OPERATING SYSTEMS LAB

PRACTICAL 4

NAME: VEDANT BHUTADA

ROLL: 69

BATCH: A4

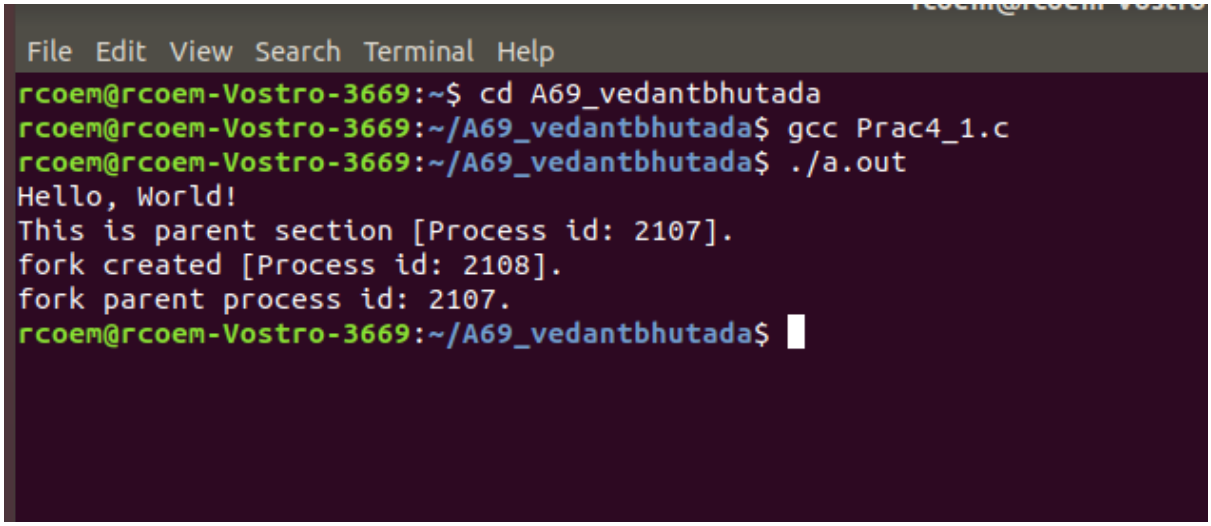
AIM: **Demonstrate process control system calls:**
fork
vfork
exec
wait
and sleep
getpid
and getppid
Also understand the concept of fork bomb, zombie states and orphan states.

CODE AND OUTPUT:

Program-1: fork() Example - C program demonstrating use of fork() in Linux

```
#include <stdio.h>
#include <unistd.h>
 int main()
{
    int id;
    printf("Hello, World!\n");
    id=fork();
    if(id>0)
    {
        /*parent process*/
        printf("This is parent section [Process id: %d].\n",getpid());
    }
    else if(id==0)
    {
        /*child process*/
        printf("fork created [Process id: %d].\n",getpid());
        printf("fork parent process id: %d.\n",getppid());
    }
    else
    {
        /*fork creation faile*/
        printf("fork creation failed!!!\n");
```

```
    }

    return0;
}
```



<u>Program-2:</u> Program on prime numbers and to print array of numbers using fork…

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
void main()
{
        int pid,n,a[10],i,t,j,flag=0;
        pid=fork();
        printf("Pid=%d\n",pid);
        if(pid==0)
        {
                printf("Enter a number to check whether prime or not:");
                scanf("%d",&n);

                        if(n==1)
                        {
                                printf("number is prime");
                        }

                        Else
                        for(j=2;j<=(n/2);j++)
                        {
                                if(n%j==0)
                                {
                                flag=1;
                                printf("not a prime number");
                                break;
```

```
                        }

                }
                        if(flag==0)
                        printf("\nprime number");
}
 else
{
        sleep(5);
        printf("\nenter 10 numbers to sort");
                for(j=0;j<10;j++)
                scanf("%d",&a[j]);
        for(i=0;i<9;i++)
        {
                for(j=0;j<9-i;j++)
                {
                        if(a[j+1]<a[j])
                        {
                        t=a[j];
                        a[j]=a[j+1];

                        a[j+1]=t;
                        }
                }
        }
printf("sorted element:\n");
for(i=0;i<10;i++)
printf("%d ",a[i]);
 }

}
```

```
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out
Pid=6515
Pid=0
Enter a number to check whether prime or not:2

prime number
enter 10 numbers to sort1
7
8
3
4
9
2
6
4
5
sorted element:
1 2 3 4 4 5 6 7 8 9 rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$
```

Program-3 : Program to create Orphan process

```
#include<stdio.h>
#include<unistd.h>
```

```
int main()
{

 pid_t p;

/* create child process */
 p=fork();

 if(p==0) {
    /* fork() returns Zero to child */
    sleep(5);
 }
 printf("The child process pid is %d parent pid %d\n", getpid(), getppid());
/*parent/child waits for 10 secs and exits*/
 sleep(10);
 printf("\nProcess %d is done its Parent pid %d...\n", getpid(), getppid());

 return 0;
}
```

```
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ gcc prac4_3.c
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out
The child process pid is 8519 parent pid 2090
The child process pid is 8520 parent pid 8519

Process 8519 is done its Parent pid 2090...
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$
Process 8520 is done its Parent pid 1011...
```

<u>Program-4:</u> Code for PROGRAM FOR ORPHAN PROCESS in C Programming

```
#include<stdio.h>

main()
{
   int id;

   printf("Before fork()\n");
   id=fork();

   if(id==0)
   {
      printf("Child has started: %d\n ",getpid());
      printf("Parent of this child : %d\n",getppid());
      printf("child prints 1 item :\n ");
      sleep(10);
      printf("child prints 2 item :\n");
   }
```

```
    else
    {
        printf("Parent has started: %d\n",getpid());
        printf("Parent of the parent proc : %d\n",getppid());
    }

    printf("After fork()");
}
```

```
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ gedit prac4_4.c
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ gcc prac4_4.c
prac4_4.c:3:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main()
 ^~~~
prac4_4.c: In function ‘main’:
prac4_4.c:8:8: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
     id=fork();
        ^~~~
prac4_4.c:12:43: warning: implicit declaration of function ‘getpid’; did you mean ‘fgetpos’? [-Wimplicit-function-declaration]
         printf("Child has started: %d\n ",getpid());
                                            ^~~~~~
                                            fgetpos
prac4_4.c:13:46: warning: implicit declaration of function ‘getppid’ [-Wimplicit-function-declaration]
         printf("Parent of this child : %d\n",getppid());
                                               ^~~~~~~
prac4_4.c:15:9: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
         sleep(5);
         ^~~~~
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out
Before fork()
Parent has started: 9827
Parent of the parent proc : 2090
After fork()Child has started: 9828
 Parent of this child : 9827
child prints 1 item :
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$  child prints 2 item :
After fork()
```

Program-5 :

```
// A C program to demonstrate Zombie Process.
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    // Fork returns process id
    // in parent process
    pid_t child_pid = fork();

    // Parent process
    if (child_pid > 0){
        printf("sleep for 5 sec.. ");
}
        sleep(10);

    // Child process
    else
        exit(0);
    return 0;
```

}

```
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out
sleep for 5 sec..
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$
```

Program-6 : execl:

```
main()
{ printf("Files in Directory are:\n");
                execl("/bin/ls","ls", "-l",0);
}
```

```
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ gcc prac4_6.c
prac4_6.c:2:1: warning: return type defaults to 'int' [-Wimplicit-int]
 main()
 ^~~~
prac4_6.c: In function 'main':
prac4_6.c:4:4: warning: implicit declaration of function 'execl' [-Wimplicit-function-declaration]
    execl("/bin/ls","ls", "-l",0);
    ^~~~~
prac4_6.c:4:4: warning: incompatible implicit declaration of built-in function 'execl'
prac4_6.c:4:4: warning: missing sentinel in function call [-Wformat=]
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out
Files in Directory are:
total 36
-rwxrwxr-x 1 rcoem rcoem 8344 Jun 10 14:06 a.out
-rw-rw-r-- 1 rcoem rcoem  529 Jun 10 13:23 Prac4_1.c
-rw-rw-r-- 1 rcoem rcoem  827 Jun 10 13:48 prac4_2.c
-rw-rw-r-- 1 rcoem rcoem  395 Jun 10 13:50 prac4_3.c
-rw-rw-r-- 1 rcoem rcoem  498 Jun 10 13:52 prac4_4.c
-rw-rw-r-- 1 rcoem rcoem  323 Jun 10 14:05 prac4_5.c
-rw-rw-r-- 1 rcoem rcoem  101 Jun 10 14:06 prac4_6.c
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$
```

Program-7 : execvp:
(Another approach for this code must be to perform CPU bound task from one process and I/O bound task from another process)

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/types.h>
#define BUF_SIZE 100
void ChildProcess();
void ParentProcess();
void Child1Process();
main(void)
{
        pid_t pid;
        pid=fork();
        if(pid==0)
                ChildProcess();
        else
                ParentProcess();
}

void ChildProcess()
{
```

```
        int i;
        char *args[]={"ls","-l",0};
        execvp("ls",args);
}

void ParentProcess()
{
        int i,t;
        char b[BUF_SIZE];
        char buffer[5];
        int f1,f2,ret;
        pid_t pid1;
        pid1=fork();
        if(pid1==0)
                Child1Process();
        else
        {
        f1=open("t1.txt",O_RDONLY|O_CREAT,0777);
        if(f1==-1)
                write(1,"Error while opening file!!!!!!",36);
        f2=open("t3.txt",O_WRONLY|O_CREAT,0777);
        if(f2==-1)
                write(1,"Error while opening file!!!!!!",36);
        else
        do{
                ret=read(f1,buffer,sizeof(buffer));
                if(ret)
                {
                        ret=write(f2,buffer,sizeof(buffer));
                }
      } while(ret);
        }
        i=wait(NULL);
        if(i!=-1)
                printf("PID=%d",i);
        i=wait(NULL);
        if(i!=-1)
                printf("PID=%d",i);
}

void Child1Process()
{
        int i;
        long int sum=0;
        for(i=1;i<=1000;i++)
                sum+=i;
        printf("CHILD CALCULATED TOTAL=%ld\n",sum);

}
```

```
    main
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out
CHILD CALCULATED TOTAL=500500
total 40
-rwxrwxr-x 1 rcoem rcoem 8728 Jun 10 14:12 a.out
-rw-rw-r-- 1 rcoem rcoem  529 Jun 10 13:23 Prac4_1.c
-rw-rw-r-- 1 rcoem rcoem  827 Jun 10 13:48 prac4_2.c
-rw-rw-r-- 1 rcoem rcoem  395 Jun 10 13:50 prac4_3.c
-rw-rw-r-- 1 rcoem rcoem  498 Jun 10 13:52 prac4_4.c
-rw-rw-r-- 1 rcoem rcoem  323 Jun 10 14:05 prac4_5.c
-rw-rw-r-- 1 rcoem rcoem  101 Jun 10 14:06 prac4_6.c
-rw-rw-r-- 1 rcoem rcoem 1049 Jun 10 14:11 prac4_7.c
-rwxrwxr-x 1 rcoem rcoem    0 Jun 10 14:12 t1.txt
-rwxrwxr-x 1 rcoem rcoem    0 Jun 10 14:12 t3.txt
-rwxrwxr-x 1 rcoem rcoem    0 Jun 10 14:08 text1.txt
-rwxrwxr-x 1 rcoem rcoem    0 Jun 10 14:08 text3.txt
PID=10466PID=10465rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$
```

<u>Program-8 :</u> Programs to estimate the working of fork() and vfork():
Using the fork() syscall

main.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<unistd.h>
#include<string.h>
#include<errno.h>
int  main()
{
  int i, value;
  int status;
  pid_t f;

  value = 0;
  i = 0;
  status = 1;
  f = fork();
  if (f < 0)
    {
      fprintf(stderr, "Error: %s - fork() < 0 (%d)\n", strerror(errno), f);
    }
  else if (f > 0)
    {
      printf("\n===== Begin Parent =====\n\n");
      printf("fork() = %d\n", f);
      printf("getpid() = %d\n", getpid());
      while (i < 10)
      {
         printf(" Parent -value = %d\n", value);
```

```
        ++value;
        ++i;
    }
  }
 else
  {
    printf("\n===== Begin Child =====\n\n");
    printf("fork() = %d\n", f);
    printf("getpid() = %d\n", getpid());
    while (i < 10)
    {
        printf("  Child -value = %d\n", value);
        ++value;
        ++i;
    }
  }
 printf("status = %d\n", status);
 printf("value = %d\n\n", value);
 printf("===== End =====\n\n");
 return 0;
}
```

```
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ gcc prac4_8.c
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out

===== Begin Parent =====

fork() = 10575
getpid() = 10574
 Parent -value = 0
 Parent -value = 1
 Parent -value = 2
 Parent -value = 3
 Parent -value = 4
 Parent -value = 5
 Parent -value = 6
 Parent -value = 7
 Parent -value = 8
 Parent -value = 9
status = 1
value = 10

===== End =====


===== Begin Child =====

fork() = 0
getpid() = 10575
  Child -value = 0
  Child -value = 1
  Child -value = 2
  Child -value = 3
  Child -value = 4
  Child -value = 5
  Child -value = 6
  Child -value = 7
  Child -value = 8
  Child -value = 9
status = 1
value = 10

===== End =====

rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$
```

Program-9: Using the vfork() syscall

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<unistd.h>
#include<string.h>
#include<errno.h>

int main()
{
int i,value;
int status;
pid_t f;


value = 0;
  i = 0;
  status = 1;
  f = vfork();
  if (f < 0)
    {
fprintf(stderr, "Error: %s - fork() < 0 (%d)\n", strerror(errno), f);
    }
  else if (f > 0)
    {
printf("\n===== Begin Parent =====\n\n");
printf("fork() = %d\n", f);
printf("getpid() = %d\n", getpid());
      while (i < 10)
      {
printf(" Parent - value = %d\n", value);
          ++value;
          ++i;
      }
    }
  else
    {
printf("\n===== Begin Child =====\n\n");
printf("fork() = %d\n", f);
printf("getpid() = %d\n", getpid());
      while (i < 10)
      {
printf("  Child - value = %d\n", value);
          ++value;
          ++i;
      }
      _exit(status);
    }
printf("status = %d\n", status);
```

```
printf("value = %d\n\n", value);
printf("===== End =====\n\n");
  return 0;
}
```

```
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ gcc prac4_9.c
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out

===== Begin Child =====

fork() = 0
getpid() = 10720
  Child - value = 0
  Child - value = 1
  Child - value = 2
  Child - value = 3
  Child - value = 4
  Child - value = 5
  Child - value = 6
  Child - value = 7
  Child - value = 8
  Child - value = 9

===== Begin Parent =====

fork() = 10720
getpid() = 10719
status = 1
value = 10

===== End =====

rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$
```

Program-10 : **C Program for vowels counting using vfork**

```
#include<stdio.h>
#include<sys/types.h>
int main()
{
int j,n,a,i,e,o,u;
      char str[50];
      a=e=i=o=u=0;
pid_tpid;
      if((pid=vfork())<0)
       {
perror("FORK ERROR");
              exit(1);
       }
       if(pid==0)
       {
printf("Counting Number of Vowels using VFORK");
```

```
printf("-------- ------ -- ------ ----- -----");
printf("Enter the String:");
            gets(str);
            _exit(1);
      }
      else
      {
            n=strlen(str);
            for(j=0;j<n;j++)
            {
                  if(str[j]=='a' || str[j]=='A')
                        a++;
                  else if(str[j]=='e' || str[j]=='E')
                        e++;
                  else if(str[j]=='i' || str[j]=='I')
                        i++;
                  else if(str[j]=='o' || str[j]=='O')
                                    o++;
                  else if(str[j]=='u' || str[j]=='U')
                        u++;
            }
printf("Vowels Counting");
printf("------ --------");
printf("Number of A  : %d",a);
printf("Number of E  : %d",e);
printf("Number of I  : %d",i);
printf("Number of O  : %d",o);
printf("Number of U  : %d",u);
printf("Total vowels : %d",a+e+i+o+u);
            exit(1);
      }
}
```

```
prac4_10.c:(.text+0xa3): warning: the `gets' function is dangerous and s
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out

Counting
Number of Vowels using VFORK
-- -- -- -- -- -- -- -- -- -- -- -- -- -- - -- -- --
Enter the String: vedant bhutada

Vowels Counting:
-- -- -- -- -- -- -- -- -- -- -- -- -- -- - -- -- --
Number of A :3
Number of E: 1
Number of I: 0
Number of O: 0
Number of U: 1
Total vowels: 5rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$
```

<u>Program-11 :</u> C program for Fork Bomb

```
#include <stdio.h>
#include <sys/types.h>

int main()
{
   while(1)
      fork();
   return 0;
}
```

```
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ gcc prac4_11.c
prac4_11.c: In function 'main':
prac4_11.c:7:8: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
        fork();
        ^~~~
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out
^C
```

//In this program fork pid, ppid will be demonstrated
// Initially child will have one parent,after the death of its father it will be assigned another father
// named init process. This can be checked by noting down the new ppid and running ps-el. run the program in background by concanating & at the end.

```
#include <stdio.h>

void main()

{

int pid1,pid,ppid;

pid1 =fork();

if(pid1==0)

{

printf("I am child process \n");

printf("child pid is %d\n",getpid());

printf("child ppid is %d\n",getppid());

printf("\n");

system("ps -el");

sleep(20);

printf("now child pid is %d\n",getpid());

printf("now child ppid is %d\n",getppid());

system("ps -el");

}
```

```
if(pid1>0)

{

printf("I am parent process \n");

printf("parent pid is %d\n",getpid());

printf("parent's ppid is %d\n",getppid());

printf("\n");

}

}
```

```
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out
I am parent process
parent pid is 24832
parent's ppid is 24805

I am child process
child pid is 24833
child ppid is 24832

rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ now child pid is 24833
now child ppid is 1011
^C
```

//In this program Zombi Process will be demonstrated
// Run this program in background and run ps-el on the command prompt and see that it is zombi Z process

```
#include <stdio.h>
void main()
{
int pid1,pid,ppid;
pid1 =fork();

if(pid1>0)
{
printf("I am parent process \n");
printf("parent pid is %d\n",getpid());
printf("parent's ppid is %d\n",getppid());
printf("\n");
sleep(5);
}
}
```

Activities Terminal Sat 14:43

rcoem@rcoem-Vostro-3669: ~/A69_vedantbhutada

File Edit View Search Terminal Help

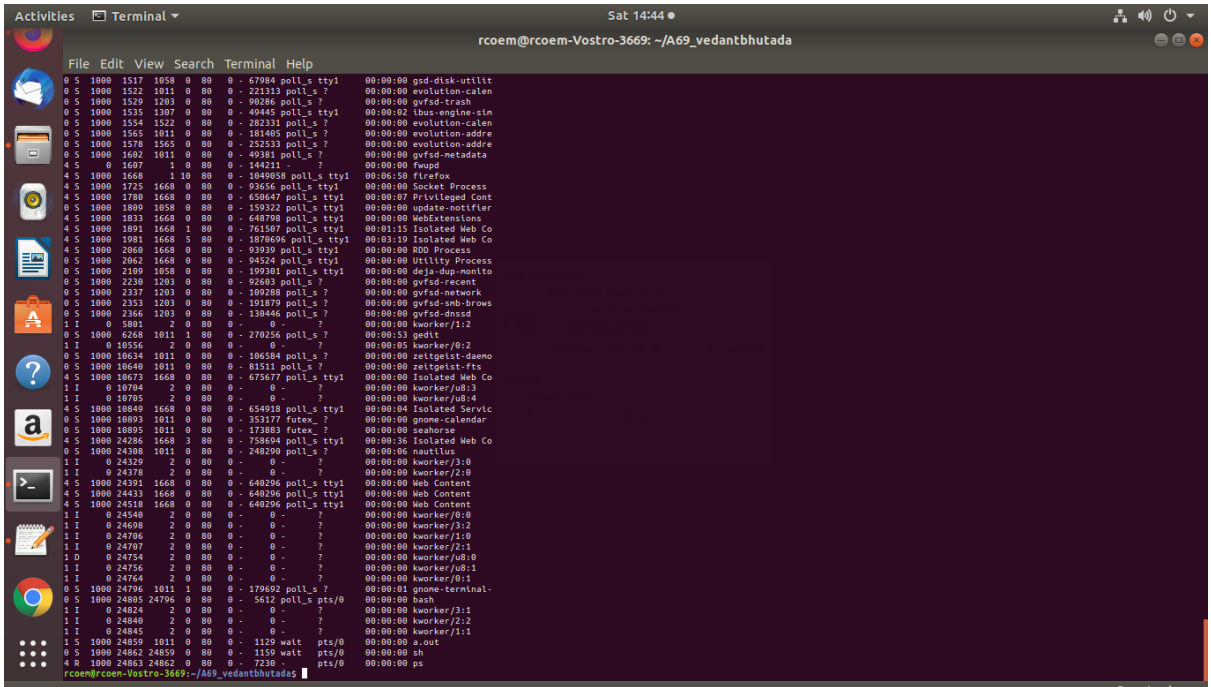
```
0 S  1000  1554  1522  0  80   0 - 282331 poll_s ?        00:00:00 evolution-cale
0 S  1000  1565  1011  0  80   0 - 181405 poll_s ?        00:00:00 evolution-addr
0 S  1000  1578  1565  0  80   0 - 252533 poll_s ?        00:00:00 evolution-addr
0 S  1000  1602  1011  0  80   0 - 49381 poll_s ?        00:00:00 gvfsd-metadata
4 S     0  1607     1  0  80   0 - 144211 -      ?        00:00:00 fwupd
4 S  1000  1668     1 10  80   0 - 1050082 poll_s tty1    00:06:45 firefox
4 S  1000  1725  1668  0  80   0 - 93656 poll_s tty1     00:00:00 Socket Process
4 S  1000  1780  1668  0  80   0 - 650647 poll_s tty1    00:00:07 Privileged Con
0 S  1000  1809  1058  0  80   0 - 159322 poll_s tty1    00:00:00 update-notifie
4 S  1000  1833  1668  0  80   0 - 648798 poll_s tty1    00:00:00 WebExtensions
4 S  1000  1891  1668  1  80   0 - 761507 poll_s tty1    00:01:13 Isolated Web C
4 S  1000  1981  1668  5  80   0 - 1070696 poll_s tty1   00:03:19 Isolated Web C
4 S  1000  2060  1668  0  80   0 - 93939 poll_s tty1     00:00:00 RDD Process
0 S  1000  2062  1668  0  80   0 - 94524 poll_s tty1     00:00:00 Utility Proces
0 S  1000  2109  1058  0  80   0 - 199301 poll_s tty1    00:00:00 deja-dup-monit
0 S  1000  2230  1203  0  80   0 - 92603 poll_s ?        00:00:00 gvfsd-recent
0 S  1000  2337  1203  0  80   0 - 109288 poll_s ?       00:00:00 gvfsd-network
0 S  1000  2353  1203  0  80   0 - 191879 poll_s ?       00:00:00 gvfsd-smb-brow
0 S  1000  2366  1203  0  80   0 - 130446 poll_s ?       00:00:00 gvfsd-dnssd
1 I     0  5801     2  0  80   0 -     0 -      ?        00:00:00 kworker/1:2
0 S  1000  6268  1011  1  80   0 - 270256 poll_s ?       00:00:53 gedit
1 I     0 10556     2  0  80   0 -     0 -      ?        00:00:05 kworker/0:2
0 S  1000 10634  1011  0  80   0 - 106584 poll_s ?       00:00:00 zeitgeist-daem
0 S  1000 10640  1011  0  80   0 - 81511 poll_s ?        00:00:00 zeitgeist-fts
4 S  1000 10673  1668  0  80   0 - 675677 poll_s tty1    00:00:00 Isolated Web C
1 I     0 10704     2  0  80   0 -     0 -      ?        00:00:00 kworker/u8:3
1 I     0 10705     2  0  80   0 -     0 -      ?        00:00:00 kworker/u8:4
4 S  1000 10849  1668  0  80   0 - 654918 poll_s tty1    00:00:04 Isolated Servi
0 S  1000 10893  1011  0  80   0 - 353177 futex_ ?       00:00:00 gnome-calendar
0 S  1000 10895  1011  0  80   0 - 173883 futex_ ?       00:00:00 seahorse
4 S  1000 24286  1668  3  80   0 - 758694 poll_s tty1    00:00:35 Isolated Web C
0 S  1000 24308  1011  0  80   0 - 248290 poll_s ?       00:00:06 nautilus
1 I     0 24329     2  0  80   0 -     0 -      ?        00:00:00 kworker/3:0
1 I     0 24378     2  0  80   0 -     0 -      ?        00:00:00 kworker/2:0
4 S  1000 24391  1668  0  80   0 - 640296 poll_s tty1    00:00:00 Web Content
4 S  1000 24433  1668  0  80   0 - 640296 poll_s tty1    00:00:00 Web Content
4 S  1000 24518  1668  0  80   0 - 640296 poll_s tty1    00:00:00 Web Content
1 I     0 24540     2  0  80   0 -     0 -      ?        00:00:00 kworker/0:0
1 I     0 24698     2  0  80   0 -     0 -      ?        00:00:00 kworker/3:2
1 I     0 24706     2  0  80   0 -     0 -      ?        00:00:00 kworker/1:0
1 I     0 24707     2  0  80   0 -     0 -      ?        00:00:00 kworker/2:1
1 D     0 24754     2  0  80   0 -     0 -      ?        00:00:00 kworker/u8:0
1 I     0 24756     2  0  80   0 -     0 -      ?        00:00:00 kworker/u8:1
1 I     0 24764     2  0  80   0 -     0 -      ?        00:00:00 kworker/0:1
0 S  1000 24796  1011  0  80   0 - 179604 poll_s ?       00:00:01 gnome-terminal
0 S  1000 24805 24796  0  80   0 -  5612 poll_s pts/0    00:00:00 bash
1 I     0 24824     2  0  80   0 -     0 -      ?        00:00:00 kworker/3:1
1 I     0 24840     2  0  80   0 -     0 -      ?        00:00:00 kworker/2:2
1 D     0 24845     2  0  80   0 -     0 -      ?        00:00:00 kworker/1:1
1 S  1000 24859  1011  0  80   0 -  1129 wait   pts/0    00:00:00 a.out
0 S  1000 24860 24859  0  80   0 -  1159 wait   pts/0    00:00:00 sh
4 R  1000 24861 24860  0  80   0 -  7230 -      pts/0    00:00:00 ps
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ now child pid is 24859
now child ppid is 1011
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 56366 -      ?        00:00:04 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:00 kthreadd
```

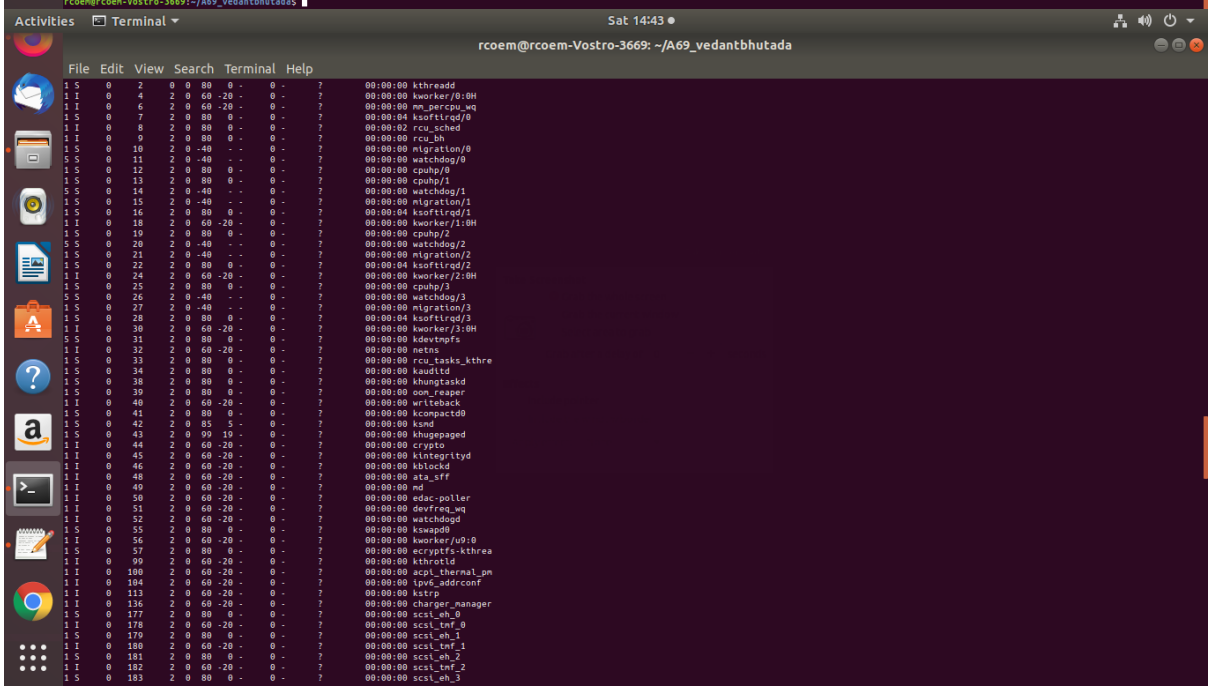
Activities Terminal Sat 14:43

rcoem@rcoem-Vostro-3669: ~/A69_vedantbhutada

File Edit View Search Terminal Help

```
set2_1.c:13:1: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
 sleep(20);
 ^~~~~~
rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ ./a.out
I am parent process
parent pid is 24858
parent's ppid is 24805

I am child process
child pid is 24859
child ppid is 24858

rcoem@rcoem-Vostro-3669:~/A69_vedantbhutada$ F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 56366 -      ?        00:00:04 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0     4     2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H
1 I     0     6     2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0     7     2  0  80   0 -     0 -      ?        00:00:04 ksoftirqd/0
1 I     0     8     2  0  80   0 -     0 -      ?        00:00:02 rcu_sched
1 I     0     9     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
5 S     0    10     2  0 -40   - -     0 -      ?        00:00:00 migration/0
5 S     0    11     2  0 -40   - -     0 -      ?        00:00:00 watchdog/0
1 S     0    12     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
1 S     0    13     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
5 S     0    14     2  0 -40   - -     0 -      ?        00:00:00 watchdog/1
5 S     0    15     2  0 -40   - -     0 -      ?        00:00:00 migration/1
1 S     0    16     2  0  80   0 -     0 -      ?        00:00:04 ksoftirqd/1
1 I     0    18     2  0  60 -20 -     0 -      ?        00:00:00 kworker/1:0H
1 S     0    19     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/2
5 S     0    20     2  0 -40   - -     0 -      ?        00:00:00 watchdog/2
5 S     0    21     2  0 -40   - -     0 -      ?        00:00:00 migration/2
1 S     0    22     2  0  80   0 -     0 -      ?        00:00:04 ksoftirqd/2
1 I     0    24     2  0  60 -20 -     0 -      ?        00:00:00 kworker/2:0H
1 S     0    25     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/3
5 S     0    26     2  0 -40   - -     0 -      ?        00:00:00 watchdog/3
5 S     0    27     2  0 -40   - -     0 -      ?        00:00:00 migration/3
1 S     0    28     2  0  80   0 -     0 -      ?        00:00:04 ksoftirqd/3
1 I     0    30     2  0  60 -20 -     0 -      ?        00:00:00 kworker/3:0H
5 S     0    31     2  0  80   0 -     0 -      ?        00:00:00 kdevtmpfs
1 I     0    32     2  0  60 -20 -     0 -      ?        00:00:00 netns
1 S     0    33     2  0  80   0 -     0 -      ?        00:00:00 rcu_tasks_kthr
1 S     0    34     2  0  80   0 -     0 -      ?        00:00:00 kauditd
1 S     0    38     2  0  80   0 -     0 -      ?        00:00:00 khungtaskd
1 S     0    39     2  0  80   0 -     0 -      ?        00:00:00 oom_reaper
1 I     0    40     2  0  60 -20 -     0 -      ?        00:00:00 writeback
1 S     0    41     2  0  80   0 -     0 -      ?        00:00:00 kcompactd0
1 S     0    42     2  0  85   5 -     0 -      ?        00:00:00 ksmd
1 S     0    43     2  0  99  19 -     0 -      ?        00:00:00 khugepaged
1 I     0    44     2  0  60 -20 -     0 -      ?        00:00:00 crypto
1 I     0    45     2  0  60 -20 -     0 -      ?        00:00:00 kintegrityd
1 I     0    46     2  0  60 -20 -     0 -      ?        00:00:00 kblockd
1 I     0    48     2  0  60 -20 -     0 -      ?        00:00:00 ata_sff
1 I     0    49     2  0  60 -20 -     0 -      ?        00:00:00 md
1 I     0    50     2  0  60 -20 -     0 -      ?        00:00:00 edac-poller
1 I     0    51     2  0  60 -20 -     0 -      ?        00:00:00 devfreq_wq
1 I     0    52     2  0  60 -20 -     0 -      ?        00:00:00 watchdogd
1 S     0    55     2  0  80   0 -     0 -      ?        00:00:00 kswapd0
```

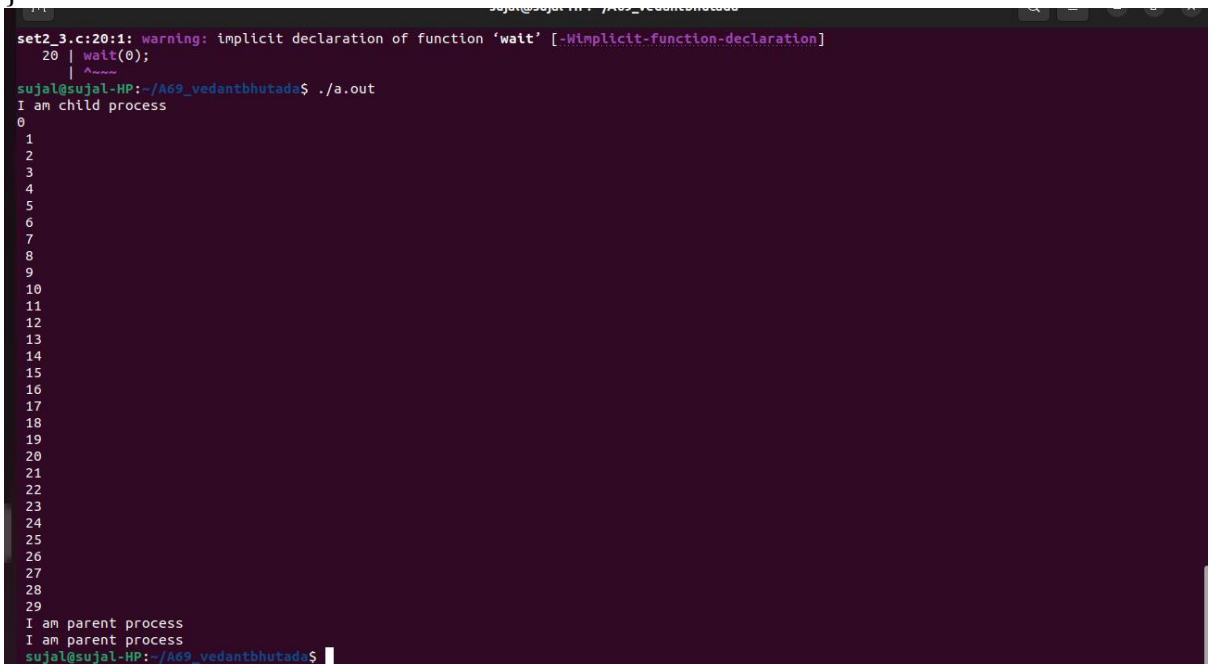




// We will demonstrate sleep and wait system calls in this programe

```
#include <stdio.h>
void main()
{
int pid1,i;
pid1 =fork();
if(pid1==0)
{
printf("I am child process \n");
  for(i=0;i<30;i++)
```

```
    {
     printf("%d\n ", i);
     sleep(1);
//system("ps -el");
    }
//exit(0);
}

else
{
wait(0);
sleep(10);
printf("I am parent process\n ");
wait(0);
//sleep(10);
printf("I am parent process\n ");
//wait(0);
}
}
```



// We will demonstrate execl system calls in this programe
// Compile this program by gcc -o first_d first_d.c
// Compile next program by gcc -o first_e first_e.c

```
#include <stdio.h>
#include <unistd.h>

void main()
{
printf("before exec my pid is %d\n",getpid());
```

```
printf("before exec my ppid is %d\n",getppid());
printf("exec starts\n");
execl("first_e","first_e",(char*)0);
printf("This will not print\n");
}
```

```
sujal@sujal-HP:~/A69_vedantbhutada$ gcc set2_4.c
sujal@sujal-HP:~/A69_vedantbhutada$ ./a.out
before exec my pid is 9515
before exec my ppid is 2227
exec starts
This will not print
sujal@sujal-HP:~/A69_vedantbhutada$
```

// We will demonstrate execl system calls in this programe
// Compile this program by gcc -o first_d first_d.c
// Compile next program by gcc -o first_e first_e.c

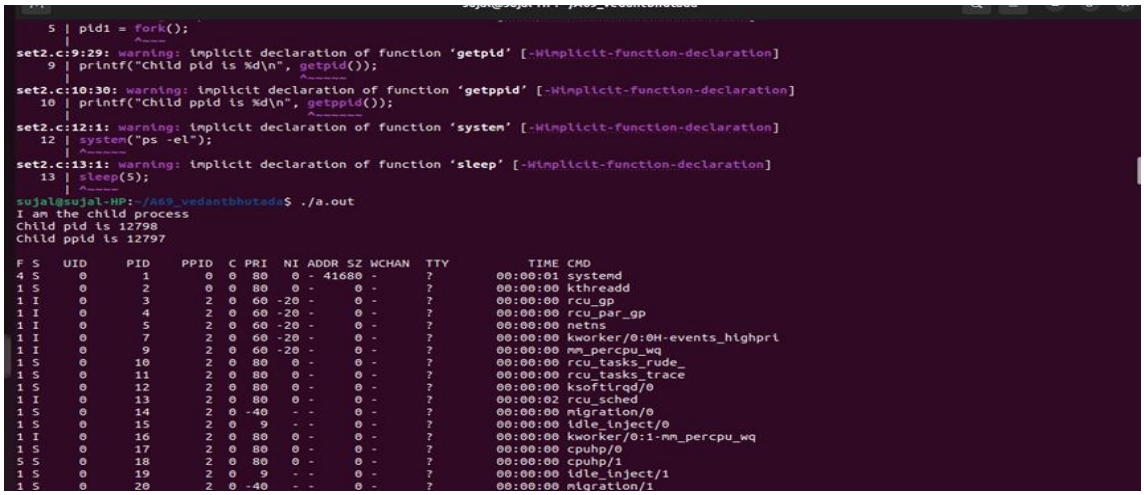
```
#include <stdio.h>
void main()
{
printf("After exec my pid is %d\n",getpid());
printf("After exec my ppid is %d\n",getppid());
printf("exec ends\n");
}
```

```
sujal@sujal-HP:~/A69_vedantbhutada$ gcc set2_5.c
set2_5.c: In function 'main':
set2_5.c:4:36: warning: implicit declaration of function 'getpid' [-Wimplicit-function-declaration]
    4 | printf("After exec my pid is %d\n",getpid());
      |                                    ^~~~~~
set2_5.c:5:37: warning: implicit declaration of function 'getppid' [-Wimplicit-function-declaration]
    5 | printf("After exec my ppid is %d\n",getppid());
      |                                     ^~~~~~~
sujal@sujal-HP:~/A69_vedantbhutada$ ./a.out
After exec my pid is 10538
After exec my ppid is 2227
exec ends
sujal@sujal-HP:~/A69_vedantbhutada$
```

Program to verify pid and ppid

```
#include <stdio.h>

void main()

{

int pid1, pid, ppid;

pid1 = fork();

if (pid1 == 0)
```

```
{
printf("I am the child process\n");
printf("Child pid is %d\n", getpid());
printf("Child ppid is %d\n", getppid());
printf("\n");
system("ps -el");
sleep(5);
printf("Now child pid is %d\n", getpid());
printf("Now child ppid is %d\n", getppid());
system("ps -el");
}
if (pid1 > 0)
{
sleep(2);
printf("I'm the parent process\n");
printf("Parent pid is %d\n", getpid());
printf("Parent ppid is %d\n", getppid());
printf("\n");
}
}
```



```
    5 | pid1 = fork();
      |        ^~~~
set2.c:9:29: warning: implicit declaration of function 'getpid' [-Wimplicit-function-declaration]
    9 | printf("Child pid is %d\n", getpid());
      |                             ^~~~~~
set2.c:10:30: warning: implicit declaration of function 'getppid' [-Wimplicit-function-declaration]
   10 | printf("Child ppid is %d\n", getppid());
      |                              ^~~~~~~
set2.c:12:1: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
   12 | system("ps -el");
      | ^~~~~~
set2.c:13:1: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
   13 | sleep(5);
      | ^~~~~
sujal@sujal-HP:~/A69_vedantbhutada$ ./a.out
I am the child process
Child pid is 12798
Child ppid is 12797

F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0       1       0  0  80   0 - 41680 -      ?        00:00:01 systemd
1 S     0       2       0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0       3       2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0       4       2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0       5       2  0  60 -20 -     0 -      ?        00:00:00 netns
1 I     0       7       2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-events_highpri
1 I     0       9       2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0      10       2  0  80   0 -     0 -      ?        00:00:00 rcu_tasks_rude_
1 S     0      11       2  0  80   0 -     0 -      ?        00:00:00 rcu_tasks_trace
1 S     0      12       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 I     0      13       2  0  80   0 -     0 -      ?        00:00:02 rcu_sched
1 S     0      14       2  0 -40   - -     0 -      ?        00:00:00 migration/0
1 S     0      15       2  0   9   - -     0 -      ?        00:00:00 idle_inject/0
1 I     0      16       2  0  80   0 -     0 -      ?        00:00:00 kworker/0:1-mm_percpu_wq
1 S     0      17       2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
5 S     0      18       2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
1 S     0      19       2  0   9   - -     0 -      ?        00:00:00 idle_inject/1
1 S     0      20       2  0 -40   - -     0 -      ?        00:00:00 migration/1
```

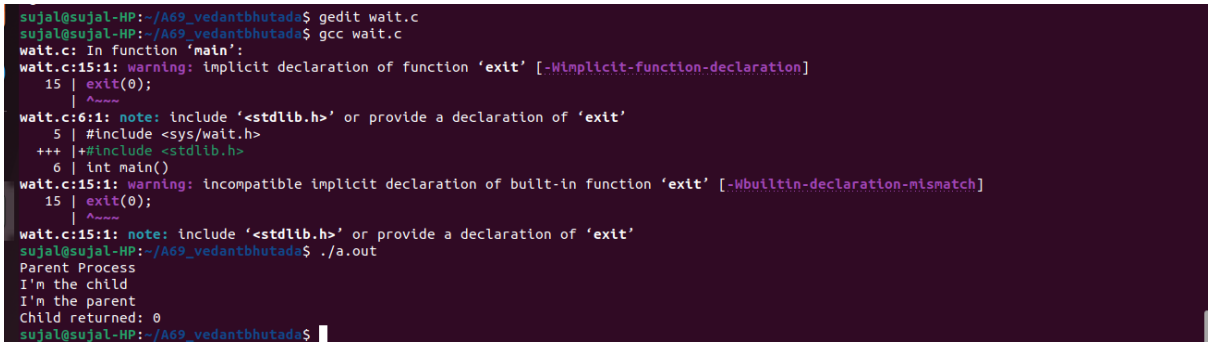
```
0 S  1000    2130    1697  0  80   0 - 800488 do_pol ?        00:00:01 gjs
0 S  1000    2153    1495  0  80   0 - 42942 do_pol ?         00:00:00 gvfsd-metadata
0 S  1000    2209    1495  0  80   0 - 207614 do_pol ?        00:00:07 gnome-terminal-
0 S  1000    2227    2209  0  80   0 -  4947 do_sel pts/0     00:00:00 bash
0 S  1000    2244    1677  0  80   0 - 125668 do_pol ?        00:00:00 update-notifier
4 S  1000    2397    1697  6  80   0 - 3105070 do_pol ?       00:01:49 firefox
0 S  1000    2490    1697  0  80   0 - 200773 ep_pol ?        00:00:10 Xwayland
0 S  1000    2509    1495  0  80   0 - 258603 do_pol ?        00:00:00 gsd-xsettings
0 S  1000    2541    1495  0  80   0 - 50706 do_pol ?         00:00:00 ibus-x11
4 S  1000    2631    2397  0  80   0 - 56541 do_pol ?         00:00:00 Socket Process
4 S  1000    2674    2397  0  80   0 - 617725 do_pol ?        00:00:02 Privileged Cont
0 S  1000    2715    1495  0  80   0 - 253609 futex_ ?        00:00:00 snap
4 S  1000    2863    2397  0  80   0 - 615062 do_pol ?        00:00:01 WebExtensions
4 S  1000    3169    2397  5  80   0 - 1840495 do_pol ?       00:01:30 Isolated Web Co
4 S  1000    3428    2397  0  80   0 - 86017 do_pol ?         00:00:00 RDD Process
0 S  1000    3430    2397  0  80   0 - 58081 do_pol ?         00:00:00 Utility Process
4 S  1000    4386    2397  1  80   0 - 673316 do_pol ?        00:00:26 file:// Content
0 S  1000    4484    1495  0  80   0 - 24148 pipe_r ?         00:00:00 oosplash
0 S  1000    4500    4484  0  80   0 - 261092 do_pol ?        00:00:08 soffice.bin
1 I     0    6158       2  0  80   0 -     0 -      ?         00:00:00 kworker/2:0-mm_percpu_wq
4 S  1000    6182    2397  0  80   0 - 605765 do_pol ?        00:00:00 Web Content
1 I     0    8820       2  0  60 -20 -     0 -      ?         00:00:03 kworker/u17:2-rtw_tx_wq
4 S  1000    8822    2397  0  80   0 - 605765 do_pol ?        00:00:00 Web Content
1 I     0    9209       2  0  80   0 -     0 -      ?         00:00:00 kworker/1:0-events
4 S  1000    9422    2397  0  80   0 - 605765 do_pol ?        00:00:00 Web Content
1 I     0    9690       2  0  80   0 -     0 -      ?         00:00:00 kworker/5:2-events
1 I     0   12442       2  0  80   0 -     0 -      ?         00:00:00 kworker/6:2-events
0 S  1000   12566    1495  1  80   0 - 202529 do_pol ?        00:00:08 gedit
1 I     0   12629       2  0  80   0 -     0 -      ?         00:00:00 kworker/4:1-events
1 I     0   12638       2  0  80   0 -     0 -      ?         00:00:00 kworker/u16:0-flush-259:0
1 I     0   12639       2  0  80   0 -     0 -      ?         00:00:00 kworker/0:0-cgroup_destroy
0 S  1000   12671    1527  0  80   0 - 99342 do_pol ?         00:00:00 gvfsd-network
0 S  1000   12688    1527  0  80   0 - 81368 do_pol ?         00:00:00 gvfsd-dnssd
1 I     0   12780       2  0  60 -20 -     0 -      ?         00:00:00 kworker/u17:1-rtw_tx_wq
1 S  1000   12798    1495  0  80   0 -   693 do_wai pts/0     00:00:00 a.out
0 S  1000   12801   12798  0  80   0 -   722 do_wai pts/0     00:00:00 sh
4 R  1000   12802   12801  0  80   0 -  5331 -      pts/0     00:00:00 ps
```

```
4 S  1000    6182    2397  0  80   0 - 605765 do_pol ?        00:00:00 Web Content
1 I     0    8820       2  0  60 -20 -     0 -      ?         00:00:03 kworker/u17:2-rtw_tx_wq
4 S  1000    8822    2397  0  80   0 - 605765 do_pol ?        00:00:00 Web Content
1 I     0    9209       2  0  80   0 -     0 -      ?         00:00:00 kworker/1:0-events
4 S  1000    9422    2397  0  80   0 - 605765 do_pol ?        00:00:00 Web Content
1 I     0    9690       2  0  80   0 -     0 -      ?         00:00:00 kworker/5:2-events
1 I     0   12442       2  0  80   0 -     0 -      ?         00:00:00 kworker/6:2-events
0 S  1000   12566    1495  1  80   0 - 202529 do_pol ?        00:00:08 gedit
1 I     0   12629       2  0  80   0 -     0 -      ?         00:00:00 kworker/4:1-events
1 I     0   12638       2  0  80   0 -     0 -      ?         00:00:00 kworker/u16:0-flush-259:0
1 I     0   12639       2  0  80   0 -     0 -      ?         00:00:00 kworker/0:0-cgroup_destroy
0 S  1000   12671    1527  0  80   0 - 99342 do_pol ?         00:00:00 gvfsd-network
0 S  1000   12688    1527  0  80   0 - 81368 do_pol ?         00:00:00 gvfsd-dnssd
1 I     0   12780       2  0  60 -20 -     0 -      ?         00:00:00 kworker/u17:1-rtw_tx_wq
0 S  1000   12797    2227  0  80   0 -   660 hrtime pts/0     00:00:00 a.out
1 S  1000   12798   12797  0  80   0 -   693 do_wai pts/0     00:00:00 a.out
0 S  1000   12799   12798  0  80   0 -   722 do_wai pts/0     00:00:00 sh
4 R  1000   12800   12799  0  80   0 -  5331 -      pts/0     00:00:00 ps
I'm the parent process
Parent pid is 12797
Parent ppid is 2227

sujal@sujal-HP:~/A69_vedantbhutada$ Now child pid is 12798
Now child ppid is 1495
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0       1       0  0  80   0 - 41680 -      ?        00:00:01 systemd
1 S     0       2       0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0       3       2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0       4       2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0       5       2  0  60 -20 -     0 -      ?        00:00:00 netns
1 I     0       7       2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-events_highpri
1 I     0       9       2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0      10       2  0  80   0 -     0 -      ?        00:00:00 rcu_tasks_rude_
1 S     0      11       2  0  80   0 -     0 -      ?        00:00:00 rcu_tasks_trace
1 S     0      12       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 I     0      13       2  0  80   0 -     0 -      ?        00:00:02 rcu_sched
1 S     0      14       2  0 -40   - -     0 -      ?        00:00:00 migration/0
1 S     0      15       2  0   9   - -     0 -      ?        00:00:00 idle_inject/0
```

Program executing wait syscall

```
#include <stdio.h>

#include <sys/stat.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main()
```

```
{
int status;
pid_t fork_return;
fork_return = fork();
if (fork_return == 0) // child process
{
sleep(2);
printf("I'm the child\n");
exit(0);
}
else // parent process
{
printf("Parent Process\n");
wait(&status);
printf("I'm the parent\n");
if (WIFEXITED(status))
printf("Child returned: %d\n", WEXITSTATUS(status));
}
}
```



Program executing waitpid

```
#include <stdio.h>
#include <sys/stat.h>
```

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
void waitexample()
{
int i, stat;
pid_t pid[5];
for (int i = -0; i < 5; i++)
{
if ((pid[i] = fork()) == 0)
{
sleep(1)
exit(100 + i);
}
}
// using waitpid() and printing exit status of children
for (i = 0; i < 5; i++)
{
pid_t cpid = waitpid(pid[i], &stat, 0);
if (WIFEXITED(stat))
printf("Child %d terminated with status %d\n", cpid,
WEXITSTATUS(stat));
}
}
int main()
{
waitexample();
return 0;
}
```

```
sujal@sujal-HP:~/A69_vedantbhutada$ gcc waitpid.c
waitpid.c: In function ‘waitexample’:
waitpid.c:15:1: warning: implicit declaration of function ‘exit’ [-Wimplicit-function-declaration]
   15 | exit(100 + i);
      | ^~~~
waitpid.c:6:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
    5 | #include <sys/wait.h>
  +++ |+#include <stdlib.h>
    6 | void waitexample()
waitpid.c:15:1: warning: incompatible implicit declaration of built-in function ‘exit’ [-Wbuiltin-declaration-mismatch]
   15 | exit(100 + i);
      | ^~~~
waitpid.c:15:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
sujal@sujal-HP:~/A69_vedantbhutada$ ./a.out
Child 12900 terminated with status 100
Child 12901 terminated with status 101
Child 12902 terminated with status 102
Child 12903 terminated with status 103
Child 12904 terminated with status 104
sujal@sujal-HP:~/A69_vedantbhutada$
```

OS prac-4_additional question—

In this, you work with the fork(), wait() and the exec*() family of functions in order to find the maximum in an array of integers.

Part 1

Write a C program *parmax.c* that creates a tree of processes in order to recursively compute the maximum in an array of integers. The process at the root of the tree reads the count n of integers in the array. An array A of size n is then populated with randomly generated integers of small values (in the range 0–127). The initially unsorted array is printed by the root process.

Any process in the tree handles a chunk of the array A. The chunk is delimited by two indices L and R. For the root process, $L = 0$ and $R = n - 1$. Any process P in the tree (including the root) first counts the number of integers in the chunk it has got. If that count is less than 10, the process P itself computes the maximum element in its chunk, prints it, and exits. If the chunk size of P is 10 or more, then P creates two child processes PL and PR which handle the chunks $[L, M]$ and $[M + 1, R]$ in A respectively, where $M = (L + R) / 2$. P waits until the two child processes PL and PR exit. It then computes the maximum of the two maximum values computed by PL and PR, prints this maximum, and exits.

Every non-root process returns to its parent (via the exit status) the maximum value for its chunk. During the printing of the maximum computed by a process P, the PID and the parent PID of P are also printed.

For $n = 50$, the ranges of the chunks handled by different processes in the tree are shown below.

```
                                  [0 – 49]
                 /                                   \
            [0 – 24]                            [25 – 49]
          /          \                        /          \
     [0 – 12]      [13 – 24]             [25 – 37]      [38 – 49]
     /    \        /      \              /      \       /      \
[0 – 6] [7 – 12] [13 – 18] [19 – 24]  [25 – 31] [32 – 37] [38 – 43] [44 – 49]
```

It is expected that your code will handle values of n in the range 50 – 100. Compile your code, and generate an executable file with the name *parmax*.

Part 2

Write a separate C code *wrapper.c* to run the executable *parmax* created in Part 1. When *parmax* exits, your wrapper function should also exit.

Submit the two C source files *parmax.c* and *wrapper.c*.

Code :

parmax.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int findmax(int *A, int L, int R)
{
int max = A[L];
for (int i = L + 1; i <= R; i++)
{
if (A[i] > max)
{
max = A[i];
}
}
return max;
}
void process(int *A, int L, int R)
{
if (R - L + 1 < 10)
{
int max = findmax(A, L, R);
printf("Process Id : %d (Parent Process Id : %d) - Max: %d\n",
getpid(), getppid(), max);
exit(max);
}
else
```

```
{
int M = (L + R) / 2;
int left_child, right_child;
int left_status, right_status;
left_child = fork();
if (left_child == 0)
{
process(A, L, M);
}
else
{
right_child = fork();
if (right_child == 0)
{
process(A, M + 1, R);
}
else
{
waitpid(left_child, &left_status, 0);
waitpid(right_child, &right_status, 0);
int max_l = WEXITSTATUS(left_status);
int max_r = WEXITSTATUS(right_status);
int max;
if (max_l > max_r)
max = max_l;
else
printf("Process Id : %d (Parent Process Id : %d) - Max:
%d\n", getpid(), getppid(), max);
exit(max);
```

```
}
}
}
}
void main()
{
int n = 50;
int A[n];
printf("Array is : \n");
for (int i = 0; i < n; i++)
{
A[i] = rand() % 128;
printf("%d ", A[i]);
}
printf("\n");
process(A, 0, n - 1);
}
```

---

wrapper.c

```
#include <stdlib.h>
#include <stdio.h>
int main()
{
system("./parmax");
return 0;
}
```

```
sujal@sujal-HP:~/A69_vedantbhutada$ gcc -o parmax parmax.c
sujal@sujal-HP:~/A69_vedantbhutada$ gcc -o wrapper wrapper.c
sujal@sujal-HP:~/A69_vedantbhutada$ ./wrapper
Array is :
103 70 105 115 81 127 74 108 41 77 58 43 114 123 99 70 124 66 84 120 27 104 103 13 118 90 46 99 51 31 73 26 102 50 13 55 49 88 35 90 37 93 5 2
3 88 105 94 84 43 50
Process Id : 12619 (Parent Process Id : 12615) - Max: 127
Process Id : 12620 (Parent Process Id : 12617) - Max: 124
Process Id : 12621 (Parent Process Id : 12618) - Max: 93
Process Id : 12622 (Parent Process Id : 12616) - Max: 99
Process Id : 12623 (Parent Process Id : 12615) - Max: 114
Process Id : 12624 (Parent Process Id : 12617) - Max: 120
Process Id : 12625 (Parent Process Id : 12618) - Max: 105
Process Id : 12626 (Parent Process Id : 12616) - Max: 102
Process Id : 12618 (Parent Process Id : 12614) - Max:32764
Process Id : 12616 (Parent Process Id : 12614) - Max:32764
Process Id : 12614 (Parent Process Id : 12612) - Max:32648
Show Applications 12 (Parent Process Id : 12611) - Max:32648
sujal@sujal-HP:~/A69_vedantbhutada$
```

---

4) Demonstrate the following questions using programs:

Q1. Create a parent-child relationship between two processes. The parent should print two statements:

A) Parent (P) is having ID <PID>

B) ID of P's Child is <PID_of_Child>

The child should print two statements:

C) Child is having ID <PID>

D) My Parent ID is <PID_of_Parent>

Make use of wait() in such a manner that the order of the four statements A, B, C and D is:

A

C

D

B

You are free to use any other relevant statement/printf as you desire and their order of execution does not

matter.

```
#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>
```

```
int main() {
    pid_t child_pid;

    child_pid = fork();

    if (child_pid == 0) {
        printf("C) Child is having ID %d\n", getpid());
        printf("D) My Parent ID is %d\n", getppid());
    } else if (child_pid > 0) {
        printf("A) Parent (P) is having ID %d\n", getpid());
        wait(NULL); // Waits for the child process to finish
        printf("B) ID of P's Child is %d\n", child_pid);
    } else {
        printf("Fork failed.\n");
        return 1;
    }

    return 0;
}
```

Q2. Create a parent-child relationship between two processes such that the Child process creates a file named Relation.txt and the Parent process write some content into it by taking the input from the user

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>
```

```
int main() {
    pid_t child_pid;

    child_pid = fork();

    if (child_pid == 0) {
        // Child process
        FILE *file = fopen("Relation.txt", "w");
        if (file == NULL) {
            printf("Error creating the file.\n");
            return 1;
        }
        fclose(file);
    } else if (child_pid > 0) {
        // Parent process
        wait(NULL); // Waits for the child process to finish

        FILE *file = fopen("Relation.txt", "a");
        if (file == NULL) {
            printf("Error opening the file.\n");
            return 1;
        }

        char content[100];
        printf("Enter the content to write into the file: ");
        fgets(content, sizeof(content), stdin);

        fprintf(file, "%s", content);

        fclose(file);
```

```
    } else {
        printf("Fork failed.\n");
        return 1;
    }

    return 0;
}
```

Q3. Write a program to create two child process. The parent process should wait for both the child to finish.

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

int main() {
    pid_t child_pid1, child_pid2;

    child_pid1 = fork();

    if (child_pid1 == 0) {
        // First child process
        printf("First child process\n");
    } else if (child_pid1 > 0) {
        child_pid2 = fork();

        if (child_pid2 == 0) {
            // Second child process
```

```
        printf("Second child process\n");
    } else if (child_pid2 > 0) {
        // Parent process
        wait(NULL); // Waits for the first child process to finish
        wait(NULL); // Waits for the second child process to finish

        printf("Both child processes have finished.\n");
    } else {
        printf("Second fork failed.\n");
        return 1;
    }
  } else {
    printf("First fork failed.\n");
    return 1;
  }

  return 0;
}
```

Q4. Can we use wait() to make the child process wait for the parent process to finish?

No, you cannot use wait() in the child process to make it wait for the parent process to finish. The wait() system call is specifically used by the parent process to wait for its child processes to finish.

Q5. What does the wait() system call return on success?

The wait() system call returns the process ID (PID) of the terminated child process on success.

CONCLUSION: Process control system calls has been studied and Linux C programs on them has been implemented.