Экзамен по ОСям

Будут сформированы “билеты” по 4 вопроса. Первый вопрос будет по практике с требованием построить какую-либо диаграмму по практическому заданию. Остальные три вопросы по теории из каждой аттестации (если у вас сдана та или иная аттестация, то вопрос по этой аттестации Вам задаваться не будет). Те, кто не смог построить диаграмму не смогут продолжить сдачу зачета. На зачет обязательно приходить со сданной практикой. Сданная практика - допуск к зачету.

1. В чем заключаются две основные функции операционной системы? Пояснить

1. Предоставление прикладным программистам (и прикладным программам, естественно) вполне понятный абстрактный набор ресурсов взамен неупорядоченного набора аппаратного обеспечения
2. Управление аппаратными ресурсами

2. В чем разница между системами с разделением времени и многозадачными системами?

Разделение времени — вариант многозадачности, при котором у каждого пользователя есть свой диалоговый терминал. ЦП по очереди предоставляется нескольким пользователям, желающим работать на машине.

3. В чем разница между режимом ядра и пользовательским режимом? Объясните, как сочетание двух отдельных режимов помогает в проектировании операционных систем.

Большинство компьютеров имеют два режима работы: режим ядра и режим пользователя. Операционная система — наиболее фундаментальная часть программного обеспечения, работающая в режиме ядра (этот режим называют еще режимом супервизора). В этом режиме она имеет полный доступ ко всему аппаратному обеспечению и может задействовать любую инструкцию, которую машина в состоянии выполнить. Вся остальная часть программного обеспечения работает в режиме пользователя, в котором доступно лишь подмножество инструкций машины. В частности, программам, работающим в режиме пользователя, запрещено использование инструкций, управляющих машиной или осуществляющих операции ввода-вывода (Input/Output — I/O).

4. При создании операционных систем одновременно решаются задачи, например, использования ресурсов, своевременности, надежности и т. д.

Приведите пример такого рода задач, требования которых могут протировечить друг другу.

- Многозадачность противоречит надежности. Чем больше программ работают одновременно, тем чаще между ними могут происходить конфликты, приводящие к "зависанию" компьютера.
- Дружественность противоречит простоте, поскольку достижение настоящей дружественности - это не простая, а сверхсложная задача
- Работоспособность в сети противоречит безопасности. С одной стороны мы хотим, чтобы наш компьютер мог запросто получать информацию со всего мира и общаться с другими компьютерами, а с другой строны, боимся, что вместе с информацией он может получить вирус.

5. Задача про время выполнения программ.

Рассмотрим систему, имеющую два центральных процессора, у каждого из которых есть два потока (работающих в режиме гипертрейдинга). Предположим, есть три запущенные програмы: P0, P1 и P2 со временем работы 5, 10 и 20 мс соответственно. Сколько времени займет полное выполнение этих программ? Следует принять во внимание, что все три программы загружают центральный процессор на 100 %, не осуществляют блокировку во время выполнения и не меняют центральный процессор, назначенный для их выполнения.

Выполнение программ может занять 20, 25, 30 или 35 мсек в зависимости от того как операционная система назначит их выполнение. Если P0 и P1 назначены на одном и том же CPU, а P2 на другом, то программы выполнятся за 20 мсек. Если P0 и P2 назначены на один CPU, а P1 на другом, то 25 мсек. Если P1 и P2 назначены на одном цп, а P0 на другом, то 30 мсек. Если все три программы будут назначены на один цп то 35 мсек

6. Почему в системах разделения времени необходима таблица процессов? Нужна ли она в операционных системах персональных компьютеорв, работающих под управлением UNIX или Windows при единственном пользователе?

Таблица процессов содержит информацию о каждом процессе: открытые файлы, состояния регистров, и т.д. В системах с одним пользователем все равно куча одновременных процесов, и таблица процессов нужна.

7. С точки зрения программиста, системный вызов похож на вызов любой другой библиотечной процедуры. Важно ли программисту знать, какая из библиотечных процедур в результате приводит к системным вызовам? Если да, то при каких обстоятельствах и почему?

Системный вызов приводит к прерыванию текущего процесса, переходу в режим ядра, выполнению вызова и возврату к выполнению процесса. Это долго, и если производительность важна, то лучше избегать системных вызовов.

8. Виртуальные машины приобрели высокую популярность по различным причинам. И тем не менее у них имеется ряд недостатков. Назовите их. Поясните разницу между эмуляцией и виртуализацией.

Недостатки:

- Не все процессоры поддерживают
- Медленно

Виртуализатор или совсем не эмулирует(имитирует) реальную машину, её архитектуру и процессор или делает это вминимальном варианте для отдельных ресурсов.

Эмулятор - полностью или почти полностью реализует для исполнения кода отдельную машину со своей архитектурой и своими ресурсами. Вплоть до того, что может быть процессор совершенно другой архитектуры.

9. Инструкции, касающиеся доступа к устройствам ввода-вывода, обычно относятся к привилегированным инструкциям, то есть они могут выполняться в режиме ядра, но не в пользовательском режиме. Назовите причину привилегированнности этих инструкций.

Устройства ввода-вывода это аппаратные ресурсы машины. Ими управляет ядро.

Причины:

- безопасность ресурсов компьютера
- скорость обработки
- предоставление программистам нормального API для работы с вводом-выводом.

10. Предположим, вам нужно разработать новую компьютерную архитектуру, которая вместо использования прерываний осуществляет аппаратное переключение процессов. Какие сведения необходимы центральному процессору? Опишите возможное устройство аппаратного переключения процессов.

Аппаратное переключение процесса производится путем записи в регистры ЦП значений, образующих аппаратный контекст процесса. Так как в состав этих регистров входят и адресные регистры (например, сегментные регистры), то замена аппаратного контекста процесса приводит к замене полного контекста процесса, выполняемого на ЦП.

11. Когда в результате прерывания или системного вызова управление передается операционной системе, используется, как правило, область стека ядра, отделенная от стека прерываемого процеса. Почему?

1. Ядро не может доверять пользовательскому стеку, там может быть что угодно.
2. Ядро может оставлять чуствительные данные в стеке, доступ к которым пользовательской программе должен быть закрыт.

12. У компьютерной системы достаточно места, чтобы хранить в основной памяти пять программ. Половину своего времени эти программы простаивают в ожидании ввода-вывода. Какая доля процессорного времени при этом тратится впустую?

$$\text{Степень загрузки ЦП } = 1 - p^n = 1 - 0.5^5 = 0.96875$$

13. Представьте себе мультипрограммную систему со степерью 6 (то есть имеющую в памяти одновременно шесть программ). Предположим, что каждый процесс проводит 40% своего времени в ожидании ввода-вывода. Каким будет процент использования времени центрального процессора?

$$\text{Степень загрузки ЦП } = 1 - p^n = 1 - 0.4^6 = 0.995904$$

14. Может ли поток быть приостановлен таймерным прерыванием? Если да, то при каких обстоятельстваъ, а если нет, то почему?

Может. Если прерывания не отключены (если программа в критической области, например)

15. В чем заключается самое большое преимущество от реализации потоков в пользовательском пространстве? А в чем заключается самый серьезный недостаток?

Можно реализовать в операционной системе, которая не поддерживает потоки (на данный момент - все). Потоки реализуются с помощью библиотеки.

Недостаток - реализация блокирующих системных вызовов.

16. Представьте себе систему реального времени с двумя голосовыми вызовами с периодичностью, равной 5 мс для каждого из них, со временем центрального процессора, затрачиваемого на каждый вызов, равным 1 мс, и с одним видеопотоком с периодичностью, равной 33 мс, со временем центрального процессора, затрачиваемого на каждый вызов, равным 11 мс. Можно ли спланировать работу такой системы?

Нет. За время обработки видео-вызова 11мс гарантированно придет аудио-вызов, который вызывается с периодичностью 5 мс.

Или да, ведь $\frac{1}{5} + \frac{1}{5} + \frac{11}{33} < 1$, и общее время, необходимое процессу, больше того времени, которое может предоставить процессор.

17. Гибкая система реального времени имеет четыре периодически возникающих события с периодами для каждого, составляющими 50, 100, 200 и 250 мс. Предположим, что эти четыре события требуют 35, 20, 10 мс и x процессорного времени соответственно. Укажите максимальное значение x, при котором система все еще поддается планированию.

$$\frac{35}{50} + \frac{20}{100} + \frac{10}{200} + \frac{x}{250} <= 1$$

$$\frac{175 + 50 + 12.5 + x}{250} <= 1$$

$$237.5 + x <= 250$$

$$x <= 12.5$$

18. Системе реального времени необходимо обработать два голосовых телефонных разговора,

каждый из которых запускается каждые 6 мс и занимает 1 мс процессорного времени при каждом использовании процессора, и один видеопоток со скоростью 25 кадров в секунду, где каждый кадр требует 20 мс процессорного времени. Поддается ли эта система планированию?

$$\frac{1}{6} + \frac{1}{6} + \frac{20}{\frac{1000}{25}}$$

$$\frac{1}{3} + \frac{1}{2} < 1$$

Поддается.

19. Рассмотрите систему, в которой желательно разделить политику и механизм планирования потоков, реализованных на уровне ядра. Предложите средства для достижения этой цели.

Нужно наличие способа параметризации алгоритма планирования, предусматривающего возможность пополнение параметров со стороны пользовательских процессов. Чтобы родительский процесс мог всесторонне управлять планированием дочерних потоков, даже если сам планированием не занимаетяс. Механизм находится в ядре, а политика устанавливается пользователем.

20. Процессу, запущенному в системе CTSS, для завершения необходимо 30 квантов времени. Сколько раз он должен быть перекачан на диск, включая самый первый раз (перед тем, как он был запущен)?

Считывание процесса с диска, 1 квант, сброс 2 кванта, сброс 4 ванта, сброс 8 квантов, сброс 16 квантов, процесс выполнен.

Всего 4 перекачивания на диск. А может быть и 5, учитывая "самый первый раз, перед тем, как он был запущен", хотя в самый первый раз он перекачивался не НА диск а С диска. Или 6, если учитывать еще и последние перекачивание на диск, хотя че его перекачивать, ведь процесс уже завершен.

Ответ: от 4 до 6.

21. В состоянии готовности к выполнению находятся пять заданий. Предполагаемое время их выполнения составляет 9, 6, 3, 5 и *x* . В какой последовательности их нужно запустить, чтобы свести к минимуму среднее время отклика? (Ответ будет зависеть от *x* .)

В порядке возрастания

22. Как в операционной системе, способной отключать прерывания, можно реализовать семафоры?

Слишком сложно, я хз

23. Могут ли два потока, принадлежащие одному и тому же процессу, быть синхронизированы с помощию семафора, реализованного в ядре, если эти потоки реализованы на уровне ядра? Ответьте на тот же вопрос применительно к потокам, реализованным на уровне пользователя. Предполагается, что к семафору не имеют доступа никакие другие потоки любых других процессов. Обоснуйте свой ответ.

Чет тоже сложно, тут знать надо...

24. Объясните, как значение кванта времени и время переключения контекста влияют друг на друга в алгоритме циклического планирования.

Установка слишком короткого кванта времени приводит к слишком частым переключениям процессов и снижает эффективность использования центрального процессора, но установка слишком длинного кванта времени может привести к слишком вялой реакции на короткие интерактивные запросы.

25. Генерация сигналов

Из пользовательских программ:

- kill (2)
- sigsend(2)
- alarm(2)

Из ядра:

- от клавиатуры
- от ошибок программирования

26. sa_flags

Поле sa_flags в struct sigaction формируется побитовым ИЛИ следующих значений:

A_ONSTACK - Используется для обработки сигналов на альтернативном сигнальном стеке.

SA_RESETHAND - Во время исполнения функции обработки сбрасывает реакцию на сигнал к SIG_DFL; обрабатываемый сигнал при этом не блокируется.

SA_NODEFER - Во время обработки сигнала сигнал не блокируется.

SA_RESTART - Системные вызовы, которые будут прерваны исполнением функции обработки, автоматически перезапускаются.

SA_SIGINFO - Используется для доступа к подробной информации о процессе, исполняющем сигнальный обработчик, такой как причина возникновения сигнала и контекст процесса в момент доставки сигнала. **SA_NOCLDWAIT** - Подавляет создание процессов-зомби.

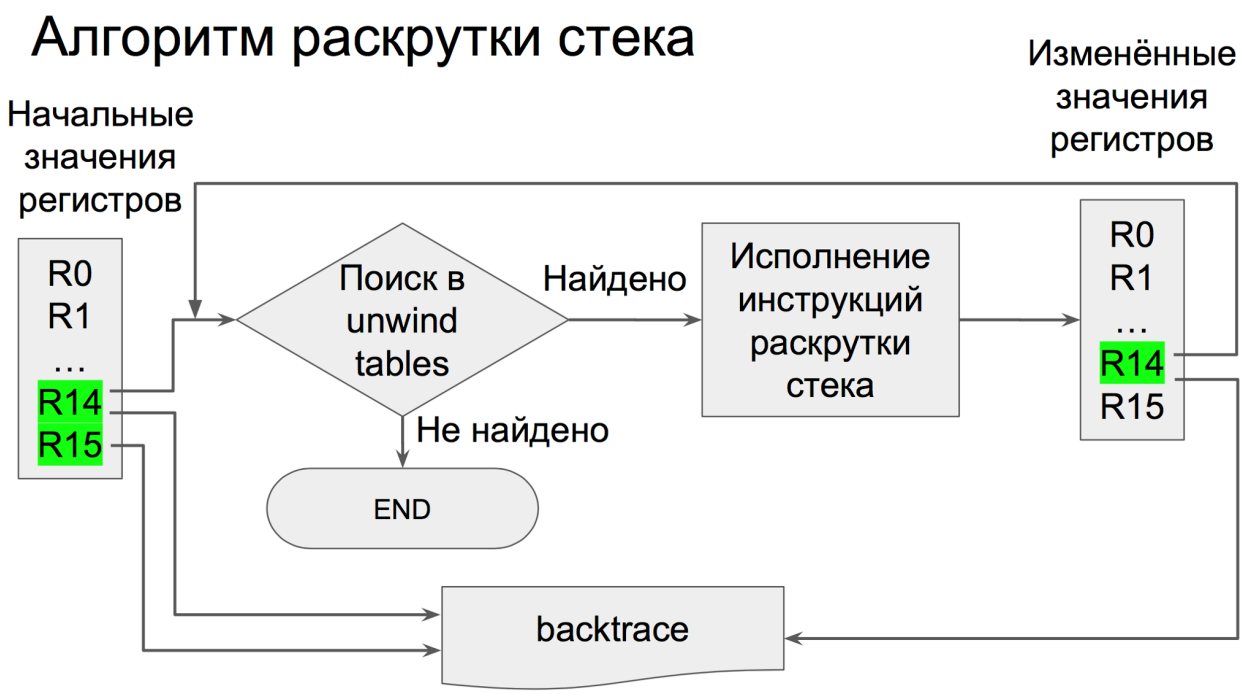
SA_NOCLDSTOP - Подавляет генерацию SIGCHLD, когда порожденные процессы останавливаются или возобновляются.

27. signal и sigset

Устанавливают адрес функции - обработчика сигнала

Системный вызов sigset аналогичен вызову signal с важным дополнением: sigset позволяет откладывать сигналы.

28. Алгоритм раскрутки стека



29. Сигналы для управления заданиями

Имя	Значение	Умолчание	Событие
SIGSTOP	23	Stop	Остановка (сигналом)
SIGTSTP	24	Stop	Остановка (пользователем)
SIGCONT	25	Ignore	Продолжение исполнения
SIGTTIN	26	Stop	Остановка при вводе с терминала
SIGTTOU	27	Stop	Остановка при выводе на терминал

30. Типы сигналов

POSIX определяет 28 сигналов, которые можно классифицировать следующим образом

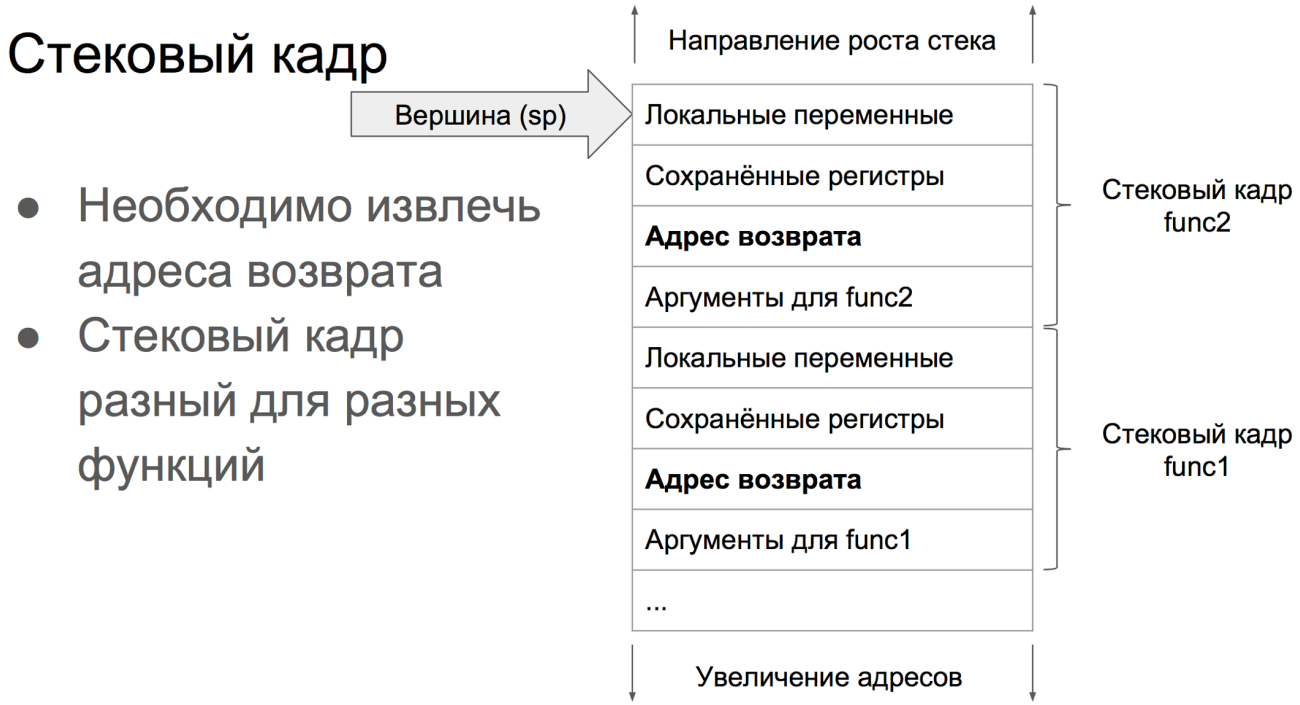
Название	Код	Действие по умолчанию	Описание	Тип
SIGABRT	6	Завершение с дампом памяти	Сигнал посылаемый функцией `abort()`	Управление
SIGALRM	14	Завершение	Сигнал истечения времени, заданного `alarm()`	Уведомление
SIGBUS	10	Завершение с дампом памяти	Неправильное обращение в физическую память	Исключение
SIGCHLD	18	Игнорируется	Дочерний процесс завершен или остановлен	Уведомление
SIGCONT	25	Продолжить выполнение	Продолжить выполнение ранее остановленного процесса	Управление
SIGFPE	8	Завершение с дампом памяти	Ошибочная арифметическая операция	Исключение
SIGHUP	1	Завершение	Закрытие терминала	Уведомление
SIGILL	4	Завершение с дампом памяти	Недопустимая инструкция процессора	Исключение
SIGINT	2	Завершение	Сигнал прерывания (Ctrl-C) с терминала	Управление
SIGKILL	9	Завершение	Безусловное завершение	Управление
SIGPIPE	13	Завершение	Запись в разорванное соединение (пайп, сокет)	Уведомление
SIGQUIT	3	Завершение с дампом памяти	Сигнал «Quit» с терминала (Ctrl-)	Управление
SIGSEGV	11	Завершение с дампом памяти	Нарушение при обращении в память	Исключение
SIGSTOP	23	Остановка процесса	Остановка выполнения процесса	Управление
SIGTERM	15	Завершение	Сигнал завершения (сигнал по умолчанию для утилиты kill)	Управление
SIGTSTP	20	Остановка процесса	Сигнал остановки с терминала (Ctrl-Z).	Управление
SIGTTIN	26	Остановка процесса	Попытка чтения с терминала фоновым процессом	Управление
SIGTTOU	27	Остановка процесса	Попытка записи на терминал фоновым процессом	Управление
SIGUSR1	16	Завершение	Пользовательский сигнал № 1	Пользовательский
SIGUSR2	17	Завершение	Пользовательский сигнал № 2	Пользовательский
SIGPOLL	22	Завершение	Событие, отслеживаемое `poll()`	Уведомление
SIGPROF	29	Завершение	Истечение таймера профилирования	Отладка
SIGSYS	12	Завершение с дампом памяти	Неправильный системный вызов	Исключение
SIGTRAP	5	Завершение с дампом памяти	Ловушка трассировки или брейкпоинт	Отладка
SIGURG	21	Игнорируется	На сокете получены срочные данные	Уведомление
SIGVTALRM	28	Завершение	Истечение «виртуального таймера»	Уведомление
SIGXCPU	30	Завершение с дампом памяти	Процесс превысил лимит процессорного времени	Исключение

Название	Код	Действие по умолчанию	Описание	Тип
SIGXFSZ	31	Завершение с дампом памяти	Процесс превысил допустимый размер файла	Исключение

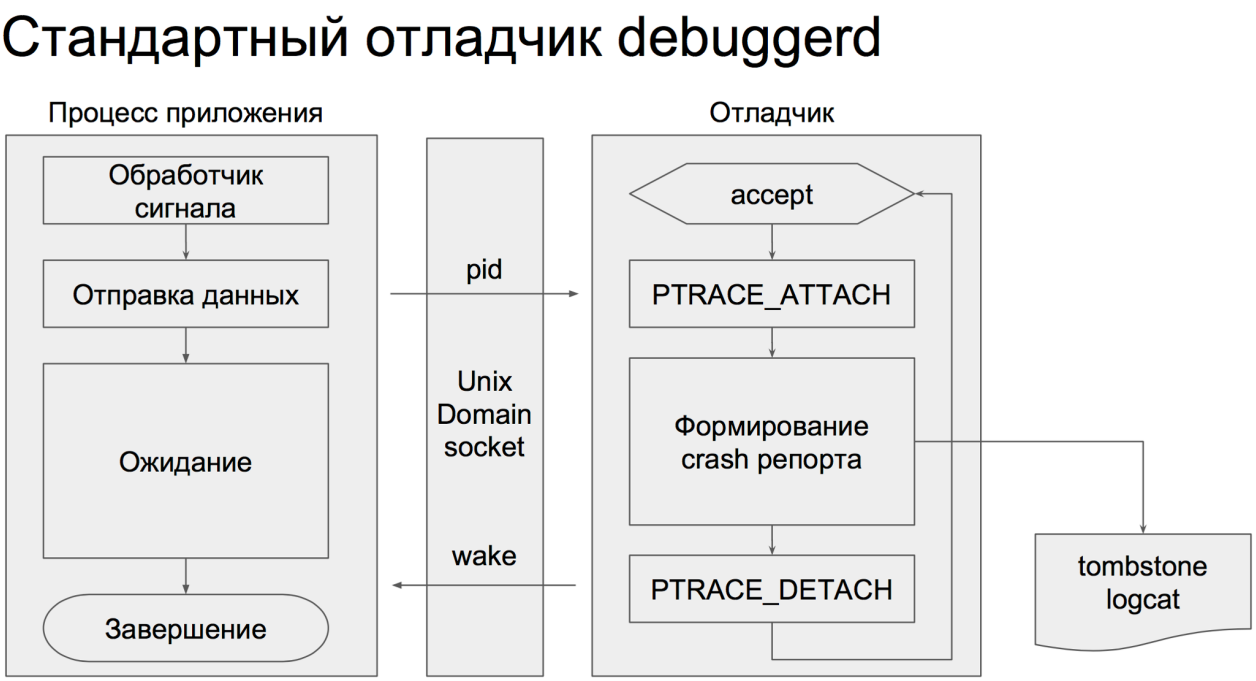
31. Основные этапы разработки ОС с пояснением каждого этапа

1. Настройка среды разработки
2. Генезис (зарождение)
3. Экран
4. Таблицы GDT и IDT
   Это служебные таблицы в памяти, хранящие дескрипторы сегментов
5. Запросы на прерывания IRQ и таймер PIT (с программируемым интервалом)
6. Страничная организация памяти
7. Память типа куча
8. Файловая система VFS и initrd
9. Многозадачность
10. Пользовательский режим

32. Стековый кадр



33. Cxxabi + стандартный отладчик



34. Планирование процессов

Исходя из трех основных состояний процесса «готов», «выполнение», «заблокирован». Планировщик должен знать, какой процесс находится в каком состоянии. Все усложняется, если ЦП содержит несколько вычислительных ядер. Поэтому в ОС вводятся различные очереди (списки) для планирования процессов.

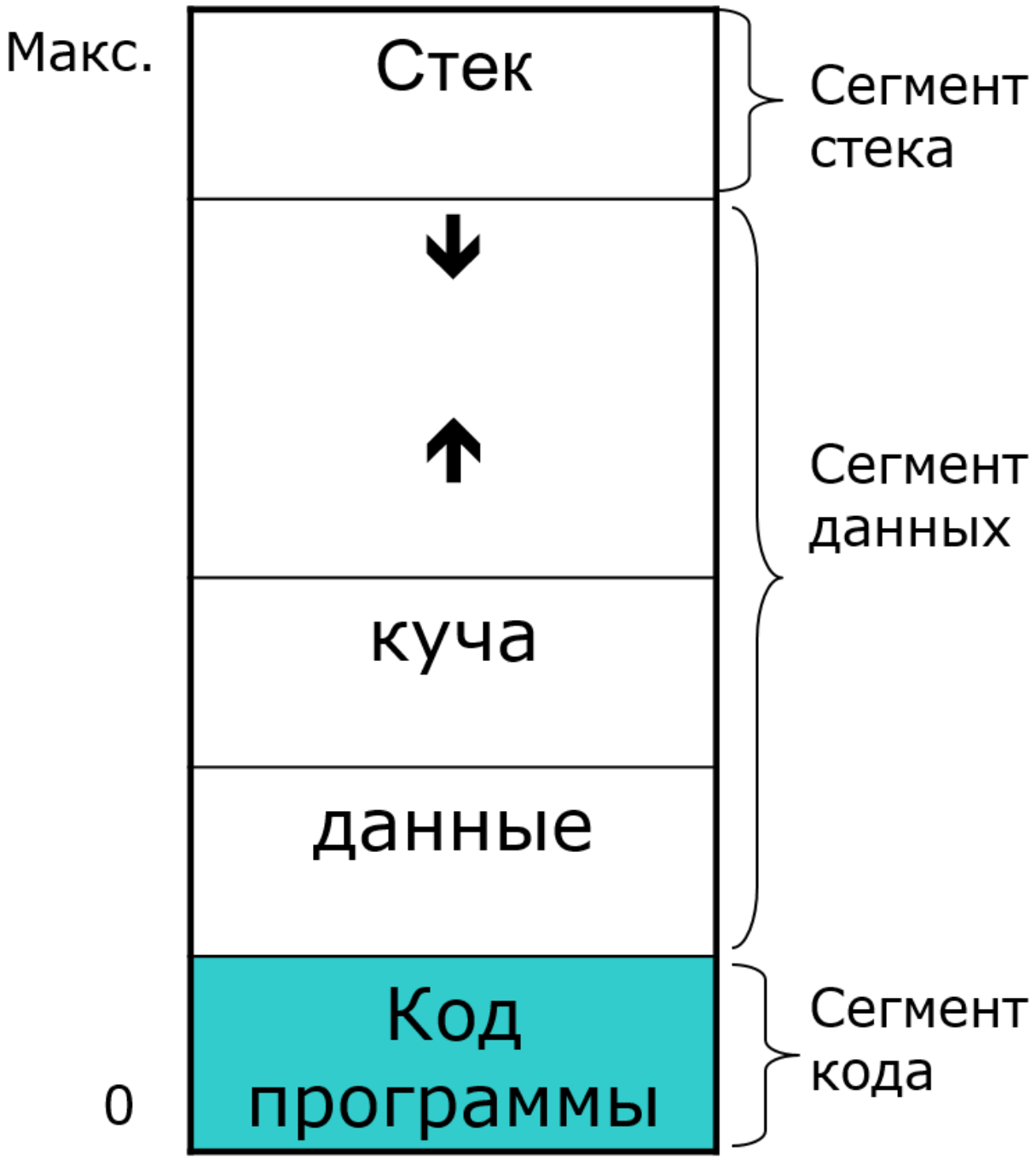
Исходя из трех состояний процесса вводятся 3 очереди:

1. Очередь задач: множество всех процессов, которые есть в системе
2. Очередь готовых: множество всех процессов, готовых для выполнения, им можно в любой момент дать квант процессорного времени и они будут выполняться.
3. Очередь ожидающих: множество всех заблокированных процессов.

35. Управление процессами

- создание процесса – завершение процесса ;
- приостановка процесса (перевод из состояния исполнение в состояние готовность) – запуск процесса (перевод из состоянияготовность в состояние исполнение);
- блокирование процесса (перевод из состояния исполнение в состояние ожидание) – разблокирование процесса (перевод из состояния ожидание в состояние готовность).

36. Процесс (физическое представление)



При запуске программы под процесс выделяется место в памяти. Адресное пространство, относящееся к процессу, делится на 3 части:

- Сегмент стека - используется для вызовов функций и системных вызовов
- Сегмента данных - переменные статические и динамические, выделяемые из кучи (все, что нужно для работы)
- Сегмент кода - код программы, обычно read-only

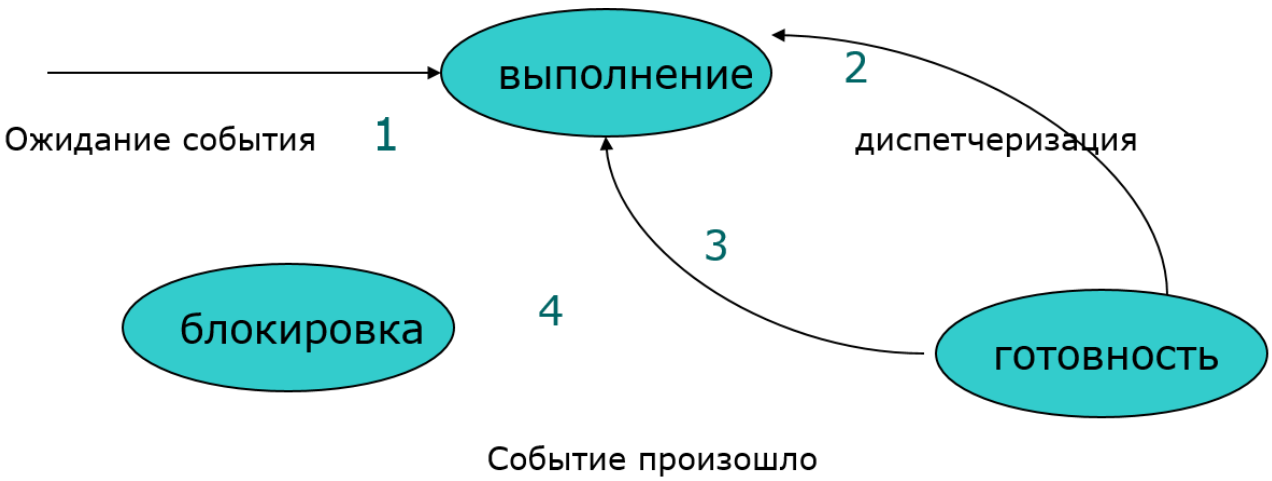
37. Структура управления процессами в ОС

- Таблица процессов
- Блок управления процессом - описывает свой процесс, которому он принадлежит, и его контекст
- Образ процесса - кусок памяти, выделенный для процесса

38. Записи таблицы процессов

Управление процессом	Управление памятью	Управление файлами
Регистры	Указатель на информацию о текстовом сегменте	Корневой каталог
Счетчик команд	Указатель на информацию о сегменте данных	Рабочий каталог
Слово состояния программы	Указатель на информацию о сегменте стека	Дескрипторы файлов
Указатель стека		Идентификатор пользователя
Состояние процесса		Идентификатор группы
Приоритет		
Параметры планирования		
Идентификатор процесса		
Родительский процесс		
Группа процесса		
Сигналы		
Время запуска процесса		
Использованное время процессора		
Время процессора, использованное дочерними процессами		
Время следующего аварийного сигнала		

39. Модель состояния процесса



40. Этапы создания процесса

1. Присвоение id
2. Выделение места
3. Инициализировать PCB (блок управления процессом)
4. Добавить процесс в очередь "готовых" к выполнению

41. Создание процесса ОС Unix

fork() - создает клон вызывающего процесса

exec() - создает клон вызывающего процесса и заменяет код на нужный процесс

42. Создание процесса ОС Windows

CreateProcess() - создает новый процес. Иерархии нет. Возвращает хэндл процесса.

43. Переключение контекста процесса

Старый процесс сохраняется в его PCB, новый загружается из его PCB

44. Параллелизм

Параллелизм – это физически одновременное выполнение для достижения наибольшей производительности(например, между двумя ядрами)

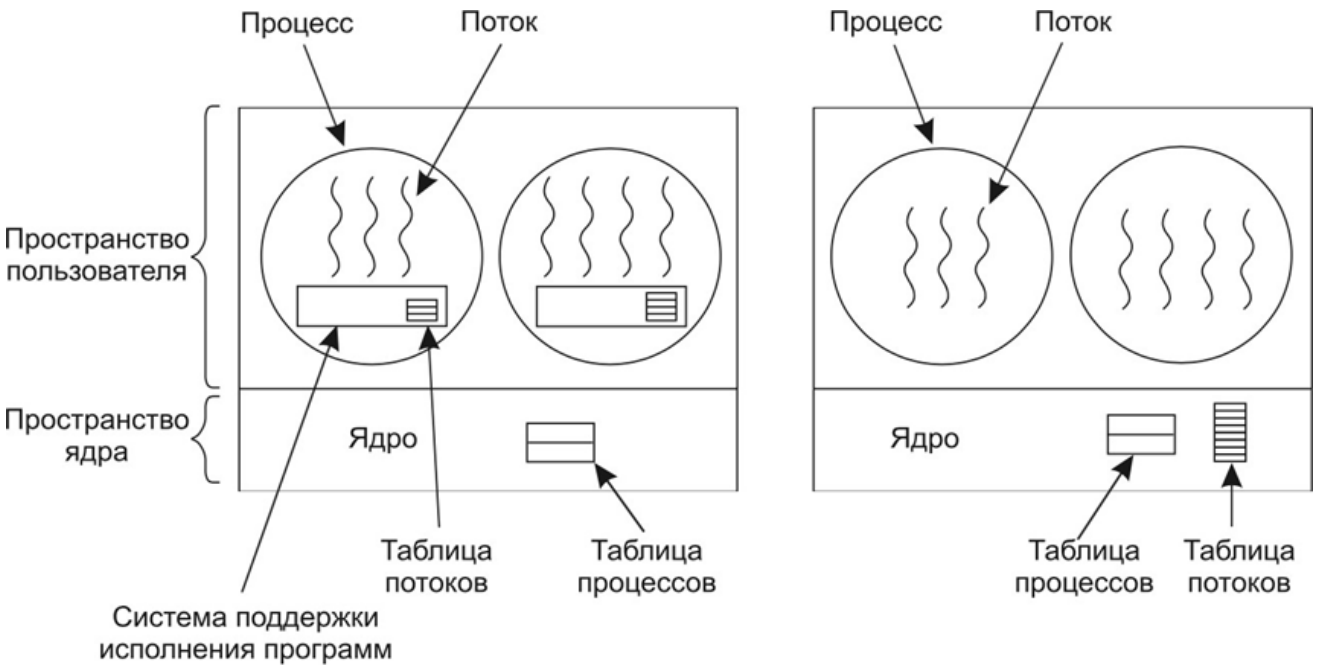
45. Потоки. Использование объектов потоками

Элементы, присущие каждому процессу	Элементы, присущие каждому потоку
Адресное пространство	Счетчик команд
Глобальные переменные	Регистры
Открытые файлы	Стек
Дочерние процессы	Состояние
Необработанные аварийные сигналы	
Сигналы и обработчики сигналов	
Учетная информация	

46. Отличие потоков от процессов

- Процессы незасисимы, а потоки - составные элементы процессов
- Процессы несут больше информации о состоянии, а потоки совместно используют информацию о состоянии
- Процессы имеют отдельное адресное пространство, а у потоков свое - только стек
- Процессы взаимодействуют только через предоставляемые системой механизмы связи
- Переключение контекста между потоками в процессе быстрее, чем переключение контекста между процессами

47. Потоки на пользовательском уровне, потоки, управляемые ядром



48. Иерархия процессов

В Unix есть иерархия процессов. Каждый процесс имеет родителя. Самый главный - init.

49. Память без использования абстракций. Свопинг. Уплотнение памяти.

Память без абстракций - программа просто видит физическую память. Реальные адреса. Работа двух программ невозможна. Хотя можно запилить многопоточную прогу, у них память одна.

Свопинг - сохранение всей проги на диск, и запуск другой проги. Замена данных. Так можно обеспечить работу двух программ без абстракций памяти.

Уплотнение памяти - перемещение всех программ, расположенных в разных местах в оперативке в нижние адреса.

50. Адресное пространство