**Layout of Program, Library Function Calls and error handling**

**Lab no# 02**

****

**Fall 2021**

**CSE-302 System Programming Lab**

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr: Ma’am Madeha sheer**

**November** 16, 2021

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

## **Task 1: Write two C programs for the analysis of the difference in executable file sizes while using:**

## **Uninitialized Static Variables**

## **Initialized Static Variables**

Use ls -l to compare the sizes of the executable modules for the above two C programs.

**Source Code:**

**Initialized Static Variable:**

#include <stdio.h>

int main()

{

printf("Intialized Task\n");

static int array[100]={2,3,4,5,6}; //we can also used single integer instead of array.

}

**Uninitialized Static Variable:**

#include <stdio.h>

int main()

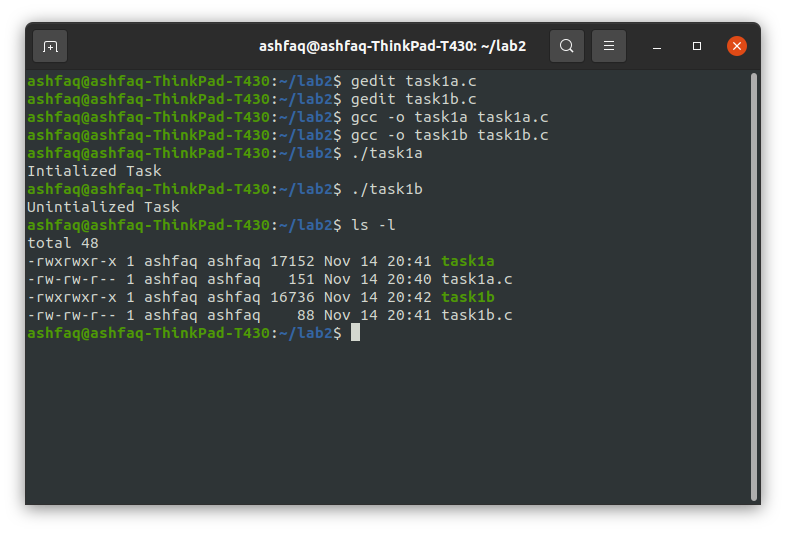
{

printf("Unintialized Task\n");

static int array[100];

}

**Output:**

****

## **Task 2: Analyze the return values and error numbers/error stings of *wait* system call on:**

## **Success**

## **Failure**

**Using:**

1. **Strerror**
2. **Perror**
3. **Success**

**Using perror:**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <sys/wait.h>

int main()

{

int x=fork(); //for child creation

if (x==-1) //child not created

{

printf("Return value of fork: %d\n",x);

perror("Error occured:");

printf("Errno value: %d\n",errno);

}

else

if (x==0) //child created

{

exit(0);

}

if (x>0) //child process executing

{

int y=wait(NULL); //parent wait for child process termination

if(y==-1) //no child process

{

printf("Return value of wait: %d\n",x);

perror("Error occured:");

printf("Errno value: %d\n",errno);

}

else //child process succesfully terminated

{

printf("Return value of wait: %d\n",x);

perror("Operation Successful:");

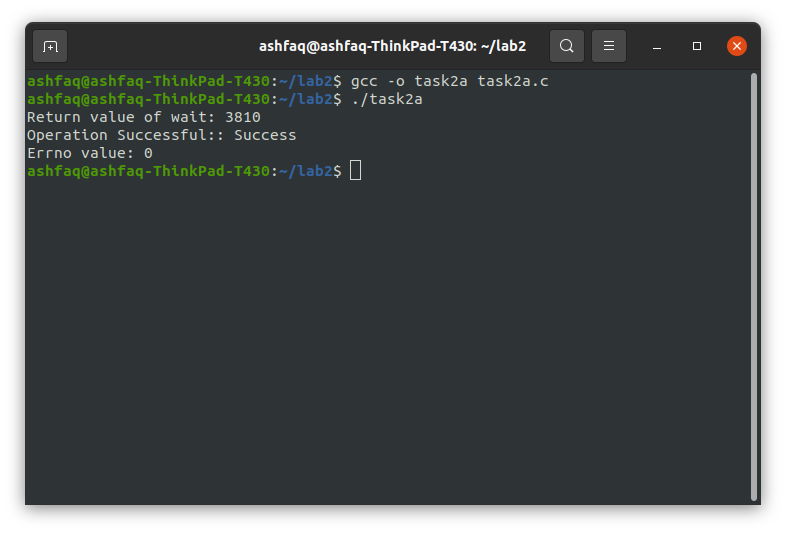
printf("Errno value: %d\n",errno);

}

}

}

**Output:**

****

**Using Strerror:**

**Source code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <sys/wait.h>

#include <string.h>

int main()

{

int x=fork(); //for child creation

if (x==-1) //child not created

{

printf("Return value of fork: %d\n",x);

printf("Error occured:%s\n",strerror(errno)); //strerror display error in sting form assigned to errno value

printf("Errno value: %d\n",errno);

}

else

if (x==0) //child created

{

exit(0);

}

if (x>0) //child process executing

{

int y=wait(NULL); //parent wait for child process termination

if(y==-1) //no child process

{

printf("Return value of wait: %d\n",x);

printf("Error occured:%s\n",strerror(errno));

printf("Errno value: %d\n",errno);

}

else //child process succesfully terminated

{

printf("Return value of wait: %d\n",x);

perror("Operation Successful:");

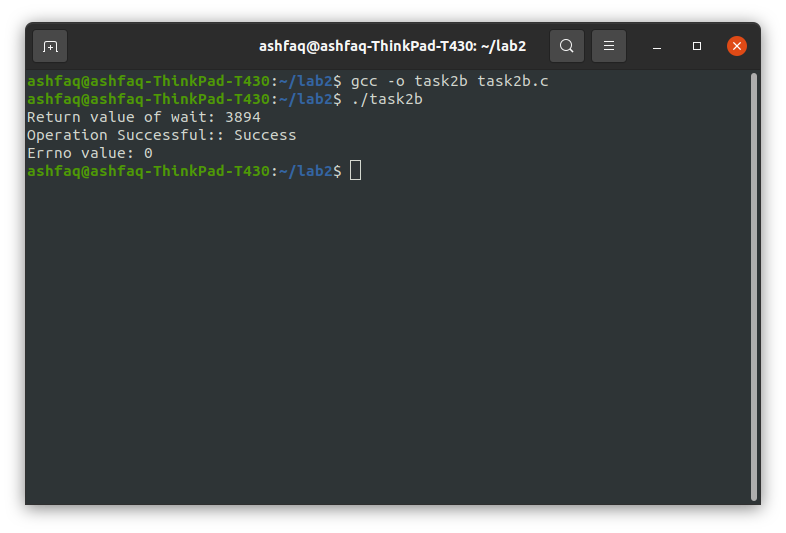
printf("Errno value: %d\n",errno);

}

}

}

**Output:**

****

1. **Failure:**

**Using Perror:**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <sys/wait.h>

int main()

{ //no fork function for child creation

int y=wait(NULL); //parent wait for child process termination

printf("Return value of wait: %d\n",y);

if(y==-1) //no child process

{

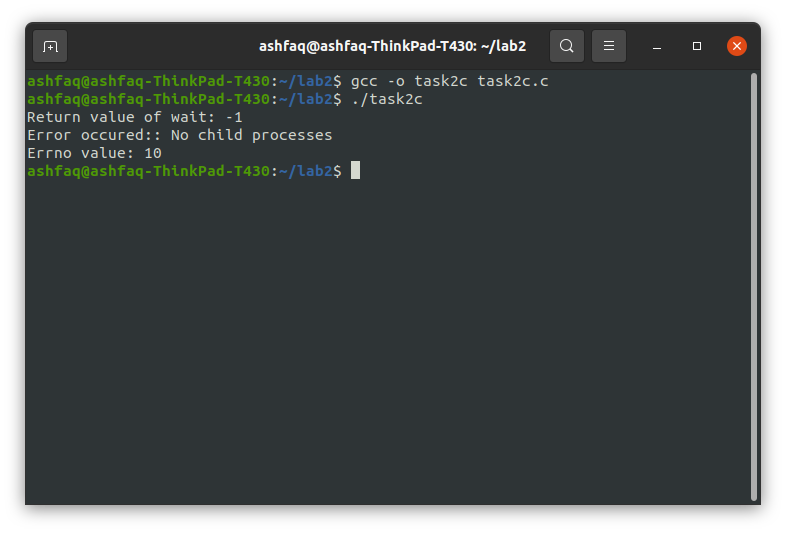
perror("Error occured:"); //display error in string form

printf("Errno value: %d\n",errno); //display errno value

}

}

**Output:**

****

**Using Strerror:**

**Source code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <sys/wait.h>

#include <string.h>

int main()

{ //no fork function for child creation

int y=wait(NULL); //parent wait for child process termination

printf("Return value of wait: %d\n",y);

if(y==-1) //no child process

{

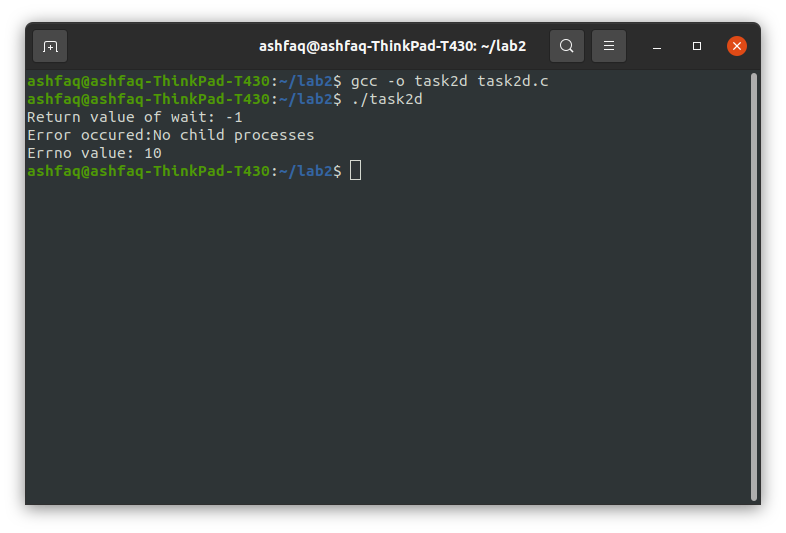
printf("Error occured:%s\n",strerror(errno)); //display error in string form

printf("Errno value: %d\n",errno); //display errno value

}

}

**Output:**

****

## **Task 3: Analyze the return values and error numbers/error stings of *close* system call on:**

## **Success**

## **Failure**

**Using:**

1. **Strerror**
2. **Perror**
3. **Success:**

**Using Strerror:**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

int main()

{

int x=close(0); //in 0 to 3 case defaults files will be closed. so file will be successful close.

printf("Function Returned value: %d\n",x);

if (x==-1)

{

printf("Error Occured: %s\n",strerror(errno));

printf("Errno Return Value: %d\n", errno);

}

else

{

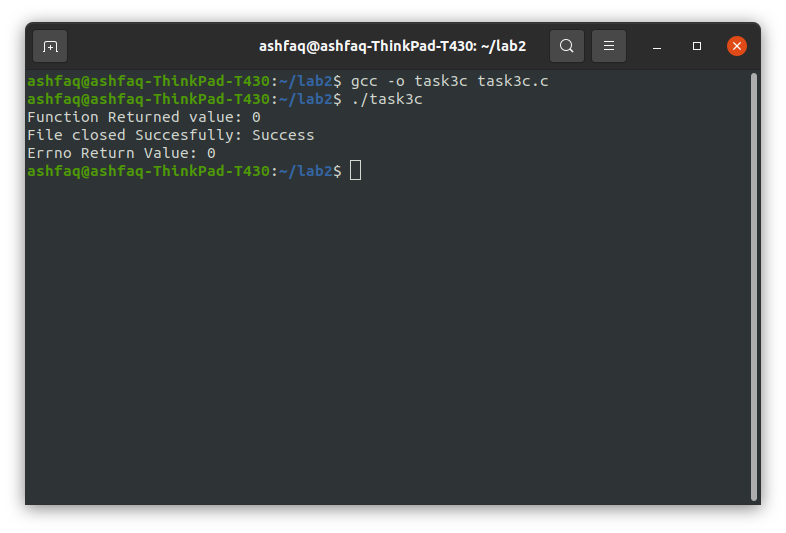
printf("File closed Succesfully: %s\n",strerror(errno));

printf("Errno Return Value: %d\n", errno);

}

}

**Output:**

****

**Using Perror:**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

int main()

{

int x=close(2); //if we don't have any opened file and we pass 0 or 1 or 2 it will not give error b/c by default in system 3 files opened

// stdin(keyboard) stdout(moniter) stderror(error file).

printf("the Return value of close function: %d\n",x);

if (x==-1)

{

perror("Error Occured:");

printf("Errno value: %d\n",errno);

}

else

{

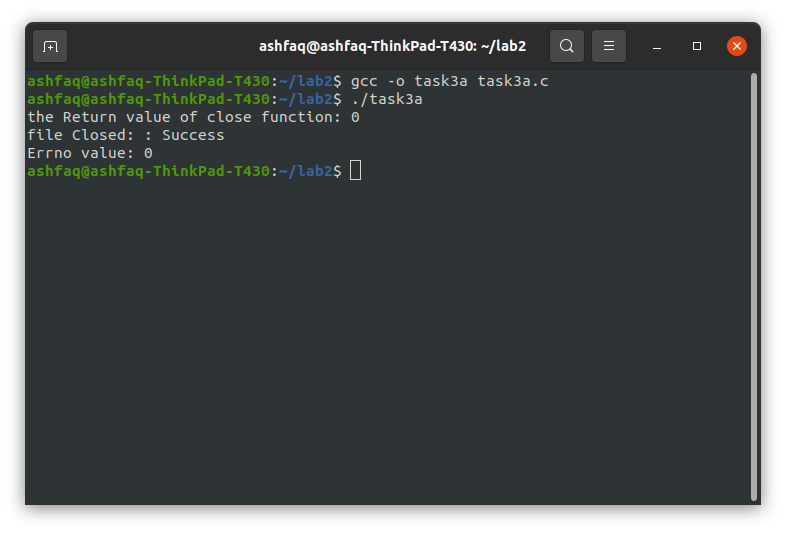
perror("file Closed: ");

printf("Errno value: %d\n",errno);

}

}

**Output:**

****

1. **Failure:**

**Using Strerror:**

**Source code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

int main()

{

int x=close(8); //there is no 8th file opened. error will occur.

printf("Function Returned value: %d\n",x);

if (x==-1)

{

printf("Error Occured: %s\n",strerror(errno));

printf("Errno Return Value: %d\n", errno);

}

else

{

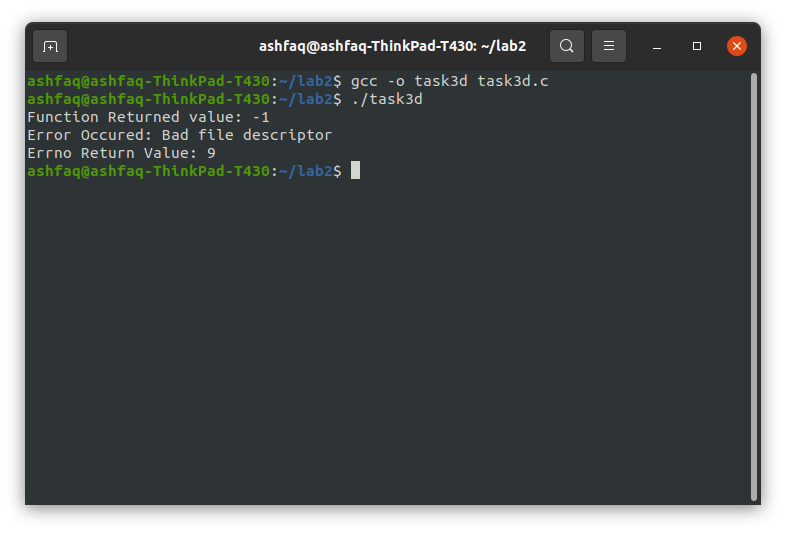
printf("File closed Succesfully: %s\n",strerror(errno));

printf("Errno Return Value: %d\n", errno);

}

}

**Output:**

****

**Using perror:**

**Source code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

int main()

{

int x=close(6); //there is no sixth file opened it will give us error.

printf("Function Returned Value: %d\n", x);

if (x==-1)

{

perror("Error Ocurred:");

printf("Errno value: %d\n",errno);

}

else

{

perror("File closed successfully:");

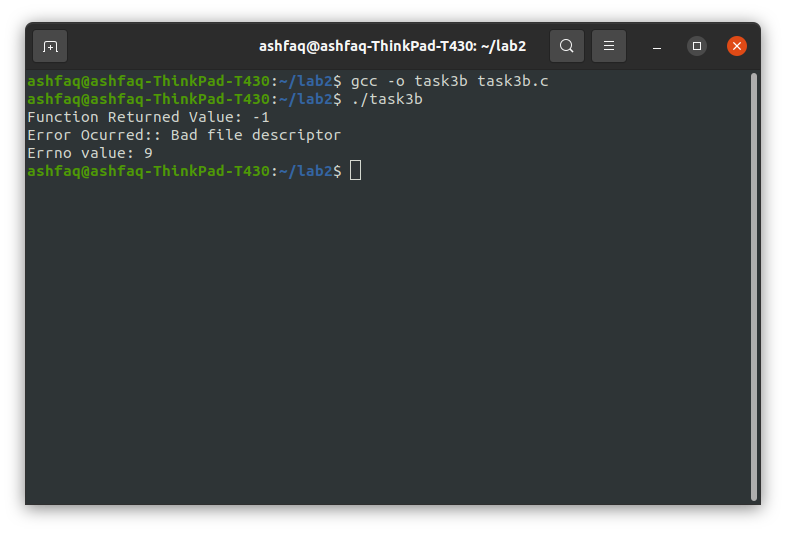
printf("Errno value: %d\n",errno);

}

return 0;

}

**Output:**



**Task 4: Print all Error messages.**

**Source code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

int main()

{

for(int i=0; i<134; i++)

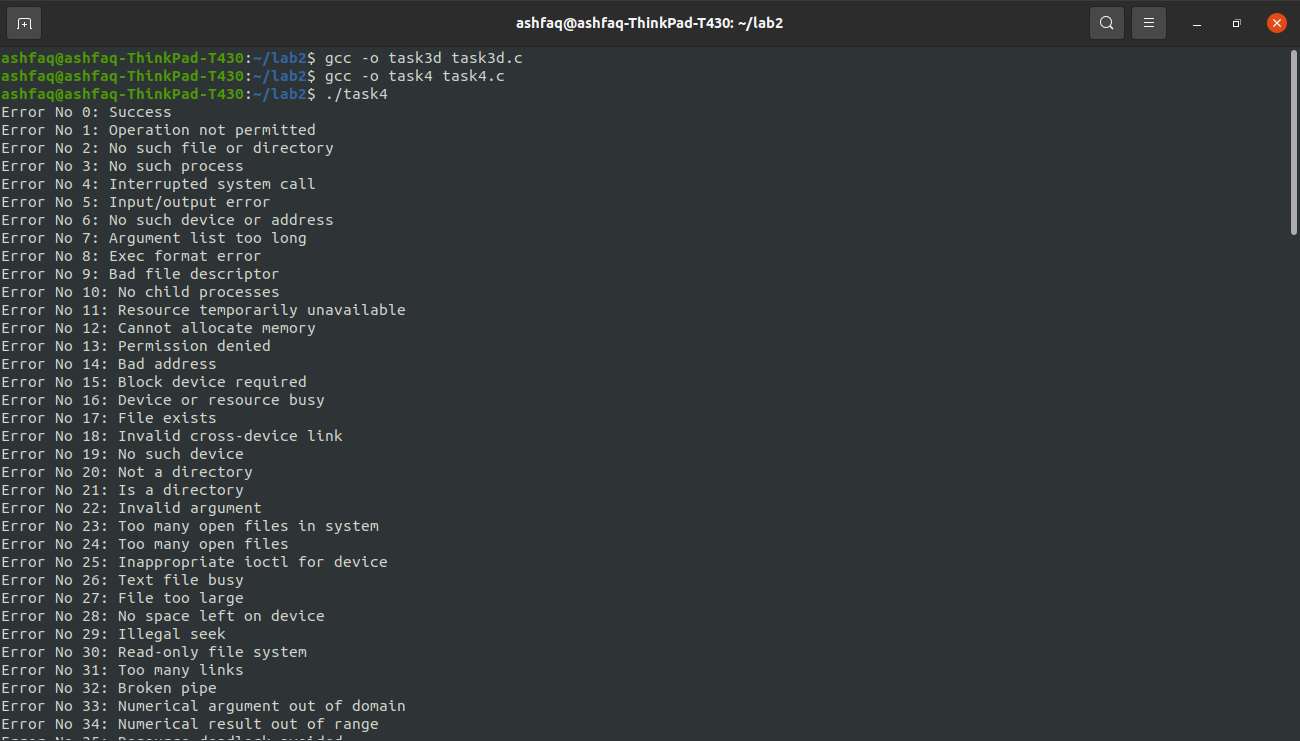
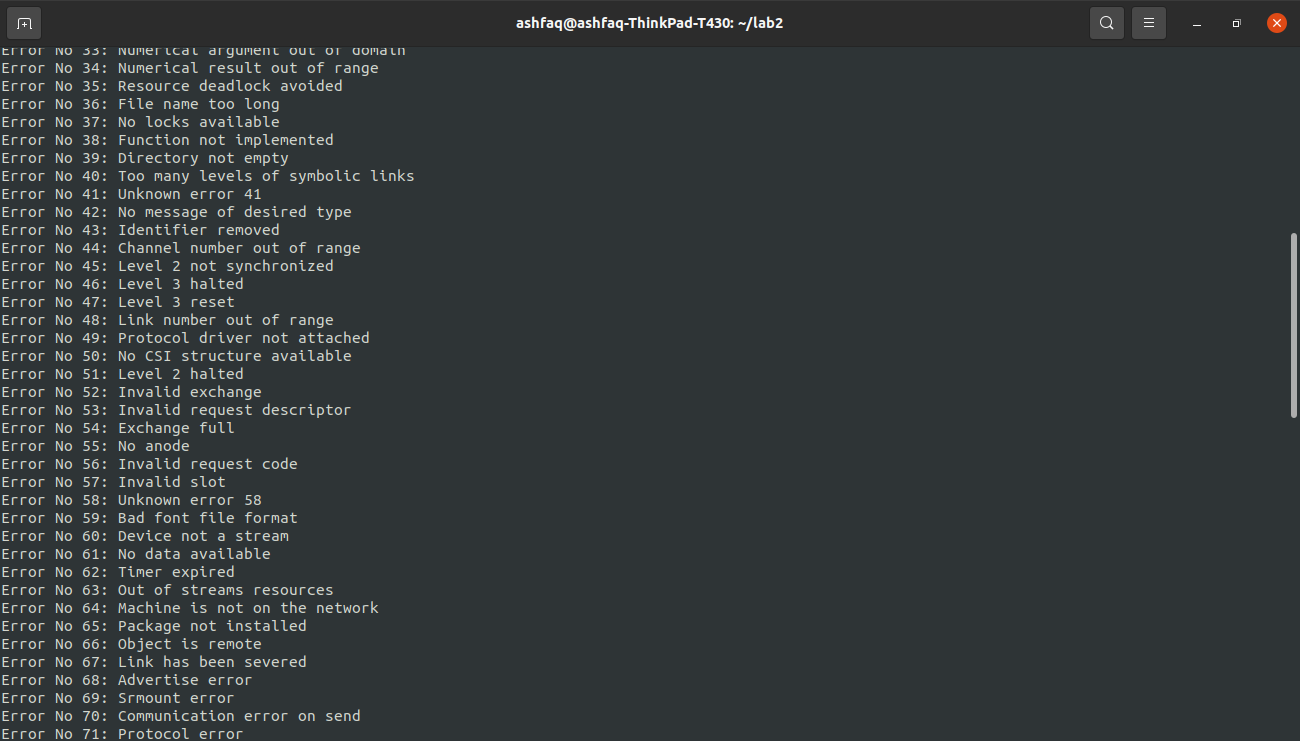
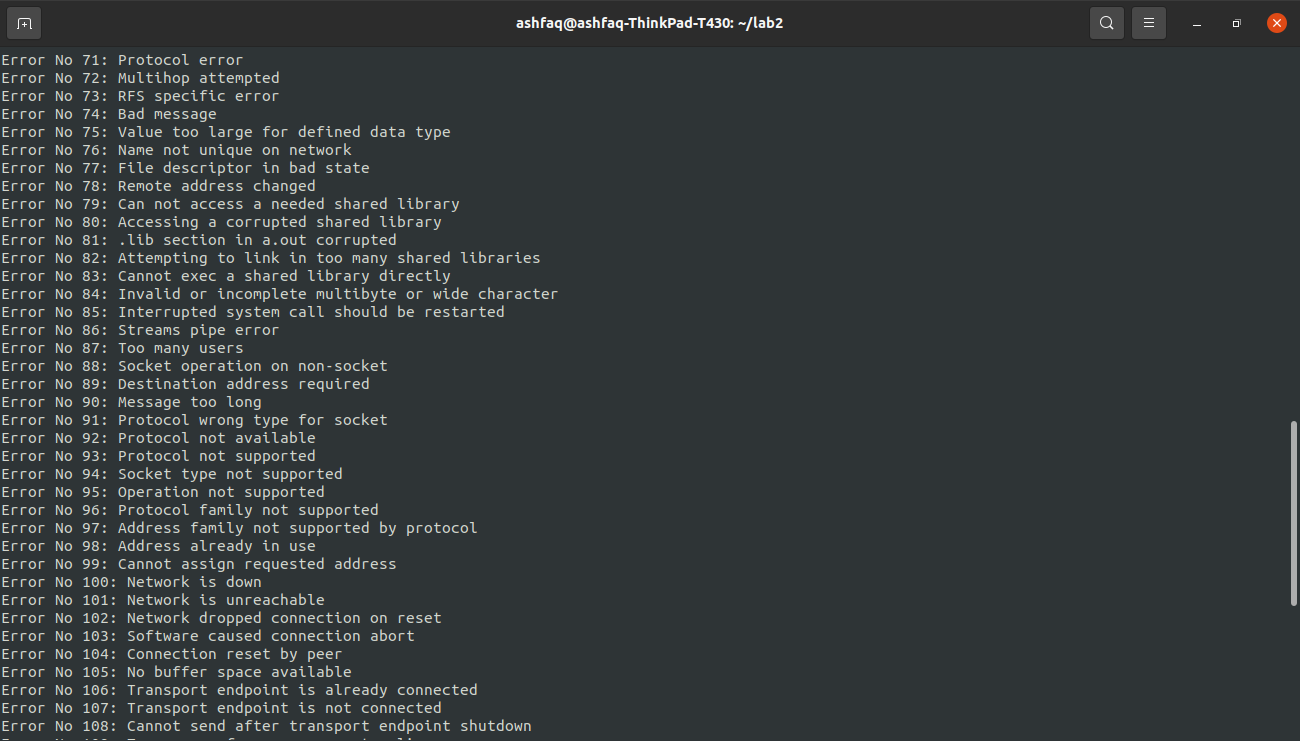
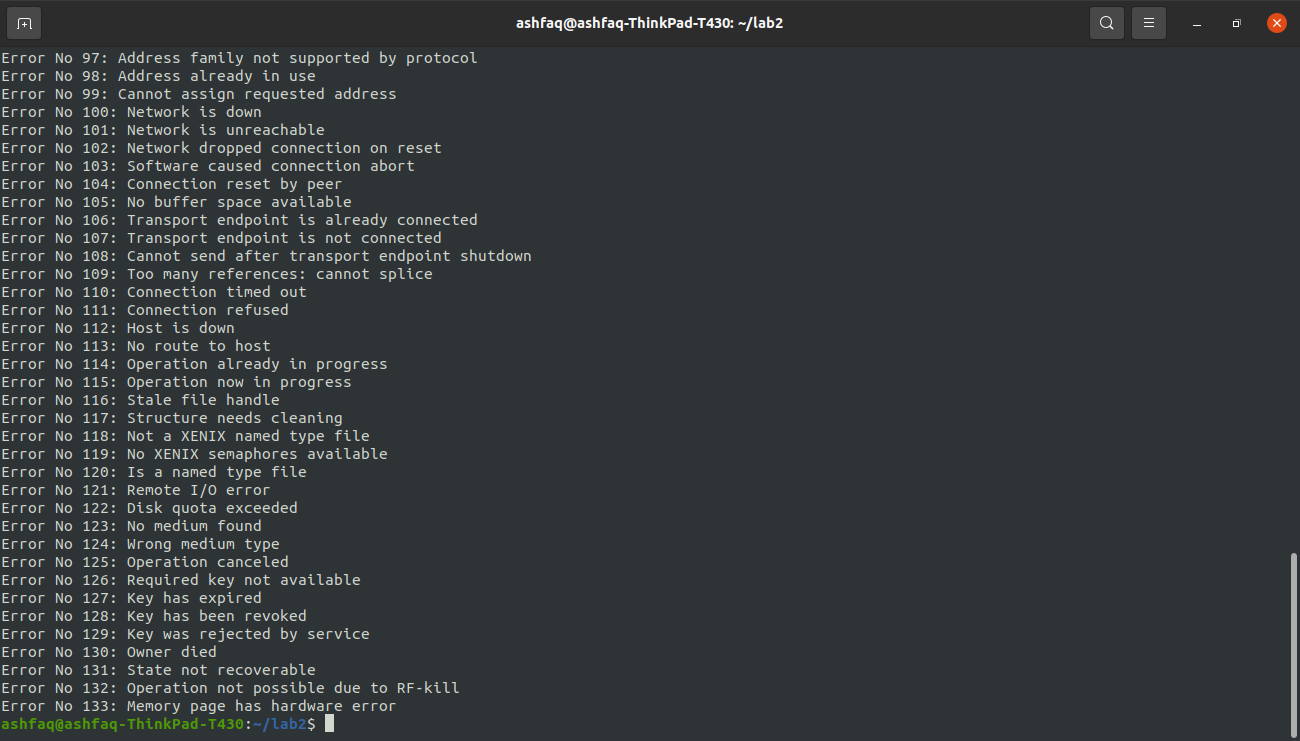
{

printf("Error No %d: %s\n",i,strerror(i));

}

}

**Output:**

**** ****  

The End