NAME: Sitara Rehman

Sap ID: 66454 BScs

Subject: Operating system Lab 08

Submitted to: Ma'am Ayesha Akram

Semester: 05

Task :01:

```
# The PWD team.                                                    #
####################################################################
[node1] (local) root@192.168.0.13 ~
$ cat > 1.c


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

int main() {
    fork(); // First fork
    fork(); // Second fork

    int pid = getpid();

    for (int i = 1; i <= 20; i++) {
        printf("Process ID: %d, Loop: %d\n", pid, i);
    }
```

DELETE EDITOR

```
[node1] (local) root@192.168.0.13 ~
$ gcc 1.c -o lab1
[node1] (local) root@192.168.0.13 ~
$ ./lab1
Process ID: 3755, Loop: 1Process ID: 3754, Loop: 1
Process ID: 3754, Loop: 2Process ID: 3756, Loop: 1
Process ID: 3756, Loop: 2
Process ID: 3756, Loop: 3

Process ID: 3756, Loop: 4Process ID: 3757, Loop: 1

Process ID: 3757, Loop: 2
Process ID: 3757, Loop: 3
Process ID: 3757, Loop: 4Process ID: 3756, Loop: 5
Process ID: 3757, Loop: 5

Process ID: 3756, Loop: 6Process ID: 3757, Loop: 6
```

Task:2

```
    }

    return 0;
}
[node1] (local) root@192.168.0.13 ~
$ gcc 2.c -o lab2
[node1] (local) root@192.168.0.13 ~
$ ./lab2
I am the Parent Process
Parent PID: 5959
Parent's Parent PID: 10
I am the Child Process
Child PID: 5960
Parent PID: 1
[node1] (local) root@192.168.0.13 ~
$
```

Task:3

```
    return 0;
}
[node1] (local) root@192.168.0.13 ~
$ gcc 3.c -o lab3
[node1] (local) root@192.168.0.13 ~
$ ./lab3
Child: My PID is 6652
Child: Exiting now...
Parent: My PID is 6651
Parent: Child has finished. Now I'm done.
[node1] (local) root@192.168.0.13 ~
$
```

Manual code 1:

```
int main(void)
{
    printf("The PID of this process (PID)= %d\n", getpid());
    printf("The PID of Parent process (PPID)= %d\n", getppid());
    return 0;
}
[node2] (local) root@192.168.0.22 ~
$ gcc lab8_e1.c -o lab8_e1
[node2] (local) root@192.168.0.22 ~
$ ./lab8_e1
The PID of this process (PID)= 1945
The PID of Parent process (PPID)= 11
[node2] (local) root@192.168.0.22 ~
```

Manual code 2:

```
    int x = 5;

    pid_t pid = getpid();

    printf("Value of X in PID= %d is %d\n", pid, x);

    return 0;
}
[node2] (local) root@192.168.0.22 ~
$ gcc lab8_e3.c -o lab8_e3
[node2] (local) root@192.168.0.22 ~
$ ./lab8_e3
Value of X in PID= 6132 is 5
Value of X in PID= 6133 is 5
[node2] (local) root@192.168.0.22 ~
$
```

Manual code 3:

```
    } else {
        printf("I'm the parent! My PID is %d and my child's PID is %d\n", getpid(),
    }

    return 0;
}
[node2] (local) root@192.168.0.22 ~
$ gcc lab8_e4.c -o lab8_e4
[node2] (local) root@192.168.0.22 ~
$ ./lab8_e4
I'm the parent! My PID is 8096 and my child's PID is 8097
I'm the child! My PID is 8097
[node2] (local) root@192.168.0.22 ~
$
```

Manual code 4:

```
}
[node2] (local) root@192.168.0.22 ~
$ gcc ex1.c -o ex1
[node2] (local) root@192.168.0.22 ~
$ ./ex1
PID of ex1.c = 10151
We are in ex2.c
PID of ex2.c = 10151
[node2] (local) root@192.168.0.22 ~
$
```

Manual code 5:

```
    }

    return 0;
}
[node2] (local) root@192.168.0.22 ~
$ gcc lab8_wait_exit.c -o lab8_wait_exit
[node2] (local) root@192.168.0.22 ~
$ ./lab8_wait_exit
Parent: Waiting for child to finish...
Child: I'm working...
Child: I'm done!
Parent: Child has finished. I can continue.
[node2] (local) root@192.168.0.22 ~
$
```