

1. Check the following limits: No. of clock ticks, Max. no. of child processes, Max. path length, Max. no. Of characters in a file name, Max. no. of open files/ process

```
#include<stdio.h>

#include<unistd.h>

#include<limits.h>

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("prg1.c",_PC_PATH_MAX));

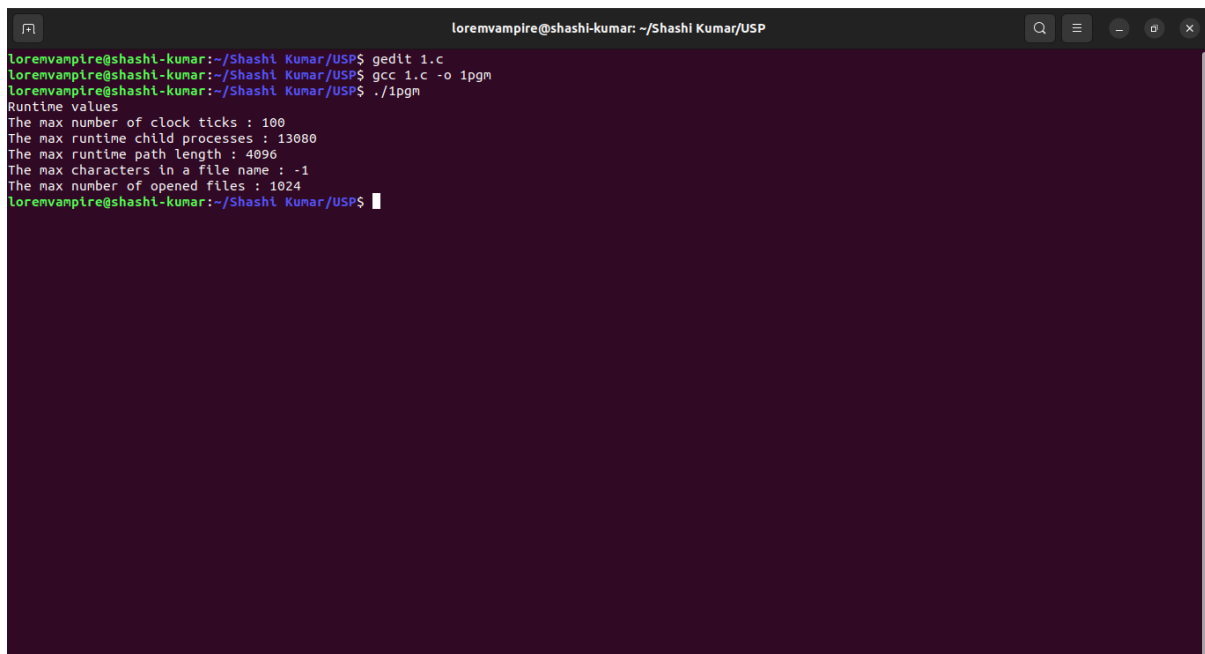
printf("The max characters in a file name : %ld\n",pathconf("prg1.c",_PC_NAME_MAX));

printf("The max number of opened files : %ld\n",sysconf(_SC_OPEN_MAX));

return 0;

}
```

## OUTPUT:

A terminal window with a dark purple background. The title bar reads 'lorenvampire@shashi-kumar: ~/Shashi Kumar/USP'. The terminal shows the following commands and output:

```
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 1.c
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 1.c -o 1pgm
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ ./1pgm
Runtime values
The max number of clock ticks : 100
The max runtime child processes : 130080
The max runtime path length : 4096
The max characters in a file name : -1
The max number of opened files : 1024
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$
```

2a. Write C program that makes copy of a file using standard I/O and System calls.

```
#include<syscall.h>

#include<stdio.h>

#include<stdlib.h>

#define BUFSIZE 1024

char buf[BUFSIZE];
```

```

int main(int argc, char** argv) {
    int src, dst, amount;
    if (argc!=3){
        printf("Usage: cp <src> <dst>\n");
        return 1;
    }
    src = open(argv[1]);
    if (src==-1) {
        printf("Unable to open %s\n", argv[1]);
        return 1;
    }
    creat(argv[2]);
    dst = open(argv[2]); if (dst==-1) {
        printf("Unable to create %s\n", argv[2]);
        return 1;
    }
    while ((amount = read(src, buf, BUFSIZE))>0) {
        write(dst, buf, amount);
    }
    close(src);
    close(dst);
    return 0;
}

```

## OUTPUT:

```

loremvampire@shashi-kumar: ~/Shashi Kumar/USP
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 2a.c
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 2a.c -o 2apgm
2a.c: In function 'main':
2a.c:12:7: warning: implicit declaration of function 'open'; did you mean 'popen'? [-Wimplicit-function-declaration]
12 | src = open(argv[1]);
    |         ^~~~~
2a.c:17:1: warning: implicit declaration of function 'creat' [-Wimplicit-function-declaration]
17 | creat(argv[2]);
    | ^~~~~
2a.c:22:18: warning: implicit declaration of function 'read'; did you mean 'freadd'? [-Wimplicit-function-declaration]
22 | while ((amount = read(src, buf, BUFSIZE))>0)
    |                  ^~~~~
2a.c:24:1: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
24 | write(dst, buf, amount);
    | ^~~~~
2a.c:26:1: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
26 | close(src);
    | ^~~~~
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./2apgm
Usage: cp <src> <dst>
loremvampire@shashi-kumar:~/Shashi Kumar/USP$

```

## 2.b. 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++ )
```

```
printf("%s\n", *ptr);
```

```
return 0;
```

```
}
```

## OUTPUT:

```
lorenvampire@shashl-kumar: ~/Shashl Kumar/USP
lorenvampire@shashl-kumar:~/Shashl Kumar/USP$ gedit 2b.c
lorenvampire@shashl-kumar:~/Shashl Kumar/USP$ gcc 2b.c -o 2bpgm
lorenvampire@shashl-kumar:~/Shashl Kumar/USP$ ./2bpgm
SHELL=/bin/bash
SESSION_MANAGER=local/shashl-kumar:@/tmp/.ICE-unix/1518,unix/shashl-kumar:/tmp/.ICE-unix/1518
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LANGUAGE=en_IN:en
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@xib:ibus
DESKTOP_SESSION=ubuntu
GTK_MODULES=gall:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/lorenvampire/Shashl Kumar/USP
LOGNAME=lorenvampire
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1518
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.MB1051
IM_CONFIG_CHECK_ENV=1
HOME=/home/lorenvampire
USERNAME=lorenvampire
IM_CONFIG_PHASE=1
LANG=en_IN
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pl=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:ml=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lzh=01;31:*.lzh=01;31:*.lzh=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.7z=01;31:*.zip=01;31:*.z=01;31:*.d=01;31:*.gz=01;31:*.l=01;31:*.l=01;31:*.l=01;31:*.xz=01;31:*.zst=01;31:*.tzt=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpt=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.w=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pex=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.m4=00;36:*.m4d=00;36:*.m4p=00;36:*.m4v=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6880
WAYLAND_DISPLAY=wayland-0
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/bb23e1f5_6f4b_4d82_8715_951055bf31a1
GNOME_SETUP_DISPLAY=:1
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=lorenvampire
GNOME_TERMINAL_SERVICE=:1.105
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=b8f059752534877b8537081a646f9982
XDG_RUNTIME_DIR=/run/user/1000
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=b8f059752534877b8537081a646f9982
_=./2bpgm
lorenvampire@shashl-kumar:~/Shashl Kumar/USP$
```

### 3. a. 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

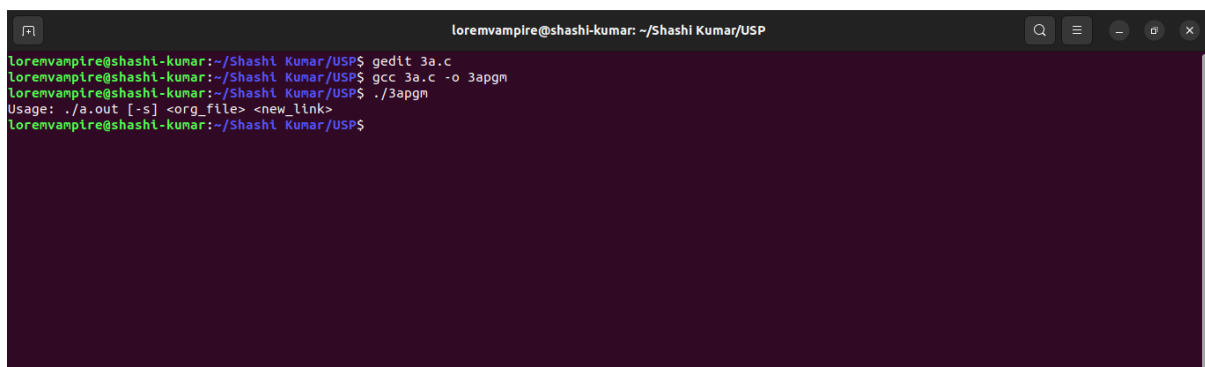
printf("Hard link created\n");

}

return 0;

}
```

### OUTPUT:

A terminal window with a dark background and light green text. The window title is 'loremvampire@shashi-kumar: ~/Shashi Kumar/USP'. The terminal shows the following commands and output: 1. 'gedit 3a.c' - opens a file in the editor. 2. 'gcc 3a.c -o 3apgm' - compiles the program. 3. './3apgm' - runs the program. 4. The program prints 'Usage: ./a.out [-s] <org\_file> <new\_link>' and returns to the prompt. The terminal window has standard Linux window controls (search, menu, back, forward, close) in the top right corner.

```
loremvampire@shashi-kumar: ~/Shashi Kumar/USP
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 3a.c
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 3a.c -o 3apgm
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./3apgm
Usage: ./a.out [-s] <org_file> <new_link>
loremvampire@shashi-kumar:~/Shashi Kumar/USP$
```

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

```
#include<stdio.h>

#include<stdlib.h>

int main()

{

for(int i=0;i<5;i++)

{

if(fork() == 0)

{

printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());

exit(0);

}

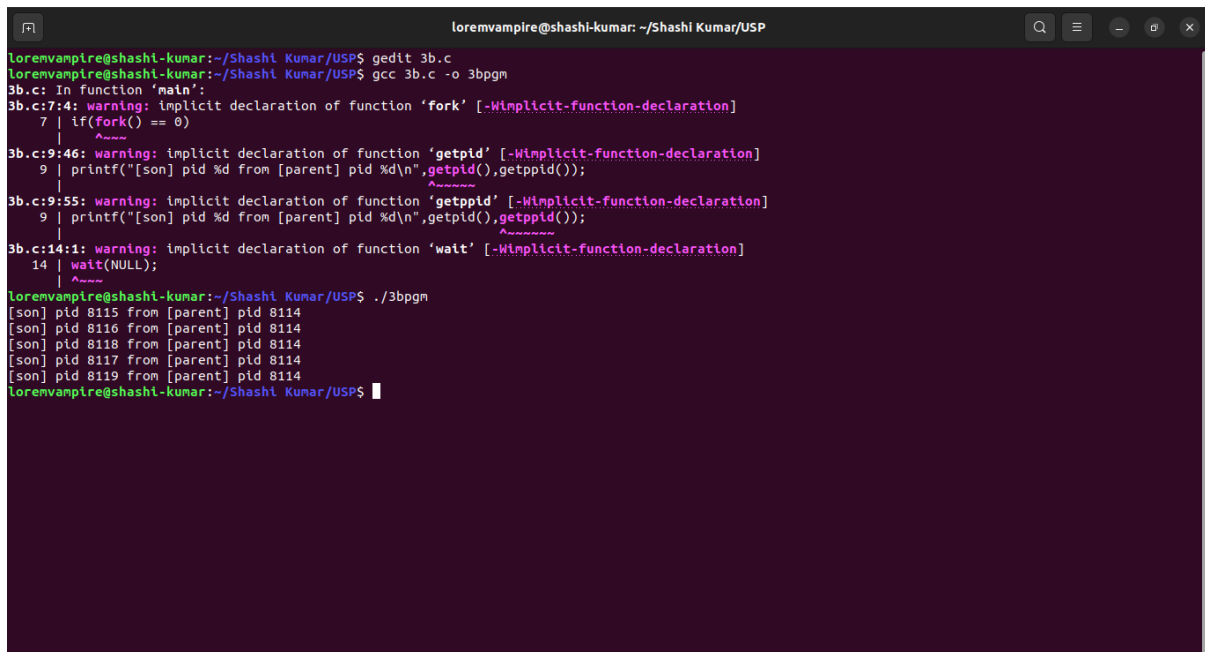
}

for(int i=0;i<5;i++)

wait(NULL);

}
```

## OUTPUT:



```
loremvampire@shashi-kumar: ~/Shashi Kumar/USP
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 3b.c
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 3b.c -o 3bpgm
3b.c: In function 'main':
3b.c:7:4: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
7 | if(fork() == 0)
  | ~~~~~
3b.c:9:46: warning: implicit declaration of function 'getpid' [-Wimplicit-function-declaration]
9 | printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());
  | ~~~~~
3b.c:9:55: warning: implicit declaration of function 'getppid' [-Wimplicit-function-declaration]
9 | printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());
  | ~~~~~
3b.c:14:11: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]
14 | wait(NULL);
    | ~~~~~
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./3bpgm
[son] pid 8115 from [parent] pid 8114
[son] pid 8116 from [parent] pid 8114
[son] pid 8118 from [parent] pid 8114
[son] pid 8117 from [parent] pid 8114
[son] pid 8119 from [parent] pid 8114
loremvampire@shashi-kumar:~/Shashi Kumar/USP$
```

4. Illustrate two processes communicating using shared memory.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <stdio.h>

int main(void) {
    pid_t pid;
    int *shared;
    int shmid;

    shmid = shmget(IPC_PRIVATE, sizeof(int), IPC_CREAT | 0666);
    printf("Shared Memory ID=%u", shmid);
    if (fork() == 0) {
        shared = shmat(shmid, (void *) 0, 0);
        printf("Child pointer %p\n", shared);
        *shared=1;
        printf("Child value=%d\n", *shared);
        sleep(2);
        printf("Child value=%d\n", *shared);
    } else {
        shared = shmat(shmid, (void *) 0, 0);
        printf("Parent pointer %p\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);
    }
}
```

## OUTPUT:

```
lorenvampire@shashi-kumar: ~/Shashi Kumar/USP
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 4.c
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 4.c -o 4pgm
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ ./4pgm
Shared Memory ID=11Parent pointer 0x7f2f3e040000
Parent value=0
Shared Memory ID=11Child pointer 0x7f2f3e040000
Child value=1
Parent value=42
Child value=42
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$
```

5. Demonstrate producer and consumer problem using semaphores.

### Producer:

```
#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<stdlib.h>

#define MAXSIZE 10

#define FIFO_NAME "myfifo"

int main()

{

int fifoid; int fd, n;

char *w;

int open_mode;

system("clear");

w=(char*)malloc(sizeof(char)*MAXSIZE);

open_mode=O_WRONLY;

fifoid=mknfif(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);
}
}

```

### **Consumer:**

```

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.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);
}
}

```

## OUTPUT:

```

Terminal 1 (Producer):
shashi
Producer :
Producer sent: shashi

kumar
Producer :
Producer sent: kumar

uvce
Producer :
Producer sent: uvce

Terminal 2 (Consumer):
Consumer read: shashi
Consumer read: kumar
Consumer read: uvce

```

6. Demonstrate round robin scheduling algorithm and calculates average waiting time and average turnaround 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("Burst Time:\t");

scanf("%d", &burst_time[i]);

temp[i] = burst_time[i];

}

printf("\nEnter Time Quantum:\t");

scanf("%d", &time_quantum);

printf("\nProcess ID\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 %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("\n\nAvg Turnaround Time:\t%f\n", average_turnaround_time);

return 0;

}

```

## OUTPUT:

```

loremvampire@shashi-kumar: ~/Shashi Kumar/USP$ gedit 6.c
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 6.c -o 6pgm
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./6pgm

Enter Total Number of Processes:      4

Enter Details of Process[1]
Arrival Time:  1
Burst Time:    3

Enter Details of Process[2]
Arrival Time:  2
Burst Time:    5

Enter Details of Process[3]
Arrival Time:  3
Burst Time:    6

Enter Details of Process[4]
Arrival Time:  4
Burst Time:    8

Enter Time Quantum:      10

Process ID      Burst Time      Turnaround Time      Waiting Time
Process[1]      3              2              -1
Process[2]      5              6              1
Process[3]      6              11             5
Process[4]      8              18             10

Average Waiting Time:  3.750000
Avg Turnaround Time:  9.250000
loremvampire@shashi-kumar:~/Shashi Kumar/USP$

```

7. Implement priority-based scheduling algorithm and calculates average waiting time and average turnaround time.

```
#include<stdio.h>

int main() {
int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n;
int 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\tBurst Time\tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++) {
    tat[i]=bt[i]+wt[i];
    total+=tat[i];

    printf("\nP[%d]\t\t %d\t\t %d\t\t %d",p[i],bt[i],wt[i],tat[i]);
}
avg_tat=total/n;

printf("\n\nAverage Waiting Time=%d",avg_wt);
printf("\n\nAverage Turnaround Time=%d\n",avg_tat);

return 0;
}

```

## OUTPUT:

```

loremvampire@shashi-kumar: ~/Shashi Kumar/USP$ gedit 7.c
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 7.c -o 7pgm
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./7pgm
Enter Total Number of Process:3

Enter Burst Time and Priority
P[1]
Burst Time:5
Priority:1
P[2]
Burst Time:3
Priority:2
P[3]
Burst Time:7
Priority:3

Process  Burst Time    Waiting Time    Turnaround Time
P[1]           5             0                5
P[2]           3             5                8
P[3]           7             8               15

Average Waiting Time=4
Average Turnaround Time=9
loremvampire@shashi-kumar:~/Shashi Kumar/USP$

```

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

**Reciever:**

```
#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;
}
```

**Writer:**

```
#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 send is : %s \n", message.mesg_text);
    return 0;
}

```

## OUTPUT:

The image shows two terminal windows side-by-side, both with the title bar 'lorenvampire@shashi-kumar: ~/Shashi Kumar/USP'.

**Left Terminal Window:**

```

lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 8r.c
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 8r.c -o 8rpgm
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ ./8rpgm
Data Received is : Hello
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$

```

**Right Terminal Window:**

```

lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 8w.c
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 8w.c -o 8wpgm
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$ ./8wpgm
Write Data : Hello
Data send is : Hello
lorenvampire@shashi-kumar:~/Shashi Kumar/USP$

```

9. Where a parent writes a message to pipe and child reads message from pipe.

**Producer:**

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<stdlib.h>
#define MAXSIZE 10
#define FIFO_NAME "myfifo"
int main()
{
int foid;
int fd, n;
char *w;
system("clear");
w=(char *)malloc(sizeof(char)*MAXSIZE);
int open_mode=O_WRONLY;
foid=mkfifo(FIFO_NAME, 0755);
if(foid== -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);
}
}

```

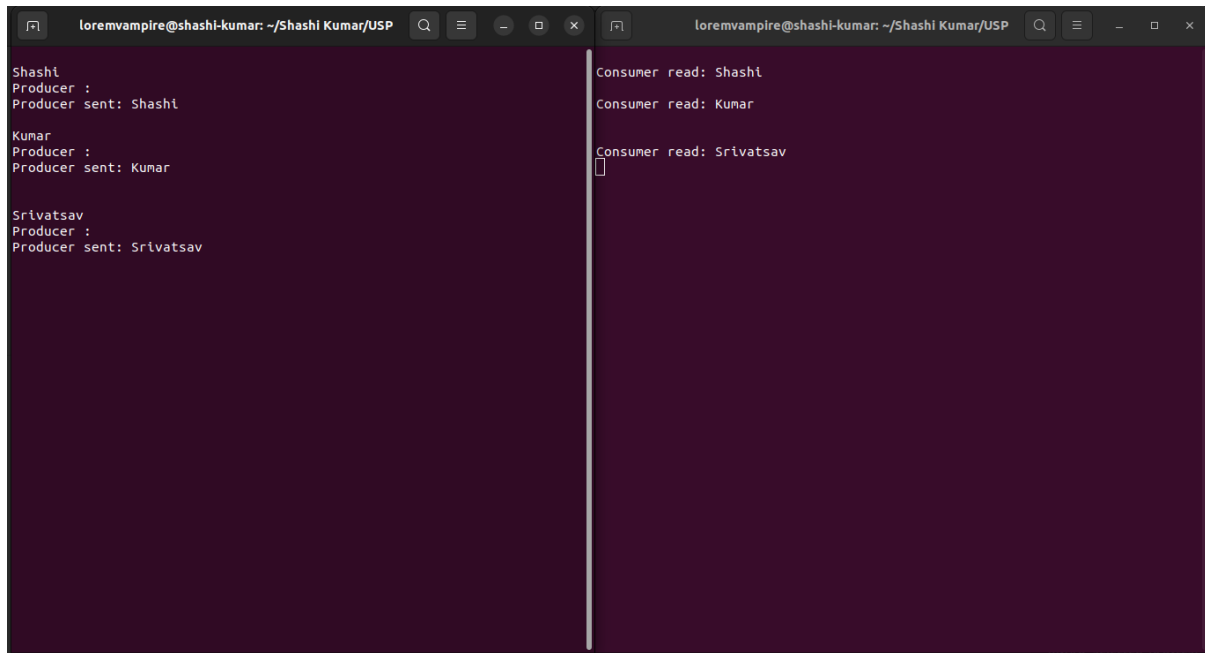
### **Consumer:**

```

#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<stdlib.h>
#define MAXSIZE 10
#define FIFO_NAME "myfifo"
int main()
{
int ffoid;
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);
}
}

```

## OUTPUT:

The image shows two side-by-side terminal windows. Both windows have a title bar that reads 'loremvampire@shashi-kumar: ~/Shashi Kumar/USP'. The left terminal window displays the output of a producer process. It shows three items being sent: 'Shashi', 'Kumar', and 'Srivatsav'. Each item is preceded by 'Producer :' and followed by 'Producer sent:'. The right terminal window displays the output of a consumer process. It shows three items being read: 'Shashi', 'Kumar', and 'Srivatsav'. Each item is preceded by 'Consumer read:'. There is a small cursor visible on the line 'Consumer read: Srivatsav'.

### 10. Demonstrate setting up a simple web server and host website on your own Linux computer.

#### What Is LAMP and How Does It Work?

The best way to create a local web server is to install LAMP, one of the most popular stacks for building and deploying dynamic web applications. The LAMP stack uses **Linux**, **Apache**, **MySQL**, and **PHP** as its foundation.

Below is a brief explanation of how LAMP works:

1. Requests will be pointed to the Apache web server whenever a user visits your website.
2. The web server will look for the requested web page file and pass the information to PHP. PHP interprets and pulls the necessary data from the MySQL database to render the web content.
3. Finally, the Apache web server delivers the web content and displays it on the user's web browser.

#### 1. Install Ubuntu Operating System

4. The first step is to install a Linux operating system. We recommend installing Ubuntu, one of the most popular Debian-based distributions. If you are a Windows user, you can choose whether to do a dual boot or a fresh installation. As for Linux users, you may skip this step.
5. You can download the Ubuntu installer from [the official directory](#). After that, create a bootable USB drive using third-party software, such as **Rufus**. Once finished, boot Ubuntu OS from the USB, and follow the installation wizard.

#### 2. Install the Apache Web Server

There are multiple steps to install and configure Apache, which we will cover in this section.

## Installing SSH Client

Before installing Apache, you need to install the SSH client on your Linux computer. First, open Terminal by pressing **Ctrl + Alt + T** and type the following command to check for updates and upgrades:

```
$ sudo apt update && sudo apt upgrade
```

Then, install the SSH client by entering the command below:

```
$ sudo apt install openssh-server
```

Once installed, activate the SSH server with the following command:

```
$ sudo systemctl enable --now ssh
```

Then, check whether the SSH server is running by entering:

```
$ sudo systemctl status ssh
```

## Installing Apache

The next step is to install the Apache web server. In the Terminal window, type the following command:

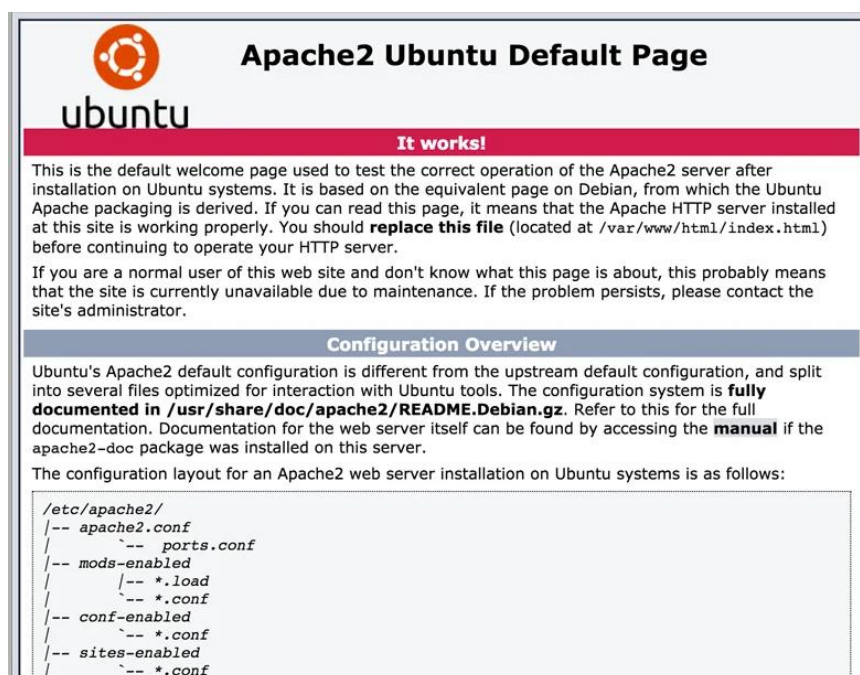
```
$ sudo apt-get install apache2
```

## Verifying the Apache Installation

After successfully installing Apache, verify the installation by entering this URL into your web browser's address bar: [http://<your\\_ip\\_address\\_here>](http://<your_ip_address_here>).

To find your local computer's IP address, navigate to **Settings** → **Network**, then click the **gear** icon on your current network interface. You can find your IP address' information on the **Details** tab.

The page should look like the following example:



### 3. Install MySQL

[MySQL](#) will be the database management system for your web application. To install MySQL, enter the command below in the Terminal window:

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

The installer will prompt you to create a password for the MySQL root user. After that, check the status of the MySQL service using the command below:

```
$ sudo systemctl status mysql
```

### 4. Install PHP

PHP is a web server scripting language for executing applications. Installing the stable version of PHP and its additional modules is highly recommended. So, in the Terminal window, input the following command:

```
$ sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql php-cgi php-curl php-json
```

Once completed, check if you have successfully installed the latest PHP version using the command below:

```
$ php -v
```

### 5. Make a Directory for Your Domain

After successfully installing the LAMP stack, you can set up the virtual host. First, you must create a specific directory for storing your website files.

Use the command **cd** to move from your active directory to the **/var/www** directory. Open Terminal and enter the following command:

```
$ cd /var/www
```

After that, create a directory using the command below:

```
$ sudo mkdir -p /var/www/domainname.com/
```

Remember to replace **domainname.com** with your own website's domain name.

Next, change the file ownership and assign necessary permissions inside the **/var/www/domainname.com** directory using the following commands:

```
$ sudo chown -R $<your_username>:$<your_username> /var/www/domainname.com
```

```
$ sudo chmod -R 755 /var/www/domainname.com
```

### 6. Create a Sample Web Page for Testing

After creating the directory to store your website files, create an HTML file or a sample web page using a text editor. In this tutorial, we will use [the Nano editor](#). To start, type the command below:

```
$ nano /var/www/domainname.com/index.html
```

Next, copy and paste the following HTML code inside the text editor:

```
<html>

<head>

<title>Welcome to My Website</title>

</head>

<body>

<h1>Thank you for visiting my humble web page!</h1>

</body>

</html>
```

Save the file by pressing **Ctrl + O**, then **Ctrl + X** to exit the text editor.

## 7. Create a Virtual Host File and Activate It

At this stage, we have successfully created a local website directory and a sample web page. The next step is to make the website accessible online. To do that, [create a virtual host file](#) inside the Apache default directory:

```
$ sudo nano /etc/apache2/sites-available/domainname.com.conf
```

Next, add the following lines of code inside the **domainname.com.conf** file:

```
1<VirtualHost *:80>

2  ServerAdmin admin@domainname.com

3  ServerName domainname.com

4  ServerAlias www.domainname.com

5  DocumentRoot /var/www/domainname.com

6  ErrorLog ${APACHE_LOG_DIR}/error.log

7  CustomLog ${APACHE_LOG_DIR}/access.log combined

8</VirtualHost>
```

Replace the information for the **ServerAdmin**, **ServerName**, **ServerAlias**, and **DocumentRoot** fields with your own settings. Then, save the changes and exit the text editor by pressing **Ctrl + O**, then **Ctrl + X**.

After that, enable the virtual host configuration file using the **a2ensite** command:

```
$ sudo a2ensite domainname.com
```

Then, disable the default configuration file using the **a2dissite** command:

```
$ sudo a2dissite 000-default.conf
```

After successfully performing these actions, restart Apache with the following command:

```
$ sudo systemctl restart apache2
```

## 8. Test Virtual Host

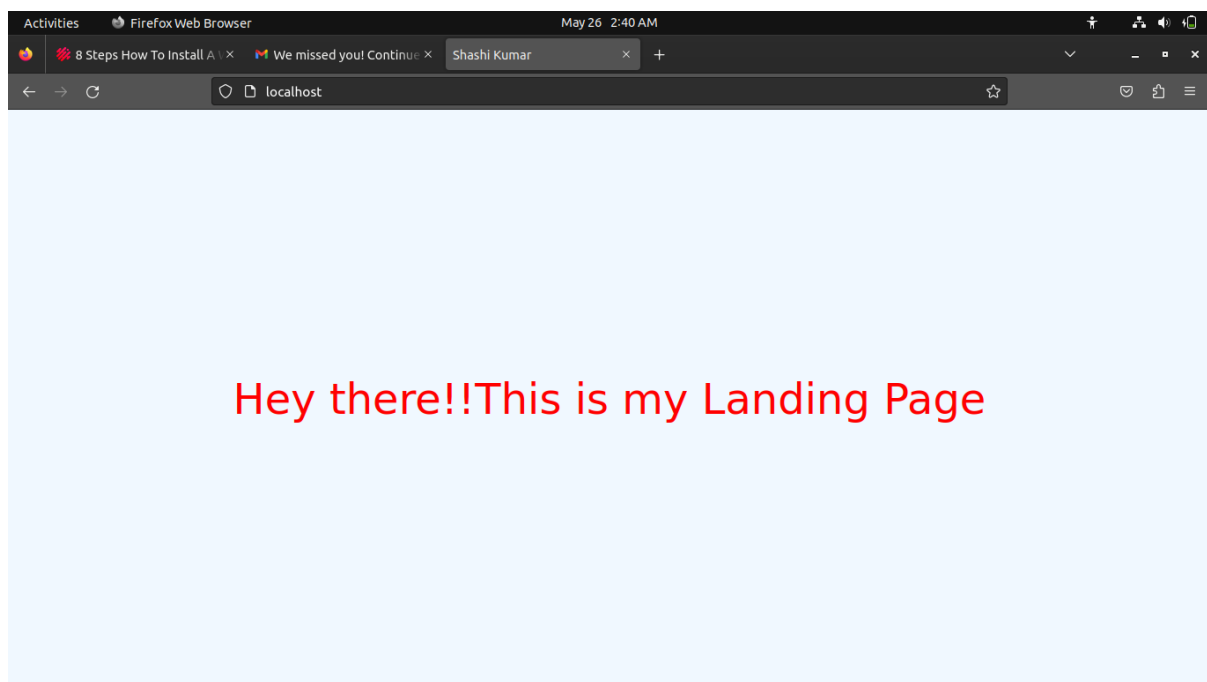
The final step is to test the virtual host. But first, it is important to perform error testing on the virtual host configuration. Type the command below in the Terminal window:

```
$ sudo apache2ctl configtest
```

If there is no error, the output of the command will show the confirmation message: **Syntax OK**. In that case, restart the Apache service using the **sudo systemctl restart apache2** command.

Finally, you can test your virtual host by entering your domain name in the web browser's address bar. If the web browser displays the sample web page, you have successfully created your own server.

## Sample Output:



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

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

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

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

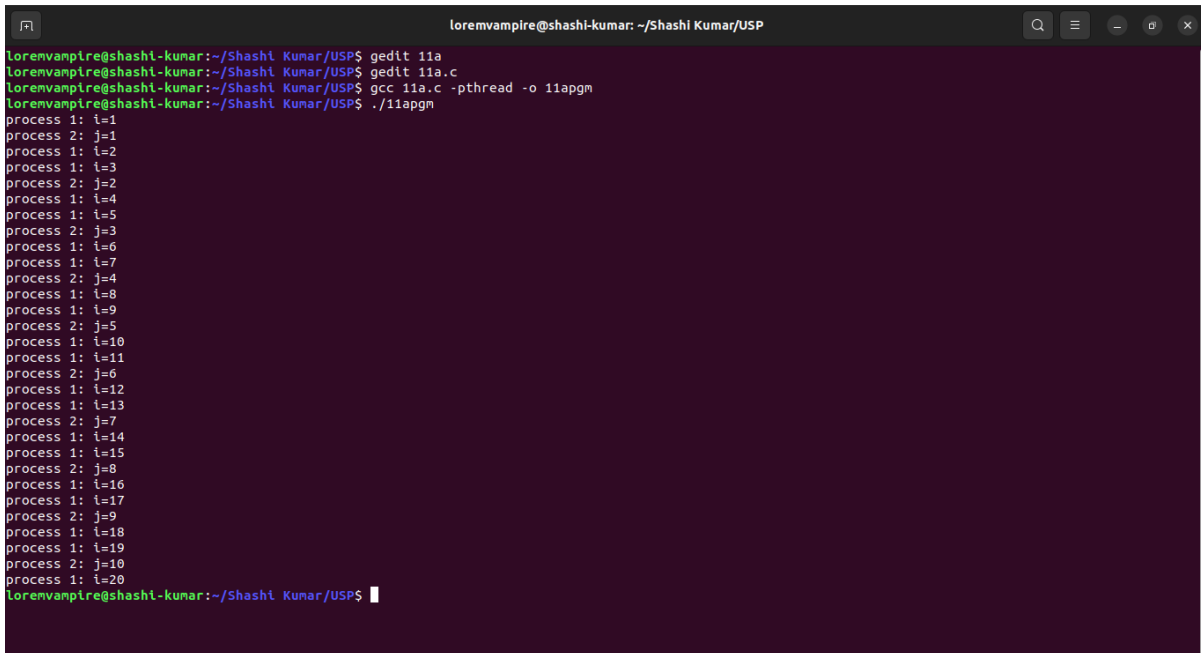
```
#include<stdlib.h>
```

```
void* myturn(void *arg)
{
    for(int i=1;i<=20;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;
}
```

## OUTPUT:



```
loremvampire@shashi-kumar: ~/Shashi Kumar/USP$ gedit 11a
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 11a.c
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 11a.c -pthread -o 11apgn
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./11apgn
process 1: i=1
process 2: j=1
process 1: i=2
process 1: i=3
process 2: j=2
process 1: i=4
process 1: i=5
process 2: j=3
process 1: i=6
process 1: i=7
process 2: j=4
process 1: i=8
process 1: i=9
process 2: j=5
process 1: i=10
process 1: i=11
process 2: j=6
process 1: i=12
process 1: i=13
process 2: j=7
process 1: i=14
process 1: i=15
process 2: j=8
process 1: i=16
process 1: i=17
process 2: j=9
process 1: i=18
process 1: i=19
process 2: j=10
process 1: i=20
loremvampire@shashi-kumar:~/Shashi Kumar/USP$
```

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){
```

```
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;

for (i = 0; i < 2; i++)

pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

pthread_exit(NULL);

return 0;

}

```

## OUTPUT:

```

loremvampire@shashi-kumar: ~/Shashi Kumar/USP
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 11b.c -o 11bpgm -lpthread
11b.c: In function 'myThreadFun':
11b.c:13:15: warning: format '%ld' expects argument of type 'long int', but argument 3 has type 'int' [-Wformat=]
   13 | printf("%ld %ld \n", pthread_self(), *myid);
      |           ^~^~^
      |           |   |
      |           |   +-- long int
      |           +----- int
      |           %d
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./11bpgm
139746519021120 1159718464
139746510628416 1159718464
main thread encountered
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
loremvampire@shashi-kumar:~/Shashi Kumar/USP$

```

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<netinet/in.h>

#include<arpa/inet.h>

#include<netdb.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 sockfd = 0, n = 0;

    char recvBuff[1024];

    struct sockaddr_in serv_addr;

    if(argc != 2)

    {
```

```

printf("\n Usage: %s \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)

{

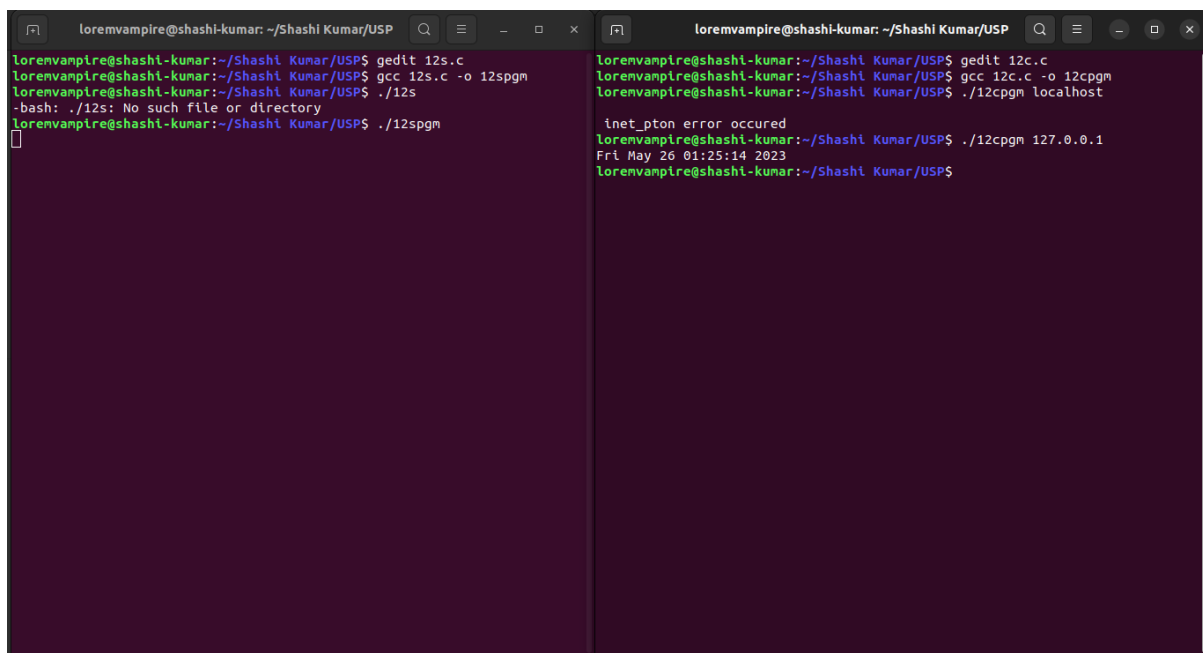
printf("\n Read error \n");

}

return 0;

}
```

## OUTPUT:



```
loremvampire@shashi-kumar: ~/Shashi Kumar/USP
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 12s.c
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 12s.c -o 12spgm
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./12s
-bash: ./12s: No such file or directory
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./12spgm

loremvampire@shashi-kumar: ~/Shashi Kumar/USP
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gedit 12c.c
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ gcc 12c.c -o 12cpgm
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./12cpgm localhost

inet_pton error occurred
loremvampire@shashi-kumar:~/Shashi Kumar/USP$ ./12cpgm 127.0.0.1
Fri May 26 01:25:14 2023
loremvampire@shashi-kumar:~/Shashi Kumar/USP$
```