Операционные системы

Курс лекций для гр. 4057/2

Лекция №1

И.В.Штурц

Содержание

Введение

- Предмет и цель курса
- Назначение и структура ОС
- Классификация режимов выполнения программ
- Классификация ОС по назначению

Раздел 1. Управление процессами

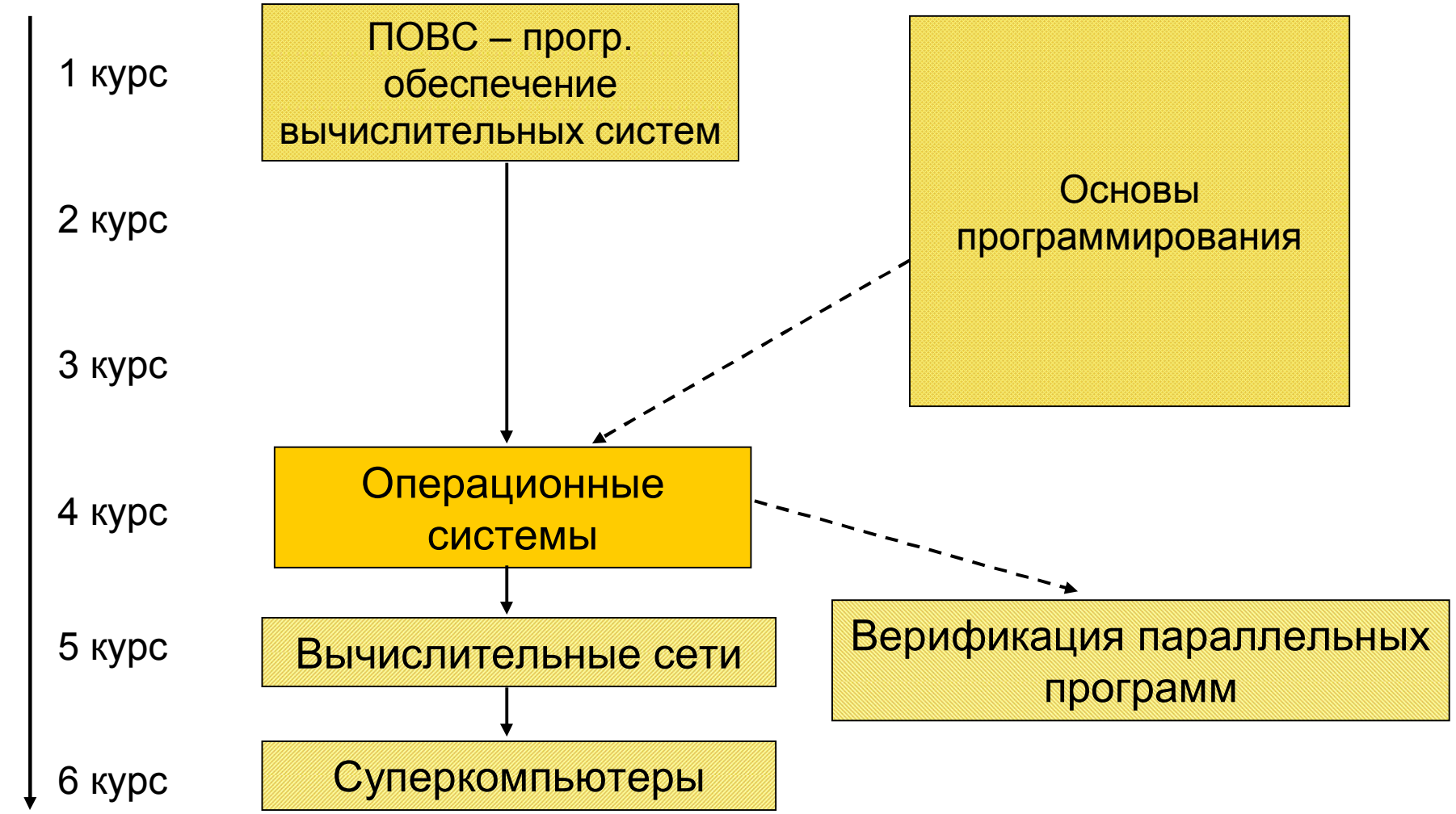
1.1 Понятие процесса и его контекста

1.2 Переключение процессов

Предмет и цель курса

- **Предмет курса:** общие принципы построения и функционирования ОС. Примеры – Windows 2000 (W2k), UNIX, Linux и некоторые другие
- **Цель:** знать принципы работы основной *системной* программы. Это необходимо *системным* программистам для:
  - для повышения производительности и надежности вычислительных систем
  - оптимизации приложений по времени и памяти
  - разработки системных программ (напр., драйверов) и аппаратно-зависимых частей прикладных программ
  - разработки параллельных программ для многоядерных процессоров и распределенных систем

Связь с другими учебными курсами



Разделы курса лекций

- в соответствии с видами ресурсов и функциями ОС:

- Управление процессами
- Управление памятью
- Ввод-вывод
- Файловая система
- Распределенные системы
- Отказоустойчивость
- Защита

Лабораторные работы – Linux:

- Командный язык
- Администрирование
- Системное программирование

Ауд. 2 кафедры, среда 16-18, Тимофеев Дмитрий Андреевич

Рекомендуемая литература

1. К.А. Коньков, В.Е. Карпов. Основы операционных систем. http://www.intuit.ru/department/os/osintro/
2. К.А. Маслинский, Г.В.Курячий. ОС Linux. http://www.intuit.ru/department/os/linux/
3. Э.Таненбаум. Современные операционные системы. – «Питер», 2004.
4. Д.Бэкон, Т.Харрис. Операционные системы. Параллельные и распределенные системы. БХВ, «Питер», 2004.
5. А.Робачевский. ОС Unix. БХВ-СПб, 1999 (или другой учебник для программистов по Unix).
6. Д.Соломон, М.Руссинович. Внутреннее устройство Windows 2000. – «Питер», 2001.

Что такое ОС ?

Несколько определений

- Интерфейс между пользователем и вычислительной аппаратурой
- Программный слой, предоставляющий возможность выполнения всех других программ
- Совокупность программ, управляющих вычислительным процессом в системе (в компьютере, многопроцессорном *кластере* или в сети)

Цели ОС

- сделать использование компьютера

- *Удобным*
  - Для конечного пользователя
    - Дружественный UI (User Interface)
    - Безопасность
  - Для программиста
    - Упростить программирование вв/вы и сетевого обмена
    - Упростить *коммунальное* использование аппаратных ресурсов
    - Предоставить функции синхронизации и коммуникации процессов
- *Эффективным* в смысле максимальной:
  - производительности
    - оценивается количеством программ, выполняемых под управлением ОС в единицу времени
  - реактивности (быстродействия)
    - оценивается временем реакции на запрос о выполнении программы

Ключевые термины

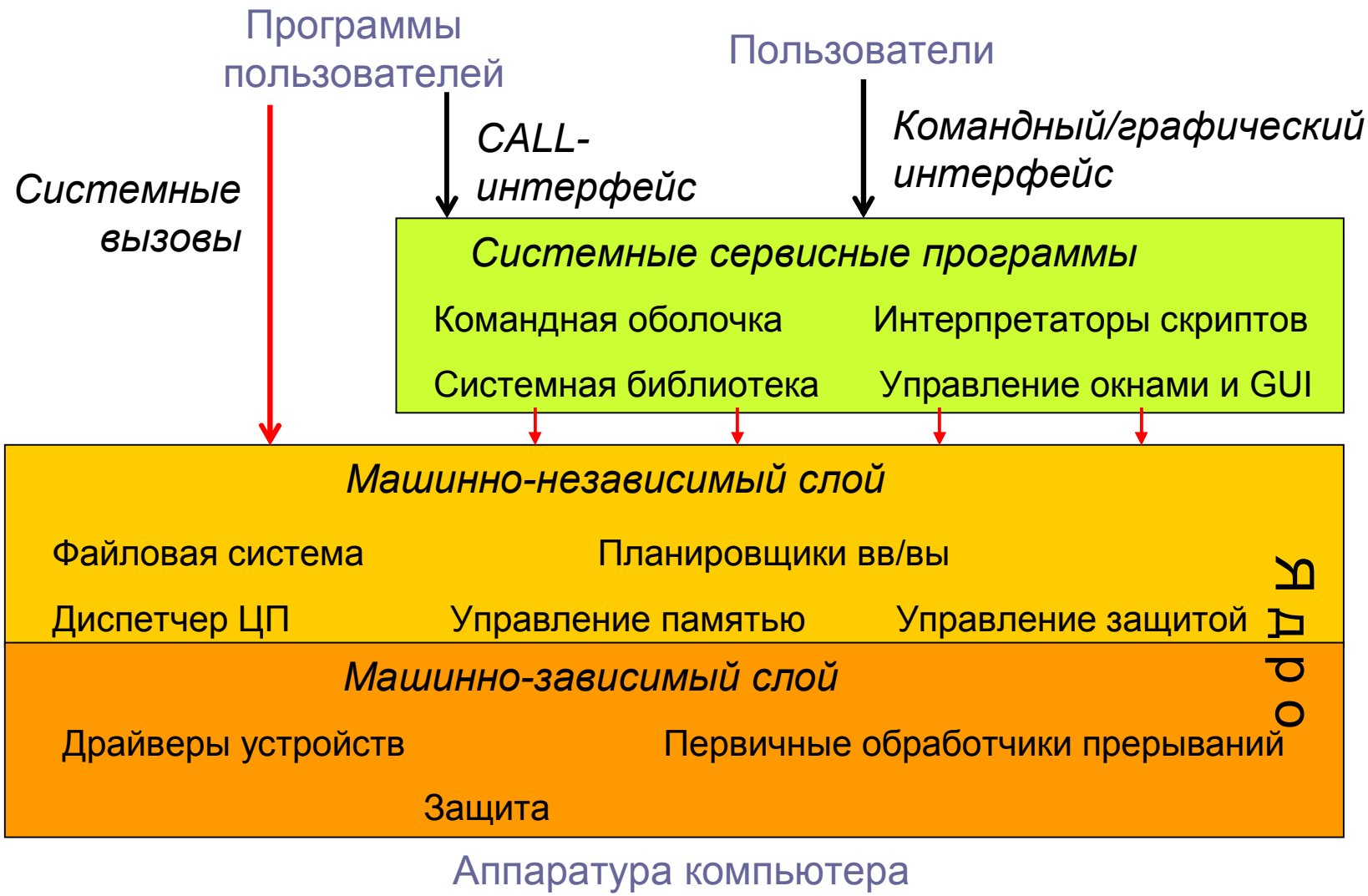
- Субъекты управления: пользователи (люди) и прикладные программы (процессы)
- Объекты управления - *ресурсы* вычислительной системы:
  - аппаратура - центральный процессор (ЦП), память и внешние устройства (диск и устр. вв/вы)
  - выполняемые программы (процессы)
  - данные (файлы)
- Функции управления:
  - Распределение ресурсов между субъектами
  - Защита ресурсов при их коммунальном использовании
  - Интерфейс для заказываемых действий:
    - Для пользователей – командный язык (в UNIX – Shell) или GUI, как в Windows
    - Для программ - набор *системных вызовов,* задействующих *функции API* (напр., Win32 API) и вызовов библиотечных программ *системных сервисов*

Понятие ядра ОС (kernel)

- *Системный вызов* начинается с программного прерывания (trap), переводящего ЦП в *режим ядра*
- *Режим ядра* дает б**о**льшие возможности по доступу к аппаратуре (память, вв/вы, защита и пр.) по сравнению с пользовательским режимом
- *Ядро ОС* – это ее основная часть, резидентная в памяти, выполняющая базовые функции управления вычислительным процессом
- Остальные программы ОС (утилиты, или сервисные) имеют статус обычных приложений и выполняются в области памяти пользователя

Мы будем изучать в основном ядро, т.е. ОС в узком смысле слова

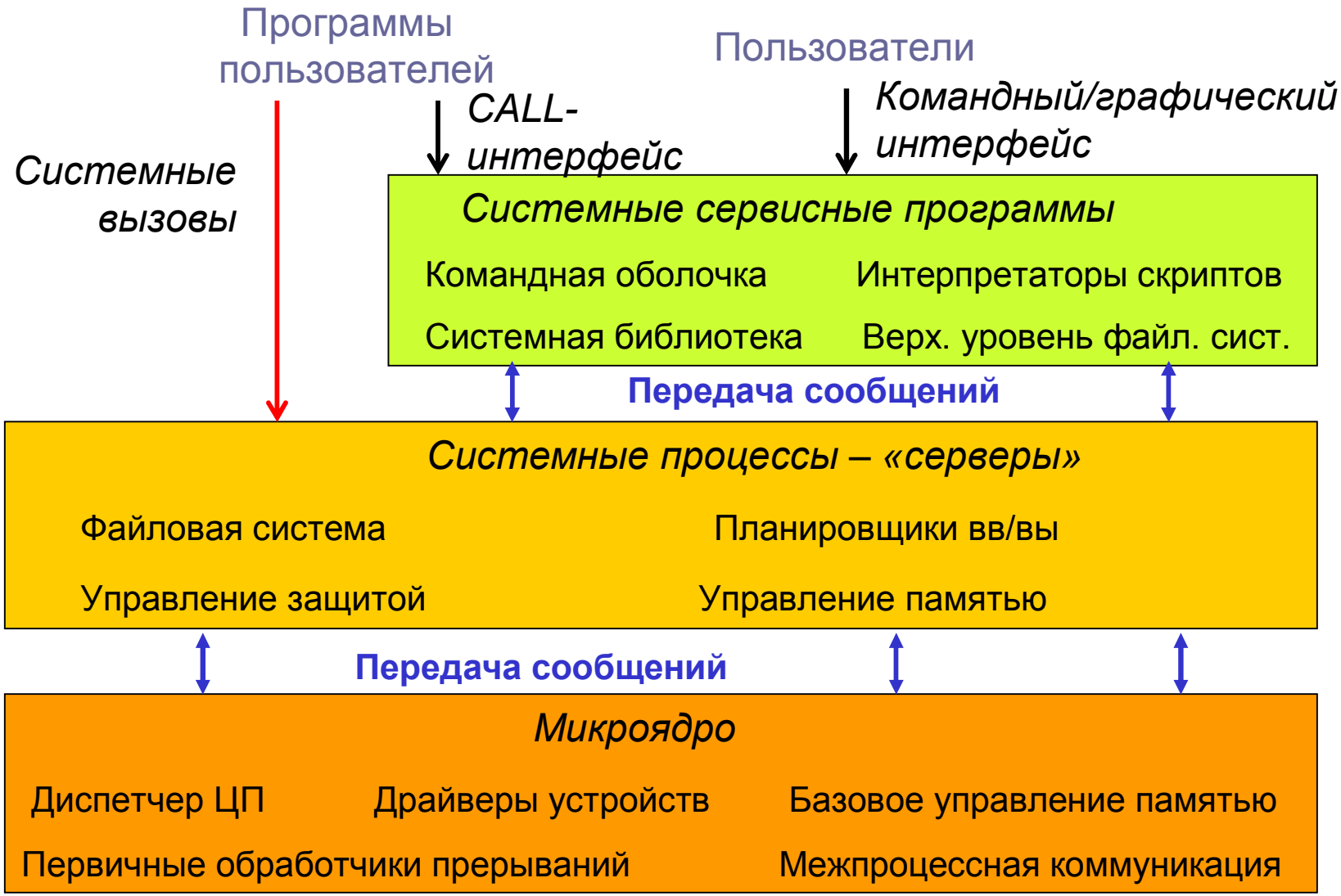
Монолитная ОС = ядро



Монолитная структура ОС

- Старейший способ организации операционных систем; пример - Unix
- Все компоненты ОС используют единое адресное пространство и взаимодействуют друг с другом путем непосредственного вызова процедур
- Сборка ядра - его компиляция - осуществляется отдельно для каждой конфигурации компьютера
- Вырожденные случаи: нет привилегированного режима ядра
  - ОС для встроенных (embedded) систем
    - Простой процессор
    - Программы известны и отлажены - «без неожиданностей»
  - Надстройки над ОС, напр. JVM – Java Virtual Machine

Структура микроядерной ОС



Микроядерная ОС

- Достоинства по сравнению с монолитной ОС
  - Гибкость благодаря высокой степени модульности: упрощается добавление новых компонентов «на ходу»
  - Повышается надежность системы, поскольку ошибка на уровне непривилегированной программы менее опасна, чем на уровне режима ядра
- Недостаток - уменьшается производительность и реактивность из-за передачи сообщений. Поэтому у современных ОС - *смешанные* архитектуры:
  - Unix BSD и MkLinux: монолитная ОС поверх микроядра Mach с минимальной функциональностью
  - Linux – монолитная ОС, где разрешается динамическая загрузка и выгрузка многих компонентов ядра – т. наз. модулей ОС
  - Windows: компоненты ядра
    - как в микроядерной ОС, взаимодействуют друг с другом путем передачи сообщений
    - как в монолитной ОС, работают в одном адресном пространстве и используют общие структуры данных

Классификация режимов выполнения программ

- история компромиссов между производительностью и реактивностью

- **Однопрограммный**: любая программа занимает все ресурсы компьютера до ее завершения – нормального или аварийного
  - 1.1 **Индивидуальный**: единственный пользователь управляет процессом выполнения программы –> потери времени при остановках
  - 1.2 **Пакетный**: *безостановочное* выполнение последовательности программ – *пакета* (batch), загружаемого оператором; пользователи не управляют процессами
- 2. **Мультипрограммный (многозадачный)**: несколько программ выполняются параллельно, т.е. совместно используют ресурсы в пространстве (память) и во времени (ЦП и УВв/Вы)
  - 2.1 **Пакетный**: программы пакета поочередно занимают ЦП и Увв/Вы, и поэтому вычисления совмещаются во времени с вв/вы
  - 2.2 **Разделения времени**: несколько пользователей управляют своими процессами с терминалов; время процессора предоставляется им поочередно, порциями – *квантами* (time slices)
  - 2.3 **Реального времени**: активность процессов синхронизируется внешними событиями; реактивность важнее производительности
  - 2.4 **Индивидуальный**: в персональных компьютерах

Общая характеристика W2k (Windows 2000)

- Универсальная, масштабируемая, мультипрограммная
- С модифицированной архитектурой микроядра: многие функции, не входящие в микроядро, выполняются в режиме ядра – для большей эффективности (поэтому системные вызовы неотличимы от вызовов сервисов)
- Объектно-ориентированная: ее ресурсы – это объекты с классами, инкапсуляцией и ограниченными наследованием и полиморфизмом
- Имеет опубликованный интерфейс Win32 API – тысячи вызовов библиотечных процедур, которые
  - либо обращаются к системным вызовам (непубликуемым)
  - либо выполняются в области памяти пользователя
- Внутренняя структура – по модели «клиент/сервер»: все служебные подсистемы обслуживают запросы (сообщения) клиентов (приложений или других модулей ОС), что упрощает структуру ОС, повышает ее надежность и масштабируемость и служит основой для распределенных вычислений

NB: более современные Win XP, Vista и Windows 7 основаны на усовершенствованном программном ядре W2k

Общая характеристика UNIX

- Универсальная, с монолитным ядром
- Развивается с 1971 г.; первая ОС, написанная на языке высокого уровня, поэтому переносимая и читабельная
- Надежность, переносимость, ясность лучше, чем у Windows
- Имеет богатый набор утилит (служебных программ) и развитый командный язык (sh, bash, csh, …).
- Графический оконный интерфейс – прикладная программа X/Window
- Наиболее важные современные версии:
  - SVR4 (System V Release 4) – наиболее универсальная коммерческая версия
  - Free BSD (Berkley Software Distribution) – бесплатная, широко применяется в академических организациях и университетах
  - POSIX - стандарт IEEE 1003.1, пересечение функциональностей System V и BSD
  - Solaris 2.x – для компьютеров Sun
  - Open Solaris – для PC

Общая характеристика Linux

- Первоначально (1991) - вариант UNIX для IBM PC, сейчас для многих платформ, в т.ч. для встроенных систем
- Соответствует стандарту POSIX; 80% системных вызовов совпадают; отличия от UNIX - в алгоритме планирования и др. деталях
- Распространяется по лицензии GPL (Debian), с открытым кодом (open source); развивается программистским сообществом на добровольных началах
- Организована в виде набора *загружаемых модулей*, динамически связываемых с ядром и удаляемых из памяти в любой момент времени
- Графический оконный интерфейс – X/Window, GNOME и KDE
- Наследует достоинства UNIX, но лучше масштабируема

Классификация популярных ОС по назначению

- Общего назначения: для настольных систем и ноутбуков
  Windows, Linux
- Для сетевых серверов и мэйнфреймов
  Windows Server, Unix, Linux
- Для мобильных устройств (смартфоны, КПК (PDA), ...)
  Windows Mobile, Symbian, Android, BADA, JVM
- Для встроенных систем реального времени
  Windows CE, Linux (Mandriva), QNX, VxWorks, RTOS, P-SOS
- Для облачных вычислений
  Android, Google Chrome OS (обе – с Linux ядром)

Раздел 1

Управление процессами

Понятия процесса и его контекста

Процесс (или задача - task) - это

- Совокупность явлений, связанных с выполнением программы на компьютере
- Минимальная единица работы в ОС, для которой выделяются системные ресурсы

  Работа - это выполнение:
  - прикладной программы
  - или системной программы: командной оболочки, сетевого обмена и т.д.
  - или скрипта (кода на языке сценариев)

- Процесс – это последовательность смены дискретных состояний
  - Микросостояния характеризуются значениями всех элементов памяти в компьютере и изменяются с каждым тактом ЦП
  - Более обобщенные состояния: после выполнения каждой команды
    - Это состояние приходится запоминать при прерывании программы, чтобы позже возобновить ее выполнение с точки прерывания (всегда с дискретностью до команды!)
    - Такое состояние процесса, или его *контекст*, складывается из двух составляющих:
      - Состояние процессора (неразделяемого ресурса) – аппаратный контекст
      - Состояние части совместно используемых ресурсов, принадлежащих процессу: памяти, открытых файлов и т.п. – программный контекст

Дескриптор процесса

Контекст процесса хранится в служебной структуре данных ОС – *дескрипторе* или *управляющем блоке* (PCB) процесса. Его типичный состав:

- Учетные и диспетчерские данные
  - идентификатор (в UNIX - proc_id, в W2k - UniqueProcessId)
  - идентификатор родительского процесса
  - владелец, пользователь
  - время создания
  - приоритет
  - диспетчерское состояние
- Аппаратный контекст
  - слово состояния программы - PSW
  - содержимое общих регистров
  - состояние памяти – таблица отображения виртуальных адресов на физические
- Программный контекст – локальная операционная среда
  - текущий каталог
  - список открытых файлов и других доступных процессу ресурсов
  - назначение логических Увв/вы - физическим
  - командная строка
  - переменные окружения

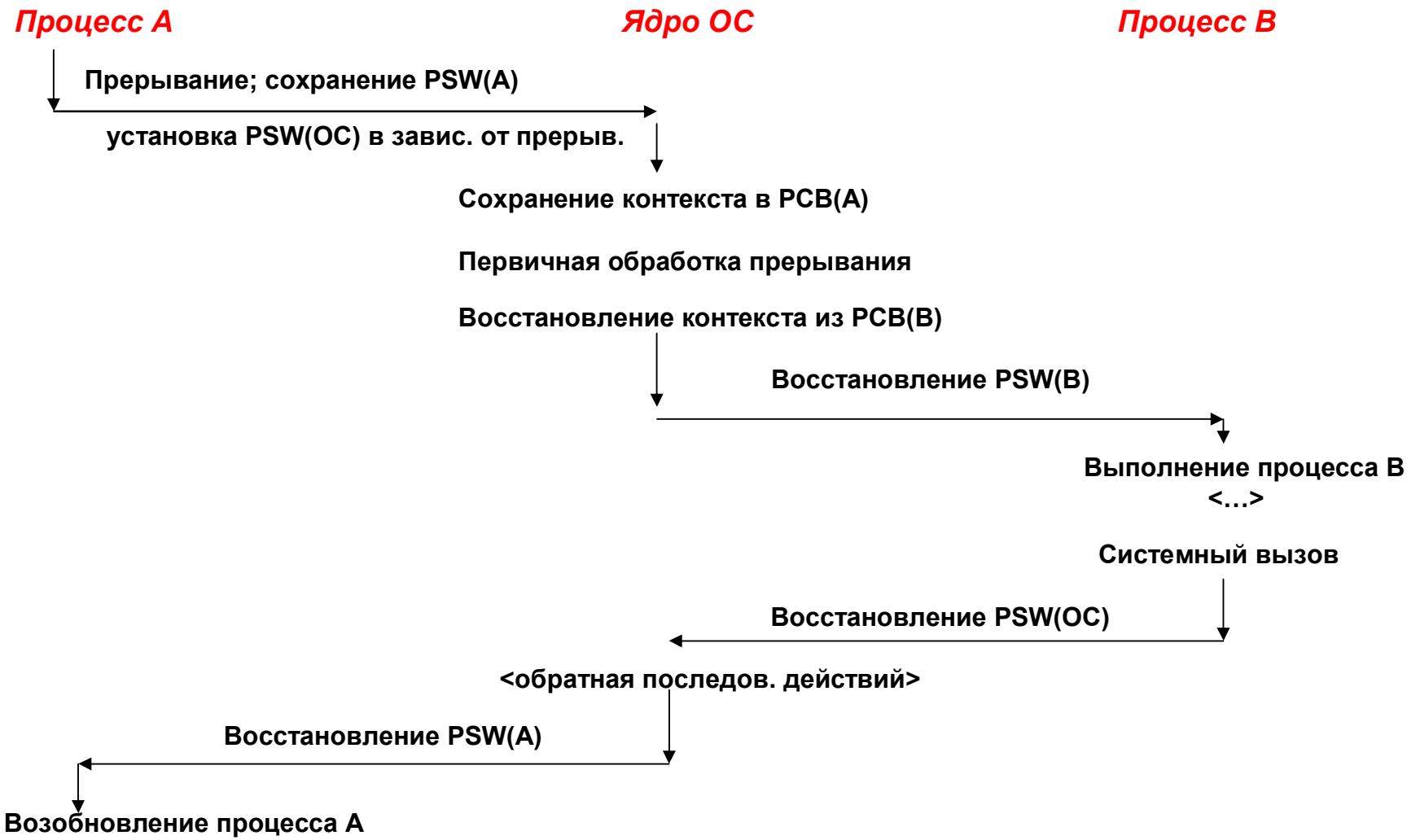
Аппаратный контекст

- В слове состояния программы - PSW (Program Status Word) хранится содержимое специальных регистров и триггеров процессора:
  - счетчик команд
  - флаги условий
  - флаги режима
  - маски прерываний

  - минимально необходимая информация для возобновления процесса с точки прерывания (всего в I80x86 – 6 байт). Во время прерывания PSW сохраняется в системном управляющем стеке

- Таблица отображения виртуальных адресов процесса на физические адреса памяти индивидуальна для каждого процесса и хранится таблице страниц (или сегментов)

Схема переключения процессов



Переключение контекста

- происходит в два этапа:

- Аппаратная смена PSW – производится автоматически в момент *прерывания*
- Программная смена (сохранение/восстановление) остального контекста – производится первичным обработчиком прерывания

- Эта операция - *переключение* или *перестановка контекста*; время ее выполнения определяет реактивность и производительность ОС совместно с аппаратурой
- Это время велико (10^3-10^5 машинных тактов): нужно сохранить и загрузить регистры, обновить таблицы и списки
- Кэш памяти начинает заполняться командами и данными новой программы, что сильно снижает производительность
- Переключение контекста может производиться тысячи раз в сек.

Первичная обработка прерываний

Два вида событий, приводящих к прерываниям:

- *асинхронное* – может произойти в любой момент; генерируется аппаратно Увв/вы и таймерами) (в W2k называется просто прерыванием (interrupt))
- *синхронное* (в W2k – *исключение* (exception)) генерируется программно:
  - ошибочная ситуация при выполнении конкретной команды (например, «деление на ноль», «защита памяти» или «отсутствие страницы»)
  - системный вызов (в W2k - вызов функции Win32 API или системного сервиса)
- Первичный обработчик прерываний (в W2k – *ловушка* (trap)) передает управление определенной части ОС. Тип (номер) прерывания используется как индекс в *таблице диспетчеризации прерываний* (interrupt dispatch table, IDT), и управление передается соответствующей *процедуре обслуживания прерывания* (interrupt service routine, ISR)
- Прерывания с более высоким приоритетом вытесняют обработку прерываний с меньшим приоритетом, причем последние *маскируются* (временно запрещаются
- Системный вызов в процессоре любой архитектуры начинается с выполнения специальной команды (в Intel x86 это *int 0x2e*); вид системного сервиса задается параметрами, передаваемыми через стек

Резюме

- ОС – программа управления вычислительным процессом
- В узком смысле термина ОС = ядро
- Удобство, модульность, производительность, реактивность и надежность – главные критерии качества ОС
- История развития ОС и усложнения режимов выполнения программ – история компромиссов между производительностью и реактивностью
- В сфере встроенных систем разнообразие ОС больше, чем для компьютеров общего назначения, где доминируют Windows, Unix и Linux
- Процесс – основная единица работы в ОС
- Контекст процесса – информация, необходимая для его возобновления после приостановки. Хранится в дескрипторе процесса вместе с учетными данными
- Время переключения контекста сильно влияет на производительность и реактивность ОС
- Синхронные прерывания возникают при выполнении определенных команд программы, асинхронные – в произвольные моменты времени

Вопросы к лекции 1

1. Почему однопрограммный пакетный режим – более производительный, чем индивидуальный?
2. Какой добавочный резерв производительности задействует пакетный мультипрограммный режим по сравнению с пакетным однопрограммным?
3. Какой из режимов обеспечивает самую высокую производительность? Реактивность?
4. В вычислительных центрах существует многопользовательский режим *удаленного ввода заданий (Remote Job Entry, RJE)*, промежуточный между 2.1 и 2.2. Он эффективен, когда несколько программистов работают в цикле подготовка/отладка программ. Охарактеризуйте его.
5. Как размер кванта времени в режиме разделения времени влияет на реактивность? На производительность?
6. В чем преимущества однопользовательского мультипрограммирования W2k по сравнению с однопрограммным режимом, единственно возможным в MS-DOS ?
7. В Win2k UniqueProcessId (сокр. PID) – целые, кратные 4. Зачем?
8. Значительную долю времени переключения контекста занимает перегрузка содержимого общих регистров в PCB и обратно. Как можно этого избежать - ценой увеличения объема аппаратуры процессора (что сделано, в частности, в архитектуре RISC–процессоров Sun Sparc) ?