

Ex.No:6	Implementation of system calls - fork, exec, getpid, exit, wait, close, stat, opendir, readdir.
----------------	---

AIM:

To write C Programs using the following system calls of UNIX operating system fork, exec, getpid, exit, wait, close, opendir and readdir.

1. PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEMS (OPENDIR, READDIR, CLOSEDIR)**ALGORITHM:**

- STEP 1: Start the program.
- STEP2: Create struct dirent.
- STEP3: declare the variable buff and pointer dptr.
- STEP4: Get the directory name as input
- STEP5: Open the directory.
- STEP6: Read the contents in directory and print it.
- STEP7: Close the directory and stop the program

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<dirent.h>
struct dirent *dptr;
int main(int argc, char *argv[])
{
    char buff[100]; DIR *dirp;
    printf("\n\n ENTER DIRECTORY NAME");
    scanf("%s", buff);
    if((dirp=open(dirp)) ==NULL)
    {
        printf("The given directory does not exist");
        exit(1);
    }
    while(dptr=readdir(dirp))
    {
        printf("%s\n",dptr->d_name);
    }
    closedir(dirp);
}
```

2. PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEM (fork, getpid, exit)

ALGORITHM:

STEP1:Start the program.
STEP 2: Declare the variables pid, pid1,pid2.
STEP3: Call fork() system call to create process.
STEP4:If pid== -1, exit.
STEP5: If pid!= -1, get the processid using getpid().
STEP6: Print the process id.
STEP7:Stop the program

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
void main()
{
int pid,pid1,pid2;
pid=fork();
if(pid== -1)
{
printf("ERROR IN PROCESS CREATION \n");
exit(1);
}
if(pid!=0)
{
pid1=getpid();
printf("\n the parent process ID is %d\n", pid1);
}
else
{
pid2=getpid();
printf("\n the child process ID is %d\n", pid2);
}
}
```

Result: Thus the C program to implement system calls of fork, exec, wait, close, opendir and readdir have been executed successfully.

Additional Programs

1. Display the contents in Child and Parent Process
2. Get the PID value of Parent by the Child process and PID value of Child by the Parent process

Display the contents in Child and Parent Process

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
char string1[] = "\n Hello";
char string2[] = "\n Welcome";
```

```

int main(void)
{
pid_t PID;
PID = fork();
if(PID==0) /* Child Process*/
printf("%s", string2);
else /* Parent Process*/
printf("%s", string1);
exit(0); /* Executed by both processes*/
}

```

2. Get the PID value of Parent by the Child process and PID value of Child by the Parent process

```

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
void main()
{
int childpid;
if(childpid=fork()) == -1
{
    printf("\n can't fork");
    exit(0);
}
else if(childpid==0) /*child process*/
{
    printf("\n Child: Child Pid=%d, Parent Pid =%d\n" , getpid(),getppid());
    exit(0);
}
else
{
/* Parent Process*/
printf("\n Parent: Child Pid=%d, Parent Pid =%d\n" , childpid, getpid());
exit(0);
}
}

```