**Исходный код**

class FCFS

{

private List<Process> procs = new List<Process>();

private List<Int32> queue = new List<Int32>();

Thread serve;

bool isServing = false, isEnd = false;

Int32 totalIdlingProcs = 0;

Stopwatch[] idleSW;

Process current;

public FCFS(Int32 totalProcs)

{

idleSW = new Stopwatch[totalProcs];

for (Int32 i = 0; i < totalProcs; i++)

idleSW[i] = new Stopwatch();

}

public void AddProcess(Process proc)

{

Console.WriteLine("FCFS: add proc with id " + proc.PID.ToString());

procs.Add(proc);

if (!isServing)

{

isServing = true;

serve = new Thread(this.ServeProcess);

serve.Start();

}

else

{

queue.Add(procs.Count);

totalIdlingProcs++;

idleSW[proc.PID].Start();

}

}

public void CheckQueue()

{

if (procs.Count > 0)

{

serve = new Thread(this.ServeProcess);

serve.Start();

}

else

{

isServing = false;

if (isEnd)

Console.WriteLine("FCFS: emulation end.");

}

}

void ServeProcess()

{

idleSW[procs[0].PID].Stop();

current = procs[0];

procs.RemoveAt(0);

//Console.WriteLine("FCFS: remove proc with id " + current.PID.ToString());

Console.WriteLine("FCFS: start serving proc with id " + current.PID.ToString());

Thread.Sleep(current.executionTime \* 100);

Console.WriteLine("FCFS: served proc with id " + current.PID.ToString());

CheckQueue();

//RemoveProcess();

}

public void EndEmulation()

{

isEnd = true;

}

public Int32 maxQueue

{

get {

Int32 res = 0;

foreach(Int32 i in queue)

if(i > res)

res = i;

return res;

}

}

public Double avgIdleTime

{

get {

Int64 res = 0;

for (Int32 i = 0; i < idleSW.Length; i++)

res += idleSW[i].ElapsedMilliseconds;

return 1.0 \* res / totalIdlingProcs / 100;

}

}

}

class SPN

{

private List<Process> procs = new List<Process>();

private List<Int32> queue = new List<Int32>();

Thread serve;

bool isServing = false, isEnd = false;

Int32 totalIdlingProcs = 0;

Stopwatch[] idleSW;

Process current;

public SPN(Int32 totalProcs)

{

idleSW = new Stopwatch[totalProcs];

for (Int32 i = 0; i < totalProcs; i++)

idleSW[i] = new Stopwatch();

}

public void AddProcess(Process proc)

{

Console.WriteLine("SPN: add proc with id " + proc.PID.ToString());

procs.Add(proc);

if (!isServing)

{

isServing = true;

serve = new Thread(this.ServeProcess);

serve.Start();

}

else

{

queue.Add(procs.Count);

totalIdlingProcs++;

idleSW[proc.PID].Start();

try

{

procs.Sort(delegate(Process p1, Process p2) { return p2.executionTime.CompareTo(p1.executionTime); });

}

catch (Exception e)

{

e.ToString();

}

}

}

public void CheckQueue()

{

if (procs.Count > 0)

{

serve = new Thread(this.ServeProcess);

serve.Start();

}

else

{

isServing = false;

if (isEnd)

Console.WriteLine("SPN: emulation end.");

}

}

void ServeProcess()

{

idleSW[procs[0].PID].Stop();

current = procs[0];

procs.RemoveAt(0);

//Console.WriteLine("SPN: remove proc with id " + current.PID.ToString());

Console.WriteLine("SPN: start serving proc with id " + current.PID.ToString());

Thread.Sleep(current.executionTime \* 100); // emulate serving

Console.WriteLine("SPN: served proc with id " + current.PID.ToString());

CheckQueue();

}

public void EndEmulation()

{

isEnd = true;

}

public Int32 maxQueue

{

get

{

Int32 res = 0;

foreach (Int32 i in queue)

if (i > res)

res = i;

return res;

}

}

public Double avgIdleTime

{

get

{

Int64 res = 0;

for (Int32 i = 0; i < idleSW.Length; i++)

res += idleSW[i].ElapsedMilliseconds;

return 1.0 \* res / totalIdlingProcs / 100;

}

}

}

class Process

{

private Int32 execTime, procID;

public Process(Int32 execTime, Int32 procID)

{

executionTime = execTime;

PID = procID;

}

public Int32 executionTime

{

get { return execTime; }

private set {

if (value >= 3 && value <= 10)

execTime = value;

}

}

public Int32 PID

{

get { return procID; }

private set { procID = value; }

}

}

class Program

{

static List<Int32> times = new List<Int32>();

static List<Int32> execTime = new List<Int32>();

static List<Process> processes = new List<Process>();

static void Main(string[] args)

{

Console.Write("Enter number of processes/delays: ");

Int32 res;

while (!int.TryParse(Console.ReadLine(), out res) || res <= 1 || res > 1000)

Console.Write("Enter number of processes/delays: ");

Random rnd = new Random();

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

{

if (i == 0)

times.Add(0);

else

times.Add(times[i-1] + rnd.Next(6, 12));

execTime.Add(rnd.Next(3, 10));

}

//times.AddRange(new int[] { 0, 6, 12, 18, 24 });//(new int[] { 0, 6, 12, 18 });

//execTime.AddRange(new int[] { 8, 9, 8, 13, 4 }); //(new int[] { 10, 10, 10, 10 });

Int32 totalExecTime = 0;

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

{

totalExecTime += execTime[i];

Console.Write(times[i] + "\t");

processes.Add(new Process(execTime[i], i));

}

Console.WriteLine();

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

Console.Write(execTime[i] + "\t");

Console.WriteLine();

Console.WriteLine("Start FCFS emulating.");

FCFS fcfs = new FCFS(execTime.Count);

Console.WriteLine("Adding processes");

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

{

//Console.WriteLine("adding proc " + processes[i].PID.ToString()+ " with execTime " + processes[i].executionTime.ToString());

fcfs.AddProcess(processes[i]);

if (i == times.Count - 1)

{

Console.WriteLine("There is no next process.\r\nWaiting for current processes emulation ending.");

fcfs.EndEmulation();

}

else

{

//Console.WriteLine("next proc in " + (times[i+1] - times[i]).ToString() +"\r\n");

Thread.Sleep((times[i + 1] - times[i]) \* 100);

}

}

Console.ReadKey(true);

Console.WriteLine("\r\nmax queue " + fcfs.maxQueue.ToString());

Console.WriteLine("avg exec time {0:F2}", 1.0 \* totalExecTime / execTime.Count);

Console.WriteLine("avg idle time {0:F2}", fcfs.avgIdleTime);

Console.WriteLine("\r\nTo start emulating SPN press any key.");

Console.ReadKey(true);

Console.WriteLine("Start SPN emulating, please wait...");

SPN spn = new SPN(execTime.Count);

Console.WriteLine("Adding processes...");

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

{

//Console.WriteLine("adding proc " + processes[i].PID.ToString()+ " with execTime " + processes[i].executionTime.ToString());

spn.AddProcess(processes[i]);

if (i == times.Count - 1)

{

Console.WriteLine("There is no next process.\r\nWaiting for current processes emulation ending.");

spn.EndEmulation();

}

else

{

//Console.WriteLine("next proc in " + (times[i+1] - times[i]).ToString() +"\r\n");

Thread.Sleep((times[i + 1] - times[i]) \* 100); // sleep to next proc

}

}

Console.ReadKey(true);

Console.WriteLine("\r\nmax queue " + spn.maxQueue.ToString());

Console.WriteLine("avg exec time {0:F2}", 1.0 \* totalExecTime / execTime.Count);

Console.WriteLine("avg idle time {0:F2}", spn.avgIdleTime);

Console.ReadKey(true);

}

}