

## **Q) IMPLEMENT ANY ONE BASIC COMMANDS OF LINUX LIKE LS, CP, MV AND OTHERS USING KERNEL APIS**

### **PROGRAM:3A**

#### **A) IMPLEMENTING" ls" COMMAND USING SYSTEM CALLS**

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <dirent.h>
int main(int argc, char *argv[]){
    DIR *dp;
    struct dirent *sd;
    dp opendir(argv[1]);
    while((sd=readdir(dp))!= NULL)
    {
        printf("%s\t",sd->d_name);
    }
    closedir(dp);
}
```

#### **OUTPUT:**

```
root@litmus:/home# gcc Iscmd.c
root@litmus:/home#/a.out dimame
f.txt fl.c
```

### **PROGRAM:3B**

#### **A) IMPLEMENTING" mv" COMMAND USING SYSTEM CALLS**

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <dirent.h>
int main(int argc, char *argv[])
{
    int i, fd1, fd2;
    char *file1, *file2, buf[2];
    file1= argv[1];
    file2=argv[2];
    printf("file1=%s file2=%s", file1, file2);
    fd1=open(file1, O_RDONLY, 0777);
    fd2=creat(file2, 0777)
    while(i=read(fd1, buf, 1)>0)
    write(fd2, buf, 1);
    remove(file1);
    close(fd1); close(fd2);
}
```

#### **OUTPUT:**

```
root@nik:$ gcc mv.c -o mv.out
root@rdk:Scat> f1
hello world
root@ndk:$/mv.out f1 f2
root@rik:Scat f1
cat: No such file or directory
root@rdk:$ cat f2
hello world
```

**Q) CREATE A CHILD PROCESS IN LINUX USING THE FORK SYSTEM CALL. FROM THE CHILD PROCESS OBTAIN THE PROCESS ID OF BOTH CHILD AND PARENT BY USING GETPID AND GETPPID SYSTEM CALL.**

### **PROGRAM:4A**

(A) CREATE CHILD PROCESS IN LINUX USING FORK SYSTEM CALL

```
#include<stdio.h>
#include <unistd.h>
int main()
{
int pid;
pid=fork();
if(pid==0)
{
printf("\n After fork");
printf("\n The new child process created by fork system call %d\n", getpid());
}
else {
printf("\n Before fork");
printf("\n The parent process id is :- %d", getppid()); printf("\n parent excuted successfully\n");
}
return 0;
}
```

### **OUTPUT:**

```
root@rajesh-optiplex-3020:$ gee B3.c
root@rajesh-optiplex-3020:$ ./a.out
```

```
Before fork
The parent process id is :-4070
parent excuted successfully
```

```
After fork
The new child process created by fork system call 4428
root@rajesh-optiplex-3020:$ ^C
root@rajesh-optiplex-3020:$
```

## **Q) WRITE A PROGRAM TO DEMONSTRATE THE CONCEPT OF NON-PREEMPTIVE SCHEDULING ALGORITHMS.**

### **PROGRAM: 5A**

#### **A) SHORTEST JOB FIRST (NON PRIMITIVE SCHEDULING)**

```
#include<stdio.h>
int main()
{
int bt[20],p[20],wt[20],tat[20],i,j,n,total=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;
}
//sorting of burst times
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;
total=0;
```

```
printf("\nProcess\t  
Burst Time  
\tWaiting  
Time\tTurnaround  
Time");
```

```

for(i=0;i<n;i++)
{
    tat[i]=bt[i]+wt[i];
    total+=tat[i];
    printf("\np%d\t %d\t %d\t\t%d",p[i],bt[i],wt[i],tat[i]);
}
avg_tat=(float)total/n;
printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\n\nAverage Turnaround Time=%f\n",avg_tat);
}

```

## OUTPUT:

```

Enter number of process:3
Enter Burst Time:
p1:12
p2:58
p3:14

Process      Burst Time      Waiting Time      Turnaround Time
p1           12             0                12
p3           14             12               26
p2           58             26               84

Average Waiting Time=12.666667
Average Turnaround Time=40.666668
-----

```

## Q) WRITE A C PROGRAM TO IMPLEMENT SOLUTION OF PRODUCER CONSUMER PROBLEM THROUGH SEMAPHORE

### PROGRAM: 6


```
#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;  
}}}
```

## OUTPUT:



```
1.Producer  
2.Consumer  
3.Exit  
Enter your choice:2  
Buffer is empty!!  
Enter your choice:1  
  
Producer produces item 1  
Enter your choice:1  
  
Producer produces item 2  
Enter your choice:1  
  
Producer produces item 3  
Enter your choice:2  
  
Consumer consumes item 3  
Enter your choice:1  
  
Producer produces item 3  
Enter your choice:2  
  
Consumer consumes item 3  
Enter your choice:2  
  
Consumer consumes item 2  
Enter your choice:2  
  
Consumer consumes item 1  
Enter your choice:2  
Buffer is empty!!  
Enter your choice:3
```



Q)WRITE A PROGRAM IN C DEMONSTRATE THE CONCEPT OF PAGE REPLACEMENT POLICIES FOR HANDLING PAGE FAULTS EG: FIFO, LRU ETC

## PROGRAM: 9B

### FIFO:

```
#include < stdio.h >
int main()
{
    int incomingStream[] = {4 , 1 , 2 , 4 , 5};
    int pageFaults = 0;
    int frames = 3;
    int m, n, s, pages;
    pages = sizeof(incomingStream)/sizeof(incomingStream[0]);
    printf(" Incoming \t Frame 1 \t Frame 2 \t Frame 3 ");
    int temp[ frames ];
    for(m = 0; m < frames; m++)
    {
        temp[m] = -1;
    }
    for(m = 0; m < pages; m++)
    {
        s = 0;
        for(n = 0; n < frames; n++)
        {
            if(incomingStream[m] == temp[n])
            {
                s++;
                pageFaults--;
            }
        }
        pageFaults++;
        if((pageFaults <= frames) && (s == 0))
        {
            temp[m] = incomingStream[m];
        }
        else if(s == 0)
        {
            temp[(pageFaults - 1) % frames] = incomingStream[m];
        }
        printf("\n");
        printf("%d\t\t\t",incomingStream[m]);
        for(n = 0; n < frames; n++)
        {
            if(temp[n] != -1)
                printf(" %d\t\t\t", temp[n]);
            else
                printf(" - \t\t\t");
        }
        printf("\nTotal Page Faults:\t%d\n", pageFaults);
        return 0;
    }
}
```

## OUTPUT:

Incoming	Frame 1	Frame 2	Frame 3	
4		4	-	-
1		4	1	-
2		4	1	2
4		4	1	2
5		5	1	2
Total Page Faults: 4				

## LRU:

```
#include <stdio.h>
int findLRU(int time[], int n)
{
    int i, minimum = time[0], pos = 0;
    for (i = 1; i < n; ++i)
    {
        if (time[i] < minimum)
        {
            minimum = time[i];
            pos = i;
        }
    }
    return pos;
}
int main()
{
    int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i,
    j, pos, faults = 0;
    printf("Enter number of frames: ");
    scanf("%d", &no_of_frames);
    printf("Enter number of pages: ");
    scanf("%d", &no_of_pages);
    printf("Enter reference string: ");
    for (i = 0; i < no_of_pages; ++i)
    {
        scanf("%d", &pages[i]);
    }
    for (i = 0; i < no_of_frames; ++i)
    {
        frames[i] = -1;
    }
    for (i = 0; i < no_of_pages; ++i)
    {
        flag1 = flag2 = 0;
        for (j = 0; j < no_of_frames; ++j)
        {
```

```

if (frames[j] == pages[i])
{
counter++;
time[j] = counter;
flag1 = flag2 = 1;
break;
}}
if (flag1 == 0)
{
for (j = 0; j < no_of_frames; ++j)
{
if (frames[j] == -1)
{
counter++;
faults++;
frames[j] = pages[i];
time[j] = counter;
flag2 = 1;
break;}}
if (flag2 == 0)
{
pos = findLRU(time, no_of_frames);
counter++;
faults++;
frames[pos] = pages[i];
time[pos] = counter;
}
printf("\n");
for (j = 0; j < no_of_frames; ++j)
{
printf("%d\t", frames[j]);
}}
printf("\nTotal Page Faults = %d", faults);
return 0;
}

```

## OUTPUT:

```

Enter number of frames: 3
Enter number of pages: 10
Enter reference string: 7 5 9 4 3 7 9 6 2 1

7      -1      -1
7      5       -1
7      5       9
4      5       9
4      3       9
4      3       7
9      3       7
9      6       7
9      6       2
1      6       2
Total Page Faults = 10

```