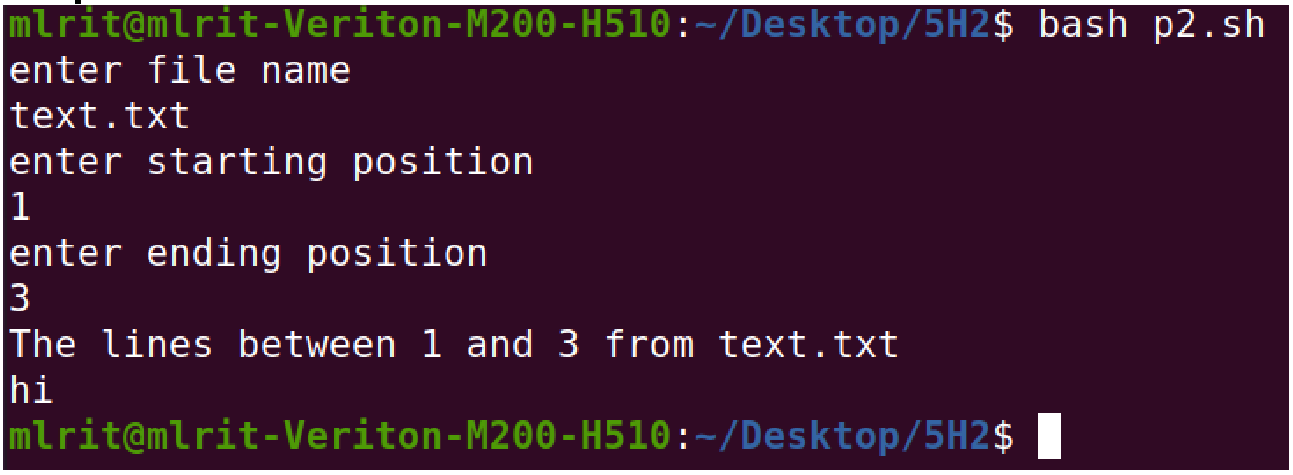
Program 2: Write a shell script that accepts a file name,starting and ending line numbers as arguments and display all the lines between the given line numbers.

```
echo "enter file name"
read f
echo 'enter starting position'
read st
echo 'enter ending position'
read end
echo 'The lines between' $st 'and' $end 'from' $f
if [ $st -lt $end ]
then
n1=`expr $st + 1`
n2=`expr $end - 1`
sed -n "$n1,$n2 p" $f
elif [ $st -gt $end ]
then
n3=`expr $st - 1`
n4=`expr $end + 1`
sed -n "$n4,$n3 p" $f
fi
```

Output:



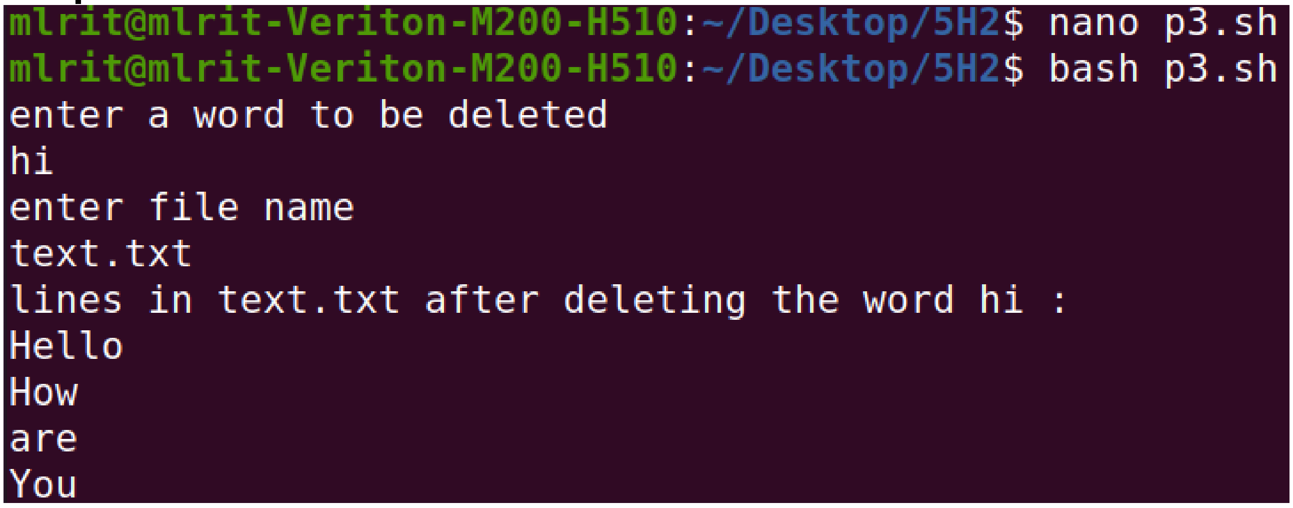
```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ bash p2.sh
enter file name
text.txt
enter starting position
1
enter ending position
3
The lines between 1 and 3 from text.txt
hi
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$
```

asdAS

Program 3: Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.

```
echo 'enter a word to be deleted'
read word
echo 'enter file name'
read fname
echo 'lines in' $fname 'after deleting the word' $word ':'
sed "/$word/d" $fname
```

Output

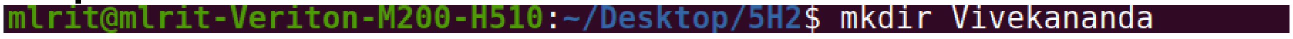


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ nano p3.sh
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ bash p3.sh
enter a word to be deleted
hi
enter file name
text.txt
lines in text.txt after deleting the word hi :
Hello
How
are
You
```

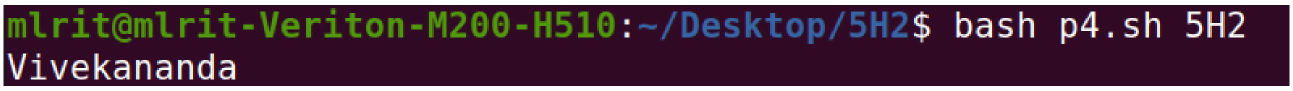
Program 4: Write a shell script that displays a list of all the files in the current directory to which the user has read,write and execute permissions.

```
for i in *
do
if [ -r $i -a -w $i -a -x $i ]
then
21
echo $i
fi
done
```

Output



```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ mkdir Vivekananda
```

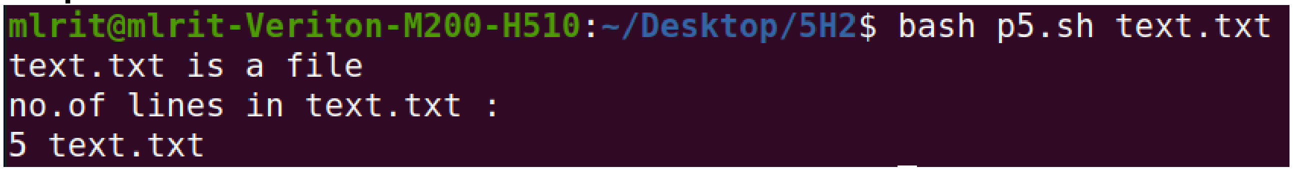


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ bash p4.sh 5H2
Vivekananda
```

Program 5: Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or directory and reports accordingly.whenever the argument is a file,the number of lines on it is also reported.

```
for fname in $*
do
if [ -f $fname ]
then
echo $fname 'is a file'
echo 'no.of lines in' $fname ':'
wc -l $fname
elif [ -d $fname ]
then
echo $fname 'is a directory'
else
echo 'Does not exist'
fi
done
```

Output:

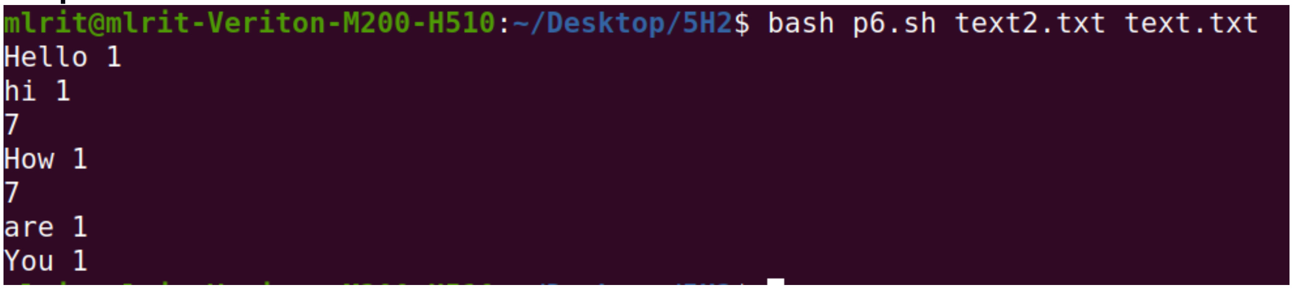


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ bash p5.sh text.txt
text.txt is a file
no.of lines in text.txt :
5 text.txt
```

Program 6: Write a shell script that accepts a file names as its arguments,counts and reports the occurrence of each word that is present in the file argument file in other argument files.

```
if [ $# -eq 0 ]
then
echo "no arguments"
else
tr " " "\n" < $1 > temp
shift
for i in $*
do
tr " " "\n" < $i > temp1
y=`wc -l < temp`
j=1
while [ $j -le $y ]
do
x=`head -n $j temp | tail -1`
c=`grep -c "$x" temp1`
echo $x $c
j=`expr $j + 1`
done
done
fi
```

Output:

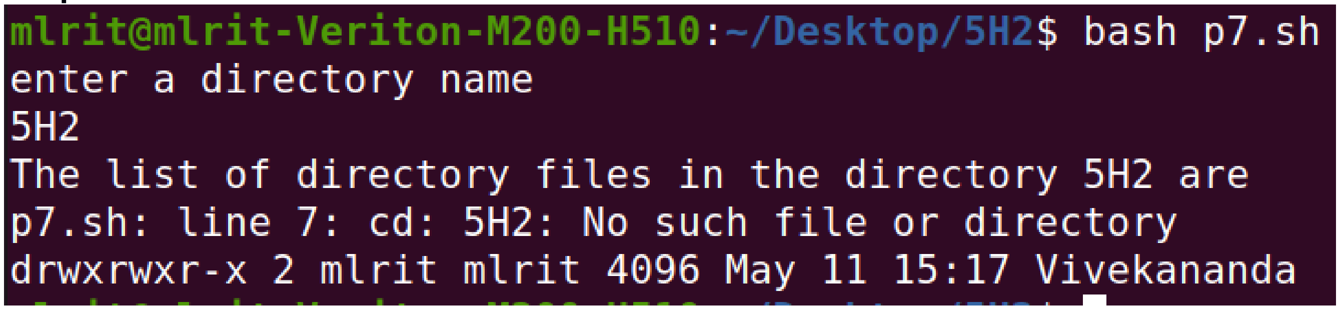


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ bash p6.sh text2.txt text.txt
Hello 1
hi 1
7
How 1
7
are 1
You 1
```

Program 7: Write a shell script to all of the directory files in a directory.

```
echo 'enter a directory name'

read dname

echo 'The list of directory files in the directory' $dname 'are'

cd $dname

ls -l | grep '^d'
```

Output:

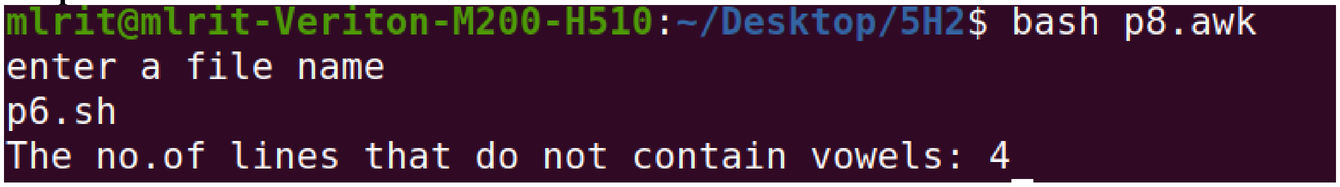


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ bash p7.sh
enter a directory name
5H2
The list of directory files in the directory 5H2 are
p7.sh: line 7: cd: 5H2: No such file or directory
drwxrwxr-x 2 mlrit mlrit 4096 May 11 15:17 Vivekananda
```

Program 8: **Write a awk script to count the number of lines in a file that do not contain vowels.**

```
echo 'enter a file name'
read fn
awk '$0 !~/[aeiou]/ { c=c+1 }
    END       { print("The no.of lines that do not contain vowels:",c) }' $fn
```

Output:

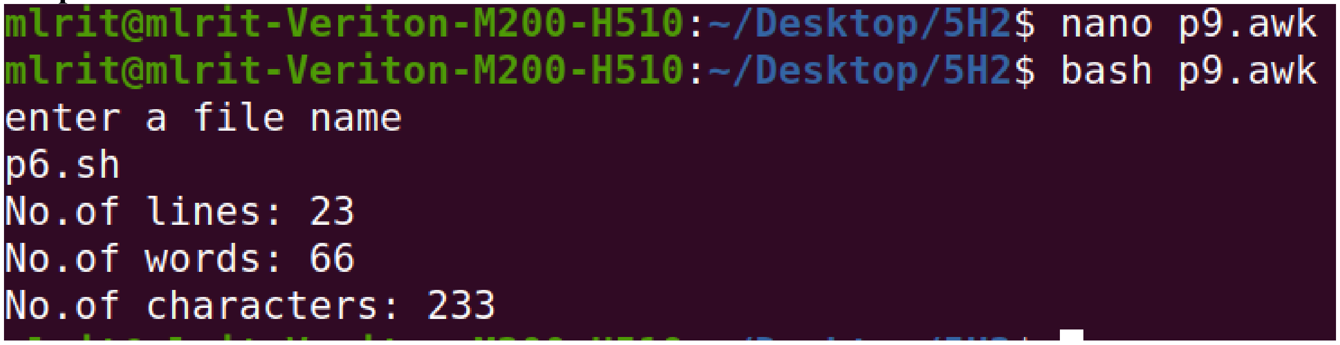


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ bash p8.awk
enter a file name
p6.sh
The no.of lines that do not contain vowels: 4
```

Program 9 : **Write a awk script to find the number of characters,words and lines in a file.**

```
echo "enter a file name"
read fn
awk '{ w=w+NF
     c=c+length($0)
    }
    END { print("No.of lines:",NR)
         print("No.of words:",w)
         print("No.of characters:",c)
       }' $fn
```

Output:

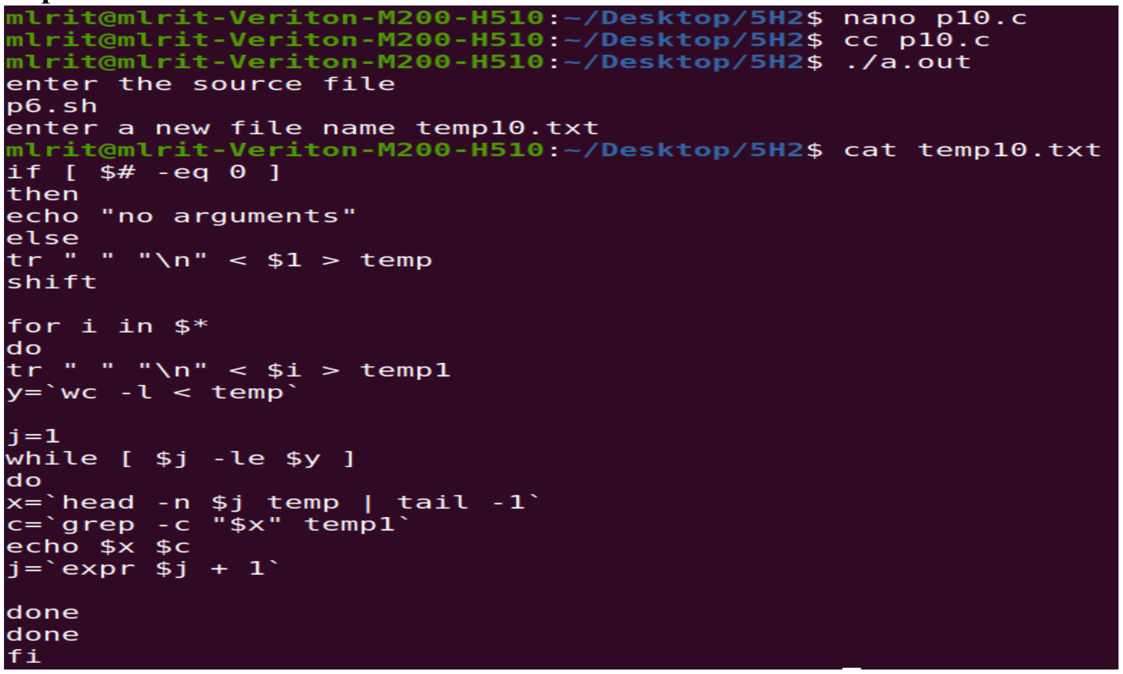


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ nano p9.awk
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ bash p9.awk
enter a file name
p6.sh
No.of lines: 23
No.of words: 66
No.of characters: 233
```

Program 10: Write a C program that makes a copy of a file using standard I/O and system calls.

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

#define MAX_SIZE 1000

void main()

{
    int fd1,fd2,r1,w1;

    char buffer[MAX_SIZE];

    char sourceName[100],destName[100];

    printf("enter the source file\n");

    scanf("%s",sourceName); printf("enter

    a new file name");

    scanf("%s",destName);

    fd1=open(sourceName,O_RDONLY);

    r1=read(fd1,buffer,MAX_SIZE);

    fd2=open(destName,O_CREAT|O_RDWR,0600);

    w1=write(fd2,buffer,r1);

}
```

Output:

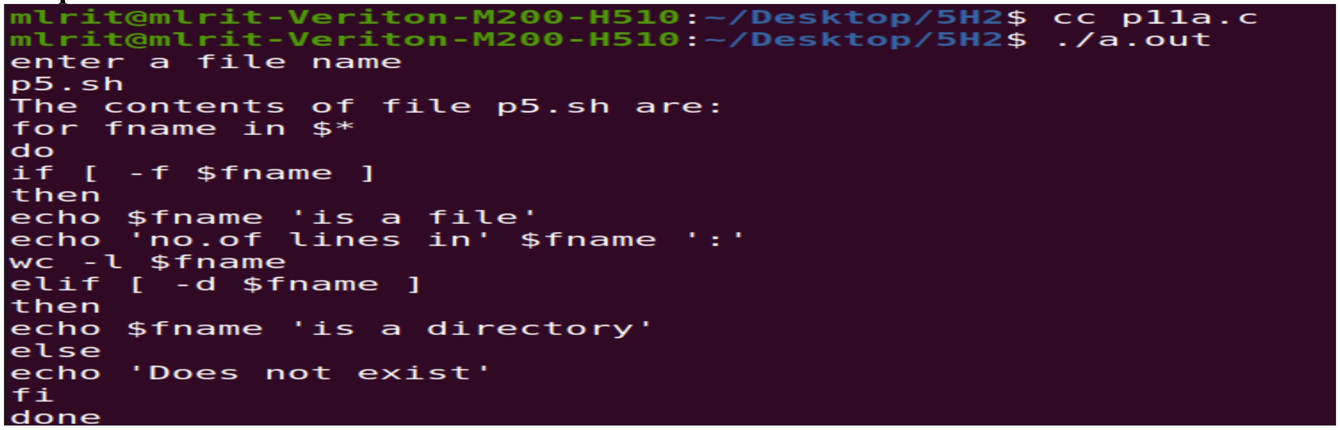


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ nano p10.c
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ cc p10.c
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ ./a.out
enter the source file
p6.sh
enter a new file name temp10.txt
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ cat temp10.txt
if [ $# -eq 0 ]
then
echo "no arguments"
else
tr " " "\n" < $1 > temp
shift

for i in $*
do
tr " " "\n" < $i > temp1
y=`wc -l < temp`

j=1
while [ $j -le $y ]
do
x=`head -n $j temp | tail -1`
c=`grep -c "$x" temp1`
echo $x $c
j=`expr $j + 1`

done
done
fi
```

Program 11:.Implement in C the following Unix commands using system calls.
a cat command

```
#include<fcntl.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#define MAX_SIZE
500
int main()

{
int fd1,n;

char
buf[MAX_SIZE],fname
[20]; printf("enter a file
name\n");
scanf("%s",fname);
fd1=open(fname,O_RD
ONLY); if(fd1==-1)
printf("the file does not
exist");
else
{

printf("The contents of
file %s are:\n",fname);
n=read(fd1,buf,MAX_S
IZE); write(1,buf,n);

}
}
```

Output:

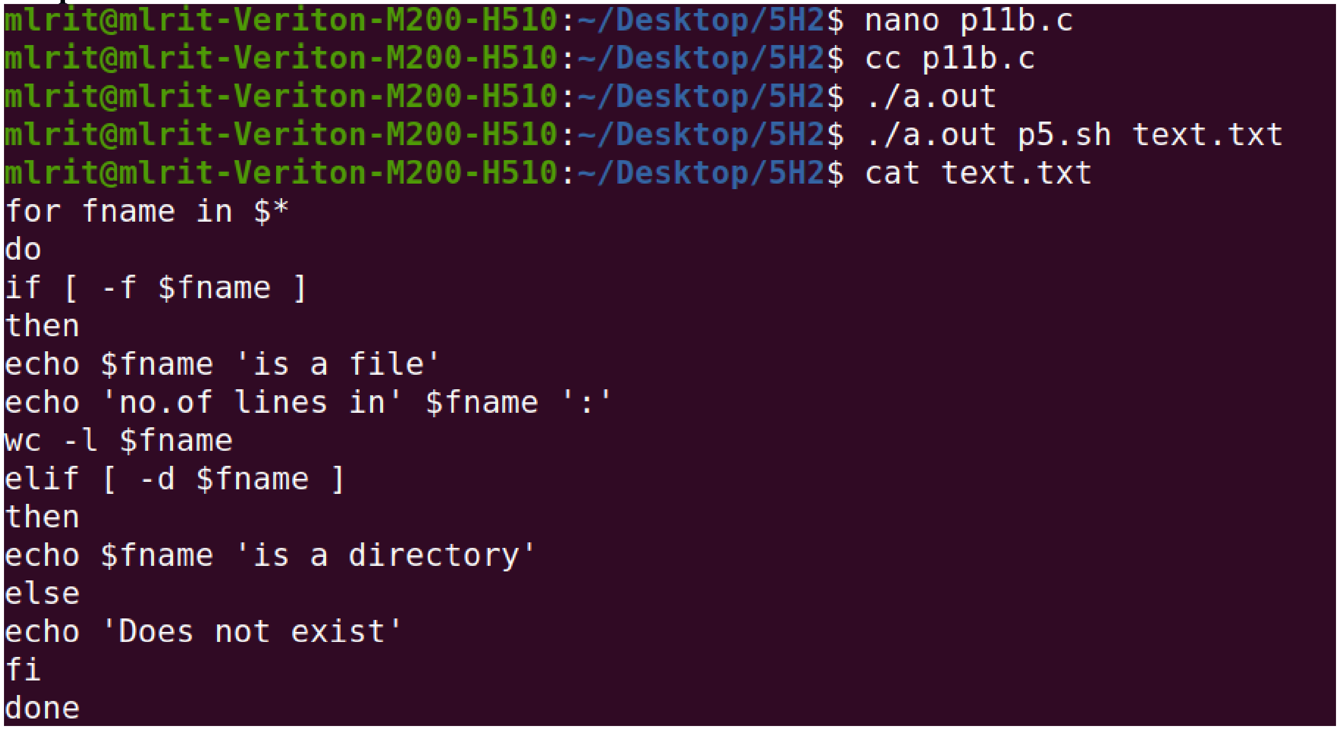


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ cc p11a.c
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ ./a.out
enter a file name
p5.sh
The contents of file p5.sh are:
for fname in $*
do
if [ -f $fname ]
then
echo $fname 'is a file'
echo 'no.of lines in' $fname ':'
wc -l $fname
elif [ -d $fname ]
then
echo $fname 'is a directory'
else
echo 'Does not exist'
fi
done
```

11b)mv command

```
#include<fcntl.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
int main(int argc,char *argv[])
{
        open(argv[1],O_RDONLY);
        creat(argv[2],S_IWUSR);
        rename(argv[1],argv[2]);
        unlink(argv[1]);
}
```

Output:

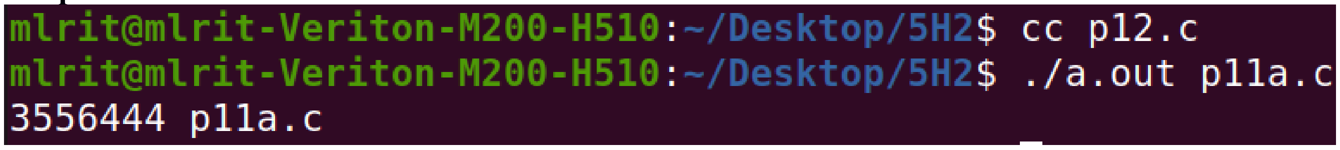


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ nano p11b.c
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ cc p11b.c
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ ./a.out
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ ./a.out p5.sh text.txt
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ cat text.txt
for fname in $*
do
if [ -f $fname ]
then
echo $fname 'is a file'
echo 'no.of lines in' $fname ':'
wc -l $fname
elif [ -d $fname ]
then
echo $fname 'is a directory'
else
echo 'Does not exist'
fi
done
```

Program 12: Write a C program to list for every file in a directory,its inode number in a given directory

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
int main(int argc,char *argv[])
{
char d[50];
if(argc==2)
{
bzero(d,sizeof(d));
strcat(d,"ls ");
strcat(d,"-i ");
strcat(d,argv[1]);
system(d);
}
else
printf("\nInvalid no.of inputs");
}
```

Output:

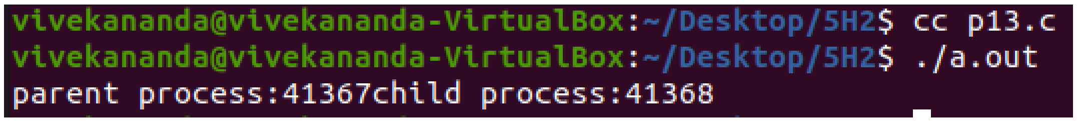


```
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ cc p12.c
mlrit@mlrit-Veriton-M200-H510:~/Desktop/5H2$ ./a.out p11a.c
3556444 p11a.c
```

Program 13: Write a C program to create a child process and allow the parent to display "parent" and the child to display "child " on the screen.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
int main()
{
pid_t pid;
pid=fork();

if(pid==-1)
printf("failed to fork");


else if(pid==0)
printf("child process:%d\n",getpid());
else
printf("parent process:%d",getpid());
}
```

Output:

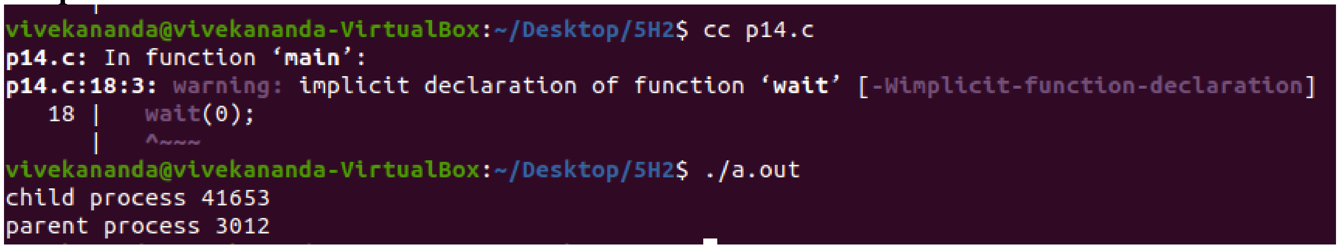


```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p13.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
parent process:41367child process:41368
```

Program 14: Write a C program to create a Zombie process

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
void main()
{
        int pid;
        pid=fork();
        if(0==pid)
        {
                printf("child process %d \n",getpid());
               exit(0);
        }
        else
        {
                wait(0);
                sleep(10);
                printf("parent process %d \n", getppid());
        }
}
```

Output:

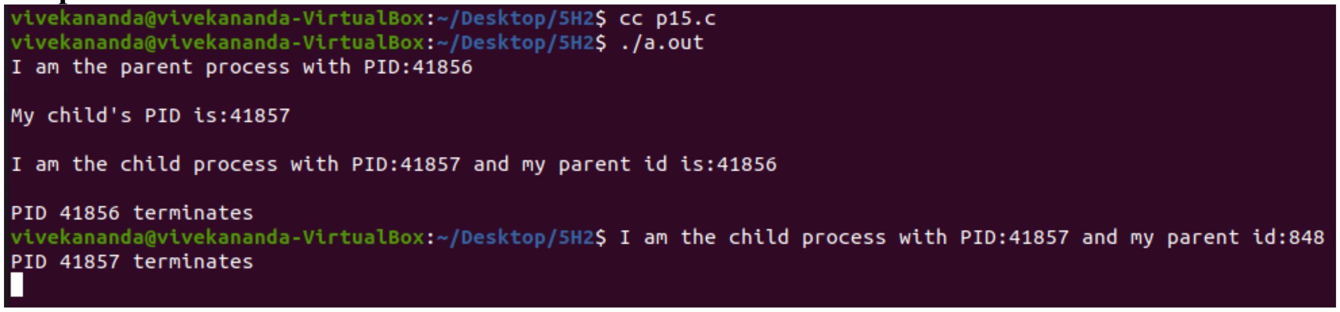


```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p14.c
p14.c: In function ‘main’:
p14.c:18:3: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   18 |   wait(0);
      |   ^~~~
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
child process 41653
parent process 3012
```

Program 15: Write a C program that illustrates how an orphan is created.
i) Orphan process

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
void main()
{
        int pid;
        pid=fork();
        if(pid==0)
        {
                printf("I am the child process with PID:%d and my parent id
is:%d\n\n",getpid(),getppid());
                sleep(15);
                printf("I am the child process with PID:%d and my parent id:%d\n",getpid(),getppid());
    }
        else
        {

                printf("I am the parent process with PID:%d\n \n",getpid());
                printf("My child's PID is:%d\n\n",pid); sleep(1);


    }
        printf("PID %d terminates \n",getpid()); /*Both processes execute this*/}
```

Output:

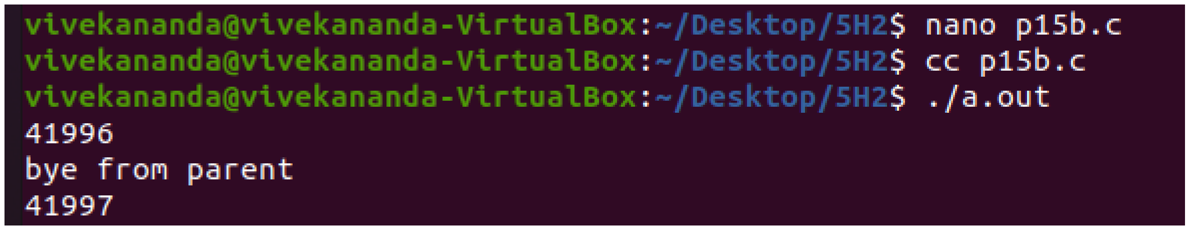


```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p15.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
I am the parent process with PID:41856

My child's PID is:41857

I am the child process with PID:41857 and my parent id is:41856

PID 41856 terminates
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ I am the child process with PID:41857 and my parent id:848
PID 41857 terminates
```

ii)Orphan process

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
 void main()
{
        int pid;
        pid=fork();
        if(0==pid)
        {
                printf("%d\n",getpid());
                sleep(5);
                printf("%d\n",getpid());
        }
        else
        {
        printf("%d\n",getpid());
        printf("bye from parent\n");
        exit(0);
        }
}
```

Output:

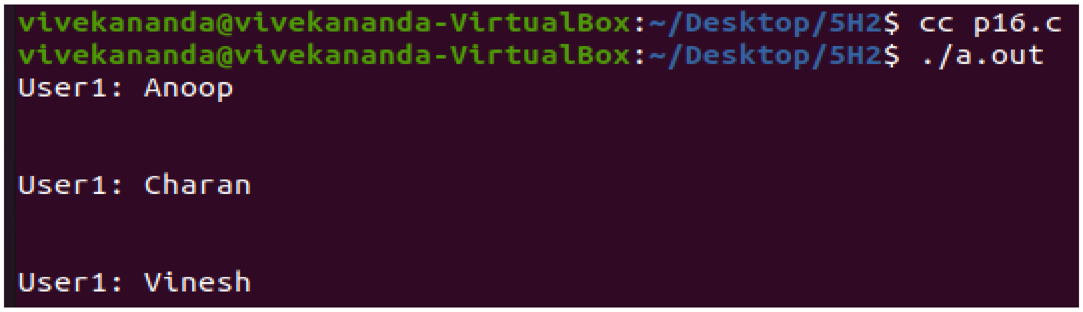


```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ nano p15b.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p15b.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
41996
bye from parent
41997
```

Program 16: Write a C program that illustrates communication between two unrelated process using named pipe(FIFO).

```
//read write
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
   int fd1;
   // FIFO file path
   char * myfifo = "/tmp/myfifo";
   // Creating the named file(FIFO)
   // mkfifo(<pathname>,<permission>)
   mkfifo(myfifo, 0666);

   char str1[80], str2[80];
   while (1)
   {
      // First open in read only and read
      fd1 = open(myfifo,O_RDONLY);
      read(fd1, str1, 80);
      // Print the read string and close
      printf("User1: %s\n", str1);
      close(fd1);
      // Now open in write mode and write
      // string taken from user.
      fd1 = open(myfifo,O_WRONLY);
      fgets(str2, 80, stdin);
      write(fd1, str2, strlen(str2)+1);
      close(fd1);
   }
   return 0;
}
```

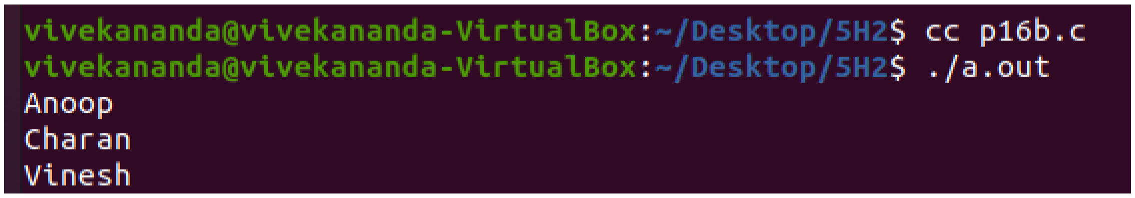
output:



```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p16.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
User1: Anoop


User1: Charan


User1: Vinesh
```

```
/fifo write read// C program to implement one side of FIFO
// This side writes first, then reads
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
   int fd;
   // FIFO file path
   char * myfifo = "/tmp/myfifo";
   // Creating the named file(FIFO)
   // mkfifo(<pathname>, <permission>)
   mkfifo(myfifo, 0666);
   char arr1[80], arr2[80];
   while (1)
   {
      // Open FIFO for write only
      fd = open(myfifo, O_WRONLY);
      // Take an input arr2ing from user.
      // 80 is maximum length
      fgets(arr2, 80, stdin);
      // Write the input arr2ing on FIFO
      // and close it
      write(fd, arr2, strlen(arr2)+1);
      close(fd);
      // Open FIFO for Read only
      fd = open(myfifo, O_RDONLY);
      // Read from FIFO
      read(fd, arr1, sizeof(arr1));
      // Print the read message
      printf("User2: %s\n", arr1);
      close(fd);
   }
   return 0;
}
```

Output:

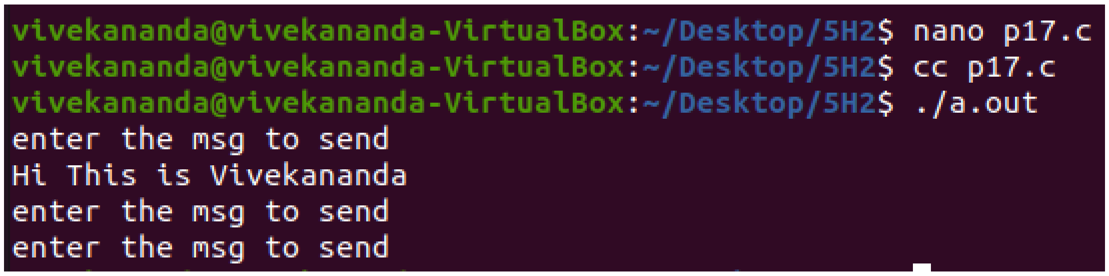


```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p16b.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
Anoop
Charan
Vinesh
```

Program 17:write a C program(sender.c) to create a message queue with read and write permissions to wrte 3 messages to it with different priority numbers.

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main(){
int qid, len,i;
char s[15];
struct
{
   long mtype;
   char mtext[15];
} message,buff;
qid = msgget(1000,IPC_CREAT|0666);
if(qid==-1)
{
   perror("message queue create failed");
   exit(1);
}
for(i=1;i<=3;i++){
   printf("enter the msg to send \n");
   scanf("%s",s);
   strcpy(message.mtext,s);
   message.mtype=i;
   len=strlen(message.mtext);
   if(msgsnd(qid,&message,len+1,0)==-1)
   {
   perror("message failed\n");
   exit(1);
   }
}
}
```

Output:

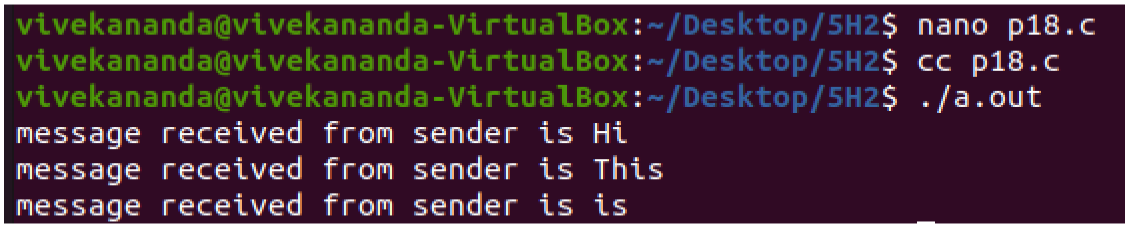


```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ nano p17.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p17.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
enter the msg to send
Hi This is Vivekananda
enter the msg to send
enter the msg to send
```

Program 18:write a C program(receiver.c) that receives the messages from sender and displays them.

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main(){
int qid, len,i;
char s[15];
struct
{
   long mtype;
   char mtext[15];
} buff;
qid = msgget(1000,IPC_CREAT|0666);
if(qid==-1)
{
   perror("message queue create failed");
   exit(1);
}
for(i=1;i<=3;i++){
   if(msgrcv(qid,&buff,15,i,0)==-1)
   {
      perror("message failed\n");
      exit(1);
   }
   printf("message received from sender is %s\n",buff.mtext);
}
}
```

Output:



```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ nano p18.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p18.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
message received from sender is Hi
message received from sender is This
message received from sender is is
```

Program 19:write client and server programs(using c) for interaction between server and client processes using Unix Domain Sockets.
a)unix domain sockets(interaction between server and client)
(server)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <stdio.h>
#define NSTRS 3
#define ADDRESS "mysocket"
/* no. of strings */
//"mysocket" /* addr to connect */
/*

  • Strings we send to the client. */

char *strs[NSTRS] = {
"This is the first string from the server.\n",
"This is the second string from the server.\n",
"This is the third string from the server.\n"
};

main()
{
char c;
FILE *fp;
int fromlen;
register int i, s, ns, len;
struct sockaddr_un saun, fsaun;

/*
  • Get a socket to work with. This socket will
  • be in the UNIX domain, and will be a
  • stream socket.
*/

if ((s = socket(AF_UNIX, SOCK_STREAM, 0)) < 0) { perror("server: socket");

exit(1);
}

/*

  • Create the address we will be binding to. */

saun.sun_family = AF_UNIX;
strcpy(saun.sun_path, ADDRESS);

/*
* Try to bind the address to the socket. We
```

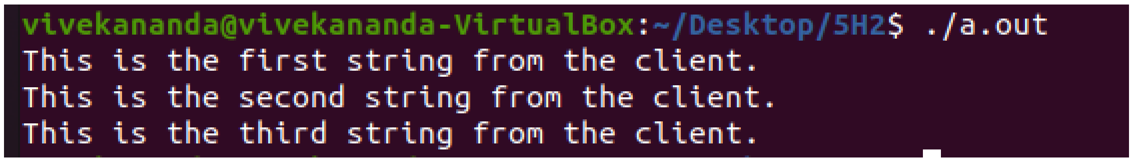
```
    • unlink the name first so that the bind won't
    • fail.
*
    • The third argument indicates the "length" of
    • the structure, not just the length of the
    • socket name.
*/
unlink(ADDRESS);

len = sizeof(saun.sun_family) + strlen(saun.sun_path);

if (bind(s, &saun, len) < 0) {
perror("server: bind");
exit(1);

}

/*

    • Listen on the socket. */

if (listen(s, 5) < 0) {
perror("server: listen");
exit(1);
}

/*
    • Accept connections. When we accept one, ns
    • will be connected to the client. fsaun will
    • contain the address of the client.
*/
if ((ns = accept(s, &fsaun, &fromlen)) < 0) {
perror("server: accept");
exit(1);
}

/*

    • We'll use stdio for reading the socket. */

fp = fdopen(ns, "r");

/*

    • First we send some strings to the client. */

for (i = 0; i < NSTRS; i++)
send(ns, strs[i], strlen(strs[i]), 0);

/*
* Then we read some strings from the client and
    • print them out. */

for (i = 0; i < NSTRS; i++) {
while ((c = fgetc(fp)) != EOF) {
putchar(c);
if (c == '\n')
break;

}
}

/*
```

- We can simply use close() to terminate the

- connection, since we're done with both sides. */

```
close(s);

exit(0);
}
```

Output:

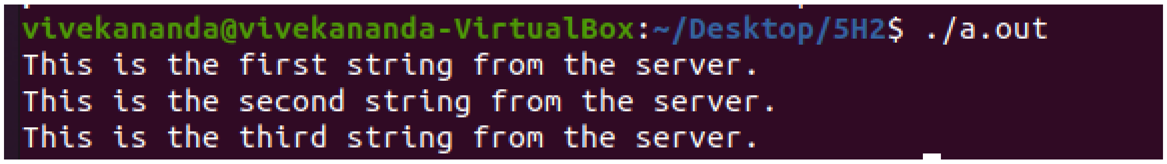


```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
This is the first string from the client.
This is the second string from the client.
This is the third string from the client.
```

20 unix domain sockets(interaction between server and client)
Client.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <stdio.h>
#define NSTRS 3
#define ADDRESS "mysocket"
/* no. of strings */
 /* addr to connect */
/*
   • Strings we send to the server. */

char *strs[NSTRS] = {
"This is the first string from the client.\n",
"This is the second string from the client.\n",
"This is the third string from the client.\n"
};
void main(){
char c;
FILE *fp;
register int i, s, len;
struct sockaddr_un saun;

/*
   • Get a socket to work with. This socket will
   • be in the UNIX domain, and will be a
   • stream socket.
*/

if ((s = socket(AF_UNIX, SOCK_STREAM, 0)) < 0) { perror("client: socket");

exit(1);
}

/*
   • Create the address we will be connecting to. */
saun.sun_family = AF_UNIX;
strcpy(saun.sun_path, ADDRESS);

/*
   • Try to connect to the address. For this to
   • succeed, the server must already have bound
   • this address, and must have issued a listen()
   • request.
*
   • The third argument indicates the "length" of
   • the structure, not just the length of the
   • socket name.
*/
len = sizeof(saun.sun_family) + strlen(saun.sun_path);

if (connect(s, &saun, len) < 0) {
perror("client: connect");
exit(1);
}
/*
    • We'll use stdio for reading
    • the socket.
*/
fp = fdopen(s, "r");
/*
    • First we read some strings from the server
    • and print them out.
*/
for (i = 0; i < NSTRS; i++) {
while ((c = fgetc(fp)) != EOF) {
putchar(c);
if (c == '\n')
break;
}
}
/*
    • Now we send some strings to the server. */
for (i = 0; i < NSTRS; i++)
send(s, strs[i], strlen(strs[i]), 0);
/*
    • We can simply use close() to terminate the
    • connection, since we're done with both sides. */
close(s);
exit(0);
}
```

Output:

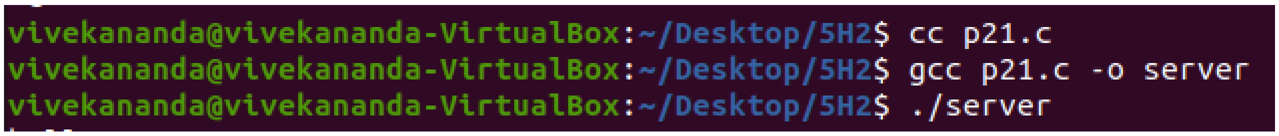


```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./a.out
This is the first string from the server.
This is the second string from the server.
This is the third string from the server.
```

21. internet sockets
a) server1.c

```
#include<netinet/in.h> //structure for storing address information
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h> //for socket APIs
#include<sys/types.h>
int main(int argc, char const* argv[])
{

    // create server socket similar to what was done in
    // client program
    int servSockD = socket(AF_INET, SOCK_STREAM, 0);

    // string store data to send to client
    char serMsg[255] = "Message from the server to the client";

    // define server address
    struct sockaddr_in servAddr;
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(9001);
    servAddr.sin_addr.s_addr = INADDR_ANY;

    // bind socket to the specified IP and port
    bind(servSockD, (struct sockaddr*)&servAddr,
         sizeof(servAddr));

    // listen for connections
    listen(servSockD, 1);
    // integer to hold client socket.
    int clientSocket = accept(servSockD, NULL, NULL);

    // send's messages to client socket
    send(clientSocket, serMsg, sizeof(serMsg), 0);

    return 0;
}
```

Output:

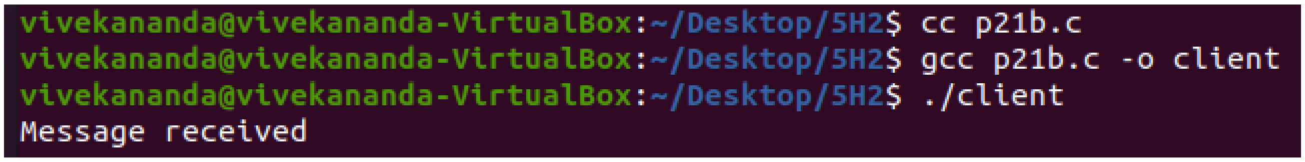


```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p21.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ gcc p21.c -o server
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./server
```

b)client.c

```
#include<netinet/in.h> //structure for storing address information
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h> //for socket APIs
#include<sys/types.h>

int main(int argc, char const* argv[])
{
   int sockD = socket(AF_INET, SOCK_STREAM, 0);

   struct sockaddr_in servAddr;

   servAddr.sin_family = AF_INET;
   servAddr.sin_port = htons(9001); // use some unused port number
   servAddr.sin_addr.s_addr = INADDR_ANY;
   int connectStatus = connect(sockD, (struct sockaddr*)&servAddr, sizeof(servAddr));

   if (connectStatus == -1) {
      printf("Error...\n");
   }

   else {
      char strData[255];

      recv(sockD, strData, sizeof(strData), 0);

      printf("Message: %s\n", strData);
   }

   return 0;
}
```

output:



```
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ cc p21b.c
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ gcc p21b.c -o client
vivekananda@vivekananda-VirtualBox:~/Desktop/5H2$ ./client
Message received
```