Sri Lanka Institute of Information Technology

IT2060 – Operating Systems and Systems Administration

Year 2, Semester 1- 2024

Assignment Report

IT23163218

Table Of Contents

Question 1: Consider the following C program and answer the questions.

```
 #include <stdio.h>
#include <unistd.h>
int main() {
   pid_t pid;
   for(int i = 0; i < 10; i++) {
      if ((pid = fork()) < 0) {
         // error
      } else if (pid == 0) {
         function_A();
         return 0;
      }
      printf("process ID: %d \n", pid); // Line A
   }
   for(int i = 0; i < 10; i++) // Line B
      wait();
   return 0;
}
```

(i) How many new processes are created in the program? Justify your answer?

- **Answer**
    - 10 new processes.

- **Justification**
    - This for loop iterates 10 times, each time fork() creates child process. Therefore, a total of 10 new processors are created.

(ii) Which process, the parent or the child, executes function A()? Justify your answer?

- **Answer**
    - Child Process

- **Justification**
    - The else if condition (pid == 0) ensures that only the child process enters this block of code, where function_A() is called. The parent process does not execute this block and continues with the loop.

(iii) Whose PID, the parent or the child, is printed in Line A? Justify your answer?

- **Answer**
    - child PID

- **Justification**
    - This, in the parent process, pid has the value of the child's PID and this value is printed in Line A. But the child process does not reach Line A because the child process returns immediately after the execution of function_A().

(iv) What is the purpose of the for loop with the wait() in Line B?

- **Answer**
    - The wait() function in Line B is used to wait for all child processes to terminate before the parent process continues

Question 2: Consider the following C program and answer the questions.

```
int main () {
   for(i = 0; i < K; i++) {
      pid = fork();
   }
}
```

(i) For K=5, How many processes are in the memory when the program is executed?

- Answer
    - There are 32 processes in the memory

(ii) Modify the above program so that only the parent process creates 3 child processes, and each newly created process calls a function CPU( ). In addition, make the parent process wait for each child's termination.

```
#include <stdio.h>
#include <unistd.h>
int main(){
    pid_t pid;
    for (int i = 0; i < 3; i++){
        pid = fork();
        if(pid==0){
            CPU(); //Child Process Executes CPU Function
            return 0;
        }
    }

    for (int i = 0; i < 3; i++){
        wait(); //Parent process wait for each child terminate
    }
    return 0;
}
```

Question 3: Consider the following program. Explain the meanings of every line in the code and mention the output in Line A?

// Declares and initializes a variable value with 40.
int value = 40;

int main() {

// Declares a variable pid using pid_t data type
pid_t pid;

// Creates a new process. The fork() function returns 0 to the child process and the child's PID to the parent process.
pid = fork();

// If the process is the child (pid == 0), it increments value by 15.
if (pid == 0) {
value = value + 15;
}

```
// If the process is the parent, it decrements value by 15 and prints the updated
value. Then after waits for the child process to terminate.
   else if (pid > 0) {
      value = value - 15;
      printf("PARENT: value= %d \n", value); // Line A
      wait(NULL);
   }
}
```

- **The output of Line A is** "PARENT: value= 25".

Question 4: Consider the following programs A and B and answer the questions given below.

(i) How many times will the fork () function be called in Program A? (i.e., how many processes are created?) Justify your answer.

```
//Program A
 int main() {
   pid_t pid;
   int i;
   for (i = 0; i < 4; i++)
      pid = fork();
}
```

- **Answer**
  - 4 times
  - 15 new processes created. There are 16 processes with Initial process(2^4 = 16)

(ii) What is the output of Line A in Program B? Justify your answer.

Answer

- Output of the Line A

- In child process >> Value= 55
- In parent process >> Value= 5

- one parent and one child. Both have their private copy of the variable value. Changes in one process do not affect the other. In the child process, the value will be incremented by 25, and in the parent process, it is decremented by 25. Subsequently, both processes print their respective values ofvalue: 55 in the child and 5 in the parent.

References

- https://www.geeksforgeeks.org/fork-system-call/
- https://stackoverflow.com/questions/19461744/how-to-make-parent-wait-for-all-child-processes-to-finish