14 Планирование

(Часть темы управления процессами. Перенос в курс ОСиС)

14.1 Планирование выполнения процессов (потоков)

14.1.1 Задачи планирования

– выбор момента смены активного процесса/потока
– выбор нового процесса/потока (из очереди готовых)
– переключение контекстов процессов/потоков

Фактически задачи планирования ставятся шире, чем распределение времени процессора по какому-либо критерию (например, на основании приоритета). Планируются также и другие ресурсы, в первую очередь память и устройства (каналы) ввода-вывода, при этом используются схожие в целом подходы, методы и алгоритмы.

14.1.2 Уровни планирования

1) Планировщик доступа – управление поступлением новых задач в систему.

2) Планировщик памяти – размещение и хранение задач.

3) Планировщик процессора – выполнение задач.

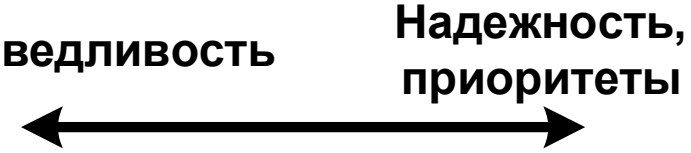
Чаще обсуждается компонент ОС, решающий задачи планирования выполнения – ***планировщик задач***, ***планировщик процессов***, ***диспетчер потоков***, ***process manager*** и т.д.
Но он взаимодействует с другими уровнями – использует их и зависит от них. Например, ***диспетчер памяти*** может выполнять запросы на выделение памяти или блокировать их в зависимости от параметров процесса – инициатора запроса (в реально многозадачной среде с высокой интенсивностью таких запросов).

14.1.3 Требования к планировщику

Общие требования – для всех систем:

– справедливость
– контролируемость
– баланс занятости, сбалансированная загрузка частей системы

Справедливость планирования – все процессы рано или поздно получают необходимые ресурсы и возможность выполняться.
Противоположность – надежность, строгость исполнения приоритетов.



Характер планирования

Противоречие – проблема ***инверсии приоритетов***.

Специфические требования:

Для систем ***пакетной*** обработки:
– пропускная способность
– время прохождения заявки («оборотное» время)
– загрузка, полнота использования ресурсов системы

Для ***интерактивных*** систем:
– время отклика
– соразмерность (соответствие) желаниям пользователя

Для систем ***реального времени***:
– конечность времени реагирования и обработки
– предсказуемость
– предотвращение деградации качества

14.1.4 Основные группы методов (алгоритмов) планирования

Деление по двум критериям: с квантованием времени или без квантования; с приоритетами или без приоритетов.

Критерии не исключают друг друга – возможны и на практике часто используются ***комбинированные*** методы

Комбинации подходов к планированию

С ***квантованием*** времени – основаны на выделении времени процессора короткими порциями – «квантами»

Переключение контекста в случаях:
– завершение активного потока (нормальное или аварийное)
– переход в состояние ожидания
– исчерпание выделенного кванта

Вариации: неравные кванты, разные дисциплины очередей, ...

С ***приоритетами*** – выбор контекста на основании приоритетов готовых к выполнению процессов (потоков)

Переключение контекста в случаях:
– завершение активного потока (нормальное или аварийное)
– переход в состояние ожидания
– появление более приоритетного процесса (потока)

Виды приоритетов: абсолютные и относительные

Относительные приоритеты – активный процесс (поток) не прерывается, переключение контекстов только по окончании кванта.

Результат – ***невытесняющая*** (***not-preemptive***) многозадачность

Абсолютные приоритеты – переключение контекста немедленно при появлении более приоритетного процесса

Результат – ***вытесняющая*** (***preemptive***) многозадачность

(Очевидно, после завершения активного процесса или перехода его в состояние ожидания переключение контекста будет независимо от границ квантов).

14.1.5 Дисциплины планирования

Основные типы

1) Циклические (Round Robin)

2) Приоритетные

3) «Кратчайший следующий» (оценка «устаревания» задач)

4) Гарантированное планирование (учет доли обязательно предоставляемого времени)

5) Лотерейные (случайные), в т.ч. с неравенством

6) «Справедливые» с учетом владельца и и др. параметров процесса

Дополнительные факторы и критерии выбора дают смешанные и модифицированные дисциплины, применяемые на практике.

Некоторые базовые дисциплины планирования

FCFS (***F***irst ***C***ome ***F***irst ***S***erve) – очередь, без вытеснения. Нет бесконечно ждущих процессов (если активный не зависнет).

RR (***R***ound ***R***obin), карусельная – циклическое планирование с квантованием. Идеально «справедливое» планирование.

SRR (***Selfish RR***), «эгоистичная» карусель – вытесняющее циклическое планирование с преимуществом текущего активного процесса.

SJN (***S***hortest ***J***ob ***N***ext) – невытесняющая, преимущество «коротких» («быстрых») задач. Минимизация штрафов за потери времени, максимизация пропускной способности для «коротких» задач.

PSJN (***P***reemptive ***SJN***) – то же, но вытесняющая.

HPRN (***H***ighest ***P***enalty ***R***atio ***N***ext) – невытесняющая с динамическими приоритетами, улучшена «справедливость»

FB (***F***oreground – ***B***ackground) – очереди готовых процессов «переднего» и «заднего» плана (приоритет на уровне очередей)

MLFB (***M***ulti-***L***evel ***FB***) – то же с большим числом очередей, эффективность для интерактивных процессов.

14.2 Тупики и борьба с тупиками

Тупик (***deadlock***) – ситуация, когда несколько взаимодействующих процессов (потоков) блокируют друг друга, обычно при конфликте за ресурсы: каждый из участников владеет частью необходимых ресурсов и ожидает получения остальных, которые заняты другими. Никто не может продолжить работу, соответственно никто не может освободить занятые ресурсы.

В терминах критических секций – ситуация вложенности секций: каждый из конкурирующих процессов, уже находясь в критическую секции, ожидает возможности войти в другую, которая в этот момент занята (нарушение базового требования к критическим секциям).

Обычно принимается, что процессы самостоятельно выйти из тупика не могут, необходимы дополнительные меры и средства.

14.2.1 Уровни (задачи) борьбы с тупиками

1) Предупреждение тупика – выбор стратегии распределения ресурсов

2) Обнаружение тупика – распознавание ситуации тупика

3) Развязка тупика – разблокирование участников

Решения – компромисс между надежностью и ограничением возможностей и производительности.

«Консервативные» и «либеральные» методы решения

14.2.2 Предупреждение тупиков

Стратегия распределения ресурсов (в т.ч., возможно, и процессорного времени) должна предотвращать возникновение тупика, при необходимости задерживая выполнение критических операций.

Стратегии – без дополнительной информации о процессах и с дополнительной информацией.

Без дополнительной информации

1) ***Последовательное*** выделение – ресурсом (или набором ресурсов) может владеть только один процесс.
Отдельные ресурсы из набора могут предоставляться поочередно, но весь набор будет считаться занятым и недоступным для других пользователей.

2) «***Залповое***» выделение – весь набор ресурсов выделяется и освобождается только целиком, даже если запрошена его часть. Необходима поддержка атомарного запроса разнородных ресурсов.

3) ***Иерархическое*** выделение – иерархия классов ресурсов, где более дефицитный ресурс относится к более высокому уровню. Запросить можно только ресурс с более высокого уровня, чем уже имеющиеся.
Проблема – построение адекватной иерархии.

С дополнительной информацией

Различные варианты «***алгоритма банкира***» на основании предварительных ***заявок***.

Реализуемая ситуация – сумма запрошенных ресурсов (в заявках) не превышает наличие ресурсов в системе, при этом никто не превышает свою заявку.

Безопасная ситуация – каждый процесс может нормально завершиться, использовав весь объем своей заявки, если до него завершится и освободит ресурсы предыдущий процесс.

Опасная ситуация – возможна нехватка ресурсов для очередного процесса. При получении такого запроса на выделение ресурсов может перейти в тупик.

Стратегия состоит в удовлетворении запросов до тех пор, пока они не делают ситуацию опасной. Отказ в удовлетворении заявки в безопасной ситуации к тупику не приводит.

Также другие близкие алгоритмы, например алгоритм ***Габермана*** (адаптирован для работы с одним классом ресурсов).

Алгоритмы с предварительными заявками – наиболее либеральные из гарантирующих избегание тупика. Производительность снижается из-за того, что не допускаются также и опасные ситуации, которые не обязательно стали бы тупиковыми.

14.2.3 Обнаружение тупиков

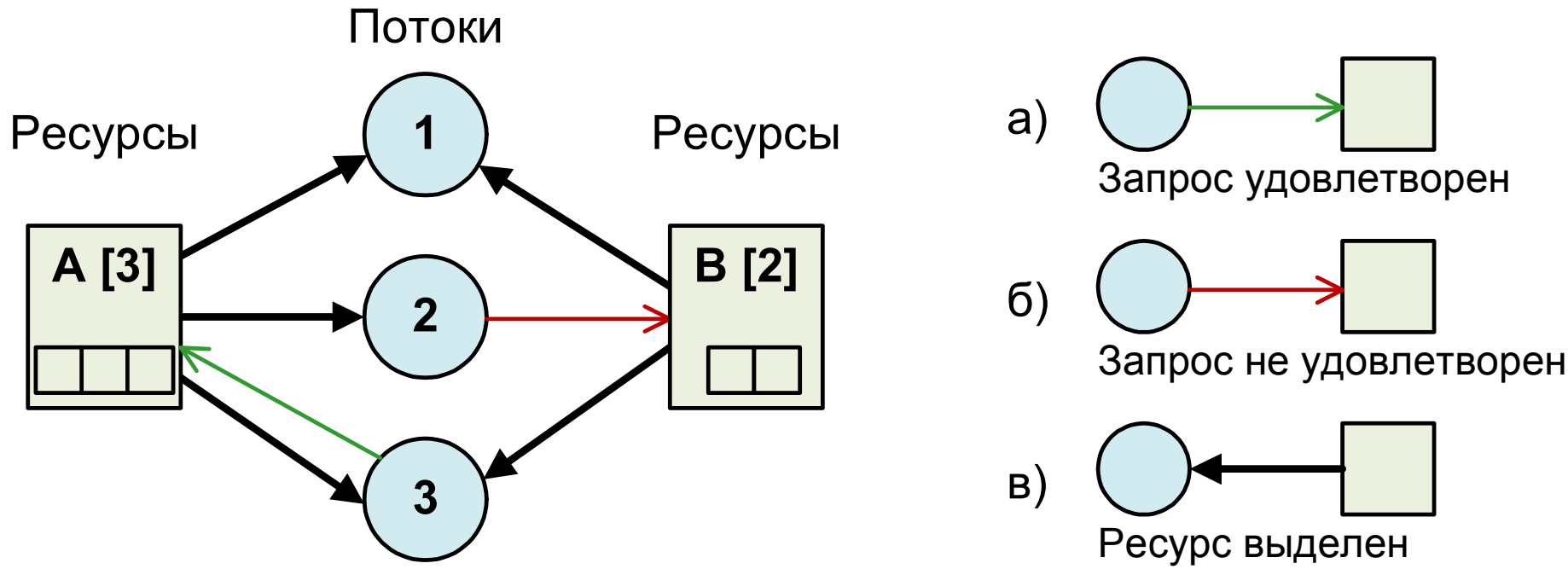
Политика точного учета

Возможные ситуации:

– ***Не тупиковая*** – можно выстроить процессы так, что первый из них сможет выполниться и завершиться, удовлетворяя все запросы, а каждый следующий может выполниться и завершиться после завершения предыдущего.
(Аналогично определению ***безопасной*** ситуации)

– ***Тупиковая*** – выстроить такую последовательность невозможно, есть взаимно блокирующиеся процессы.

Ситуацию можно представить графом.
Граф не обязательно связный.



Граф распределения ресурсов

Граф – не статический, он отражает мгновенное состояние в системе и непрерывно перестраивается.

Преимущества: точное, достоверное, проверяемое решение; возможность оптимально загрузить ресурсы без излишнего резервирования.

Недостатки: вычислительно сложная задача, значительные затраты на непрерывный анализ и поиск тупиков.

Оптимизация: проверка выполняется только при получении очередного запроса на ресурс, т.к. ситуация может стать тупиковой только в этот момент.
Выигрыш – общие вычислительные затраты
Проигрыш – более длительное выполнение запросов

«Ленивая» политика

Игнорирование проблемы в расчете на самоорганизацию и малую вероятность действительно неразрешимого тупика («страусовый» алгоритм).

Не предсказание и предупреждение тупика, а обнаружение фактически сложившегося тупика и его ликвидация.

Недостатки: не гарантирует решение проблемы тупиков. Неприемлемо при высоких требованиях к надежности.

Преимущества: простота и экономичность решения. Типично для универсальных и «настольных» ОС.

Основная мера для повышения надежности – ограничение времени ожидания отдельно взятых потоков, выявление зависших потоков.
Технические решения: использование ***тайм-аутов***, «сторожевых» таймеров (***WDT***) и т.п.

14.2.4 Развязка тупиков

Так или иначе, прерывание текущего хода выполнения конфликтующих процессов

Принудительное ***высвобождение*** (***выгрузка***) ***ресурса*** – если возможно (проблема целостности, некорректного состояния ресурса и/или процесса)

Откат, ***восстановление*** или ***перезапуск*** – например, механизм ***транзакций*** в БД

Уничтожение процесса – проблема выбора наименее ценного (минимизация потерь). Обычно процесс, наименее использующий наиболее дефицитные ресурсы, либо «зависший». Методы выбора удаляемого процесса – в общем аналогичны используемым при выгрузке-загрузке страниц памяти.

14.2.5 Устранение голодания

«***Голодание***» (***starvation***) – систематическое недополучение ресурсов с сохранением потенциальной возможности выполняться (без взаимной блокировки).
В целом менее критичная проблема.

– Строгая ***очередь*** без приоритетов – справедливое выделение ресурсов, но риск тупика

– ***Приоритеты*** – более правильная дисциплина выделения ресурсов, но риск «голодания»

– ***Динамическая*** перестройка очереди (динамические приоритеты) – риск инверсии приоритетов