# PRACTICAL – 5

**Aim:** **Write programs using the following system calls of UNIX operating system: fork, exec, getpid, exit, wait, stat, readdir, opendir.**

**A. Write a program to execute fork () and find out the process id by getpid() system call.**

**B. Write a program to execute following system call fork (), execl(), getpid(), exit(), wait() for a process.**

**C. Write a program to find out status of named file (program of working stat () system cal**

**A:** **Write a program to execute fork () and find out the process id by getpid() system call.**

**PROGRAM:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

void main()

{

int i;

printf("fork: pid=%d\n",getpid());

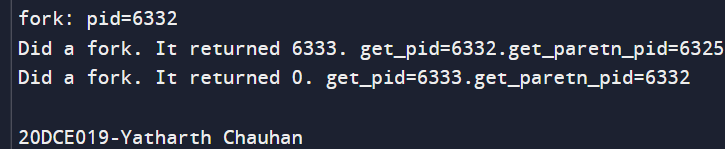
i=fork();

printf("Did a fork. It returned %d. get\_pid=%d.get\_paretn\_pid=%d\n",i,getpid(),getppid());

printf("\n20DCE019-Yatharth Chauhan\n");

}

**OUTPUT:**

****

**B. Write a program to execute following system call fork (), execl(), getpid(), exit(), wait() for a process.**

**PROGRAM:**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <sys/wait.h>

int main()

{

pid\_t pid;

int ret = 1;

int status;

pid = fork();

if (pid == -1)

{

printf("can't fork, error occurred\n");

exit(EXIT\_FAILURE);

}

else if (pid == 0)

{

printf("child process, pid = %u\n", getpid());

printf("parent of child process, pid = %u\n", getppid());

char \*argv\_list[] = {"ls", "-lart", "/home", NULL};

execv("ls", argv\_list);

exit(0);

}

else

{

printf("Parent Of parent process, pid = %u\n", getppid());

printf("parent process, pid = %u\n", getpid());

if (waitpid(pid, &status, 0) > 0)

{

if (WIFEXITED(status) && !WEXITSTATUS(status))

printf("program execution successful\n");

else if (WIFEXITED(status) && WEXITSTATUS(status))

{

if (WEXITSTATUS(status) == 127)

{

printf("execv failed\n");

}

else

printf("program terminated normally,"

" but returned a non-zero status\n");

}

else

printf("program didn't terminate normally\n");

}

else

{

printf("waitpid() failed\n");

}

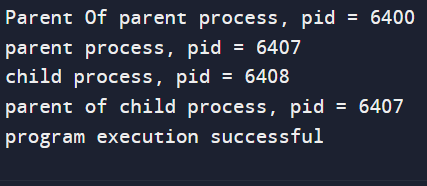
exit(0);

}

return 0;

}

**OUTPUT:**

****

**For Wait:**

#include<stdio.h>

#include<sys/wait.h>

#include<unistd.h>

int main()

{

if(fork()==0)

{

printf("get\_pid=%d.get\_parent\_pid=%d\n",getpid(),getppid());

printf("HC: hello from child\n");

}

else

{

wait(NULL);

printf("get\_pid=%d .get\_parent\_pid=%d\n",getpid(),getppid());

printf("HP: hello from parent\n");

printf("CT: child has terminated\n");

}

printf("get\_pid=%d.get\_parent\_pid=%d\n",getpid(),getppid());

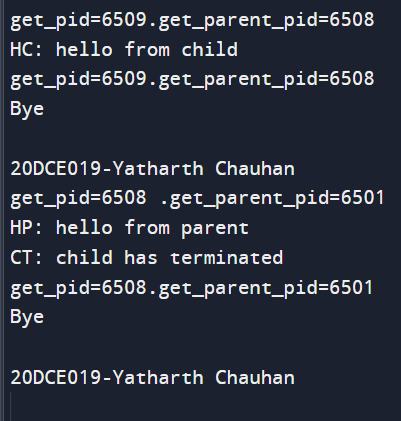
printf("Bye\n");

printf("\n20DCE019-Yatharth Chauhan\n");

return 0;

}

**OUTPUT:**

****

**C:** **Write a program to find out status of named file (program of working stat () system call.**

**PROGRAM:**

#include<stdio.h>

#include<sys/stat.h>

int main()

{

struct stat sofile;

stat("p5\_3.c" &sofile);

printf("st\_atime=%ldin" sofile.st\_atime);

printf(“st\_uid=%u\n" sofile.st\_uid);

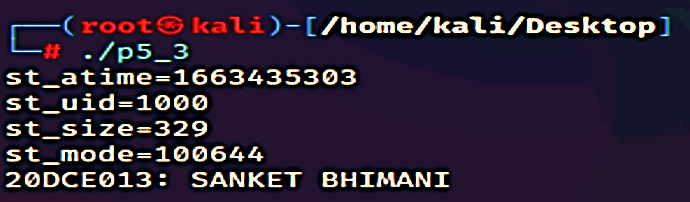
printf( "st\_size=%ld\n sofile.st\_size);

printf(“st\_mode=%o\n” sofile.st\_mode);

return 0;

}

**OUTPUT:**



**CONCLUSION:** In this practical we learnt program maintenance using make utility. We created a program spread over 4 files and maintained it using make utility.