ATMA RAM SANATAN DHARMA COLLEGE
University of Delhi

SUBMITTED BY:

Aditi Sharma

Roll no : 18055

Bsc.(Hons) Computer Science

SUBMITTED TO:

Dr. Parul Jain

Department of Computer Science

PRACTICAL 5

Objective:

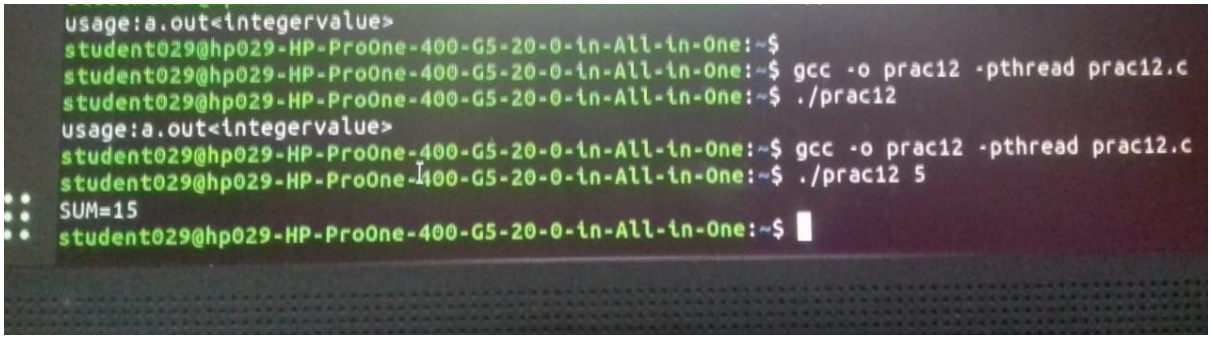
Write a program to copy files using system calls.

Code:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
void copy(int,int);
void display(int);
main(int argc,char *argv[])
{
	int fold,fnew;
	if(argc!=3)
	{
		printf("Two Arguments Required");
		exit(1);
	}
	fold=open(argv[1],0);
	if(fold==-1)
	{
		printf("Unable to Open the File\n%s",argv[1]);
		exit(1);
	}
	fnew=creat(argv[2],0666);
	if(fnew==-1)
	{
		printf("Unable to Open the File\n%s",argv[2]);
		exit(1);
	}
	copy(fold,fnew);
	exit(0); close(fold);
	close(fnew);
	fnew=open(argv[2],0);
	printf("New File:\n");
	display(fnew);
	close(fnew);
	exit(0);
}
void copy(int old,int new)
{
	nt count=0;
	char buffer[512];
	while((count=read(old,buffer,sizeof(buffer)))>0)
```

```
        {
                write(new,buffer,count);
        }
}
void display(int fnew)
{
        int count=0,i;
        char buffer[512];
        while((count=read(fnew,buffer,sizeof(buffer)))>0)
        {
                for(i=0;i<count;i++)
                {
                        printf("%c",buffer[i]);
                }
        }
        for(i=0;i<count;i++)
        {
                printf("%c",buffer[i]);
        }
}
```

Output:



PRACTICAL 11

Objective:

Write a program to implement SRJF scheduling algorithm.

Code:

```
#include <stdio.h>
int main()
{
	int n, ari[10],bur[10],total=0,I,j,small,temp,procs[100],k,waiting[10],finish[10];
	float tavg=0.0,wavg=0.0;
	printf("\nEnter number of process: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("Enter the arrival time of processes %d:\t",i);
		scanf("%d",&ari[i]);
		printf("Enter the burst time of processes %d:\t",i);
		scanf("%d",&bur[i]);
		waiting[i]=0;
		total+=bur[i];
	}
	for(i=0;i<n;i++)
	{
		for(j=i+1;j<n;j++)
		{
			if(ari[i]>ari[j])
			{
				temp=ari[i];
				ari[i]=ari[j];
				ari[j]=temp;
				temp=bur[i];
				bur[i]= bur [j];
				bur[j]=temp;
			}
		}
	}
	for(i=0;i<total;i++)
	{
		small=3200;
		for(j=0;j<n;j++)
		{
			if(bur[j]!=0 && (ari[j]<=i) && (bur[j]<small)
			{
				small=bur[j];
				k=j;
			}
```

```
                }
                bur[k]--;
                procs[i]=k;
        }
        k=0;
        for(i=0;i<total;i++)
        {
                for(j=0;j<n;j++)
                {
                        if(procs[i]==j)
                        {
                                finish[j]=i;
                                waiting[j]++;
                        }
                }
        }
        for(i=0;i<n;i++)
        {
                printf("\nProcesses %d:-Finish Time=>%d TurnAround Time=>%d Waiting
Time=>%d\n",i+1 ,finish[i]+1,(finish[i]-ari[i])+1,((( finish[i]+1)-waiting[i])-ari[i]);
                wavg=wavg+((( finish[i]+1)-waiting[i])-ari[i]);
                tavg=tavg+(( finish[i]-ari[i])+1);
        }
        printf("\nWavg=>\t%f\n Tavg=>\t5f\n",(wavg/n),(tavg/n));
        return 0;
}
```

Output:

```
                :~$ gcc -o 11 11.c
                :~$ ./11
Enter the number of process:3
Enter the arrival time of process0;     0
Enter the burst time of process 0:      7
Enter the arrival time of process1;     1
Enter the burst time of process 1:      3
Enter the arrival time of process2;     3
Enter the burst time of process 2:      4

 Process:1
FINISH TIME===>14       TURNAROUND TIME===>14   WAITING TIME==>7
 Process:2
FINISH TIME===>4        TURNAROUND TIME===>3    WAITING TIME==>0
 Process:3
FINISH TIME===>8        TURNAROUND TIME===>5    WAITING TIME==>1
 WAvg==>        2.666667
 TAVG==>        7.333333
```

PRACTICAL 12

Objective:

Write a program to calculate sum of n numbers using thread library.

Code:

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
int sum;
void *runner(void *param);
int main(int argc, char *argv[])
{
	pthread_t tid;
	pthread_attr_t attr;
	if(argc!=2)
	{
		fprintf(stderr,"usage:a.out\n");
		return -1;
	}
	if(atoi(argv[1])
	{
		fprintf(stderr,"%d must be >=0\n",atoi(argv[1]));
		return -1;
	}
	pthread_attr_init(&attr);
	pthread_create(&tid,&attr,runner,argv[1]);
	pthread_join(tid,NULL);
	printf("SUM=%d\n",sum);
	return 0;
}
void *runner(void *param)
{
	int i,upper=atoi(param);
	sum=0;
	for(i=1;i<=upper;i++)
		sum+=i;
	pthread_exit(0);
}
```

Output:

```
recognized: File truncated
collect2: error: ld returned 1 exit status
student032@hp032-HP-ProOne-400-G5-20-0-in-All-in-One:~$ gcc -o prac5  prac5.c  new.txt
/usr/bin/ld:demo.txt: file format not recognized; treating as linker script
/usr/bin/ld:demo.txt:1: syntax error
collect2: error: ld returned 1 exit status
student032@hp032-HP-ProOne-400-G5-20-0-in-All-in-One:~$ gcc -o prac5  prac5.c  demo.txt
student032@hp032-HP-ProOne-400-G5-20-0-in-All-in-One:~$ gcc -o prac5  prac5.c
student032@hp032-HP-ProOne-400-G5-20-0-in-All-in-One:~$ ./prac5 new.txt demo.txt
student032@hp032-HP-ProOne-400-G5-20-0-in-All-in-One:~$ gcc -o prac5  prac5.c
student032@hp032-HP-ProOne-400-G5-20-0-in-All-in-One:~$ ./prac5 new.txt demo.txt
uuhiojoyh4yhvbcog vu pro thgp0r9ie[fcoe[vgokrfbm
student032@hp032-HP-ProOne-400-G5-20-0-in-All-in-One:~$ cat demo.txt
student032@hp032-HP-ProOne-400-G5-20-0-in-All-in-One:~$ gcc -o prac12 prac12.c
/tmp/ccH9xPgf.o: In function `main':
prac12.c:(.text+0xc9): undefined reference to `pthread_create'
prac12.c:(.text+0xda): undefined reference to `pthread_
collect2: error: ld
```

PRACTICAL 13

Objective:

Write a program to implement first-fit, best-fit and worst-fit allocation strategies.

Code:

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

//function to enter values in array

void accept(int a[],int n)
{
	int i;
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
}

//function to display array

void display(int a[],int n)
{
	int i;
	printf("\n\n");
	for(i=0;i<n;i++)
	{
		printf("\t%d ",a[i]);
	}
}

//function to sort given array

void sort(int a[],int n)
{
	int i,j,temp;
	for(i=0;i<n-1;i++)
	{
		for(j=0;j<n-1;j++)
		{
```

```
                        if(a[j]>a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }
}

//reverse sort

void revsort(int a[],int n)
{
        int i,j,temp;
        for(i=0;i<n-1;i++)
        {
                for(j=0;j<n-1;j++)
                {
                        if(a[j]<a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }
}

// first fit algo

void first_fit(int psize[],int np,int msize[],int nm)
{
        int i,j,in_fr,ex_fr,flag[30]={0};
        in_fr=ex_fr=0;
        for(i=0;i<np;i++)
        {
                for(j=0;j<nm;j++)
                {
                        if(flag[j]==0 && msize[j]>=psize[i])
                        {
                                flag[j]=1;
                                in_fr=in_fr+msize[j]-psize[i];
                                break;
                        }
                }
                if(j==nm)
                        printf("\n\nTHERE IS NO SPACE FOR PROCESS %d ",i);
```

```
        }
        for(i=0;i<nm;i++)
        {
                if(flag[i]==0)
                        ex_fr=ex_fr+msize[i];
        }
        printf("\n\nPROCESSES::");
        display(psize,np);
        printf("\n\nMEMORY HOLES::");
        display(msize,nm);
        printf("\n\nTOTAL SUM OF INTERNAL FRAGMENTATION = %d ",in_fr);
        printf("\n\nTOTAL SUM OF EXTERNAL FRAGMENTATION = %d ",ex_fr);
}
void best_fit(int psize[],int np,int msize[],int nm)
{
        int i,j,in_fr,ex_fr,temp[30],flag[30]={0};
         in_fr=ex_fr=0;
        for(i=0;i<nm;i++)
                temp[i]=msize[i]
        sort(temp,nm);
        for(i=0;i<np;i++)
        {
                for(i=0;i<nm;i++)
                {
                        if(flag[j]==0 && temp[j]>=psize[i])
                        {
                                flag[j]=1;
                                in_fr=in_fr+temp[j]-psize[i];
                                break;
                        }
                }
                if(j==nm)
                        printf("\n\nTHERE IS NO SPACE FOR PROCESS %d ",i);
        }
        for(i=0;i<nm;i++)
        {
                if(flag[i]==0)
                        ex_fr=ex_fr+temp[i];
        }
        printf("\n\nPROCESSES::");
        display(psize,np);
        printf("\n\nMEMORY HOLES::");
        display(temp,nm);
        printf("\n\nTOTAL SUM OF INTERNAL FRAGMENTATION = %d ",in_fr);
        printf("\n\nTOTAL SUM OF EXTERNAL FRAGMENTATION = %d ",ex_fr);
}

void worst_fit(int psize[],int np,int msize[],int nm)
```

```
{
        int i,j,in_fr,ex_fr,temp[30],flag[30]={0};
         in_fr=ex_fr=0;
         for(i=0;i<nm;i++)
                temp[i]=msize[i];
        revsort(temp,nm);
        for(i=0;i<np;i++)
        {
                for(j=0;j<nm;j++)
                {
                        if(flag[j]==0 && temp[j]>=psize[i])
                        {
                                flag[j]=1;
                                in_fr=in_fr+temp[j]-psize[i];
                                break;
                        }
                }
                if(j==nm)
                        printf("\n\nTHERE IS NO SPACE FOR PROCESS %d ",i);
        }
        for(i=0;i<nm;i++)
        {
                if(flag[i]==0)
                        ex_fr=ex_fr+temp[i];
        }
        printf("\n\nPROCESSES::");
        display(psize,np);
        printf("\n\nMEMORY HOLES::");
        display(temp,nm);
        printf("\n\nTOTAL SUM OF INTERNAL FRAGMENTATION = %d ",in_fr);
        printf("\n\nTOTAL SUM OF EXTERNAL FRAGMENTATION = %d ",ex_fr);
}

void main()
{
        nt ch,np,nm,psize[30],msize[30];
        printf("\nENTER NO OF PROCESSES::");
        scanf("%d",&np);
        printf("\n\nENTER SIZES OF PROCESSES::");
        accept(psize,np);
        printf("\nENTER NO MEMORY HOLES::");
        scanf("%d",&nm); printf("\n\nENTER SIZES OF MEMORY HOLES::");
        accept(msize,nm);

        while(1)
        {
                printf("\n\n\t\t**MAIN MENU**");
                printf("\n\n\tMEMORY MANAGEMENT");
```

```
                printf("\n\n\t1.FIRST FIT");
                printf("\n\n\t2.BEST FIT");
                printf("\n\n\t3.WORST FIT");
                printf("\n\n\t4.QUIT");
                printf("\n\nENTER YOUR CHOICE::");
                scanf("%d",&ch);

                switch(ch)
                {
                        case 1:
                                printf("\n\nFIRST FIT::\n");
                                first_fit(psize,np,msize,nm);
                                break;
                        case 2:
                                printf("\n\n\tBEST FIT::\n");
                                best_fit(psize,np,msize,nm);
                                break;
                        case 3:
                                printf("\n\n\tWORST FIT::\n");
                                worst_fit(psize,np,msize,nm);
                                break;
                        case 4:
                                exit(0);
                        default:
                                printf("\n\nPLEASE ENTER CORRECT CHOICE!!");
                }
        }
}
```

Output:

```
collect2: error: ld returned 1 exit status
                    :~$ gcc -o 13 13.c
                    :~$ ./13

ENTER NO OF PROCESSES::5

ENTER SIZES OF PROCESSES::40 50 60 70 10

ENTER NO MEMORY HOLES::3

ENTER SIZES OF MEMORY HOLES::67
54
23


                **MAIN MENU**

        MEMORY MANAGEMENT

        1.FIRST FIT

        2.BEST FIT

        3.WORST FIT

        4.QUIT
```

```
        4.QUIT
ENTER YOUR CHOICE::1

FIRST FIT::

THERE IS NO SPACE FOR PROCESS 2
THERE IS NO SPACE FOR PROCESS 3
PROCESSES::
        40      50      60      70      10
MEMORY HOLES::
        67      54      23
TOTAL SUM OF INTERNAL FRAGMENTATION = 44
TOTAL SUM OF EXTERNAL FRAGMENTATION = 0
                **MAIN MENU**
        MEMORY MANAGEMENT
        1.FIRST FIT
```

```
        1.FIRST FIT
        2.BEST FIT
        3.WORST FIT
        4.QUIT
ENTER YOUR CHOICE::2

        BEST FIT::

THERE IS NO SPACE FOR PROCESS 2
THERE IS NO SPACE FOR PROCESS 3
PROCESSES::
        40      50      60      70      10
MEMORY HOLES::
        23      54      67
TOTAL SUM OF INTERNAL FRAGMENTATION = 44
```

```
TOTAL SUM OF INTERNAL FRAGMENTATION = 44
TOTAL SUM OF EXTERNAL FRAGMENTATION = 0
                **MAIN MENU**
        MEMORY MANAGEMENT
        1.FIRST FIT
        2.BEST FIT
        3.WORST FIT
        4.QUIT
ENTER YOUR CHOICE::3

        WORST FIT::

THERE IS NO SPACE FOR PROCESS 2
THERE IS NO SPACE FOR PROCESS 3
PROCESSES::
        40      50      60      70      10
```

```
        40      50      60      70      10
MEMORY HOLES::
        67      54      23
TOTAL SUM OF INTERNAL FRAGMENTATION = 44
TOTAL SUM OF EXTERNAL FRAGMENTATION = 0
                **MAIN MENU**
        MEMORY MANAGEMENT
        1.FIRST FIT
        2.BEST FIT
        3.WORST FIT
        4.QUIT
ENTER YOUR CHOICE::3

        WORST FIT::
```

```
        WORST FIT::

THERE IS NO SPACE FOR PROCESS 2
THERE IS NO SPACE FOR PROCESS 3
PROCESSES::
        40      50      60      70      10
MEMORY HOLES::
        67      54      23
TOTAL SUM OF INTERNAL FRAGMENTATION = 44
TOTAL SUM OF EXTERNAL FRAGMENTATION = 0
                **MAIN MENU**
        MEMORY MANAGEMENT
        1.FIRST FIT
        2.BEST FIT
```

```
THERE IS NO SPACE FOR PROCESS 3
PROCESSES::
        40      50      60      70      10
MEMORY HOLES::
        67      54      23
TOTAL SUM OF INTERNAL FRAGMENTATION = 44
TOTAL SUM OF EXTERNAL FRAGMENTATION = 0
                **MAIN MENU**
        MEMORY MANAGEMENT
        1.FIRST FIT
        2.BEST FIT
        3.WORST FIT
        4.QUIT
ENTER YOUR CHOICE::4
```