**Контрольная работа 1-05**

**Вариант 10 (решения)**

За разговоры с соседом -3 балла за каждый разговор.

1. (14 баллов) Рассмотрим однопроцессорную вычислительную систему с объемом оперативной памяти 200 Mb, в которой используется схема организации памяти с динамическими (переменными) разделами. Для долгосрочного планирования процессов в ней применен алгоритм SJF. В систему поступают пять заданий с различной длительностью и различным объемом занимаемой памяти по следующей схеме:

|  |  |  |  |
| --- | --- | --- | --- |
| Номер задания | Момент поступления в очередь заданий | Время исполнения (CPU burst) | Объем занимаемой памяти |
| 1 | 0 | 3 | 80 Mb |
| 2 | 2 | 4 | 50 Mb |
| 3 | 3 | 5 | 60 Mb |
| 4 | 4 | 2 | 80 Mb |
| 5 | 5 | 1 | 10 Mb |

Вычислите среднее время между стартом задания и его завершением (turnaround time) и среднее время ожидания (waiting time) для следующих комбинаций алгоритмов краткосрочного планирования и стратегий размещения процессов в памяти:

1. RR (Round Robin) и first fit (первый подходящий);
2. RR и best fit (наиболее подходящий);
3. FCFS (First Come First Served) и first fit;
4. FCFS и best fit.

При вычислениях считать, что процессы не совершают операций ввода-вывода, величину кванта времени принять равной 1. Временами переключения контекста, рождения процессов и работы алгоритмов планирования пренебречь. Освобождение памяти, занятой процессами, происходит немедленно по истечении их CPU burst. Краткосрочное планирование осуществляется после рождения новых процессов в текущий момент времени. Для алгоритма RR принять, что родившиеся процессы добавляются в **САМЫЙ** конец очереди готовых процессов (**ПОСЛЕ** процесса, перешедшего в состояние ***готовность*** из состояния ***исполнение*** в это время).

***Решение:***

* 1. Рассмотрим выполнение процессов в системе для алгоритма RR и стратегии first fit. По вертикали в таблице отложены номера процессов, по горизонтали — промежутки времени. Столбец 0 соответствует временному интервалу от 0 до 1. Буква И означает состояние исполнения, буква Г — состояние готовности, буква О — ожидание в очереди заданий. Под таблицей приведено распределение памяти, а еще ниже — содержимое очереди заданий.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | И | И | И |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  | Г | И | Г | И | Г | Г | И | Г | И |  |  |  |  |
| 3 |  |  |  | Г | И | Г | И | Г | Г | И | Г | И | Г | И |  |
| 4 |  |  |  |  | О | О | О | О | О | О | О | Г | И | Г | И |
| 5 |  |  |  |  |  | Г | Г | И |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 80 P1 | 80 P1 | 80 P1 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 |
| 20 | 20 | 10 P5 | 10 P5 | 10 P5 | 20 | 20 | 20 | 80 P4 | 80 P4 | 80 P4 | 80 P4 |
| 10 | 10 | 10 |
| 120 | 120 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 |
| 70 | 70 | 70 | 70 | 70 | 70 | 70 | 70 | 70 |
| 60 | 60 | 60 |  |
| 60 |
|  |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|  |  |  |  | P4 | P4 | P4 | P4 | P4 | P4 | P4 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Среднее время между стартом задания и его завершением: tt = (3 + 9 +11 + 11 + 3)/5 = 7.4.  
Среднее время ожидания: wt = (0 + 5 + 6 + 9 + 2)/5 = 4.4.

* 1. Рассмотрим выполнение процессов в системе для алгоритма RR и стратегии best fit.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | И | И | И |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  | Г | И | Г | И | Г | Г | Г | И | Г | Г | И |  |  |
| 3 |  |  |  | Г | И | Г | Г | И | Г | Г | Г | И | Г | И | И |
| 4 |  |  |  |  | Г | Г | И | Г | Г | Г | И |  |  |  |  |
| 5 |  |  |  |  |  | Г | Г | Г | И |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 80 P1 | 80 P1 | 80 P1 | 80 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 | 80 | 130 | 130 |
| 120 | 120 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 |
| 70 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 |
| 10 | 10 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 | 10 | 10 | 10 | 10 | 10 |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Среднее время между стартом задания и его завершением: tt = (3 + 11 +12 + 7 + 4)/5 = 7.4.  
Среднее время ожидания: wt = (0 + 7 + 7 + 5 + 3)/5 = 4.4.

* 1. Рассмотрим выполнение процессов в системе для алгоритма FCFS и стратегии first fit.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | И | И | И |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  | Г | И | И | И | И |  |  |  |  |  |  |  |  |
| 3 |  |  |  | Г | Г | Г | Г | И | И | И | И | И |  |  |  |
| 4 |  |  |  |  | О | О | О | Г | Г | Г | Г | Г | Г | И | И |
| 5 |  |  |  |  |  | Г | Г | Г | Г | Г | Г | Г | И |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 80 P1 | 80 P1 | 80 P1 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 | 60 | 70 |
| 20 | 20 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 |
| 10 | 10 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 |
| 120 | 120 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 |
| 70 | 70 | 70 | 70 | 70 |
| 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|  |  |  |  | P4 | P4 | P4 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Среднее время между стартом задания и его завершением: tt = (3 + 5 + 9 + 11 + 8)/5 = 7.2.  
Среднее время ожидания: wt = (0 + 1 + 4 + 9 + 7)/5 = 4.2.

* 1. Рассмотрим выполнение процессов в системе для алгоритма FCFS и стратегии best fit.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | И | И | И |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  | Г | И | И | И | И |  |  |  |  |  |  |  |  |
| 3 |  |  |  | Г | Г | Г | Г | И | И | И | И | И |  |  |  |
| 4 |  |  |  |  | Г | Г | Г | Г | Г | Г | Г | Г | И | И |  |
| 5 |  |  |  |  |  | Г | Г | Г | Г | Г | Г | Г | Г | Г | И |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 80 P1 | 80 P1 | 80 P1 | 80 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 80 P4 | 190 |
| 120 | 120 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 P2 | 50 | 50 | 50 | 50 | 50 | 110 | 110 |
| 70 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 | 60 P3 |
| 10 | 10 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 | 10 P5 |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Среднее время между стартом задания и его завершением: tt = (3 + 5 + 9 + 10 + 10)/5 = 7.4.  
Среднее время ожидания: wt = (0 + 1 + 4 + 8 + 9)/5 = 4.4.

*Оценка:*

За каждый алгоритм со стратегией — по 3 балла. Если времена нахождения в очереди заданий включены в подсчет времен — еще 2 балла на всю задачу

1. (12 баллов) В диком каннибальском племени вокруг котла с пищей спят дикари и повар. Изначально в котле находится N порций мяса. Дикари по очереди просыпаются, берут из котла порцию мяса, съедают его и засыпают снова. Дикарь, не обнаруживший мяса в котле, будит повара. Повар находит добычу и снова готовит N порций, не подпуская никого к котлу во время приготовления, после чего тоже засыпает. Используя семафоры Дейкстры и разделяемые переменные, постройте корректную модель происходящего, описав поведение каждого из дикарей и повара с помощью отдельных процессов.

***Решение:***

Заводим 2 семафора lock\_cauldron (для ограничения доступа к котлу) и need\_eat (для активизации повара) и разделяемую переменную Nportion (для количества порций в котле).

Semaphore lock\_cauldron = 1, need\_eat = 0;

Shared int Nportion = N;

|  |  |
| --- | --- |
| Для дикарей  While(1){  P(lock\_cauldron);  if(Nportion == 0)  {Разбудить повара; V(need\_eat);}  **else{**  **Взять порцию; Nportion--;**  V(lock\_cauldron); Съесть порцию; Поспать;  }  } | Для повара  While(1){  P(need\_eat);  Найти добычу и приготовить еду;  Nportion = N;  V(lock\_cauldron);  Лечь спать;  } |

*Оценка:*

Грубые ошибки: нет взаимоисключения, тупиковые ситуации, убитые за попытку взять пищу не вовремя дикари — -8 баллов, средней тяжести: циклы ожидания, прохождение дикарями критических участков без совершения разумных действий — -4 балла. Полный балл только за полностью правильный ответ.

1. (6 баллов) В вычислительной системе с сегментно-страничной организацией памяти и 32-х битовым адресом максимальный размер сегмента составляет 4 Mb, а размер страницы памяти 512 Kb. Для некоторого процесса в этой системе таблица сегментов имеет вид:

|  |  |
| --- | --- |
| Номер сегмента | Длина сегмента |
| 0 | 0x180000 |
| 1 | 0x080000 |

Таблицы страниц, находящихся в памяти, для сегментов 0 и 1 приведены ниже:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сегмент 0 | |  | Сегмент 1 | |
| Номер страницы | Номер кадра  (десятичный) |  | Номер страницы | Номер кадра  (десятичный) |
| 0 | 18 |  | 0 | 32 |
| 3 | 0 |  | 1 | 63 |

Каким физическим адресам соответствуют логические адреса: 0x000f0236, 0х00470111, 0x00502005?

***Решение:***

4 Mb — это 222 байт, т.е. под номер сегмента в логическом адресе отводится 10 бит, а 22 бита — под смещение внутри сегмента. Размер страницы 512 Kb — это 219 байт, т.е. из смещения внутри сегмента 19 бит отводится под смещение внутри страницы, а 3 бита — под номер страницы.

**0x000f0236** —> сегмент 0, смещение 0x0f0236 —> сегмент 0, страница 1, смещение 0x00070236 —>**error**,   
**0x00470111** —> сегмент 1, смещение 0x070111 —> сегмент 1, страница 0, смещение 0x00070236 — > кадр 32, смещение 0x00070236 —> **0x01070236**,   
**0x00502005** —> сегмент 1, смещение 0x102005 —> смещение больше размера сегмента —> **error**.

*Оценка:*

По 2 балла за адрес:

1. (6 баллов) Ответьте на следующие вопросы
2. Для чего в мониторах Хора применяются условные переменные? Можно ли придумать задачу на взаимодействие процессов, которая решалась бы с помощью мониторов Хора без использования условных переменных?
3. В чем заключается разница между процессом и нитями исполнения (thread’ами), реализованными на уровне ядра ОС?

***Решение:***

1. Условные переменные в мониторах Хора, как правило, применяются для взаимной синхронизации процессов, требующейся для обеспечения правильной очередности их доступа к разделяемым ресурсам. Задачу, для решения которой можно обойтись мониторами Хора без использования условных переменных придумать очень легко. Это может быть, например, задача «подсчет количества запусков программ» из семинара 6-7.
2. В системах, поддерживающих нити исполнения на уровне ядра, thread’ы представляют собой единицы исполнения, а процессы — единицы выделения ресурсов. Процесс представляется как совокупность взаимодействующих нитей и выделенных ему ресурсов. Нити процесса разделяют его программный код, глобальные переменные и системные ресурсы, но каждая нить имеет свой собственный программный счетчик, свое содержимое регистров и свой собственный стек. Планирование использования процессора происходит в терминах нитей, а управление памятью и другими системными ресурсами остается в терминах процессов.

*Оценка:*

За каждый пункт предполагается по 3 балла.