Basics of Computing – Chapter 3
An Introduction to Operating Systems

Cory L. Strope

Introduction
Hardware/Software

Hardware is the physical components of a machine that allow computation to occur.

- Processor, main memory, I/O devices, system bus

Software is programs or sets of instructions that tell the hardware what to do

- Word processor, web browser, operating system.

Software
Classification

Application Software are programs that perform a specific task.

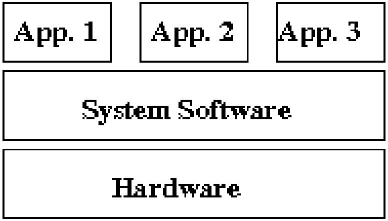
- Application software will vary from computer to computer.
- Examples: Word, Photoshop, video games.

System Software: Programs that operate or maintain a computer.

- Will usually be the same on any computer.
- Windows OS, disk defragmenter, etc.

Software
System Software

System software provides the environment in which application software executes.



What if application software had to deal directly with the hardware?

Software
Operating Systems (OSes)

The most important software on a computer is the **operating system**.

- Manages the execution of programs, allocation of resources and maintenance of the computing environment.
- Very complex: We will limit our survey of OSes to single processor systems.
- Types of OSes:
  - Batch processing
  - Interactive processing
  - Time-sharing
  - Multitasking

Batching Processing

Batch processing is the execution of jobs by collecting them in a single batch, then executing them without further interaction with the user.

- In early days, anyone who wanted to run a program was required to submit it, along with any required data and special directions about the program's requirements to the computer operator.
- The operator would load the materials into the machine's mass storage where the operating system could access them for execution.

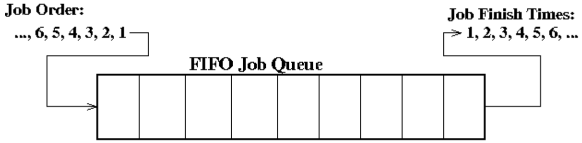
In essence, the operator was a major part of the operating system.

Batching Processing
Jobs

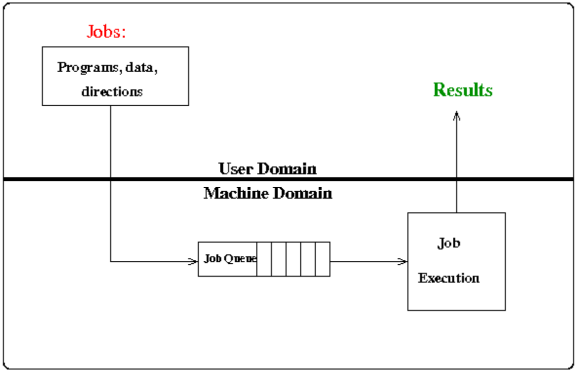
The materials and special instructions given to the operator form a **job**.

If we want to execute several jobs, we place them in the **job queue**, where they will wait until they can be executed.

A **queue** is a storage structure in which objects (jobs) are ordered **first-in, first-out (FIFO)**.



Batching Processing



A more realistic queue model is the **priority queue** where different jobs receive different priorities.

Batching Processing
Job Control Language

In batch processing, jobs were accompanied by a set of instructions explaining the steps required to prepare the machine for that particular job.

- Instructions are coded in a **job control language (JCL)**, and are stored with the job in the job queue.

When a job is selected for execution, the OS prints the instructions at a printer where they can be read and followed by the computer operator.

Batching Processing
Drawbacks

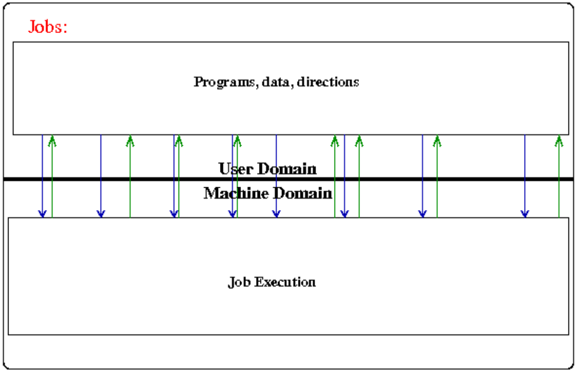
Traditional batch processing imposes the restriction that the user has no interaction with the program once it is on the job queue.

- Applications must have all data and processing decisions are established in advance.
- No computer games, word processors, ...

We need operating systems that are **interactive** with the user.

Interactive Processing
Overview

The need for interaction between the user and machine lead to the concept of **interactive processing**.



Interactive Processing
Overview

Interaction with the user raised new issues:

- **Response time**: Type a word, and after 5 minutes, it appears on the screen??
- **Real-time processing** is providing services to a user in a timely manner.

cse.unl.edu: How can an interactive system that be used by more than one user?

Time-Sharing

To manage a group of users at the same time, we need to divide processing time between users.

- Finish the task of each user, one after another (similar to the idea of a bank teller)
- **Time-sharing**: Divide the CPU time into intervals (**time slices**), and portion out time slices to each user/job in the system.

Time-Sharing
Multitasking

Multitasking is a different concept of time-sharing.

- Time-sharing shares time slices between different users.
- **Multitasking** is when a time-sharing system is begin used in a single-user environment.

Personal computers have advanced to the point of begin multitasking machines.

Time-Sharing and Multitasking
Single-Processor System Issues

Time-sharing and multitasking machines greatly improve the efficiency of the machine, but add more **overhead** to the computation.

- The computer has to divide the job into parts, and keep track of which part of which job is begin executed, whose job it is, where the job is stored, etc.
- Additionally, one job may be more important than others, so we may need to assign more time to it.

Operating System Storage

The operating system is “system” software, i.e. programs that operate or maintain a computer.

Recall that software must be loaded into the main memory (RAM) in order to be executed by the CPU.

- RAM cannot store programs indefinitely. When the computer is shut off, RAM loses all information it previously held.
- The OS resides **on the hard disk** of your machine.

When the machine is turned on, the OS is loaded into RAM, allowing the CPU to start executing the OS.

Operating System Storage
Machine Turn-Ons

The process of starting the operating system is known as **boot strapping**, or for short, **booting**.

As system software, the OS loads programs into the main memory, initializes the program counter in the CPU to run the program, etc.

What loads the OS??

- At startup, the CPU contains an address to main memory in a reserved portion that is non-volatile and unalterable, called **Read Only Memory (ROM)**.
- ROM contains the program called **the bootstrap**, the first program that the CPU executes when it is turned on.

Operating System Storage
Bootstrap

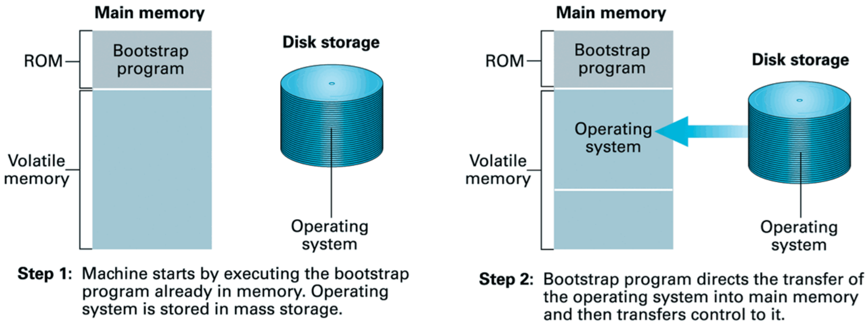
Bootstrap:

- The program that is executed automatically when the machine is turned on.
- Directs the CPU to transfer the OS from a predetermined spot in the hard drive into main memory.

Once the OS is in memory, the bootstrap directs the CPU to execute a JUMP instruction to that area in memory.

- At this point, the OS takes over and begins controlling the machine's activities.

Operating System Storage
Bootstrap


Main memory
ROM
Bootstrap program
Volatile memory
Disk storage
Operating system
Step 1: Machine starts by executing the bootstrap program already in memory. Operating system is stored in mass storage.
Main memory
ROM
Bootstrap program
Operating system
Volatile memory
Disk storage
Operating system
Step 2: Bootstrap program directs the transfer of the operating system into main memory and then transfers control to it.

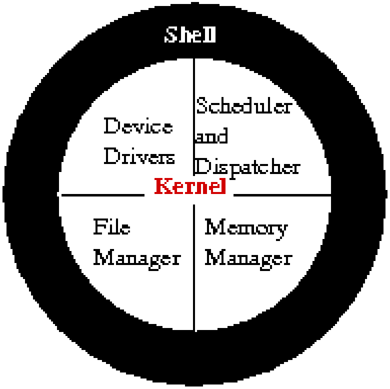
Why don't we store the OS software in ROM?

Operating System Components

An OS has two main components:

- **Kernel**: The internal part of the OS. The kernel contains those software components that perform the very basic functions required by the computer.
- **Shell**: The interface between a user and the kernel. The job of the shell is to communicate with the user or users of the machine.

Kernel Components



File Manager

The job of the **file manager** is to keep records of all the files stored in the mass storage devices.

A record for a file can contain:

- The location of the file,
- Which users are allowed to access the various files,
- What portions of mass storage are available to new files or extensions to existing files.

The records are stored in an area of the main memory called a **file descriptor**.

File Manager

Other file manager operations:

- **Allow for grouping of files**: Groups files into directories or folders, keeping track of which files belong to which folder.
- **Allow access to files**: All requests for accessing files by other software programs goes through the file manager.
  - “Opening” a file: The process of requesting the permission of the file manager to access a file.

How hard is this task? What happens when you store a small file, but you then want to extend it by writing more information...

Device Drivers

The most rapidly changing components in computing are the peripheral devices.

- We should be able to separate the technical aspects of the peripheral devices from the machine, i.e. make the device conform to the machine, not make the machine conform to the device.
- If not, we need to change the machine every time a new CD drive appears.

Device drivers are the software units that communicate with peripheral devices (e.g. printer, modem, keyboard, monitor, webcam, ...).
For example, when printing a file:

- The file manager asks the device driver to print the file.
- The technical details of this instruction (which is printer dependent) is taken care of by the device driver of that machine.
- The file manager does not need to know the type, or brand, of the printer!

Scheduler and Dispatcher
Overview

In time-sharing or multitasking systems, the **scheduler** determines which activities are to be considered for execution.

The **dispatcher** controls the allocation of time slices to these activities.

More on this later in the chapter.

Memory Manager

The **memory manager** is responsible for coordinating the machine's use of main memory. Difficult? Depends...

Trivial: Environment → Machine performs only one task at a time.

- Load, wait for execution, replace with next program, wait for execution, ...

Difficult: In a multi-tasking/multi-user environment, several programs need to run at the same time.

- You are writing your homework while listening to music, all the while chatting with your parents asking for money and surfing Facebook after an enthralling game of Minesweeper.
- Has to keep track of different programs while loading and storing them back and forth between the main memory and mass storage.
- What happens if we run out of memory? Can the memory fit all the apps we want to run? Are some apps larger than the size of the memory?

Memory Manager

Virtual Memory Concept

When memory is limited (say 128 MB):

- What if we want to open a program that is 256 MB?
- What if we have two programs, the first is 101 MB, the second is 28 MB. What happens?

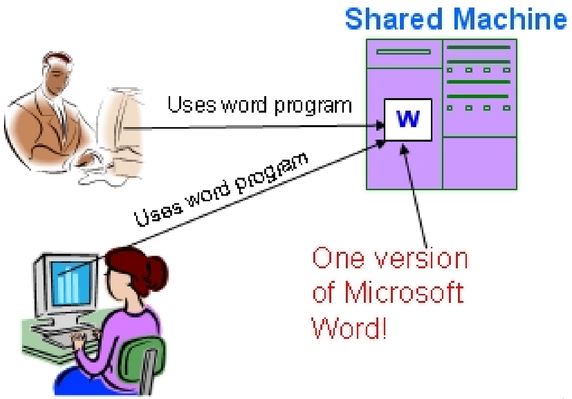
Solution: The memory manager creates the illusion of larger memory.

- The memory manager reserves a larger space on the magnetic disk (say 256 MB).
- All bit patterns needed are recorded on the disk as if it was main memory.
- The data is split up into uniform length pieces, called **pages**.
- Only the pages that are needed are loaded into memory. When different pages are needed, old pages are written to the disk, new pages are brought in.

This method is called **Virtual Memory**.

Programs vs. Processes

Consider a multiuser system, with two users using the same program, at the same time



Even though both users are using the same *program*, each user has their own **process**.

Program vs. Process

A **program** is a static set of directions, where a **process** is a dynamic activity whose properties change as time progresses. In other words:

A process is a program in execution.

On the previous slide:

- Microsoft Word is a *program*.
- The suit and the dress opened their own *process* by starting Word and updating their own file (Two processes were created, one for each).

Any software entity that runs on your machine will be associated with a process (whether it is an application or system software).

Process Coordination

The Operating System's task is to manage and coordinate the operations of various operations.

- When you are using your machine, several processes are created.
- In a multitasking or time-sharing machine, several processes compete for the machine's resources (e.g. CPU time, memory, disk access, etc.).

Coordination involves ensuring that each process has the resources (space in memory, CPU time, etc.) that it needs. At the most basic level, the OS stops independent processes from interfering with one another.

- OS needs to know information about each process.

Both the processes and the OS work together to maintain the process information.

Process Coordination

Requirements of the Process

Each process keeps information about its current status in a table called the **process state** $\longrightarrow$ The "snapshot" of the machine at a given time.

- Current position in the program being executed (value of the PC).
- Values in the CPU registers and associated memory cells.

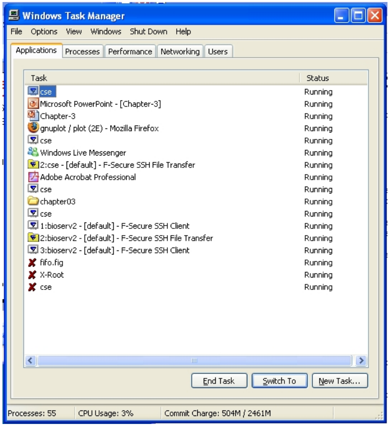
The process state can be used to restore the same environment that the process was in before its "time-slice" ended.

Process Coordination

Requirements of the OS

The OS keeps track of which processes are active, and what resources they are using in the **process table**.

- The process table is stored as a block of information residing in the main memory.


Windows Task Manager
File Options View Windows Shut Down Help
Applications Processes Performance Networking Users
End Task
Switch To
New Task...

Task	Status
cse	Running
Microsoft PowerPoint - [Chapter-3]	Running
Chapter-3	Running
gnuplot / plot (2E) - Mozilla Firefox	Running
cse	Running
Windows Live Messenger	Running
2:cse - [default] - F-Secure SSH File Transfer	Running
Adobe Acrobat Professional	Running
cse	Running
chapter03	Running
cse	Running
1:bioserv2 - [default] - F-Secure SSH Client	Running
2:bioserv2 - [default] - F-Secure SSH File Transfer	Running
3:bioserv2 - [default] - F-Secure SSH Client	Running
fifo.fig	Running
X-Root	Running
cse	Running

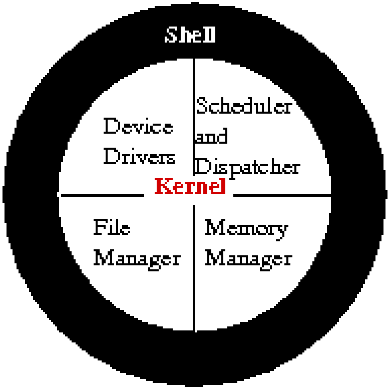
Processes: 55 CPU Usage: 3% Commit Charge: 504M / 2461M

Process Administration

Given that we have the process table along with the process states:

- How does the OS use the information?
- Which part of the OS uses this information?

Process Administration
Scheduler and Dispatcher



The **scheduler** maintains a record of the processes present in the system, i.e. it maintains the process table:

- A new process is introduced: Scheduler creates a process for the task by placing a new entry in the process table.
- A process completes: Scheduler removes process from process table.

Process Administration
Scheduler (Cont'd...)

- Maintains information about whether processes are waiting or ready.[1].
  - A process is **ready** if it is in a state from which its progress can continue.
  - A process is **waiting** if its progress is currently delayed until some external event occurs (such as the completion of a disk access).

What monitors the execution of the process?

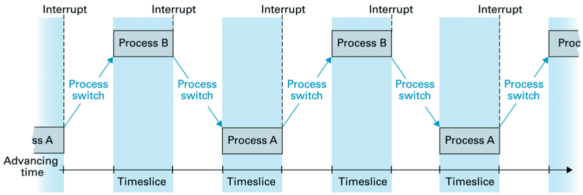
[1]NOTE: Each task in the OS is handled by a specific component! The scheduler does *not* know how much memory a process uses. Only the memory manager has this information.

Process Administration

Dispatcher

The **dispatcher** is the component of the kernel that ensures that the scheduled processes are executed.

- In time-sharing systems, this is accomplished by dividing the CPUs attention into **time-slices**, typically no more that 50 ms.
- Switching the CPUs attention from one task to another (after the end of a time slice) is called **process switching** or **context switching**.



Introduction

An important task of the operating systems is the allocation machine's resources is to the processes in the system.

- A Machine resource means any device or feature within the machine.
- For example, access to files is a resource: If two users want to open the same file, who should do that first? Another example can be memory space.

Recall that in time-sharing and multitasking systems, the processes compete for CPU time.

- One solution to this problem: Using time-slices.

Process Competition

Sharing a Printer

Consider a time-sharing operating system controlling the activities of a machine with a single printer:

When a process needs to print, it will ask the OS for access to the printer.

- If there are no other processes using the printer at the time of the request, the OS will grant access to the requesting process.
- If a process is currently using the printer, the operating system will queue the process in the waiting list for the printer.

How can the OS accomplish this task? I.e. how does the OS know the printer is in use?

Process Competition

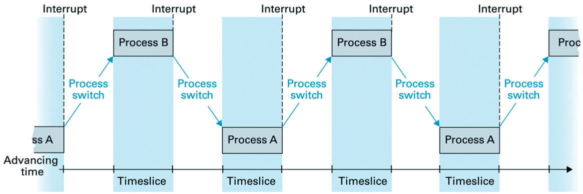
One way the OS can use to mark whether the printer is in use is through a **flag bit**.

- A single bit, where if the bit is 1, there is a process using the printer, 0 otherwise.

When a process requests the printer, the OS looks up the flag bit and sees if there is a process currently using the printer.

Often, the problem is far more complex!

Process Competition
Context Switching



Each process gets a time slice... What will happen if the interrupt occurs just before the OS sets the ‘use’ flag of the printer?

Process Competition

Context Switching Problem

Each process gets a time slice... What will happen if the interrupt occurs just before the OS sets the 'use' flag of the printer?

Two processes will be granted access to the printer *at the same time*.

Solution?

1. If the OS is able to run the operation that checks the printer flag,
2. Disallow any Interrupts, until
3. The OS either sets the printer 'use' flag and begins printing OR finds the printer 'use' flag is set and puts new print job on the printer queue.

Operations that can be executed without in interrupt are called **Atomic Operations**.

Sharing Main Memory

Using the time-sharing model, consider two processes containg their own set of instructions, however, they share a cell in main memory.

Process 1:

1. Execute instructions...
2. Check M[05]:
    2.1 If *M[05]* $= 0$: Set M[05] $= 1$.
3. Execute instructions...

Process 2:

1. Execute instructions...
2. Check M[05]:
    2.1 If *M[05]* $\neq 0$: Increment M[05].
3. Execute instructions...

Main Memory:

00	
01	
02	
03	
04	
05	0
06	
07	

Sharing Main Memory
The Critical Section

The **Critical Section** is the part of the program in which a process accesses a shared memory location.

Process 1:

1. Execute instructions...
2. Check M[05]:
    2.1 If *M[05]* $= 0$: Set M[05] $= 1$.
3. Execute instructions...

Process 2:

1. Execute instructions...
2. Check M[05]:
    2.1 If *M[05]* $\neq 0$: Increment M[05].
3. Execute instructions...

Sharing Main Memory
The Critical Section

Process 1:

1. Execute instructions...
2. Check M[05]:
   2.1 If *M[05]* $= 0$: Set M[05] $=$ 1.
3. Execute instructions...

Process 2:

1. Execute instructions...
2. Check M[05]:
   2.1 If *M[05]* $\neq 0$: Increment M[05].
3. Execute instructions...

Scenario: Process 2 starts first:

1. Process 2 checks M[05], finds 0.
2. Process 2's time expires, CPU switches to process 1.
3. Process 1 checks M[05], finds 0, sets it to 1. Time expires, CPU switches to process 2.

Process 2 increments M[05]?

Critical Section
Access

A **Semaphore** can be thought of as a gate that controls access to the critical section of a program.

- Created by placing special instructions before and after the critical section.
- Verifies that when the CPU starts to execute instructions inside the critical section of a process, no other process can interrupt until the execution of the critical section is completed.

Process 1:

1. Execute instructions...
2. Semaphore statement to enter critical section
3. Check M[05]:
    3.1 If $M[05] = 0$: Set M[05] = 1.
4. Semaphore statement to exit critical section
5. Execute instructions...

Process 2:

1. Execute instructions...
2. Semaphore statement to enter critical section
3. Check M[05]:
    3.1 If $M[05] \neq 0$: Increment M[05].
4. Semaphore statement to exit critical section
5. Execute instructions...

Deadlocks
Defined

A **Deadlock** is a situation that happens when two or more processes are blocked from processing because each is waiting for access to resources that are allocated to another process.

- One process may have access to the machine's printer, but is waiting for the CD-ROM drive, while another process has access to the CR-ROM drive, but is waiting for the printer.

When this happens, the performance of the machine is severely degraded.

Criteria for a Deadlock

A deadlock can occur only if all of the following conditions are satisfied at the same time[2]:

1. There is competition for non-shareable resources (e.g. printer, hard drive, modem).
2. The resources are requested on a partial basis; that is, having received some resources, a process will return later to request more.
3. Once a resource has been allocated, it cannot be forcibly retrieved.

[2]Note: If any of the three conditions is not satisfied, the deadlock can be broken.

There are three main approaches to solve the problem of deadlocks:

1. **Deadlock avoidance techniques**: Try to avoid a deadlock in the first place.
2. **Deadlock detection and correction techniques**: Let a deadlock happen, detect it, then solve it.
3. **Deadlock ignorance and programmer stupidity**: Ignore the problem altogether and pretend that a deadlock will never occur.

Which approach do you think is most common in today's OSes? In a CS class?

Handling Deadlocks
Solution Type 1, Deadlock Avoidance

Keep in mind: Breaking any single deadlock condition will solve the deadlock problem!!

Deadlock condition: There is competition for non-sharable resouces.

- We can break this by making a non-sharable resource a sharable one.
- **Spooling** is a printer technique that can be used for holding data in order to output at a later (and more convenient) time. This makes the printer appear sharable to the user.

Deadlock condition: The resources are requested on a partial basis.

- Each process asks for all the resources it needs *at one time*.

Handling Deadlocks

Solution Type 2, Deadlock Detection and Correction

Keep in mind: Breaking any single deadlock condition will solve the deadlock problem!!

Deadlock condition: *Unknowable*. To detect a deadlock, we need to *forcibly* retrieve resources to break the deadlock.

- Kill one process, allowing others to continue execution.

Software Categories

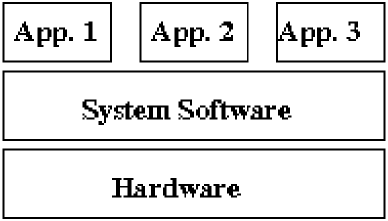
System software:

- Operating System
- Utilities (disk defragmenter, etc.)

Application software:

- Word processors, web browsers, games, etc.

System software provides the environment in which the application software executes.



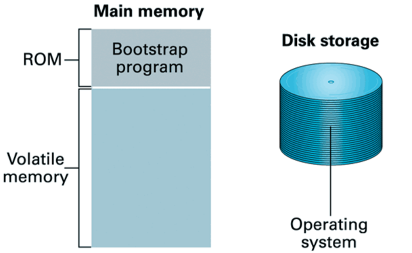
Important Concepts – NOT COMPREHENSIVE

Process: A program in execution.

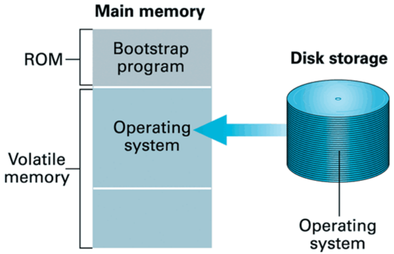
Processing Models:

- Batch processing
- Interactive (real-time) processing
- Time-sharing processing
- Multitasking

Important Concepts – NOT COMPREHENSIVE

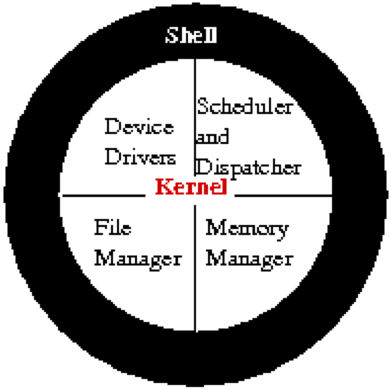


Step 1: Machine starts by executing the bootstrap program already in memory. Operating system is stored in mass storage.



Step 2: Bootstrap program directs the transfer of the operating system into main memory and then transfers control to it.

Important Concepts – NOT COMPREHENSIVE



Important Concepts – NOT COMPREHENSIVE

Process management:

- Resources
- Time slice
  - Illusion of concurrent processing vs. efficiency
- Critical section problem
  - Atomic operation
  - Semaphores

Deadlock:

- Deadlock conditions
  - Competition for non-shareable resources
  - Resources are requested on a partial basis
  - Once requested, a resource cannot be forcibly retrieved
- Handling deadlock
  - Remove any of the three conditions.

Important Concepts – NOT COMPREHENSIVE

Questions over Chapter 3?