Operating Systems

Dr. B.Majhi

Department Of Computer Science and Engineering

National Institute Of Technology Rourkela

Introduction

What is an OS?

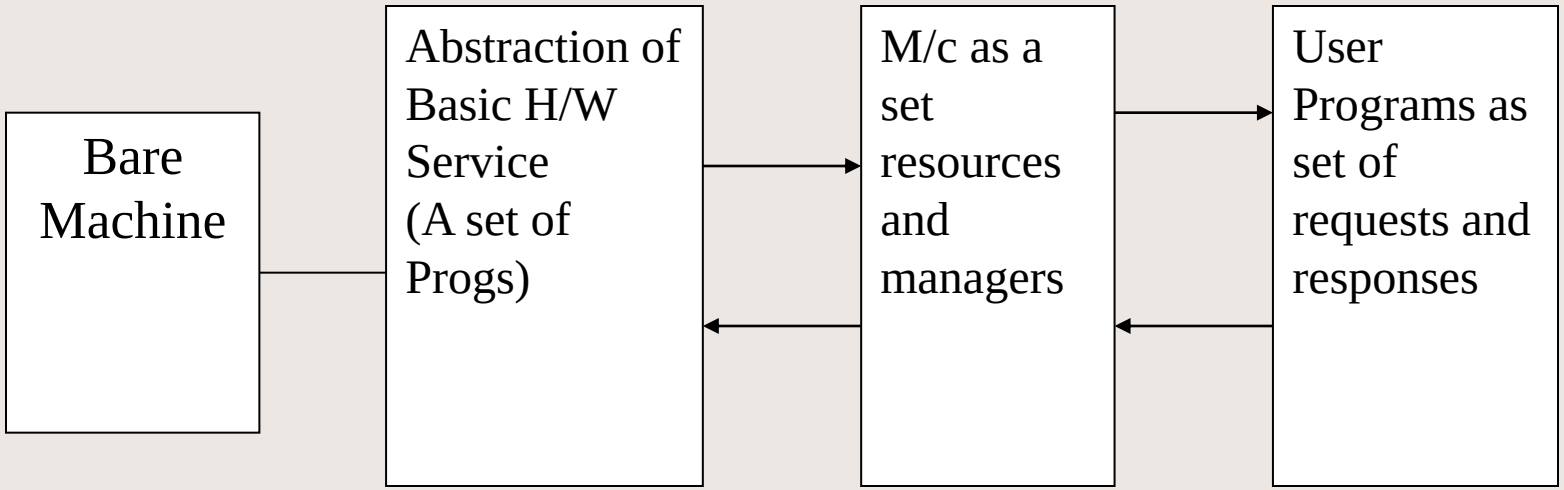
- A collection of systems programs entrusted with effective utilization of a Computer System.

- Effective utilization :
  - Is it efficient, good service(quick response), flexibility, convenience
  - No unique interpretation of effective : Have meaning in a context e.g. space time trade off, M/c code vs. HLL

- OS objectives are complex functions of
  - System Configuration : CPU characteristics, Memory (speed and capacity)
  - User expectations : number crunching, databases, graphics, real time applications
  - Usage environment: Batch/interactive, Distributed/non-distributed, Isolated/Networked

Operating Systems

- OS acts like a Govt., Manager of resources, Interface between user and M/C (virtual m/c)
- OS=S/W infrastructure added on Bare hardware
- computer hardware + operating system = usable machine



Goals of an operating system

1. To maximise utilisation of the hardware resources for computing user jobs - sharing the hardware resources among different user programs
2. To maximise the productivity of users using the system by maximising the usability of the system - presenting an easy to use interface for managing data and program execution
3. To provide a system call interface for software developers to use - eliminating need for application programmers to write their own software to manipulate hardware.

Evolution of OS

- Changes in design perspectives, and Computer Architectures
- Need to study to introduce the concept of sharing and protection, fundamental terminology
- Unit of Work: Executing a program: It may consists several activities like compiling, , linking, loading etc. Hence a unit work can be replaced by the term JOB.
- A job is a **sequence** of related programs: If one fails other cannot proceed. Job and program is used interchangeably.

Early Systems – Bare machine (early 1950s)

- Structure
  - Large machines run from console
  - Single user system - Programmer/User as operator
  - Paper tape or punched cards
- Early Software - Assemblers, compilers, Linkers, loaders + device drivers
- Secure
- Inefficient use of expensive resources
  - Low CPU utilization
  - Significant amount of setup time

Simple Batch Systems

- Hire an operator
- User != operator
- Reduce setup time by batching similar jobs
- Automatic job sequencing – automatically transfers control from one job to another. First rudimentary operating system.
- Resident monitor
  - initial control in monitor
  - control transfers to job
  - when job completes control transfers back to monitor

Simple Batch Systems

- Batch Monitor
  - Current Job of Batch
  - Punched card and .BAT file of DOS
  - Standard control cards : specify start and end of job

Example:

//JOB

//EXEC FORTRAN → FORTRAN source program

/*

//EXEC → Data

/* →End of data

/& → End of job car

Batch Monitor

Batch Monitor

- Resides permanently in memory, hence called resident monitor

Functions

- Initiates a job
- Check the normal or abnormal end of job

H/W and S/ W support

- Hardware memory protection
- Read operations should be performed through the batch monitor to detect end of data and end of job cards.

Parts of resident monitor

- Control card interpreter – responsible for reading and carrying out instructions on the cards.
- Loader – loads systems programs and applications programs into memory.
- Device drivers – know special characteristics and properties for each of the system's I/O devices.

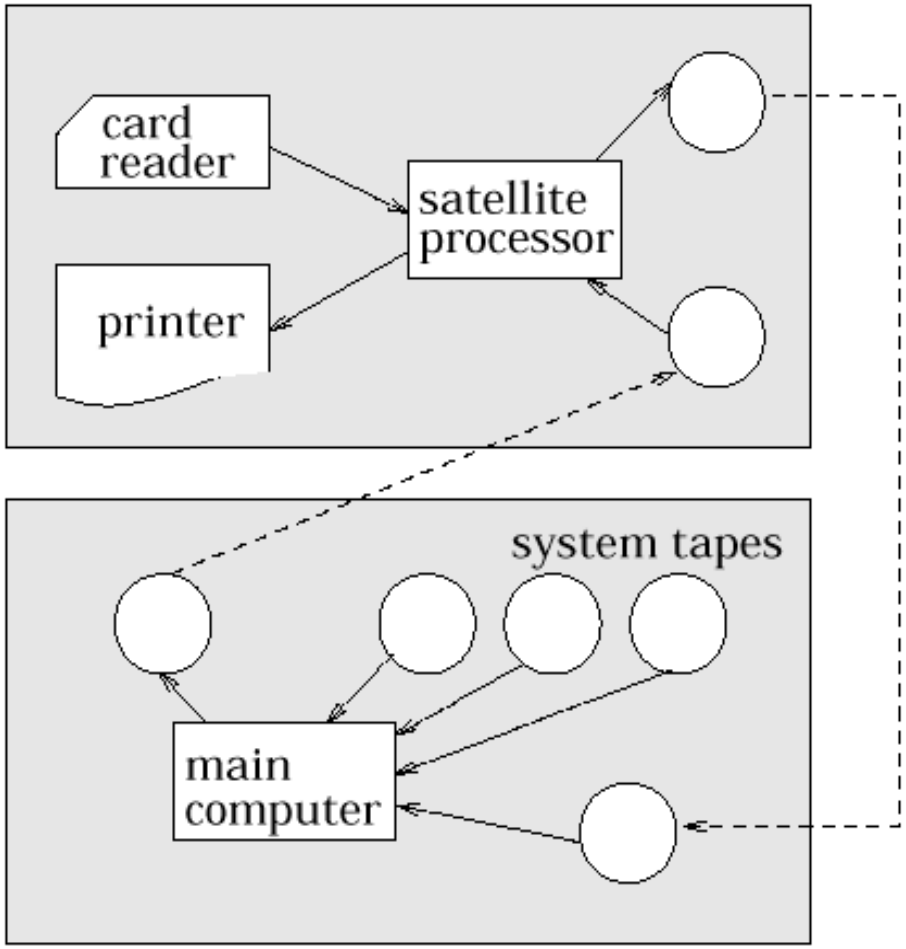
- Problem:

  Slow Performance – I/O and CPU could not overlap; card reader very slow.

- Solution:

  Off-line operation – speed up computation by loading jobs into memory from tapes,with card reading and line printing done off-line.

Batch with Off-Line Operation



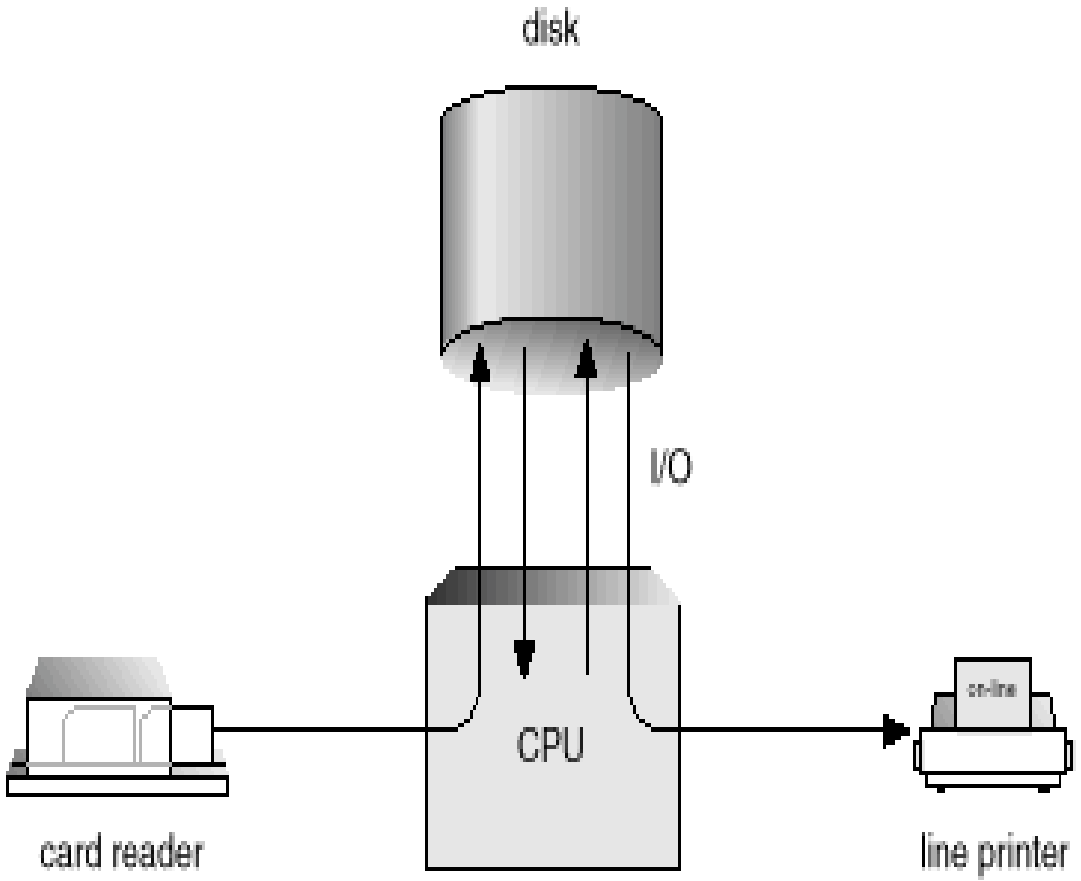
Off-Line Operation (Cont.)

- Advantage – main computer not constrained by the speed of the card readers and line printers, but only by the speed of faster magnetic tape units.
- No changes need to be made to the application programs to change from direct to off-line I/O operation.
- Real gain – possibility of using multiple reader-to-tape and tape-to-printer systems for one CPU.

Batch with Spooling

- Overlap I/O of one job with computation of another job. While executing one job, the OS:
  - reads next job from card reader into a storage area on the disk (job queue).
  - outputs printout of previous job from disk to printer.
  - Spool = Simultaneous Peripheral Operation On-Line
- Job pool – data structure that allows the OS to select which job to run next in order to increase CPU utilization.

Spooling



Benefits of Batch

- TAT: elapsed time between submission of the job for processing and the return of its results to the user. Depends on
- Batch formation time,
- Execution of all the jobs in the batch
- Time involved in printing and sorting the jobs
- Hence TAT of a job does not relate at all to its own execution time. Thus user is not benefited as compared to one-job-at-a-time environment
- But M/C utilization improves, benefit all users indirectly i.e. Throughput increases.

Sharing & Protection

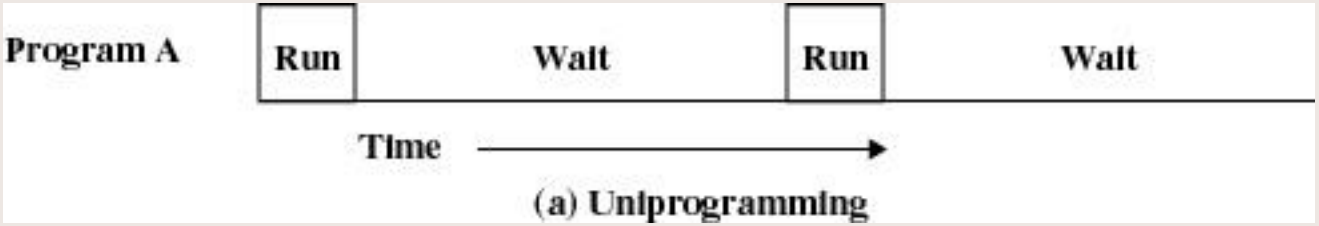
- Protecting user programs from each other
- At any time one user job resides in memory, it shares memory with resident monitor. Fence register for protection
- Is protection necessary among users, Yes…..
- Even if resources are accessed serially by a user program, it could indirectly interfere with next job. It happens if it reads past its own end-of-data. The next job will not execute properly.
- Hence all input/output are done thru resident monitor, to avoid this situation

Multiprogramming OS

- The objective of a MP system is to reduce the wastage of CPU power due to idling during input-output operations.
- Early systems were using IO operations through CPU, and no processing was performed during data transfer. When I/O gets completed, CPU is interrupted and next instruction was getting executed.
- To avoid CPU involvement I/O Channels, DMA Device were introduced.
- I/O devices are connected to IO channels rather than CPU and have direct paths to memory. An IO channel initiates its operation by getting signal from CPU and interrupts the CPU at the end of data transfer.

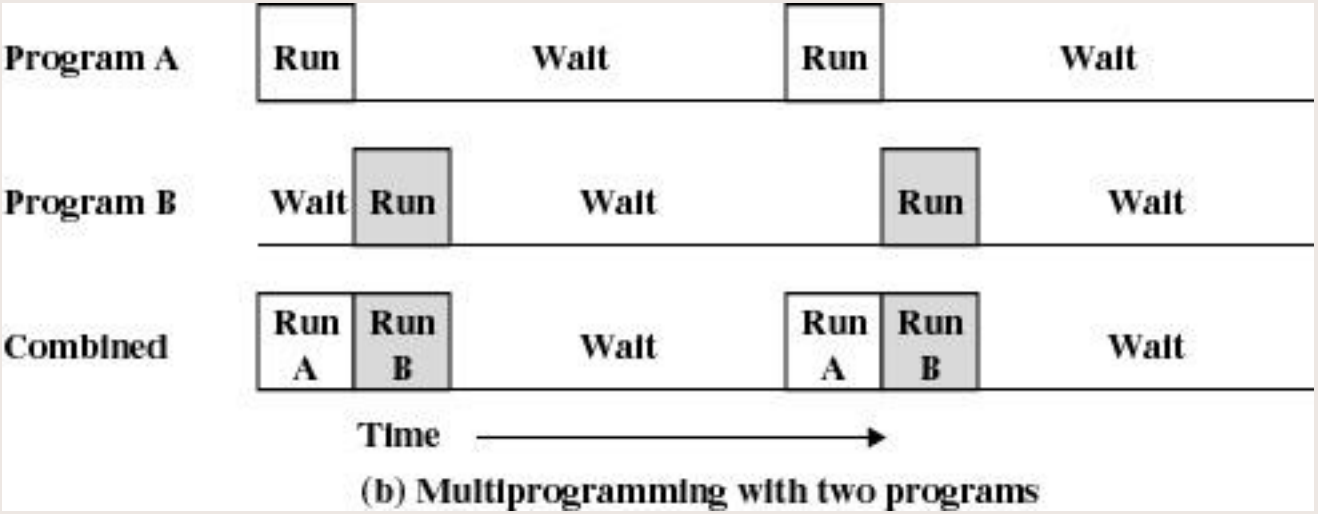
Uniprogramming

- Processor must wait for I/O instruction to complete before preceding

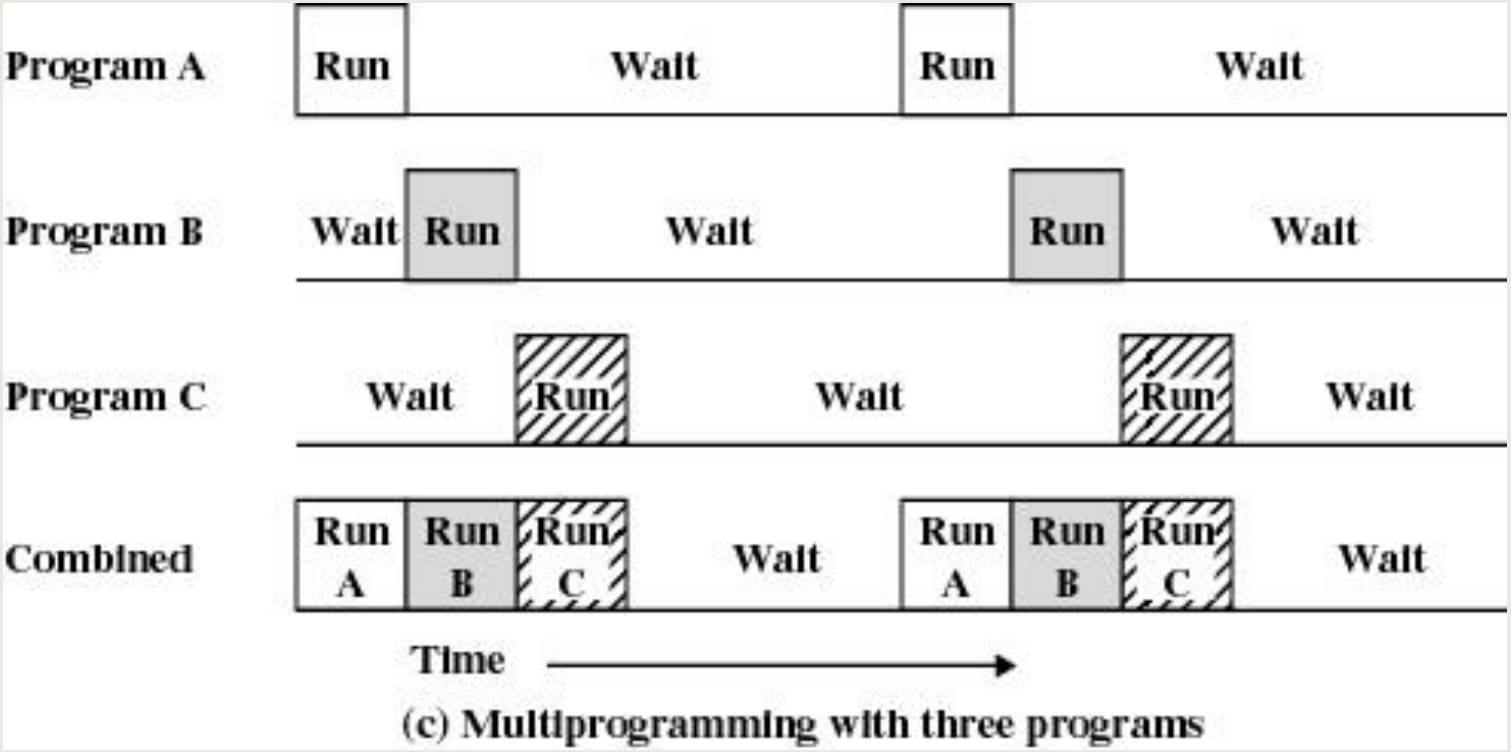


Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



Multiprogramming



(c) Multiprogramming with three programs

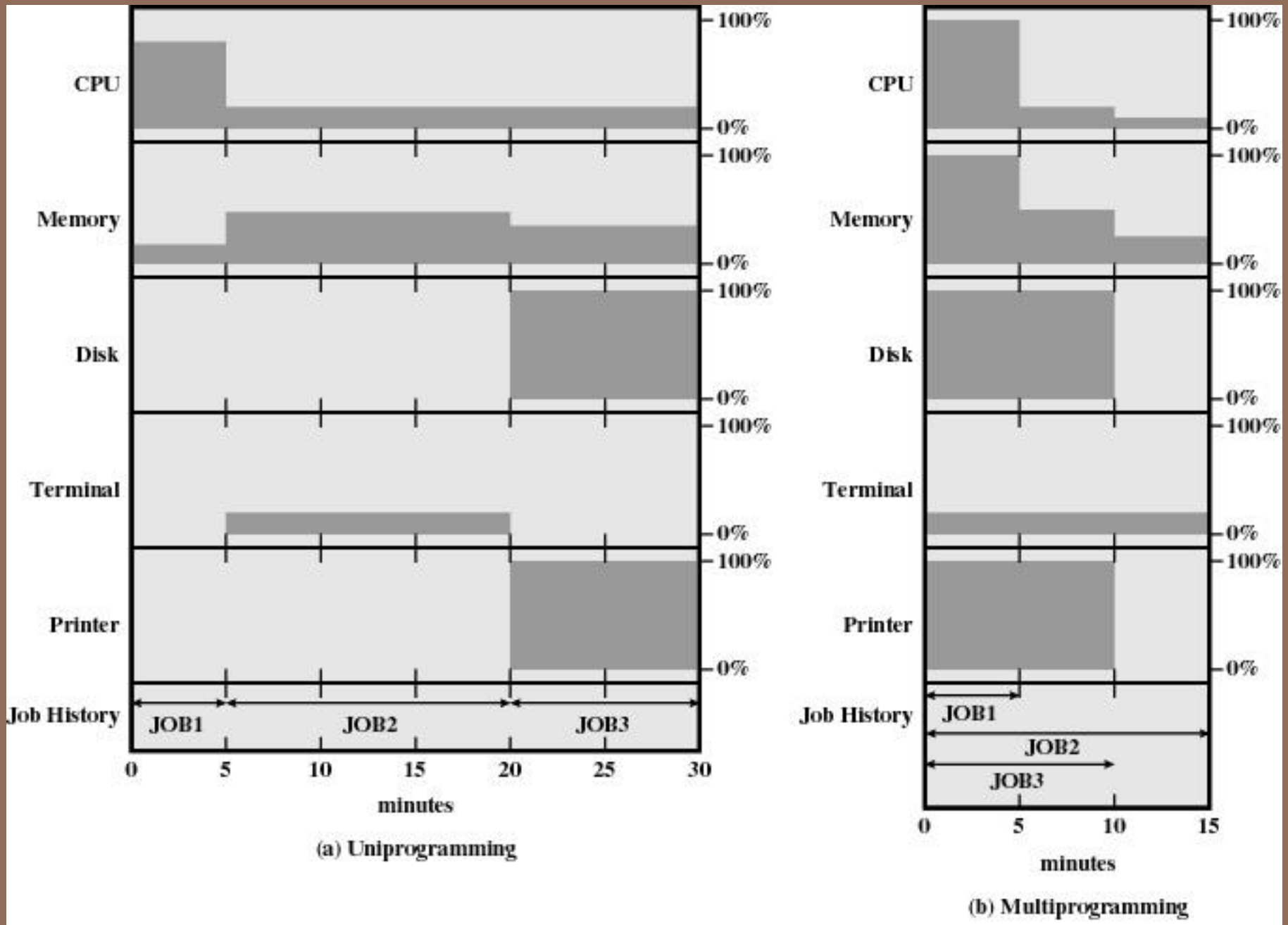


Figure 2.6 Utilization Histograms

Example

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min.	15 min.	10 min.
Memory required	50K	100 K	80 K
Need disk?	No	No	Yes
Need terminal	No	Yes	No
Need printer?	No	No	Yes

Effects of Multiprogramming

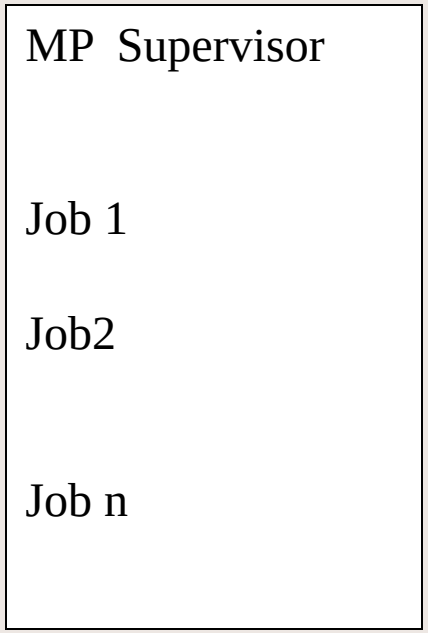
	Uniprogramming	Multiprogramming
Processor use	22%	43%
Memory use	30%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min.	15 min.
Throughput rate	6 jobs/hr	12 jobs/hr
Mean response time	18 min.	10 min.

Multiprogramming OS

- Data Independence
  - IO channels and CPU work concurrently
  - Meaningful, when two do carry out computations with IO in a program
  - READ A,B,C
  - D=A+B
  - It is essential to have data independence of CPU and IO activities, else the program would compute wrong results
- Solutions
  - Buffering the input earlier for computation and fetch next input: how early?
  - Use more than one program in the system. CPU activity of one program is independent of IO activity of other program.(MP utilizes this approach)

Multiprogramming OS

- MP Organization
  - When IO is in progress for job 1 CPU can execute
  - Job 2 and resumes job1 when IO completed
- MP Supervisor functions
  - Allocate different memory areas and different
  - IO devices to the jobs
  - Organize the execution of jobs
  - Determine the start and end of each job's IO
  - Implement protection against the interference in other jobs' IO and processing



Sharing

1. Sharing of memory and IO devices: implemented through mutually exclusive allocation to a job over its lifetime.

2. CPU sharing: notion of *CPU scheduling* and *program priority* as 2/3 jobs may require CPU at the same time.

M/C Architecture

1. Capability to perform CPU and IO activities concurrently: IO Channels, DMA

2. H/W feature for memory protection

3. Privileged state for CPU: IO initiation, IO completions are done in this state. In non-privileged state only user program runs and any attempt to perform IO by such programs would result in hardware interrupt.

Protection (memory)

1. Memory keys

2. Memory Bounds

Memory keys

1. Memory is divided into areas of standard sizes

2. Each area is assigned a memory key, which is typically a small integer number (done by MP OS at the time of memory allocation to all jobs). These keys are stored in special memory, called a *protection key storage*.

3. User job is also given the memory key stored in PSR

4. CPU is only permitted to access a memory with a key which matches with key in PSR, else memory protection violation interrupt is raised.

Memory Bounds

1.Memory start register and memory end register in CPU is checked with every address generated. If outside the range, memory protection violation is raised.

2.The set of control registers of the CPU is called PSR :

(Processor Status Register)

MP Supervisor Functions

1. Allocating memory: Start and end registers are loaded with CPU.

2. IO devices are allocated in a Mutual Exclusive way.

3. Organizing the execution of a job : CPU state is saved before CPU is taken away, so that it can resume the job.

CPU State: control register(PSR) + CPU registers accessible to programs

4.Managing IO operations: IO operations started in privileged mode through system call. The MP supervisor initiates IO on behalf of user and IO completion is indicated by an interrupt to the supervisor.

5. Preventing mutual interference: User job is executed in non-privileged mode. Attempts at interference in other jobs' memory or IO now lead to interrupts.

When and why CPU is taken away from a job?

Internal reasons:

1.Job wishes to perform IO operation

2.Job finishes its execution

External reasons:

1.Violation : memory, resource limits

2.Some other job(high priority) has to be executed

When CPU is given back?

Internal reasons:A job finishes its execution

External reasons:When Systems decide to execute this job

EVENT Monitoring

1. An event is said to occur when the corresponding condition becomes true in the system for the first time.(no even occurs if the condition remains true)

2. To organize the execution of jobs in the system, MP OS monitor the occurrence of events.

- Start of IO for executing job
- IO operation finishes
- Constraint violation for an executing job
- Finish of a job

Program state

Indicates what a program is doing currently and can do at preset moment thru state tag

➔ CPU state + state tag

➔ state tag : Running, ready, blocked, finish

➔ changes to state tag occurs due to events(internal & external)

➔ for event monitoring system uses interrupt h/w and interrupt is generated for each event (s/w or h/w interrupt)

Interrupt

1. Interrupt is recorded by IC field in PSR

2. Current PSR is saved into saved PSR location

3. The PSR is loaded with new PSR (ISR)

4. Contents of new PSR location are defined according to the privileges required for performing the interrupt processing actions

Interrupt Processing Actions

1. ➔Changes the state tag to ready

   ➔Save the CPU state to saved location

2. Analyze the IC and determine the cause of interrupt

3. Perform the appropriate interrupt processing action (serve ISR)

4. Decide next program to run

- Decision to take up next program : priorities
- For effective utilization of resources priorities are set up by MP OS
  - CPU bound job
  - IO bound job
  - Program mix
  - Higher priority to CPU or IO bound jobs
  - Good overlap between CPU and IO activities
  - Priorities to IO bound jobs utilizes the resources properly

❖ Degree of multiprogramming (m): no. of jobs in the system

1. m larger ➔ good utilization, but less memory for each program

2. Hence a trade off decides about m with good program mix

❖ Benefits of MP OS

1.Throughput : improves as better utilization

2.TAT : may not improve, rather may deteriorate

❖ Time Sharing Systems(TSS)

1.To provide better service to users in terms of TAT and response time

2.Interactive vs. non-interactive jobs

❖ Providing good response time

(a) Serve the users by turn

(b) Don't allow monopolization

Resulting policy is RR with time slicing

❖Time slice (δ) : largest chunk of CPU time that a program is allowed to run when scheduled

1. Program may need to perform IO before time slice elapses
2. Program continues to execute for full time slice (timer interrupt)

❖ In any of the cases CPU is taken away, i.e program is pre-empted in the second case

❖ Response time (r) = n * δ ,n is the number of users. But normally r < δ

To reduce r

1. δ should be small , but it will lead to scheduling overhead and r= n * (δ + O), O is the scheduling overhead. For t seconds computational time it has to be scheduled |t/δ| + 1 times before completion. Hence effective response time would be much worse.

2. For large δ, it may increase r as most of the programs will consume all δ seconds of CPU time.

Difference between MP and TSS

1. Dynamic program priority unlike static in MP
2. Degree of MP is determined by acceptable r and δ , not by memory limitations
3. $n = k. (r/\delta)$, k is a small integer
4. no programs can execute larger than δ seconds
5. Memory management : program size > (memory capacity /n) due to swapping possibility
6. At any moment, the system only holds some programs in the primary memory
    a) Next few programs in scheduling queue
    b) Programs performing IO

Real Time OS

1. Is it immediate response
2. Response with short fraction of time
3. No unique definition for real time

Real time application

1. if not appropriate response time system will fail
2. deadline (tolerable response time)
3. Satellite data acquisition system : 10 weather maps / seconds, the response time is 100msecs
4. Weather forecasting system, Airline reservation system

How to satisfy response requirements?

1. Worst case response time should be less than the application's response requirement
2. Situations:

a) Small δ , overhead more, doesn’t guarantee a worst response time

b) Priority to real time applications, improves the response time but interferes the RR scheduling

c) Multitasking : improves the response time in a guaranteed manner (overlapping of IO and processing)

d) Dedicated systems

PORTABLE OS

1. Problems:

   a) Architecture and OS have strong ties due to which computer manufactures face difficulties in marketing new Architectures as OS development take substantial time

   b) Develop new set of skills for each OS and M/c

2. Hence portability is required and standardization is required from both user and architecture point of view
3. Two components

   a) Policy : governs the use of resource, RR policy
   b) Mechanisms : implements the policies, RR implementation

4. Portable OS is structured so as to segregate the policy from the mechanism. This makes the policy implementation of a m/c on which the OS is to be used
5. Mechanism : Kernel of OS to be rewritten for the new m/c
6. UNIX supports portability, layered OS
7. Now work is going on for micro-kernel level to introduce more portability.

UNIX

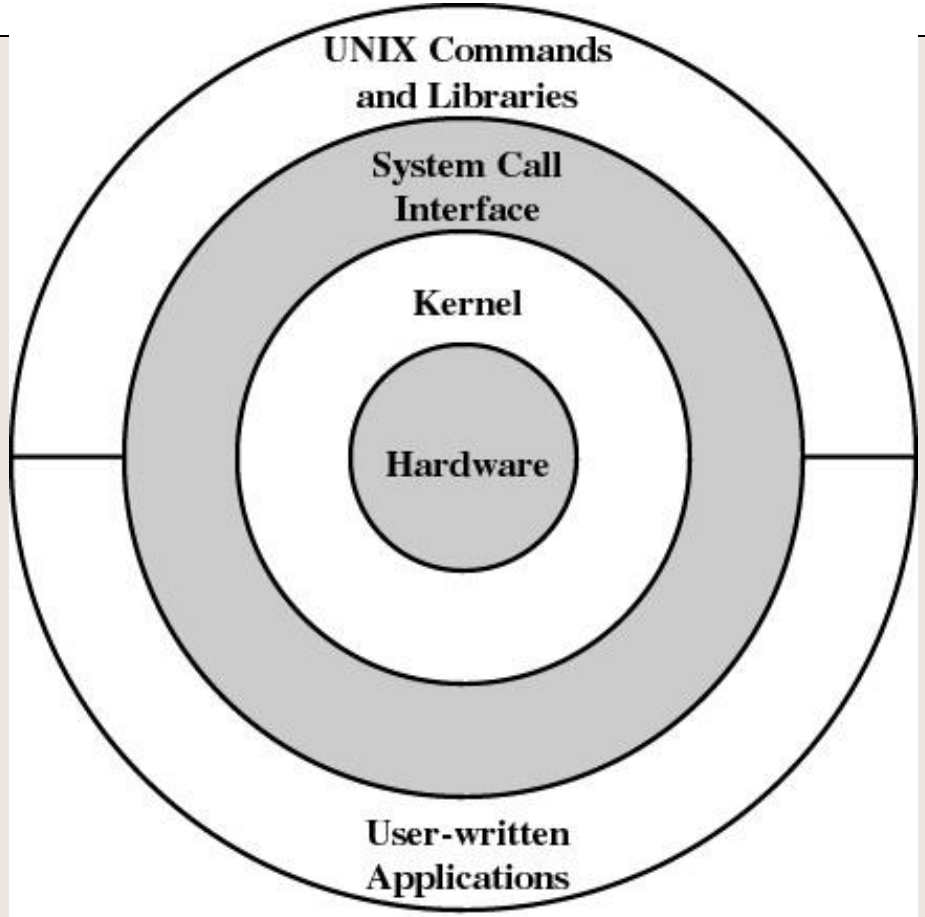


Figure 2.15 General UNIX Architecture

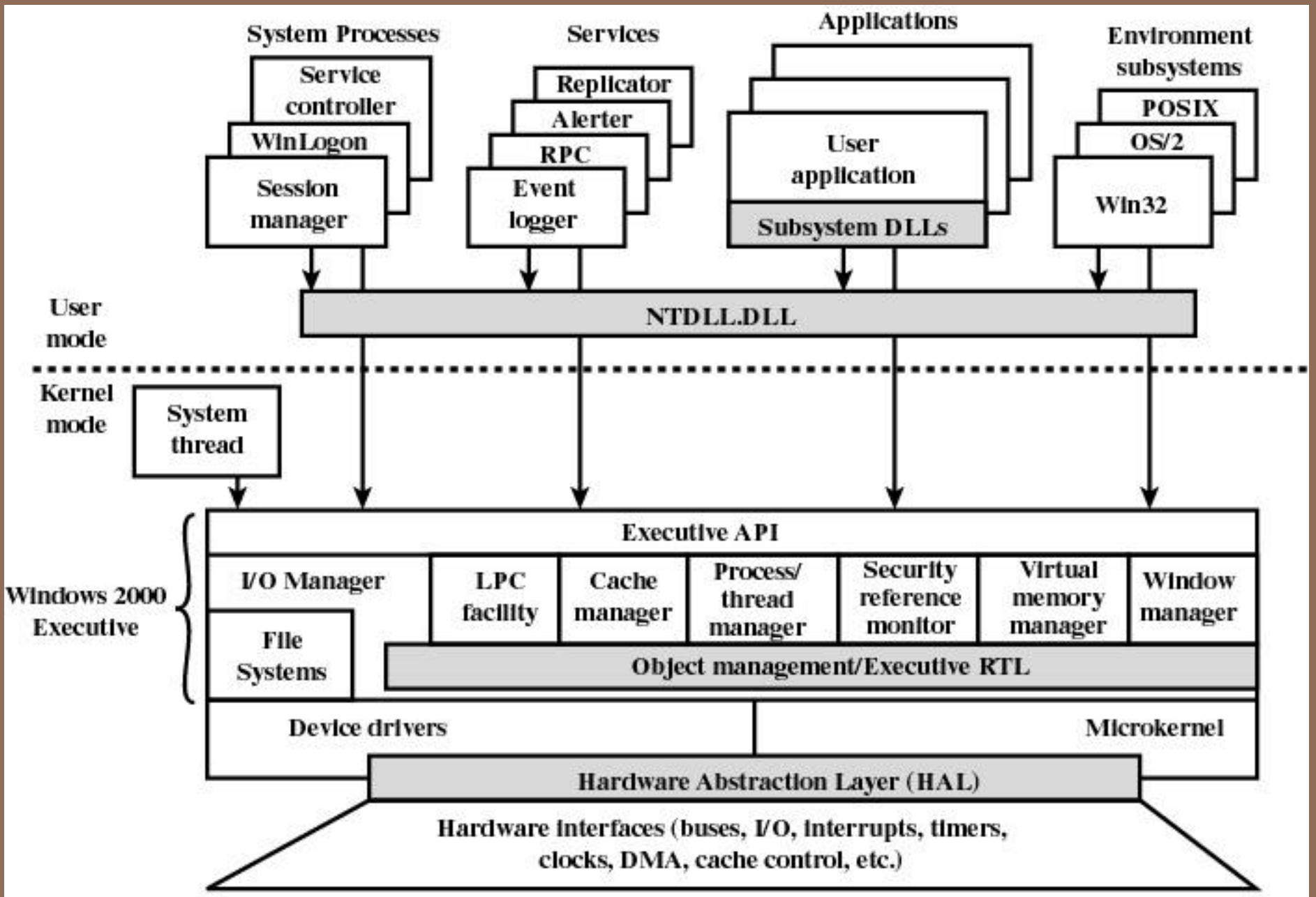


Figure 2.13 Windows 2000 Architecture