**Student Name:** Saiteja.m

**Student ID:**11608248

**Email Address:**Saiteja888466@gmail.com

**GitHub Link:**

**Question1:** Write a C program using the fork() system call that generates this sequence in the child process. The starting number will be provided from the user. For example, if 8 is passed as a parameter on the command line, the child process will output 8, 4, 2, 1. Because the parent and child processes have their own copies of the data, it will be necessary for the child to output the sequence. Have the parent invoke the wait() call to wait for the child process to complete before exiting the program. Perform necessary error checking to ensure that a positive integer is passed on the command line.

**Code:**

#include<stdio.h>

#include<sys/types.h>

int main(int argc, char \*argv[]) {

if( argc > 2 ) {

printf("Too many arguments supplied.\n");

exit(1);

}

int n = atoi(argv[1]);

pid\_t id;

id = fork();

if(id==0){//child process

printf("the sequence is:\n");

while(1){

if(n==1){

printf("%d ",n);

break;

}

if(n%2==0){

printf("%d ",n);

n = n/2;

}

else{

printf("%d ",n);

n = 3\*n+1;

}

}

}

else if(id >0){//parent process

wait();

}

else{

printf("error in creating process");

}

printf("\n");

return 0;

}

**Description:**

We should generate sequence 8,4,2,1 by the child process using fork()(system call) the starting is taken by the user if user passed 8 then the sequence should be 8,4,,2,1 because parent and child are having 2 different copies while child is processing the sequence then parent should be in waiting.

**Algorithm:**

Sequence(int argc , Int \*argv)

if(input>2)

print(“two many arguments supplied”);

if(id==0)(creation of child process)

print(“the sequence is:”);

while true

if(n==1)

print("%d ",n);

if(n%2==0){

print("%d ",n);

n = n/2;

else{

printf("%d ",n);

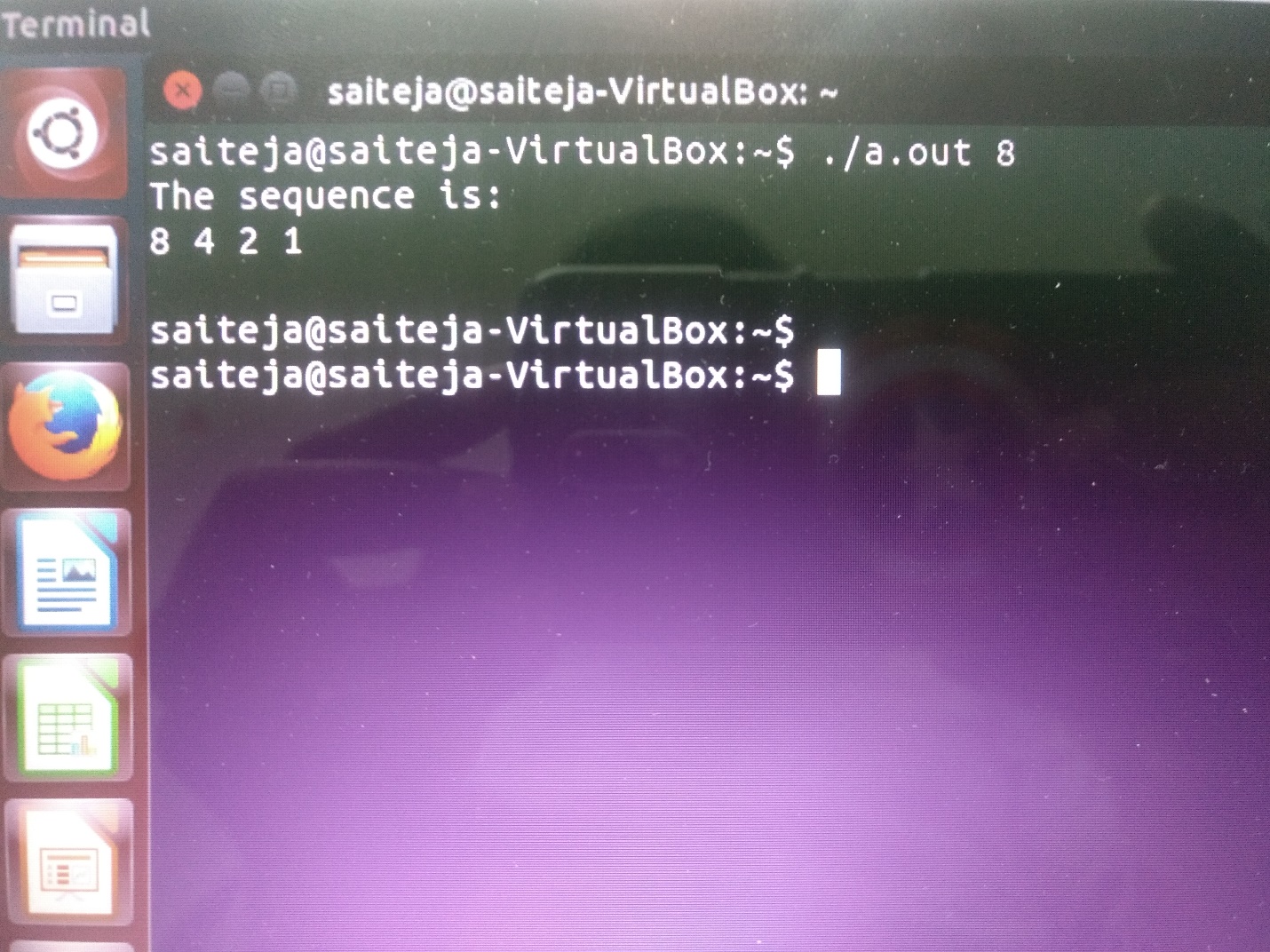
n = 3\*n+1;

elseif(id>0)(parent process)

wait();

else

print(“error in creating process”);

**Test cases: **

**Question2:** Design a file-copying program named filecopy using ordinary pipes. This program will be passed two parameters: the name of the file to be copied and the name of the copied file. The program will then create an ordinary pipe and write the contents of the file to be copied to the pipe. The child process will read this file from the pipe and write it to the destination file. For example, if we invoke the program as follows: filecopy input.txt copy.txt The file input.txt will be written to the pipe. The child process will read the contents of this file and write it to the destination file copy.txt.

Code:

#include<stdio.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<string.h>

#include<stdlib.h>

int main( int argc, char\* argv[] ) {

int fdone[2];

pid\_t childid;

char readBuff[50];

char writeBuff[50];

int readCounter;

pipe( fdone );

if( argc < 3 ) {

printf( "Atleast need 2 params " );

exit(1);

}

int fileOpen = open( argv[1], 0 );

int targetFile = open( argv[2], 0666 );

if ( fileOpen == -1 || targetFile == -1 ) {

printf( "Opening file failed " );

exit(1);

}

childid = fork();

sa

if( childid == 0 ) {

close( fdone[1] );

read( fdone[0], readBuff, sizeof( readBuff ) );

printf( "The recived string is : %s", readBuff );

write( targetFile, readBuff, strlen( readBuff ) + 1 );

} else {

close( fdone[0] );

while( (readCounter = read( fileOpen, readBuff, sizeof( readBuff ) ) > 0 ) ) {

write( fdone[1], readBuff, sizeof( readBuff ) );

}

close( fdone[1] );

}

}

**Description:**

Copying of one file to another file by using ordinary pipes and file names are given by the user on the runtime so the data of one file copy to the other file by using the pipes.

**Algorithm:**

**Filecopying algorithm**

if( argc < 3 )

printf( "Atleast need 2 params " );

if ( fileOpen == -1 || targetFile == -1 )

printf( "Opening file failed " );

if( childid == 0 )

close( fdone[1] );

read( fdone[0], readBuff, sizeof( readBuff ) );

printf( "The recived string is : %s", readBuff );

write( targetFile, readBuff, strlen( readBuff ) + 1 );

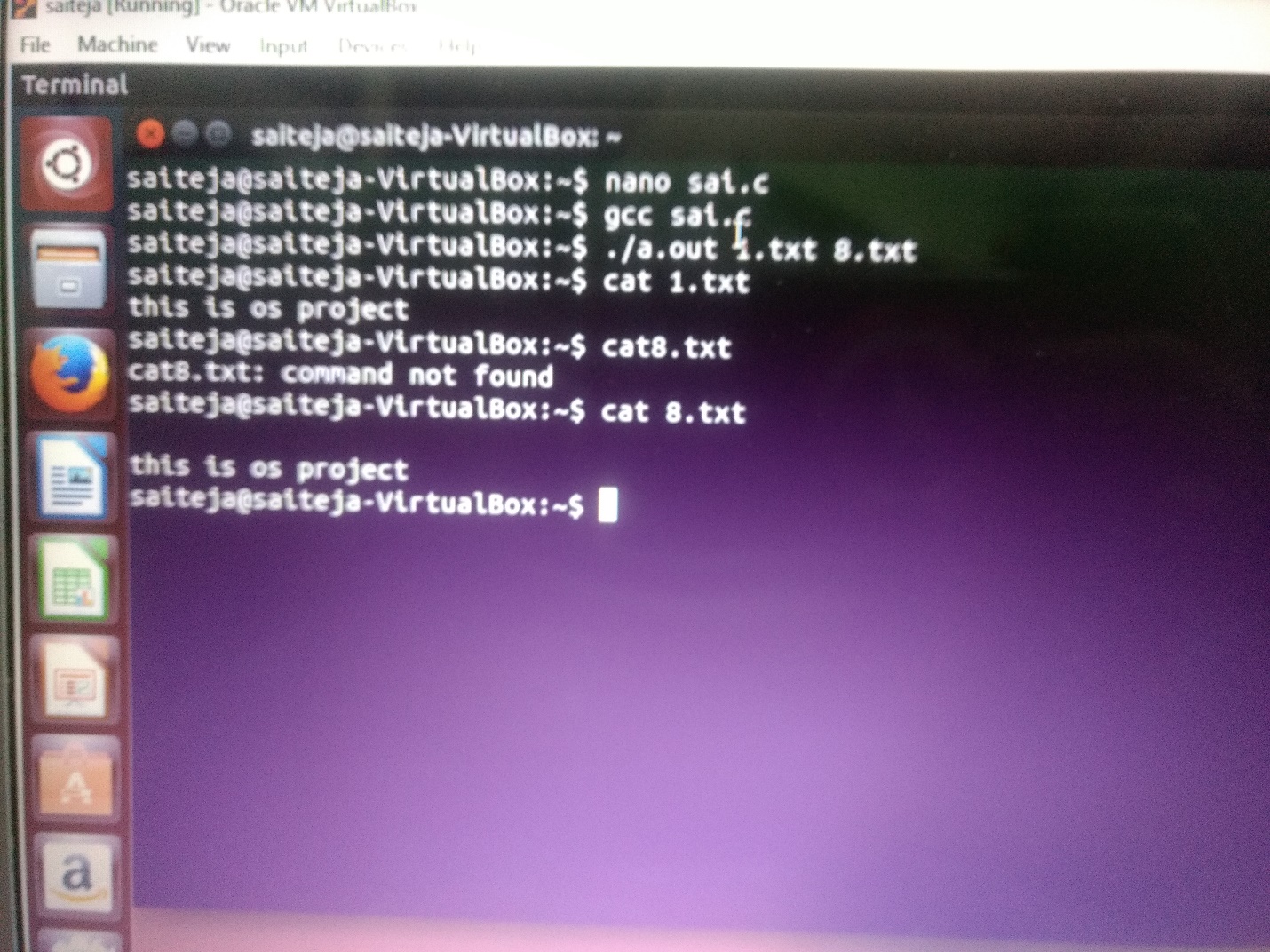
else

close( fdone[0] );

while( (readCounter = read( fileOpen, readBuff, sizeof( readBuff ) ) > 0 ) )

write( fdone[1], readBuff, sizeof( readBuff ) );

**Test cases:**

****