

Homework 3- Unit 3

CGS3763: Operating Systems Concepts

Due June 21, 2020

1. **(10 points)** Including the initial parent process, how many processes are created by the following program? Explain.

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

int main()
{
    int i;

    for (i = 0; i < 4; i++)
        fork();

    return 0;
}
```

2. **(10 points)** Explain the circumstances under which the line of code `printf("LINE J")` in the following program will be reached.

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

int main()
{
    pid_t pid;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        execlp("/bin/ls", "ls", NULL);
        printf("LINE J");
    }
    else { /* parent process */
        /* parent will wait for the child to complete */
        wait(NULL);
        printf("Child Complete");
    }

    return 0;
}
```

3. **(10 points)** What are the benefits and detriments of each of the following? Consider both the systems and the programmers' levels.
- a) Symmetric and asymmetric communication
 - b) Automatic and explicit buffering
 - c) Send by copy and send by reference
 - d) Fixed-sized and variable-sized messages?
4. **(10 points)** Give an example of a situation in which ordinary pipes are more suitable than named pipes and an example of a situation in which named pipes are more suitable than ordinary pipes.
5. **(10 points)** Consider the RPC mechanism. Describe the undesirable consequences that could arise from not enforcing either the "at most once" or "exactly once" semantic. Describe possible uses for a mechanism that has neither of these guarantees.
6. **(10 points)** Using either a UNIX or a Linux system, write a C program that forks a child process that ultimately becomes a zombie process. This zombie process must remain in the system for at least 10 seconds. Process states can be obtained from the command

ps -l

The process states are shown below the S column; processes with a state of Z are zombies. The process identifier (pid) of the child process is listed in the PID column, and that of the parent is listed in the PPID column. Perhaps the easiest way to determine that the child process is indeed a zombie is to run the program that you have written in the background (using the &) and then run the command **ps -l** to determine whether the child is a zombie process. Because you do not want too many zombie processes existing in the system, you will need to remove the one that you have created. The easiest way to do that is to terminate the parent process using the kill command. For example, if the process id of the parent is 4884, you would enter

kill -9 4884

7. **(10 points)** Using the following program in, 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.)

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

int main()
{
    pid_t pid, pid1;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        pid1 = getpid();
        printf("child: pid = %d",pid); /* A */
        printf("child: pid1 = %d",pid1); /* B */
    }
    else { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d",pid); /* C */
        printf("parent: pid1 = %d",pid1); /* D */
        wait(NULL);
    }

    return 0;
}
```

8. **(10 points)** Design a program using ordinary pipes in which one process sends a string message to a second process, and the second process reverses the case of each character in the message and sends it back to the first process. For example, if the first process sends the message Hi There, the second process will return hI tHERE. This will require using two pipes, one for sending the origin al message from the first to the second process and the other for sending the modified message from the second to the first process. You can write this program using either UNIX or Windows pipes.

9. **(10 points)** Using the program shown below, explain what the output will be at lines X and Y.

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

#define SIZE 5

int nums[SIZE] = {0,1,2,3,4};

int main()
{
    int i;
    pid_t pid;

    pid = fork();

    if (pid == 0) {
        for (i = 0; i < SIZE; i++) {
            nums[i] *= -i;
            printf("CHILD: %d ",nums[i]); /* LINE X */
        }
    }
    else if (pid > 0) {
        wait(NULL);
        for (i = 0; i < SIZE; i++)
            printf("PARENT: %d ",nums[i]); /* LINE Y */
    }

    return 0;
}
```

10. **(10 points)** Design a file-copying program named filecopy using ordinary pipes. This program will be passed two parameters: the name of the file to be copied and the name of the copied file. The program will then create an ordinary pipe and write the contents of the file to be copied to the pipe. The child process will read this file from the pipe and write it to the destination file. For example, if we invoke the program as follows: filecopy input.txt copy.txt the file input.txt will be written to the pipe. The child process will read the contents of this file and write it to the destination filecopy.txt. You may write this program using either UNIX or Windows pipes.