# King Saud University
**Department of Computer Science**
**CSC227:Operating Systems**
**Tutorial No. 1.1**

Q 1)

a) What abstracts the hardware? Why? Give an example of such abstraction.

- OS and application programs abstract the hardware of a computer.
- This abstraction helps in controlling and coordinating the use of hardware among various application programs for various users.
    - OS Example: Windows, Linux
    - Applications: Compilers, Editors.

b) What are the two main objectives of Operating System design

- Resource allocation, for example, CPU time, memory space, I/O devices.
- Control program, which is controlling user programs to prevent errors and improper use of computer.

c) What is the difference between:

(i) mono-programming and multi-programming.

- In a monoprogram machine a single job occupies the system from start until end. In such a case CPU sits idle when there is a need of user action.
- A multiprogram machine increases CPU utilization by organizing jobs so that CPU always has something to execute.

(ii) mono-processor and multi-processor.

- In a monoprocessor machine there is only one processor which is responsible for the execution.
- In a multi processor machine there are more than one machine that may share the execution.

(iii) CPU-scheduler and job-scheduler.

- CPU scheduler assigns CPU to a process.
- Job scheduler selects job for execution.

Q 2)

What is the definition of the following terms:

a) Processor

The processor (also called Central Processing Unit, or CPU) is the part of the computer that actually works with the data and runs the programs.

b) Program

A sequence of instructions that a computer can interpret and execute.

c) Process

A process is a running instance of a program, including all variables and other state.

d) Time sharing

A way of sharing out computer facilities between a number of people who want to use the computer at the same time.

# King Saud University
# Department of Computer Science
# CSC227:Operating Systems
# Tutorial No. 1.2

**Q 1.** What are the three main 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.
- Resource allocator. To allocate the 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 user programs to prevent errors and improper use of the computer, and (2) management of the operation and control of I/O devices.

**Q 2.** What is the main advantage of multiprogramming?

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.** What are the main differences between operating systems for mainframe computers and PCs?

The design goals of operating systems for those machines are quite different.
PCs are inexpensive, so wasted resources like CPU cycles are inconsequential. Resources are wasted to improve usability and increase software user interface functionality. Mainframes are the opposite, so resource use is maximized, at the expense of ease of use.

**Q 4.** Define the essential properties of the following types of operating systems:

a) Batch
b) Time sharing
c) Real time
d) Network
e) Distributed
f) Clustered
g) Handheld

a) **<u>Batch</u>**:
Jobs with similar needs are batched together and run through the computer as a group by an operator or automatic job sequencer. Performance is increased by attempting to keep CPU and I/O devices busy at all times through buffering, off-line operation, spooling, and multiprogramming. Batch is good for executing large jobs that need little interaction; it can be submitted and picked up later.

b) **<u>Time sharing</u>**.
Uses CPU scheduling and multiprogramming to provide economical interactive use of a system. The CPU switches rapidly from one user to another. Instead of having a job defined by spooled card images, each program reads its next control card from the terminal, and output is normally printed immediately to the screen.

c) **Real time**.
Often used in a dedicated application. The system reads information from sensors and must respond within a fixed amount of time to ensure correct performance.

d) **Network:**
A network operating system is an operating system that provides features such as file sharing across the network, and that includes a communication scheme that allows different processes on different computers to exchange messages.

e) **Distributed**.
Distributes computation among several physical processors. The processors do not share memory or a clock. Instead, each processor has its own local memory. They communicate with each other through various communication lines, such as a high-speed bus or telephone line.

**f)** **Clustered**
Like parallel systems, clustered systems gather together multiple CPUs to accomplish computational work. Clustered systems differ from parallel systems, however, in that they are composed of two or more individual systems coupled together.

g) **Handheld**
operating systems are designed to provide an environment in which a user can easily interface with the computer to execute programs. many varieties of handheld computers have come into fashion. These devices are mostly standalone, used singly by individual users. Some are connected to networks, either directly by wire or (more often) through wireless modems.

**Q 5.** What network configuration would best suit the following environments?

a) An office:

**Ans:** Local Area Network (LAN) connects network devices over a relatively short distance.

b) A university campus

**Ans:** LAN

c) A state

**Ans:** Metropolitan Area Network (MAN) - a network spanning a physical area larger than a LAN but smaller than a WAN, such as a city. A MAN is typically owned and operated by a single entity such as a government body or large corporation.

d) A nation

**Ans:** WAN - Wide Area Network (WAN) covers a large physical distance. The Internet is the largest WAN.

**Q 6.** What are the tradeoffs inherent in handheld computers?

- Limitation of size in memory. Usually in megabytes compared to gigabytes in computers.
- Processor speed is limited because of less power ability. the operating system has to be designed for low power consumption.
- Lack of physical space limits the I/O capability. The input method are small screen based keyboard or hand-writing recognition while small display screen for output.

**Q 7.** What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?
The main difficulty is keeping the operating system within the fixed time constraints of a real-time system. If the system does not complete a task in a certain time, it could cause a breakdown of the entire system it is running. Therefore when writing an operating system for a real-time system, the writer must be sure that his scheduling schemes don't allow response time to exceed the time constraint.

**Q 8.** Which of the following instructions should be privileged?

a) Set value of timer.
b) Read the clock.
c) Clear memory.
d) Issue a trap instruction.
e) Turn off interrupts.
f) Modify entries in device-status table.
g) Switch from user to kernel mode.
h) Access I/O device.

The following operations need to be privileged: Set value of timer, clear memory, turn off interrupts, modify entries in device-status table, access I/O device. The rest can be performed in user mode.

**Q 9.** What is the difference between mono-programming and multi-programming?

- In a mono-program machine a single job occupies the system from start until end. In such a case CPU sits idle when there is a need of user action.
- A multi-program machine increases CPU utilization by organizing jobs so that CPU always has something to execute.

**Q 10.** What is the difference between CPU-scheduler and job-scheduler?

- CPU scheduler assigns CPU to a process.
- Job scheduler selects job for execution.

# King Saud University
# Department of Computer Science
# CSC227:Operating Systems
# Tutorial No. 6

## Exercise 1

Below is a table of four CPU-bound processes P1, P2, P3, and P4 and their associated arrival times and CPU burst times.

a) Draw Gantt charts illustrating the execution of these processes using SRTF scheduling.
b) What is the average turnaround time
c) What is the average waiting time

| Process | Arrival | Burst Time |
|---|---|---|
| **P1** | 0 | 8 |
| **P2** | 1 | 4 |
| **P3** | 2 | 9 |
| **P4** | 3 | 5 |

## Exercise 2

Below is a table of four CPU-bound processes P1, P2, P3, and P4 and their associated arrival times and CPU burst times.

| Process | Arrival | Burst Time |
|---|---|---|
| **P1** | 0 | 7 |
| **P2** | 2 | 4 |
| **P3** | 4 | 1 |
| **P4** | 5 | 4 |

a) Draw Gantt charts illustrating the execution of these processes using SRTF scheduling.
b) What is the average turnaround time
c) What is the average waiting time

**Exercise 3**

Below is a table of five CPU-bound processes P1, P2, P3, P4, and P5 and their associated arrival times and CPU burst times.

| Process | Arrival | Burst Time |
|---|---|---|
| **P1** | 0 | 10 |
| **P2** | 0 | 29 |
| **P3** | 0 | 3 |
| **P4** | 0 | 7 |
| **P5** | 0 | 12 |

a) Draw Gantt charts illustrating the execution of these processes using RR (quantum=10) scheduling.
b) What is the average turnaround time
c) What is the average waiting time

**King Saud University**
**Department of Computer Science**
**CSC227:Operating Systems**
**Tutorial No. 3**

Q 1) What is a process?
Executable program with its own address space and process control block.

Q 2) How does a child process differ from its parent?
c. It uses separate address space and separate stack pointer from the parent pointer.

Q 3) What is a process control block (PCB)?
Answer:
A process control block is a data structure used by an operating system to manage processes.

Q 4) Is a PCB found all operating systems?
Answer:
Process control blocks are found in all modern operating systems.

Q 5) Is PCB in all operating systems the same?
The structure of PCBs will be different on each operating system though.

Q 6) What does process context switching involves?
Answer:
Changing Process Control Block structures between an old and a new process

Q 7) Context switching operation represents a pure overhead in the OS process management. What does it means?
Answer:
It means that no other useful work can be done during the context switch operation.

Q 8) List at least FIVE other kinds of data that will be stored in a PCB that is not shown in PCB diagram.
Answer:
PCB contains a great deal of information about the process, including:
- The scheduling priority
- Information about the user and groups associated with this process
- Pointer to the parent process PCB
- Pointer to a list of child process PCBs
- Pointers to the process's data and machine code in memory
- Pointers to open files and other resources held by the process

Q 9) Describe the actions a kernel takes to context switch between processes.
Answer:
In general, the operating system must save the state of the currently running process and restore the state of the process scheduled to be run next. Saving the state of a process typically includes the values of all the CPU registers in addition to memory allocation.

Context switches must also perform many architecture-specific operations, including flushing data and instruction caches.

Q 10) A computer has multiple register sets. Describe the actions of a context switch if the new context is already loaded into one of the register sets. What else must happen if the new context is in memory rather than a register set, and all the register sets are in use?

Answer:
The CPU current-register-set pointer is changed to point to the set containing the new context, which takes very little time. If the context is in memory, one of the contexts in a register set must be chosen and moved to memory, and the new context must be loaded from memory into the set. This process takes a little more time than on systems with one set of registers, depending on how a replacement victim is selected.

Q 11) What is the advantage of restricting a child process to a subset of the parent's process?
Answer: the processes will limit within the resource allocated to parent and thus will not exhaust system resources.

Q 12) List the reasons when parent may terminate the execution of its child.
Answer: the child has exceeded its usage of system resources, the task assigned to the child is no longer required, the parent is exiting and the OS does not allow a child to continue if its parent terminates.)

Q 13) How does provision of multiple registers help in context switch?
Answer: if multiple registers are provided then the context switch simply requires changing the pointer to the current register set. Of course, if there are more process than the number of registers available, the OS must copy the register data to and form memory.

Q 14) Mention the three different queues a process may be place in by the OS before it completes execution.
Answer: job, ready, waiting (device)

Q 15) In a context switch, the OS made changes to some fields in the PCB of the current running process. Mention two fields in the PCB that will be updated with new information.
Answer: register, state

Q 16) How many times the message will be printed?

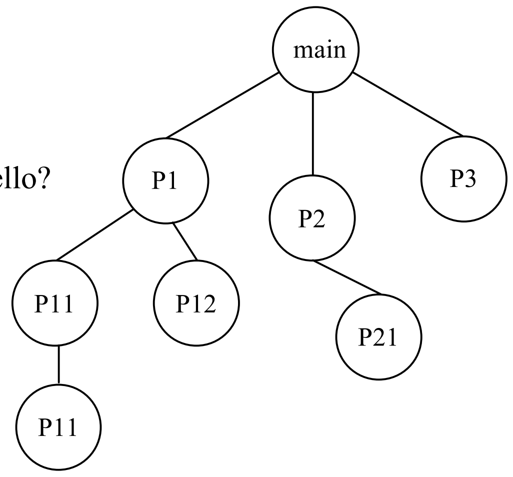
```
main()
{
        fork();
        printf("Hello world");
}
```

Answer.: 2 times. Child is created at fork() and every line after fork will be executed twice once by parent (i.e. printf("Hello world");) and once by child (again printf("Hello world");)

Q 17) Consider the following C language program, what are the possible outputs?

```
#include <stdio.h>
int num;
int main(){
   num = 1;
   fork();
   num = num + 1;
   printf(num);
}
```

Answer: It will print value of num two times one from parent and one from child.



Q 18) How many times does the program below print Hello?

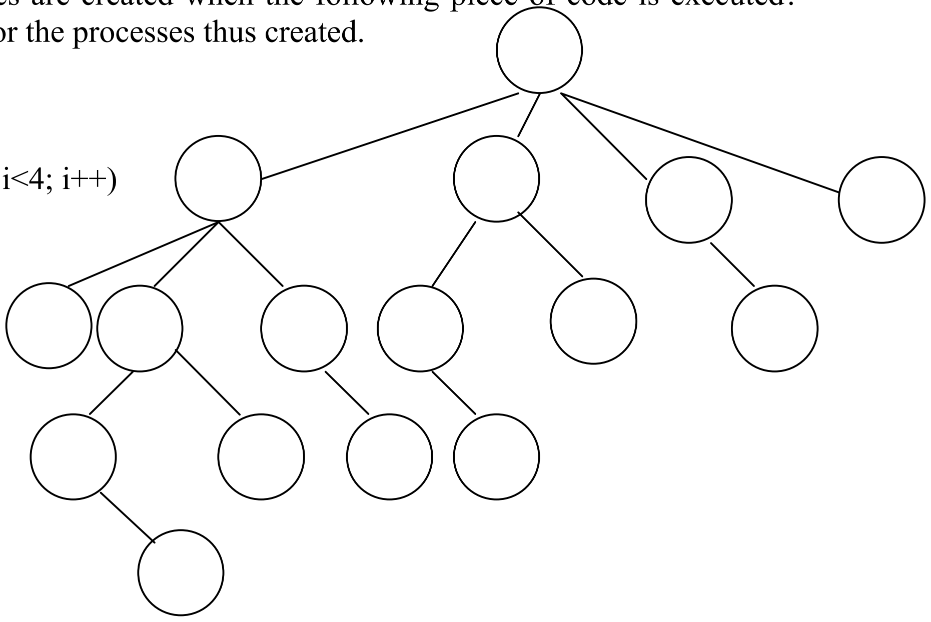
```
include <stdio.h>
int main()
{
     fork();
     fork();
     fork();
     printf( "Hello\n" );
}
```

Answer: 8 times.

Q 19) How many processes are created when the following piece of code is executed? Draw the process tree for the processes thus created.

```
int main() {
     int i;

     for (i=0; i<4; i++)
     fork();
     return 1;
}
```



Answer: 15 + main

The diagram is not accurate please consider the notes from the tutorial lecture

**King Saud University**
**Department of Computer Science**
**CSC227:Operating Systems**
**Tutorial No. 4**

**Ex 1.** 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.
- 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.

**Ex 2.** What resources are used when a thread is created? How do they differ from those used when a process is created?

**Answer:** A context must be created, including a register set storage location for storage during context switching, and a local stack to record the procedure call arguments, return values, and return addresses, and thread-local storage. A process creation results in memory being allocated for program instructions and data, as well as thread-like storage. Code may also be loaded into the allocated memory.

**Ex 3.** Describe the actions taken by a kernel to switch context

a) Among threads.
b) Among processes.

**Answer:**

a) The thread context must be saved (registers and accounting if appropriate), and another thread's context must be loaded.
b) The same as (a), plus the memory context must be stored and that of the next process must be loaded.

**Ex 4.** Describe similarities and differences between thread and process.

In many respect threads operate in the same way as that of processes. Some of the similarities and differences are:

**<u>Similarities</u>**

- Like processes threads share CPU and only one thread active (running) at a time.
- Like processes, threads within a processes execute sequentially.
- Like processes, thread can create children.

- And like process, if one thread is blocked, another thread can run.

**<u>Differences</u>**

- 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.

**Ex 5.** Why threads are used in operating system?

Following are some reasons why we use threads in designing operating systems.

- A process with multiple threads make a great server for example printer server.
- Because threads can share common data, they do not need to use interprocess communication.
- Because of the very nature, threads can take advantage of multiprocessors.
- Threads are cheap in the sense that

a) They only need a stack and storage for registers therefore, threads are cheap to create.

b) Threads use very little resources of an operating system in which they are working. That is, threads do not need new address space, global data, program code or operating system resources.

c) Context switching is fast when working with threads. The reason is that we only have to save and/or restore PC, SP and registers.

d) The biggest drawback is that there is no protection between threads.

**Ex 6.** 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.

# CSC 227 Tutorial 5

**Question 1**

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

| Process | Burst Time | Priority |
|---|---|---|
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, 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)?

**Question 2**

Suppose that the following processes arrive for execution at the times indicated. Each process will run the listed amount of time. In answering the questions, use nonpreemptive scheduling and base all decisions on the information you have at the time the decision must be made.

| Process | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0.0 | 8 |
| P2 | 0.4 | 4 |
| P3 | 1.0 | 1 |

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

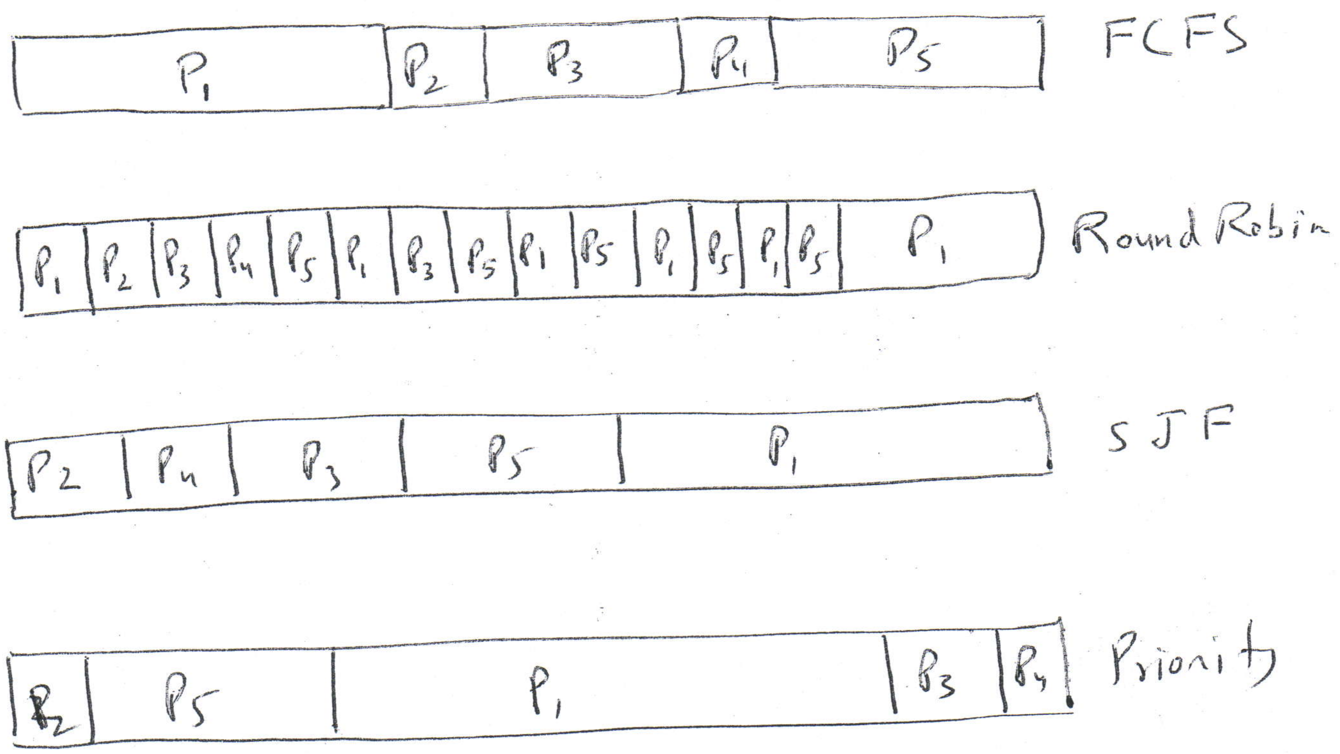
c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes P1 and P2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.

Tutorial 4

[illegible]

## Answer 1

a) The Gantt charts are



$P_1$
$P_2$
$P_3$
$P_4$
$P_5$
FCFS
$P_1$ $P_2$ $P_3$ $P_4$ $P_5$ $P_1$ $P_3$ $P_5$ $P_1$ $P_5$ $P_1$ $P_5$ $P_1$ $P_5$ $P_1$
Round Robin
$P_2$ $P_4$ $P_3$ $P_5$ $P_1$
SJF
$P_2$ $P_5$ $P_1$ $P_3$ $P_4$
Priority

b) Turn around 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 |

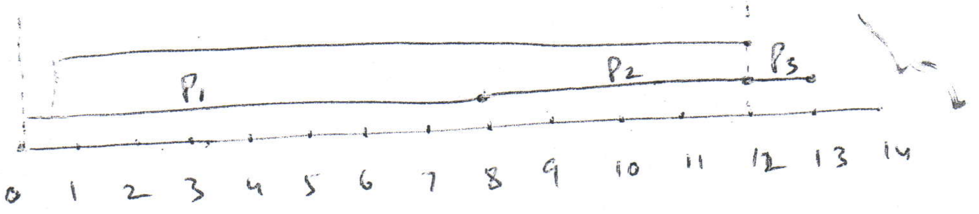
c) Waiting time = |turnaround time − burst time|

|  | 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) SJF 3.2

Answer 2 ! $TT$ = finishing time − arrival time

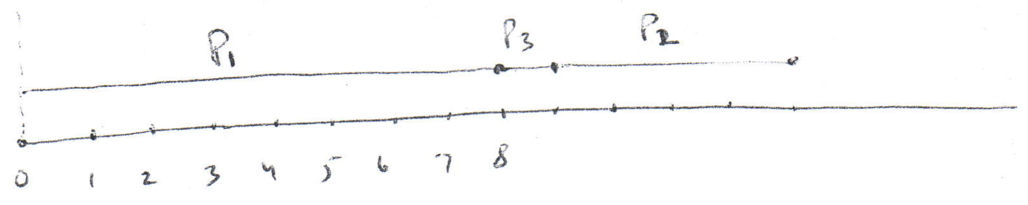
a) Turn around time.



(FCFS)

$TT_1 = 8.0$ , $TT_2 = 11.6$ , $TT_3 = 12$

$TT)_{avg} = 31.6/3 = 10.53$

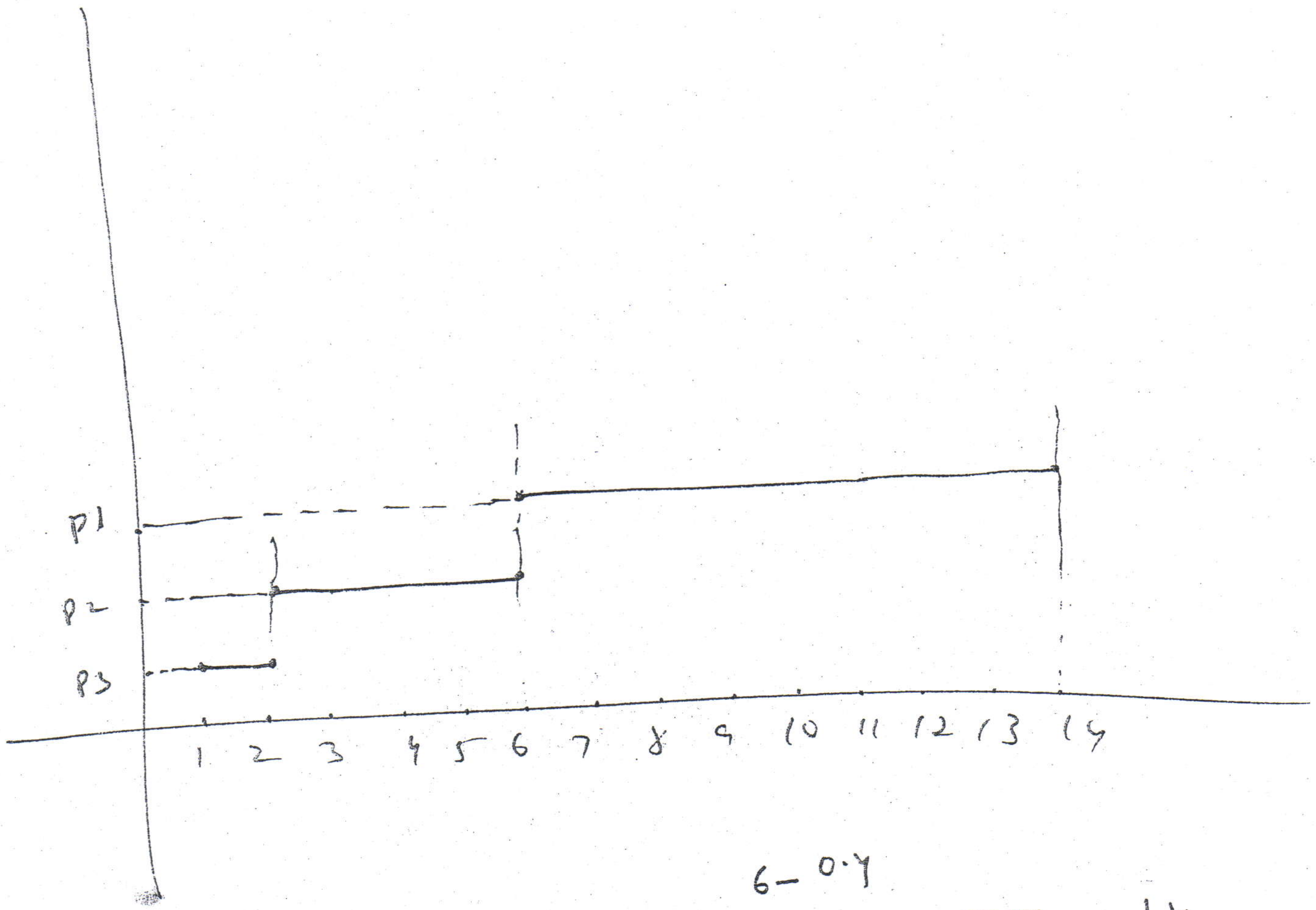


SJF

$TT_1 = 8$ , $TT_2 = 2.6$ , $TT_3 = 8$

$TT)_{avg} = 28.6/3 = 9.53$

2(c)



$TT_3 = 1.0$ $TT_2 = \overset{6-0.4}{5.6}$ $TT_1 = 14$

$$\frac{\begin{array}{r} 14 \\ 5.6 \\ 1 \\ \hline 20.6 \end{array}}{3} = 6.67$$

$WT_3 = 0$ $WT_2 = 1.6$ $WT_1 = 6$

**King Saud University**
**Department of Computer Science**
**CSC227:Operating Systems**
**Tutorial No. 7**

1) **Describe the difference between deadlock and starvation.**

The difference between deadlock and starvation is that with deadlock none of the threads in a set of threads are able to make progress because the events they are waiting for can only be triggered by other threads in the set that are also blocked. With starvation, some threads may make progress while others fail to ever make progress because for example, they are losing in all races for a semaphore. As another example, if a LIFO queue is used for locks, some threads may starve if they happen to end up at the bottom of the queue

2) **Is it possible to have a deadlock involving only one single process? Explain your answer.**

No, because of the hold-and-wait condition. If you are holding resources, you are not waiting for them.

3) **Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock- free.**

Suppose the system is deadlocked. This implies that each process is holding one resource and is waiting for one more. Since there are three processes and four resources, one process must be able to obtain two resources. This process requires no more resources and, therefore it will return its resources when done.

4) **Can a system detect that some of its processes are starving? If you answer yes, explain how it can. If you answer no, explain how the system can deal with the starvation problem.**

No. A process starves if it never gets to run. If a given process has not run for 1 second, has it starved? How about if it has not run for 10 seconds? A minute? An hour? None of these indicate that a process is starving, since it may get to run in the next second. However, as the amount of time that a process has waited gets longer and longer, the probability that it is starving goes up.
This argument depends on the fact that no numeric criteria exist for declaring a process to have starved. If, on the other hand, we declared a process to have starved if it waits 10 seconds without ruining, then we might be able to answer "yes".

5) **What is the meaning of the term busy waiting? What other kinds of waiting are there in an operating system? Can busy waiting be avoided altogether? Explain your answer.**

Busy waiting means that a process is waiting for a condition to be satisfied in a tight loop without relinquish the processor. Alternatively, a process could wait by relinquishing the

processor, and block on a condition and wait to be awakened at some appropriate time in the
future. Busy waiting can be avoided but incurs the overhead associated with putting a process to
sleep and having to wake it up when the appropriate program state is reached.

**King Saud University**
**Department of Computer Science**
**CSC227:Operating Systems**
**Tutorial No. 8**

**Exercise 1)**
Consider a virtual address space of eight pages with 1024 bytes each, mapped onto a physical memory of 32 frames. How many bits are used in the virtual address ? How many bits are used in the physical address ?

Solution:
There are 13 bits in the virtual address.
There are 15 bits in the physical address.

**Exercise 2)**
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?

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.
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.
Worst-Fit:
212K is put in 600K partition.
417K is put in 500K partition.
112K is put in 388K partition.
426K must wait.
In this example, Best-Fit turns out to be the best.

**Exercise 3)**
Consider a paging system with the page table stored in memory.

a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?
b) If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.)

Solution:
A paged memory reference would take 400 nanoseconds; 200 nanoseconds to access the page table and 200 nanoseconds to access the word in memory.

The effective memory access time is:
E.A.T. = 0.75 * (200 nanoseconds) + 0.25 * (400 nanoseconds) = 250 nanoseconds

**Exercise 4)**
Consider a segmentation system where virtual space can have up to $2^{14}$ segments of $2^{18}$ bytes

a) How many bits represent the address field?
b) What is the maximum size of SMT?
c) Assume a program P1 is loaded into memory with the following SMT:

| Segment | Limit | Base |
|---|---|---|
| 0 | 4000 | 1000 |
| 1 | 8000 | 2000 |

Translate the address (1,50) to an absolute address.

d) If the access time to the memory is 200 ns and associative memory access time is 4 ns, what is the effective access time if hit ratio is 80%,

Solution:

a) 32 bits
b) length is $2^{14}$ (number of segments) each filed is 32 bits (4 bytes), then size is $2^{14} * 2^{2} = 2^{16}$ bytes.
c) <#segment, Offset> ➔ <1, 50> ➔ offset is less than limit ➔50 < 8000 (OK) ➔ <base, offset> ➔ <2000 , 50> ➔ 2000 + 50 = 2050.

   One access for table + one access to physical address □ 200 + 200 = 400ns

   (200 + 4) * 0.80 + (400 + 4) * 0.20 = 163.2 + 80.8= 244 ns