OS_Lab4

K. Ram Mohan

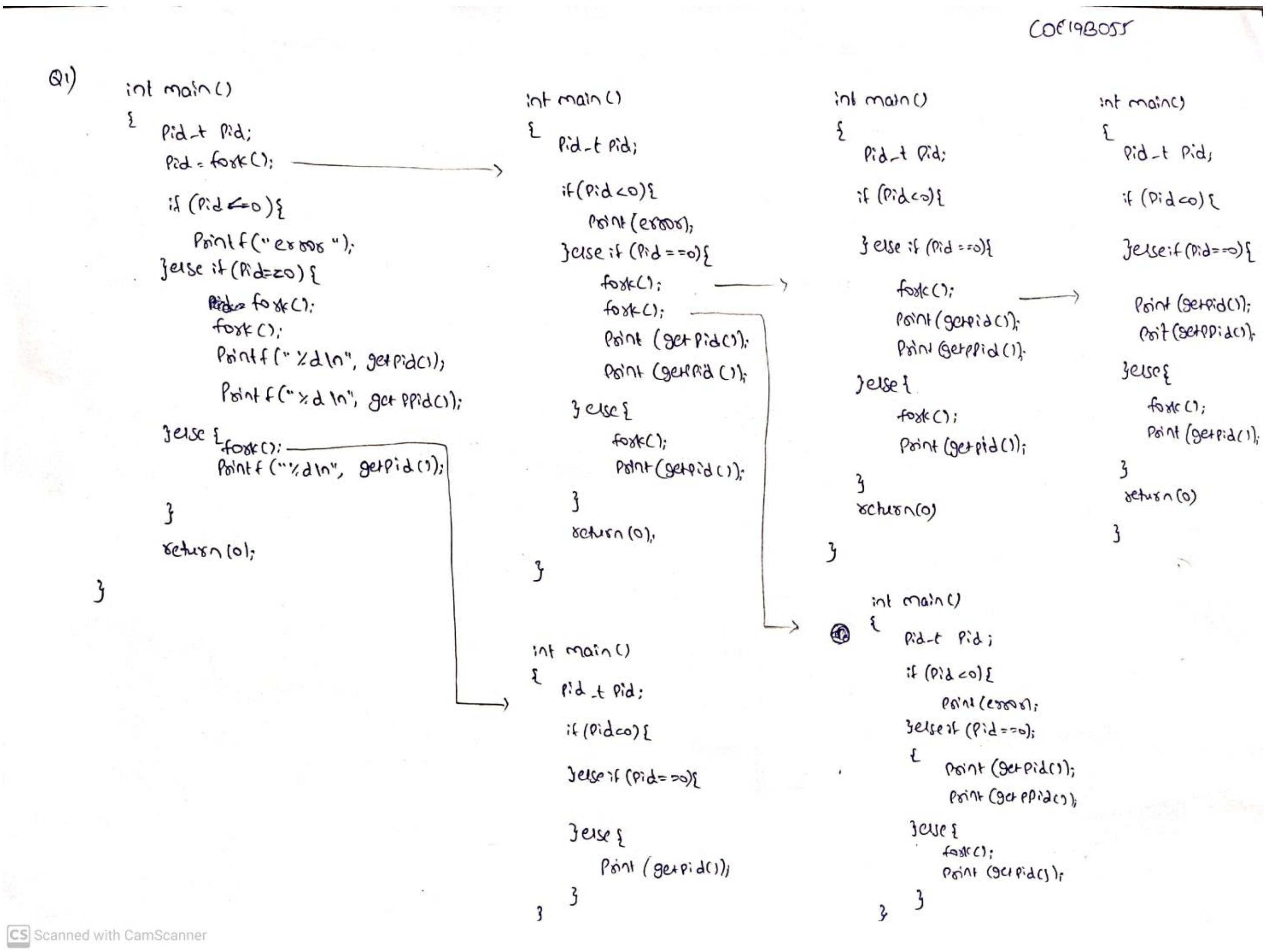
COE19B055

Q1)

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>


int main()

{

   pid_t pid;

   pid = fork();


   if(pid<0){

       printf("Error in fork call");

   }else if (pid==0)

   {

       pid = fork();

       pid = fork();

       printf("Process id of me is %d\n", getpid());

       printf("Process id of my parent is %d\n", getppid());

   }else if(pid>0){

       pid = fork();

       printf("Process id of mw is %d\n", getpid());

   }

   return(0);

}
```

COE19B055

Q1)



```
int main()
{   Pid_t Pid;
    Pid = fork();

    if (Pid <=0){
        Printf("error");
    }else if(Pid==0){
        ~~Pid=~~ fork();
        fork();
        Printf("%d\n", getPid());
        Printf("%d\n", get PPid());
    }else {fork();
        Printf("%d\n", getPid());

    }
    return(0);
}
```

```
int main()
{   Pid_t Pid;

    if(Pid<0){
        Print(error);
    }else if (Pid==0){
        fork();
        fork();
        Print (get Pid());
        Print (getPPid ());
    }else{
        fork();
        Print(getPid());
    }
    return(0);
}
```

```
int main()
{
    Pid_t Pid;

    if (Pid<0){

    }else if (Pid ==0){

        fork();
        Print(getPid());
        Print (getPPid());
    }else{
        fork();
        Print (getpid());
    }
    return(0)
}
```

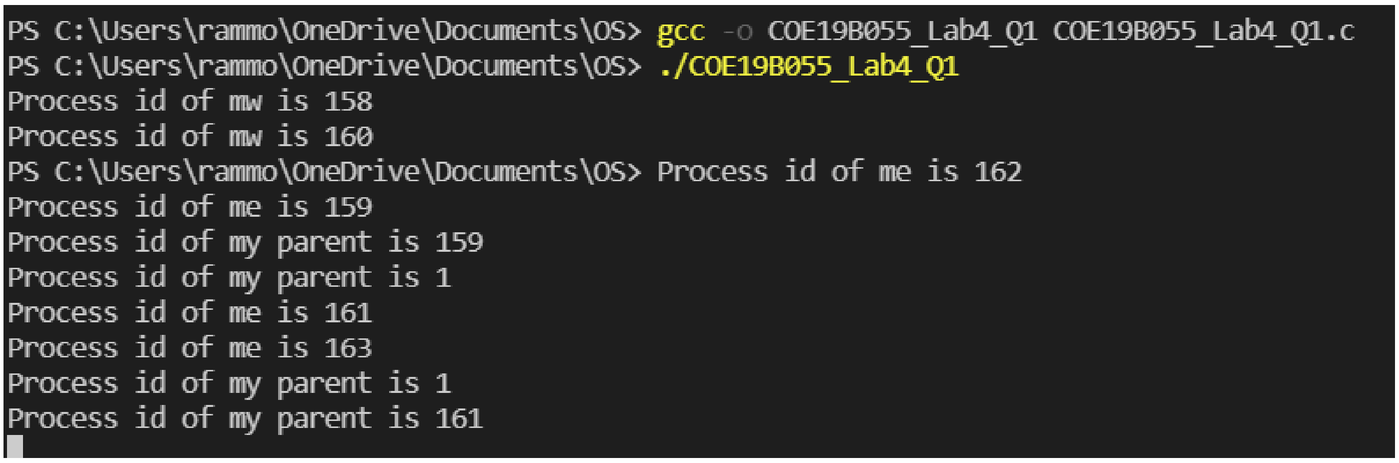
```
int main()
{
    Pid_t Pid;

    if (Pid<0){

    }elseif(Pid==0){

        Print (getPid());
        Prit (getPPid());
    }else{
        fork();
        Print (getPid();
    }
    return(0)
}
```

```
int main()
{   Pid_t Pid;

    if(Pid<0){

    }else if(Pid==0){

    }else{
        Print (getPid());
    }
}
```

```
int main()
{   Pid_t Pid;
    if (Pid<0){
        Print(error);
    }else if (Pid==0);
    {
        Print (getPid());
        Print (getPPid());
    }else{
        fork();
        Print (getPid());
    }
}
```

Scanned with CamScanner

Output:



```
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q1 COE19B055_Lab4_Q1.c
PS C:\Users\rammo\OneDrive\Documents\OS> ./COE19B055_Lab4_Q1
Process id of mw is 158
Process id of mw is 160
PS C:\Users\rammo\OneDrive\Documents\OS> Process id of me is 162
Process id of me is 159
Process id of my parent is 159
Process id of my parent is 1
Process id of me is 161
Process id of me is 163
Process id of my parent is 1
Process id of my parent is 161
```

Q2)

FILE1

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>


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

{

   char* args[] = {NULL};

   execv("./COE19B055_Lab4_Q2_1", args);

}
```

FILE2

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>

#include <sys/tree.h>


int main()

{

   printf("In COE19B055_Lab4_Q2_1\n");

   execlp("ls", "ls", NULL);

   execlp("pstree", "pstree", NULL);

}
```

COE19B055

Q2) To display the content of ls and pstree we can use command " execlp() " .

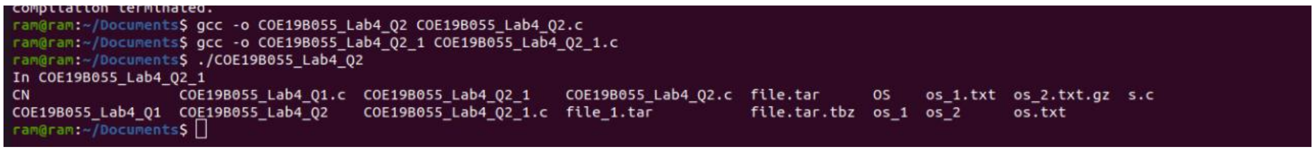
Synatx:-

execlp ("ls", "ls" ,NULL);
execlp ("pstree", "pstree", NULL);

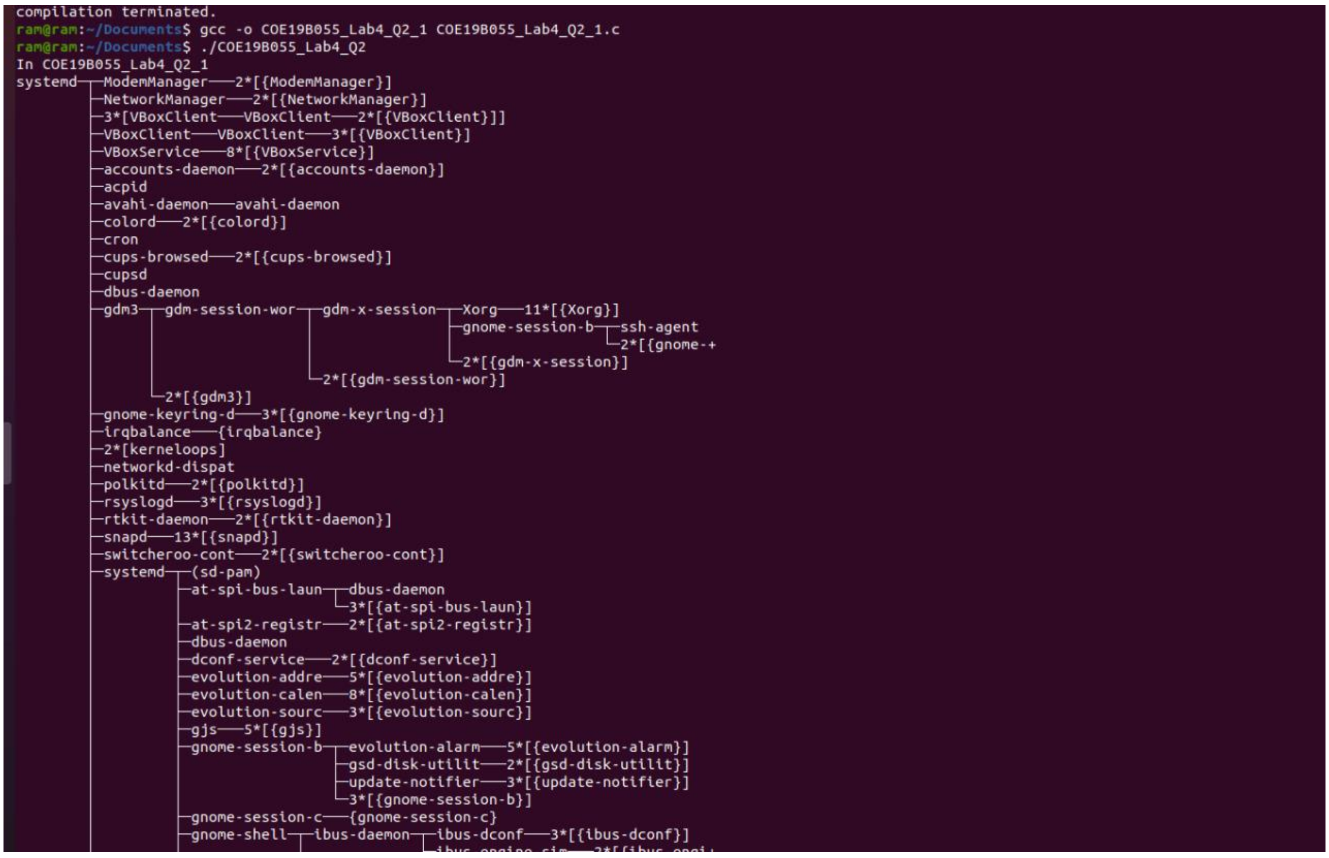
We can call only one to display content either 'ls' or 'pstree' at a time.

From file1 we will call file 2 using "execv()" in file 2 we will use "execlp()" to display content.

Output:



```
compilation terminated.
ram@ram:~/Documents$ gcc -o COE19B055_Lab4_Q2 COE19B055_Lab4_Q2.c
ram@ram:~/Documents$ gcc -o COE19B055_Lab4_Q2_1 COE19B055_Lab4_Q2_1.c
ram@ram:~/Documents$ ./COE19B055_Lab4_Q2
In COE19B055_Lab4_Q2_1
CN                   COE19B055_Lab4_Q1.c  COE19B055_Lab4_Q2_1    COE19B055_Lab4_Q2.c  file.tar      OS    os_1.txt  os_2.txt.gz  s.c
COE19B055_Lab4_Q1    COE19B055_Lab4_Q2    COE19B055_Lab4_Q2_1.c  file_1.tar           file.tar.tbz  os_1  os_2      os.txt
ram@ram:~/Documents$ 
```



```
compilation terminated.
ram@ram:~/Documents$ gcc -o COE19B055_Lab4_Q2_1 COE19B055_Lab4_Q2_1.c
ram@ram:~/Documents$ ./COE19B055_Lab4_Q2
In COE19B055_Lab4_Q2_1
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─3*[VBoxClient───VBoxClient───2*[{VBoxClient}]]
        ├─VBoxClient───VBoxClient───3*[{VBoxClient}]
        ├─VBoxService───8*[{VBoxService}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─avahi-daemon───avahi-daemon
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd
        ├─dbus-daemon
        ├─gdm3─┬─gdm-session-wor─┬─gdm-x-session─┬─Xorg───11*[{Xorg}]
        │      │                 │               ├─gnome-session-b─┬─ssh-agent
        │      │                 │               │                 └─2*[{gnome-+
        │      │                 │               └─2*[{gdm-x-session}]
        │      │                 └─2*[{gdm-session-wor}]
        │      └─2*[{gdm3}]
        ├─gnome-keyring-d───3*[{gnome-keyring-d}]
        ├─irqbalance───{irqbalance}
        ├─2*[kerneloops]
        ├─networkd-dispat
        ├─polkitd───2*[{polkitd}]
        ├─rsyslogd───3*[{rsyslogd}]
        ├─rtkit-daemon───2*[{rtkit-daemon}]
        ├─snapd───13*[{snapd}]
        ├─switcheroo-cont───2*[{switcheroo-cont}]
        ├─systemd─┬─(sd-pam)
        │         ├─at-spi-bus-laun─┬─dbus-daemon
        │         │                 └─3*[{at-spi-bus-laun}]
        │         ├─at-spi2-registr───2*[{at-spi2-registr}]
        │         ├─dbus-daemon
        │         ├─dconf-service───2*[{dconf-service}]
        │         ├─evolution-addre───5*[{evolution-addre}]
        │         ├─evolution-calen───8*[{evolution-calen}]
        │         ├─evolution-sourc───3*[{evolution-sourc}]
        │         ├─gjs───5*[{gjs}]
        │         ├─gnome-session-b─┬─evolution-alarm───5*[{evolution-alarm}]
        │         │                 ├─gsd-disk-utilit───2*[{gsd-disk-utilit}]
        │         │                 ├─update-notifier───3*[{update-notifier}]
        │         │                 └─3*[{gnome-session-b}]
        │         ├─gnome-session-c───{gnome-session-c}
        │         ├─gnome-shell─┬─ibus-daemon─┬─ibus-dconf───3*[{ibus-dconf}]
        │         │             │             ├─ibus-engine-sim───2*[{ibus-engi+
```

Q3)

FIXED FILE: We are going to call this fixed file into other files

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>

#include <sys/tree.h>


int main()

{

   printf("In COE19B055_Lab4_Q3_a\n");

   for(int i=0; i<2; i++)

   {

       printf("%d ", i);

   }

   printf("\n");

}
```

execl():

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>


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

{

   char* args[] = {NULL};

   execl("./COE19B055_Lab4_Q3_a", "./COE19B055_Lab4_Q3_a", args);

   printf("In COE19B055_Lab4_Q3_execl\n");

}
```

execve():

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>

int main(int argc, char* argv[])
{
    char* args[] = {NULL};
    char *const env = {NULL};
    execve("./COE19B055_Lab4_Q3_a", args, NULL);
    printf("In COE19B055_Lab4_Q3_execve\n");
}
```

execvp():

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>

int main(int argc, char* argv[])
{
    char* args[] = {NULL};
    execvp("./COE19B055_Lab4_Q3_a", args);
    printf("In COE19B055_Lab4_Q3_execvp\n");
}
```

execle():

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>


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

{

   char* args[] = {NULL};

   char *const env = {NULL};

   execle("./COE19B055_Lab4_Q3_a", "./COE19B055_Lab4_Q3_a", args, args, NULL, env);

   printf("In COE19B055_Lab4_Q3_execle\n");

}
```

execv():

Used in Q2.

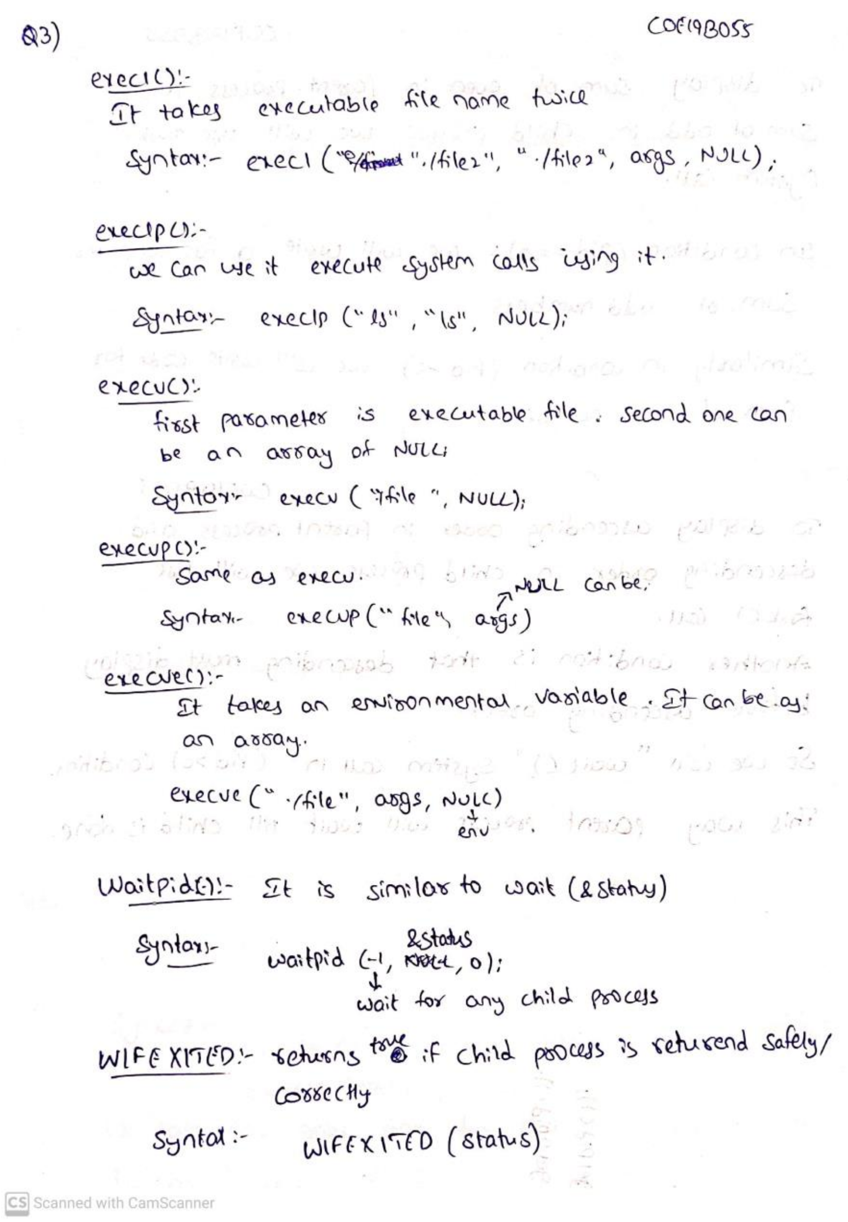
execlp():

Used in Q2.

waitpid() and WIFEXITED():

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>


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

{

   pid_t pid;
```

```
    pid = fork();


    if(pid==0)
    {
        printf("In child process\n");
    }


    int status;
    if(pid>0)
    {
        waitpid(pid, &status, 0);
        if(WIFEXITED(status))
        {
            printf("Child Process is exited safely\n");
        }
    }
    return(0);
}
```



Q3) COE19B055

execl():-
It takes executable file name twice
Syntax:- execl ("~~[illegible]~~./file2", "./file2", args, NULL);

execlp():-
we can use it execute system calls using it.
Syntax:- execlp ("ls", "ls", NULL);

execv():
first parameter is executable file. second one can be an array of NULL;
Syntax:- execv ("./file", NULL);

execvp():-
Same as execv.
Syntax:- execvp (" file", $\overset{\text{NULL can be.}}{\text{args}}$)

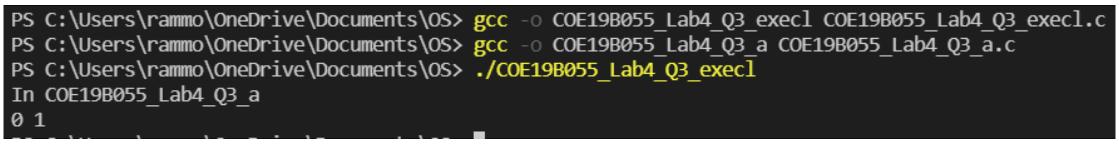
execve():-
It takes an environmental variable. It can be as an array.
execve (" ./file", args, $\underset{\text{env}}{\text{NULL}}$)

Waitpid():- It is similar to wait (&status)
Syntax:- waitpid ($\underset{\text{wait for any child process}}{-1}$, $\overset{\text{\&status}}{\text{~~NULL~~}}$, 0);

WIFEXITED:- returns true if child process is returend safely/ correctly
Syntax:- WIFEXITED (status)

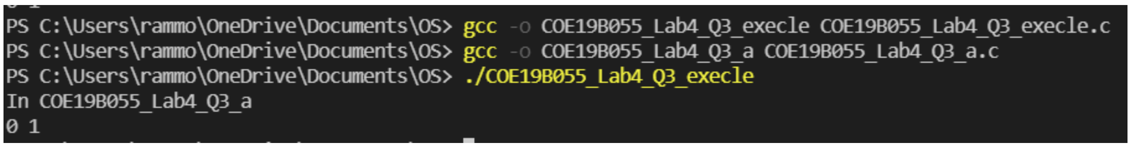
Output:

Using execl():



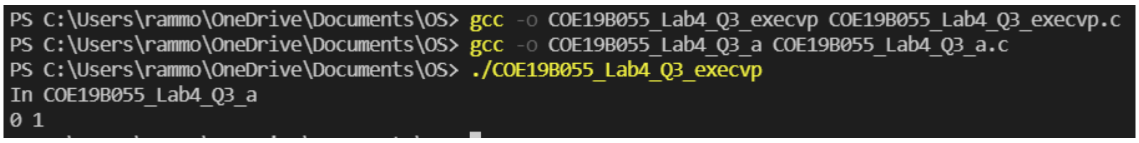
```
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_execl COE19B055_Lab4_Q3_execl.c
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_a COE19B055_Lab4_Q3_a.c
PS C:\Users\rammo\OneDrive\Documents\OS> ./COE19B055_Lab4_Q3_execl
In COE19B055_Lab4_Q3_a
0 1
```

Using execle():



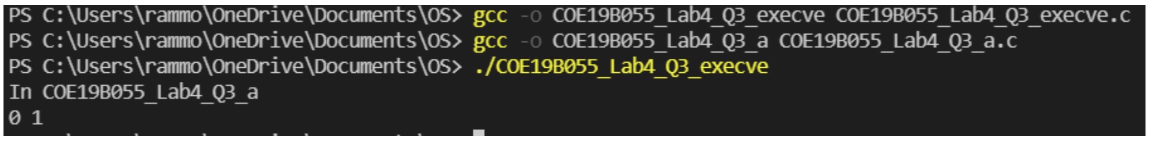
```
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_execle COE19B055_Lab4_Q3_execle.c
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_a COE19B055_Lab4_Q3_a.c
PS C:\Users\rammo\OneDrive\Documents\OS> ./COE19B055_Lab4_Q3_execle
In COE19B055_Lab4_Q3_a
0 1
```

Using execvp():



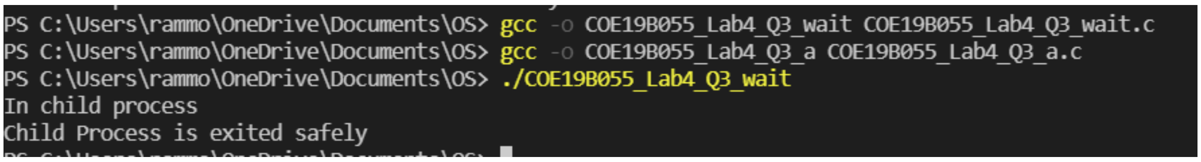
```
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_execvp COE19B055_Lab4_Q3_execvp.c
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_a COE19B055_Lab4_Q3_a.c
PS C:\Users\rammo\OneDrive\Documents\OS> ./COE19B055_Lab4_Q3_execvp
In COE19B055_Lab4_Q3_a
0 1
```

Using execve():



```
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_execve COE19B055_Lab4_Q3_execve.c
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_a COE19B055_Lab4_Q3_a.c
PS C:\Users\rammo\OneDrive\Documents\OS> ./COE19B055_Lab4_Q3_execve
In COE19B055_Lab4_Q3_a
0 1
```

Using waitpid() and WIFEIXTED():

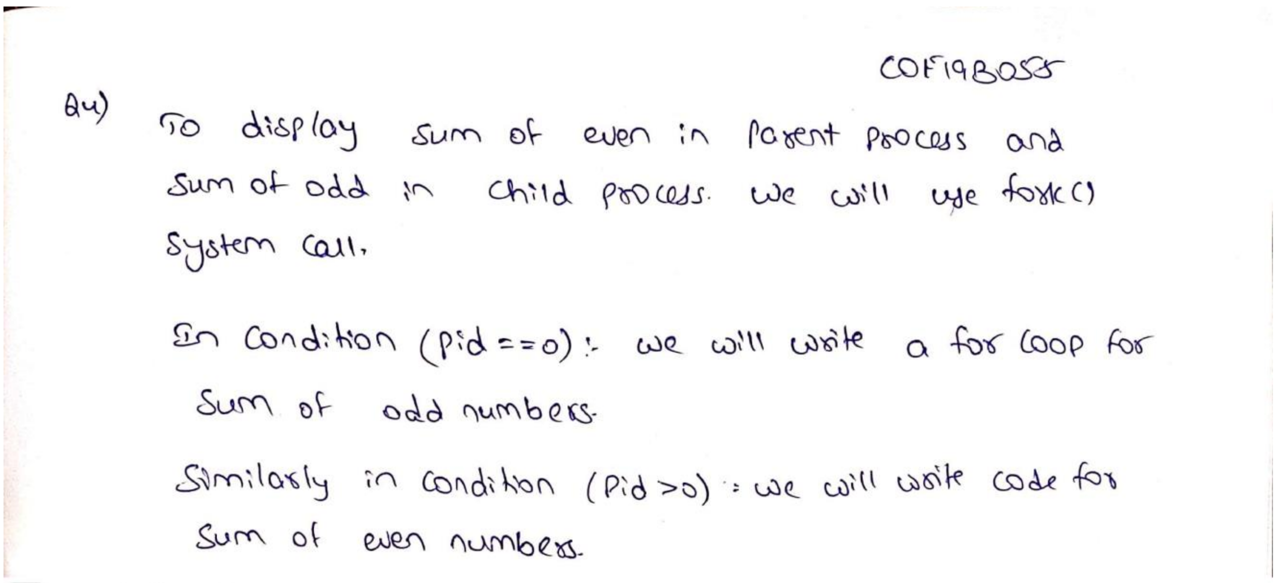


```
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_wait COE19B055_Lab4_Q3_wait.c
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q3_a COE19B055_Lab4_Q3_a.c
PS C:\Users\rammo\OneDrive\Documents\OS> ./COE19B055_Lab4_Q3_wait
In child process
Child Process is exited safely
```

Q4)

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>


int main()

{

   pid_t pid;

   int n, sum_odd=0, sum_even=0, i;


   printf("Enter n: ");

   scanf("%d", &n);


   while (n<1)

   {

      printf("Enter a positive number: ");

      scanf("%d", &n);

   }


   pid = fork();
```

```
  if(pid<0){
    printf("Error in fork process\n");
  }else if(pid==0){
    for(i=1; i<=n; i=i+2)
    {
      sum_odd = sum_odd + i;
    }
    printf("Sum of ODD is: %d\n", sum_odd);
  }else if(pid>0){
    for(i=2; i<=n; i=i+2)
    {
      sum_even = sum_even + i;
    }
    printf("Sum of EVEN is: %d\n", sum_even);
  }
}
```



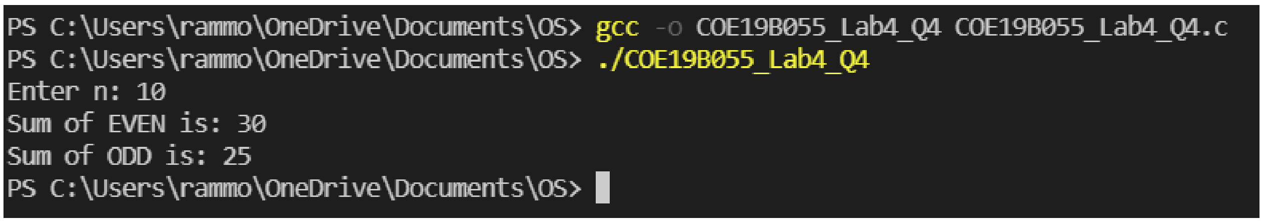
COE19B055

Q4) To display sum of even in parent process and sum of odd in child process. We will use fork() system call.

In condition (pid==0):- we will write a for loop for sum of odd numbers.

Similarly in condition (pid>0): we will write code for sum of even numbers.

Output:

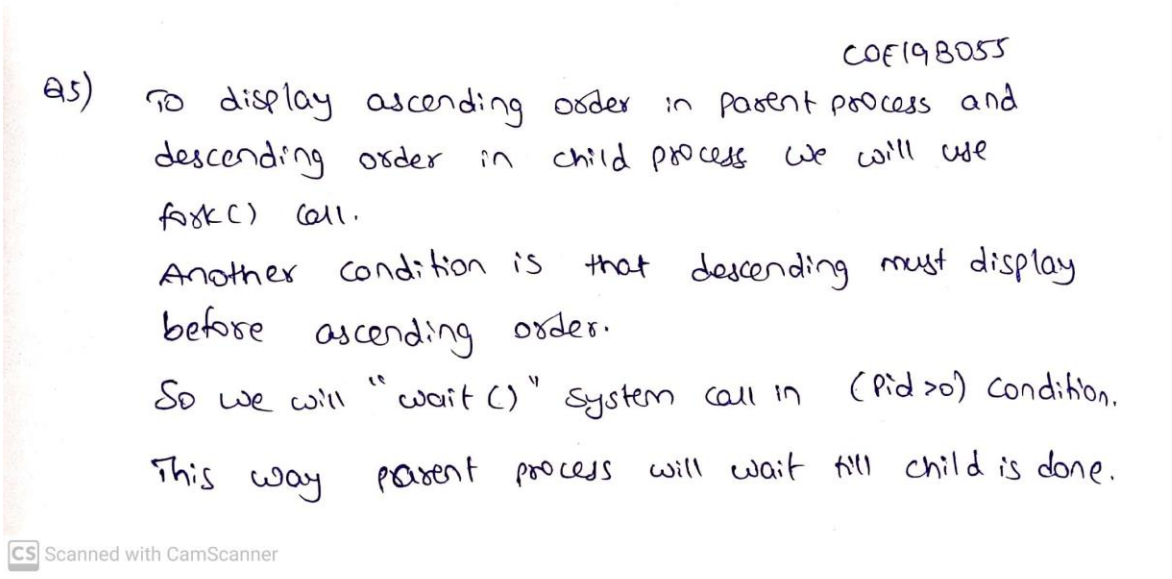


```
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q4 COE19B055_Lab4_Q4.c
PS C:\Users\rammo\OneDrive\Documents\OS> ./COE19B055_Lab4_Q4
Enter n: 10
Sum of EVEN is: 30
Sum of ODD is: 25
PS C:\Users\rammo\OneDrive\Documents\OS>
```

Q5)

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>


void swap(int *p,int *q)

{

   int temp;

   temp=*p;

   *p=*q;

   *q=temp;

}


int main()

{

   pid_t pid;

   int n, i, j, arr[10];


   i = 0;

   while (i<10)

   {

      printf("Enter a number %d : ", i+1);

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

      i++;

   }


   pid = fork();


   if(pid<0){
    printf("Error in fork process\n");
}else if(pid==0){
    for(i=0; i<9; i++)
    {
        for(j=0; j<9-i; j++)
        {
            if(arr[j]<arr[j+1])
            {
                swap(&arr[j], &arr[j+1]);
            }
        }
    }
    printf("Descending order of given is: ");
    for(i=0; i<10; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}else if(pid>0){
    for(i=0; i<9; i++)
    {
        for(j=0; j<9-i; j++)
        {
            if(arr[j]>arr[j+1])
            {
                swap(&arr[j], &arr[j+1]);
            }
        }
    }
    wait(NULL);  /* OR  waitpid(-1, NULL, 0);*/
    printf("Ascending order of given is: ");
    for(i=0; i<10; i++)
```

```
      {
        printf("%d ", arr[i]);
      }
      printf("\n");
    }
}
```



COE19B055

Q5) To display ascending order in parent process and descending order in child process we will use fork() call.

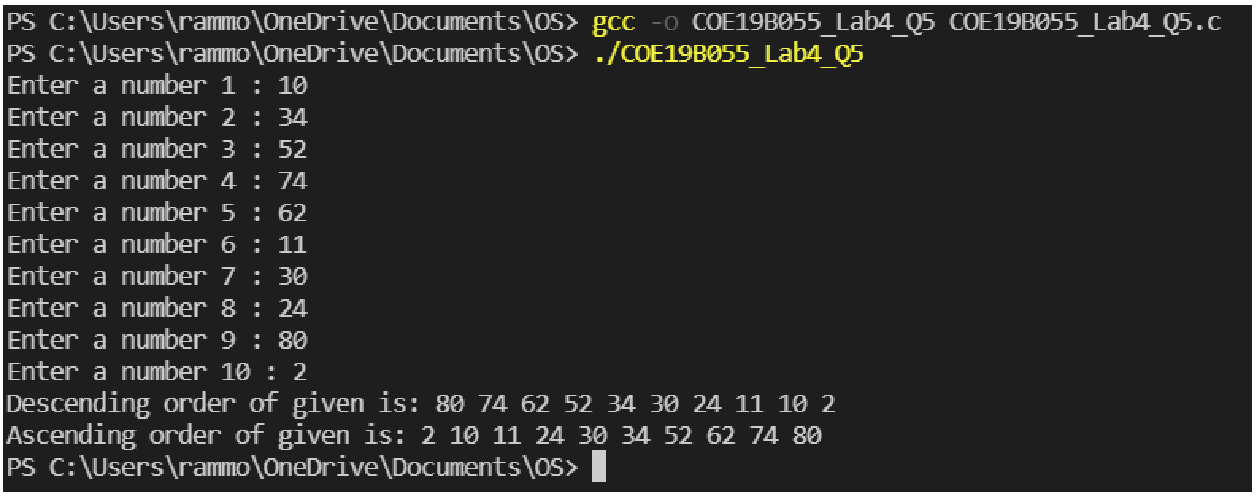
Another condition is that descending must display before ascending order.

So we will "wait()" system call in (Pid>0) condition.

This way parent process will wait till child is done.

Scanned with CamScanner

Output:



```
PS C:\Users\rammo\OneDrive\Documents\OS> gcc -o COE19B055_Lab4_Q5 COE19B055_Lab4_Q5.c
PS C:\Users\rammo\OneDrive\Documents\OS> ./COE19B055_Lab4_Q5
Enter a number 1 : 10
Enter a number 2 : 34
Enter a number 3 : 52
Enter a number 4 : 74
Enter a number 5 : 62
Enter a number 6 : 11
Enter a number 7 : 30
Enter a number 8 : 24
Enter a number 9 : 80
Enter a number 10 : 2
Descending order of given is: 80 74 62 52 34 30 24 11 10 2
Ascending order of given is: 2 10 11 24 30 34 52 62 74 80
PS C:\Users\rammo\OneDrive\Documents\OS>
```