用c语言实现对n个进程采用动态优先权算法的调度详细代码：

/\*define the structure of process\*/

#include <stdio.h>

#define N 10

#define P proc

struct process{

int id;

int py; /\*priority\*/

int ct; /\*cputime\*/

int at; /\*alltime\*/

int sb; /\*startblock\*/

int bt; /\*blocktime\*/

int st; /\*state:1 ready;-1 block;0 finish\*/

};

struct process proc[N];

int main(){

int i,j,n;

int max\_pt,priority;

int timeslice=0;

int ready\_q[N],block\_q[N];

char ch;

FILE \*fp;

char \*fname="c:\\a.txt";

fp=fopen(fname,"w+");

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

ready\_q[i]=0; block\_q[i]=0;

}

n=5; /\* 定义进程数量为5 \*/

/\* 初始化进程数组 \*/

P[1].id=0; P[1].py=9; P[1].ct=0; P[1].at=3; P[1].sb=2; P[1].bt=3;

P[2].id=1; P[2].py=38; P[2].ct=0; P[2].at=3; P[2].sb=-1; P[2].bt=0;

P[3].id=2; P[3].py=30; P[3].ct=0; P[3].at=6; P[3].sb=-1; P[3].bt=0;

P[4].id=3; P[4].py=29; P[4].ct=0; P[4].at=3; P[4].sb=-1; P[4].bt=0;

P[5].id=4; P[5].py=0; P[5].ct=0; P[5].at=4; P[5].sb=-1; P[5].bt=0;

/\* 输出初始化的进程数组内容\*/

fprintf(fp,"\n\n RUNNING RPOC:%d\n",-1);

fprintf(fp," READY\_QUEUE :");

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

fprintf(fp,"->id%d",i-1);

fprintf(fp,"\n");

fprintf(fp," BLOCK\_QUEUE :\n ");

fprintf(fp,"===================================================== ===");

fprintf(fp,"\n %-12s","ID");

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

fprintf(fp,"%8d",i-1);

fprintf(fp,"\n %-12s","PRIORITY");

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

fprintf(fp,"%8d",P[i].py);

fprintf(fp,"\n %-12s","CPUTIME");

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

fprintf(fp,"%8d",P[i].ct);

fprintf(fp,"\n %-12s","ALLTIME");

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

fprintf(fp,"%8d",P[i].at);

fprintf(fp,"\n %-12s","STARTBLOCK");

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

fprintf(fp,"%8d",P[i].sb);

fprintf(fp,"\n %-12s","BLOCKTIME");

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

fprintf(fp,"%8d",P[i].bt);

fprintf(fp,"\n %-12s","STATE");

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

P[i].st=1;

fprintf(fp,"%8s","READY");

}

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

P[i].st=1;

ready\_q[i]=i;

}

ready\_q[0]=n;

/\* ready\_q[0]表示就绪队列中进程个数，block\_q[0]表示阻塞队列中进程个数\*/

/\* 进行运算\*/

do{

timeslice++; /\* 时间片加1 \*/

/\*

max\_pt记录优先级最高进程的pt,ready\_q数组记录就绪队列的顺序

priority记录优先级最高进程的ID

此程序段使ready\_q排序

\*/

for(i=1,max\_pt=-1;i<=ready\_q[0];i++)

if (P[ready\_q[i]].py>max\_pt){

max\_pt=P[ready\_q[i]].py;

j=i;

priority=ready\_q[i];

}

for(i=j;i<ready\_q[0];i++)

ready\_q[i]=ready\_q[i+1];

ready\_q[0]--;

/\* 按原则操作各进程，实质是按原则改变数组各值\*/

P[priority].py-=3;

if (P[priority].py<0) P[priority].py=0;

P[priority].ct++;

P[priority].at--;

/\* 除执行进程外，其他进程的操作\*/

for(i=1;i<=ready\_q[0];i++)

P[ready\_q[i]].py+=1;

/\* 对阻塞队列和就绪队列赋值\*/

/\*? 进程被阻塞的时间BLOCKTIME，表示已阻塞的进程再等待BLOCKTIME个时间片后，进程将转换成就绪状态；\*/

for(i=1;i<=block\_q[0];i++)

{

P[block\_q[i]].bt-=1; /\* 阻塞队列中的进程的blocktime－1 \*/

if (P[block\_q[i]].bt==0) /\* 如果该进程的blocktime＝0，说明该进程需要进入就绪队列\*/

{ P[block\_q[i]].st=1; /\* 将该进程的st状态置1，将其从阻塞队列移入就绪队列\*/

block\_q[0]--;

ready\_q[0]++;

ready\_q[ready\_q[0]]=i;

}

}

if(P[priority].at==0) /\* 对当前执行进程，如果at＝0说明执行完毕\*/

{

P[priority].st=0;

P[priority].py=-1;

}

else if (P[priority].sb==P[priority].ct) /\* 进程的阻塞时间STARTBLOCK，\*/

/\* 表示当进程再运行STARTBLOCK个时间片后，\*/

/\* 进程将进入阻塞状态；\*/

{

P[priority].st=-1;

block\_q[0]++;

block\_q[block\_q[0]]=priority;

}

else /\* 以上两个条件都不符合，直接进入就绪队列\*/

{

ready\_q[0]++;

ready\_q[ready\_q[0]]=priority;

}

/\* 运行一次后输出运行结果\*/

fprintf(fp,"\n\n RUNNING RPOC:%d TIMELICE:%d\n",priority-1,timeslice);

fprintf(fp," READY\_QUEUE :");

for(i=1;i<=ready\_q[0];i++)

fprintf(fp,"->id%d",ready\_q[i]-1);

fprintf(fp,"\n");

fprintf(fp," BLOCK\_QUEUE :");

for(i=1;i<=block\_q[0];i++)

fprintf(fp,"->id%d",block\_q[i]-1);

fprintf(fp,"\n");

fprintf(fp,"===================================================== ===");

fprintf(fp,"\n %-12s","ID");

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

fprintf(fp,"%8d",i-1);

fprintf(fp,"\n %-12s","PRIORITY");

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

fprintf(fp,"%8d",P[i].py);

fprintf(fp,"\n %-12s","CPUTIME");

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

fprintf(fp,"%8d",P[i].ct);

fprintf(fp,"\n %-12s","ALLTIME");

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

fprintf(fp,"%8d",P[i].at);

fprintf(fp,"\n %-12s","STARTBLOCK");

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

fprintf(fp,"%8d",P[i].sb);

fprintf(fp,"\n %-12s","BLOCKTIME");

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

fprintf(fp,"%8d",P[i].bt);

fprintf(fp,"\n %-12s","STATE");

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

{

if(P[i].st==1)

fprintf(fp,"%8s","READY");

else if (P[i].st==0)

fprintf(fp,"%8s","FINISH");

else

fprintf(fp,"%8s","BLOCK");

}

fprintf(fp,"\n");

}while((ready\_q[0]!=0)||(block\_q[0]!=0)); /\* 循环结束条件：就绪队列和阻塞队列全为空\*/

}