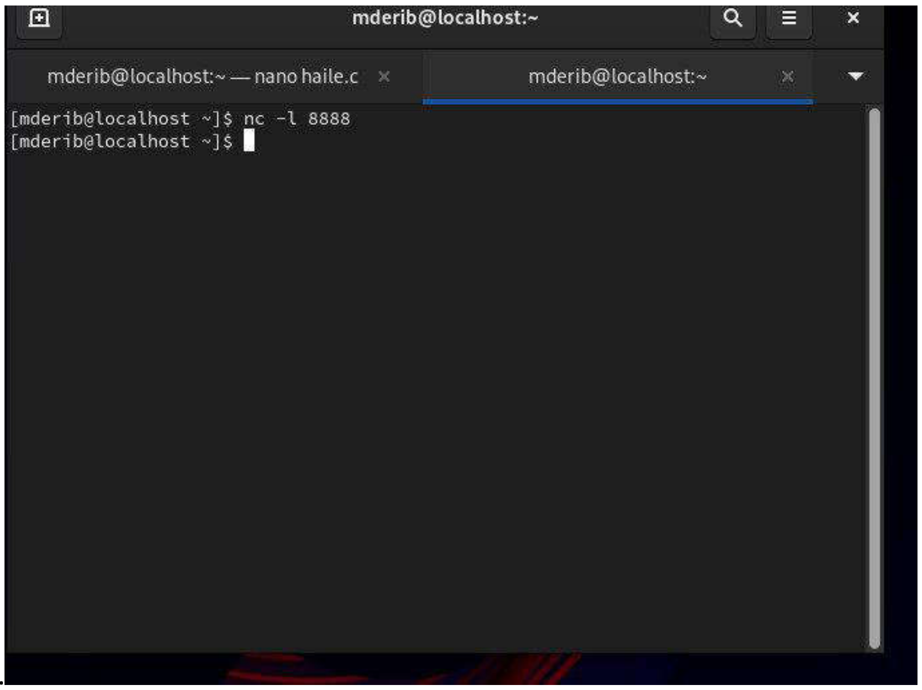
System call

Code for the system call

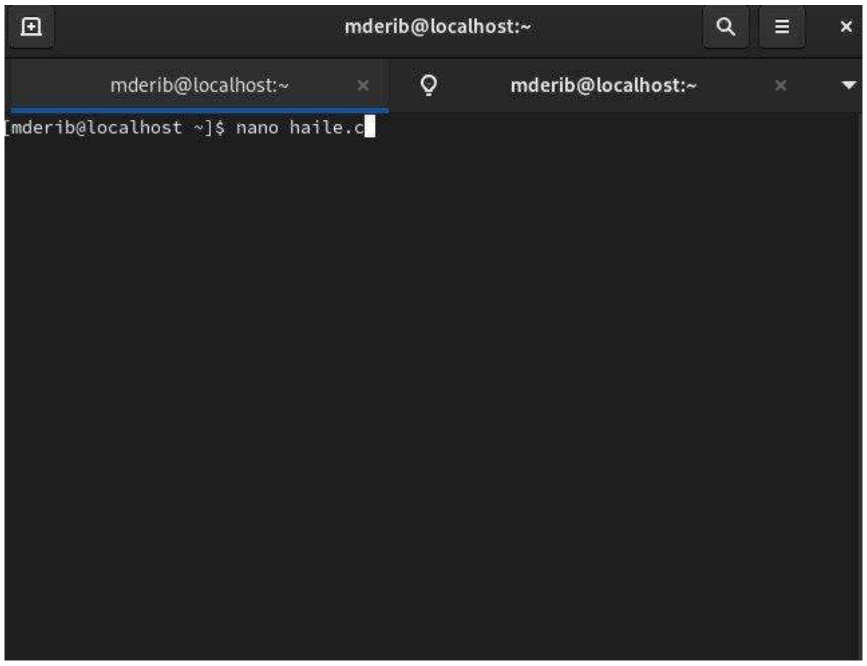
```
#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>


int main()

{

    int sock = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in server;

    server.sin_family = AF_INET;

    server.sin_port = htons(8888);

    server.sin_addr.s_addr = inet_addr("127.0.0.1");


    if (connect(sock, (struct sockaddr*)&server, sizeof(server)) == 0) {

        printf("connected to server");

    } else {

        perror("connection failed");

    }


    close(sock);

    return 0;

}
```



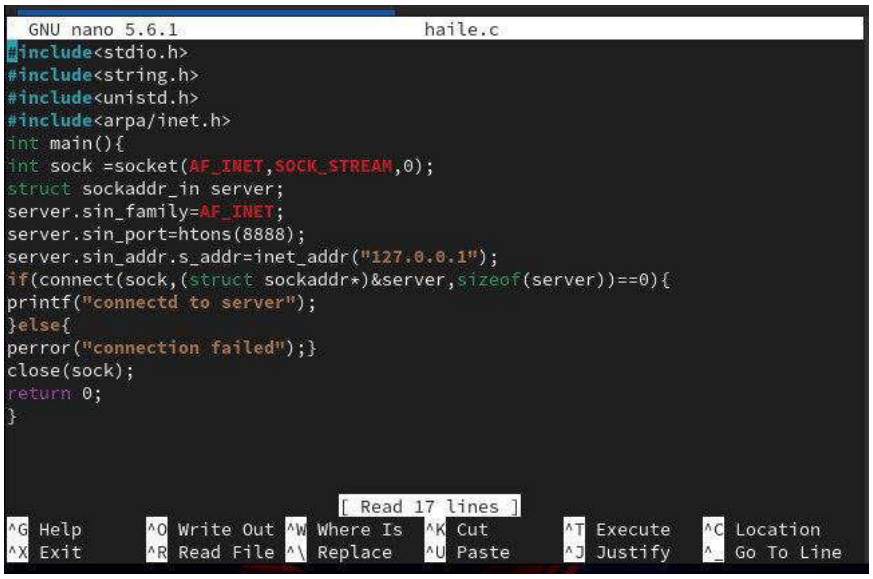
```
[mderib@localhost ~]$ nc -l 8888
[mderib@localhost ~]$
```

.

The screenshot shows a terminal window on a Linux system where the user has run the command `nc -l 8888`. This starts a Netcat listener on port 8888, waiting for incoming TCP connections. It's likely being used to test a client program, such as the C socket code shown previously

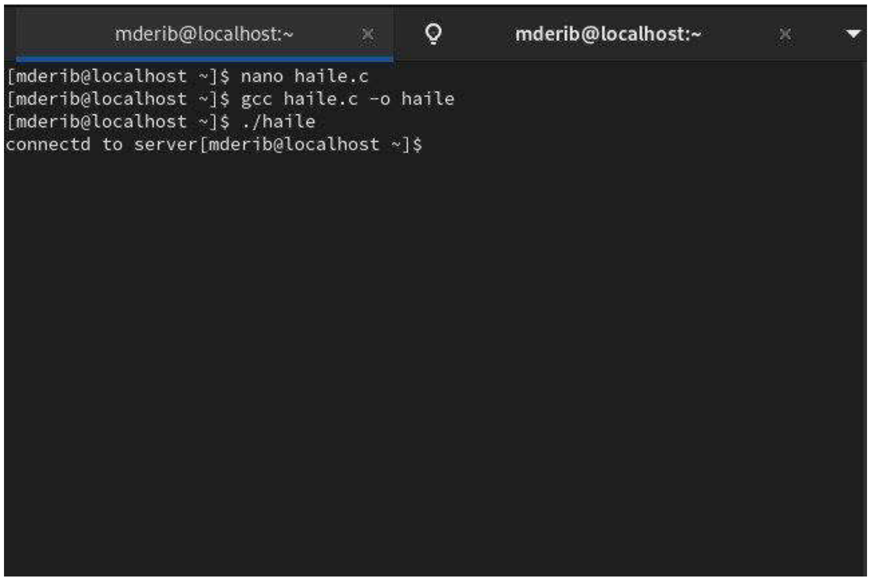


```
[mderib@localhost ~]$ nano haile.c
```



```
  GNU nano 5.6.1                          haile.c
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
int main(){
int sock =socket(AF_INET,SOCK_STREAM,0);
struct sockaddr_in server;
server.sin_family=AF_INET;
server.sin_port=htons(8888);
server.sin_addr.s_addr=inet_addr("127.0.0.1");
if(connect(sock,(struct sockaddr*)&server,sizeof(server))==0){
printf("connectd to server");
}else{
perror("connection failed");}
close(sock);
return 0;
}



                               [ Read 17 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

Here is implementing the connect system call by using the code given in the above page



```
[mderib@localhost ~]$ nano haile.c
[mderib@localhost ~]$ gcc haile.c -o haile
[mderib@localhost ~]$ ./haile
connectd to server[mderib@localhost ~]$
```

The result after implementing the code