1) Write a program to create processes in Linux using fork () system call.

```
#include<stdio.h>
#include <unistd.h>
int main(void)
{int pid=fork();
if(pid == -1)
{printf("fork failed!");}
else if(pid == 0)
{printf("Hello from the child process! %d\n", getpid());}
else
{printf("Hello from the parent process! %d\n", getppid());}}
```

2) To write some data on the standard output device (by default – monitor).

```
#include<stdio.h>
#include<unistd.h>
int main()
{
int count;
count=write(1,"hello\n",6);
printf("Total bytes written: %d\n",count);
}
```

3)To read data from the standard input device and write it on the screen.

```
#include<unistd.h>
int main()
{
char buff[20];
read(0,buff,10);//read 10 bytes from standard input device(keyboard), store in buffer (buff)
write(1,buff,10);//print 10 bytes from the buffer on the screen
}
```

4) Write a Program to send a message from parent process to child process using pipe().

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{int fd[2],n;
char buffer[100];
pid_t p;
pipe(fd); //creates a unidirectional pipe with two end fd[0] and fd[1]
p=fork();
if(p>0) //parent{
printf("Parent Passing value to child\n");
write(fd[1],"hello\n",6); //fd[1] is the write end of the pipe
}else // child{
printf("Child printing received value\n");
n=read(fd[0],buffer,100); //fd[0] is the read end of the pipe
write(1,buffer,n);}}
```

5) Write a program using open() system call to read the first 10 characters of an existing file “xyz.txt” and print them on screen.

```
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdio.h>
int main()
{
int n,fd;
char buff[50];
fd=open("test.txt",O_RDONLY);
printf("The file descriptor of the file is: %d\n",fd);
n=read(fd,buff,10);
write(1,buff,n);
}
```

6) Write a program to read 10 characters from file “test.txt” and write them into non-existing file “abc.txt”.

```
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
int main(){
int n,fd,fd1;
char buff[50];
fd=open("test.txt",O_RDONLY);
n=read(fd,buff,10);
fd1=open("towrite.txt",O_WRONLY|O_CREAT,0642);
O_WRONLY and O_CREAT
write(fd1,buff,n);}
```

7) Write a program to create a shared memory segment which attaches itself to it and then writes some content into the shared memory segment.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main(){
int i;
void *shared_memory;
char buff[100];
int shmid;
shmid=shmget ((key_t)2345, 1024, 0666|IPC_CREAT);
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0);
printf("Process attached at %p\n",shared_memory);
printf("Enter some data to write to shared memory\n");
read(0,buff,100);
strcpy(shared_memory,buff);
printf("You wrote : %s\n",(char *)shared_memory);}
```

8) Program to create threads in Linux. Thread prints 1-5 while the main process prints 21-25.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
void *thread_function(void *arg);
int i,j;
int main() {
pthread_t a_thread;
pthread_create(&a_thread, NULL, thread_function, NULL);
pthread_join(a_thread, NULL);
printf("Inside Main Program\n");
for(j=21;j<26;j++){
printf("%d\n",j);
sleep(1);}}
void *thread_function(void *arg) {
printf("Inside Thread\n");
for(i=1;i<6;i++)
{printf("%d\n",i);
sleep(1);}}
gcc thread.c -lpthread
```

9) Write a program to create a thread to add two numbers.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#include<string.h>
void *thread_function(void *arg);
int num[2]={3,5};
int main() {
pthread_t a_thread;
void *result;
pthread_create(&a_thread, NULL, thread_function,(void *) num);
pthread_join(a_thread, &result);
printf("Inside main process\n");
printf("Thread returned:%s\n",(char *)result);}
void *thread_function(void *arg) {
printf("Inside thread\n");
int *x=arg;
int sum=x[0]+x[1];
printf("sum is %d\n",sum);
pthread_exit("sum calculated");}
```

10)Write a program to implement Basic Process scheduling algorithm SJF.

```
#include<stdio.h>
int main(){
int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,totalT=0,pos,temp;
float avg_wt,avg_tat;
printf("Enter number of process:");
scanf("%d",&n);
printf("\nEnter Burst Time:\n");
for(i=0;i<n;i++){
```

```
printf("p%d:",i+1);
scanf("%d",&bt[i]);
p[i]=i+1;}
for(i=0;i<n;i++)
{pos=i;
for(j=i+1;j<n;j++)
{if(bt[j]<bt[pos])
pos=j;}
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=(float)total/n;
printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++){
tat[i]=bt[i]+wt[i];
totalT+=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=(float)totalT/n;
printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\nAverage Turnaround Time=%f",avg_tat);}
```

11) Write a program to implement Basic Process scheduling algorithm FCFS.

```
#include <stdio.h>
int main(){
int pid[15];
int bt[15];
int n;
printf("Enter the number of processes: ");
scanf("%d",&n);
printf("Enter process id of all the processes: ");
for(int i=0;i<n;i++)
{scanf("%d",&pid[i]);}
printf("Enter burst time of all the processes: ");
for(int i=0;i<n;i++)
{scanf("%d",&bt[i]);}
int i, wt[n];
wt[0]=0;
for(i=1; i<n; i++)
{wt[i]= bt[i-1]+ wt[i-1];}
printf("Process ID Burst Time Waiting Time TurnAround Time\n");
float twt=0.0;
float tat= 0.0;
for(i=0; i<n; i++){
```

```
printf("%d\t\t", pid[i]);
printf("%d\t\t", bt[i]);
printf("%d\t\t", wt[i]);
printf("%d\t\t", bt[i]+wt[i]);
printf("\n");
twt += wt[i];
tat += (wt[i]+bt[i]);}
float att,awt;
awt = twt/n;
att = tat/n;
printf("Avg. waiting time= %f\n",awt);
printf("Avg. turnaround time= %f",att);}
```

12) Write a program to demonstrate Semaphores in Linux.

```
#include<pthread.h>
#include<stdio.h>
#include<unistd.h>
void *fun1();
void *fun2();
int shared=1;
int main(){
pthread_t thread1,thread2;
pthread_create(&thread1,NULL,fun1,NULL);
pthread_create(&thread2,NULL,fun2,NULL);
pthread_join(thread1,NULL);
pthread_join(thread2,NULL);
printf("Final value of shared is %d\n",shared);}
void *fun1(){
int x;
x=shared;
printf("Thread1 reads the value of shared variable as %d\n",x);
x++;
printf("Local updation by thread1 %d\n",x);
sleep(1);
shared=x;
printf("value of shared variable updated by thread1 %d\n",shared);}
void *fun2(){
int y;
y=shared;
printf("Thread2 reads the value of shared variable as %d\n",y);
y--;
printf("Local updation by thread2 %d\n",y);
sleep(1);
shared=y;
printf("value of shared variable updated by thread2 %d\n",shared);}
```

13) Write a program to demonstrate race conditions in Linux.

```
#include<pthread.h>
#include<stdio.h>
#include<unistd.h>
void *fun1();
void *fun2();
```

```
int shared=1;
int main(){
pthread_t thread1,thread2;
pthread_create(&thread1,NULL,fun1,NULL);
pthread_create(&thread2,NULL,fun2,NULL);
pthread_join(thread1,NULL);
pthread_join(thread2,NULL);
printf("Final value of shared is %d\n",shared);}
void *fun1(){
int x;
x=shared;
printf("Thread1 reads the value of shared variable as %d\n",x);
x++;
printf("Local updation by thread1 %d\n",x);
sleep(1);
shared=x;
printf("value of shared variable updated by thread1 %d\n",shared);}
void *fun2(){
int y;
y=shared;
printf("Thread2 reads the value of shared variable as %d\n",y);
y--;
printf("Local updation by thread2 %d\n",y);
sleep(1);
shared=y;
printf("value of shared variable updated by thread2 %d\n",shared);}
```

14) Write a program to demonstrate paging.

```
#include<stdio.h>
void main(){
int memsize=32;
int pagesize, nofpage;
int p[100];
int frameno, offset;
int logadd, phyadd;
int i;
int choice=0;
printf("\nYour memsize is %d ",memsize);
printf("\nEnter page size:");
scanf("%d",&pagesize);
nofpage=memsize/pagesize;
for(i=0;i<nofpage;i++){
printf("\nEnter the frame of page%d:",i);
scanf("%d",&p[i]);}
do{
printf("\nEnter a logical address:");
scanf("%d",&logadd);
frameno=logadd/pagesize;
offset=logadd%pagesize;
phyadd=(p[frameno]*pagesize)+offset;
```

```
printf("\nPhysical address is:%d",phyadd);
printf("\nDo you want to continue(1/0)?:");
scanf("%d",&choice);
}while(choice==1);}
```

15) Write a program to demonstrate Disk scheduling algorithms like FCFS.

```
#include<stdio.h>
#include<math.h>
void main(){
int i,sum=0,n,st;
int a[20],b[20],dd[20];
do{
printf("\nEnter the block number between 0 and 200: ");
scanf("%d",&st);}
while((st>=200)||(st<0));
printf("\nOur disk head is on the %d block",st);
a[0]=st;
printf("\nEnter the no. of request: ");
scanf("%d",&n);
printf("\nEnter request: ");
for(i=1;i<=n;i++){
printf("\nEnter %d request: ",i);
scanf("%d",&a[i]);
do{
if((a[i]>200)||(a[i]<0)){
printf("\nBlock number must be between 0 and 200!");
}}while((a[i]>200)||(a[i]<0));}
for(i=0;i<=n;i++)
dd[i]=a[i];
printf("\n\t\tFIRST COME FIRST SERVE: ");
printf("\nDISK QUEUE:");
for(i=0;i<=n;i++)
printf("\t%d",a[i]);
printf("\n\nACCESS ORDER:");
for(i=0;i<=n;i++){
printf("\t%d",dd[i]);
if(i!=n)
sum+=abs(dd[i]-dd[i+1]);}
printf("\n\nTotal no. of head movements: %d",sum);}
```

16) Write a program to demonstrate various file allocation methods.

```
#include<stdio.h>
void main(){
int i,j,n,block[20],start;
printf("Enter the no. of file:\n");
scanf("%d",&n);
printf("Enter the number of blocks needed foreach file:\n");
for(i=0;i<n;i++)
scanf("%d",&block[i]);
start=0;
printf("\t\tFile name\tStart\tSize of file\t\t\n");
```

```
printf("\n\t\tFile1\t\t%d\t\t\t%d\n",start,block[0]);
for(i=2;i<=n;i++){
start=start+block[i-2];
printf("\t\tFile%d\t\t%d\t\t%d\n",i,start,block[i-1]);}}
```

Extras:

//fork()

```
#include<stdio.h>
#include<unistd.h>
int main()
{int a,b,n;
pid_t ret_value;
printf("\n The Process id is %d \n",getpid());
ret_value=fork();
if(ret_value<0)
{printf("\n fork failure\n");
}else if (ret_value==0)
{printf("*****Child process*****\n");
printf("The process id is %d and parent id is %d \n ", getpid(),getppid());
printf("Enter a number to check even or odd\n",n);
scanf("%d",&n);
if(n%2 == 0)
printf("number %d is even\n",n);
else
printf("number %d is odd\n",n);
}else
{wait();
printf("parent process\n");
printf("The process id id %d \n", getpid());
printf("Enter 2 numbers to check maximum or minimum\n");
scanf("%d%d",&a,&b);
if (a>b)
printf("%d is greater than %d\n",a,b);
else
printf("%d is greater than %d\n",b,a);
}return 0;}
```