

SIKSHA 'O' ANUSANDHAN

DEEMED TO BE UNIVERSITY

Admission Batch : 2021 - 25

Session : 2023 - 24

Laboratory Assignment #4

DESIGN OF OPERATING SYSTEMS (CSE 4049)

Submitted By -

Name : Arya Aditya Dash
Registration No. : 2141004081
Branch & Section : CSE - B
Semester : 5th Semester



**Department of Computer Science & Engineering
Faculty of Engineering & Technology (ITER)**

Jagamohan Nagar, Jagamara, Bhubaneswar, Odisha – 751030

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 (), execl ()) for process handling in Unix environment.

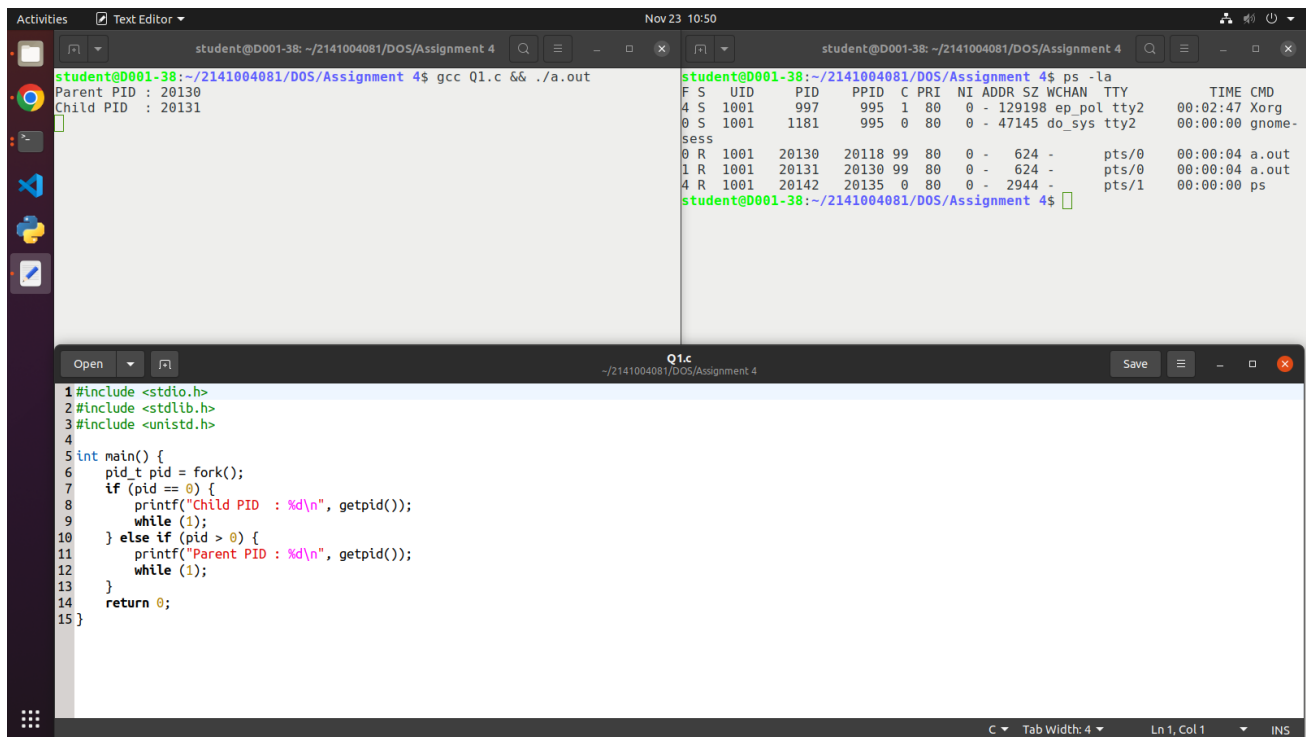
Q1. Write a C program to create a child process using fork() system call. The child process will print the message "Child" with its process identifier and then continue in an indefinite loop. The parent process will print the message "Parent" with its process identifier and then continue in an indefinite 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

Command:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    pid_t pid = fork();
    if (pid == 0) {
        printf("Child PID : %d\n", getpid());
        exit (0);
    } else if (pid > 0) {
        printf("Parent PID : %d\n", getpid());
        wait (NULL);
        while (1);
    }
    return 0;
}
```

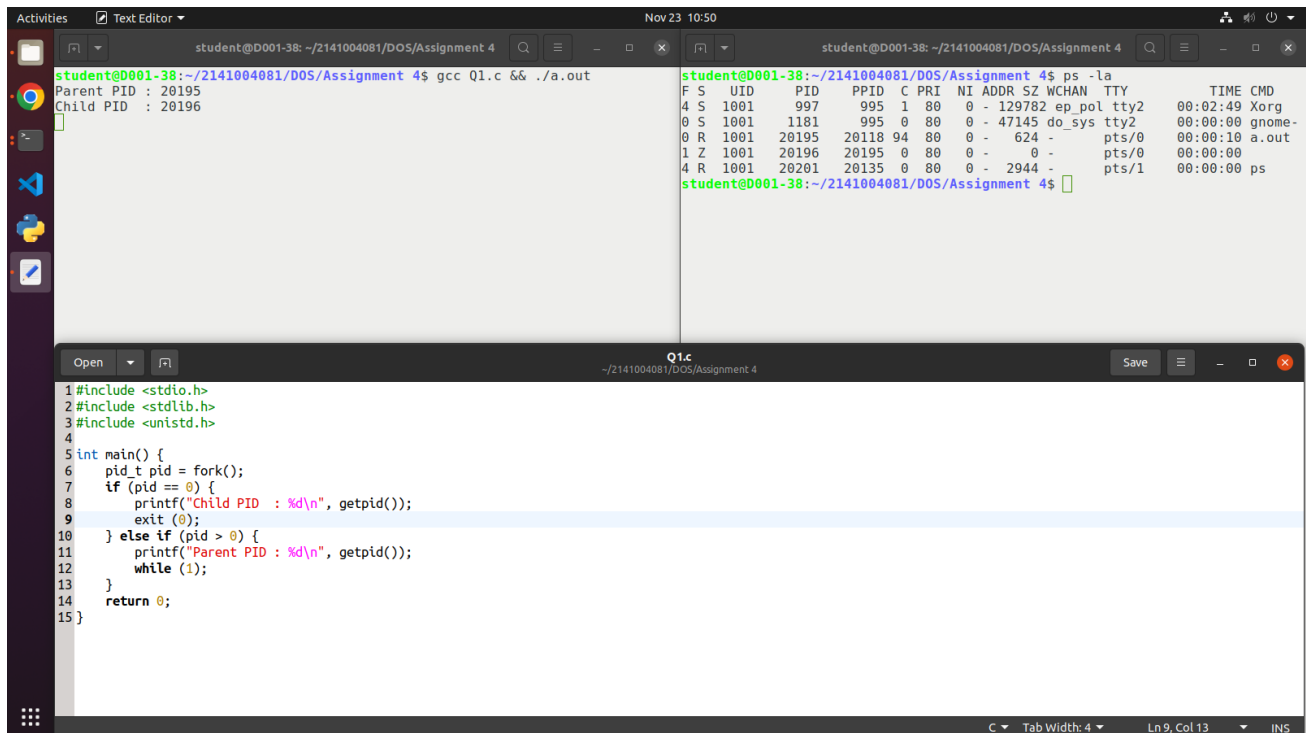
Output:



```
student@D001-38: ~/2141004081/DOS/Assignment 4$ gcc Q1.c && ./a.out
Parent PID : 20130
Child PID : 20131
```

```
student@D001-38:~/2141004081/DOS/Assignment 4$ ps -la
F S  UID      PID     PPID  C  PRI  NI ADDR SZ WCHAN  TTY      TIME CMD
4 S  1001      997      995  1   80   0 - 129198 ep_pol tty2      00:02:47 Xorg
0 S  1001     1181      995  0   80   0 - 47145 do_sys tty2      00:00:00 gnome-
sess
0 R  1001    20130    20118  99   80   0 - 624 -      pts/0    00:00:04 a.out
1 R  1001    20131    20130  99   80   0 - 624 -      pts/0    00:00:04 a.out
4 R  1001    20142    20135  0   80   0 - 2944 -      pts/1    00:00:00 ps
student@D001-38:~/2141004081/DOS/Assignment 4$
```

```
1#include <stdio.h>
2#include <stdlib.h>
3#include <unistd.h>
4
5int main() {
6    pid_t pid = fork();
7    if (pid == 0) {
8        printf("Child PID : %d\n", getpid());
9        while (1);
10    } else if (pid > 0) {
11        printf("Parent PID : %d\n", getpid());
12        while (1);
13    }
14    return 0;
15}
```



```
student@D001-38: ~/2141004081/DOS/Assignment 4$ gcc Q1.c && ./a.out
Parent PID : 20195
Child PID : 20196
```

```
student@D001-38:~/2141004081/DOS/Assignment 4$ ps -la
F S  UID      PID     PPID  C  PRI  NI ADDR SZ WCHAN  TTY      TIME CMD
4 S  1001      997      995  1   80   0 - 129782 ep_pol tty2      00:02:49 Xorg
0 S  1001     1181      995  0   80   0 - 47145 do_sys tty2      00:00:00 gnome-
0 R  1001    20195    20118  94   80   0 - 624 -      pts/0    00:00:10 a.out
1 Z  1001    20196    20195  0   80   0 - 0 -      pts/0    00:00:00
4 R  1001    20201    20135  0   80   0 - 2944 -      pts/1    00:00:00 ps
student@D001-38:~/2141004081/DOS/Assignment 4$
```

```
1#include <stdio.h>
2#include <stdlib.h>
3#include <unistd.h>
4
5int main() {
6    pid_t pid = fork();
7    if (pid == 0) {
8        printf("Child PID : %d\n", getpid());
9        exit (0);
10    } else if (pid > 0) {
11        printf("Parent PID : %d\n", getpid());
12        while (1);
13    }
14    return 0;
15}
```

Activities Text Editor Nov 23 10:51

student@D001-38: ~/2141004081/DOS/Assignment 4

```
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q1.c && ./a.out
Parent PID : 20280
Child PID : 20281
student@D001-38:~/2141004081/DOS/Assignment 4$
```

student@D001-38:~/2141004081/DOS/Assignment 4\$ ps -la

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	1001	997	995	1	80	0	-	129782	ep_pol	tty2	00:02:51	Xorg
0	S	1001	1181	995	0	80	0	-	47145	do_sys	tty2	00:00:00	gnome-
1	R	1001	20281	979	99	80	0	-	624	-	pts/0	00:00:03	a.out
4	R	1001	20283	20135	0	80	0	-	2944	-	pts/1	00:00:00	ps

student@D001-38:~/2141004081/DOS/Assignment 4\$

Q1.c

```
1#include <stdio.h>
2#include <stdlib.h>
3#include <unistd.h>
4
5int main() {
6    pid_t pid = fork();
7    if (pid == 0) {
8        printf("Child PID : %d\n", getpid());
9        while (1);
10    } else if (pid > 0) {
11        printf("Parent PID : %d\n", getpid());
12        exit (0);
13    }
14    return 0;
15}
```

Activities Text Editor Nov 23 10:55

student@D001-38: ~/2141004081/DOS/Assignment 4

```
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q1.c && ./a.out
Parent PID : 20352
Child PID : 20353
student@D001-38:~/2141004081/DOS/Assignment 4$
```

student@D001-38:~/2141004081/DOS/Assignment 4\$ ps -la

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	1001	997	995	1	80	0	-	130333	ep_pol	tty2	00:02:52	Xorg
0	S	1001	1181	995	0	80	0	-	47145	do_sys	tty2	00:00:00	gnome-
1	R	1001	20281	979	99	80	0	-	624	-	pts/0	00:03:05	a.out
0	S	1001	20352	20118	0	80	0	-	624	do_wai	pts/0	00:00:00	a.out
1	R	1001	20353	20352	97	80	0	-	624	-	pts/0	00:00:02	a.out
4	R	1001	20355	20135	0	80	0	-	2944	-	pts/1	00:00:00	ps

student@D001-38:~/2141004081/DOS/Assignment 4\$

Q1.c

```
1#include <stdio.h>
2#include <stdlib.h>
3#include <unistd.h>
4#include <sys/wait.h>
5
6int main() {
7    pid_t pid = fork();
8    if (pid == 0) {
9        printf("Child PID : %d\n", getpid());
10        while (1);
11    } else if (pid > 0) {
12        printf("Parent PID : %d\n", getpid());
13        wait (NULL);
14        while (1);
15    }
16    return 0;
17}
```

The screenshot shows a Linux desktop with a text editor window titled 'Q1.c' and a terminal window. The text editor contains the following C code:

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

The terminal window shows the output of the program:

```
student@D001-38: ~/Z141004081/DOS/Assignment 4 $ gcc Q1.c && ./a.out
Parent PID : 20406
Child PID : 20407
```

Below the terminal window, a 'ps -la' command is executed, showing the process list:

```
student@D001-38:~/Z141004081/DOS/Assignment 4$ ps -la
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   1001      997      995  1 80   0 - 129798 ep_pol tty2      00:02:54 Xorg
0 S   1001     1181      995  0 80   0 - 47145  do_sys  tty2      00:00:00 gnome-
1 R   1001    20281     979  99 80   0 - 624 -      pts/0      00:04:07 a.out
0 R   1001    20406    20118  83 80   0 - 624 -      pts/0      00:00:03 a.out
4 R   1001    20409    20135  0 80   0 - 2944 -      pts/1      00:00:00 ps
```

Q2. Trace the output of the following codes:

A.

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

The screenshot shows a Linux desktop with a text editor window titled 'Q2A.c' and a terminal window. The text editor contains the following C code:

```
1#include <stdio.h>
2
3#include <unistd.h>
4
5int main() {
6    if (fork() == 0)
7        printf("1");
8    else
9        printf("2");
10    printf("3");
11    return 0;
12}
```

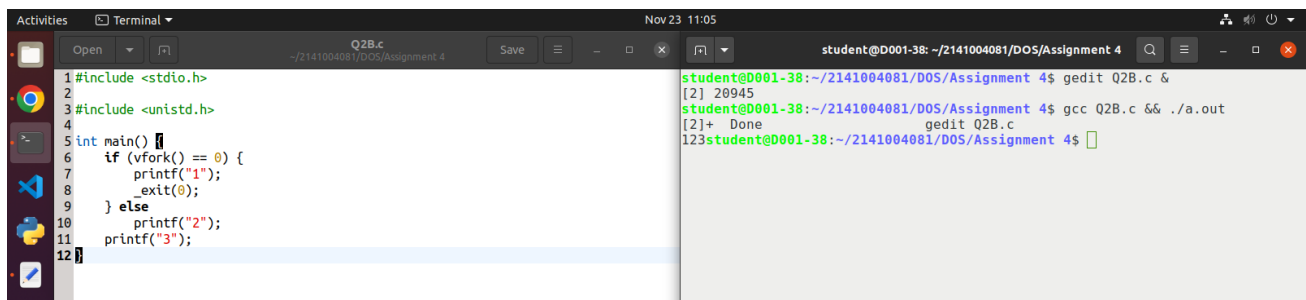
The terminal window shows the output of the program:

```
student@D001-38:~/Z141004081/DOS/Assignment 4$ gedit Q2A.c &
[2] 20893
student@D001-38:~/Z141004081/DOS/Assignment 4$ gcc Q2A.c && ./a.out
[2]+  Done                  gedit Q2A.c
2313student@D001-38:~/Z141004081/DOS/Assignment 4$
```

```

B.  int main( ) {
        if(vfork()==0) {
            printf("1");
            exit(0);
        } else
            printf("2");
            printf("3");
    }

```



```

student@D001-38: ~/Z141004081/DOS/Assignment 4
student@D001-38:~/Z141004081/DOS/Assignment 4$ gedit Q2B.c &
[2] 20945
student@D001-38:~/Z141004081/DOS/Assignment 4$ gcc Q2B.c && ./a.out
[2]+  Done                  gedit Q2B.c
123student@D001-38:~/Z141004081/DOS/Assignment 4$

```

```

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;
    }

```



```

student@D001-38:~/Z141004081/DOS/Assignment 4$ gedit Q2C.c &
[2] 21081
student@D001-38:~/Z141004081/DOS/Assignment 4$ gcc Q2C.c && ./a.out
[2]+  Done                  gedit Q2C.c
Child: 6Parent: 6student@D001-38:~/Z141004081/DOS/Assignment 4$

```

```

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;
    }

```

The screenshot shows a Linux desktop environment with a terminal window and a code editor. The code editor displays the C code for the program, which uses `vfork()` to create a child process. The terminal shows the execution of the program, which prints "Child: 6" and "Parent: 7".

```

1 #include <stdio.h>
2
3 #include <unistd.h>
4
5 #include <sys/wait.h>
6
7 int main()
8 {
9     pid_t pid;
10    int i = 5;
11    pid = vfork();
12    i = i + 1;
13    if (pid == 0)
14    {
15        printf("Child: %d", i);
16        _exit(0);
17    } else
18    {
19        printf("Parent: %d", i);
20    }
21    return 0;
22 }

```

```

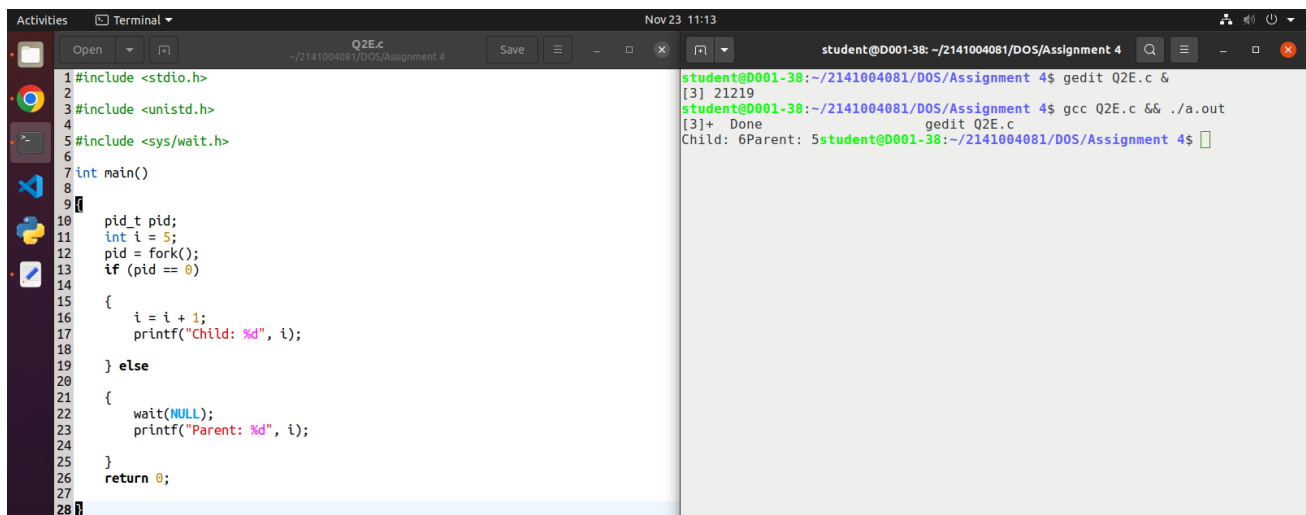
student@D001-38: ~/2141004081/DOS/Assignment 4
student@D001-38:~/2141004081/DOS/Assignment 4$ gedit Q2D.c &
[2] 21152
[1] Done
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2D.c && ./a.out
Child: 6Parent: 7student@D001-38:~/2141004081/DOS/Assignment 4$

```

```

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;
    }

```

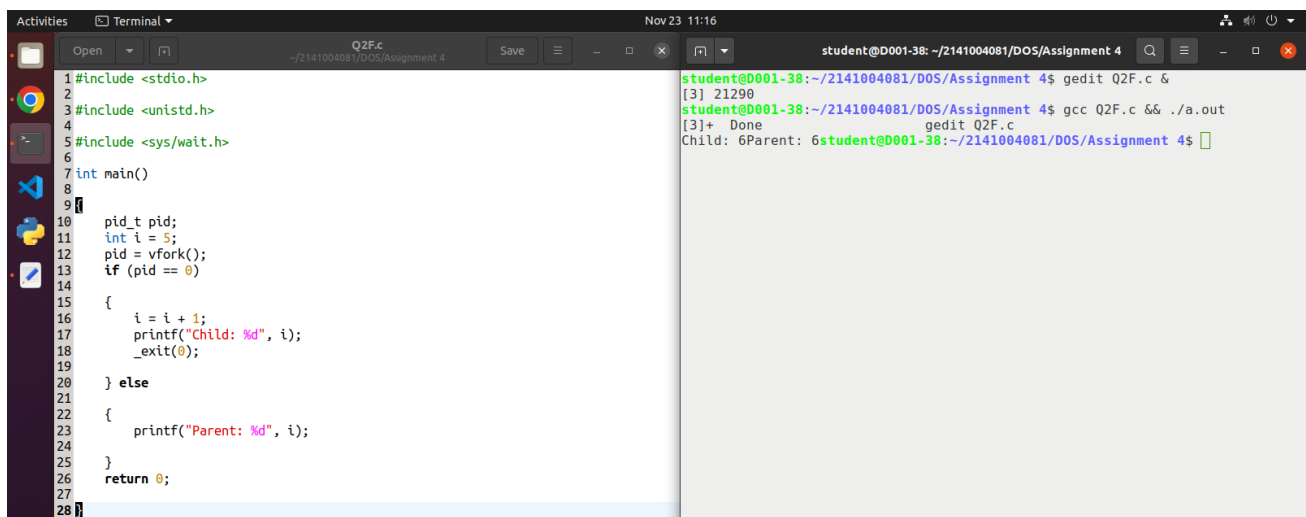


```
1#include <stdio.h>
2
3#include <unistd.h>
4
5#include <sys/wait.h>
6
7int main()
8{
9    pid_t pid;
10    int i = 5;
11    pid = fork();
12    if (pid == 0)
13    {
14        i = i + 1;
15        printf("Child: %d", i);
16    } else
17    {
18        wait(NULL);
19        printf("Parent: %d", i);
20    }
21    return 0;
22}
```

```
student@D001-38: ~/Z141004081/DOS/Assignment 4$ gcc Q2E.c &
[3] 21219
student@D001-38:~/Z141004081/DOS/Assignment 4$ gcc Q2E.c && ./a.out
[3]+  Done                  gcc Q2E.c
Child: 6Parent: 5student@D001-38:~/Z141004081/DOS/Assignment 4$
```

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;
}
```

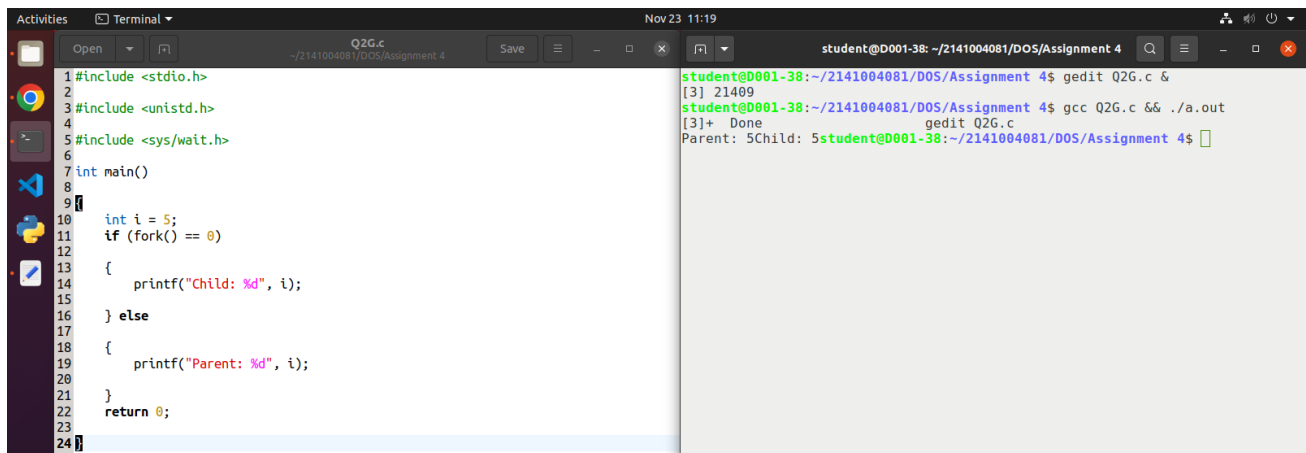


```
1#include <stdio.h>
2
3#include <unistd.h>
4
5#include <sys/wait.h>
6
7int main()
8{
9    pid_t pid;
10    int i = 5;
11    pid = vfork();
12    if (pid == 0)
13    {
14        i = i + 1;
15        printf("Child: %d", i);
16        _exit(0);
17    } else
18    {
19        printf("Parent: %d", i);
20    }
21    return 0;
22}
```

```
student@D001-38:~/Z141004081/DOS/Assignment 4$ gcc Q2F.c &
[3] 21298
student@D001-38:~/Z141004081/DOS/Assignment 4$ gcc Q2F.c && ./a.out
[3]+  Done                  gcc Q2F.c
Child: 6Parent: 6student@D001-38:~/Z141004081/DOS/Assignment 4$
```



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



```

1 #include <stdio.h>
2
3 #include <unistd.h>
4
5 #include <sys/wait.h>
6
7 int main()
8 {
9     int i = 5;
10    if (fork() == 0)
11    {
12        printf("Child: %d", i);
13    } else
14    {
15        printf("Parent: %d", i);
16    }
17    return 0;
18 }
19
20
21
22
23
24

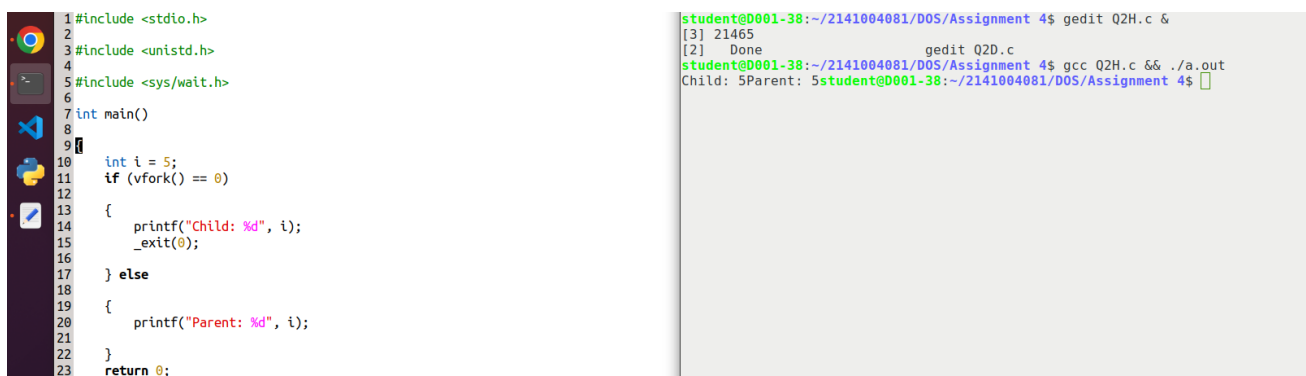
```

```

student@D001-38: ~/2141004081/DOS/Assignment 4
[3] 21409
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2G.c && ./a.out
[3]+  Done                  gedit Q2G.c
Parent: 5Child: 5student@D001-38:~/2141004081/DOS/Assignment 4$

```

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



```

1 #include <stdio.h>
2
3 #include <unistd.h>
4
5 #include <sys/wait.h>
6
7 int main()
8 {
9     int i = 5;
10    if (vfork() == 0)
11    {
12        printf("Child: %d", i);
13        _exit(0);
14    } else
15    {
16        printf("Parent: %d", i);
17    }
18    return 0;
19 }
20
21
22
23
24

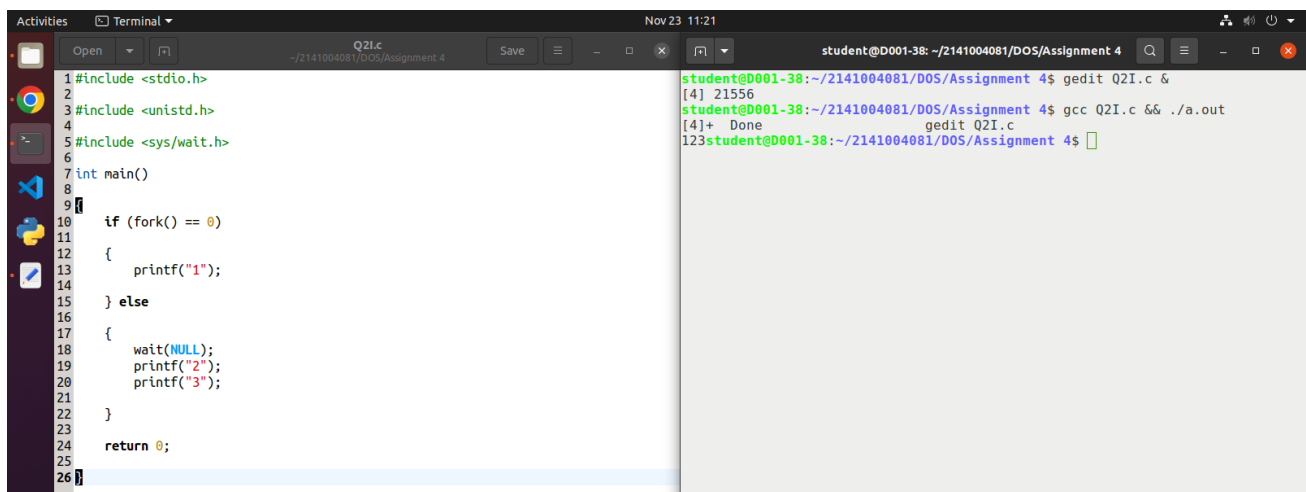
```

```

student@D001-38:~/2141004081/DOS/Assignment 4$ gedit Q2H.c &
[3] 21465
[2] Done                  gedit Q2D.c
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2H.c && ./a.out
Child: 5Parent: 5student@D001-38:~/2141004081/DOS/Assignment 4$

```

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



The screenshot shows a Linux desktop with a dark theme. On the left is a vertical dock with icons for Activities, Files, Firefox, and several application launchers. The main window is a code editor titled 'Q2I.c' with a file path of '~/.2141004081/DOS/Assignment 4'. The code in the editor is the same as in block I. To the right of the code editor is a terminal window titled 'student@D001-38: ~/2141004081/DOS/Assignment 4'. The terminal shows the following commands and output:

```
student@D001-38:~/2141004081/DOS/Assignment 4$ gedit Q2I.c &
[4] 21556
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2I.c && ./a.out
[4]+  Done                  gedit Q2I.c
123student@D001-38:~/2141004081/DOS/Assignment 4$
```

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

```
1#include <stdio.h>
2
3#include <unistd.h>
4
5#include <sys/wait.h>
6
7int main()
8{
9    if (vfork() == 0)
10    {
11        printf("1");
12        _exit(0);
13    } else
14    {
15        printf("2");
16        printf("3");
17    }
18    return 0;
19}
```

```
student@D001-38: ~/2141004081/DOS/Assignment 4$ gedit Q2J.c &
[4] 21616
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2J.c && ./a.out
[4]+  Done                  gedit Q2J.c
123student@D001-38:~/2141004081/DOS/Assignment 4$
```

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;
}
```

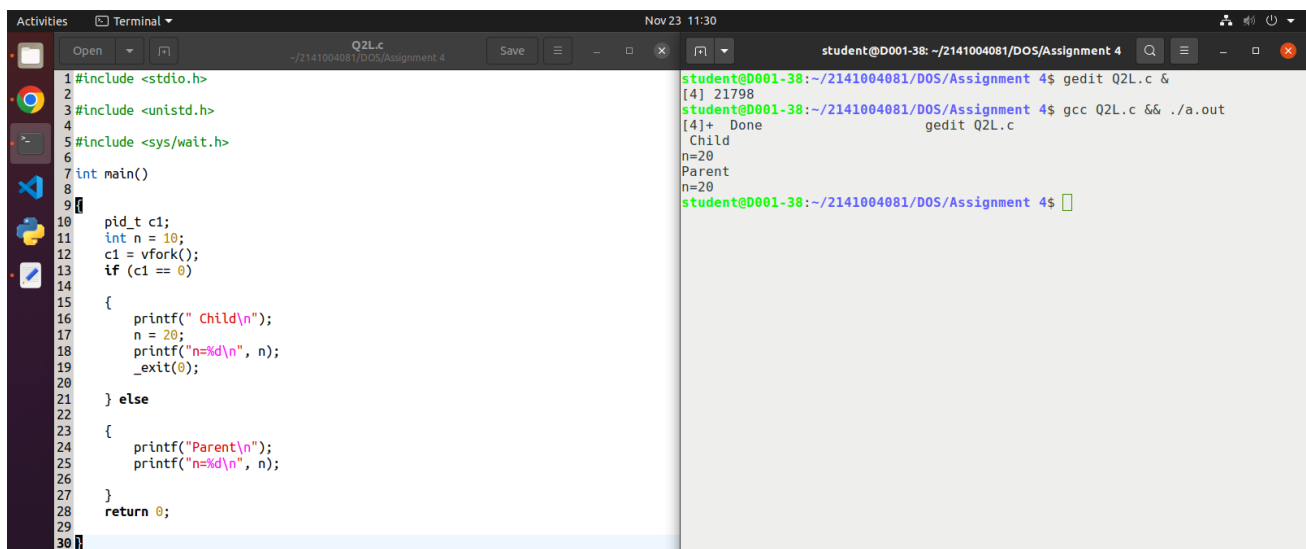
```
1#include <stdio.h>
2
3#include <unistd.h>
4
5#include <sys/wait.h>
6
7int main()
8{
9    pid_t c1;
10    int n = 10;
11    c1 = fork();
12    if (c1 == 0) {
13        printf(" Child\n ");
14        n = 20;
15        printf("n=%d\n ", n);
16    } else {
17        wait(NULL);
18        printf("Parent\n ");
19        printf("n=%d\n ", n);
20    }
21    return 0;
22}
```

```
student@D001-38:~/2141004081/DOS/Assignment 4$ gedit Q2K.c &
[4] 21731
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2K.c && ./a.out
[4]+  Done                  gedit Q2K.c
Child
n=20
Parent
n=10
student@D001-38:~/2141004081/DOS/Assignment 4$
```

```

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;
}

```



The screenshot shows a terminal window with two panes. The left pane displays the source code of a C program named Q2L.c. The code uses vfork() to create a child process. The parent process prints "Parent\n" and the child process prints " Child\n". Both processes then print their value of n. The right pane shows the output of the program, which is " Child\n", "n=20\n", "Parent\n", and "n=20\n".

```

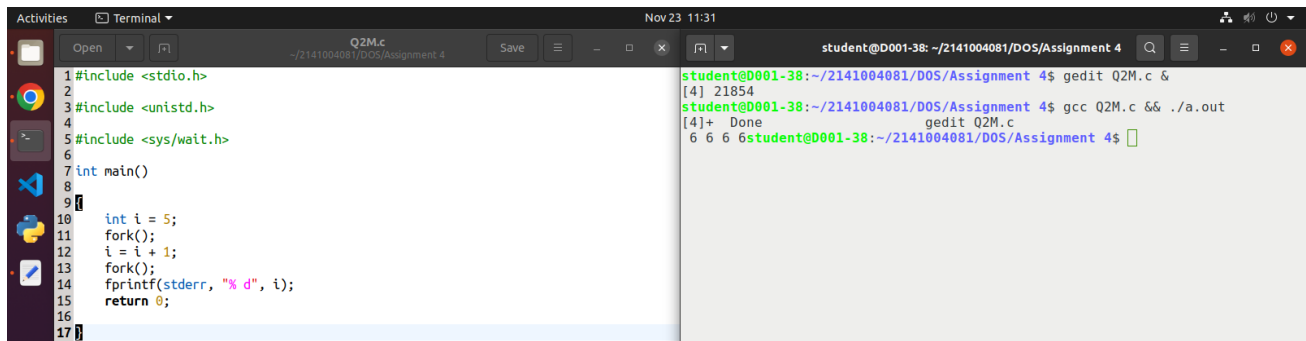
student@D001-38: ~/2141004081/DOS/Assignment 4
[4] 21798
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2L.c && ./a.out
[4]+  Done                  gedit Q2L.c
 Child
n=20
Parent
n=20
student@D001-38:~/2141004081/DOS/Assignment 4$

```

```

M.  int main() {
    int i = 5;
    fork();
    i = i + 1;
    fork();
    fprintf(stderr, "% d", i);
    return 0;
}

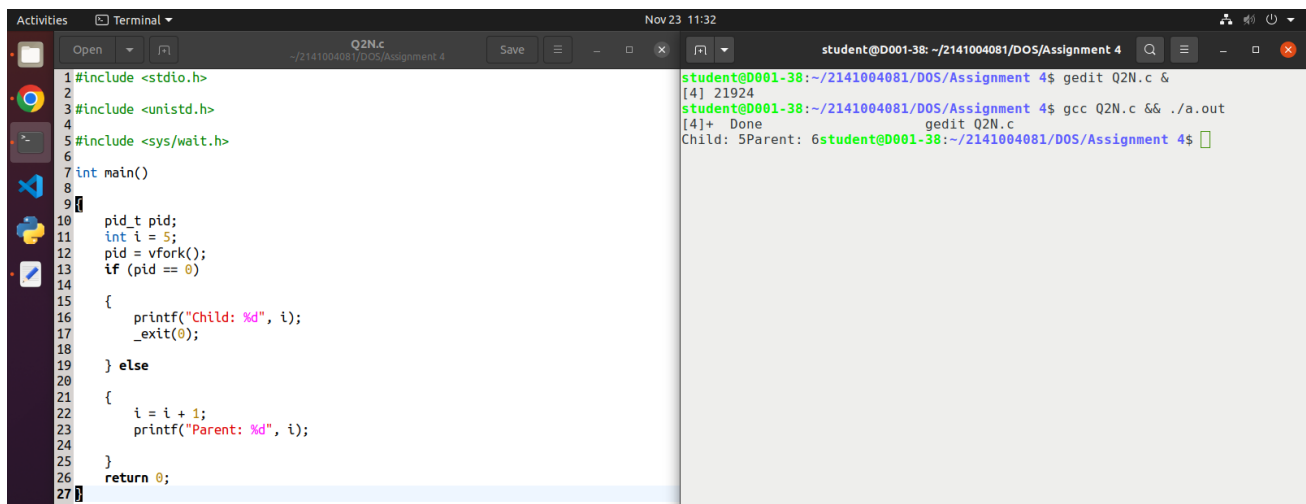
```



```
1 #include <stdio.h>
2
3 #include <unistd.h>
4
5 #include <sys/wait.h>
6
7 int main()
8 {
9     int i = 5;
10    fork();
11    i = i + 1;
12    fork();
13    fprintf(stderr, "%d", i);
14    return 0;
15 }
16
17
```

```
student@D001-38: ~/2141004081/DOS/Assignment 4$ gedit Q2M.c &
[4] 21854
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2M.c && ./a.out
[4]+  Done                  gedit Q2M.c
6 6 6 student@D001-38:~/2141004081/DOS/Assignment 4$
```

```
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;
}
```



```
1 #include <stdio.h>
2
3 #include <unistd.h>
4
5 #include <sys/wait.h>
6
7 int main()
8 {
9     pid_t pid;
10    int i = 5;
11    pid = vfork();
12    if (pid == 0)
13    {
14        printf("Child: %d", i);
15        _exit(0);
16    }
17    else
18    {
19        i = i + 1;
20        printf("Parent: %d", i);
21    }
22    return 0;
23 }
24
25
26
27
```

```
student@D001-38:~/2141004081/DOS/Assignment 4$ gedit Q2N.c &
[4] 21924
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2N.c && ./a.out
[4]+  Done                  gedit Q2N.c
Child: 5Parent: 6 student@D001-38:~/2141004081/DOS/Assignment 4$
```

```

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

```

```

student@D001-38: ~/2141004081/DOS/Assignment 4
student@D001-38:~/2141004081/DOS/Assignment 4$ gedit Q20.c &
[4] 21982
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q20.c && ./a.out
[4]+  Done                  gedit Q20.c
46student@D001-38:~/2141004081/DOS/Assignment 4$

```

```

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

```

```

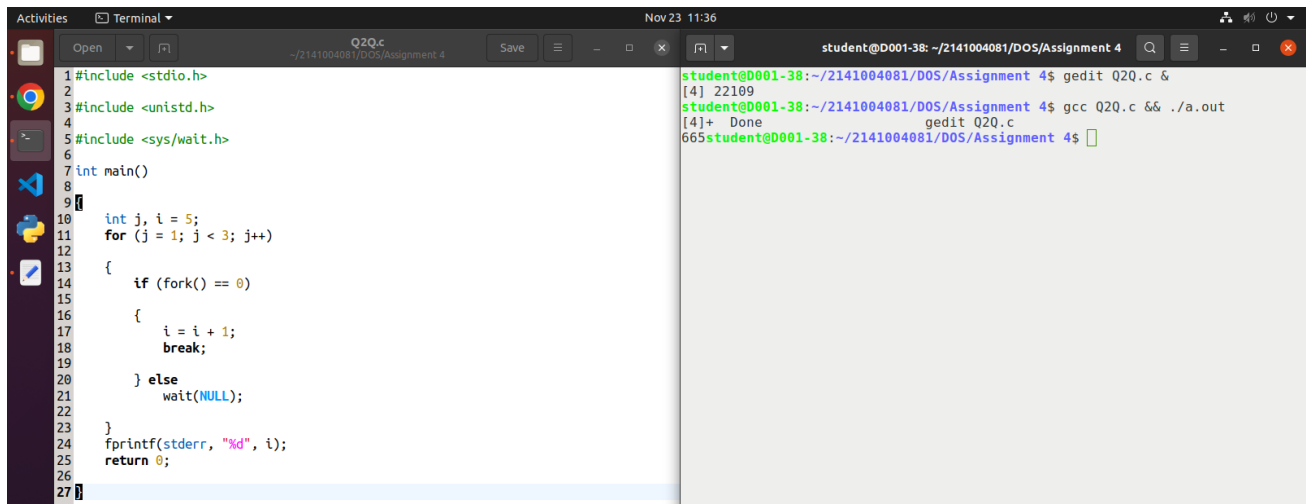
student@D001-38: ~/2141004081/DOS/Assignment 4
student@D001-38:~/2141004081/DOS/Assignment 4$ gedit Q2P.c &
[4] 22040
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2P.c && ./a.out
[4]+  Done                  gedit Q2P.c
5student@D001-38:~/2141004081/DOS/Assignment 4$

```

```

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

```



```

student@D001-38: ~/2141004081/DOS/Assignment 4
student@D001-38:~/2141004081/DOS/Assignment 4$ gedit Q20.c &
[4] 22109
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q20.c && ./a.out
[4]+  Done                  gedit Q20.c
665student@D001-38:~/2141004081/DOS/Assignment 4$

```

```

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;
}

```

```
1#include <stdio.h>
2
3#include <unistd.h>
4
5#include <sys/wait.h>
6
7int main()
8{
9    int j, i = 5;
10   for (j = 1; j < 3; j++)
11   {
12       if (fork() != 0)
13       {
14           i = i - 1;
15           break;
16       }
17   }
18   printf(stderr, "%d", i);
19   return 0;
20 }
```

```
student@D001-38: ~/2141004081/DOS/Assignment 4$ gedit Q2R.c &
[4] 22170
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2R.c && ./a.out
[4]+  Done                  gedit Q2R.c
445student@D001-38:~/2141004081/DOS/Assignment 4$
```

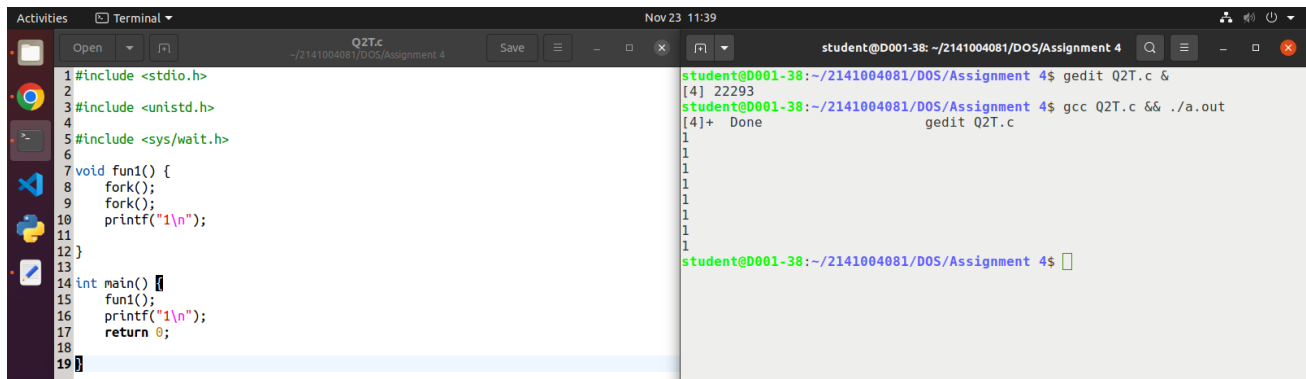
```
S.  int main() {
        if (fork() == 0)
            if (fork())
                printf("1\n");
        return 0;
    }
```

```
1#include <stdio.h>
2
3#include <unistd.h>
4
5#include <sys/wait.h>
6
7int main()
8{
9    if (fork() == 0)
10    {
11        if (fork())
12            printf("1\n");
13    }
14    return 0;
15 }
```

```
student@D001-38:~/2141004081/DOS/Assignment 4$ gedit Q2S.c &
[4] 22233
student@D001-38:~/2141004081/DOS/Assignment 4$ gcc Q2S.c && ./a.out
[4]+  Done                  gedit Q2S.c
1
student@D001-38:~/2141004081/DOS/Assignment 4$
```

```
T.  void fun1() {
        fork();
        fork();
        printf("1\n");
    }

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

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.

Command:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
void childProcess1(char *file1, char *file2) {
    printf("First child process created. PID: %d,
Parent PID: %d\n", getpid(), getppid());
    execlp("cp", "cp", file1, file2, NULL);
    perror("exec");
    exit(EXIT_FAILURE);
}
void childProcess2(char *file2) {
    printf("Second child process created. PID: %d,
Parent PID: %d\n", getpid(), getppid());
```

```
        execlp("cat", "cat", file2, NULL);
        perror("exec");
        exit(EXIT_FAILURE);
    }
    void childProcess3(char *file2) {
        printf("Third child process created. PID: %d,
Parent PID: %d\n", getpid(), getppid());
        execlp("sort", "sort", "-r", file2, NULL);
        perror("exec");
        exit(EXIT_FAILURE);
    }
    int main() {
        char *file1 = "input.txt";
        char *file2 = "output.txt";
        pid_t child1 = fork();
        if (child1 == 0) {
            childProcess1(file1, file2);
        } else if (child1 == -1) {
            perror("fork");
            exit(EXIT_FAILURE);
        } else {
            sleep(1);
            pid_t child2 = fork();
            if (child2 == 0) {
                childProcess2(file2);
            } else if (child2 == -1) {
                perror("fork");
                exit(EXIT_FAILURE);
            } else {
                sleep(1);
                pid_t child3 = fork();
                if (child3 == 0) {
                    childProcess3(file2);
                } else if (child3 == -1) {
                    perror("fork");
                    exit(EXIT_FAILURE);
                }
            }
        }
    }
}
```

```

    } else {
        sleep(1);
        wait(NULL);
        wait(NULL);
        wait(NULL);
        printf("Parent process completed. PID:
%d\n", getpid());
    }
}
}
return 0;
}

```

Output:

```

arya@arya:~/2141004081/DOS/Lab4$ gedit L4Q3.c &
[2] 5615
arya@arya:~/2141004081/DOS/Lab4$ cat > input.txt
Hello, I am inside file named input.txt
^C
[2]+  Done                  gedit L4Q3.c
arya@arya:~/2141004081/DOS/Lab4$ cat > output.txt
^C
arya@arya:~/2141004081/DOS/Lab4$ gcc L4Q3.c && ./a.out
First child process created. PID: 5635, Parent PID: 5634
Second child process created. PID: 5636, Parent PID: 5634
Hello, I am inside file named input.txt
Third child process created. PID: 5639, Parent PID: 5634
Hello, I am inside file named input.txt
Parent process completed. PID: 5634
arya@arya:~/2141004081/DOS/Lab4$

```

Q3. 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.

Command:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int isPrime(int num) {
    if (num <= 1) return 0;
    for (int i = 2; i * i <= num; i++)
        if (num % i == 0) return 0;
    return 1;
}
void generateFibonacci(int length, int *fibonacci) {
    fibonacci[0] = 0;
    fibonacci[1] = 1;
    for (int i = 2; i < length; i++)
        fibonacci[i] = fibonacci[i - 1] + fibonacci[i
- 2];
}
int main() {
    int length;
    printf("Enter length of Fibonacci series: ");
    scanf("%d", &length);
    pid_t child = fork();
    if (child == 0) {
        int *fibonacci = (int *)malloc(length *
sizeof(int));
        generateFibonacci(length, fibonacci);
        printf("Fibonacci series: ");
```

```

        for (int i = 0; i < length; i++) printf("%d ",
fibonacci[i]);
        printf("\n");
        printf("Prime Fibonacci number: ");
        for (int i = 2; i < length; i++)
            if (isPrime(fibonacci[i])) {
                printf("%d (at position %d)\n",
fibonacci[i], i + 1);
                break;
            }
        free(fibonacci);
    } else if (child == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    } else {
        wait(NULL);
        printf("Parent process completed.\n");
    }
    return 0;
}

```

Output:

The screenshot shows a terminal window with two panes. The left pane displays the source code of a C program named `L4Q4.c`. The code includes headers for `stdio.h`, `stdlib.h`, `unistd.h`, `sys/types.h`, and `sys/wait.h`. It defines a function `isPrime` to check if a number is prime, a function `generateFibonacci` to generate a Fibonacci series, and a `main` function that prompts the user for the length of the series, forks a child process to generate and print the series, and then waits for the child to complete before printing the prime Fibonacci number.

The right pane shows the terminal output of the program. The user enters the length of the series as 15, and the program outputs the Fibonacci series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377. It then identifies the prime Fibonacci number as 2 at position 4 and prints "Parent process completed."

```

arya@arya:~/2141004081/DOS/Lab4$ gedit L4Q4.c &
[2] 5752
arya@arya:~/2141004081/DOS/Lab4$ gcc L4Q4.c && ./a.out
[2]+  Done                  gedit L4Q4.c
Enter length of Fibonacci series: 15
Fibonacci series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
Prime Fibonacci number: 2 (at position 4)
Parent process completed.
arya@arya:~/2141004081/DOS/Lab4$

```