Layout of Program, Library Function Calls and error handling

LAB # 02

Fall 2020

CSE302L System Programming Lab

Submitted by: **Shah Raza**

Registration No. : **18PWCSE1658**

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. Madiha Sher

Saturday, December 5th, 2020

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Lab Objective(s):

- Understand and implement Error Handling.
- Understand the difference between Initialized and Uninitialized static variables.

Task # 01:

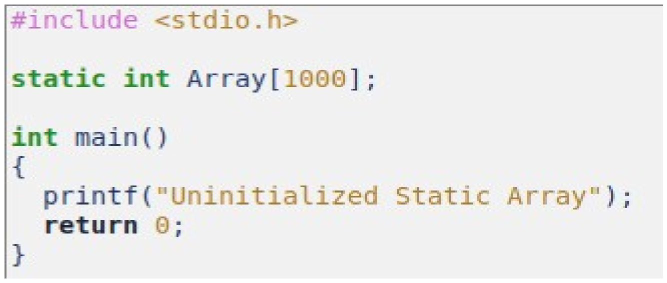
Write two C programs for the analysis of the difference in executable file sizes while using:

1. **Uninitialized Static Variables**
2. **Initialized Static Variables**

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

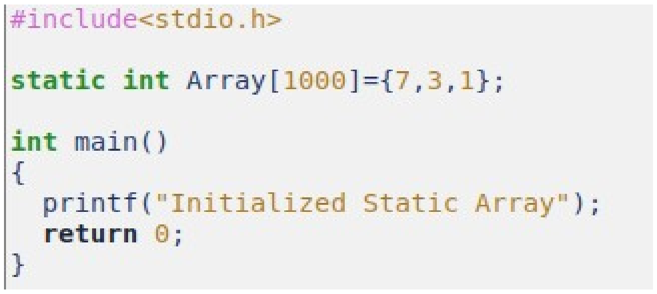
Code:

Uninitialized Static Variables:



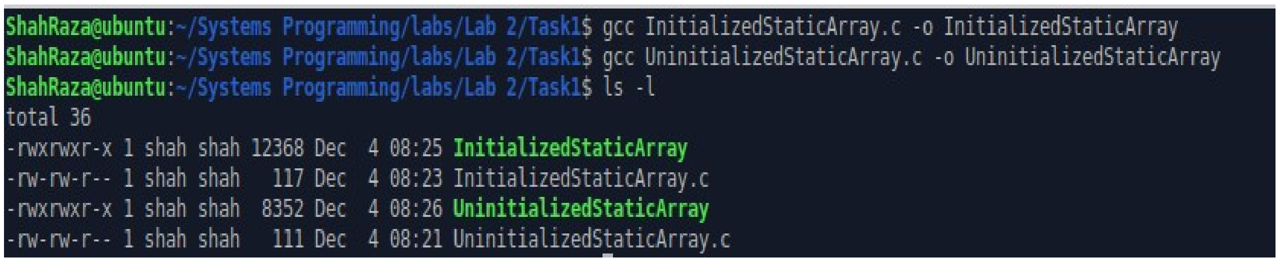
```c
#include <stdio.h>

static int Array[1000];

int main()
{
  printf("Uninitialized Static Array");
  return 0;
}
```

Initialized Static Variables:



```c
#include<stdio.h>

static int Array[1000]={7,3,1};

int main()
{
  printf("Initialized Static Array");
  return 0;
}
```

Output / Graphs / Plots / Results:



```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task1$ gcc InitializedStaticArray.c -o InitializedStaticArray
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task1$ gcc UninitializedStaticArray.c -o UninitializedStaticArray
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task1$ ls -l
total 36
-rwxrwxr-x 1 shah shah 12368 Dec  4 08:25 InitializedStaticArray
-rw-rw-r-- 1 shah shah   117 Dec  4 08:23 InitializedStaticArray.c
-rwxrwxr-x 1 shah shah  8352 Dec  4 08:26 UninitializedStaticArray
-rw-rw-r-- 1 shah shah   111 Dec  4 08:21 UninitializedStaticArray.c
```

Task # 02:

Analyze the return values and error numbers/error stings of wait system call on:

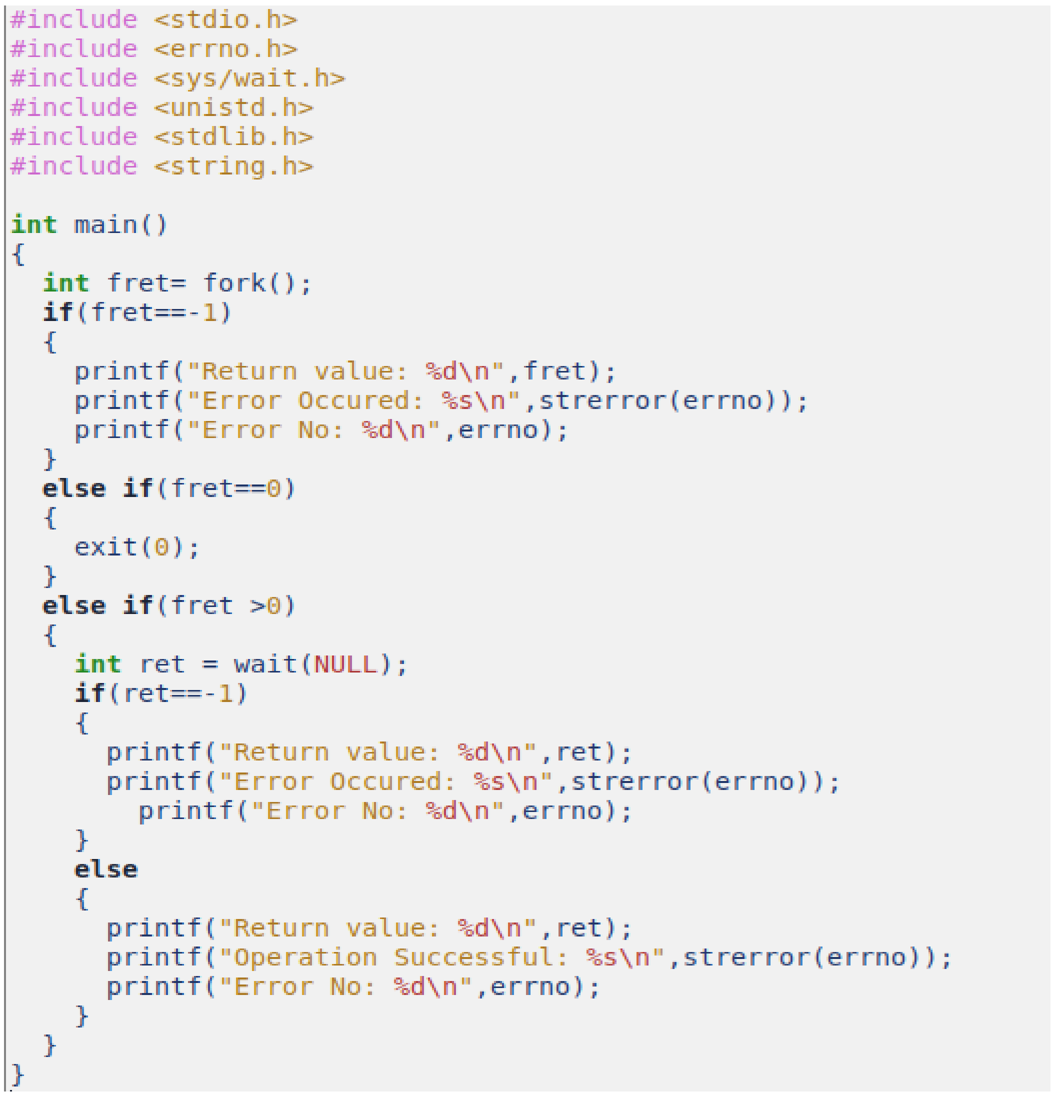
1. **Success**
2. **Failure**

Using:

1. **Strerror**
2. **Perror**

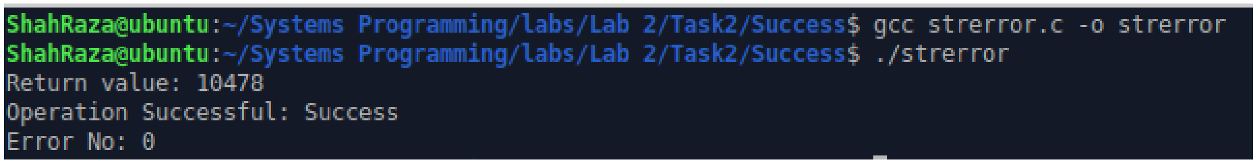
Success:

Strerror:



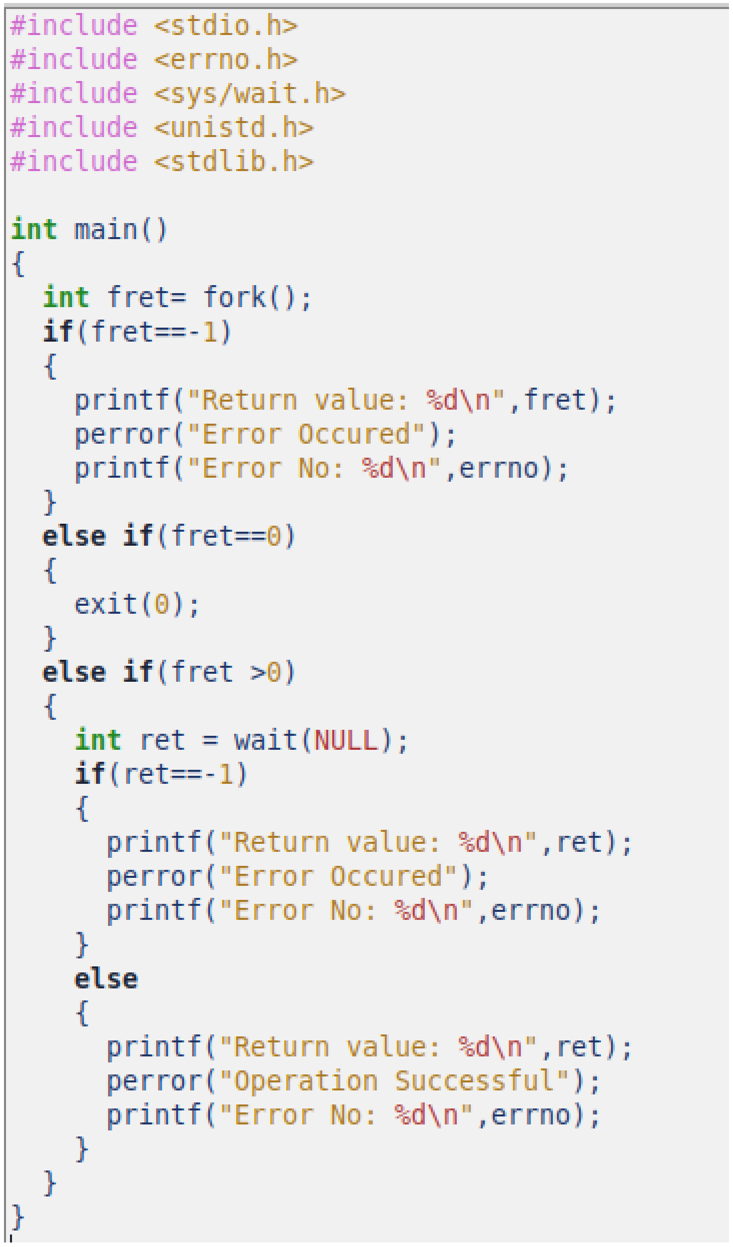
```
#include <stdio.h>
#include <errno.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main()
{
  int fret= fork();
  if(fret==-1)
  {
    printf("Return value: %d\n",fret);
    printf("Error Occured: %s\n",strerror(errno));
    printf("Error No: %d\n",errno);
  }
  else if(fret==0)
  {
    exit(0);
  }
  else if(fret >0)
  {
    int ret = wait(NULL);
    if(ret==-1)
    {
      printf("Return value: %d\n",ret);
      printf("Error Occured: %s\n",strerror(errno));
        printf("Error No: %d\n",errno);
    }
    else
    {
      printf("Return value: %d\n",ret);
      printf("Operation Successful: %s\n",strerror(errno));
      printf("Error No: %d\n",errno);
    }
  }
}
```

Output:



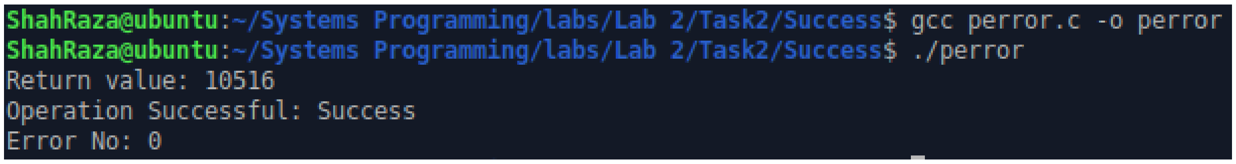
```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task2/Success$ gcc strerror.c -o strerror
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task2/Success$ ./strerror
Return value: 10478
Operation Successful: Success
Error No: 0
```

Perror:



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

int main()
{
  int fret= fork();
  if(fret==-1)
  {
    printf("Return value: %d\n",fret);
    perror("Error Occured");
    printf("Error No: %d\n",errno);
  }
  else if(fret==0)
  {
    exit(0);
  }
  else if(fret >0)
  {
    int ret = wait(NULL);
    if(ret==-1)
    {
      printf("Return value: %d\n",ret);
      perror("Error Occured");
      printf("Error No: %d\n",errno);
    }
    else
    {
      printf("Return value: %d\n",ret);
      perror("Operation Successful");
      printf("Error No: %d\n",errno);
    }
  }
}
```

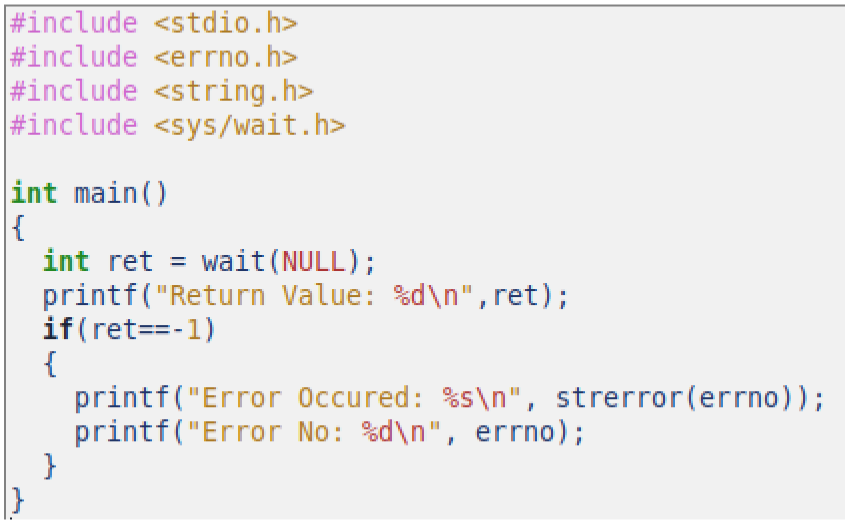
Output:



```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task2/Success$ gcc perror.c -o perror
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task2/Success$ ./perror
Return value: 10516
Operation Successful: Success
Error No: 0
```

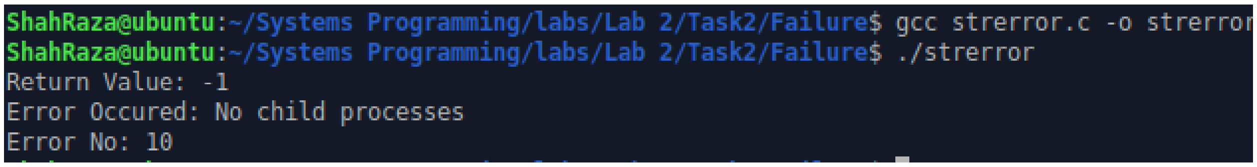
Failure:

Strerror:



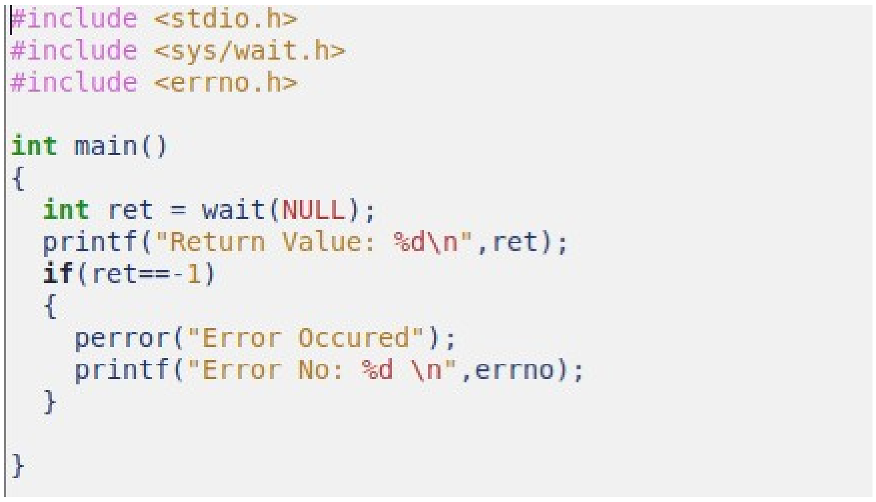
```c
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <sys/wait.h>

int main()
{
  int ret = wait(NULL);
  printf("Return Value: %d\n",ret);
  if(ret==-1)
  {
    printf("Error Occured: %s\n", strerror(errno));
    printf("Error No: %d\n", errno);
  }
}
```

Output:



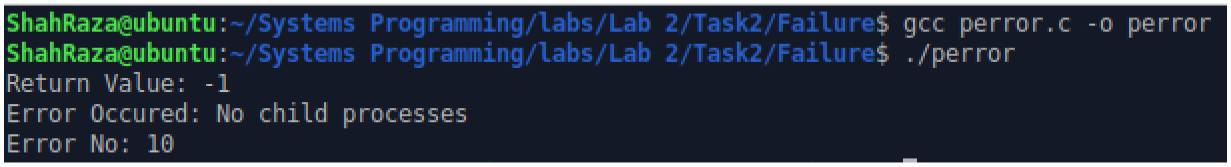
```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task2/Failure$ gcc strerror.c -o strerror
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task2/Failure$ ./strerror
Return Value: -1
Error Occured: No child processes
Error No: 10
```

Perror:



```c
#include <stdio.h>
#include <sys/wait.h>
#include <errno.h>

int main()
{
  int ret = wait(NULL);
  printf("Return Value: %d\n",ret);
  if(ret==-1)
  {
    perror("Error Occured");
    printf("Error No: %d \n",errno);
  }

}
```

Output:



```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task2/Failure$ gcc perror.c -o perror
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task2/Failure$ ./perror
Return Value: -1
Error Occured: No child processes
Error No: 10
```

Task 3:

Analyze the return values and error numbers/error stings of close system call on:

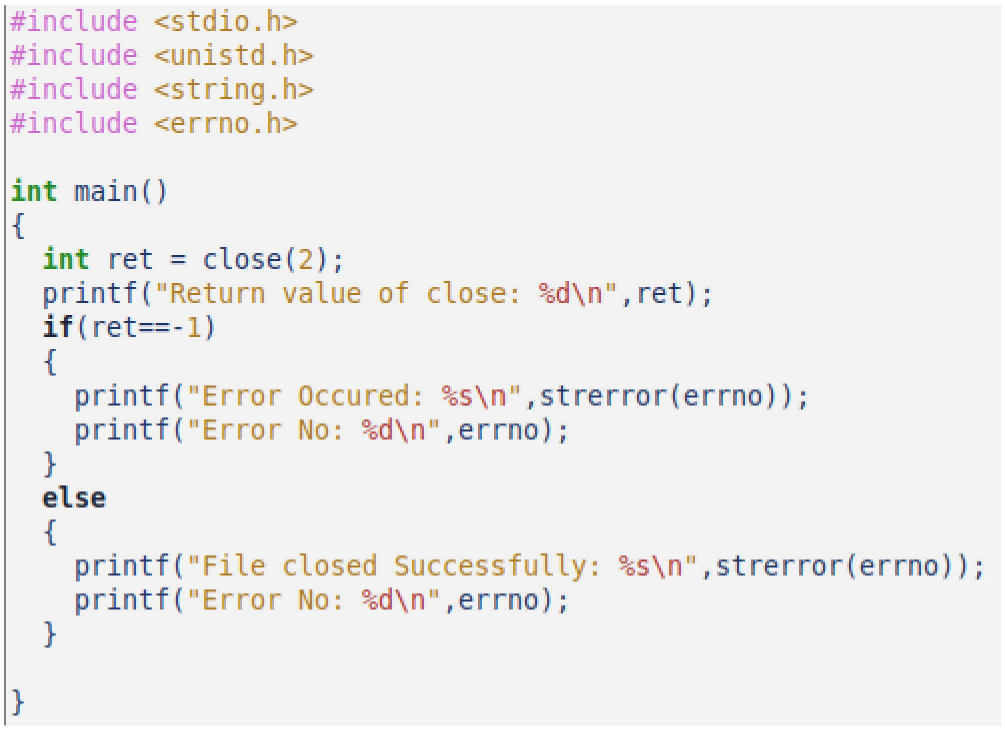
1. **Success**
2. **Failure**

Using:

1. **Strerror**
2. **Perror**

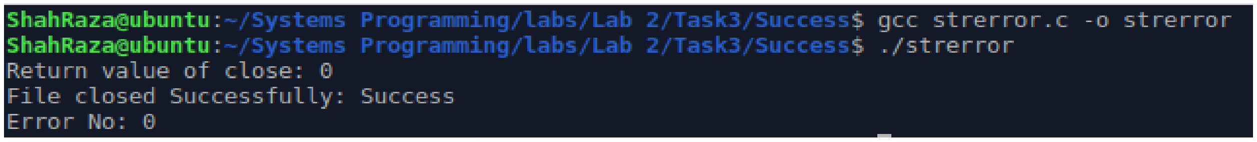
Success:

Strerror:



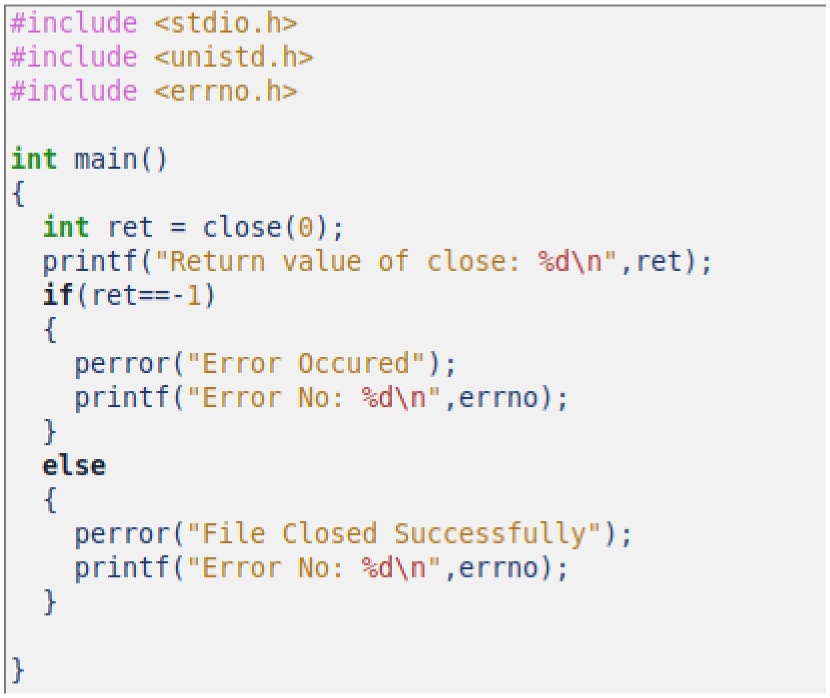
```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

int main()
{
  int ret = close(2);
  printf("Return value of close: %d\n",ret);
  if(ret==-1)
  {
    printf("Error Occured: %s\n",strerror(errno));
    printf("Error No: %d\n",errno);
  }
  else
  {
    printf("File closed Successfully: %s\n",strerror(errno));
    printf("Error No: %d\n",errno);
  }

}
```

Output:



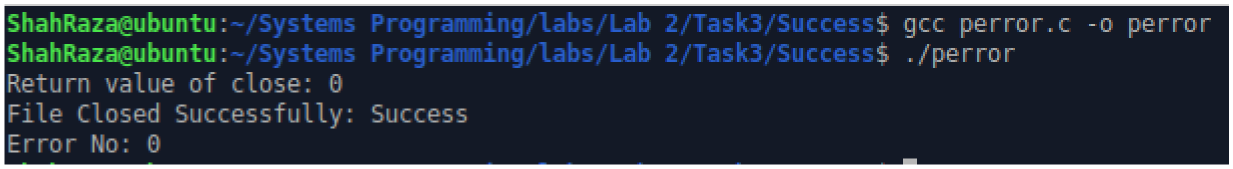
```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task3/Success$ gcc strerror.c -o strerror
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task3/Success$ ./strerror
Return value of close: 0
File closed Successfully: Success
Error No: 0
```

Perror:



```c
#include <stdio.h>
#include <unistd.h>
#include <errno.h>

int main()
{
  int ret = close(0);
  printf("Return value of close: %d\n",ret);
  if(ret==-1)
  {
    perror("Error Occured");
    printf("Error No: %d\n",errno);
  }
  else
  {
    perror("File Closed Successfully");
    printf("Error No: %d\n",errno);
  }

}
```

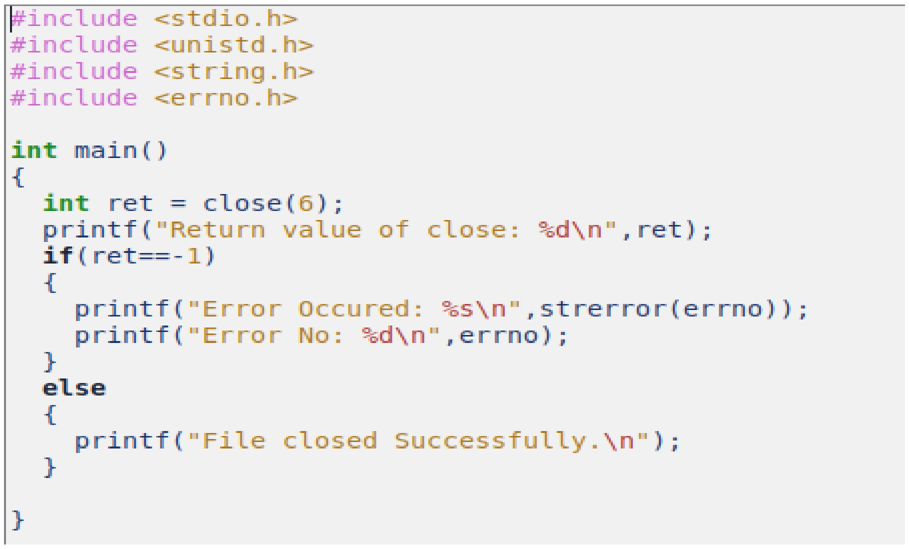
Output:



```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task3/Success$ gcc perror.c -o perror
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task3/Success$ ./perror
Return value of close: 0
File Closed Successfully: Success
Error No: 0
```

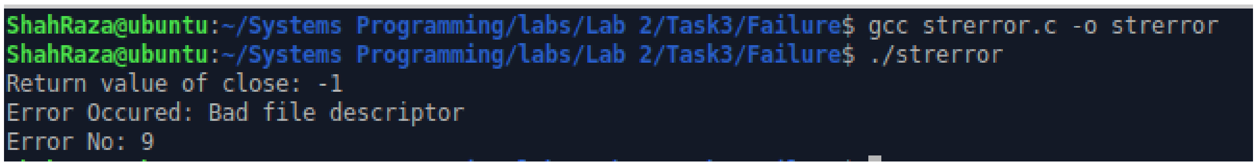
Failure:

Strerror:



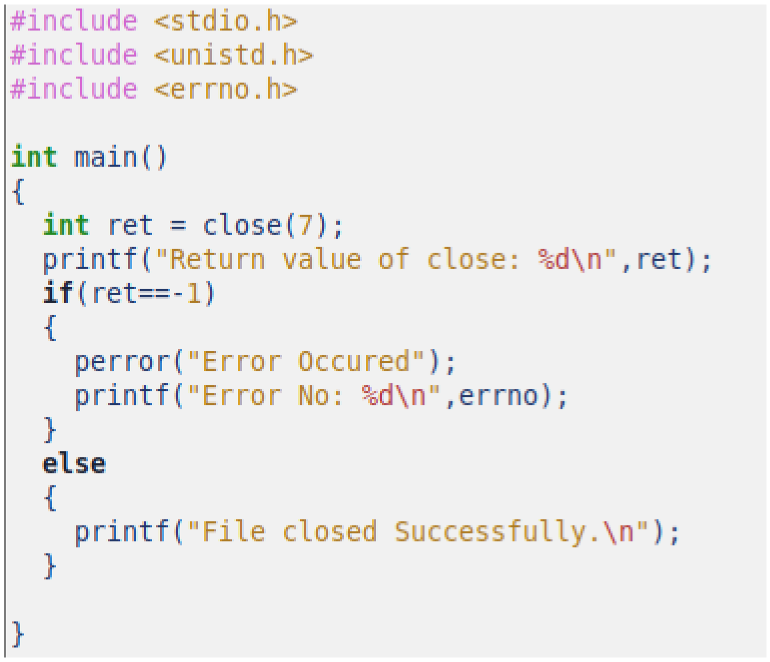
```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

int main()
{
  int ret = close(6);
  printf("Return value of close: %d\n",ret);
  if(ret==-1)
  {
    printf("Error Occured: %s\n",strerror(errno));
    printf("Error No: %d\n",errno);
  }
  else
  {
    printf("File closed Successfully.\n");
  }

}
```

Output:



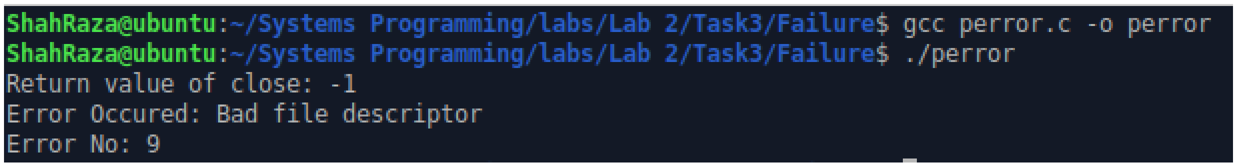
```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task3/Failure$ gcc strerror.c -o strerror
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task3/Failure$ ./strerror
Return value of close: -1
Error Occured: Bad file descriptor
Error No: 9
```

Perror:



```c
#include <stdio.h>
#include <unistd.h>
#include <errno.h>

int main()
{
  int ret = close(7);
  printf("Return value of close: %d\n",ret);
  if(ret==-1)
  {
    perror("Error Occured");
    printf("Error No: %d\n",errno);
  }
  else
  {
    printf("File closed Successfully.\n");
  }

}
```

Output:

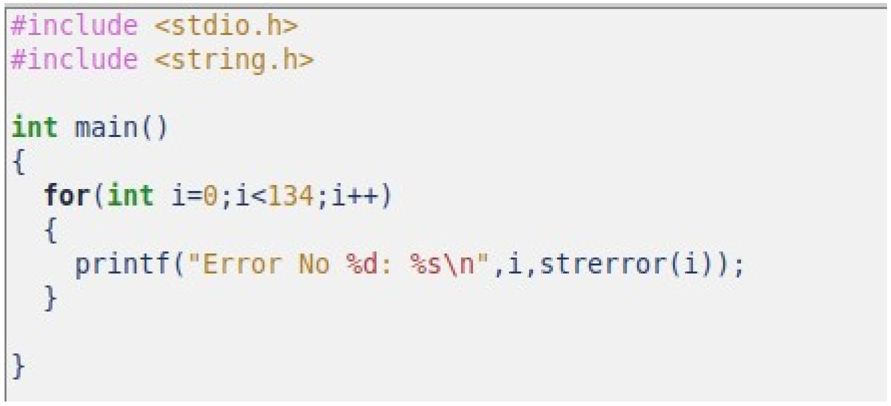


```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task3/Failure$ gcc perror.c -o perror
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task3/Failure$ ./perror
Return value of close: -1
Error Occured: Bad file descriptor
Error No: 9
```

Task 4:

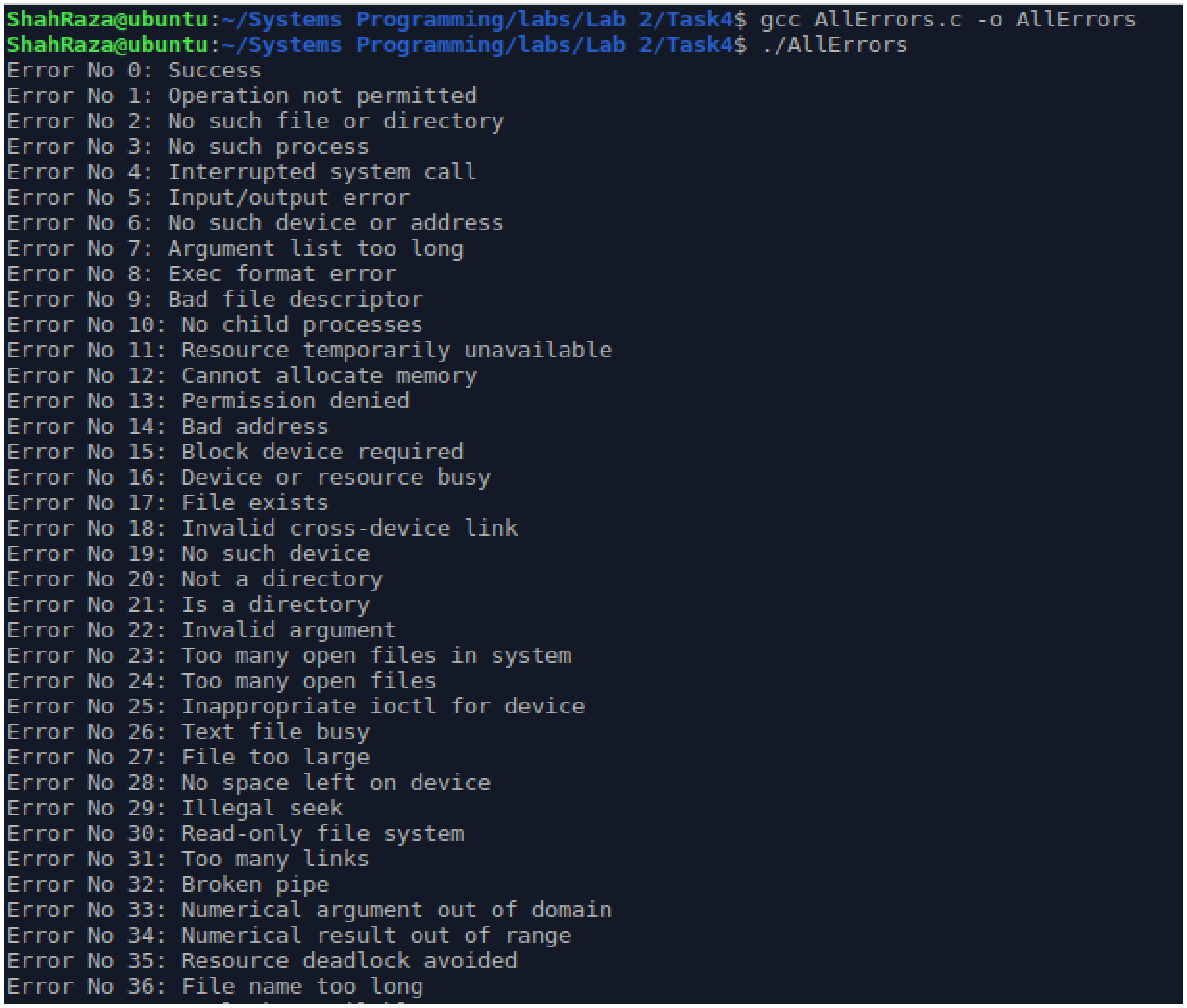
Print all the Error Messages.

Code:

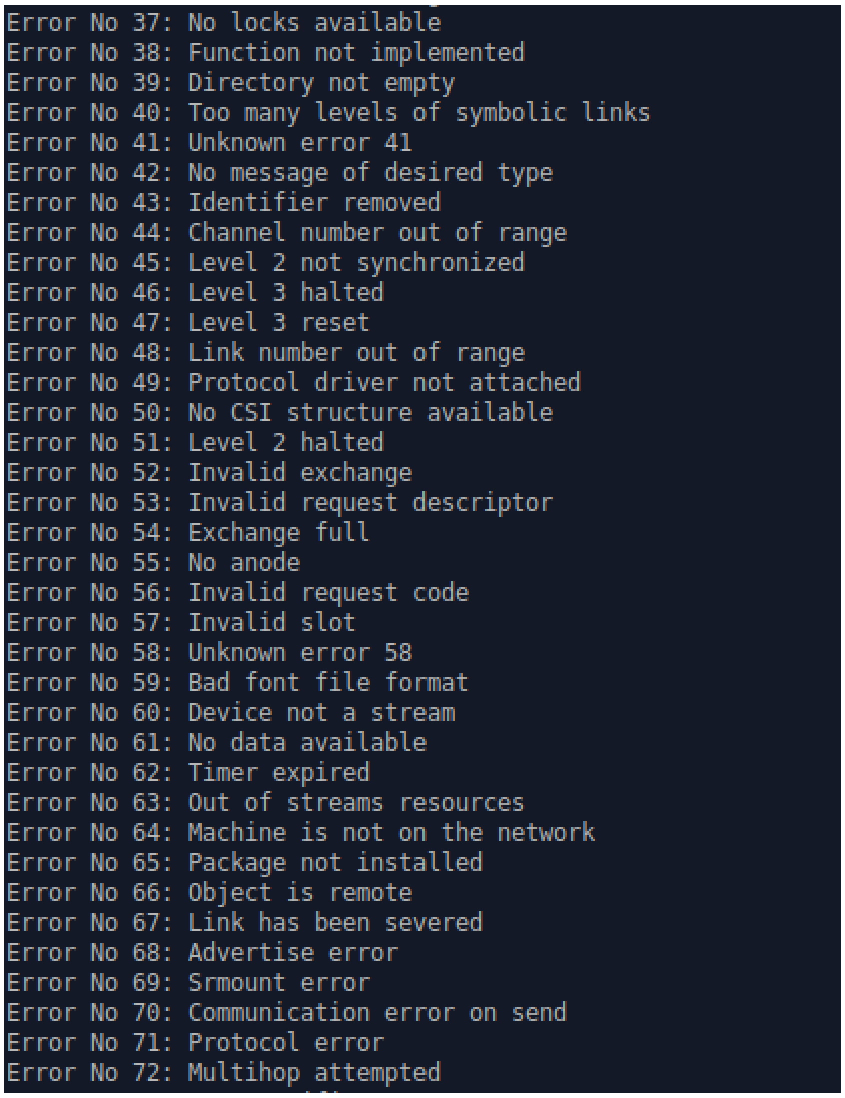


```c
#include <stdio.h>
#include <string.h>

int main()
{
  for(int i=0;i<134;i++)
  {
    printf("Error No %d: %s\n",i,strerror(i));
  }

}
```

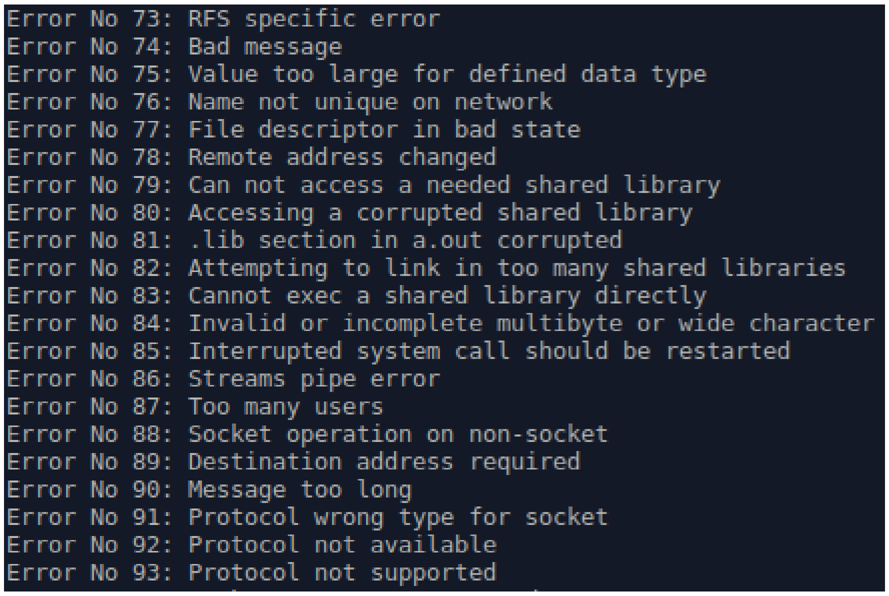
Output:



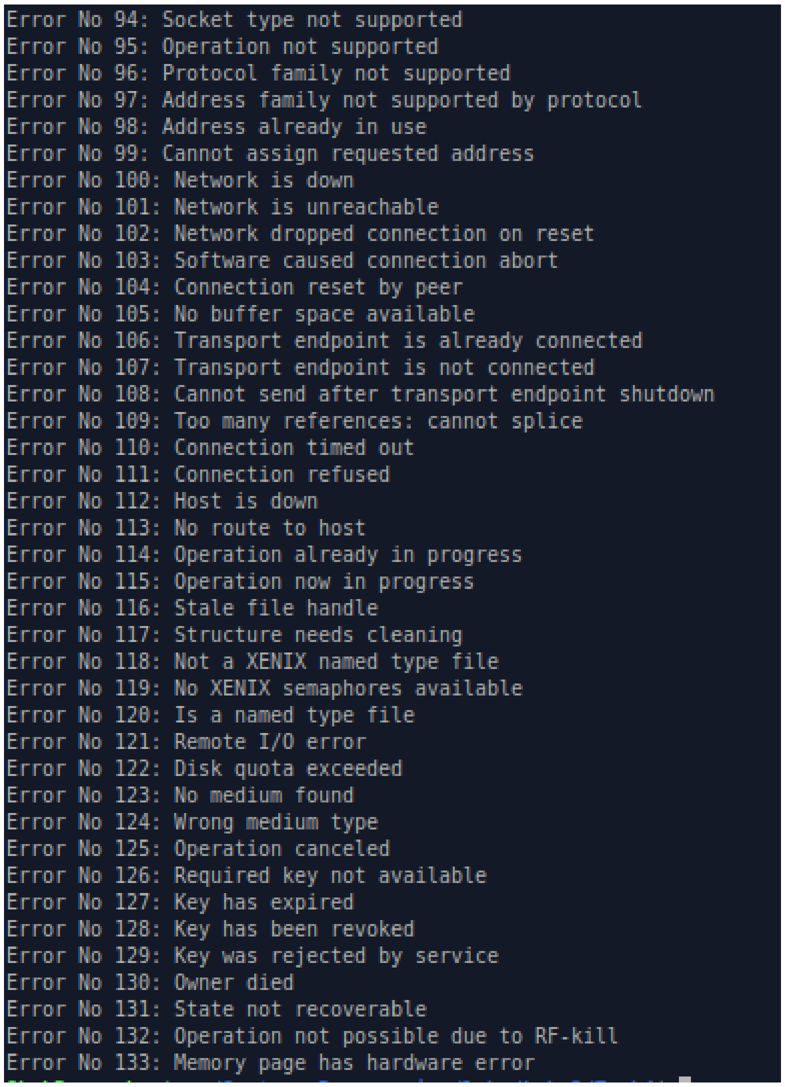
```
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task4$ gcc AllErrors.c -o AllErrors
ShahRaza@ubuntu:~/Systems Programming/labs/Lab 2/Task4$ ./AllErrors
Error No 0: Success
Error No 1: Operation not permitted
Error No 2: No such file or directory
Error No 3: No such process
Error No 4: Interrupted system call
Error No 5: Input/output error
Error No 6: No such device or address
Error No 7: Argument list too long
Error No 8: Exec format error
Error No 9: Bad file descriptor
Error No 10: No child processes
Error No 11: Resource temporarily unavailable
Error No 12: Cannot allocate memory
Error No 13: Permission denied
Error No 14: Bad address
Error No 15: Block device required
Error No 16: Device or resource busy
Error No 17: File exists
Error No 18: Invalid cross-device link
Error No 19: No such device
Error No 20: Not a directory
Error No 21: Is a directory
Error No 22: Invalid argument
Error No 23: Too many open files in system
Error No 24: Too many open files
Error No 25: Inappropriate ioctl for device
Error No 26: Text file busy
Error No 27: File too large
Error No 28: No space left on device
Error No 29: Illegal seek
Error No 30: Read-only file system
Error No 31: Too many links
Error No 32: Broken pipe
Error No 33: Numerical argument out of domain
Error No 34: Numerical result out of range
Error No 35: Resource deadlock avoided
Error No 36: File name too long
```



```
Error No 37: No locks available
Error No 38: Function not implemented
Error No 39: Directory not empty
Error No 40: Too many levels of symbolic links
Error No 41: Unknown error 41
Error No 42: No message of desired type
Error No 43: Identifier removed
Error No 44: Channel number out of range
Error No 45: Level 2 not synchronized
Error No 46: Level 3 halted
Error No 47: Level 3 reset
Error No 48: Link number out of range
Error No 49: Protocol driver not attached
Error No 50: No CSI structure available
Error No 51: Level 2 halted
Error No 52: Invalid exchange
Error No 53: Invalid request descriptor
Error No 54: Exchange full
Error No 55: No anode
Error No 56: Invalid request code
Error No 57: Invalid slot
Error No 58: Unknown error 58
Error No 59: Bad font file format
Error No 60: Device not a stream
Error No 61: No data available
Error No 62: Timer expired
Error No 63: Out of streams resources
Error No 64: Machine is not on the network
Error No 65: Package not installed
Error No 66: Object is remote
Error No 67: Link has been severed
Error No 68: Advertise error
Error No 69: Srmount error
Error No 70: Communication error on send
Error No 71: Protocol error
Error No 72: Multihop attempted
```



```
Error No 73: RFS specific error
Error No 74: Bad message
Error No 75: Value too large for defined data type
Error No 76: Name not unique on network
Error No 77: File descriptor in bad state
Error No 78: Remote address changed
Error No 79: Can not access a needed shared library
Error No 80: Accessing a corrupted shared library
Error No 81: .lib section in a.out corrupted
Error No 82: Attempting to link in too many shared libraries
Error No 83: Cannot exec a shared library directly
Error No 84: Invalid or incomplete multibyte or wide character
Error No 85: Interrupted system call should be restarted
Error No 86: Streams pipe error
Error No 87: Too many users
Error No 88: Socket operation on non-socket
Error No 89: Destination address required
Error No 90: Message too long
Error No 91: Protocol wrong type for socket
Error No 92: Protocol not available
Error No 93: Protocol not supported
```



```
Error No 94: Socket type not supported
Error No 95: Operation not supported
Error No 96: Protocol family not supported
Error No 97: Address family not supported by protocol
Error No 98: Address already in use
Error No 99: Cannot assign requested address
Error No 100: Network is down
Error No 101: Network is unreachable
Error No 102: Network dropped connection on reset
Error No 103: Software caused connection abort
Error No 104: Connection reset by peer
Error No 105: No buffer space available
Error No 106: Transport endpoint is already connected
Error No 107: Transport endpoint is not connected
Error No 108: Cannot send after transport endpoint shutdown
Error No 109: Too many references: cannot splice
Error No 110: Connection timed out
Error No 111: Connection refused
Error No 112: Host is down
Error No 113: No route to host
Error No 114: Operation already in progress
Error No 115: Operation now in progress
Error No 116: Stale file handle
Error No 117: Structure needs cleaning
Error No 118: Not a XENIX named type file
Error No 119: No XENIX semaphores available
Error No 120: Is a named type file
Error No 121: Remote I/O error
Error No 122: Disk quota exceeded
Error No 123: No medium found
Error No 124: Wrong medium type
Error No 125: Operation canceled
Error No 126: Required key not available
Error No 127: Key has expired
Error No 128: Key has been revoked
Error No 129: Key was rejected by service
Error No 130: Owner died
Error No 131: State not recoverable
Error No 132: Operation not possible due to RF-kill
Error No 133: Memory page has hardware error
```