

Основы современных операционных систем

Лекция 13

Сафонов Владимир Олегович

Профессор кафедры информатики,

Заведующий лабораторией Java-технологии

мат-мех. факультета СПбГУ

Email: vosafonov@gmail.com

Сайт лаборатории: <http://polyhimnie.math.spbu.ru/jtl>

Тупики (deadlocks)

- **Модель системы**
- **Характеристики тупиков**
- **Обработка тупиков**
- **Предотвращение тупиков**
- **Как избежать тупиков**
- **Обнаружение тупиков**
- **Восстановление после выхода из тупика**
- **Комбинированный подход к обработке тупиков**

Проблема тупиков

- Тупик - множество заблокированных процессов, каждый из которых владеет некоторым ресурсом и ожидает ресурса, которым владеет какой-либо другой процесс из этого множества
- Пример
 - Система имеет два внешних устройства.
 - P_1 и P_2 – каждый пользуется некоторым устройством и требует использования другого устройства.
- Пример
 - семафоры A и B , инициализированные 1

P_1

P_2

$wait(A); wait(B)$

$wait(B); wait(A)$

Модель системы

- Типы ресурсов - R_1, R_2, \dots, R_m
Процессор, память, устройства ввода-вывода
- Каждый тип ресурса R_i имеет W_i экземпляров.
- Каждый процесс использует ресурс с помощью одним из следующих способов:
 - request (запрос)
 - use (использование)
 - release (освобождение)

Характеристики тупика

Тупик может возникнуть, если одновременно выполняются четыре условия:

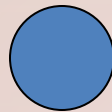
- **Взаимное исключение:** Только один процесс в каждый момент времени может получить доступ к ресурсу
- **Удержание и ожидание:** процесс, удерживающий один ресурс, ожидает приобретения других ресурсов, которыми обладают другие процессы.
- **Отсутствие прерываний:** ресурс может быть освобожден процессом только добровольно, когда процесс завершает свою работу.
- **Циклическое ожидание:** существует множество процессов $\{P_0, P_1, \dots, P_n\}$, таких, что P_0 ожидает ресурса, которым обладает P_1 , P_1 ожидает ресурса, которым обладает P_2, \dots, P_{n-1} ожидает ресурса, которым обладает P_n , а P_n ожидает ресурса, которым владеет P_0 .

Граф распределения ресурсов

- Множество вершин V и множество дуг E .
- V подразделяется на два типа вершин:
 - $P = \{P_1, P_2, \dots, P_n\}$, множество всех процессов в системе.
 - $R = \{R_1, R_2, \dots, R_m\}$, множество всех ресурсов в системе.
- Дуга типа “запрос” (request edge) – направленная дуга $P_i \rightarrow R_j$
- Дуга типа “присваивание” (assignment edge) – направленная дуга $R_j \rightarrow P_i$

Граф распределения ресурсов (продолжение)

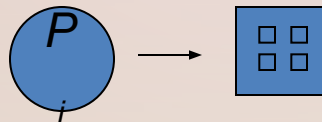
- Процесс



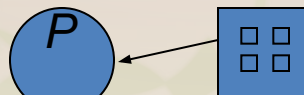
- Тип ресурса, имеющий 4 экземпляра



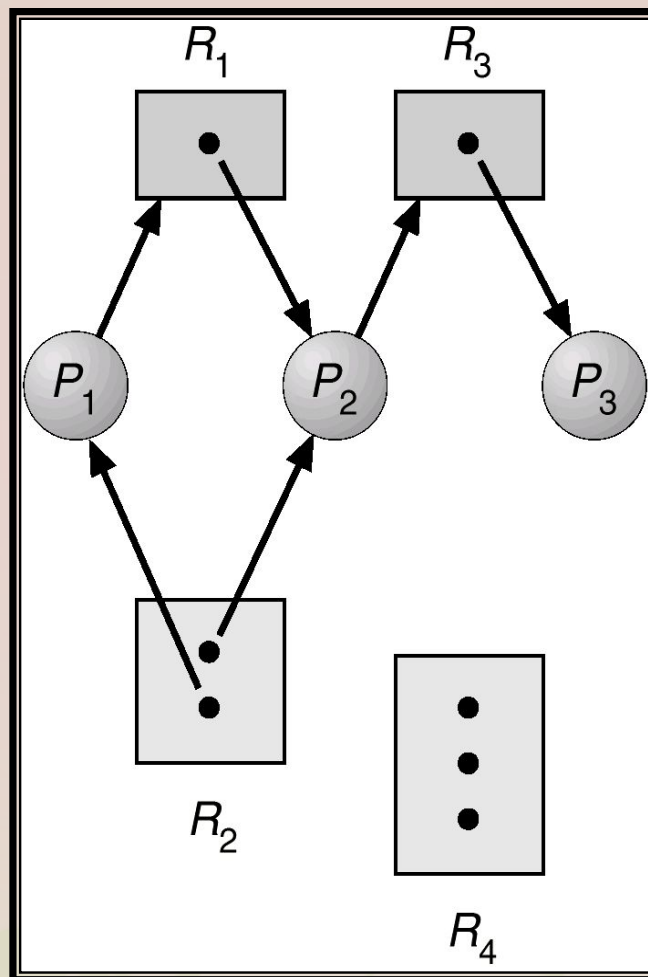
- P_i запрашивает экземпляр ресурса R_j



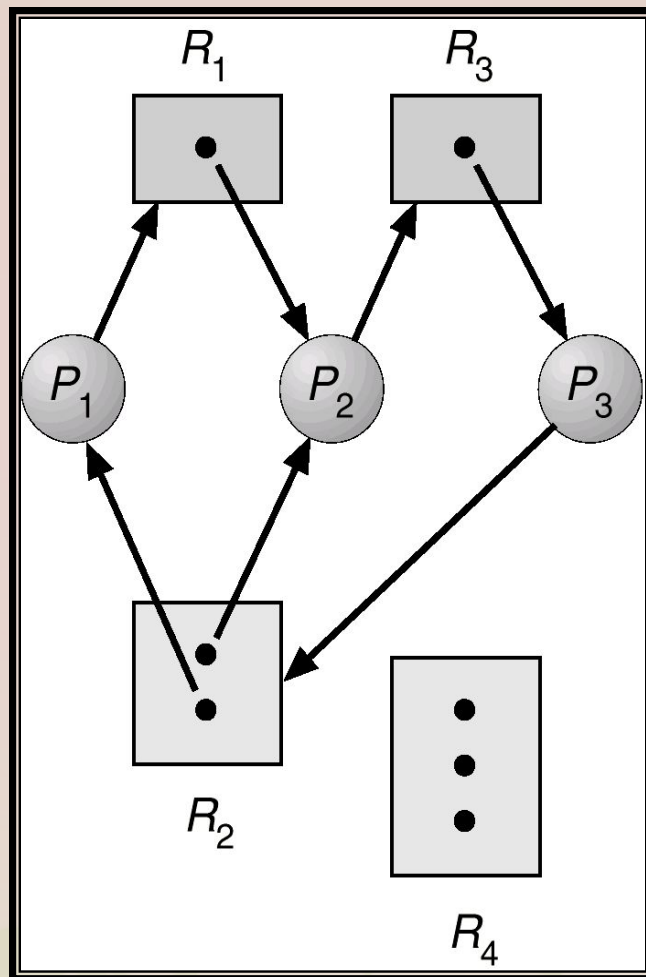
- P_i удерживает экземпляр ресурса R_j



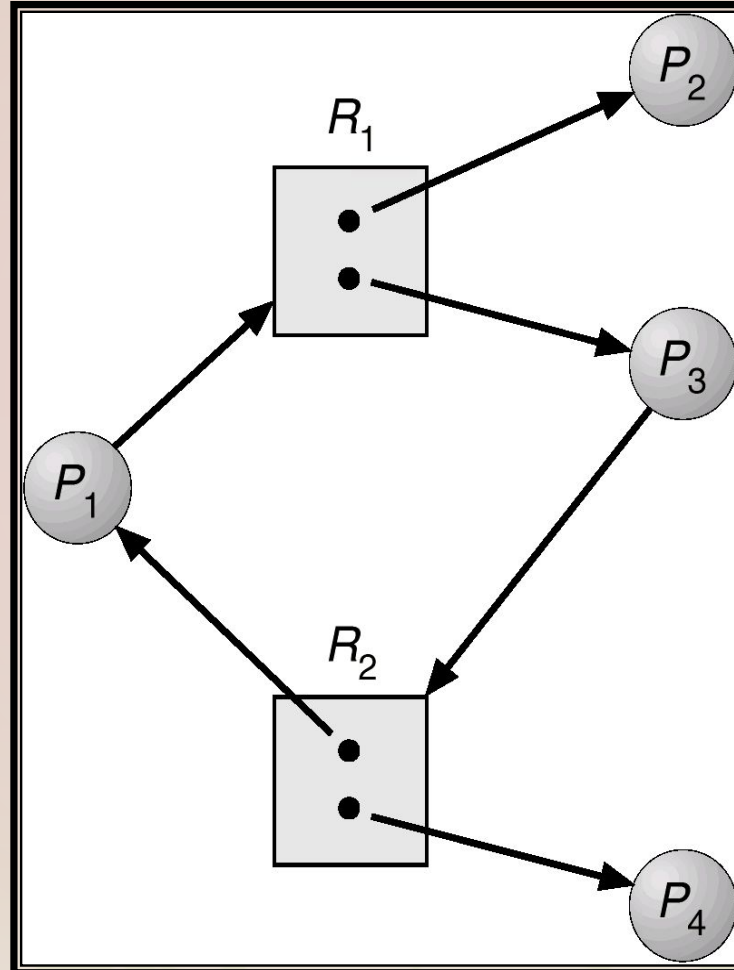
Пример графа распределения ресурсов



Граф распределения ресурсов с тупиком



Граф распределения ресурсов с циклом, но без тупика



Основные утверждения (факты)

- **Граф не содержит циклов \Rightarrow тупиков нет.**
- **Граф содержит цикл \Rightarrow**
 - **Если ресурсов каждого типа только по одному экземпляру, то тупик.**
 - **Если ресурсов по несколько экземпляров, то возможность тупика.**

Методы обработки тупиков

- Убедиться в том, что система никогда не войдет в состояние тупика.
- Допустить, чтобы система могла входить в состояние тупика, но затем восстанавливалась (recover).
- Игнорировать эту проблему, но претендовать на то, что в системе тупики невозможны 😊 (используется практически во всех ОС, включая UNIX).

Предотвращение тупиков

Ограничить методы запросов

- **Взаимное исключение – не требуется для разделяемых ресурсов; должно соблюдаться только для не разделяемых ресурсов.**
- **Удержание и ожидание – необходимо гарантировать, чтобы, когда процесс запрашивает некоторый ресурс, он не владел бы никакими другими ресурсами.**
- **Либо требовать от процессов, чтобы они запрашивали и приобретали все необходимые ресурсы до начала их исполнения, либо требовать, чтобы процесс, запрашивающий ресурс, ничем больше не обладал.**
- **Приводит к недостаточному использованию ресурсов; возможна ситуация “голодания” (starvation).**

Предотвращение тупиков (продолжение)

- Отсутствие перераспределения ресурсов –
 - Если процесс, обладающий некоторыми ресурсами, запрашивает другой ресурс, который не может быть ему немедленно выделен, то все ресурсы, которыми он обладает, должны быть освобождены.
 - Перераспределяемые ресурсы добавляются к списку ресурсов, которые ожидает процесс.
 - Процесс будет перезапущен только в случае, если он может вновь получить все старые ресурсы, которыми он обладал, а также все новые ресурсы, которые он запрашивает.
- Циклическое ожидание – ввести общую нумерацию (упорядочение) всех типов ресурсов, и потребовать, чтобы процесс запрашивал ресурсы только в порядке возрастания номеров.

Как избежать тупиков

Данные методы требуют, чтобы система обладала дополнительной априорной информацией

- Самая простая и полезная модель требует, чтобы каждый процесс указывал максимальный объем ресурсов каждого типа, которые могут ему понадобиться (как в ранних ОС – паспорт задачи и т.д.).
- Алгоритм избежания тупиков динамически анализирует состояние распределения ресурсов, чтобы убедиться, что никогда не может возникнуть ситуация циклического ожидания.
- Состояние распределения ресурсов определяется как объем доступных и распределенных ресурсов, а также максимальные требования процессов.

Q & A

- **Вопросы и ответы**