1. True or False. [5 marks]

1) In RR scheduling, the time quantum should be small with respect to the context-switch time. _______

Ans: False

2) System programs run in user space. _______

Ans: True

3) The difference between a program and a process is that a program is an active entity while a process is a passive entity.

Ans: False

4) For a single-processor system, there will never be more than one process in the Running state.

Ans: True

5) System calls can be run in either user mode or kernel mode.

Ans: False

2. Multiple-Choice Questions: Choose the answer that best applies. [10 marks]

1) The CPU scheduler is also known as _____________.

C) Short-term scheduler

2) Consider the following C program. How many times hello is printed? _______

```
#include <stdio.h>
#include <unistd.h>
int main()
{
   int pid;
   printf("hello\n");
   pid = fork();
```

```
    if(pid == 0) {
                printf("hello\n");
                exit(1);
   }
    fork();
    printf("hello\n");
    return 0;
}
```

A) 4

3) The ____ scheduling algorithm is designed especially for time-sharing systems.

C) RR

4) What are some other terms for kernel mode? ____

D) All of the above

5) Which of the following scheduling algorithms must be nonpreemptive? _________

C) FCFS

6) A significant problem with priority scheduling algorithms is _____.

B) starvation

7) A ____ provides an API for creating and managing threads.

C) thread library

8) The ____ of a process contains temporary data such as function parameters, return addresses, and local variables.

D) stack

9) System calls are said to be expensive because? ____________

A) they require transition between the user and kernel modes

10) Imagine that a host with IP address 150.55.66.77 wishes to download a file from the web server at IP address 202.28.15.123. Select a valid socket pair for a connection between this pair of hosts. ___________

C) 150.55.66.77:2000 and 202.28.15.123:80

3. Fill in the appropriate word, phrase or value in the space provided [0.5 mark each unless otherwise specified in the question (at the end of the question), 12 marks]

1) process (**1 mark**)

2) CPU utilization, **Throughput**, **Turnaround time**, **Waiting time**, and **Response time**.

3) Shortest-Job-First (or SJF) (**1 mark**)

4) the processor (or CPU), **memory (or RAM)**, **I/O devices**, **File storage**, **network connections**

5) Program counter (or PC)

6) Process Control Block (or PCB) (**1 mark**)

7) new, **running**, **waiting (or blocking)**, **ready**, **terminated**.

8) shared memory and **message passing**.

Short Answer Questions

4. What is Zombie process? (**2 marks**)

Ans: A process that has completed execution (or is terminated) [**1 mark**] but still has an entry in the process table (or the exit status has not been read or collected by its parent) [**1 mark**]

5. What is Convoy effect in FCFS CPU scheduling? (**2 marks**)

Ans: In FCFS CPU scheduling, if short process arrives behind long process [**1 mark**], it results in long waiting time [**1 mark**]

6. What happens during a context switch? (**3 marks**)

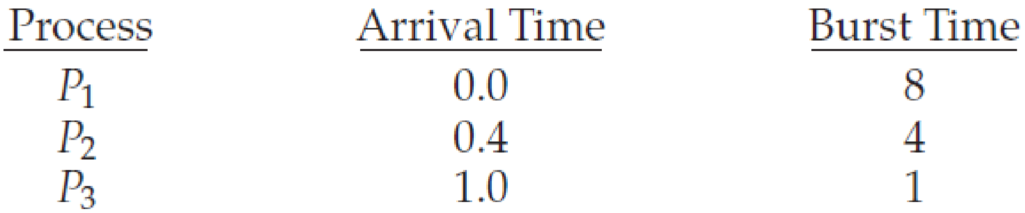
Ans: The state of the currently running process is saved (**1 mark**) into a PCB, and the next process is started by retrieving or loading (**1 mark**) information from its PCB
Saved into or loading from **PCB** (**1 mark**)

7. In Unix systems, what system calls have to be executed by a command interpreter or shell in order to start a new process? What do these system calls actually do? (**5 marks**)

Ans: In Unix systems, a fork (**1 mark**) system call followed (**1 mark**) by an exec (**1 mark**) system call need to be performed to start a new process. <u>Note that you need to mention the order of these two system calls, which worth 1 mark.</u>
The fork call clones the currently executing process (**1 mark**), while the exec call replaces the calling process or the process that calls it (**1 mark**).

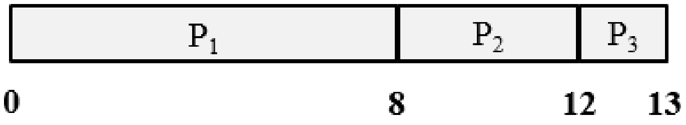
8. Unix and Linux systems start from an initial process (or init process). a) What is process ID of the init process? It forks child processes, child processes fork child processes. This parent and child relation form a tree. b) What is the tree called? c) Explain the role of the init process on UNIX and Linux systems in regard to process termination. (**3 marks**)
Ans:
a) pid=1 (**1 mark**)
b) process tree (**1 mark**)
c) In Unix/Linux, **it adopts orphan process** (**1 mark**). If an orphan process terminates, the init process cleans or reads its exit status and remove it from the system.

9. Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made. Most of the marks will be given for the calculations that demonstrate how you obtained the answer. Give Gantt Charts to help in your explanation.



Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a) What is the average turnaround time for these processes with the FCFS scheduling algorithm? (**3 marks**)

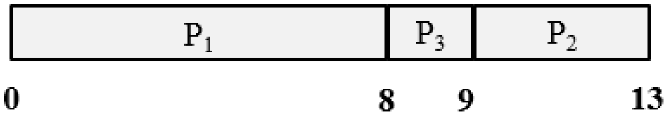


[**1 mark**]

Turnaround time for P_1 = 8; P_2 = 11.6; P_3 = 12 [**1 mark**]

Average turnaround time: (8 + 11.6 + 12)/3 = **10.53** [**1 mark**]

b) What is the average turnaround time for these processes with the SJF scheduling algorithm? (**3 marks**)



[**1 mark**]

turnaround time for P_1 = 8; P_2 = 12.6; P_3 = 8 [**1 mark**]

Average turnaround time: (8 + 12.6 + 8)/3 = 9.53 [**1 mark**]

10. Including the initial parent process, how many processes are created by the C program shown below? (**3 marks**)

```
#include <stdio.h>
#include <unistd.h>
int main()
{
pid_t pid;
pid = fork();
if (pid == 0) {
        exit(0);
}
fork();
fork();
return 0;
}
```

Ans: **5**

11. Consider the following C code. What output will be at Line A?

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int value = 5;
int main()
{
pid_t pid;
pid = fork();
if (pid == 0) { /* child process */
value += 15;
return 0;
}
else if (pid > 0) { /* parent process */
wait(NULL);
printf("PARENT: value = %d",value); /* LINE A */
return 0;
}
}
```

LINE A: ______ **PARENT: value = 5________________________**

5 (-1 mark for format**)**

12. Consider the following C code, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.) (**8 marks**)

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int main()
{
pid_t pid, pid1;
pid = fork();
if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
}
else if (pid == 0) {
        pid1 = getpid();
        printf("child: pid = %d",pid); /* A */
        printf("child: pid1 = %d",pid1); /* B */
}
else {
        pid1 = getpid();
        printf("parent: pid = %d",pid); /* C */
        printf("parent: pid1 = %d",pid1); /* D */
        wait(NULL);
}
return 0;
}
```

Line A: pid =______**0**____________ (**2 marks**)

Line B: pid1 =_____**2603**__________ (**2 marks**)

Line C: pid =_____**2603**__________ (**2 marks**)

Line D: pid1 =_____**2600**__________ (**2 marks**)