1. Понятие операционной системы…

Проблемы использования современных ЭВМ

- Сложность эффективного управления устройствами
  - Обеспечение независимости функционирования устройств
  - Введение устройств управления для отдельных функциональных подсистем ЭВМ
  - Организация взаимодействия на основе механизма событий и прерываний
  - Аппаратная поддержка обработки прерываний (регистр состояний , операции сохранения и восстановления состояния)
- Трудоемкость использования сложных устройств
  - Разработка программного обеспечения для снижения сложности использования устройств (драйверы)
- Неравномерность загрузки устройств при однопрограммном режиме использования ЭВМ

1. Понятие операционной системы

Определение 5.1. Под операционной системой обычно понимается комплекс взаимосвязанных системных программ, выполняемый на компьютере для снижения сложности эффективного использования ЭВМ

2. Функции операционной системы

- Управление работой устройств ЭВМ
- Обеспечение средств более простого использования ресурсов для пользователей (программистов)
- Распределение ресурсов ЭВМ при многопрограммном режиме работы компьютера

1. Концепция процесса…

- Основой для построения моделей функционирования набора одновременно выполняемых программ является понятие *процесса*
- Представление множества одновременно выполняемых программ в виде набора процессов позволяет:
  - выполнить анализ проблем организации *взаимодействия процессов*, обычно возникающих при потреблении ресурсов,
  - определить моменты и способы обеспечения с*инхронизации и взаимоисключения процессов* при использовании неразделяемых ресурсов
  - изучить условия возникновения или доказать отсутствие *тупиков* в ходе выполнения программ из-за нехватки ресурсов

1. Концепция процесса…

- Процесс может пониматься как "*некоторая последовательность команд, претендующая наравне с другими процессами программы на использование процессора для своего выполнения*".
- Конкретизация понятия процесса зависит от целей исследования параллельных программ. Для анализа проблем организации взаимодействия процессов процесс можно рассматривать как последовательность команд

$$p_n = (i_1, i_2, \cdots, i_n)$$

(для простоты изложения материала будем предполагать, что процесс описывается единственной командной последовательностью)

1. Концепция процесса…

Динамика развития процесса определяется моментами времен начала выполнения команд (*траектория процесса*)

$$t(p_n) = t_p = (\tau_1, \tau_2, \ldots, \tau_n)$$

где τ_j, $1 \le j \le n$, есть время начала выполнения команды τ_j

Предположения:

- Команды процесса исполняются строго последовательно,
- Команды в ходе своей реализации не могут быть приостановлены (т.е. являются неделимыми)
- Команды имеют одинаковую длительность выполнения, равную 1

$$\forall i, \quad 1 \le i < n \quad \Rightarrow \quad \tau_{i+1} \ge \tau_i + 1$$

1. Концепция процесса

- $\tau_{i+1} = \tau_i + 1$ - процесс *активен*,
- $\tau_{i+1} \geq \tau_i + 1$ - процесс *приостановлен* (т.е. находится в состоянии *ожидания* или *блокировки*)

2. Модель выполняемого набора программ как системы процессов…

- Понятие процесса может быть использовано в качестве основного *конструктивного элемента* для построения модели выполняемого набора программ как системы *взаимодействующих процессов*.
- Подобное представление набора программ позволяет:
  - получить более компактные (поддающиеся анализу) вычислительные схемы многопрограммного режима работы ЭВМ,
  - выделить в явном виде моменты взаимодействия процессов при потреблении ресурсов ЭВМ,
  - Оценивать эффективность применяемых методов распределения ресурсов

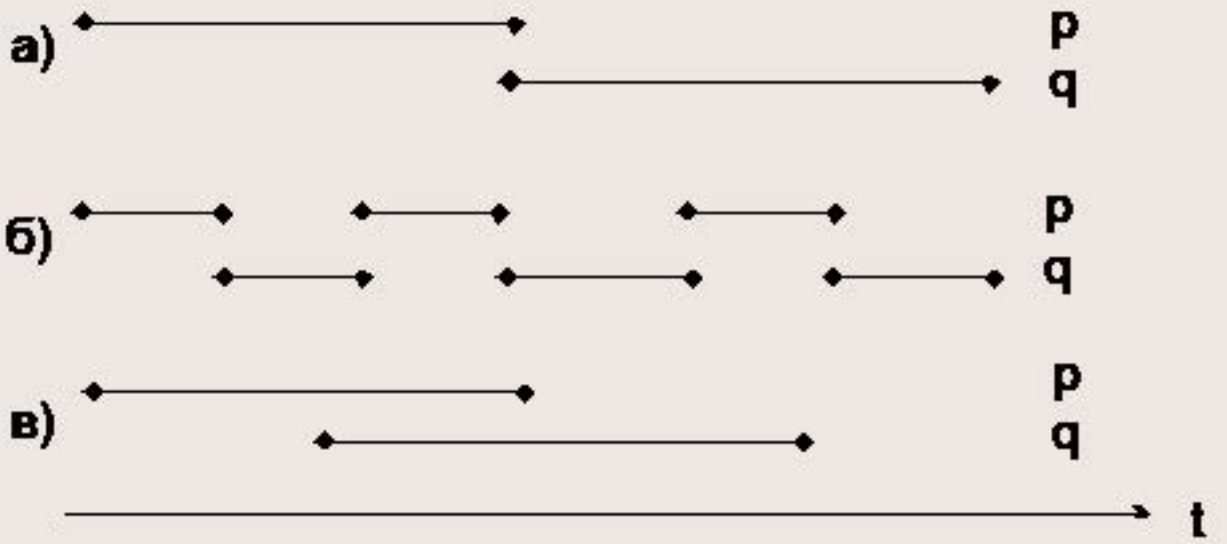
2. Модель выполняемого набора программ как системы процессов…

Возможные типовые варианты соотношений временных траекторий на примере двух одновременно выполняемых процессов ***p*** и ***q*** состоят в следующем :

– выполнение процессов осуществляется строго последовательно, т.е. процесс ***q*** начинает свое выполнение только после полного завершения процесса ***p*** (*однопрограммный режим* работы ЭВМ),

– выполнение процессов может осуществляться одновременно, но в каждый момент времени могут исполняться команды только какого либо одного процесса (*режим разделения времени* или *многопрограммный режим* работы ЭВМ),

– *параллельное выполнение* процессов, когда одновременно могут выполняться команды нескольких процессов (данный режим исполнения процессов осуществим только при наличии в вычислительной системе нескольких процессоров).


а)
p
q
б)
p
q
в)
p
q
t

3. Понятие ресурса

- Понятие *ресурса* обычно используется для обозначения любых объектов вычислительной системы, которые могут быть использованы процессом для своего выполнения. В качестве ресурса может рассматриваться процесс, память, программы, данные и т.п.
- Различают следующие категории ресурсов:
  - *выделяемые* (монопольно используемые, неперераспределяемые) ресурсы;
  - *повторно распределяемые ресурсы*;
  - *разделяемые* ресурсы;
  - *многократно используемые* (реентерабельные) ресурсы.

4. Описание состояния вычислительной системы...

Состояние вычислительной системы может быть представлено в виде ориентированного графа (V,E) со следующей интерпретацией и условиями:

- Множество V разделено на два взаимно пересекающихся подмножества P и R, представляющие *процессы*

$$P=(p_1, p_2,\ldots, p_n)$$

и *ресурсы*

$$R=(R_1, R_2,\ldots, R_m)$$

ВС.

- Граф является "двудольным" по отношению к подмножествам вершин P и R, т.е. каждое ребро $e \in E$ соединяет вершину P с вершиной R. Если ребро имеет вид $e=(p_i, R_j)$, то e есть ребро *запроса* и интерпретируется как запрос от процесса p_i на единицу ресурса R_j. Если ребро e имеет вид $e=(R_j, p_i)$, то e есть ребро *назначения* и выражает назначение единицы ресурса R_j процессу p_i.

4. Описание состояния вычислительной системы…

- Для каждого ресурса $R_j \in R$ существует целое $k_j \geq 0$, обозначающее количество единиц ресурса R_j.
- Пусть $|(a,b)|$ - число ребер, направленных от вершины a к вершине b. Тогда при принятых обозначениях для ребер графа должны выполняться условия:
  - Может быть сделано не более k_j назначений (распределений) для ресурса R_j, т.е.

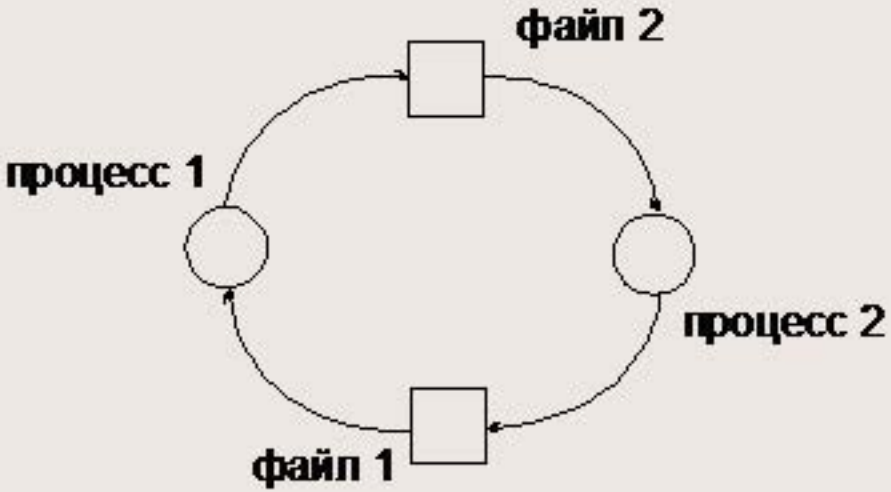
$$\sum_i |(R_j, p_i)| \leq k_j \; 1 \leq j \leq m \; ;$$

  - Сумма запросов и распределений относительно любого процесса для конкретного ресурса не может превышать количества доступных единиц, т.е.

$$|(R_j, p_i)| + |(p_i, R_j)| \leq k_j \; 1 \leq j \leq n \; , \; 1 \leq j \leq m \; .$$

4. Описание состояния вычислительной системы..

Граф, построенный с соблюдением всех перечисленных правил, называется как *граф "процесс-ресурс"*.



Ресурс 1 (файл 1) выделен процессу 1, который, в свою очередь, выдал запрос на ресурс 2 (файл 2). Процесс 2 владеет ресурсом 2 и нуждается для своего продолжения в ресурсе 1.

5. Правила изменения состояний…

Состояние ВС, представленное в виде графа "процесс-ресурс", изменяется только в результате *запросов*, *освобождений* или *приобретений* ресурсов каким-либо из процессов программы.

- **Запрос.** Если программа находится в состоянии S и процесс p_i не имеет невыполненных запросов, то p_i может запросить любое число ресурсов. Тогда программа переходит в состояние T

$$S \xrightarrow{i} T$$

Состояние T отличается от S только дополнительными ребрами запроса p_i от к затребованным ресурсам.

5. Правила изменения состояний…

- **Приобретение**. Операционная система может изменить состояние программы S на состояние T в результате операции приобретения ресурсов процессом p_i, тогда и только тогда, когда p_i имеет запросы на выделение ресурсов и все такие запросы могут быть удовлетворены, т.е. если

$$\forall R_j : (p_i, R_j) \in E \quad \Rightarrow \quad (p_i, R_j) + \sum_l \left| (R_j, p_l) \right| \le k_j$$

Граф T идентичен S за исключением того, что все ребра запроса (p_i, R_j) для p_i обратны ребрам (R_j, p_i), что отражает выполненное распределение ресурсов.

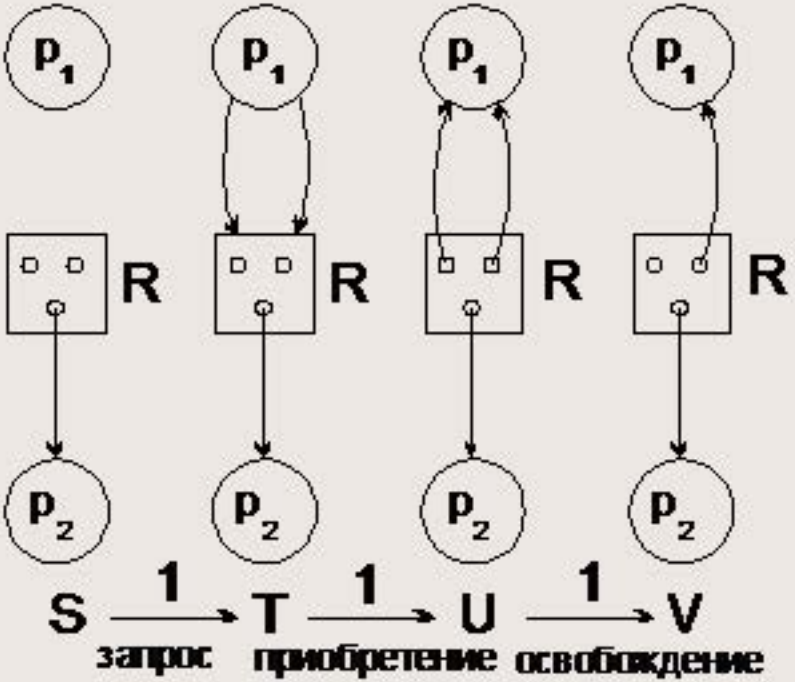
5. Правила изменения состояний…

• **Освобождение**. Процесс p_i может вызвать переход из состояния S в состояние T с помощью освобождения ресурсов тогда и только тогда, когда p_i не имеет запросов, а имеет некоторые распределенные ресурсы, т.е.

$$\forall R_j : (p_i, R_j) \notin E \quad \exists R_j : (R_j, p_i) \in E$$

В этой операции p_i может освободить любое непустое подмножество своих ресурсов. Результирующее состояние T идентично исходному состоянию S за исключением того, что в T отсутствуют некоторые ребра приобретения из S (из S удаляются ребра (R_j, p_i) каждой освобожденной единицы ресурса R_j).

5. Правила изменения состояний…



Для примера показаны состояния ВС с одним ресурсом емкости 3 и двумя процессами после выполнения операций запроса, приобретения и освобождения ресурсов для первого процесса.

5. Правила изменения состояний

При рассмотрении переходов вычислительной системы из состояния в состояние важно отметить, что *поведение процессов является недетерминированным* – при соблюдении приведенных выше ограничений выполнение любой операции любого процесса возможно в любое время

6. Модель вычислительной системы…

• Под *вычислительной системой* будем понимать модель

$$\langle \Sigma, P \rangle$$

где Σ есть множество состояний ВС **(S,T,U,…)**,
а P представляет множество процессов $(p_1, p_2, \ldots, p_n)$.

• *Процесс* $p_i \in R$ есть частичная функция, отображающая состояния ВС в непустые подмножества состояний

$$p_i : \Sigma \to \{\Sigma\}$$

где $\{\Sigma\}$ есть множество всех подмножеств Σ.

• Обозначим множество состояний, в которые может перейти ВС при помощи процесса p_i (*область значений процесса* p_i) при нахождении программы в состоянии S через $p_i(S)$. Возможность перехода ВС из состояния S в состояние T в результате некоторой операции над ресурсами в процессе p_i (т.е. $T \in p_i(S)$) будем пояснять при помощи записи

$$S \xrightarrow{i} T$$

6. Модель вычислительной системы

- Обобщим данное обозначение для указания достижимости состояния T из состояния S в результате выполнения некоторого произвольного количества переходов в программе

$$S \xrightarrow{*} T \Leftrightarrow (S = T) \vee (\exists p_i \in P : S \xrightarrow{i} T) \vee (\exists p_i \in P, U \in \Sigma : S \xrightarrow{i} U, U \xrightarrow{*} T)$$

7. Обнаружение и исключение тупиков…

Проблема тупиков является значительной при организации параллельных вычислений:

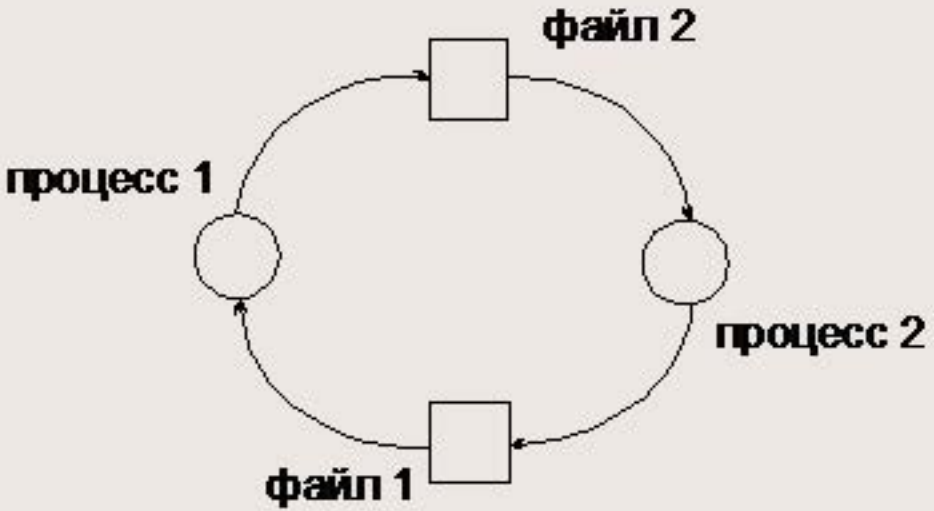
- Сложность диагностирования состояния тупика (система выполняет длительные расчеты или "зависла" из-за тупика),
- Необходимость определенных специальных действий для выхода из тупика,
- Возможность потери данных при восстановлении системы при устранении тупика.

7. Обнаружение и исключение тупиков…

В самом общем виде *тупик (клинч, дедлок, взаимная блокировка, смертельное объятие)* может быть определен как ситуация, в которой один или несколько процессов ожидают какого-либо события, которое никогда не произойдет.

Состояние тупика может наступить не только вследствие логических ошибок, допущенных при разработке параллельных программ, но и в результате возникновения тех или иных событий в вычислительной системе (выход из строя отдельных устройств, нехватка ресурсов и т.п.).

7. Обнаружение и исключение тупиков…



Ситуация тупика: первому процессу для продолжения работы требуется файл второго процесса и одновременно второму процессу необходим файл первого процесса

7. Обнаружение и исключение тупиков…

Рассмотрим один из аспектов проблемы тупика – анализ причин возникновения тупиковых ситуаций при использовании разделяемых ресурсов. Могут быть выделены следующие **необходимые условия тупика**:

– процессы требуют предоставления им права монопольного управления ресурсами, которые им выделяются (*условие взаимоисключения*);

– процессы удерживают за собой ресурсы, уже выделенные им, ожидая в то же время выделения дополнительных ресурсов (условие *ожидания* ресурсов);

– ресурсы нельзя отобрать у процессов, удерживающих их, пока эти ресурсы не будут использованы для завершения работы (условие *неперераспределяемости*);

- существует кольцевая цепь процессов, в которой каждый процесс удерживает за собой один или более ресурсов, требующихся следующему процессу цепи (условие *кругового ожидания*).

7. Обнаружение и исключение тупиков…

Как результат, для обеспечения отсутствия тупиков необходимо исключить возникновение, по крайней мере, одного из рассмотренных условий.

Модель программы в виде графа "процесс-ресурс" может быть использована для обнаруживания ситуации кругового ожидания.

7. Обнаружение и исключение тупиков…

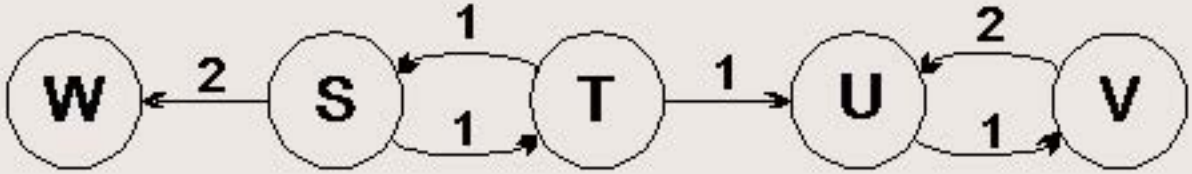
С учетом построенной модели и введенных обозначений можно выделить ряд ситуаций, возникающих при выполнении программы и представляющих интерес при рассмотрении проблемы тупика:

- *процесс* p_i *заблокирован* в состоянии S, если программа не может изменить свое состояние при помощи этого процесса, т.е. если $p_i(S)=\varnothing$;
- *процесс* p_i находится в *тупике* в состоянии S, если этот процесс является заблокированным в любом состоянии T, достижимом из состояния , т.е.

$$\forall T : S \xrightarrow{*} T \Rightarrow p_i(T) =$$

7. Обнаружение и исключение тупиков…

- *состояние* ***S*** называется *тупиковым*, если существует процесс p_i, находящийся в тупике в этом состоянии;
- *состояние* ***S*** есть *безопасное состояние*, если любое состояние ***T***, достижимое из ***S***, не является тупиковым.



Пример: состояния U и V являются безопасными, состояния S и T и W не являются безопасными, а состояние W есть состояние тупика.

7. Обнаружение и исключение тупиков…

Рассмотренная модель программы может быть использована для определения возможных состояний программы, обнаружения и недопущения тупиков.
В• качестве возможных теоретических результатов такого анализа может быть приведена теорема [Шоу].
Теорема 5.1. Граф "процесс-ресурс" для состояния программы с ресурсами единичной емкости указывает на состояние тупика тогда и только тогда, когда он содержит цикл.

Заключение

- Операционная система обеспечивает возможность более простого использования ЭВМ (автоматизация управления устройствами, обеспечение удобного виртуального уровня работы с аппаратурой компьютера, эффективное распределение ресурсов)
- Моделирование работы вычислительной системы как комплекса процессов и ресурсов
- Обнаружение и исключение тупиковых ситуаций при функционировании вычислительной системы