Fokane Sakshi Anil
TE-A-42

Assignment No: 10

UNIX System Calls

- Aim : Implemented UNIX System calls like for process management.

- problem Statement : To write a program to implement UNIX System calls like for process management.

- pre-requisite :- 1) Explain concept of System Calls.
  2) Explain State diagram working of new process.

- Software Requirements :-

Sno	facilities required	Quantity
1	System	1
2	O/S	Ubuntu Kylin
3	S/w name	Turbo or GCC

- Hardware requirements - NO.

- Objectives :- 1) To understand UNIX System Call. 2) To understand concept of process management.
  3) Implementation of some System call or O.S

- Theory : System Call :-
  when a program in user mode

require access to RAM or a hardware resource. This is done via something called a System call.
generally System Call are made by the User level program in the following situation:

- creating opening closing & deleting files in the file System.
- creating and managing new process
- creating a connection in the network, sending & receiving packets.
- Requesting access to a hardware device, like a mouse or a printer.

- Kernel Mode :-
When CPU is Kernel mode the code being executed can access any memory address and any hardware resource.
- In user mode, if any program crashes only that particular program is halted.
- That means the system will be in a Safe state even if a program in user mode crashes.
- Hence most programs in an O.S run in user mode.

- System Call Basics :
- Since System calls are function, we need to include the proper header files
- most System calls have a meaningful

return values.
a) Usually 1-or negative value indicate an error.
b) a specific error code is place in a global variable called error.

System calls for process
- Pid_t fork(void)
- int execl(char *name, char *ag(), (char *)0)
- Pid_t wait(int *status)
- void exit(int status)
- int kill(pid_t pid, int sig)

- Unix System Calls :
  - P.s command : The Ps command is used to provide information about the currently running process, including their process identification N°.

- Fork Command :
  The fork() system call is used to create process.

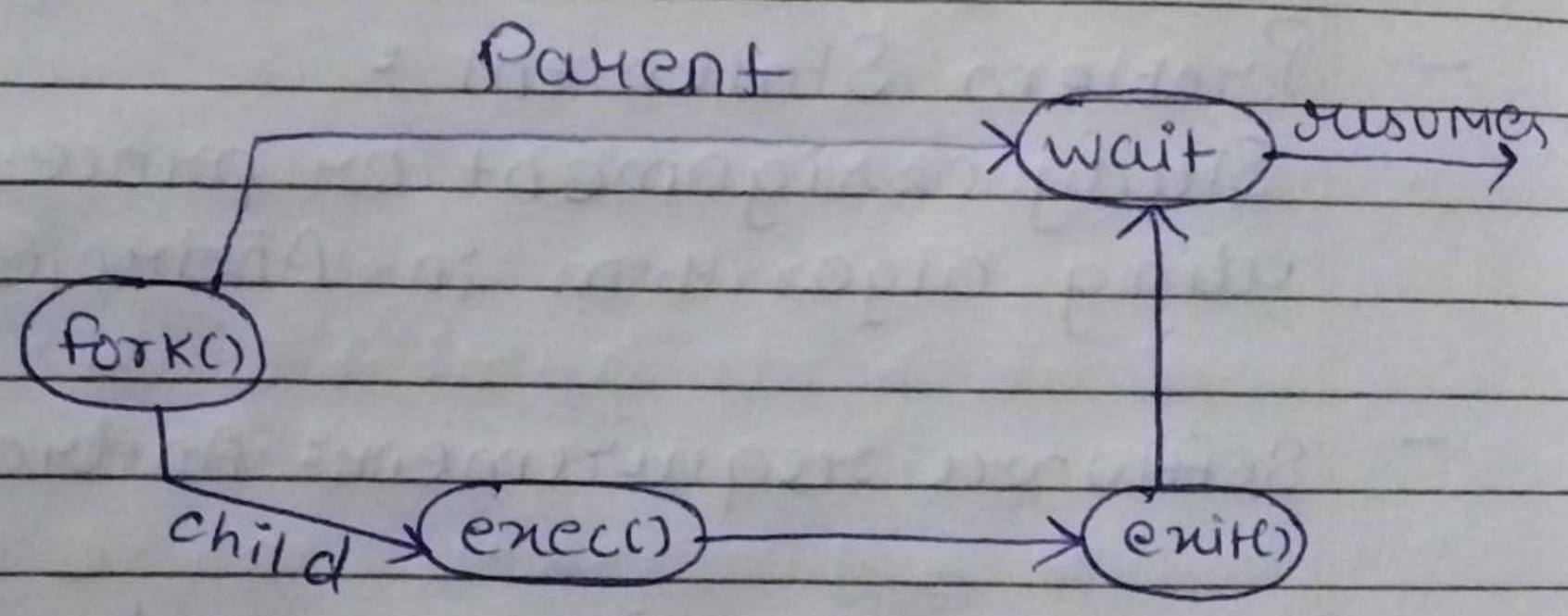
- join Command :
  The join command in UNIX is a command line utility for joining lines of two files on a common field.

- Execl() command :

The exec() system call is also used to create But there is one big difference between fork() & exec() calls.

- wait():

A call to wait() blocks the calling process until one of the its child process exits or a signal is received.



Conclusion :-

Thus, we have the process system call program & studied various system calls.

Assignment No. 10

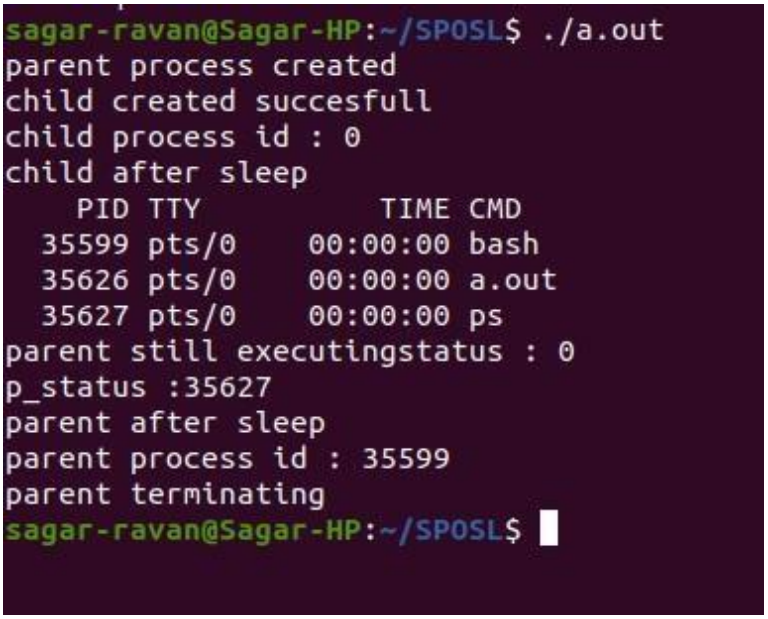
Name:-Fokane Sakshi Anil
Std-Div:-TE-A
Roll No:-42

1. Code:
Problem Statement : Write a C program to create a child process using fork system call. Display Status of running processes used in child process(EXEC) & terminate child process before completion of parent task(wait).

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


int main()
{ pid_t pid , ppid , p_status ;
        int status ;
        printf("parent process created \n");
        pid = fork();
        if(pid ==0)
        {
                printf("child created succesfull\n");
                printf("child process id : %d \n",
                pid); sleep(10); printf("child after
                sleep \n");
                execlp("/bin/ps","ps",NULL);

                printf("child terminating\n");
                exit(0);
        }
        else
        { printf("parent still executing"); p_status
                = wait(&status); printf("status :
                %d \n",status); printf("p_status
                :%d \n",p_status); sleep(10);
```

```
            printf("parent after sleep\n");
            ppid = getppid();
            printf("parent process id : %d\n",ppid);
            printf("parent terminating\n");
            exit(0);
        }

        return 0;
}
```

OUTPUT:



```
sagar-ravan@Sagar-HP:~/SPOSL$ ./a.out
parent process created
child created succesfull
child process id : 0
child after sleep
    PID TTY          TIME CMD
  35599 pts/0    00:00:00 bash
  35626 pts/0    00:00:00 a.out
  35627 pts/0    00:00:00 ps
parent still executingstatus : 0
p_status :35627
parent after sleep
parent process id : 35599
parent terminating
sagar-ravan@Sagar-HP:~/SPOSL$
```