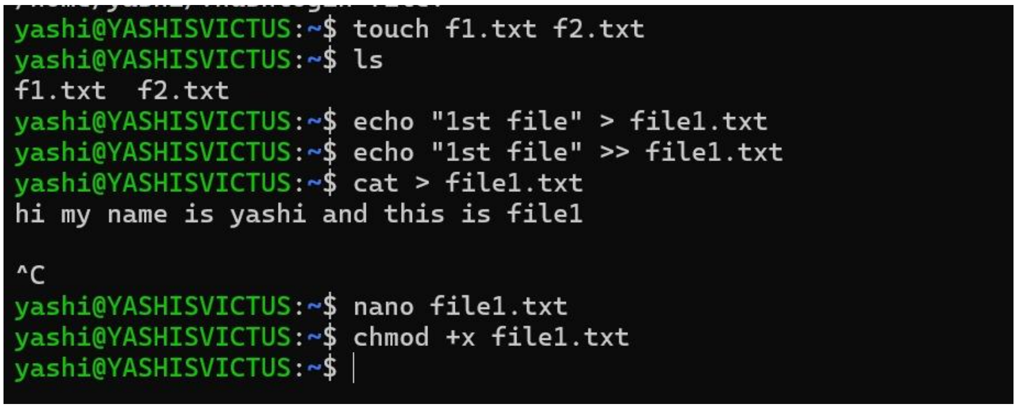
OPERATING SYSTEMS LAB

ASSIGNMENT 1

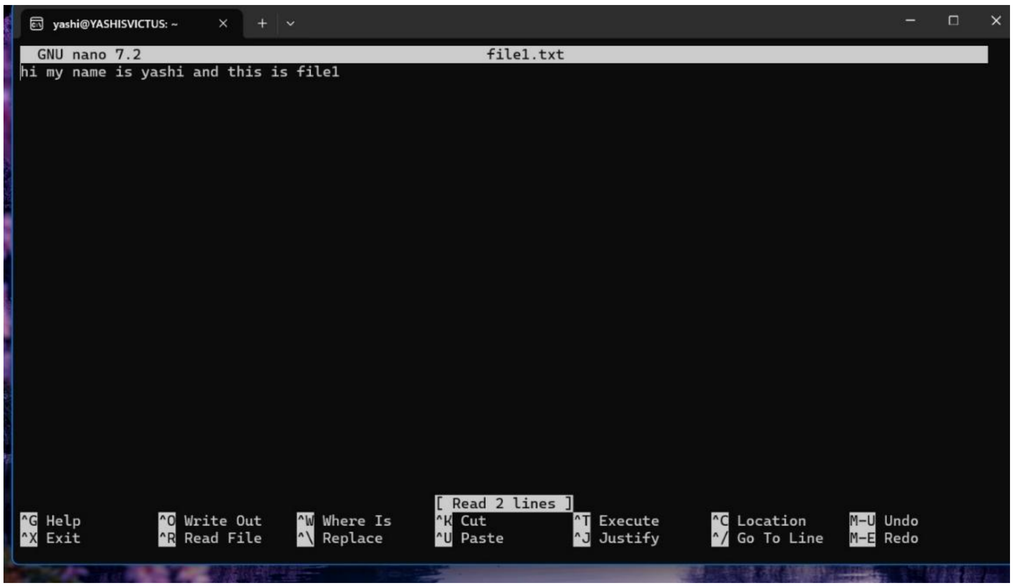
NAME - YASHASWINI KANAUJIA

REG. NO. - 23BCE0622

Basic Commands



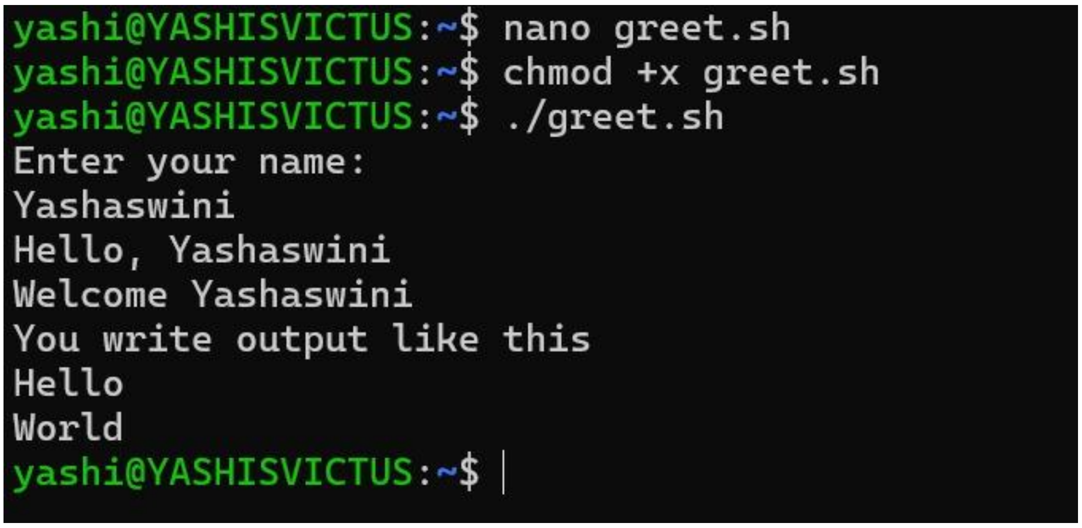
```
yashi@YASHISVICTUS:~$ touch f1.txt f2.txt
yashi@YASHISVICTUS:~$ ls
f1.txt  f2.txt
yashi@YASHISVICTUS:~$ echo "1st file" > file1.txt
yashi@YASHISVICTUS:~$ echo "1st file" >> file1.txt
yashi@YASHISVICTUS:~$ cat > file1.txt
hi my name is yashi and this is file1

^C
yashi@YASHISVICTUS:~$ nano file1.txt
yashi@YASHISVICTUS:~$ chmod +x file1.txt
yashi@YASHISVICTUS:~$ |
```



```
yashi@YASHISVICTUS: ~
  GNU nano 7.2                              file1.txt
hi my name is yashi and this is file1

                                  [ Read 2 lines ]
^G Help       ^O Write Out   ^W Where Is    ^K Cut       ^T Execute    ^C Location    M-U Undo
^X Exit       ^R Read File   ^\ Replace     ^U Paste     ^J Justify    ^/ Go To Line  M-E Redo
```

Basic Code:

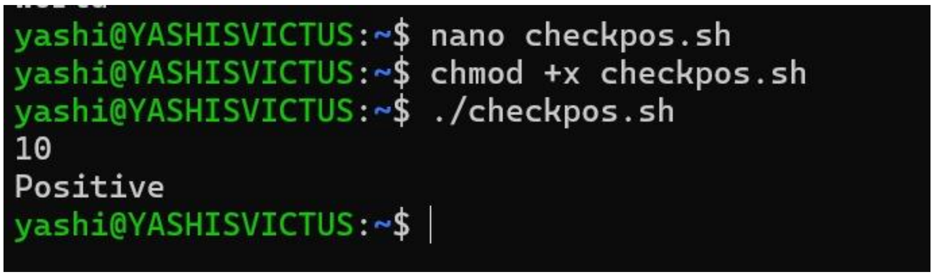
```
#!/bin/bash
echo "Enter your name: "
read name
echo "Hello, $name"
printf "Welcome %s\n" "$name"
echo "You write output like this"
echo "Hello" > f1.txt
echo "World" > f2.txt
cat f1.txt
cat f2.txt
```



```
yashi@YASHISVICTUS:~$ nano greet.sh
yashi@YASHISVICTUS:~$ chmod +x greet.sh
yashi@YASHISVICTUS:~$ ./greet.sh
Enter your name:
Yashaswini
Hello, Yashaswini
Welcome Yashaswini
You write output like this
Hello
World
yashi@YASHISVICTUS:~$
```

Check if Number is Positive, Negative or Zero:

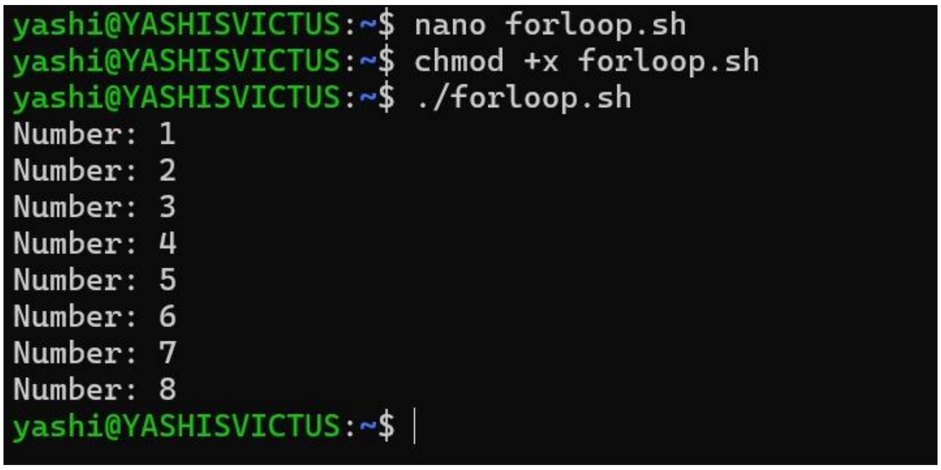
```
read num
if [ $num -gt 0 ]; then
   echo "Positive"
elif [ $num -lt 0 ]; then
   echo "Negative"
else
   echo "Zero"
fi
```



```
yashi@YASHISVICTUS:~$ nano checkpos.sh
yashi@YASHISVICTUS:~$ chmod +x checkpos.sh
yashi@YASHISVICTUS:~$ ./checkpos.sh
10
Positive
yashi@YASHISVICTUS:~$
```

For Loop:

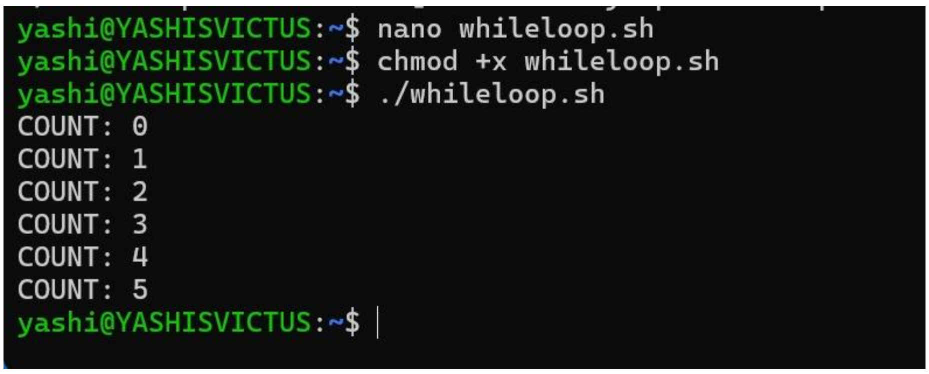
```
#!/bin/bash
for i in 1 2 3 4 5
do
   echo "Number: $i"
done
```



```
yashi@YASHISVICTUS:~$ nano forloop.sh
yashi@YASHISVICTUS:~$ chmod +x forloop.sh
yashi@YASHISVICTUS:~$ ./forloop.sh
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
Number: 6
Number: 7
Number: 8
yashi@YASHISVICTUS:~$
```

While Loop:

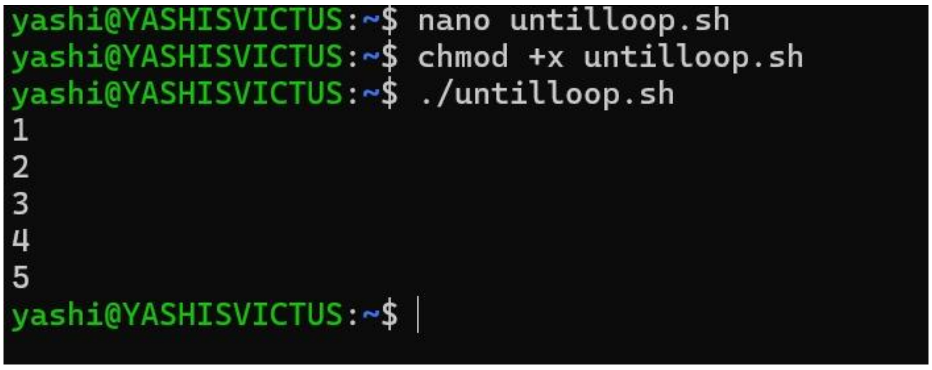
```
#!/bin/bash

count=0

while [ $count -le 5 ]

do

   echo "COUNT: $count"

   count=$((count + 1))

done
```



```
yashi@YASHISVICTUS:~$ nano whileloop.sh
yashi@YASHISVICTUS:~$ chmod +x whileloop.sh
yashi@YASHISVICTUS:~$ ./whileloop.sh
COUNT: 0
COUNT: 1
COUNT: 2
COUNT: 3
COUNT: 4
COUNT: 5
yashi@YASHISVICTUS:~$ |
```

Until Loop:

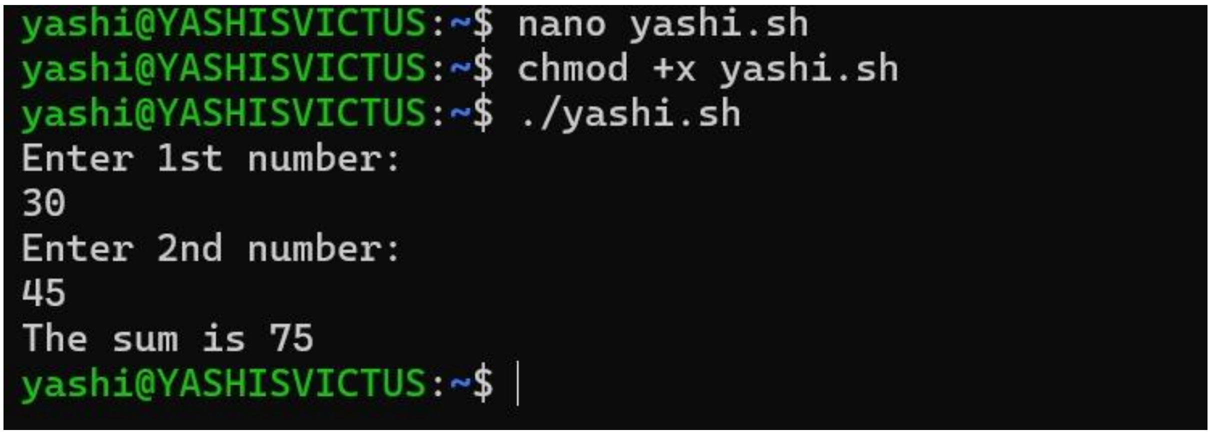
```
#!/bin/bash

n=1

until [ $n -gt 5 ]

do

   echo $n

   n=$((n+1))

done
```



```
yashi@YASHISVICTUS:~$ nano untilloop.sh
yashi@YASHISVICTUS:~$ chmod +x untilloop.sh
yashi@YASHISVICTUS:~$ ./untilloop.sh
1
2
3
4
5
yashi@YASHISVICTUS:~$ |
```

Addition of 2 Numbers:

```
#!/bin/bash

echo "Enter 1st number: "

read num1

echo "Enter 2nd number: "

read num2

sum=$((num1 + num2))

echo "The sum is $sum"
```



```
yashi@YASHISVICTUS:~$ nano yashi.sh
yashi@YASHISVICTUS:~$ chmod +x yashi.sh
yashi@YASHISVICTUS:~$ ./yashi.sh
Enter 1st number:
30
Enter 2nd number:
45
The sum is 75
yashi@YASHISVICTUS:~$
```

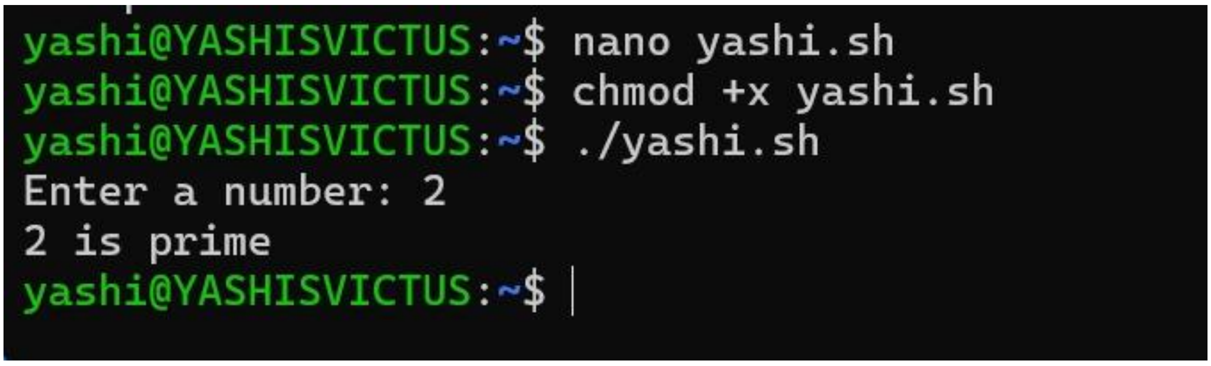
Check for Prime Number:

```
#!/bin/bash
echo "Enter a number: "
read num
count=0
for (( i=1; i<=num; i++ ))
do
   if [ $((num % i)) -eq 0 ]
   then
      count=$((count+1))
   fi
done
if [ $count -eq 2 ]
then
   echo "$num is prime"
else
   echo "$num is not prime"
fi
```

OR

```
read -p "Enter a number: " num
if [ "$num" -eq 1 ]; then
   echo "$num is not prime"
   exit
fi
is_prime=1
for ((i = 2; i * i <= num; i++))
do
```

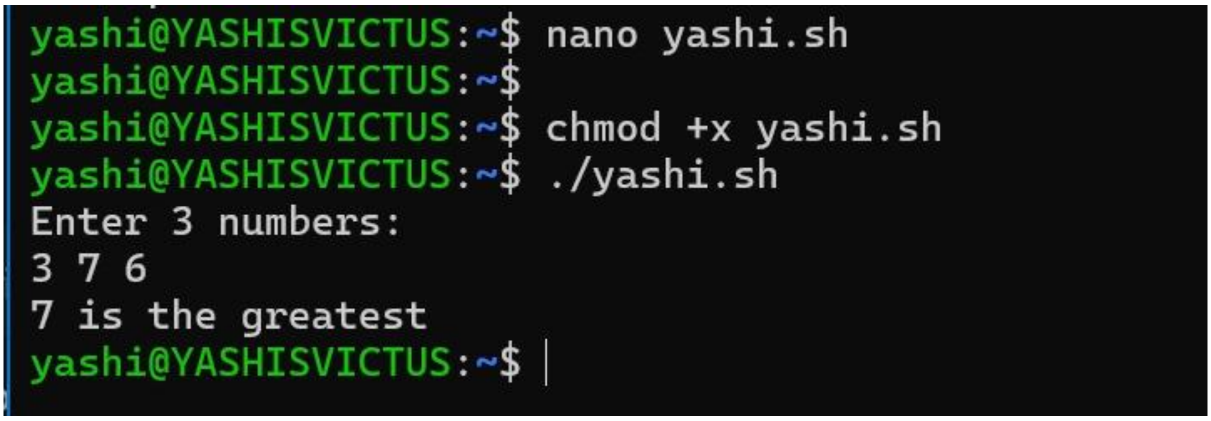
```
    if [ $((num % i)) -eq 0 ]; then
        is_prime=0
        break
    fi
done
if [ "$is_prime" -eq 1 ]; then
    echo "$num is prime"
else
    echo "$num is not prime"
fi
```



```
yashi@YASHISVICTUS:~$ nano yashi.sh
yashi@YASHISVICTUS:~$ chmod +x yashi.sh
yashi@YASHISVICTUS:~$ ./yashi.sh
Enter a number: 2
2 is prime
yashi@YASHISVICTUS:~$
```

Greatest Among 3 Numbers:

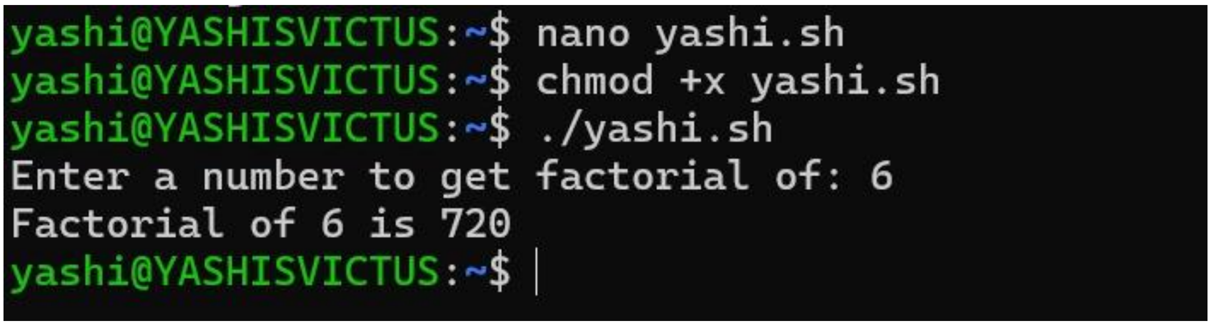
```
#!/bin/bash
echo "Enter 3 numbers: "
read a b c
if [ $a -gt $b ] && [ $a -gt $c ]; then
   echo "$a is the greatest"
elif [ $b -gt $a ] && [ $b -gt $c ]; then
   echo "$b is the greatest"
else
   echo "$c is the greatest"
fi
```



```
yashi@YASHISVICTUS:~$ nano yashi.sh
yashi@YASHISVICTUS:~$
yashi@YASHISVICTUS:~$ chmod +x yashi.sh
yashi@YASHISVICTUS:~$ ./yashi.sh
Enter 3 numbers:
3 7 6
7 is the greatest
yashi@YASHISVICTUS:~$
```

Factorial of a Number:

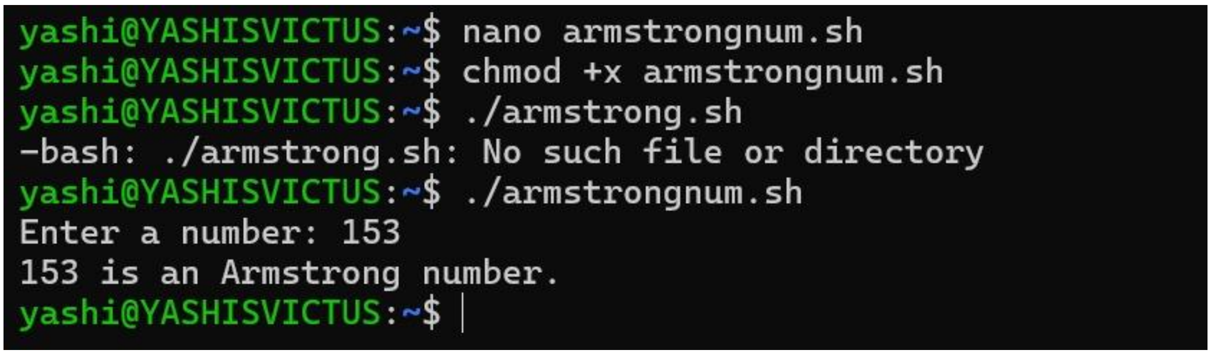
```
#!/bin/bash

read -p "Enter a number to get factorial of: " num

factorial=1

if [ $num -eq 0 ]; then

   echo "Factorial of $num is 1"

   exit

fi

for (( i=1; i<=num; i++ )); do

   factorial=$((factorial * i))

done

echo "Factorial of $num is $factorial"
```



```
yashi@YASHISVICTUS:~$ nano yashi.sh
yashi@YASHISVICTUS:~$ chmod +x yashi.sh
yashi@YASHISVICTUS:~$ ./yashi.sh
Enter a number to get factorial of: 6
Factorial of 6 is 720
yashi@YASHISVICTUS:~$
```

Armstrong Number:

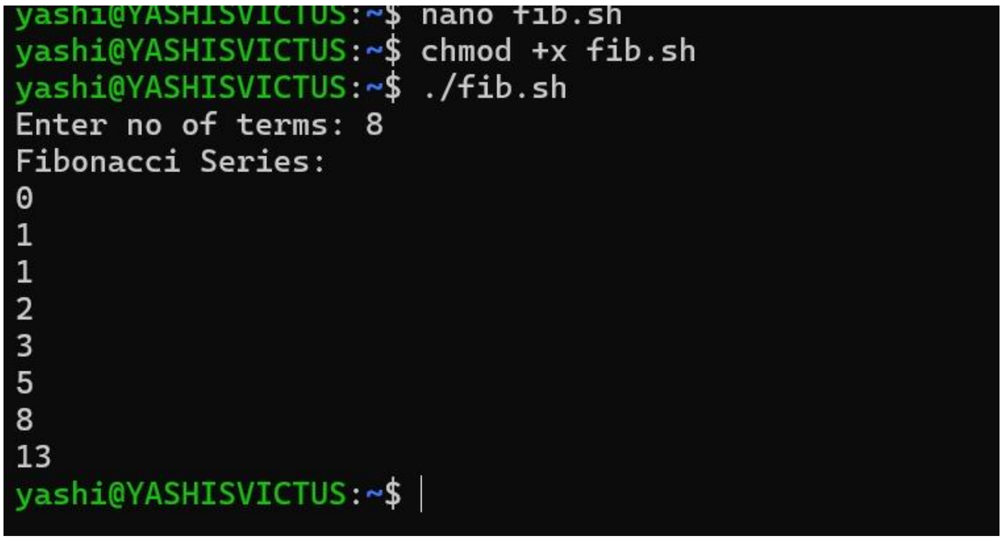
```bash
#!/bin/bash
read -p "Enter a number: " num
copy=$num
sum=0
n=0
temp=$copy
while [ $temp -gt 0 ]; do
    temp=$((temp / 10))
    n=$((n + 1))
done
copy=$num
while [ $copy -gt 0 ]; do
    j=$((copy % 10))
    pow=1
    for (( i=1; i<=n; i++ )); do
        pow=$((pow * j))
    done
    sum=$((sum + pow))
    copy=$((copy / 10))
done
if [ $sum -eq $num ]; then
    echo "$num is an Armstrong number."
else
    echo "$num is NOT an Armstrong number."
fi
```



```
yashi@YASHISVICTUS:~$ nano armstrongnum.sh
yashi@YASHISVICTUS:~$ chmod +x armstrongnum.sh
yashi@YASHISVICTUS:~$ ./armstrong.sh
-bash: ./armstrong.sh: No such file or directory
yashi@YASHISVICTUS:~$ ./armstrongnum.sh
Enter a number: 153
153 is an Armstrong number.
yashi@YASHISVICTUS:~$
```

Fibonacci Series:

```
#!/bin/bash

read -p "Enter no of terms: " num

echo "Fibonacci Series: "

if [ "$num" -eq 1 ]; then

   echo "0"

else

   echo "0"

   echo "1"

   f=0

   s=1

   count=2

   while [ $count -lt $num ]; do

      k=$((f + s))

      echo "$k"

      f=$s

      s=$k

      count=$((count + 1))

   done

fi
```

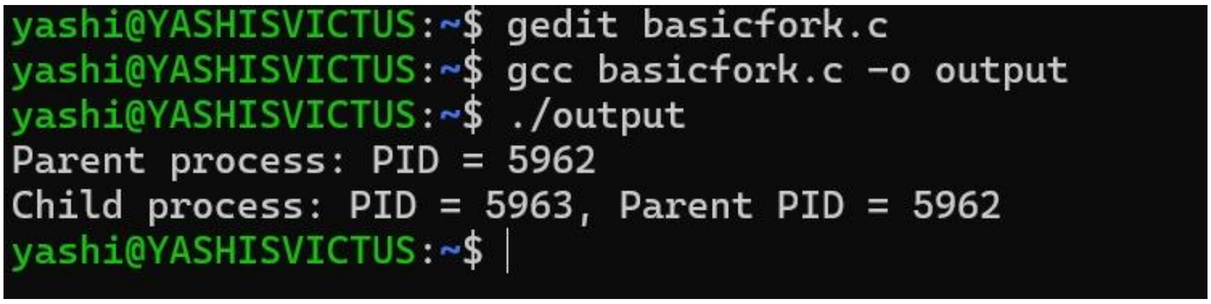


```
yashi@YASHISVICTUS:~$ nano fib.sh
yashi@YASHISVICTUS:~$ chmod +x fib.sh
yashi@YASHISVICTUS:~$ ./fib.sh
Enter no of terms: 8
Fibonacci Series:
0
1
1
2
3
5
8
13
yashi@YASHISVICTUS:~$
```

Experiment 1: Basic Fork - Parent and Child Identification

1)

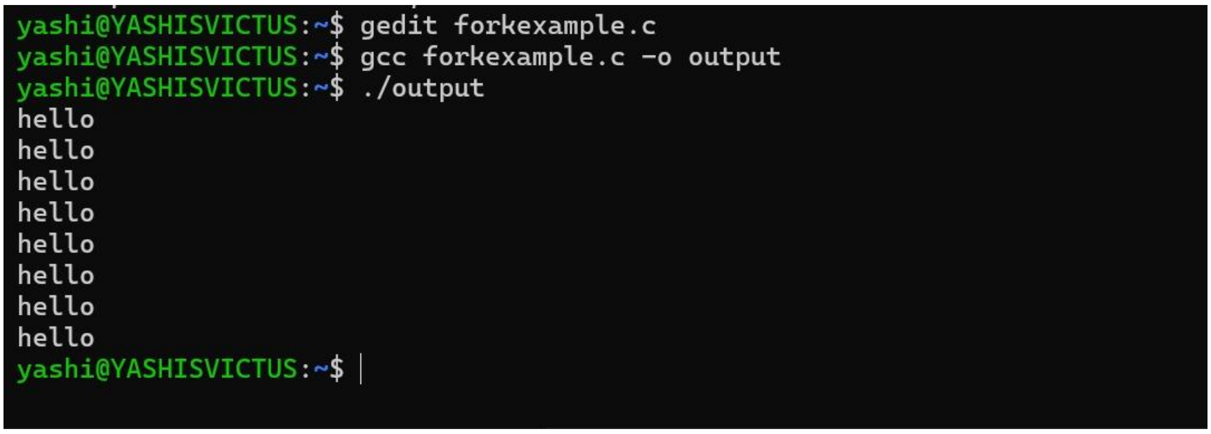
```
#include <stdio.h>

#include <unistd.h>

int main() {

   pid_t pid = fork();

   if (pid > 0) {

      printf("Parent process: PID = %d\n", getpid());

   } else if (pid == 0) {

      printf("Child process: PID = %d, Parent PID = %d\n", getpid(), getppid());

   } else {

      printf("Fork failed!\n");

   }

   return 0;

}
```



```
yashi@YASHISVICTUS:~$ gedit basicfork.c
yashi@YASHISVICTUS:~$ gcc basicfork.c -o output
yashi@YASHISVICTUS:~$ ./output
Parent process: PID = 5962
Child process: PID = 5963, Parent PID = 5962
yashi@YASHISVICTUS:~$
```

2)

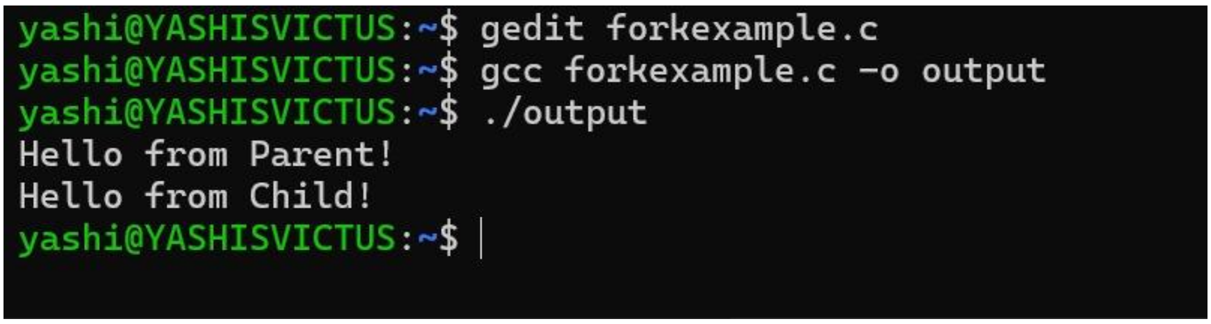
```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
   fork();
   fork();
   fork();
   printf("hello\n");
   return 0;
}
```



```
yashi@YASHISVICTUS:~$ gedit forkexample.c
yashi@YASHISVICTUS:~$ gcc forkexample.c -o output
yashi@YASHISVICTUS:~$ ./output
hello
hello
hello
hello
hello
hello
hello
hello
yashi@YASHISVICTUS:~$
```

3)

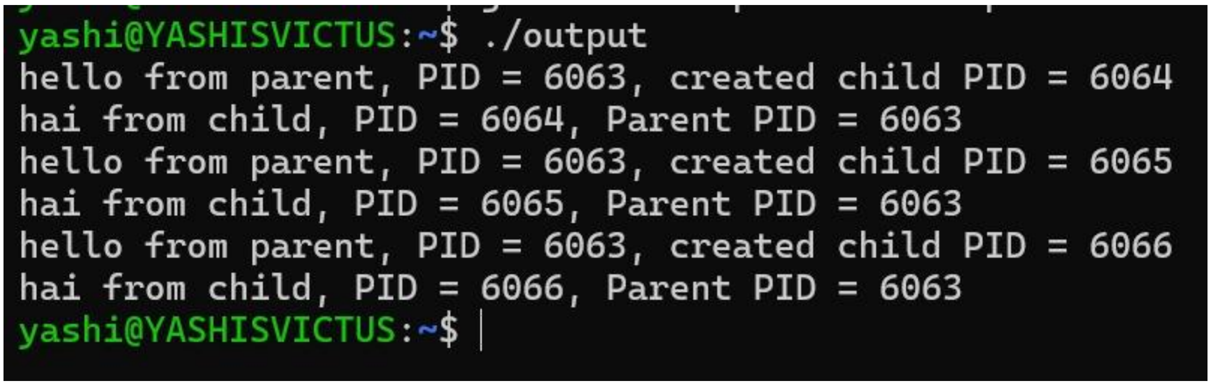
```c
#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

void forkexample()

{

   pid_t p;

   p = fork();

   if (p < 0)

   {

      perror("fork fail");

      exit(1);

   }

   else if (p == 0) // Child process because return value zero

      printf("Hello from Child!\n");

   else // Parent process because return value non-zero.

      printf("Hello from Parent!\n");

}

int main()

{

   forkexample();

   return 0;

}
```



```
yashi@YASHISVICTUS:~$ gedit forkexample.c
yashi@YASHISVICTUS:~$ gcc forkexample.c -o output
yashi@YASHISVICTUS:~$ ./output
Hello from Parent!
Hello from Child!
yashi@YASHISVICTUS:~$
```

4)

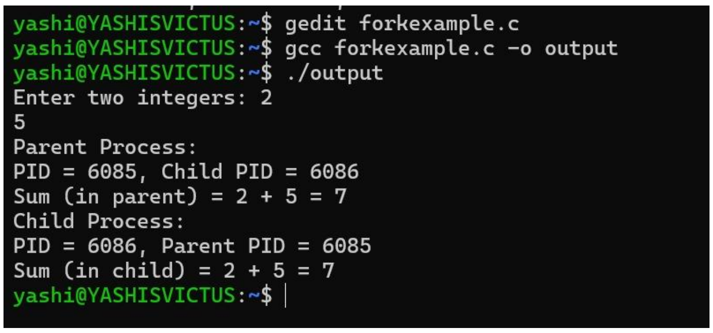
```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main() {
   int i;
   pid_t pid;
   for (i = 0; i < 3; i++) { // Change this number to create more children
      pid = fork();
      if (pid == 0) {
         // Child process
         printf("hai from child, PID = %d, Parent PID = %d\n", getpid(), getppid());
         return 0; // End child process after printing
      }
      else if (pid > 0) // Parent process
         printf("hello from parent, PID = %d, created child PID = %d\n", getpid(), pid);
      else // fork failed
         printf("fork failed\n");
   }
   return 0;
}
```



```
yashi@YASHISVICTUS:~$ ./output
hello from parent, PID = 6063, created child PID = 6064
hai from child, PID = 6064, Parent PID = 6063
hello from parent, PID = 6063, created child PID = 6065
hai from child, PID = 6065, Parent PID = 6063
hello from parent, PID = 6063, created child PID = 6066
hai from child, PID = 6066, Parent PID = 6063
yashi@YASHISVICTUS:~$
```

5)

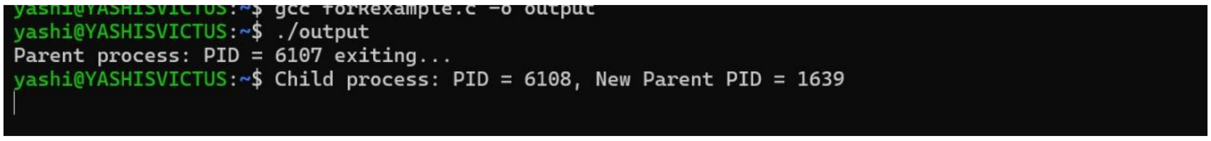
```c
#include <stdio.h>

#include <unistd.h>

int main() {

    int a, b, sum;

    printf("Enter two integers: ");

    scanf("%d %d", &a, &b);

    pid_t pid = fork();

    if (pid < 0) {

        printf("Fork failed!\n");

        return 1;

    } else if (pid == 0) { // Child process

        sum = a + b;

        printf("Child Process:\n");

        printf("PID = %d, Parent PID = %d\n", getpid(), getppid());

        printf("Sum (in child) = %d + %d = %d\n", a, b, sum);

    } else { // Parent process

        sum = a + b;

        printf("Parent Process:\n");

        printf("PID = %d, Child PID = %d\n", getpid(), pid);

        printf("Sum (in parent) = %d + %d = %d\n", a, b, sum);

    }

    return 0;

}
```



```
yashi@YASHISVICTUS:~$ gedit forkexample.c
yashi@YASHISVICTUS:~$ gcc forkexample.c -o output
yashi@YASHISVICTUS:~$ ./output
Enter two integers: 2
5
Parent Process:
PID = 6085, Child PID = 6086
Sum (in parent) = 2 + 5 = 7
Child Process:
PID = 6086, Parent PID = 6085
Sum (in child) = 2 + 5 = 7
yashi@YASHISVICTUS:~$
```

Experiment 2: Orphan Process Creation

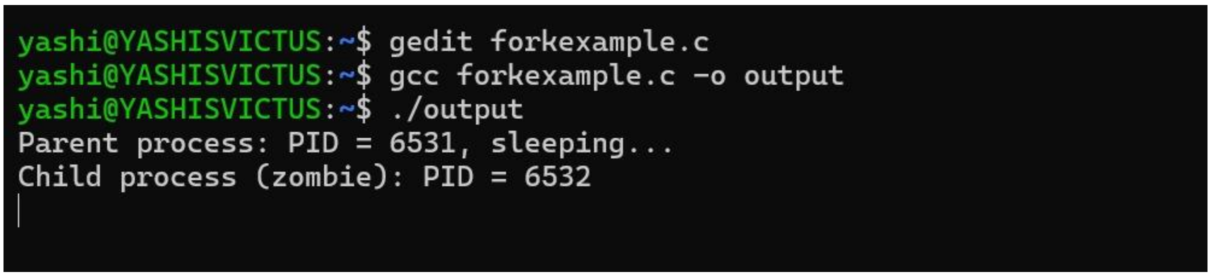
```
#include <stdio.h>

#include <unistd.h>

int main() {

   pid_t pid = fork();

   if (pid == 0) {

      sleep(5);

      printf("Child process: PID = %d, New Parent PID = %d\n", getpid(), getppid());

   } else {

      printf("Parent process: PID = %d exiting...\n", getpid());

   }

   return 0;

}
```



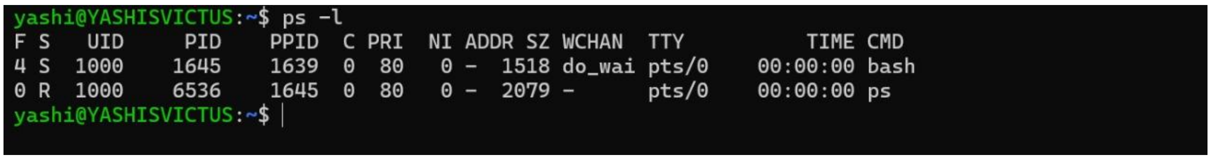
```
yashi@YASHISVICTUS:~$ gcc forkexample.c -o output
yashi@YASHISVICTUS:~$ ./output
Parent process: PID = 6107 exiting...
yashi@YASHISVICTUS:~$ Child process: PID = 6108, New Parent PID = 1639
```

Experiment 3: Zombie Process Creation

```c
#include <stdio.h>

#include <unistd.h>

int main() {

    pid_t pid = fork();

    if (pid == 0) {

        printf("Child process (zombie): PID = %d\n", getpid());

        // Child process ends immediately

    } else {

        printf("Parent process: PID = %d, sleeping...\n", getpid());

        sleep(10);

    }

    return 0;

}
```



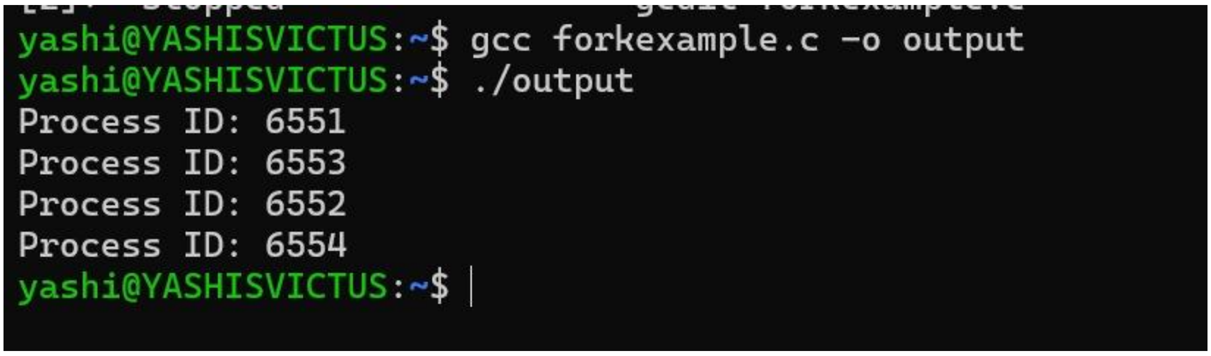
```
yashi@YASHISVICTUS:~$ gedit forkexample.c
yashi@YASHISVICTUS:~$ gcc forkexample.c -o output
yashi@YASHISVICTUS:~$ ./output
Parent process: PID = 6531, sleeping...
Child process (zombie): PID = 6532
```



```
yashi@YASHISVICTUS:~$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000    1645    1639  0  80   0 -  1518 do_wai pts/0    00:00:00 bash
0 R  1000    6536    1645  0  80   0 -  2079 -      pts/0    00:00:00 ps
yashi@YASHISVICTUS:~$
```

Bonus: Multiple fork() Calls

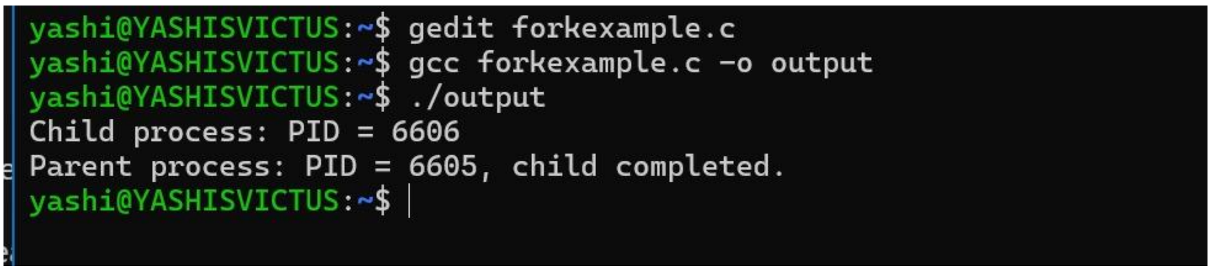
```c
#include <stdio.h>

#include <unistd.h>

int main() {

   fork();

   fork();

   printf("Process ID: %d\n", getpid());

   return 0;

}
```



```
yashi@YASHISVICTUS:~$ gcc forkexample.c -o output
yashi@YASHISVICTUS:~$ ./output
Process ID: 6551
Process ID: 6553
Process ID: 6552
Process ID: 6554
yashi@YASHISVICTUS:~$ 
```

Bonus: Using wait() to Prevent Zombie

```c
#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

    pid_t pid = fork();

    if (pid == 0) {

        printf("Child process: PID = %d\n", getpid());

    } else {

        wait(NULL);

        printf("Parent process: PID = %d, child completed.\n", getpid());

    }

    return 0;

}
```



```
yashi@YASHISVICTUS:~$ gedit forkexample.c
yashi@YASHISVICTUS:~$ gcc forkexample.c -o output
yashi@YASHISVICTUS:~$ ./output
Child process: PID = 6606
Parent process: PID = 6605, child completed.
yashi@YASHISVICTUS:~$
```

Assessment

Aim: To demonstrate the concepts of orphan and zombie processes in a Unix-like operating system by writing a C program. The program should:

1. Create an orphan process: Show how a child process continues executing after its parent terminates, and the init (or system) process adopts it.
2. Create a zombie process: Demonstrate how a child process becomes a zombie when it terminates but its parent does not collect its exit status using the wait() system call.

The program must use fork(), sleep(), and wait() appropriately, and include clear messages to indicate process states (e.g., "Child running," "Parent exiting," "Zombie created"). Additionally, students are expected to observe process states using tools like ps -elf or top to verify the creation of orphan and zombie processes.

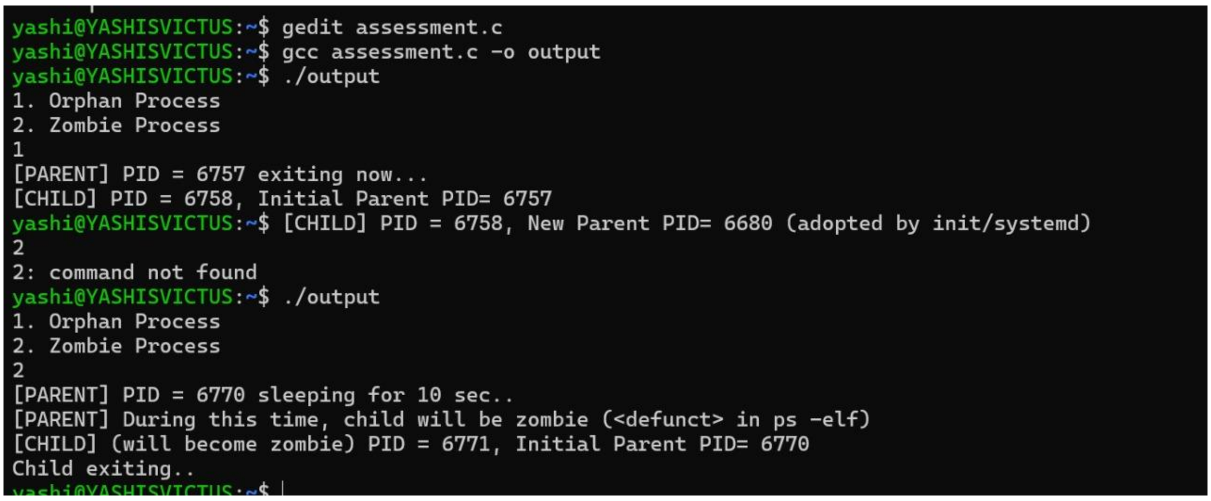
Program:

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>


int main(void) {

int choice;

printf("1. Orphan Process\n");

printf("2. Zombie Process\n");

if (scanf("%d", &choice) != 1) return 0;


if (choice == 1) {

   pid_t pid = fork();


   if (pid < 0) {

      perror("fork");
```

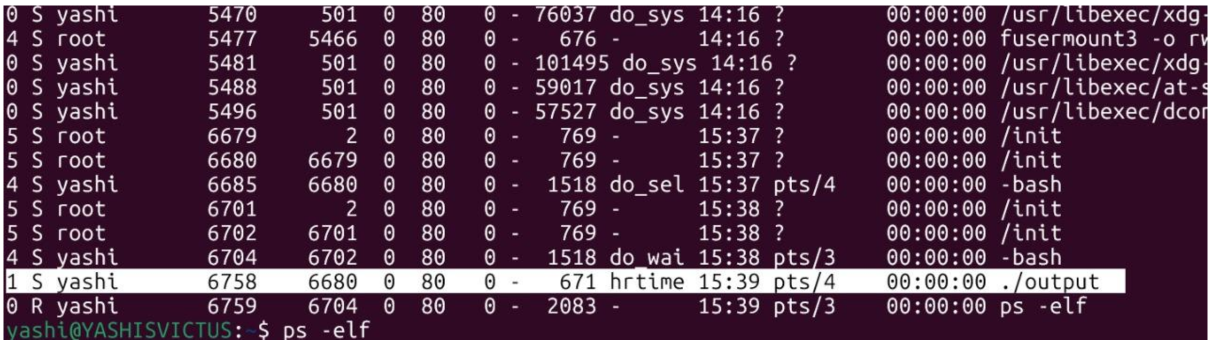
```
        exit(1);
    } else if (pid == 0) {
        // Child
        pid_t mypid = getpid();
        pid_t ppid_before = getppid();
        printf("[CHILD] PID = %d, Initial Parent PID= %d\n", mypid, ppid_before);

        // Wait a bit so parent can exit and init/systemd can adopt us
        sleep(2);

        pid_t ppid_after = getppid();
        printf("[CHILD] PID = %d, New Parent PID= %d (adopted by init/systemd)\n",
               mypid, ppid_after);
        // keep the child around a moment so you can see in ps
        sleep(5);
        exit(0);
    } else {
        // Parent
        printf("[PARENT] PID = %d exiting now...\n", getpid());
        // Parent exits immediately; do not wait() so the child becomes orphan and is
adopted
        exit(0);
    }
} else if (choice == 2) {
    pid_t pid = fork();

    if (pid < 0) {
        perror("fork");
        exit(1);
    } else if (pid == 0) {
        // Child will exit quickly, becoming zombie until parent wait()s.
        printf("[CHILD] (will become zombie) PID = %d, Initial Parent PID= %d\n",
               getpid(), getppid());
        printf("Child exiting..\n");
        exit(0);
    } else {
        // Parent intentionally sleeps before wait() to show zombie state
        printf("[PARENT] PID = %d sleeping for 10 sec..\n", getpid());
        printf("[PARENT] During this time, child will be zombie (<defunct> in ps -elf)\n");
        sleep(10);

        // Now reap the child
        int status = 0;
        waitpid(pid, &status, 0);
        // Small delay so user sees that zombie is gone after wait
        sleep(1);
    }
} else {
    printf("Invalid choice\n");
}

return 0;
}
```

Output:

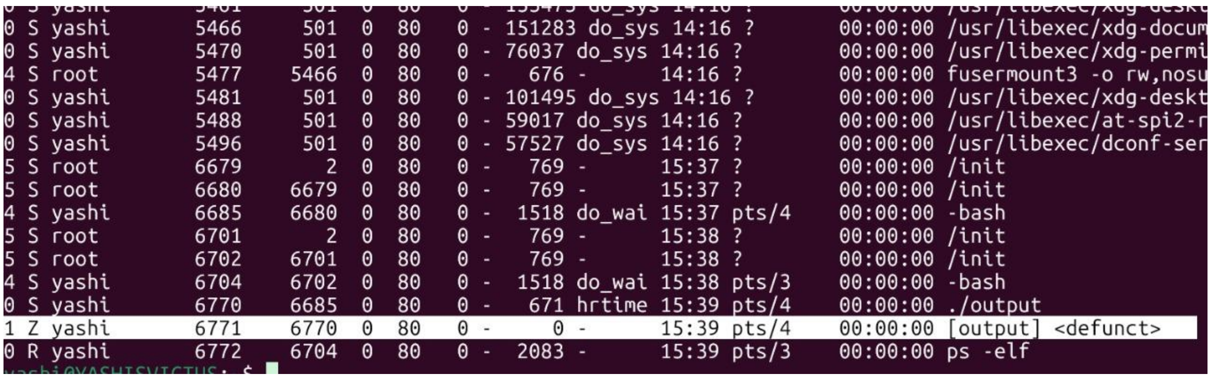
(On my computer)



```
yashi@YASHISVICTUS:~$ gedit assessment.c
yashi@YASHISVICTUS:~$ gcc assessment.c -o output
yashi@YASHISVICTUS:~$ ./output
1. Orphan Process
2. Zombie Process
1
[PARENT] PID = 6757 exiting now...
[CHILD] PID = 6758, Initial Parent PID= 6757
yashi@YASHISVICTUS:~$ [CHILD] PID = 6758, New Parent PID= 6680 (adopted by init/systemd)
2
2: command not found
yashi@YASHISVICTUS:~$ ./output
1. Orphan Process
2. Zombie Process
2
[PARENT] PID = 6770 sleeping for 10 sec..
[PARENT] During this time, child will be zombie (<defunct> in ps -elf)
[CHILD] (will become zombie) PID = 6771, Initial Parent PID= 6770
Child exiting..
yashi@YASHISVICTUS:~$
```

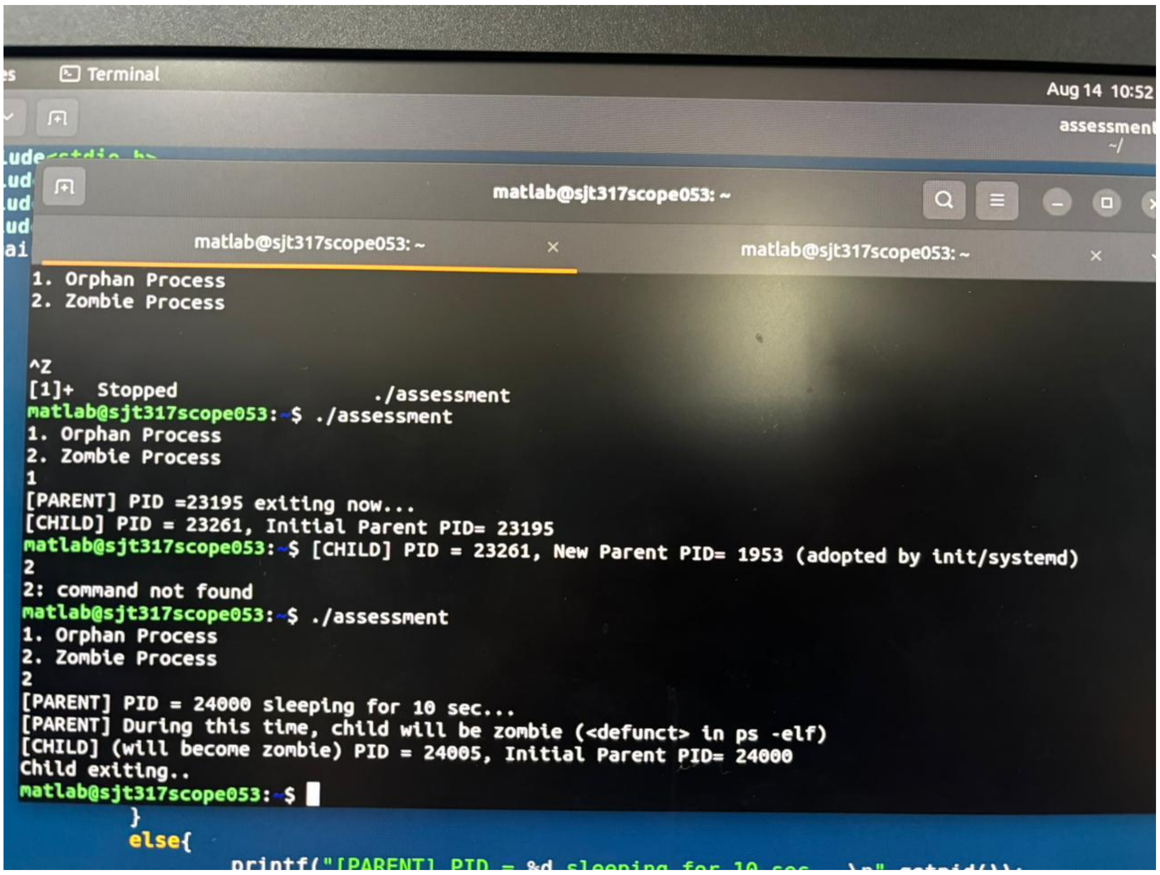


```
0 S yashi       5470     501  0  80   0 -  76037 do_sys 14:16 ?        00:00:00 /usr/libexec/xdg
4 S root        5477    5466  0  80   0 -    676 -      14:16 ?        00:00:00 fusermount3 -o rw
0 S yashi       5481     501  0  80   0 - 101495 do_sys 14:16 ?        00:00:00 /usr/libexec/xdg
0 S yashi       5488     501  0  80   0 -  59017 do_sys 14:16 ?        00:00:00 /usr/libexec/at-s
0 S yashi       5496     501  0  80   0 -  57527 do_sys 14:16 ?        00:00:00 /usr/libexec/dcon
5 S root        6679       2  0  80   0 -    769 -      15:37 ?        00:00:00 /init
5 S root        6680    6679  0  80   0 -    769 -      15:37 ?        00:00:00 /init
4 S yashi       6685    6680  0  80   0 -   1518 do_sel 15:37 pts/4    00:00:00 -bash
5 S root        6701       2  0  80   0 -    769 -      15:38 ?        00:00:00 /init
5 S root        6702    6701  0  80   0 -    769 -      15:38 ?        00:00:00 /init
4 S yashi       6704    6702  0  80   0 -   1518 do_wai 15:38 pts/3    00:00:00 -bash
1 S yashi       6758    6680  0  80   0 -    671 hrtime 15:39 pts/4    00:00:00 ./output
0 R yashi       6759    6704  0  80   0 -   2083 -      15:39 pts/3    00:00:00 ps -elf
yashi@YASHISVICTUS:~$ ps -elf
```

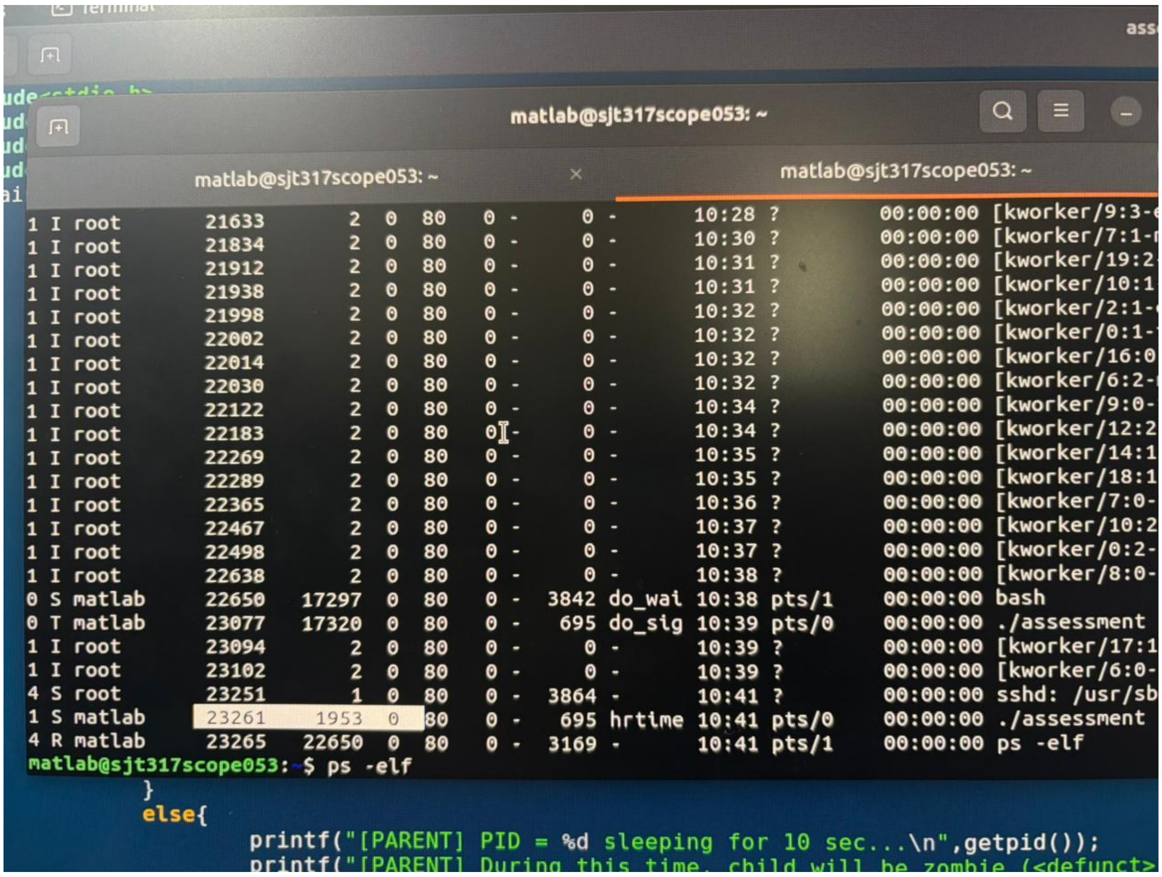


```
0 S yashi      5466     501  0  80   0 - 151283 do_sys 14:16 ?        00:00:00 /usr/libexec/xdg-docum
0 S yashi      5470     501  0  80   0 -  76037 do_sys 14:16 ?        00:00:00 /usr/libexec/xdg-permi
4 S root       5477    5466  0  80   0 -    676 -      14:16 ?        00:00:00 fusermount3 -o rw,nosu
0 S yashi      5481     501  0  80   0 - 101495 do_sys 14:16 ?        00:00:00 /usr/libexec/xdg-deskt
0 S yashi      5488     501  0  80   0 -  59017 do_sys 14:16 ?        00:00:00 /usr/libexec/at-spi2-r
0 S yashi      5496     501  0  80   0 -  57527 do_sys 14:16 ?        00:00:00 /usr/libexec/dconf-ser
5 S root       6679       2  0  80   0 -    769 -      15:37 ?        00:00:00 /init
5 S root       6680    6679  0  80   0 -    769 -      15:37 ?        00:00:00 /init
4 S yashi      6685    6680  0  80   0 -   1518 do_wai 15:37 pts/4    00:00:00 -bash
5 S root       6701       2  0  80   0 -    769 -      15:38 ?        00:00:00 /init
5 S root       6702    6701  0  80   0 -    769 -      15:38 ?        00:00:00 /init
4 S yashi      6704    6702  0  80   0 -   1518 do_wai 15:38 pts/3    00:00:00 -bash
0 S yashi      6770    6685  0  80   0 -    671 hrtime 15:39 pts/4    00:00:00 ./output
1 Z yashi      6771    6770  0  80   0 -      0 -      15:39 pts/4    00:00:00 [output] <defunct>
0 R yashi      6772    6704  0  80   0 -   2083 -      15:39 pts/3    00:00:00 ps -elf
yashi@YASHISVICTUS:~$
```

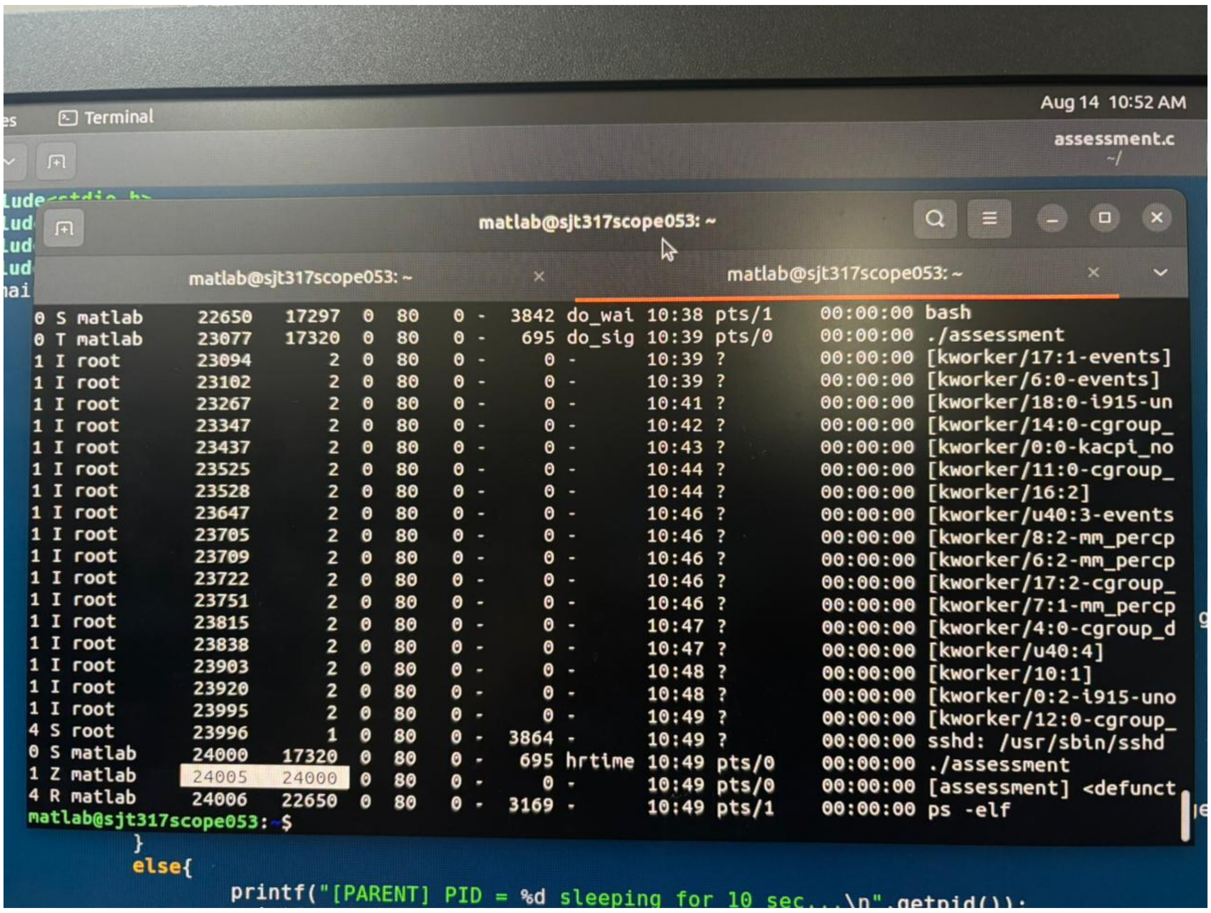
(In the lab)



```
1. Orphan Process
2. Zombie Process


^Z
[1]+  Stopped                 ./assessment
matlab@sjt317scope053:~$ ./assessment
1. Orphan Process
2. Zombie Process
1
[PARENT] PID =23195 exiting now...
[CHILD] PID = 23261, Initial Parent PID= 23195
matlab@sjt317scope053:~$ [CHILD] PID = 23261, New Parent PID= 1953 (adopted by init/systemd)
2
2: command not found
matlab@sjt317scope053:~$ ./assessment
1. Orphan Process
2. Zombie Process
2
[PARENT] PID = 24000 sleeping for 10 sec...
[PARENT] During this time, child will be zombie (<defunct> in ps -elf)
[CHILD] (will become zombie) PID = 24005, Initial Parent PID= 24000
Child exiting..
matlab@sjt317scope053:~$ 
        }
        else{
```



```
1 I root      21633      2  0  80   0 -     0 -      10:28 ?      00:00:00 [kworker/9:3-
1 I root      21834      2  0  80   0 -     0 -      10:30 ?      00:00:00 [kworker/7:1-
1 I root      21912      2  0  80   0 -     0 -      10:31 ?      00:00:00 [kworker/19:2
1 I root      21938      2  0  80   0 -     0 -      10:31 ?      00:00:00 [kworker/10:1
1 I root      21998      2  0  80   0 -     0 -      10:32 ?      00:00:00 [kworker/2:1-
1 I root      22002      2  0  80   0 -     0 -      10:32 ?      00:00:00 [kworker/0:1-
1 I root      22014      2  0  80   0 -     0 -      10:32 ?      00:00:00 [kworker/16:0
1 I root      22030      2  0  80   0 -     0 -      10:32 ?      00:00:00 [kworker/6:2-
1 I root      22122      2  0  80   0 -     0 -      10:34 ?      00:00:00 [kworker/9:0-
1 I root      22183      2  0  80   0 -     0 -      10:34 ?      00:00:00 [kworker/12:2
1 I root      22269      2  0  80   0 -     0 -      10:35 ?      00:00:00 [kworker/14:1
1 I root      22289      2  0  80   0 -     0 -      10:35 ?      00:00:00 [kworker/18:1
1 I root      22365      2  0  80   0 -     0 -      10:36 ?      00:00:00 [kworker/7:0-
1 I root      22467      2  0  80   0 -     0 -      10:37 ?      00:00:00 [kworker/10:2
1 I root      22498      2  0  80   0 -     0 -      10:37 ?      00:00:00 [kworker/0:2-
1 I root      22638      2  0  80   0 -     0 -      10:38 ?      00:00:00 [kworker/8:0-
0 S matlab    22650  17297  0  80   0 -  3842 do_wai 10:38 pts/1  00:00:00 bash
0 T matlab    23077  17320  0  80   0 -   695 do_sig 10:39 pts/0  00:00:00 ./assessment
1 I root      23094      2  0  80   0 -     0 -      10:39 ?      00:00:00 [kworker/17:1
1 I root      23102      2  0  80   0 -     0 -      10:39 ?      00:00:00 [kworker/6:0-
4 S root      23251      1  0  80   0 -  3864 -      10:41 ?      00:00:00 sshd: /usr/sb
1 S matlab    23261   1953  0  80   0 -   695 hrtime 10:41 pts/0  00:00:00 ./assessment
4 R matlab    23265  22650  0  80   0 -  3169 -      10:41 pts/1  00:00:00 ps -elf
matlab@sjt317scope053:~$ ps -elf
        }
        else{
            printf("[PARENT] PID = %d sleeping for 10 sec...\n",getpid());
```



```
0 S matlab    22650  17297  0  80   0 -  3842 do_wai 10:38 pts/1     00:00:00 bash
0 T matlab    23077  17320  0  80   0 -   695 do_sig 10:39 pts/0     00:00:00 ./assessment
1 I root      23094      2  0  80   0 -     0 -      10:39 ?         00:00:00 [kworker/17:1-events]
1 I root      23102      2  0  80   0 -     0 -      10:39 ?         00:00:00 [kworker/6:0-events]
1 I root      23267      2  0  80   0 -     0 -      10:41 ?         00:00:00 [kworker/18:0-i915-un
1 I root      23347      2  0  80   0 -     0 -      10:42 ?         00:00:00 [kworker/14:0-cgroup_
1 I root      23437      2  0  80   0 -     0 -      10:43 ?         00:00:00 [kworker/0:0-kacpi_no
1 I root      23525      2  0  80   0 -     0 -      10:44 ?         00:00:00 [kworker/11:0-cgroup_
1 I root      23528      2  0  80   0 -     0 -      10:44 ?         00:00:00 [kworker/16:2]
1 I root      23647      2  0  80   0 -     0 -      10:46 ?         00:00:00 [kworker/u40:3-events
1 I root      23705      2  0  80   0 -     0 -      10:46 ?         00:00:00 [kworker/8:2-mm_percp
1 I root      23709      2  0  80   0 -     0 -      10:46 ?         00:00:00 [kworker/6:2-mm_percp
1 I root      23722      2  0  80   0 -     0 -      10:46 ?         00:00:00 [kworker/17:2-cgroup_
1 I root      23751      2  0  80   0 -     0 -      10:46 ?         00:00:00 [kworker/7:1-mm_percp
1 I root      23815      2  0  80   0 -     0 -      10:47 ?         00:00:00 [kworker/4:0-cgroup_d
1 I root      23838      2  0  80   0 -     0 -      10:47 ?         00:00:00 [kworker/u40:4]
1 I root      23903      2  0  80   0 -     0 -      10:48 ?         00:00:00 [kworker/10:1]
1 I root      23920      2  0  80   0 -     0 -      10:48 ?         00:00:00 [kworker/0:2-i915-uno
1 I root      23995      2  0  80   0 -     0 -      10:49 ?         00:00:00 [kworker/12:0-cgroup_
4 S root      23996      1  0  80   0 -  3864 -      10:49 ?         00:00:00 sshd: /usr/sbin/sshd
0 S matlab    24000  17320  0  80   0 -   695 hrtime 10:49 pts/0     00:00:00 ./assessment
1 Z matlab    24005  24000  0  80   0 -     0 -      10:49 pts/0     00:00:00 [assessment] <defunct
4 R matlab    24006  22650  0  80   0 -  3169 -      10:49 pts/1     00:00:00 ps -elf
matlab@sjt317scope053:~$
        }
        else{
                printf("[PARENT] PID = %d sleeping for 10 sec...\n",getpid());
```