/10

Multiple Choice (3 points each)

1. For a single-processor system,
   A. processes spend long times waiting to execute.
   B. there will never be more than one running process.
   C. input-output always causes CPU slowdown.
   D. process scheduling is always optimal.

2. What is the main difference between traps and interrupts?
   A. How they are initiated.
   B. The kind of code that's used to handle them.
   C. Whether or not the scheduler is called.
   D. How the operating system returns for them.

3. A table points to routines that handle interrupts is called _____.
   A. interrupt handler
   B. interrupt vector
   C. interrupt indicator
   D. interrupt signal

4. Most often, application programs access system resources using _____.
   A. system calls
   B. kernel threads
   C. user threads
   D. application program interfaces

5. The major difficulty in designing a layered operating system approach is
   A. making sure each layer is easily converted to modules.
   B. making sure that each layer hides certain data structures, hardware, and operations.
   C. debugging a particular layer.

D. appropriately defining the various layers.

6. When a process is created using the classical `fork()` system call, which of the following is *not* inherited by the child process?
   A. Process address space
   B. Process ID
   C. User ID
   D. Open files
   E. Signal handlers
   F. None of the above

7. Which of the following is *not* true about message passing?
   A. In direct communication, the sending process specifies the receiving process (usually by ID).
   B. In indirect communication, the sending process specifies a mailbox rather than a process.
   C. In indirect communication a link may be associated with more than two processes.
   D. In direct communication, multiple links may exist between a pair of processes.

8. A thread control block _____.
   A. is managed by the parent process
   B. contains the same information as the process control block
   C. has the identical structure as the process control block
   D. does not include information about the parent process resource allocation

9. When a process is accessing its heap space, it exists in the _____.
   A. Running state
   B. Waiting state
   C. Terminating state
   D. Ready state

10. Long-term scheduling is performed _____.
    A. typically on submitted jobs
    B. when processes must be moved from waiting to ready state
    C. on processes in the ready queue
    D. All of the above are correct

11. In scheduling, the term *aging* involves

(A) higher priority processes preventing low-priority processes from ever getting the CPU.

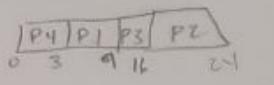
X (B) gradually increasing the priority of a process so that a process will eventually execute.

C. processes that are ready to run but stuck waiting indefinitely for the CPU.

D. processes being stuck in ready queues so long that they die.

12. Consider the following set of processes, the length of the CPU burst time given in milliseconds.

Process	Burst Time
P1	6
P2	8
P3	7
P4	3



24/4 = 6

3

Assuming the above processes being scheduled with the SJF scheduling algorithm,

A. the waiting time for process P1 is 3 ms.

B. the waiting time for process P1 is 0 ms.

X C. the waiting time for process P1 is 16 ms.

(D.) the waiting time for process P1 is 9 ms.

True or False (2 points each)

13. True/False T The two primary purposes of an operating system are to manage the resources of the computer and to provide a convenient interface to the hardware for programmers.

14. True/False F Non-preemptive scheduling algorithms are better for interactive jobs since they tend to favor jobs that require quick responses.

10

15. True/False F A context switch from one process to another can be accomplished without executing OS code in kernel mode.

X 16. True/False F An advantage of implementing threads in user space is that they don't incur the overhead of having the OS schedule their execution.

17. True/False T All processes in UNIX first translate to a zombie process upon termination.

X 18. True/False T Shortest Remaining Time First is the best preemptive scheduling algorithm that can be implemented in an operating system.

19. True/False F In a monolithic kernel, most operating system components, e.g., memory management, inter-process communication, and basic synchronization modules, execute outside the kernel.

Short Answer (5 points each)

20. What effect does the size of the time quantum have on the performance of a round robin algorithm?

The effect that the time quantum has on the performance of round robin algorithm is large and it takes time.

1

21. When a process executes a `fork()` system call, a duplicate process (i.e. the child process) is created. How does the code in the processes (since it is identical in both the parent and child processes) know which process is the parent and which is the child?

It knowns which one is which by the Process ID, the process ID is not inherited by the child process. how to distinguish?

2

22. When multiple processes need to cooperate, there is a choice between shared memory and message passing. Compare and contrast these two techniques. Make sure to clarify the role of the operating system in each.

Message passing is easier to implement than shared memory for inter-computer communication.

Shared memory is for a network protocol does not apply to OS.

1

23. Describe the difference between "CPU-bound" processes and "I/O-bound" processes.

CPU-bound processes can be expected to hold the CPU for long as CPU bound processes are ones that are implementing algorithms.

I/O bound processes will not stay the processes for very long

4

Long Questions

24. (15 points) Suppose that the following jobs arrive as indicated for scheduling and execution on a single CPU. Draw the Gantt chart and calculate the average waiting time for each of the following CPU scheduling algorithms.

NJ+ turn. time

Job	Arrival Time	CPU Burst Time (msec)	Priority
J_1	0	12	1(Gold)
J_2	2	4	3(Bronze)
J_3	5	2	1(Gold)
J_4	8	10	3(Bronze)
J_5	10	6	2(Silver)

(a) First-Come, First-Served (FCFS).

(b) Non-preemptive Shortest Job First (SJF).

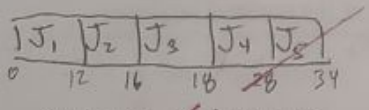
(c) Preemptive Shortest Job First (a.k.a. Shortest Remaining Time First)

(d) Round Robin with time quantum of 4 (assume that new jobs will be inserted at the end of the ready queue).

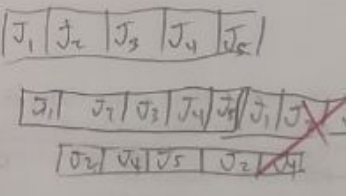
(e) (Preemptive) Priority Scheduling.

7

a) FCFS



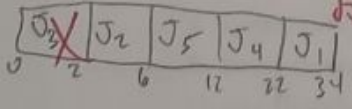
34+28+18+16+12 / 5 = 21.6

d) 

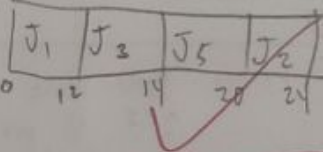
= 31+28+23+15 / 5 = 20.8

b) SJF

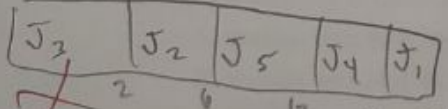
J5 arrives at time 5



34+22+12+6+2 / 5 = 15.2

e) 

34+24+20+ / 5 = 20.8

c) 

= 15.2

25. (7 points) You write a UNIX shell, but instead of calling `fork()` then `exec()` to launch a new job, you insert a subtle difference: the code first calls `exec()` and then calls `fork()` as shown below.

```
shell(..) {
   ...
   exec(cmd, args);
   fork();
   ...
}
```

Does it work? What is the impact of this change to the shell, if any? Explain your answer.

It will work, but there will be a problem it will lead to deal lock

2

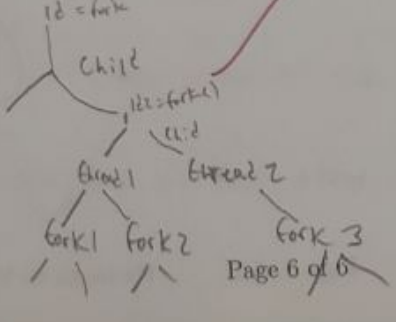
26. (8 points) How many times does the following program print "Hello World"? Draw a simple tree diagram to show the parent-child hierarchy of the spawned processes.

```
#include <stdio.h>
#include <unistd.h>

int  main()  {
   int i;
   for (i = 0; i <  3; i++)
     fork();
   printf("Hello World\n");
   return 0;
}
```

Prints Hello World 3 times, 3 fork calls
$2^3 = 8$ processes

8



End of exam.