Operating Systems Laboratory

WEEK 3: Process Creation and Termination

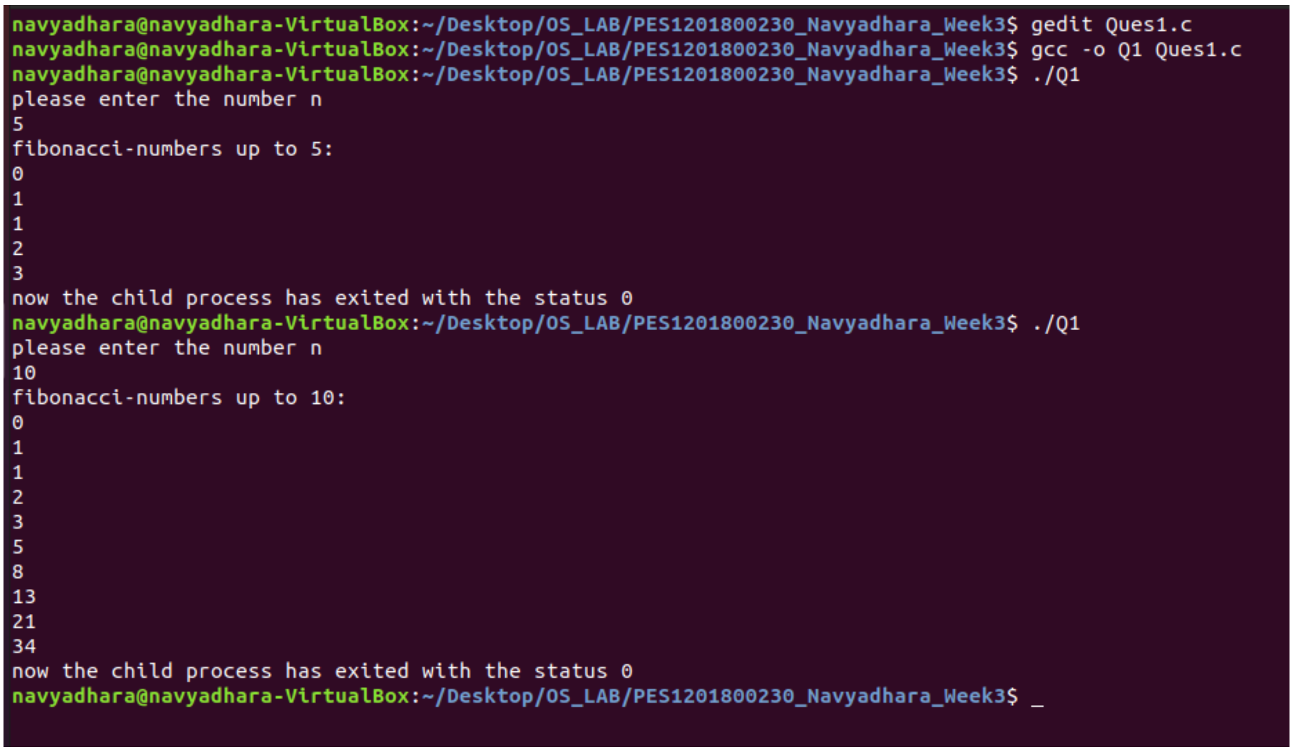
OBJECTIVE: Understanding Parent-Child Process relationship and the working of fork(),exec() and wait()

Name: Navyadhara Gana Sai G
SRN: PES1201800230
Sec: 5A

Basic Understanding Of the system calls:

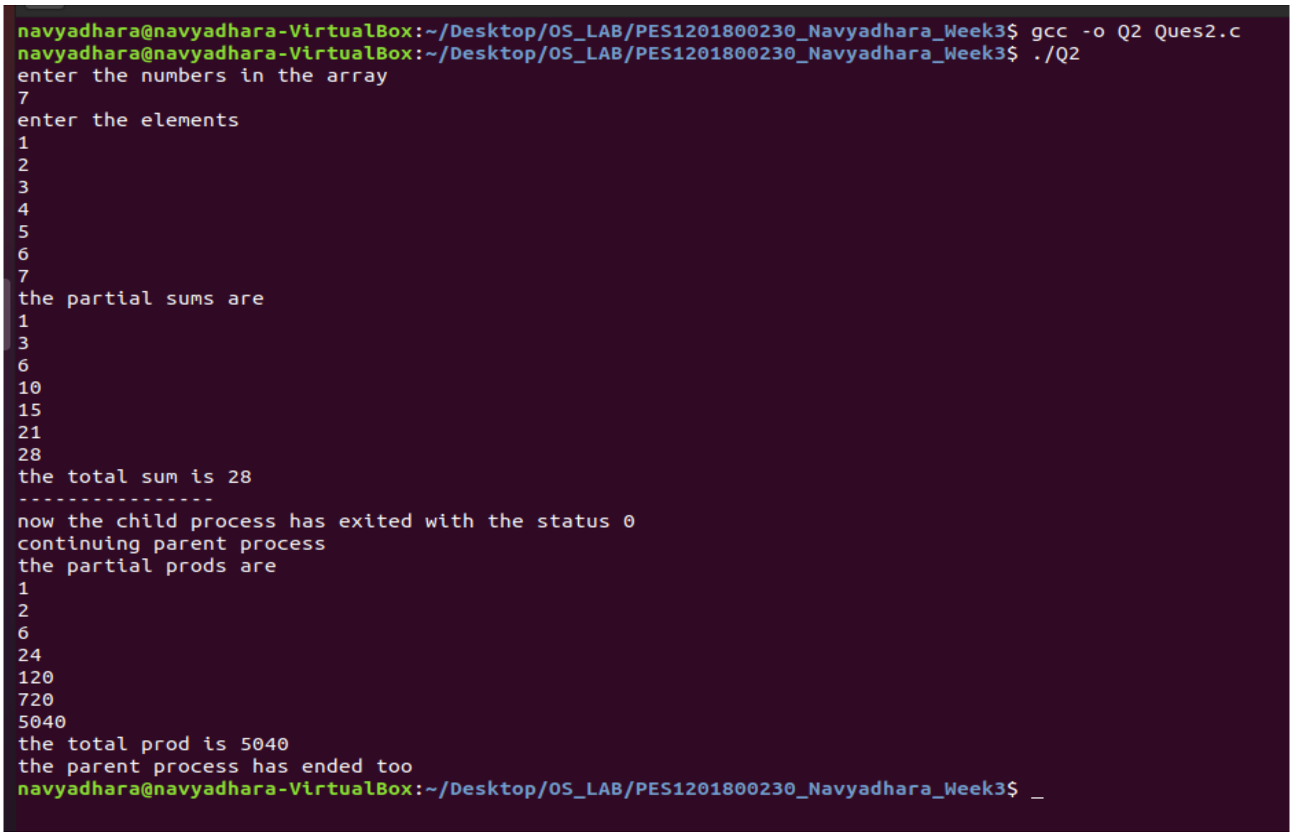
- **Fork(): System call fork() is used to create processes. It takes no arguments and returns a process ID. The purpose of fork() is to create a new process, which becomes the child process of the caller**
- **Exec() : Used to execute files residing in an active process. Using it will replace old files with the new from the process**
- **Wait(): It is responsible to call exit() and terminate processes. It takes an integer as input.**

Question 1: Fibonnaci Series



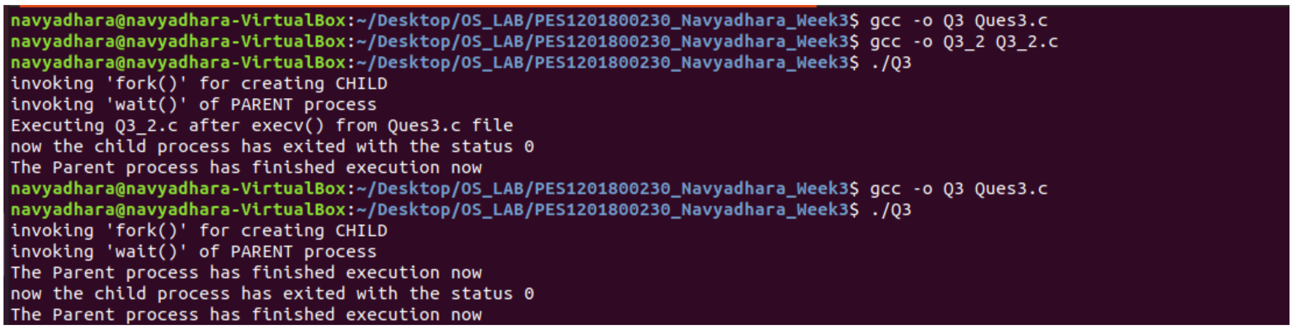
```
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ gedit Ques1.c
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ gcc -o Q1 Ques1.c
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ ./Q1
please enter the number n
5
fibonacci-numbers up to 5:
0
1
1
2
3
now the child process has exited with the status 0
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ ./Q1
please enter the number n
10
fibonacci-numbers up to 10:
0
1
1
2
3
5
8
13
21
34
now the child process has exited with the status 0
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ _
```

Question 2: Partial Sums and Partial Products



```
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ gcc -o Q2 Ques2.c
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ ./Q2
enter the numbers in the array
7
enter the elements
1
2
3
4
5
6
7
the partial sums are
1
3
6
10
15
21
28
the total sum is 28
----------------
now the child process has exited with the status 0
continuing parent process
the partial prods are
1
2
6
24
120
720
5040
the total prod is 5040
the parent process has ended too
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ _
```

Question 3: All three system calls in one place



```
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ gcc -o Q3 Ques3.c
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ gcc -o Q3_2 Q3_2.c
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ ./Q3
invoking 'fork()' for creating CHILD
invoking 'wait()' of PARENT process
Executing Q3_2.c after execv() from Ques3.c file
now the child process has exited with the status 0
The Parent process has finished execution now
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ gcc -o Q3 Ques3.c
navyadhara@navyadhara-VirtualBox:~/Desktop/OS_LAB/PES1201800230_Navyadhara_Week3$ ./Q3
invoking 'fork()' for creating CHILD
invoking 'wait()' of PARENT process
The Parent process has finished execution now
now the child process has exited with the status 0
The Parent process has finished execution now
```

Questions:

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

Ans) When a process is terminated, it briefly moves to the zombie state and remains in that state until the parent invokes a call to wait(). When this occurs, the process id as well as entry in the process table are both released. However, if a parent does not invoke wait(), the child process remains a zombie as long as the parent remains alive. Once the parent process terminates, the init process becomes the new parent of the zombie. Periodically, the init process calls wait() which ultimately releases the pid and entry in the process table of the zombie process.

2)What is a subreaper process?
Ans) A subreaper is an orphan process which becomes so after fulfilling of init(1) role of its descendent processes. A process is marked as a subreaper when its parent is terminated and =the nearest still living ancestor subreaper will receive a `SIGCHLD` signal and be able to `wait(2)` on the process to discover its termination status.

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

Ans) A "defunct" process (also called "zombie") is a process that is actually finished which depends on a parent process which for some reason has not accepted the knowledge that it is finished and should be terminated. The reason for processes to be listed as defunct is that something has gone wrong with the parent process.

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

Ans) **$ ps -elf | grep Z**
This command identifies the zombie processes.

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

Ans) A child process inherits most of its attributes, such as file descriptors, from its parent. In Unix, a child process is typically created as a copy of the parent, using the fork system call. The child process can then overlay itself with a different program (using exec) as required.