Операционные системы
Файловые системы

Лекция №07 по АКОС ФИВТ ПМИ

Зачем нужны ОС

- Компьютеры - это вычислительные системы для архитектур с разными процессорами
- У этих процессоров различные модели памяти
- Взаимодействие с внешним миром через порты ввода-вывода, прямую запись в память, обработка прерываний...

Вам действительно нужно это всё знать?

Компоненты ОС

- Ядро
- Стандартные библиотеки
- Минимальная (?) среда для взаимодействия с внешним миром

Абстракции на уровне API

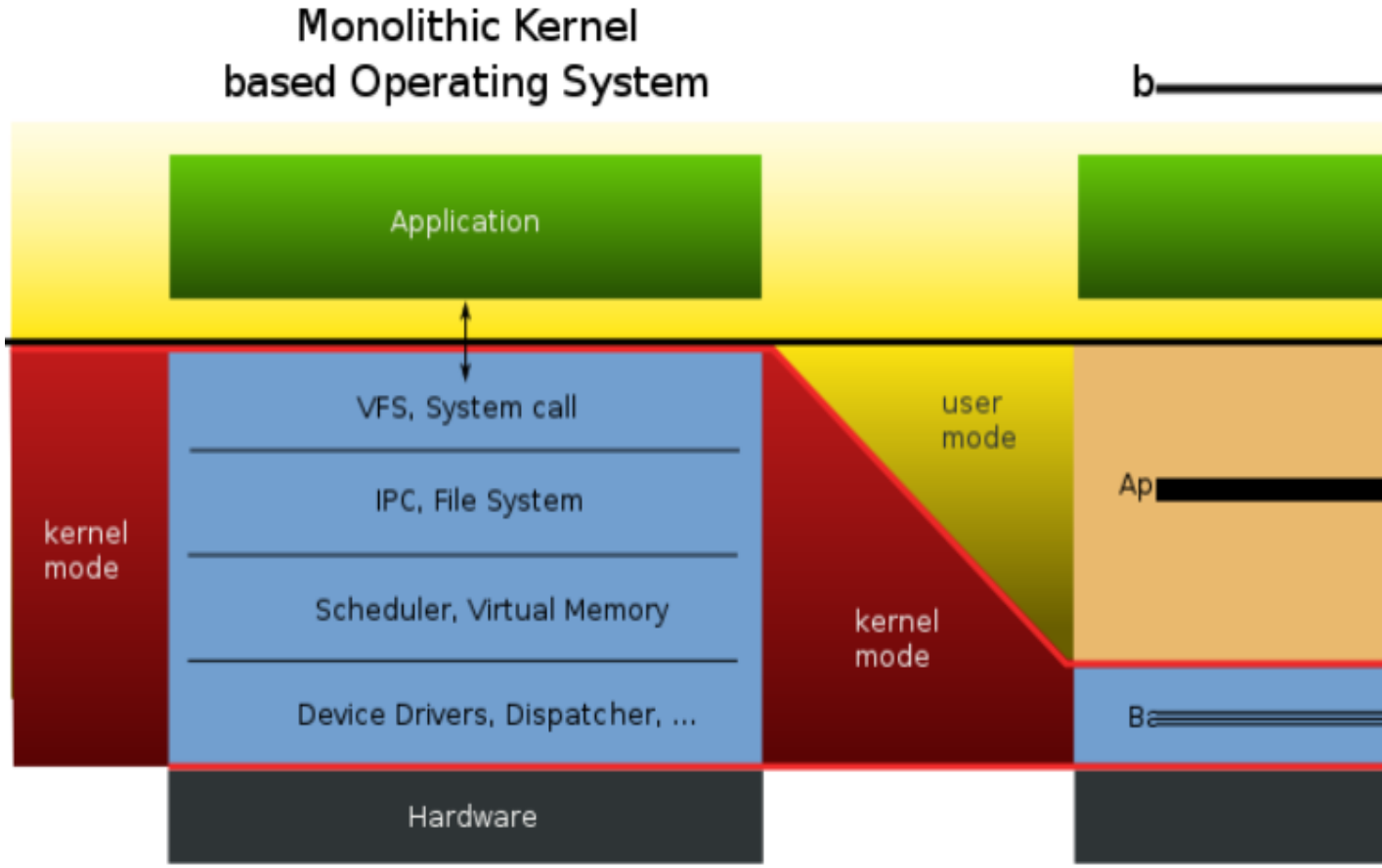
Стандартные библиотеки:

- Стандарты языка Си: ISO/IEC 9899:2011
- Стандарты языка C++: ISO/IEC 14882:2017

Стандарты на взаимодействие с ОС:

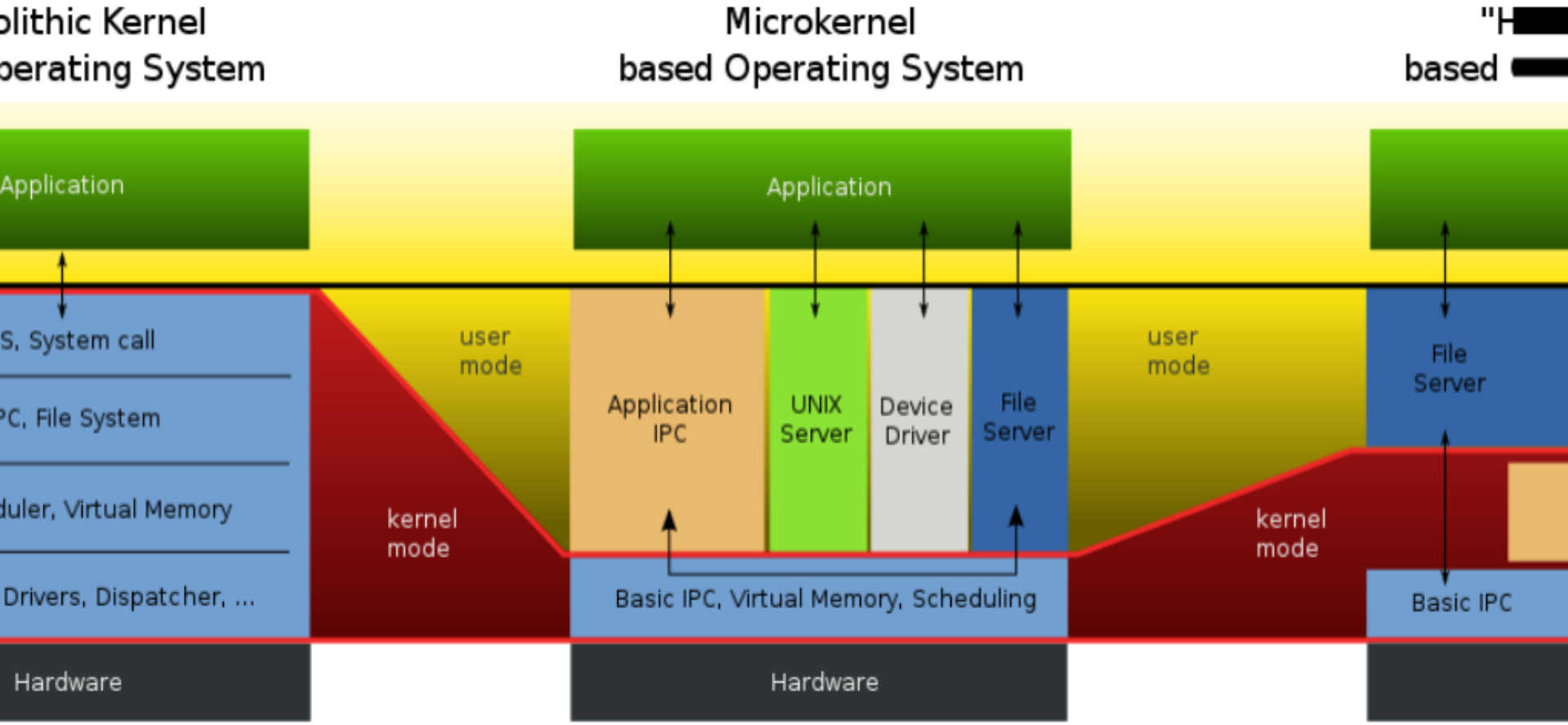
- ***P***ortable ***O***perating ***S***ystem ***I***nterface based on UNI***X*** (POSIX): IEEE 1003.1-2017
- WinAPI: внутренний стандарт Microsoft

Функциональность ядер: монолит



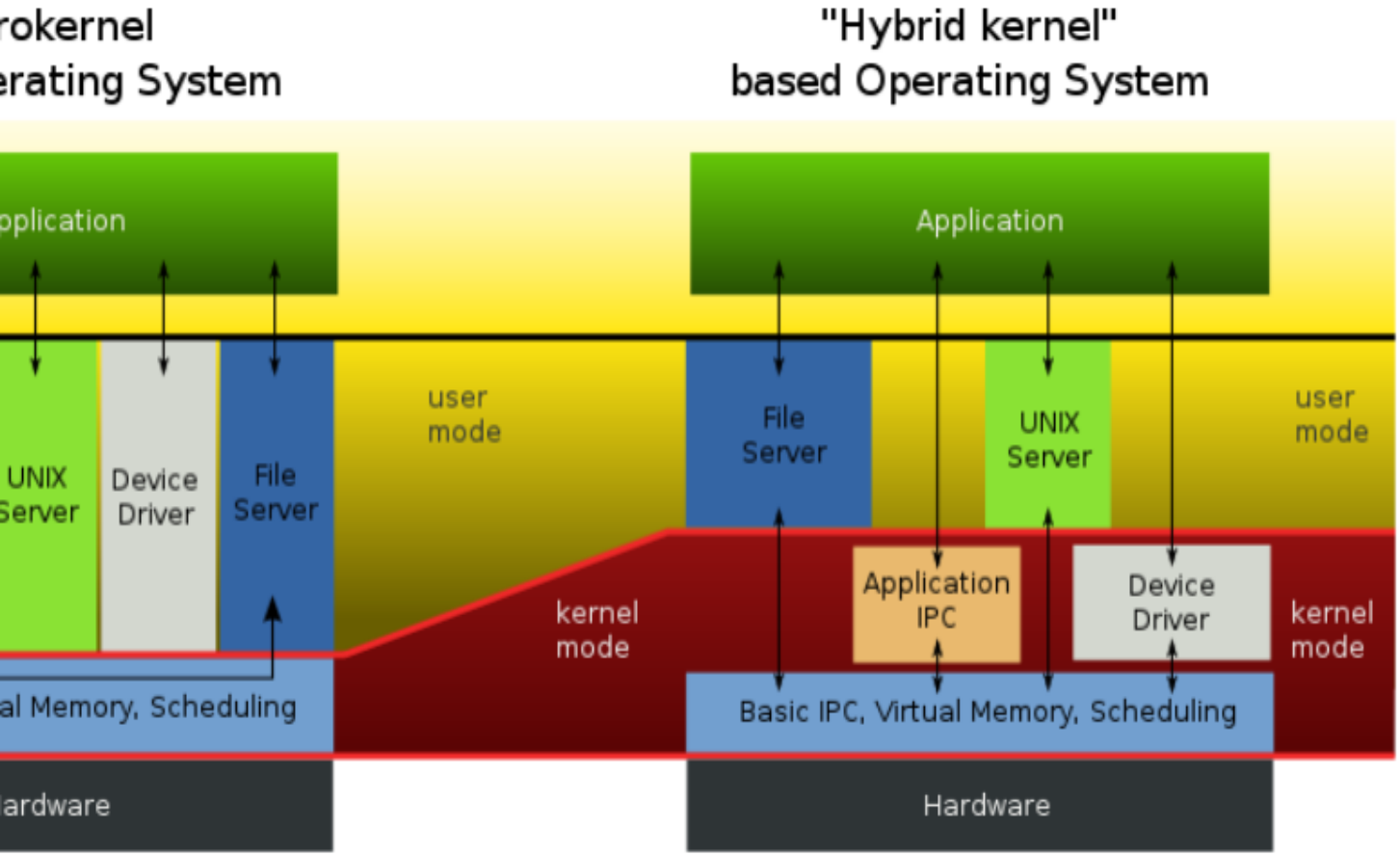
Linux, FreeBSD etc.

Функциональность ядер: микроядро



QNX, MINIX3

Функциональность ядер: гибридные



Windows, Haiku, MacOS X

Уровни абстракции

- Прикладные программы
- Высокоуровневые библиотеки
- libstdc++
- glibc -- POSIX
- Ядро -- POSIX

Взаимодействие с библиотеками

- Библиотеки загружаются в адресное пространство прикладной программы
- Вызов функций - обычным образом (плюс прыжок через PLT)
- Аргументы адресуются произвольным образом и имеют произвольную структуру

Взаимодействие с ядром

- int 0x80
  - номер системного вызова в eax
  - аргументы - в ebx, ecx, edx, ebp
- syscall / sysret
  - номер системного вызова в rax
  - аргументы - в rdi, rsi, rdx, rcx

- Аргументы - только целочисленные:
  - целые значения
  - указатели

Взаимодействие с ядром

- vdso - виртуальная «библиотека» для read-only функций
- На архитектуре x86_64:

```
__vdso_clock_gettime
__vdso_getcpu
__vdso_gettimeofday
__vdso_time
```

Подсистемы ядра

- Драйверы устройств
- Управление памятью
- Планировщик процессов
- Межпроцессное взаимодействие
- **Виртуальная файловая система**

Файловая система UNIX

Virtual ***F***ile ***S***ystem

- Единое адресное пространство для всех подключенных устройств, в том числе сетевых
- Один API для взаимодействия со всеми файлами
- Древовидная иерархия

Виды файлов

- Регулярный файл
- Каталог
- Файлы-устройства (блочные и символьные)
- Символические (но не жёсткие) ссылки
- Именованные каналы (FIFO)
- Сокеты

Демо: файловая система /dev

Регулярные файлы

- Обычные данные
- Доступна операция seek
- Содержимое файла и его формат не регламентируется ОС

Каталоги

- Каталог - это именованный файл, который где-то хранится
- Содержимое каталога - набор записей struct dirent

Ссылки

- Символические ссылки - специальные файлы, которые хранят имя целевого файла
- С точки зрения обычных функций работы с файлами, - символические ссылки не отличаются от файлов, на которые они ссылаются
- Жёсткие ссылки - это не ссылки, а дополнительные имена файлов в каталоге

Устройства

- Абстракция для упрощения взаимодействия пользовательских программ с устройствами
- Символьные устройства позволяют принимать/передавать данные
- Блочные устройства допускают операцию seek
- Be careful!
  dd if=/dev/zero of=/dev/sda # facepalm

Каналы FIFO и сокеты

- Механизм межпроцессного взаимодействия
- Данные физически не хранятся на дисках
- First In - First Out
- Сокеты от каналов отличаются возможностью подключения нескольких клиентов

Физические файловые системы

- Корень системы VFS - каталог /
- Монтирование файловой системы - процесс подключения физической ФС в дерево основной системы
- команды mount/umount и файл /etc/fstab

- Данные внутри отдельных ФС всегда консистентны → не возможны жёсткие ссылки на файлы из другой ФС

Виды физических ФС

- На диске - fat, ntfs, ext2/3/4
- На диске, но без файлов - swap
- Сетевые - smbfs, nfs
- Виртуальные - tmpfs, sshfs, overlayfs

Отображение файлов на диске в VFS

- Физическая ФС имеет свой *сессионный* порядковый номер (st_dev):
  major (24 бита) - тип устройства
  minor (8 бит) - порядковый номер
- /dev/loop - отображение произвольного файла на "устройство"
- Внутри физической файловой системы у каждого файла есть свой порядковый номер (inode, st_ino)
- Если формат ФС не поддерживает inode, то его эмуляция - задача драйвера
- Пара (st_dev, st_ino) однозначно позволяет идентифицировать файл в VFS

Файловые дескрипторы

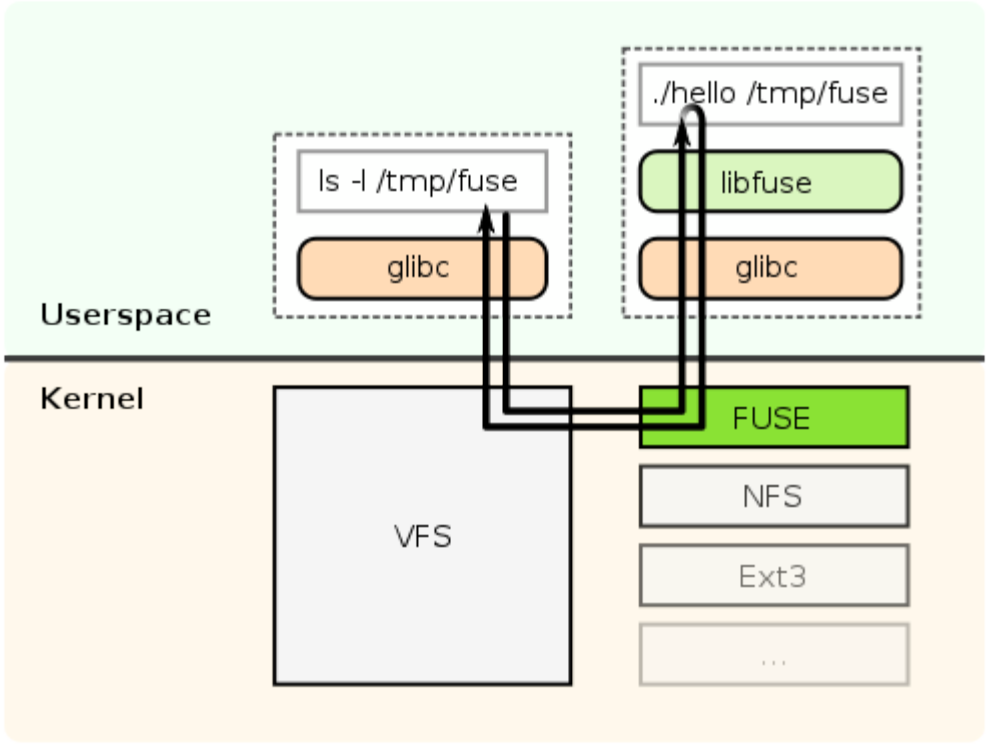
- Пара (st_dev, st_ino) однозначно позволяет идентифицировать файл в VFS
- Пользовательскому процессу файл не доступен, если он не открыт для текущего процесса
- Файловый дескриптор - порядковый номер внутри отдельного процесса
- Для каждого процесса все файловые дескрипторы уникальны
- Количество файловых дескрипторов для каждого процесса - ограничено

Файловые дескрипторы

- В начальный момент *времени как правило* (но не обязательно) существуют файловые дескрипторы:
  - stdin (0)
  - stdout (1)
  - stderr (2)
- При открытии нового файла используется следующий доступный порядковый номер

Демо: файловый дескриптор с любым номером

FUSE - файловая система своими руками



- Библиотека для реализации ФС
- Можно писать не только на Си, но и на Python
- Работает в пространстве пользователя
- Взаимодейсвтие через /dev/fuse

Целостность данных

- Ядро Linux старается использовать всю доступную физическую память в качестве буфера ввода-вывода
- Сброс данных на диск - при отмонтировании ФС или sync
- Могут существовать дополнительные буферы вне VFS, например буфер USB

```
int fsync(int fd) /* принудительная
        синхронизация данных */
```

Целостность данных

- Целостность файлов
  - тут трудно что-то сделать программными средствами
  - дублирование RAID-1+
- Целостность структуры ФС
  - журналирование: при изменении данных ФС, сначала создаётся и синхронизируется новая запись, и только в случае успеха - удаляется старая
  - при повреждении данных выполняется откат старой записи