IOPS 332 C

Homework Set 3

Q1. **C**hoose the best option from the following.

(a). Which of the following is not a function of the operating system?

(i) Generate interrupts
(ii) making the computer sytem convenient to use
(iii) manage i/o devices
(iv) protecting user programs from one another

(b). Choose the best definition of process :

(i) An executable program
(ii) program code +contents of processor's registers+stack
(iii) program code +contents of processor's registers+stack+PCB+ready queue
(iv) program code +contents of processor's registers

(c). What is the difference between multithreading and multiprocessing?
(i) multiple threads share code and data sections
(ii) only processes require context switching
(iii) only threads can support parallelism

(d). What is the advantage of multiprogramming:

(i) efficient use of CPU
(ii) fast response
(iii) efficent use of disk

(e). Which component ensures that process can execute within its own address space?
(i) I/O device
(ii) memory addressing hardware
(ii) stack pointers

(f). Which of the following instructions should be priviledged?
(i) Read data from disk
(ii) set priority of process
(iii) read the clock

(g). Threads of the same task....I. Share the same address space II. Reduce context switching overhead III. Are protected from each other the same weight as heavy weight processes.

(i) Only I
(ii) I and II
(iii) I, II and III

(h). Assume a single thread kernel OS running multiple user thraeds. If one user thread requests a read system call then....

(i) other system threads continue to run

(ii) some user threads run and some are blocked
(iii) all user threads are blocked

(i). Which of the following statements about the process state transitions are FALSE:

(i) When a running process receives interrupt, it goes to ready state.
(ii) Upon finishing, the running process exits and goes to terminated state.
(iii) A ready state goes to running state when the scheduler schedules it.
(iv) An I/O process on completion of I/O request goes back to running process.

10.
(j)

2. Consider the following code fragment:

```
if (fork() == 0)
    { a = a + 5; printf("%d,%d\n", a, &a); }
else { a = a – 5; printf("%d,%d\n", a, &a); }
```

Let u, v be the values printed by the parent process, and x, y be the values printed by the child process. Which one of the following is TRUE?
(a) u = x + 10 and v = y
(b) u = x + 10 and v != y
(c) u + 10 = x and v = y
(d) u + 10 = x and v != y

3. A uni-processor computer system only has two processes, both of which alternate 10 ms CPU bursts with 90 ms I/O bursts. Both the processes were created at nearly the same time. The I/O of both processes can proceed in parallel. Which of the following scheduling strategies will result in the least CPU utilization (over a long period of time) for this system?
(a) First come first served scheduling
(b) Shortest remaining time first scheduling
(c) Static priority scheduling with different priorities for the two processes
(d) Round robin scheduling with a time quantum of 5 ms

4. A uni-processor computer system only has two processes, both of which alternate 10 ms CPU bursts with 90 ms I/O bursts. Both the processes were created at nearly the same time. The I/O of both processes can proceed in parallel. Which ofthe following scheduling strategies will result in the least CPU utilization (over a long period of time) for this system?
(i) First come first served scheduling
(ii) Shortest remaining time first scheduling
(iii) Static priority scheduling with different priorities for the two processes
(iv) Round robin scheduling with a time quantum of 5 ms

5. The following program:
main(){
if(fork()>0)
sleep(100);
}
results in the creation of:
(i) an orphan process
(ii) a zombie process
(iii) a process that executes forever
(iv) None of these

6. You write a UNIX shell, but instead of calling fork() then exec() to launch a new job, you instead insert a subtle difference: the code first calls exec() and then calls fork() like the following:

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

Does it work? What is the impact of this change to the shell, if any? (Explain)

5. (A) Assume you want to implement a web-server for YouTube by using multithreading, where each thread serves one incoming request by loading a video file from the disk. Assume the OS only provides the normal blocking read system call for disk reads, do you think user-level threads or kernel-level threads should be used? Why?

(B) Now you want to implement a web-server for Facebook, to serve each user's "Home" page (the first page you see after you log in). This time your web-server needs to perform many tasks: load the news feeds from each of your friends, load the advertisement, check for new messages, etc. Now you want to implement your web-server by using multithreading, and have one thread to perform each of the tasks, and later these threads will cooperate with each other to collectively construct the "Home" page. For performance reasons, Facebook makes sure that all the data these threads need is already cached in the memory (so theydon't have to perform any disk I/O). Do you think user-level threads or kernel-level threads should be used? Why?

6. Consider the workload in the following table :

Process	Burst Time	Priority	Arrival Time
P1	10	4	1
P2	12	3	0
P3	5	2	2
P4	8	1	4

Draw the Gannt chart for preemptive shortest job first, preemptive priority scheduling and round robin scheduling (time slice =2). What is the average waiting time and response time in each case?

7. The following code in C uses an IPC mechanism to communicate between two processes. Identify the IPC mechanism used and document the code (fill the comments sections in the code) to highlight the usage of the different POSIX system calls used for IPC. Additionally, you need to comment on the output.

```
main(){
      int shmid,status;
      int *a, *b;
      int i;

    shmid = shmget(IPC_PRIVATE, 2*sizeof(int), 0777|IPC_CREAT);

          /*--------Comments----- */

   if (fork() == 0) {                          /*--------Comments----- */

          b = (int *) shmat(shmid, 0, 0);      /*--------Comments */

          for( i=0; i< 10; i++) {

                sleep(5);

             printf("\t\t\t Child reads: %d,%d\n",b[0],b[1]); /*----Comments---- */

          }

     shmdt(b);                                 /*----Comments---- */

    }

     else {                                          /*----Comments---- */

          a = (int *) shmat(shmid, 0, 0);      /*----Comments----- */

          a[0] = 0; a[1] = 1;

           for( i=0; i< 10; i++) {
                sleep(1);
                a[0] = a[0] + a[1];
                a[1] = a[0] + a[1];
                printf("Parent writes: %d,%d\n",a[0],a[1]);  /*----Comments---- */
           }

          wait(&status);                       /*-----Comments----- */

            shmdt(a);                                 /*----Comments----- */

          shmctl(shmid, IPC_RMID, 0);          /*----Comments------ */

}}
```