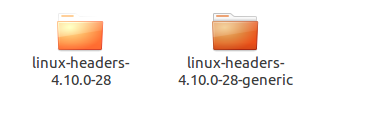
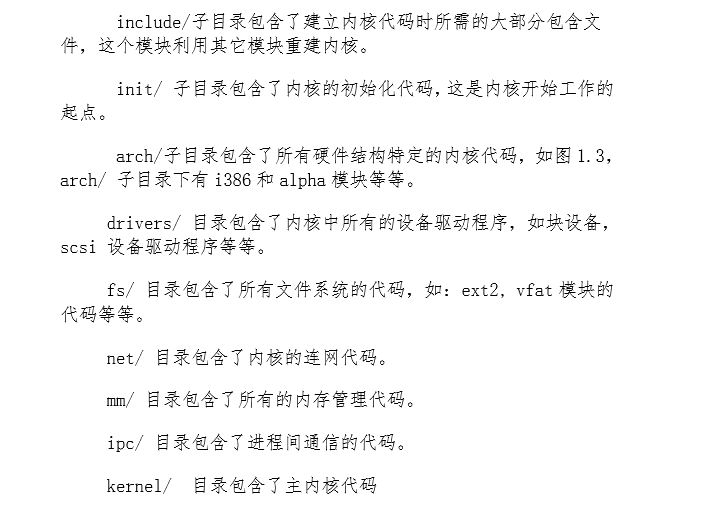
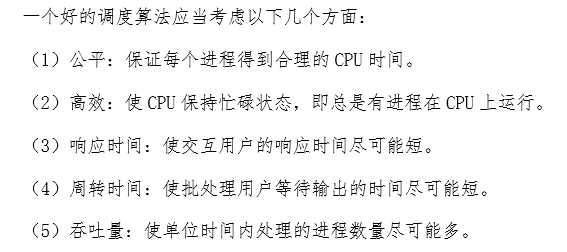
/usr/src/linux



Linux目录和子目录：



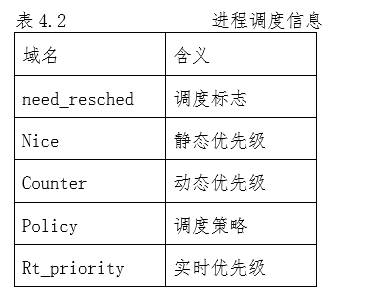


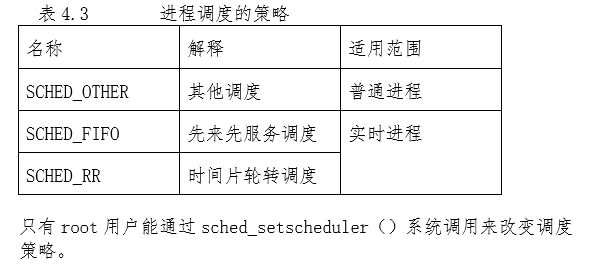
**Ⅰ．时间片轮转调度算法**

**Ⅱ．优先权调度算法**

**Ⅲ．多级反馈队列调度**

**Ⅳ**．**实时调度**





代码：

https://blog.csdn.net/KoalaZB/article/details/53563028

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<pthread.h>

#include<time.h>

#include<iostream>

#define Thread\_Num 20

using namespace std;

pthread\_mutex\_t Device\_mutex ;

//Virtual PCB of threads

struct VirtualPCB

{

int tid;

int priority;

int waittime;

int runtime;

int arrivetime;

int visited;

int tempruntime;

public:

int gettid()

{

return tid;

}

int getwaittime()

{

return waittime;

}

int getpriority()

{

return priority;

}

int getruntime()

{

return runtime;

}

int getarrivetime()

{

return arrivetime;

}

void setvisit(int a)

{

visited=a;

}

int getvisit()

{

return visited;

}

int gettempruntime()

{

return tempruntime;

}

void setwaittime(int n)

{

waittime = n;

}

void settempruntime(int n)

{

tempruntime = tempruntime - n;

}

}TCB[Thread\_Num];

//Function to initial virtual PCB

void t\_init()

{

int n;

srand(time(NULL));

for(n =0;n<Thread\_Num;n++)

{

TCB[n].tid = n + 1;//用线程创建序号作为虚拟进程id

//用随机数随机产生虚拟PCB的值

TCB[n].priority = 1 + rand()%19;

TCB[n].runtime = 1 + rand()%19;

TCB[n].arrivetime = 0;//模拟时，默认进程按创建顺序依次在0时刻到达

TCB[n].waittime = 0;

TCB[n].visited =0;

TCB[n].tempruntime = TCB[n].runtime;

}

}

//Threads run function

void \*t\_print(void \*arg)

{

int n = \*(int \*)arg;//get argument

while(1)

{

pthread\_mutex\_lock(&Device\_mutex);

printf("Thread\_%-2d: ",n);

printf("tid:%-2d priority:%-2d runtime:%-2d \n",TCB[n-1].gettid(),TCB[n-1].priority,TCB[n-1].runtime);

pthread\_mutex\_unlock(&Device\_mutex);

sleep(1);

break;

}

//printf("Error %d\n",n);

pthread\_exit(0);

}

//First come first service schedule function

void FCFS()

{

cout<<"-----------FCFS:"<<endl;

int i,j;

int start = 0;

float waittime = 0;

float avwait = 0;

for(i=0;i<Thread\_Num/2;i++)

{

for(j=0;j<Thread\_Num;j++){

if(TCB[j].getarrivetime()==i && TCB[j].getvisit()==0){

printf("Thread: %-2d Start: %-3d Runtime: %-2d\n",TCB[j].gettid(),start,TCB[j].getruntime());

waittime = waittime + (float)start;

start = start + TCB[j].getruntime();

TCB[j].setvisit(1);

}

}

}

avwait = waittime / (float)Thread\_Num;

printf("Total waitting time : %f\n",waittime);

printf("Average waitting time : %f\n",avwait);

}

//Shortest job first schedule function

void SJF()

{

for(int k=0 ;k<Thread\_Num;k++)

{

TCB[k].setvisit(0);

}

cout<<"-------------SJF:"<<endl;

int i,j;

int start = 0;

float waittime = 0;

float avwait = 0;

for(i=1;i<Thread\_Num;i++)

{

for(j=0;j<Thread\_Num;j++){

if(TCB[j].getruntime()==i && TCB[j].getvisit()==0){

printf("Thread: %-2d Start: %-3d Runtime: %-2d\n",TCB[j].gettid(),start,TCB[j].getruntime());

waittime = waittime + (float)start;

start = start + TCB[j].getruntime();

TCB[j].setvisit(1);

}

}

}

avwait = waittime / (float)Thread\_Num;

printf("Total waitting time : %f\n",waittime);

printf("Average waitting time : %f\n",avwait);

}

//Round R schedule function

void RR(int r)

{

cout<<"--------------RR:"<<endl;

int start = 0;

float waittime = 0;

float avwait = 0;

for(int i=0;i<Thread\_Num;i++)

{

int totaltime = totaltime + TCB[i].getruntime();

TCB[i].setvisit(0);

}

for(int j=0;j<20\*Thread\_Num;j=j+r)

{

int k = (j%(20\*r))/r;

if(TCB[k].gettempruntime() > 0){

int tepruntime = r;

if(TCB[k].gettempruntime()-r<=0){

tepruntime = TCB[k].gettempruntime();

TCB[k].setwaittime(start + tepruntime - TCB[k].getruntime());

}

printf("Thread: %-2d Start: %-3d Runtime:%-2d \n",TCB[k].gettid(), start,tepruntime);

start = start + tepruntime;

TCB[k].settempruntime(r) ;

}

}

for(int m=0;m<Thread\_Num;m++)

{

waittime += TCB[m].getwaittime();

//printf("TCB[%d].getwaittime():%d\n",m+1,TCB[m].getwaittime());

}

avwait = waittime / (float)Thread\_Num;

printf("Total waitting time : %f\n",waittime);

printf("Average waitting time : %f\n",avwait);

}

//Priority schedule function

void Priority()

{

for(int k=0 ;k<Thread\_Num;k++)

{

TCB[k].setvisit(0);

}

cout<<"-----------Priority:"<<endl;

int i,j;

int start = 0;

float waittime = 0;

float avwait = 0;

for(i=1;i<Thread\_Num;i++)

{

for(j=0;j<Thread\_Num;j++){

if(TCB[j].getpriority()==i && TCB[j].getvisit()==0){

printf("Thread: %-2d Start: %-3d Runtime: %-2d\n",TCB[j].gettid(),start,TCB[j].getruntime());

waittime = waittime + (float)start;

start = start + TCB[j].getruntime();

TCB[j].setvisit(1);

}

}

}

avwait = waittime / (float)Thread\_Num;

printf("Total waitting time : %f\n",waittime);

printf("Average waitting time : %f\n",avwait);

}

//Main thread execute function to create 20 children threads

void \*Children(void\*)

{

int ret[Thread\_Num];

t\_init();

pthread\_t tid[Thread\_Num];

pthread\_mutex\_init(&Device\_mutex,NULL);

int i,j;

for(i=0;i<Thread\_Num;i++)

{

int k =i+1;

ret[i] = pthread\_create(&tid[i],NULL,&t\_print, &k);

if(ret[i] == 0) {

sleep(1);

}

else{

printf("Thread\_%-2d failed!\n",i+1);

}

}

for(j=0;j<Thread\_Num;j++)

pthread\_join (tid[i], NULL);

pthread\_mutex\_destroy(&Device\_mutex);

pthread\_exit(0);

}

int main()

{

int ret1;

pthread\_t tid1;//Declare main thread

ret1 = pthread\_create(&tid1,NULL,&Children,NULL);//Create main thread

if(ret1 == 0)

{

printf("Main Thread ok!\n");

sleep(20);

}

else{

printf("Thread failed!\n");

}

FCFS();

SJF();

cout<<"Please enter RR time:\n";//Request RR time

int rr;

scanf("%d",&rr);

RR(rr);

Priority();

return 0;

}