Addis Ababa University

Addis Ababa Institute of Technology

School of Information Technology and Engineering

Lab Report

Course Name: System Programming

Course Code: ITSE-3133

Name: Aymen Mohammednur

Id: ATR/5985/11

Section: Software 02

Instructor Name: Ms. Arisema Mezgebe

Submission Date: Sunday June 13/2021

Table of Contents

Part I: Process Creation and Process Id

Practical 1

process.c U

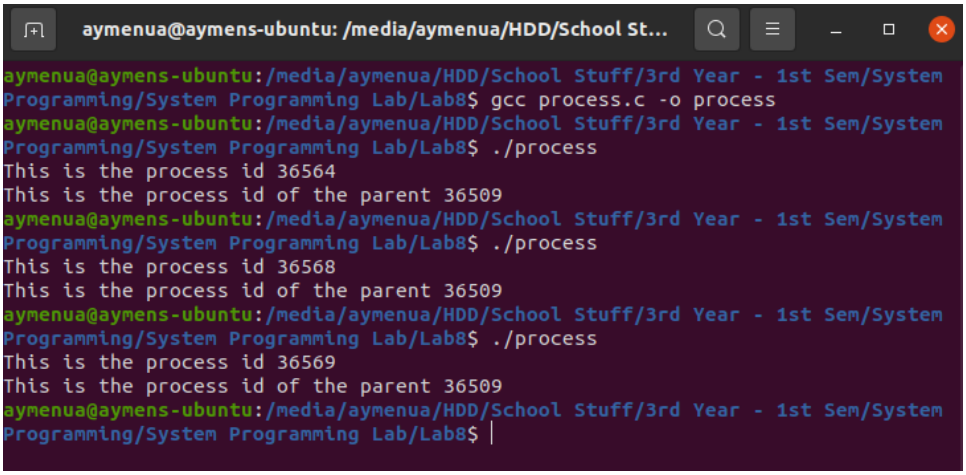
process.c > ...

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

void main(){
    printf("This is the process id %d\n", getpid());
    printf("This is the process id of the parent %d\n", getppid());
}
```

Questions

1. Compile and run the above program called process.c. What is displayed (printed) by the program?



2. The process id on the first run is 36564.
3. The process id of the parent is 36509.
4. The parent process is the one which doesn't change its value every time we run the program. It is the parent of the child calling process.

Practical 2

fork1.c U

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

void main()
{
    pid_t pid = fork();
    int x = 25;
    if(pid == 0) // child process returned
    {
        x += 5;
        printf("\nChild! process id: %d\n",getpid()); // child process id
        printf("Child! parent process id: %d\n",getppid()); // child's parent pid
        printf("Child! parent process id: %d\n",getppid());
        printf("The value of x is: %d\n",x);
    }
    else // parent proces returned
    {
        x -= 5;
        printf("Parent! process id: %d\n",getpid()); // parent process id
        printf("Parent! parent process id: %d\n",getppid()); // parent's parent pid
        printf("The value of x is: %d\n",x);
    }
    printf("Good bye from process with id : %d\n",getpid()); // goodbye id, executed by both processes
}
```

Questions

1. Compile and run the above program called fork1.c. What is displayed (printed) by the program?

```
aymenua@aymens-ubuntu: /media/aymenua/HDD/School St...
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ gcc fork1.c -o fork
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ ./fork
Parent! process id: 37428
Parent! parent process id: 36509
The value of x is: 20
Good bye from process with id : 37428

Child! process id: 37429
Child! parent process id: 37428
Child! parent process id: 37428
The value of x is: 30
Good bye from process with id : 37429
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$
```

2. Both processes executed their operations at the same time.
3. Process id of the parent is 37428 and process id of the child is 37429.
4. The last line (line 24) which is the printf statement is executed by both the child and parent process.

Practical 3

The fork2.c code is shown below…

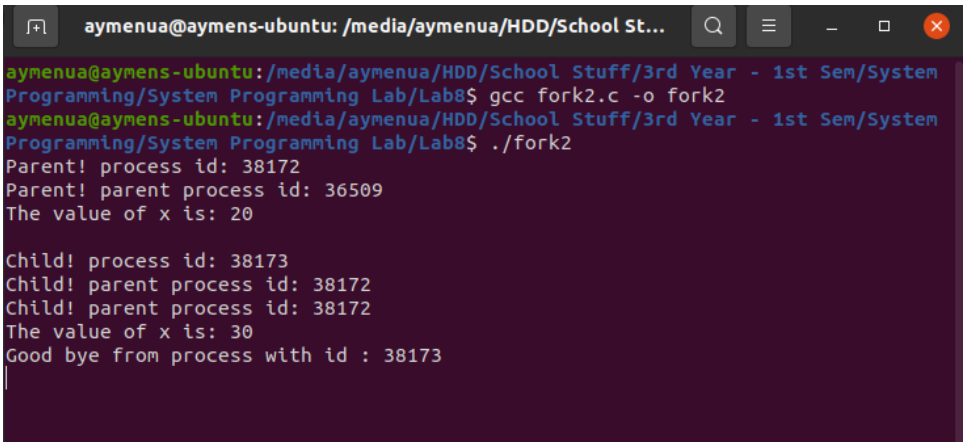
fork2.c U

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

void main()
{
    pid_t pid = fork();
    int x = 25;
    if(pid == 0)
    {
        x += 5;
        printf("\nChild! process id: %d\n",getpid());
        printf("Child! parent process id: %d\n",getppid());
        printf("Child! parent process id: %d\n",getppid());
        printf("The value of x is: %d\n",x);
    }
    else
    {
        x -= 5;
        printf("Parent! process id: %d\n",getpid());
        printf("Parent! parent process id: %d\n",getppid());
        printf("The value of x is: %d\n",x);
        sleep(10);
    }
    printf("Good bye from process with id : %d\n",getpid());
}
```

Questions

1. Compile and run the above program called fork2.c. What is displayed (printed) by the program?



2. Both processes executed at the same time but the final statement from the parent process is suspended for 10 seconds. So, it is executed after the child's final statement is printed out.

Part II: Process Termination and Waiting for Children

Practical 4

reap.c

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

void main()
{
    pid_t pid;
    pid = fork ();
    if (pid == 0)
    {
        printf ("I am child PID %d\n", getpid ());
    }
    else
    {
        printf ("I am parent PID %d\n", getpid());
    }
}
```

Questions

1. Write the above program reap.c and execute it

```
aymenua@aymens-ubuntu: /media/aymenua/HDD/School St...
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ gcc reap.c -o reap
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ ./reap
I am parent PID 38510
I am child PID 38511
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ ./reap
I am parent PID 38512
I am child PID 38513
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$
```

2. Modify the above program (reap.c) by adding the correct form of wait() and exit(0), so that the child terminates successfully and the parent waits for the child to finish execution displaying “I am child …”

reap.c

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

void main()
{
    pid_t pid;
    pid = fork ();
    if (pid == 0)
    {
        // child process
        printf ("I am child PID %d\n", getpid());
    }
    else
    {
        wait(NULL); // wait for child process to finish executing
        printf ("I am parent PID %d\n", getpid());
    }
    exit(0); // exit with status code 0
}
```

```
aymenua@aymens-ubuntu: /media/aymenua/HDD/School St...
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ gcc reap.c -o reap
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ ./reap
I am child PID 39534
I am parent PID 39533
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ ./reap
I am child PID 39542
I am parent PID 39541
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$
```

3. I added wait(NULL) in line number 18 (right after the else statement) to suspend it from executing until the child process was done. And exit(0) was added in line number 21 (after the else block) for both processes to execute.

4. The child process prints “I am...” first. This is because we have added the wait(NULL) command thus the parent will wait for the child to finish executing.

Part III: Command Line Arguments

The count.c code is shown below…

count.c

```c
#include <stdio.h>
#include <stdlib.h>

void main(int argc, char *argv[]) {
    int n = 0;
    /* Findout if any word passed to program */
    if(argc == 1)
    {
        printf("No word to examine. \n");
        exit(0);
    }
    /* Loop to count characters */
    while(argv[1][n++] != '\0');

    /*print result */
    printf("The word %s has %d characters .\n",argv[1],n);
}

```

The above code has the following output...

```
aymenua@aymens-ubuntu: /media/aymenua/HDD/School St...
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ gcc count.c -o count
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ ./count
No word to examine.
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$ ./count Aymen
The word Aymen has 6 characters .
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System Programming/System Programming Lab/Lab8$
```

Part IV: Running other programs using execv system call

Questions

1. Write execute.c whose code is shown. Compile and run the program.

execute.c

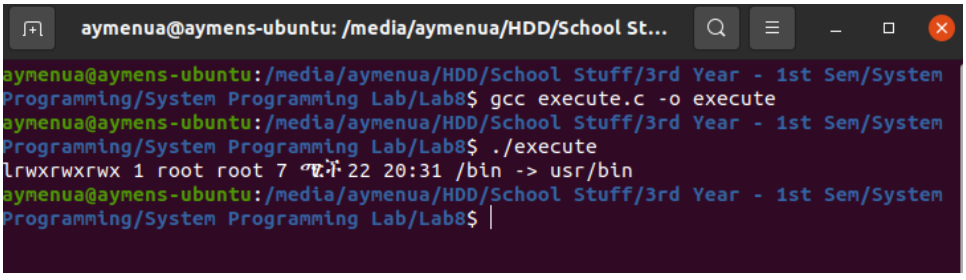
```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

void main()
{
    char *newargs[4];
    pid_t childpid;

    newargs[0] = "/bin/ls";
    newargs[1] = "-l";
    newargs[2] = "/bin";
    newargs[3] = NULL; /* Indicate end of args array */

    childpid = fork();
    if (childpid == 0)
    {   /* child code */
        execv("/bin/ls", newargs);
    }
    else
    {   /* parent code */
        wait(NULL); /*wait for child to finish */
    }
}
```

Output of the above code is shown below...



2. Modify above program (execute.c) so that it runs the command mkdir abc. mkdir is a program that creates a directory (folder) and the argument as a name for the directory.

execute.c

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

void main()
{
    char *newargs[4];
    pid_t childpid;

    newargs[0] = "mkdir";
    newargs[1] = "aymen";
    // newargs[2] = "/bin";
    newargs[3] = NULL; /* Indicate end of args array */

    childpid = fork();
    if (childpid == 0)
    {   /* child code */
        execv("/bin/mkdir", newargs);
    }
    else
    {   /* parent code */
        wait(NULL); /*wait for child to finish */
    }
}
```

3. Compile and run the program (the modified execute.c).

```
aymenua@aymens-ubuntu: /media/aymenua/HDD/School St...
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System
Programming/System Programming Lab/Lab8$ gcc execute.c -o execute
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System
Programming/System Programming Lab/Lab8$ ./execute
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System
Programming/System Programming Lab/Lab8$ ls -l
total 127
drwxrwxrwx 1 aymenua aymenua     0 ??  12 21:41 aymen
-rwxrwxrwx 1 aymenua aymenua 16784 ??  12 21:30 count
-rwxrwxrwx 1 aymenua aymenua   381 ??  12 21:29 count.c
-rwxrwxrwx 1 aymenua aymenua 16832 ??  12 21:41 execute
-rwxrwxrwx 1 aymenua aymenua   509 ??  12 21:39 execute.c
-rwxrwxrwx 1 aymenua aymenua 16832 ??  12 20:54 fork
-rwxrwxrwx 1 aymenua aymenua   845 ??  12 20:01 fork1.c
-rwxrwxrwx 1 aymenua aymenua 16872 ??  12 21:00 fork2
-rwxrwxrwx 1 aymenua aymenua   685 ??  12 19:51 fork2.c
-rwxrwxrwx 1 aymenua aymenua 16792 ??  12 20:46 process
-rwxrwxrwx 1 aymenua aymenua   207 ??  12 19:44 process.c
-rwxrwxrwx 1 aymenua aymenua 16864 ??  12 21:15 reap
-rwxrwxrwx 1 aymenua aymenua   462 ??  12 21:14 reap.c
aymenua@aymens-ubuntu:/media/aymenua/HDD/School Stuff/3rd Year - 1st Sem/System
Programming/System Programming Lab/Lab8$
```

4. As we can see, the directory named “aymen” was successfully created.