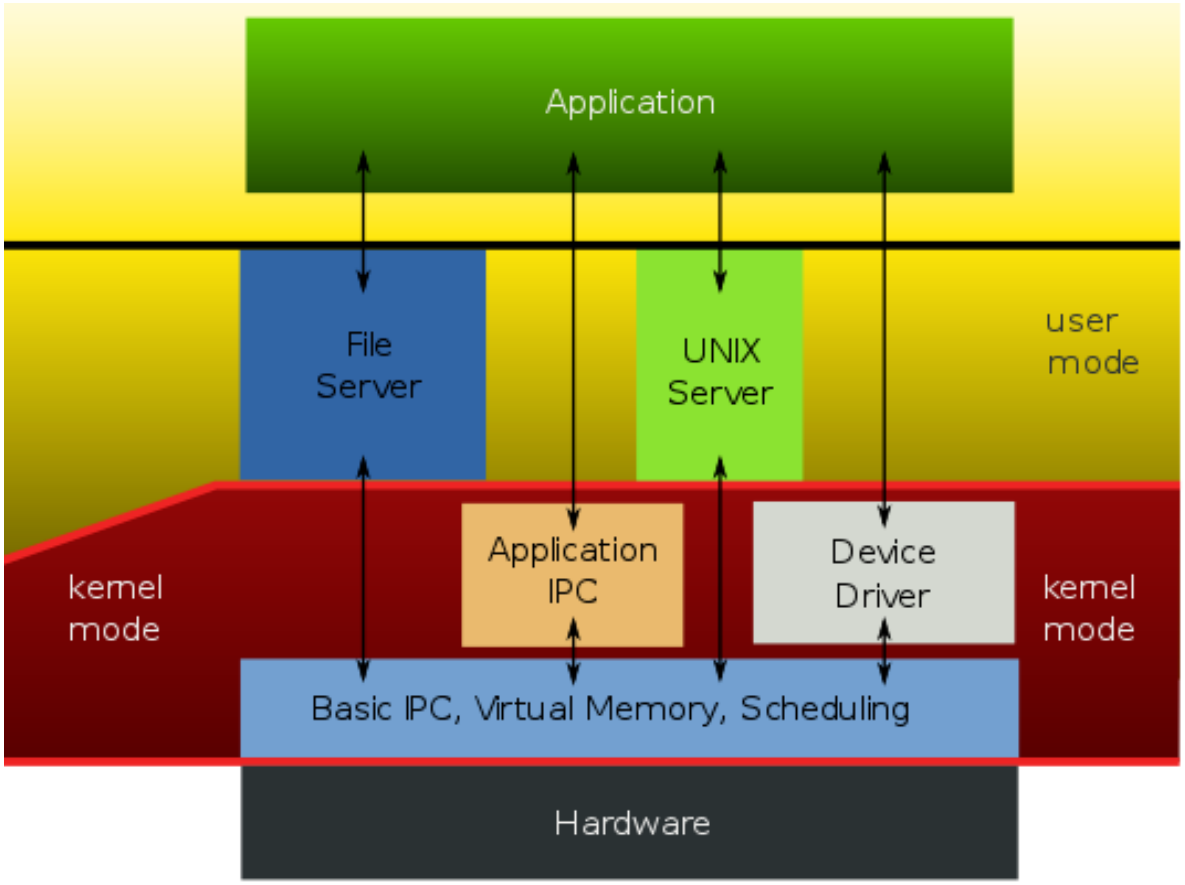
Basis for comparison	Microkernel	Monolithic kernel
Size (small / large)		
Execution (slow / fast)		
Extendible (easy / difficult)		
What will happen if a service crashes?		
Example		

Hybrid Kernel Architecture

- Combine the best of both
  - Speed and simple design of a monolithic kernel
  - Modularity and stability of a microkernel

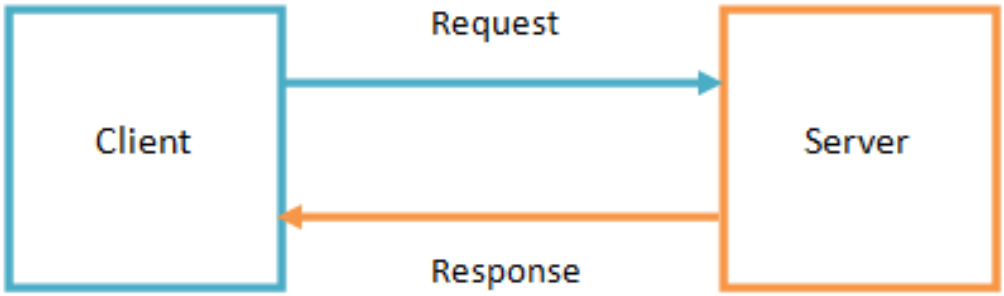
- Still similar to a monolithic kernel
  - Disadvantages still apply here

Hybrid Kernel Architecture



Hybrid kernels are used in most commercial operating systems such as Microsoft Windows NT 3.1, NT 3.5, NT 3.51, NT 4.0, 2000, XP, Vista, 7, 8, 8.1 and 10.

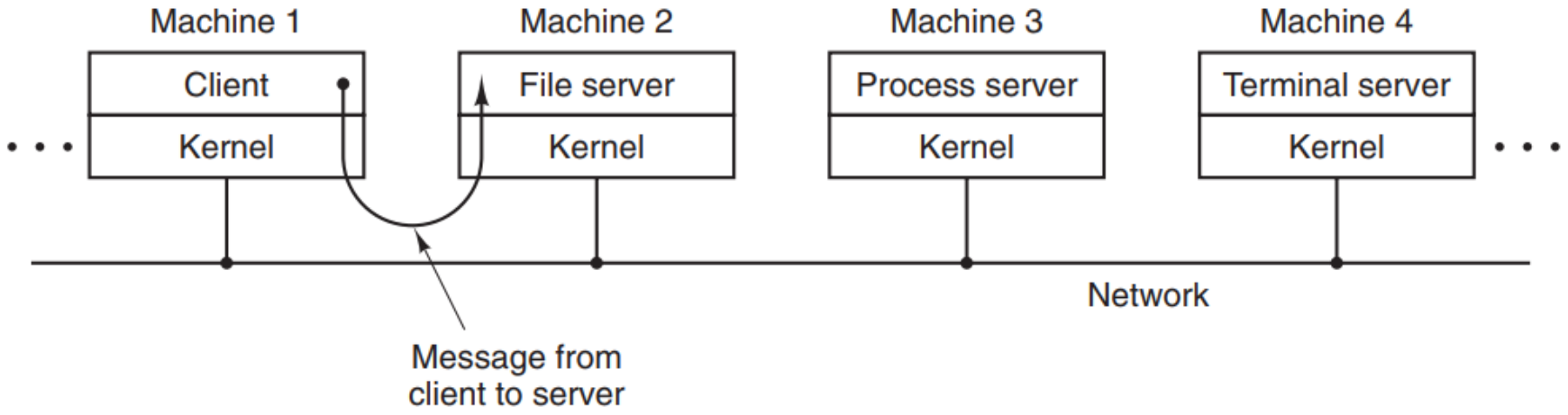
Client-Server model



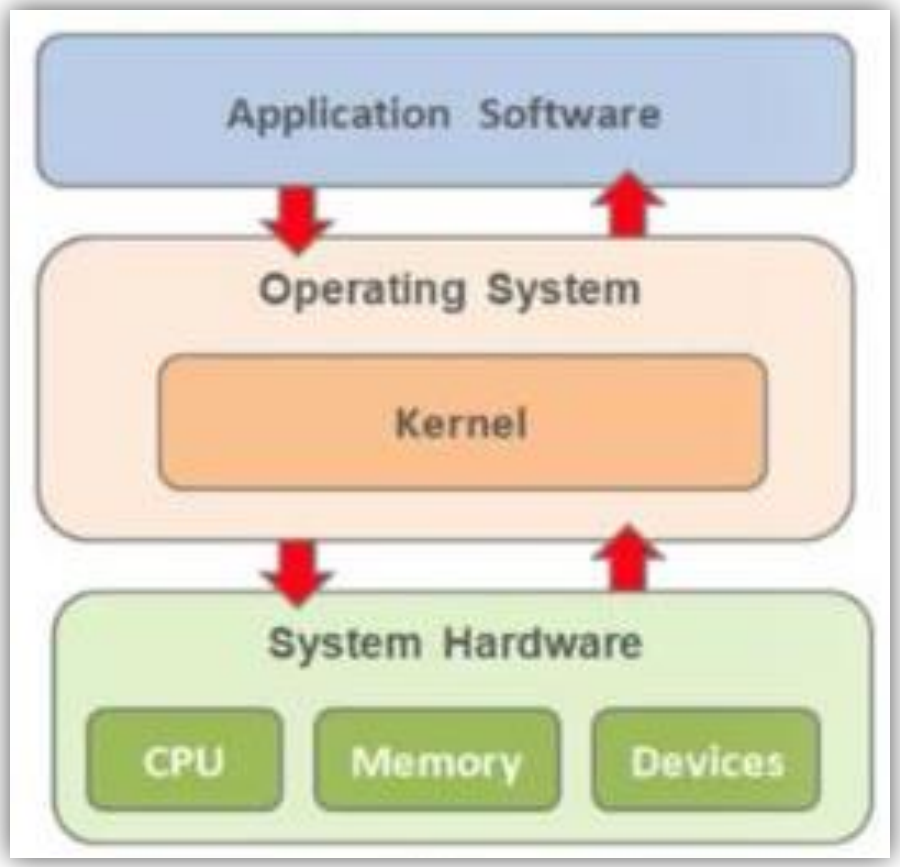
- A slight variation of the microkernel idea is to distinguish two classes of processes,
  - the servers, each of which provides some service, and
  - the clients, which use these services.
- Communication between clients and servers is often by message passing.
  - To obtain a service, a client process constructs a message saying what it wants and sends it to the appropriate service. The service then does the work and sends back the answer.

Client-Server model

- A generalization of this idea is to have the clients and servers run on different computers, connected by a local or wide-area network
- Since clients communicate with servers by sending messages, the clients need not know whether the messages are handled locally on their own machines, or whether they are sent across a network to servers on a remote machine.



Kernel vs OS?



Kernel is a part of OS.

OS	Kernel
It is a system software.	It is system software which is an important part of the OS.
One major purpose of an OS is to provide security.	The main purpose of a kernel is to manage memory, disk and task.
It provides an interface between hardware and user.	It provides an interface (system call interface) between application and hardware.
Without an OS a computer cannot run.	Without a kernel, an OS cannot be run.
Single OS, multiuser OS, multiprocessor OS, etc are the types of OS.	Monolithic and Micro kernels are examples for the types of kernel.
It is the first program to begin when the computer boots up.	It is the first program to start when the OS runs.