Familiarization with Process Management in Linux environment

Q1. 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.

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.

Program:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(){
   pid_t pid = fork();
   printf("pid value = %d\n", pid);
   if(pid < 0){
      printf("fork fail\n");
   }
   else if(pid == 0){
      printf("I am the child.\n");
```

```
    printf("Child process pid = %d\n", getpid());
    printf("Parent process pid = %d\n", getppid());
  }
  else {
    printf("I am the parent.\n");
    printf("Child process pid = %d\n", getpid());
    printf("Parent process pid = %d\n", getppid());
  }
}
```

Output:

```
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit q1.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc q1.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
pid value = 7921
I am the parent.
Child process pid = 7920
Parent process pid = 7859
pid value = 0
I am the child.
Child process pid = 7921
Parent process pid = 1884
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$
```

Q2.Trace the output of the following codes.

a)

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

```
}
```

b)

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

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;
```

```
}
```

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;
}
```

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;
}
```

f)

```
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;
}
```

g)

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

```
}
```

h)

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

i)

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

Output:

```
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit q2a.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc q2a.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
2313ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit q2b.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc q2b.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
123ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit q2c.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc q2c.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
123ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit q2d.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc q2d.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
Child: 6Parent: 7ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$
```

```
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit p11.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc p11.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
123ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit p12.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc p12.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
123ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit p13.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc p13.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
 Child
n=20
Parent
n=10
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$
```

j)

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

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;
}
```

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;
}
```

m)

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

n)

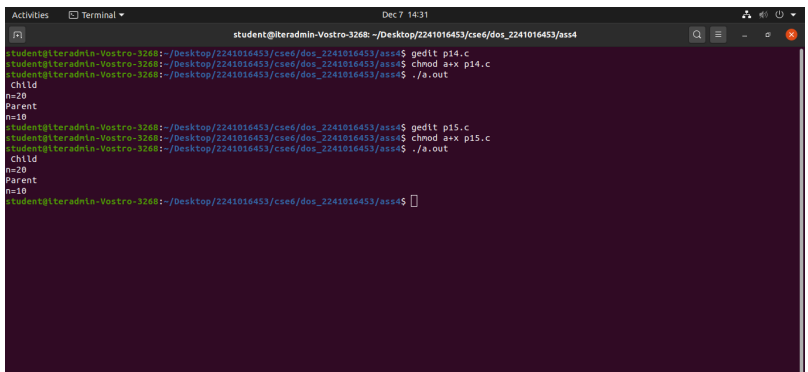
```
int main() {
    pid_t pid;
    int i = 5;
    pid = vfork();
    if(pid == 0) {
        printf("Child: %d", i);
        _exit(0);
    } else {
        i = i + 1;
        printf("Parent: %d", i);
```

```
    }
    return 0;
}
```

o)

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

```
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit p5.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc p5.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
Child: 6Parent: 6ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit p6.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc p6.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
Child: 6Parent: 7ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit p7.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc p7.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
Child: 6Parent: 5ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit p8.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc p8.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
Child: 6Parent: 6ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit p9.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc p9.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
Parent: 5Child: 5ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit p10.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc q10.c
cc1: fatal error: q10.c: No such file or directory
compilation terminated.
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc p10.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
Child: 5Parent: 5ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$
```



```
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ gedit p16.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ chmod a+x p16.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ ./a.out
 Child
n=20
Parent
n=10
```

```
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ gedit p17.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ chmod a+x p17.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ ./a.out
 Child
n=20
Parent
n=10
```

```
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ gedit p18.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ chmod a+x p19.c
chmod: cannot access 'p19.c': No such file or directory
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ chmod a+x p18.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ ./a.out
 Child
n=20
Parent
n=10
```

```
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ gedit p19.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ chmod a+x p19.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ ./a.out
 Child
n=20
Parent
n=10
```

p)

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

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;
}
```

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;
}
```

s)

```
int main() {
```

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

```
    return 0;   }
```

```
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ gedit p20.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ chmod a+x p20.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ ./a.out
 Child
n=20
Parent
n=10
```

```
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ gedit p21.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ chmod a+x p21.c
student@iteradmin-Vostro-3268:~/Desktop/2241016453/cse6/dos_2241016453/ass4$ ./a.out
 Child
n=20
Parent
n=10
```

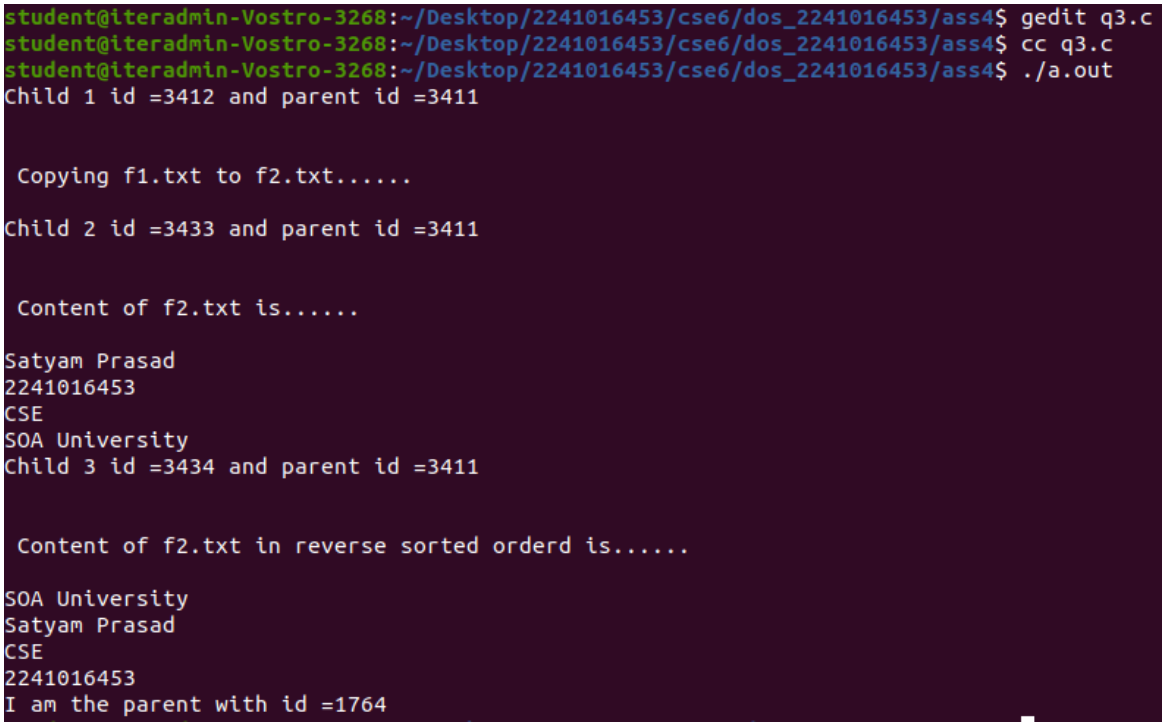
Q3. Write a C program that will create three child process to perform the following operations respectively: - First child will copy the content of file1 to file2 - Second child will display the content of file2 - Third child will display the sorted content of file2 in reverse order. - Each child process 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.

PROGRAM:

```
#include<stdio.h>

#include<unistd.h>

int main() {

   if(fork() > 0)

      sleep(1);

   else {
```

```
    printf("Child 1 id = %d and parent id = %d\n", getpid(), getppid());
    printf("\n\n Copying f1.txt to f2.txt.....\n\n");
    execlp("cp", "cp", "-f", "f1.txt", "f2.txt", NULL);
  }
  if(fork() > 0)
    sleep(1);
  else {
    printf("Child 2 id = %d and parent id = %d\n", getpid(), getppid());
    printf("\n\n Content of f2.txt is.....\n\n");
    execlp("cat", "cat", "f2.txt", NULL);
  }
  if(fork() > 0)
    sleep(1);
  else {
    printf("Child 3 id = %d and parent id = %d\n", getpid(), getppid());
    printf("\n\n Content of f2.txt in reverse sorted order is.....\n\n");
    execlp("sort", "sort", "-r", "f2.txt", NULL);
  }
  printf("I am the parent with id = %d\n", getppid());
  return 0;
}
```

OUTPUT:



Q4. 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.

Program:

```
#include<stdio.h>

#include<stdlib.h>
```

```
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
 int prime(int a) {
   if(a == 1)
      return 0;
   for(int i = 2; i < a; i++) {
      if(a % i == 0)
         return 0;
   }
   return a;
}
 int main() {
   printf("Enter the length: ");
   int n = 0;
   scanf("%d", &n);
   int fib[n];
   pid_t pid = vfork();
   if(pid == 0) {
      int a = 0, b = 1;
      fib[0] = a;
      fib[1] = b;
       int k = 2;
       int c = n;
       while(c > 0) {
```

```
        a = a + b;
        b = a + b;
        fib[k++] = a;
        fib[k++] = b;
        c -= 2;
      }
      _exit(0);
    } else {
      wait(NULL);
      for(int i = 0; i < n; i++)
        printf("%d,", fib[i]);
      printf("\nPrimes are:\n");
      for(int i = 0; i < n; i++) {
        if(prime(fib[i]))
          printf("%d : %d \n", fib[i], i);
      }
}
}
```

OUTPUT:

```
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ gedit q4.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ cc q4.c
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$ ./a.out
Enter the length: 5
0,1,1,2,3,
Primes are:
2 : 3
3 : 4
ubuntu@ubuntu:~/Desktop/2241016453_dos/ass4$
```