Планирование процессов

- В многозадачных операционных системах возможны ситуации, когда более одного процесса находятся в состоянии готовности и хотят получить доступ к процессору
- Планировщик — часть операционной системы, которая реализует алгоритм планирования
- Существуют различные типы поведения процессов:
  - ограничены вычислительными возможностями ЭВМ
  - ограничены возможностями ввода/вывода

Когда планировать?

- При завершении процесса (обязательно)
- При блокировании на ввод/вывод или примитивами синхронизации с «засыпанием» (обязательно)
- При создании нового процесса
- При прерывании ввода/вывода
- При прерывании от таймера
  - **невытесняющий** алгоритм планирования: процесс выполняется до блокирования или завершения
  - **вытесняющий** алгоритм планирования: процесс выполняется в течение некоторого времени, затем принудительно переключается

Окружение и категории алгоритмов планирования

Среды (окружение), в которых происходит планирование:

- **Системы пакетной обработки**: нет пользователей, нужна производительность. Подходят невытесняющие алгоритмы или вытесняющие с длительным промежутком исполнения
- **Интерактивные системы**: необходимо вытеснение
- **Системы реального времени**: вытеснение необязательно, поскольку процессы короткие

Цели алгоритмов планирования

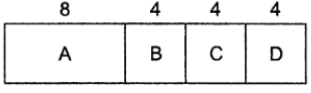
- Цели зависят от среды (окружения), но выделяются и общие
- Все системы:
  - равноправие: предоставление каждому процессу справедливой доли процессорного времени
  - применение политик: наблюдение за соблюдением установленной политики (пример политики: процессы безопасности могут быть запущены в любой момент, даже если нужно задержать другие процессы)
  - баланс: обеспечение работой всех компонентов системы
- Системы пакетной обработки:
  - пропускная способность: выполнение максимального количества заданий в единицу времени
  - время оборота: минимизация времени, затрачиваемого на ожидание обслуживания и обработку задания
  - коэффициент использования процессора: обеспечение постоянной занятости процессора
- Интерактивные системы:
  - время отклика: быстрая реакция на запросы
  - пропорциональность: соответствие ожиданиям пользователя (пользователи имеют распространенное представление, сколько выполняется та или иная задача)
- Системы реального времени:
  - соответствие временным ограничениям: во избежание потери данных, поступающих непрерывно
  - предсказуемость: избежание потери качества в мультимедиа системах (возможна потеря небольшого объема данных, но время исполнения процесса должно быть предсказуемо в целом)

Планирование в системах пакетной обработки

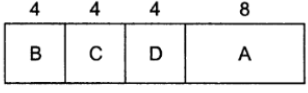
- Первым пришел – первым обслужен
  - Процессы обслуживаются в порядке поступления запросов на использование процессора.
  - Новые процессы помещаются в конец очереди
  - Заблокированные процессы помещаются в конец очереди
  - «+»: простота и легкость
  - «−»: неоптимальная загрузка ресурсов (вычислительный процесс, изредка обращающийся к вводу/выводу и несколько процессов со многими операциями ввода/вывода; последние будут ждать первого)

- Самое короткое задание – первое
  - Предположение: время выполнения процессов известно заранее (на рисунке A,B,C,D — задания, числа — время исполнения)
  - «+»: минимальное время оборота
  - «−»: должны быть в наличии сразу все задания

- Задание с наименьшим временем выполнения — следующее
  - Первым выбирается задание с наименьшим оставшимся временем исполнения
  - Время выполнения заданий должно быть известно
  - Если вновь появившееся задание можно выполнить быстрее — текущее приостанавливается и запускается новое



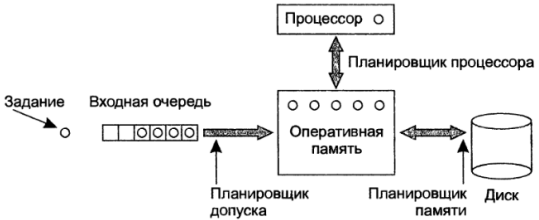
а



б

Трехуровневое планирование

- Входная очередь хранится на жестком диске
- Планирование допуска:
  - сочетание заданий для процессора и ввода/вывода
  - короткие задания сначала
- Планировщик памяти работает, когда не хватает оперативной памяти для всех процессов; перемещение заданий в/из памяти - долгий процесс
- Планировщик процессора использует подходящий алгоритм

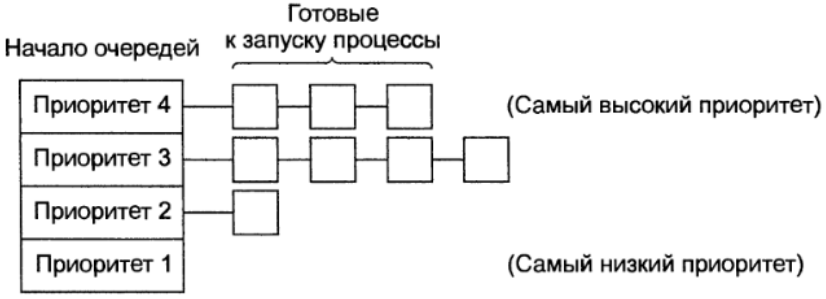


Планирование в интерактивных системах

- Только двухуровневое планирование
- Циклическое (карусельное) планирование
  - каждому процессу предоставляется квант времени
  - по истечении кванта времени происходит переключение на следующий процесс, а текущий отправляется в конец очереди
  - прост в реализации (нужно поддерживать список готовых к исполнению процессов)
  - важно выбрать размер кванта времени
- Приоритетное планирование
  - каждому процессу присваивается приоритет и управление передается процессу с наибольшим приоритетом
  - приоритет остальных процессов увеличивается с каждым тактом (или переключением между другими процессами), после выполнения приоритет восстанавливается до первоначального значения
  - приоритет может задаваться статически, может изменяться динамически
  - удобно считать приоритет в обратном порядке, т.е. минимальное числовое значение означает максимальный приоритет

Планирование в интерактивных системах

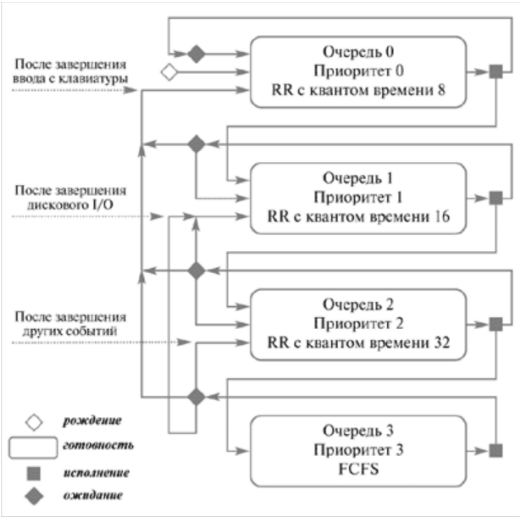
- Многоуровневое планирование:
  - процессы группируются в классы по приоритетам
  - между классами применяется приоритетное планирование, внутри класса — циклическое



Планирование в интерактивных системах

Многоуровневые очереди с обратной связью:

- возможна миграция процессов между очередями приоритетов (за счет механизма обратной связи)
- переход из очереди 0 в очередь 1 и далее осуществляется, если процесс полностью израсходовал свой квант времени
- FCFS — «первый пришел — первый обслужен»



Планирование в интерактивных системах

Гарантированное планирование

- N интерактивно работающих пользователя
- Гарантия: каждый из пользователей получит $1/N$ часть процессорного времени
- T_i — время нахождения пользователя в системе (время с начала сеанса), t_i — полученное время процессора
- Коэффициент справедливость для каждого пользователя: $t_i \cdot N / T_i$
- Квант времени получает пользователь (один из процессов пользователя) с минимальным коэффициентом справедливости

Планирование в интерактивных системах

Лотерейное планирование

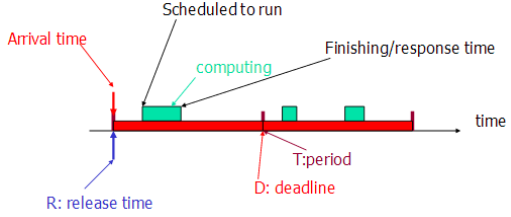
- Процессам раздаются «лотерейные билеты»
- В момент принятия решения выбирается случайный билет и управление передается процессу, у которого этот билет находится
- Для реализации приоритетов билеты могут выдаваться неравномерно, в разных объемах
- Возможна передача билетов между процессами
- Легко реализуется точная балансировка нагрузки и обеспечение качества обслуживания
- Прототип планировщика реализован для микроядра Mach 3.0

Планирование в системах реального времени

- Свойства систем реального времени
  - Жесткие системы реального времени (обязательно укладываться в сроки выполнения задания)
  - Мягкие системы реального времени
- Короткие, предсказуемые процессы, которые обрабатывают (реагируют) на внешние события
- Внешние события
  - периодические
  - непериодические

Планирование в системах реального времени

- C (computing) — время, необходимое для обработки события
- R — время впуска задания (совпадает с временем поступления события для упрощения моделей, пренебрегаем переключением)
- T — период поступления события
- D — время, до истечения которого должна закончится обработка события (совпадает с T для упрощения модели)
- C/T — утилизация (загрузка) ЦПУ задачей
- $U = \sum C_i/T_i$ – утилизация ЦПУ всеми задачами
- Если $U > 1$ – перегрузка процессора
- Если $U \leq 1$ – все задания будут закончены к сроку



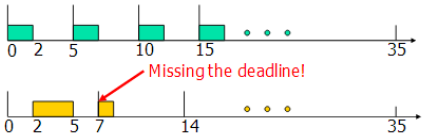
Планирование в системах реального времени: RMS

- RMS (Rate-monotonic Scheduling) — планирование с приоритетом, пропорциональным частоте
  - каждый периодический процесс должен быть завершен за время его периода
  - ни один процесс не должен зависеть от любого другого
  - каждому процессу требуется одинаковое процессорное время на каждом интервале
  - у непериодических процессов нет жестких сроков
  - прерывание происходит без накладных расходов (для упрощения модели системы)

- Используется для процессов, удовлетворяющих условиям:
- Каждому процессу назначается приоритет, равный частоте возникновения событий процесса (чем меньше задача – тем выше приоритет). Всегда запускается процесс с наивысшим приоритетом, прерывая по необходимости работающий процесс

Планирование в системах реального времени: RMS

- Не всегда работает, даже если $U < 1$, например:
  - $T_1(2,5); T_2(4,7)$
  - $U = 2/5 + 4/7 = 34/35$
  - $Pr(T_1) = 1, Pr(T_2) = 2$



- Достаточное условие планирования: $U \leq n * (2^{1/n} - 1)$
- «+»
  - легкий в реализации
  - стабильный (низкоприоритетные задания могут не успевать обрабатываться, но высокоприоритетные будут работать)
- «—»
  - низкая загрузка ЦПУ
  - требование $D = T$
  - работа только с независимыми заданиями
- Все «—» можно решить, кроме низкой загрузки

Планирование в системах реального времени: EDF

- EDF (Earliest Deadline First) – вначале процессы с самым коротким временем окончания
- Когда поступает новая задача, происходит сортировка очереди задач таким образом, что ближайшая к своему завершению получает наивысший приоритет
- Текущая задача продолжает выполнятся, если только новая не получает наивысший приоритет
- Если любой другой алгоритм сможет распланировать задачи, то и EDF сможет
- $U \leq 1$ – необходимое и достаточное условие планирование
- «+»
  - простой в теории и реализации
  - простая проверка на возможность планирования
  - оптимальный
  - наилучшая загрузка ЦПУ
- «—»
  - сложно реализовать на практике, нужно быстро сортировать очередь
  - нестабильный, нельзя предсказать, какие задачи могут быть невыполнены

Планирование в ОС Linux

- Три типа процессов:
  - Интерактивные: постоянно взаимодействующие с пользователем, при поступлении команды (прерывания) от пользователя должны быть быстро «разбужены».
  - Пакетные процессы: работают в фоновом режиме и не нуждаются во взаимодействии с пользователем
  - Процессы реального времени: жесткие требования к планированию, нельзя блокировать ради процессов с низким приоритетом
- Планировщик Linux использует эвристический алгоритм для определения интерактивного или пакетного процесса, предпочитает интерактивные процессы
- Реализованы три класса планирования:
  - SCHED_FIFO: процессы, работающие в реальном времени, планируются по принципу «первым вошел—первым обслужен»
  - SCHED_RR: процессы, работающие в реальном времени, планируются по круговому принципу
  - SCHED_NORMAL: обычные процессы

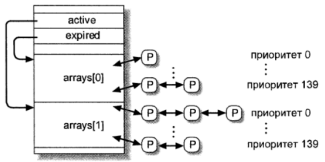
Планирование в ОС Linux: SCHED_NORMAL

- Каждый процесс имеет свой статический приоритет (от 100, наивысший, до 139)
- Статический приоритет определяет базовый квант времени
- Каждый процесс имеет динамический приоритет (от 100 до 139) = max(100, min(статический-бонус+5, 139))
- Бонус - число от 0 до 10, зависит от среднего времени сна процесса (предыдущая история процесса)
- Среднее время сна — среднее количество наносекунд, проведенных процессом в ожидании (среднее, поскольку в разных режимах ожидания время считается неравномерно), не более 1 секунды, уменьшается пока процесс работает
- Существует таблица соответствия между среднем временем сна и бонусом
- Среднее время сна учитывается при отнесении процесса к интерактивным или пакетным

$$\text{базовый квант времени (в мс)} = \begin{cases} (140 - \text{статический приоритет}) \times 20, \\ \text{если статический приоритет} < 120 \\ (140 - \text{статический приоритет}) \times 5, \\ \text{если статический приоритет} \geq 120 \end{cases}$$

Планирование в ОС Linux: SCHED_NORMAL

- для предотвращения голодания процесс, исчерпавший квант времени, замещается низкоприоритетным процессом, чей квант времени не истек; для реализации этого механизма поддерживается два непересекающихся набора выполняющихся процессов:
  - **активные**: не исчерпали своего кванта времени, им разрешено работать
  - **с истекшим квантом**: процессам запрещено работать, пока у всех активных не закончатся кванты времени
- повышение интерактивных процессов:
  - активный пакетный процесс, исчерпавший квант, обязательно переходит в список процессов с истекшим квантом
  - активный интерактивный процесс, исчерпавший квант, обычно остается активным, планировщик выделяет ему новый квант
  - активный интерактивный процесс, исчерпавший квант, переносится в список процессов с истекшим квантом, если более «старый» процесс с истекшим квантом ждет достаточно долго или если процесс с истекшим квантом имеет более высокий приоритет



Планирование в Linux: процессы реального времени

- Каждый процесс реального времени имеет свой приоритет реального времени, от 1 до 99
- Планировщик выбирает процессы с высоким приоритетом
- Процессы реального времени всегда считаются активными
- Процессы с одинаковым приоритетом обслуживаются в круговой очереди
- Процесс реального времени замещается:
  - При появлении процесса с высоким приоритетом
  - При блокировки, остановке, уничтожении, добровольном освобождении
  - Процесс работает в реальном времени по дисциплине SCHED_RR и исчерпал свой квант времени
- Продолжительность базового кванта времени у процессов реального времени, работающих по принципу SCHED_RR, зависит от статического приоритета а не от приоритета реального времени

Планирование в ОС Windows

- Выделены две группы приоритетов: реального времени и переменные
- Вытесняющий планировщик с учетом приоритетов
- Если появляется поток с более высоким приоритетом, текущий поток вытесняется
- В классе потоков реального времени все потоки имеют фиксированный приоритет (от 16 до 31); потоки с одинаковым приоритетом располагаются в круговой очереди
- В классе переменных приоритетов
  - Поток начинает работать с базовым приоритетом (1-15)
  - Приоритет увеличивается
    - когда завершается операция ввода/вывода на связанном с ней потоке, слагаемое зависит от устройства (1 – для диска, 2 – для последовательной линии, 6 – для клавиатуры и 8 – для звуковой карты)
    - при разблокировании на семафоре (+2, +1)
    - при пробуждении графического потока при доступном оконном вводе
  - Приоритет уменьшается, когда поток исчерпал свой квант (-1, но до базового уровня)
  - Если окно становится окном переднего плана, все его потоки получают более длительный квант времени
  - Потоки с одинаковым приоритетом располагаются в круговой очереди

Планирование в ОС Windows