**Tina Borundia**

**C1-16**

**Practical 3**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Fork1.c** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code :

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

int main(void)

{

printf("Hello \n");

fork(); // Generate multiple copies

printf("bye\n");

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

return 0;

}

Output:

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

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

Hello

bye

4023

bye

4024

**Fork2.c** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code :

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<stdlib.h>

int main(void)

{

// Make two process which run same

// program after this instruction

// p==0 ----> child

// p>0 ----> parent

// p<0 ----> Paging / error

pid\_t p = fork();

if(p<0){

perror("fork fail");

exit(1);

}

printf("Hello World!,child process\_id(pid)=%d\n",getpid());

printf("Hello World!,parent process\_id(pid)=%d\n",getppid());

return 0;

}

Output:

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc fork2.c

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

Hello World!,child process\_id(pid)=4094

Hello World!,parent process\_id(pid)=2443

Hello World!,child process\_id(pid)=4095

Hello World!,parent process\_id(pid)=1556

**Fork3.c** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code :

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

int main(void)

{

fork();

fork();

fork();

printf("Hello \n");

return 0;

}

Output:

rcoem@rcoem-Vostro-3910:~/c1tina$ gedit fork3.c

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc fork3.c

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

Hello

Hello

Hello

Hello

Hello

Hello

Hello

Hello

**Fork4.c** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code :

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

void forkexample()

{

pid\_t p;

p=fork();

if(p<0){

perror("fork fails");

exit(1);

}

//child process because return value zero

else if(p==0){

printf("Hello from Child!\n");

}

// parent process because return value non-zero

else{

printf("Hello from Parent!\n");

}

}

int main(){

forkexample();

return 0;

}

Output:

rcoem@rcoem-Vostro-3910:~/c1tina$ gedit fork4.c

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc fork4.c

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

Hello from Parent!

Hello from Child!

**Fork5.c** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code :

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

void forkexample()

{

int x=1;

pid\_t p=fork();

if(p<0){

perror("fork fails");

exit(1);

}

else if(p==0){

printf("Child has x = %d\n",++x);

}

else{

printf("Parent has x = %d\n",--x);

}

}

int main(){

forkexample();

return 0;

}

Output:

rcoem@rcoem-Vostro-3910:~/c1tina$ gedit fork5.c

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc fork5.c

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

Parent has x = 0

Child has x = 2

**Fork5.c** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code :

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

void forkexample()

{

int x=1;

pid\_t p=fork();

if(p<0){

perror("fork fails");

exit(1);

}

else if(p==0){

printf("Child has x = %d\n",++x);

}

else{

printf("Parent has x = %d\n",--x);

}

}

int main(){

forkexample();

return 0;

}

Output:

rcoem@rcoem-Vostro-3910:~/c1tina$ gedit fork5.c

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc fork5.c

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

Parent has x = 0

Child has x = 2

**Fork6.c** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code :

#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(5);

}

else{

// Parent process

wait(NULL);

printf("Parent Process\n");

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

sleep(5);

}

}

Output:

rcoem@rcoem-Vostro-3910:~/c1tina$ gedit fork6.c

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc fork6.c

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

The process ID is 4461

Child Process

Process ID is 4462

Parent Process

Process ID is 4461

**Fork 7.c** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code :**

#include<stdio.h>

#include<sys/wait.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

int n,a,b;

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 and parent id is %d\n",getpid(),getppid());

printf("Enter the number to check EVEN or ODD\n");

scanf("%d",&n);

if(n%2==0){

printf("Number %d is EVEN\n",n);

}

else{

printf("Number %d is ODD\n",n);

}

sleep(5);

}

else{

// Parent process

wait(NULL);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Parent Process\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

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

printf("Enter two Number to check greatest\n");

scanf("%d%d",&a,&b);

if(a>b){

printf("%d is greatest than %d\n",a,b);

}

else{

printf("%d is greatest than %d\n",b,a);

}

sleep(5);

}

return 0;

}

**Output :**

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc fork7.c

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

The process ID is 4663

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Child Process\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Process ID is 4664 and parent id is 4663

Enter the number to check EVEN or ODD

9

Number 9 is ODD

6

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Parent Process\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Process ID is 4663

Enter two Number to check greatest

7 9

7 is greatest than 6

**Fork8.c** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code:**

#include<stdio.h>

#include<sys/wait.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

int n,key,idx,a;

int c=0,d=0;

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 and parent id is %d\n",getpid(),getppid());

printf("Enter the number of element in array\n");

scanf("%d",&n);

int A[n];

printf("Enter element in array\n");

for(int i=0;i<n;i++){

scanf("%d",&A[i]);

}

printf("Enter the key\n");

scanf("%d",&key);

for(int i=0;i<n;i++){

if(A[i]==key){

idx=i;

c++;

break;

}

}

if(c==0){

printf("Element not found");

}

else{

printf("Element found at index %d",idx);

}

sleep(10);

}

else{

// Parent process

wait(NULL);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Parent Process\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

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

printf("Enter a Number to check prime\n");

scanf("%d",&a);

for(int i=2;i<a;i++){

if(i%2==0){

d++;

break;

}

}

if(d>0){

printf("Prime");

}

else{

printf(" Not Prime");

}

sleep(5);

}

return 0;

}

**Output :**

rcoem@rcoem-Vostro-3910:~/c1tina$ gedit fork8.c

rcoem@rcoem-Vostro-3910:~/c1tina$ gcc fork8.c

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

The process ID is 4814

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Child Process\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Process ID is 4815 and parent id is 4814

Enter the number of element in array

6

Enter element in array

5 7 3 8 6 9

Enter the key

8

Element found at index 3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Parent Process\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Process ID is 4814

Enter a Number to check prime

7

Prime