Jigar Makwana

Assignment 1

Due Date 09/29/17

Purpose of the project:

Main purpose of the assignment is to understand and different uses the function fork. The other objective is to learn use of the execve. By doing this assignment we learn child and parent process.

Background of the assignment:

To understand the different processes and functions in operating system. Forking is used to create child process of the main process. It takes no arguments and returns a process ID.

Algorithms/Functions used in assignment:

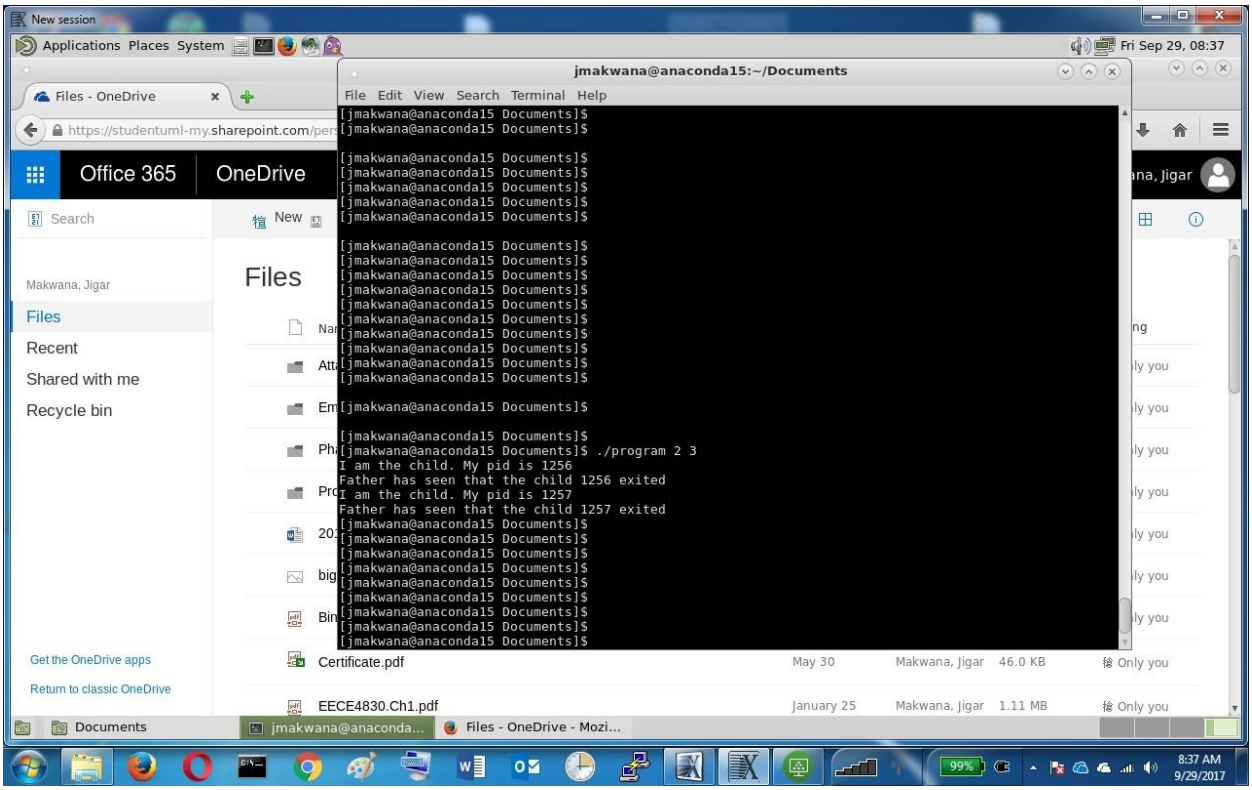
In the first program, I used the fork function to create a child process from main process and then I got the child ID.

In the second program, I used the execve function where the child spawns another process and execute the different command separately.

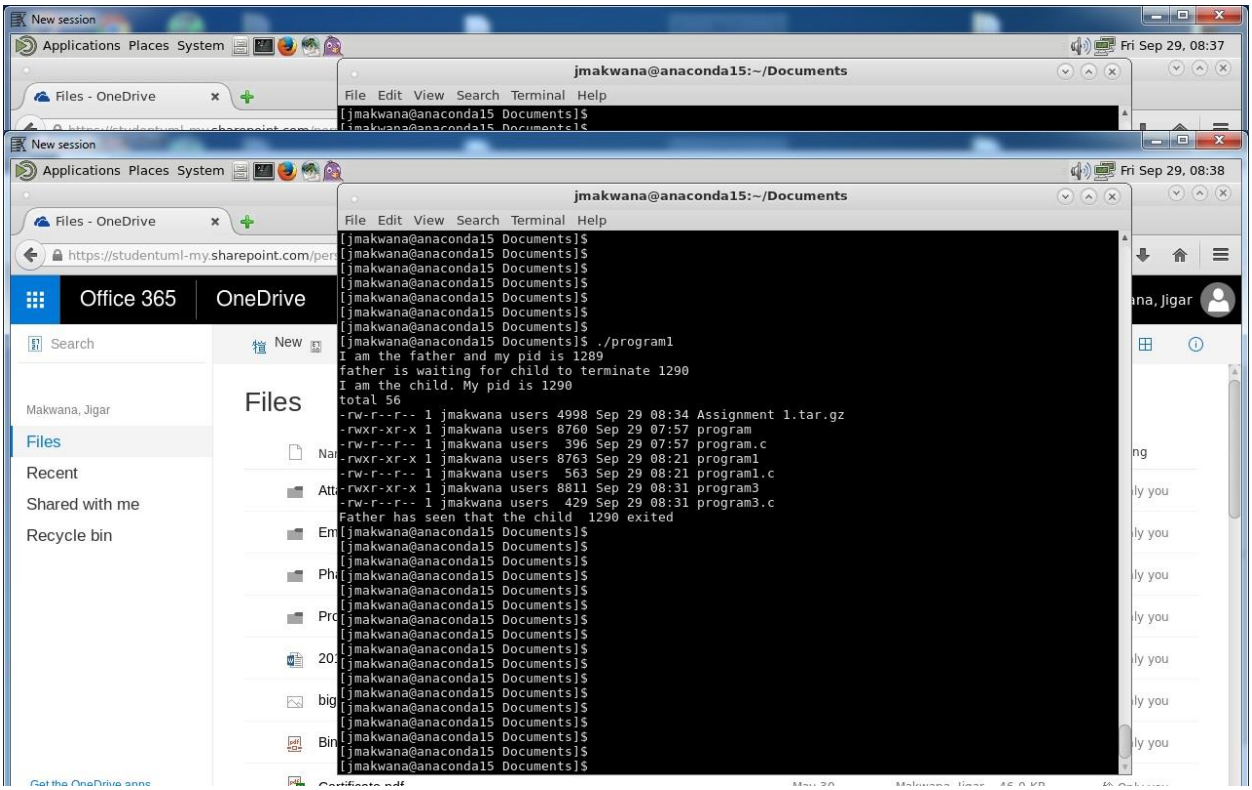
In the third program, I used the atoi function to convert the argument vector into the integer, by doing that I can set the limit twice the count of the argument vector and get the expected results.

Result:

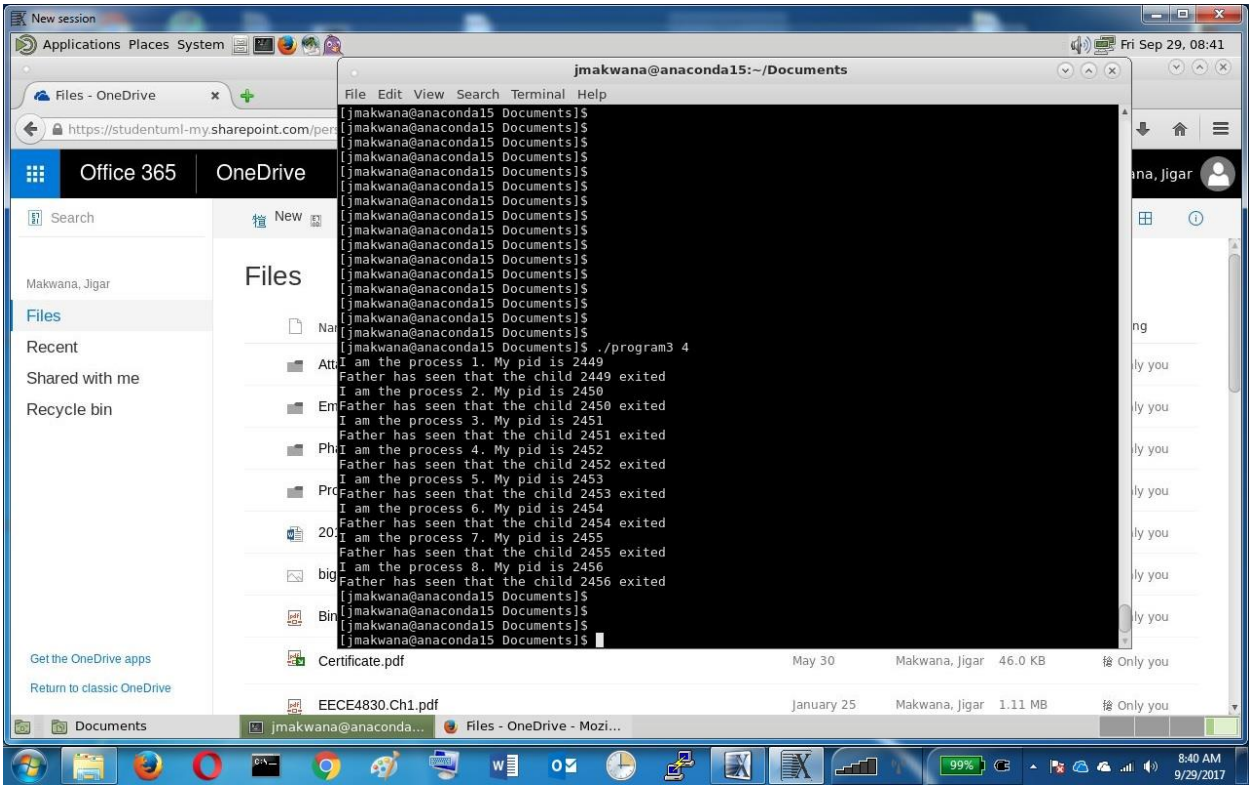
Part 1.a)



Part 1.b)


New session
Applications Places System
Fri Sep 29, 08:37
jmakwana@anaconda15:~/Documents
File Edit View Search Terminal Help
[jmakwana@anaconda15 Documents]$
New session
Applications Places System
Fri Sep 29, 08:38
jmakwana@anaconda15:~/Documents
File Edit View Search Terminal Help
[jmakwana@anaconda15 Documents]$
[jmakwana@anaconda15 Documents]$
[jmakwana@anaconda15 Documents]$
[jmakwana@anaconda15 Documents]$
[jmakwana@anaconda15 Documents]$
[jmakwana@anaconda15 Documents]$
[jmakwana@anaconda15 Documents]$
[jmakwana@anaconda15 Documents]$ ./program1
I am the father and my pid is 1289
father is waiting for child to terminate 1290
I am the child. My pid is 1290
total 56
-rw-r--r-- 1 jmakwana users 4998 Sep 29 08:34 Assignment 1.tar.gz
-rwxr-xr-x 1 jmakwana users 8760 Sep 29 07:57 program
-rw-r--r-- 1 jmakwana users 396 Sep 29 07:57 program.c
-rwxr-xr-x 1 jmakwana users 8763 Sep 29 08:21 program1
-rw-r--r-- 1 jmakwana users 563 Sep 29 08:21 program1.c
-rwxr-xr-x 1 jmakwana users 8811 Sep 29 08:31 program3
-rw-r--r-- 1 jmakwana users 429 Sep 29 08:31 program3.c
Father has seen that the child 1290 exited
[jmakwana@anaconda15 Documents]$
Files - OneDrive
https://studentuml-my.sharepoint.com/
Office 365
OneDrive
Search
New
Makwana, Jigar
Files
Recent
Shared with me
Recycle bin
Files

Part 2)



Observations:

By doing the assignment I observed that a process creates children processes and we can execute the single program by dividing the program into many different branches. By doing this we can execute the operation in parallel and much more efficient way.

Conclusion:

By doing the assignment I concluded that a process can be executed in parallel and the result of each sub-process can be sent to the main process. That will make the execution much simpler and less time consuming.

Source Code:

P1.a)

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

void main(int argc, char **argv){

      int res,k;
      for (k=1;k<argc;k++)
      {
      res=fork();
      if(res == 0){
                  printf("I am the child. My pid is %d\n",getpid());
                  exit();
                  }
      else {
            int child_pid = res;
            waitpid(child_pid, NULL,0);
            printf("Father has seen that the child %d exited\n",
child_pid);
            }

            }
      }
```

P1.b)

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

void main(int argc, char **argv){

      int res;

      res=fork();
      if(res == 0){
                  char *args[] = {"ls", "-l",NULL};
                  printf("I am the child. My pid is %d\n", getpid());
                  res = execve("/bin/ls", args,NULL);
                  }
      else {
            int child_pid = res;
            printf("I am the father and my pid is %d\n", getpid());
            printf("father is waiting for child to terminate
%d\n",child_pid);
            waitpid(child_pid, NULL,0);
            printf("Father has seen that the child  %d exited\n",
child_pid);
            }


      }
```

P2)

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

void main(int argc, char **argv){

      int res,j,k=1;
      j = atoi (argv[k]);
      for (k=1;k<=2*j;k++)
      {
      res=fork();
      if(res == 0){
                  printf("I am the process %d. My pid is
%d\n",k,getpid());
                  exit(0);
                  }
      else {
```

```
            int child_pid = res;
            waitpid(child_pid, NULL,0);
            printf("Father has seen that the child %d exited\n",
child_pid);
            }

            }
      }
```