Unit 5 Sample Problems - Processes

In this sample problem set, we will practice concepts of processes and process management.

- Length: 60 minutes with discussion.
- Questions: Q1, Q3-Q5. (optional: Q2, Q6)

Process Concept

1. [Acuña] There are four general regions to a process in memory: stack, heap, data, and text. **List and explain** which of these must be separated for each program running and which might be combined globally. (Don't worry about security or programs misbehaving.) [2 points]

2. [Bush] A PCB typically contains a list of the files that the process has open. **Explain** how the PCB concept could be used to avoid concurrent IO issues? [2 points]

Process Scheduling

3. [Bahremand] **Explain** why the queue diagram in slide 9 has a *continuous cycle* that flows between the listed queues and resources? Is a cycle necessary? [2 points]

Operations on Processes

4. [Lisonbee] **Trace** the program below, identify the values of the pids at lines A, B, C, D, E, and F. (Assume that the actual pid of Process 1 is 1502, Process 2 is 1505, and Process 3 is 1507. Also assume that fork will always succeed.) [4 points]

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t temp1, temp2, temp3;
    temp1 = fork();

    if (temp1 < 0) { /* Error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (temp1 == 0) { /* Process 2 */
        temp2 = fork();

        if (temp2 < 0) { /* Error occurred */
            fprintf(stderr, "Fork Failed");
            return 1;
        }
        else if (temp2 == 0) { /* Process 3 */
            temp3 = getpid();
            printf("temp2 = %d", temp2); /* A */
            printf("temp3 = %d", temp3); /* B */
        }
        else { /* Process 2 */
            temp3 = getpid();
            printf("temp2 = %d", temp2); /* C */
            printf("temp3 = %d", temp3); /* D */
            wait(NULL);
        }
    }
    else { /* Process 1 */
        temp2 = getpid();
        printf("temp1 = %d", temp1); /* E */
        printf("temp2 = %d", temp2); /* F */
        wait(NULL);
    }
    return 0;
}
```

5. [Lisonbee] **Trace** the following program and then list all of the possible outputs that could be generated. (Assume that fork will always succeed.) [2 points]

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid1, pid2;
    pid1 = fork();

    if (pid1 < 0) {
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid1 == 0) {
        printf("A");
        pid2 = fork();

        if (pid2 < 0) {
            fprintf(stderr, "Fork Failed");
            return 1;
        }
        else if (pid2 == 0) {
            printf("B");
        }
        else {
            wait(NULL);
            printf("C");
        }
    }
    else {
        printf("D");
    }
    return 0;
}
```

Interprocess Communication

6. [Lisonbee] Consider a system where two processes (a producer and a consumer) use a message-passing system to communicate, and each process does work at different rates. The producer can produce (perform work) at any rate, but the consumer has to wait for the producer to complete its task before moving on. Based on this system's needs, **explain** whether a synchronous or asynchronous communication system would be a better choice and why. [2 points]

Examples of IPC Systems

Communication in Client-Server Systems