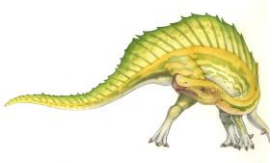
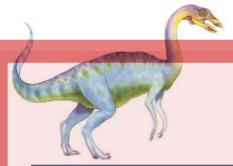


Operating System Definition (Cont.)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.





Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds

- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention (appear).

- Only one interrupt is generated per block, rather than the one interrupt per byte



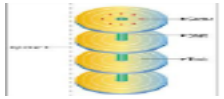


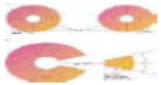
Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile** (loses its content when power is off.)

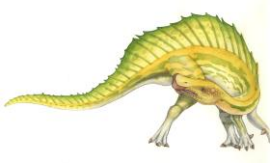


- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity





- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer





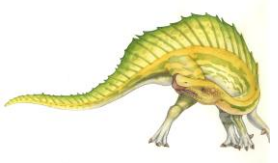
Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility (power off)

كيف تفقد البيانات عند انقطاع الطاقة

- **Caching** – copying information into faster storage system; main memory can be viewed as a *cache* for secondary storage

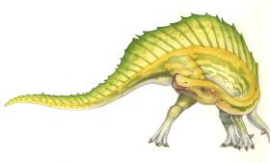
Volatile is used to describe memory content that is lost when the power is interrupted or switched off. Your computer's ordinary memory (or RAM) is volatile memory.





Computer-System Architecture

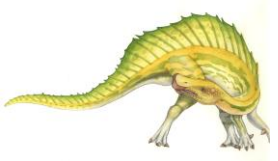
- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput** (more work done in less time)
    2. **Economy of scale** (cost less than single-processor)
    3. **Increased reliability** – **graceful degradation** or **fault tolerance**
  - Two types:
    1. **Asymmetric Multiprocessing** (each PROC→a specific task(mstr))
    2. **Symmetric Multiprocessing** (each PROC → all tasks within OS)





Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a *passive entity*, process is an *active entity*.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- 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 has one program counter per thread
- 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 / threads





Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** (increase) allows user to change to effective ID with more rights



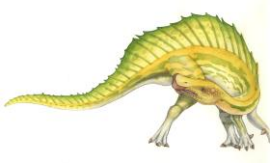


Ch 2

Operating System Services

خدمات اساسية 2/1

- Operating systems provide an environment for execution of programs and services to programs and users
- One set of operating-system services provides functions that are helpful to the user:
  - **User interface** - Almost all operating systems have a user interface (UI).
    - Varies between **Command-Line (CLI)**, **Graphics User Interface (GUI)**, **Batch processing** (the execution of a series of programs on a computer without human interaction.
  - **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device
  - **File-system manipulation** - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

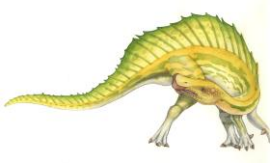




Operating System Services (Cont.)

2/2

- **Communications** – Processes may exchange information, on the same computer or between computers over a network
  - Communications may be via shared memory or through message passing (packets moved by the OS)
- **Error detection** – OS needs to be constantly aware of possible errors
  - May occur in the CPU and memory hardware, in I/O devices, in user program
  - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

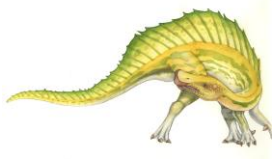




Operating System Services (Cont.)

خدمات إضافية

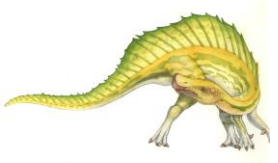
- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - **Resource allocation** - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
  - **Accounting** - To keep track of which users use how much and what kinds of computer resources
  - **Protection and security** - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - **Protection** involves ensuring that all access to system resources is controlled
    - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts
    - If a system is to be protected and secure, precautions must be instituted throughout it. A chain (series) is only as strong as its weakest link.





System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
  - Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in *registers*
    - In some cases, may be more parameters than registers
  - Parameters stored in a *block*, or table, in memory, and address of block passed as a parameter in a register
    - This approach taken by Linux and Solaris
  - Parameters placed, or *pushed*, onto the *stack* by the program and *popped* off the stack by the operating system
  - Block and stack methods do not limit the number or length of parameters being passed



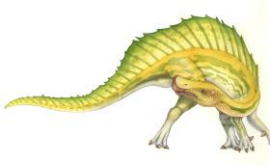


Types of System Calls

معلم معلم

جاء سابقًا خيارات: الشغل الفلاني تبع اي عملية والجواب كان احد اللي هددتهم بأحمر

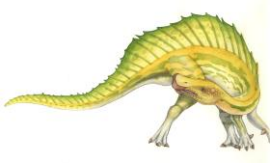
- Process control — Processes
  - end, abort
  - load, execute
  - create process, terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
- File management — files
  - create file, delete file
  - open, close file
  - read, write, reposition — مشتركة مع الديفايس
  - get and set file attributes





Types of System Calls (Cont.)

- Device management devices
  - request device, release device
  - read, write, reposition مشتركة مع الفايلز
  - get device attributes, set device attributes
  - logically attach or detach devices
- Information maintenance data→set, get
  - get time or date, set time or date خيارات
  - get system data, set system data
  - get and set process, file, or device attributes مكس بين اللي قبل
- Communications Communication
  - create, delete communication connection
  - send, receive messages
  - transfer status information information maintenance لاختصوفا لـ
  - attach and detach remote devices device management لاختصوفا لـ





Virtual Machines

- A **virtual machine** takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
- A virtual machine provides an interface *identical* to the underlying bare hardware.
- The operating system **host** creates the illusion that a process has its own processor and (virtual memory).
- Each **guest** provided with a (virtual) copy of underlying computer.

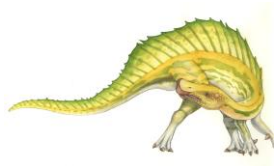
5. An abstract hardware of a single computer into several different execution environments:
   a. Virtual machine.
   b. Live migration
   c. Cloud computing
   d. Network Address Translation (NAT).





Operating-System Debugging

- **Debugging** is finding and fixing errors, or **bugs**
- OSes generate **log files** containing error information
- Failure of an application can generate **core dump** file capturing memory of the process
- Operating system failure can generate **crash dump** file containing kernel memory
- Beyond crashes, performance tuning can optimize system performance
- Kernighan's Law: "**Debugging** is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."
- DTrace tool in Solaris, FreeBSD, Mac OS X allows live instrumentation on production systems
  - **Probes** fire when code is executed, capturing state data and sending it to consumers of those probes

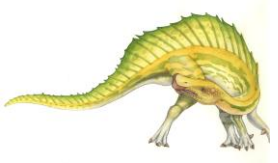




System Boot

- Operating system must be made available to hardware so hardware can start it
  - Small piece of code – **bootstrap loader**, locates the kernel, loads it into memory, and starts it
    - Sometimes two-step process where **boot block** at fixed location loads bootstrap loader
    - When power initialized on system, execution starts at a fixed memory location
      - Firmware used to hold initial boot code

معهم اختبار سابق

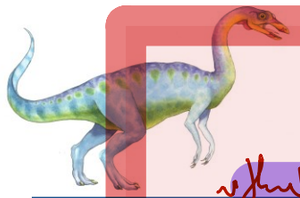




Process Concept

- An operating system executes a variety of programs:
  - Batch system – **jobs**
  - Time-shared systems – **user programs** or **tasks or Active program**
- Textbook uses the terms **job** and ***process*** almost interchangeably
- **Process** – a program in execution; process execution must progress in sequential fashion
- Multiple parts :
  - The **program code**, also called **text section**
  - Current activity including **program counter**, processor registers
  - **Stack** containing temporary data
    - Function parameters, return addresses, local variables
  - **Data section** containing global variables
  - **Heap** containing memory dynamically allocated during run time
- **Program** is ***passive*** entity stored on disk (**executable file**), **process** is ***active***
  - **Program** becomes process when executable file loaded into memory {Multi}
- Execution of program started via GUI mouse clicks, command line entry of its name, etc
- One program can be several processes {how example}
  - Consider multiple users executing the same program

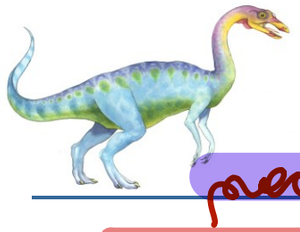




Process State

- As a process executes, it changes **state**
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor {Memory}
  - **terminated**: The process has finished execution (Passive/program}



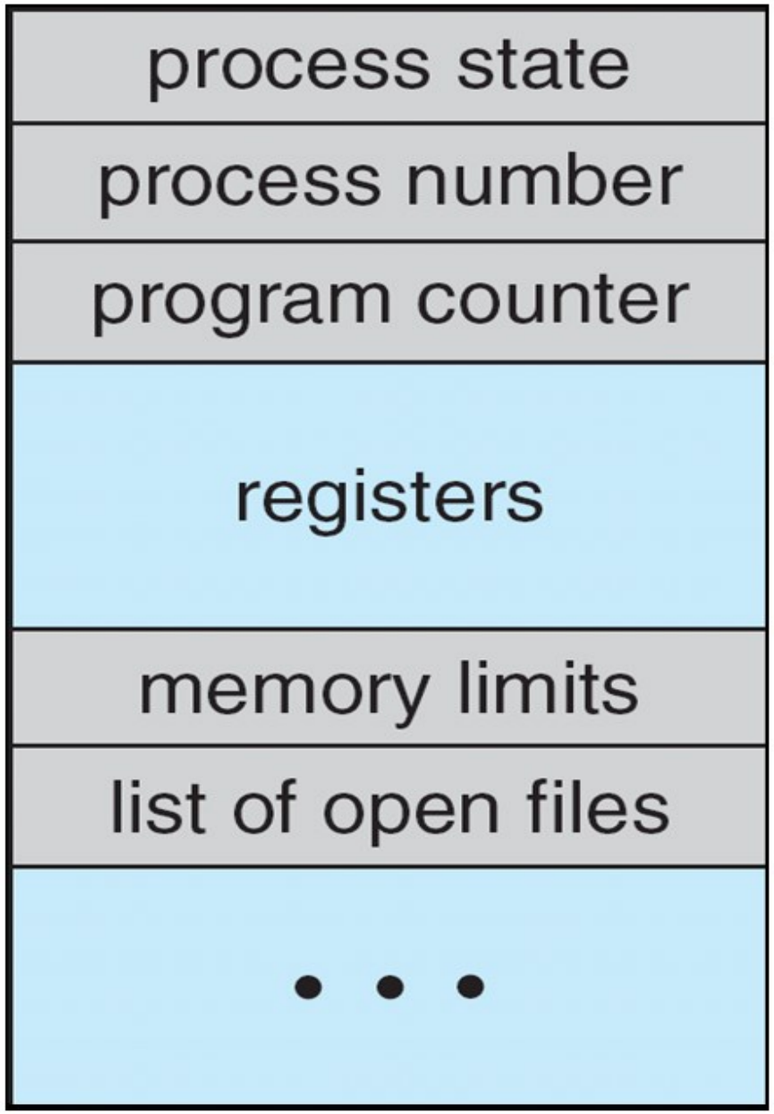


Process Control Block (PCB)

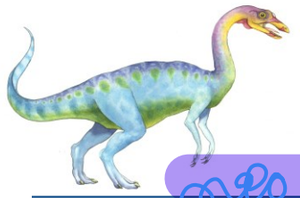
Information associated with each process

(also called **task control block**)

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files







Process Scheduling

- Maximize CPU use, quickly switch processes onto CPU for time sharing
- **Process scheduler (S/W):** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes:
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute // {belong to}
  - **Device queues** – set of processes waiting for an I/O device
  - Processes migrate among the various queues // (State diagram /Process life cycle)





Schedulers

- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
- مهم: فرقان **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
- Short-term scheduler is invoked very frequently (milliseconds) ⇒ (must be fast)
- Long-term scheduler is invoked very infrequently (seconds, minutes) ⇒ (may be slow)
- The long-term scheduler controls the **degree of multiprogramming**
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts



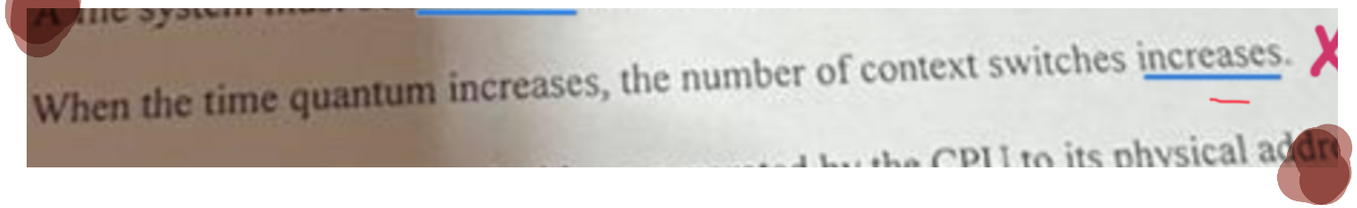






Context Switch

- When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB
- Context-switch time is overhead; the system does no useful work while switching
  - The more complex the OS and the PCB -> longer the context switch
- Time dependent on hardware support







Cooperating Processes

- **Independent** process cannot affect or be affected by the execution of another process
- **Cooperating** process can affect or be affected by the execution of another process

- Advantages of process cooperation
  - Information sharing
  - Computation speed-up
  - Modularity
  - Convenience





Interprocess Communication – Message Passing

- Mechanism for processes to communicate and to synchronize their actions

  - Message system – processes communicate with each other without resorting to shared variables

  - IPC facility provides two operations:
    - **`send`**(*message*) – message size fixed or variable
    - **`receive`**(*message*)

  - If *P* and *Q* wish to communicate, they need to:
    - establish a ***communication link*** between them
    - exchange messages via send/receive

  - Implementation of communication link
    - physical (e.g., shared memory, hardware bus)
    - logical (e.g., direct or indirect, synchronous or asynchronous, automatic or explicit buffering)





Direct Communication

- Processes must name each other explicitly:
  - **`send`** (*P, message*) – send a message to process P
  - **`receive`**(*Q, message*) – receive a message from process Q

- Properties of communication link
  - Links are established automatically
  - A link is associated with exactly one pair of communicating processes
  - Between each pair there exists exactly one link
  - The link may be unidirectional, but is usually bi-directional





Indirect Communication

- Messages are directed and received from mailboxes
  - Each mailbox has a unique id
  - Processes can communicate only if they share a mailbox
- Properties of communication link
  - Link established only if processes share a common mailbox
  - A link may be associated with many processes
  - Each pair of processes may share several communication links
  - Link may be unidirectional or bi-directional





Solution to Critical-Section Problem

1. **Mutual Exclusion** - If process P_i is executing in its critical section, then no other processes can be executing in their critical sections
2. **Progress** - If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely
3. **Bounded Waiting** - A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted

- Two approaches depending on if kernel is preemptive or non-preemptive
  - **Preemptive** – allows preemption of process when running in kernel mode
  - **Non-preemptive** – runs until exits kernel mode, blocks, or voluntarily yields CPU
    - Essentially free of race conditions in kernel mode

- Under non_preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the **process** either by terminating or by switching to the waiting state. This scheduling method was used by Microsoft Windows.





Peterson's Solution

ذكرها باختبار قال T/F هل هي تعتبر حل لـ Critical Section؟ وجاب نفس السؤال بنفس الاختبار كفرافات قال تعتبر حل لأنه؟

- It is a good algorithmic description of solving the problem, Peterson's solution is restricted to two processes.

Critical Section

- Two process solution only, no guarantees that Peterson's solution will work correctly in some architectures.

سلبيات الحل

- Assume that the **`load`** and **`store`** instructions are atomic; that is, cannot be interrupted
- The two processes share two variables:
  - **`int turn;`**
  - **`Boolean flag[2]`**
- The variable **`turn`** indicates whose turn it is to enter the critical section
- The **`flag`** array is used to indicate if a process is ready to enter the critical section. **`flag[i] = true`** implies that process P_i is ready.
- What is the disadvantage of Peterson's solution to the critical section solution?





Semaphore Usage

- **Counting semaphore** – integer value can range over an unrestricted domain
- **Binary semaphore** – integer value can range only between 0 and 1
  - Then a **mutex lock**
- Can implement a counting semaphore ***S*** as a binary semaphore
- Can solve various synchronization problems
- Consider P_1 and P_2 that require S_1 to happen before S_2

```
P1:
    S1;
    signal(synch);
P2:
    wait(synch);
    S2;
```





CPU Scheduler

- **Short-term scheduler** selects from among the processes in ready queue, and allocates the CPU to one of them
  - Queue may be ordered in various ways
- 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**
  - Consider access to shared data
  - Consider preemption while in kernel mode
  - Consider interrupts occurring during crucial OS activities





Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)





Scheduling Algorithm Optimization Criteria

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

نبي نزيد الشغل والانتاجية (max)

وما نضيع الوقت (min)

Question 2, 1x6=6 Marks

Mark the below Statements True or False:

1. One of the goals of scheduling techniques is to maximize turnaround time. ? F [✗F]
2. CPU utilization means keeping the CPU as busy as possible. [T]

السلايد السابق





Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user





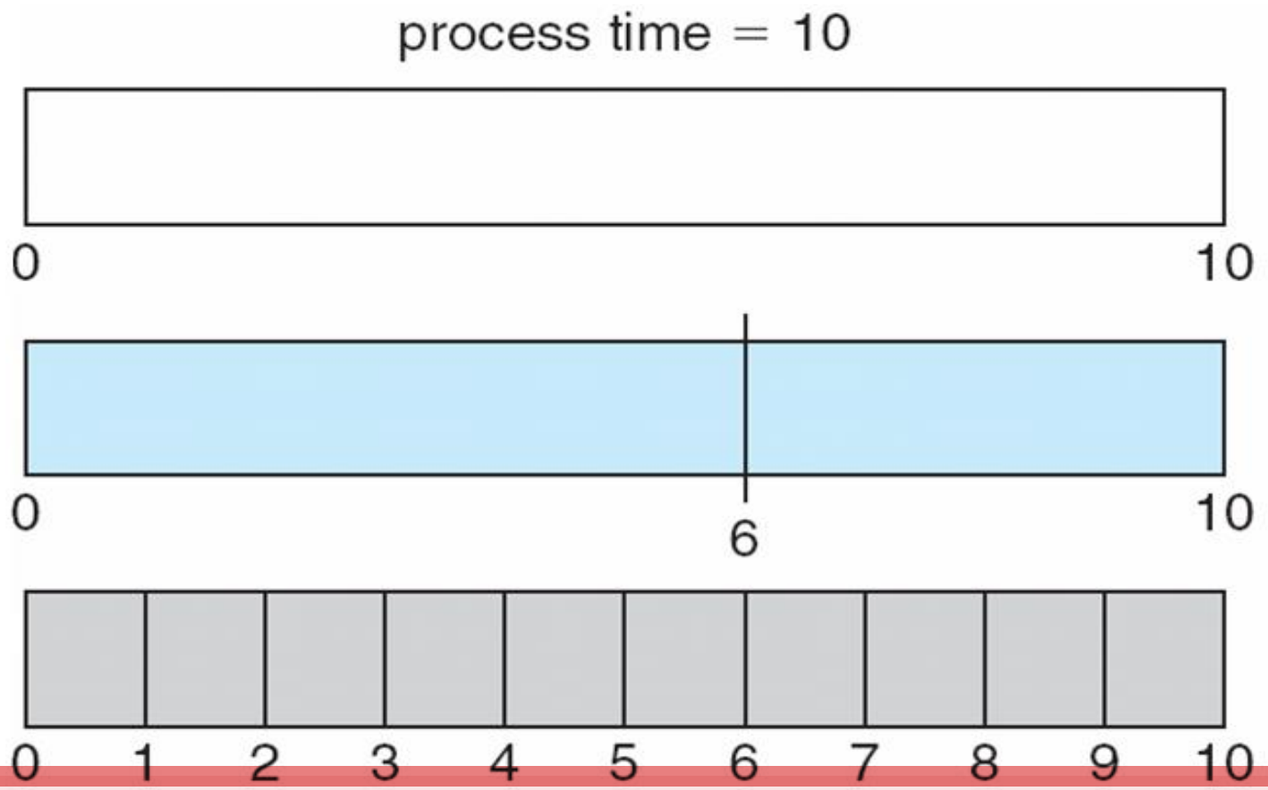
Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - Nonpreemptive

- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Problem ≡ **Starvation** – low priority processes may never execute
- Solution ≡ **Aging** – as time progresses increase the priority of the process



Time Quantum and Context Switch Time



quantum	context switches
12	0
6	1
1	9

When the time quantum increases, the number of context switches increases. ✗

F