李富荣

软件工程 | 2012118068

操作系统实训项目报告

项目一：单处理机进程调度

1. **基本信息**

1，项目信息：单处理机进程调度

2，完成人信息：李富荣 2012118068

3，完成日期：2015.5.24

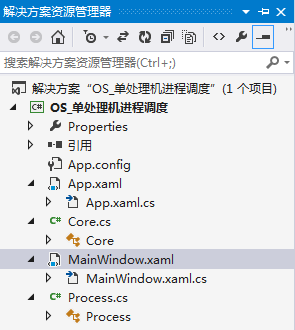
1. **实验内容与目的**

1，加深进程概念理解，明确进程与程序区别。

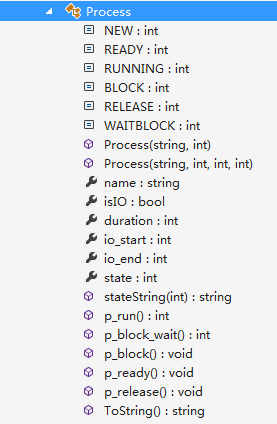
2，理解操作系统中 进程的组织、创建和调度等方法。

1. **主要设计思路和流程图**

使用Microsoft Visual Studio 2013为开发工具，在windows环境下开发，类视图如下所示：



Process进程实体类



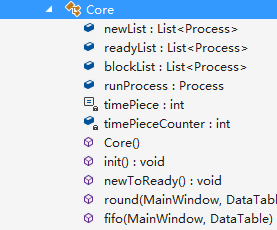
进程有五个属性成员name进程名，isIO是否IO，duration执行时间，io\_start IO开始时间，io\_end IO结束时间。

两个构造方法，从界面读取属性成员取值，如果有IO调用4个参数的构造方法，如果没有IO调用2个参数构造方法。

方法P\_打头的方法通过改变state属性，实现状态的改变

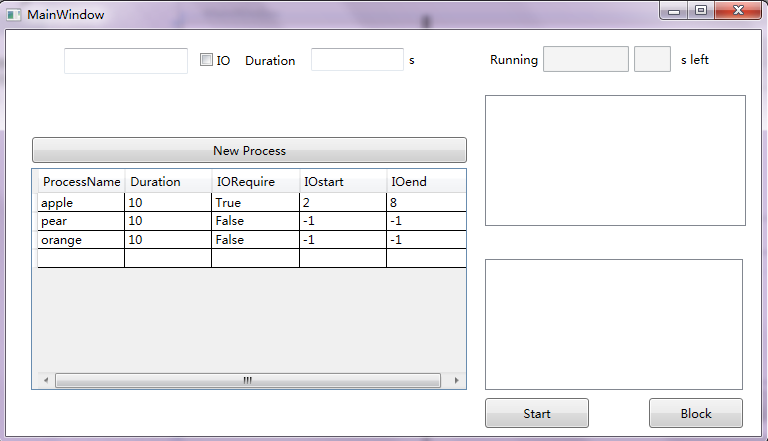
ToString方法返回进程信息，stateString方法辅助toString方法返回状态进程状态信息

Core 进程管理类



Core类作为进程的管理类，拥有3个队列，应该是5个状态，但处理机每次只让一个进程运行，则设定一个runProcess属性存放当前处于运行态的进程，在这里没有设定退出队列，因为没有实际意义，另外设定一个时间片常量属性，描述时间片大小，timeCounter变量给那些等待IO完成的进程计时，使其在事件完成后能够回到就绪队列。Core的构造方法进行初始化，init方法在一开始创建3个进程，newToReady做一个从ready到New的计时事件，round和fifo分别模拟时间片调度和先来先服务的调度算法。

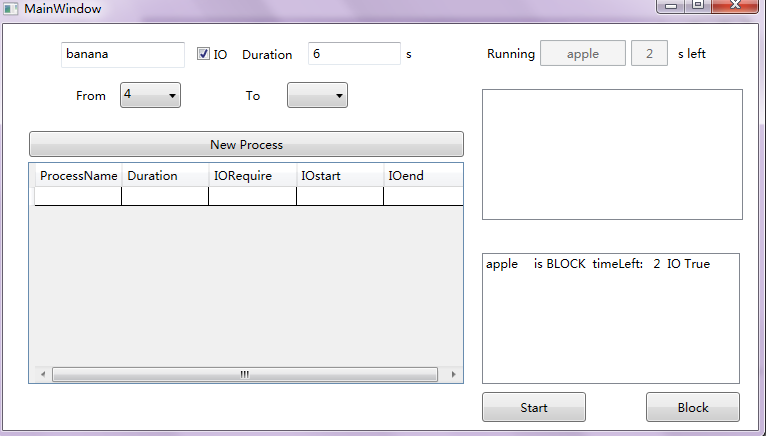
主界面视图



应用C#WPF中的定时器DispatcherTimer每隔1秒刷新界面，展示运行信息。左侧的DataGrid存放newList中的信息。右侧上面的DataGrid存放ReadyList中的信息，下面是BlockList中的情况显示

NewProcess按键事件：读取进程名，是否IO，以及IO开始结束时间，创建进程

IO checkbox状态变换时间，选择IO请求，出现两个selectBox



Block按键事件：中断正在运行的进程

Start按键点击事件：恢复block队列中最早进入的进程回到就绪队列

DisPatcher Timer 计时事件，刷新界面显示状态信息，调用Core中的进程调度算法

1. **主要数据结构及其说明**

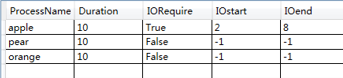
**List<Process> ReadyList; Process类型就绪队列**

**List<Process> BlockList; Process类型终止队列**

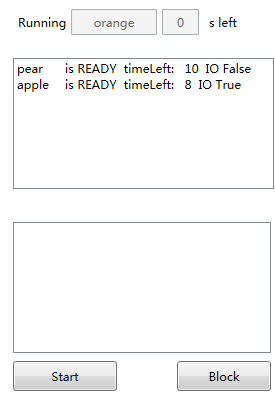
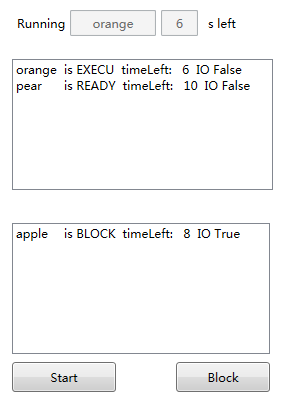
**List<Process> NewList; Process类型新建队列**

**Process runProcess Process类型存放正在运行的进程信息**

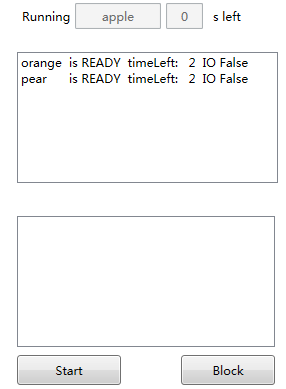
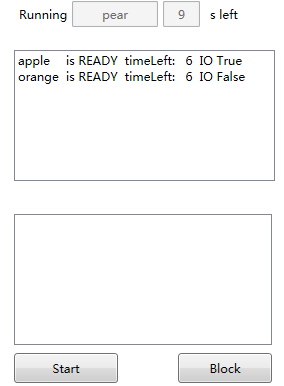
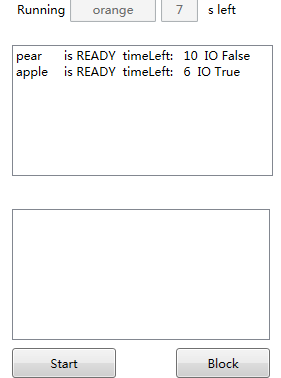
1. **程序运行时的初值和运行结果**



程序启动，配置调度策略为fifo，计时器过1秒，自动将NewList中进程挂到ReadyLIst中，运行程序能够明显看出这暂停的一秒的时间。因为apple有IO请求，因此它在运行2秒后会自动挂起，在pear运行到剩下4秒时，在进入ready队列，根据进入readyList 的顺序，各个进程逐个运行完毕

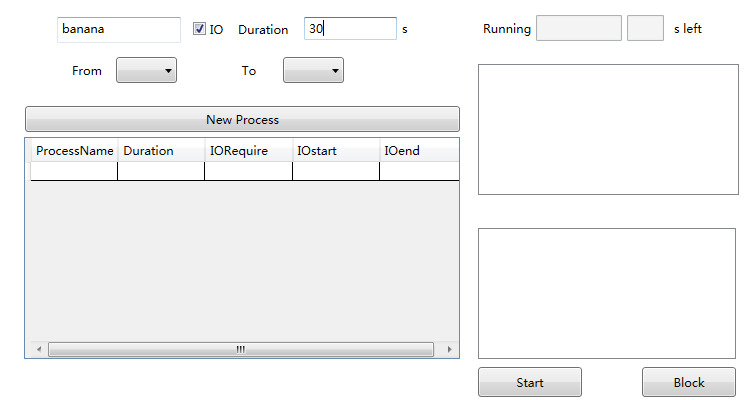


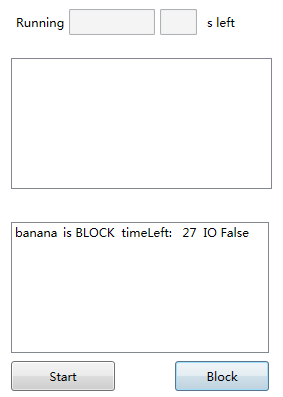
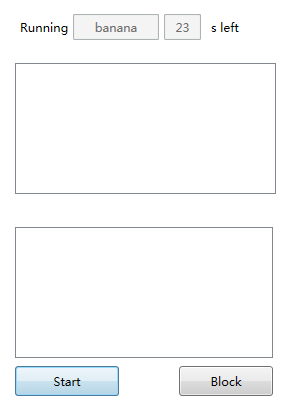
配置调度策略为时间片，时间片为3，则apple会自动挂起，orange运行3秒也自动挂起，直至所有进程运行完毕



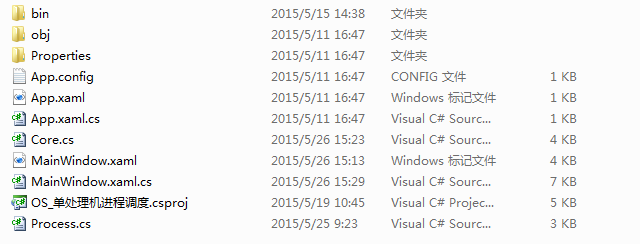
当然你可以将正在运行的任何进程通过点击break按键阻塞，通过点击start按钮恢复到就绪队列

***新建一个banana进程运行30秒，点击block终止，点击start恢复***



1. **源程序&注释**



**Process.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OS\_单处理机进程调度

{

public class Process

{

public const int NEW = 0;

public const int READY = 1;

public const int RUNNING = 2;

public const int BLOCK = 3;

public const int RELEASE = 4;

public const int WAITBLOCK = 5;

//无IO的时候引用

public Process(string \_name,int \_duration)

{

name = \_name;

duration = \_duration;

isIO = false;

io\_start = -1;

io\_end = -1;

state = NEW;

}

//IO的时候引用

public Process(string \_name,int \_duration,int \_io\_start,int \_io\_end)

{

name = \_name;

duration = \_duration;

io\_start = \_io\_start;

io\_end = \_io\_end;

isIO = true;

state = NEW;

}

public string name { get; set; }

public bool isIO { get; set; }

public int duration { get; set;}

public int io\_start { get; set; }

public int io\_end { get; set; }

public int state { get; set; }

public string stateString(int \_state)

{

string stateString;

switch (\_state)

{

case 0: stateString = "NEW"; break;

case 1: stateString = "READY"; break;

case 2: stateString = "EXECU"; break;

case 3: stateString = "BLOCK"; break;

case 4: stateString = "RELEASE"; break;

default: stateString = "NOT EXIST\t"; break;

}

return stateString;

}

public int p\_run()

{

state = RUNNING;

if (duration > 0)

{

duration = duration - 1;

}

else

{

duration = 0;

}

if (duration == io\_end)

{

return WAITBLOCK;

}

else

{

return RUNNING;

}

}

public int p\_block\_wait()

{

if (io\_start < io\_end)

{

io\_start++;

}

if (io\_start == io\_end)

{

return 1;

}

else

{

return 0;

}

}

public void p\_block()

{

state = BLOCK;

}

public void p\_ready()

{

state = READY;

}

public void p\_release()

{

state = RELEASE;

}

public override string ToString()

{

return name+"\tis "+stateString(state)+" timeLeft: "+duration+" IO "+isIO;

}

}

}

**Core.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Timers;

using System.Data;

namespace OS\_单处理机进程调度

{

public class Core

{

public List<Process> newList;

public List<Process> readyList;

public List<Process> blockList;

public Process runProcess=null;

const int timePiece = 3;

int timePieceCounter = 0;

public Core()

{

newList = new List<Process>();

readyList = new List<Process>();

blockList = new List<Process>();

init();

}

public void init()

{

newList.Add(new Process("apple", 10,2,8));

newList.Add(new Process("pear", 10));

newList.Add(new Process("orange", 10));

}

public void newToReady()

{

for (int i = 0; i < newList.Count; i++)

{

if (newList[i].state == 0)

{

newList[i].p\_ready();

readyList.Add(newList[i]);

newList.RemoveAt(i);

}

}

}

public void round(MainWindow main, DataTable dt)

{

if (readyList.Count > 0)

{

runProcess = readyList[0];

readyList.Remove(runProcess);

if (runProcess.duration > 0)

{

runProcess.p\_run();

main.tb\_run\_process.Text = runProcess.name;

main.tb\_run\_process\_duration.Text = runProcess.duration.ToString();

main.showData(dt);

timePieceCounter++;

if (runProcess.state == 2&&timePieceCounter<=timePiece)

{

readyList.Insert(0, runProcess);

}

else

{

timePieceCounter = 0;

runProcess.p\_ready();

readyList.Add(runProcess);

}

}

}

else

{

main.showData(dt);

main.tb\_run\_process.Text = "";

main.tb\_run\_process\_duration.Text = "";

}

}

public void fifo(MainWindow main,DataTable dt)

{

if (readyList.Count > 0)

{

runProcess = readyList[0];

readyList.Remove(runProcess);

if (runProcess.duration > 0)

{

int flag = runProcess.p\_run();

main.tb\_run\_process.Text = runProcess.name;

main.tb\_run\_process\_duration.Text = runProcess.duration.ToString();

if (flag == 5)

{

runProcess.p\_block();

blockList.Add(runProcess);

}

else

{

if (runProcess.state == 2)

{

readyList.Insert(0, runProcess);

}

}

}

//如果组赛队列不为空，等待io，完毕后挂入就绪队列

if (blockList.Count > 0)

{

Process temp2 = blockList[0];

int tempN = temp2.io\_end;

temp2.io\_end--;

if (temp2.io\_start == temp2.io\_end)

{

temp2.p\_ready();

//temp2.duration = temp2.io\_start;

readyList.Add(temp2);

blockList.Remove(temp2);

}

}

}

main.showData(dt);

}

}

}

**MainWindow.xaml.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Windows.Threading;

using System.Timers;

namespace OS\_单处理机进程调度

{

/// <summary>

/// MainWindow.xaml 的交互逻辑

/// </summary>

public partial class MainWindow : Window

{

DataTable dt;

Core core;

private DispatcherTimer timer;

public MainWindow()

{

InitializeComponent();

Loaded += new RoutedEventHandler(MainWindow\_loaded);

}

//界面方法组

public void init()

{

core = new Core();

//建立表

dt = new DataTable();

DataColumn dc = new DataColumn("Process\_Name", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("Duration", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("IO\_Require", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("IO\_start", typeof(string));

dt.Columns.Add(dc);

dc = new DataColumn("IO\_end", typeof(string));

dt.Columns.Add(dc);

//设置datdagrid样式，显示刚才的数据表

dgShow.CanUserResizeColumns = false;

dgShow.ColumnWidth = ((dgShow.Width) / 5);

showData(dt);

}

public void showData(DataTable \_dt)

{

\_dt.Clear();

listbox\_ready.Items.Clear();

listbox\_block.Items.Clear();

for (int i = 0; i < core.newList.Count; i++)

{

dt.Rows.Add(core.newList[i].name, core.newList[i].duration, core.newList[i].isIO, core.newList[i].io\_start, core.newList[i].io\_end);

}

dgShow.ItemsSource = \_dt.DefaultView;

for (int i = 0; i < core.readyList.Count; i++)

{

listbox\_ready.Items.Add(core.readyList[i]);

}

for (int i = 0; i < core.blockList.Count; i++)

{

listbox\_block.Items.Add(core.blockList[i]);

}

}

private void bt\_new\_Click(object sender, RoutedEventArgs e)

{

if (tb\_p\_name.Text != "" && tb\_duration.Text != "")

{

string tempName = tb\_p\_name.Text.ToString();

int tempDuration = Convert.ToInt32(tb\_duration.Text.ToString());

bool tempIsIO = (bool)checkbox\_IO.IsChecked;

Process temp;

if (tempIsIO && cb\_from1.SelectedItem != null && cb\_to.SelectedItem != null)

{

int tempIoStart = Convert.ToInt32(cb\_from1.SelectedItem.ToString());

int tempIoEnd = Convert.ToInt32(cb\_to.SelectedItem.ToString());

temp = new Process(tempName, tempDuration, tempIoStart, tempIoEnd);

}

else

{

temp = new Process(tempName, tempDuration);

}

core.newList.Add(temp);

showData(dt);

}

}

private void checkbox\_IO\_Click(object sender, RoutedEventArgs e)

{

if (checkbox\_IO.IsChecked == true)

{

cb\_from1.Visibility = Visibility.Visible;

cb\_to.Visibility = Visibility.Visible;

label\_from.Visibility = Visibility.Visible;

label\_to.Visibility = Visibility.Visible;

}

else if (checkbox\_IO.IsChecked == false)

{

cb\_from1.Visibility = Visibility.Hidden;

cb\_to.Visibility = Visibility.Hidden;

label\_from.Visibility = Visibility.Hidden;

label\_to.Visibility = Visibility.Hidden;

}

}

private void tb\_duration\_TextChanged(object sender, TextChangedEventArgs e)

{

int temp = Convert.ToInt32(tb\_duration.Text.ToString());

cb\_from1.Items.Clear();

cb\_to.Items.Clear();

if (temp > 0)

{

for (int i = 0; i < temp; i++)

{

cb\_from1.Items.Add(i);

}

}

if (!cb\_from1.Items.IsEmpty)

{

for (int i = 0; i < temp; i++)

{

cb\_to.Items.Add(i);

}

}

}

private void cb\_from1\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (cb\_from1.SelectedItem != null)

{

int length = Convert.ToInt32(tb\_duration.Text.ToString());

int temp = Convert.ToInt32(cb\_from1.SelectedItem.ToString());

cb\_to.Items.Clear();

for (int i = temp; i < length - 1; i++)

{

cb\_to.Items.Add(i + 1);

}

}

}

private void bt\_break\_Click(object sender, RoutedEventArgs e)

{

core.readyList.Remove(core.runProcess);

core.blockList.Add(core.runProcess);

core.runProcess.p\_block();

showData(dt);

}

private void bt\_release\_Click(object sender, RoutedEventArgs e)

{

if (core.blockList != null)

{

Process temp = core.blockList[0];

core.blockList.RemoveAt(0);

core.readyList.Add(temp);

temp.p\_ready();

showData(dt);

}

}

//时间方法组

void MainWindow\_loaded(Object sender,RoutedEventArgs e)

{

init();

timer = new DispatcherTimer();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick+=timer2\_Tick;

timer.Tick += timer1\_Tick;

timer.Start();

}

public void timer1\_Tick(Object sender, EventArgs e)

{

core.round(this,dt);

}

public void timer2\_Tick(Object sender, EventArgs e)

{

core.newToReady();

}

}

}