Laboratory Assignments 4
Subject: Design Principles of Operating Systems
Subject code: CSE 3249

Assignment 4: Familiarization with Process Management in Linux environment.

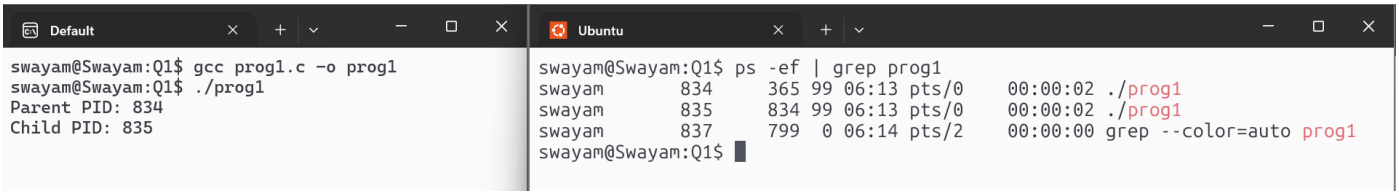
Objective of this Assignment:

- To trace the different states of a process during its execution.
- To learn the use of different system calls such as (fork(),vfork(),wait(),exec()) for process handling in Unix/Linux environment.

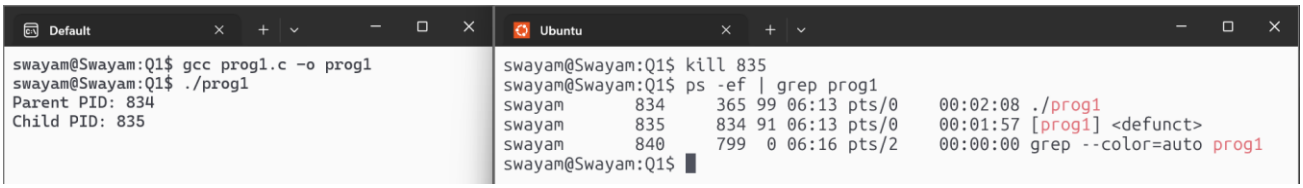
1. Write a C program to create a child process using fork() system call. The child process will print the message "Child" with its process identifier and then continue in an indefinite loop. The parent process will print the message "Parent" with its process identifier and then continue in an indefinite loop.

```
swayam@Swayam:Q1$ cat prog1.c
#include <stdio.h>
#include <unistd.h>

int main()
{
    pid_t pid;
    pid = fork();

    if (pid == 0) {
        printf("Child PID: %d\n", getpid());
        while (1);
    } else {
        printf("Parent PID: %d\n", getpid());
        while (1);
    }
    return 0;
}
swayam@Swayam:Q1$
```

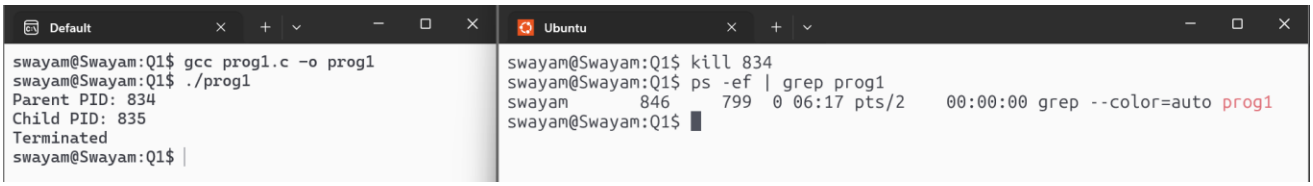
a) Run the program and trace the state of both processes.



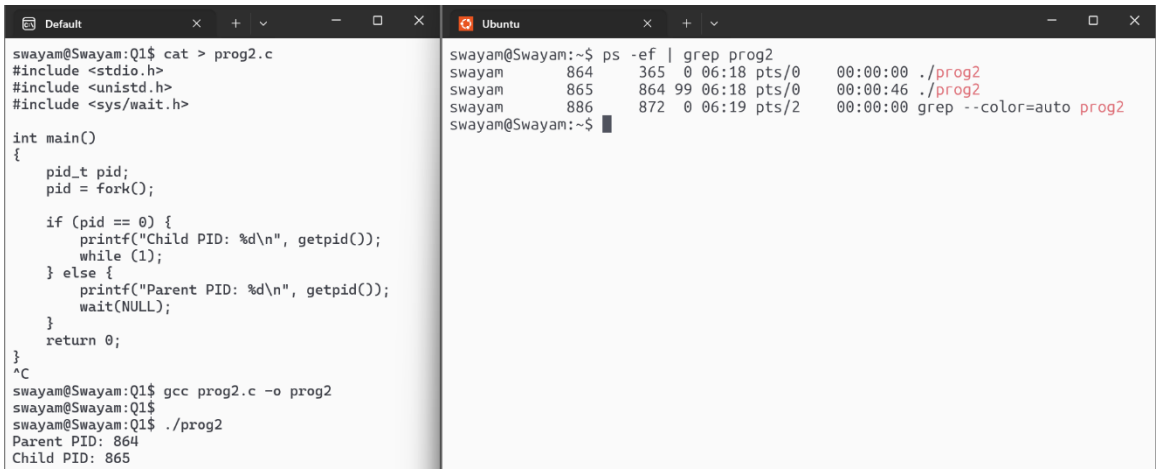
b) Terminate the child process. Then trace the state of processes.



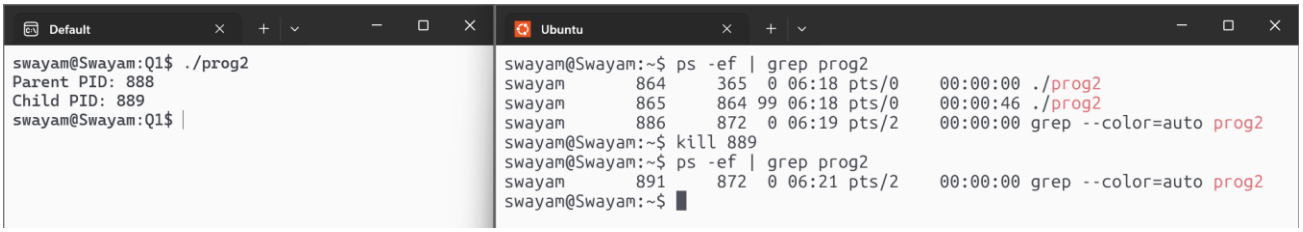
c) Run the program and trace the state of both processes. Terminate the parent process. Then trace the state of processes.



d) Modify the program so that the parent process after displaying the message will wait for child process to complete its task. Again run the program and trace the state of both processes.



e) Terminate the child process. Then trace the state of processes.



2. Trace the output of the following codes:

a)
```
int main( )
{
    if(fork()==0)
        printf("1");
    else
        printf("2");
    printf("3");
return 0;
}
```

```
swayam@Swayam:Q2$ gcc prog1.c -o prog1
swayam@Swayam:Q2$ ./prog1
2313swayam@Swayam:Q2$
```

b)
```
int main( )
{
    if(vfork()==0)
    {
        printf("1");
        _exit(0);

    }
    else
        printf("2");
        printf("3");
}
```

```
swayam@Swayam:Q2$ gcc prog2.c -o prog2
swayam@Swayam:Q2$ ./prog2
123swayam@Swayam:Q2$
```

c)
```
int main( )
{
  pid_t pid; int
 i=5;
 pid=fork();
 i=i+1; if(pid=
 =0)
 {
      printf("Child: %d",i);
  }
  else

  {
      wait(NULL);
      printf("Parent: %d",i);
   }
   return 0;
}
```

```
swayam@Swayam:Q2$ gcc prog3.c -o prog3
swayam@Swayam:Q2$ ./prog3
Child: 6
Parent: 6
swayam@Swayam:Q2$
```

d)
```
int main( )
{
  pid_t pid; int
  i=5;
  pid=vfork();
  i=i+1;
  if(pid==0)
 {
      printf("Child: %d",i);
      _exit(0);
  }

  else
 {
      printf("Parent: %d",i);
  }
  return 0;
}
```

```
swayam@Swayam:Q2$ gcc prog4.c -o prog4
swayam@Swayam:Q2$ ./prog4
Child: 6Parent: 7swayam@Swayam:Q2$
```

e)

```
int main( )
{
 pid_t pid;
 int i=5;
 pid=fork();
 if(pid= =0)
 {
    i=i+1;
    printf("Child: %d",i);
 }
  else
 {
   wait(NULL);
   printf("Parent: %d",i);
  }
  return 0;
 }
```

```
swayam@Swayam:Q2$ gcc prog5.c -o prog5
swayam@Swayam:Q2$ ./prog5
Child: 6Parent: 5swayam@Swayam:Q2$
```

f)

```
int main( )
{
  pid_t pid;
  int i=5;
   pid=vfork();
   if(pid==0)
   {
      i=i+1;
      printf("Child: %d",i);
       _exit(0);
   }
   else
   {
      printf("Parent: %d",i);
   }
   return 0;
  }
```

```
swayam@Swayam:Q2$ gcc prog6.c -o prog6
swayam@Swayam:Q2$ ./prog6
Child: 6Parent: 6swayam@Swayam:Q2$
```

g)

```
int main( )
{
 int i=5;
 if(fork( )==0)
 {
     printf("Child: %d",i);
 }
 else
 {
    printf("Parent: %d",i);
 }
 return 0;
}
```

```
swayam@Swayam:Q2$ gcc prog7.c -o prog7
swayam@Swayam:Q2$ ./prog7
Parent: 5Child: 5swayam@Swayam:Q2$
```

h)

```
int main( )
{
 int i=5;
 if(vfork( )==0)
 {
    printf("Child: %d",i);
    _exit(0);
 }
 else
 {
    printf("Parent: %d",i);
 }
  return 0;
}
```

```
swayam@Swayam:Q2$ gcc prog8.c -o prog8
swayam@Swayam:Q2$ ./prog8
Child: 5Parent: 5swayam@Swayam:Q2$
```

i)
```
int main( )
{
 if(fork( )==0)
 {
     printf("1");
 }
 else
 {
      wait(NULL);
      printf("2");
      printf("3");
  }
  return 0;
 }
```

```
swayam@Swayam:Q2$ gcc prog9.c -o prog9
swayam@Swayam:Q2$ ./prog9
123swayam@Swayam:Q2$
```

j)
```
int main( )
{
 if(vfork( )==0)
 {
     printf("1");
     _exit(0);
 }
 else
 {
     printf("2");
    printf("3");
 }
 return 0;
}
```

```
swayam@Swayam:Q2$ gcc prog10.c -o prog10
swayam@Swayam:Q2$ ./prog10
123swayam@Swayam:Q2$
```

k)
```
int main( )
{
pid_t c1;
int n=10;
c1=fork( );
if(c1==0)
{
     printf(" Child\n");
     n=20;
     printf("n=%d \n",n);
 }
 else
{
      wait(NULL);
      printf("Parent\n");
      printf("n=%d \n",n);
 }
  return 0;
}
```

```
swayam@Swayam:Q2$ gcc prog11.c -o prog11
swayam@Swayam:Q2$ ./prog11
Child
n = 20
Parent
n = 10
swayam@Swayam:Q2$
```

l)
```
int main( )
{
 pid_t c1;
 int n=10;
 c1=vfork( );
 if(c1==0)
 {
      printf(" Child\n");
      n=20;
      printf("n=%d \n",n);
      _exit(0);
  }
  else
  {
       printf("Parent\n");
       printf("n=%d \n",n);
  }
   return 0;
  }
```

```
swayam@Swayam:Q2$ gcc prog12.c -o prog12
swayam@Swayam:Q2$ ./prog12
 Child
n=20
Parent
n=20
swayam@Swayam:Q2$
```

m)
```
int main( )
{
 int i=5;
 fork();
 i=i+1;
 fork();
 printf ( "% d",i);
 return 0;
}
```

```
swayam@Swayam:Q2$ gcc prog13.c -o prog13
swayam@Swayam:Q2$ ./prog13
6666swayam@Swayam:Q2$
```

n)
```
int main( )
{
 pid_t pid;
 int i=5;
 pid=vfork();
 if(pid==0)
 {
      printf("Child: %d",i);
      _exit(0);
 }
  else
  {
       i=i+1;
       printf("Parent: %d",i);
   }
   return 0;
}
```

```
swayam@Swayam:Q2$ gcc prog14.c -o prog14
swayam@Swayam:Q2$ ./prog14
Child: 5
Parent: 6
swayam@Swayam:Q2$
```

o)
```
int main( )
{
 int i=5;
 if(fork()==0)
 i=i+1;
 else
 i=i-1;
 printf("%d",i);
 return 0;
}
```

```
swayam@Swayam:Q2$ cat prog15.c
#include <stdio.h>
#include<unistd.h>
int main( ){
        int i=5;
        if(fork()==0)
        i=i+1;
        else
        i=i-1;
        printf("%d",i);
        return 0;
}
swayam@Swayam:Q2$ gcc prog15.c -o prog15
swayam@Swayam:Q2$ ./prog15
46swayam@Swayam:Q2$
```

p)
```
int main( )
{
 int i=5;
 if(vfork()==0)
 {
      i=i+1;
      _exit(0);
  }
  else
       i=i-1;
   fprintf(stderr,"%d",i);
   return 0;
   }
```

```
swayam@Swayam:Q2$ cat prog16.c
#include <stdio.h>
#include<unistd.h>
int main(){
        int i=5;
        if(vfork()==0){
                i=i+1;
                _exit(0);
        }
        else
                i=i-1;
        fprintf(stderr,"%d",i);
        return 0;
}
swayam@Swayam:Q2$ gcc prog16.c -o prog16
swayam@Swayam:Q2$ ./prog16
5swayam@Swayam:Q2$
```

q)
```
int main( )
{
int j,i=5;
for(j=1;j<3;j++)
{
    if(fork()==0)
    {
        i=i+1;
        break;
    }
else
  wait(NULL);
}
printf("%d",i);
return 0;
}
```

```
swayam@Swayam:Q2$ cat prog17.c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    int j, i = 5;

    for (j = 1; j < 3; j++) {
        if (fork() == 0) {   // Child
            i = i + 1;
            break;
        } else {             // Parent
            wait(NULL);
        }
    }

    printf("%d\n", i);
    return 0;
}
swayam@Swayam:Q2$ gcc prog17.c -o prog17
swayam@Swayam:Q2$ ./prog17
6
6
5
swayam@Swayam:Q2$
```

r)
```
int main( )
{
int j,i=5;
for(j=1;j<3;j++)
{
    if(fork()!=0)
    {
        i=i-1;
        break;
    }
}
fprintf(stderr,"%d",i);
return 0;
}
```

```
swayam@Swayam:Q2$ cat prog18.c
#include <stdio.h>
#include<unistd.h>
int main(){
        int j,i=5;
        for(j=1;j<3;j++){
                if(fork()!=0){
                        i=i-1;
                        break;
                }
        }
        fprintf(stderr,"%d",i);
        return 0;
}
swayam@Swayam:Q2$ gcc prog18.c -o prog18
swayam@Swayam:Q2$ ./prog18
4swayam@Swayam:Q2$ 45
```

s)
```
int main( )
{
if(fork() == 0)
if(fork())
printf("1\n");

return 0;
}
```

```
swayam@Swayam:Q2$ cat prog19.c
#include <stdio.h>
#include<unistd.h>
int main(){
        if(fork() == 0)
        if(fork())
        printf("1\n");
        return 0;
}
swayam@Swayam:Q2$ gcc prog19.c -o prog19
swayam@Swayam:Q2$ ./prog19
swayam@Swayam:Q2$ 1
```

t)
```
void fun1(){
fork();
fork();
printf("1\n");
}
 int main() {
 fun1();
 printf("1\n");
 return 0;
 }
```

```
swayam@Swayam:Q2$ gcc prog20.c -o prog20
swayam@Swayam:Q2$ ./prog20
1
1
1
1
1
1
1
1
swayam@Swayam:Q2$
```

3. Write a C program that will create three child process to perform the following operations respectively:
    - First child will copy the content of file1 to file2
    - Second child will display the content of file2
    - Third child will display the sorted content of file2 in reverse order.
    - Each child process being created will display its id and its parent process id with appropriate message.
    - The parent process will be delayed for 1 second after creation of each child process. It will display appropriate message with its id after completion of all the child processes.

Ubuntu

```
swayam@Swayam:Q3$ echo -e "banana\napple\ncherry" > file1
swayam@Swayam:Q3$ cat > q3.c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main()
{
    pid_t pid;

    pid = fork();
    if (pid == 0) {
        printf("First Child | PID=%d PPID=%d\n", getpid(), getppid());
        execlp("cp", "cp", "file1", "file2", NULL);
    }
    sleep(1);

    pid = fork();
    if (pid == 0) {
        printf("Second Child | PID=%d PPID=%d\n", getpid(), getppid());
        execlp("cat", "cat", "file2", NULL);
    }
    sleep(1);

    pid = fork();
    if (pid == 0) {
        printf("Third Child | PID=%d PPID=%d\n", getpid(), getppid());
        execlp("sort", "sort", "-r", "file2", NULL);
    }

    wait(NULL);
    wait(NULL);
    wait(NULL);

    printf("Parent Process Completed | PID=%d\n", getpid());
    return 0;
}
^C
swayam@Swayam:Q3$ gcc q3.c -o q3
swayam@Swayam:Q3$ ./q3
First Child | PID=410 PPID=409
Second Child | PID=411 PPID=409
banana
apple
cherry
Third Child | PID=412 PPID=409
cherry
banana
apple
Parent Process Completed | PID=409
swayam@Swayam:Q3$
```

4. Write a C program that will create a child process to generate a Fibonacci series of specified length and store it in an array. The parent process will wait for the child to complete its task and then display the Fibonacci series and then display the prime Fibonacci number in the series along with its position with appropriate message.

Ubuntu

```
swayam@Swayam:Lab4$ cat > q4.c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/mman.h>

int isPrime(int n)
{
    if (n < 2) return 0;
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0) return 0;
    return 1;
}

int main()
{
    int n;
    printf("Enter length of Fibonacci series: ");
    scanf("%d", &n);

    int *fib = mmap(NULL, n * sizeof(int),
                    PROT_READ | PROT_WRITE,
                    MAP_SHARED | MAP_ANONYMOUS, -1, 0);

    pid_t pid = fork();

    if (pid == 0) {
        fib[0] = 0;
        if (n > 1) fib[1] = 1;
        for (int i = 2; i < n; i++)
            fib[i] = fib[i - 1] + fib[i - 2];
        _exit(0);
    } else {
        wait(NULL);

        printf("Fibonacci Series:\n");
        for (int i = 0; i < n; i++)
            printf("%d ", fib[i]);
        printf("\n");

        printf("Prime Fibonacci Numbers:\n");
        for (int i = 0; i < n; i++) {
            if (isPrime(fib[i]))
                printf("Prime %d at position %d\n", fib[i], i + 1);
        }
    }

    return 0;
}
^C
swayam@Swayam:Lab4$ gcc q4.c -o q4
swayam@Swayam:Lab4$ ./q4
Enter length of Fibonacci series: 7
Fibonacci Series:
0 1 1 2 3 5 8
Prime Fibonacci Numbers:
Prime 2 at position 4
Prime 3 at position 5
Prime 5 at position 6
swayam@Swayam:Lab4$
```