10. Demonstrate setting up a simple Web Server and Host Website on Your Own Linux Computer.

- Prerequisites to Setup Web Server
- How to Setup A Web Server
- Install Apache2
- Install MySQL
- Install PHP

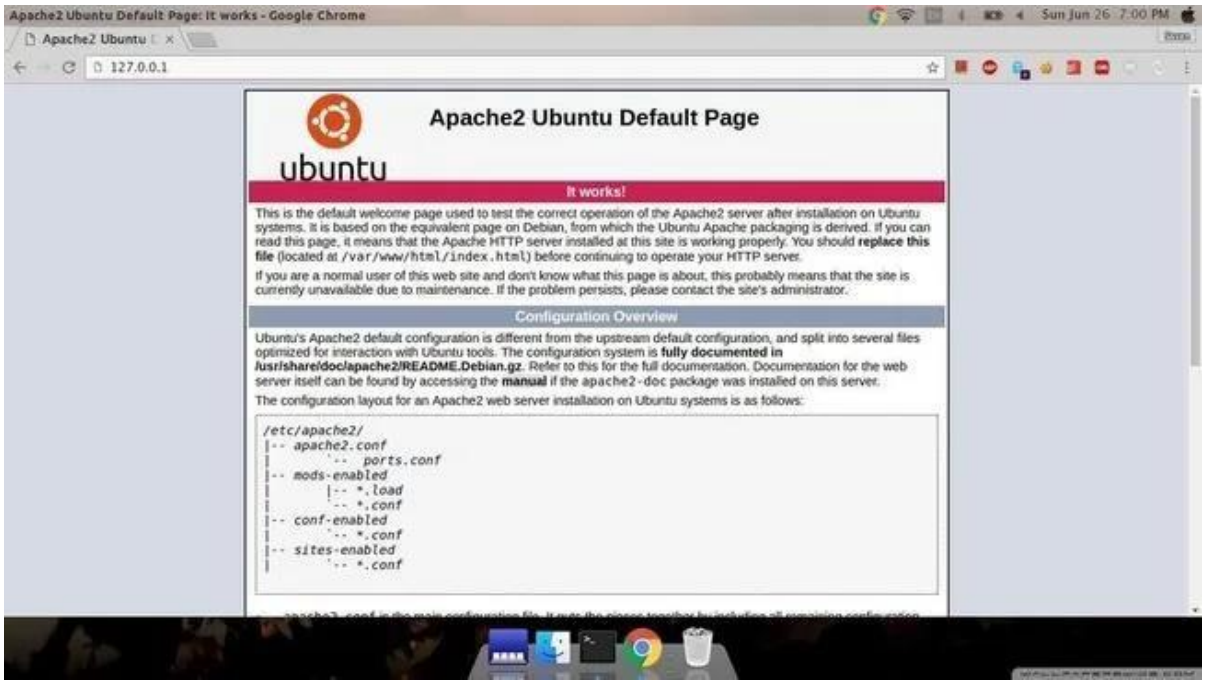
Prerequisites to Setup Web Server
To setup web server on your own Linux computer, we'll require the following three components to be installed –

- **Apache2:** apache2 is open-source HTTP server. It is still the most popular web-server used worldwide today.
- **php and php SQLite component**: PHP is a server-side scripting language. PHP and its component will help you to interact with a backend MySQL database for your website.
- **MySQL**: MySQL is a database solution in which you shall be storing your data in the table.

If you have installed the above components, you can skip this part and move to the next part.

How to Setup A Web Server
Install Apache2
Apache is open source web-server software that powers much of the web today. It is maintained by apache-http-project. Explore more here: https://httpd.apache.org/

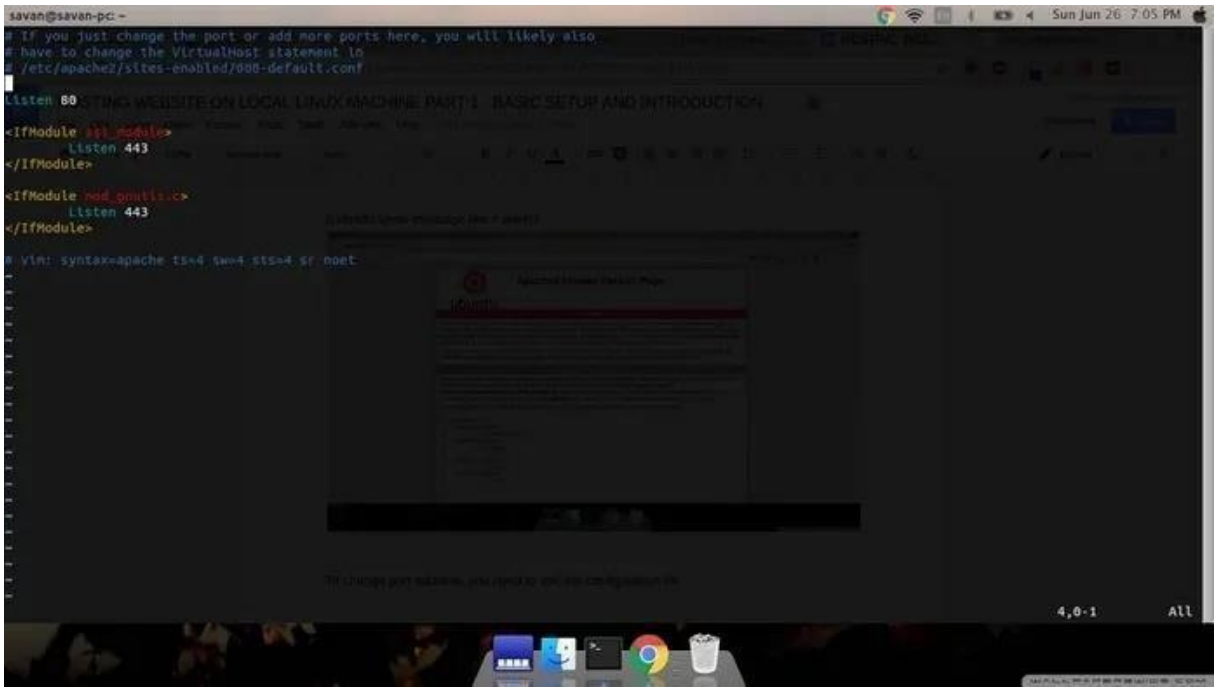
Open your terminal and type in commands –
```
sudo apt-get updatesudo apt-get install apache2
```



To check if apache2 is installed properly –

```
sudo service apache2 restart
```

Open your web browser and open the link using ip–address of your server. If you are practicing locally, you can type in localhost or 127.0.0.1. By default, Apache runs on port 80 and hence you need not provide the port number in your browser.
127.0.0.1 or ip-address of your server. For example198.162.12.52
It should show a message as it works! To change port address, you need to edit the configuration file at **/etc/apache2/ports.conf** and change the Listen 80 to your desired port number. After edit, you need to restart the apache2 server.



To restart apache2 web server -

```
sudo service apache2 restart
```

Install MySQL
MySQL is the database management solution that helps you to store and retrieve data in tables.
Install php5-mysql component.

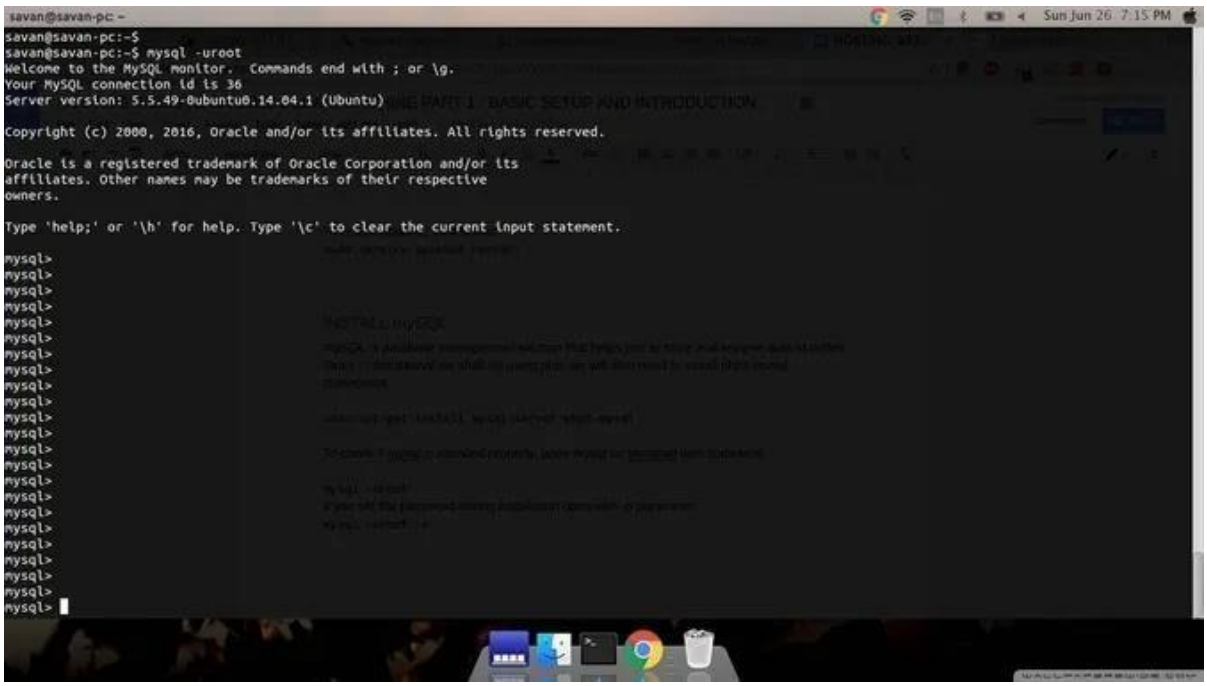
```
sudo apt-get install mysql-server php5-mysql
```

To check if MySQL is installed properly, open MySQL on terminal with command –

```
mysql -uroot
```

If you set the password during installation open with -p parameter –

```
mysql -uroot -p
```

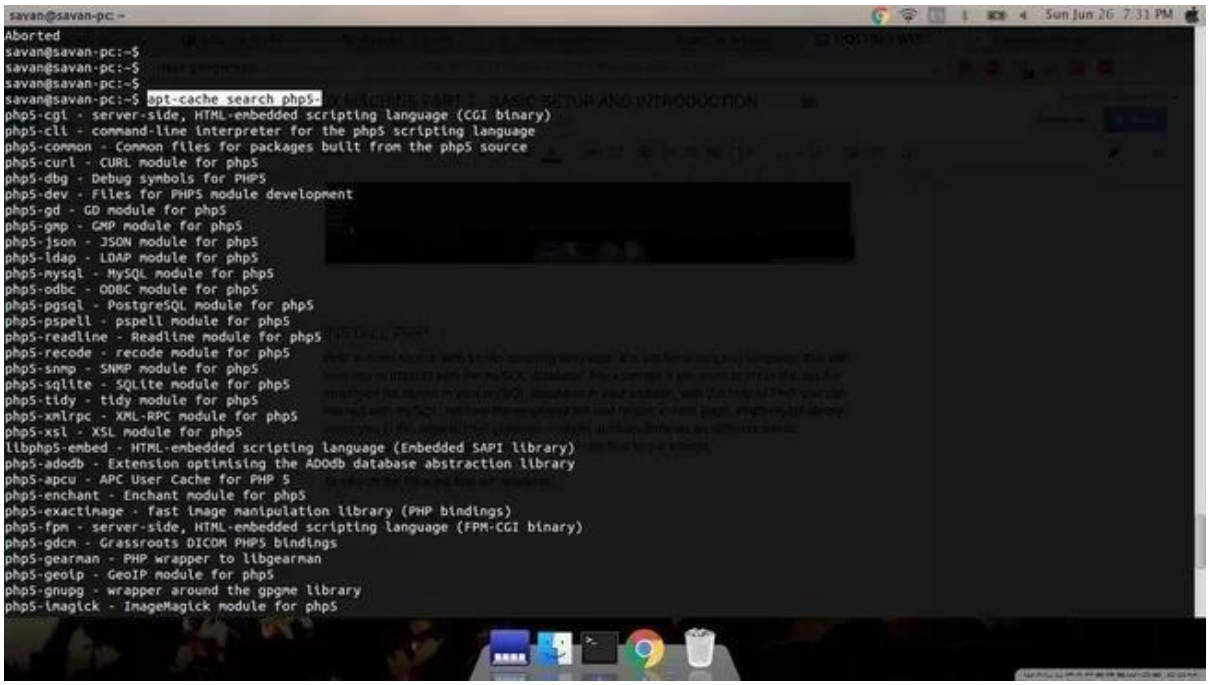


Install PHP

PHP is an open-source web server scripting language. It is a back-end scripting language that will help you to interact with the MySQL database. For example, if you want to show the tabular employee list stored in your MySQL database in your website, with the help of PHP you can interact with MySQL, retrieve the employee list and render in html page. php5-mysql library helps you in this regard. PHP provides multiple auxiliary libraries for different needs. Php5-mysql is one among them.

To search the libraries that are available.

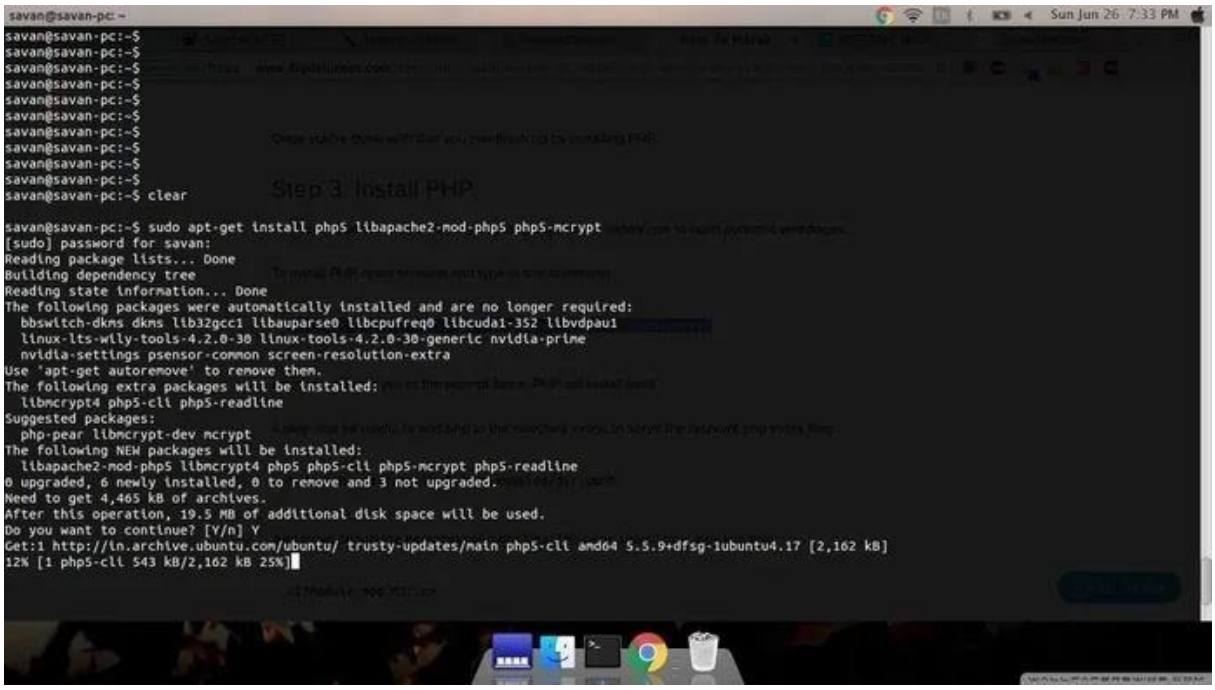
```
apt-cache search php5-
```



To install PHP and php5-mysql

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt

sudo apt-get install php5-sqlite
```

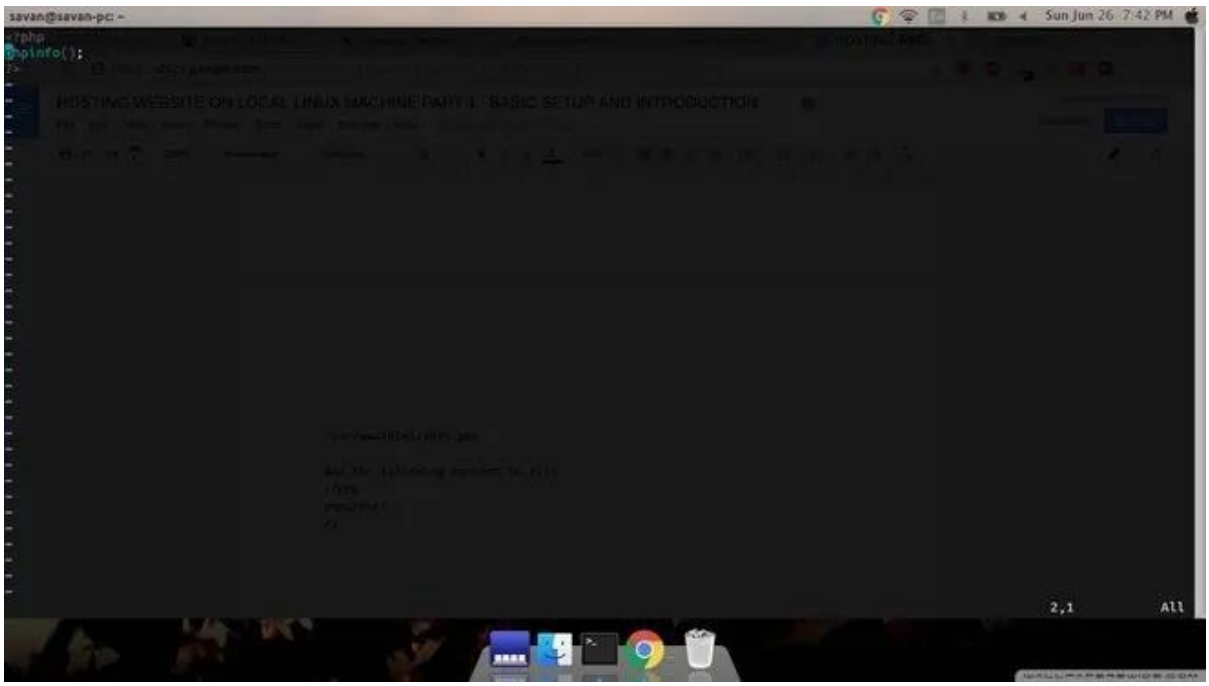


To check if PHP is installed correctly, make file **/var/www/html/info.php** and add the following content to this file -

```
<?php

phpinfo();

?>
```

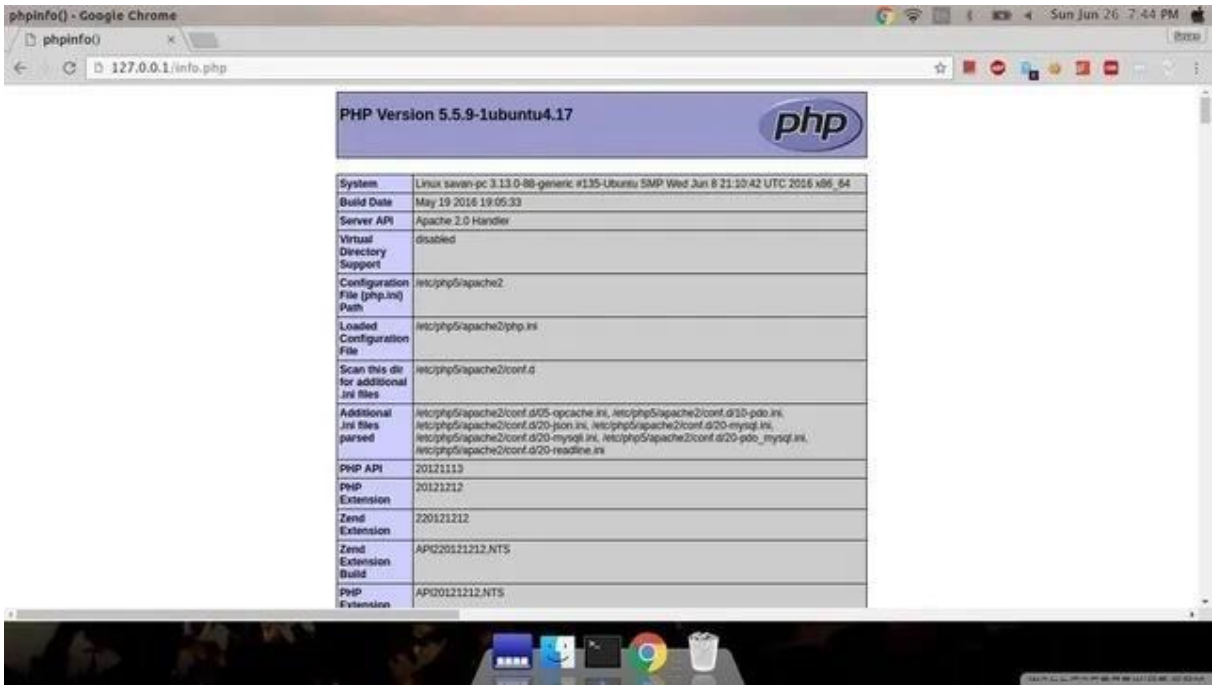


Restart apache2

```
sudo service apache2 restart
```

Open web browser and navigate to

127.0.0.1/info.php If you are using remote server replace ip with server's ip address. Upon success, you should see the following webpage –



11 b. Create two threads using pthreads. Here, main thread creates 5 other threads for 5 times and each new thread print "Hello World" message with its thread number.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

// The function to be executed by all threads
void *myNewThread(void *vargp){
   printf("Hello world\n");
}

void *myThreadFun(void *vargp)
{
 // Store the value argument passed to this thread
 int *myid = (int *)vargp;
   printf("%ld %ld ", pthread_self(), *myid);
   int maint = pthread_self();
 if(maint == (*myid) )
   {
      int i,j;
      printf("main thread encountered\n");
      pthread_t nid;
```

```
        for (i = 0; i < 5; i++)
            for(j =0;j<5;j++)
                pthread_create(&nid, NULL, myNewThread, (void *)&nid);
    }
}

int main()
{
 int i;
 pthread_t tid;

 // Let us create three threads
 for (i = 0; i < 2; i++)
        pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

 pthread_exit(NULL);
 return 0;
}
```

// gcc filename.c -lpthread -o user
./user

12. Using Socket APIs establish communication between remote and local processes.

Socket Server Example

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>

int main(int argc, char *argv[])
{
    int listenfd = 0, connfd = 0;
    struct sockaddr_in serv_addr;

    char sendBuff[1025];
    time_t ticks;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
```

```
    memset(&serv_addr, '0', sizeof(serv_addr));
  memset(sendBuff, '0', sizeof(sendBuff));

  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
  serv_addr.sin_port = htons(5000);

  bind(listenfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));

  listen(listenfd, 10);

  while(1)
  {
     connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);


     ticks = time(NULL);

     snprintf(sendBuff, sizeof(sendBuff), "%.24s\r\n", ctime(&ticks));

     write(connfd, sendBuff, strlen(sendBuff));



     close(connfd);

     sleep(1);

   }

}
```

Socket Client Example

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    int sockfd = 0, n = 0;
    char recvBuff[1024];
    struct sockaddr_in serv_addr;
    if(argc != 2)
   {
        printf("\n Usage: %s <ip of server> \n",argv[0]);
        return 1;
    }

    memset(recvBuff, '0',sizeof(recvBuff));
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
            printf("\n Error : Could not create socket \n");
        return 1;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000);

    if(inet_pton(AF_INET, argv[1], &serv_addr.sin_addr)<=0)
    {
        printf("\n inet_pton error occured\n");
        return 1;
    }

    if( connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
      printf("\n Error : Connect Failed \n");
      return 1;
    }

    while ( (n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)
    {
        recvBuff[n] = 0;
        if(fputs(recvBuff, stdout) == EOF)
        {
            printf("\n Error : Fputs error\n");
        }
    }

    if(n < 0)
    {
```

```
        printf("\n Read error \n");
    }

    return 0;
}


OUTPUT

$ ./newsy 127.0.0.1

Thu May 27 22:22:14 2021
```