

Department of Information Technology, Faculty of Computing

Learning Objectives: Students will be able to learn UNIX process management system calls and library functions.

Step 1:

Consider the following C program.

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

for(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(i = 0; i < 10; i++) //Line B
wait();
```

- (i) How many new processes are created in the program? Why?
- (ii) Which process, the parent or the child, executes function_A()? Why?
- (iii) Whose PID, the parent or the child, is printed in Line A? Why?
- (iv) What is the purpose of the loop (with its wait()) in Line B?

Step 2:

Consider the following C program.

```
// Assume variables i and pid, and constant N have been properly defined,
and/or initialized and there is no syntax error.
int main ()
{
    for(i =0; i < N; i++) {
        pid=fork ();
    }
```

Department of Information Technology, Faculty of Computing

}

- (i) For $N=5$, How many processes are created when the program is executed?
- (ii) Modify the program so that only the parent process creates 3 child processes, and each new created process calls a function CPU(). In addition, make the parent process wait for each child's termination.

Step 3:

Consider the following program. What will be the output in Line A?

```
int value = 60;
int main()
{
    pid_t pid;
    pid = fork();
    if (pid == 0) {
        value = value + 20;
    }
    else if (pid > 0) {
        value = value -20;
        printf("PARENT: value= %d \n", value); //Line A
        wait(NULL);
    }
}
```

Step 5:

Consider the following programs **A** and **B**.

- (i) How many times will the fork () function be called in **Program A**? (*i.e.*, how many processes are created?) Justify your answer.
- (ii) What is the output of **Line A** in **Program B**? Justify your answer.

Department of Information Technology, Faculty of Computing

// Program A

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

```
}
```

// Program B

```
int value = 30;
int main()
{
    pid_t pid;
    pid = fork();
    if (pid == 0)

        value = value + 15;

    else if (pid > 0) {

        value = value -15;
        wait (NULL);

    }
    printf("Value= %d \n", value); //Line A
}
```