Красавин Евгений Васильевич доцент кафедры ИУК4

Цвет учебника – виртуальный

06.09

Введение в операционные системы

ОС обеспечивает удобство посредством представления расширенной машины и рациональное управление ресурсами компьютера

Управление ресурсами включает планирование ресурса и отслеживание состояний

Эволюция ОС

1945 – 1955 – использовались реле и электронные лампы

1955 – 1965 – появление полупроводниковых элементов, произошло разделение персонала на программистов и операторов, эксплуатационников и разработчиков вычислительных машин, появились первые алгоритмические языки и компиляторы, первые системы пакетной обработки, которые просто автоматизировали запуск одной программы за другой, по итогам пакетной обработки был разработан язык управления заданиями, с помощью которой программист сообщал системе и оператору, какую работу он хочет выполнить, совокупность нескольких заданий получило название пакетозаданий, компьютеры использовались для научных и технических вычислений, появились языки программирования Fortran и Assembler и первые ОС

1965 – 1980 – переход к интегральным микросхемам, создание семейств программно совместимых машин, а программная совместимость требовала программной совместимости операционных систем, важнейшим достижением являлась реализация мультипрограммирования (одновременного выполнения нескольких программ), другое нововведение – спулинг – способ организации вычислительного процесса, в соответствии с которым задания считывались с перфокарт на диск в том темпе, в каком они появлялись в вычислительном центре, а затем после завершения очередного задания новое загружалось в освободившийся раздел, появился новый тип ОС с разделением времени

1980 – наше время – появление больших интегральных схем, персональных компьютеров, доминирование на рынке двух ОС – однопрограммной однопользовательской MS DOS и мультипрограммной мультипользовательской UNIX, – с развитием сетей бурно развивались сетевые ОС

13.09

Классификация ОС

По размеру:

* ОС мейнфреймов – в основном ориентированы на обработку множества одновременных заданий, большинству из которых требуется огромное количество операций ввода-вывода. Предлагают 3 вида обслуживания:
  + Пакетная обработка
  + Обработка транзакций
  + Разделение во времени
* Серверные ОС – работают на серверах, рабочих станциях и некоторых мейнфреймах. Одновременно обслуживают множество пользователей и позволяют делить им между собой программные и аппаратные ресурсы
* Многопроцессорные ОС
  + Мультипроцессорные ОС
    - Ассиметричные – ОС целиком выполняется только на одном процессоре, распределяя прикладные задачи по остальным
    - Симметричные – ОС полностью децентрализована, используется весь пул процессоров, разделяемых между системными и прикладными
  + Многомашинные системы и кластеры
  + Глобальные распределенные системы
* ОС для ПК – предоставляют удобный интерфейс для одного пользователя
* ОС реального времени – главным параметром является время. Существует гибкая система реального времени, в которой допустимы редкие пропуски сроков выполнения операции
* Встроенные ОС
* ОС для смарт-карт

В зависимости от выбранной архитектуры организации различают:

* Монолитные – параметры перемещаются в строго определенные места – регистры или стек, – после чего выполняется команда прерывания – вызов ядра, – которая переключает машину из режима пользователя в режим ядра и передает управление операционной системе. Структура такой ОС:
  + Главная программа, которая вызывает служебную процедуру
  + Набор служебных процедур, выполняющих системные вызовы
  + Набор утилит, обслуживающих служебные процедуры
* Многоуровневые системы
* Виртуальные машины. Экзо-ядро
* Модель клиент-сервер. Микроядро

20.09

Управление процессами

Процесс (задача) – абстракция, описывающая выполняющуюся программу

Подсистема управления процессами планирует выполнение процессов, распределяет процессорное время между несколькими одновременно существующими в системе процессами, а также занимается создание и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами и поддерживает взаимодействие между процессами

Реальный процессор переключается с процесса на процесс, но рассматривается набор процессов, идущих параллельно, то есть процессор переключается от программы к программе, и это переключение называется многозадачностью.

Состояние процессов

В многозадачной многопроцессорной системе процесс может находиться в одном из трех основных состояний:

* Выполнение – активное состояние процесса, во время которого 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 некого события
* Готовность – пассивное состояние процесса, но в этом случае процесс заблокирован в связи с внешним по отношению к нему обстоятельством, то есть процесс имеет все требуемые для него ресурсы, он готов выполняться, однако процессор занят выполнением другого процесса

В состоянии выполнения в однопроцессорной системе может находиться только один процесс, чего не скажешь о процессах готовности и ожидания. Жизненный цикл процесса начинается с состояния готовности. При активизации процесс переходит в состояние выполнения и находится в нем до тех пор, пока либо сам не освободит процессор, перейдя в состояние ожидания, либо будет насильно вытеснен, возвращаясь в состояние готовности. В состояние готовности процесс переходит из состояния ожидания, когда это событие произойдет.

Создание и завершение процесса

Для того, чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды. Состояние операционной среды отображается состоянием регистров, программного счетчика, режимом работы процессора, указателем на открытые файлы, информацией о незавершенных операциях ввода-вывода и др. Эта информация называется контекстом процесса. Операционной системе для реализации планирования процессов требуется дополнительная информация: идентификатор процесса, состояние процесса, данные о степени привилегированности процесса и др. Такую информацию называют дескриптором процесса. Очереди процессов представляют собой дескрипторы отдельных процессов, объединенных в списки.

Создать процесс – значит:

* Создать информационный структуры, описывающие данный процесс
* Включить дескриптор нового процесса в очередь процессов
* Загрузить кодовый сегмент процесса в оперативную память или область свопинга

Фоновые процессы называются демонами. Процессы могут создаваться не только в момент загрузки, но и по просьбе текущего процесса. Процесс завершается вследствие следующих событий:

* Обычный выход, преднамеренно
* Выход по ошибке, преднамеренно
* Выход по неисправимой ошибке, непреднамеренно
* Уничтожение другим процессом, непреднамеренно

Алгоритмы планирования процессов

Планирование включает:

* Определение момента времени для смены выполняемого процесса
* Выбор процесса для выполнения из очереди готовых процессов
* Переключение контекста старого и нового процессов

Алгоритмы планирования можно разделить на 2 группы:

* Основанные на квантовании
* Основанные на приоритетах

При квантовании смена активного процесса происходит, если:

* Процесс завершился и покинул систему
* Произошла ошибка
* Процесс перешел в состояние ожидания
* Исчерпан квант процессорного времени, отведенный данному процессу

Кванты, выделяемые процессором, могут быть одинаковыми или изменяться в разные периоды жизни. Очередь готовых процессов может быть организована:

* Циклически по правилу FIFO
* Циклически по правилу LIFO

Приоритет – число, характеризующее степень привилегированности процесса при использовании ресурсов вычислительной машины. Чем выше приоритет – тем выше привилегии. Приоритет может назначаться директивно администратором системы либо операционной системой по определенному правилу. Приоритет может оставаться фиксированным на протяжении жизни процесса либо изменяться во времени в соответствии с некоторым законом и быть динамическим. Существует 2 разновидности приоритетных алгоритмов:

* Алгоритмы, использующие относительные приоритеты
* Алгоритмы, использующие абсолютные приоритеты

В системах с относительным приоритетом активный процесс выполняется до тех пора, пока сам не покинет процессор, переходя в состояние ожидания. В системах с абсолютным приоритетом выполнения активного процесса прерывается при еще одном условии – если в очереди готовых процессов появился процесс с более высоким приоритетом.

Вытесняющий и невытесняющие алгоритмы планирования

Существует два типа процедур планирования: вытесняющие и невытесняющие. Невытесняющая многозадачность – это способ планирования процессов, при котором активный процесс выполняется до тех пор, пока сам не отдаст управление планировщику ОС. Вытесняющая – способ, при котором решение о переключении процессора с выполнения одного процесса на выполнение другого принимается планировщиком ОС.

При невытесняющей многозадачности механизм планирования распределен между системой и прикладными программами. Преимущество невытесняющих систем – более высокая скорость переключения с задачи на задачу. Почти все системы написаны на вытесняющей многозадачности.

27.09

Планирование в системах реального времени

Системы реального времени делятся на жесткие системы реального времени, что означает наличие жестких сроков для каждой задачи, и гибкие системы реального времени, в которых нарушение временного графика нежелательно, но допустимо.

Системы реального времени, удовлетворяющие этому условию, называются поддающимися планированию

Событие с номером i поступает с периодом Pi и на его обработку уходит ci времени

Алгоритмы для систем реального времени могут статическими и динамическими

Статические – решения о планировании принимаются до запуска система

Динамические – решения о планировании принимаются по ходу

Статические алгоритмы заранее назначают каждому процессу фиксированный приоритет, после чего выполняют приоритетное планирование с переключением. У динамических алгоритмов нет фиксированных приоритетов.

Работа алгоритмов планирования RMS и EDF

Алгоритм планирования – планирование с приоритетом пропорциональном частоте

Алгоритм используется при следующих условиях:

1. Каждый периодический процесс должен быть завершен за время его периода
2. Ни один процесс не должен зависеть от любого другого процесса
3. Каждому процессу требуется одинаковое процессорное время на каждом интервале
4. У непериодических процессов нет жестких сроков
5. Прерывание процесса происходит мгновенно без накладных расходов

Алгоритм RMS работает, назначая каждому процессу фиксированный приоритет, равный частоте возникновения процесса

Алгоритм EDF – процесс с ближайшим сроком завершения выполняется в первую очередь

Средства синхронизации и взаимодействия процессов

Критическая секция

Критическая секция – часть программы, в которой осуществляется доступ к разделяемым данным. Чтобы исключить эффект «гонок» в отношении к некотором ресурсу, необходимо обеспечить, чтобы в каждый момент в критической секции, связанной с этим ресурсом, находился максимум один процесс. Этот прием называют взаимным исключением.

Простейший способ взаимного исключение – запрещение прерывания. Другой способ – использование блокирующих переменных. Операция проверки и установки блокирующей переменной должна быть неделимой, желательно иметь единую команду «проверка-установка» либо реализовывать программные примитивы, запрещающие прерывания на протяжении всей операции «проверки-установки».

Обобщающее средство синхронизации процессов вводит два примитива, обозначаемые P и V, оперирует целыми неотрицательными переменными, называемыми семафорами. В частном случае семафор может принимать значение только 0 или 1. Он превращается в блокируемую переменную, называемую Мьютекс.

Тупики

Существует проблема синхронизации – взаимные блокировки – называемые дедлоками, клинчами или тупиками. Проблема тупиков включает предотвращение тупиков, распознавание тупиков и восстановление системы после тупиков. Для написания корректных программ используется высокоуровневое средство синхронизации, называемое монитором. Монитор – набор процедур, переменных и структур данных. Имеет свойство: только один процесс может быть активным по отношению к монитору. Существует несколько ЯП, поддерживающих мониторы.

Потоки

Современные ОС предлагают использовать новый механизм многопоточной обработки – мультитрединг. Потоки имею собственные программный счетчик, стек, регистры состояния, разделяют адресное пространство, глобальные переменные, открытые файлы, таймеры, семафоры, статистическую информацию, дочерние процессы. Поток реализуются в пространстве пользователя или ядре.

04.10

Управление памятью

Распределению подлежит вся память, не занятая системой. ОС обычно располагается в самых младших адресах. Функциями ОС по управлению памятью является:

* Отслеживание свободной и занятой памяти
* Выделение памяти процессам и освобождение их при завершении
* Вытеснение процессов из оперативной памяти на диск и возвращение их в оперативную память
* Настройка адресов программы на конкретную область физической памяти

Типы адресов

* Символьные имена
* Виртуальные адреса
* Физические адреса



Совокупность адресов процесса называется виртуальным адресным пространством

Физические адреса соответствуют номерам ячеек оперативной памяти, где в действительности расположены или будут расположены переменные и команды

Переход от виртуальных адресов к физическим осуществляется двумя способами:

1. Специальная системная программа, перемещающая загрузчик. Он на основании исходных у него данных о начальном адресе физических данных и информации транслятора выполняет загрузку программы с заменой адресов с виртуальных на физические
2. Динамическое преобразование аппаратными средствами заключается в том, что программа загружается в неизменном виде в виртуальных адресах. ОС фиксирует смещение действительного расположение программного кода относительно виртуального адресного пространства, а во время выполнения программы при каждом обращении к виртуальной памяти выполняется преобразование виртуального адреса в физический. Отображением виртуальных адресов на физические занимается диспетчер памяти, являющийся частью процессора.

Второй способ более гибкий, так как допускает перемещение программ во время их выполнения.

Методы распределение памяти без использования дискового пространства

1. Распределение памяти фиксированными разделами. Самый простой способ управления оперативной памятью – разделение на несколько разделов фиксированной величины (вручную оператором). Очередная задача помещается либо в общую очередь, либо в очередь к какому-то разделу. Подсистема управления памятью, сравнивая размер программы и разделов, выбирает подходящий раздел, выполняет настройку программы и адресов. Достоинства: простота. Недостатки: неэффективное использование памяти, уровень мультипрограммирования ограничен числом разделов.
2. Разделение памяти разделами переменной величины. Сначала память свободна, каждой поступающей задаче выделяется необходимая ей память. После завершения задачи память освобождается и на это место может быть загружена другая задача. Задачами ОС при этом являются:
   1. ведение таблиц свободных и занятых областей, в которых указываются начальные адреса и размеры участков памяти;
   2. при поступлении новой задачи – анализ запроса, просмотр таблицы свободных областей и выбор раздела, размер которого достаточен для поступившей задачи;
   3. загрузка задачи в выделенный ей раздел, корректировка таблиц свободных и занятый областей;
   4. после завершения задачи – корректировка таблиц свободных и занятых областей

Недостаток: фрагментация памяти.

1. Перемещаемые разделы. Для борьбы с фрагментации происходит перемещение участков либо в сторону старших, либо в сторону младших адресов. Поэтому у ОС добавляется еще одна задача – периодически сжимать память. Процесс дефрагментации называется сжатием. Если процесс может иметь 2 увеличивающихся сегмента: сегмента данных и стека. Предлагается предоставлять вверху области для стека, который расширяется вниз, и сегмента данных, который расширяется вверх.

Методы распределение памяти с использованием дискового пространства

Виртуальная память – совокупность программно-аппаратных средств, позволяющих пользователям писать программы, размер которых превосходит имеющуюся оперативную память. Виртуальная память решает задачи:

* Размещает данные в запоминающих устройствах разного типа
* Перемещает при необходимости между запоминающими устройствами разного типа
* Преобразует виртуальные адреса в физические

Эти действия выполняются автоматически без участия программиста.

Страничное разделение памяти

11.10

Виртуальное адресное пространство каждого процесса делится на части одинакового фиксированного для данной системы размера, называемого виртуальными страницами. Размер виртуального адресного пространства не кратен размеру страницу, поэтому последняя страница дополняется фиктивной областью. Оперативная память делится на части, называемые физическими страницами, размер обычно выбирается равным степени двойки.

При загрузке процесса часть его виртуальных страниц помещается в оперативную память, а остальные – на диск. При загрузке ОС создает для каждого процесса таблицу страниц, в которой устанавливается соответствие между номерами виртуальных и физических страниц для страниц, загруженных в оперативную память, или делается отметка о том, что виртуальная страница выгружена на диск. Кроме того, содержится признак модификации страницы, признак невыгружаемости, признак обращения к странице и др. При каждом обращении к памяти происходит чтение из таблицы страниц информации. Если эта страница находится в оперативной памяти, выполняется преобразование виртуального адреса в физический, если выгружена на диск – происходит страничное прерывание.

Алгоритм замещения страниц

* NRU – очень грубый – делит страницы на 4 класса в зависимости от состояния битов R и M обращения и модификации (0 – не было обращений и изменений, 1 – не было обращений, страница изменена, 2 – было обращение, страницы не изменена, 3 – произошло обращение и изменение). Для вытеснения выбирается страница с наименьшим номером.
* FIFO – может выгрузить важные страницы – отслеживает порядок загрузки страниц в память, храня их в связанном списке
* Вторая попытка – значительное усовершенствование FIFO – аналогичен FIFO, но проверяет, используется ли страница или нет
* Часы – реалистичный – модификация второй попытки, но более быстродействующий
* LRU – отличный алгоритм, но его можно осуществить целиком – страница, не использовавшаяся дольше всего, требует специального аппаратного обеспечения, поддерживающего связанный список страниц, содержащихся в памяти, где последняя использованная страница находится в начале списка, а дольше всех неиспользуемая – в конце, список должен обновляться при каждом обращении к памяти
* NFU – довольно грубое приближение алгоритма NRU – редко использовавшиеся страницы, модификация LRU, создается программный счетчик, связанный с каждой страницей в памяти и изначально равный нулю, во время каждого прерывания по таймеру исследуются страницы в памяти, бит R обращения прибавляется к счетчику
* Старение – эффективный алгоритм, хорошо аппроксимирующий алгоритм LRU
* WSClock – хороший рациональный алгоритм
* Рабочий набор – немного дорог для реализации

Два последних алгоритма используют рабочий набор, многие системы со страничной организацией пытаются отслеживать рабочий набор каждого процесса и обеспечивают его нахождения в памяти до запуска процесса. Загрузка страниц перед тем, как разрешить процессу работу, называется опережающей подкачкой страниц.

Механизм преобразования виртуального адреса в физический при страничной организации памяти

При каждом обращении к оперативной памяти аппаратными средствами выполняется:

* На основании начального адреса таблицы страниц (регистр), номеру виртуальной страницы (старшие разряды виртуального адреса) и длины записи в таблице страниц (системная константа) определяется адрес нужной записи в таблице
* Из этой записи извлекается номер физической таблицы
* К номеру физической таблицы присоединяется смещение

Простейший алгоритм распределения страничного пространства на диске заключается в поддержке специальной области подкачки – свопинга – на диске. При загрузке системы эта область является пустой, имеет единую запись с начальным адресом и размером. Когда запускается первый процесс, резервируется участок области подкачки размером с этот процесс. Новым процессам предоставляются участки области подкачки, равные по размерам их областям памяти. Когда процессы завершаются, дисковое пространство освобождается. Область подкачки управляется как список свободных участков. Вычисление адреса для записи страницы: прибавляется смещение страницы внутри виртуального адреса пространства к области начала подкачки. При запуске процесса 2 способа:

1. Копируется полный образ процесса в область подкачки и при необходимости переносится в память
2. Загружается весь процесс в память и постранично выгружается на диск, когда это требуется

Сегментное распределение

Виртуальное адресное пространство процесса делится на сегменты, размер которых определяется программистом с учетом смыслового значения хранящейся в ней информации. При загрузке процесса часть сегментов помещается в оперативную память, часть размещается в дисковой памяти. Система с сегментной организацией функционирует аналогично системе со страничной организацией.

Сегментно-страничное распределение

Виртуальное адресное пространство процесса делится на сегменты, а каждый сегмент – на виртуальные страницы, которые нумеруются в пределах сегмента. Загрузка процесса выполняется постранично, для каждого сегмента создается своя таблица страниц.

Свопинг

При свопинге весь процесс перемещается между процессором и диском целиком.

Иерархия запоминающих устройств, принцип кэширвоания

Кэш-память – это способ организации совместного функционирования двух типов запоминающий устройств. Алгоритм запроса с кэш-памятью:

1. Рассматривается содержимое кэш-памяти
2. Если данные обнаруживаются в кэш-памяти, они считываются из нее
3. Если данных нет, они копируются в кэш-память, а затем передаются в процессор

Объективные свойства:

1. Пространственная локальность – если было обращение по некоему адресу, то велика вероятность обращения к соседнему адресу
2. Временная локальность – если произошло обращение по некоему адресу, то велика вероятность обращения по тому же адресу

Для ускорения процесса преобразования виртуального адреса в физический используется аппаратное решение – таблица TLB.

18.10

Средства аппаратной поддержки управления памятью

Средства поддержки сегментации памяти

Физическое адресное пространство х86 процессора составляет для 32-раряздного процессора 4 Гб, что определяется 32-разрядной шиной адреса.

Физическая память является линейной. Виртуальный адрес, используемый в программе, представляет собой пару: номер сегмента и смещение внутри сегмента. Смещение хранится в поле команды, а номер сегмента – в одной из 6 сегментных регистров процессора, каждый из которых является 16-битным. Средства сегментации образуют верхний уровень управления памятью, а средства страничной организации – нижний уровень. Средства страничной организации могут быть как включены, так и выключены за счет установки определенного бита управляющим регистром процессора.

Количество сегментов определяется размером поля, отведенном в сегментном регистре под номер сегмента. Селектор (структура данных в сегментном регистре) предназначен для выбора дескриптора определенного сегмента из таблиц дескрипторов сегмента. Дескриптор сегмента описывает все характеристики сегмента, необходимые для проверки правильности доступа к нему и нахождении его в физическом адресном пространстве. Процессор х86 поддерживает 2 таблицы дескрипторов сегмента: глобальную и локальную. Глобальная предназначена для описания сегментов операционной системы и сегментов межзадачного взаимодействия, а локальная – бля остальных задач. Глобальная таблица одна, локальных – столько, сколько в системе выполняется задач, при этом активной в каждый момент времени может быть только одна локальная таблица.

Селектор состоит из 3 полей: 13 бит – номер сегмента в таблицах глобальной и локальной, 1 бит – указатель типа используемой таблицы дескрипторов, 2 бит – текущие права доступа задачи. Разрядность поля индекса определяет максимальное число глобальных и локальных сегментов по 8 Кб (всего 16). С учетом максимального размера сегмента 4 Гб каждая задача при чистой сегментной организации виртуальной памяти работает в виртуальном адресном пространстве 64 Тб.

Когда задаче необходимо получить доступ к ячейке физической памяти, то для выбора дескриптора виртуального сегмента используется значение селектора из соответствующего сегментного регистра процессора. Значение поля типа таблицы указывается на то, какую таблицу использовать. Для хранения этих таблиц используется оперативная память. Для того, чтобы процессор смог найти в физической памяти глобальную таблицу, ее полный физический адрес начала таблицы (32 бита), также размер (16 бит) хранится в специальном регистре процессора.

Каждый дескриптор GDT и LDT имеет размер 8 байт, поэтому максимальный размер этих страниц – 64 Кб (8к дескрипторов по 8 байт). Для извлечения нужного дескриптора из таблицы процессора складывает базовый адрес таблицы GDT со сдвинутым на 3 разряда влево (умножение на 8) значением поля индекса из сегментного регистра и получает физический линейный адрес нужного дескриптора физической памяти. Таблица GT постоянно присутствует в физической памяти, поэтому процессор извлекает по этому адресу нужный дескриптор сегмента и помещает его во внутренний программно недоступный регистр процессора.

Процессор сначала определяет правильность адреса, сравнивая смещение и размер сегмента. В случае выхода за границы сегмента происходит прерывание типа исключение. Потом процессор проверяет права доступа задачи к данному сегменту, сравнивая значения полей CPL селектора и DPL дескриптора. Самый высокий уровень – нулевой – используется ядром. В случае, когда селектор указывает на LDT, виртуальный адрес преобразуется в физический аналогичным образом, но для доступа к самой LDT добавляется дополнительный этап: в процессоре регистр указывает на размещение таблицы LDT не прямо, а косвенно. Сначала по значению селектора определяется физический адрес дескриптора из таблицы GDT, описывающего начало расположения таблицы в физической памяти, а затем с помощью селектора задачи определяется физический адрес нужного дескриптора. Далее процесс аналогичен таблице GDT.