Assignment No-10

Aim- Implement UNIX system calls like for process management.

Problem statement -
To write a program to implement UNIX system calls like for process management.

Pre-requisites -
① Explain concepts of system call.

② Explain state diagram working of new process

Software Requirements -

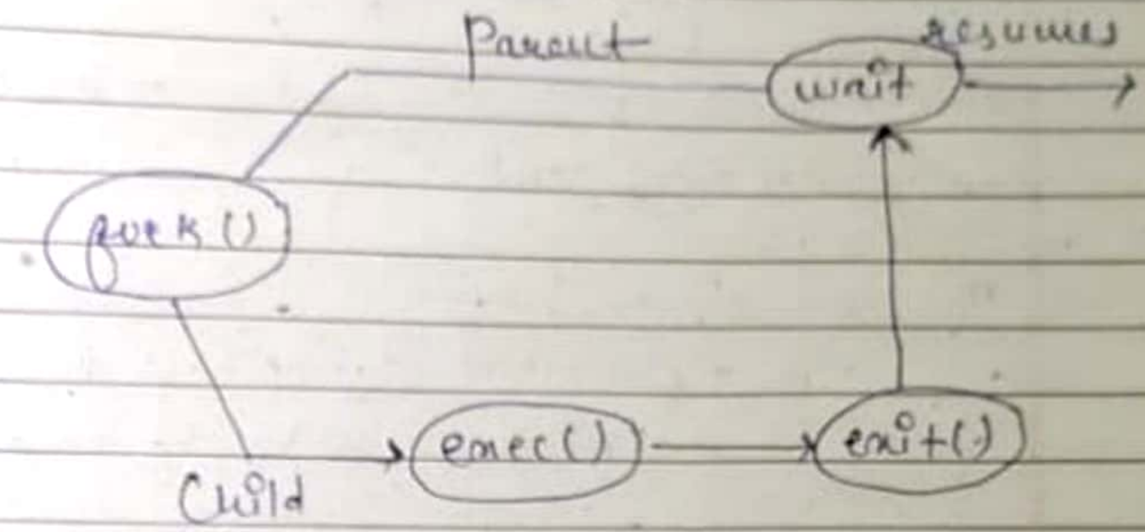
Sr No	facilities required	Quantity.
1	System	1
2	o/s	Ubuntu Kylin
3	S/w name	C Turbo or GCC

Hardware Requirements -
No

Objectives -
① To understand UNIX system call.
② To understand Concept of process management
③ Implementation of some system call of OS

UNIX system calls -

- PS command
- fork command.
- join command.
- exec() command.
- wait() command.



Conclusion -

Thus, the process system call program is implemented & studied various system calls.

Assignment No. 10 [UNIX System Calls]

Problem Satement: To write a program to implement UNIX system calls like for process Management.

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).

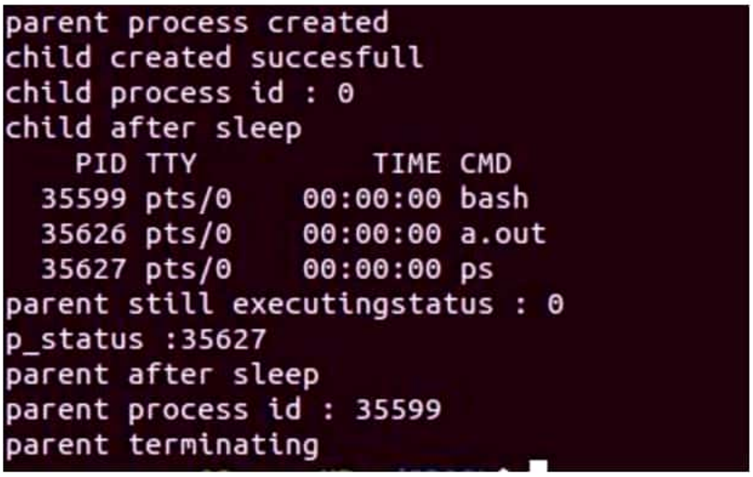
```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>


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:



```
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
```