21.dining philosopher

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <unistd.h>

#define N 5

#define LEFT (i + N - 1) % N

#define RIGHT (i + 1) % N

#define THINKING 0

#define HUNGRY 1

#define EATING 2

int state[N];

sem\_t mutex;

sem\_t s[N];

void test(int i) {

if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {

state[i] = EATING;

sem\_post(&s[i]);

}

}

void take\_forks(int i) {

sem\_wait(&mutex);

state[i] = HUNGRY;

test(i);

sem\_post(&mutex);

sem\_wait(&s[i]);

}

void put\_forks(int i) {

sem\_wait(&mutex);

state[i] = THINKING;

test(LEFT);

test(RIGHT);

sem\_post(&mutex);

}

void \*philosopher(void \*num) {

int i = \*(int\*)num;

while (1) {

printf("Philosopher %d is thinking\n", i + 1);

sleep(1);

take\_forks(i);

printf("Philosopher %d is eating\n", i + 1);

sleep(1);

put\_forks(i);

}

}

int main() {

int i;

pthread\_t thread\_id[N];

int num[N];

sem\_init(&mutex, 0, 1);

for (i = 0; i < N; i++) {

sem\_init(&s[i], 0, 0);

}

for (i = 0; i < N; i++) {

num[i] = i;

pthread\_create(&thread\_id[i], NULL, philosopher, &num[i]);

}

for (i = 0; i < N; i++) {

pthread\_join(thread\_id[i], NULL);

}

}

22.two level directory system

#include<string.h>

#include<stdlib.h>

#include<stdio.h>

struct

{

char dname[10],fname[10][10];

int fcnt;

}dir[10];

int main()

{

int i,ch,dcnt,k;

char f[30], d[30];

dcnt=0;

while(1)

{

printf("\n\n1. Create Directory\t2. Create File\t3. Delete File");

printf("\n4. Search File\t\t5. Display\t6. Exit\tEnter your choice -- ");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nEnter name of directory -- ");

scanf("%s", dir[dcnt].dname);

dir[dcnt].fcnt=0;

dcnt++;

printf("Directory created");

break;

case 2: printf("\nEnter name of the directory -- ");

scanf("%s",d);

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

if(strcmp(d,dir[i].dname)==0)

{

printf("Enter name of the file -- ");

scanf("%s",dir[i].fname[dir[i].fcnt]);

printf("File created");

break;

}

if(i==dcnt)

printf("Directory %s not found",d);

break;

case 3: printf("\nEnter name of the directory -- ");

scanf("%s",d);

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

{

if(strcmp(d,dir[i].dname)==0)

{

printf("Enter name of the file -- ");

scanf("%s",f);

for(k=0;k<dir[i].fcnt;k++)

{

if(strcmp(f, dir[i].fname[k])==0)

{

printf("File %s is deleted ",f);

dir[i].fcnt--;

strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);

goto jmp;

}

}

printf("File %s not found",f);

goto jmp;

}

}

printf("Directory %s not found",d);

jmp : break;

case 4: printf("\nEnter name of the directory -- ");

scanf("%s",d);

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

{

if(strcmp(d,dir[i].dname)==0)

{

printf("Enter the name of the file -- ");

scanf("%s",f);

for(k=0;k<dir[i].fcnt;k++)

{

if(strcmp(f, dir[i].fname[k])==0)

{

printf("File %s is found ",f);

goto jmp1;

}

}

printf("File %s not found",f);

goto jmp1;

}

}

printf("Directory %s not found",d);

jmp1: break;

case 5: if(dcnt==0)

printf("\nNo Directory's ");

else

{

printf("\nDirectory\tFiles");

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

{

printf("\n%s\t\t",dir[i].dname);

for(k=0;k<dir[i].fcnt;k++)

printf("\t%s",dir[i].fname[k]);

}

}

break;

default:exit(0);

return 0;

}

}

}

23. scan disk scheduling

#include<stdio.h>

int absoluteValue(int);

int main()

{

int queue[25],n,headposition,i,j,k,seek=0, maxrange,

difference,temp,queue1[20],queue2[20],temp1=0,temp2=0;

float averageSeekTime;

printf("Enter the maximum range of Disk: ");

scanf("%d",&maxrange);

printf("Enter the number of queue requests: ");

scanf("%d",&n);

printf("Enter the initial head position: ");

scanf("%d",&headposition);

printf("Enter the disk positions to be read(queue): ");

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

{

scanf("%d",&temp);

if(temp>headposition)

{

queue1[temp1]=temp;

temp1++;

}

else

{

queue2[temp2]=temp;

temp2++;

}

}

for(i=0;i<temp1-1;i++)

{

for(j=i+1;j<temp1;j++)

{

if(queue1[i]>queue1[j])

{

temp=queue1[i];

queue1[i]=queue1[j];

queue1[j]=temp;

}

}

}

for(i=0;i<temp2-1;i++)

{

for(j=i+1;j<temp2;j++)

{

if(queue2[i]<queue2[j])

{

temp=queue2[i];

queue2[i]=queue2[j];

queue2[j]=temp;

}

}

}

for(i=1,j=0;j<temp1;i++,j++)

{

queue[i]=queue1[j];

}

queue[i]=maxrange;

for(i=temp1+2,j=0;j<temp2;i++,j++)

{

queue[i]=queue2[j];

}

queue[i]=0;

queue[0]=headposition;

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

{

difference = absoluteValue(queue[j+1]-queue[j]);

seek = seek + difference;

printf("Disk head moves from position %d to %d with Seek %d \n",

queue[j], queue[j+1], difference);

}

averageSeekTime = seek/(float)n;

printf("Total Seek Time= %d\n", seek);

printf("Average Seek Time= %f\n", averageSeekTime);

}

int absoluteValue(int x)

{

if(x>0)

{

return x;

}

else

{

return x\*-1;

}

return 0;

}