Практическая работа №4

Тема: Синхронизация потоков

Теория:

Жизненный цикл процесса от его порождения до завершения в любой многозадачной (многопроцессной) системе содержит ряд состояний и переходов между ними в зависимости от предоставляемых и доступных процессу ресурсов в каждый момент времени. Часть состояний являются активными, часть пассивными по отношению к возможности использования основного ресурса – процессора. Количество состояний и переходов зависит от конкретной операционной системы и степени детализации представления жизненного цикла. В любой системе выделяют три основных типа состояний: выполнение – активные состояния процесса, когда процесс обладает всеми необходимыми ресурсами и непосредственно выполняется процессором; ожидание – пассивное состояние процесса, процесс заблокирован, он не может выполняться по своим внутренним причинам. Он ждет осуществления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого процесса, освобождения какого-либо необходимого ему ресурса.Обобщенно можно сказать, что процессу не хватает какого-либо ресурса, отличного от процессора. Для этого типа состояний характерно наличие очередей процессов по ресурсам в соответствии с приоритетами доступа к каждому из этих ресурсов; готовность – также пассивное состояние процесса, но в этом случае процесс заблокирован в связи с внешними по отношению к нему обстоятельствами: процесс имеет все требуемые для него ресурсы, он готов выполняться, однако процессор занят выполнением другого процесса. Для этого типа состояний также характерно наличие очередей процессов по приоритетам по отношению к процессору.

Очереди процессов представляют собой дескрипторы отдельных процессов, объединенные в списки. Таким образом, каждый дескриптор, кроме всего прочего, содержит, по крайней мере, один указатель на другой дескриптор, соседствующий с ним в очереди. Такая организация очередей позволяет легко их переупорядочивать, включать и исключать процессы, переводить процессы из одного состояния в другое.В ходе жизненного цикла каждый процесс переходит из одного состояния в другое в соответствии с алгоритмом планирования процессов, реализуемым в данной операционной системе.

Переход от выполнения одного процесса или потока к другому осуществляется в результате планирования – работы по определению того, в

какой момент необходимо прервать выполнение текущего активного процесса (потока) и какому процессу (потоку) предоставить возможность выполняться. Планирование потоков осуществляется на основе информации, хранящейся в описателях процессов и потоков. ОС общего назначения в большинстве случаев планируют выполнение потоков независимо от того, принадлежат ли потоки одному процессу или разным. В системе, не поддерживающей потоки, соблюдаются все описываемые здесь подходы, применительно к процессам. В дальнейшем для краткости будем говорить только о потоках, подразумевая, что все это верно и для процессов. Таким образом, планирование выполняет решение двух задач: - определение момента времени для смены текущего активного потока; - выбор потока для выполнения из очереди готовых потоков. Обе задачи решаются программными средствами. Существует множество различных алгоритмов планирования, по-разному решающих выше перечисленные задачи, преследующих различные цели и обеспечивающих различное качество мультипрограммирования.

Существует обширный класс средств ОС, с помощью которых обеспечивается взаимная синхронизация процессов и потоков. Потребность в синхронизации потоков связана с совместным использованием ресурсов вычислительной системы. Чтобы избежать условий гонки и взаимоблокировок, необходимо синхронизировать доступ нескольких потоков с общими ресурсами. Синхронизация также необходима, чтобы убедиться, что код взаимозависимости выполняется в правильной последовательности.

Существует несколько объектов, дескриптор которых можно использовать для синхронизации нескольких потоков. К этим объектам относятся:

- Входные буферы консоли
- События
- Mutexes
- Процессы
- Семафоры
- Потоки
- Таймеры

Состояние каждого из этих объектов либо сигнализируется, либо не сигнализируется. При указании дескриптора для любого из этих объектов в вызове одной из функций ожидания выполнение вызывающего потока блокируется до тех пор, пока состояние указанного объекта не будет сигнализировать.

Некоторые из этих объектов полезны при блокировке потока до тех пор, пока не произойдет какое-то событие. Например, дескриптор буфера ввода консоли сигнализируется при непрочитанных входных данных, таких как нажатие клавиши или нажатие кнопки мыши. Дескрипторы процессов и потоков сигналит о завершении процесса или потока. Это позволяет процессу, например, создать дочерний процесс, а затем заблокировать собственное выполнение до завершения нового процесса.

Другие объекты полезны для защиты общих ресурсов от одновременного доступа. Например, у нескольких потоков может быть дескриптор объекта мьютекса. Перед доступом к общему ресурсу потоки должны вызвать одну из функций ожидания, чтобы ожидать передачи сигнала о состоянии мьютекса. Когда мьютекс становится сигналом, для доступа к ресурсу освобождается только один ожидая поток. Состояние мьютекса немедленно сбрасывается так, чтобы другие ожидающие потоки оставались заблокированными. После завершения работы потока с ресурсом необходимо задать состояние мьютекса, чтобы разрешить другим потокам доступ к ресурсу.

Ход практической работы:

1. Реализовать синхронизацию процессов в трех контрольных точках (смотри рисунок) с помощью объекта синхронизации ядра – «семафор» (по варианту задания). Все процессы в контрольных точках запускаются, процессом, пришедшим в контрольную точку первым. Разработать алгоритм синхронизации взаимодействующих процессов в контрольных точках и приложение на языке высокого уровня. Создать интерфейс программы или консольное приложение, для контроля синхронизации процессов в контрольных точках.

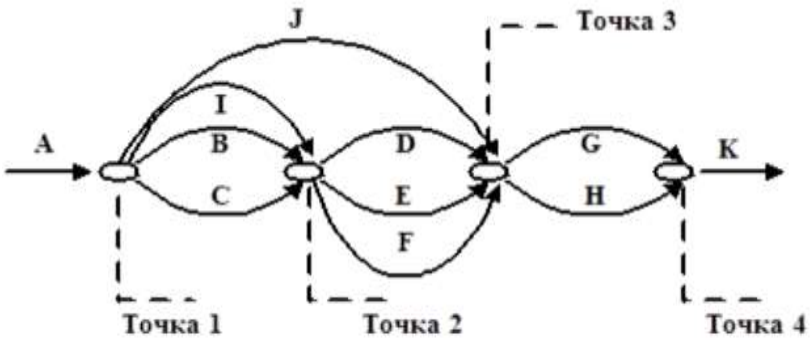
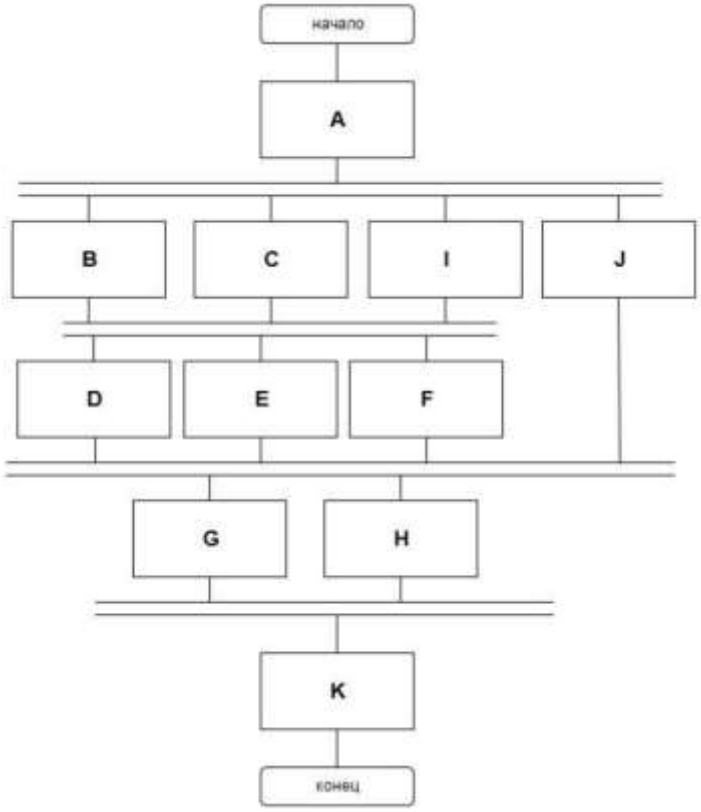


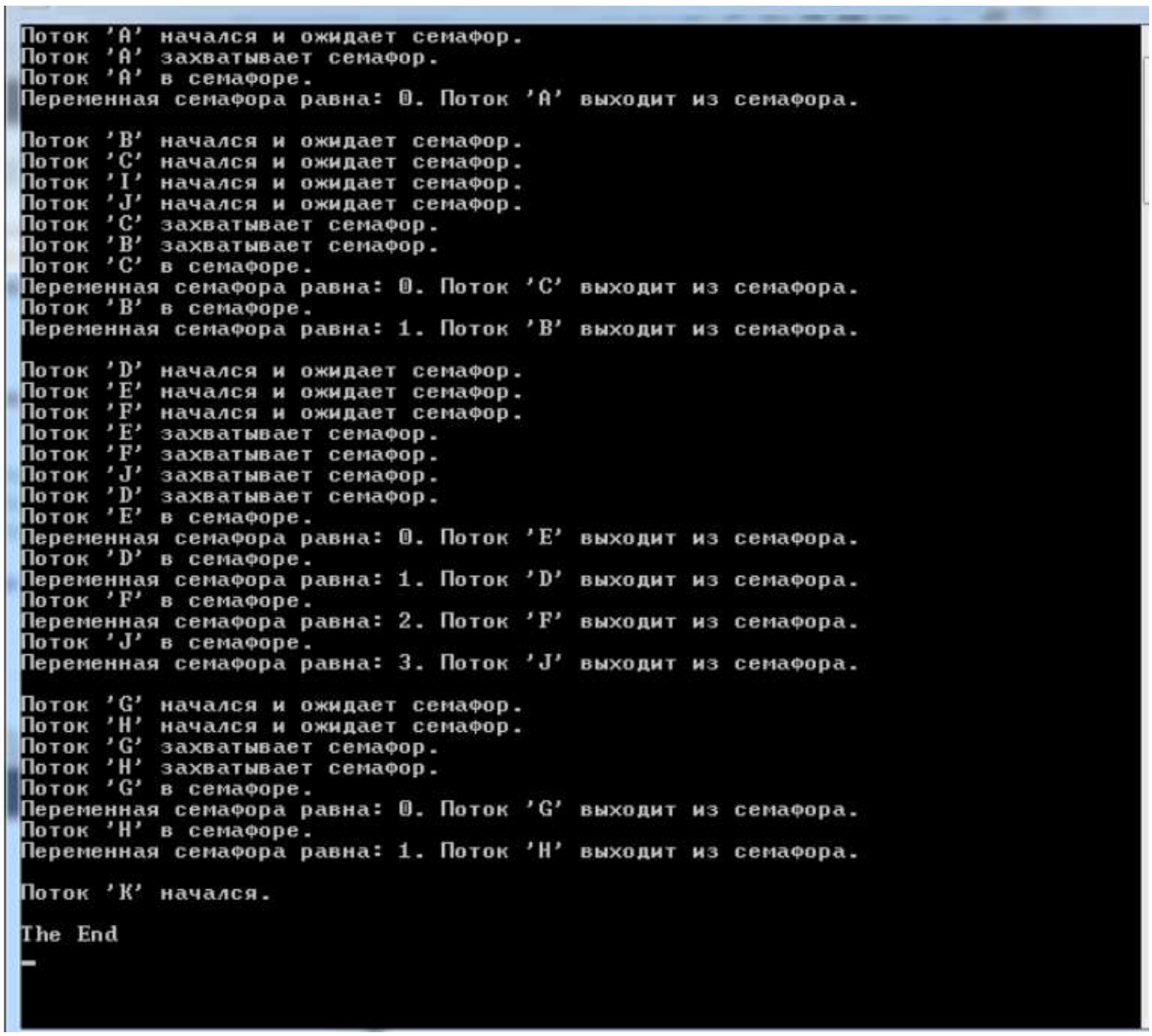
Схема взаимодействия параллельно выполняющихся задач

Схема алгоритма работы программы

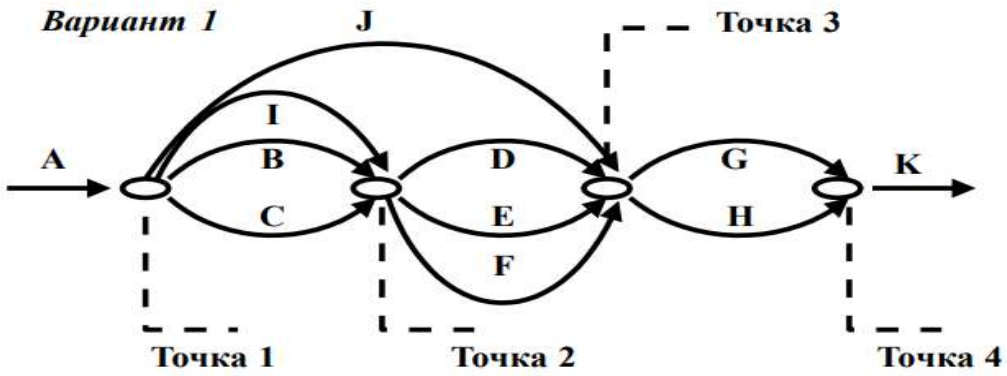


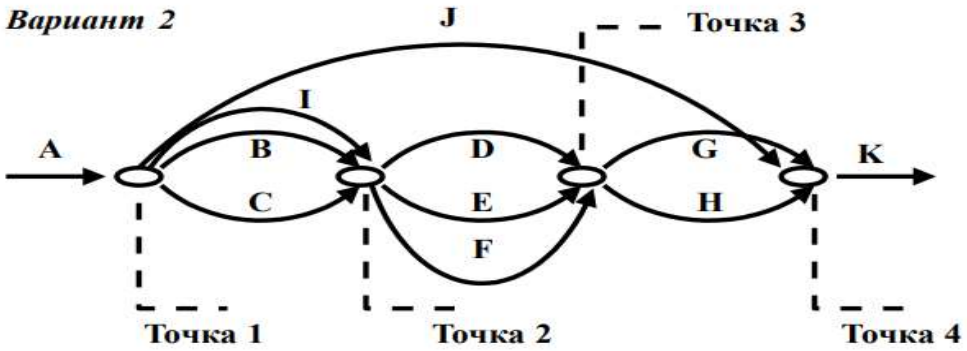
В соответствии с алгоритмом: Процесс A запускает процессы B, C, I, J, процесс, выполняющийся первым из процессов B, C, I, запускает процессы D, E, F, а процесс J конкурирует в запуске процессов G, H с первым, выполнившимся из процессов D, E, F и, в заключении, первый, выполнившийся из процессов G, H, запускает процесс K, который завершает программу. Синхронизация (блокировка) процессов происходит в точках 2, 3, 4.

Результат работы консольного приложения представлен на рисунке. Информация о запуске и выполнении потоков в контрольных точках выводится на экран в текстовом режиме.



Варианты заданий:





Вопросы:

1. Какие параллельные процессы называются независимыми, какиевзаимодействующими?

2. Что такое метод критической секции?

3. Какое число потоков является оптимальным для конкретной вычислительной системы?

4. Общие понятия о программах, процессах и потоках выполнения?

5. Жизненный цикл процесса?

6. Виды межпроцессного взаимодействия?

7. Семафоры?

Требования к содержанию и оформлению отчета

Отчет по практической работе должен содержать:

а) титульный лист;

б) описание хода выполнения работы команд в ОС Windows и AstraLinux (либо любая версия Linux) и снимки экрана;

в) ответы на вопросы;

г) отчет по практическим работам загружается на СДО (online-edu.mirea.ru).