|  |
| --- |
| #include<stdio.h> |
|  | #include<stdlib.h> |
|  | #include<strings.h> |
|  | #define MAX 50 |
|  | #define INF 32767 |
|  | int time; |
|  | typedef struct process |
|  | { |
|  | int pid,AT,BT,CT,TAT,WT,RT,type; |
|  | }process; |
|  | typedef struct queue{ |
|  | int q[MAX]; |
|  | int f,r; |
|  | }queue; |
|  |  |
|  | void display(process p[],int n) |
|  | { |
|  | int i; |
|  | printf("Pid AT BT CT TAT WT type\n"); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("%3d %2d %2d %2d %3d %2d %d\n",p[i].pid,p[i].AT,p[i].BT,p[i].CT,p[i].TAT,p[i].WT,p[i].type); |
|  | } |
|  | printf("\n"); |
|  | } |
|  |  |
|  | void insert(queue \*t,int ele) |
|  | { |
|  | t->r++; |
|  | t->q[t->r] = ele; |
|  | } |
|  | int queueFront(queue \*t){ |
|  | return t->q[t->f]; |
|  | } |
|  | int delete1(queue \*t){ |
|  | int i=t->f; |
|  | int z=t->q[t->f]; |
|  | while(i < t->r){ |
|  | t->q[i] = t->q[i+1]; |
|  | i++; |
|  | } |
|  | t->r--; |
|  | return z; |
|  | } |
|  |  |
|  | void sort(process p[],int n) |
|  | { |
|  | int i,j; |
|  | process val; |
|  | for(i=0;i<n-1;i++) |
|  | { |
|  | val = p[i+1]; |
|  | for(j=i;j>=0;j--) |
|  | if(val.type<p[j].type) |
|  | p[j+1] = p[j]; |
|  | else |
|  | break; |
|  | p[j+1] = val; |
|  | } |
|  | } |
|  | void sortAT(process p[],int n) |
|  | { |
|  | int i,j; |
|  | process val; |
|  | for(i=0;i<n-1;i++) |
|  | { |
|  | val = p[i+1]; |
|  | for(j=i;j>=0;j--) |
|  | if(val.AT<p[j].AT) |
|  | p[j+1] = p[j]; |
|  | else |
|  | break; |
|  | p[j+1] = val; |
|  | } |
|  | } |
|  | int minRT(process p[],int n,int time) |
|  | { |
|  | int min=INF,pos; |
|  | for(int i=0;i<n;i++) |
|  | { |
|  | if(p[i].type==1) |
|  | if(p[i].AT <= time&&p[i].RT != 0) |
|  | if(p[i].RT<min) |
|  | { |
|  | min = p[i].RT; |
|  | pos = i; |
|  | } |
|  | if(p[i].AT>time) |
|  | break; |
|  | } |
|  | return pos; |
|  | } |
|  | int srtn(process p[],int n,int time) |
|  | { |
|  | int i = minRT(p,n,time); |
|  | p[i].RT--; |
|  | if(p[i].RT == 0) |
|  | { |
|  | p[i].CT = time+1; |
|  | p[i].TAT = p[i].CT - p[i].AT; |
|  | p[i].WT = p[i].TAT - p[i].BT; |
|  | } |
|  | return p[i].pid; |
|  | } |
|  | int fcfs(process p[],int n,int time) |
|  | { |
|  | int i; |
|  | for(i=0;i<n;i++) |
|  | { |
|  | if(p[i].type == 3&&p[i].RT != 0) |
|  | break; |
|  | } |
|  | p[i].RT--; |
|  | if(p[i].RT == 0) |
|  | { |
|  | p[i].CT = time+1; |
|  | p[i].TAT = p[i].CT - p[i].AT; |
|  | p[i].WT = p[i].TAT - p[i].BT; |
|  | } |
|  | return p[i].pid; |
|  | } |
|  | int rr(process p[],queue \*t,int tq,int n,int time) |
|  | { |
|  | int i,runTime,cp;//cp -> currnet process |
|  | cp = queueFront(t); |
|  | for(i=0;i<n;i++) |
|  | if(p[i].pid == cp) |
|  | break; |
|  | p[i].RT--; |
|  | runTime = p[i].BT-p[i].RT; |
|  | if(runTime%tq == 0 && p[i].RT != 0) |
|  | { |
|  | delete1(t); |
|  | insert(t,cp); |
|  | } |
|  | if(p[i].RT == 0) |
|  | { |
|  | p[i].CT = time+1; |
|  | p[i].TAT = p[i].CT - p[i].AT; |
|  | p[i].WT = p[i].TAT - p[i].BT; |
|  | delete1(t); |
|  | } |
|  | return p[i].pid; |
|  | } |
|  | float avgTAT(process p[],int n) |
|  | { |
|  | int i; |
|  | float avg=0; |
|  | for(i=0;i<n;i++) |
|  | avg += p[i].TAT; |
|  | avg = avg / n; |
|  | return avg; |
|  | } |
|  | float avgWT(process p[],int n) |
|  | { |
|  | int i; |
|  | float avg=0; |
|  | for(i=0;i<n;i++) |
|  | avg += p[i].WT; |
|  | avg = avg / n; |
|  | return avg; |
|  | } |
|  | void mlqueue(process p[],int n,int tq,int chart[]) |
|  | { |
|  | int t,i,j,T=0,hp; //T --> total time, hp --> highest priority |
|  | int cur\_pid; |
|  | queue x; |
|  | x.f = 0; |
|  | x.r = -1; |
|  | for(i=0;i<n;i++) |
|  | T += p[i].BT; |
|  | sortAT(p,n); |
|  | sort(p,n); |
|  | for(t=0;t<T;t++) |
|  | { |
|  | for(i=0;i<n;i++) |
|  | if(p[i].type == 2&&p[i].AT==t) |
|  | insert(&x,p[i].pid); |
|  | for(i=0;i<n;i++) |
|  | if(p[i].AT <= t&&p[i].RT != 0) |
|  | break; |
|  | if(i==n) |
|  | break; |
|  | switch(p[i].type) |
|  | { |
|  | case 1 :cur\_pid = srtn(p,n,t); |
|  | break; |
|  | case 2 :cur\_pid = rr(p,&x,tq,n,t); |
|  | break; |
|  | case 3 :cur\_pid = fcfs(p,n,t); |
|  | break; |
|  | default:printf("Invalid Type!\n"); |
|  | exit(1); |
|  | } |
|  | chart[t] = cur\_pid; |
|  | } |
|  | } |
|  | void saveOutput(process p[],int n,float avg\_tat,float avg\_wt,int chart[]) |
|  | { |
|  | FILE \*fp; |
|  | char fileName[34]; |
|  | int T=0; |
|  | for(int i=0;i<n;i++) |
|  | T += p[i].BT; |
|  |  |
|  | printf("Enter file name: "); |
|  | fflush(stdin); |
|  | scanf("%s",&fileName); |
|  | strcat(fileName, ".txt"); |
|  | fp = fopen(fileName,"w"); |
|  | if(fp == NULL) |
|  | { |
|  | printf("cannot open file\n"); |
|  | exit(4); |
|  | } |
|  | fprintf(fp,"%d\n",n); |
|  | for(int i=0;i<n;i++) |
|  | { |
|  | fprintf(fp,"%3d %2d %2d ",p[i].pid,p[i].AT,p[i].BT); |
|  | fprintf(fp,"%2d %3d %2d ",p[i].CT,p[i].TAT,p[i].WT); |
|  | fprintf(fp,"%d\n",p[i].type); |
|  | } |
|  | for(int i=0;i<T;i++) |
|  | fprintf(fp,"%d ",chart[i]); |
|  | fprintf(fp,"\n%f %f\n",avg\_tat,avg\_wt); |
|  | fclose(fp); |
|  | } |
|  | void main() |
|  | { |
|  | int i,n,q,chart[MAX]; |
|  | process p[MAX]; |
|  | float avg\_tat,avg\_wt; |
|  | printf("Enter no. of processes: "); |
|  | scanf("%d",&n); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("Enter arrival and burst time for p%d: ",i+1); |
|  | scanf("%d%d",&p[i].AT,&p[i].BT); |
|  | printf("Enter type of process: 1.System 2.Interactive 3.Batch\n>>>"); |
|  | scanf("%d",&p[i].type); |
|  | p[i].pid = i+1; |
|  | p[i].RT = p[i].BT; |
|  | } |
|  | printf("Enter time quantum: "); |
|  | scanf("%d",&q); |
|  | mlqueue(p,n,q,chart); |
|  | display(p,n); |
|  | avg\_tat = avgTAT(p,n); |
|  | printf("average TAT = %.2f ms\n",avg\_tat); |
|  | avg\_wt = avgWT(p,n); |
|  | printf("average WT= %.2f ms\n",avg\_wt); |
|  | saveOutput(p,n,avg\_tat,avg\_wt,chart); |
|  | } |