Dẫn tới giải thuật này có nhiều dị bản khác nhau và sẽ tối ưu hay không tối ưu phụ thuộc vào trưng dụng CPU.

Shortest Remain Time(SRT)

Tương tự như SJF nhưng trong thuật toán này, độ ưu tiên thực hiện các tiến trình dựa vào thời gian cần thiết để thực hiện nốt tiến trình(bằng tổng thời gian trừ đi thời gian đã thực hiện). Như vậy, trong thuật toán này cần phải thường xuyên cập nhật thông tin về giời gian đã thực hiện của tiến trình. Đồng thời, chế độ phân bổ lại giờ CPU cũng phải được áp dụng nếu không sẽ làm mất tình ưu việc của thuật toán.

Ưu điểm :

Thời gian chờ đợi,tồn tại trong hệ thống của mỗi tiến trình đều ngắn

Thuật toán tối ưu nhất

Nhược điểm :

Việc cài đặt thuật toán khá phức tạp

Cần quản lý chặt chẽ việc điều phối các tiến trình

Quản lý thời gian đến của mỗi tiến trình

CÀI ĐẶT THUẬT TOÁN

Mô hình cài đặt thuật toán

Cấu trúc dữ liệu

Tiến trình

Cấu trúc dữ liệu đề xuất cho việc quả lý tiến trình được xây dựng thành một lớp nhằm tạo điều kiện cho việc quản lý các tiến trình được dễ dàng.

Code

class TIENTRINH

{

private:

int stt;

int t\_den;

int t\_xuly;

int t\_cho;

int finish;

public:

TIENTRINH();

TIENTRINH(int stt,int t\_den,int t\_xuly);

void insert(int stt,int t\_den,int t\_xuly);

int getT\_DEN();

void setT\_DEN(int a);

int getT\_XULY();

void setT\_XULY(int a);

int getT\_CHO();

void setT\_CHO(int a);

int getFINISH();

void setFINISH(int a);

int getSTT();

};

Stt : số thứ tự của tiến trình

t\_den : thời gian đến của tiến trình

t\_xuly : thời gian xữ lý của tiến trình

t\_cho : thời gian chờ của tiến trình

finish : thời gian hoàn thành của tiến trình

và các hàng thiết lập và lấy thông tin

Ready List

Ready list tổ chức theo danh sách liên kết chỉ chứa số thứ tự của các tiến trình.Và việc tổ chức các tiến trình vào ra trong ready list tuân theo các giải thuật được dùng trên danh sách liên kết.

Code

struct DS

{

int id;

DS \*next;

};

typedef DS\* list;

Id: chứa số thứ tự tiến trình trong Ready List

Input

Input được tổ chức theo danh sách liên kết đơn nhằm lưu giữ các giá trị khi nhập các tiến trình và là dữ liệu để phục hồi lại các tiến trình nhằm để tránh các trường hợp sai lệnh và mất dữ liệu khi xử lý.

Code

struct Input

{

int den,xuly;

Input \*next;

};

typedef Input\* IN;

den: thời gian đến của tiến trình khi nhập liệu

xử lý: thời gian xử lý của tiến trình khi nhập liệu

Thuật toán xử lý chung

Việc cài đặt thuật toán được mô phòng theo cách làm việc của CPU và tất các thuật toán con đều theo mô hình thuật toán này.

Tiến trình ở đầu danh sách sẽ được ưu tiên xữ lý trước và nó chiếm dụng CPU tại thời điểm đó.

Việc đi kèm thèo là xem xét thời gian xữ lý các tiến trình đã hết chưa. Nếu đã hết thì nghĩa là hoàn thành việc xữ lý, ngược lại thì tiếp tục xữ lý theo thuật toán.

Xong mỗi chu kỳ của CPU ( 1 quantum ) thì cập nhật lại danh sách để loại bỏ các tiến trình đã hoàn thành hay sắp xếp hay thêm các tiến trình mới vào

Hình 3.1.2-1. Sơ đồ thuật toán đề xuất chung cho các giải thuật

Thuật toán

First In First Out(FIFO)

Hình 3.2.1-1.Thuật toán FIFO

Code

void FIFO()

{

int time=0,ok=1,i,j=0,ID;

while(ok)

{

ID=-1;

PrintRL(ready,time);

listBox2->Items->Add("------------------");

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

{

// nap readylist luc bat dau

if(tt[j].getT\_DEN()==time && j<n )

{ them(j);

j++;

listBox2->Items->Add("Time = "+time.ToString()+" : Nap tien trinh : "+(tt[j-1].getSTT()).ToString());

}

// nen ton tai tt trong readylist thi lam,ko thi thoat quantum

if(ready)

{

ID=(\*ready).id;

listBox2->Items->Add("Time = "+time.ToString()+" : xu ly tien trinh : "+(tt[ID].getSTT()).ToString());

if(tt[ID].getT\_XULY()>0)

{

// tang thoi gian cho cua cac tt trong ready

tangT\_CHO(ready,ID);

tt[ID].setT\_XULY(tt[ID].getT\_XULY() - 1);

if(tt[ID].getT\_XULY()==0) xoa();

}

time++;

if(tt[ID].getT\_XULY()==0)

{ tt[ID].setFINISH(time);

listBox2->Items->Add("Time = "+time.ToString()+" : hoan thanh tien trinh : "+(tt[ID].getSTT()).ToString());

break;

}

}

else

{

tangT\_CHO(ready,-1);

time++;

break;

}

}

listBox2->Items->Add("-------Hoan thanh chu ky-------");

listBox2->Items->Add("------------------");

if(checkFinish()) ok=0;

}

TIME=time

}

Round Robin(RR)

Hình 3.2.2-1.Round Robin

Code

void RR()

{

int time=0,ok=1,i,j=0,ID,check;

while(ok)

{

ID=-1;

check=0;

PrintRL(ready,time);

listBox2->Items->Add("------------------");

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

{// nap readylist luc bat dau

if(tt[j].getT\_DEN()==time && j<n )

{ them(j);

j++;

listBox2->Items->Add("Time = "+time.ToString()+" : Nap tien trinh : "+(tt[j-1].getSTT()).ToString());

}

// nen ton tai tt trong readylist thi lam,ko thi thoat quantum

if(ready)

{

ID=(\*ready).id;

listBox2->Items->Add("Time = "+time.ToString()+" : xu ly tien trinh : "+(tt[ID].getSTT()).ToString());

if(tt[ID].getT\_XULY()>0)

{

// tang thoi gian cho cua cac tt trong ready

tangT\_CHO(ready,ID);

tt[ID].setT\_XULY(tt[ID].getT\_XULY() - 1);

if(tt[ID].getT\_XULY()==0) xoa();

}

time++;

if(tt[ID].getT\_XULY()==0)

{ tt[ID].setFINISH(time);

listBox2->Items->Add("Time = "+time.ToString()+" : hoan thanh tien trinh : "+(tt[ID].getSTT()).ToString());

check=1; // \*\*\*\*

break; // \*\*\*\*

}

}

else

{

tangT\_CHO(ready,-1);

time++;

break;

}

}

if(!check) hoanvi();

listBox2->Items->Add("-------Hoan thanh chu ky-------");

listBox2->Items->Add("------------------");

if(checkFinish()) ok=0;

}

TIME=time;

}

Shortest Job First(SRT)

Hình 3.2.3-1. Shortest Job First

Code

void SJF()

{

int time=0,ok=1,i,j=0,ID;

while(ok)

{

ID=-1;

PrintRL(ready,time);

listBox2->Items->Add("------------------");

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

{

// nap readylist luc bat dau

if(tt[j].getT\_DEN()==time && j<n )

{ them(j);

j++;

listBox2->Items->Add("Time = "+time.ToString()+" : Nap tien trinh : "+(tt[j-1].getSTT()).ToString());

}

// nen ton tai tt trong readylist thi lam,ko thi thoat quantum

if(ready)

{

ID=(\*ready).id;

listBox2->Items->Add("Time = "+time.ToString()+" : xu ly tien trinh : "+(tt[ID].getSTT()).ToString());

if(tt[ID].getT\_XULY()>0)

{

// tang thoi gian cho cua cac tt trong ready

tangT\_CHO(ready,ID);

tt[ID].setT\_XULY(tt[ID].getT\_XULY() - 1);

if(tt[ID].getT\_XULY()==0) xoa();

}

time++;

if(tt[ID].getT\_XULY()==0)

{ tt[ID].setFINISH(time);

listBox2->Items->Add("Time = "+time.ToString()+" : hoan thanh tien trinh : "+(tt[ID].getSTT()).ToString());

break;

}

}

else

{

tangT\_CHO(ready,-1);

time++;

break;

}

}

sapxep();

listBox2->Items->Add("-------Hoan thanh chu ky-------");

listBox2->Items->Add("------------------");

if(checkFinish()) ok=0;

}

TIME=time;

}

Shortest Remain Time(SRT)

Hình 3.2.4-1.Shortest Remain Time

Code

void SRT()

{

int time=0,ok=1,i,j=0,ID;

while(ok)

{

ID=-1;

PrintRL(ready,time);

listBox2->Items->Add("------------------");

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

{

// nap readylist luc bat dau

if(tt[j].getT\_DEN()==time && j<n )

{ them(j);

j++;

listBox2->Items->Add("Time = "+time.ToString()+" : Nap tien trinh : "+(tt[j-1].getSTT()).ToString());

}

// nen ton tai tt trong readylist thi lam,ko thi thoat quantum

if(ready)

{

ID=(\*ready).id;

sapxep();

if( ID != (\*ready).id ) break; //\*\*\*\*

listBox2->Items->Add("Time = "+time.ToString()+" : xu ly tien trinh : "+(tt[ID].getSTT()).ToString());

if(tt[ID].getT\_XULY()>0)

{

// tang thoi gian cho cua cac tt trong ready

tangT\_CHO(ready,ID);

tt[ID].setT\_XULY(tt[ID].getT\_XULY() - 1);

if(tt[ID].getT\_XULY()==0) xoa();

}

time++;

if(tt[ID].getT\_XULY()==0)

{ tt[ID].setFINISH(time);

listBox2->Items->Add("Time = "+time.ToString()+" : hoan thanh tien trinh : "+(tt[ID].getSTT()).ToString());

break;

}

}

else

{

tangT\_CHO(ready,-1);

time++;

break;

}

}

sapxep();

listBox2->Items->Add("-------Hoan thanh chu ky-------");

listBox2->Items->Add("------------------");

if(checkFinish()) ok=0;

}

TIME=time;

}

XÂY DỰNG CHƯƠNG TRÌNH DEMO

Các modun chính

Input: dùng để nhập dữ liệu

Nhập từ file

Nhập từ bàn phím

Ouput: hiển thị các tiến trình đã nhập

Control: lựa chọn các giải thuật

First In First Out

Round Robin

Shortest Job First

Shortest Remain Time

Result

About: thông ti về chương trình

Info

Help

Exit: thoát khỏi chương trình

Môi trường phát triển

Sử dụng ngôn ngữ C++ và giao diện windows để cho thao tác nhập dữ liệu, mô phỏng được dễ dàng.

Giao diện của chương trình

About

Info

Hình 4.3.1-1.Hiển thị thông tin về đồ án môn học

Help

Hình 4.3.1-2.Hiển thị trợ giúp cho việc thao tác trên chương trình

Input

File

Hình 4.3.2-1.Nhập dữ liệu từ file

Bàn phím

Hình 4.3.2-2.Nhập dữ liệu từ bàn phím

Output

Hình 4.3.3-1.Hiển thị thông tin các tiến trình đã nhập

Control

FIFO

Hình 4.3.4-1.Điều khiển giải thuật FIFO và tùy chỉnh quantum tùy ý

RR

Hình 4.3.4-2.Round Robin

SJF

Hình 4.3.4-3.Shortest Job First

SRT

Hình 4.3.4-4.Shortest Remain Time

Result

Hình 4.3.4-5.Hiển thị bản so sánh giữa các tiến trình và lựa chọn giải thuật tối ưu nhất

Đánh giá và nhận xét

Giao diện chương trình bắt mắt

Các modun được bố trí hợp lí và gọn gàn

KẾT LUẬN

Xây dựng thành công chương trình mô phỏng các giải thuật định thời CPU

Qua đó nắm bắt rõ các giải thuật lập lịch

TÀI LIỆU

Nguyễn Kim Tuấn.Giáo Trình Lý Thuyết Hệ Điều Hành.Đại học Huế,trường đại học khoa học,khoa công nghê thông tin, Huế 06/2004, 217t

Đặng Vũ Tùng. Giáo Trình Lý Thuyết Hệ Điều Hành.Sở giáo dục và đào tạo Hà Nội, nhà xuất bản Hà Nội 2005, 165t.

Trần Hồ Thủy Tiên.Giáo Tìình Nguyên Lý Hệ Điều Hành.Đại học Đà Nẵng, Trường đại học Bách Khoa, Khoa Công Nghệ Thông Tin 01/04/2010.