20CO24
Khan Arshad Abdulla 5/3/2022

EXPERIMENT NO-4

Aim: **Create a child process in Linux using the fork system call. From the child process obtain the process ID of both child and parent by using getpid and getppid system call.**

Theory:

What is a System Call?

A system call is a method for a computer program to request a service from the kernel of the operating system. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel.A system call is a way for a user program to interface with the operating system. The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request. A system call can be written in assembly language or a high-level language like C or C++

API(Application Program Interface) connects the operating system's functions to user programs. It acts as a link between the operating system and a process

System Calls can be grouped into major 5 categories:

Process Control:

Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

File Management:

File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

Device Management:

Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

Information Maintenance:

Information maintenance is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

20CO24
Khan Arshad Abdulla 5/3/2022

Communication:

Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

What is fork()?

The **fork ()** system calls aids in the creation of processes. When a process uses the fork() system call, it creates a replicate of itself. The parent process is the existing process, and the child process is the new process. Although, the child process is equivalent to the parent process. When creating the child process, the parent state like open files, address space, and variables are copied to the child process. In other words, the child and parent processes are located in separate physical address spaces. As a result, the modification in the parent process doesn't appear in the child process.

It is a command that allows a process to copy itself.
The parent and child processes are in separate address spaces in the fork().
There is a child process and a parent process after calling the fork() function.
The fork() makes a child's process equal to the parent's process.

Process	**Windows**	**Unix**
	CreateProcess()	Fork()
Process Control	ExitProcess()	Exit()
	WaitForSingleObject()	Wait()

Program:

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

  int x, y;

  int i = fork();

    if (i == 0)

    {
```

```
		printf("parent = %d, Child = %d\n", getppid(), getpid());
	}
	else if(i<0)
	{
		printf("Child process not created\n");
	}
	else
	{
		x = wait(NULL);
		y = waitpid(i,NULL,1);
		printf("parent = %d, GrandParent = %d\n", getpid(), getppid());
		printf("Child Process = %d Completed Execution\n", x);
		printf("GrandChild Process = %d Completed Execution\n", y);
	}
}
```

Output:

```
parent = 2216, Child = 2217
parent = 2216, GrandParent = 2214
Child Process = 2217 Completed Execution
GrandChild Process = -1 Completed Execution


------------------
(program exited with code: 0)
Press return to continue
```

20CO24
Khan Arshad Abdulla 5/3/2022

Conclusion:

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. After a new child process is created, both processes will execute the next instruction following the fork() system call. Therefore, we have to distinguish the parent from the child.