fork(), wait() & exec() in Linux – example tasks

Task1:

Write a program where a child process is created to implement Fibonacci series.

Solution:

First of all create a .c file by using the following commands as:

```
$ vi task1.c
```

After this write the following code in the file on the Terminal as:

Code:

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

#include <stdio.h>

#include <unistd.h>

int main()

{

int pid;

int status;

/* fork a child process */

pid = fork();

if (pid < 0) { /* error occurred */

fprintf(stderr, "Fork Failed");

return 1;

}

else if (pid == 0) { /* child process */

printf("its a child process\n");

printf("the fibonacci series is printed in it. it is: \n");

int a=0, b=1, sum=0;

int no;
printf("-----------------------\n");

printf("Enter a number: \n");

scanf("%d", &no);

printf("-----------------------\n");

while(a<=no)

{

printf("%d\n", a);

sum=a+b;

a=b;

b=sum;

}

}

else { /* parent process */

/* parent will wait for the child to complete */

wait(&status);

printf("Child Complete\n");

}

return 0;

}
```

After this press “ESC” button twice and write the command:

```
:wq
```

(To save the written code and exit from the file)

After this write the following commands:

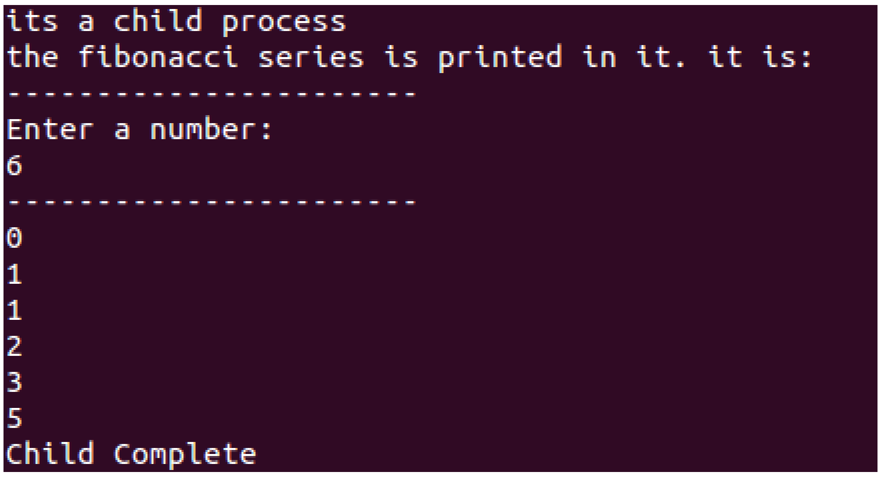
```
$ chmod 755 task1.c

$ gcc -o task1 task1.c

$ ./task1
```

- With the first command permissions for the file are changed.

- Second command compiles the code with the name task1 after this we can see the compiled file in the directory we are working.
- After that the compiled code is executed by the third command.

Output:

The output of the code was as:



```
its a child process
the fibonacci series is printed in it. it is:
-----------------------
Enter a number:
6
-----------------------
0
1
1
2
3
5
Child Complete
```

(The program successfully creates a child process so it tells that it is in a child process after that it takes in a number by the user and displays Fibonacci series less than that number.)

Task2:

Write a program where a child process is created to execute a command which accepts a pathname as argument and creates the components in that pathname and parent process executes a command and checks that the required components are successfully made by child process.

Solution:

First of all create a .c file by using the following commands as:

```
$ vi task2.c
```

After this write the following code in the file on the Terminal as:

Code:

```
#include<stdio.h>

#include<unistd.h>

#include<sys/wait.h>

int main()

{

pid_t pid, status;
pid=fork();

if(pid ==0)

{

/*Child Process creates the directories*/

execl("/bin/mkdir","mkdir", "-p", "Desktop/myfolder",NULL);

}

else if (pid >0)

{

/*Parent Process confirms the creation of directories, created in the child process*/

wait(&status);

printf("Parent!!!\n");

execl("/bin/ls","ls", "-aR",NULL);

/* in "-aR", 'a' lists all files while 'R' lists files in Reverse order*/

}

}
```

After this press "ESC" button twice and write the command:

```
:wq
```

(To save the written code and exit from the file)

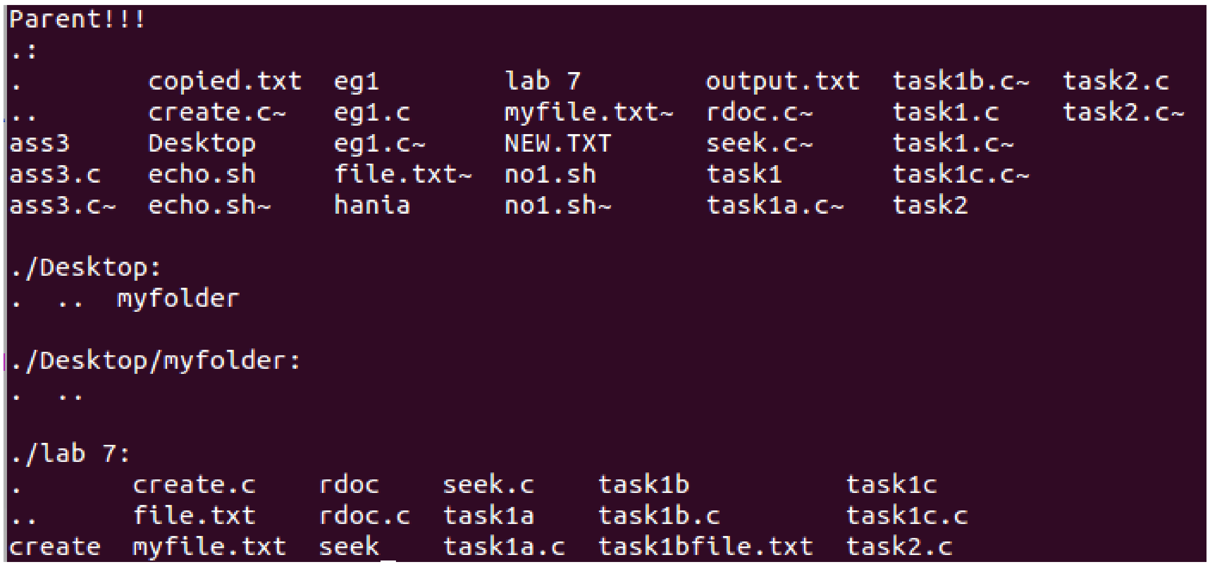
After this write the following commands:

```
$ chmod 755 task2.c

$ gcc -o task2 task2.c

$ ./task2
```

- With the first command permissions for the file are changed.
- Second command compiles the code with the name task2 after this we can see the compiled file in the directory we are working.
- After that the compiled code is executed by the third command.

Output:

The output of the code was as:



```
Parent!!!
.:
.        copied.txt  eg1        lab 7        output.txt  task1b.c~  task2.c
..       create.c~   eg1.c      myfile.txt~  rdoc.c~     task1.c    task2.c~
ass3     Desktop     eg1.c~     NEW.TXT      seek.c~     task1.c~
ass3.c   echo.sh     file.txt~  no1.sh       task1       task1c.c~
ass3.c~  echo.sh~    hania      no1.sh~      task1a.c~   task2

./Desktop:
.  ..  myfolder

./Desktop/myfolder:
.  ..

./lab 7:
.       create.c    rdoc    seek.c    task1b          task1c
..      file.txt    rdoc.c  task1a    task1b.c        task1c.c
create  myfile.txt  seek    task1a.c  task1bfile.txt  task2.c
```

(In the above code I have passed the path in the argument as "Desktop/myfolder" and thus when my code runs first it creates those directories then the parent process confirms its creation by the "-aR" command.)