CSE2005

Operating Systems

Lab CAT

Name: Ripunjay Narula

Registration No.: 19BCE0470

Software: Ubuntu for Windows

Q4. Parent child process creation using fork( ) and exec() system call

Checking the Process Identifier

Assigning new task to child

Providing the path name and program name to exec()

Synchronizing Parent and child process using wait()

CODE:

labcat.c:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
fork();
printf("Finding PID\n");
printf(" PID: %d\n",getpid());
return 0;
```

```
}
```

labcatf1.c:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
printf("PID of labcatf1.c = %d\n",getpid());
char*args[]= {"Operating Systems",NULL};
execv("./file2",args);
return 0;
}
```

labcatf2.c:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(){
printf(" In labcatf2.c\n");
printf("PID of labcatf2.c : %d \n",getpid());
return 0;
```

}

labcatf3.c [With Wait()]:

```
#include<stdio.h>
#include<sys/types.h>
#include <sys/wait.h>
#include<unistd.h>
int main(int argc , char *argv[]){
        int id= fork();
        int n,i;
                if(id==0){
                printf("\nParent Process: %d\n",getpid());
                                n=1; }
                                        else{
                                        printf("\n Child Process: %d
\n",getpid());
                                                        n=5; }
                                            printf("\n");
                                            if(id!=0){
                                            pid_t wait(int *id); }

for(i=n;i<n+5;i++){

printf("%d ",n); }

if(id!=0){
```

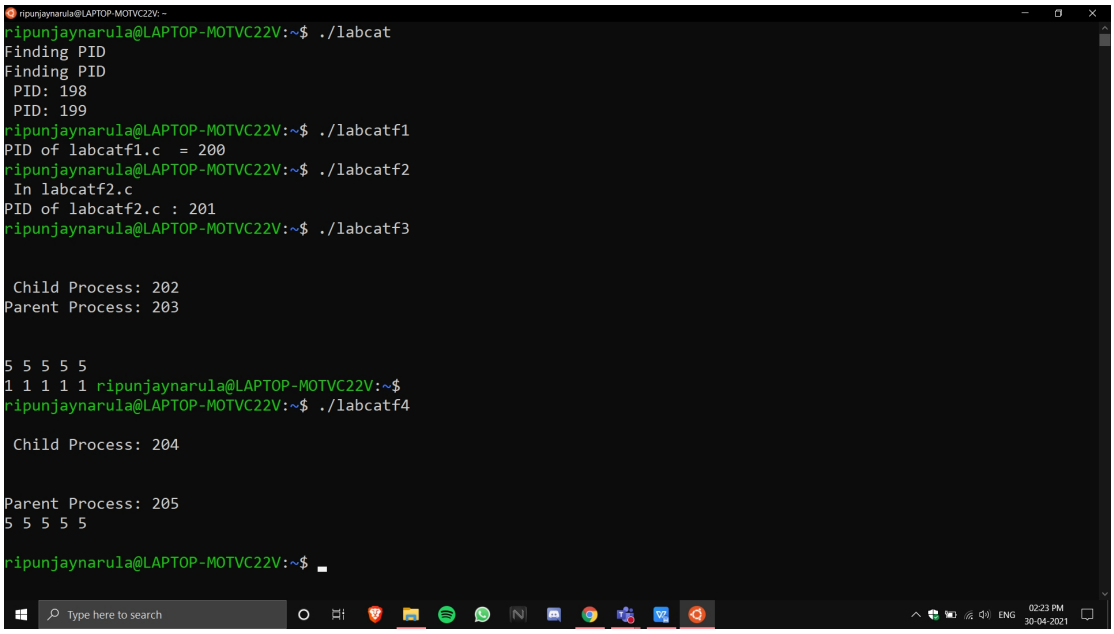
```
printf("\n"); }

return 0; }
```

labcatf4.c[Without Wait()]:

```
#include<stdio.h>
#include<sys/types.h>
#include <sys/wait.h>
#include<unistd.h>
int main(int argc , char *argv[]){
        int id= fork();
        int n,i;
                if(id==0){
                printf("\nParent Process: %d\n",getpid());
                                n=1; }
                                        else{
                                        printf("\n Child Process: %d
\n",getpid());
                                                        n=5; }
                                                    printf("\n");


for(i=n;i<n+5;i++){

printf("%d ",n); }

if(id!=0){

printf("\n"); }

return 0;
```

}

OUTPUT:


ripunjaynarula@LAPTOP-MOTVC22V: ~
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./labcat
Finding PID
Finding PID
PID: 198
PID: 199
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./labcatf1
PID of labcatf1.c = 200
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./labcatf2
In labcatf2.c
PID of labcatf2.c : 201
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./labcatf3
Child Process: 202
Parent Process: 203
5 5 5 5 5
1 1 1 1 1 ripunjaynarula@LAPTOP-MOTVC22V:~$
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./labcatf4
Child Process: 204
Parent Process: 205
5 5 5 5 5
ripunjaynarula@LAPTOP-MOTVC22V:~$
Type here to search
ENG
02:23 PM
30-04-2021