

# Laboratory Assignments 4

## Subject: Design of Operating Systems

### Subject code: CSE 4049

#### Assignment 4: Familiarization with Process Management in Unix 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.
  - 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.

#### Solution:

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

int main(){
    if(fork()==0){
        printf("child: %d\n",getpid());
        while(1){
        }
        exit(0);
    }
    else{
        printf("parent: %d\n",getpid());
        while(1){
        }
    }
    return 0;
}
```

```
student@D001-37: ~/2141019362/DOS_2141019362/lab4$ ./q1
parent: 1874
child: 1875
```

```
student@D001-37: ~/2141019362/DOS_2141019362/lab4$ ps -la
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
4 S 1001 971 968 2 80 0 - 134127 ep_pol tty2 00:00:29 Xorg
0 S 1001 1156 968 0 80 0 - 49272 do_sys tty2 00:00:00 gnome-session-b
0 S 1001 1799 1663 1 80 0 - 208170 do_sys pts/0 00:00:04 gedit
1 R 1001 1850 951 99 80 0 - 624 - pts/0 00:02:35 q1
0 T 1001 1867 1663 72 80 0 - 624 do_sig pts/0 00:00:52 q1
1 Z 1001 1868 1867 21 80 0 - 0 - pts/0 00:00:15 q1 <defunct>
0 R 1001 1874 1663 98 80 0 - 624 - pts/0 00:00:04 q1
1 R 1001 1875 1874 98 80 0 - 624 - pts/0 00:00:04 q1
0 R 1001 1876 1825 0 80 0 - 5102 - pts/1 00:00:00 ps

student@D001-37: ~/2141019362/DOS_2141019362/lab4$ kill -9 1875
student@D001-37: ~/2141019362/DOS_2141019362/lab4$ ps -la
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
4 S 1001 971 968 2 80 0 - 134111 ep_pol tty2 00:00:29 Xorg
0 S 1001 1156 968 0 80 0 - 49272 do_sys tty2 00:00:00 gnome-session-b
0 S 1001 1799 1663 1 80 0 - 208170 do_sys pts/0 00:00:04 gedit
1 R 1001 1850 951 99 80 0 - 624 - pts/0 00:02:46 q1
0 T 1001 1867 1663 63 80 0 - 624 do_sig pts/0 00:00:52 q1
1 Z 1001 1868 1867 18 80 0 - 0 - pts/0 00:00:15 q1 <defunct>
0 R 1001 1874 1663 97 80 0 - 624 - pts/0 00:00:15 q1
1 Z 1001 1875 1874 79 80 0 - 0 - pts/0 00:00:12 q1 <defunct>
0 R 1001 1880 1825 0 80 0 - 5102 - pts/1 00:00:00 ps
```

```
Activities Terminal Nov 22 14:31 student@D001-37: ~/2141019362/DOS_2141019362/lab4
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5 #include <sys/types.h>
6
7 int main(){
8     if(fork()==0){
9         printf("child :%d\n",getpid());
10        while(1){
11
12        }
13        exit(0);
14    }
15    else{
16        printf("parent: %d\n",getpid());
17        while(1){
18
19        }
20    }
21    return 0;
22 }
```

```
student@D001-37:~/2141019362/DOS_2141019362/lab4$ ./q1
parent: 2135
child :2136
Killed
student@D001-37:~/2141019362/DOS_2141019362/lab4$
```

```
student@D001-37:~/2141019362/DOS_2141019362/lab4$ ps -la
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
4 S 1001 971 968 2 80 0 - 136719 ep_pol tty2 00:00:31 Xorg
0 S 1001 1156 968 0 80 0 - 49272 do_sys tty2 00:00:00 gnome-session-b
0 S 1001 1799 1663 0 80 0 - 208170 do_sys pts/0 00:00:04 gedit
1 R 1001 1850 951 99 80 0 - 624 - pts/0 00:04:34 q1
0 T 1001 1867 1663 27 80 0 - 624 do_sig pts/0 00:00:52 q1
1 Z 1001 1868 1867 8 80 0 - 0 - pts/0 00:00:15 q1 <defunct>
0 T 1001 1874 1663 85 80 0 - 624 do_sig pts/0 00:01:46 q1
1 Z 1001 1875 1874 10 80 0 - 0 - pts/0 00:00:12 q1 <defunct>
0 R 1001 2135 1663 99 80 0 - 624 - pts/0 00:00:08 q1
1 R 1001 2136 2135 99 80 0 - 624 - pts/0 00:00:08 q1
0 R 1001 2140 1825 0 80 0 - 5102 - pts/1 00:00:00 ps
student@D001-37:~/2141019362/DOS_2141019362/lab4$ kill -9 2135
student@D001-37:~/2141019362/DOS_2141019362/lab4$ ps -la
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
4 S 1001 971 968 2 80 0 - 136703 ep_pol tty2 00:00:31 Xorg
0 S 1001 1156 968 0 80 0 - 49272 do_sys tty2 00:00:00 gnome-session-b
0 S 1001 1799 1663 0 80 0 - 208170 do_sys pts/0 00:00:04 gedit
1 R 1001 1850 951 99 80 0 - 624 - pts/0 00:04:46 q1
0 T 1001 1867 1663 25 80 0 - 624 do_sig pts/0 00:00:52 q1
1 Z 1001 1868 1867 7 80 0 - 0 - pts/0 00:00:15 q1 <defunct>
0 T 1001 1874 1663 78 80 0 - 624 do_sig pts/0 00:01:46 q1
1 Z 1001 1875 1874 9 80 0 - 0 - pts/0 00:00:12 q1 <defunct>
1 R 1001 2136 951 99 80 0 - 624 - pts/0 00:00:20 q1
0 R 1001 2141 1825 0 80 0 - 5102 - pts/1 00:00:00 ps
student@D001-37:~/2141019362/DOS_2141019362/lab4$
```

```
Activities Terminal Nov 22 14:35 student@D001-37: ~/2141019362/DOS_2141019362/lab4
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5 #include <sys/types.h>
6
7 int main(){
8
9     if(fork()==0){
10        printf("child :%d\n",getpid());
11        while(1){
12
13        }
14        exit(0);
15    }
16    else{
17        wait(NULL);
18        printf("parent: %d\n",getpid());
19        while(1){
20
21        }
22    }
23    return 0;
24 }
```

```
student@D001-37:~/2141019362/DOS_2141019362/lab4$ gcc q1.c -o q1
student@D001-37:~/2141019362/DOS_2141019362/lab4$ ./q1
child :2262
parent: 2261
student@D001-37:~/2141019362/DOS_2141019362/lab4$ gcc q1.c -o q1
student@D001-37:~/2141019362/DOS_2141019362/lab4$ ./q1
child :2286
```

```
student@D001-37:~/2141019362/DOS_2141019362/lab4$ ps -la
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
4 S 1001 971 968 2 80 0 - 136687 ep_pol tty2 00:00:34 Xorg
0 S 1001 1156 968 0 80 0 - 49272 do_sys tty2 00:00:00 gnome-session-b
0 S 1001 1799 1663 0 80 0 - 208170 do_sys pts/0 00:00:05 gedit
1 R 1001 1850 951 99 80 0 - 624 - pts/0 00:00:26 q1
0 T 1001 1867 1663 10 80 0 - 624 do_sig pts/0 00:00:52 q1
1 Z 1001 1868 1867 3 80 0 - 0 - pts/0 00:00:15 q1 <defunct>
0 T 1001 1874 1663 25 80 0 - 624 do_sig pts/0 00:01:46 q1
1 Z 1001 1875 1874 3 80 0 - 0 - pts/0 00:00:12 q1 <defunct>
0 R 1001 2136 951 99 80 0 - 624 - pts/0 00:04:59 q1
0 T 1001 2252 1663 3 80 0 - 624 do_sig pts/0 00:00:04 q1
0 T 1001 2253 2252 3 80 0 - 624 do_sig pts/0 00:00:04 q1
0 S 1001 2285 1663 0 80 0 - 591 do_wai pts/0 00:00:00 q1
0 R 1001 2286 2285 99 80 0 - 624 - pts/0 00:00:24 q1
0 R 1001 2302 1825 0 80 0 - 5102 - pts/1 00:00:00 ps
student@D001-37:~/2141019362/DOS_2141019362/lab4$
```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5 #include <sys/types.h>
6
7 int main()
8 {
9     if(fork()==0){
10         printf("child: %d\n",getpid());
11         while(1){}
12         exit(0);
13     }
14     else{
15         wait(NULL);
16         printf("parent: %d\n",getpid());
17         while(1){}
18     }
19 }
20
21 return 0;

```

```

student@D001-37: ~/2141019362/DOS_2141019362/lab4$ gcc q1.c -o q1
student@D001-37: ~/2141019362/DOS_2141019362/lab4$ ./q1
child: 2262
parent: 2261
student@D001-37: ~/2141019362/DOS_2141019362/lab4$ gcc q1.c -o q1
student@D001-37: ~/2141019362/DOS_2141019362/lab4$ ./q1
child: 2286
parent: 2285

```

```

pa: command not found
student@D001-37: ~/2141019362/DOS_2141019362/lab4$ ps -la
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1001      971       0  0  0  0 - 136687 ep_pol tty2      00:00:34 Xorg
0 S  1001     1156       0  0  0  0 - 492722 do_sys tty2      00:00:00 gnome-session-b
0 S  1001     1799     1663  0  0  0 - 208170 do_sys pts/0      00:00:05 gedit
1 R  1001     1850     951  99  80  0 - 624 - pts/0      00:09:26 q1
0 T  1001     1867     1663  10  80  0 - 624 do_sig pts/0      00:00:52 q1
1 Z  1001     1868     1867  3  80  0 - 0 - pts/0      00:00:15 q1 <defunct>
0 T  1001     1874     1663  25  80  0 - 624 do_sig pts/0      00:01:46 q1
1 Z  1001     1875     1874  3  80  0 - 0 - pts/0      00:00:12 q1 <defunct>
1 R  1001     2136     951  99  80  0 - 624 - pts/0      00:04:59 q1
0 T  1001     2252     1663  3  80  0 - 624 do_sig pts/0      00:00:04 q1
1 T  1001     2253     2252  3  80  0 - 624 do_sig pts/0      00:00:04 q1
0 S  1001     2285     1663  0  80  0 - 591 do_wai pts/0      00:00:00 q1
1 R  1001     2286     2285  99  80  0 - 624 - pts/0      00:00:24 q1
0 R  1001     2302     1825  0  80  0 - 5102 - pts/1      00:00:00 ps
student@D001-37: ~/2141019362/DOS_2141019362/lab4$ kill -9 2286
student@D001-37: ~/2141019362/DOS_2141019362/lab4$ ps -la
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1001      971       0  0  0  0 - 136703 ep_pol tty2      00:00:36 Xorg
0 S  1001     1156       0  0  0  0 - 492722 do_sys tty2      00:00:00 gnome-session-b
0 S  1001     1799     1663  0  0  0 - 208170 do_sys pts/0      00:00:06 gedit
1 R  1001     1850     951  99  80  0 - 624 - pts/0      00:10:34 q1
0 T  1001     1867     1663  9  80  0 - 624 do_sig pts/0      00:00:52 q1
1 Z  1001     1868     1867  2  80  0 - 0 - pts/0      00:00:15 q1 <defunct>
0 T  1001     1874     1663  21  80  0 - 624 do_sig pts/0      00:01:46 q1
1 Z  1001     1875     1874  2  80  0 - 0 - pts/0      00:00:12 q1 <defunct>
1 R  1001     2136     951  99  80  0 - 624 - pts/0      00:06:08 q1
0 T  1001     2252     1663  2  80  0 - 624 do_sig pts/0      00:00:04 q1
1 T  1001     2253     2252  2  80  0 - 624 do_sig pts/0      00:00:04 q1
0 R  1001     2285     1663  13  80  0 - 624 - pts/0      00:00:12 q1
0 R  1001     2420     1825  0  80  0 - 5102 - pts/1      00:00:00 ps

```

2. Trace the output of the following codes:

<p>a) <code>int main( )</code>  <code>{</code>              <code>if(fork()==0)</code>  <code>printf("1");</code>        <code>else</code>              <code>printf("2");</code>  <code>printf("3");</code>  <code>return 0; }</code></p>	<p>b) <code>int main( )</code>  <code>{</code>              <code>if(vfork()==0)</code>  <code>{</code>                  <code>printf("1");</code>              <code>_exit(0);</code>              <code>}</code>              <code>else</code>              <code>printf("2");</code>              <code>printf("3");</code>              <code>}</code></p>

<pre> 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; } </pre>	<pre> 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; } </pre>
<pre> 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; } </pre>	<pre> 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; } </pre>

<pre> g)  int main( )     { int i=5;     if(fork( )==0)     {         printf("Child: %d",i);     } else     {         printf("Parent: %d",i);     }     return 0;     } </pre>	<pre> h)  int main( )     { int i=5;     if(vfork( )==0)     {         printf("Child: %d",i);         _exit(0);     } else     {         printf("Parent: %d",i);     }     return 0;     } </pre>
--	---

<pre> i)  int main( )     {     if(fork( )==0)     {         printf("1");     } else     {         wait(NULL);     }     printf("2");     printf("3");     }     return 0;     } </pre>	<pre> j)  int main( )     {     if(vfork( )==0)     {         printf("1");         _exit(0);     }     else     {         printf("2");         printf("3");     }     return 0;     } </pre>
---	--

<pre> 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;     } </pre>	<pre> 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;     } </pre>
---	--

<pre> m)  int main( )     { int i=5; fork(); i=i+1; fork();     fprintf ( stderr,"%d",i);     return 0;     } </pre>	<pre> 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; } </pre>
<pre> o)  int main( )     {      int i=5; if(fork()==0)     i=i+1; else    i=i-1;     fprintf(stderr,"%d",i);     return 0;     } </pre>	<pre> p)  int main( )     { int i=5;     if(vfork()==0)     { i=i+1; _exit(0);     }     else i=i-1;         fprintf(stderr,"%d",i);         return 0;     } </pre>
<pre> q)  int main( )     { int j,i=5;     for(j=1;j&lt;3;j++)     {         if(fork()==0)         { i=i+1; break;         }     else         wait(NULL);     }     fprintf(stderr,"%d",i);     return 0;     } </pre>	<pre> r)  int main( )     {         int j,i=5;         for(j=1;j&lt;3;j++)         {             if(fork()!=0)             { i=i-1; break;             }         }         fprintf(stderr,"%d",i);         return 0;     } </pre>

<pre> s)  int main( )     {         if(fork() == 0)     if(fork())     printf("1\n");     return 0;     } </pre>	<pre> t)  void fun1(){         fork();     fork();         printf("1\n");     }      int      main() {         fun1();         printf("1\n");         return 0;     } </pre>
--	--

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.

## Solution:

The screenshot shows a C program in a text editor and its execution in a terminal. The program creates three child processes using `fork()` and `execlp()`. Each child process performs a specific task: copying file1 to file2, displaying file2's content, and displaying the sorted content of file2 in reverse order. The parent process delays for 1 second after each child creation and displays its ID after all children complete.

```

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

int main(){
    pid_t child1=fork();
    if(child1==0){
        printf("Child1: %d Parent: %d\n",getpid(),getppid());
        execlp("cp","cp","file1.txt","file2.txt",NULL);
    }
    else{
        sleep(1);
    }

    pid_t child2=fork();
    if(child2==0){
        printf("Child2: %d Parent: %d\n",getpid(),getppid());
        execlp("cat","cat","file2.txt",NULL);
    }
    else{
        sleep(1);
    }

    pid_t child3=fork();
    if(child3==0){
        printf("Child3: %d Parent: %d\n",getpid(),getppid());
        execlp("sort","sort","-r","file2.txt",NULL);
    }
    else{
        sleep(1);
    }

    return 0;
}

```

Terminal Output:

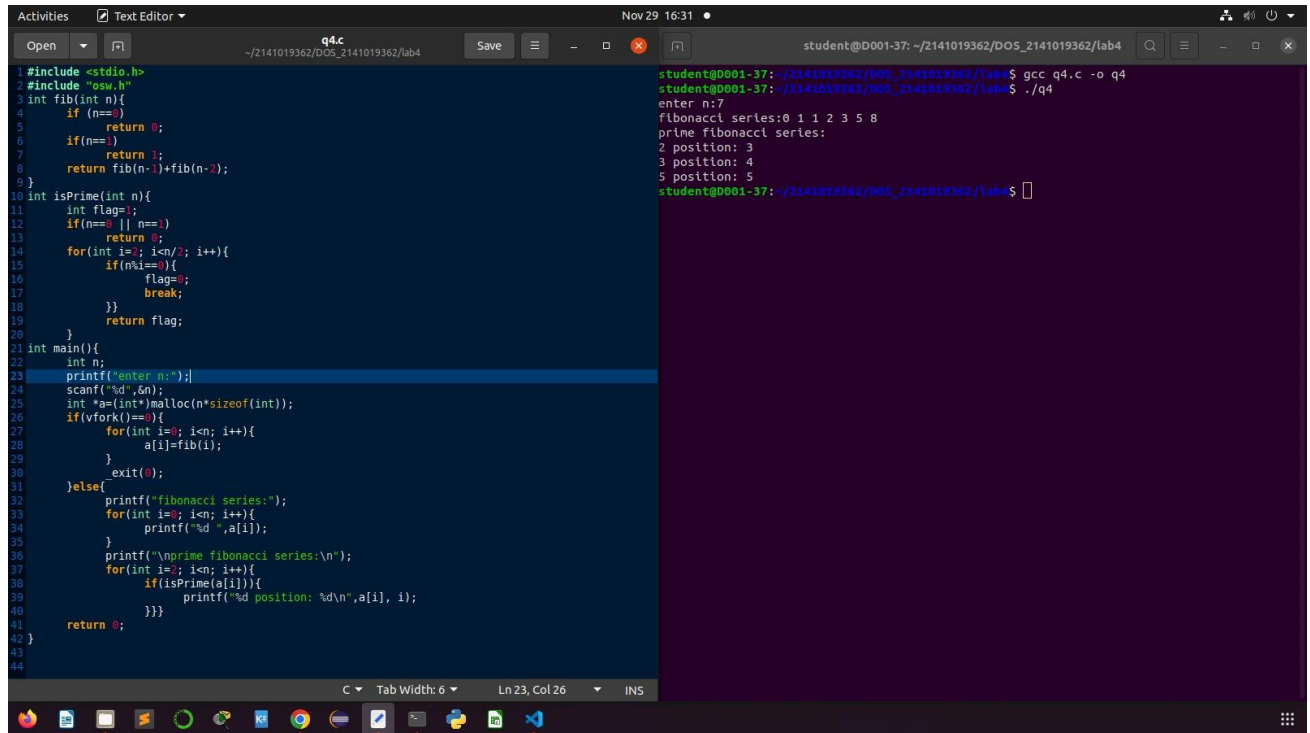
```

student@D001-37: ~/2141019362/DOS_2141019362/lab4
3 | execlp("cp","cp","file1.txt","file2.txt");
~~~~~
::22:16: error: 'file2' undeclared (first use in this function); did you mean 'fileno'
2 | execlp("cat","file2,NULL);
~~~~~
::22:16: note: each undeclared identifier is reported only once for each function it
in
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$ gcc q3.c -o q3
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$ ./q3
d1: 4373 Parent: 4372
d2: 4374 Parent: 4372
d3: 4375 Parent: 4374
d3: 4376 Parent: 4372
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$ gedit file2.txt
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$ gcc q3.c -o q3
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$ ./q3
d1: 4455 Parent: 4454
d2: 4456 Parent: 4454
d3: 4457 Parent: 4456
d3: 4458 Parent: 4454
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$ cat file2.txt
file2.txt: No such file or directory
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$ cat file2.txt
rsh
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$ gcc q3.c -o q3
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$ ./q3
d1: 4480 Parent: 4479
d2: 4481 Parent: 4479
rsh
d3: 4482 Parent: 4479
rsh
lent@D001-37: ~/2141019362/DOS_2141019362/lab4$

```

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.

### Solution:



```
1 #include <stdio.h>
2 #include "osw.h"
3 int fib(int n){
4     if (n==0)
5         return 0;
6     if (n==1)
7         return 1;
8     return fib(n-1)+fib(n-2);
9 }
10 int isPrime(int n){
11     int flag=1;
12     if(n==0 || n==1)
13         return 0;
14     for(int i=2; i<n/2; i++){
15         if(n%i==0){
16             flag=0;
17             break;
18         }
19     }
20     return flag;
21 }
22 int main(){
23     int n;
24     printf("enter n:");
25     scanf("%d",&n);
26     int *a=(int*)malloc(n*sizeof(int));
27     if(vfork()==0){
28         for(int i=0; i<n; i++){
29             a[i]=fib(i);
30         }
31         exit(0);
32     }else{
33         printf("fibonacci series:");
34         for(int i=0; i<n; i++){
35             printf("%d ",a[i]);
36         }
37         printf("\nprime fibonacci series:\n");
38         for(int i=0; i<n; i++){
39             if(isPrime(a[i])){
40                 printf("%d position: %d\n",a[i], i);
41             }
42         }
43     }
44     return 0;
45 }
```

```
student@D001-37: ~/Z141019362/DOS_2141019362/lab4$ gcc q4.c -o q4
student@D001-37: ~/Z141019362/DOS_2141019362/lab4$ ./q4
enter n:7
fibonacci series:0 1 1 2 3 5 8
prime fibonacci series:
2 position: 3
3 position: 4
5 position: 5
student@D001-37: ~/Z141019362/DOS_2141019362/lab4$
```