Основы операционных систем 1 модуль.

Андрей Тищенко

2024/2025

Лекция 2 сентября.

1 Структура вычислительной системы.

Вычислительная система состоит нескольких частей:

- Пользователь. (алгоритмы и алгоритмические языки)
- Прикладные программы. (ЦГ, matlab и т.п.)
- Системные программы. (системное программирование)
- Операционная система. (основы операционных систем)
- Техническое обеспечение. (архитектура ЭВМ и языки ассемблера)

Прекрасная притча про слепцов и слона.

1.1 Что такое операционная система?

Различные точки зрения (что такое ОС?):

- Распорядитель ресурсов.
- Защитник пользователей и программ.
- Виртуальная машина (фокусник, виртуальная память).
- Кот в мешке (попросил загрузить ОС, значит всё загруженное и есть ОС).

- Постоянно функционирующее ядро.

Проще сказать не что такое операционная система, а для чего она нужна и чем занимается. Современные ОС это продукт эволюции вычислительных систем, поэтому стоит эту эволюцию рассмотреть.

1.1.1 Эволюция вычислительных систем

Удобство, стоимость и производительность - самые главные факторы отбора в эволюции операционных система.
Условные этапы развития вычислительных систем:

1-й период. (1945-1955гг.) Научно-исследовательнская работа в области вычислительной техники.

- Ламповые машины.
- Нет разделения персонала.
- Вход программы коммутацией или перфокартами.
- Одновременное выполнение толко одной операции.
- Появление прообразов первых компиляторов.
- Нет операционных систем.

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

- Транзисторные машины
- Происходит разделение персонала (появление малочисленной касты программистов)
- Бурное развитие алгоритмических языков (напиши 10 000 строк на асме и отладь, это стало причиной появления первых языков программирования)
- Ввод заадния колодой перфокарт
- Вывод результатов на печать
- Пакеты заданий и системы пакетной обработки.

3-й период. (начало 60-х годов - 1980 гг.) Конец этого периода указан точно. Тактовая частота значительно повысилась из-за появления микросхем.

- Машины на интегральных схемах.
- Использование спулинга spooling. (появление процессоров ввода/вывода, которые включаются вместо центрального процессора)
- Планирование заданий (из-за появления магнитных дисков вместо ленты).
- Мультипрограммные пакетные системы. (в память загружается несколько программ, пока происходит ввод/вывод на в одной программе ЦП передаётся другой программе)
- Системы разделения времени (time-sharing).
- Интерактивная отладка программ.
- Виртуальная память.
- Появление семейств ЭВМ.

Мультипрограммирование и эволюция вычислительных систем.

Software	**Hardware**
Планирование заданий	Защита памяти
Управление памятью	Сохранение контекста
Сохранение контекста	Механизм прерывания
Планирование использования процессора	Привилегированные команды
Системные вызовы	
Средства коммуникации	
Средства синхронизации	

Интерактивная отладка стала возможна благодаря изобретению терминала, но возникла новая проблема: нужно хранить информацию разных пользователей так, чтобы они не могли портить данные других пользователей, также потребовался единый стандарт хранения данных в компьютере.

Хотелось разрешить как можно большему числу пользователей одновременно пользоваться вычислительной машиной. Для этого нужно было экономить оперативную память. С целью решения этой проблемы программы подгружаются в машину по частям (только нужные в данный момент куски). На этой концепции основана виртуальная память.

Семейства ЭВМ были нужны для того, чтобы маленькие кампании могли покупать маломощные компьютеры, писать свой код на них, по мере роста кампании покупать более мощные машины того же семейства и исполнять такой же код без переписывания и перекомпиляции.

4-й период (1980 - 2005 гг.)

- Машины на больших интегральных схемах (БИС).
- Персональные ЭВМ.
- Дружественное программное обеспечение.
- Сетевые и распределённые операционные системы.

Поскольку теперь любой человек мог стать владельцем персонального ЭВМ, было необходимо создать дружелюбный интерфейс для предоставления человеку без специального образования пользоваться персональной машиной.

Из-за увелечения количества компьютеров увеличилась важность компьютерных сетей. Пользователи, работающие на сетевой ОС должны были понимать как доставать и обрабатывать файлы из сервера.

Владелец распределённой операционной системы не должен понимать, где находится его файл, обращение к файлу на локальной машине и к файлу на сервере, управляемом распределённой ОС, должно выглядеть для пользователя одинаково.

Широкое использование ЭВМ в быту, образовании и на производстве.

5-й период (2005-?? гг.)

- Машины на многоядерных процессорах.
- Мобильные компьютеры.
- Высокопроизводительные вычислительные системы.
- Облачные технологии.
- Виртуализация выполнения программ.

Можно виртуализовать программы и операционные системы.

Период Глобальной компьютеризации.

1.2 Основные функции ОС

- Планирование заданий и использования процессора. (кто и после кого будет выполняться)
- Обеспечение программ средствами коммуникации и синхронизации.
- Управление памятью.
- Управление файловой системой.
- Управление вводом-выводом.
- Обеспечение безопасности.

Операционные системы существуют потому что на данный момент их существование - это разумный способ использования вычислительных систем.

2 Архитектурные особенности построения ОС

Внутреннее строение ОС (разновидности)

Монолитное ядро.

- Каждая процедура (функция) может вызвать каждую.
- Все процедуры работают в привилегированном режиме.
- Ядро совпадает со всей операционной системой.
- Пользовательские программы взаимодействуют с ядром через системные вызовы.

Многоуровневые (Layered) системы.

- Процедура уровня K может вызывать только процедуры уровня $K-1$
- Все или почти все уровни работают в привилегированном режиме
- Ядро совпадает или почти совпадает со всей операционной системой

- Пользовательские программы взаимодействуют с ОС через интерфейс пользователя

Самый низкий уровень - Hardware. Самый высокий - интерфейс пользователя. В самой первой такой машине выглядело так:

1. Интерфейс пользователя.
2. Управление ввода вывода.
3. Связь с консолью.
4. Управление памятью.
5. Планирование заданий и программ.

Работает медленнее, чем система монолитного ядра, но отладка и модификация сильно упрощается.

Микроядерная (microkernel) архитектура.

Функции микроядра:

- Взаимодействие между программами.
- Планирование использования процессора.
- Первичная обработка прерываний и процессов ввода вывода
- Управление памятью.

Устройство системы:

- Микроядро составляет лишь малую часть ОС.
- В привилегированном режиме работает только микроядро.
- Взаимодействие частей ОС между собой и с программами пользователей путем передачи сообщений через микроядро.

Все части ОС, кроме микроядра, могут быть заменены без прерывания работы ОС. Микроядро сразу же сможет обратиться к заменённой части системы.

Микроядро в таком концепте работает весьма медленно, поэтому на практике для повышения быстродействия функционал микроядра расширили.

Виртуальные машины

Каждому пользователю предоставляется своя копия виртуального hardware

Новая микроядерная архитектура (экзоядерная архитектура).

- Взаимодествие между программами
- Выделение и выспобождение физических ресурсов
- Контроль прав доступа

Вся остальная функциональность выделяется на поддержание абстракций, которые разделены на библиотеки. Считается спорным подходом и на практике не используется (только в учебных ОС).

Смешанные системы.

Монолитное ядро - необходимость перекомпиляции при каждом изменении, сложность отладки, высокая скорость работы.
Многоуровневые системы - необходимость перекомпиляции при изменениях, отлаживается только изменённый уровень, меньшая скорость работы.
Микроядро - простота отладки, возможность замены компонент, без перекомпиляции и остановки системы, очень медленные.
Все три подхода используются в совеременных ОС.

3 Понятие процесса. Операции над процессами.

Уточнение терминологии

Термин *программа* - не может использоваться для описания происходящего внутри ОС.
Термин *задание* - не может использоваться для описания происходящего внутри ОС.
Изначально эти термины были придуманы для статических объектов, а нам нужно описывать динамические объекты.
Введём термин *процесс* для описания динамических объектов.
Теперь под *процессом* будем понимать совокупность:

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

Вся эта совокупность находится под управлением операционной системы.
Важно: Процесс $\neq$ программа, которая исполняется.

- для исполнения одной программы может организовываться несколько процессов
- В рамках одного процесса может исполняться несколько программ
- В рамках процесса может исполняться код, отсутствующий в программе.

Состояния процесса

1. Исполняется
2. Не исполняется

Новые процессы не исполняются. Переход из состояний происходит в случае выбора его для исполнения или приостановки. После исполнения процесс выходит из ОС.

Примитивная система, которая не является верной, так как процесс может "не исполняться разными способами".

1. Готовность
2. Исполнение
3. Ожидание

Новые процессы попадают в состояние готовности. Выбор готового процесса для исполнения прерывает в состояние исполнения. В случае прерывания исполняемого процесса он переходит в состояние готовности. Если для исполнения нужно дождаться какого-то события, то он переходит в состояние ожидания, после исполнения ожидаемого, он переходит в состояние готовности. Однако эту схему тоже стоит дополнить:

1. Рождение

2. Готовность
3. Исполнение
4. Ожидание
5. Закончил исполнение

Новые процессы попадают в состояние рождения, после прохождения допуска к планированию процесс попадает в состояние готовности. При завершении работы процесса он попадает в состояние "закончил исполнение". **Важно:** переход между состояниями процесса осуществляет ОС.

Лекция 16 сентября

Операции над процессами:

- Создание процесса - завершение процесса
- Запуск процесса - приостановка процесса
- блокировка процесса - разблокирование процессса
- изменение приоритета

Чтобы ОС могла взаимодействовать с процессами, она должна иметь какое-то представление об этом объекте. Эта информация хранится в PCB (process control block).
Содержимое PCB

- Состояние процесса
- Програмнный счётчик (следующая команда в процессе)
- содержимое регистров
- данные для планирования использования процессора и управления памятью
- учётная информация
- сведения об устройствах ввода-вывода, связанных с процессом (возможно файлы, с которыми взаимодействует).

Регистровый контекст: программный счётчик, содержимое регистров.
Всё остальное входит в системный контекст.
Всё, что лежит в PCB лежит в ядре ОС, код и данные программы называются пользовательским контекстом, входят в адресное пространство процесса.
Создание процесса:

- Присвоение идентификационного номера
- Порождение нового PCB с состоянием процесса "рождение"
- Выделение ресурсов (из ресурсов родителя и ОС)
- Занесение в адресное пространство кода и установка значения программного счётчика. Может создаться дубликат родителя, владеющий всей информацией родителя до момента желания создать дочерний процесс. Информация процесса может быть получена из файла (пользовательский контекст убирается, вместо него через системный вызов открывается исполняемый файл и подгружается как пользовательский контекст).
- Окончание заполнения PCB
- Изменение состояния процесса на "готовность"

Завершение процесса: Это состояние необходимо для того, чтобы родитель мог узнать, по какой причине ребёнок завершил процесс.

- Изменение состояния процесса на "закончил исполнение"
- Освобождение ресурсов (остаётся только PCB процесса, в нём остаётся учётная информация, родитель)
- Очистка соответствующих элементов в PCB
- Сохранение в PCB информации о причинах завершения.

После окончания родительского процесса некоторые ОС убивают всех его детей, а некоторые (например Windows и Unix) позволяют детям жить.
Но в таком случае возникает вопрос с новым родителем. Например у процесса 251 был ребёнок 1000, после окончания 251 1000 продолжил работать, ОС создала новый процесс 251, который не должен являться

родителем 1000. Поэтому осиротевшие процесса усыновляются процессами, которые существуют, пока ОС не прекратит работу.
Если процессы используют какой-то ресурс (например файл), ему присваивается счётчик, равный количеству процессов, использующих этот ресурс. Освобождение происходит если один из процессов явно это попросил или если счётчик стал равен нулю.
Запуск процесса:

- Изменение состояния процесса на "исполнение"
- Обеспечение наличия в оперативной памяти информации, необходимой для его выполнения
- Восстановление значений регистров
- передача управления по адресу, на который указывает программный счётчик.

PCB меняется только при переключении состояния процесса. Во время работы процесса ничего в PCB не заносится (так как это было бы непроизводительно).
Приостановка процесса:

- Автоматическое сохранение программного счётчик и части регистров (работа hardware)
- Передача управления по специальному адресу (hardware)
- Сохранение динамической части... UNFINISHED

Блокирование процесса

- Сохранение контекста процесса в PCB.
- Обработка системного вызова (разбираемся, чего мы ждём)
- Перевод процесса в состояние "ожидание"

Разблокирование процесса:

- Уточнение тоо, какое именно событие произошло
- Проверка наличия процесса, ожидающего этого события
- Перевод ожидающего процесса в состояние "готовность"
- Обработка произошедшего события

4 Кооперация процессов и основные аспекты ее логической организации

Основные причины объединения усилий:

- Повешение скорости решения задач.
- Совместное использование данных.
- Модульная конструкция какой-либо системы.
- Для удобства работы пользователя (например, отладчик).

Кооперативные или взаимодейтсвующие процессы - это процессы, которые влияют на поведение друг друга путем обмена информацией.
Категории средств взаимодействия:

- Сигнальные (переданная информация зачастую ограничивается битами).
- Канальные (создание логического канала связи. Данные из одного процесса могут быть получены при желании другого, тогда произойдёт передача запрошенных данных)
- Разделяемая память (доступна обоим процессам).

Как устанавливаестя связь

- Нужна или не нужна инициализация?
- Способы адресации
    1. Прямая адресация
        - симметричная
        - асимметричная
    2. Непрямая или косвенная адресация

Информационная валентность процессов и средств связи

- Сколько процессов может быть ассоциировано с конкретным средством связи?

- Сколько идентичных средств связи может быть задействовано между двумя процессами?
- Направленность связи
  - Симплексная связь (только в одну сторону)
  - Полудуплексная связь (двусторонняя, но только по очереди, как рация)
  - Дуплексная связь (двусторонняя)