

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 41

OS Practical 4

1. Create chain of processes where one parent have exactly one child as per sample chain given by Instructor.

Code:

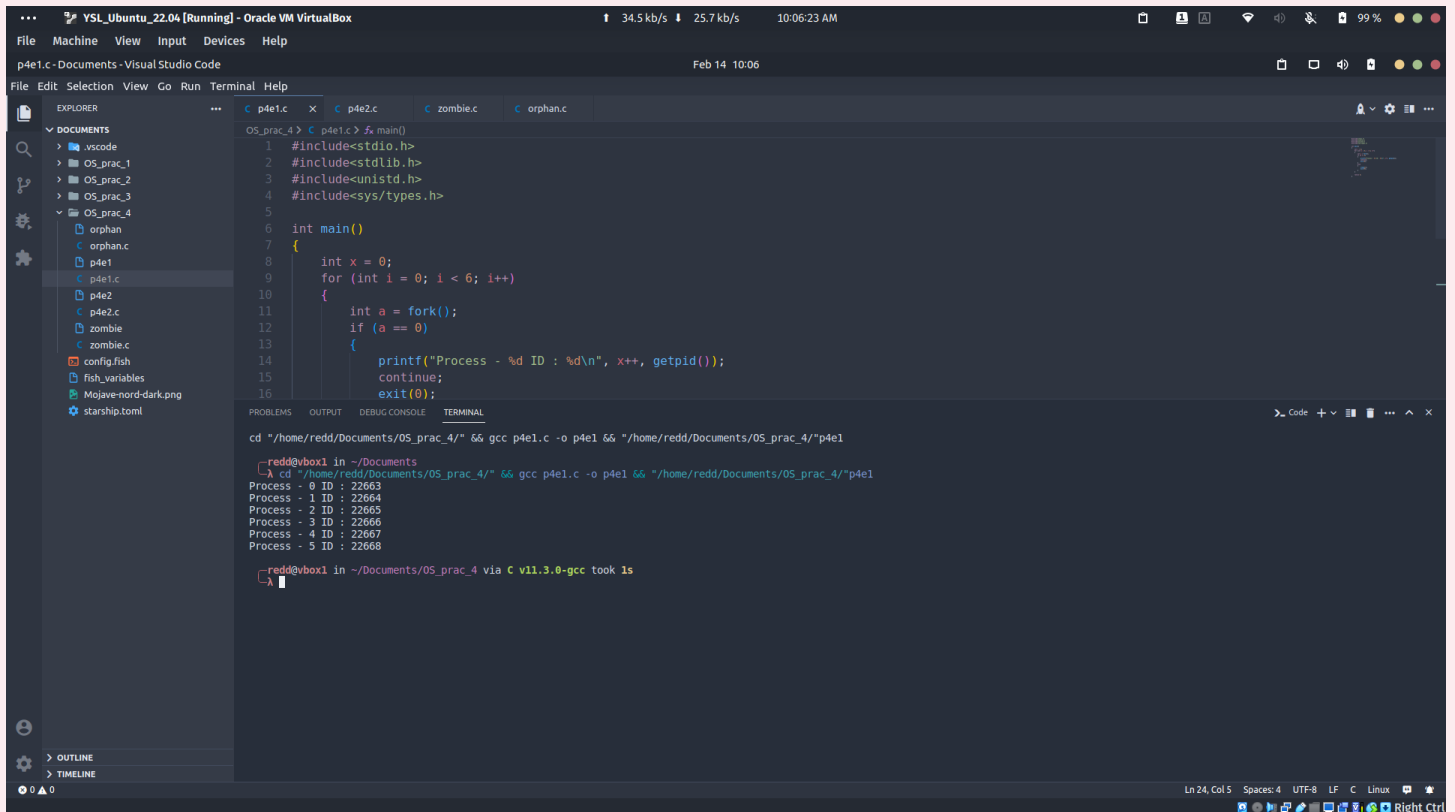
```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>

int main()
{
    int x = 0;
    for (int i = 0; i < 6; i++)
    {
        int a = fork();
        if (a == 0)
        {
            printf("Process - %d ID : %d\n", x++, getpid());
            continue;
            exit(0);
        }
        else
        {
            sleep(1);
            exit(0);
        }
    }

    return 0;
}
```

Output:

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 41
OS Practical 4



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

int main()
{
    int x = 0;
    for (int i = 0; i < 6; i++)
    {
        int a = fork();
        if (a == 0)
        {
            printf("Process - %d ID : %d\n", x++, getpid());
            continue;
            exit(0);
        }
    }
}
```

```
cd "/home/redd/Documents/OS_prac_4/" && gcc p4e1.c -o p4e1 && "/home/redd/Documents/OS_prac_4/"p4e1
redd@vbox1 in ~/Documents
└─A cd "/home/redd/Documents/OS_prac_4/" && gcc p4e1.c -o p4e1 && "/home/redd/Documents/OS_prac_4/"p4e1
Process - 0 ID : 22663
Process - 1 ID : 22664
Process - 2 ID : 22665
Process - 3 ID : 22666
Process - 4 ID : 22667
Process - 5 ID : 22668
redd@vbox1 in ~/Documents/OS_prac_4 via C v11.3.0-gcc took 1s
└─A
```

2. Jack want to sort out coins of various values as per below input. Design program to sort the coins in parent process and print the reversely sorted coins in child process

Code :

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

int main()
{
    int x = 0, i, j;
    int num[6];
```

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 41

OS Practical 4

```
for (int y = 0; y < 6; y++)
{
    scanf("%d", &num[y]);
}

for (int i = 0; i < 6; i++)
{
    int a = fork();
    if (a == 0)
    {
        for (i = 0; i < 6; ++i)
        {
            for (j = i + 1; j < 6; ++j)
            {
                if (num[i] > num[j])
                {
                    a = num[i];
                    num[i] = num[j];
                    num[j] = a;
                }
            }
        }
        for (int y = 0; y < 6; y++)
        {
            printf("%d ", num[y]);
        }
    }
    else
    {
        sleep(1);
        exit(0);
        for (i = 0; i < 6; ++i)
        {
            for (j = i + 1; j < 6; ++j)
```

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 41

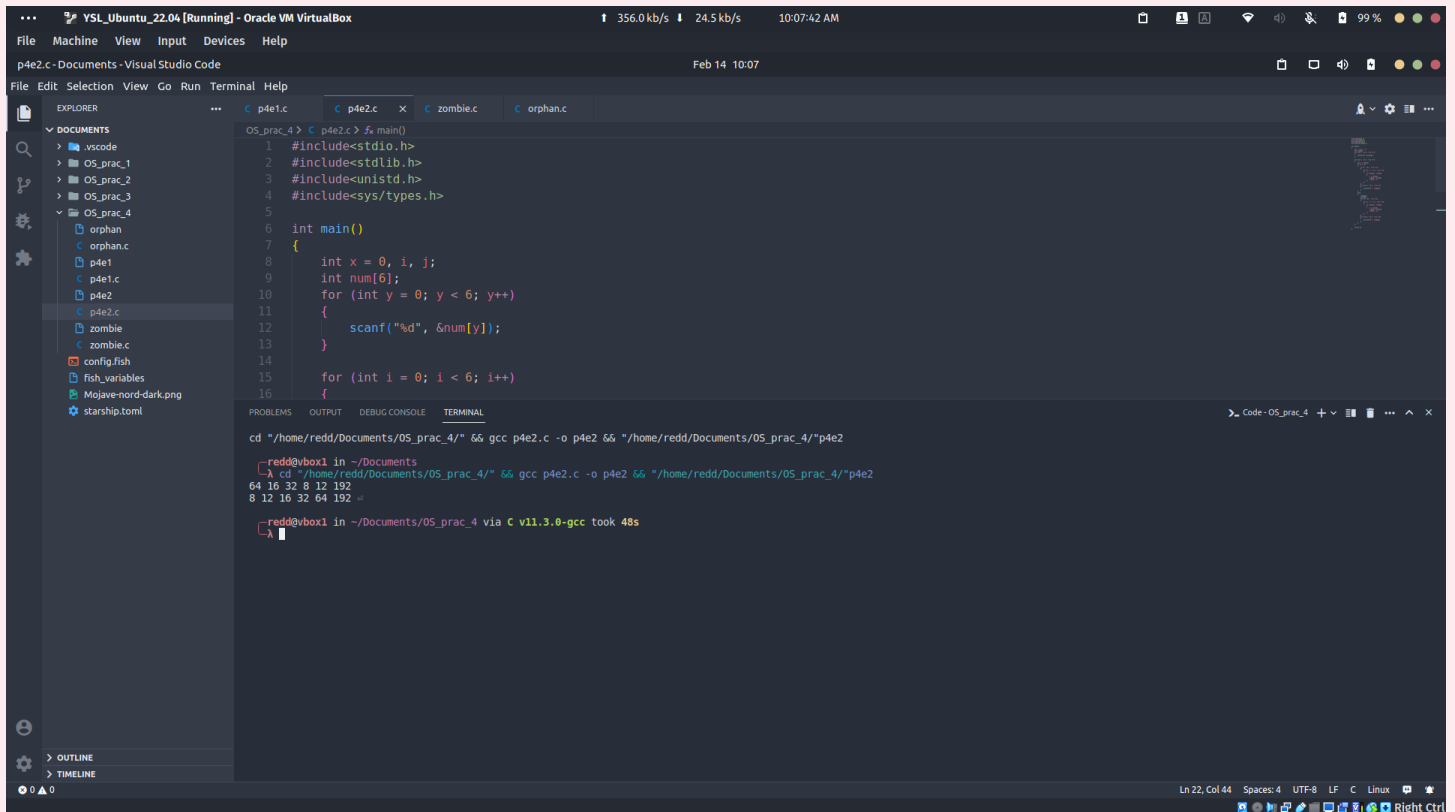
OS Practical 4

```
        {
            if (num[i] > num[j])
            {
                a = num[i];
                num[i] = num[j];
                num[j] = a;
            }
        }
    }
    for (int y = 0; y < 6; y++)
    {
        printf("%d ", num[y]);
    }
}

return 0;
}
```

Output :

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 41
OS Practical 4



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

int main()
{
    int x = 0, i, j;
    int num[6];
    for (int y = 0; y < 6; y++)
    {
        scanf("%d", &num[y]);
    }
    for (int i = 0; i < 6; i++)
    {
```

```
cd ~/home/redd/Documents/OS_prac_4/" 66 gcc p4e2.c -o p4e2 66 ~/home/redd/Documents/OS_prac_4/"p4e2
└─redd@vbox1 in ~/Documents
└─┤ cd ~/home/redd/Documents/OS_prac_4/" 66 gcc p4e2.c -o p4e2 66 ~/home/redd/Documents/OS_prac_4/"p4e2
64 16 32 8 12 192
8 12 16 32 64 192
└─redd@vbox1 in ~/Documents/OS_prac_4 via C v11.3.0-gcc took 48s
└─┤
```

3. Demonstrate Zombi and Orphan state of processes using fork().

Code (Zombie) :

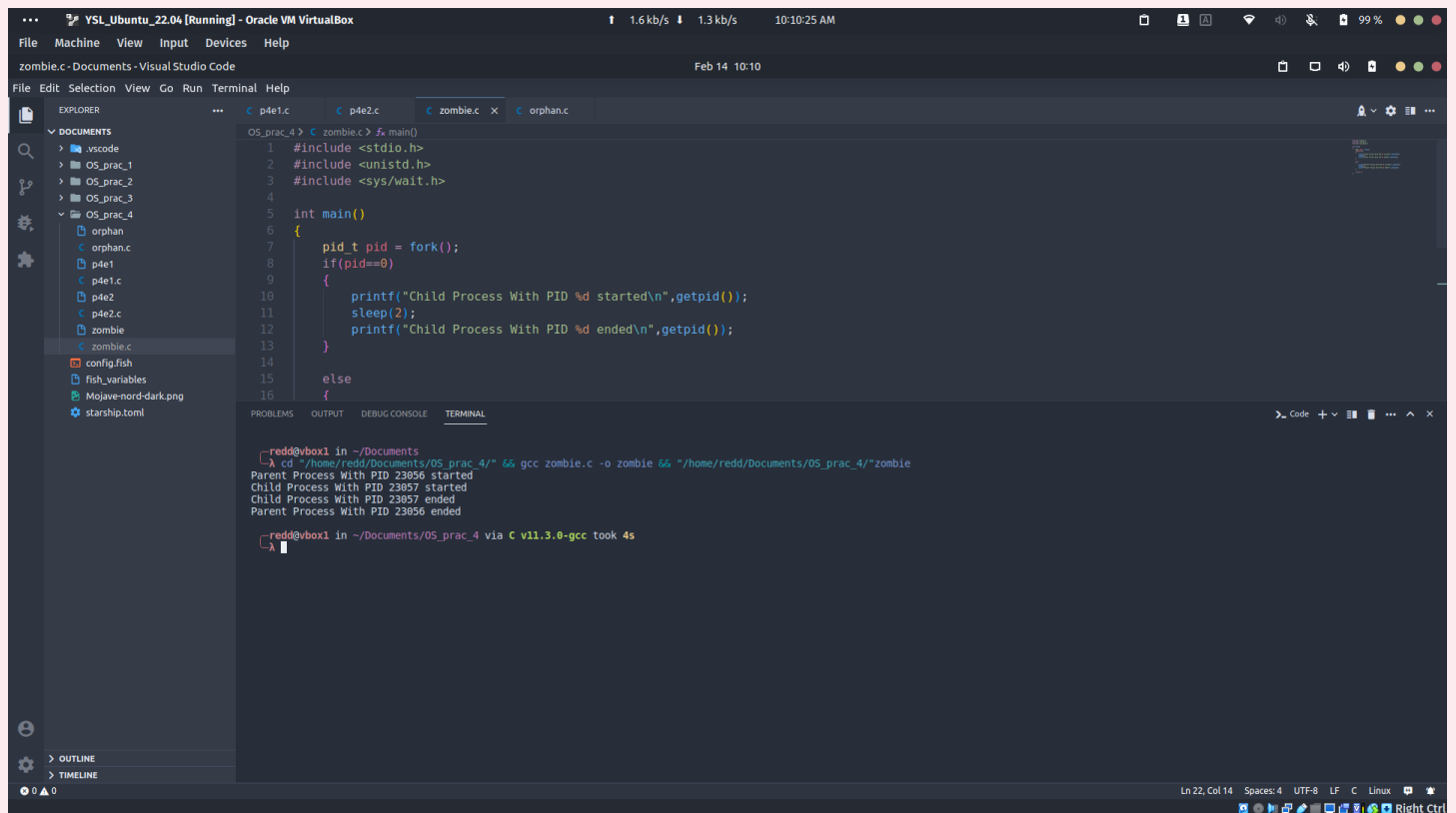
```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main()
{
    pid_t pid = fork();
    if(pid==0)
    {
        printf("Child Process With PID %d started\n",getpid());
        sleep(2);
        printf("Child Process With PID %d ended\n",getpid());
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 41
OS Practical 4

```
}  
  
else  
{  
  
    printf("Parent Process With PID %d started\n",getpid());  
    sleep(4);  
    printf("Parent Process With PID %d ended\n",getpid());  
  
}  
  
return 0;  
}
```

Output :



The screenshot shows a Visual Studio Code editor window with a C program named `zombie.c` open. The program is a simple fork-based process creation example. The terminal output shows the execution of the program, which prints the parent and child process IDs and sleeps for 4 seconds.

```
OS_prac_4 > c zombie.c > $ ./main()
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/wait.h>
4
5 int main()
6 {
7     pid_t pid = fork();
8     if(pid==0)
9     {
10         printf("Child Process With PID %d started\n",getpid());
11         sleep(2);
12         printf("Child Process With PID %d ended\n",getpid());
13     }
14     else
15     {
16
```

Terminal Output:

```
red@redbox1 in ~/Documents
└─$ cd ~/home/red@redbox1/Documents/OS_prac_4/" && gcc zombie.c -o zombie && ./zombie
Parent Process With PID 23856 started
Child Process With PID 23857 started
Child Process With PID 23857 ended
Parent Process With PID 23856 ended
red@redbox1 in ~/Documents/OS_prac_4 via C v11.3.0-gcc took 4s
└─$
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 41
OS Practical 4

Code (Orphan):

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

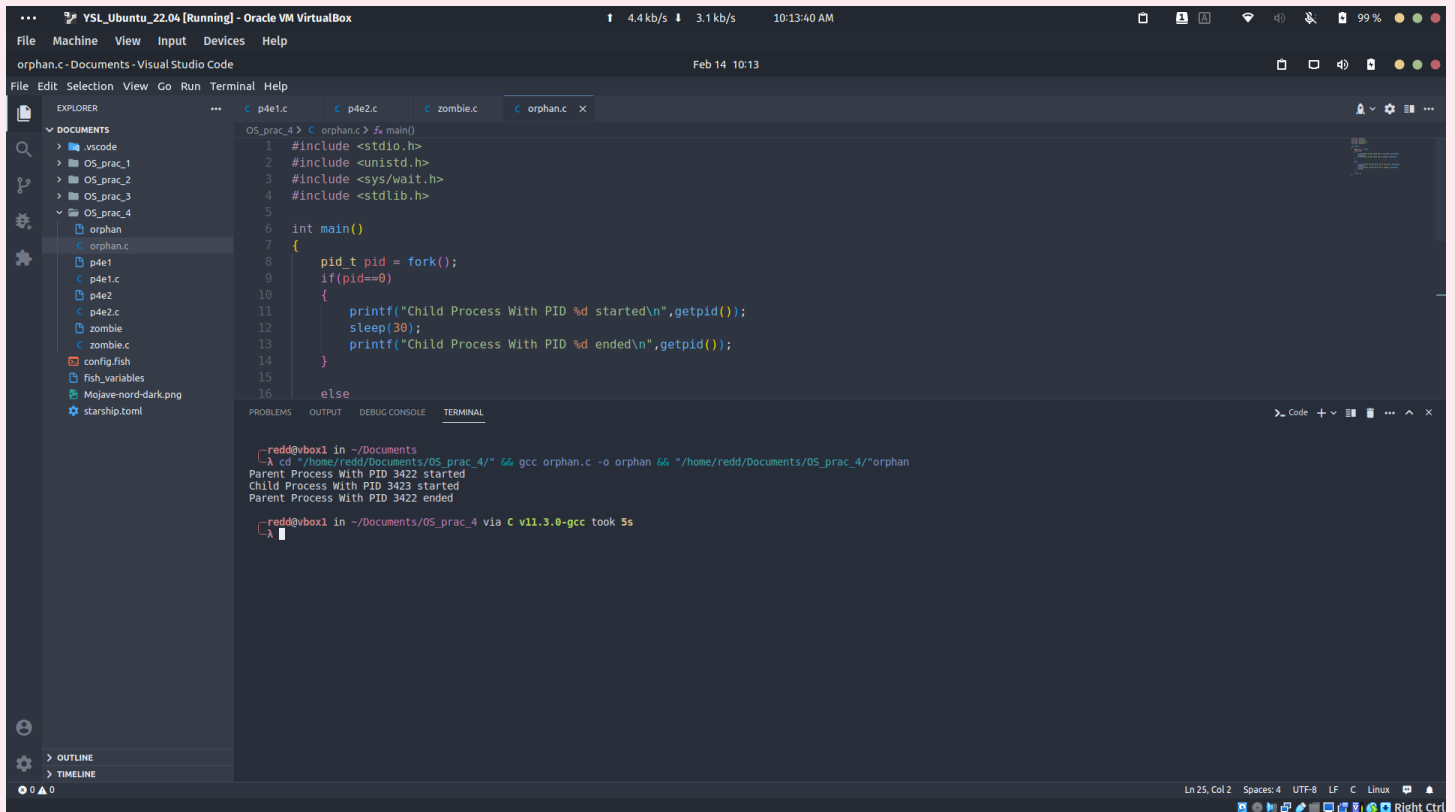
int main()
{
    pid_t pid = fork();
    if(pid==0)
    {
        printf("Child Process With PID %d started\n",getpid());
        sleep(30);
        printf("Child Process With PID %d ended\n",getpid());
    }

    else
    {
        printf("Parent Process With PID %d started\n",getpid());
        sleep(5);
        printf("Parent Process With PID %d ended\n",getpid());
        exit(0);
    }

    return 0;
}
```

Output :

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 41
OS Practical 4



The screenshot shows a Visual Studio Code editor window titled "YSL_Ubuntu_22.04 [Running] - Oracle VM VirtualBox". The editor is open to a file named "orphan.c" in the "OS_prac_4" directory. The code in "orphan.c" is a C program that demonstrates process forking. It includes headers for `<stdio.h>`, `<unistd.h>`, `<sys/wait.h>`, and `<stdlib.h>`. The `main` function calls `fork()` to create a child process. If the fork is successful (`pid == 0`), the child process prints its PID, sleeps for 30 seconds, and then prints "ended". The parent process prints its PID and "started".

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/wait.h>
4 #include <stdlib.h>
5
6 int main()
7 {
8     pid_t pid = fork();
9     if(pid==0)
10     {
11         printf("Child Process With PID %d started\n",getpid());
12         sleep(30);
13         printf("Child Process With PID %d ended\n",getpid());
14     }
15     else
16
```

The terminal output shows the execution of the program. The user runs `gcc orphan.c -o orphan` and then `./orphan`. The output shows the parent process starting with PID 3422, the child process starting with PID 3423, and the child process ending with PID 3422.

```
red@vbox1 in ~/Documents
└─$ cd ~/home/redd/documents/OS_prac_4/" && gcc orphan.c -o orphan && "/home/redd/Documents/OS_prac_4/"orphan
Parent Process With PID 3422 started
Child Process With PID 3423 started
Parent Process With PID 3422 ended
red@vbox1 in ~/Documents/OS_prac_4 via C v11.3.0-gcc took 5s
└─$
```