**Tina Borundia**

**C1-16**

**Practical 3**



**examplePrac3.c :**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

int main(int argc,char \*argv[])

{

printf("PID of examplePrac3.c = %d\n",getpid());

char \*args[]={"Hello","C","Programming",NULL};

execv("./hello",args);

printf("back to examplePrac3.c\n");

return 0;

}

**hello.c**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

int main(int argc,char \*argv[])

{

printf("we are in hello.c\n");

printf("PID of hello.c = %d\n",getpid());

return 0;

}

**Output :**

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc examplePrac3.c

rcoem@rcoem-Vostro-3910:~/c1tina$ ./a.out

PID of examplePrac3.c = 4417

back to examplePrac3.c

**excel.c:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

void main(){

char \*temp,\*temp1,\*temp2;

temp="hello";

temp1="Funny";

temp2="world";

printf("PID of execl.c = %d\n",getpid());

execl("hello",temp,temp1,temp,NULL);

printf("Error");

}

**hello.c :**

#include<stdio.h>

void main(int argc,char \*argv[],char \*envp[])

{

printf("Filename: %s\n",argv[0]);

printf("%s %s\n",argv[1],argv[2]);

}

**Output :**

rcoem@rcoem-Vostro-3910:~/Desktop/Excel$ gcc excel.c -o excel

rcoem@rcoem-Vostro-3910:~/Desktop/Excel$ ./excel

Filename: hello

Funny hello

**execvp.c :**

#include<stdio.h>

#include<unistd.h>

int main()

{

if(fork()==0)

{

char \*args[3]={"ls","-l",NULL};

execvp(args[0],args);

//Does not get here

}

else

{

printf("Parent!\n");

}

printf("Only parent!\n");

return 0;

}

**Output :**

rcoem@rcoem-Vostro-3910:~/Desktop/Excel$ gcc execvp.c -o execvp

rcoem@rcoem-Vostro-3910:~/Desktop/Excel$ ./execvp

Parent!

Only parent!

**fork6.c\_\_\_\_\_\_\_\_\_Zombiee Process:**

#include<stdio.h>

#include<sys/wait.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

pid\_t ret\_value;

printf("The process ID is %d\n",getpid());

ret\_value=fork();

if(ret\_value<0){

// Fork system call failed

printf("fork fails");

}

else if(ret\_value==0){

// child process

printf("Child Process\n");

printf("Process ID is %d\n",getpid());

sleep(10);

}

else{

// Parent process

//wait(NULL);

printf("Parent Process\n");

printf("Process ID is %d\n",getppid());

sleep(20);

}

}

**Output :**

rcoem@rcoem-Vostro-3910:~/c1tina$ ps -a1

PID TTY STAT TIME COMMAND

1 ? Ss 0:01 /sbin/init splash

1683 tty2 Ssl+ 0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME\_

1685 tty2 Sl+ 1:49 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user

1694 tty2 Sl+ 0:00 /usr/libexec/gnome-session-binary --systemd --syste

4224 pts/1 Ss 0:00 bash

4271 pts/0 Ss 0:00 bash

4284 pts/0 S+ 0:00 ./a.out

4285 pts/0 Z+ 0:00 [a.out] <defunct>

4287 pts/1 R+ 0:00 ps -a1

**orp.c :**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

int main()

{

//fork() create a child process

int pid=fork();

if(pid>0)

{

//getpid() return process id

// while getppid() will return parent process id

printf("Parent process\n");

printf("ID : %d\n",getpid());

}

else if(pid==0)

{

printf("Child Process\n");

//getpid() will return process id of child process

printf("ID : %d\n",getpid());

// getpid() will return process id of child process

// printf("parent -ID : %d\n\n",getpid());

sleep(10);

// At this time parent process has finished

// So if u will check parent process id

//It will show different process id

printf("\nChild Process\n");

printf("ID: %d\n",getpid());

printf("parent -ID : %d\n",getppid());

}

else

{

printf("failed to create the child process");

}

return 0;

}

**Output** :

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc orp2.c

rcoem@rcoem-Vostro-3910:~/c1tina$ ./a.out

Parent process

ID : 4547

Child Process

ID : 4548

rcoem@rcoem-Vostro-3910:~/c1tina$

Child Process

ID: 4548

parent -ID : 1573

**vfork.c :**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

void main()

{

int n=10;

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

pid\_t pidv =vfork(); //creating child class

if(pidv==0)

{

//child Process

printf("\t This is child \n");

printf("\t\tPID(child): %d\n",getpid());

//sleep(3);

exit(0);

}

else if(pidv>0)

{

//Parent process

printf("\t This is Parent \n");

printf("\t\tPID(parent): %d\n",getpid());

}

else

{

printf("Fork error");

// exit(ERROR\_FAILURE);

}

printf("value of n : %d\n",n);

//return 0;

}

**Output:**

rcoem@rcoem-Vostro-3910:~/c1tina$ gedit vfork.c

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc vfork.c

rcoem@rcoem-Vostro-3910:~/c1tina$ ./a.out

Before fork pid : 4505

This is child

PID(child): 4506

This is Parent

PID(parent): 4505

value of n : 10