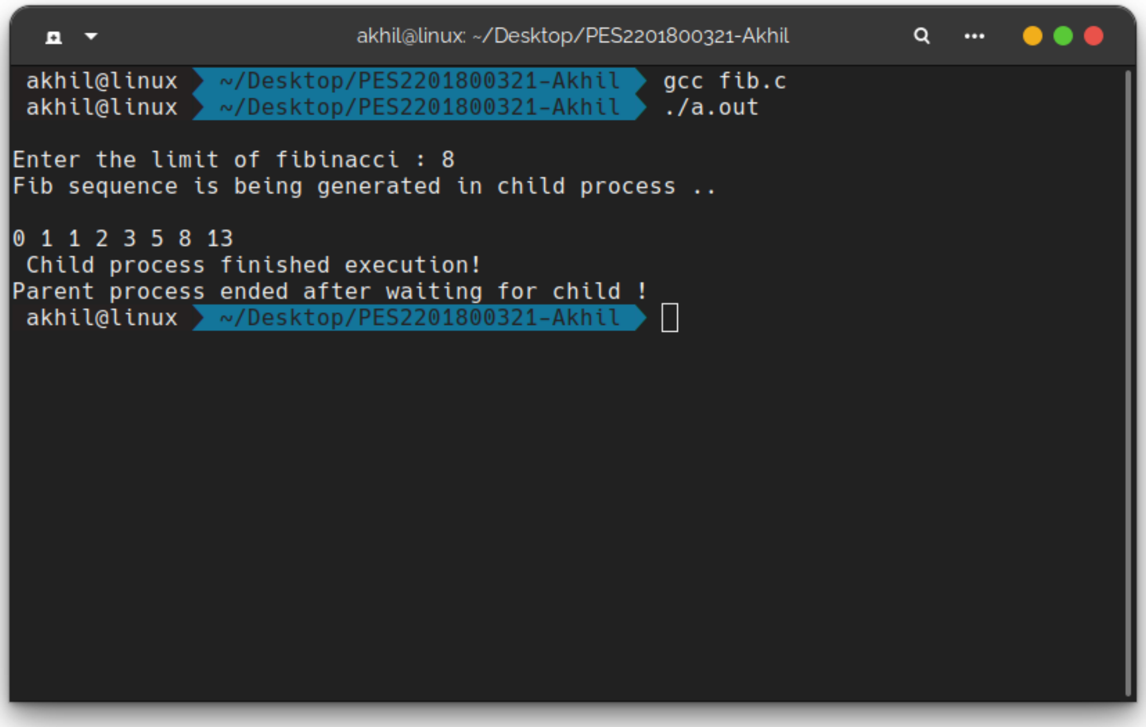
OS LAB WEEK -3

1. Print fibonacci sequence given the limit, using the child process.

OUTPUT:



akhil@linux: ~/Desktop/PES2201800321-Akhil

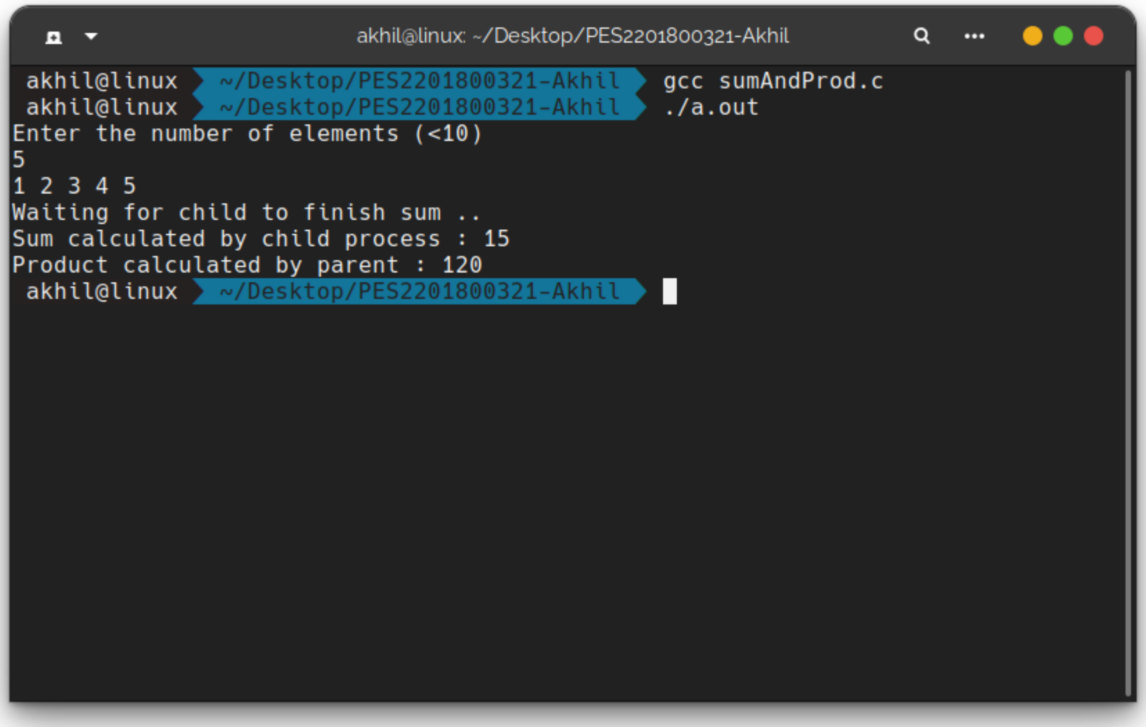
```
 akhil@linux  ~/Desktop/PES2201800321-Akhil  gcc fib.c
 akhil@linux  ~/Desktop/PES2201800321-Akhil  ./a.out

Enter the limit of fibinacci : 8
Fib sequence is being generated in child process ..

0 1 1 2 3 5 8 13
 Child process finished execution!
Parent process ended after waiting for child !
 akhil@linux  ~/Desktop/PES2201800321-Akhil  
```

2. Program to compute sum of array in the child process and product of array in parent process.

OUTPUT:

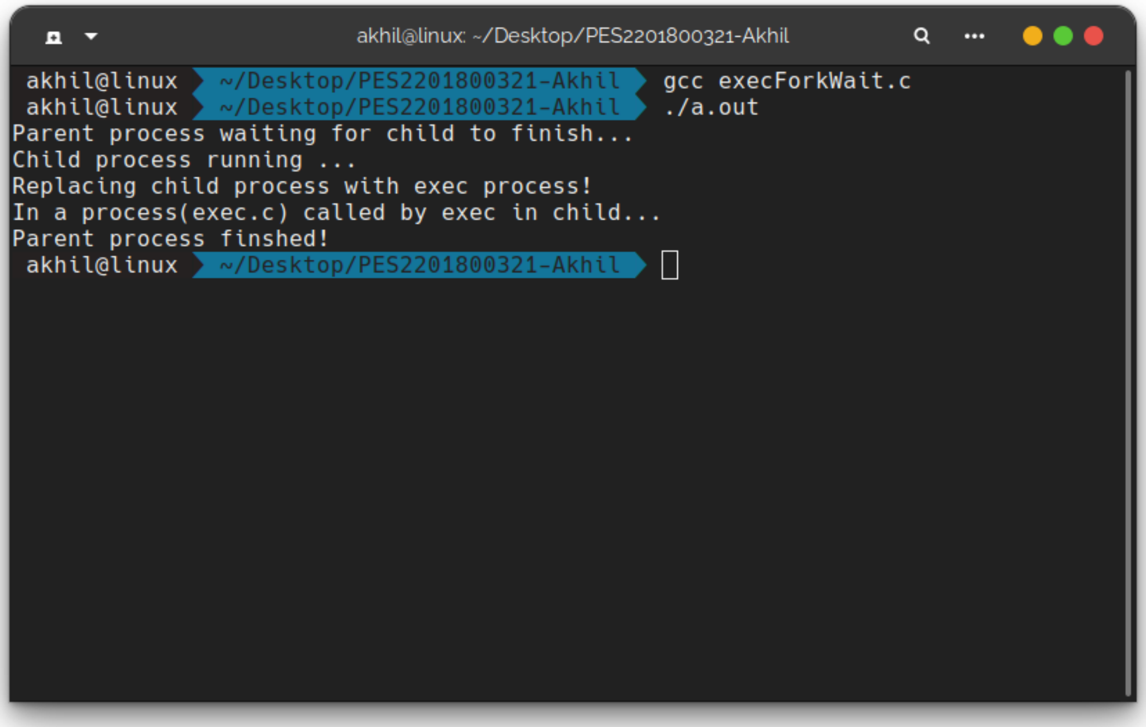


akhil@linux: ~/Desktop/PES2201800321-Akhil

```
 akhil@linux  ~/Desktop/PES2201800321-Akhil  gcc sumAndProd.c
 akhil@linux  ~/Desktop/PES2201800321-Akhil  ./a.out
Enter the number of elements (<10)
5
1 2 3 4 5
Waiting for child to finish sum ..
Sum calculated by child process : 15
Product calculated by parent : 120
 akhil@linux  ~/Desktop/PES2201800321-Akhil 
```

3. Write a program to show functionalities of fork(), exec family of functions and wait() in the same program.

OUTPUT:



```
 akhil@linux  ~/Desktop/PES2201800321-Akhil  gcc execForkWait.c
 akhil@linux  ~/Desktop/PES2201800321-Akhil  ./a.out
Parent process waiting for child to finish...
Child process running ...
Replacing child process with exec process!
In a process(exec.c) called by exec in child...
Parent process finshed!
 akhil@linux  ~/Desktop/PES2201800321-Akhil 
```

QUESTIONS AND ANSWERS

1. **What is the role of the init process on UNIX and Linux systems in regard to process termination?**

> 💡 When a process is terminated, it moves to the defunct state and remains in that state until the parent calls wait using wait() or its derivatives. However, if a parent does not invoke the wait function, the child process remains defunct as long as the parent process remains alive. Once the parent process terminates, the init process becomes the new parent of the defunct process. Periodically, the init process calls wait() which ultimately releases the pid.

2. **What is a subreaper process?**

💡 A subreaper fulfils the role of init for its descendant processes. Upon termination of a process that is orphaned and marked as having a subreaper, the nearest still living ancestor subreaper will receive a SIGCHLD signal and be able to wait on the process to discover its termination status. With the concept of subreapers, the user-space service manager now becomes the new parent of a child calling wait() continuously to avoid defunct process, instead of init.

3. **What causes a defunct process on the Linux system and how can you avoid it?**

💡 Defunct processes are processes that have terminated but they remain visible to the Linux operating system until the parent process reads it’s status. Once the status of the process has been read, the operating system removes it.
A parent process must always call wait (or one of its variants) on their child processes in order to let the kernel know that the terminated child can be cleaned up.

4. **How can you identify zombie processes on the Linux system?**

💡 Zombie processes can be found with the ps command. Within the ps output there is a STAT column which will show the processes current status, a zombie process will have Z as the status. In addition to the STAT column zombies commonly have the words <defunct> in the CMD column as well.

5. **What does child process inherit from its parent?**

💡 A child process inherits most of its attributes, ex: file descriptors, from its parent. In Linux, a child process is created as a copy of the parent, using the fork(). The child process can then differentiate itself with a different program (using exec or its types) as required.