Operating system

Day-4:Programmes

Name:G.Tejaswi

Reg.no:192211748

31. Construct a C program to simulate the First in First Out paging technique of memory

management.

#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("\n total page faults:\t%d\n",pagefaults);

return 0;

}

32. Construct a C program to simulate the Least Recently Used paging technique of memory

management.

#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("\n\nTotal Page Faults=%d",faults);

return 0;

}

33. Construct a C program to simulate the optimal paging technique of memory management.

#include<stdio.h>

int main()

{

int no\_of\_frames,no\_of\_pages,frames[10],pages[30],temp[10],flag1,flag2,flag3,i,j,

k,pos,max,faults=0;

printf("enter number of frames: ");

scanf("%d",&no\_of\_frames);

printf("enter number of pages: ");

scanf("%d",&no\_of\_pages);

printf("enter page 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])

{

flag1=flag2=1;

break;

}

}

if(flag1==0)

{

for(j=0;j<no\_of\_frames;j++)

{

if(frames[j]==-1)

{

faults++;

frames[j]=pages[i];

flag2=1;

break;

}

}

}

if(flag2==0)

{

flag3=0;

for(j=0;j<no\_of\_frames;j++)

{

temp[j]=-1;

for(k=i+1;k<no\_of\_pages;k++)

{

if(frames[j]==pages[k])

{

temp[j]=k;

break;

}

}

}

for(j=0;j<no\_of\_frames;j++)

{

if(temp[j]==-1)

{

pos=j;

flag3=1;

break;

}

}

if(flag3==0)

{

max=temp[0];

pos=0;

for(j=1;j<no\_of\_frames;j++)

{

if(temp[j]>max)

{

max=temp[j];

pos=j;

}

}

}

frames[pos]=pages[i];

faults++;

}

printf("\n");

for(j=0;j<no\_of\_frames;j++)

{

printf("%d\t",frames[j]);

}

}

printf("\n\ntotal page faults=%d",faults);

return 0;

}

34. Consider a file system where the records of the file are stored one after another both physically and logically. A record of the file can only be accessed by reading all the previous

records. Design a C program to simulate the file allocation strategy.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int main()

{

int f[50],i,st,len,j,c,k,count=0;

for(i=0;i<50;i++)

f[i]=0;

printf("files allocated are:\n");

x:count=0;

printf("enter starting block and lengthnof files: ");

scanf("%d%d",&st,&len);

for(k=st;k<(st+len);k++)

if(f[k]==0)

count++;

if(len==count)

{

for(j=st;j<(st+len);j++)

if(f[j]==0)

{

f[j]=1;

printf("%d\t%d\n",j,f[j]);

}

if(j!=(st+len-1))

printf("the file is allocated to disk\n");

}

else

printf("the file is not allocated\n");

printf("do you want to enter more file(yes-1\no-0)");

scanf("%d",&c);

if(c==1)

goto x;

else

exit(0);

getch();

}

35. Consider a file system that brings all the file pointers together into an index block. The ith

entry in the index block points to the ith block of the file. Design a C program to simulate the file

allocation strategy.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int main()

{

int f[50],index[50],i,n,st,len,j,c,k,ind,count=0;

for(i=0;i<50;i++)

f[i]=0;

x:printf("enter the index block: ");

scanf("%d",&ind);

if(f[ind]!=1)

{

printf("enter no of blocks needed and no of files for the index %d on the disk:\n",ind);

scanf("%d",&n);

}

else

{

printf("%d index is already allocated\n",ind);

goto x;

}

y:count=0;

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

{

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

if(f[index[i]]==0)

count++;

}

if(count==n)

{

for(j=0;j<n;j++)

f[index[j]]=1;

printf("allocated\n");

printf("file indexed\n");

for(k=0;k<n;k++)

printf("%d---------->%d:%d\n",ind,index[k],f[index[k]]);

}

else

{

printf("file is the index is alredy allocated\n");

printf("enter another file indexed");

goto y;

}

printf("do you want to enter more file(yes-1/no-0)");

scanf("%d",&c);

if(c==1)

goto x;

else

exit(0);

getch();

}

36. With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered

anywhere on the disk. The directory contains a pointer to the first and last blocks of the file. Each

block contains a pointer to the next block. Design a C program to simulate the file allocation

strategy.

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int main()

{

int f[50],p,i,st,len,j,c,k,a;

//clrscr();

for(i=0;i<50;i++)

f[i]=0;

printf("enter how many blocks already allocated: ");

scanf("%d",&p);

printf("enter blocks already allocated: ");

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

{

scanf("%d",&a);

f[a]=1;

}

x:printf("enter index atarting block and length: ");

scanf("%d%d",&st,&len);

k=len;

if(f[st]==0)

{

for(j=st;j<(st+k);j++)

{

if(f[j]==0)

{

f[j]=1;

printf("%d------->%d\n",j,f[j]);

}

else

{

printf("%dblocknis already allocated\n",j);

k++;

}

}

}

else

printf("%d starting block is already allocated\n",st);

printf("do you want to enter more file(yes-1/no-0)");

scanf("%d",&c);

if(c==1)

goto x;

else

exit(0);

getch();

}

37.Construct a C program to simulate the First Come First Served disk scheduling

algorithm.

#include<stdio.h>

#include<stdlib.h>

int main()

{

int ReadyQueue[100],i,n,TotalHeadmov=0,initial;

scanf("%d",&n);

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

{

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

}

scanf("%d",&initial);

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

{

TotalHeadmov=TotalHeadmov+abs(ReadyQueue[i]-initial);

initial=ReadyQueue[i];

}

printf("Total head Movement=%d",TotalHeadmov);

}

38. Design a C program to simulate SCAN disk scheduling algorithm.

#include<stdio.h>

#include<stdlib.h>

#define MAX\_TRACKS 100

void sortArray(int arr[],int size)

{

int i,j,temp;

for(i=0;i<size-1;i++)

{

for(j=0;j<size-i-1;j++)

{

if(arr[j]>arr[j+1])

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

}

void simulateSCAN(int tracks[],int numtracks,int currentposition)

{

int i;

int headmovement=0;

int closesttrack=-1;

sortArray(tracks, numtracks);

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

{

if(tracks[i]>=currentposition)

{

closesttrack=i;

break;

}

}

if(closesttrack==-1)

{

closesttrack=numtracks-1;

}

for(i=closesttrack;i<numtracks;i++)

{

headmovement+=abs(tracks[i]-currentposition);

currentposition=tracks[i];

}

headmovement+=abs(tracks[numtracks-1]-tracks[0]);

currentposition=tracks[0];

headmovement+=abs(tracks[0]-tracks[closesttrack]);

for(i=closesttrack+1;i<numtracks;i++)

{

headmovement+=abs(tracks[i]-currentposition);

currentposition=tracks[i];

}

float averageheadmovement=(float)headmovement/numtracks;

printf("average headmovement:%.2f\n",averageheadmovement);

}

int main()

{

int tracks[MAX\_TRACKS];

int numtracks;

int currentposition;

printf("enter the number of tracks: ");

scanf("%d",&numtracks);

printf("enter the track positions: ");

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

{

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

}

printf("enter the current position of the head: ");

scanf("%d",&currentposition);

simulateSCAN(tracks,numtracks,currentposition);

return 0;

}

39. Develop a C program to simulate C-SCAN disk scheduling algorithm.

#include<stdio.h>

#include<stdlib.h>

#define MAX\_TRACKS 100

void sortarray(int arr[],int size)

{

int i,j,temp;

for(i=0;i<size-1;i++)

{

for(j=0;j<size-i-1;j++)

{

if(arr[j]>arr[j+1])

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

}

int findclosesttrack(int tracks[],int numtracks,int currentposition,int direction)

{

int i;

int closesttrack=-1;

if(direction==1)

{

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

{

if(tracks[i]>=currentposition)

{

closesttrack=i;

break;

}

}

if(closesttrack==-1)

{

closesttrack=numtracks-1;

}

else

{

for(i=numtracks-1;i>=0;i--)

{

if(tracks[i]<=currentposition)

{

closesttrack=i;

break;

}

}

if(closesttrack==-1)

{

closesttrack=0;

}

}

return closesttrack;

}

}

void simulateCSCAN(int tracks[],int numtracks,int currentposition)

{

int i,direction=1;

int headmovement=0;

sortarray(tracks, numtracks);

int closesttrack=findclosesttrack(tracks,numtracks,currentposition,direction);

for(i=closesttrack;i<numtracks;i++)

{

headmovement+=abs(tracks[i]-currentposition);

currentposition=tracks[i];

}

headmovement+=abs(tracks[numtracks-1]-tracks[0]);

currentposition=tracks[0];

headmovement+=abs(tracks[0]-tracks[closesttrack]);

for(i=closesttrack+1;i<numtracks;i++)

{

headmovement+=abs(tracks[i]-currentposition);

currentposition=tracks[i];

}

float averageheadmovement=(float)headmovement/numtracks;

printf("average head movement:%.2f\n",averageheadmovement);

}

int main()

{

int tracks[MAX\_TRACKS];

int numtracks;

int currentposition;

printf("enter the number of tracks: ");

scanf("%d",&numtracks);

printf("enter the track positions: ");

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

{

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

}

printf("enter the current position of head: ");

scanf("%d",&currentposition);

simulateCSCAN(tracks,numtracks,currentposition);

return 0;

}

40. Illustrate the various File Access Permission and different types users in Linux.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_LINE\_LENGTH 256

int main(int argc, char \*argv[])

{

if (argc != 2) {

fprintf(stderr, "Usage: %s <pattern>\n", argv[0]);

exit(EXIT\_FAILURE);

}

char pattern[MAX\_LINE\_LENGTH];

strcpy(pattern, argv[1]);

char line[MAX\_LINE\_LENGTH];

while (fgets(line, sizeof(line), stdin) != NULL)

{

if (strstr(line, pattern) != NULL) {

printf("%s", line);

}

}

return 0;

}