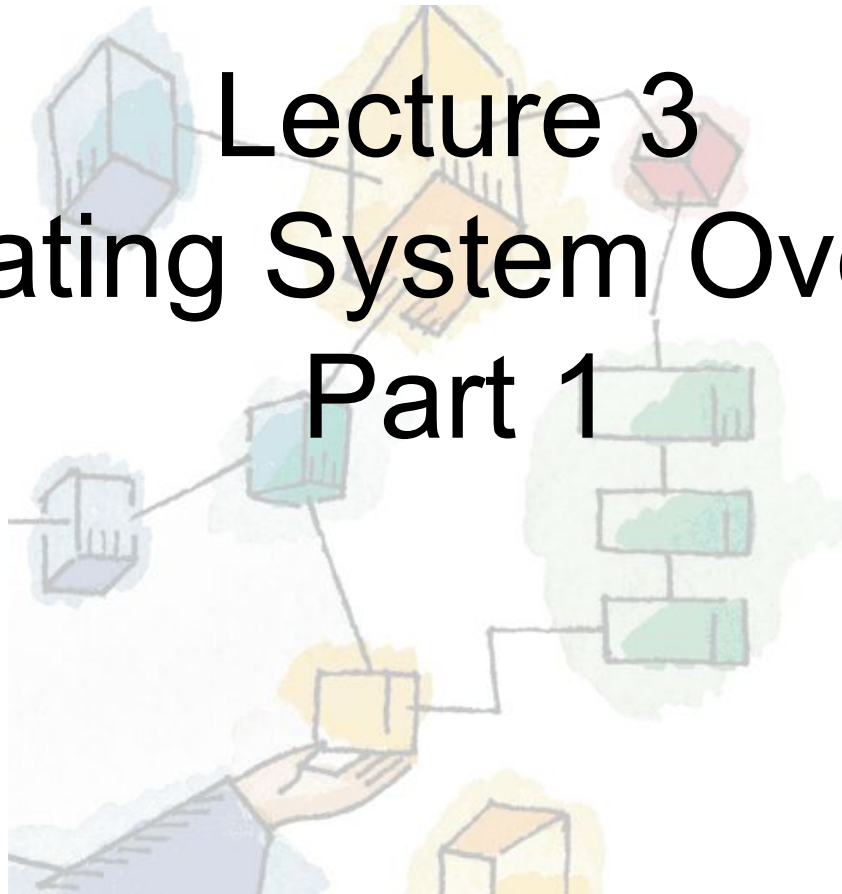


*Operating Systems:
Internals and Design Principles, 6/E*
William Stallings

Lecture 3

Operating System Overview.

Part 1



Patricia Roy
Manatee Community College, Venice,
FL

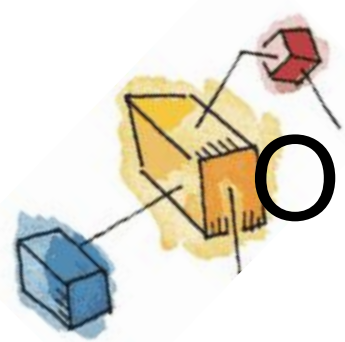
©2008, Prentice Hall



Outline

- **Operating system functions and objectives**
 - The OS as a user/computer interface
 - The OS as a resource manager
 - Ease of evolution of an OS
- **Evolution of operating systems**
 - Serial processing
 - Simple batch systems
 - Multiprogrammed batched systems
 - Time-sharing systems

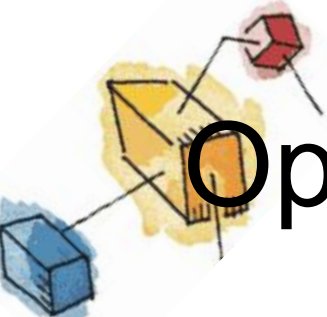




Operating system functions

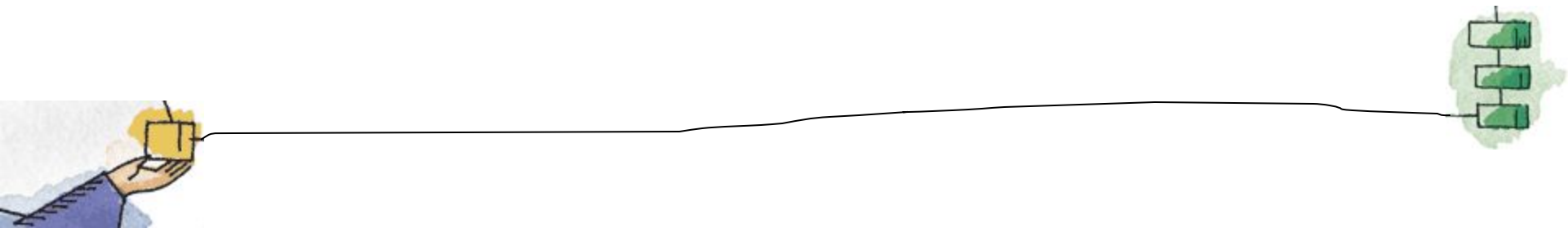
- A program that controls the execution of application programs
- An interface between applications and hardware





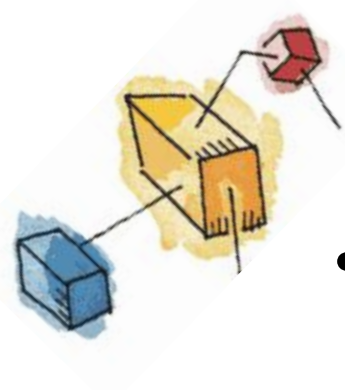
Operating System Objectives

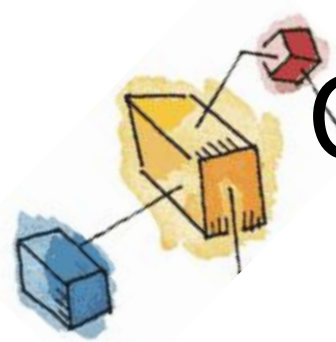
- Convenience
- Efficiency
- Ability to evolve



Outline

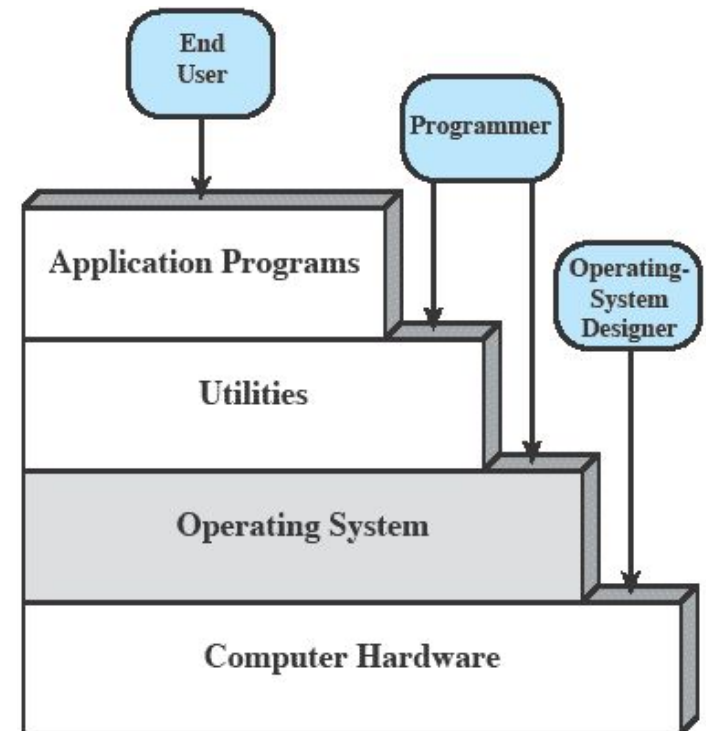
- Operating system functions and objectives
 - **The OS as a user/computer interface**
 - The OS as a resource manager
 - Ease of evolution of an OS
- Evolution of operating systems
 - Serial processing
 - Simple batch systems
 - Multiprogrammed batched systems
 - Time-sharing systems

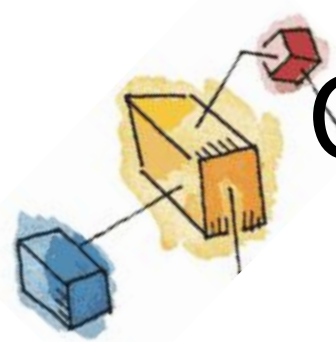




Convenience: the OS as a user/computer interface

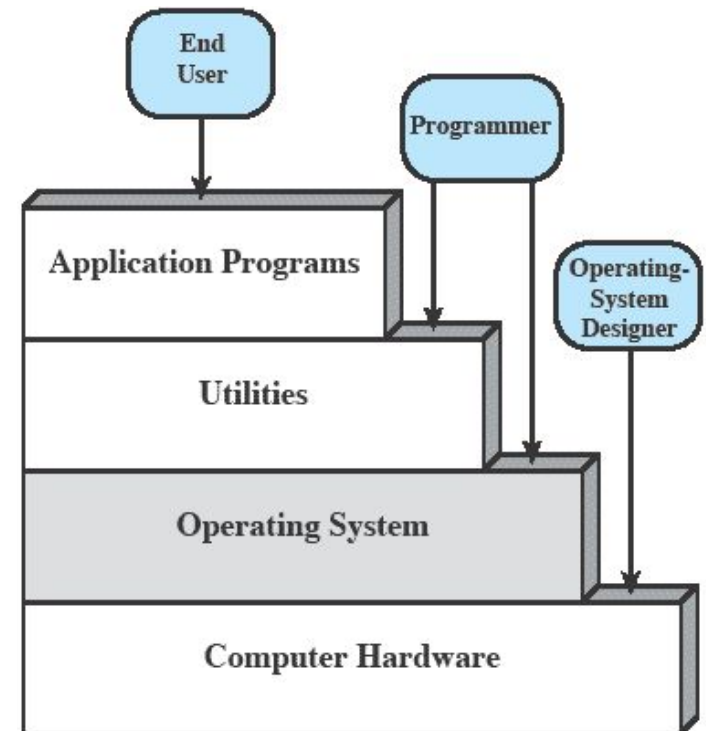
- The hardware and software viewed in a layered or hierarchical fashion
- The end user
 - not concerned with the details of the computer hardware
 - views a computer system in terms of a set of applications

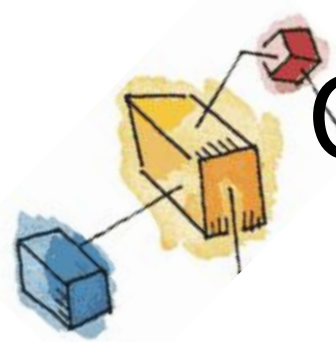




Convenience: the OS as a user/computer interface

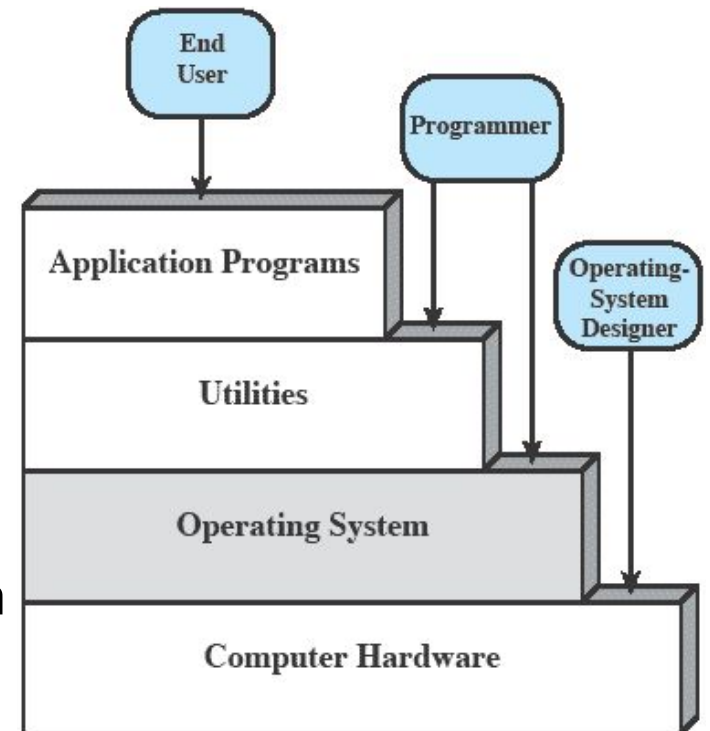
- An application
 - developed by an application programmer
 - expressed in a programming language
- Develop an application as a set of machine instructions
 - programmer completely responsible for controlling the computer hardware
 - overwhelmingly complex





Convenience: the OS as a user/computer interface

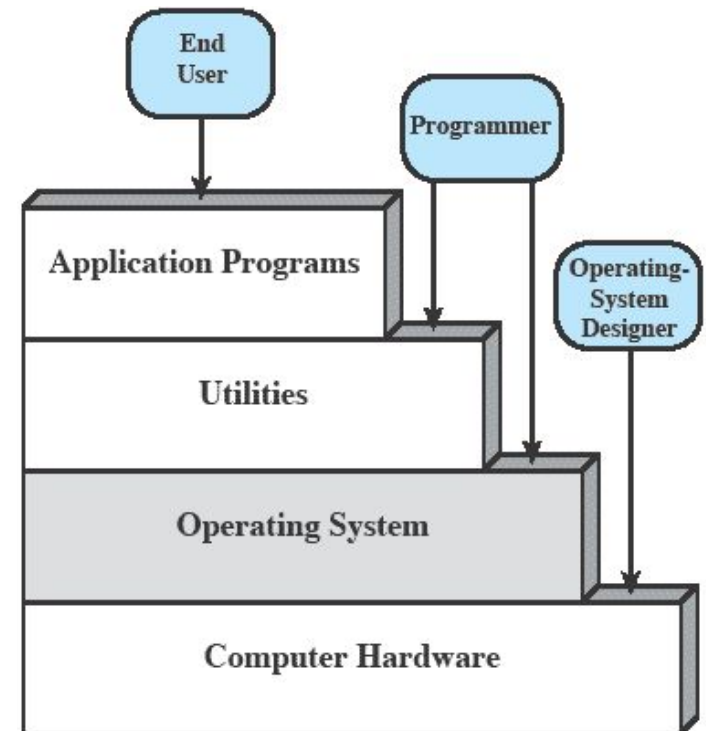
- Develop an application with a set of system programs
- Utilities implement frequently used functions that assist in
 - program creation
 - the management of files
 - the control of I/O devices
- Utilities are
 - used by a programmer to develop an application
 - invoked by an application to perform certain functions

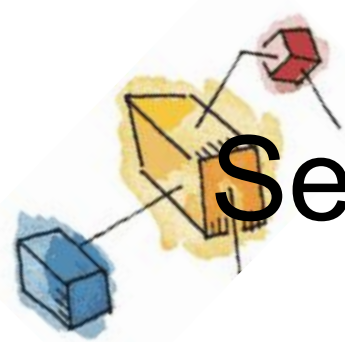




Convenience: the OS as a user/computer interface

- The most important collection of system programs comprises the OS
- The OS
 - masks the details of the hardware from the programmer
 - provides the programmer with a convenient interface for using the system
 - makes it easier for the programmer/application to access/use the facilities and services

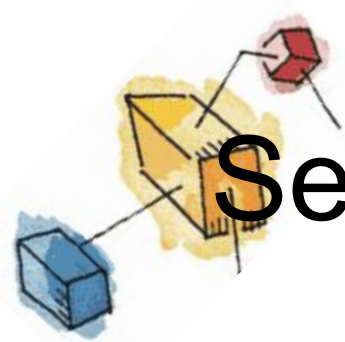




Services Provided by the OS

- Program development
 - Editors and debuggers – assist the programmer in creating programs
- Program execution
 - To execute a program
 - instructions and data must be loaded into main memory
 - I/O devices and files must be initialized
 - other resources must be prepared





Services Provided by the OS

- Access to I/O devices
 - Each I/O device requires its own set of instructions/control signals for operation
 - The OS provides
 - a uniform interface that hides these details
 - I/O devices are accessed using simple reads and writes





Services Provided by the OS

- Controlled access to files
 - The OS reflects
 - the nature of the I/O device (disk drive, tape drive)
 - the structure of the data contained in the files on the storage medium
 - with multiple users, provides protection mechanisms to control access to the files





Services Provided by the OS

- System access
 - For shared or public systems, the OS
 - controls access to the system as a whole
 - controls access to specific system resources
 - provides protection of resources and data from unauthorized users
 - resolves conflicts for resource contention





Services Provided by the OS

- Error detection and response
 - Internal and external hardware errors
 - a memory error
 - a device failure or malfunction
 - Software errors
 - division by zero
 - attempt to access forbidden memory location
 - inability of the OS to grant the request of an application
 - The OS must provide a response that clears the error condition with the least impact on running apps
 - ending the program that caused the error
 - retrying the operation
 - reporting the error to the application





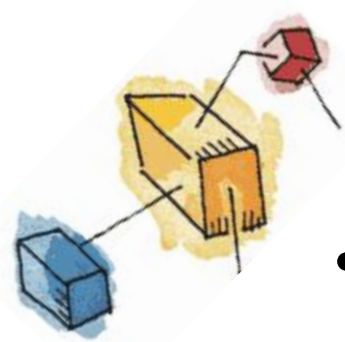
Services Provided by the OS

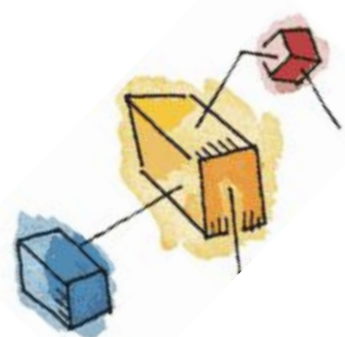
- Accounting
 - Collect usage statistics for various resources
 - Monitor performance parameters
 - response time
 - On any system, used to
 - anticipate the need for future enhancements
 - tune the system to improve performance
 - On a multiuser system, used for
 - billing purposes



Outline

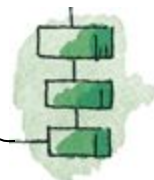
- Operating system functions and objectives
 - The OS as a user/computer interface
 - **The OS as a resource manager**
 - Ease of evolution of an OS
- Evolution of operating systems
 - Serial processing
 - Simple batch systems
 - Multiprogrammed batched systems
 - Time-sharing systems

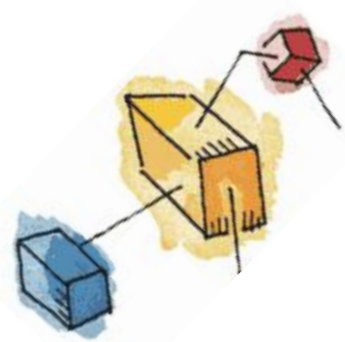




Efficiency: the OS as a resource manager

- Responsible for managing resources
 - the movement, storage, processing of data
 - the control of these functions
- Normally, a control mechanism is
 - external to that which is controlled
 - a distinct and separate part of that which is controlled
- With the OS, control mechanism is unusual in two respects

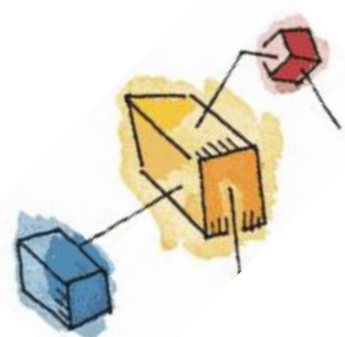




Efficiency: the OS as a resource manager

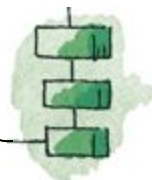
- 1) The OS functions same way as ordinary computer software
 - It is a program that is executed by the processor

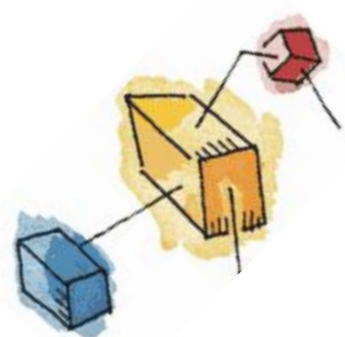




Efficiency: the OS as a resource manager

- 2) The OS relinquishes control for the processor and then resumes control
 - the OS directs the processor in
 - the use of the other system resources
 - the timing of its execution of other programs
 - the **processor** must
 - cease (stop) executing the OS program
 - execute other program





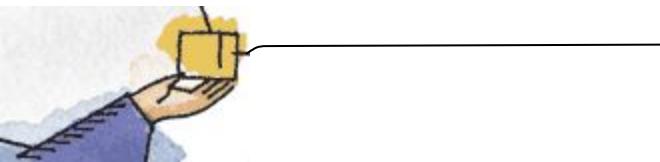
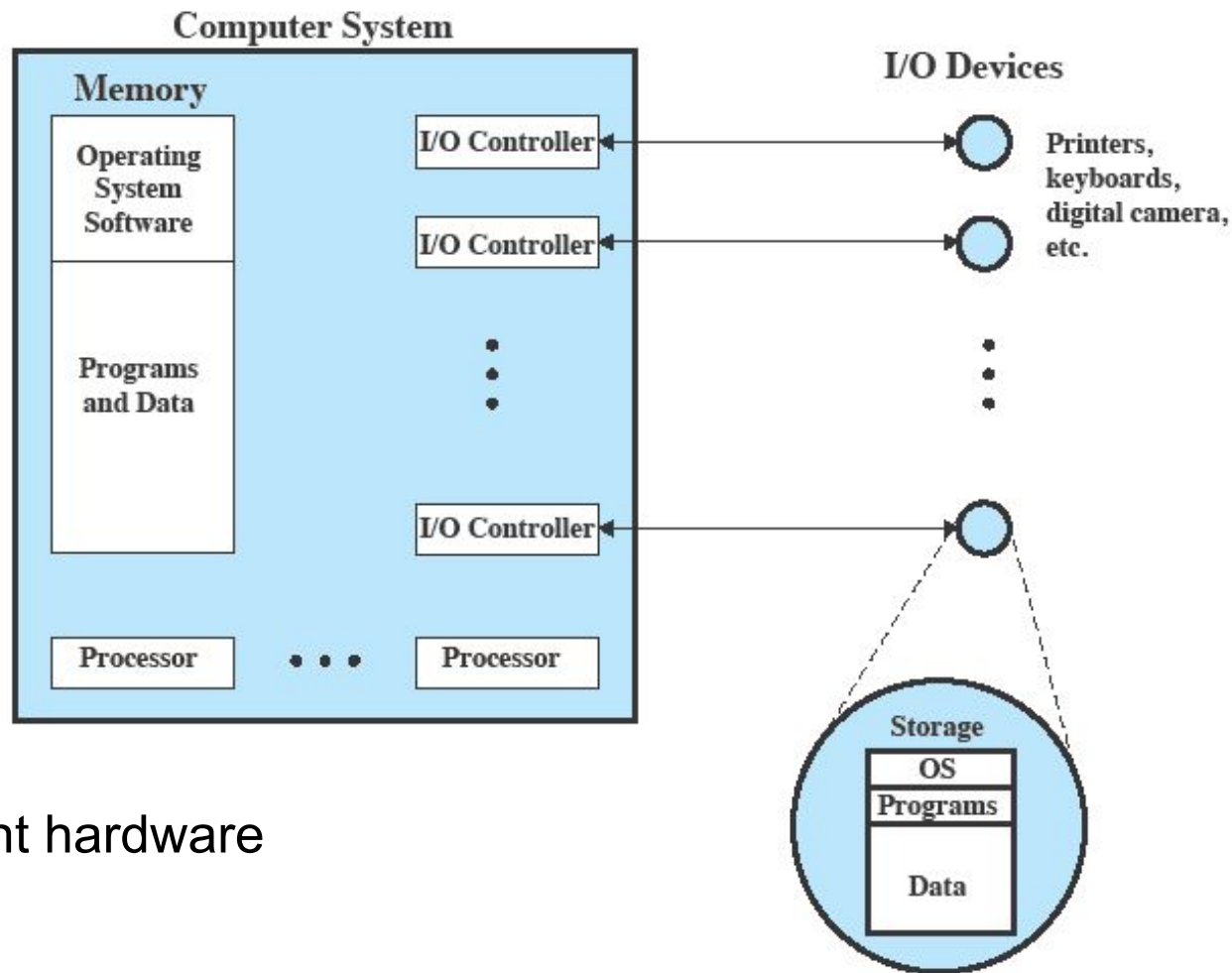
Efficiency: the OS as a resource manager

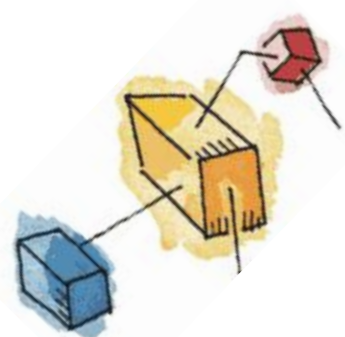
Kernel (also called the nucleus)

- portion of the OS that is in main memory
- contains most frequently used functions

Allocation of main memory is controlled jointly by

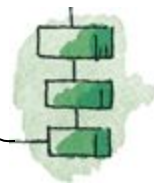
- the OS
- memory management hardware





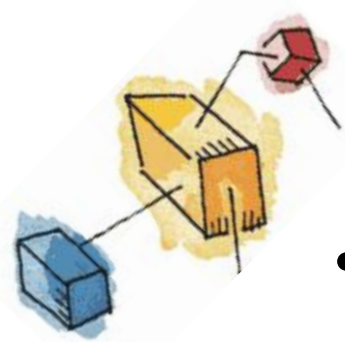
Efficiency: the OS as a resource manager

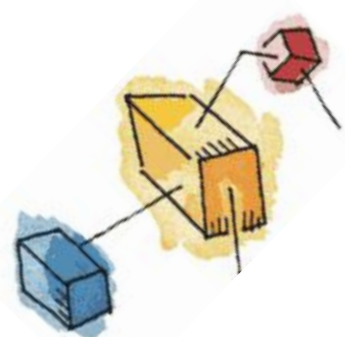
- The OS
 - decides when an I/O device can be used by a program
 - controls access to files
 - controls use of files
- The processor itself is a resource
 - the OS must determine how much processor time is to be devoted to the execution of a particular program



Outline

- Operating system functions and objectives
 - The OS as a user/computer interface
 - The OS as a resource manager
 - **Ease of evolution of an OS**
- Evolution of operating systems
 - Serial processing
 - Simple batch systems
 - Multiprogrammed batched systems
 - Time-sharing systems

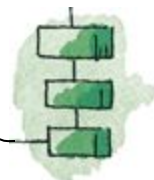




Ability to evolve: ease of evolution of the OS

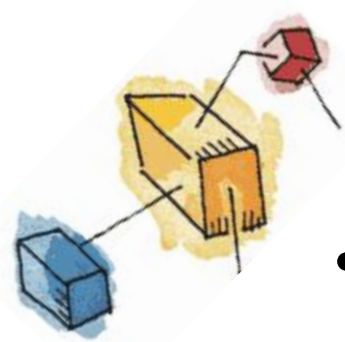
Reasons

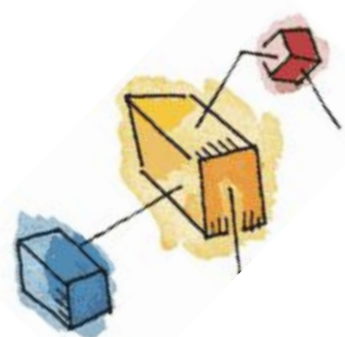
- Hardware upgrades plus new types of hardware
- New services
 - in response to user demand
 - in response to the needs of system managers
- Fixes
 - any OS has faults -> fixes are made (may introduce new faults)



Outline

- Operating system functions and objectives
 - The OS as a user/computer interface
 - The OS as a resource manager
 - Ease of evolution of an OS
- **Evolution of operating systems**
 - **Serial processing**
 - Simple batch systems
 - Multiprogrammed batched systems
 - Time-sharing systems





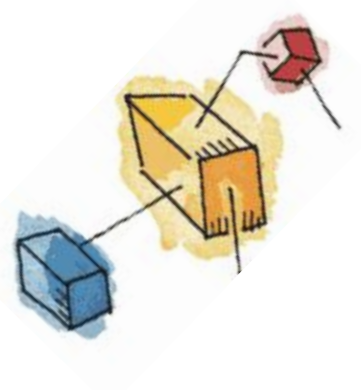
Evolution of Operating Systems

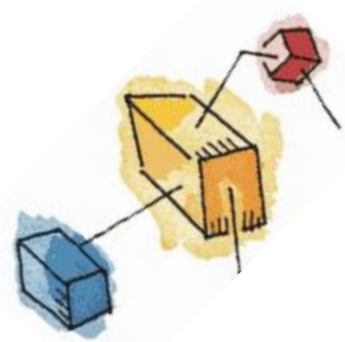
- Serial processing (late 1940s-mid-1950s)
 - Programmer interacted with the computer hardware - there were no OS
 - Computers were run from a console consisting of
 - Display lights
 - Toggle switches
 - Some form of input device
 - Printer



Serial processing (late 1940s-mid-1950s)

- Programs in machine code were loaded via input device (card reader)
- If an error halted the program, the error condition was indicated by the lights
- If the program proceeded to a normal completion, the output appeared on the printer





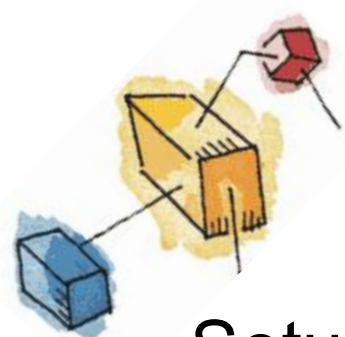
Serial processing (late 1940s-mid-1950s)

The early systems presented two main problems

– Schedule time

- Hardcopy sign-up sheet to reserve computer time
- Sign-up for an hour and finish in 45 minutes => resulted in wasted computer processing time
- Not finish in the allotted time => run into problems, forced to stop before resolving the problem



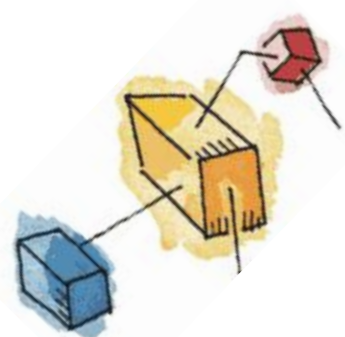


Serial processing (late 1940s-mid-1950s)

– Setup time

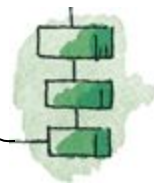
- A single program (called **job**) involved
 - loading the compiler
 - loading source program (high-level language program)
 - saving compiled program (object program)
 - loading and linking together object program and common functions
- Each of these steps could involve mounting or dismounting tapes
- Considerable amount of time was spent just in setting up the program to run





Serial processing (late 1940s-mid-1950s)

- Serial processing – users have access to the computer in series
- Various system software tools developed for efficiency
 - Libraries of common functions
 - Linkers
 - Loaders
 - Debuggers
 - I/O driver routines

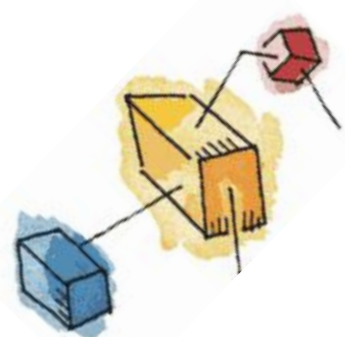




Outline

- Operating system functions and objectives
 - The OS as a user/computer interface
 - The OS as a resource manager
 - Ease of evolution of an OS
- **Evolution of operating systems**
 - Serial processing
 - **Simple batch systems**
 - Multiprogrammed batched systems
 - Time-sharing systems

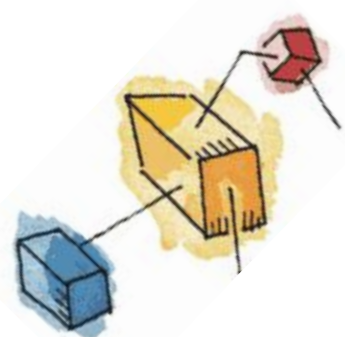




Evolution of Operating Systems

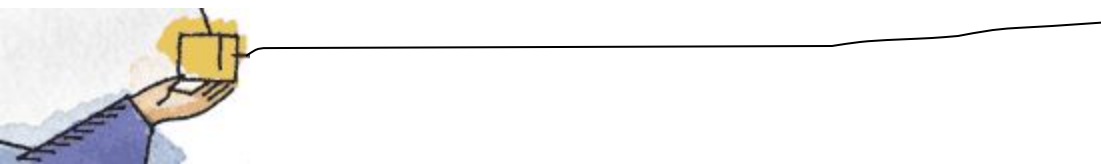
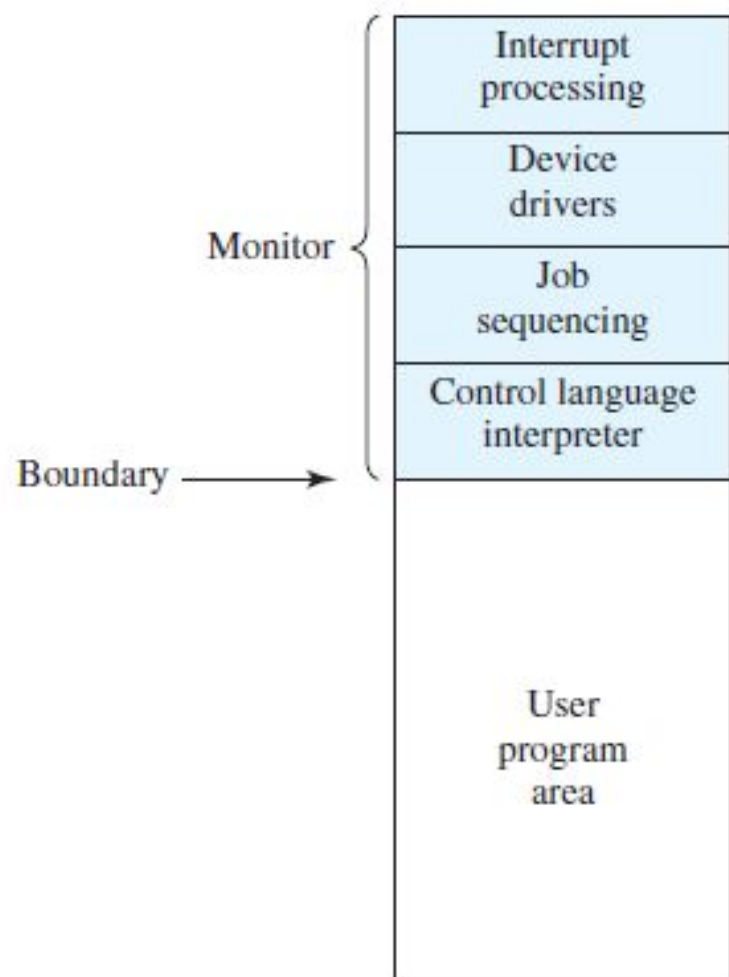
- To improve processor utilization the concept of a batch operating system was developed
- The first developed in mid-1950s by General Motors for the use on an IBM 701
- Simple batch-processing scheme
 - Use of the **monitor**
 - user no longer has direct access to the processor
 - job on cards/tapes submitted to a computer operator
 - computer operator batches the jobs together sequentially and places the entire batch on an input device
 - Each program after completing processing branches back to the monitor
 - The monitor automatically begins loading the next program

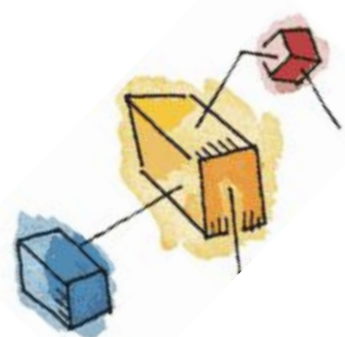




Simple batch systems

- Monitor controls the sequence of events
- It is always in main memory and available for execution (resident monitor)
- Utilities and common functions – loaded as subroutines





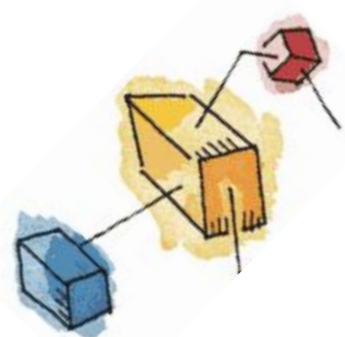
Simple batch systems

- The monitor improves
 - Scheduling
 - a batch of jobs is queued up
 - jobs are executed as rapidly as possible
 - no idle time
 - Setup time
 - with each job, instructions are included in **job control language** (JCL)
 - special type of programming language
 - provides instruction to the monitor
 - what compiler to use
 - what data to use

```
$JOB  
$FTN  
•  
•  
• } FORTRAN instructions
```

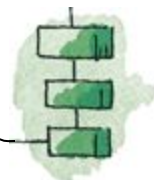
```
$LOAD  
$RUN  
•  
•  
• } Data  
$END
```

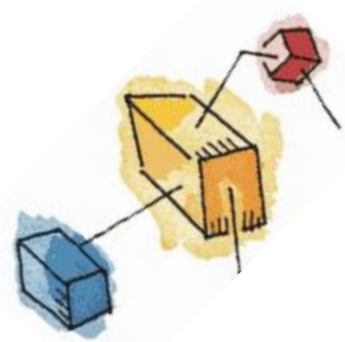




Hardware Features

- Memory protection
 - Does not allow the memory area containing the monitor to be altered
- Timer
 - Prevents a job from monopolizing the system
 - Set at the beginning of each job
 - If expires, user program is stopped
 - Control returns to the monitor

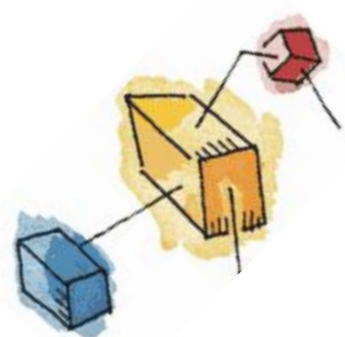




Hardware Features

- Privileged instructions
 - Certain machine level instructions can only be executed by the monitor
 - If a user programs wished to perform I/O, it must request the monitor
- Interrupts
 - Early computer models did not have this capability

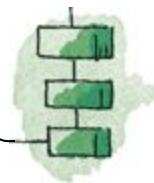




Memory Protection

Modes of operation

- User program executes in user mode
 - Certain instructions may not be executed
- Monitor executes in system mode
 - Kernel mode
 - Privileged instructions are executed
 - Protected areas of memory may be accessed





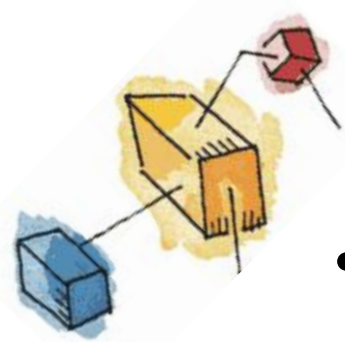
Processor time

- Processor time alternates between
 - execution of user program and
 - execution of the monitor
- Two sacrifices
 - some main memory is given over to the monitor
 - some processor time is consumed by the monitor
- Despite this utilization is improved



Outline

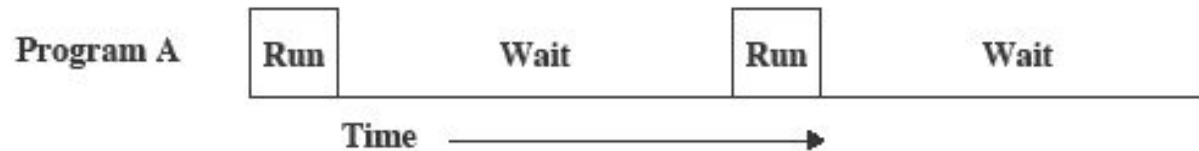
- Operating system functions and objectives
 - The OS as a user/computer interface
 - The OS as a resource manager
 - Ease of evolution of an OS
- **Evolution of operating systems**
 - Serial processing
 - Simple batch systems
 - **Multiprogrammed batched systems**
 - Time-sharing systems





Uniprogramming

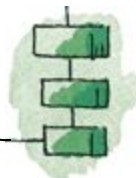
- I/O devices are slow compared to the processor
- Processor must wait for I/O instruction to complete before proceeding



- A file of records. 100 machine instructions performed per record. 96% of time processor spends waiting for the I/O

Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	<u>15 μs</u>
TOTAL	31 μs

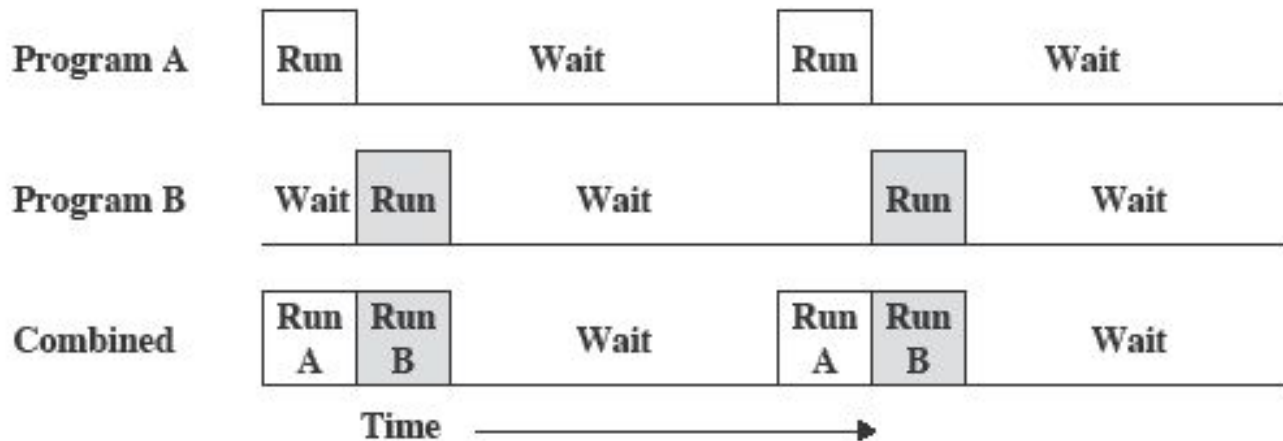
$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$





Multiprogramming / multitasking

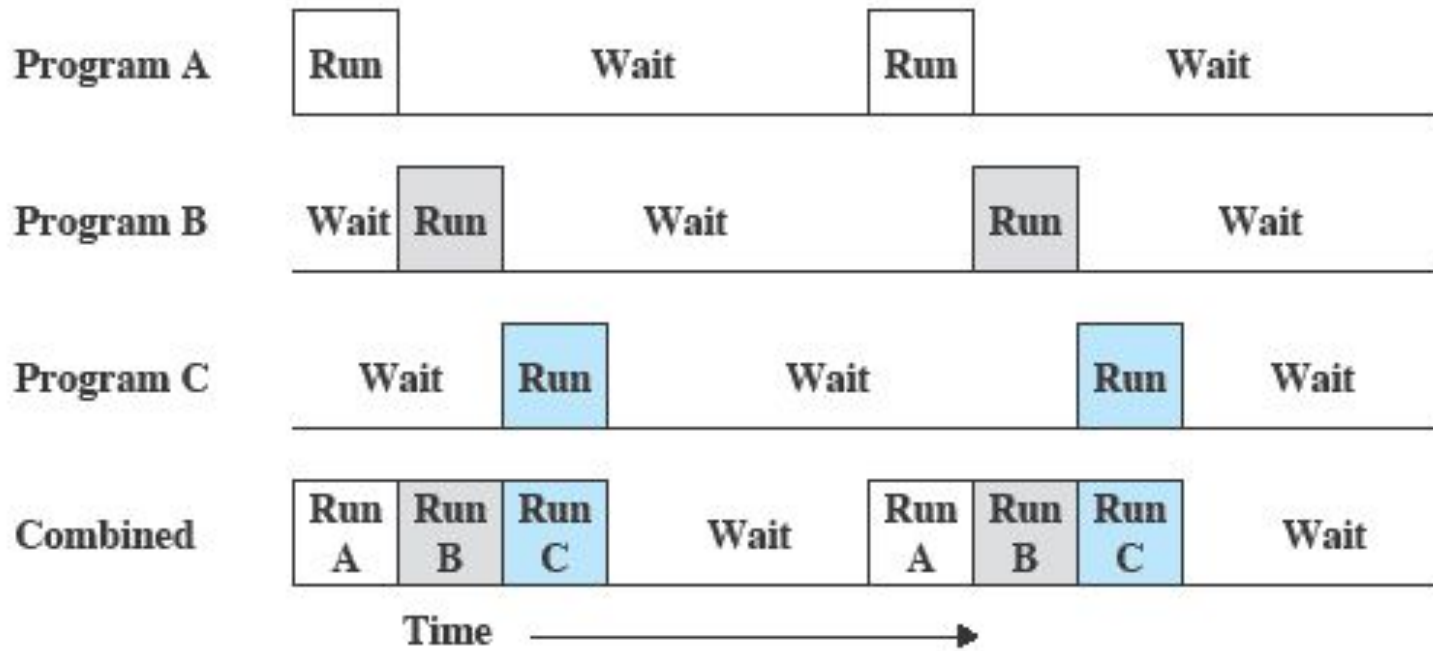
- Memory holds the OS and several programs
- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs



Multiprogramming / multitasking

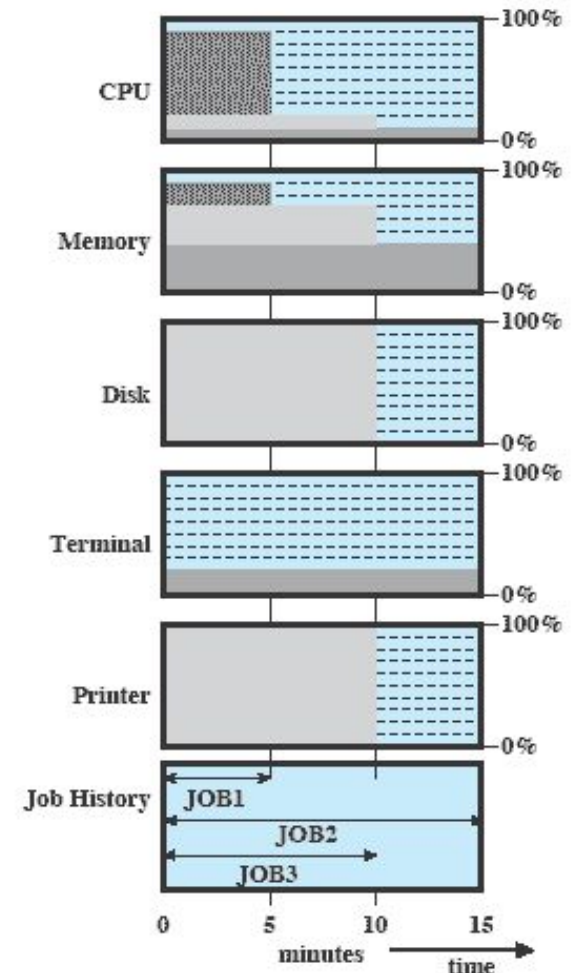
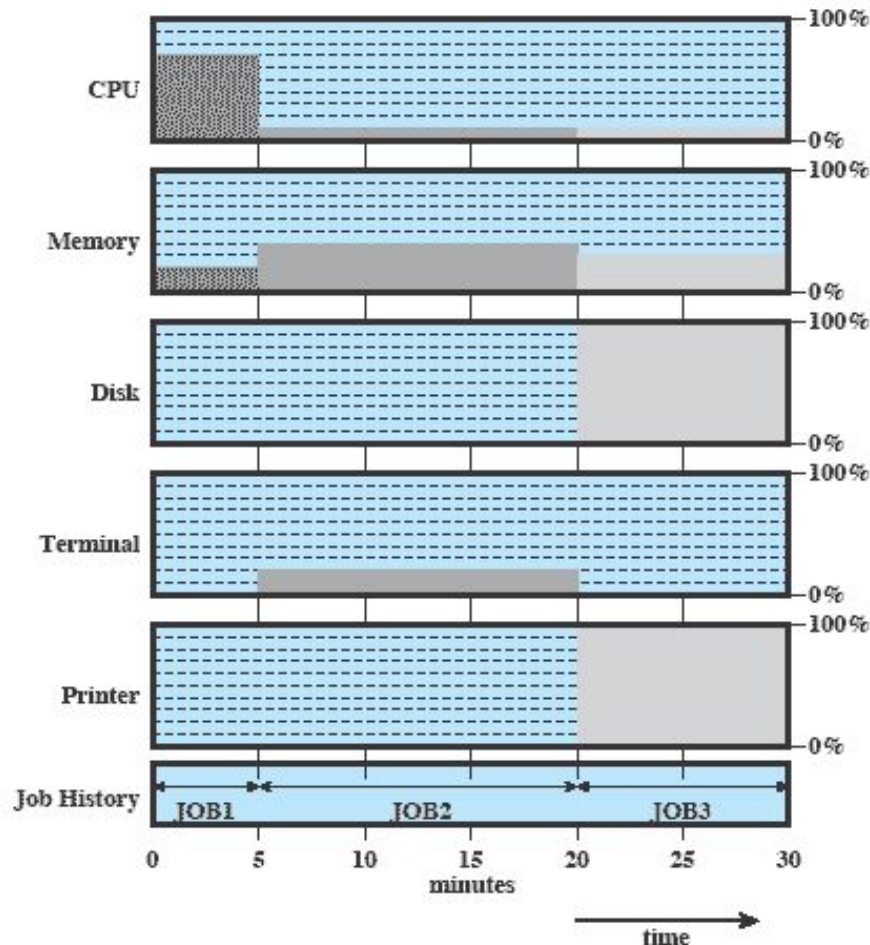


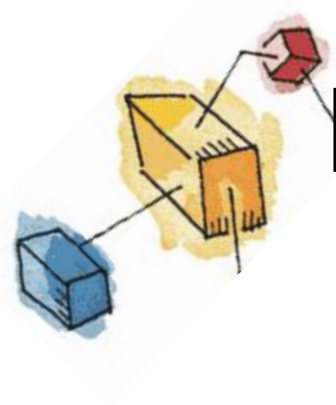
(c) Multiprogramming with three programs



Utilization Histograms

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes





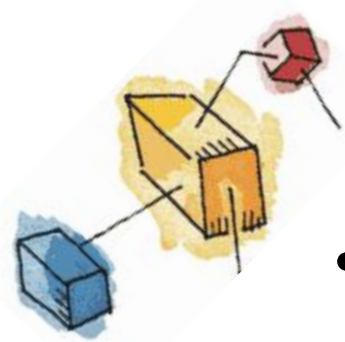
Effect of multiprogramming on resource utilization

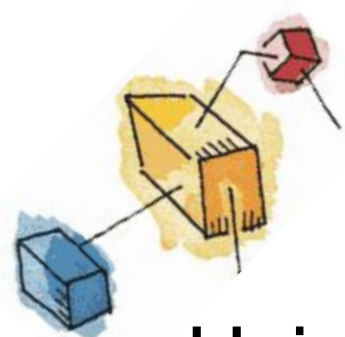
	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min



Outline

- Operating system functions and objectives
 - The OS as a user/computer interface
 - The OS as a resource manager
 - Ease of evolution of an OS
- **Evolution of operating systems**
 - Serial processing
 - Simple batch systems
 - Multiprogrammed batched systems
 - **Time-sharing systems**

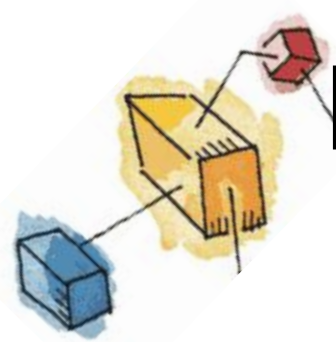




Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor's time is shared among multiple users
- Multiple users simultaneously access the system through terminals
- OS interleaves the execution of each user program in a short burst or quantum of computation

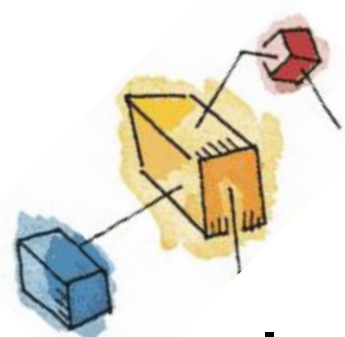




Batch Multiprogramming versus Time Sharing

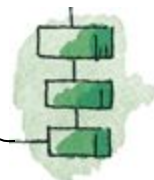
	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

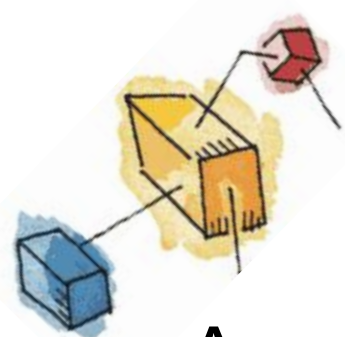




Time Sharing Systems

- In 1961 Project MAC group at MIT developed Compatible Time-Sharing System (CTSS) for IBM 701 (later 7094)
- The system ran on a computer with 32000 36-bit words of main memory
- Resident monitor consumed 5000 words
- User's program and data were loaded into the remaining 27000 words of main memory

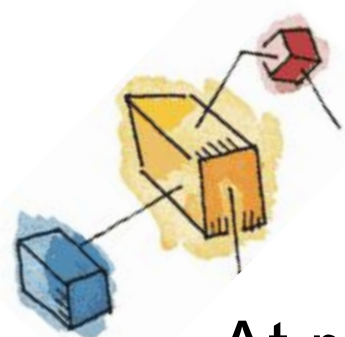




Time Sharing Systems

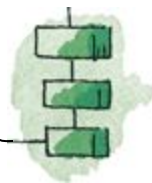
- A system clock generated interrupts at a rate of approximately one every 0.2 seconds
- At each clock interrupt
 - the OS regained control
 - assigned the processor to another user
 - this technique is known as time slicing





Time Sharing Systems

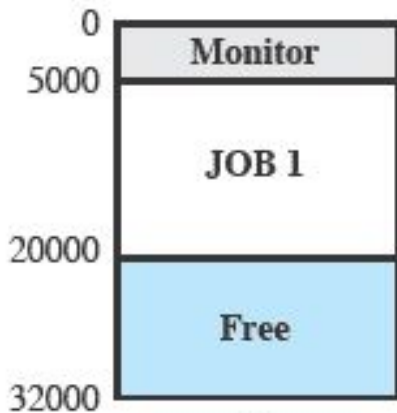
- At regular time intervals
 - the current user would be preempted
 - another user loaded in
- The old user program and data
 - were written out to disk before new user programs and data were read in
 - were restored in main memory when that program was next given a turn
- To minimize disk traffic, user memory was written out when incoming program would overwrite it



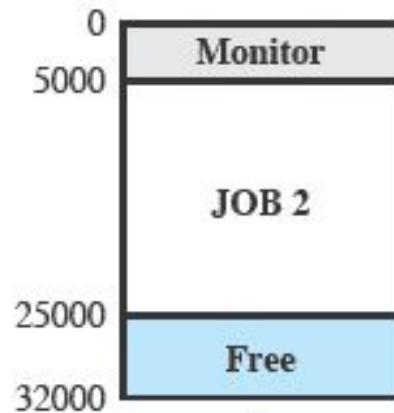
CTSS Operation

- Four interactive users
- Memory requirements

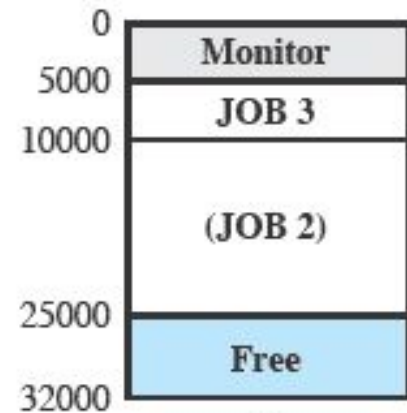
- JOB1: 15,000
- JOB2: 20,000
- JOB3: 5000
- JOB4: 10,000



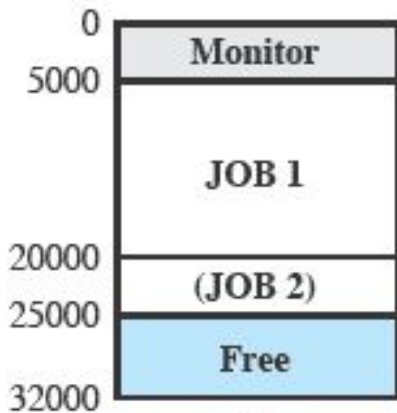
(a)



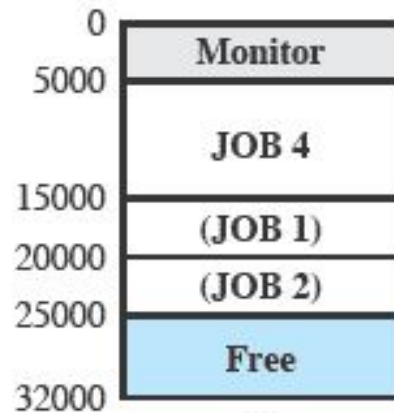
(b)



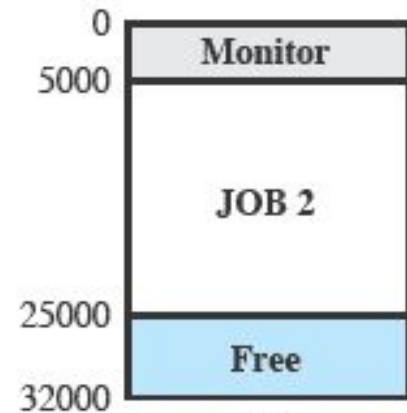
(c)



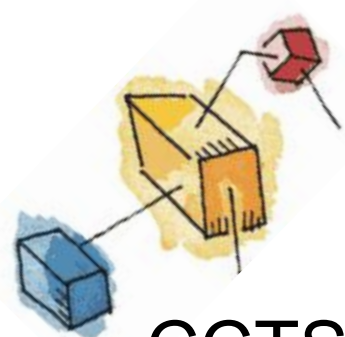
(d)



(e)



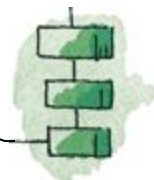
(f)

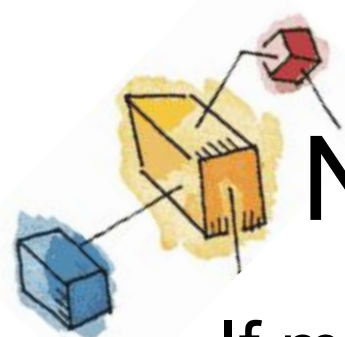


Time Sharing Systems

CCTS approach

- Was extremely simple => minimized the size of the monitor
- A job always loaded into the same locations in memory => there was no need for relocation techniques at load time
- The techniques of writing out minimized disk activity
- Supported a maximum of 32 users (running on IBM 7094)





New problems for the OS

- If multiple jobs are in memory
 - they must be protected from interfering with each other (by modifying each other's data)
- With multiple interactive users
 - the file system must be protected so that only authorized users have access to a particular file
- The contention for resources (printers, mass storage devices)
 - must be handled

