OPERATING SYSTEM PROJECT

NAME: E.SRIRAM

Section: K1619

Reg no:11603985

Roll no: b-30

Q4. Design a scheduling program to implements a Queue with two levels: Level 1 : Fixed priority preemptive Scheduling Level 2 : Round Robin Scheduling For a Fixed priority preemptive Scheduling (Queue 1), the Priority 0 is highest priority. If one process P1 is scheduled and running, another process P2 with higher priority comes. The New process (high priority) process P2 preempts currently running process P1 and process P1 will go to second level queue. Time for which process will strictly execute must be considered in the multiples of 2. All the processes in second level queue will complete their execution according to round robin scheduling.

Consider: 1. Queue 2 will be processed after Queue 1 becomes empty. 2. Priority of Queue 2 has lower priority than in Queue 1.

#include<iostream>

#include<algorithm>

using namespace std;

struct process{

int id, arrival\_time, burst\_time, priority, status;

}p[10];

bool compare(process a, process b)

{

return a.arrival\_time < b.arrival\_time;

}

int main()

{

int n,i;

cout<<"Enter the number of processes:\n";

cin>>n;

cout<<"Enter the arrival time of each process:\n";

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

{

cout<<"Enter the arrival time of P"<<i+1<<":"<<endl;

cin>>p[i].arrival\_time;

p[i].id = i+1;

}

cout<<"Enter the burst time of each process:\n";

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

{

cout<<"Enter the burst time of P"<<i+1<<":"<<endl;

cin>>p[i].burst\_time;

}

cout<<"Enter the priority of each process:\n";

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

{

cout<<"Enter the priority of P"<<i+1<<":"<<endl;

cin>>p[i].priority;

}

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

p[i].status = 0;

sort(p, p+n, compare);

int time = 0;

int count = 0, count2 = 0;

i=0;

while(count!=n)

{

if(p[i].status!=1 && p[i].status!=2)

{

cout<<"Processing "<<p[i].id<<endl;

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

{

if(p[j].status != 1 && p[j].status!=2 && (p[j].priority < p[i].priority) && (p[j].arrival\_time <= time))

{

cout<<"Process P"<<p[i].id<<" is interuppted by P"<<p[j].id<<".Pushing it into queue2"<<endl;

cout<<"Process P"<<p[j].id<<" is processing"<<endl;

p[i].status = 2;

count2++;

time += p[j].burst\_time;

p[j].status = 1;

cout<<"Process P"<<p[j].id<<" is completely processed"<<endl;

count += 2;

break;

}

}

if(p[i].status != 2)

{

time += p[i].burst\_time;

p[i].status = 1;

cout<<"Process P"<<p[i].id<<" is completely processed"<<endl;

count++;

}

}

i = (i+1)%n;

}

int tq = 2;

time = 0;

count = 0;

cout<<"Executing Queue2"<<endl;

i=0;

while(count != count2)

{

if(p[i].status == 2)

{

if(p[i].burst\_time<2)

{

p[i].burst\_time -= 1;

time += 1;

p[i].status = 1;

cout<<"Completed P"<<p[i].id<<endl;

count++;

}

else

{

p[i].burst\_time -= 2;

time += tq;

cout<<"Processing P"<<p[i].id<<endl;

}

}

i = (i+1)%n;

}

return 0;

}

