

# SHELL PROGRAMS

## SEARCHING A SUBSTRING IN GIVEN TEXT

### PROGRAM:

```
echo Enter main string:
read main
l1=`echo $main | wc -c`
l1=`expr $l1 - 1`
echo Enter sub string:
read sub
l2=`echo $sub | wc -c`
l2=`expr $l2 - 1`
n=1
m=1
pos=0
while [ $n -le $l1 ]
do
a=`echo $main | cut -c $n`
b=`echo $sub | cut -c $m`
if [ $a = $b ]
then
n=`expr $n + 1`
m=`expr $m + 1`
pos=`expr $n - $l2`
r=`expr $m - 1`
if [ $r -eq $l2 ]
then
break
fi
else
pos=0
m=1
n=`expr $n + 1`
fi
done
echo Position of sub string in main string is $pos
```

### OUTPUT:

Enter main string:

This is a shell pgm

Enter sub string:

shell

Position of sub string in main string is 11

## MENU BASED MATH CALCULATOR

### PROGRAM:

```
echo " Menu Based Calculator"
echo "Enter the Operands"
read a
read b
echo "Enter the Operator"
read o
case $o in
"+" ) echo "$a + $b" = `expr $a + $b`;
 "-" ) echo "$a + $b" = `expr $a - $b`;
 "*" ) echo "$a + $b" = `expr $a * $b`;
 "/" ) echo "$a + $b" = `expr $a / $b`;
 * ) echo " Inavlid Operation"
esac
```

### OUTPUT:

```
Menu Based Calculator
Enter the Operands
4
6
Enter the Operator
+
4 + 6 = 10
```

## CONVERTING ALL FILENAMES FROM LOWERCASE TO UPPERCASE

### PROGRAM

```
for i in *
do
echo Before Converting to uppcase the filename is
echo $i
j=`echo $i | tr '[a-z]' '[A-Z]`
echo After Converting to uppcase the filename is
echo $j
mv $i $j
done
```

### OUTPUT

```
Before Converting to upper case the filename is
cse.sh
After Converting to uppcase the filename is
CSE.SH
```

## PRINTING PATTERN USING LOOP STATEMENT

### PROGRAM

```
echo "Enter the Limit "  
read n  
echo "Pattern"  
for (( i = 1 ; i < $n ; i++ ))  
do  
for (( j = 1 ; j <= i ; j++ ))  
do  
echo -n " $ "  
done  
echo " "  
done
```

### OUTPUT

Enter the Limit

3

Pattern

```
$  
$ $  
$ $ $
```

## CONVERTING THE FILENAME FROM UPPERCASE TO LOWERCASE

### PROGRAM

```
echo -n "Enter the Filename"  
read filename  
if [ ! -f $filename ];  
then  
echo "Filename $filename does not exists"  
exit 1  
fi  
tr ' [A-Z]' '[a-z]' < $filename
```

### OUTPUT

Enter the Filename

CSE.sh

cse.sh

## SHOWING VARIOUS SYSTEM INFORMATION

### PROGRAM

```
echo "SYSTEM INFORMATION"
echo "Hello ,${LOGNAME}"
echo "Current Date is = $(date)"
echo "User is 'who I am'"
echo "Current Directory = $(pwd)"
echo "Network Name and Node Name = $(uname -n)"
echo "Kernal Name =$(uname -s)"
echo "Kernal Version=$(uname -v)"
echo "Kernal Release =$(uname -r)"
echo "Kernal OS =$(uname -o)"
echo "Proessor Type = $(uname -p)"
echo "Kernel Machine Information = $(uname -m)"
echo "All Information =$(uname -a)"
```

### OUTPUT

```
SYSTEM INFORMATION
Hello, 3CSE-A
Current date is = Mar 17 08:38:58 IST 2014
Kernal Name = Linux
User is Who I am
Current Directory = 11scs122
Network name and Node name = linuxmint
Kernal Versio n= #1-Ubuntu SMP Fri Apr 16 08:10:02 UTC 2010
Kernal OS = GNU/Linux
kernal release =2.6.32-21-generic
Kernal Processor Type = 2.6.33.85.fc13.i686.PAE
Kernal All Information = Linux main lab 2.6.33.85.fc1.3 i686.PAE
= #1-Ubuntu SMP Fri Apr 16 08:10:02 UTC 2010
I686 i686 i686 GNU/Linux
```

# C PROGRAMS

## IMPLEMENTATION OF PROCESS SCHEDULING MECHANISM

### FIRST COME FIRST SERVE SCHEDULING

#### PROGRAM

```
#include<stdio.h>
#include<conio.h>
void main()
{
int nop,wt[10],tw,tat[10],ttat,i,j,bt[10],t;
float awt,atat;
clrscr();
awt=0.0;
atat=0.0;
printf("Enter the no.of process:");
scanf("%d",&nop);
for(i=0;i<nop;i++)
{
printf("Enter the burst time for process %d: ", i);
scanf("%d",&bt[i]);
}
wt[0]=0;
tat[0]=bt[0];
tw=wt[0];
ttat=tat[0];
for(i=1;i<nop;i++)
{
wt[i]=wt[i-1]+bt[i-1];
tat[i]=wt[i]+bt[i];
tw+=wt[i];
ttat+=tat[i];
}
awt=(float)tw/nop;
atat=(float)ttat/nop;
printf("\nProcessid\tBurstTime\tWaitingTime\tTurnaroundTime\n");
for(i=0;i<nop;i++)
printf("%d\t%d\t%d\t%d\n",i,bt[i],wt[i],tat[i]);
printf("\nTotal Waiting Time:%d\n",tw);
printf("\nTotal Around Time:%d\n",ttat);
printf("\nAverage Waiting Time:%f\n",awt);
printf("\nAverage Total Around Time:%f\n",atat);
getch();
}
```

## SHORT JOB FIRST SCHEDULING

### PROGRAM

```
#include<stdio.h>
#include<conio.h>
void main()
{
int nop,wt[10],tw,tat[10],ttat,i,j,bt[10],t;
float awt,atat;
clrscr();
awt=0.0;
atat=0.0;
printf("Enter the no.of process:");
scanf("%d",&nop);
for(i=0;i<nop;i++)
{
printf("Enter the burst time for process %d: ", i);
scanf("%d",&bt[i]);
}
for(i=0;i<nop;i++)
{
for(j=i+1;j<nop;j++)
{
if(bt[i]>=bt[j])
{
t=bt[i];
bt[i]=bt[j];
bt[j]=t;
}
}
}
wt[0]=0;
tat[0]=bt[0];
tw=wt[0];
ttat=tat[0];
for(i=1;i<nop;i++)
{
wt[i]=wt[i-1]+bt[i-1];
tat[i]=wt[i]+bt[i];
tw+=wt[i];
ttat+=tat[i];
}
awt=(float)tw/nop;
atat=(float)ttat/nop;
printf("\nProcessid\tBurstTime\tWaitingTime\tTurnaroundTime\n");
for(i=0;i<nop;i++)
printf("%d\t%d\t%d\t%d\n",i,bt[i],wt[i],tat[i]);
printf("\nTotal Waiting Time:%d\n",tw);
```

```

printf("\nTotal Around Time:%d\n",ttat);
printf("\nAverage Waiting Time:%f\n",awt);
printf("\nAverage Total Around Time:%f\n",atat);
getch();
}

```

## PRIORITY QUEUE SCHEDULING

### PROGRAM

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int nop,t,wt[10],tw,tat[10],ttat,i,j,p[10],b[10],tmp;
    float awt, atat;
    clrscr();
    awt=0.0;
    atat=0.0;0
    printf("Enter the number of process:");
    scanf("%d",&nop);
    for(i=0;i<nop;i++)
    {
        printf("Enter the burst time of Process %d:",i);
        scanf("%d",&b[i]);
    }
    for(i=0;i<nop;i++)
        printf("Enter the priority number of each Process %d:",i);
        scanf("%d",&p[i]);
    }
    for(i=0;i<nop;i++)
    {
        for(j=i+1;j<nop;j++)
        {
            if(p[i]>p[j])
            {
                t=p[i];
                p[i]=p[j];
                p[j]=t;
                tmp=b[i];
                b[i]=b[j];
                b[j]=tmp;
            }
        }
    }
}

```

```

wt[0]=0;
tat[0]=b[0];
tw=wt[0];
ttat=tat[0];
for(i=1;i<nop;i++)
{
    wt[i]=wt[i-1]+b[i-1];
    tat[i]=wt[i]+b[i];
    tw+=wt[i];
    ttat+=tat[i];
}
awt=(float)tw/nop;
atat=(float)ttat/nop;
printf("Process No:\tPriority:\tBurst Time:\tWaiting Time\tTurnaround Time:\n");
for(i=0;i<nop;i++)
    printf("%d\t%d\t%d\t%d\t%d\n",i,p[i],b[i],wt[i],tat[i]);

printf("Total TurnAround Time:%d\n",ttat);
printf("Total Waiting Time:%d\n",tw);
printf("Average Waiting Time:%f\n",awt);
printf("Average Turnaround Time:%f\n",atat);
getch();
}

```

## READER – WRITER PROBLEM

### PROGRAM

```

#include<stdio.h>
#include<conio.h>
#include<process.h>
void main()
{
    typedef int semaphore;
    semaphore sread=0, swrite=0;
    int ch,r=0;
    clrscr();
    printf("\nReader writer");
    do
    {
        printf("\nMenu");
        printf("\n\t 1.Read from file");
        printf("\n\t 2.Write to file");
        printf("\n\t 3.Exit the reader");
        printf("\n\t 4.Exit the writer");
        printf("\n\t 5.Exit");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
    }
}

```



```

switch(ch)
{
case 1: if(swrite==0)
        {
            sread=1;
            r+=1;
            printf("\nReader %d reads",r);
        }
        else
        {printf("\n Not possible");
        }
        break;
case 2: if(sread==0 && swrite==0)
        {
            swrite=1;
            printf("\nWriter in Progress");
        }
        else if(swrite==1)
        {printf("\nWriter writes the files");
        }
        else if(sread==1)
        {printf("\nCannot write while reader reads the file");
        }
        else
        printf("\nCannot write file");
        break;
case 3: if(r!=0)
        {
            printf("\n The reader %d closes the file",r);
            r-=1;
        }
        else if(r==0)
        {
            printf("\n Currently no readers access the file");
            sread=0;
        }
        else if(r==1)
        {
            printf("\nOnly 1 reader file");
        }
        else
            printf("%d reader are reading the file\n",r);

        break;
case 4: if (swrite==1)
        {
            printf("\nWriter close the file");
            swrite=0;
        }
        else

```

```

        printf("\nThere is no writer in the file");
        break;
case 5: exit(0);
}
}
while(ch<6);
getch();
}

```

## **DINING PHILOSOPHER'S PROBLEM**

### **PROBLEM**

```

#include<stdio.h>
#include<conio.h>
#define LEFT (i+4) %5
#define RIGHT (i+1) %5
#define THINKING 0
#define HUNGRY 1
#define EATING 2
int state[5];
void put_forks(int);
void test(int);
void take_forks(int);
void philosopher(int i)
{
    if(state[i]==0)
    {
        take_forks(i);
        if(state[i]==EATING)
            printf("\n Eating in process....");
        put_forks(i);
    }
}
void put_forks(int i)
{
    state[i]=THINKING;
    printf("\n philosopher %d completed its works",i);
    test(LEFT);
    test(RIGHT);
}
void take_forks(int i)
{
    state[i]=HUNGRY;
    test(i);
}
void test(int i)

```

```

{
if(state[i]==HUNGRY && state[LEFT]!=EATING && state[RIGHT]!=EATING)
{
printf("\n philosopher %d can eat",i);
state[i]=EATING;
}
}
void main()
{
int i;
clrscr();
for(i=1;i<=5;i++)
state[i]=0;
printf("\n\t\t Dining Philosopher Problem");
printf("\n\t\t.....");
for(i=1;i<=5;i++)
{
printf("\n\n the philosopher %d falls hungry\n",i);
philosopher(i);
}
getch();
}

```

## MEMORY MANAGEMENT SCHEME

### PROGRAM

```

#include<stdio.h>
#include<conio.h>
void main()
{
int f3[20],f2[20],r[20],r1[20],ms,bod,sb[20],nsb[20],nsb1[20],np,sp[20];
int f[20],i,j,l,k,z[20],s=0;
clrscr();
printf("enter the memory size:");
scanf("%d",&ms);
printf("\n enter the number of block of division of memory:");
scanf("%d",&bod);
printf("enter the size of each block:");
for(i=1;i<=bod;i++)
{
printf("\nBlock[%d]:",i);
scanf("%d",&sb[i]);
f[i]=1;
f2[i]=1;
f3[i]=1;
r[i]=1;
r1[i]=1;

```

```

z[i]=sb[i];
}
printf("\nenter the number of process:");
scanf("%d",&np);
printf("\nenter the size of each process:");
for(i=1;i<=np;i++)
{
printf("\nprocess[%d]:",i);
scanf("%d",&sp[i]);
}
printf("\n FIRST FIT ");
printf("\n ***** ");
for(i=1;i<=np;i++)
{
for(j=1;j<=bod;j++)
{
if((sb[j]>=sb[i]) && (f[j]!=0))
{
printf("\n Process p[%d] is allocated to Block[%d]",i,j);
f[j]=0;
z[j]=sb[j]-sp[i];
s++;
goto l1;
}
}
printf("\n process p[%d] cannot be allocated",i);
l1:
printf(" ");
}
printf("\n\n Remaining space left in each block \n");
printf("\n ***** \n");
for(i=1;i<=bod;i++)
{
printf("\n Block[%d]: free space =%d",i,z[i]);
}
printf("\n\nUnallocated Blocks");
printf("\n *****");
for(i=1;i<=bod;i++)
{
if(f[i]!=0)
{
printf("\n Block [%d] unallocated",i);
}
}
if(s==bod)
printf("\n No Block is left unallocated");
getch();
clrscr();
s=0;
getch();

```

```

printf("\n\n BEST FIT ");
printf("\n  ***** ");
for(i=2;i<=bod;i++)
{
for(j=1;j<i;j++)
{
if(sb[i]>=sb[j])
r[i]++;
else
r[j]++;
}
}
for(i=1;i<=bod;i++)
{
nsb[r[i]]=sb[i];
z[r[i]]=sb[i];
}
for(i=1;i<=np;i++)
{
for(j=1;j<=bod;j++)
{
if((nsb[j]>=sp[i]) && (f2[j]!=0))
{
for(k=1;k<=bod;k++)
{
if(r[k]==j)
l=k;
}
printf("\nProcess p[%d] is allocated to Block[%d]",i,l);
f2[j]=0;
z[j]=nsb[j]-sp[i];
s++;
goto l2;
}
}
printf("\n process p[%d] cannot be allocated",i);
l2:
printf(" ");
}
printf("\n free space in each block \n");
printf("  ***** \n");
for(i=1;i<=bod;i++)
printf("\nBlock [%d]: free space =%d",i,z[r[i]]);

printf("\n\nUnallocated Blocks");
printf(" \n *****");
for(i=1;i<=bod;i++)
{
if(f2[r[i]]!=0)
{

```

```

printf("\n Block [%d] unallocated",i);
}
}
if(s==bod)
printf("\n No Block is left unallocated");
getch();
clrscr();
s=0;
getch();
printf("\n\n WORST FIT ");
printf("\n ***** ");
for(i=2;i<=bod;i++)
{
for(j=1;j<i;j++)
{
if(sb[i]<=sb[j])
r1[i]++;
else
r1[j]++;
}
}
for(i=1;i<=bod;i++)
{
nsb1[r1[i]]=sb[i];
z[r1[i]]=sb[i];
}
for(i=1;i<=np;i++)
{
for(j=1;j<=bod;j++)
{
if((nsb1[j]>=sp[i]) && (f3[j]!=0))
{
for(k=1;k<=bod;k++)
{
if(r1[k]==j)
l=k;
}
printf("\nProcess p[%d] is allocated to Block[%d]",i,l);
f3[j]=0;
z[j]=nsb1[j]-sp[i];
s++;
goto l3;
}
}
printf("\n process p[%d] cannot be allocated",i);
l3:
printf(" ");
}
printf("\n free space in each block \n");
printf(" ***** \n");

```

```

for(i=1;i<=bod;i++)
printf("\nBlock [%d]: free space =%d",i,z[r1[i]]);

printf("\n\nUnallocated Blocks");
printf(" \n *****");
for(i=1;i<=bod;i++)
{
if(f3[r1[i]]!=0)
{
printf("\n Block [%d] unallocated",i);
}
}
if(s==bod)
printf("\n No Block is left unallocated");
getch();
printf("\n");
}
}

```

## BANKERS ALGORITHM

### PROGRAM

```

#include<stdio.h>
#include<conio.h>
int np,nr,r[10],safe[10],ava[10],aval[10],re[10],f[10],i,j,flag,z,index,pid;
int m[10][10],need[10][10],all[10][10];
void resource()
{
printf("\nEnter the no. of resources: ");
scanf("%d",&nr);
printf("\nEnter the resources instances \n");
for(i=0;i<nr;i++)
scanf("%d",&r[i]);
}
void alloc()
{
printf("\nEnter the no of process: ");
scanf("%d",&np);
for(i=0;i<np;i++)
{
f[i]=0;
for(j=0;j<nr;j++)
{
printf("\n Resource %d for %d ",j+1,i+1);
scanf("%d",&all[i][j]);
}}}
void maxreq()

```

```

{
printf("\nEnter the maximum request for each process \n");
for(i=0;i<np;i++)
for(j=0;j<nr;j++)
scanf("%d",&m[i][j]);
printf("\nThe Available Matrix\n");
printf("-----\n");
for(i=0;i<nr;i++)
{
z=0;
for(j=0;j<np;j++)
z+=all[j][i];
ava[i]=r[i]-z;
printf("%d\t",ava[i]);
aval[i]=ava[i];
}}
void needcal()
{
printf("\n");
printf("\nThe Need Matrix \n");
printf("-----\n");
for(i=0;i<np;i++)
{
printf("\n");
for(j=0;j<nr;j++)
{
need[i][j]=m[i][j]-all[i][j];
printf("%d\t",need[i][j]);
}}
printf("\n\n");
}
void request()
{
flag=0;index=0;
printf("\nEnter the requesting process id:");
scanf("%d",&pid);
printf("\nEnter the resource instance required \n");
for(i=0;i<nr;i++)
{
scanf("%d",&re[i]);
if(re[i]>m[pid][i]);
flag=1;
}
if(flag==0)
{
for(i=0;i<nr;i++)
need[pid][i]=re[i];
for(i=0;i<np;i++)
{
printf("\n");

```



```

for(j=0;j<nr;j++)
printf("%d \t",need[i][j]);
}}
else
{
printf("\n request exceeds maximum request");
exit(0);
}.
}
void out()
{
printf("The safe sequensce is\n");
for(i=0;i<np;i++)
printf("p[%d]\t",safe[i]);
printf("\n\n");
}
void safety()
{
flag=0;i=0;j=0;z=0;index=0;
while(1)
{
if(z++ > 2*np)
{
printf("\n no safe sequence");
exit(0);
}
for(j=0;j<nr;j++)
{
if(need[i][j] <=ava[j]&&f[i] !=1)
{
flag=0;
}
else
{
flag=1;
break;
}
}
if(flag==0)
{
f[i]=1;
safe[index]=i;
for(j=0;j<nr;j++)
ava[j]+= all[i][j];
index++;
if(index >=np)
return;
}
i++;
if(i>=np)

```

```

i=0;
}
}
void main()
{
clrscr();
resource();
alloc();
maxreq();
needcal();
safety();
out();
for(i=0;i<np;i++)
{ fl[i]=0;safe[i]=0;}
request();
for(j=0;j<nr;j++)
ava[j]=aval[j];
safety();
out();
}

```

## **PRODUCER CONSUMER PROBLEM**

### **PROBLEM**

```

#include<stdio.h>
#include<conio.h>
int main()
{
int s,n,b=0,p=0,c=0;
clrscr();
printf("\n producer and consumer problem");
do
{
printf("\n menu");
printf("\n 1.producer an item");
printf("\n 2.consumer an item");
printf("\n 3.add item to the buffer");
printf("\n 4.display status");
printf("\n 5.exit");
printf("\n enter the choice");
scanf("%d",&s);
switch(s)
{
case 1:
p=p+1;
printf("\n item to be produced");
break;

```

```

case 2:
if(b!=0)
{
c=c+1;
b=b-1;
printf("\n item to be consumed");
}
else
{
printf("\n the buffer is empty please wait...");
}
break;
case 3:
if(b<n)
{
if(p!=0)
{
b=b+1;
printf("\n item added to buffer");
}
else
printf("\n no.of items to add...");
}
else
printf("\n buffer is full,please wait");
break;
case 4:
printf("no.of items produced :%d",p);
printf("\n no.of consumed items:%d",c);
printf("\n no.of buffered item:%d",b);
break;
case 5:exit(0);
}
}
while(s<=5);
getch();
return 0;
}

```

## **MEMORY MANAGEMENT SCHEME – PAGING**

### **PROGRAM USING C++**

```

#include<iostream.h>
#include<conio.h>
struct paging
{
    int frameNumber,valid;
};
void main()

```

```

{
    int noOfPages = 0, baseAddress = 0, noOfFrames = 0, sizeOfMM = 0, sizeOfLM = 0,
    FrameSize = 0, physicalAddress = 0, disp = 0, pageNo = 0, logicalAddress = 0;
    paging pageTable[10], frameTable[10];
    clrscr();
    cout << "Program for Paging techniques - Fixed Size partition";
    cout << "\n\nEnter the base address of physical memory:";
    cin >> baseAddress;
    cout << "\nEnter the size of Main Memory:";
    cin >> sizeOfMM;
    cout << "\nEnter the size of Main Memory Frame:";
    cin >> FrameSize;
    noOfFrames = sizeOfMM / FrameSize;
    cout << "\nTotal no. of frames in Main Memory is " << noOfFrames;
    cout << "\n Enter the size of Logical Memory:";
    cin >> sizeOfLM;
    noOfPages = sizeOfLM / FrameSize;
    cout << "\n Total no. of pages in Logical Memory is " << noOfPages;
    cout << "\n Enter the frame values in Page Table\n";
    for(int i = 0; i < noOfPages; i++)
    {
        while(1)
        {
            cout << "\n Page " << i << " is stored in frame number:";
            cin >> pageTable[i].frameNumber;
            pageTable[pageTable[i].frameNumber].valid = 1;
            frameTable[pageTable[i].frameNumber].valid = i;
            break;
        }
    }
    cout << "\n \t PAGE TABLE";
    cout << "\nIndex\t\tFrame Number\tValid_Bit\n\n";
    for(i = 0; i < noOfPages; i++)
    {
        cout << i << "\t\t" << pageTable[i].frameNumber <<
        "\t\t"<<pageTable[pageTable[i].frameNumber].valid << "\t\n";
    }

    for(i=0;i<noOfPages; i++)
    {
        cout << "Enter the logical addresss for mapping process:";
        cin >> logicalAddress;
        pageNo = logicalAddress / FrameSize;
        disp = logicalAddress % FrameSize;
        physicalAddress = baseAddress + ((pageTable[pageNo].frameNumber - 1) *
        FrameSize) + disp;
        cout << "Physical Address value is " << physicalAddress << "\n";
    }
}

```



