Operating Systems

Szu-Chi Chung

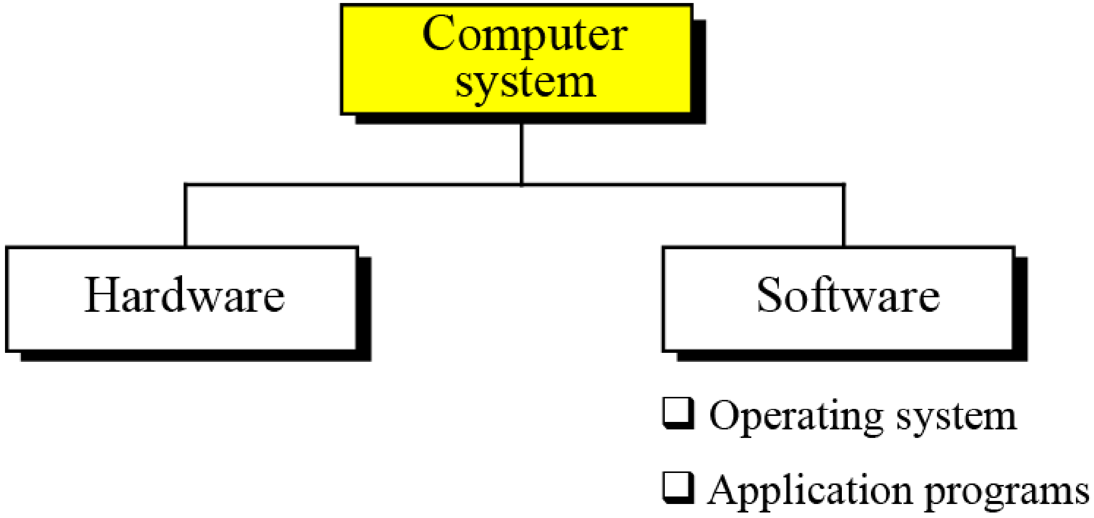
Department of Applied Mathematics, National Sun Yat-sen University

What we are going to study in this semester

- Introduction
- Data representation and operation
  - Number Systems
  - Data Storage
  - Operations on Data
- Computer hardware
  - Computer organization
  - Computer network and internet
- Computer software
  - Programming Languages (Python)
  - Operating system
  - Algorithms
- Data organization and abstraction
  - File Structure
  - Data Structure
  - Abstract Data Type
- Advance topics
  - Security
  - Artificial intelligence (Recorded lecture)
  - Data compression (Recorded lecture)
- Not covered
  - Software engineering
  - Databases
  - Theory of computation
  - Social media and social issues

Introduction

- Computer software is divided into two broad categories: the operating system and application programs
  - Application programs use computer hardware to solve users' problems
  - The operating system, on the other hand, controls access to hardware by users

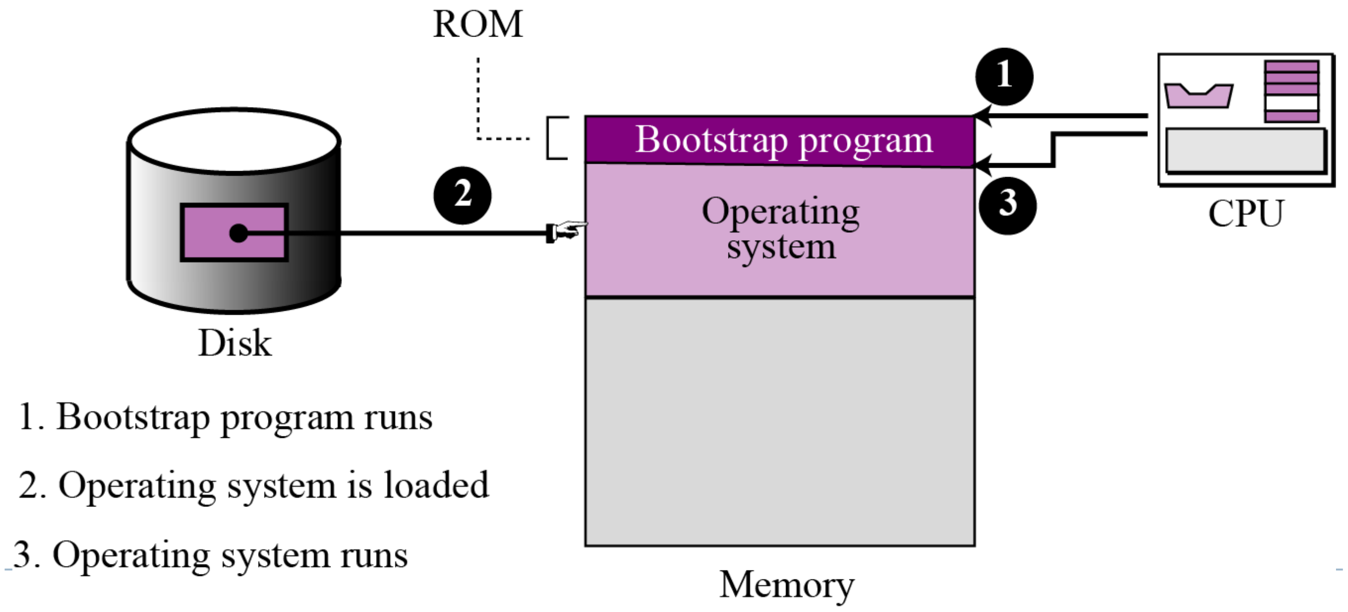


Operating system

1. An operating system is an interface between the hardware of a computer and the user (programs or humans)
2. An operating system is a set of programs that facilitates the execution of other programs
3. An operating system acts as a general manager supervising the activity of each component in the computer system
   - The operating system checks that hardware and software resources are used efficiently, and when there is a conflict in using a resource, the operating system mediates to solve it
   - For example, it is responsible for loading other programs into memory for execution

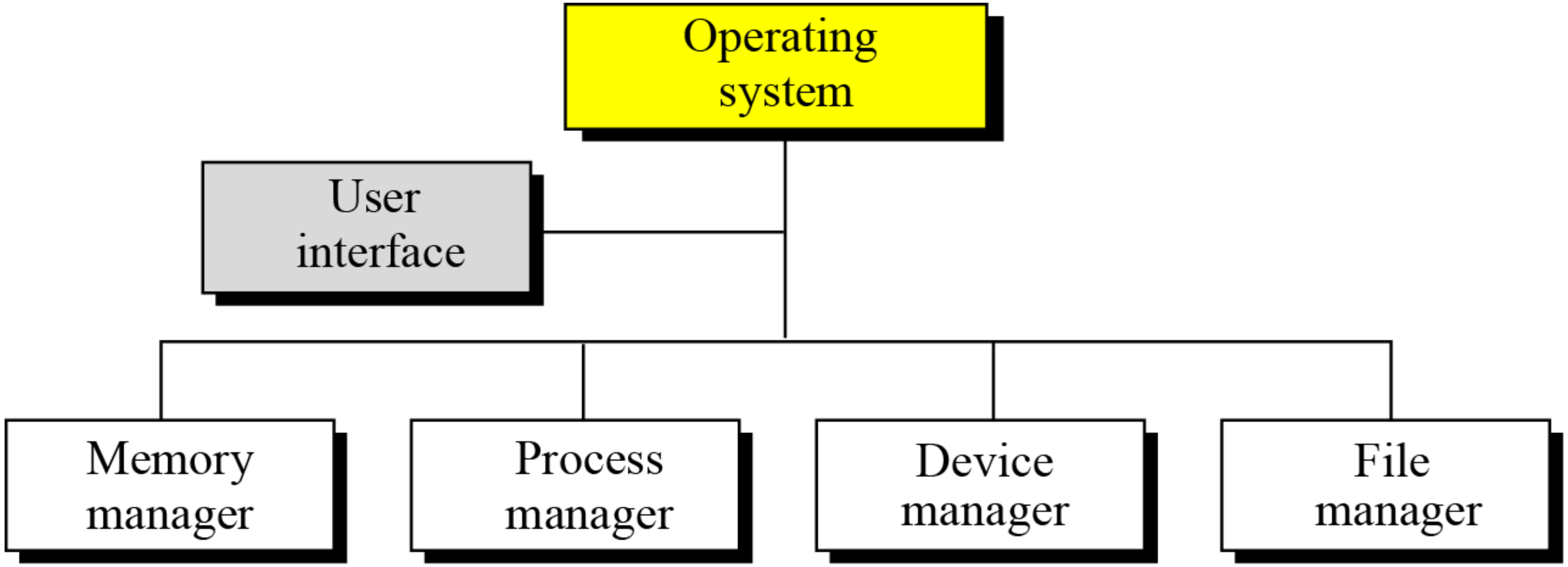
Bootstrap process

- However, the operating system itself is a program that needs to be loaded into the memory and run!
  - A very small section of memory is made of ROM and holds a small program called the *bootstrap program* which is pointed by the program counter when the computer is on
  - This program is only responsible for loading the operating system into RAM and the program counter in the CPU is set to the first instruction of the operating system



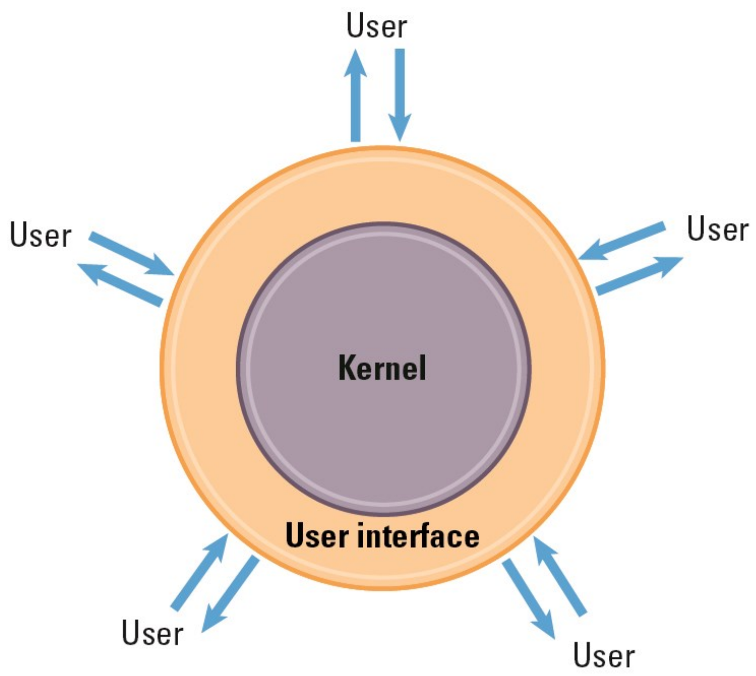
Components

- An operating system needs to manage different resources in a computer system
  - A modern operating system has at least four duties *memory manager, process manager*, *device manager*, and *file manager* which are the *kernel*
  - Operating system also has a component called a user interface or a *shell* which is responsible for communication outside the operating system



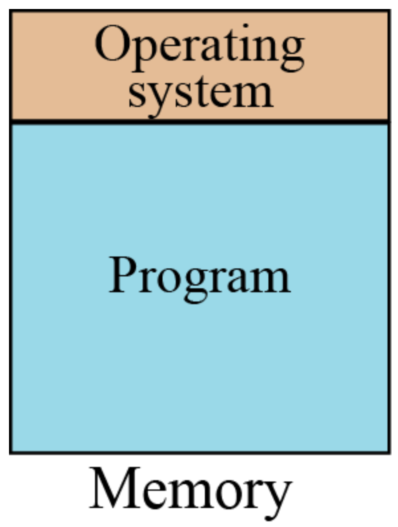
User interface

- User interface a program that accepts requests from users (processes) and interprets them for the rest of the operating system
  - Shells communicated with users through textual messages using a keyboard and monitor screen
  - A GUI (graphical user interface) allows users to issue commands by using a mouse or other input device to manipulate the icons
  - Touch screens allow users to manipulate icons directly with their fingers



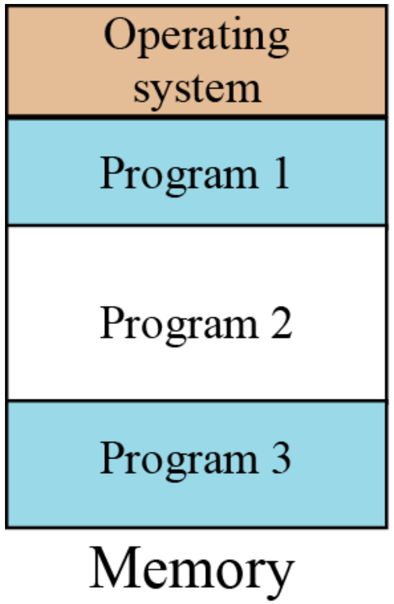
Memory manager

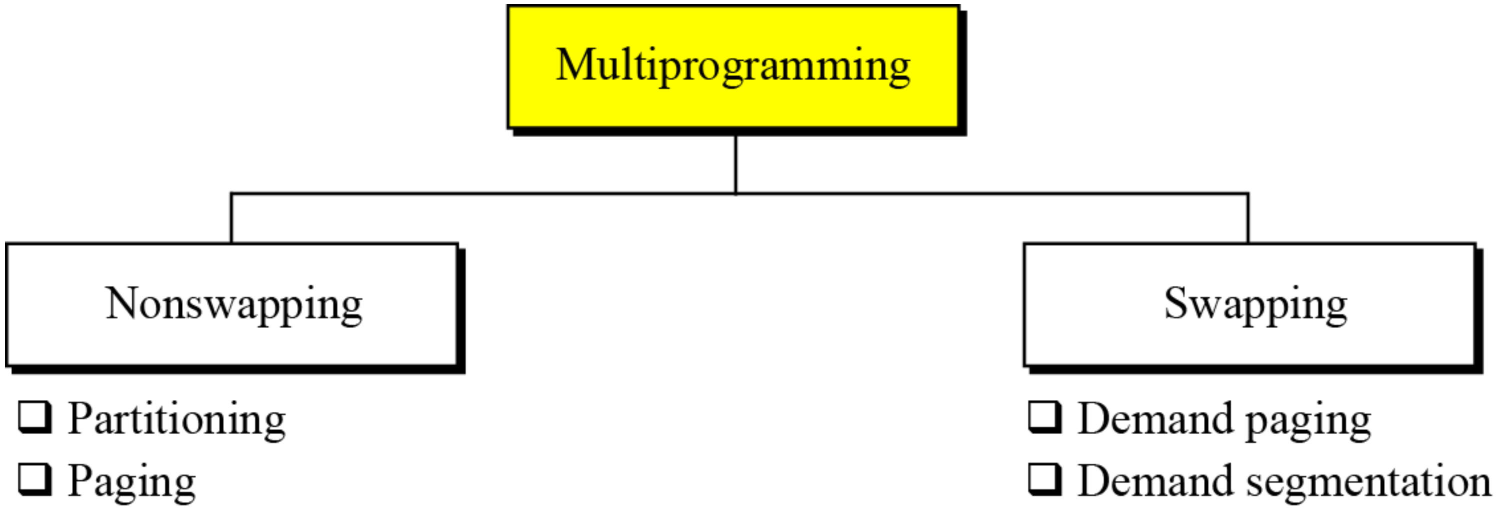
- Memory allocation must be managed to prevent applications from running out of memory
  - It can be divided into two broad categories of memory management: *monoprogramming* and *multiprogramming*
  - In monoprogramming, most of the memory capacity is dedicated to a single program
    - In this configuration, the whole program is in memory for execution
    - The job of the memory manager is to load the program into memory, run it, and replace it with the next program
    - This is a very inefficient use of memory and CPU time since we need to wait for the I/O device



Memory manager

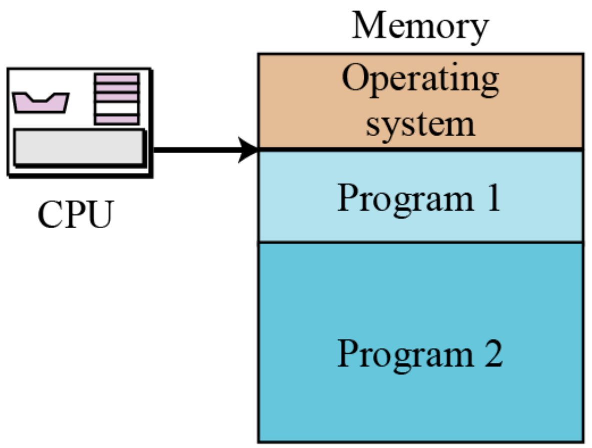
- In multiprogramming, more than one program is in memory at the same time
  - They are executed concurrently, with the CPU switching rapidly between the programs
  - Multiprogramming can be further divided into several categories
    - Nonswapping means that the program remains in memory for the duration of execution



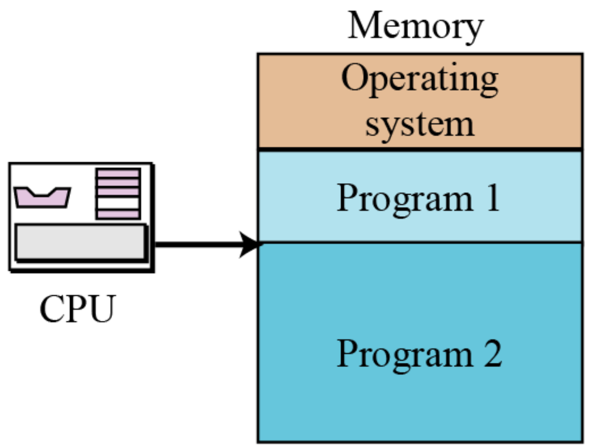


Memory manager - Partitioning

- In *partitioning*, memory is divided into variable-length partitions
  - Each section or partition holds one program. The CPU switches between programs
    - It starts with one program, executing some instructions until it either encounters an input/output operation or the time allocated for that program has expired
    - The CPU then saves the address of the memory location where the last instruction was executed and moves to the next program



a. CPU starts executing program 1



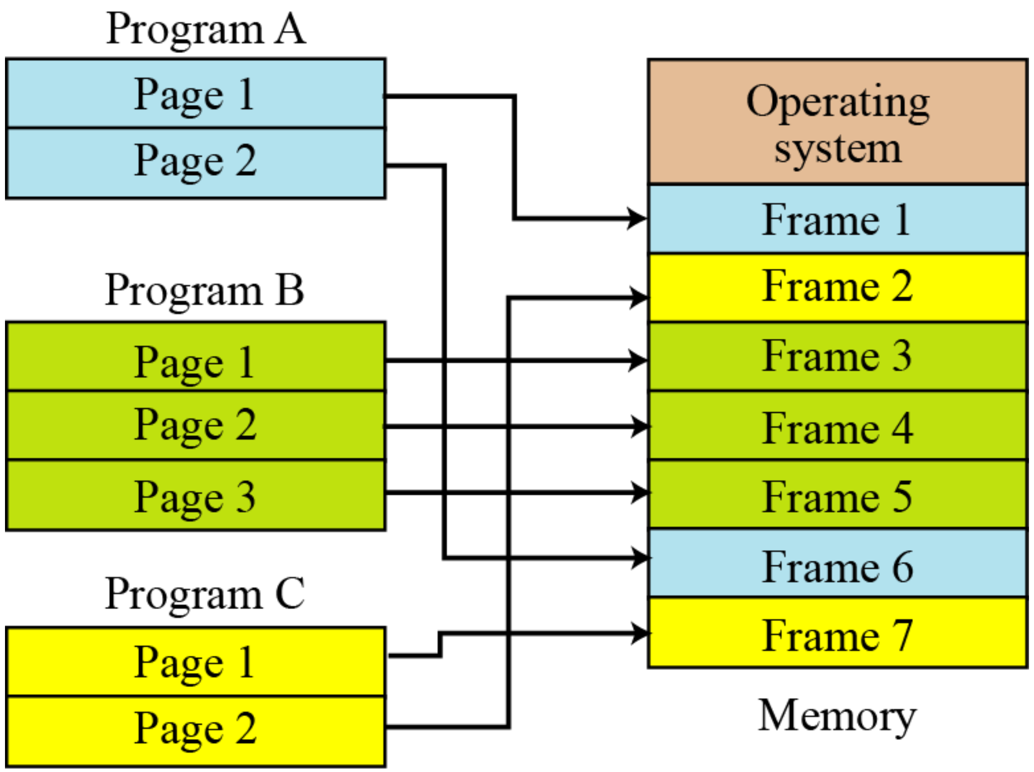
b. CPU starts executing program 2

Memory manager - Partitioning

- Each program is entirely in memory and occupies contiguous locations. However, there are some issues
  - The size of the partitions has to be determined beforehand by the memory manager
  - Even if partitioning is perfect when the computer is started, there may be some holes after completed programs are replaced by new ones
  - When there are many holes, the memory manager can compact the partitions to remove the holes and create new partitions, but this creates extra overhead on the system!

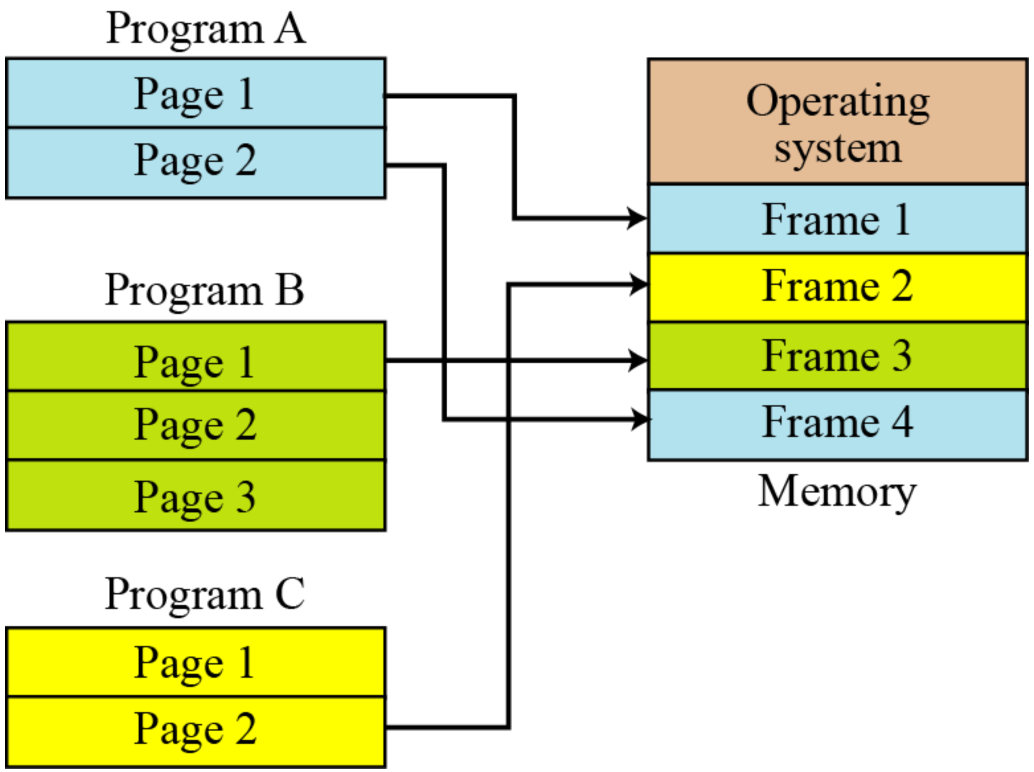
Memory manager - Paging

- In *paging*, memory is divided into equally sized sections called frames. Programs are also divided, into equally sized sections called pages
  - The size of a page and a frame is usually the same and equal to the size of the block used by the system to retrieve information from a storage device
  - The program does not have to be contiguous in memory
  - Paging improves efficiency to some extent, but larger program still needs to be in memory before being executed



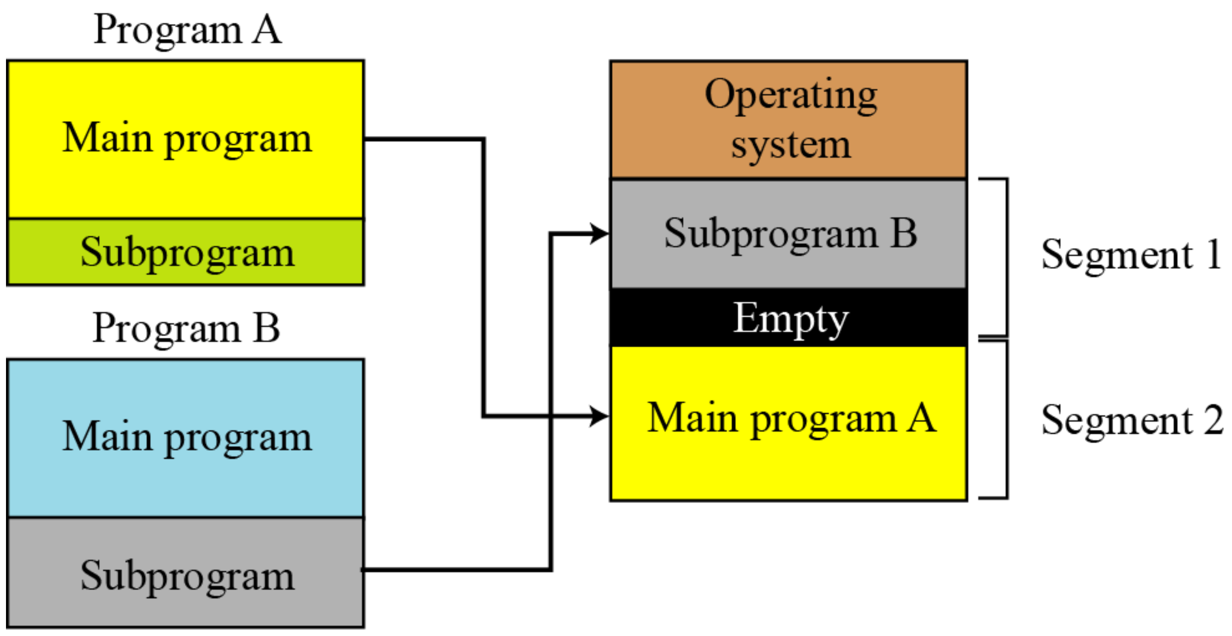
Memory manager - Demand paging

- Paging still requires that the entire program be in memory for execution
  - In *demand paging* the program is divided into pages, but the pages can be loaded into memory one by one, executed, and replaced by another page
  - The program can be swapped between memory and disk!



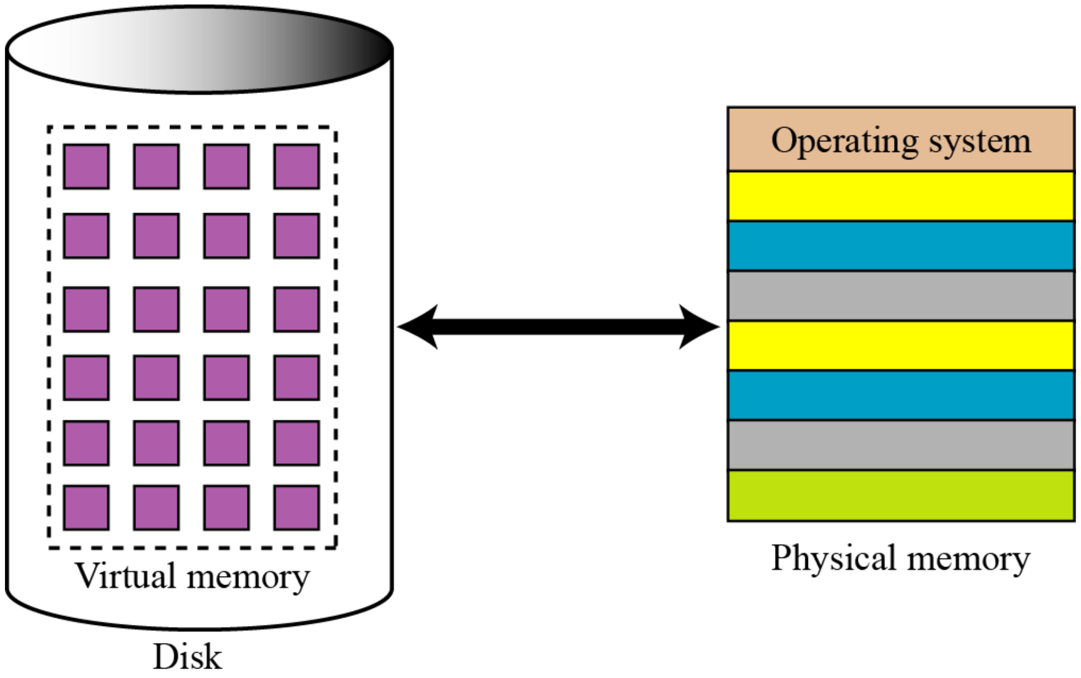
Memory manager - Demand segmentation

- In *demand segmentation*, the program is divided into segments that match the programmer's view
  - In demand segmentation, the program is divided into modules that match the programmer's view
  - Since segments in memory are still of equal size, part of a segment may remain empty



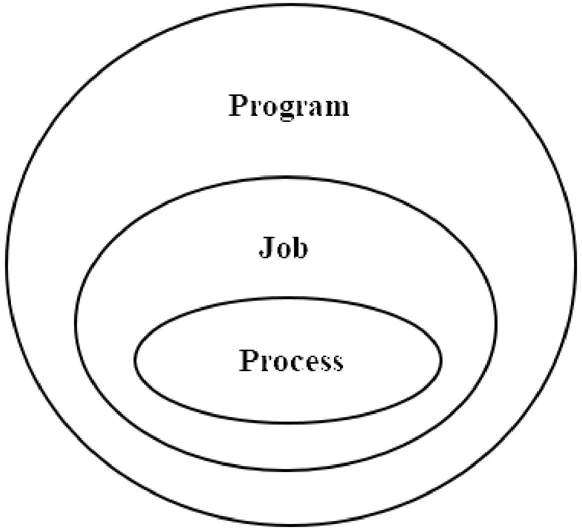
Memory manager - Virtual memory

- Demand paging and demand segmentation mean that, when a program is being executed, part of the program is in memory and part is on disk!
  - This large "fictional" memory space created by paging is called *virtual memory*



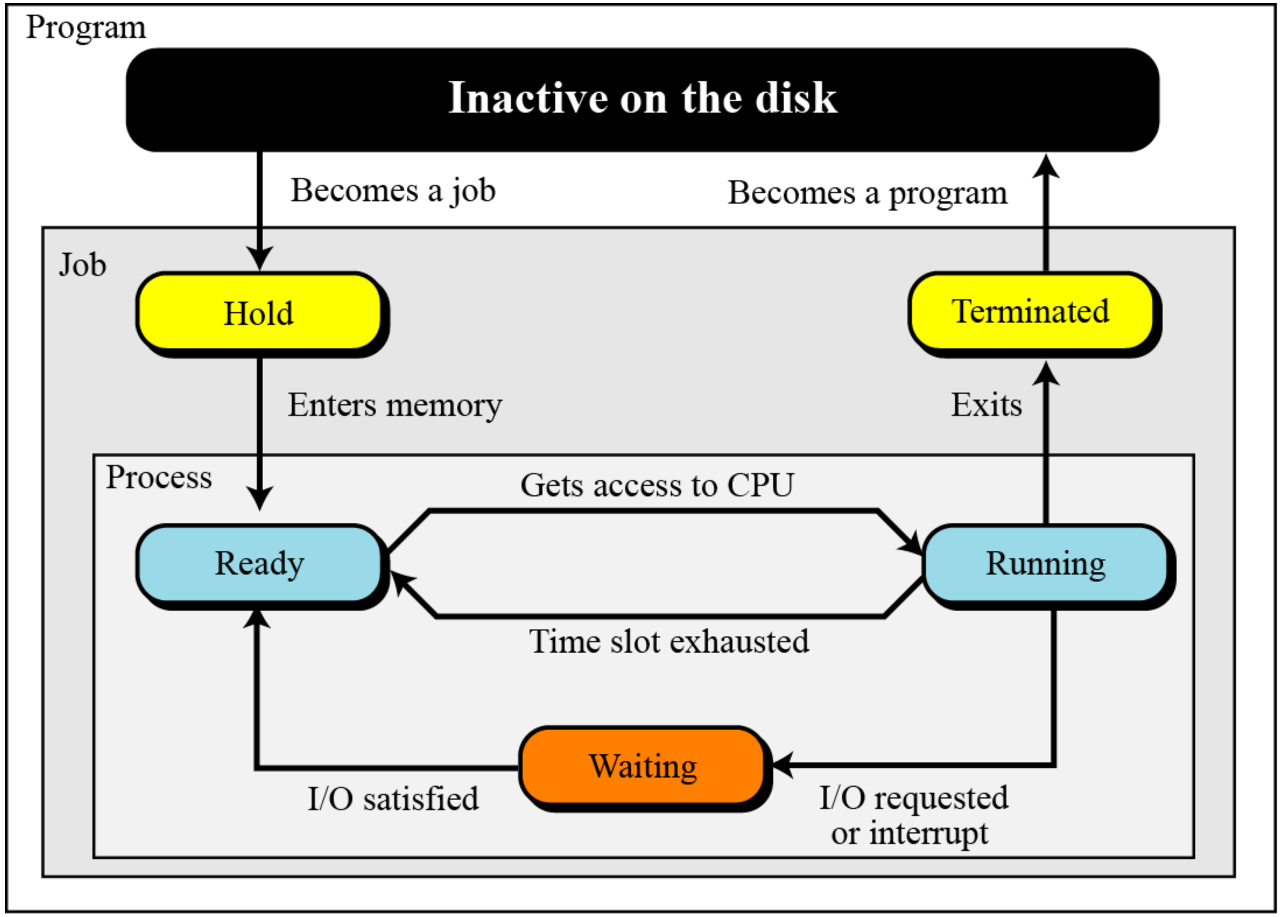
Process manager

- Modern operating systems use three terms that refer to a set of instructions
  - Program: A program is a nonactive set of instructions stored on a disk. It may or may not become a job
  - Job: A program becomes a job from the moment it is selected for execution until it has finished running and becomes a program again
    - During this time a job may or may not be executed and it may be on disk or in memory
  - Process: A process is a job that is being run in memory
    - As long as the job is in memory, it is a process



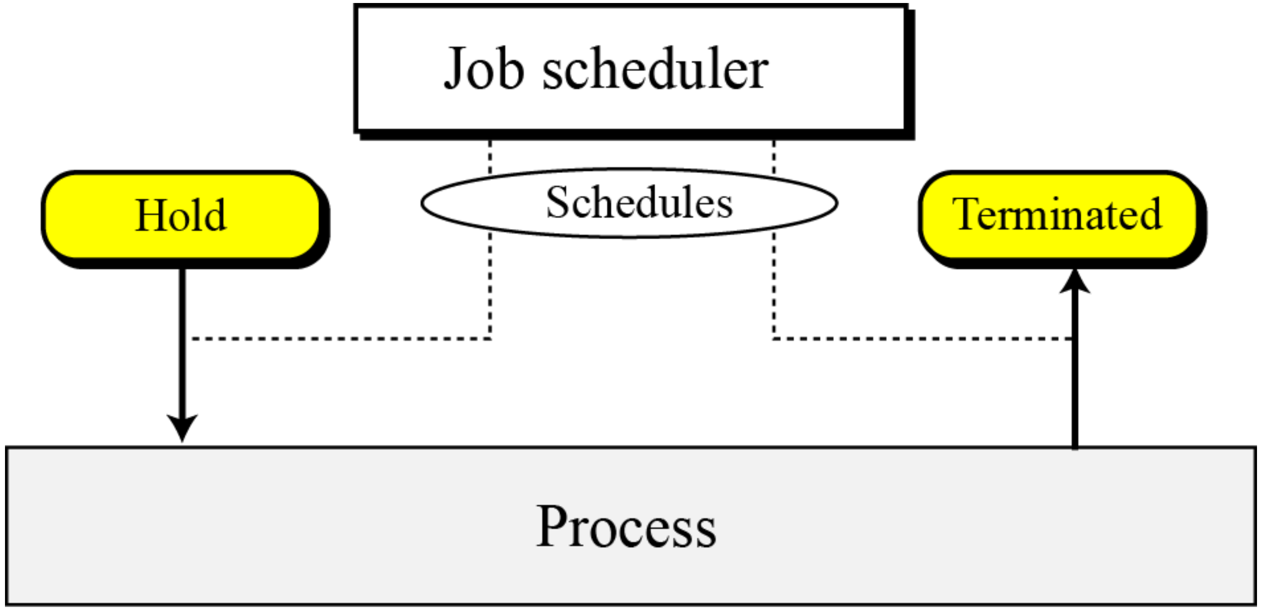
Process manager - State diagrams

- The relationship can be illustrated with a *state diagram*
   1. A program becomes a job when selected by the operating system and brought to the *hold state*
   2. When there is memory space available to load the program totally or partially, the job moves to the *ready state* and becomes a process
   3. It remains in memory and in this state until the CPU can execute it, moving to the *running state*
   4. When it encounters I/O it enters *wait state* and it goes into the *terminated state* when finishing



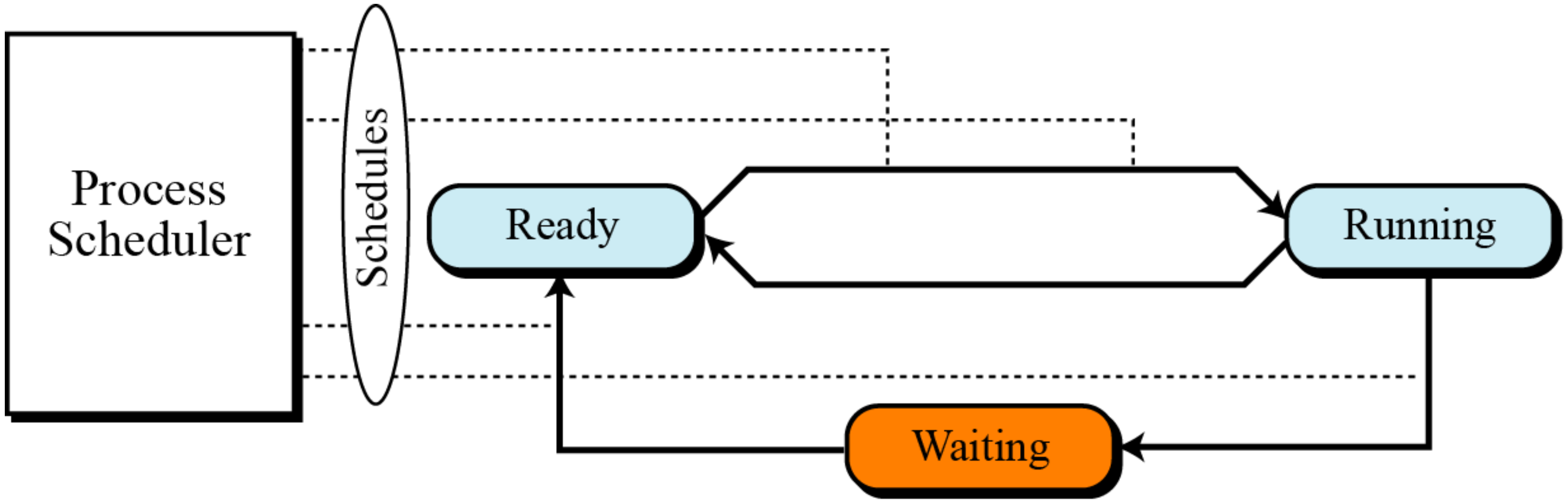
Process manager - Schedulers

- To move a job or process from one state to another, the process manager uses two schedulers: the *job scheduler* and the *process scheduler*
  - A job scheduler is responsible for creating a process from a job and terminating a process



Process manager - Schedulers

- The process scheduler moves a process from one state to another
  - It moves a process from the running state to the waiting state when the process is waiting for some event (interrupt) to happen
  - It moves a process from the running state to the ready state if the process' time allotment has expired

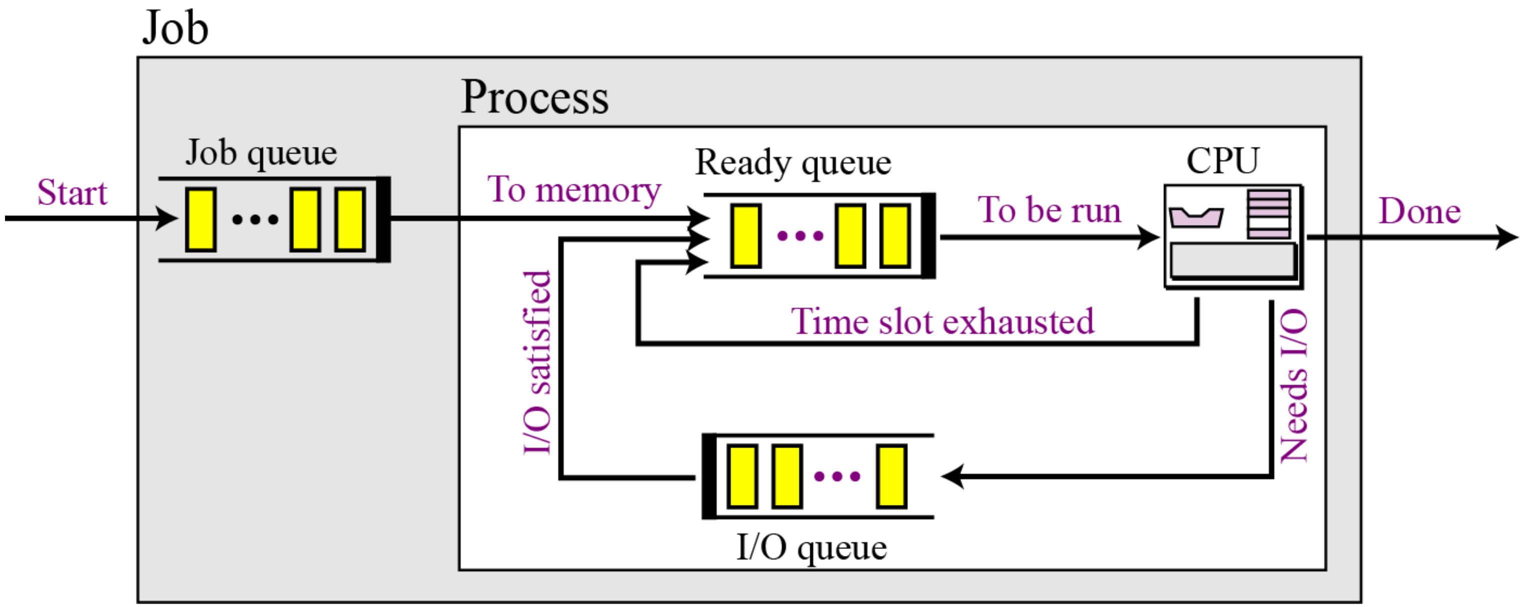


Process manager - Queuing

- In reality, there are many jobs and many processes competing with each other for computer resources
  - To handle multiple processes and jobs, the process manager uses *queues*
  - There is a *job control block* or *process control block* associated with each job or process that stores the information about the job or process
    - The process manager stores the job or process control block in the queues instead of the job or process itself!

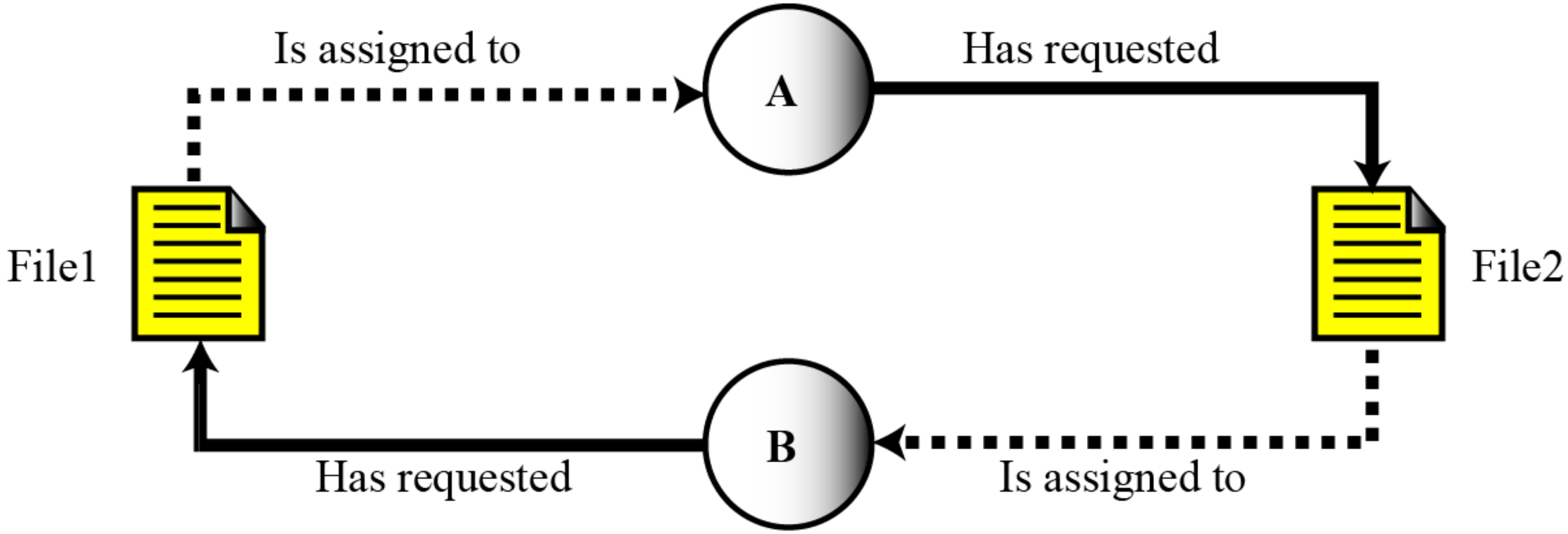
Process manager - Queuing

- An OS can have several queue and the process manager can have different policies for selecting the next job or process from a queue



Process manager - Process synchronization

- The whole idea behind process management is to synchronize different processes with different resources
  - We can have two problematic situations: *deadlock* and *starvation*
  - Deadlock occurs if the operating system allows a process to start running without first checking to see if the required resources are ready, and allows a process to hold a resource as long as it wants

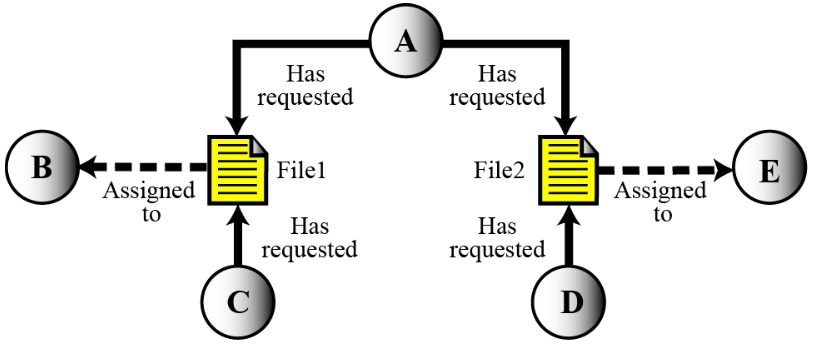


Process manager - Process synchronization

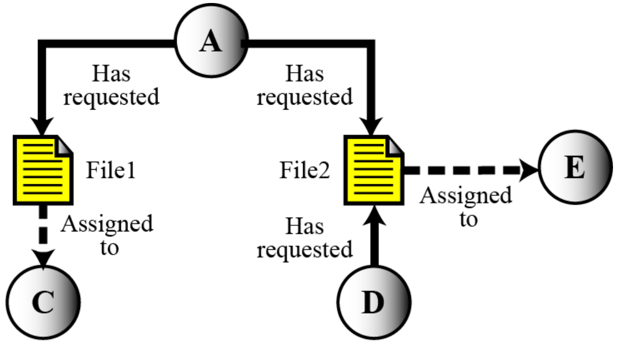
- Solutions
  - Not to allow a process to start running until the required resources are free
  - Limit the time a process can hold a resource
- There are four necessary conditions for deadlock
  - Mutual exclusion: Only one process can hold a resource
  - Resource holding: A process holds a resource even though it cannot use it until other resources are available
  - No preemption: The operating system cannot temporarily reallocate a resource (Priority)
  - Circular waiting: All processes and resources involved form a loop

Process manager - Process synchronization

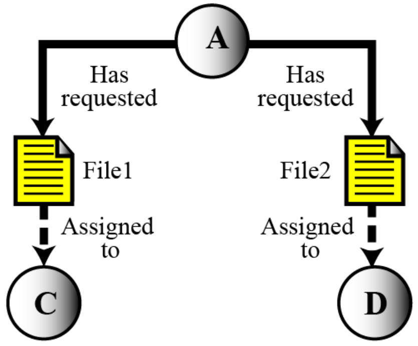
- Starvation is the opposite of deadlock. It can happen when the operating system puts too many resource restrictions on a process



a. Process A needs both File1 and File2



b. Process A still needs both File1 and File2



c. Process A still needs both File1 and File2 (starving)

Device manager

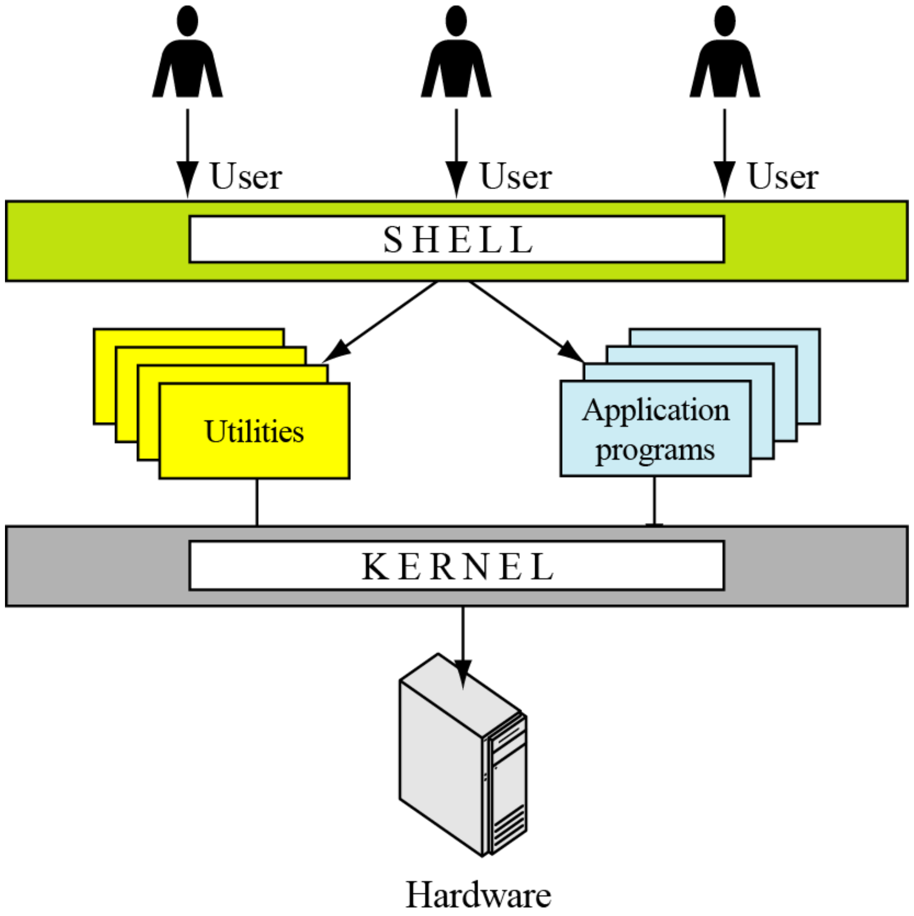
- The device manager, or input/output manager, is responsible for access to input/output devices
  - The device manager monitors every input/output device constantly to ensure that the device is functioning properly. The manager also needs to know when a device has finished serving one process and is ready to serve the next process in the queue
  - The device manager maintains a queue for each input/output device or one or more queues for similar input/output devices. For example, if there are two fast printers in the system, the manager can have one queue for each or one queue for both
  - The device manager controls the different policies for accessing input/output devices. For example, it may use FIFO for one device and the shortest length first for another

File manager

- File manager is used to controlling the access to files
  - The file manager controls access to files. Access is permitted only by permitted applications and/or users, and the type of access can vary
  - The file manager supervises the creation, deletion, modification and naming of files
  - The file manager supervises the storage of files: how they are stored, where they are stored, and so on
  - The file manager is responsible for archiving and backups

Example - UNIX

- UNIX consists of four major components: the *kernel*, the *shell*, a standard set of *utilities*, and *application programs*
- Kernel
  - It contains the most basic parts of the operating system: memory management, process management, device management, and file management
  - Another component of the kernel consists of a collection of *device drivers*, which are the software units that communicate with the controllers



Example - UNIX

- Shell
  - It receives and interprets the commands entered by the user
- Utilities
  - A utility is a standard UNIX program that provides a support process for users
  - Three common utilities are text editors, search programs, and sort programs
- Applications
  - Applications in UNIX are programs that are not a standard part of the operating system distribution
  - They provide extended capabilities to the system

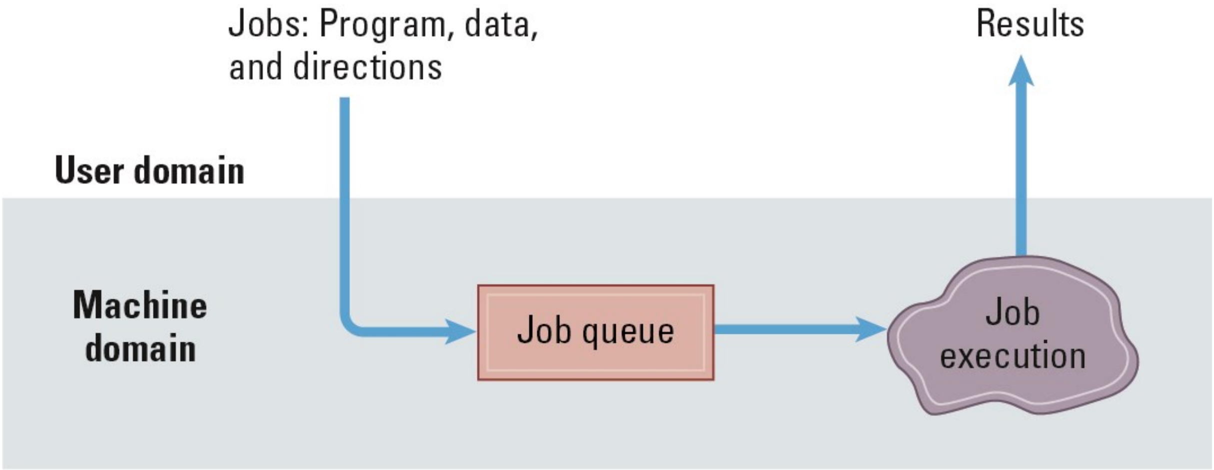
Appendix

Memory manager - Demand paging and segmentation

- *Demand paging and segmentation* can be combined to further improve the efficiency of the system
  - A segment may be too large to fit any available free space in memory. Memory can be divided into frames, and a module can be divided into pages. The pages of a module can then be loaded into memory one by one and executed

Evolution - Batch systems

- Batch operating systems were designed in the 1950s to control mainframe
  - At that time, computers used punched cards for input, line printers for output, and tape drives for secondary storage media
  - Each program to be executed was called a *job* and the punched cards were fed into the computer by an operator
  - Operating systems during this era were very simple: they only ensured that all of the computer's resources were transferred from one job to the next!



Time-sharing systems

- The idea of time sharing arises: resources could be shared between different jobs, with each job being allocated a portion of time to use a resource
  - One means of implementing time-sharing is to apply the technique called *multiprogramming*
  - For example, when one program is using an input/output device, the CPU is free and can be used by another program
- Time-sharing is hidden from the user - each user has the impression that the whole system is serving them exclusively!
  - The operating system now had to do *scheduling*: allocating resources to different programs and deciding which program should use which resource, and when
  - The user could directly interact with the system without going through an operator!

Personal systems

- When personal computers were introduced, there was a need for an operating system for this new type of computer
  - During this era, single-user operating systems such as DOS (Disk Operating System) were introduced

Parallel systems

- The need for more speed and efficiency led to the design of parallel systems: multiple CPUs on the same machine
  - Each CPU can be used to serve one program or a part of a program, which means that many tasks can be accomplished in parallel instead of serially
  - The operating systems required for this are more complex than those that support single CPUs

Distributed systems

- A job that was previously done on one computer can now be shared between computers that may be thousands of miles apart by network
  - A program may need files located in different parts of the world
  - Operating systems combine features of the previous generation with new duties such as controlling security

Real-time systems

- A real-time system is expected to do a task within specific time constraints
  - Examples can be found in traffic control, patient monitoring, or military control systems
  - The application program can sometimes be an embedded system such as a component of a large system, such as the control system in an automobile
  - The requirements for a real-time operating system are often different than those for a general-purpose system!

Example - UNIX

- UNIX is a very powerful operating system with three outstanding features
  1. UNIX is a portable operating system that can be moved from one platform to another without many changes. The reason is that it is written mostly in the C language (instead of a machine language specific to a particular computer system)
  2. UNIX has a powerful set of utilities (commands) that can be combined (in an executable file called a script) to solve many problems that require programming in other operating systems
  3. It is device-independent, because it includes device drivers in the operating system itself, which means that it can be easily configured to run any device

Example - Linux

- The initial kernel of Linux was similar to a small subset of UNIX and now it mainly contain the following components
  - Kernel
  - System libraries
    - The system libraries hold a set of functions used by the application programs, including the shell, to interact with the kernel
  - System utilities
    - The system utilities are individual programs that use the services provided by the system libraries to perform management tasks
- Linux supports the standard Internet protocols
- Linux' provides the security aspects defined traditionally for UNIX, such as authentication and access control

Example - Windows

- The design goals released by Microsoft are
  - Extensibility
    - Windows is designed as a modular architecture with several layers. The purpose is to let the higher layers to be changed with time without affecting the lower layers
  - Portability
    - Windows, like UNIX, is mostly written in C or C++ and the code is independent of the machine language of the computer on which it is running
  - Reliability
    - Windows was designed to handle error conditions including protection from malicious software
  - Compatibility
    - Windows was designed to run programs written for earlier versions of Windows
  - Performance
    - Windows was designed to have a fast response time to applications that run on top of the operating system

Example - Windows

- Windows uses a layered architecture
  - Hardware abstraction layer (HAL)
    - Hides hardware differences from the upper layers
  - Kernel
    - The kernel is the heart of the operating system. It is an object-oriented piece of software that sees any entity as an object
  - Executive (Runs in kernel (privileged) mode)
    - The Windows executive provides services for the whole operating system. It is made up of six subsystems: object manager, security reference monitor, process manager, virtual memory manager, local procedure call facility, and the I/O manager
  - Environmental subsystems (Runs in the user mode)
    - These are subsystems designed to allow Windows to run application programs designed for Windows, for other operating systems, or for earlier versions of Windows

Example - Windows