Assignment

**iCSE316**

**i**

**iSubmission:i8thiAprili2020i**

**iSection-K18AW**

**Name-Anujikumar**

**RegistrationiNo:-11802721i**

**Email-anujdwivedi88699@gmail.comi**

**RolliNo:-62**

**Githubilink:-https://github.com/settings/admin**

**Solutionino:-10**

1. WriteiaiCiprogramitoisolveitheifollowingiproblem:iSupposeithatiaidiskidriveihasi5,000icylinders,inumberedi0itoi4999.iTheidriveiisicurrentlyiservingiairequestuestiaticylinder143,ianditheipreviousirequestuestiwasiaticylinderi125.iTheiqueueiofipendingirequestuests,iiniFIFO

order,is:

86,i1470,i913,i1774,i948,i1509,i1022,i1750,i130

#include<stdio.h>  
#include<stdlib.h>  
intimain()  
{  
iintihd,j,n;  
iprintf("\nEnteriheadiposition");  
iscanf("%d",&hd);i//headipointsitoicurrentiposition  
iprintf("\nEnterinumberiofirequestuests");  
iscanf("%d",&n);  
iintirequest[n];i//requestiisitheirequestuestilist  
ifor(j=0;j<n;i++)  
i{  
iscanf("%d",&request[j]);  
i}  
iintidifference=request[0]-hd;  
iif(difference<0)  
i{  
idifference=difference\*-1;  
i}  
ifor(j=1;j<n;i++)  
i{  
iif((request[j]-request[j-1])>0)  
idifference=difference+(request[j]-request[j-1]);  
ielse  
idifference=difference+(request[j-1]-request[j]);  
i}  
iprintf("Seekingitimer=:i%d\n",difference);  
igetch();  
}

**Explanation:**

**Librariesiused:**

**i**<stdio>i:ithisilibraryiisiusediforiinputiandioutput.

i<stdlib>i:ithisilibraryiisiusediforistring.

Initheiquestionivaluesiareigivenibutitheyiareiconstantivaluesibuti

Initheicodeiwritteniaboveiweicaniactuallyichangeitheivaluesiasi

i

Periourirequestuirement.Theirequestsiareifullfillediinitheiorderiiniwhichitheyicome.Thisialgorithmidoesinoticauseistarvationiproblem.

Totaliheadimovementsithatioccuriwhileiservingitheseirequestsiare:

i(125-86)+(1470-86)+(1470-913)+(1774-913)+(1774-948)+(1509-948)+

(1509-1022)+(1750-1022)+(1750-130)

=>7063

0i500i1000i1500i2000i2500i3000i3500i4000i4999

Complexityiofiaboveicode:in\*n

I am attempting to write a program that calculates various statistical priority values for a list of numbers. This program will be passed a series of numbers on the command line and will then create three separate worker threads. One thread will determine the average, the second will find the maximum, and the third will find the minimum.

This is my code. It compiles and runs, but crashes with a segmentation fault.

The variables representing the average, minimum, and maximum values will be stored  
globally. The worker threads will set these values, and the parent thread will output the  
values once the workers have exited.

This program will be passed a series of numbers on the command line and will then create three separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third determine the minimum value.

**Solutionino:-17**

1. DesigniaischedulingiprogramitoiimplementsiaiQueueiwithitwoilevels:

Leveli1i:iFixedipriorityipreemptiveiScheduling

Leveli2:iRoundiRobiniScheduling

ForiaiFixedipriorityipreemptiveiSchedulingi(Queue1),itheiPriorityi0iisihighestipriority.iIfioneiprocessiP1iisischedulediandirunning,ianotheriprocessiP2iwithihigheripriorityicomes.iTheiNewiprocessi(highipriority)iprocessiP2ipreemptsicurrentlyirunningiprocessiP1iandiprocessiP1iwilligoitoisecondileveliqueue.iTimeiforiwhichiprocessiwillistrictlyiexecuteimustibeiconsiderediinitheimultiplesiofi2.iAllitheiprocessesiinisecondileveliqueueiwillicompleteitheiriexecutioniaccordingitoiroundirobinischeduling.

Consider:i1.iQueuei2iwillibeiprocessediafteriQueuei1ibecomesiempty.i

2.iPriorityiofiQueuei2ihasiloweripriorityithaniiniQueuei1.

Code:

#include<stdio.h>

#include<string.h>

#include<conio.h>

voidimain()

{

icharip[10][5],temp[5];

iintii,j,pt[10],wt[10],totwt=0,pr[10],temp1,n;

ifloatiavgwt;

iprintf("enterinoiofiprocesses:");

iscanf("%d",&n);

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

i{

iprintf("enteriprocess%diname:",i+1);

iscanf("%s",&p[i]);

iprintf("enteriprocessitime:");

iscanf("%d",&pt[i]);

iprintf("enteripriority:");

iscanf("%d",&pr[i]);

i}

ifor(i=0;i<n-1;i++)

i{

ifor(j=i+1;j<n;j++)

i{

iif(pr[i]>pr[j])

i{

itemp1=pr[i];

ipr[i]=pr[j];

ipr[j]=temp1;

itemp1=pt[i];

ipt[i]=pt[j];

ipt[j]=temp1;

istrcpy(temp,p[i]);

istrcpy(p[i],p[j]);

istrcpy(p[j],temp);

i}

i}

i}

iwt[0]=0;

ifor(i=1;i<n;i++)

i{

iwt[i]=wt[i-1]+wt[i-1];

itotwt=totwt+wt[i];

i}

iavgwt=(float)totwt/n;

iprintf("p\_name**\t**ip\_time**\t**ipriority**\t**iw\_time**\n**");

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

i{

iprintf("i%s**\t**i%d**\t**i%d**\t**i%d**\n**"i,p[i],pt[i],pr[i],wt[i]);

i}

iprintf("totaliwaitingitime=%d**\n**iavgiwaitingitime=%f",totwt,avgwt);

i

iintits,pid[10],need[10],wt1[10],tat[10],i1,j1,n2,n1;

iintibt[10],flag[10],ttat=0,twt=0;

ifloatiawt,atat;

iprintf("**\n**EnteritheinumberiofiProcessorsi**\n**");

iscanf("%d",&n);

in1=n;

iprintf("**\n**iEnteritheiTimeslicei**\n**");

iscanf("%d",&ts);

ifor(i=1;i<=n;i++)

i{

iprintf("**\n**iEnteritheiprocessiIDi%d",i);

iscanf("%d",&pid[i]);

iprintf("**\n**iEnteritheiBurstiTimeiforitheiprocess");

iscanf("%d",&bt[i]);

ineed[i]=bt[i];

i}

ifor(i=1;i<=n;i++)

i{

iflag[i]=1;

iwt[i]=0;

i}

iwhile(n!=0)

i{

ifor(i=1;i<=n;i++)

i{

iif(need[i]>=ts)

i{

ifor(j=1;j<=n;j++)

i{

iif((i!=j)&&(flag[i]==1)&&(need[j]!=0))

iwt[j]+=ts;

i}

ineed[i]-=ts;

iif(need[i]==0)

i{

iflag[i]=0;

in--;

i}

i}

ielse

i{

ifor(j=1;j<=n;j++)

i{

iif((i!=j)&&(flag[i]==1)&&(need[j]!=0))

iwt[j]+=need[i];

i}

ineed[i]=0;

in--;

iflag[i]=0;

i}

i}

i}

ifor(i=1;i<=n1;i++)

i{

itat[i]=wt[i]+bt[i];

itwt=twt+wt[i];

ittat=ttat+tat[i];

i}

iawt=(float)twt/n1;

iatat=(float)ttat/n1;

iprintf("**\n\n**iProcessi**\t**iProcessiIDi**\t**iBurstTimei**\t**iWaitingiTimei**\t**iTurnaroundTimei**\n**i");

ifor(i=1;i<=n1;i++)

i{

iprintf("**\n**i%5di**\t**i%5di**\t\t**i%5di**\t\t**i%5di**\t\t**i%5di**\n**",ii,pid[i],bt[i],wt[i],tat[i]);

i}

iprintf("**\n**iTheiaverageiWaitingiTime=4.2f",awt);

iprintf("**\n**iTheiaverageiTurniaroundiTime=4.2f",atat);

}

**Explanation:**

**Librariesiused:**

**i**<stdio>i:ithisilibraryiisiusediforiinputiandioutput.

i<stdlib>i:ithisilibraryiisiusediforistring.

Fixedipriorityipreumptiveischeduling

Itiisiaischedulingisystemiusediinirealitimeisystems.

Complexityiofiaboveicode:inlogn

**Round Robin** is a scheduling algorithm in cpu where each process is assigns a fixed time slot in a cyclic way. ... One of the most commonly used technique in CPU scheduling as a core. It is preeumptive as processses are assigns cpu only for a fixed amount of time at most.