Intel Studio: Операционные Системы

Лекция 1 – Введение

Александр Мощук
Алексей Линёв
Кирилл Погорельский

somes slides are adapted from the OS course at the University of Washington

Добро Пожаловать

- Обзор курса
  - Преподаватели
  - Структура курса
  - Учебные материалы
  - Правила прохождения курса
  - Требования к слушателям
- Обзор операционных систем
  - Функции ОС
  - Эволюция ОС

Web-страница

- Все необходимые материалы вы можете найти на http://www.nnstudio.ru/courses/2006su/
  - Расписание
  - Задания
  - Презентации с лекций
  - Полезные ссылки
  - Форум для вопросов

Преподавательский состав

- Алексей Линёв (liniov@nnstudio.ru)
- Александр Мощук (moshchuk@nnstudio.ru)
- Кирилл Погорельский (pogorelskiy@nnstudio.ru)

О курсе

- Первый курс в программе Intel Studio
- Мы познакомимся с основными понятиями ОС
  - Организация, планирование, параллельность и синхронизация, управление памятью, файловые системы, ...
- Цель – понять, не только как пользоваться ОС, но и как *написать* ОС
  - Большая практическая нагрузка

Учебный процесс

- Лекции – три раза в неделю
  - На русском, английская терминология
  - Интерактивные
- Семинары – 1-2 раза в неделю
  - Объяснение заданий, решений
  - Практические темы
  - Интерактивные
- Задания
  - Письменные домашние задания
  - Проекты (лабораторные задания)

Проекты

- Лучший способ узнать, как работает ОС – написать ОС самому
- 6 проектов, покрывающих основные понятия и компоненты ОС
  - Разминка по C
  - Системные вызовы и шеллы
  - Потоки / синхронизация
  - Виртуальная память
  - Файловые системы
  - Безопасность
- Интересные
- Трудные
  - Начинать заранее!
- Первые два – индивидуальные

Ресурсы

- Персональные компьютеры
  - Новые, быстрые
  - Linux (Fedora Core 4)
  - Доступ в интернет
- Учебник
  - *Современные Операционные Системы* (А. Таненбаум)
  - Дополняет, но не заменяет лекции
  - Читайте материалы перед занятиями
- Другие источники
  - множество источников в Internet; чтение ряда из них является обязательным, некоторых других - рекомендуемым

Оценки

- 50% - проекты
  - автоматические тесты
  - дизайн/качество кода
- 30% - письменные тесты
  - 10% - промежуточный
  - 20% - финальный
- 20% - домашняя работа

Правила

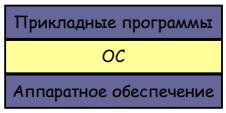
- Сотрудничество
  - Списывание запрещается
- Сдача
  - Домашние задания (в начале лекций)
  - Проекты (электронно)
- Сроки
  - Сдача заданий позже срока: 0 баллов (строго)
  - Один пропуск для сдачи на день позже

Ваши обязанности

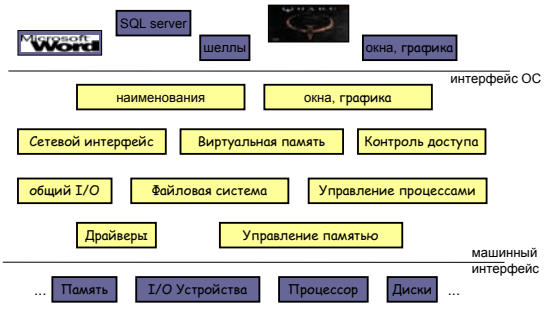
- Вовремя выполнять и сдавать задания
- Посещать все лекции и семинары
- Участвовать в обсуждениях
- Регулярно проверять электронную почту
- Во многом должны будете разбираться сами
  - изучать исходный код
  - google
  - ресурсы на web-странице

Что такое Операционная Система?

- Операционная система (ОС):
  - Программное обеспечение, управляющее аппаратными ресурсами и предоставляющее прикладным программам удобные абстракции
  - Набор утилит, упрощающих процесс разработки ПО
  - “Весь тот код, который вы не написали,” при разработке своего приложения



Из чего состоит ОС



ОС и аппаратные средства

- ОС является *посредником* при выделении приложениям аппаратных ресурсов
  - Вычислительные ресурсы (центральный процессор, ЦП)
  - Оперативная память и устройства долговременного хранения данных (жесткий диск и пр.)
  - Сетевая связь (протоколы TCP/IP, сетевые карты и т.д.)
  - Устройства ввода/вывода (клавиатура, монитор, звуковая карта...)
- ОС выполняет *абстрагирование* аппаратного обеспечения до уровня *логических ресурсов* с четко определенными интерфейсами
  - Процессы (ЦП, память)
  - Файлы (диск)
    - программы (последовательности инструкций)
  - Сокеты / sockets (сетевые соединения)

Зачем нужна ОС?

- Прикладным программам:
  - *упрощение* программирования
    - Использование высокоуровневых абстракций (напр. файлов) вместо низкоуровневых подробностей (регистров устройств и т.д.)
    - Абстракции могут быть *повторно использованы* во многих программах
  - *переносимость* (между системами с различными конфигурациями и архитектурой) – *portability*
    - Независимость от конкретных устройств (сетевая карта 3Com или Intel?)

Зачем нужны ОС?

- Пользователям:
  - *безопасность*
    - Программы “видят” только собственную “виртуальную машину”, думают что им предоставлен отдельный компьютер
    - ОС *защищает* программы друг от друга
    - ОС *справедливо распределяет* ресурсы между программами
  - *эффективность* (стоимость и производительность)
    - *совместное использование* одного компьютера несколькими пользователями
    - *параллельное* исполнение нескольких программ

Основные вопросы

- **Структура (structure)**
  - как построена ОС?
- **Совместное использование (sharing)**
  - как ресурсы распределяются между пользователями?
- **Именование (naming)**
  - какие имена имеют ресурсы (с точки зрения программ/пользователей)?
- **Безопасность (security)**
  - как обеспечивается целостность ОС и ресурсов?
- **Защита (protection)**
  - как пользователи/программы защищены друг от друга?
- **Производительность (performance)**
  - как сделать так, чтобы все работало быстро?
- **Надежность (reliability)**
  - каковы последствия сбоя (в аппаратном или программном обеспечении)?
- **Расширяемость (extensibility)**
  - можем ли мы добавлять новые возможности?

Основные вопросы - 2

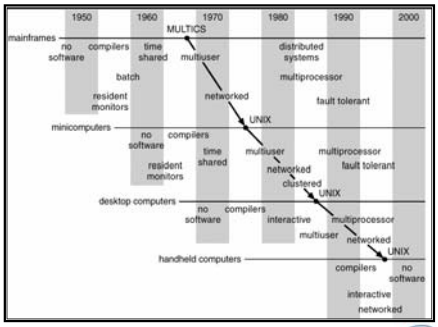
- **Гибкость (flexibility)**
  - мы готовы к поддержке новых приложений?
- **Взаимодействие (communication)**
  - как программы обмениваются данными локально и через сеть?
- **Параллельность (concurrency)**
  - как параллельная активность (вычислительная и ввод/вывод) организуется и управляется?
- **Масштабируемость (scalability)**
  - что произойдет при увеличении требований и ресурсов?
- **Сохраняемость (persistence)**
  - как сохранять данные между запусками программы?
- **Распределенность (distribution)**
  - Как несколько компьютеров взаимодействуют между собой?
- **Аудит (accounting)**
  - как учитывать использование ресурсов и, возможно, начислять плату за их использование?

Нет верных и неверных решений,
все решения – компромиссы

Отвечают ли требованиям существующие ОС?

Отвечают ли требованиям существующие ОС?

Эволюция идей и компьютеров



Тенденции

- «Онтогонез повторяет филогонез»
- «Тот, кто не помнит прошлого – обречен на его повторение»
- Возникают новые проблемы и задачи, изменяются уже известные
  - Эволюция ПК повторяет эволюцию миникомпьютеров, которая повторяет эволюцию мэйнфреймов
  - Повсеместное использование ПК повышает значимость защищенности и безопасности

Защищенность и безопасность: примеры

- отсутствует
- ОС от моих программ
- моих программ от программ других пользователей
- от доступа пользователей-злоумышленников
- от доступа программ-злоумышленников
- отказ в обслуживании (denial of service)
- распределенный вариант отказ в обслуживании (DDoS)
- подмена адреса (MAC или IP) (spoofing)
- spam
- черви (worms)
- вирусы
- ПО, которое вы загрузили и выполнили (ошибки, трояны)
- ПО, которое было загружено и выполнено без вашего ведома (cookies, spyware)

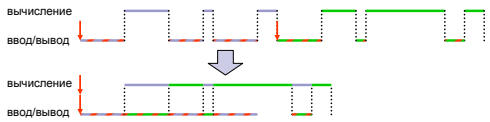
Эволюция ОС

- В самом начале...
  - ОС была просто библиотекой функций
    - подключалась к вашей программе
  - Программа загружалась в память целиком и выполнялась
  - Интерфейс пользователя - набор переключателей и мигающих ламп
- Затем появились системы *пакетной обработки* (*batch systems*)
  - ОС занимала часть оперативной памяти
  - ОС загружала очередную задачу в память с устройства чтения перфокарт
    - задача выполнялась
    - печатался вывод и дамп памяти (memory dump) – зачем?
    - процесс повторялся
  - устройства чтения перфокарт и принтеры были очень медленными
    - центральный процессор подолгу простаивал (потеря времени и $$)

Spooling («подкачка»)

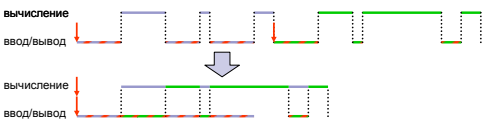
- Диски были гораздо быстрее перфокарт и принтеров
- Spool (Simultaneous Peripheral Operations On-Line)
  - Пока одна задача выполняется, можно загрузить следующею задачу с устройства чтения перфокарт на диск
    - медленное чтение перфокарт выполняется параллельно с вычислениями ЦП
  - можно даже загрузить на диск несколько задач
    - ОС должна определять, какую задачу выполнить следующей
    - *планирование задач*
  - однако, ЦП все еще бездействует в моменты обращения программ к периферийным устройствам

Многозадачность (Multiprogramming)



- Увеличение *эффективности использования* ЭВМ
  - Несколько готовых к выполнению задач держатся в памяти
  - Ввод-вывод одной задачи производится параллельно с вычислениями другой
    - Пока одна задача ждет завершения ввода/вывода, ОС запускает на ЦП выполнение инструкций другой задачи

Многозадачность (Multiprogramming)



- Цель: улучшение *пропускной способности (throughput)*
  - возможно, за счет *времени отклика (response time)*
- Для получения ускорения, нужны *асинхронные* устройства ввода/вывода
  - Нужен механизм, сообщающий о завершении операций I/O
    - прерывания (interrupts)
    - циклический опрос (polling)

Разделение времени
Timesharing

- Вариант многозадачности
- Создан для интерактивной работы
  - Несколько терминалов у одной ЭВМ
  - каждый пользователь считает, что ему полностью выделена вычислительная система
- Цель: улучшение времени отклика
  - возможно за счет пропускной способности
- Time-slicing (разделение времени ЦП)
  - Разделение процессорного времени поровну между пользователями
  - Для интерактивных задач (напр. редакторов), возможно переключаться между программами и пользователями быстрее, чем создается загрузка для системы

ОС разделения времени

- CTSS (1961) – одна из самых первых систем, поддерживавших режим разделения времени
- MULTICS (1968) – первая “большая” ОС разделения времени
  - в Multics могут быть найдены зачатки многих концепций теории ОС
  - “синдром второй системы”

Персональные компьютеры

- Аппартные средства стали достаточно дешевыми, чтобы каждый мог владеть своим ПК
  - Xerox Alto (1972) – первый ПК
- Влияние на ОС?


Xerox Alto (1972)


IBM PC (1981)

Параллельные ОС

- Некоторые приложения могут выполняться в виде нескольких параллельно запущенных *процессов* или *потоков*
  - ускорение при одновременном выполнении нескольких потоков/процессов на нескольких ЦП
  - необходима поддержка разделения программы на параллельные компоненты со стороны ОС и языка программирования
  - необходимо наличие в ОС поддержки взаимодействия параллельно выполняющихся компонент
- Много разновидностей параллельных ЭВМ:
  - SMPs (симметричные мультипроцессоры)
  - MPPs (massively parallel processors – сотни/тысячи)
  - NOWs (networks of workstations – сети рабочих станций)
  - computational grid (напр. SETI @home)

Распределенные ОС

- Обеспечивают использование территориально распределенных ресурсов
  - рабочие станции локальной сети
  - сервера на интернете
- Обеспечивают взаимодействие программ
  - Межпроцессное взаимодействие (interprocess communication)
- Предоставляют в совместное использование распределенные ресурсы (аппаратные и программные)
  - балансировка нагрузки (load balancing), идентификация, контроль доступа
- Основная цель – *возможность доступа к ресурсам*
  - а не ускорение

Операционные системы реального времени

- Жесткие временные требования
  - Цель RTOS: удовлетворить их
- Жесткая система реального времени
  - ОС гарантирует, что действия произойдут вовремя
  - Примеры: TCAS, системы управления производством, health monitors

- Гибкая система реального времени
  - ОС *старается* закончить выполнение к сроку
    - Не делает гарантий, но ограничивает возможную задержку
  - Примеры: цифровое аудио, мультимедийный системы

- «Реальное время» - предсказуемость, не скорость

Встроенные (embedded) ОС

- Вездесущие вычисления
  - Дешевые процессоры встроенны повсюду
  - Сколько их у вас с собой?
  - Мобильные телефоны, PDA, iPod, сетевые компьютеры
- Обычно очень ограниченные аппаратные характеристики
  - Медленные процессоры
  - Мало оперативной памяти
  - Отсутствие жесткого диска
  - Часто - ориентированность на выполнение одного конкретного приложения
  - ограниченная мощность

- Но все это быстро изменяется

Заключение

- В этом курсе мы изучим:
  - основные компоненты ОС
  - структуру этих компонент
  - наиболее важные и распространенные программные интерфейсы
  - принципы, лежащие в основе функционирования ОС
  - алгоритмы, реализующие эти принципы

Заключение

- А так же, вы будете знать:
  - как работают современные ОС, их главные недостатки
  - современные технологические тенденции, их влияние на ОС
  - суть новейших исследований по ОС
  - английскую терминологию
- Вы приобретете новый практический опыт
  - разработка сложных программ в среде Unix
  - программирование (в частности низкого уровня) на C
  - умение разбираться в сложных чужих системах и изменять их
  - умение изменять/компилировать/отлаживать ядро Linux и модули к нему

Заключение

- Использование полученных знаний
  - Возможно, вы никогда не создадите свою ОС
  - но инженер-программист или ученый в области информационных технологий должен иметь базовую подготовку
  - ОС хорошо иллюстрируют несколько видов архитектурных и технических компромиссов, с которыми вам придется столкнуться в будущем – компромиссов между стоимостью, производительностью, функциональностью, сложностью, управляемостью и т.д.

Что нужно сделать

- Освоиться на рабочем месте
  - Войти в систему, проверить login/password, проверить сетевой доступ
- Найти и прочитать web-страницу курса
- Начать читать Таненбаума – главу №1
- Начать работать над проектом 0
  - доступен на web-странице в настоящий момент