Операционные системы. Лекции

Сурова София, БПИ192

23 января 2021

Литература:

1. В.Е. Карпов, К.А. Коньков, Основы операционных систем Курс лекций, издание 3, М.: Физматкнига, 2019
2. В.Е. Карпов, К.А. Коньков, Основы операционных систем, 1-е и 2-е издания, М.:Intuit.ru (лекции и практика)
3. Э.Таненбаум, Х.Бос, Современные операционные системы, 4-е издание, Спб.: Питер, 2015
4. Вильям Столлингс, Операционные системы, издание 4-е, М.: Вильямс, 2004
5. William Stallings, Operating Systems: Internals and Design Principles, 8th Edition, PErson Education, Inc., 2015
6. Avi Silberschatz, Peter Baer Galvin, Greg Gagne, Operating System Concepts 10th edition, Wiley, 2018

Темы:

1. Обзор. Введение (лекции 1,2)
2. Архитектурные особенности построения ОС (лекция 2)
3. Понятие процесса. Операции над процессами (лекция 3)

Тема №1. Обзор. Введение

Ссылка на лекции

https://youtu.be/1diuy4utjio, https://youtu.be/zBc4ggpDEZA

Содержание темы

- что такое операционная система,
- эволюция операционных систем,
- функции операционной системы,
- заключение.

Что такое операционная система?

1. Распорядитель ресурсов (процессорное время, оперативная память, место на жестком диске, количество устройств ввода-вывода)
   Когда происходит разделение программами, например, одного принтера, ожидается упорядоченный результат, а не хаотичный захват принтера для печати информации.

2. Защитник пользователей и программ
   Разные пользователи могут запускать много программ одновременно, и они не могут вмешиваться в работу друг друга или быть атакованы из вне.

3. Виртуальная машина
   Чтобы работать с вычислительной машиной на уровне абстракции, на низком уровне непосредственно с устройствами, пользователю необходимо иметь достаточный уровень компетенции и знаний об этих устройствах.

4. Кот в мешке
   Смутные представления пользователя, который столкнулся с операционной системой при покупке, например, нового ноутбука, где операционная система (Windows, Linux) является некоторой установленной средой.

5. Постоянно функционирующее ядро
   Программа, которая постоянно находится в оперативной памяти вычислительной машины и постоянно функционирует, обеспечивая функционирование и взаимодействие всех остальных программ.

Проще сказать, не что такое операционная система, а какие функции она выполняет. Для определения функций операционной системы рассмотрим эволюцию вычислительных систем, так как она связана с эволюцией и изменением операционных систем.

Эволюция вычислительных систем

1-й период (1945 - 1955): Научно-исследовательская работа в области вычислительной техники

- Ламповые машины
- Десятичная, а не двоичная арифметика
- Перепрограммирование осуществлялось с помощью комутаторов, перфокарт и переключателей и занимало 2-3 дня
- Нет разделения персонала
- Одновременное выполнение только одной операции (либо программирование, либо переключение тумблеров, либо определение набора перфокарт, либо работа машины, либо интерпретация результатов работы машины)
- Появление прообразов первых компиляторов
- Нет операционных систем

2-й период (1955 - начало 60х гг.): Начало использования ЭВМ в научных и коммерческих целях

- Транзисторные машины (менее габаритные, выделяют менее тепла, более надежные, менее дорогие)
- Разделение персонала (сборка, эксплуатация и программирование вычислительной системы выполняется разными коллективами)
- Развитие алгоритмических языков (Algol, Fortran и др.)
- Ввод программы колодой первокарт и вывод результата на печать
- Пакеты заданий и системы пакетной обработки (нынешние библиотеки)

3-й период (начало 60х гг. - 1980): Широкое использование ЭВМ в научных и коммерческих целях

- Машины на интегральных схемах
- Использование спулинга (spooling) (устройства ввода-вывода не успевали за скоростью вычислительных систем, поэтому входные и выходные данные считывались с магнитного диска и записывались на него)
- Планирование заданий
- Мультипрограммные пакетные системы (например, пока одна программа занимается вводом-выводом, то может быть выполнена другая программа, не прибегающая к вводу-выводу, для повышения эффективности работы вычислительной машины и сокращения времени простоя)
- Системы разделения времени (time-sharing) (чтобы не фиксировать события, после которых можно передать процессор от одной программы к другой, можно установить время (оно называется квантовым временем), в течение которого программа владеет процессором
- Виртуальная память (чтобы увеличить производительность вычислительной системы, необходимо увеличить количество программ, которые можно разместить в вычислительной системе, а для этого можно держать программу в оперативной памяти не целиком, а только необходимые команды для обработки данных)
- Интерактивная отладка программ, файловые системы
- Семейства ЭВМ (совместимость ЭВМ с предыдущими и переносимость программ)

Немного про мультипрограммирование

- Software
  - Планирование заданий (определение, какое задание будет загружено очередным)
  - Управление памятью (при размещении в памяти нескольких программ необходимо вести учёт свободной и занятой памяти)
  - Сохранение контекста (при передаче процессора от одной программы к другой необходимо фиксировать и запоминать состояние программ, чтобы программы могли выполнять работу с того момента, когда процессор был передан)
  - Планирование использования процессора (при завершении программы и освобождения процессора необходимо принять решение, какую программу из очереди выполнить следующей)
  - Системные вызовы (способы обратиться к специализированной части ПО, чтобы программы пользователя могли выполнить привилегированную команду)
  - Средства коммуникации (хоть программы друг от друга и защищены, но они могут решать одну задачу командно, и необходим обмен данными между программами)
  - Средства синхронизации (некоторые правила и очередность коммуникации программ)
- Hardware
  - Защита памяти (работающие программы не должны иметь доступ к памяти, занятой другой программой)
  - Сохранение контекста
  - Механизм прерываний (если программа занималась вводом-выводом и на это время у неё отобрали процессор, то необходимо оповестить процессор о прекращении ввода-вывода программой, чтобы процессор снова перешёл к ней)
  - Привилегированные команды (не все команды являются равноценными)

4-й период (1980 - 2005 гг.): Широкое использование ЭВМ в быту, в образовании, на производстве

- Машины на больших интегральных схемах (БИС) (на кристалле помещается процессор целиком)
- Персональные ЭВМ (уменьшились размеры компьютеров, они упали в стоимости и стали доступны обычным пользователям)
- Дружественное ПО (обычные пользователи не имеют глубокие знания об устройстве вычислительных систем, поэтому начинается развитие интерфейса между пользователем и вычислительной машиной и вместе с тем происходит некая деградация в развитии вычислительных машин)
- Сетевые и распределенные операционные системы (появление большого количества компьютеров у пользователей приводит к появлению сетей, которые позволяют объединять вычислительные системы между собой для совместного использования)

5-й период (2005 - ?? гг.): Глобальная компьютеризация

- Машины на многоядерных процессорах (несколько ядер на кристалле)
- Мобильные компьютеры (уменьшились размеры компьютеров,а вычислительная мощность сохранилась)
- Высокопроизводительные вычислительные системы (несколько процессоров с многими ядрами могут связываться между собой посредством быстрого сетевого интерфейса или другой быстрой коммуникационной среды)
- Виртуализация выполнения программ (детали работы hardware скрываются от пользователя, например, детали работы с дисководом, памятью и др.)
- Облачные технологии (хранение на компьютере лишь части данных, остальная часть хранится на облаке, на более мощных серверах)

Как говорилось выше, эволюция вычислительных систем поможет нам сделать выводы о функциях операционной системы.

Основные функции операционной системы

- Планирование заданий и использование процессора (определение, какое задание поступит очередным в вычислительную машину и какая программа в очереди захватит процессор)
- Обеспечение программ средствами коммуникации и синхронизации (для решения совместных задач)
- Управление памятью (размещение программ и виртуальная память)
- Управление файловой системой (хранение информации различных пользователей)
- Управление вводом-выводом (привилегированная операция)
- Обеспечение безопасности (забота о безопасности программ и пользователей)

Заключение

Мы рассмотрели эволюцию операционных систем, но не говорили о том, что он является единственным и правильным. Операционные системы существуют, потому что на данный момент их существование - это разумный способ использования вычислительных систем.

Тема №2. Архитектурные особенности построения ОС

Ссылка на лекции

https://youtu.be/zBc4ggpDEZA

Содержание темы

- введение,
- монолитное ядро,
- многоуровневые системы,
- микроядерная архитектура,
- виртуальные машины,
- экзоядерная архитектура,
- заключение.

Введение

Мы рассмотрели, что такое операционная система и какие функции она выполняет, но теперь нас интересует, как операционная система выполняет эти функции, поэтому перейдём к рассмотрению нескольких подходов к архитектуре операционной системы.

Монолитное ядро

- Операционная система - программа, состоящая из набора функций или процедур
- Каждая процедура может вызывать любую другую процедуру
- Все процедуры работают в привилегированном режиме (для уменьшения трат времени при переходе от одной процедуре к другой на смену пользовательского или привилегированного режима)
- Ядро совпадает со всей операционной системой
- Пользовательские программы взаимодействуют с ядром через системные вызовы (точки входа в ядро - это вкрапления в наш монолит)

Достоинства: высокая скорость работы.
Недостатки: нахождение большого объёма в оперативной памяти, необходимость перекомпиляции при каждом изменении, сложность отладки.

Многоуровневые (Layered) системы

- Операционная система - это набор уровней: нижний - hardware, верхний - интерфейс пользователя, а также промежуточные уровни
- Процедура уровня K может вызвать только процедуры уровня K-1
- Почти все уровни (кроме уровня интерфейса пользователя) работают в привилегированном режиме
- Ядро совпадает или почти совпадает со всей операционной системой
- Пользовательские программы взаимодействуют с операционной системой через интерфейс пользователя

Достоинства: отладка только измененного уровня.
Недостатки: нахождение большого объёма в оперативной памяти, необходимость перекомпиляции при изменениях, меньшая скорость работы из-за прохода по всем уровням.
Пример - система THE, имеющая 6 уровней: (0) hardware, (1) планирование задач и процессов, (2) управление памятью, (3) драйвер связи с консолью, (4) управление вводом-выводом, (5) интерфейс пользователя.

Микроядерная (microkernel) архитектура

- Операционная система включает в себя микроядро с функциями обеспечения взаимодействия между программами, планирования использования процессора, первичной обработки прерываний и операций ввода-вывода, базового управления памятью, а также включает в себя менеджер сети, менеджер памяти, менеджер файлов и другие
- В привилегированном режиме работает только микроядро (остальные части - в пользовательском)
- Взаимодействие частей ОС между собой и с программами пользователей происходит путем передачи сообщений через микроядро

Достоинства: простота отладки, возможность замены компонент без перекомпиляции и остановки системы.
Недостатки: очень медленная скорость работы (микроядро - "бутылочное горлышко"операционной системы).

Виртуальные машины

- Каждому пользователю предоставляется своя копия виртуального hardware (реальное hardware ↔ реальная ОС ↔ виртуальное hardware ↔ Linux/Windows10/MS-DOS/? ↔ пользователь)

Новая микроядерная архитектура - экзоядерная архитектура

- Операционная система включает в себя микроядро теперь с базовыми функциями обеспечения взаимодействия между программами, выделения и высвобождения физических ресурсов, контроля прав доступа
- В привилегированном режиме работает только микроядро (остальные части - в пользовательском)
- Менеджеры и остальные функции микроядра выносятся в библиотеки для различных пользователей и поддерживают абстракции высокого уровня (файлы, виртуальная память, абстракции под Windows, абстракции под Linux и другие)
- Взаимодействие ОС с программами пользователей происходит путем библиотек

Заключение

Стоит отметить, что в чистом виде ни одна из приведенных архитектур в современных операционных системах не используется, обычно используются определенные смешанные системы, потому что каждый подход имеет свои достоинства и недостатки.

Тема №3. Понятие процесса. Операции над процессами

Ссылка на лекции

https://youtu.be/akHyhNmEllc

Содержание темы

- введение,
- терминология,
- жизнедеятельность процесса,
- операции,
- модель процесса,
- создание процесса,
- завершение процесса.

Введение

"Программа" в обычном понимании не может использоваться для описания происходящего внутри ОС. Например, два студента запускают на выполнение программу sqrt с различными входными данными. С пользовательской точки зрения программы идентичны, но на уровне вычислительной машины они будут выполняться по-разному. "Задание" тоже не может использоваться для описания происходящего внутри ОС. Так мы приходим к термину "процесс".

Терминология

Термин "процесс" характеризует находящуюся под управлением ОС совокупность

- набора исполняющихся команд,
- ассоциированных с ним ресурсов,
- текущего момента его выполнения.

Неправильно считать, что процесс - это программа, которая исполняется, потому что для исполнения одной программы может организовываться несколько процессов, в рамках одного процесса может исполняться несколько программ и даже код, отсутствующий в программе.

Диаграмма жизнедеятельности процесса

- рождение - то, что происходит с процессом при его появлении
- готовность - процесс может работать, как только ему передадут процессор
- ожидание - процесс может работать, когда в системе произойдёт определенное событие
- закончил исполнение - то, что происходит после завершения процесса

Операции над процессами

- создание процесса: рождение → готовность
  завершение процесса: исполнение → закончил исполнение
- запуск процесса: готовность → исполнение
  приостановка процесса: исполнение → готовность
- блокирование процесса: исполнение → ожидание
  разблокирование процесса: ожидание → готовность
- изменение приоритета

Одноразовые операции: создание процесса, завершение процесса
Многоразовые операции: запуск процесса, приостановка процесса, блокирование процесса, разблокирование процесса, изменение приоритета

Модель процесса

Введем модель процесса в ОС, чтобы понять, как организованы операции над процессами в ОС.
Process Control Block (PCB) - структура данных, которая хранит

- состояние процесса
- программный счетчик - какая команда должна выполняться следующей
- содержимое регистров
- данные для планирования использования процессора и управления памятью
- учетная информация: кто запустил процесс, сколько времени проработал процесс, сколько времени запрошено и т.д.
- сведения об устройствах ввода-вывода, связанных с процессом

Содержимое PCB меняется не от программы к программе, а только при совершении ОС операций над процессом.

Контекст процессора:

- регистровый контекст: программный счетчик, содержимое регистров
- системный контекст: состояние процесса, данные для планирования использования процессора и управления памятью, учетная информация, сведения об устройствах ввода-вывода, связанных с процессом
- пользовательский контекст: код и данные в адресном пространстве

Процессы связаны отношениями родитель-ребенок и образуют генеалогические леса.
Теперь перейдем к рассмотрению операций над процессом, начиная с одноразовых.

Создание процесса

1. Порождение нового PCB с состоянием процесса "рождение"
2. Присвоение идентификационного номера
3. Выделение ресурсов (адресного пространства, устройств ввода-вывода) из ресурсов родителя или из свободных ресурсов ОС
4. Занесение в адресное пространство кода и установка значения программного счетчика, клонируя родителя (UNIX) или используя информацию с внешнего носителя, файла (Windows)
5. Окончание заполнение PCB
6. Изменение состояния процесса на "готовность"

Когда процесс-родитель породил другой процесс, то в зависимости от ОС эти два процесса или начинают жить совместно в вычислительной системе, или родитель ждёт, пока не завершится процесс-ребенок.

Завершение процесса

1. Изменение состояния процесса на "закончил исполнение" (даже если процесс был убит)
2. Освобождение ресурсов (PCB остаётся)
3. Очистка соответствующих элементов в PCB
4. Сохранение в PCB информации о причинах завершения

Но даже при очистке ресурсов и элементов PCB процесс не покинет вычислительную систему, пока хранится причина смерти процесса. Такие процессы называют процессами-зомби. Когда процесс-родитель или завершится, или запросит причину "смерти" процесса, или скажет ОС, что не интересуется причиной "смерти" процесса, то данный процесс-ребенок окончательно покинет вычислительную систему.
Когда умирает процесс-ребёнок, то с родителем ничего страшного не происходит, некоторые ОС информируют родителя о завершении процесса-ребёнка.

Когда умирает процесс-родитель, то ОС может завершить всё поддерево потомков для данного процесса, но современные ОС используют другой подход. PCB хранит идентификационные номера процессов, которые их породили, поэтому, чтобы не возникало коллизий при смерти процесса-родителя и нарушения целостности генеалогического дерева процессов (когда идентификатор указывает на новый процесс, занявший идентификатор умершего родителя), идетификатор процесса-родителя у детей (сирот) заменяется на идентификатор процесса, который будет жить, пока живет операционная система.