Lectures / Notes

Subject: INTRODUCTION TO OPERATING SYSTEM (2205)

For BCS (4th Semester)

Prepared by: Sir Muneer A. Shaikh

2205-INTRODUCTION TO OPERATING SYSTEM

Introduction: Introduction to operating system O/S. Monitors, Multiprogramming, and Multiprocessing, different classes of computers.

Process: Process Concepts, process models, critical selection problem, Inter-Process Communication.

Deadlocks: Deadlocks characterization, Deadlock prevention and detection, Dijkstra's method. Mutex, Semaphores, Philosopher's problems.

Scheduling: Scheduling concept, Scheduling Algorithm, first-come-first-served, shortest-job, first priority, etc.

Memory Management: Memory hierarchy, paging techniques, Segmentation, Paged Segmentation.

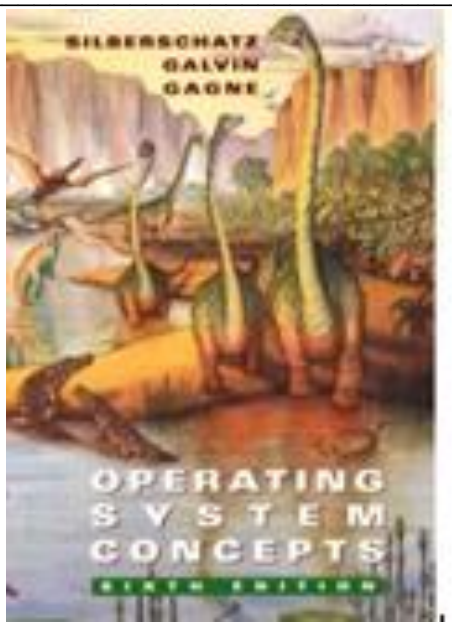
Virtual Memory: Paging Demand, Page Replacement, Algorithms, Allocation of frames, Thrashing.

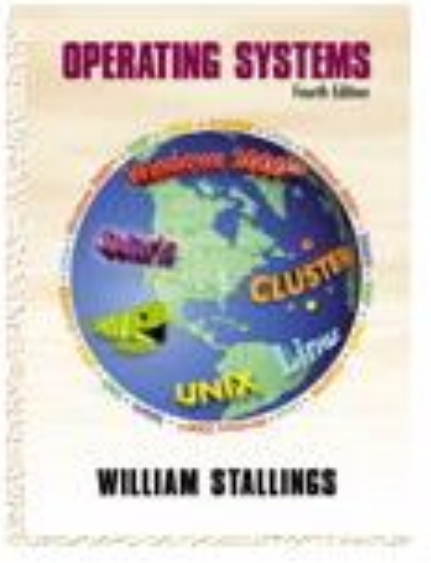
File System: File concept, Access Methods, File protection.

Introduction to Distributed Systems: Study and analysis of a currently use Operating System.

Books Recommended:

1. Stalling. W. Operating Systems Fundamentals.
2. Silberchatz. A. Peterson. JL. Operating System Concepts.



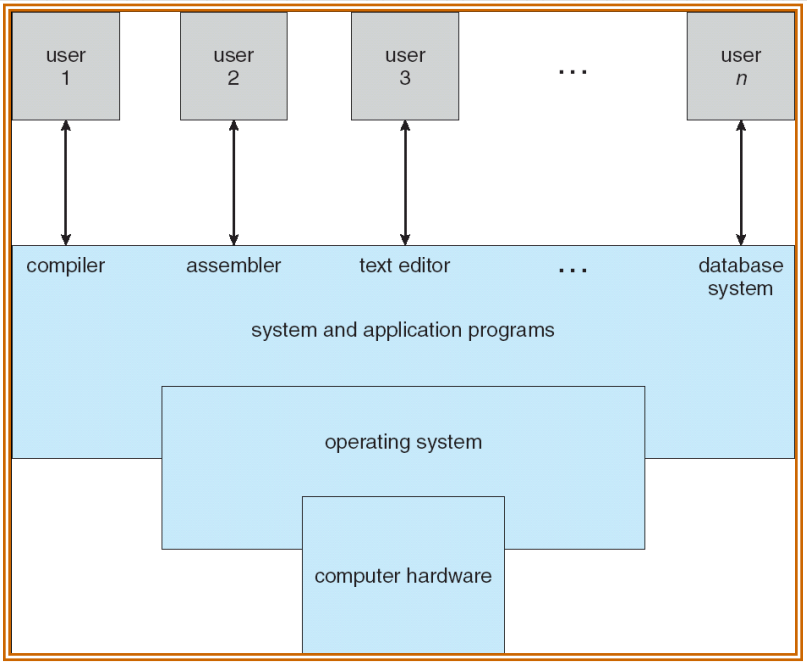


Lecture # 1

Chapter 1 : Introduction to Operating systems

Computer System Structure

- Computer system can be divided into four components
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers



Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EEPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Operating system:

- An operating system (OS) is software (programs and data) that runs on computers and manages the computer hardware and provides common services for efficient execution of various application software.
- OS acts as an intermediary between a user of a computer and the computer hardware. It enables the user to communicate with the hardware and get tasks done.
- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer
- Windows 7, Windows XP, OS/2, and UNIX are all operating systems.

Operating system goals:

- Executeusers programs and make solving user problems easier.
- Make the computer system convenient to use

Parts of Operating System:

- An operating system consists of many parts.
- **Kernel:** One of the most important components that running at all times on the computer is the kernel. This controls low-level processes that the average user usually cannot see: it controls how memory is read and written, the order in which processes are executed, how information is received and sent by devices like the monitor, keyboard and mouse, and deciding how to interpret information received by networks.
- **User Interface:** The user interface is the part of the operating system that interacts with the computer user directly, allowing them to control and use programs. The user interface may be graphical with icons and a desktop, or textual, with a command line.

Lecture # 2

Some OS Functions

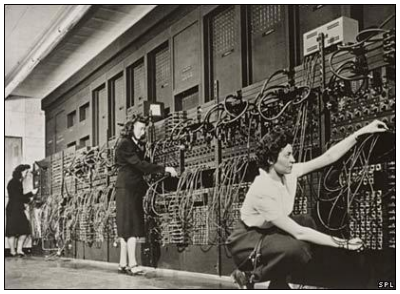
- The operating system manages many kinds of activities ranging from user programs to system programs like printer spooler, name servers, file server etc.
  - User authentication
  - Processor allocation (process scheduling)
  - File Management
  - Memory Management
  - Device Management
  - Network Management
  - Providing a user interface (graphical or not) to system functions and resources
  - Process synchronization
  - Interprocess communication.

History of Operating Systems

- Historically operating systems have been tightly related to the computer architecture, it is good idea to study the history of operating systems from the architecture of the computers on which they run.
- Operating systems have evolved through a number of distinct phases or generations which corresponds roughly to the decades.

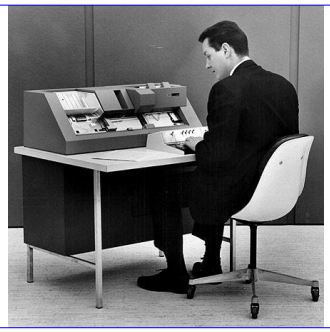
First generation 1945 - 1955

- vacuum tubes, plug boards
- Still very slow and used for scientific calculations
- No OS was needed
- Programs were entered by setting some switches
- It all started with computer hardware in about 1940s.



Second generation 1955 – 1965

- transistors, batch systems
- Universities started to buy computers (spending millions of dollars)
- Punched cards were used
- To run a job (a program or a set of related programs) first punch it and give the deck to the operators and wait for the output (batch operation)
- Computers were single user



Third generation 1965 – 1980

- Integrated Circuits
- The concept of spooling was introduced by 3rd generation operating systems and a memory partition was available, it is loaded directly to memory.
- Third generation OSs were still not interactive, i.e., whenever there is an error, programmers would realize it after a couple of hours.
- This problem lead to timesharing systems where the user has an online terminal

Fourth generation 1980 – present

- Personal computers were developed after LSI (Large Scale Integration) circuits were invented.
- First Microcomputer:
  - Intel 8080 CPU + attached 8-inch floppy disk
  - First disk based OS CP/M (Control Program for Microcomputers)
- In 1980s IBM designed the IBM PC and contacted Bill Gates for an operating System.
- Two operating systems have dominated the personal computer scene: MS-DOS, written by Microsoft, Inc. for the IBM PC, Intel 8088 CPU and its successors,

and UNIX, which is dominant on the large personal computers using the Motorola 6899 CPU family.

Operating Systems Services

Following are the five services provided by operating systems to the convenience of the users.

1. **Program Execution**
   - The purpose of a computer system is to allow the user to execute programs. So the operating systems provide an environment where the user can conveniently run programs. Running a program involves the allocating and de-allocating memory, CPU scheduling in case of multiprocessing.
2. **I/O Operations**
   - Each program requires an input and produces output. This involves the use of I/O. The operating systems hides the user the details of underlying hardware for the I/O. So the operating systems by providing I/O makes it convenient for the users to run programs.
3. **File System Manipulation**
   - The output of a program may need to be written into new files or input taken from some files. The operating systems provide this service.
4. **Communications**
   - There are instances where processes need to communicate with each other to exchange information. It may be between processes running on the same computer or running on the different computers. By providing this service the operating system relieves the user of the worry of passing messages between processes.

Error Detection

- An error is one part of the system may cause malfunctioning of the complete system. To avoid such a situation the operating system constantly monitors the system for detecting the errors. This relieves the user of the worry of errors propagating to various part of the system and causing malfunctioning.

Lecture # 3

Types of Operating Systems

- There are generally four types, categorized based on the types of computers they control and the sort of applications they support. The categories are:

1. **Real-time operating system (RTOS)** - Real-time operating systems are used to control machinery, scientific instruments and industrial systems.
2. **Single-user, single task** - As the name implies, this operating system is designed to manage the computer so that one user can effectively do one thing at a time. The Palm OS for Palm handheld computers is a good example of a modern single-user, single-task operating system.
3. **Single-user, multi-tasking** - This is the type of operating system most people use on their desktop and laptop computers today. Microsoft's Windows and Apple's MacOS platforms are both examples of operating systems that will let a single user have several programs in operation at the same time.
4. **Multi-user** - A multi-user operating system allows many different users to take advantage of the computer's resources simultaneously. Unix, VMS and mainframe operating systems, such as MVS, are examples of multi-user operating systems.

Kernel:

- Portion of operating system that is in main memory
- Contains most frequently used functions
- Also called the nucleus

Types of Kernel:

(1) Macro kernel: A large operating system core that provides a wide range of services.

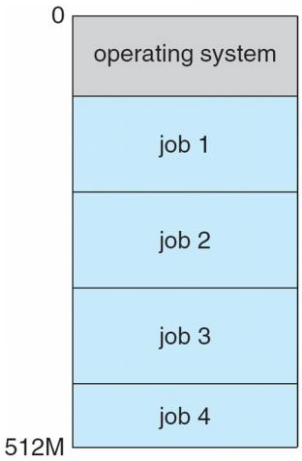
(2) Microkernel: A small privileged operating system core that provides process scheduling, memory management, and communication services and relies on other processes to perform some of the function traditionally associated with the operating system kernel.

(3) Monolithic Kernel: A large kernel containing virtually the complete operating system, including scheduling, file system, device drivers and memory management. All the functional components of the kernel have access to all of its internal data structures and

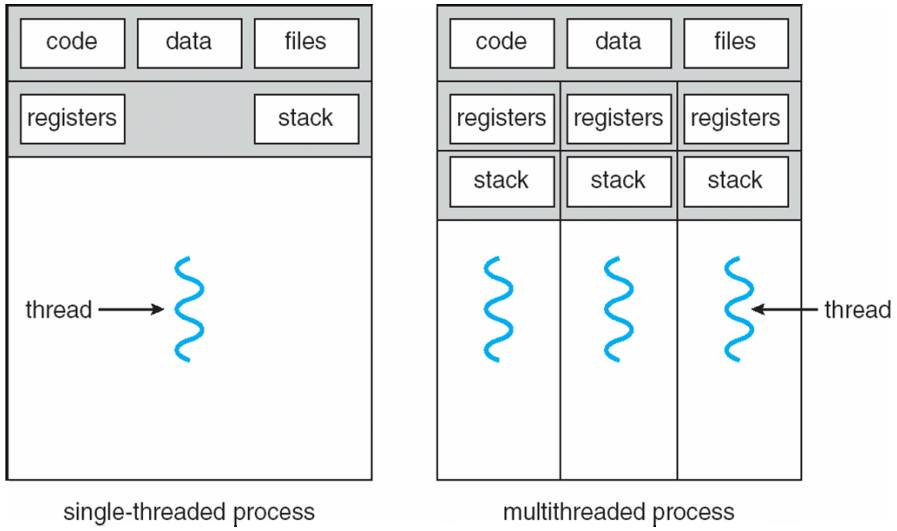
routines. Typically, a monolithic kernel is implemented as a single process, with all elements sharing the same address space.

Modern Operating Systems

- **Multiprogramming**
  - Use interrupts to run multiple programs simultaneously
  - Requires CPU scheduling to choose the next job to run.

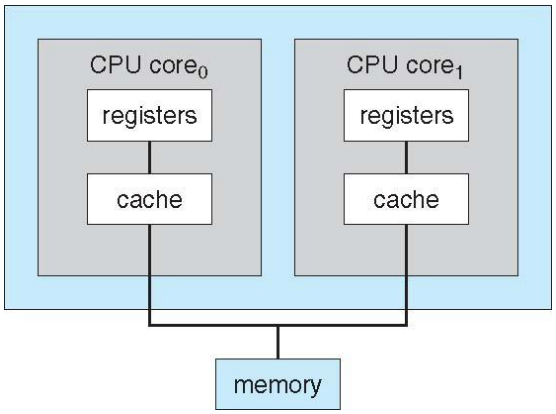


- **Multithreading**
  - Process is divided into threads that can run concurrently
    - Thread
      - Dispatchable unit of work
      - executes sequentially and is interruptable
    - Process is a collection of one or more threads



- **Symmetric multiprocessing (SMP)**

- There are multiple processors
- These processors share same main memory and I/O facilities
- All processors can perform the same functions
- A Dual-Core Design:



- **Distributed operating systems**
  - Collection of separate, possibly heterogeneous, systems networked together
  - Network Operating System provides features between systems across network
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

- **Special-Purpose Systems**
  - Real-time embedded systems most prevalent form of computers
    - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Multimedia systems
    - Streams of data must be delivered according to time restrictions
  - Handheld systems

- PDAs, smart phones, limited CPU, memory, power
- Reduced feature set OS, limited I/O

Lecture # 4

Linux vs. Windows

Windows

- Windows has different versions. It started with Win3.x, Win9x, WinME, WinNT, Win2000, WinXP. It is a proprietary software produced by Microsoft.
- Windows is expensive to run. The price to purchase a full version of Windows XP Professional is about USD400.
- Only one copy of Windows may be used on a computer. Activation with Microsoft is needed.
- Windows is a proprietary closed source software. Codes are not released to the public.

Linux

- Linux has different versions, depending on which vendor develops and runs it. Linux vendors include: Linspire, **Red Hat**, SuSE, Ubuntu, Mandriva, Knoppix, Slackware, Caldera, Debian
- Linux is cheap or free to run. It can be downloaded from various Linux vendors.
- Linux may run off a server so numerous computers may have access to the program.
- Linux is an open source software. Source codes are freely distributed to the public, of which programmers had reviewed to improve performance, eliminate bugs and strengthen security.

Definitions:

Shell: It is a piece of software, usually a separate program that provides direct communication between the user and the operating system. This usually, but not always, takes the form of a command-line interface. Example of shell is the MS-DOS command interface program COMMAND.COM.

Batch processing: A situation common in early mainframe environments in which many tasks were scheduled to run at a specific time late in the evening. The user never directly interacted with the computer in batch processing.

Client: A computer that accesses shared network resources provided by server.

Server A computer that provides shared resources to network users

Device A generic term for a computer subsystem. Printers, serial ports, and disk drives are referred to as devices.

Job: A set of computational steps packaged to run as a unit.

Multiprocessing: A mode f operation that provides for parallel processing by two or more processors of a multiprocessor. There are two primary types of multiprocessing: Symmetric multiprocessing (SMP) & Asymmetric Multiprocessing. In SMP each processor runs an identical copy of the operating system, and these copies communicate with one another as needed. In asymmetric multiprocessing, each processor is assigned a specific task. A master processor controls the system and schedules & allocates work to the salve processors.

Multiprogramming: A mode of operation that provides for the interleaved execution of two or more computer programs by a single processor. The same as multitasking.

Multitasking In operating systems, the ability to divide computer time among two or more running programs simultaneously. There are two primary types of multitasking: preemptive and nonpreemptive. In preemptive multitasking, the operating system can take control of the processor without the task's cooperation. In nonpreemptive multitasking, the processor is never taken from a task. The task itself decides when to give up the processor.

Time Sharing: The concurrent use of a device by a number of users.

Real Time system: An operating system that must schedule and manage real-time tasks.

Cluster: A group of interconnected, whole computers working together as a unified computing resource.

Distributed Operating System: A common operating system shared by a network of computers.

spooling (simultaneous peripheral operations on line): In spooling, a high-speed device like a disk interposed between a running program and a low-speed device involved with the program in input/output. Instead of writing directly to a printer, for example, outputs are written to the disk. Programs can run to completion faster, and other programs can be initiated sooner when the printer becomes available, the outputs may be printed.

Questions:

Q#1: What are the three main purposes of an operating system?

Answer: There are several possible purposes of an operating system:

- To provide an environment for a computer user to execute programs on computer hardware in a convenient and efficient manner.
- To allocate the separate resources of the computer as needed to solve the problem given. The allocation process should be as fair and efficient as possible.
- As a control program it serves two major functions: 1) supervision of the execution of ser programs to prevent errors and improper use of the computer, and 2) management f the operation and control of I/O devices.

Q#2: What is the main advantage of multiprogramming?

Answer: Multiprogramming makes efficient use of the CPU by overlapping the demands for the CPU and its I/O devices from various users. It attempts to increase CPU utilization by always having something for the CPU to execute.

Q#3: Describe the differences between symmetric and asymmetric multiprocessing. What are three advantages and one disadvantage of multiprocessor systems?

Answer: Symmetric multiprocessing treats all processors as equals, and I/O can be processed on any CPU. Asymmetric multiprocessing has one master CPU and the remainder CPUs are slaves. The master distributes tasks among the slaves, and I/O is usually done by the master only. Multiprocessors can save money, by not duplicating power supplies, housings, and peripherals. They can execute programs more quickly, and can have increased reliability. They are also more complex in both hardware and software than uniprocessor systems.

Q#4: What is spooling?

Answer: An acronym for "Simultaneous Peripheral Operation On-Line." It uses the disk as a large buffer for outputting data to line printers and other devices (like microfilm). It can also be used for input, but is generally used for output. Its main use is to prevent two users from alternately printing lines to the line printer on the same page, getting their output completely mixed together. It also helps in reducing idle time and overlapped I/O and CPU.

Q#5: How do I/O-bound and CPU-bound programs differ?

Answer:

- In I/O-bound programs, the CPU remains idle much of the time.
- In CPU-bound programs, the I/O processor remains idle.

Note: I/O is never "faster" than CPU.

Q#6: What is multiprogramming?

Answer: In multiprogramming, several programs are in memory concurrently; the system switches among the programs for efficient processing, and minimal idle time.

Fill in the blanks:

1) The _____ is a program that manages the computer hardware. (Operating system)
2) The _____ controls and coordinates the use of the hardware among various application programs for various users. (Operating system)
3) The operating system is designed to maximize _____. (Resource utilization)
4) The two goals of operating system are _____ & _____. (Convenience & Efficiency)
5) _____ increases CPU utilization by organizing jobs so that the CPU always has one to execute.
   (Multiprogramming)
6) If several jobs are ready to run at the same time, the system must choose among them. Making this decision is called _____. (CPU scheduling)
7) An _____ computer system provides direct communication between the user and the system.
   (interactive)
8) A _____ operating system allows many users to share the computer simultaneously. (time-shared)
9) A program loaded into memory and executing is commonly referred to as a _____. (Process)
10) The multiprocessing systems have three main advantages:
    (increased throughput, economy of scale & increased reliability)
11) All modern operating systems – including Windows 2000 & Linux – now provide support for _____ multiprocessing. (symmetric)

12) Distributed systems depend on _____ for their functionality.

(networking)

13) Centralized systems today act as _____ systems to satisfy requests generated by client systems.

(server)

14) Systems that control scientific experiments, medical imaging systems, industrial control systems are _____ systems. (real-time)

15) To improve the overall performance of the computer system, developers introduced the concept of _____.

(multiprogramming)

True / False:

1. The operating system is similar to a government.
   (T)
2. The operating system does not act as the manager of the resources.
   (F)
3. A more common definition is that the operating system is the one program running at all times on the computer system.
   (T)
4. The primary goal of some operating system is convenience for the user.
   (T)
5. The design of an operating system is an easy task. (F / complex task)
6. The operating system in the early computers was fairly simple.
   (T)
7. In a non-multi-programmed system, The CPU would not sit idle.
   (F)
8. In a multiprogramming system, the operating system simply switches to, and executes job and so on.
   (T)
9. All the jobs that enter the system are kept in the job pool.
   (T)
10. Time sharing (or multitasking) is a physical extension of multiprogramming.
    (F/logical)
11. Time sharing systems also provide mechanisms for concurrent execution.
    (T)
12. Multiprogramming and time sharing are not the central themes of modern operating systems.
    (F)
13. SMP mans that all processors are peers; no master-salve relationship exists between processors.(T)
14. Clustered systems are similar to parallel systems. (F / differ)

15. Multiprogramming also allows time sharing.

(T)

Lecture#5

Chapter 2 : Processes

Process

- The term "process" was first used by the designers of the MULTICS in 1960's. Since then, the term process, used somewhat interchangeably with 'task' or 'job'. The process has been given many definitions for instance
  - A program in Execution.
  - An asynchronous activity.
  - The entity to which processors are assigned.
- But the definition "Program in Execution" seems to be most frequently used.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- In computing, a process is an instance of a computer program that is being executed. Process is not the same as program. A process is more than a program code. A process is an 'active' entity as oppose to program which consider being a 'passive' entity.
- Process, on the other hand, includes:
  - Current value of Program Counter (PC)
  - Contents of the processors registers
  - Value of the variables
  - The process stack (SP) which typically contains temporary data such as subroutine parameter, return address, and temporary variables.
  - A data section that contains global variables.

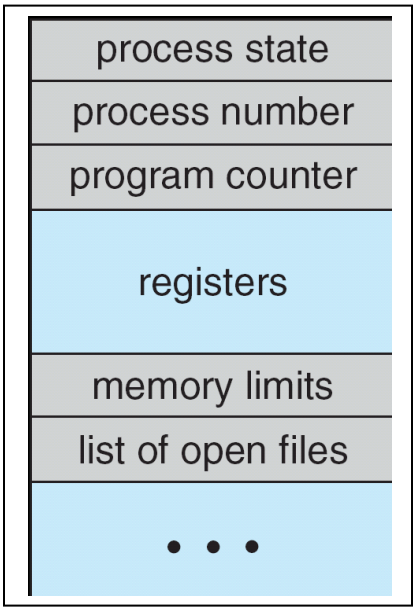
- All these processes can potentially execute concurrently, with the CPU (or CPUs) multiplexed among them. By switching the CPU between processes, the operating system can make the computer more productive.
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes

Resources:

- Resource: something a process uses
- Examples of computer resources
  - Printers
  - Semaphores / locks
  - Tables (in a database)
- Processes need access to resources in reasonable order
- Two types of resources:
  - Preemptable resources: can be taken away from a process with no ill effects
  - Nonpreemptable resources: will cause the process to fail if taken away
- Users on a network can share computer resources, such as hard disks, printers, modems, CD-ROM drives, and even the processor.

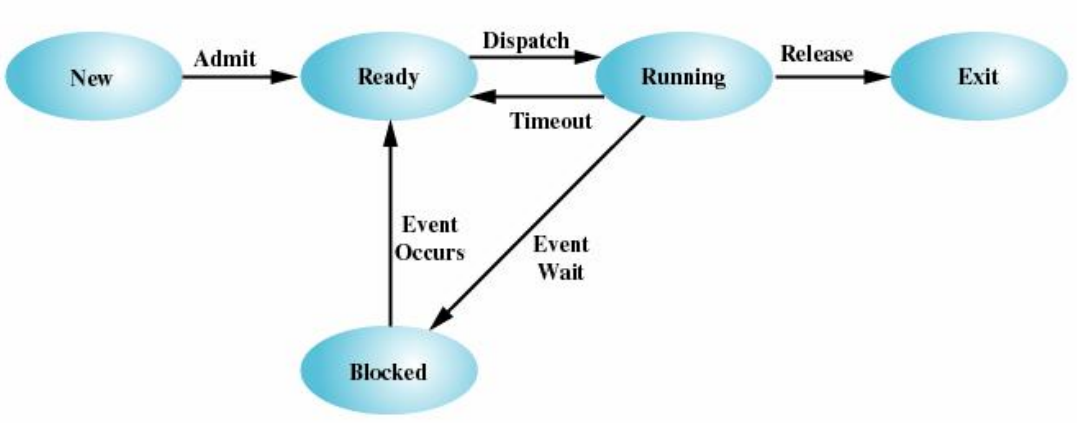
Process Control Block

- A process in an operating system is represented by a data structure known as a process control block (PCB) or process descriptor.
- The PCB contains important information about the specific process including
  - Process ID (PID)
  - Process State (Running, Ready, Blocked)
  - Process Priority
  - Parent Process ID
  - Child Process ID
  - Program counter
  - CPU registers
  - CPU scheduling information
  - Memory-management information
  - Accounting information
  - I/O status information



Lecture # 6

Process States

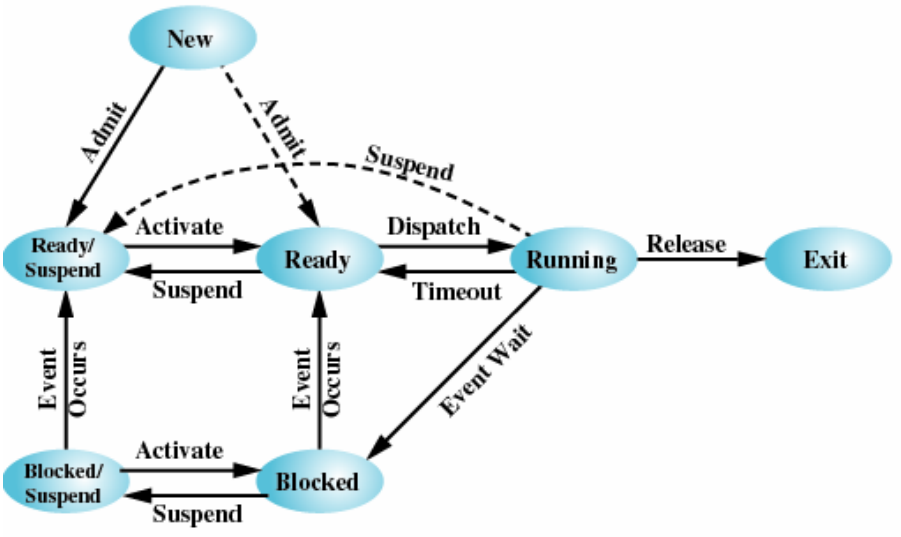


- As a process executes, it changes *state*
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **blocked**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a process
  - **exit**: The process has finished execution
- Logically, the 'Running' and 'Ready' states are similar. In both cases the process is willing to run, only in the case of 'Ready' state, there is temporarily no CPU available for it.
- The 'Blocked' state is different from the 'Running' and 'Ready' states in that the process cannot run, even if the CPU is available.

Process State Transitions

- Following are six (6) possible transitions among above mentioned five (5) states
  1. Block (process-name): Running → Block.
  2. Time-Run-Out (process-name): Running → Ready.
  3. Dispatch (process-name): Ready → Running.
  4. Wakeup (process-name): Blocked → Ready.
  5. Admitted (process-name): New → Ready.
  6. Exit (process-name): Running → Terminated.

Seven-State Process Model



Ready/Suspend:

- In systems that support virtual memory, a process may be swapped out, that is removed from main memory and placed in virtual memory by the mid-term scheduler. From here the process may be swapped back into the waiting state.

Blocked/Suspend:

- Processes that are blocked may also be swapped out. In this event the process is both swapped out and blocked, and may be swapped back in again under the same circumstances as a swapped out and waiting process (although in this case, the process will move to the blocked state, and may still be waiting for a resource to become available).

Process Management Activities

- The operating system is responsible for the following activities in connection with process management:
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication
  - Providing mechanisms for deadlock handling

Process Operations:

- Create
- Termination / Destroy
- Suspend
- Resume
- Preempt
- Dispatch
- Change priority
- Block
- Awaken
- Enable inter-process communications

Process Creation

- In general-purpose systems, some way is needed to create processes as needed during operation.
- There are four principal events led to processes creation.
  - User logs on.
  - User starts a program.
  - Operating systems creates process to provide service, e.g., to manage printer.
  - Some program starts another process, e.g., Netscape calls xv to display a picture.
- A process may create a new process, creating process is called parent process and the created one is called the child processes.
- After a process is created, both the parent and child have their own distinct address space.

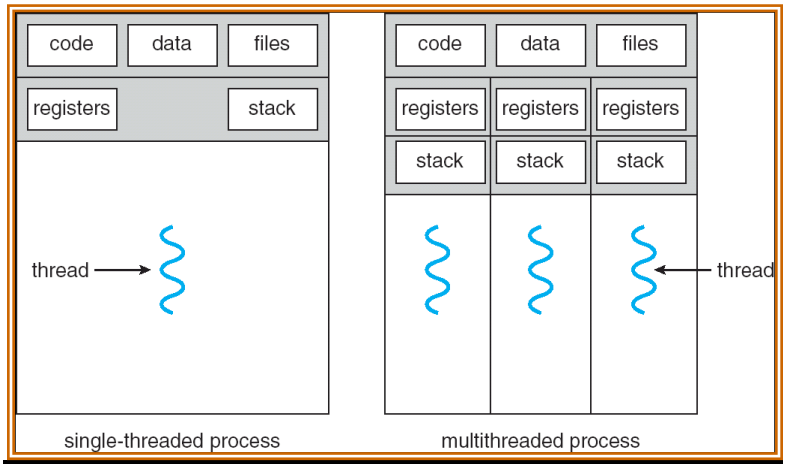
Process Termination

- A process terminates when it finishes executing its last statement.
- Its resources are returned to the system, and its process control block (PCB) is erased i.e., the PCB's memory space is returned to a free memory pool.

Lecture # 7

Threads

- Dispatchable unit of work
- executes sequentially and is interruptable
- Process is a collection of one or more threads
- Because threads have some of the properties of processes, they are sometimes called lightweight processes.
- In a process, threads allow multiple executions of streams.
- In many respect, threads are popular way to improve application through parallelism.
- The CPU switches rapidly back and forth among the threads giving illusion that the threads are running in parallel.
- Like a traditional process i.e., process with one thread, a thread can be in any of several states (Running, Blocked, Ready or Terminated).
- Each thread has its own stack.
- An operating system that has thread facility, the basic unit of CPU utilization is a thread.
- A thread has or consists of a program counter (PC), a register set, and a stack space.
- Threads are not independent of one other like processes as a result threads shares with other threads their code section, data section, OS resources also known as task, such as open files and signals.



- Single-threaded process has one program counter specifying location of next instruction to execute
    - Process executes instructions sequentially, one at a time, until completion

- Multi-threaded process is divided into threads that can run concurrently

Processes Vs Threads

Similarities

- Like processes threads share CPU and only one thread active (running) at a time.
- Like processes, threads within a process execute sequentially.
- Like processes, thread can create children.
- And like process, if one thread is blocked, another thread can run.

Differences

- Unlike processes, threads are not independent of one another.
- Unlike processes, all threads can access every address in the task.
- Unlike processes, threads are design to assist one other. Note that processes might or might not assist one another because processes may originate from different users.

Types of Threads

1. **User-Level Threads**
    - User-level threads implement in user-level libraries, rather than via systems calls, so thread switching does not need to call operating system and to cause interrupt to the kernel.
    - In fact, the kernel knows nothing about user-level threads and manages them as if they were single-threaded processes.
    - Three primary thread libraries:
        - POSIX threads
        - Win32 threads
        - Java threads

Advantages:

- The most obvious advantage of this technique is that a user-level threads package can be implemented on an Operating System that does not support threads.
- User-level threads do not require modification to operating systems.
- Simple Representation: Each thread is represented simply by a PC, registers, stack and a small control block, all stored in the user process address space.
- Simple Management: This simply means that creating a thread, switching between threads and synchronization between threads can all be done without intervention of the kernel.

- Fast and Efficient: Thread switching is not much more expensive than a procedure call.

Disadvantages:

- There is a lack of coordination between threads and operating system kernel. Therefore, process as whole gets one time slice irrespective of whether process has one thread or 1000 threads within. It is up to each thread to relinquish control to other threads.
- User-level threads require non-blocking systems call i.e., a multithreaded kernel. Otherwise, entire process will blocked in the kernel, even if there are runable threads left in the processes. For example, if one thread causes a page fault, the process blocks.

2. Kernel-Level Threads

- In this method, the kernel knows about and manages the threads.
- Instead of thread table in each process, the kernel has a thread table that keeps track of all threads in the system. In addition, the kernel also maintains the traditional process table to keep track of processes. Operating Systems kernel provides system call to create and manage threads.
- Examples
  - Windows XP/2000
  - Solaris
  - Linux

Advantages:

- Because kernel has full knowledge of all threads, Scheduler may decide to give more time to a process having large number of threads than process having small number of threads.
- Kernel-level threads are especially good for applications that frequently block.

Disadvantages:

- The kernel-level threads are slow and inefficient. For instance, threads operations are hundreds of times slower than that of user-level threads.
- Since kernel must manage and schedule threads as well as processes. It require a full thread control block (TCB) for each thread to maintain information about threads. As a result there is significant overhead and increased in kernel complexity.

Context Switch:

- Action performed by the operating system to remove a process from the run state and replace it with another.
    - Step 1: save the complete state of the running process (save the register contents, etc.)
    - Step 2: restore the complete state of the ready process which is to run (restore the register contents, etc.)
- Context-switch time is overhead; the system does no useful work while switching

Lecture#8

Inter-process Communication

- Inter-Process communications or IPC is the mechanism whereby one process can communicate, that is, exchange data, with another process.
- Inter-process communication (IPC) is a set of techniques for the exchange of data among multiple threads in one or more processes.
- Processes may be running on one or more computers connected by a network.
- IPC techniques are divided into methods. Main IPC Methods:
  - Message passing,
  - Synchronization (Semaphores),
  - Shared memory, and
  - Remote procedure calls (RPC).

Kind of Synchronization / IPC

- ✓ Mutual-Exclusion (MUTEX)
- ✓ Semaphores
- ✓ Monitors

Mutual-Exclusion

- A condition in which there is a set of processes, only one of which is able to access a given resource or perform a given function at any time.

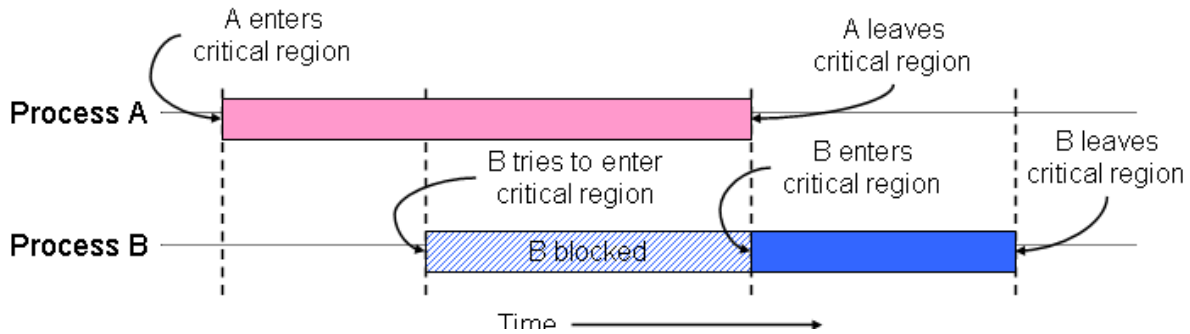
Critical Section / region:

- A section of code that only one process at a time can be executing

Mutual Exclusion Conditions

- Use critical regions to provide *mutual exclusion* and help fix race conditions
- Four conditions to provide mutual exclusion
  - No two processes simultaneously in critical region
  - No assumptions made about speeds or numbers of CPUs

- No process running outside its critical region may block another process
- No process must wait forever to enter its critical region



Mutual Exclusion Example

- Using a queue with shared memory, while one process is executing ``add", no other process should be allowed to access the queue data structure.
- **Example:** In a file update, where records are read, modified and written back, no other process should be allowed access to a record while one process is changing it.
- **Example:** On a single lane bridge, no other vehicle should be allowed on the bridge while a vehicle is on it.

Busy waiting: strict alternation

- Use a shared variable (turn) to keep track of whose turn it is
- Waiting process continually reads the variable to see if it can proceed
  - This is called a *spin lock* because the waiting process "spins" in a tight loop reading the variable
- Avoids race conditions, but doesn't satisfy criterion 3 for critical regions

Eliminating busy waiting

- Problem: previous solutions waste CPU time
  - Both hardware and software solutions require spin locks
  - Allow processes to sleep while they wait to execute their critical sections
- Solution: use semaphores
  - Synchronization mechanism that doesn't require busy waiting

Lecture#9

Semaphore:

- An integer value used for signaling among processes.
- The fundamental principle is this: Two or more processes can cooperate by means of simple signals, such that a process can be forced to stop as a specified place until it has received a specific signal.
- To achieve the desired effect, we can view the semaphore as a variable that has an integer value upon which three operations are defined:
  i. A semaphore may be initialized to a non-negative value.
  ii. The wait operation decrements the semaphore value. If the value becomes negative, then the process executing the wait is blocked.
  iii. The signal operation increments the semaphore value. If the value is not positive, then a process blocked by a wait operation is unblocked.

Types of semaphores

- Two different types of semaphores
  - Counting semaphores
  - Binary semaphores
- Counting semaphore
  - Value can range over an unrestricted range
- Binary semaphore
  - Only two values possible
    - 1 means the semaphore is available
    - 0 means a process has acquired the semaphore
  - May be simpler to implement
- For both counting semaphores and binary semaphores, a queue is used to hold processes waiting on the semaphore. The question arises of the order in which processes are removed from such a queue.
- The fairest policy is first-in-first-out(FIFO): The process that has been blocked the longest is released from the queue first; a semaphore whose definition includes this policy is called a **Strong Semaphore**. A semaphore that does not specify the order in which processes are removed from the queue is a **Weak Semaphore**.

A program Mutual Exclusion Using Semaphore:

```
/* program mutual exclusion */
Const int n= /* number of processes */

Semaphore s=1;
Void P(int i)
{
while (true)
{
waits(s);
/* critical section */
signal(s);
}
}
Void main()
{
parbegin(P(1), P(2), ……,P(n))
}
```

Monitors

- A *monitor* is another kind of high-level synchronization primitive
  - One monitor has multiple entry points
  - Only one process may be in the monitor at any time
  - Enforces mutual exclusion - less chance for programming errors
- Monitors provided by high-level language
  - Variables belonging to monitor are protected from simultaneous access
  - Procedures in monitor are guaranteed to have mutual exclusion
- Monitor implementation
  - Language / compiler handles implementation
  - Can be implemented using semaphores

Classical synchronization problems

- Bounded Buffer
  - Multiple producers and consumers
  - Synchronize access to shared buffer
- Readers & Writers
  - Many processes that may read and/or write

- Only one writer allowed at any time
- Many readers allowed, but not while a process is writing

- Dining Philosophers
  - Resource allocation problem
  - N processes and limited resources to perform sequence of tasks
- Goal: use semaphores to implement solutions to these problems

Lecture#10

Problem: ***The Cigarette-Smokers Problem***. Consider a system with three *smoker* processes and one *agent* process. Each smoker continuously rolls a cigarette and then smokes it. But to roll and smoke a cigarette, the smoker needs three ingredients: tobacco, paper, and matches. One of the smoker processes has paper, another has tobacco, and the third has matches. The agent has an infinite supply of all three materials. The agent places two of the ingredients on the table. The smoker who has the remaining ingredient then makes and smokes a cigarette, signaling the agent on completion. The agent then puts out another two of the three ingredients, and the cycle repeats. Write a program to synchronize the agent and the smokers.

Answer: The shared data structures are

var *a*: array [0...2] *of* semaphore {initially = 0}

agent: semaphore {initially = 1}

The agent process code is as follows:

rep

repeat

Set *i, j* to a value between 0 and 2.

wait(*agent*);

signal(*a*[*i*]);

signal(*a*[*j*]);

until *false*;

Each smoker process needs two ingredients represented by integers *r* and *s* each with value between 0 and 2.

Rep

repeat

wait(*a*[*r*]);

wait(*a*[*s*]);

"smoke"

signal(*agent*);

until *false*;

Producer/Consumer Problem

- One or more producers are generating data and placing these in a buffer
- A single consumer is taking items out of the buffer one at time
- Only one producer or consumer may access the buffer at any one time

producer:

```
while (true) {
        /*      produce
item v */
        b[in] = v;
        in++;
}
```

consumer:

```
while (true) {
        while   (in   <=
out)
        /*do    nothing
*/;
        w = b[out];
        out++;
        /*      consume
item w */
}
```

Barbershop Problem

An other example of the use of semaphores to implement concurrency, we consider a simple barbershop problem. Our barbershop has three chairs, three barbers, and a waiting area that can accommodate four customers on a sofa and that has standing room for additional customers (as shown in fig.). In this example, we assume that the barbershop will process 50 customers.

A customer will not enter the shop if it is filled to capacity with other customers. Once inside, the customer takes a seat on the sofa or stands if the sofa is filled. When a barber is free, the customer that has been on the sofa the longest is served and if there are any standing customers, the one that has been in the shop the longest takes a seat on the sofa. When a customer's haircut is finished, any barber can accept payment, but because there is only one cash register, payment is accepted for one customer at a time. The barbers divide their time among cutting hair, accepting payment and sleeping in their chair waiting for a customer.

- ☐ The barber shop has only three chairs for customers so only three barbers are available
- ☐ The waiting area has the capacity of four customers
- ☐ Only 20 customers can be accommodated within the shop at any time

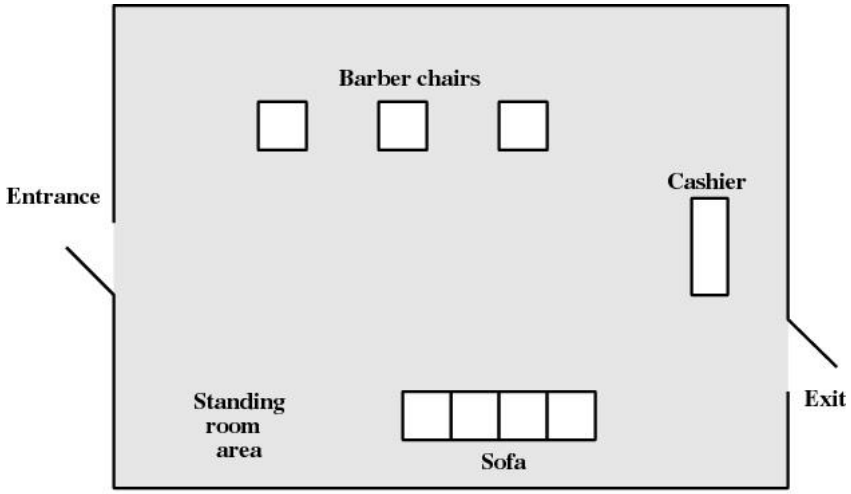


Figure 5.18 The Barbershop

- ☐ When a barber is free, the longest waiting customer on the sofa moves to the chair
- ☐ The longest standing customer then sits on the sofa
- ☐ Payment is accepted one at a time
- ☐ Barbers cut hair, accept payment and sleep if no customers are around

```
/* program barbershop1 */

semaphore max_capacity = 20;
semaphore sofa = 4;
semaphore barber_chair = 3;
semaphore coord = 3;
semaphore cust_ready = 0, finished = 0, leave_b_chair = 0, payment= 0,      receipt = 0;

void main()
{
parbegin (customer, …. 50 times…, customer, barber, barber, barber, cashier);
}

//  50 customers, 3 barbers and one cashier routine

 Void customer()
 {
 wait(max_capacity);
 enter_shop();
 wait(sofa);
 sit_on_sofa();
 wait(Barbaer_chair);
 get_up_from_sofa();
 signal(sofa);
 sit_in_barber_chair();
 signal(cust_ready);
 wait(finished);
 leave_barber_chair();
 signal(leave_b_chair);
 pay();
 signal(payment);
 wait(receipt);
 exit_shop();
 signal(max_capacity)
 }

 Void Barber()
 {
 while (true)
 {
 wait(cust_ready);
 wait(coord);
 cut_hair():
 signal(coord);
 signal(finished);
 wait(leave_b_chair);
 signal(barber_chair);
 }
 }

 Void Cachier()
 {
 while (true)
 {
 wait(payment);
 wait(coord);
 accept_pay();
 signal(coord);
 signal(receipt);
 }
 }
```

Philosophers Problem

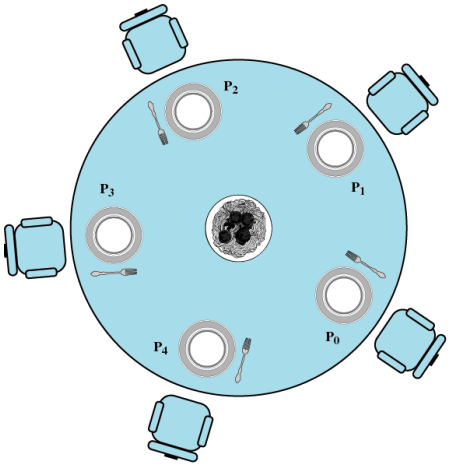


Figure 6.11 Dining Arrangement for Philosophers

- *N* philosophers around a table
  - All are hungry
  - All like to think
- *N* chopsticks available
  - 1 between each pair of philosophers
- Philosophers need two chopsticks to eat
- Philosophers alternate between eating and thinking
- Goal: coordinate use of chopsticks
- Use a semaphore for each chopstick
- A hungry philosopher
  - Gets the chopstick to his right
  - Gets the chopstick to his left
  - Eats
  - Puts down the chopsticks
- Potential problems?
  - Deadlock
  - Fairness

Coding:

```
Shared variables
const int n;
// initialize to 1
Semaphore chopstick[n];

Code for philosopher i
while(1) {
  chopstick[i].down();
  chopstick[(i+1)%n].down();
  // eat
  chopstick[i].up();
  chopstick[(i+1)%n].up();
  // think
}
```

Definitions

Concurrent: Pertaining to processes, that take place within a common interval of time during which they may have to alternately share common resources.

Scheduling: selecting jobs or tasks that are to be dispatched.

Synchronization: situation in which two or more processes coordinate their activities based on a condition.

Device A generic term for a computer subsystem. Printers, serial ports, and disk drives are referred to as devices.
Event An action or occurrence to which a program might respond. Examples of events are mouse clicks, key presses, and mouse movements.

Interrupt: A suspension of a process. An electronic signal sent to a computer's CPU to indicate that an event has taken place which requires the processor's attention.

Thread: A light weight process. An execution context that is independently scheduled but shares a single address space with other threads.

Message: A block of information that may be exchanged between processes as a means of communication.

Busy Waiting: The repeated execution of a loop of code while waiting for an event to occur.

Dispatch: To allocate time on a processor to jobs or tasks that are ready for execution.

Mailbox: A data structure shared among a number of processes that is used as a queue for messages. Messages are sent to the mailbox and retrieved from the mailbox rather than passing directly from sender to receiver.

Review Questions

Q#1: What is a process?

Answer: A program in execution.

Q#2: What is a PCB?

Answer: Process control block contains various data structures.

Q#3: What is the "degree of multiprogramming?"

Answer: Number of jobs in the current mix of running/waiting jobs.

Q#4: What is time sharing? What kind of scheduling does it involve?

Answer: Time sharing is many users interactively using a system "simultaneously;" each user gets a share of CPU-time, after other users have gotten their share. It uses Mediumterm scheduling, such as round-robin for the foreground.

Q#5: What is swapping?

Answer: Copying a process out of memory onto a fast disk or drum, to allow space for other active processes; it will be copied back into memory when space is ample.

Q#6: What is a context switch?

Answer: The time needed to switch from one job to another.

Q#7: Describe the producer/consumer problem.

Answer: The consumer can't be allowed to use a result until the producer has created that result, and the producer can't be allowed to create results if the buffers are all full.

Q#8: Describe the differences among short-term, medium-term, and long-term scheduling.

Answer:

Short-term (CPU scheduler): selects from jobs in memory, those jobs which are ready to execute, and allocates the CPU to them.

Medium-term: used especially with time-sharing systems as an intermediate scheduling level. A swapping scheme is implemented to remove partially run programs from memory and reinstate them later to continue where they left off.

Long-term (job scheduler): determines which jobs are brought into memory for processing.

The primary difference is in the frequency of their execution. The short-term must select a new process quite often. Long-term is used much less often since it handles placing jobs in the system, and may wait a while for a job to finish before it admits another one.

Q#10: What two advantages do threads have over multiple processes? What major disadvantage do they have? Suggest one application that would benefit from the use of threads, and one that would not.

Answer: Threads are very inexpensive to create and destroy, and they use very little resources while they exist. They do use CPU time for instance, but they don't have totally separate memory spaces. Unfortunately, threads must "trust" each other to not damage shared data. For instance, one thread could destroy data that all the other threads rely on, while the same could not happen between processes unless they used a system feature to allow them to share data. Any program that may do more than one task at once could benefit from multitasking. For instance, a program that reads input, processes it, and outputs it could have three threads, one for each task. "Single-minded" processes would not benefit from multiple threads; for instance, a program that displays the time of day.

Q#11: What are the differences between user-level threads and kernel-supported threads? Under what circumstances is one type "better" than the other?

Answer: User-level threads have no kernel support, so they are very inexpensive to create, destroy, and switch among. However, if one blocks, the whole process blocks. Kernel-supported threads are more expensive because system calls are needed to create and destroy them and the kernel must schedule them. They are more powerful because they are independently scheduled and block individually.

Q#12: What is a critical section?

Answer: A section of code that only one process at a time can be executing.

Q#13: What is the critical-section problem?

Answer: To design an algorithm that allows at most one process into the critical section at a time, without deadlock.

Q#14: Describe the Dining Philosophers problem.

Answer: Five philosophers around a table can either eat or think; they share five chopsticks; but each needs two to eat. The problem is to design an algorithm so that there is no deadlock or starvation.

Q#15. What is busy waiting?

Answer: The repeated execution of a loop of code while waiting for an event to occur is called busy-waiting. The CPU is not engaged in any real productive activity during this period, and the process does not progress toward completion.

Fill in the blanks:

1. Informally, a _____ is a program in execution. (Process)
2. Each process may be in one of the following states: (New, Running, Waiting, Ready, Terminated)
3. Only process can be _____ on any processor at any instant, although many processes may be ready and waiting.
   (running)
4. Each process is represented in the operating system by a _____. (process control block)
5. Many modern operating systems have extended the process concept to allow a process to have multiple _____ of executions.
   (Threads)
6. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the _____. (Ready queue)
7. The list of processes waiting for a particular I/O device is called a _____. (Device queue)
8. The selection of process is carried out by the appropriate _____. (Scheduler)
9. The two main categories of schedulers are: (long-term & short-term schedulers)
10. In general, most processes can be described as either /O bound or ____. (CPU bound)
11. Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process, this task is known as a _____. (context switch)
12. A process may create several new processes, via a _____ system call, during the course of execution. (create-process)
13. The creating process is called a _____, whereas the new processes are called the _____ of that process. (Parent, children)
14. A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the ____ system call. (exit)
15. The concurrent processes executing in the operating system may be either _____ processes or ____ processes. (independent , cooperating)
16. A process is _____ if it can not affect or be affected by the other processes executing in the system. (independent)
17. Any process that shares data with other processes is a _____ process. (cooperating)

18. Concurrent execution of cooperating processes requires a _____ mechanism that allow processes to communicate with other one another. (inter-process communication ICP)
19. The _____ queue contains all the processes that are ready to execute and are waiting for the CPU. (Ready queue)
20. The ___ scheduling is the selection of one process from the ready queue. (CPU)

True / False:

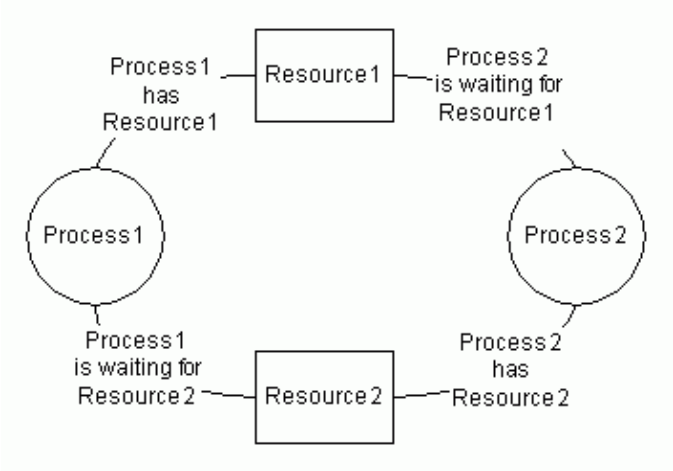
1. By switching the CPU between processes, the operating system can make the computer more productive. (T)
2. As a process executes, it does not change state. (F)
3. The objective of time-sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running. (T)
4. A uni-processor system can have many running processes. (f / only one)
5. Ready queue is generally stored as a linked list. (T)
6. Each device has not its own device queue. (F)
7. A new process is initially put in the ready queue. (T)
8. A process migrates between the various scheduling queues throughout its life time. (T)
9. The long-term scheduler must select a new process for the CPU frequently. (F)
10. The long-term scheduler controls the degree of multiprogramming. (T)
11. The operating sysem must provide a mechanism for process creation & termination. (T)
12. When a process creates a sub-process, that sub-process may not be able to obtain its resources directly from the operating system (F)
13. Any process that does not share any data with any other process is independent. (T)
14. The state of a process is defined by that process's current activity. (T)
15. A process, when it is not executing, it is placed in running state. (F/waiting queue)
16. Concurrent execution requires mechanisms for process creation and deletion. (T)
17. The cooperating processes must not have the means to communicate with each other. (F)
18. The shared-memory method requires communicating processes to share some variables. (T)
19. The message system method allows the processes to exchange messages. (T)
20. The two schemes for inter-process communication, shared-memory & message systems are mutually exclusive, and can not be used simultaneously within a single operating system. (F)

Lecture # 11

Chapter 3 : Deadlocks

Deadlock:

- A situation that occurs when multiple processes are waiting for the availability of resources that will not become available because it is being held by another process that is in a similar wait state.
- In deadlock, none of the processes can
  - Run
  - Release resources
  - Be awakened
- The resources may be either physical or logical. Examples of physical resources are Printers, CD ROM Drivers, Memory Space, and CPU. Examples of logical resources are Files, Semaphores, and Monitors.
- Deadlocks occur when …
  - Processes are granted exclusive access to devices or software constructs (resources)
  - Each deadlocked process needs a resource held by another deadlocked process
- The simplest example of deadlock is where process 1 has been allocated non-shareable resources *1*, say, a CD ROM drive, and process 2 has be allocated non-sharable resource 2, say, a printer. Now, if it turns out that process 1 needs resource *2* (printer) to proceed and process 2 needs resource *1*(the CD ROM drive) to proceed and these are the only two processes in the system, each is blocked the other and all useful work in the system stops.
- The system is in deadlock state because each process holds a resource being requested by the other process neither process is willing to release the resource it holds.



Using resources

- Sequence of events required to use a resource
  - Request the resource
  - Use the resource
  - Release the resource
- Can't use the resource if request is denied
  - Requesting process has options
    - Block and wait for resource
    - Continue (if possible) without it: may be able to use an alternate resource
    - Process fails with error code

Preemptable and Nonpreemptable Resources

- Resources come in two flavors: preemptable and nonpreemptable.
- A preemptable resource is one that can be taken away from the process with no ill effects. Memory is an example of a preemptable resource.
- On the other hand, a nonpreemptable resource is one that cannot be taken away from process (without causing ill effect). For example, *CD* resources are not preemptable at an arbitrary moment.
- Reallocating resources can resolve deadlocks that involve preemptable resources.
- Deadlocks that involve nonpreemptable resources are difficult to deal with.

Four Necessary Conditions for Deadlock

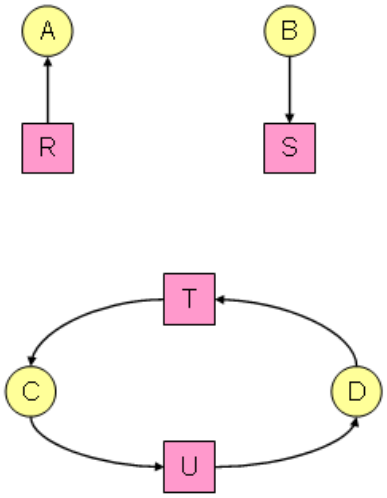
Deadlock can arise if four conditions hold simultaneously.

- **Mutual exclusion**
  - Each resource is assigned to at most one process.
  - Only one process at a time can use a resource.
- **Hold and wait**
  - A process holding at least one resource is waiting to acquire additional resources held by other processes
- **No preemption**
  - Previously granted resources cannot be forcibly taken away
  - A resource can be released only voluntarily by the process holding it, after that process has completed its task.

- **Circular wait**
  - There must be a circular chain of 2 or more processes where each is waiting for a resource held by the next member of the chain
  - There exists a set {*P*0, *P*1, …, *P*0} of waiting processes such that *P*0 is waiting for a resource that is held by *P*1, *P*1 is waiting for a resource that is held by *P*2, …, *Pn*–1 is waiting for a resource that is held by *P*n, and *Pn* is waiting for a resource that is held by *P*0.

Lecture # 12

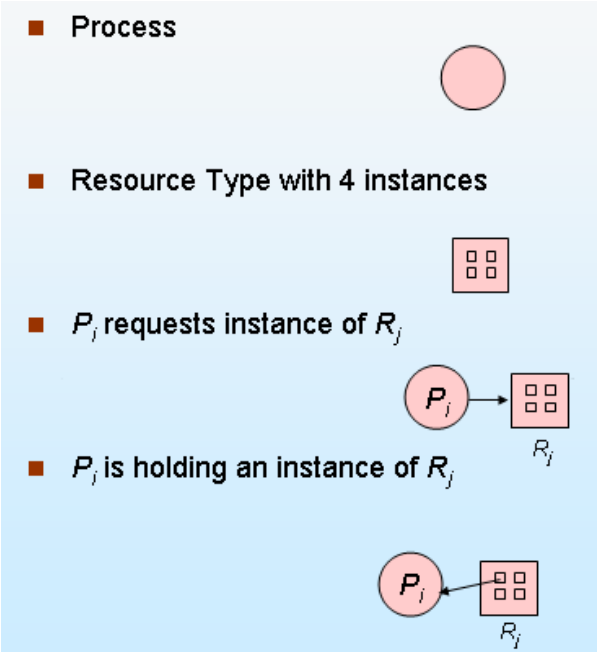
Resource allocation graphs



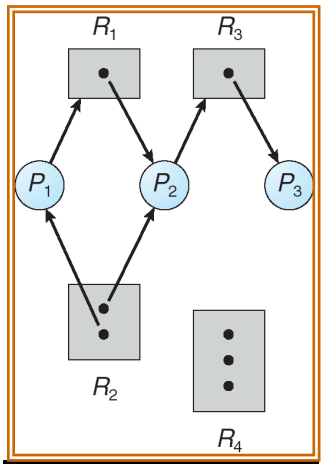
- Resource allocation modeled by directed graphs
- Example 1:
  - Resource R assigned to process A
- Example 2:
  - Process B is requesting / waiting for resource S
- Example 3:
  - Process C holds T, waiting for U
  - Process D holds U, waiting for T
  - C and D are in deadlock!

A set of vertices *V* and a set of edges *E*.

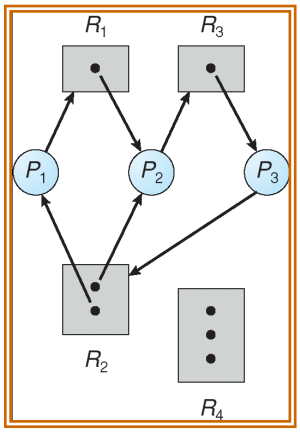
- V is partitioned into two types:
  - $P = \{P1, P2, \ldots, Pn\}$, the set consisting of all the processes in the system.
  - $R = \{R1, R2, \ldots, Rm\}$, the set consisting of all resource types in the system.
- request edge – directed edge $P1 \rightarrow Rj$
- assignment edge – directed edge $Rj \rightarrow Pi$



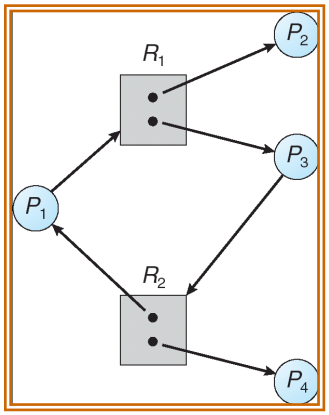
Example of a Resource Allocation Graph



Resource Allocation Graph With A Deadlock



Resource Allocation Graph With A Cycle But No Deadlock



Basic Facts

- If graph contains no cycles ⇒ no deadlock.
- If graph contains a cycle ⇒
  - if only one instance per resource type, then deadlock.
  - if several instances per resource type, possibility of deadlock.

Lecture # 13

Deadlock Solutions

- In general, there are three strategies of dealing with deadlock problem:
    - Deadlock Prevention
    - Deadlock Avoidance
    - Deadlock Detection & Recovery

Deadlock Prevention:

- Deadlock Prevention is a technique that guarantees that a deadlock will not occur.
- Prevention is achieved by assuring that one of the necessary conditions for deadlock is not met.
- **Mutual Exclusion** – not required for sharable resources; must hold for nonsharable resources.
- **Hold and Wait** – must guarantee that whenever a process requests a resource, it does not hold any other resources.
    - Require process to request and be allocated all its resources before it begins execution, or allow process to request resources only when the process has none.
    - Low resource utilization; starvation possible
- **No Preemption** –
    - If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released.
    - Preempted resources are added to the list of resources for which the process is waiting.
    - Process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting.
- **Circular Wait** – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration.

For example, provide a global numbering of all the resources, as shown

1 ≡ Scanner
2 ≡ Printer
3 ≡ Plotter
4 ≡ CD Drive
5 ≡ Speaker

- Now the rule is this: processes can request resources whenever they want to, but all requests must be made in numerical order. A process may request first printer and then a tape drive (order: 2, 4), but it may not request first a plotter and then a printer (order: 3, 2). The problem with this strategy is that it may be impossible to find an ordering that satisfies everyone.

Deadlock prevention: summary

- Mutual exclusion
  - Spool everything
- Hold and wait
  - Request all resources initially
- No preemption
  - Take resources away
- Circular wait
  - Order resources numerically

Lecture # 14

Deadlock Avoidance:

- This method differs from deadlock prevention, which guarantees that deadlock cannot occur by denying one of the necessary conditions of deadlock.
- If the necessary conditions for a deadlock are in place, it is still possible to avoid deadlock by being careful when resources are allocated.
- Requires that the system has some additional *a priori* information available.
- Simplest and most useful model requires that each process declare the *maximum number* of resources of each type that it may need.
- The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition.
- Resource-allocation *state* is defined by the number of available and allocated resources, and the maximum demands of the processes.

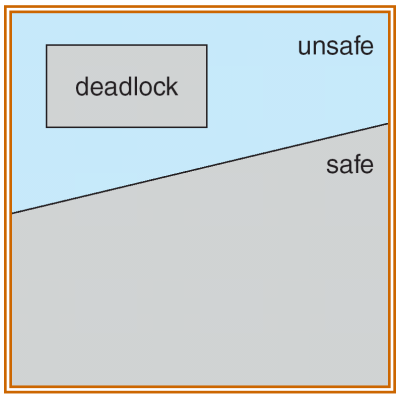
Safe State

- When a process requests an available resource, system must decide if immediate allocation leaves the system in a safe state.
- System is in safe state if there exists a safe sequence of all processes.
- Sequence <P1, P2, …, Pn> is safe if for each Pi, the resources that Pi can still request can be satisfied by currently available resources + resources held by all the Pj, with j<I.
    - If Pi resource needs are not immediately available, then *Pi* can wait until all *Pj* have finished.
    - When *Pj* is finished, *P*i can obtain needed resources, execute, return allocated resources, and terminate.
    - When *Pi* terminates, *Pi*+1 can obtain its needed resources, and so on.

Basic Facts

- If a system is in safe state ⇒ no deadlocks.
- If a system is in unsafe state ⇒ possibility of deadlock.
- Avoidance ⇒ ensure that a system will never enter an unsafe state.

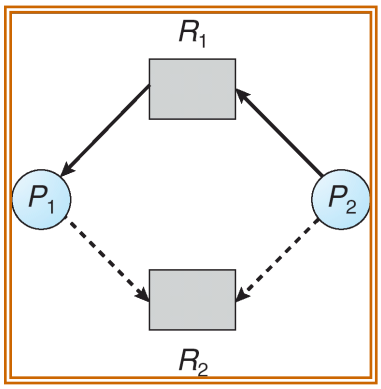
Safe, Unsafe , Deadlock State



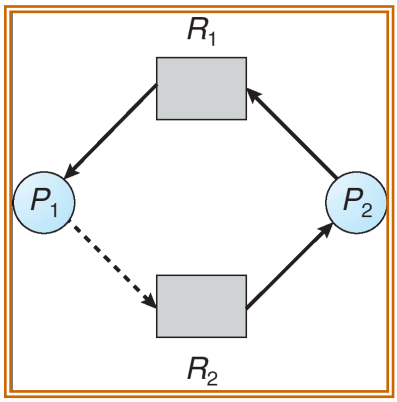
Resource-Allocation Graph Algorithm

- *Claim edge Pi* $\rightarrow$ *Rj* indicated that process *Pj* may request resource *Rj*; represented by a dashed line.
- Claim edge converts to request edge when a process requests a resource.
- When a resource is released by a process, assignment edge reconverts to a claim edge.
- Resources must be claimed *a priori* in the system.

Resource-Allocation Graph For Deadlock Avoidance



Unsafe State In Resource-Allocation Graph



Lecture # 15

Banker's Algorithm

- Perhaps the most famous deadlock avoidance algorithm is the Banker's algorithm. So named because the process is analogous to that used by a banker in deciding if a loan can be safely made.

- In this analogy

  Customers ≡ processes
  Units ≡ resources, say, CD drive
  Banker ≡ Operating System

- Multiple instances.
- Each process must a priori claim maximum use.
- When a process requests a resource it may have to wait.
- When a process gets all its resources it must return them in a finite amount of time.

Data Structures for the Banker's Algorithm

Let *n* = number of processes, and *m* = number of resources types.

- *Available:* Vector of length *m*. If available [*j*] = *k*, there are *k* instances of resource type *Rj* available.
- *Max: n x m* matrix. If *Max* [*i,j*] = *k*, then process *Pi* may request at most *k* instances of resource type *Rj*.
- *Allocation: n* x *m* matrix. If Allocation[*i,j*] = *k* then *Pi* is currently allocated *k* instances of *Rj*.
- *Need: n* x *m* matrix. If *Need*[*i,j*] = *k*, then *Pi* may need *k* more instances of *Rj* to complete its task.

  Need [*i,j*] = *Max*[*i,j*] – *Allocation* [*i,j*].

- **Important Note:** It is important to note that an unsafe state does not imply the existence or even the eventual existence a deadlock. What an unsafe state does imply is simply that some unfortunate sequence of events might lead to a deadlock.

Example of Banker's Algorithm

- 5 processes P_0 through P_4; 3 resource types A (10 instances), B (5instances, and C (7 instances).
- Snapshot at time T_0:

	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

- The content of the matrix. Need is defined to be Max – Allocation.

	Need
	A B C
P_0	7 4 3
P_1	1 2 2
P_2	6 0 0
P_3	0 1 1
P_4	4 3 1

- The system is in a safe state since the sequence < P_1, P_3, P_4, P_2, P_0> satisfies safety criteria.

Let, P_1 request for (1,0,2)

- Check that Request ≤ Available (that is, (1,0,2) ≤ (3,3,2) ⇒ true.

	Allocation	Need	Available
	A B C	A B C	A B C
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 1	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

- Executing safety algorithm shows that sequence <P1, P3, P4, P0, P2> satisfies safety requirement.
- Can request for (3,3,0) by P4 be granted?
- Can request for (0,2,0) by P0 be granted?

Lecture # 16

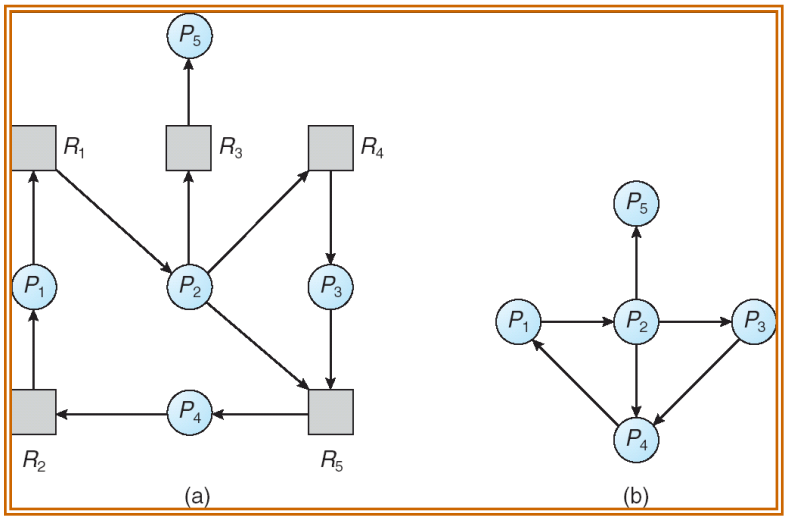
Deadlock Detection & Recovery:

- Allow system to enter deadlock state
- Deadlock Detection is a technique in which requested resources are always granted when available. Periodically the operating system tests for deadlock.
- Deadlock detection is the process of actually determining that a deadlock exists and identifying the processes and resources involved in the deadlock.
- The basic idea is to check allocation against resource availability for all possible allocation sequences to determine if the system is in deadlocked state.

Detecting deadlocks using graphs

- Process holdings and requests in the table and in the graph (they're equivalent)
- Graph contains a cycle => deadlock!
  - Easy to pick out by looking at it (in this case)
  - Need to mechanically detect deadlock

Resource-Allocation Graph and Wait-for Graph



Resource-Allocation Graph **Corresponding wait-for graph**

Deadlock detection algorithm

- General idea: try to find cycles in the resource allocation graph
- Algorithm: depth-first search at each node
  - Mark arcs as they're traversed
  - Build list of visited nodes
  - If node to be added is already on the list, a cycle exists!
- Cycle == deadlock

Resources with multiple instances

- Previous algorithm only works if there's one instance of each resource
- If there are multiple instances of each resource, we need a different method
  - Track current usage and requests for each process
  - To detect deadlock, try to find a scenario where all processes can finish
  - If no such scenario exists, we have deadlock

Several Instances of a Resource Type

- ***Available:*** A vector of length *m* indicates the number of available resources of each type.
- ***Allocation:*** An *n* x *m* matrix defines the number of resources of each type currently allocated to each process.
- ***Request:*** An *n* x *m* matrix indicates the current request of each process. If *Request* [*ij*] = *k*, then process *Pi* is requesting *k* more instances of resource type. *Rj*.

Example of Detection Algorithm

- Five processes P_0 through P_4; three resource types A (7 instances), B (2 instances), and C (6 instances).
- Snapshot at time T_0:

	Allocation	Request	Available
	A B C	A B C	A B C
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 0	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

- Sequence $<P_0, P_2, P_3, P_1, P_4>$ will result in $Finish[i]$ = true for all i.

- P_2 requests an additional instance of type C.

	Request
	A B C
P_0	0 0 0
P_1	2 0 1
P_2	0 0 1
P_3	1 0 0
P_4	0 0 2

- State of system?
  - Can reclaim resources held by process P_0, but insufficient resources to fulfill other processes; requests.
  - Deadlock exists, consisting of processes P_1, P_2, P_3, and P_4.

Recovering from deadlock

- A technique for aborting the deadlock
- **Recovery through preemption**
  - Take a resource from some other process
- **Recovery through rollback**
  - Checkpoint a process periodically
  - Use this saved state to restart the process if it is found deadlocked
- **Recovery through killing processes**
  - kill one of the processes in the deadlock cycle

Review Questions

Q#1: List types of resources we might consider in deadlock problems on computers.

Answer: CPU cycles, memory space, files, I/O devices, CD ROM drives, printers.

Q#2: What is a system resource-allocation graph (RAG) in general?

Answer: A graph which shows the resources and processes, and the relationships among them.

Q#3: What is a safe state?

Answer: A set of resource allocations such that the system can allocate resources to each process (up to its max) in some order, and still avoid a deadlock.

Q#4: List the data structures needed for the banker's algorithm.

Answer:

- available vector *Available*(m)
- demand matrix *Max*(n,m)
- allocation matrix *Allocation*(n,m)
- need matrix *Need*(n,m)

Q#5: Summarize the banker's algorithm.

Answer:

a. If request for process *i* exceeds its need, error has occurred.

b. If request of process *i* exceeds available resources, process *i* must wait.

c. The system temporarily allocates the resources process *i* wants; if the state is unsafe, the allocation is postponed.

Q#6: How can we determine whether current state is "safe" in systems with only one instance of each resource type?

Answer: State is unsafe if any cycle exists.

Q#7: What does a wait-for graph show?

Answer: Shows which process is waiting for other process (es).

Q#8: List three options for breaking an existing deadlock.

Answer:

a. Violate mutual exclusion, risking data.

b. Abort a process.

c. Preempt resources of some process.

Q#9: What is starvation?

Answer: System is not deadlocked, but at least one process is indefinitely postponed.

Problem:

Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix *Need*?

b. Is the system in a safe state?

c. If a request from process P_1 arrives for (0,4,2,0), can the request be granted immediately?

Answer:

a. Since *Need* = *Max* − *Allocation*, the content of *Need* is

A B C D
0 0 0 0
0 7 5 0
1 0 0 2
0 0 2 0
0 6 4 2

b. Yes, the sequence $<P_0, P_2, P_1, P_3, P_4>$ satisfies the safety requirement.

c. Yes. Since

i. $(0,4,2,0) \leq Available = (1,5,2,0)$

ii. $(0,4,2,0) \leq Max_i = (1,7,5,0)$

iii. The new system state after the allocation is made is

	Allocation	*Max*	*Need*	*Available*
P_0	0 0 1 2	0 0 1 2	0 0 0 0	1 1 0 0
P_1	1 4 2 0	1 7 5 0	0 3 3 0	
P_2	1 3 5 4	2 3 5 6	1 0 0 2	
P_3	0 6 3 2	0 6 5 2	0 0 2 0	
P_4	0 0 1 4	0 6 5 6	0 6 4 2	

and the sequence $< P_0, P_2, P_1, P_3, P_4 >$ satisfies the safety requirement.

Fill in the blanks:

(i) A process requests resources; if the resources are not available at that time, the process enters a _____ state. (Wait)

(ii) Under the normal mode of operation, a process may utilize a resource in only the _____ sequence. (Request ----→ Use ---→ Release)

(iii) The request and release of resources are _____. (System Calls)

(iv) A _____ records whether each resource is free or allocated, and, if a resource is allocated, to which process. (System Table)

(v) A set of processes is in a _____ state when every process in the set is waiting for an event that can be caused only by another process in the set.

(Deadlock)

(vi) A deadlock situation can arise if the following four conditions hold simultaneously in the system:

(a) Mutual Exclusion (b) Hold & Wait (c) No preemption (d) Circular wait

(vii) Deadlock can be described more precisely in terms of a directed graph called _____.

(Resource Allocation Graph RAG)

(viii) Methods for handling deadlocks are :

(a) Prevention & Avoidance (b) Detection & Recovery (C) Ignore deadlocks

(ix) To ensure that deadlocks never occur, the system can use a _____ or _____ scheme.

(Deadlock Prevention, Deadlock Avoidance)

(x) _____ is a set of methods for ensuring that at least one of the necessary conditions can not hold. (Deadlock Prevention)

(xi) _____, deadlock handling method, requires that the operating system be given in advance additional information concerning which resources a process will request and use during its lifetime.

(Deadlock Avoidance)

(xii) In deadlock prevention method, the protocols used for not occurring hold & wait condition have two main disadvantages:

(a) Resource utilization may be low (b) Starvation is possible.

(xiii) A _____ algorithm dynamically examines the resources-allocation state to ensure that a circular wait condition can never exist. (Deadlock Avoidance)

(xiv) A state is _____ if the system can allocate resources to each process in some order and still avoid a deadlock. (Safe)

(xvi) Deadlock avoidance algorithm is commonly known as the _____. (Banker's algorithm)

(xvii) A _____ algorithm that examines the state of the system to determine whether a deadlock has occurred. (Deadlock Detection)

(xviii) Deadlock Detection algorithm uses a variant of the resources-allocation graph, called a _____. (Wait-for graph)

(xix) Two options for breaking deadlock (recovery) are:

(a) Process Termination (b) Resources Preemption

(xx) Resources Preemption is a technique for breaking deadlock (Deadlock Recovery), if this technique is required to deal with deadlocks, what are the three issues need to be addressed.

(a) Selecting a victim (b) Rollback (c) Starvation

True / False:

(i) A process must request a resource before using it. And must release the resource after using it. (T)

(ii) The number of resources requested may not exceed the total number of resources available in the system. (T)

(iii) If the graph contains no cycles, then processes in the system are deadlocked. (F)

(iv) If a system does not employ either deadlock prevention or a deadlock avoidance algorithm, then a deadlock situation may occur. (T)

(v) In deadlock prevention method, the mutual-exclusion condition must hold for sharable resources. (F/ non-sharable)

(vi) Read only files are a good example of sharable resources. (T)

(vii) In deadlock prevention method, a process never needs to wait for sharable resources. (T)

(viii) Method for avoiding deadlock is to require additional information about how resources are to be requested. (T)

(ix) A safe state is a deadlock state. (F)

(x) An unsafe state may lead to deadlock. (T)

(xi) A deadlock state is a unsafe state: all unsafe states are deadlocks. (F)

(xii) A deadlock exists in the system if and only if the wait-for graph contains a cycle. (T)

Lecture # 17

Chapter 4 : CPU Scheduling

CPU Scheduling

- Scheduling the processor among all ready processes
- The assignment of physical processor to processes allows processor to accomplish work. The problem of determining when processor should be assigned and to which process, is called processor scheduling or CPU scheduling.
- CPU scheduling is the basis of multiprogrammed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.
- When more than one process is runable, the operating system must decide which one first. The part of the operating system concerned with this decision is called the scheduler, and algorithm it uses is called the scheduling algorithm.

CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- CPU scheduling decisions may take place when a process:
    1. Switches from running to waiting state
    2. Switches from running to ready state
    3. Switches from waiting to ready
    4. Terminates
- Scheduling under 1 and 4 is *nonpreemptive*
- All other scheduling is *preemptive*

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this nvolves:
    - switching context
    - switching to user mode

- jumping to the proper location in the user program to restart that program

Dispatch latency/ witch time

- time it takes for the dispatcher to stop one process and start another running

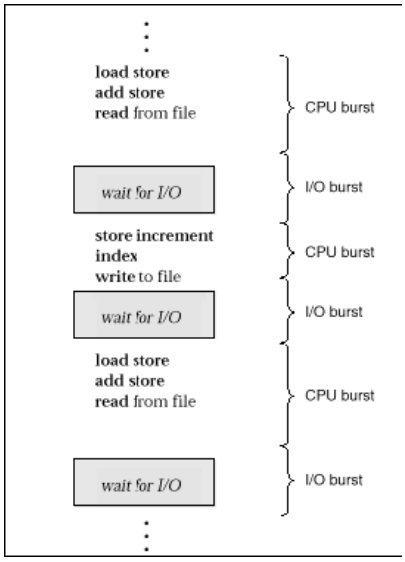
CPU-Scheduling Criteria

- Possible CPU-Scheduling criteria are:
  - **CPU Utilization**: Keep CPU utilization as high as possible.
  - **Throughput**: number of processes completed per unit time.
  - **Turnaround Time**: mean time from submission to completion of process.
  - **Waiting Time**: Amount of time spent ready to run but not running.
  - **Response Time**: Time between submission of requests and first response to the request.

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

CPU/IO burst cycle



- A process will run for a while (the CPU burst), perform some IO (the IO burst), then run for a while more (the next CPU burst) & so on….
- Processes require alternate use of processor and I/O in a repetitive fashion
- Each cycle consist of a CPU burst followed by an I/O burst

  - A process terminates on a CPU burst

  - ✓ **IO Bound processes**: processes that perform lots of IO operations. Each IO operation is followed by a short CPU burst to process the IO, then more IO happens.
  - ✓ **CPU bound processes**: processes that perform lots of computation and do little IO. Tend to have a few long CPU bursts.

Preemptive vs. Non-preemptive scheduling.

- The Scheduling algorithms can be divided into two categories with respect to how they deal with clock interrupts.
- **Nonpreemptive**

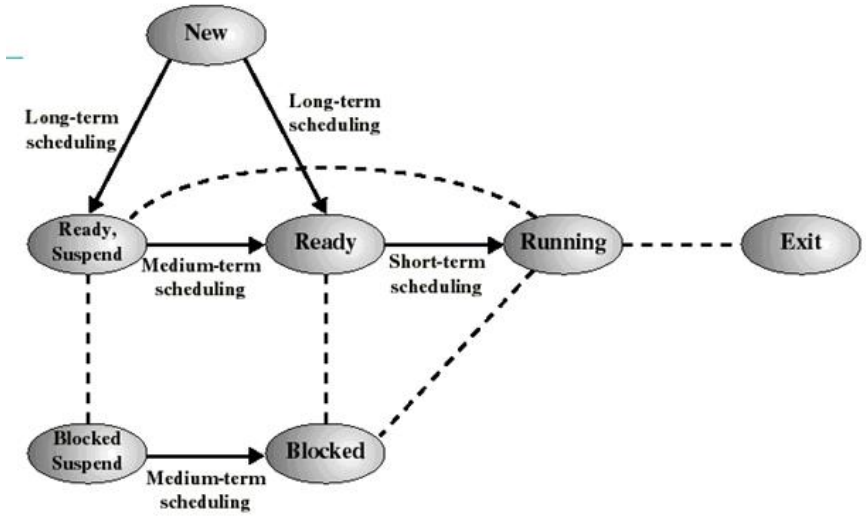
  - ✓ Once a process is in the running state, it will continue until it terminates or blocks for an I/O

- **Preemptive**

  - ✓ Currently running process may be interrupted and moved to the Ready state by the OS

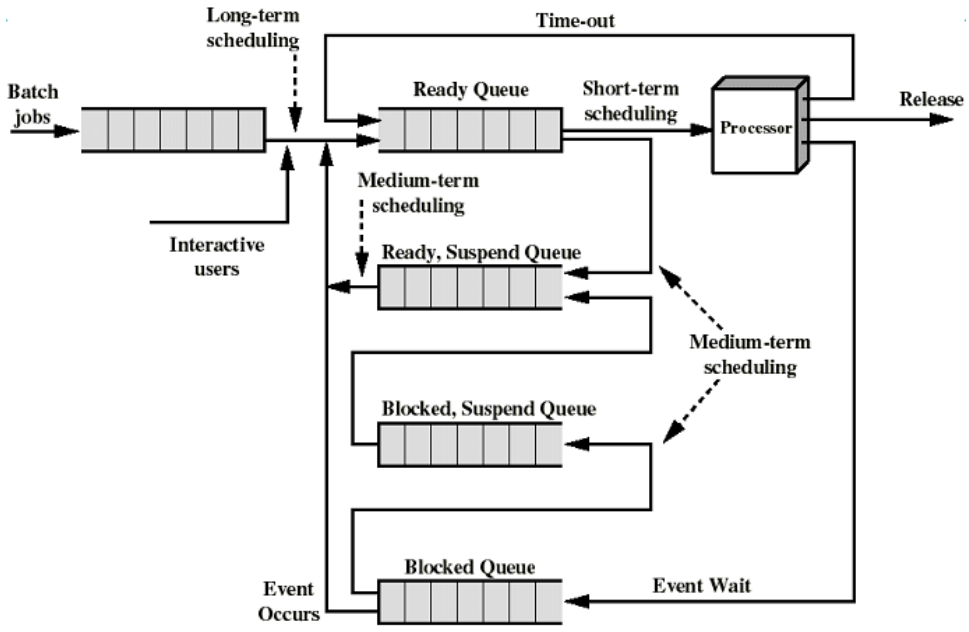
Lecture # 18

Classification of Scheduling Activity



- **Long-term**: which process to admit?
- **Medium-term**: which process to swap in or out?
- **Short-term**: which ready process to execute next?

Queuing Diagram for Scheduling



CPU Scheduling Algorithms:

(i) First Come First Served (FCFS) ----→ Non-preemptive

(ii) Round Robin (RR) ---→ Preemptive

(iii) Shortest Process Next (SPN) or Shortest Job First (SJF) ---→ Either

(vi) Shortest Remaining Time (SRT) ---→ Non-preemptive

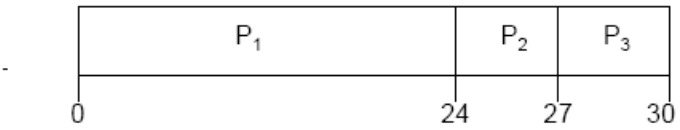
(v) Priority --→ Either

1. First Come First Served (FCFS):

- First-Come-First-Served algorithm is the simplest scheduling algorithm.
- Processes are dispatched according to their arrival time on the ready queue.
- Being a nonpreemptive discipline, once a process has a CPU, it runs to completion.
- Consider performance of FCFS algorithm for three compute-bound processes.

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:



- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

- What about if processes come in order P2, P3, P1? What is
  - Waiting Time? (3 + 3) / 2 = 6
  - Turnaround Time? (3 + 6 + 30) = 13.

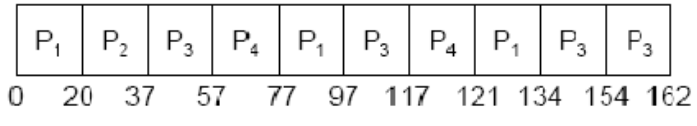
2. **Round Robin (RR):**

- One of the oldest, simplest, fairest and most widely used algorithm is round robin (RR).
- In the round robin scheduling, processes are dispatched in a FIFO manner but are given a limited amount of CPU time called a time-slice or a quantum.
- If a process does not complete before its CPU-time expires, the CPU is preempted and given to the next process waiting in a queue.
- The preempted process is then placed at the back of the ready list.
- Round Robin Scheduling is preemptive.
- The only interesting issue with round robin scheme is the length of the quantum.
- Setting the quantum too short causes too many context switches and lower the CPU efficiency. On the other hand, setting the quantum too long may cause poor response time and appoximates FCFS.

Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better *response*

3. **Shortest Job First / Shortest Process Next (SPN):**

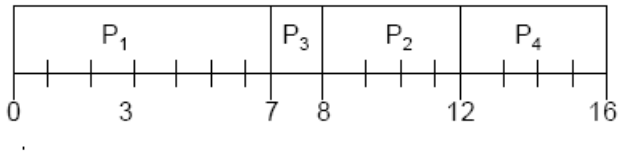
- Shortest-Job-First (SJF) is an either non-preemptive or preemptive discipline in which waiting job (or process) with the smallest estimated run-time-to-completion is run next.
- In other words, when CPU is available, it is assigned to the process that has smallest next CPU burst.

- The SJF scheduling is especially appropriate for batch jobs for which the run times are known in advance.
- Since the SJF scheduling algorithm gives the minimum average time for a given set of processes, it is probably optimal.

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)

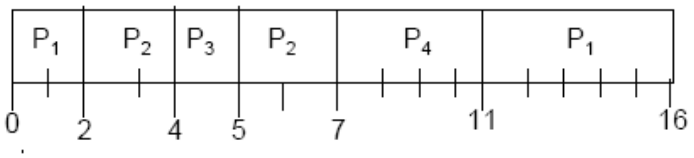


- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

Lecture # 19

4. Shortest Remaining Time (SRT):

- The SRT is the preemtive counterpart of SJF and useful in time-sharing environment.
- In SRT scheduling, the process with the smallest estimated run-time to completion is run next, including new arrivals.

5. Priority Scheduling:

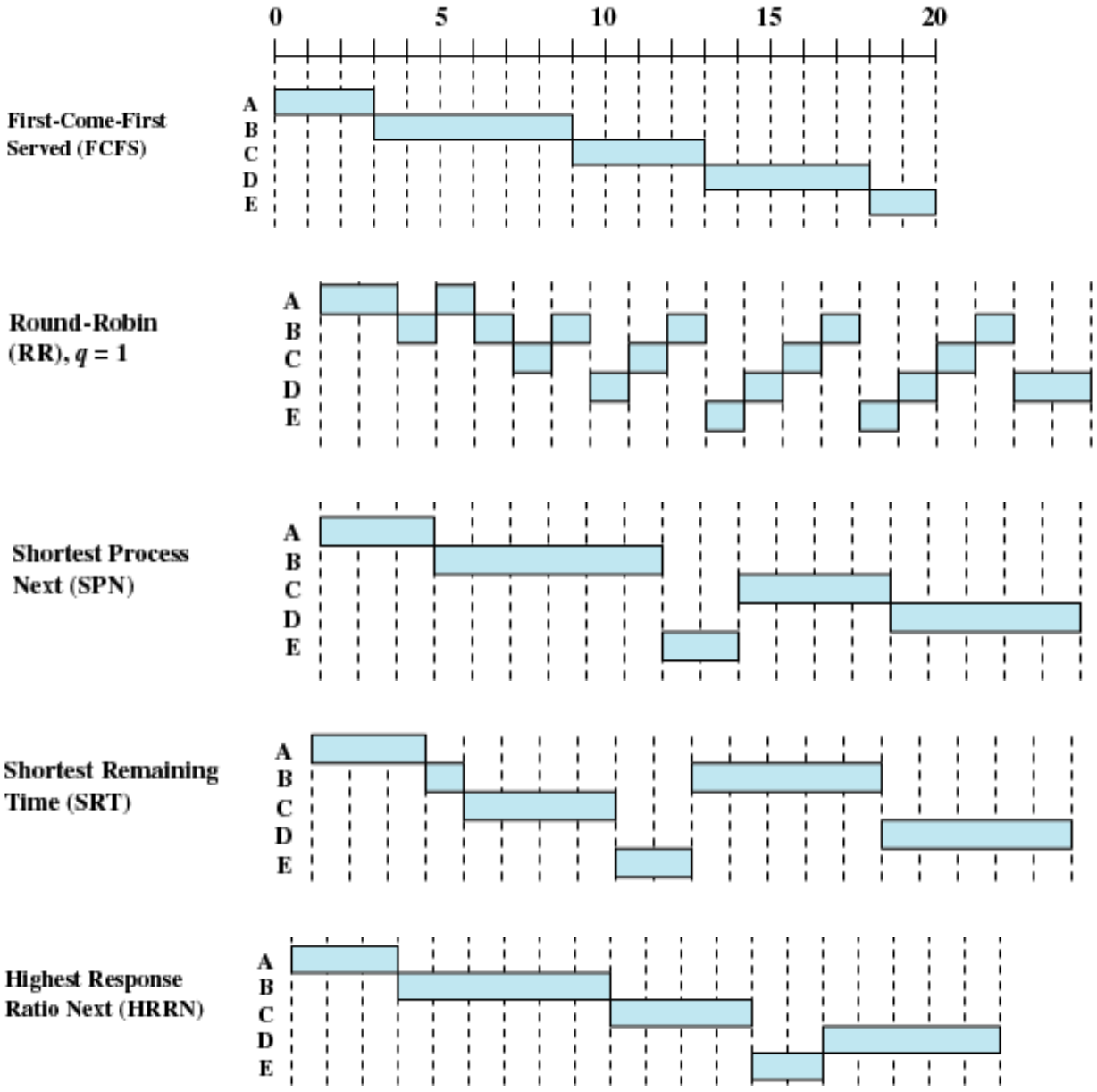
- Each process is assigned a priority, and priority is allowed to run.
- Equal-Priority processes are scheduled in FCFS order.
- The shortest-Job-First (SJF) algorithm is a special case of general priority scheduling algorithm.
- Priority scheduling can be either preemptive or non preemptive
    - A preemptive priority algorithm will preemptive the CPU if the priority of the newly arrival process is higher than the priority of the currently running process.
    - A non-preemptive priority algorithm will simply put the new process at the head of the ready queue.

Highest-Response-Ratio-Next (HRRN) Scheduling:

Response Ratio = (wait time + expected service time) / expected service time

Process Scheduling Example

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2


0
5
10
15
20
First-Come-First Served (FCFS)
Round-Robin (RR), q = 1
Shortest Process Next (SPN)
Shortest Remaining Time (SRT)
Highest Response Ratio Next (HRRN)
A
B
C
D
E

Q#1: Define the difference between preemptive and nonpreemptive scheduling. State why strict nonpreemptive scheduling is unlikely to be used in a computer center.

Answer: Preemptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. Nonpreemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

Q#2: What advantage is there in having different time-quantum sizes on different levels of a multilevel queuing system?

Answer: Processes which need more frequent servicing, for instance interactive processes such as editors, can be in a queue with a small time quantum. Processes with no need for frequent servicing can be in a queue with a larger quantum, requiring fewer context switches to complete the processing, making more efficient use of the computer.

Q#3. What are turnaround time and response time?

Answer: Turnaround time is the interval between the submission of a job and its completion. Response time is the interval between submission of a request, and the first response to that request.

Q#4: Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

Process	**Burst Time**	**Priority**
*P*1	10	3
*P*2	1	1
*P*3	2	3
*P*4	1	4
*P*5	5	2

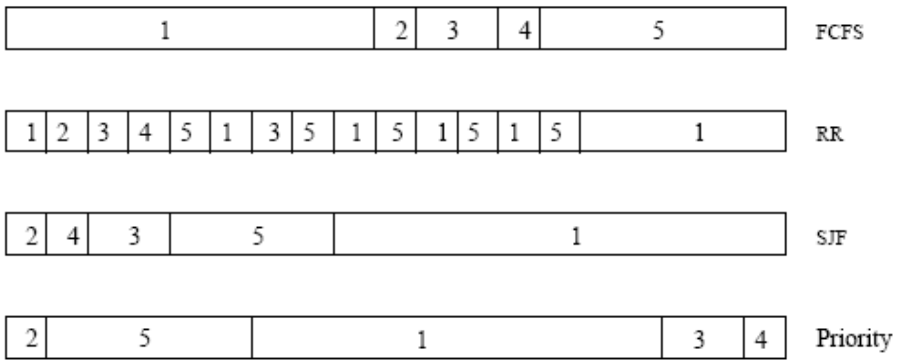
The processes are assumed to have arrived in the order *P*1, *P*2, *P*3, *P*4, *P*5, all at time 0.

a. Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, a nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1) scheduling.
b. What is the turnaround time of each process for each of the scheduling algorithms in part a?
c. What is the waiting time of each process for each of the scheduling algorithms in part a?

d. Which of the schedules in part a results in the minimal average waiting time (over all processes)?

Answer:

a. The four Gantt charts are



b. Turnaround time

	FCFS	RR	SJF	Priority
P_1	10	19	19	16
P_2	11	2	1	1
P_3	13	7	4	18
P_4	14	4	2	19
P_5	19	14	9	6

c. Waiting time (turnaround time minus burst time)

	FCFS	RR	SJF	Priority
P_1	0	9	9	6
P_2	10	1	0	0
P_3	11	5	2	16
P_4	13	3	1	18
P_5	14	9	4	1

d. Shortest Job First

Review Questions

Q#1: What is a CPU burst? An I/O burst?

Answer:

- CPU burst: a time interval when a process uses CPU only.
- I/O burst: a time interval when a process uses I/O devices only.

Q#2: What does "preemptive" mean?

Answer: Cause one process to temporarily halt, in order to run another.

Q#3: How are priority and SJF related?

Answer: Shortest jobs have highest priority.

Q#4: What is indefinite blocking? How can it occur?

Answer: Also called starvation. A process with low priority that never gets a chance to execute. Can occur if CPU is continually busy with higher priority jobs.

Q#5: What is "aging"?

Answer: Gradual increase of priority with age of job, to prevent "starvation."

Fill in the blanks:

(i) The CPU scheduling is the basis of _____ operating systems. (Multiprogrammed)

(ii) The objective of multiprogramming is to have some process running at all times, in order to to maximize _____.
(CPU Utilization)

(iii) Process execution consists of a cycle of CPU execution and _____.
(I/O wait)

(iv) The _____ program might have a few very long CPU Bursts.
(CPU bound)

(v) The selection process is carried out by the _____.
(Short term Scheduler or CPU Scheduler)

(vi) The ____ is the module that gives the control of the CPU to the process selected by the short-term scheduler.
(Dispatcher)

(vii) The time taken by the dispatcher to stop one process and start another running is known as the _____.
(Dispatch Latency)

(viii) The simplest CPU scheduling algorithm is the _____.
(First-come First-served)

(ix) The _____ scheduling algorithm is designed especially for the time sharing system.
(Round-Robin)

(x) A small unit of time in RR scheduling algorithm is called _____.
(Time Quantum or Time slice)

(xi) Both priority and SJF scheduling algorithms may suffer from _____.
(Starvation)

(xii) _____ is a technique to prevent starvation.
(Aging)

True / False:

(i)	Scheduling is a fundamental operating system function.	(T)
(ii)	Almost all computer resources are scheduled before use.	(T)
(iii)	CPU is not one of the primary computer resources.	(F)
(iv)	An I/O bound program would typically have many very short CPU bursts.	(T)
(v)	We want to minimize CPU utilization and throughput.	(F)
(vi)	We want to minimize the turnaround time, waiting time and response time.	(T)
(vii)	The average waiting time under the FCFS policy, however, is often quite short.	(F)
(viii)	FCFS Scheduling algorithm is non-preemptive.	(T)
(ix)	The SJF scheduling algorithm is provably optimal.	(T)
(x)	The SJF scheduling algorithm is a special case of the FCFS algorithm.	(F)

Lecture # 20

Chapter 5 : Memory management

Memory hierarchy

- Different levels of memory
  - Cache: small amount of fast, expensive memory
    - L1 (level 1) cache: usually on the CPU chip
    - L2 & L3 cache: off-chip, made of SRAM
  - Main memory: medium-speed, medium price memory (DRAM)
  - Disk: many gigabytes of slow, cheap, non-volatile storage
- Memory manager handles the memory hierarchy

Memory Management

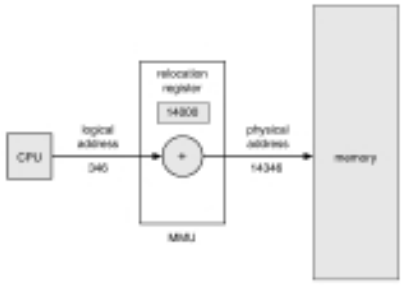
- Memory consists of a large array of words or bytes; each has its own address. The program must be in main memory before execution. The management for the programs to hold in main memory for execution is called memory management.
- **Address Space**: The range of addresses available to a computer program.
- **Base Address:** An address that is used as the origin in the calculation of addresses in the execution of a computer program.

Logical vs. Physical Address Space

- ***Logical address*** – generated by the CPU; also referred to as *virtual address*.
- ***Physical address*** – address seen by the memory unit.
- Logical and physical addresses are the same in compile time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme.
- Address generation
  - Physical address: location in actual memory
  - Logical address: location from the process's point of view
  - Physical address = base + logical address
  - Hardware translates virtual address to *physical address*
- Translation done by the ***M****emory* ***M****anagement* ***U****nit*

Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address; Usually on the same chip as the CPU
- Only physical addresses leave the CPU/MMU chip
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.
- The user program deals with *logical* addresses; it never sees the *real* physical addresses.

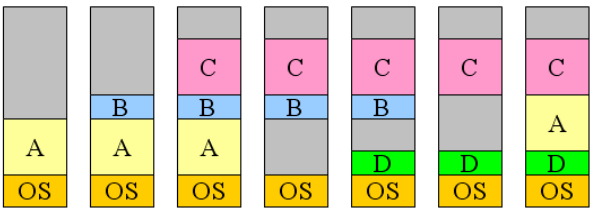


Dynamic Loading

- With dynamic loading, a routine is not loaded until it is called.
- Better memory-space utilization; unused routine is never loaded.
- Useful when large amounts of code are needed to handle infrequently occurring cases.
- No special support from the operating system is required implemented through program design.

Swapping

- A process can be *swapped* temporarily out of memory to a *backing store*, and then brought back into memory for continued execution.
- Backing store – fast disk



- Memory allocation changes as
  - Processes come into memory
  - Processes leave memory
    - Swapped to disk
    - Complete execution
- Gray regions are unused memory

Lecture # 21

Contiguous Memory Allocation:

- Each process is contained in a single contiguous section of memory.
- **Strategies for Dynamic Storage-Allocation:**
  - **First-fit**: Allocate the *first* hole that is big enough.
  - **Best-fit**: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
  - **Next fit:** the first suitable after the previously allocated hole
  - **Worst-fit**: Allocate the *largest* hole; must also search entire list. Produces the largest leftover hole.
- **Example**: Given memory partitions of 100K, 500K, 200K, 300K, and 600K (in order), how would each of the First-fit, Best-fit, and Worst-fit algorithms place processes of 212K, 417K, 112K, and 426K (in order)? Which algorithm makes the most efficient use of memory?

  Answer:

  1. **First-fit:**
     - 212K is put in 500K partition
     - 417K is put in 600K partition
     - 112K is put in 288K partition (new partition 288K = 500K -212K)
     - 426K must wait
  2. **Best-fit:**

- 212K is put in 300K partition
- 417K is put in 500K partition
- 112K is put in 200K partition
- 426K is put in 600K partition

  3. **worst-fit:**
     - 212K is put in 600K partition
     - 417K is put in 500K partition
     - 112K is put in 388K partition
     - 426K must wait

Problems with swapping

- Process must fit into physical memory (impossible to run larger processes)
- Memory becomes fragmented
  - External fragmentation: lots of small free areas
  - Compaction needed to reassemble larger free areas
- Processes are either in memory or on disk: half and half doesn't do any good

Overlays

- It Keeps in memory only those instructions and data that are needed at any given time.
- Implemented by user, no special support needed from operating system, programming design of overlay structure is complex.
- Still doesn't solve the problem of fragmentation or partially resident processes

Fragmentation

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous.
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.
- Reduce external fragmentation by compaction
- Shuffle memory contents to place all free memory together in one large block.
- Compaction is possible *only* if relocation is dynamic, and is done at execution time.

Q. #1: Explain the difference between logical and physical addresses.

Answer:

- Logical addresses are those generated by user programs relative to location 0 in memory.
- Physical addresses are the actual addresses used to fetch and store data in memory.

Q. #2: Explain the difference between internal and external fragmentation.

Answer: Internal fragmentation is the area in a region or a page that is not used by the job occupying that region or page. This space is unavailable for use by the system until that job is finished and the page or region is released.

Q. #3: Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 0,430

b. 1,10

c. 2,500

d. 3,400

e. 4,112

Answer:

a. 219 + 430 = 649

b. 2300 + 10 = 2310

c. illegal reference, trap to operating system

d. 1327 + 400 = 1727

e. illegal reference, trap to operating system

Q. #4: What is the disadvantage of using overlays?

Answer: Programmer must carefully design the program and data structures so that the overlays won't interfere with each other.

Q. #6: What is the advantage of dynamic loading?

Answer: Routines that are never used are never loaded; thus more free memory.

Q. #7: What is segmentation?

Answer: Breaking program up into its logical segments, and allocating space for these segments into memory separately. The segments may be of variable length, and need not be allocated contiguously.

Fill in the blanks:

(xiii) Memory consists of large array of words / bytes, each with its own _____. (address)

(xiv) The _____ maps the virtual address to physical address. (memory Management Unit)

(xv) An address generated by the CPU is commonly referred to as a _____. (Logical Address)

(xvi) Swapping is also called _____. (Roll in, Roll out)

(xvii) Initially, all the main memory is available for user processes, and is considered as one large block of available memory, is called a _____. (Hole)

(xviii) The _____ is a memory allocation strategy that.
(First Fit)

(vii) The memory allocation strategies suffer from _____. (External Fragmentation)

(viii) One solution for the external fragmentation is _____. (Compaction)

(ix) The _____ has one entry for each physical page frame, indicating whether the later frame is free or allocated. (Frame table)

(x) The _____ memory management scheme allocates the variable space. (segmentation)

True / False:

(i) We usually refer to the logical address as a virtual address. (T)

(ii) Dynamic loading requires special support from the operating system. (F)

(iii) The overlays enables a process to be lager than the amount of memory allocated to it. (T)

(iv) The memory fragmentation can be internal as well as external. (T)

(v) The page size in the paging memory management scheme is defined by the operating system. (T)

Lecture # 22

Chapter 6 : Virtual Memory

Virtual Storage / Memory

- Virtual memory is a technique that allows the execution of processes that may not be completely in memory.
- **Basic idea**: allow the OS to hand out more memory than exists on the system
- Keep recently used stuff in physical memory
- Move less recently used stuff to disk
- Keep all of this hidden from processes
- Processes still see an address space from 0 – max address
- Movement of information to and from disk handled by the OS without process help
- Virtual memory (VM) especially helpful in multiprogrammed system
- CPU schedules process B while process A waits for its memory to be retrieved from disk
- This technique frees programmers from the concern of memory storage limitations.
- The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of secondary memory available and not by the actual number of main storage locations.
- Virtual memory is commonly implemented by Demand Paging.

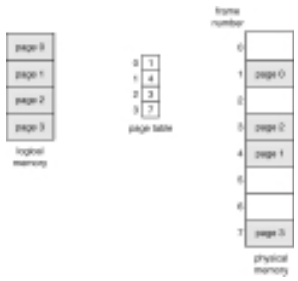
Non-contiguous Memory Management Schemes

- The following are the memory management schemes:
  - Paging
  - Segmentation

Paging

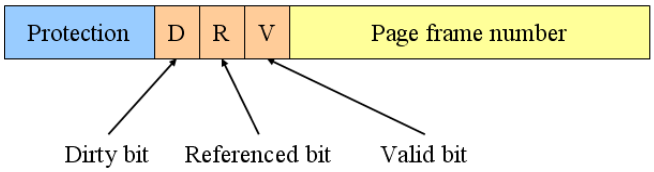
- The Paging is a memory management scheme that permits the physical address space of a process to be non-contiguous.
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8192 bytes).
- Divide logical memory into blocks of same size called **pages**.
- Keep track of all free frames.
- To run a program of size *n* pages, need to find *n* free frames and load program.

- Set up a page table to translate logical to physical addresses.
- Internal fragmentation.
- **Demand Paging**: It is the transfer of a page from secondary storage to main memory at the moment of need.
- **Page Fault**: Occurs when the page containing a referenced word is not in main memory. This causes an interrupt and requires that the proper page be brought into main memory.



Page Table

- Each entry in the page table contains
  - **Valid bit**: set if this logical page number has a corresponding physical frame in memory
    - If not valid, remainder of PTE is irrelevant
  - **Page frame number**: page in physical memory
  - **Referenced bit**: set if data on the page has been accessed
  - **Dirty (modified) bit** :set if data on the page has been modified
  - Protection information



Segmentation

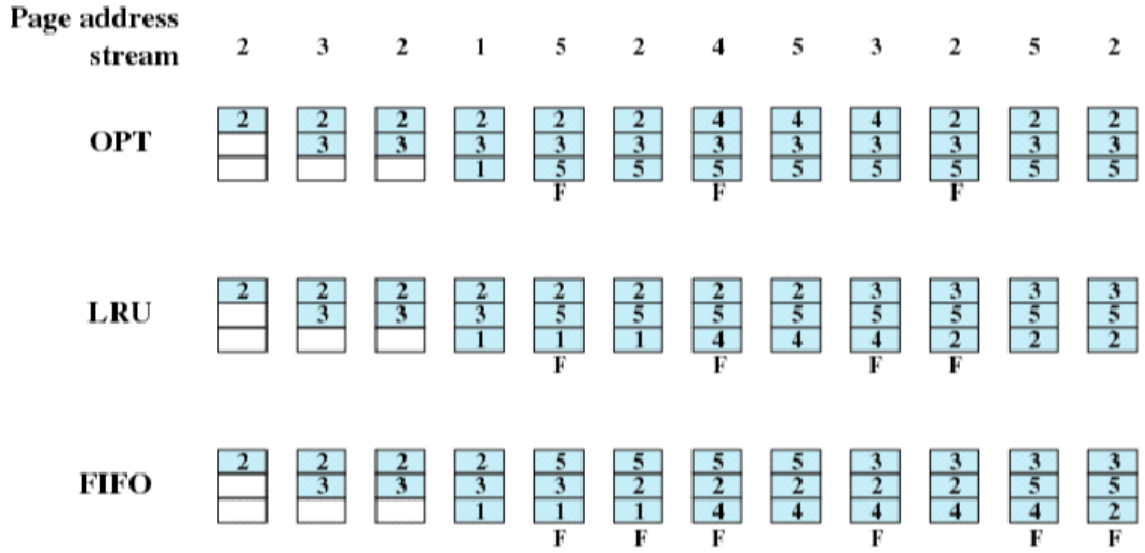
- The division of a program or application into segments as part of a virtual memory scheme.
- **Segment**: In virtual memory, a block that has a virtual address. The blocks of a program may be of unequal length and may even be of dynamically varying length.
- **Compaction**: A technique used when memory is divided into variable-size partitions. From time to time, the operating system shifts the partitions so that they are contiguous and so that all of the free memory is together in one block.

Lecture # 23

Page Replacement Algorithms:

- Page Replacement Algorithms are used to decide what pages to page out (swap out) when a page needs to be allocated. That happens when a page fault occurs and free page cannot be used to satisfy allocation (either because there are none, or because number of free pages is lower than some threshold).
- Page replacement also increases CPU utilization and throughput.
- Page Replacement Algorithms are:

a) **Random**: This algorithm replaces the page in memory by random selection.
b) **Optimal:** An optimal page replacement algorithm has the lowest page-fault. It replace the page that will not be used for the longest period of time.
c) **First in First Out (FIFO**): Algorithm treats the page frames allocated to a process as a circular buffer, and pages are removed in round-robin style.
d) **Least Recently Used (LRU)**: Replaces the page in memory that has not been referenced for the longest time.
e) **Most Frequently Used (MFU)**: Replaces the page in memory that has the smallest counter, in other words the page is replaced that has been accessed the least number of times.



- **Allocation of frames algorithms:** There are two schemes:

    a) Equal Allocation
    b) Proportional Allocation

- **Thrashing:** A phenomenon in virtual memory schemes, in which the processor spends most of its time swapping pages rather than executing instructions.

Q. #1: When do page faults occur? Describe the actions taken by the operating system when a page fault occurs.
Answer: A page fault occurs when an access to a page that has not been brought into main memory takes place. The operating system verifies the memory access, aborting the program if it is invalid. If it is valid a free frame is located and I/O requested to read the needed page into the free frame. Upon completion of I/O, the process table and page table are updated and the instruction is restarted.

Q. #2: Consider the following page reference string:
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.
How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, or seven frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.

- **LRU replacement**
- **FIFO replacement**
- **Optimal replacement**

Answer:

Number of frames	LRU	FIFO	Optimal
1	20	20	20
2	18	18	15
3	15	16	11
4	10	14	8
5	8	10	7
6	7	10	7
7	7	7	7

Q. #3: What is virtual memory?
Answer: A set of techniques and hardware allowing us to execute a program even when not entirely in memory.

Q. #4: What is demand paging?
Answer: Design where a page is never swapped into memory unless needed.

Q. #5: List advantages of demand paging.
Answer: Decreases swap time and the amount of free physical memory, allows higher degree of multiprogramming.

Q. #6: Why is there a valid/invalid bit? Where is it kept?
Answer: To indicate whether an address is invalid, or a page is swapped out. It is kept in the page-frame table.

Q. #7: What is a page fault?
Answer: An interrupt caused by program needing a specific page not yet in memory.

Q. #8: What is page replacement?
Answer: Selecting a frame (preferably not in use) as a victim; swap it out; swap in the desired page into this frame; restarting program.

Q. #9: How many swaps are needed for pure page replacement?
Answer: Two: first one out, second one in.

Q. #10: What does the modify (dirty) bit mean?
Answer: If set, the page has been modified, and must be written back to backing store before being used as a victim.

Q. #11: What is the ideal page-replacement scheme?
Answer: OPTimal, or MINimum page-fault method. Replace the page that will not be used for the longest future time.

Q. #12: What is thrashing?
Answer: State where the system spends an excessive amount of time on paging, compared to the execution of processes.

Q. #13: Why are page sizes always powers of 2?

Answer: Recall that paging is implemented by breaking up an address into a page and offset number. It is most efficient to break the address into X page bits and Y offset bits, rather than perform arithmetic on the address to calculate the page number and offset. Because each bit position represents a power of 2, splitting an address between bits results in a page size that is a power of 2.

Q. #14: Consider a logical address space of eight pages of 1024 words each, mapped onto physical memory of 32 frames.

a. How many bits are there in the logical address?

b. How many bits are there in the physical address?

Answer:

a. Logical address: 13 bits

b. Physical address: 15 bits

Lecture # 24

Chapter 7 : File Systems

File System:

- A set of system software that provides services to users and applications in the use of files, including file access, directory maintenance, and access control.
- In computing, a file system is a method for storing and organizing computer files and the data they contain to make it easy to find and access them.
- File systems may use a storage device such as a hard disk or CD-ROM and involve maintaining the physical location of the files.
- More formally, a file system is a set of abstract data types that are implemented for the storage, hierarchical organization, manipulation, navigation, access, and retrieval of data.
- The file system consists of two distinct parts:
  1. Files → each storing related data.
  2. Directory structure → which organizes and provides information about all files in the system.

Types of file systems:

- File system types can be classified into:

- **Disk file systems** (e.g. FAT, NTFS) : A disk file system is a file system designed for the storage of files on a data storage device most commonly a disk drive, which might be directly or indirectly connected to the computer
- **Network file systems** (e.g. NFS): A network file system (also known as a distributed file system) is a file system where the files are accessed over a network , potentially simultaneously by several computers.
- **Special purpose file systems** (e.g. RTOS): A special purpose file system is basically any file system that is not a disk file system or network file system. This includes systems where the files are arranged dynamically by software , intended for such purposes as communication between computer processes or temporary file space.

Access Method:

- The method that is used to find a file, a record or a set of records.
- The set of rules that defines how a computer puts data onto the network cable and takes data from the cable.
- When data is moving on the network, access methods help to regulate the flow of network traffic.
- Types of access methods:
  - **Direct Access**: The capability to obtain data from a storage device or to enter data into a storage device in a sequence independent at their relative position, by means of addresses that indicate the physical location of the data.
  - **Indexed Access**: Pertaining to the organization and accessing of the records of a storage structure through a separate index to the location of the stored records.
  - **Sequential Access**: The capability to enter data into a storage or a data medium in the same sequence as the data are ordered or to obtain data in the same order as they were entered.
  - **Indexed Sequential Access**: Pertaining to the organization and accessing of the records of a storage structure through an index of the keys that are stored in arbitrarily partitioned sequential files.

Record: A group of data elements treated as a unit.

File: A set of related records treated as a unit. File Attributes are:

- Name
- Type
- Location
- Size
- Protection
- Time, date and user identification

File Allocation Table: A table that indicates the physical location on secondary storage of the space allocated to a file. There is one file allocation table for each file.

Disk Allocation Table: A table that indicates which blocks on the secondary storage are free and available for allocation to files.

Q. #1: What is a file?

Answer: A named collection of related data defined by the creator, recorded on secondary storage.

Q. #2: How can an index file be used to speed up the access in direct-access files?

Answer: Have an index in memory; the index gives the key and the disk location of its corresponding record. Scan the index to find the record you want, and then access it directly.

Q. #3: What is a file path name?

Answer: A list of the directories, subdirectories, and files we must traverse to reach a file from the root directory.

Absolutely Important UNIX Commands

cat f	List contents of file
cat f1 f2 >f3	Concatenates f1(file 1) & f2(file 2) into f3(file 3)
cd	returns you to your home or main directory
cd /	takes you to the root, as far up (to the left) as far as possible
cd	to move down (right in the pathname) a directory
cd ..	moves you up (left in pathname) a directory; likewise,
cd ../../..	moves you up (left in the pathname) 3 directory levels
chmod ###	changes your protections. The order is: you|group|universe (rwxrwxrwx). There will be either a d or - before it. If there's a d, then it's a directory. If there's not, then it's a file. You set the protections in the order rwx (read=1, write=2, execute=4). So, to set the protections for the directory directoryname: you rwx, group r-x, universe r--, you would enter: chmod 751 .
clear	to clear screen
compress	compresses the file filename and puts a .Z extension on it. To uncompress it, type uncompress
cp f1 f2	Copy file f1 into f2
cp -r D1D2	copies the directory D1 and renames it D2

^-c (ctrl-c)	to kill a running process
^-d (ctrl-d)	to close an open window
df	gives disk usage
diff f1 f2	Lists file differences
dig host	domain name, IP address, and alias information for the given host.
dosdir	to do a "dir" (~ls in UNIX) on a DOS floppy in the disk drive
dosread	to read a file from a DOS floppy to your computer account
doswrite	to write a file from your computer account to a DOS floppy
du	lists all subdirectories and their sizes (in blocks?) and total directory size (in blocks?) (takes a long time)
du -a	lists all files and their sizes (in blocks?) in present directory and total directory size (in blocks?) (takes a long time)
du -s	lists overall directory size (in blocks?) (long but clean)
env	shows current environment set-up
find	Searches the named directory and it"s sub-directories for files. Most frequently called like this: `find ./ -name "t*" -print` Which searches the current directory (and all of its sub-directories) for any files that begin with the letter "t" and then prints them out. If you are looking for a specific filename, then replace "t*" with "filename", and "find" will print out all incidences of this file.
finger @.	(e.g., finger johndoe@ksu.edu fingers Johndoe at Kent State University)
ftp	establishes an ftp link with machinename
gzip	produces files with a .gz extension.
gunzip	decompress files created by gzip, compress or pack.
ispell f	Interactively checks the spelling of the file f, giving logical alternatives to the misspelled words. Type "?" to get help. "ispell" can be accessed from the command line, and also through emacs with M-x ispell-buffer.
kill -9 -1	(from a remotely logged-in site) kills all running processes (essentially forces a logout) *not to be used unless nothing else works* kill -9 process-id# - kills a running process
lpq	shows UNIX print queue

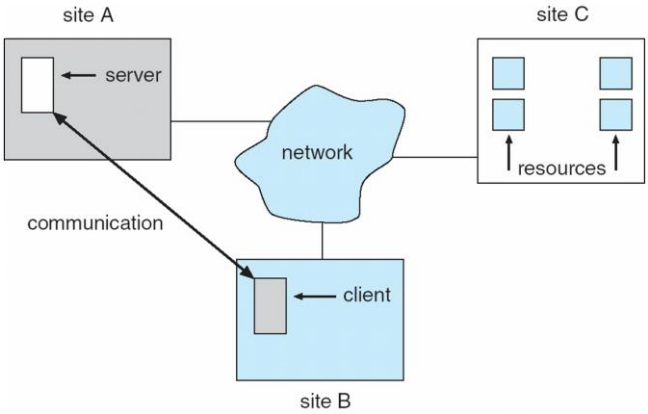
lpr	to print the file
lpqrm job#	removes job from printer queue
ls	shows listing of files in present directory
ls -a	shows listing of all files in present directory
ls -l	shows long listing of files in present directory
ls -la | more	shows long listing of all files in present directory
man command	shows help on a specific command.
mkdir D	creates a new directory called D
more	to view the contents of a file without making changes to it one screen at a time. Hit q to quit more.
mv f1 f2	Rename file f1 as f2
mv f1D	moves the file called f1 to the directory D
nslookup host	domain name, IP address, and alias information for the given host. e.g., nslookup www.kent.edu gives related data for www.kent.edu
passwd	to change your password (takes an hour or so to take effect on all machines)
ping host	to test if the host is up and running.
pwd	present working directory
ps	Shows processes running
ps -flu	Shows detailed description of processes running
pquota	Shows printer quota
quota -v	Shows current disk usage and limits.
rlogin	allows you to remotely log in to another machine on which you have access privileges
rm f	Delete (removes) the file f.
rm -i f	To be prompted for confirmation before you remove a file f, at the UNIX prompt, type
rm dir D	Delete (removes) the empty directory D
rm - r D	removes the directory named D and its contents - use with caution
s f	Alphabetically sort f.
talk	establishes an e-talk session with user@machinename
tar	combines multiple files into one or vice-versa

telnet	allows you to remotely log in to another machine on which you have access privileges
uncompress	uncompresses filename.Z
users	shows who's logged in on the machine
vi	to open the file called filename in the vi text editor
who	Shows who is currently logged on the system.
whoami	shows username of person logged in that window
whois domain_name	lists the domain registration record, e.g., whois kent.edu will produce the domain record for kent.edu
*	wild card character representing any # or characters
date	shows the time and date
date -u	shows greenwich mean time
.	a short cut that stands for the location you are at in a pathway. ex. cp (file (though a pathway) (. (the location you are at)
..	move to parent directory from any comand ex. mv (file name) .. or cd .. etc.
pwd	shows where you are in the pathway
?	wild card character representing one character, can be used in succesion
~	abbreviation for the home file ex. ls ~ lists files in home dir w/o moving there

Lecture # 25

Chapter 8 : Distributed Systems

- **Distributed system** is collection of loosely coupled processors/computers interconnected by a communications network
- Processors variously called *nodes, computers, machines, hosts*
  - *Site* is location of the processor
- Reasons for distributed systems
  - Resource sharing
    - sharing and printing files at remote sites
    - processing information in a distributed database
    - using remote specialized hardware devices
  - Computation speedup – **load sharing**
  - Reliability – detect and recover from site failure, function transfer, reintegrate failed site
- Communication – message passing



Types of Distributed Operating Systems

- Network Operating Systems
- Distributed Operating Systems

Network-Operating Systems

- Users are aware of multiplicity of machines. Access to resources of various machines is done explicitly by:
  - Remote logging into the appropriate remote machine (telnet, ssh)

- Remote Desktop (Microsoft Windows)
- Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism

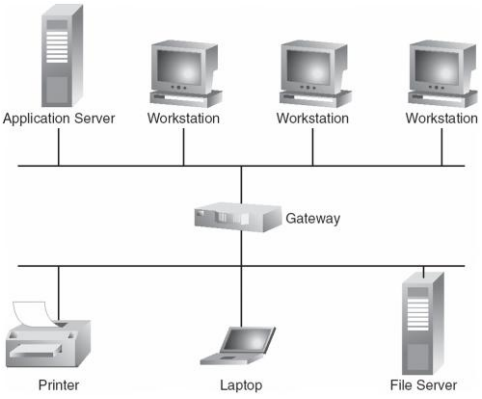
Distributed-Operating Systems

- Users not aware of multiplicity of machines
  - Access to remote resources similar to access to local resources
- Data Migration – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task
- Computation Migration – transfer the computation, rather than the data, across the system
- Process Migration – execute an entire process, or parts of it, at different sites
  - **Load balancing** – distribute processes across network to even the workload
  - **Computation speedup** – subprocesses can run concurrently on different sites
  - **Hardware preference** – process execution may require specialized processor
  - **Software preference** – required software may be available at only a particular site
  - **Data access** – run process remotely, rather than transfer all data locally

Network Structure

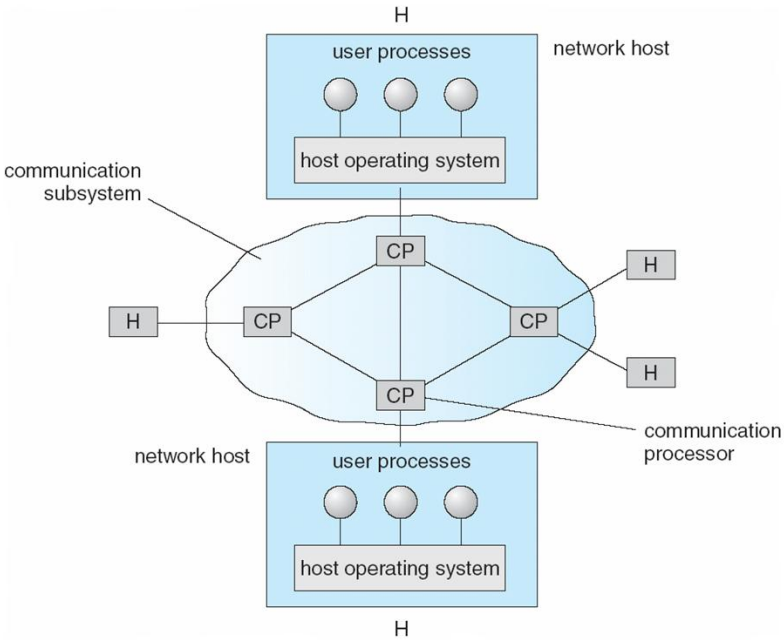
- **Local-Area Network** (**LAN**) – designed to cover small geographical area.
  - Multiaccess bus, ring, or star network
  - Speed ≈ 10 – 100 megabits/second
  - Broadcast is fast and cheap
  - Nodes:
    - usually workstations and/or personal computers
    - a few (usually one or two) mainframes

Depiction of typical LAN



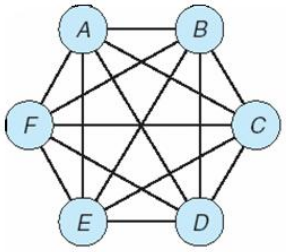
- **Wide-Area Network** (**WAN**) – links geographically separated sites
  - Point-to-point connections over long-haul lines (often leased from a phone company)
  - Speed ≈ 1.544 – 45 megbits/second
  - Broadcast usually requires multiple messages
  - Nodes:
    - usually a high percentage of mainframes

Communication Processors in a Wide-Area Network

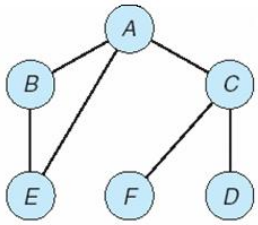


Network Topology

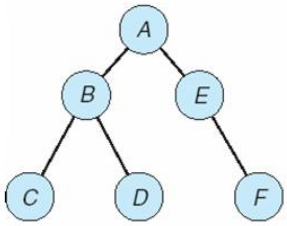
- Sites in the system can be physically connected in a variety of ways; they are compared with respect to the following criteria:
  - **Installation cost -** How expensive is it to link the various sites in the system?
  - **Communication cost -** How long does it take to send a message from site *A* to site *B*?
  - **Reliability -** If a link or a site in the system fails, can the remaining sites still communicate with each other?
- The various topologies are depicted as graphs whose nodes correspond to sites
  - An edge from node *A* to node *B* corresponds to a direct connection between the two sites
- The following six items depict various network topologies



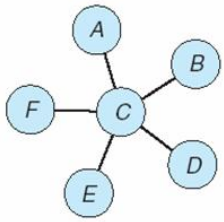
fully connected network



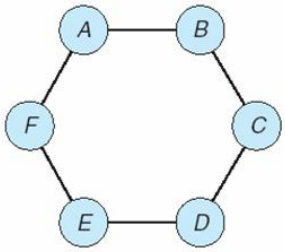
partially connected network



tree-structured network



star network



ring network

Communication Structure

The design of a *communication* network must address four basic issues:

- **Naming and name resolution** - How do two processes locate each other to communicate?
- **Routing strategies** - How are messages sent through the network?
- **Connection strategies** - How do two processes send a sequence of messages?
- **Contention** - The network is a shared resource, so how do we resolve conflicting demands for its use?

Naming and Name Resolution

- Name systems in the network
- Address messages with the process-id
- Identify processes on remote systems by

  <host-name, identifier> pair
- **Domain name service** (**DNS**) – specifies the naming structure of the hosts, as well as name to address resolution (Internet)

Routing Strategies

- **Fixed routing** - A path from *A* to *B* is specified in advance; path changes only if a hardware failure disables it
    - Since the shortest path is usually chosen, communication costs are minimized
    - Fixed routing cannot adapt to load changes
    - Ensures that messages will be delivered in the order in which they were sent
- **Virtual circuit** - A path from *A* to *B* is fixed for the duration of one session. Different sessions involving messages from *A* to *B* may have different paths
    - Partial remedy to adapting to load changes
    - Ensures that messages will be delivered in the order in which they were sent
- **Dynamic routing** - The path used to send a message form site *A* to site *B* is chosen only when a message is sent
    - Usually a site sends a message to another site on the link least used at that particular time
    - Adapts to load changes by avoiding routing messages on heavily used path
    - Messages may arrive out of order
        - This problem can be remedied by appending a sequence number to each message

Connection Strategies

- **Circuit switching** - A permanent physical link is established for the duration of the communication (i.e., telephone system)
- **Message switching** - A temporary link is established for the duration of one message transfer (i.e., post-office mailing system)
- **Packet switching** - Messages of variable length are divided into fixed-length packets which are sent to the destination
    - Each packet may take a different path through the network
    - The packets must be reassembled into messages as they arrive
- Circuit switching requires setup time, but incurs less overhead for shipping each message, and may waste network bandwidth
    - Message and packet switching require less setup time, but incur more overhead per message

Design Issues

- **Transparency** – the distributed system should appear as a conventional, centralized system to the user
- **Fault tolerance** – the distributed system should continue to function in the face of failure
- **Scalability** – as demands increase, the system should easily accept the addition of new resources to accommodate the increased demand
- **Clusters** – a collection of semi-autonomous machines that acts as a single system

Example: Networking

- The transmission of a network packet between hosts on an Ethernet network
- Every host has a unique IP address and a corresponding Ethernet (MAC) address
- Communication requires both addresses
- Domain Name Service (DNS) can be used to acquire IP addresses
- Address Resolution Protocol (ARP) is used to map MAC addresses to IP addresses
- If the hosts are on the same network, ARP can be used
    - If the hosts are on different networks, the sending host will send the packet to a *router* which routes the packet to the destination network

Karachi University Question Papers

Department of Computer Science
University of Karachi
BCS Program (4th Semester)

2205-Intorduction to Operating System
Terminal Examination Date: 20-July-2016
Max. Time: 2 Hours Max. Marks: 48

Seat Number	Name
Enrolment Number	Institute

Instructions:
1) Please return the question paper along with your answer script.
2) Attempt any all questions on the question paper.
3) Clearly mark the correct choice in MCQs. (ONLY ONE CORRECT CHOICE PER MCQ)

PART-A- Objective Type (20 marks) **Time Allowed: 30 min**

Question # 1 Mark T- True or F- False **(5, 0.5 for each)**

1. The hardware allows privileged instructions to be executed only in kernel mode.
2. DMA transfer decrease contention (traffic or load) on the system bus.
3. Time Sharing System was introduced to reduce the time as compare to serial processing.
4. A socket is identified by an IP Address concatenated with a port number.
5. Hardware failure interrupt support multi-programming system.
6. The only way a process can change from the Blocked state to the Ready state is if an I/O operation finishes.
7. Shortest Process Next (SPN) scheduling policy is a preemptive version of Shortest Remaining Time (SRT).
8. MS word run in the kernel mode.
9. CPU fetches the instruction from memory according to the value of program counter
10. The Interval from the submission of a process to the time of completion is called response time

Question # 2 Select the correct answer. **(5, 0.5 for each)**

1. Multitasking is also known as.

   a) Multi-threading b) Multiprocessing
   c) Multitasking d) Multiprogramming

2. Process State in which, process is in virtual memory and awaiting an event. a.

   a) Ready b) Blocked
   c) Ready Suspend d) Blocked suspend

3. In the context of operations system, a PCB is.

   a) An environment contaminant b) A kind of a queue
   c) A machine instruction. d) A data structure containing information about a process.

4. Single User, Multitasking Operating system.

   a) Windows XP b) DOS
   c) UNIX d) Windows Server 2008

5. Which system call returns the process identifier of a terminated child?
   a) wait b) exit c) fork d) get

6. Concurrency control technique that may cause a busy waiting state.
   a) Semaphore b) Monitor
   c) Message Passing d) Peterson's Algorithm

7. An IPC facility provides at-least two operations: (choose two)
   a) write message b) delete message c) send message d) receive message

8. The Zero Capacity queue :
   a) is referred to as a message system with buffering
   b) is referred to as a message system with no buffering
   c) is referred to as a link
   d) None of these

9. To access the services of operating system, the interface is provided by the
   a) system calls b) API c) library d) assembly instructions

10. Process Scheduling policy that may cause Starvation.
    a. Round-Robin b. Shortest Process Next
    c. FIFO d. Option 'b' and 'c'

Question # 3 Fill in the blanks **(10 marks)**

1. Number of processes executed in a unit time is called _____ of the system.
2. A program is in execution is called _____.
3. _____ gives control of the CPU to the process selected by the short-term scheduler?
4. JCL stand for _____
5. In _____ state(s), the process is in secondary memory and awaiting an event.
6. It is possible for two or more programs to be hung up waiting for each other is known as _____
7. _____ is a situation in which two or more process coordinate their activities based on a condition.
8. The two main problem in serial processing are _____, and _____
9. MIMD can be categorized as _____ and _____.
10. _____term is used when process is moved from main memory to virtual memory.

Department of Computer Science
University of Karachi
BCS Program (4th Semester)

2205-Introduction to Operating System
Terminal Examination Date: 20-July-2016
Max. Time: 2 Hours Max. Marks: 48

Seat Number	Name
Enrolment Number	Institute

Instructions:
1) Please return the question paper.
2) Attempt any all questions on the ANSWER COPY.
3) Provide answers in the given space. No extra sheet will be given.

PART-B- Subjective Type (28 marks) **Time Allowed: 1.5 hrs**

Q1: Define the following terms: (Marks: 04 , 0.5 for each)
a. DMA:
b. Cooperating processes:
c. Virtual memory:
d. Starvation:
e. time-sharing system:
f. Memory Address Register:
g. Context switching:
h. long Term Scheduling:

Q2. List down any four reasons of process creation, termination and suspension. (Marks: 2)

Q3. To the point answers:

a: What constitutes the critical mass of process image? (Marks:01)

b: What does it mean by memory protection (Marks:01)

c: Draw the block diagram of 5 states process model. (Marks: 02)

d. List down any two problems of the 1st attempts of Peterson algorithm for mutual exclusion. Also write down algorithm of Peterson good attempt (Marks:03)

e: Briefly discuss O.S History. (Marks:02)

f: Concurrency is the major goal of operating system, but sometimes it will cause major problems, discuss these problems. (Marks:01)

g. Why operating system need to create process image and list down major fields of process image. (Marks:02)

Q4. write short notes on (Marks: 10 , 2.5 for each)
a. Deadlock Conditions
b. Deadlock Detection and recovery
c. Concurrency Control Techniques.
d. Scheduling Algorithms

Department of Computer Science
University of Karachi
BCS Program (6th Semester)

2205-Introduction to Operating Systems
Terminal Examination Date: 29-June-2015
Max. Time: 2½ Hours Max. Marks: 48

Seat Number:	Name:
Enrollment Number:	Institute/College:

Instructions:

- Attempt ALL questions on the question paper.
- Please return the Question Paper along with your answer script.
- Clearly mark the correct choice in MCQs. (ONLY ONE CORRECT CHOICE PER MCQ)

Part A: Max Marks 30
Time Allowed : 40min

Part – A – Objective Type (30 marks) Time Allowed: 60 min

Note: Attempt all questions

Question # 1 Mark only **T or F at left column**. T- True or F- False **(11 Marks , 0.5 for each)**

	Processor context is an example of non-preempt-able resource.
	In Blocked state, process is in secondary memory and awaiting an event.
	Shortest Process Next (SPN) scheduling policy is a preemptive version of Shortest Remaining Time (SRT).
	chmod 333 is used to give full permissions to everyone to execute file.
	In fixed size memory partitioning memory wastage is called External Fragmentation.
	When a process generates I/O request, it will be moved in suspended state.
	Preemptive scheduling algorithm can switch processes on timer interrupt.
	In a system with single memory shared by multiple processors operating system cannot access all processors until it supports SMP.
	In UNIX, while a process running in kernel mode, it may not be preempted.
	Round robin scheduling algorithm may cause starvation.
	The command "**cd ..**" is used to change current directory to the parent directory.
	Device drivers are used to translate user I/O function calls into specific hardware device I/O requests.
	DMA is a specialized processor that is used to transfer data between I/O devices and memory.
	NTFS is less secure than FAT32.
	In kernel level threads, thread switching involves process switching.
	If a semaphore is initialized with 1, then the first call of primitive "wait" will block the calling process.
	In Deadlock Prevention dead lock can be broken by terminating processes one by one.
	In Deadlock Detection technique deadlock can always be prevented by removing any deadlock conditions.
	Windows NT and Windows Vista both are examples of client operating systems.
	In Message Passing, many-to-many relationship between sender and receiver is useful for Client/Server interaction.
	Turnaround Time is the interval of time between the submission of a process and its completion.
	We can take advantage of SMP in user level threads.

Question # 2 Fill in the **blanks in left column (9 marks , 0.1 for each)**

	__waiting state means process take processor time while waiting.
	__command is used to change file current access time in unix/linux.
	__command with __value will be used to give execute only permissions to owner only.
	__is used to make a file or directory visible in multiple parts of the system's file tree.
	__term is used when process is moved from main memory to virtual memory.
	Memory compaction is used to remove __ fragmentation.
	__part of the Process Control Block is used to save/restore processor registers.
	Execution code in UNIX process image is known as process __.
	When a process generates I/O request, it will be moved in __state.
	MBR Stands for __
	_____ is the last instruction executed in Operating System Booting.
	In SMP multiprocessing S stand for ___.
	___ordering is used to prevent circular wait condition
	Unix uses ___ algorithm/approach to handle Deadlock
	SIMD can be categorized as ___ and __ processors
POSIX Stands for	_____.

Question # 3 Write the correct answer **(A,B,C or D only one answer at left column).**
(10 Marks, 0.5 for each)

	Multitasking is also known as. a. Multithreading b. Multiprocessing c. Multiprogressing d. Multiprogramming
	Windows NT run MS-DOS programs by using the subsystem. a. NTVDM b. Win64 subsystem c. POSIX d. OS/2
	Process State in which, process is in secondary memory and awaiting an event. a. Ready b. Blocked c. Ready Suspend d. Blocked suspend
	Process Scheduling policy that may cause Starvation. a. Round-Robin b. Shortest Process Next c. FIFO d. Option 'b' and 'c'
	Which one is a non-preempt-able resource. a. Network b. Memory c. Processor d. Printer
	Which one is the example of Multi-User, Multitasking Operating System. a. Windows XP b. DOS c. UNIX d. Windows-9x
	Which one is a preemptive scheduling algorithm. a. Round-Robin b. Shortest Process Next c. FIFO d. Option 'b' and 'c'
	Command that is used to find current working directory in unix/linux. a. cwd b. pwd c. cd d. pmd
	System Call that is used to generate clone of the calling process in unix/linux. a. system() b. exec() c. fork() d. spawn()
	Which one is the fastest memory? a. Disk Cache b. Main Memory c. Cache d. Registers
	When Intel compatible system boots it start working in ___ addressing mode. a. Real b. Protected c. Virtual d. Fast
	Operating system that is installed in a smart card can be categorized as a. Desktop b. Embedded c. General Purpose d. All of these
	Which partition is used in UNIX/LINUX for virtual memory a. root b. boot c. swap d. /
	Maximum memory that can be addressed in a 32bit system is__. a. 32-GB b. 1-GB c. 1-MB d. 4-GB
	In __memory management technique, processes are partitioned into fixed size of blocks known as__. a.Paging, frames b.Segmentation, frames c.Paging, pages d.Segmentation , pages
	Computing device that is used to connect end user with a mini-computer is known as. a. Microprocessor b. Terminal c. I/O Interface d. Client OS
	Zero sector containing _____ is called boot sector. a. 0xAA55 b. 0xFFFF c. 0x7C00 d. 0xFFF0
	Which one is a non-preemptive scheduling algorithm. a. FIFO b. Shortest Remaining Time c. Round Robin d. Option 'b' and 'c'
	Command that is used to rename a file or directory in unix/linux. a. rm b. mv c. cp d. replace
	Maximum memory that can be addressed in Intel 8086 system is__. a. 32-GB b. 1-GB c. 1-MB d. 4-GB

Department of Computer Science
University of Karachi
BCS Program (6th Semester)

2205-Introduction to Operating Systems
Terminal Examination Date: 29-June-2015
Max. Time: 2½ Hours Max. Marks: 48

Seat Number:	Name:
Enrollment Number:	Institute/College:

Instructions:
- Attempt ALL questions on the question paper.
- Please return the Question Paper along with your answer script.
- Provide Answers in the given space. No extra sheet will be given.

Part B: Max Marks 18
Time Allowed : 1:50hours

Part – B – Objective Type (18 marks) Time Allowed: 120 min

Note: Attempt all questions

Question No 1. (3 marks, 0.5 for each)

a. What is an operating system, what are the main functions of operating system?

b. Define the following systems. **(1.5)**

1) Multithreading.

2) Memory Fragmentation.

3) Time Slicing.

c. What are the main functions of micro-kernel? Is micro-kernel better then large monolithic kernel? Give reason for your answer.

d. Windows NT is structured to support applications written for Windows NT, Windows 9x, DOS and several other operating systems. How Windows NT provides this functionality?

Question No 2. (1 marks)

a. Draw complete 5 State - Process state transition diagram.

Question # 3 Perform **Shortest Process Next** scheduling (A_T=Arrival /S_T=Service time) **(1 Marks)**

Process Information			Clock Time																				
ID	A_T	S_T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P0	0	5																					
P1	1	4																					
P2	2	3																					
P3	3	2																					
P4	4	1																					

Question # 4 Memory Allocation using Buddy System:(2 Marks)
Mark memory cells that will be allocated on request and then released how memory cells combine.

Memory cells unit marking	512k							
	256k				256k			
	128k		128k		128k		128k	
	64k	64k	64k	64k	64k	64k	64k	64k
Example of Allocation and Release								
Request A- 110k	A (110k)		128k		256k			
Release A- 110k	512k							
Now Allocate by dividing in cells & Marking and Release by combining the cells								
Request B-49k								
Request C-120k								
Request D-65k								
Release B-49k								
Release D-65k								

Question # 5. (2 Marks, 0.5 for each)

By using First-Fit (**FF**), Next-Fit (**NF**), Best-Fit (**BF**) and Worst-Fit (**WF**) memory placement algorithm, place the following process in available memory partitions. Gray shaded (underlined numbers) blocks are already allocated.

Process ID	Process Size
P1	3K
P2	16K
P3	1K
P4	9K
P5	32K

Write the process IDs in the given chart to solve your problem. Show if further partitioning required.

	1k	5k	3k	10k	16k	16k	33k	5k	9k	13k	5k	2k	11k	10k	118k
FF	P_3	-	P_1	-	P_2	-	P_4	last allocated block		-		-		-	P_5
NF		-	P_1	-	P_2	-	P_4 P_4			-		-		-	P_5
BF	P_3	-	P_1	-	P_2	-	P_4		P_4	-		-		-	
WF															P_1 P_2 P_3 P_4 P_5

Question No 6. Concurrency Control & Issues (4 Marks)

a. List down 4 necessary conditions for deadlock. (1)
1. _____
2. _____
3. _____
4. _____

b. List down 4 techniques to handle deadlock. (1)
1. _____
2. _____
3. _____
4. _____

c. Provide the code for Producer/Consumer Problem by using Binary Semaphores.(2)

Producer	Consumer

Department of Computer Science **University of Karachi** BCS Program (4th Semester)	**2205--Introduction to Operating Systems** Terminal Examination Date: 19-Dec-2014 Max. Time: 2:30 Hrs Max. Marks: 48
Seat # :	Name:
Enrollment #:	Institute:

Instructions:

- Attempt ALL questions on the question paper.
- Please return the Question Paper along with your answer script.
- Clearly mark the correct choice in MCQs. (ONLY ONE CORRECT CHOICE PER MCQ)

Part A: Max Marks 30
Time Allowed : 60min

Part – A – Objective Type (30 marks) Time Allowed: 60 min

Note: Attempt all questions

Question # 1 Mark only **T or F at left column**. T- True or F- False **(11 Marks , 0.5 for each)**

	In fixed size memory partitioning memory wastage is called Internal Fragmentation.
	When a process generates I/O request, it will be moved in suspended state.
	Non-Preemptive scheduling algorithm can switch processes on timer interrupt.
	In a system with single memory shared by multiple processors operating system cannot access all processors until it supports SMP.
	Processor context is an example of preempt-able resource.
	In Blocked state, process is in secondary memory and awaiting an event.
	Shortest Process Next (SPN) scheduling policy is a preemptive version of Shortest Remaining Time (SRT).
	chmod 777 is used to give full permissions to everyone to execute file.
	In UNIX, while a process running in kernel mode, it may not be preempted.
	Round robin scheduling algorithm may cause starvation.
	The command "**cd .**" is used to change current directory to the parent directory.
	Device drivers are used translate user I/O function calls into specific hardware device I/O requests.
	DMA is a specialized processor that is used to transfer data between I/O devices and memory.
	NTFS is less secure than FAT64.
	Windows NT and Windows Vista both are examples of client operating systems.
	In Message Passing, many-to-many relationship between sender and receiver is useful for Client/Server interaction.
	Turnaround Time is the interval of time between the submission of a process and its completion.
	We can take advantage of SMP in user level threads.
	In kernel level threads, thread switching involves process switching.
	If a semaphore is initialized with 0, then the first call of primitive "wait" will block the calling process.
	In Deadlock Prevention dead lock can be broken by terminating processes one by one.
	In Deadlock Detection technique deadlock can always be prevented by removing any deadlock conditions.

Question # 2 Fill in the **blanks in left column (9 marks , 0.5 for each)**

	Memory compaction is used to remove __ fragmentation.
	__part of the Process Control Block is used to save/restore processor registers.
	Execution code in UNIX process image is known as process __.
	When a process generates I/O request, it will be moved in __state.
	MBR Stands for __
	__waiting state means process take processor time while waiting.
	__command is used to change file current access time in unix/linux.
	__command with __value will be used to give execute only permissions to owner only
	__is used to make a file or directory visible in multiple parts of the system's file tree.
	__term is used when process is moved from main memory to virtual memory.
	_____ is the last instruction executed in Operating System Booting.
	In SMP multiprocessing S stand for ___.
	___ordering is used to prevent circular wait condition
	Unix uses ___ algorithm/approach to handle Deadlock
	SIMD can be categorized as ___ and __ processors
POSIX Stands for	_____

Question # 3 Write the correct answer **(A,B,C or D only one answer at left column).**
(10 Marks, 0.5 for each)

	Multitasking is also known as. a. Multithreading b. Multiprocessing c. Multiprogressing d. Multiprogramming
	Windows NT run MS-DOS programs by using the subsystem. a. NTVDM b. Win64 subsystem c. POSIX d. OS/2
	Process State in which, process is in secondary memory and awaiting an event. a. Ready b. Blocked c. Ready Suspend d. Blocked suspend
	Process Scheduling policy that may cause Starvation. a. Round-Robin b. Shortest Process Next c. FIFO d. Option 'b' and 'c'
	Which one is a non-preempt-able resource. a. Network b. Memory c. Processor d. Printer
	Which one is the example of Multi-User, Multitasking Operating System. a. Windows XP b. DOS c. UNIX d. Windows-9x
	Which one is a preemptive scheduling algorithm. a. Round-Robin b. Shortest Process Next c. FIFO d. Option 'b' and 'c'
	Command that is used to find current working directory in unix/linux. a. cwd b. pwd c. cd d. pwf
	System Call that is used to generate clone of the calling process in unix/linux. a. system() b. exec() c. fork() d. spawn()
	Which one is the fastest memory? a. Disk Cache b. Main Memory c. Cache d. Registers
	When Intel compatible system boots it start working in ___ addressing mode. a. Real b. Protected c. Virtual d. Fast
	Operating system that is installed in a smart card can be categorized as a. Desktop b. Embedded c. General Purpose d. All of these
	Which partition is used in UNIX/LINUX for virtual memory a. root b. boot c. swap d. /
	Maximum memory that can be addressed in a 32bit system is__. a. 32-GB b. 1-GB c. 1-MB d. 4-GB
	In __memory management technique, processes are partitioned into fixed size of blocks known as__. a.Paging, frames b.Segmentation, frames c.Paging, pages d.Segmentation , pages
	Computing devise that is used to connect end user with a mini-computer is known as. a. Microprocessor b. Terminal c. I/O Interface d. Client OS
	Zero sector containing _____ is called boot sector. a. 0xAA55 b. 0xFFFF c. 0x7C00 d. 0xFFF0
	Which one is a non-preemptive scheduling algorithm. a. FIFO b. Shortest Remaining Time c. Round Robin d. Option 'b' and 'c'
	Command that is used to rename a file or directory in unix/linux. a. rm b. mv c. cp d. replace
	Maximum memory that can be addressed in Intel 8086 system is__. a. 32-GB b. 1-GB c. 1-MB d. 4-GB

Department of Computer Science **2205–Introduction to Operating Systems**
University of Karachi Terminal Examination Date: 19-Dec-2014
BCS Program (4th Semester) Max. Time: 2:30 Hrs Max. Marks: 48

Seat # :	Name:
Enrollment #:	Institute:

Instructions:
- Attempt ALL questions on the question paper.
- Please return the Question Paper along with your answer script.
- Provide Answers in the given space. No extra sheet will be given.

Part B: Max Marks 18
Time Allowed : 2:00hours

Part – B – Objective Type (18 marks) Time Allowed: 120 min

Note: Attempt all questions

Question No 1. (3 marks, 0.5 for each)

a. What is an operating system, what are the main functions of operating system?

b. Define the following systems. **(1.5)**

1) Multitasking.

2) Memory Compaction.

3) Timesharing.

c. What are the main functions of micro-kernel? Is micro-kernel batter then large monolithic kernel? Give reason for your answer.

d. Windows NT is structured to support applications written for Windows NT, Windows 9x, DOS and several other operating systems. How Windows NT provides this functionality?

Question No 2. (1 marks)

a. Draw complete 7 State - Process state transition diagram.

Question # 3 Perform **Shortest Process Next** scheduling (A_T=Arrival /S_T=Service time) **(1 Marks)**

Process Information			Clock Time																			
ID	A_T	S_T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P0	0	3																					
P1	1	4																					
P2	2	3																					
P3	3	5																					
P4	4	3																					

Question # 4 Memory Allocation using Buddy System:(2 Marks)
Mark memory cells that will be allocated on request and then released how memory cells combine.

Memory cells unit marking	512k							
	256k				256k			
	128k		128k		128k		128k	
	64k	64k	64k	64k	64k	64k	64k	64k
Example of Allocation and Release								
Request A- 110k	A (110k)		128k		256k			
Release A- 110k	512k							
Now Allocate by dividing in cells & Marking and Release by combining the cells								
Request B-120k								
Request C-240k								
Request D-60k								
Release B-120k								
Release D-60k								

Question # 5. (2 Marks, 0.5 for each)

By using First-Fit (**FF**), Next-Fit (**NF**), Best-Fit (**BF**) and Worst-Fit (WF) memory placement algorithm, place the following process in available memory partitions. Gray shaded (underlined numbers) blocks are already allocated.

Process ID	Process Size
P1	2K
P2	1K
P3	5K
P4	16K
P5	32K

Write the process IDs in the given chart to solve your problem. Show if further partitioning required.

	1k	5k	3k	10k	16k	16k	33k	5k	9k	13k	5k	2k	11k	10k	118k
FF		-		-		-		Last allocated block		-		-		-	
NF		-		-		-				-		-		-	
BF		-		-		-				-		-		-	
WF															

Question No 6. Concurrency Control & Issues (4 Marks)

a. List down 4 necessary conditions for deadlock. (1)

1. _____
2. _____
3. _____
4. _____

b. List down 4 techniques to handle deadlock. (1)

1. _____
2. _____
3. _____
4. _____

c. Provide the code for Producer/Consumer Problem by using Binary Semaphores.(2)

Producer	**Consumer**

Question No 7. Process Scheduling. (5 marks , 2.5 for each)

A scenario is given in the following table. For each determine the **mean process turnaround time**. Ignore process witching overhead. At= Arrival Time, St= Service Time. FIFO (First In First Out)

FIFO

	P1	P2	P3	P4	P5
At	0	2	4	6	8
St	6	3	4	4	5
Turn Around Time	6	7	8	10	13
Mean Turn Around Time	8.8				

FIFO (First In First Out)

Round Robin (Quantum Size 3)

Process No	P1	P2	P3	P4	P5
At	0	2	4	6	8
St	4	6	5	4	3
Turn Around Time	14	14	17	16	11
Mean Turn Around Time	14				

Department of Computer Science
University of Karachi
BCS Program (4th Semester)

2205-Introduction to Operating System Dec-2013
Terminal Examination Date: 08-Jan-2014
Max Time: 03 hrs Max Marks: 48

Seat Number:	Name:
Enrolment Number:	Institute:

Instructions:

1)Do not write on question paper except Name, Seat Number and Enrollment Number.

2) *Please return the question paper along with your answer script.*

Part-A (Objective) 13 Marks (Max. 45 Mins)

Q. Choose the correct answer:

1. A Dead-lock in an Operating System is
   a. Desirable process b. Undesirable process c. definite waiting process d. All of the above
2. Time sharing provides
   a. Disk management b. File system management c. Concurrent execution d. All of the above
3. A static partitioned memory management system has a total of six partitions. If one is allocated to the operating system, this will allow a total of _____
   a. Five user jobs b. Six user jobs c. Thirty-two user jobs d. Thirty-six user jobs
4. Which of the following scheduling policy is well suited for time shared operating system?
   a. Shortest job first b. Round Robin c. FCFS d. Elevator
5. A scheduler which selects processes from secondary storage device is called?
   a. Short term scheduler b. long term scheduler c. Medium term scheduler d. Process scheduler
6. Which of the following is not supported by the operating system?
   a. Protection b. Accounting c. Compilation d. I/O operation
7. In the process management Round-robin method is essentially the pre-emptive version of
   a. FILO b. FIFO c. SSF d. Longest time first
8. What problem is solved by Banker's algorithm?
   a. Mutual exclusion b. Deadlock recovery c. Deadlock avoidance d. Cache coherence
9. What is a shell?
   a. Hardware b. Command interpreter c. Part in compiler d. Tool in CPU scheduling
10. The Hardware mechanism that enables a device to notify the CPU is called
    a. Polling b. Interrupt c. System Call d. None of the them
11. _____ begins at the root and follows a path down to the specified file.
    a. Relative path name b. Absolute path name c. Standalone name d. All of the them
12. Virtual Memory is commonly implemented by _____.
    a. Segmentation b. Swapping c. Demand Paging d. None of them
13. A program at the time of executing is called?
    a. Dynamic program b. Static program c. Binded program d. A process

True /False:

14. System calls allow user-level processes to request services of the operating system.
15. Keep track of which parts of memory are currently being used and by whom are an activity of operating system in regard to process management.
16. When a process creates a new process using the fork() operation, stack state is shared between the parent process and the child process.
17. Kernel threads need not be associated with a process where as every user thread belongs to a process.
18. Rotational time is crucial time while accessing data on the disk.
19. The problem of fragmentation arises in heap allocation.
20. Cryptography technique is used in job scheduling.
21. Inter process communication can be done through system calls.
22. A page fault occurs when the process enters the blocked state.
23. The number of processes completed per unit time is known as throughput.
24. Mutual exclusion is a high level abstraction over Semaphore.
25. A critical region is a region prone to deadlock.
26. Real time systems are used for monitoring events as they occur.

Department of Computer Science
University of Karachi
BCS Program (4th Semester)

2205-Introduction to Operating System
Terminal Examination Date: 08-Jan-2014
Max Time: 03 hrs Max Marks: 48

Seat Number:	Name:
Enrollment Number:	Institute:

Instructions:

*1) Attempt any **FOUR** questions. All questions carry equal mark.*
2) Do not write on question paper except Name, Seat Number and Enrollment Number.
3) *Please return the question paper along with your answer script.*

Part-B (Descriptive) 35 Marks (Max. 135 Mins)

Question No 1

a. What are the conditions necessary for a deadlock situation to occur? Define each condition.
b. How recovery is achieved from deadlock using resource preemption?
c. Consider the following table:

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Answer the following questions using the banker's algorithm:
1) Is the system in a safe state?
2)If a request from process P_1 arrives from (0,4,2,0) can the request be granted immediately?

Question No 2

a. What the two models of interposes communication? What are the strengths and weaknesses of the two approaches?
b. What is the purpose of the command interpreter? Why is it usually separate from the kernel?
c. Describe three general methods for passing parameters to the operating system?

Question No 3

a. Determine the sequence of execution of processes for each of the following scheduling algorithms.
   1. FCFS 2. Round Robin with a quantum size = 5
b. What is the turnaround time for each job, the waiting time for each job, and the average waiting time for all jobs each of the scheduling algorithms?

	Process	Arrival Time	Burst Time
c	1	0	14
d	2	2	12
e	3	5	10
f	4	7	4
g	5	19	7

Question No 4

a. Define the following :
   i) Dynamic loading ii) Dynamic linking iii) overlays
b. What is virtual memory? Why it is needed?
c. How virtual memory can be implemented? Define each technique.

Question No 5

a. What are the responsibilities of operating system in connection with process management?
b. In a multiprogramming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems.
   (i)What are two such problems?
   (ii)Can we ensure the same degree of security in a time shared machine as in dedicated machine? Explain your answer.
c. What are two advantages of multiprocessor system?

Department of Computer Science
University of Karachi
BCS Program (4th Semester)

2205-Introduction to Operating System
Terminal Examination Date: 23-June-2014
Max Time: 03 Hours Max Marks: 48

Seat Number:	Name:
Enrolment Number:	Institute:

Part –A (Objective) 12 Marks (Max. 45 Minutes)

Instructions:

1) Do not write on question paper except Name, Seat Number and Enrollment Number.
2) Return the question paper along with your answer script.

Choose the correct answers (encircle your answer):

1) The run time mapping from virtual (logical) to physical address is done by a hardware device called:
   a. Memory Management Unit b. Translation Lookaside Buffer c. Page Offset
2) Multiprogramming increases _____ by organizing jobs:
   a. Memory Utilization b. CPU Utilization c. H/W Utilization d. Both a & b
3) The ready queue is nothing but a list of _____ implemented as a linked list.
   a. Processor b. Processes c. PCBs d. both b and c
4) CPU switching from one process to another requires saving the state of the current process and loading the latest state of the next process, is known as:
   a. Context Switch b. Packet Switch c. Process Switch d. Processor Switch
5) Module that gives control of the CPU to the process selected by the short-term scheduler is known as:
   a. Dispatcher b. Scheduler c. Manager
6) Number of processes completed per time unit is called:
   a. Turnaround Time b. Response Time c. Throughput d. CPU Utilization
7) The file system NTFS stands for
   a. New Type File System b. Never Terminated File System c. New Technology File System
   d. Non Terminated File System
8) The FCFS algorithm is implemented by using a _____ structure for the ready queue.
   a. FIFO b. FILO c. Stack d. Heap
9) An address generated by the CPU is referred to as a:
   a. Logical Address b. Physical Address c. Variable Address d. Network Address
10) If binding is done at execution time then swap in has to / can be at the _____ as before.
   a. variable address b. different location c. same location d. both b and c
11) IPC stands for:
   a. Internal Processor Conflicts b. Inter Process Communication c. Inter Process Conflicts
12) In IPC, a mailbox is owned by:
   a. Processor b. Process c. Operating System d. both b and c

Mark True / False on the left side of statement number:

1) Shared bus organization usually supports only 2 processors.
2) Processors communicate with each other and the shared memory through the shared bus.
3) The bus and the memory being shared resources there is always a possibility of contention.
4) In case a deadlock occurs the system must detect the deadlock.
5) Memory compaction is a solution to overcome external fragmentation.
6) Including too much functionality in a kernel results in low flexibility at a higher level.
7) A long-term scheduler executes very frequently since new processes are not created at the same pace at which processes need to be executed.
8) The main disadvantage with the Shortest-Job-First Scheduling algorithm lies in knowing the length of the next CPU burst
9) Semaphores is provided by hardware.
10) One or more processes involved in deadlock could be terminated to break the deadlock.
11) RPC can be considered as a special case of a generalized remote message-passing scheme.
12) To make a system work in multiprogramming mode, we should have multiple processors.

Department of Computer Science
University of Karachi
BCS Program (4th Semester)

2205-Introduction to Operating System
Terminal Examination Date: 23-June-2014
Max Time: 03 Hours Max Marks: 48

Seat Number:	Name:
Enrolment Number:	Institute:

Part –B (Descriptive) 36 Marks (Max 2:15 Minutes)

Instructions:

1) Attempt any FOUR questions. All questions carry equal marks.
2) Do not write on question paper except Name, Seat Number and Enrollment Number.
3) Return the question paper along with your answer script.

Question No 1.

a) What are the differences between a process and a thread ? Describe some benefits of using threads ? (4)

b) There are 5 processes: (5)

	A	B	C	D	E	
with arrival times:	0	1	3	9	12	seconds
and their running time is	4	6	2	3	7	seconds

What is the average waiting time using first-come-first-served, shortest-job-first, shortest-remaining-time-first-scheduling and round-robin (with 1 second quantum) scheduling? Which algorithm is optimal in terms of waiting time.

Question No 2.

a) What is a Race Condition and how it can be avoided. (3)
b) What is Critical Section, and what is Critical Section Problem ? (3)
c)What 3 conditions must be satisfied by any solution of critical section problem, discuss each? (3)

Question No 3.

a) Discuss Swapping and discuss in detail the issues related with it. (3)
b) What is a busy waiting condition in semaphore implementation. How this can be avoided. (3)
c) Define Deadlock. (3)

Question No 4.

a) Write a note on Virtual Memory OR Segmentation (5)

b) What is the copy-on-write feature and under what circumstances is it beneficial to use this feature. (4)

OR

A certain computer provides its users with a virtual-memory space of 2^{32} byes. The computer has 2^{18} bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4,096 bytes. A user process generates the virtual address of 11123456 (0001 0001 0001 0010 0011 0100 0101 0110) . Explain how the system establishes the corresponding physical location.

Question No 5.

a) Explain the difference between internal and external fragmentation, by what means they can be avoided?(4)
b) Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory (5)

*****