Министерство Образования Республики Беларусь

Учреждение образования

«Гомельский государственный технический университет

имени П.О.Сухого»

Кафедра «Информатика»

**Лабораторная работа № 6**

по дисциплине: **«Операционные Системы и Среды»**

**Планирование процессов**

Выполнил студент

группы ИП-32

*Кулинкин К.А.*

Проверил старший преподаватель

*Самовендюк Н.В.*

Гомель 2019 г.

**Цель работы:** изучить типовые алгоритмы планирования процессов

**Задание 1. Не вытесняющие алгоритмы планирования процессов**

Выполнить различные алгоритмы планирований – First-Come, First-Served (FCFS) (прямой и обратный), Round Robin (RR), Shortest-Job-First (SJF) (не вытесняющий), Shortest-Job-First (SJF) (не вытесняющий приоритетный) для данных приведенных в таблице 2.1 в соответствии со своим вариантом (номер по журналу). Вычислить полное время выполнения все процессов и каждого в отдельности, время ожидание для каждого процесса. Рассчитать среднее время выполнения процесса и среднее время ожидания. Результаты оформить в виде таблиц иллюстрирующих работу процессов.

**Задание 2 Вытесняющие алгоритмы планирования процессов**

Выполнить различные алгоритмы планирований – Shortest-Job-First (SJF) (вытесняющий) и Shortest-Job-First (SJF) (приоритетный) для данных приведенных в таблице 2.1 в соответствии со своим вариантом. Вычислить полное время выполнения все процессов и каждого в отдельности, время ожидание для каждого процесса. Рассчитать среднее время выполнения процесса и среднее время ожидания. Результаты оформить в виде таблиц иллюстрирующих работу процессов.

Таблица 2.1 Варианты заданий

C:\Users\KotlaWan\Desktop\2.png

**Примечания:**

Для алгоритма Round Robin (RR) величина кванта времени 3 для всех вариантов.

Для приоритетных алгоритмов меньшее значение соответствует более высокому приоритету.

**Задание 3.**

Разработать программную реализацию алгоритмов задания 1 и 2.

Сравнить полученные результаты и сделать выводы.

**Практическая часть:**

**Задание 1**

**FCFS прямой**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| P0 |  |  |  | Г | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | И | И | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  | И |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |  | И | И | И | И | И | И |  |  |  |  |  |  |

Ожидание: (1+0+0+0)/4=0,25

Выполнение: (3+4+1+6)/4=3,50

**FCFS обратный**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| P0 |  |  |  |  |  |  |  | Г | Г | Г | Г | Г | Г | И | И |  |  |  |  |
| P1 |  |  |  |  |  |  | Г | Г | Г | И | И | И | И |  |  |  |  |  |  |
| P2 | И |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  | И | И | И | И | И | И |  |  |  |  |  |  |  |  |  |  |

Ожидание: (6+3+0+0)/4=2,25

Выполнение: (8+7+1+6)/4=5,50

**RR**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| P0 |  |  |  | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | И | И | И | Г | Г | И |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  | И |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |  | И | И | И | И | И | И |  |  |  |  |  |  |

Ожидание: (0+2+0+0)/4=0,50

Выполнение: (2+5+1+6)/4=3,50

**SJF (не вытесняющий)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| P0 |  |  |  | Г | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | И | И | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  | И |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |  | И | И | И | И | И | И |  |  |  |  |  |  |

Ожидание: (1+0+0+0)/4=0,25

Выполнение: (3+4+1+6)/4=3,50

**SJF(не вытесняющий приоритетный)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| P0 |  |  |  | Г | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | И | И | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  | И |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |  | И | И | И | И | И | И |  |  |  |  |  |  |

Ожидание: (1+0+0+0)/4=0,25

Выполнение: (3+4+1+6)/4=3,50

**Задание 2**

**(SJF) (вытесняющий)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| P0 |  |  |  | Г | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | И | И | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  | И |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |  | И | И | И | И | И | И |  |  |  |  |  |  |

Ожидание: (1+0+0+0)/4=0,25

Выполнение: (3+4+1+6)/4=3,50

**(SJF) (вытесняющий приоритетный)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| P0 |  |  |  | И | И |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | И | И | И | Г | Г | И |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  | И |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |  | И | И | И | И | И | И |  |  |  |  |  |  |

Ожидание: (0+2+0+0)/4=0,50

Выполнение: (2+5+1+6)/4=3,50

**Задание 3**

**Листинг программы:**

Файл Program.cs:

|  |
| --- |
| using System;  namespace lab6  {  class Program  {  static void Main(string[] args)  {  Process[] processes = new Process[]  {  new Process(2, 3, 3), //p0  new Process(4, 0, 4), //p1  new Process(1, 6, 1), //p2  new Process(6, 7, 3) //p3  };    IProcessPlanningAlgorythm[] algorythms = new IProcessPlanningAlgorythm[]  {  new FcfsDirect(), // FCFS direct  new FcfsBackward(), // FCFS back  new RoundRobin(3), // RR  new Sjf(false, false), // Sjf not dislodging, not priorital  new Sjf(false, true), // Sjf not dislodging, priorital  new Sjf(true, false), // Sjf dislodging, not priorital  new Sjf(true, true) // Sjf dislodging, priorital  };    foreach (var a in algorythms)  {  Console.WriteLine(a.ToString());  Console.WriteLine(a.Execute(processes));  Console.WriteLine();  }  Console.ReadKey();  }  }  } |

Файл Process.cs:

|  |
| --- |
| namespace lab6  {  public enum ProcessState { Void, Ready, Running }  public class Process : System.ICloneable  {  public int Duration { get; set; }  public int Appearance { get; set; }  public int Priority { get; set; }  public int DurationResource { get; set; }  public ProcessState State { get; set; } = ProcessState.Void;  public bool IsComplete { get { return DurationResource == 0; } }  public Process(int dur, int appear, int prior)  {  DurationResource = Duration = dur;  Appearance = appear;  Priority = prior;  }  public void ResetProcess()  {  DurationResource = Duration;  State = ProcessState.Void;  }  public object Clone()  {  return new Process(Duration, Appearance, Priority);  }  }  } |

Файл ProcessComparers.cs:

|  |
| --- |
| using System.Collections.Generic;  namespace lab6  {  class ProcessPriorityComparer : IComparer<Process>  {  ProcessDurationComparer durationComparer = new ProcessDurationComparer();  public int Compare(Process x, Process y)  {  var res = x.Priority.CompareTo(y.Priority);  return res != 0 ? res : durationComparer.Compare(x, y);  }  }  class ProcessDurationComparer : IComparer<Process>  {  public int Compare(Process x, Process y)  {  return x.DurationResource.CompareTo(y.DurationResource);  }  }  } |

Файл ProcessRunningSequence.cs:

|  |
| --- |
| using System;  using System.Text;  namespace lab6  {  class ProcessRunningSequence  {  public static char VoidSymbol { get; set; } = ' ';  public static char ReadySymbol { get; set; } = 'Г';  public static char RunningSymbol { get; set; } = 'И';    public int MaxT { get; set; } = 0;  public int ProcessesCount { get { return sequence.Length; } }  ProcessState[][] sequence;  public ProcessRunningSequence(int processesCount, int maxT)  {  sequence = new ProcessState[processesCount][];  for (int i = 0; i < sequence.Length; i++)  {  sequence[i] = new ProcessState[maxT];  }  MaxT = maxT;  }  public void SetSequenceAtTick(ProcessState[] sequence, int tickI)  {  if (sequence.Length != ProcessesCount)  throw new ArgumentException();  for (int i = 0; i < ProcessesCount; i++)  this.sequence[i][tickI] = sequence[i];    if (MaxT < sequence.Length)  MaxT = sequence.Length;  }  private static char GetValueSymbol(ProcessState value)  {  return  value == ProcessState.Running ? RunningSymbol :  value == ProcessState.Ready ? ReadySymbol :  VoidSymbol;  }  public override string ToString()  {  StringBuilder stringBuilder = new StringBuilder(new string('\_', 7+6\* MaxT) + Environment.NewLine );  stringBuilder.Append($"| p\\t |");  for(int i = 0; i < MaxT; i++)  stringBuilder.Append($" {i:d3} |");  stringBuilder.Append(Environment.NewLine);  for (int i = 0; i < sequence.Length; i++)  {  stringBuilder.Append($"| p{i:d2} |");  for (int j = 0; j < MaxT; j++)  stringBuilder.Append($" {GetValueSymbol(sequence[i][j])} |");  stringBuilder.Append(Environment.NewLine);  }  stringBuilder.Append(new string('\_', 7+6\*MaxT));  return stringBuilder.ToString();  }  }  } |

Файл IProcessPlanningAlgorythm.cs:

|  |
| --- |
| namespace lab6  {  interface IProcessPlanningAlgorythm  {  ProcessRunningSequence Execute(Process[] processes);  }  } |

Файл PlanningAlgorythmsHelper.cs:

|  |
| --- |
| namespace lab6  {  static class PlanningAlgorythmsHelper  {  public static void WriteLog(Process[] processes, ProcessRunningSequence runningSequence, int i)  {  var log = new ProcessState[processes.Length];  for (int j = 0; j < log.Length; j++)  log[j] = processes[j].State;  runningSequence.SetSequenceAtTick(log, i);  }  }  } |

Файл FcfsDirect.cs:

|  |
| --- |
| using System.Collections.Generic;  using System.Linq;  namespace lab6  {  class FcfsDirect : IProcessPlanningAlgorythm  {  public ProcessRunningSequence Execute(Process[] processes)  {  ProcessRunningSequence runningSequence = new ProcessRunningSequence(processes.Length, 19);  Queue<Process> queue = new Queue<Process>();  for (int i = 0; i < runningSequence.MaxT; i++)  {  // add all appearing processes at the current tick to the queue  foreach (var item in processes.Where(x => x.Appearance == i && x.State == ProcessState.Void)) //  {  item.State = ProcessState.Ready;  queue.Enqueue(item);  }  if (queue.Count > 0)  {  if (queue.Peek().IsComplete)  {  // go to the next element in queue and set its state as void  queue.Dequeue().State = ProcessState.Void;  }  if (queue.Count > 0)  queue.Peek().State = ProcessState.Running;  }    PlanningAlgorythmsHelper.WriteLog(processes, runningSequence, i);  if (queue.Count > 0)  queue.Peek().DurationResource--;  }  foreach (var item in processes)  {  item.ResetProcess();  }  return runningSequence;  }  }  } |

Файл FcfsBackward.cs:

|  |
| --- |
| using System;  namespace lab6  {  class FcfsBackward : IProcessPlanningAlgorythm  {  public ProcessRunningSequence Execute(Process[] processes)  {  var processesSorted = new Process[processes.Length];  for (int i = 0; i < processes.Length; i++)  processesSorted[i] = processes[i].Clone() as Process;  var appearance = new int[processes.Length];  for (int i = 0; i < processesSorted.Length; i++)  appearance[i] = processesSorted[i].Appearance;    Array.Reverse(appearance);  for (int i = 0; i < processesSorted.Length; i++)  processesSorted[i].Appearance = appearance[i];    return new FcfsDirect().Execute(processesSorted);  }  }  } |

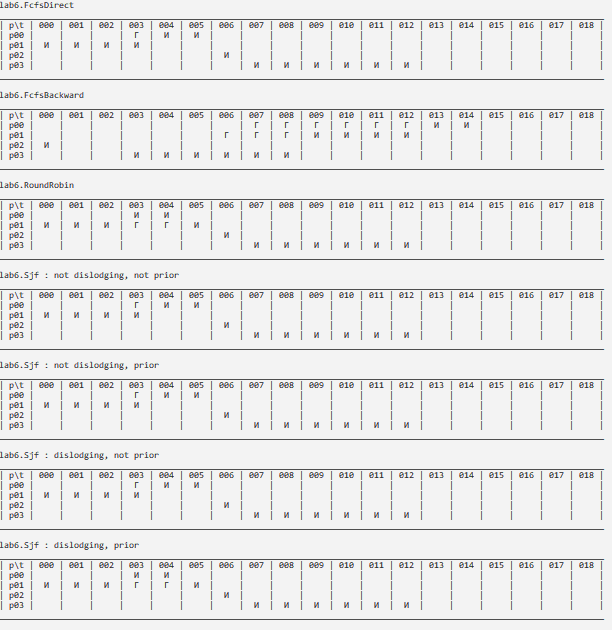
Файл RoundRobin.cs:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  namespace lab6  {  class RoundRobin : IProcessPlanningAlgorythm  {  public int CycleTicks { get; set; }  public RoundRobin(int cycleTicks)  {  if (cycleTicks < 1)  throw new ArgumentException();  CycleTicks = cycleTicks;  }  public ProcessRunningSequence Execute(Process[] processes)  {  ProcessRunningSequence runningSequence = new ProcessRunningSequence(processes.Length, 19);  Queue<Process> queue = new Queue<Process>();  int cycleCounter = 0;  for (int i = 0; i < runningSequence.MaxT; i++, cycleCounter++)  {  // add all appearing processes at the current tick to the queue  foreach (var item in processes.Where(x => x.Appearance == i)) //&& x.State == ProcessState.Void  {  item.State = ProcessState.Ready;  queue.Enqueue(item);  }  if (queue.Count > 0)  {  if (queue.Peek().IsComplete)  {  // go to the next element in queue and set its state as void  queue.Dequeue().State = ProcessState.Void;  if (queue.Count > 0)  queue.Peek().State = ProcessState.Running;  cycleCounter = 0;  }  else if((cycleCounter+1) % (CycleTicks+1) == 0)  {  var deq = queue.Dequeue();  deq.State = ProcessState.Ready;  queue.Enqueue(deq);  cycleCounter = 0;  queue.Peek().State = ProcessState.Running;  }  else  if (queue.Count > 0)  queue.Peek().State = ProcessState.Running;  }    PlanningAlgorythmsHelper.WriteLog(processes, runningSequence, i);  if (queue.Count > 0)  queue.Peek().DurationResource--;  }  foreach (var item in processes)  {  item.ResetProcess();  }  return runningSequence;  }  }  } |

Файл Sjf.cs:

|  |
| --- |
| using System.Collections.Generic;  using System.Linq;  namespace lab6  {  class Sjf : IProcessPlanningAlgorythm  {  IComparer<Process> comparer;  bool dislodging;  public Sjf(bool dislodging = false, bool prioritial = false)  {  if (prioritial)  comparer = new ProcessPriorityComparer();  else  comparer = new ProcessDurationComparer();  this.dislodging = dislodging;  }  public ProcessRunningSequence Execute(Process[] processes)  {  ProcessRunningSequence runningSequence = new ProcessRunningSequence(processes.Length, 19);  Queue<Process> queue = new Queue<Process>();  // add all appearing processes at the btggining to the queue  // then choose min element as the beggining of the queue  int k = 0;  for (k = 0; k < runningSequence.MaxT && queue.Count == 0; k++)  {  foreach (var item in processes.Where(x => x.Appearance == k))  {  item.State = ProcessState.Ready;  queue.Enqueue(item);  }  RollQueueToItem(queue, MinItem(queue));  }  if (queue.Count > 0)  {  queue.Peek().State = ProcessState.Running;  queue.Peek().DurationResource--;  PlanningAlgorythmsHelper.WriteLog(processes, runningSequence, k-1);  }  for (int i = k; i < runningSequence.MaxT; i++)  {  // add all appearing processes at the current tick to the queue  int queuePrevCount = queue.Count();  foreach (var item in processes.Where(x => x.Appearance == i && x.State == ProcessState.Void))  {  item.State = ProcessState.Ready;  queue.Enqueue(item);  }  if (dislodging && queuePrevCount - queue.Count() != 0 && queue.Peek().DurationResource != 0)  {  queue.Peek().State = ProcessState.Ready;  RollQueueToItem(queue, MinItem(queue));  queue.Peek().State = ProcessState.Running;  }  if (queue.Count > 0)  {  if (queue.Peek().IsComplete)  {  // set the complete process state as void  queue.Dequeue().State = ProcessState.Void;  // go to the next min element in queue  if (queue.Count > 0)  RollQueueToItem(queue, MinItem(queue));  }  if (queue.Count > 0)  queue.Peek().State = ProcessState.Running;  }  PlanningAlgorythmsHelper.WriteLog(processes, runningSequence, i);  if (queue.Count > 0)  queue.Peek().DurationResource--;  }  foreach (var item in processes)  {  item.ResetProcess();  }  return runningSequence;  }  public Process MinItem(Queue<Process> processes)  {  // processes may not be void  Process minItem = processes.Dequeue();  processes.Enqueue(minItem);  for (int i = 1; i < processes.Count; i++)  {  var item = processes.Dequeue();  if (comparer.Compare(item, minItem) < 0)  minItem = item;  processes.Enqueue(item);  }  return minItem;  }  public void RollQueueToItem(Queue<Process> processes, Process process)  {  // processes may not be void  for (int i = 0; i < processes.Count; i++)  {  var item = processes.Peek();  if (comparer.Compare(item, process) != 0)  {  item = processes.Dequeue();  processes.Enqueue(item);  }  }  }  public override string ToString()  {  return $"{base.ToString()} : {(dislodging ? "" : "not ")}dislodging, " +  $"{(comparer.GetType() == typeof(ProcessPriorityComparer) ? "" : "not ")}prior";  }  }  } |

**Результаты:**

****

**Вывод:** В ходе выполнения данной лабораторной работы были изучены типовые алгоритмы планирования процессов