

**NAME : ADITYA RAJ PANDIT**

**REG NO : 23BRS1157**

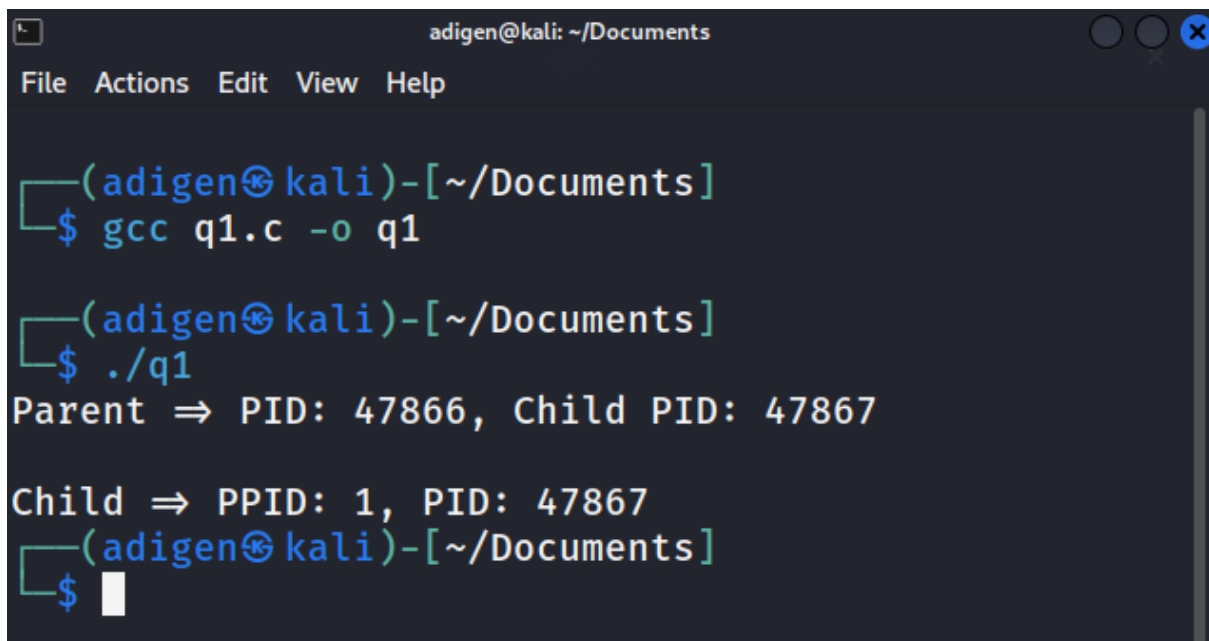
**SYSTEM CALLS**

## **ASSIGNMENT :**

**1) Create a process and print parent ID and Child ID**



```
1#include <stdio.h>
2#include <sys/types.h>
3#include <unistd.h>
4
5int main() {
6    pid_t pid = fork(); // Create a new process
7
8    if (pid == 0) {
9        // Child process
10       printf("Child => PPID: %d, PID: %d\n", getppid(), getpid());
11    } else if (pid > 0) {
12        // Parent process
13        printf("Parent => PID: %d, Child PID: %d\n", getpid(), pid);
14    } else {
15        // Fork failed
16        printf("Fork failed!\n");
17        return 1;
18    }
19
20    return 0;
21}
22
```



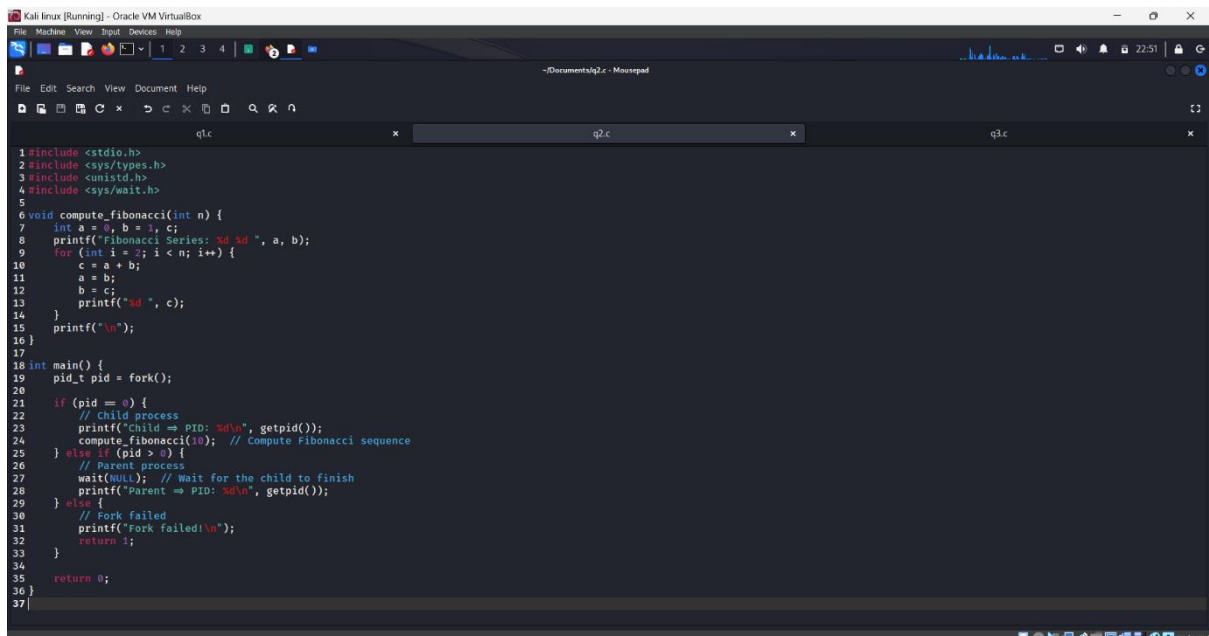
```
adigen@kali: ~/Documents
File Actions Edit View Help

(adigen@kali)-[~/Documents]
$ gcc q1.c -o q1

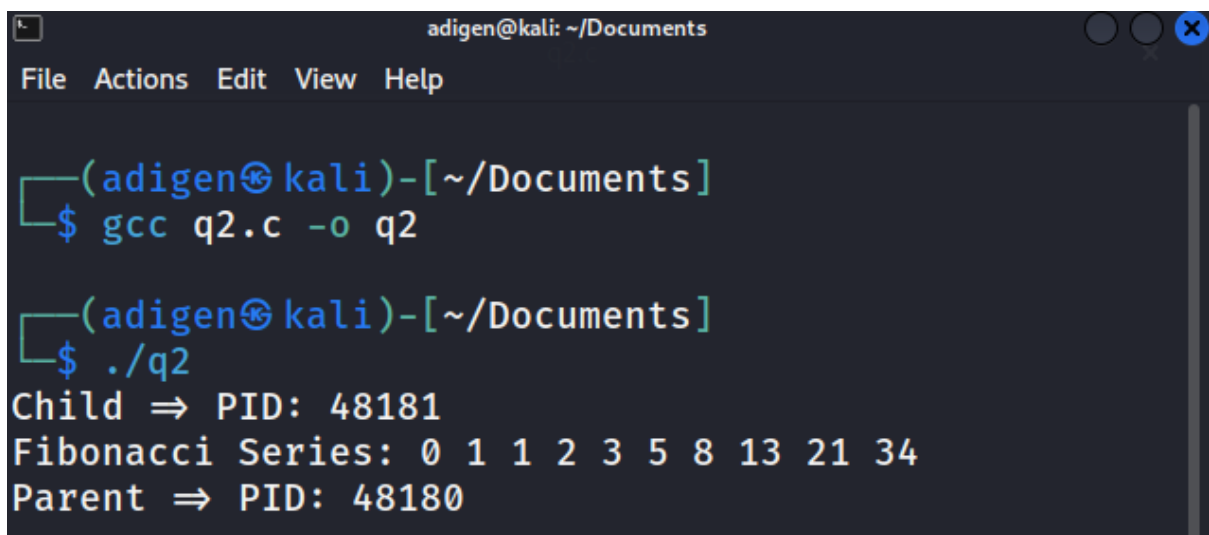
(adigen@kali)-[~/Documents]
$ ./q1
Parent => PID: 47866, Child PID: 47867

Child => PPID: 1, PID: 47867
(adigen@kali)-[~/Documents]
$
```

## 2) Create a Process and let the child do some task like computing Fibonacci



```
1#include <stdio.h>
2#include <sys/types.h>
3#include <unistd.h>
4#include <sys/wait.h>
5
6void compute_fibonacci(int n) {
7    int a = 0, b = 1, c;
8    printf("Fibonacci Series: %d %d ", a, b);
9    for (int i = 2; i < n; i++) {
10        c = a + b;
11        a = b;
12        b = c;
13        printf("%d ", c);
14    }
15    printf("\n");
16}
17
18int main() {
19    pid_t pid = fork();
20
21    if (pid == 0) {
22        // Child process
23        printf("Child => PID: %d\n", getpid());
24        compute_fibonacci(10); // Compute Fibonacci sequence
25    } else if (pid > 0) {
26        // Parent process
27        wait(NULL); // Wait for the child to finish
28        printf("Parent => PID: %d\n", getpid());
29    } else {
30        // Fork failed
31        printf("fork failed!\n");
32        return 1;
33    }
34
35    return 0;
36}
37
```



```
adigen@kali: ~/Documents
File Actions Edit View Help

(adigen@kali)-[~/Documents]
$ gcc q2.c -o q2

(adigen@kali)-[~/Documents]
$ ./q2
Child => PID: 48181
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
Parent => PID: 48180
```

### 3) Create a process and compute factorial in child and Fibonacci in parent as executable

```
Kali Linux [Running] - Oracle VM VirtualBox
File Media View Spool Devices Help
--Documents/q3.c - Mousepad
File Edit Search View Document Help
q1.c x q2.c x q3.c x

1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5
6 int compute_factorial(int n) {
7     if (n == 0)
8         return 1;
9     return n * compute_factorial(n - 1);
10 }
11
12 void compute_fibonacci(int n) {
13     int a = 0, b = 1, c;
14     printf("Fibonacci Series: %d %d ", a, b);
15     for (int i = 2; i < n; i++) {
16         c = a + b;
17         a = b;
18         b = c;
19         printf("%d ", c);
20     }
21     printf("\n");
22 }
23
24 int main() {
25     pid_t pid = fork();
26
27     if (pid == 0) {
28         // Child process: Compute factorial
29         int num = 5;
30         printf("Child => PID: %d\n", getpid());
31         printf("Factorial of %d is: %d\n", num, compute_factorial(num));
32     } else if (pid > 0) {
33         // Parent process: Compute Fibonacci
34         wait(NULL); // Wait for the child to finish
35         printf("Parent => PID: %d\n", getpid());
36         compute_fibonacci(10); // Compute Fibonacci sequence
37     } else {
38         // Fork failed
39         printf("fork failed!\n");
40         return 1;
41     }
42 }
```

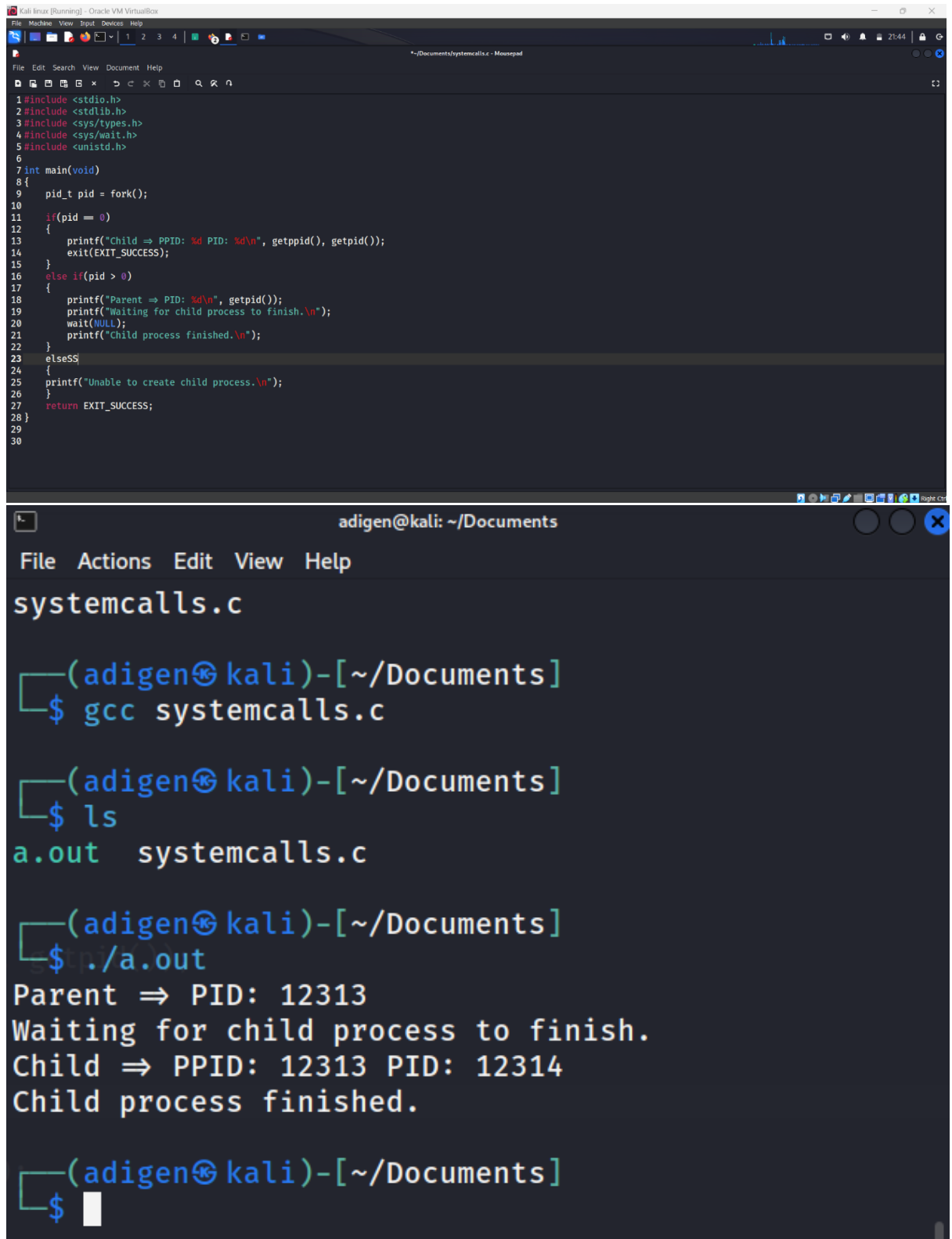
```
adigen@kali: ~/Documents
File Actions Edit View Help

(adigen@kali)-[~/Documents]
$ gcc q3.c -o q3

(adigen@kali)-[~/Documents]
$ ./q3
Child => PID: 48543
Factorial of 5 is: 120
Parent => PID: 48542
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

## CLASS WORK :

- 1) Write programs using the following system calls of UNIX operating system: fork, getpid, getppid, exit



The image shows a Kali Linux virtual machine environment. The top window is a text editor (likely VS Code) showing the source code for a C program named `systemcalls.c`. The code uses `fork()` to create a child process, `getpid()` and `getppid()` to retrieve process IDs, and `exit()` to terminate the child process. The bottom window is a terminal showing the compilation and execution of the program.

```
1#include <stdio.h>
2#include <stdlib.h>
3#include <sys/types.h>
4#include <sys/wait.h>
5#include <unistd.h>
6
7int main(void)
8{
9    pid_t pid = fork();
10
11    if(pid == 0)
12    {
13        printf("Child => PPID: %d PID: %d\n", getppid(), getpid());
14        exit(EXIT_SUCCESS);
15    }
16    else if(pid > 0)
17    {
18        printf("Parent => PID: %d\n", getpid());
19        printf("Waiting for child process to finish.\n");
20        wait(NULL);
21        printf("Child process finished.\n");
22    }
23    else
24    {
25        printf("Unable to create child process.\n");
26    }
27    return EXIT_SUCCESS;
28 }
29
30
```

Terminal output:

```
adigen@kali: ~/Documents
File Actions Edit View Help
systemcalls.c

(adigen@kali)-[~/Documents]
$ gcc systemcalls.c

(adigen@kali)-[~/Documents]
$ ls
a.out  systemcalls.c

(adigen@kali)-[~/Documents]
$ ./a.out
Parent => PID: 12313
Waiting for child process to finish.
Child => PPID: 12313 PID: 12314
Child process finished.

(adigen@kali)-[~/Documents]
$
```

- 2) Program for wait () system call which makes the parent process wait for the child to finish.

```
Kali Linux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
~ /Documents/wait_syscall.c - Mousepad
File Edit Search View Document Help
1 #include<unistd.h>
2 #include<sys/types.h>
3 #include<stdio.h>
4 #include<sys/wait.h>
5 int main()
6 {
7     pid_t p;
8     printf("before fork\n");
9     p=fork();
10    if(p==0)//child
11    {
12        printf("I am child having id %d\n",getpid());
13        printf("My parent's id is %d\n",getppid());
14    }
15    else//parent
16    {
17        wait(NULL);
18        printf("My child's id is %d\n",p);
19        printf("I am parent having id %d\n",getpid());
20    }
21    printf("Common\n");
22 }
23
```

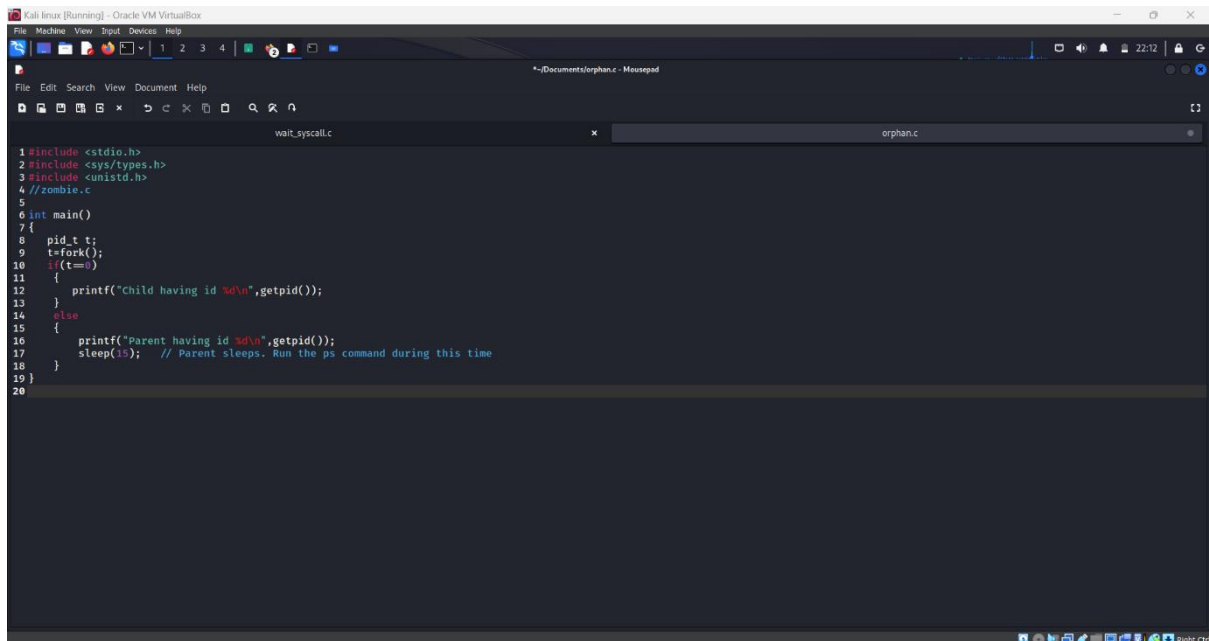
```
adigen@kali: ~/Documents
File Actions Edit View Help
└─$ gcc wait_syscall.c -o wait

(adigen@kali)-[~/Documents]
└─$ ls
a.out  systemcalls.c  wait  wait_syscall.c

(adigen@kali)-[~/Documents]
└─$ ./wait
before fork
I am child having id 17889
My parent's id is 17888
Common
My child's id is 17889
I am parent having id 17888
Common

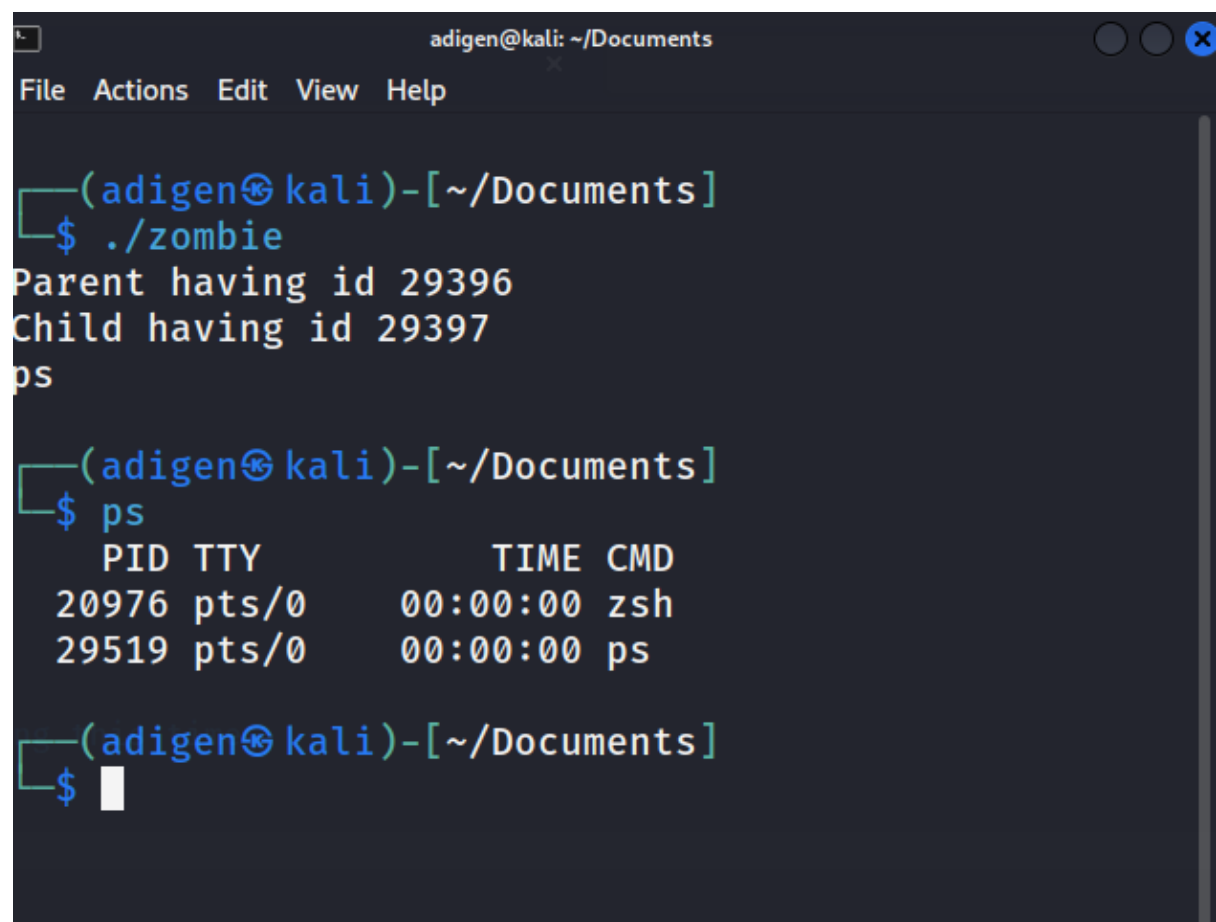
(adigen@kali)-[~/Documents]
└─$
```

### 3) Demonstrates the creation and termination of a zombie process



The screenshot shows a code editor window titled "orphan.c" with the following C code:

```
1#include <stdio.h>
2#include <sys/types.h>
3#include <unistd.h>
4//zombie.c
5
6int main()
7{
8    pid_t t;
9    t=fork();
10   if(t==0)
11   {
12       printf("Child having id %d\n",getpid());
13   }
14   else
15   {
16       printf("Parent having id %d\n",getpid());
17       sleep(15); // Parent sleeps. Run the ps command during this time
18   }
19 }
20
```



The screenshot shows a terminal window with the following output:

```
adigen@kali: ~/Documents
File Actions Edit View Help

(adigen@kali)-[~/Documents]
$ ./zombie
Parent having id 29396
Child having id 29397
ps

(adigen@kali)-[~/Documents]
$ ps
  PID TTY          TIME CMD
 20976 pts/0        00:00:00 zsh
 29519 pts/0        00:00:00 ps

(adigen@kali)-[~/Documents]
$
```

#### 4) Demonstrate the creation of an orphan process

```
Kali Linux [Running] - Oracle VM VirtualBox
File  Media  View  Spool  Devices  Help
~Documents/orphan1.c - Mousepad

1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <unistd.h>
4
5 int main()
6 {
7     pid_t p;
8     p=fork();
9     if(p==0)
10     {
11         sleep(5); //child goes to sleep and in the meantime parent terminates
12         printf("I am child having PID %d\n",getpid());
13         printf("My parent PID is %d\n",getppid());
14     }
15     else
16     {
17         printf("I am parent having PID %d\n",getpid());
18         printf("My child PID is %d\n",p);
19     }
20     return 0;
21 }
22
```

```
adigen@kali: ~/Documents
File  Actions  Edit  View  Help

(adigen@kali)-[~/Documents]
$ gcc orphan1.c -o orph

(adigen@kali)-[~/Documents]
$ ./orph
I am parent having PID 33658
My child PID is 33659

(adigen@kali)-[~/Documents]
$ I am child having PID 33659
My parent PID is 1

();;
```

