UNIX Programming Lab Manual

PROGRAM-1

Check POSIX runtime limits

Write a C/C++ POSIX compliant program to check the following limits:

1. Number of clock ticks

2. Max number of child processes

3. Max path length

4. Max number of characters in a file name

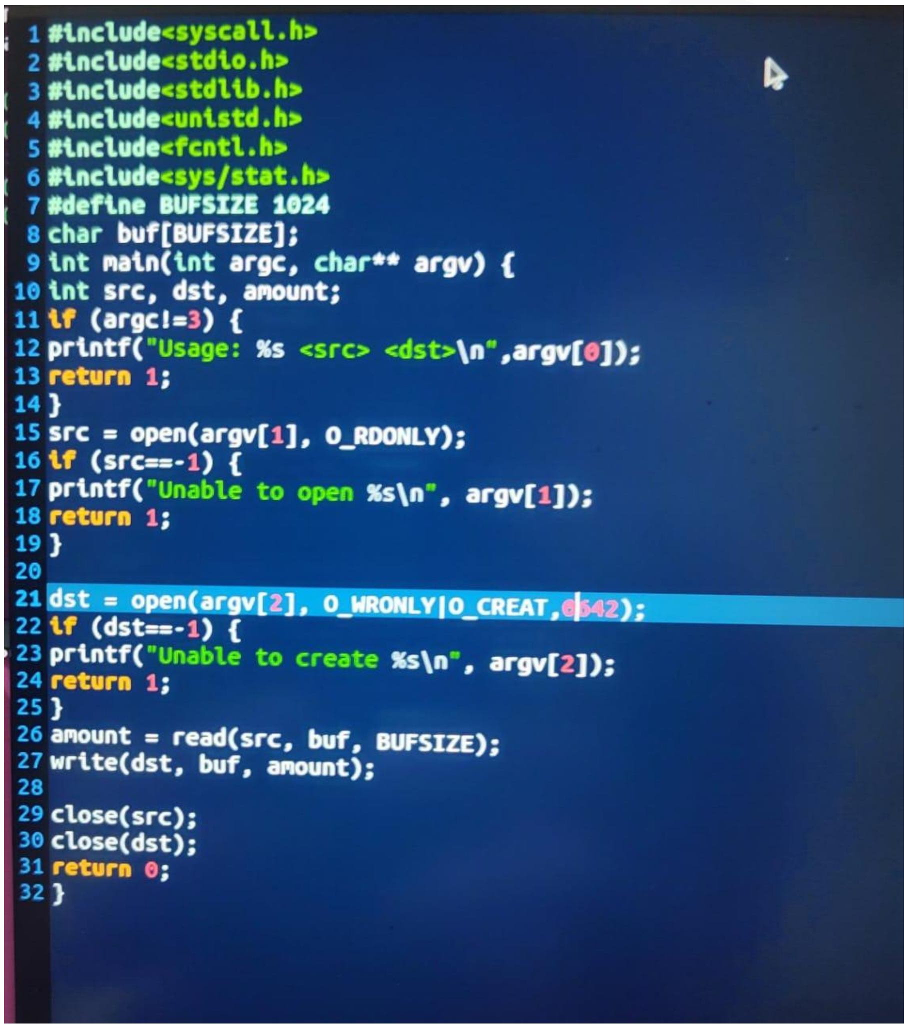
5. Max number of open files/processes

```
#include<stdio.h>
#include<unistd.h>
#include<limits.h>
//#define _POSIX_SOURCE
//#define _POSIX_C_SOURCE 199309L

int main()
{
printf("Runtime values\n");
printf("The max number of clock ticks : %ld\n",sysconf(_SC_CLK_TCK));
printf("The max runtime child processes : %ld\n",sysconf(_SC_CHILD_MAX));
printf("The max runtime path length :%ld\n",pathconf("usp1.cpp",_PC_PATH_MAX));
printf("The max characters in a file name :%ld\n",pathconf("usp1.cpp",_PC_NAME_MAX));
printf("The max number of opened files : %ld\n",sysconf(_SC_OPEN_MAX));
return 0;
}
```

2a. Copy of a file using system calls.

```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
    FILE *fptr1, *fptr2;
    char filename[100], c;

    printf("Enter the filename to open for reading \n");
    scanf("%s", filename);
    // Open one file for reading
    fptr1 = fopen(filename, "r");
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }

    printf("Enter the filename to open for writing \n");
    scanf("%s", filename);

    // Open another file for writing
    fptr2 = fopen(filename, "w");
    if (fptr2 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
      // Read contents from file
    c = fgetc(fptr1);
    while (c != EOF)
    {
        fputc(c, fptr2);
        c = fgetc(fptr1);
    }

    printf("\nContents copied to %s\n", filename);
    fclose(fptr1);
    fclose(fptr2);
    return 0;
}
```

2a - easier program



```
#include<syscall.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/stat.h>
#define BUFSIZE 1024
char buf[BUFSIZE];
int main(int argc, char** argv) {
int src, dst, amount;
if (argc!=3) {
printf("Usage: %s <src> <dst>\n",argv[0]);
return 1;
}
src = open(argv[1], O_RDONLY);
if (src==-1) {
printf("Unable to open %s\n", argv[1]);
return 1;
}

dst = open(argv[2], O_WRONLY|O_CREAT,0642);
if (dst==-1) {
printf("Unable to create %s\n", argv[2]);
return 1;
}
amount = read(src, buf, BUFSIZE);
write(dst, buf, amount);

close(src);
close(dst);
return 0;
}
```

2b Output the contents of its Environment list.

```
#include<stdio.h>
int main(int argc, char* argv[ ])
{
int i;
char **ptr;
extern char **environ;
for( ptr = environ; *ptr != 0; ptr++ ) /*echo all env strings*/
printf("%s\n", *ptr);
return 0;
}
```

3a.Emulate the UNIX ln command

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<string.h>
int main(int argc, char * argv[])
{
if(argc < 3 || argc > 4 || (argc == 4 && strcmp(argv[1],"-s")))
{
printf("Usage: ./a.out [-s] <org_file> <new_link>\n");
return 1;
}
if(argc == 4)
{
if((symlink(argv[2], argv[3])) == -1)
printf("Cannot create symbolic link\n") ;
else printf("Symbolic link created\n") ;
}
else
{
if((link(argv[1], argv[2])) == -1)
printf("Cannot create hard link\n") ;
else
printf("Hard link created\n") ;
}
return 0;
}
```

3b. Create a child from parent process using fork() and counter counts till 5 in both processes and display.

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
 #include <sys/types.h>
  #include <sys/wait.h>
int main()
{
for(int i=0;i<5;i++) // loop will run n times (n=5)
{
if(fork() == 0)
{
printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());
exit(0);
}
}
for(int i=0;i<5;i++) // loop will run n times (n=5)
wait(NULL);
}
```

4. Write a C program that illustrates 2 processes communicating using shared memory.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <string.h>
#include<stdio.h>
#include <errno.h>
int main(void) {
pid_t pid;
int *shared; /* pointer to the shm */
int shmid;
shmid = shmget(IPC_PRIVATE, sizeof(int), IPC_CREAT | 0666);
printf("Shared Memory ID=%u",shmid);
if (fork() == 0) { /* Child */
/* Attach to shared memory and print the pointer */
shared = shmat(shmid, (void *) 0, 0);
printf("Child pointer %d\n", *shared);
*shared=1;
printf("Child value=%d\n", *shared);
sleep(2);
printf("Child value=%d\n", *shared);
} else { /* Parent */
/* Attach to shared memory and print the pointer */
shared = shmat(shmid, (void *) 0, 0);
printf("Parent pointer %d\n", *shared);
printf("Parent value=%d\n", *shared);
sleep(1);
*shared=42;
printf("Parent value=%d\n", *shared);
sleep(5);
shmctl(shmid, IPC_RMID, 0);
}
}
```

PROGRAM-5

5. Write a C program that implements producer –consumer system with two processes using semaphores.

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

#define BUFFER_SIZE 5

int buffer[BUFFER_SIZE];
int in = 0, out = 0;
int producerPrompt = 0;

sem_t mutex, empty, full;

void *producer(void *arg) {
    int item;

    for (int i = 0; i < BUFFER_SIZE; i++) {
        printf("Enter item to produce: ");
        scanf("%d", &item);

        sem_wait(&empty);  // Wait for an empty slot in the buffer
        sem_wait(&mutex);  // Obtain exclusive access to the buffer
          // Lock a semaphore -> decreases the value

        buffer[in] = item;
        printf("Producer produced item: %d\n", item);
        in = (in + 1) % BUFFER_SIZE;
        sem_post(&mutex);  // Release exclusive access to the buffer
        sem_post(&full);   // Signal that a new item is available

         //unlock a semaphore -> increases the value

        producerPrompt = 1;  // Signal that producer has prompted

        while (producerPrompt) {
            // Wait until consumer consumes the item
        }
    }

    pthread_exit(NULL);
}

void *consumer(void *arg) {
    int item;

    for (int i = 0; i < BUFFER_SIZE; i++) {
        sem_wait(&full);   // Wait for an item to be available
        sem_wait(&mutex);  // Obtain exclusive access to the buffer

        item = buffer[out];
        printf("Consumer consumed item: %d\n", item);
        out = (out + 1) % BUFFER_SIZE;

        sem_post(&mutex);  // Release exclusive access to the buffer
        sem_post(&empty);  // Signal that an empty slot is available

        producerPrompt = 0;  // Signal that consumer has consumed the item
    }

    pthread_exit(NULL);
}

int main() {
    pthread_t producerThread, consumerThread;

    // Initialize semaphores
    sem_init(&mutex, 0, 1);     //int sem_init (sem_t *sem, int pshared, unsigned int value)
    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);

    // Create producer and consumer threads
    pthread_create(&producerThread, NULL, producer, NULL);
    pthread_create(&consumerThread, NULL, consumer, NULL);

    // Wait for threads to finish
    pthread_join(producerThread, NULL);
    pthread_join(consumerThread, NULL);

    // Destroy semaphores
    sem_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);

    return 0;
}
```

PROGRAM 5: Another version - GFG

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

int mutex = 1;

int full = 0;

int empty = 10, x = 0;

void producer()
{
    --mutex;

    ++full;

    --empty;

    x++;
    printf("\nProducer produces"
           "item %d",
           x);

    ++mutex;
}

void consumer()
{
    --mutex;
    --full;

    ++empty;
    printf("\nConsumer consumes "
           "item %d",
           x);
    x--;

    ++mutex;
}

int main()
{
    int n, i;
    printf("\n1. Press 1 for Producer"
           "\n2. Press 2 for Consumer"
           "\n3. Press 3 for Exit");


#pragma omp critical

    for (i = 1; i > 0; i++) {

        printf("\nEnter your choice:");
        scanf("%d", &n);

        switch (n) {
        case 1:

            if ((mutex == 1)
                && (empty != 0)) {
```

```
                producer();
            }

            else {
                printf("Buffer is full!");
            }
            break;

        case 2:

            if ((mutex == 1)
                && (full != 0)) {
                consumer();
            }

            else {
                printf("Buffer is empty!");
            }
            break;

        case 3:
            exit(0);
            break;
        }
    }
}
```

6. Demonstrate round robin scheduling algorithm and calculate average waiting time and average turn around time

```
#include<stdio.h>
int main()
{
int i, limit, total = 0, x, counter = 0, time_quantum;
int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10], temp[10];
float average_wait_time, average_turnaround_time;
printf("\nEnter Total Number of Processes:t=");
scanf("%d", &limit);
x = limit;
for(i = 0; i < limit; i++)
{
printf("\nEnter Details of Process[%d]\n", i + 1);
printf("Arrival Time:t=");
scanf("%d", &arrival_time[i]);printf("\nBurst Time:t");
scanf("%d", &burst_time[i]);
temp[i] = burst_time[i];
}
printf("\nEnter Time Quantum:t=");
scanf("%d", &time_quantum);
printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n");
for(total = 0, i = 0; x != 0;)
{
if(temp[i] <= time_quantum && temp[i] > 0)


{
total = total + temp[i];
temp[i] = 0;
counter = 1;
}
else if(temp[i] > 0)
{
temp[i] = temp[i] - time_quantum;
total = total + time_quantum;
}
if(temp[i] == 0 && counter == 1)
{
x--;
printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst_time[i], total - arrival_time[i],
total - arrival_time[i] - burst_time[i]);
wait_time = wait_time + total - arrival_time[i] - burst_time[i];
turnaround_time = turnaround_time + total - arrival_time[i];
counter = 0;
}
if(i == limit - 1)
{
i = 0;
}
else if(arrival_time[i + 1] <= total)
{
i++;
}
else{


i = 0;
}
}
average_wait_time = wait_time * 1.0 / limit;
```

```
average_turnaround_time = turnaround_time * 1.0 / limit;
printf("\n\nAverage Waiting Time:t=%f", average_wait_time);
printf("\nAvg Turnaround Time:t=%f\n", average_turnaround_time);return 0;
}
```

7.Implement Priority based scheduling algorithm and calculate average waiting time and turn around time

```
#include<stdio.h>
int main()
{
int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
printf("Enter Total Number of Process:");
scanf("%d",&n);
printf("\nEnter Burst Time and Priority\n");
for(i=0;i<n;i++)
{
printf("\nP[%d]\n",i+1);
printf("Burst Time:");
scanf("%d",&bt[i]);
printf("Priority:");
scanf("%d",&pr[i]);
p[i]=i+1;

}

for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(pr[j]<pr[pos])

pos=j;
}temp=pr[i];
pr[i]=pr[pos];
pr[pos]=temp;
temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;
temp=p[i];
p[i]=p[pos];
p[pos]=temp;
}
wt[0]=0;

for(i=1;i<n;i++)
{
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
total+=wt[i];
}
avg_wt=total/n;
total=0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{ tat[i]=bt[i]+wt[i]; //calculate turnaround time
total+=tat[i];

printf("\nP[%d]\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
}
avg_tat=total/n; //average turnaround time
printf("\n\nAverage Waiting Time=%d",avg_wt);
printf("\n\nAverage Waiting Time=%d",avg_wt);
printf("\nAverage Turnaround Time=%d\n",avg_tat);
```

```
return 0;
}
```

8.Act as sender to send data in message queues and receiver that reads data from message queue

Sender

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MAX 10

struct mesg_buffer {
 long mesg_type;
 char mesg_text[100];
} message;
int main()
{
 key_t key;
 int msgid;
 key = ftok("progfile", 65);
 msgid = msgget(key, 0666 | IPC_CREAT);
 message.mesg_type = 1;
 printf("Write Data : ");
 fgets(message.mesg_text,MAX,stdin);
 msgsnd(msgid, &message, sizeof(message), 0);
 printf("Data sent is : %s \n", message.mesg_text);
 return 0;
}
```

Receiver

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>


struct mesg_buffer {
 long mesg_type;
 char mesg_text[100];
} message;
int main()
{
 key_t key;
 int msgid;
 key = ftok("progfile", 65);
 msgid = msgget(key, 0666 | IPC_CREAT);
 msgrcv(msgid, &message, sizeof(message), 1, 0);
 printf("Data Received is : %s \n", message.mesg_text);
 msgctl(msgid, IPC_RMID, NULL);
 return 0;
}
```

9.Write a program where parent writes message to a pipe and child reads message from the pipe

Parent

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<stdlib.h>
#include <sys/stat.h>
#define MAXSIZE 10
#define FIFO_NAME "myfifo"
int main()
{
int fifoid;
int fd, n;
char *w;
system("clear");

w=(char *)malloc(sizeof(char)*MAXSIZE);
int open_mode=O_WRONLY;
fifoid=mkfifo(FIFO_NAME, 0755);
if(fifoid==-1)
{
printf("\nError: Named pipe cannot be Created\n");
exit(0);
}
if( (fd=open(FIFO_NAME, open_mode)) < 0 )
{
printf("\nError: Named pipe cannot be opened\n");
exit(0);
}
while(1)
{
printf("\nProducer :");
fflush(stdin);
read(0, w, MAXSIZE);
n=write(fd, w, MAXSIZE);
if(n > 0)
printf("\nProducer sent: %s", w);
}
}
```

Child

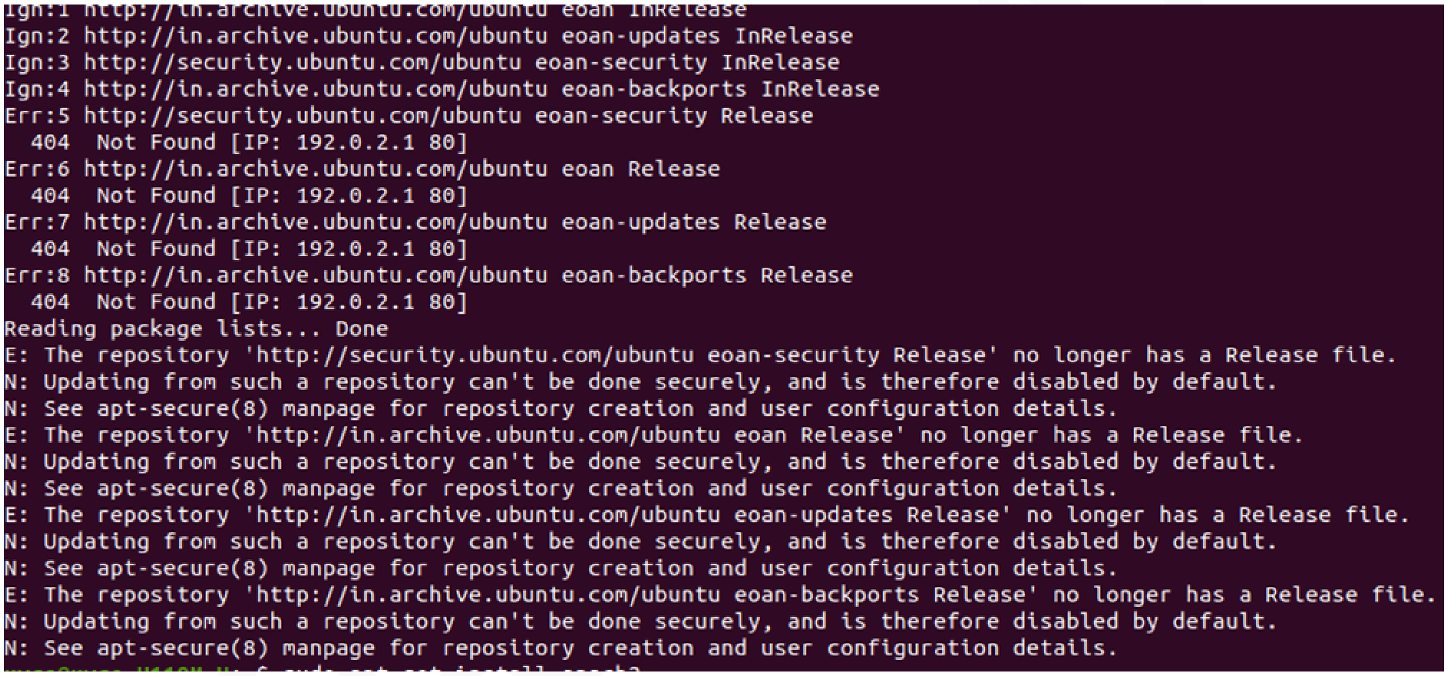
```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<stdlib.h>
#include <sys/stat.h>
#define MAXSIZE 10

#define FIFO_NAME "myfifo"
int main()
{
int fifoid;
int fd, n;
char *r;
system("clear");
r=(char *)malloc(sizeof(char)*MAXSIZE);
int open_mode=O_RDONLY;
if( (fd=open(FIFO_NAME, open_mode)) < 0 )
{
printf("\nError: Named pipe cannot be opened\n");
exit(0);
}
while(1)
{
n=read(fd, r, MAXSIZE);
if(n > 0)
printf("\nConsumer read: %s", r);
}
}
```

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

Installing apache2 and php-7.3

Note: The commands should be executed in the **terminal**.

Step 1: run the command ‘sudo apt update’ without quotes.



```
Ign:1 http://in.archive.ubuntu.com/ubuntu eoan InRelease
Ign:2 http://in.archive.ubuntu.com/ubuntu eoan-updates InRelease
Ign:3 http://security.ubuntu.com/ubuntu eoan-security InRelease
Ign:4 http://in.archive.ubuntu.com/ubuntu eoan-backports InRelease
Err:5 http://security.ubuntu.com/ubuntu eoan-security Release
  404  Not Found [IP: 192.0.2.1 80]
Err:6 http://in.archive.ubuntu.com/ubuntu eoan Release
  404  Not Found [IP: 192.0.2.1 80]
Err:7 http://in.archive.ubuntu.com/ubuntu eoan-updates Release
  404  Not Found [IP: 192.0.2.1 80]
Err:8 http://in.archive.ubuntu.com/ubuntu eoan-backports Release
  404  Not Found [IP: 192.0.2.1 80]
Reading package lists... Done
E: The repository 'http://security.ubuntu.com/ubuntu eoan-security Release' no longer has a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
E: The repository 'http://in.archive.ubuntu.com/ubuntu eoan Release' no longer has a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
E: The repository 'http://in.archive.ubuntu.com/ubuntu eoan-updates Release' no longer has a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
E: The repository 'http://in.archive.ubuntu.com/ubuntu eoan-backports Release' no longer has a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

If you get the above error:

go to this website:
https://www.digitalocean.com/community/questions/unable-to-apt-update-my-ubuntu-19-04

and run the commands given in the first answer.

The commands are:

1. sudo sed -i -re 's/([a-z]{2}\.)?archive.ubuntu.com|security.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list

2. sudo apt-get update && sudo apt-get dist-upgrade

It will install the legacy packages required.

Incase you do not get the error in the above picture, continue from step 2.

Step 2: Installing apache2

Run the commands:

1. sudo apt update

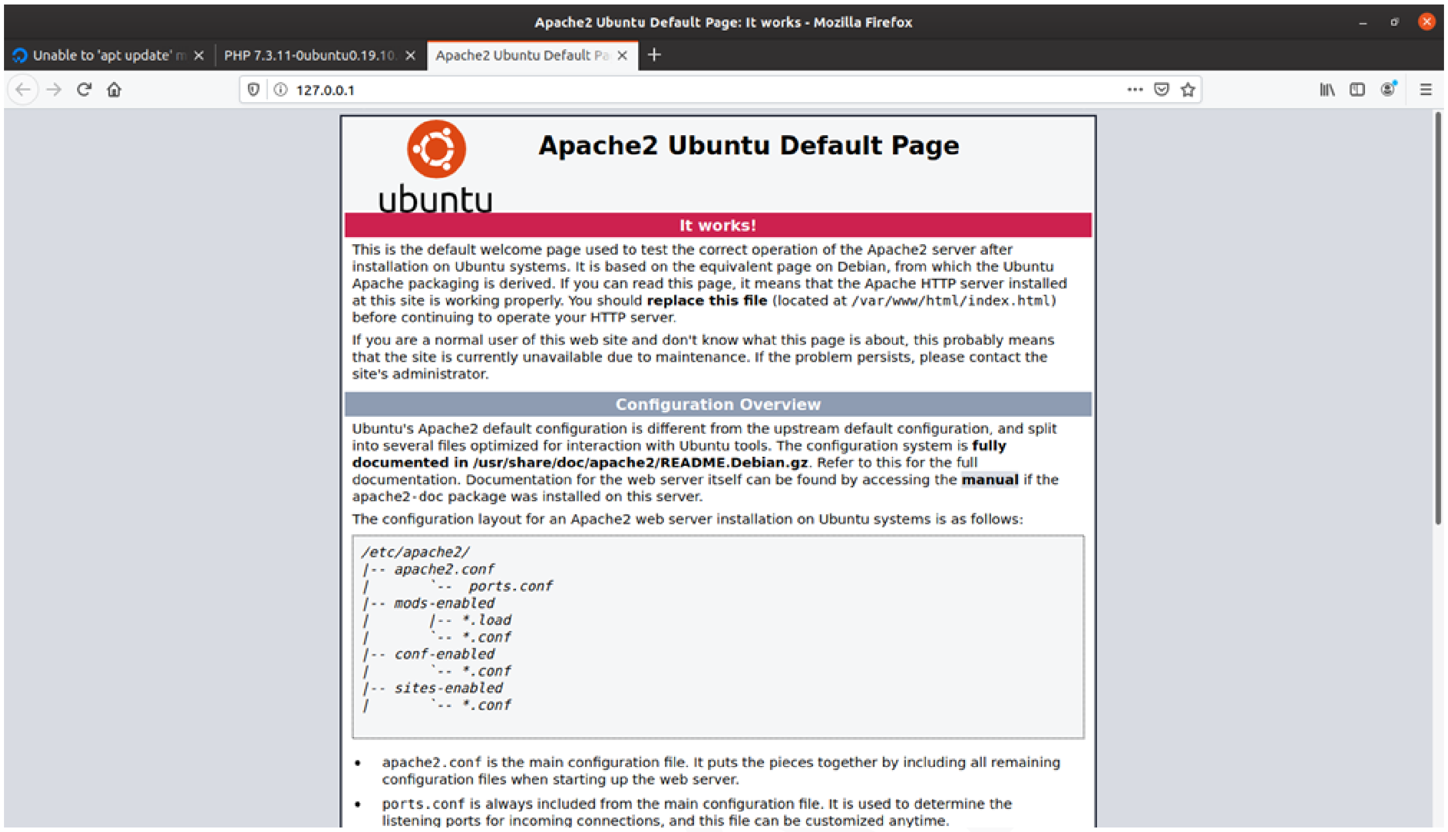
2. sudo apt-get install apache2

To check if apache2 has been installed properly:

1. Execute the command –> sudo service apache2 restart

2. Open a browser

2. Type 127.0.0.1 in the address bar on the window.


Apache2 Ubuntu Default Page: It works - Mozilla Firefox
Unable to 'apt update'
PHP 7.3.11-0ubuntu0.19.10
Apache2 Ubuntu Default Pa
127.0.0.1
ubuntu

Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.

Your screen should be as the above image.

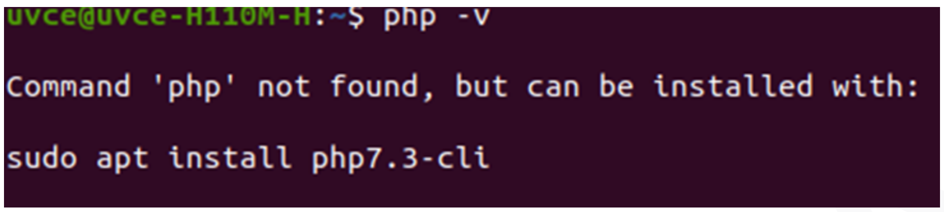
Step 3: Install MySQL server

Execute the command: sudo apt-get install mysql-server

Step 4: Install PHP

1. Run the command –> php -v

2. If the php is installed it will return the version number.



```
uvce@uvce-H110M-H:~$ php -v
Command 'php' not found, but can be installed with:
sudo apt install php7.3-cli
```

3. If not, it will return an error and at the end will give a command to install the required file.

4. Copy, paste and run the command and give ‘y’ without quotes when prompted.

5. Kindly note down the version number of PHP.

Step 5: Install necessary files to to connect PHP to Apache2 and MySQL

1. Run the command > sudo apt-get install php libapache2-mod-php php-mysql -y

2. After installation make sure apache2 and mysql are running.

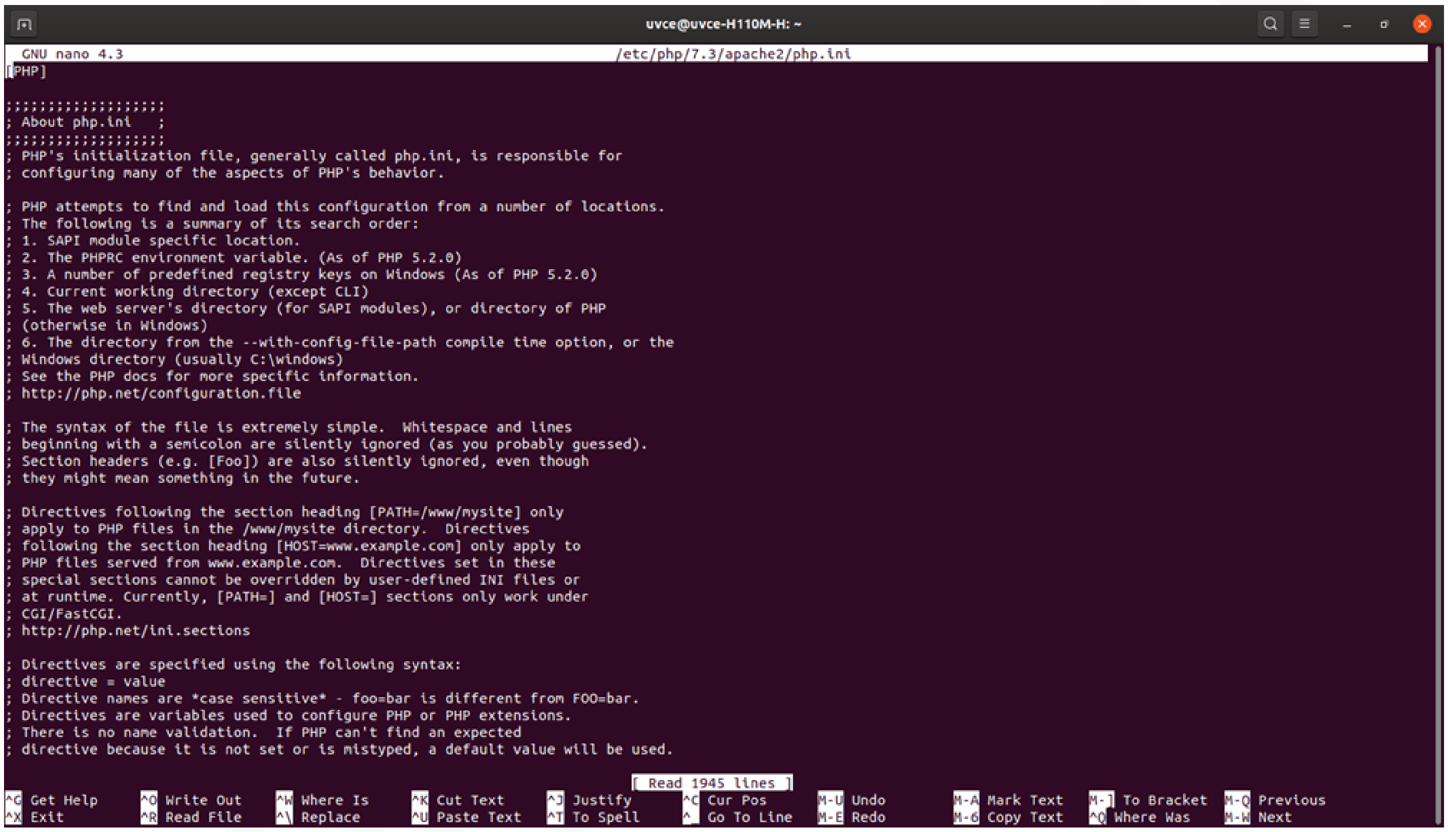
3. To check and enable that execute the following:

```
sudo systemctl start apache2
sudo systemctl enable apache2
sudo systemctl start mysql
sudo systemctl enable mysql
```

4. Run the command **sudo nano *etc/*php/ <php_version_number>/apache2/php.ini**

For example: If your PHP version number is 7.3, the above command will be:

sudo nano etc/php/7.3/apache2/php.ini



```
uvce@uvce-H110M-H: ~
  GNU nano 4.3                                          /etc/php/7.3/apache2/php.ini
[PHP]

;;;;;;;;;;;;;;;;;;;
; About php.ini   ;
;;;;;;;;;;;;;;;;;;;
; PHP's initialization file, generally called php.ini, is responsible for
; configuring many of the aspects of PHP's behavior.

; PHP attempts to find and load this configuration from a number of locations.
; The following is a summary of its search order:
; 1. SAPI module specific location.
; 2. The PHPRC environment variable. (As of PHP 5.2.0)
; 3. A number of predefined registry keys on Windows (As of PHP 5.2.0)
; 4. Current working directory (except CLI)
; 5. The web server's directory (for SAPI modules), or directory of PHP
; (otherwise in Windows)
; 6. The directory from the --with-config-file-path compile time option, or the
; Windows directory (usually C:\windows)
; See the PHP docs for more specific information.
; http://php.net/configuration.file

; The syntax of the file is extremely simple.  Whitespace and lines
; beginning with a semicolon are silently ignored (as you probably guessed).
; Section headers (e.g. [Foo]) are also silently ignored, even though
; they might mean something in the future.

; Directives following the section heading [PATH=/www/mysite] only
; apply to PHP files in the /www/mysite directory.  Directives
; following the section heading [HOST=www.example.com] only apply to
; PHP files served from www.example.com.  Directives set in these
; special sections cannot be overridden by user-defined INI files or
; at runtime. Currently, [PATH=] and [HOST=] sections only work under
; CGI/FastCGI.
; http://php.net/ini.sections

; Directives are specified using the following syntax:
; directive = value
; Directive names are *case sensitive* - foo=bar is different from FOO=bar.
; Directives are variables used to configure PHP or PHP extensions.
; There is no name validation.  If PHP can't find an expected
; directive because it is not set or is mistyped, a default value will be used.

                                                            [ Read 1945 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text  M-] To Bracket M-Q Previous
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text  ^Q Where Was   M-W Next
```

5. The command will return a file something like above. This just ensures that you have installed PHP correctly.

6. Run the command -> **sudo nano *var/*www/html/info.php**

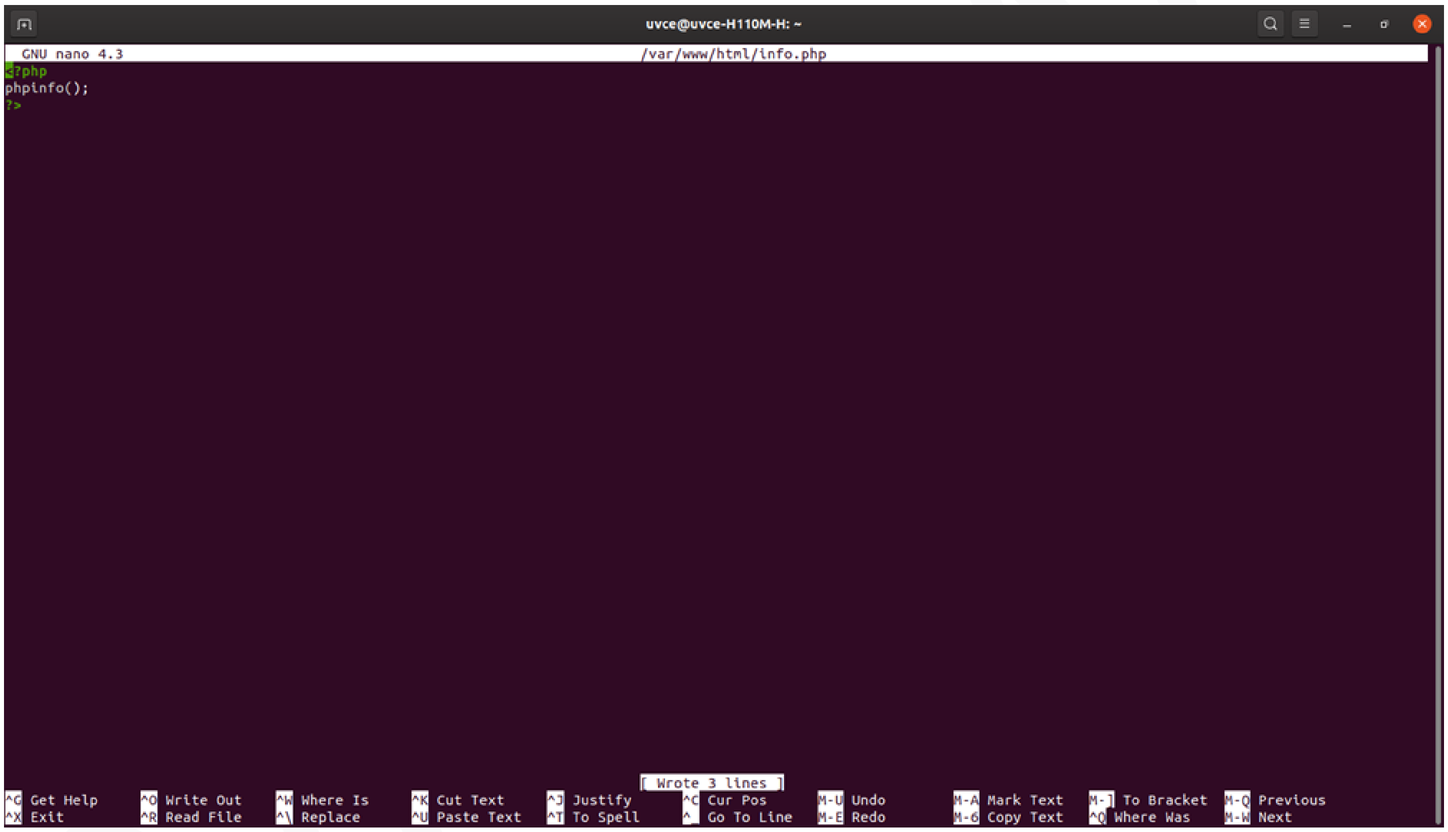
This will create a new file named info.php. Type the following code in the file:

<?php

phpinfo();

?>

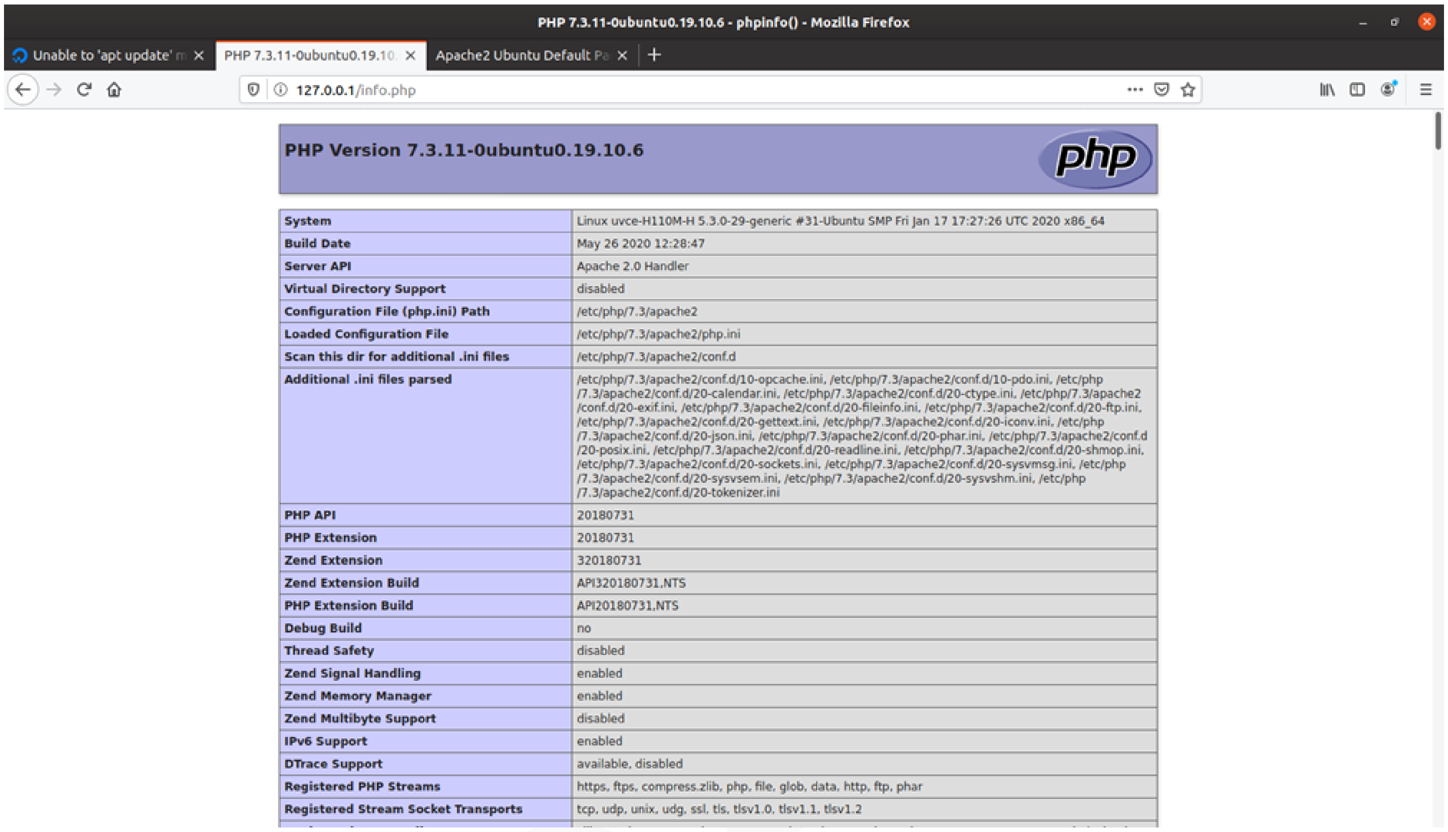
Press Ctrl+S and Ctrl+X to save and exit the file.



```
uvce@uvce-H110M-H: ~
  GNU nano 4.3                                          /var/www/html/info.php
<?php
phpinfo();
?>

                                              [ Wrote 3 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text  M-] To Bracket  M-Q Previous
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo      M-6 Copy Text  ^Q Where Was    M-W Next
```

Step 6: Check if PHP is working on Apache2

Go to web browser and type the URL -> 1**27.0.0.1/info.php**


PHP 7.3.11-0ubuntu0.19.10.6 - phpinfo() - Mozilla Firefox
Unable to 'apt update' m
PHP 7.3.11-0ubuntu0.19.10.
Apache2 Ubuntu Default Pa
127.0.0.1/info.php

PHP Version 7.3.11-0ubuntu0.19.10.6

System	Linux uvce-H110M-H 5.3.0-29-generic #31-Ubuntu SMP Fri Jan 17 17:27:26 UTC 2020 x86_64
Build Date	May 26 2020 12:28:47
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.3/apache2
Loaded Configuration File	/etc/php/7.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.3/apache2/conf.d
Additional .ini files parsed	/etc/php/7.3/apache2/conf.d/10-opcache.ini, /etc/php/7.3/apache2/conf.d/10-pdo.ini, /etc/php/7.3/apache2/conf.d/20-calendar.ini, /etc/php/7.3/apache2/conf.d/20-ctype.ini, /etc/php/7.3/apache2/conf.d/20-exif.ini, /etc/php/7.3/apache2/conf.d/20-fileinfo.ini, /etc/php/7.3/apache2/conf.d/20-ftp.ini, /etc/php/7.3/apache2/conf.d/20-gettext.ini, /etc/php/7.3/apache2/conf.d/20-iconv.ini, /etc/php/7.3/apache2/conf.d/20-json.ini, /etc/php/7.3/apache2/conf.d/20-phar.ini, /etc/php/7.3/apache2/conf.d/20-posix.ini, /etc/php/7.3/apache2/conf.d/20-readline.ini, /etc/php/7.3/apache2/conf.d/20-shmop.ini, /etc/php/7.3/apache2/conf.d/20-sockets.ini, /etc/php/7.3/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.3/apache2/conf.d/20-sysvsem.ini, /etc/php/7.3/apache2/conf.d/20-sysvshm.ini, /etc/php/7.3/apache2/conf.d/20-tokenizer.ini
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,NTS
PHP Extension Build	API20180731,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2

You should get a window like below which shows the details about PHP.

Congratulations. You have successfully installed the required modules for Question 10.

11(a). Create two threads using pthread, where both thread count until 100 and joins later.

```
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
#include<stdlib.h>
void* myturn(void *arg)
{


 for(int i=1;i<=10;i++)
 {
 sleep(1);
 printf("process 1: i=%d\n",i);
 }
 return NULL;
}
void yourturn()
{
 for(int i=1;i<=10;i++)
 {
 sleep(2);
 printf("process 2: j=%d\n",i);
 }
}
int main()
{
 pthread_t newthread;
 pthread_create(&newthread,NULL,myturn,NULL);
 yourturn();
 pthread_join(newthread,NULL);
 return 0;
}
```

11b 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.

Question is Unclear so decide which code to refer (This one or manual)

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

#define NUM_THREADS 5

void *myNewThread(void *vargp) {
    pthread_t tid = pthread_self();
    printf("Hello World from thread %ld\n", tid);
   pthread_exit(NULL);
}

void *myThreadFun(void *vargp) {
    int i;
    pthread_t tid[NUM_THREADS];

    for (i = 0; i < NUM_THREADS; i++) {
    for(int j =0;j<5;j++)
        pthread_create(&tid[i], NULL, myNewThread, NULL);
    }

    for (i = 0; i < NUM_THREADS; i++) {
        pthread_join(tid[i], NULL);
    }

    pthread_exit(NULL);
}
```

```
int main() {
    pthread_t t1,t2;

    pthread_create(&t1, NULL, myThreadFun, NULL);
  //  pthread_create(&t2, NULL, myThreadFun, NULL);

    pthread_join(t1, NULL);
    //pthread_join(t2, NULL);
    return 0;
}
```

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

Server

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

#define PORT 8000
#define BUFFER_SIZE 1024

int main()
{
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[BUFFER_SIZE];

    // Create socket
    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd < 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Set address parameters
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // Bind socket to specified address and port
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(server_fd, 3) < 0)
    {
        perror("listen failed");
        exit(EXIT_FAILURE);
    }

    // Accept incoming connection
    new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t *)&addrlen);
    if (new_socket < 0)
    {
        perror("accept failed");
        exit(EXIT_FAILURE);
    }

    // Read client message into buffer
    read(new_socket, buffer, BUFFER_SIZE);
    printf("Client message: %s\n", buffer);

    // Send response to client
    const char *response = "Hello from server";
```

```
    write(new_socket, response, strlen(response));

    // Close sockets
    close(new_socket);
    close(server_fd);

    return 0;
}
```

Client

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

#define PORT 8000
#define BUFFER_SIZE 1024

int main()
{
    int sock;
    struct sockaddr_in serv_addr;
    char buffer[BUFFER_SIZE];

    // Create socket
    sock = socket(AF_INET, SOCK_STREAM, 0);
```

```
if (sock < 0)
{
    perror("socket failed");
    exit(EXIT_FAILURE);
}

// Set server address parameters
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(PORT);

// Connect to server
if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
{
    perror("connect failed");
    exit(EXIT_FAILURE);
}

// Get message from user for server
printf("Enter a message for the server: ");
fgets(buffer, BUFFER_SIZE, stdin);

// Send message to server
write(sock, buffer, strlen(buffer));

// Clear buffer
memset(buffer, 0, BUFFER_SIZE);

// Read server response into buffer
read(sock, buffer, BUFFER_SIZE);
printf("Server response: %s\n", buffer);
```

```
    // Close socket
    close(sock);

    return 0;
}
```