LAB 8

System calls for process creation(fork and its related calls)

Anjal K K
17 R4

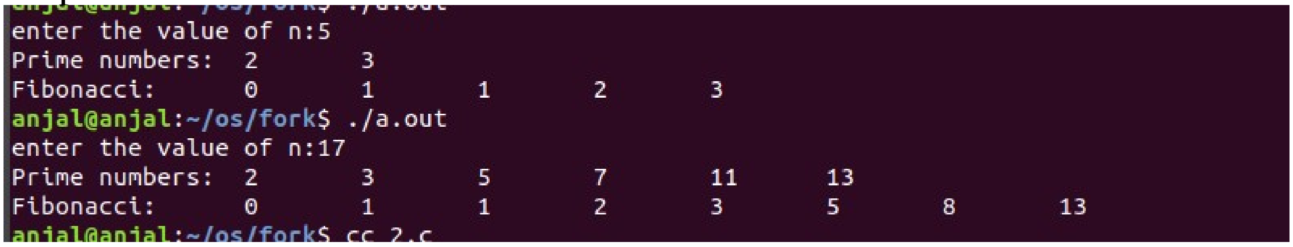
1.Program to accept the limiting value ‘n’ as input and genrate the Fibonacci sequence of n numbers using the child process while the parent process generate the first n prime numbers.

Code:

```
#include <stdio.h>
#include <unistd.h>
void main(){
  int n;
  printf("enter the value of n:");
  scanf("%d",&n);
  if(fork()==0){
    int a=0,b=1;
    printf("Fibonacci:\t");
    printf("%d\t",a);
    int sum=1;
    while(sum<n){
      printf("%d\t",sum);
      sum=a+b;
      a=b;
      b=sum;
    }
    printf("\n");
  }
  else
  {
    wait(NULL);
    printf("Prime numbers:\t");
    for(int i=1;i<n;i++){
      int count=0;
      for(int j=2;j<=i/2;j++){
        if(i%j==0){
          count++;
          break;
        }
      }
      if(count==0 && i!=1)
        printf("%d\t",i);
    }
    printf("\n");
  }

}
```

output:



```
enter the value of n:5
Prime numbers:  2       3
Fibonacci:      0       1       1       2       3
anjal@anjal:~/os/fork$ ./a.out
enter the value of n:17
Prime numbers:  2       3       5       7       11      13
Fibonacci:      0       1       1       2       3       5       8       13
anjal@anjal:~/os/fork$ cc 2.c
```

2.Generate an N level hierarchy of processes and also display the parent id of the process

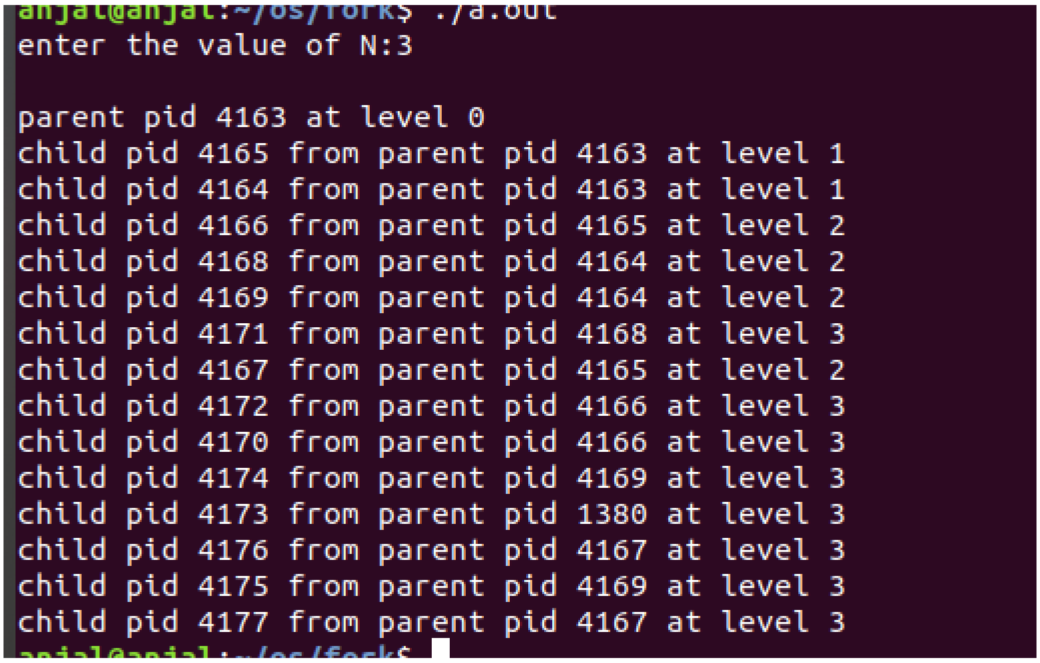
code:

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

void main(){
  int n;
  printf("enter the value of N:");
  scanf("%d",&n);
  printf("\nparent pid %d at level 0\n",getpid());
  for(int i=1;i<=n;i++){
    if(fork()==0)
      printf("child pid %d from parent pid %d at level %d\
n",getpid(),getppid(),i);
    else if(fork()==0)
```

```
      printf("child pid %d from parent pid %d at level %d\
n",getpid(),getppid(),i);
    else
    {
      wait(NULL);
      exit(0);
    }
  }
}
```

output:



```
anjal@anjal:~/os/fork$ ./a.out
enter the value of N:3

parent pid 4163 at level 0
child pid 4165 from parent pid 4163 at level 1
child pid 4164 from parent pid 4163 at level 1
child pid 4166 from parent pid 4165 at level 2
child pid 4168 from parent pid 4164 at level 2
child pid 4169 from parent pid 4164 at level 2
child pid 4171 from parent pid 4168 at level 3
child pid 4167 from parent pid 4165 at level 2
child pid 4172 from parent pid 4166 at level 3
child pid 4170 from parent pid 4166 at level 3
child pid 4174 from parent pid 4169 at level 3
child pid 4173 from parent pid 1380 at level 3
child pid 4176 from parent pid 4167 at level 3
child pid 4175 from parent pid 4169 at level 3
child pid 4177 from parent pid 4167 at level 3
anjal@anjal:~/os/fork$
```

3.

```
                    A
              B            C
           D  E   F            G
           H
           I
```

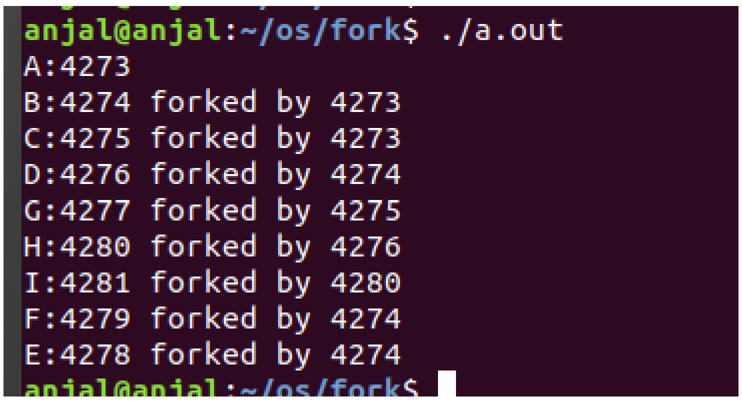
code:

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

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

void main(){
  printf("A:%d\n",getpid());
  if(fork()==0){
    printf("B:%d forked by %d\n",getpid(),getppid());
    if(fork()==0){
      printf("D:%d forked by %d\n",getpid(),getppid());
      if(fork()==0){
        printf("H:%d forked by %d\n",getpid(),getppid());
        if(fork()==0){
          printf("I:%d forked by %d\n",getpid(),getppid());
        }
        else
          wait(NULL);
      }
      else
        wait(NULL);
    }
    else if(fork()==0){
      printf("E:%d forked by %d\n",getpid(),getppid());
    }
    else if(fork()==0){
      printf("F:%d forked by %d\n",getpid(),getppid());
    }
    else
      wait(NULL);
  }  else if(fork()==0){
      printf("C:%d forked by %d\n",getpid(),getppid());
      if(fork()==0){
        printf("G:%d forked by %d\n",getpid(),getppid());
      }
      else
        wait(NULL);
    }
    else
      wait(NULL);
}
```

output:



```
anjal@anjal:~/os/fork$ ./a.out
A:4273
B:4274 forked by 4273
C:4275 forked by 4273
D:4276 forked by 4274
G:4277 forked by 4275
H:4280 forked by 4276
I:4281 forked by 4280
F:4279 forked by 4274
E:4278 forked by 4274
anjal@anjal:~/os/fork$
```