

Операционные системы

Рязанова Наталья Юрьевна

2019

Оглавление

1	История	2
1.1	Первое поколение	2
1.2	Второе поколение	2
1.3	Третье поколение	2
1.4	Прерывания	3
1.5	Терминалы	3
1.6	Виды системного ПО	4
1.7	Четвертое поколение	4
1.8	Эра сверхбольших интегральных схем	4
1.9	Дисциплины курса	4
1.10	Операционная система	4
1.10.1	Главные ресурсы программы	5
1.10.2	Классификация ОС	5
1.10.3	Системы реального времени	5
2	Разработка операционных систем и их особенности	6
2.1	Иерархическая структура операционной системы	6
2.2	Управление процессорами	7
2.2.1	Диаграмма состояния процесса	7
2.2.2	Клонирование и диспетчеризация	7
2.2.3	Планирование в системах разделения времени	8
2.2.4	Уровни наблюдения	9
2.3	Переключение в режим ядра	9
2.4	Переключение контекста	10
2.5	Потоки	10
2.5.1	Однопоточная модель	11

Глава 1

История

Деления на поколения можно считать очень условными.

Компьютер - программно управляемое устройство.

Современным ПК часть времени управляет ОС, часть времени - приложения.

1.1 Первое поколение

В 40-х годах потребность в больших, быстрых, точных вычислениях увеличилась из-за гонки вооружений.

- В 1944 году в США была создана MARK I - первая вычислительная машина на электромагнитных реле.
- В 1946 году - первая электронно-цифровая машина на электромагнитных лампах - UNIVAK.
- 1945-46 гг. - создание первого поколения ЭВМ. Длится до 1955 года. Особенности: использовались электронные лампы, ЗУ на линиях задержки, ЗУ вращающегося типа, концепция хранения программ. Для ввода/вывода - перфокарты, печатающее устройство.

Первые серийные машины - MARK I, UNIVAK I, LEO I.

Все программы выполнялись в **абсолютных адресах** - адресах байта/слова (2 байта) и т.д. физической памяти. В эти же годы разработали ENIAC. В группу разработки в 1944 г. вошел Джон Фон Нейман. В 1945 г. он опубликовал доклад, в котором определены основные принципы построения компьютерной машины, которую называют компьютером. В 1946 г. - статья "Предварительная конструкция ЭВМ" в ней была описана формальная организация работы машины.

Принцип хранимой программы - данные и команды хранятся в одной памяти. Для того, чтобы к ним обращаться, они хранятся в определенных адресах. Требуется это счетчик команд, он хранит адрес следующей команды.

1.2 Второе поколение

С середины 50-х годов - отсчет второго поколения ЭВМ. Появились диоды и триоды (транзисторы) и ОЗУ на магнитных сердечниках.

Для серийного производства машины нужна техническая документация.

IBM 1401 → IBM 7094 → 1401

Для автоматического перескока с 1 задания на другое понадобилось специальное ПО, которое стали называть ОС. Была создана серия оброчных и отладочных программ, которые помогали программисту. Для управления заданиями был разработан специальный язык - язык управления заданиями.

1.3 Третье поколение

С начала 60-х годов - 3 поколение ЭВМ. Появились микросхемы. К 3 поколению относится полноценное появление **архитектуры ЭВМ**. Появилось в связи с идеей мультипрограммными обработками и реализацией распараллеливания функций. Для ускорения переключения между заданиями - загрузка нескольких заданий в ОЗУ.

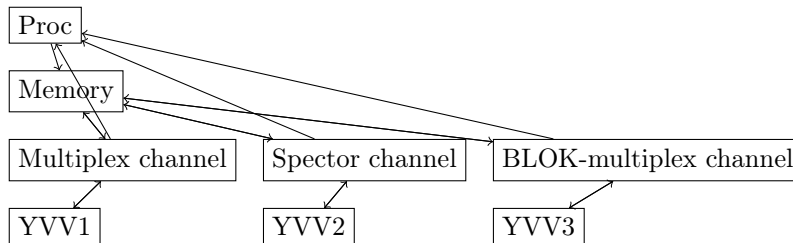
Большое число программ разделяет одно адресное пространство физической памяти.

Процесс - программа в стадии выполнения.

Перед ОС появились новые задачи, появился менеджер памяти для выделения/распределения/очистки. Нужно было обеспечить защиту памяти выделенной для программы.

В 3 поколении появилась начальная архитектура. **Канал** - программно управляемое устройство, в задачи которого входило управление внешними устройствами.

IBM 360 - машина третьего поколения.



1.4 Прерывания

1. Системные вызовы (программные прерывания)

2. Исключения (исключительные ситуации)

- Исправимые
- Неисправимые

3. Аппаратные прерывания

- Таймер
- Ввод-вывод
- ОТ действий итератора

Прерывания являются синхронными. Программные прерывания являются асинхронными и несвязанными ни с какой другой работой в системе.

1.5 Терминалы

Операционные системы для IBM 360:

- OS/360
- TSS/360

Для того, чтобы обеспечить комфортную работу большого числа пользователей время процессора стали квантовать. Такие системы всегда мультипрограммные (системы разделения времени). По истечении кванта система должна решить какому процессу выделить следующий квант, это делается планировщиком диспетчером (организует очередь процессов, а диспетчер непосредственно выделяет память процессу). Таким образом в системах разделения времени основная функция - декремент кванта. Кроме этого, там есть ряд других важнейших функций. Обработчик прерывания выполняется на очень высоком уровне приоритетов (например в windows видно, что все прерывания от внешних устройств принято называть девайсами приоритет выше, чем dpc/dispatch). Выполнение обработчика прерывания никакой процесс прервать не может, поэтому код должен завершаться быстрее. В результате каждый пользователь работал независимо от других. Система должна гарантировать время ответа. Система должна была успевать найти ошибку в программе или запрос ввода вывода. TSS была очень медленная и была разработана другая ОС **CP/CMS**.

OS Multics на платформе 6000 series Multiplexed Information and computing service. Изначально создавалась как система замера времени и стала прародителем UNIX.

Уже IBM 360 разрабатывалась как серия машин (то есть имели разный состав периферии и мощность). Также очень сильно отразилось на программировании, что IBM разделила стоимости харда и софта.

1.6 Виды системного ПО

- Системное программное обеспечение - ОС и утилиты ОС
- Системы программирования

1.7 Четвертое поколение

Начало четвертого поколения относится к 1970 году. Это уже большие интегральные схемы, меняются элементы памяти, объем запоминающего устройства увеличивается, габариты уменьшаются, мощность увеличивается.

Для формирования изображения используется смена интенсивности пикселя раstra (телевизионная развертка).

Появление IBM 370

Виртуальн 370	Виртуальн 370	Виртуальн 370
CMS	CMS	CMS
VM 370		
Аппаратное обеспечение 370		

PDP-11 уже почти современный UNIX. Первоначально UNIX называлась UNICS (что обыгрывало название MULTICS).

API. Каждая из фирм становилась монополистом.

UNIX сразу писалась как система времени. POSIX (portable operating system interface). Был предложен стандарт IEEE.

Ричард Столмен является основателем проекта GNU. GNU это рекурсивный акроним от английского GNU's not UNIX. Свои работы начал 1983 году и его задачей было разработать ПО с открытым исходным кодом. В 1992 году Линус Торвалдс начал разработку ОС под той же системой. На сегодняшний день это GNU Linux.

В 1964 году Сеймуром Крейем был создан первый компьютер СДС 6600. В 1972 году он основал свою первую собственную фирму Kreiry Search, там он создал самые быстрые компьютеры Krei 1 и Krei 2. Первый был первым коммерчески успешным векторным компьютером.

1.8 Эра сверхбольших интегральным схем

Очень мощные машины, способные решать очень широкий круг задач

Современные операционные системы являются системами разделения времени.

1.9 Дисциплины курса

1. Управление процессорами
2. Управление памятью (оперативная память)
3. Взаимодействие параллельных процессов
4. Управление данными (файловые подсистемы)
5. Управление внешними устройствами (В UNIX все - файл)

1.10 Операционная система

Операционная система - это комплект программ, которые совместно управляют ресурсами вычислительной системы и процессами, использующим эти ресурсы при вычислениях.

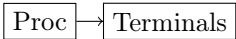
Ресурс - это любой из компонентов вычислительной системы и предоставляемые ею возможности.

Управление ресурсами заключается в том, что выделяются ресурсы процессом (основная задача ОС). Процессы создаются по мере необходимости, ресурсы выделяются по мере надобности.

1.10.1 Главные ресурсы программы

1. Процессорное время
2. Объем памяти
3. Ключи защиты
4. Системные таблицы
5. Реентирабельные коды - переход чистой процедуры (которая не изменяет саму себя)

1.10.2 Классификация ОС

1. Однопрограммная пакетной обработки
2. Мультипрограммной пакетной обработки
3. Системы разделения времени

4. Системы реального времени
5. Персональные операционные системы

1.10.3 Системы реального времени

Реальное время в операционных системах - это способность операционной системы обеспечить требуемый уровень сервиса в определенный промежуток времени в отличие от систем общего назначения. ОСРВ обеспечивает ответ системы за определенный промежуток времени, то есть обслуживание запроса и это всегда запрос системы или внешнего процесса.

- Жесткое реальное время - это когда интервал установлен жестко и не может быть превышен.
- Мягкое реальное время - возможность небольших отклонений от величины интервала.

Время ответа зависит от внешнего для системы объекта. Например, высчитывается химическая реакция которая длится месяц.

Глава 2

Разработка операционных систем и их особенности

2.1 Иерархическая структура операционной системы



Интерфейсы разделяются на:

- Прозрачные - позволяет обращаться через уровни
- Полупрозрачные - какие то обращения доступны к вышележащему уровню
- Непрозрачные - строго к низлежащему уровню

1- Символьный уровень

2- Именованние файлов

Таблица 2.1: Структура ядра ОС UNIX 4.4 BSD

Системные вызовы				//////////	Сис. выз.	//////////		
Управление терминалом		1	2	3	4	Сокеты	Обработка сигналов	Создание и завершение процесса
Необработанный телетайп	Обработанный телетайп	Файловая система	Виртуальная память		Сетевые протоколы	Маршрутизация	Планирование процессов	
	Дисциплины линий связи	Буферный КЭШ	Страничный КЭШ					
Драйверы символьных устройств		Драйверы блочных устройств			Сетевые устройства	Диспетчеризация процессов		

3- Отображение адреса

4- Страничные прерывания

///- Аппаратные и эмулируемые прерывания

2.2 Управление процессорами

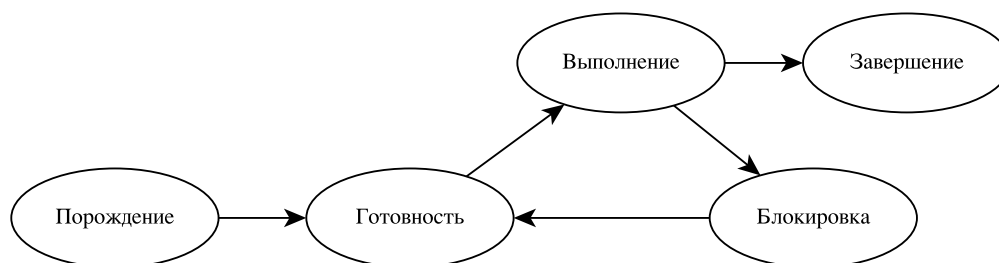
Планирование и диспетчеризация процесса в процессе чего выделяется процессорное время.

Планирование (scheduling) - это постановка процесса в очередь на выполнение.

Диспетчеризация - непосредственно выделение процессу процессорного времени.

2.2.1 Диаграмма состояния процесса

Основная таблица - таблица процессов, она содержит дескрипторы (описатели) процессов. Идентификатором процесса является номер строки в таблице процессов (порядковый номер)



В состоянии готовности находится большое количество процессов, которым необходимо выделить процессорное время. Очередь выстраивается в соответствии с реализованной в системе дисциплиной планирования.

В состоянии завершения у программы забираются все ресурсы и создаются в пулл свободных ресурсов

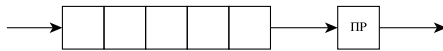
2.2.2 Клонирование и диспетчеризация

Классификация алгоритмов клонирования Планировщик шедьюлер (организация очереди процессов к какому либо ресурсу)

1. С преключением (процесс может выполняться от начала и до конца) и без преключения (процесс может быть снят с управления)
2. С приоритетами и без приоритетов
3. С вытеснением (процесс может быть вытеснен другим процессом с более высоким приоритетом) и без вытеснения (вытеснение возможно только в системе с приоритетом)

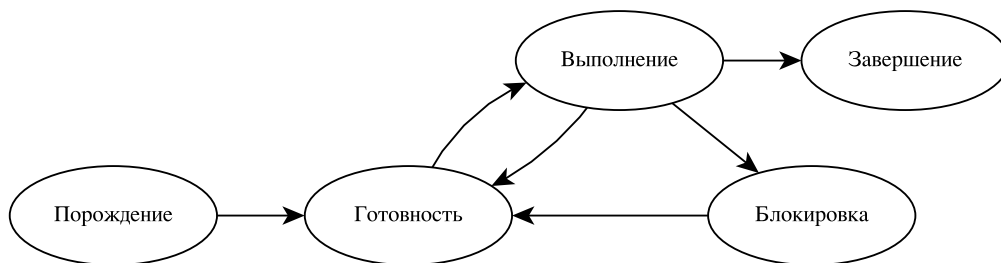
Мультипрограммная система пакетной обработки

1. **FIFO** - first in first out



2. **SJF** - shortest job first. Приводило к бесконечному откладыванию, то есть задания с большим процессорным временем все время откладывались в конец очереди
3. **SRT** - shortest reaming time (наименьшее оставшееся время). Процесс может быть прерван если поступит процесс с меньшим оценочным временем выполнения, чем время оставшееся текущему процессу до его завершения (на основе заявленного времени). Усугубляет ситуацию с бесконечным откладыванием.
4. **HRN** - highest response rationext (наибольшее относительное время ответа). Приоритет вычисляется по формуле

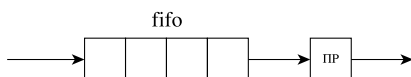
$$\frac{t_w + t_s}{t_s}, \text{ где } t_w - \text{время ожидания, } t_s - \text{запрошенное время}$$



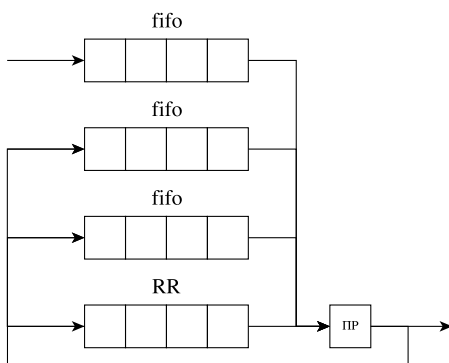
2.2.3 Планирование в системах разделения времени

Время в системах выдается квантами.

1. **RR** - циклическое планирование, процессы выстраиваются в очередь, но по истечении кванта ставятся в конец очереди



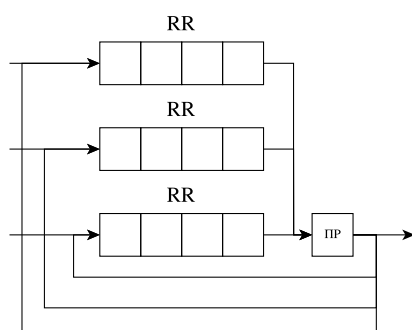
2. Многоуровневые очереди или алгоритм адаптивного планирования



В первую очередь с наивысшим приоритетом поступают только что созданные процессы или завершившие блокировку в ожидании завершения ввода-вывода. Для 1 очереди квант процесса выбирается таким образом, чтобы наибольшее число процессов успело или завершится или выдаст запрос на ввод-вывод. Если процесс не успел завершиться за по истечении кванта он поступает в слудющую очередь с более низким приоритетом, и так далее пока не окажется в последней очереди с самым низким приоритетом по алгоритму. В этой очереди крутится так называемый "холостой процесс поскольку система не может ничего не делать.

Пересчитываться могут только пользовательские приоритеты (приложений). Стараются быть справедливым, чтобы у всех процессов было примерно одинаковое время, но также учитывается время простоя процесса. В старом UNIX используется формула.

Интерактивные - процессы, запрашивающие ввод-вывод с клавиатуры, мыши. Самые высоко-приоритетные операции.



2.2.4 Уровни наблюдения

1. Последовательное выполнение
2. Квант выполняется 1 процесс, квант - другой (квазипараллельное). С точки зрения пользователя программы выполняются параллельно, с точки зрения процесса - последовательно
3. Параллельное выполнение. Реально параллельное, по времени совмещается выполнение команд программы.

Выполнение → готовность - вытеснен или истек квант

Выполнение → блокировка - запрос дополнительного ресурса

2.3 Переключение в режим ядра

Процесс - главная абстракция системы. UNIX декларирует следующим образом: процесс часть времени выполняет собственный код, и тогда он выполняется в **режиме задачи**, а часть времени выполняет реинтерабельный код операционной системы, и тогда он выполняется в **режиме ядра**.

Системные вызовы - запрос процесса на системные ресурсы (ввод-вывод). Ни одна операционная система не позволяет процессу напрямую обратиться к устройствам ввода-вывода. Если бы обращались напрямую, то такая система была бы незащищенной. Для обслуживания системного вызова процесс должен перейти в режим ядра, в режиме ядра выполняется реинтерабельный код операционной системы. Реинтерабельный код - код чистой процедуры (чистая процедура не модифицирует саму себя, то есть вынесены все данные). Коды ядра работают с системными таблицами (коды ядра реинтерабельны). Процедуры сами являются ресурсами, которые могут запрашивать процессы, разные процессы могут находиться в разных точках одной и той же процедуры. При выполнении системного вызова процесс переключается в режим ядра и выполняет реинтерабельный код операционной системы. После того, как итемный вызов обслужен, процесс опять переключается в режим задачи и выполняет собственный код.

В режиме ядра процесс переносится в очередь готовых процессов, выполняют действия для конкретного процесса. Затем когда процесс получает квант, он начинает выполнять собственный код. Если процесс запросил дополнительный ресурс, то процесс переключится в режим ядра и будет блокирован, будет находиться в режиме блокировки пока процесс не получит нужный ему ресурс, после чего система поместит процесс в очередь готовых процессов и после получения кванта процесс может продолжить выполнение своего кода до конца.

Три события, переводящих систему в режим ядра

1. Системные вызовы
2. Исключения
3. Аппаратные прерывания

2.4 Переключение контекста

Аппаратный контекст - регистры процессора (не все)

Сохранение регистров поддерживается аппаратно.

Полный контекст состоит из аппаратного контекста и информации о выделенных ресурсах, в частности о выделенной памяти. Это структуры. Все в системе описывается соответствующими структурами.

UNIX BSD \rightarrow struct proc

UNIX → struct task_struct

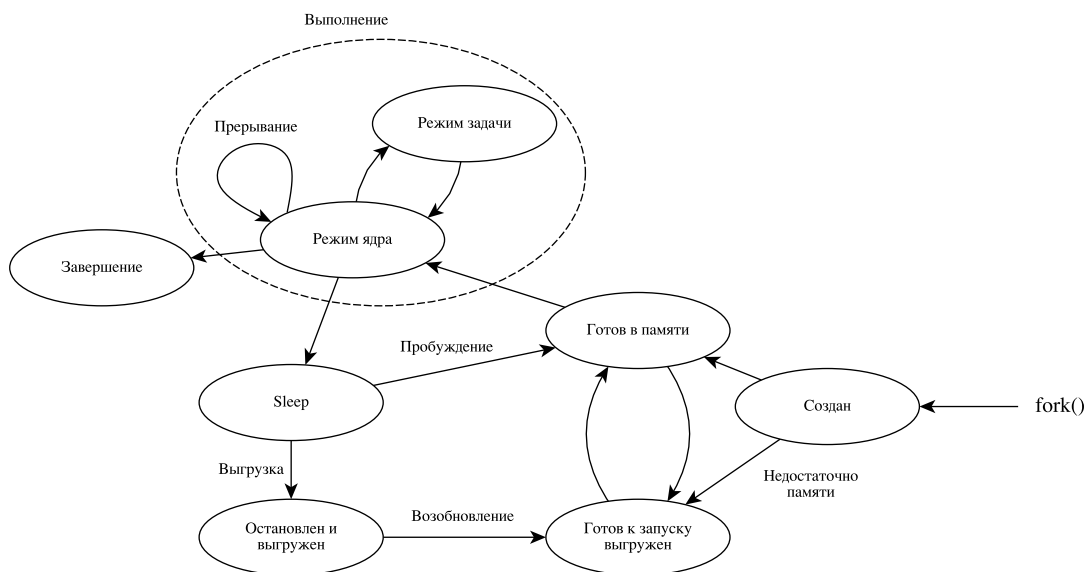
Вся эта информация нужна системе для того, чтобы иметь возможность выполнения процесса.

Понятие контекста является важнейшим в системе.

Полный контекст переключается тогда, когда процесс возвращается в готовность или ...

Диаграмма состояний процесса в UNIX

Любой процесс вызывается системным вызовом Fork. Он создает новый процесс.



2.5 Потоки

Чтобы сократить накладные расходы на переключение контекста (это очень затратно)

Возникло два типа потоков

1. Потоки на уровне пользователя
2. Потоки на уровне ядра

В качестве потока выделяются части кода процесса, которые могут выполняться параллельно с другими частями кода процессора. В любом случае владельцем ресурсов в процессе является процесс. Поток выполняется в адресном пространстве процесса (так как не имеет своего адресного пространства). Поток оказывается владельцем счетчика команд (аппаратного контекста).

Существуют многопоточные и однопоточные модели процесса.

2.5.1 Однопоточная модель

