NORTHEASTERN UNIVERSITY
COLLEGE OF ENGINEERING
Department of Electrical & Computer Engineering

95

EECE7376: Operating Systems: Interface and Implementation **Dr. Emad Aboelela**
Spring 2024 – Quiz 1 **Time Allowed: 1.5 Hour**

Student Name: Student ID:

This is a **closed book** and **closed notes** exam. The exam has **6 questions**
You **are not allowed** to use any **electronic** device nor **scratch** papers
No answers outside these exam pages will be graded (you can write on **both sides**)
Make sure to **write your full name** on the empty side of your cheat sheet
You cannot leave the exam room once the exam starts (except in case of an emergency)

For all given programs in this quiz assume all needed libraries header files are included and all called library functions are executed without errors.

9

Q1. (9 Points)

a) The following program runs two threads and waits for them to finish.

```
volatile int counter = 1000;
void *worker(void *arg) {
  int c = counter;
  Spin(.25);   //sleep for 0.25 second
  c--;
  counter = c;
  return NULL;       }

int main(int argc, char *argv[]) {
  pthread_t p1, p2;
  Pthread_create(&p1, NULL, worker, NULL);
  Pthread_create(&p2, NULL, worker, NULL);
  Pthread_join(p1, NULL);
  Pthread_join(p2, NULL);
  printf("%d\n", counter);
  return 0;
}
```

0093

List all possible outputs on the computer screen when the code runs. Explain your answer.

b) The following program modified the program in (a) by replacing the global variable `counter` with a local one in the `main()` function and passes its address to function `worker()`.

```
void *worker(void *arg) {
  int* cp = (int*)arg;
  int c = *cp;
  Spin(0.25);   //sleep for 0.25 second
  c--;
  *cp = c;
  return NULL;       }

int main(int argc, char *argv[]) {
  int counter = 1000;
  pthread_t p1, p2;
  Pthread_create(&p1, NULL, worker, &counter);
  Pthread_create(&p2, NULL, worker, &counter);
  Pthread_join(p1, NULL);
  Pthread_join(p2, NULL);
  printf("%d\n", counter);
  return 0;
}
```

List all possible outputs on the computer screen when the code runs. Explain your answer.

c) Repeat part (b) above if the following code line in function `worker()` is removed:

```
*cp = c;
```

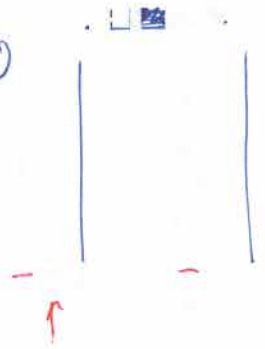
Explain your answer.

Q2. (8 Points)

Given the following code:

```
int main(int argc, char *argv[]) {
 printf("a\n");
 int fr = fork();
 if (fr) printf("b\n");
 printf("c\n");
 return 0;
}
```

a) List all possible outputs on the computer screen when the code runs. Explain your answer.



b) In the above code assume calling `Spin(5)` to sleep for 5 second in the `if` statement block as follows: `if (fr) {Spin(5); printf("b\n");}`
What will be the expected output on the computer screen assuming that the OS scheduler does not keep a process in the "Ready" state for more than one second? Explain your answer.

Q3. (15 Points)

The following program represents a parent process that forks a child process and waits for it to finish. Assume that:

14

- The text file "input.txt" contains one line with the following word: **Welcome!**
- The text file "output.txt" is empty (notice that this file is opened for append).

```
int main(int argc, char *argv[]) {
 printf("Hello!\n");
 int rc = fork();
 close(STDOUT_FILENO);
 int fd=open("./output.txt", O_CREAT|O_WRONLY|O_APPEND, S_IRWXU);
 if (rc == 0) { // cHILD
        char *myargs[3];
        myargs[0] = strdup("wc"); //The Linux wc command
        myargs[1] = strdup("input.txt");
        myargs[2] = NULL;
        execvp(myargs[0], myargs);
  } else                          {
       int wc = wait(NULL);
       printf("Bye!\n");
       fsync(fd);
       close(fd);                 }
 return 0;
}
```

a) What will be the outputs on the computer screen when the code runs?

Hello!

b) What will be the contents of the file `output.txt` after the code runs?

8
Bye

← output from 'wc input.txt' command. I'm not s

-1

c) Assume `close(STDOUT_FILENO)` in the above code is replace with `close(STDIN_FILENO)`. What will be the outputs on the computer screen when the modified code runs?

Q4. (24 Points)

The shown program utilizes two pipes between a parent process and its child process. The parent uses `pipe1` to send a character to the child. The child uses `pipe2` to send a character to the parent.

24

a) What will be the outputs on the computer screen when the program runs?

b) What line of code needs to be added in the shown code to ensure that the program will always output the following on the screen?
Parent Child Parent Child

```
int main (){
  int pipe1[2], pipe2[2];
  pipe(pipe1);
  pipe(pipe2);

  int rc = fork();
  int i;
  char c;

  if (rc == 0) {
   for (i = 0; i<2; i++) {
     read(pipe1[0], &c, sizeof(c));
     printf("Child ");
     fflush(stdout);
     write(pipe2[1], &c, sizeof(c));
   }
   }
   else {
    for (i = 0; i <2; i++) {
     read(pipe2[0], &c, sizeof(c));
     printf("Parent ");
     fflush(stdout);
     write(pipe1[1], &c, sizeof(c));
     }
    wait(NULL);
    printf("\n");
   }
 return 0;
}
```

c) What line of code needs to be added in the shown code to ensure that the program will always output the following on the screen?
Child Parent Child Parent

Q5. (14 Points)

4

a) For the OS one-queue scheduling policies: FCFS, non-preemptive SJF, and RR , which of them is generally:

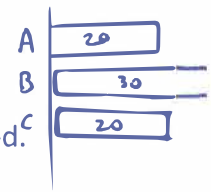
FIFO
Convoy EFFECT

i. the worst for the turn-around time metric? RR
ii. the best for the response time metric? RR
iii. the worst for its high number of context switching?
iv. the best to avoid the starvation problem? RR

b) Rule 5 of the **MLFQ** scheduler states that after some time period ***S***, move all the jobs in the system to the topmost queue. Explain the effect of having ***S*** either too short or too long on the response and turnaround times of the jobs.

prob

Q6. (30 Points)



26

Assume we have three jobs that enter a system all at time zero and need to be scheduled.

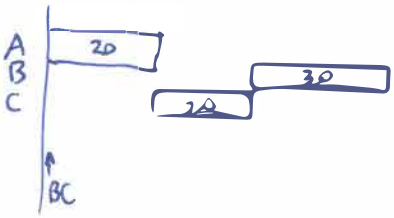
- The first job that enters is called ***A***, and it needs 20 seconds of CPU time.
- The second, which arrives just after ***A***, is called ***B***, and it needs 30 seconds of CPU time.
- The third, ***C***, arrives just after ***B***, and needs 20 seconds of CPU time.

NO

RR: x<y → x←y

Ignore the time needed in context switching and assume **only** for the round-robin scheduling policy if job *x* arrives just before job *y*, the scheduler will schedule *x* before *y*. All other policies consider all jobs arrive at the same time. Answer the following questions (**show all steps**):

a) If shortest-job-first (SJF) policy is used, what is ***B***'s turnaround time?



$T_{turnaround}_{b} = 20 + 20 + 30 =$

P_a

-2

b) If longest-job-first (LJF) policy is used, what is ***B***'s turnaround time?

$T_{turnaround}_{b}$ P_a

c) If round-robin policy (with a time slice length of 1 second) is used, what is ***A***'s turnaround time?

P_a

d) If round-robin policy (with a time slice length of 1 second) is used, what is ***B***'s turnaround time?

-2 (e) For a round-robin policy, what is the smallest value of its time slice that will result in ***B*** finishes before ***C*** ?

Make time slice of RR bigger than P_B (30 sec).

End of the Quiz Questions