Операционные системы
Часть 1. Обзор

Клименков С.В.
Версия 1.0.0
30.08.2020
vk.com/serge_klimenkov

tuneit

Организация курса

0.1

- О курсе
  - Лекции
  - Практики
  - Экзамены
- Баллы
- Литература

Контактная информация

- Технические вопросы
  https://vk.com/serge_klimenkov
- Лекции
  https://youtube.com/c/SergeKlimenkov
- Материалы по курсу
  https://se.ifmo.ru/
- Комната 374
- Техническая беседа в ВК
- ~~ИСУ, Электронная почта~~

Лекции

- Общее введение
- Процессы
- Память
- Межпроцессное взаимодействие
- Ввод-вывод
- Файловые системы
- Загрузка ОС

Лабы

- На данный момент планируется две штуки
  - Мониторинг ядра
  - Разработка псевдо-драйвера
- Могут быть вариации, если найдем более подходящий вариант

БАРС

- На данный момент тайна, покрытая мраком

Литература

- Stallings, William. Operating systems: internals and design principles. Global Edition © Pearson Education Limited 2018, ISBN: 978-1-292-21429-0
- Yosifovich, Pavel, David A. Solomon, and Alex Ionescu. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more. Microsoft Press, 2017, ISBN: 978-0-7356-8418-8
- Mauerer, Wolfgang. Professional Linux kernel architecture. John Wiley & Sons, 2010, ISBN: 978-0-470-34343-2
- Robert, Love. Linux kernel development. Pearson Education, 2010, ISBN-13: 978-0-672-32946-3

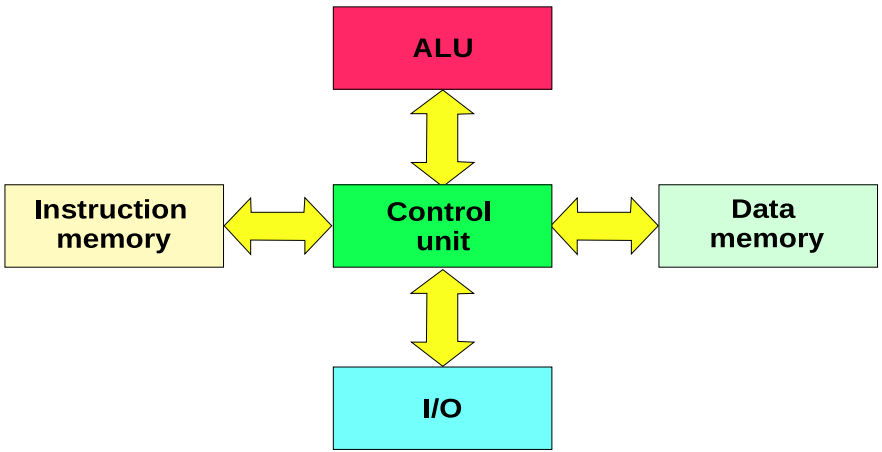
Архитектура компьютерных систем

1.1

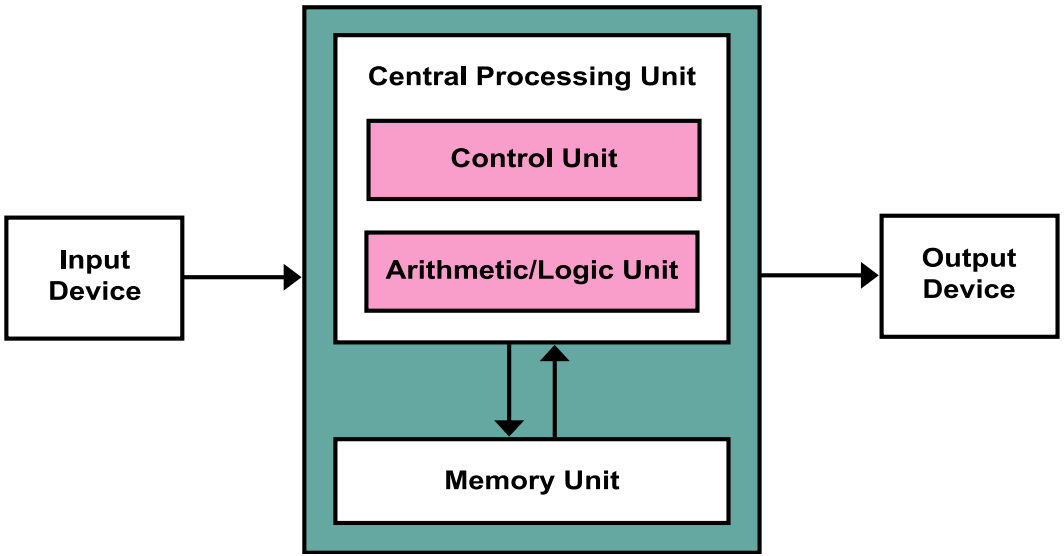
- Неймоновская и Гарвадская Архитектура
- UMA
- NUMA

Архитектура компьютерных ситем

Гарвардская архитектура



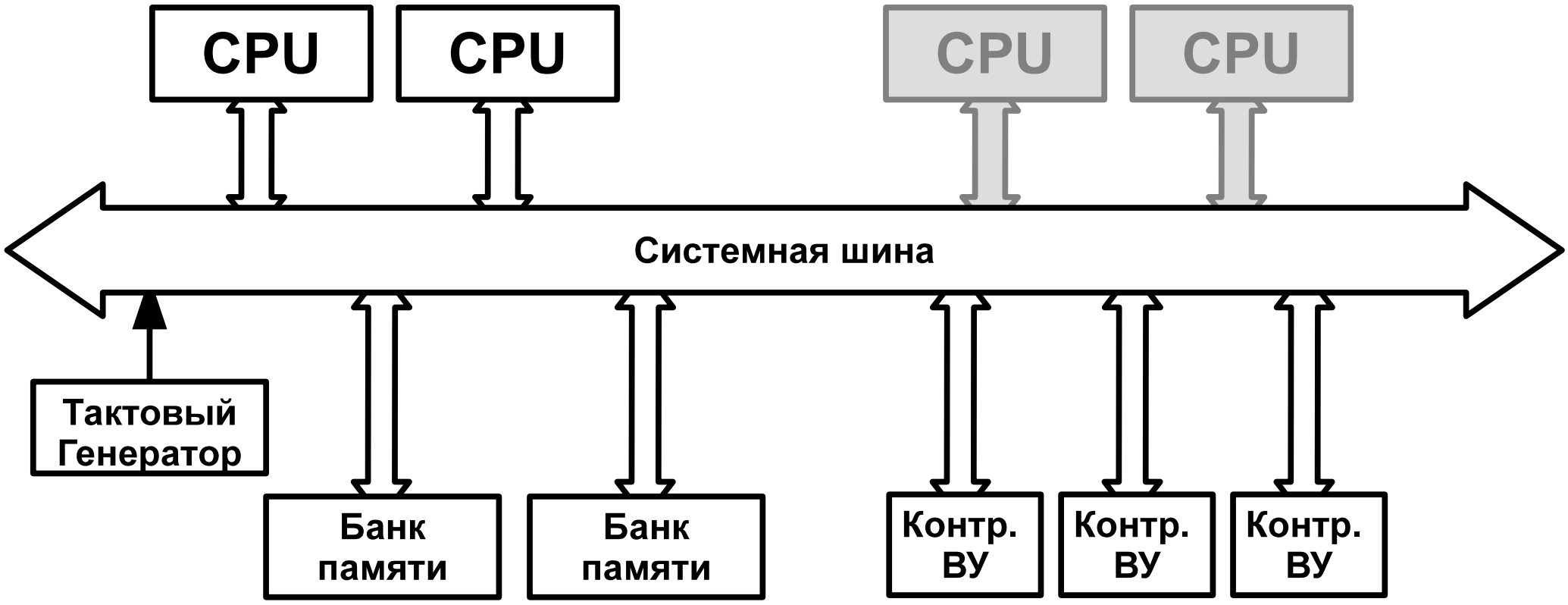
Архитектура фон Неймана



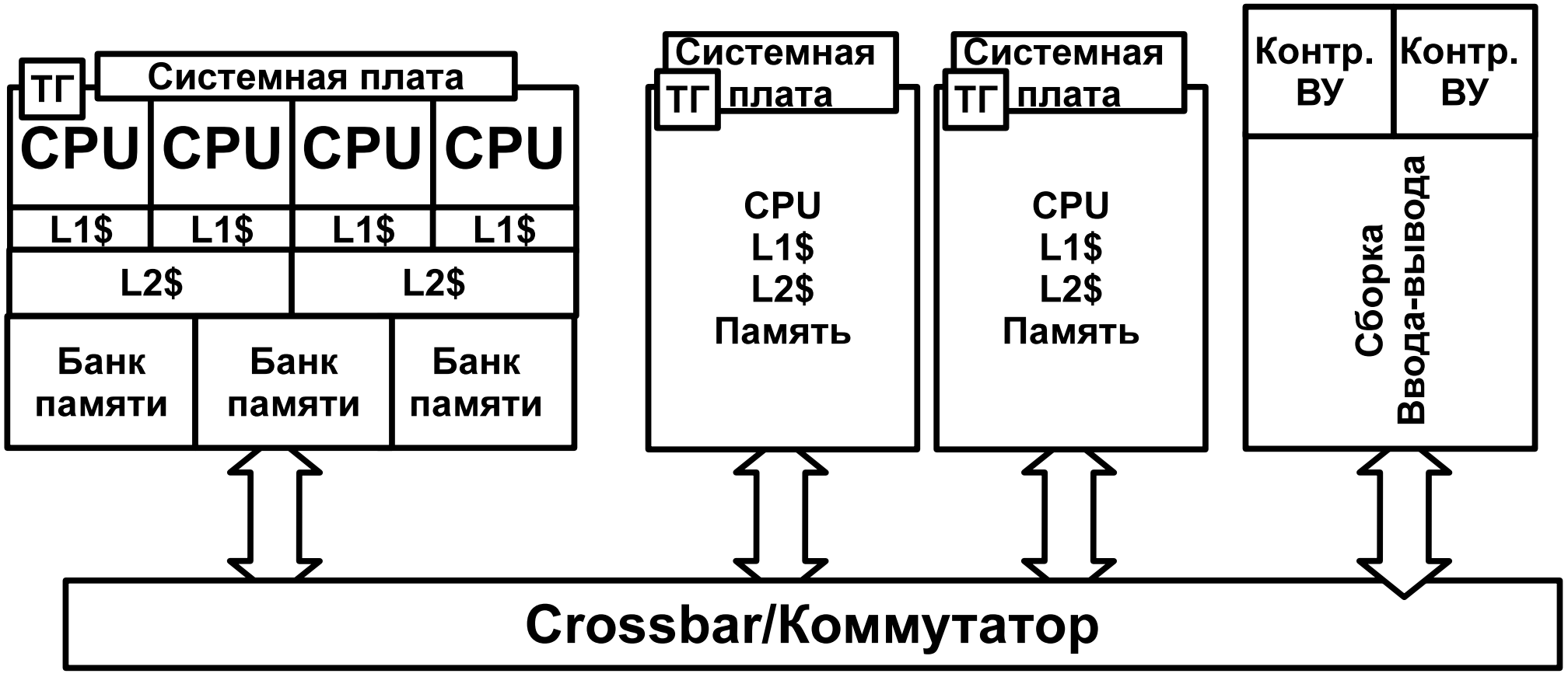
Архитектура ЭВМ Дж. фон Неймана

- Принцип однородности памяти
  - команды и данные хранятся в одной и той же памяти (внешне неразличимы)
- Принцип адресности
  - память состоит из пронумерованных ячеек, процессору доступна любая ячейка
- Принцип программного управления
  - вычисления представлены в виде программы, состоящей из последовательности команд
- Принцип двоичного кодирования
  - вся информация, как данные, так и команды, кодируются двоичными цифрами 0 и 1

UMA — Uniform Memory Access



NUMA — Non Uniform Memory Access

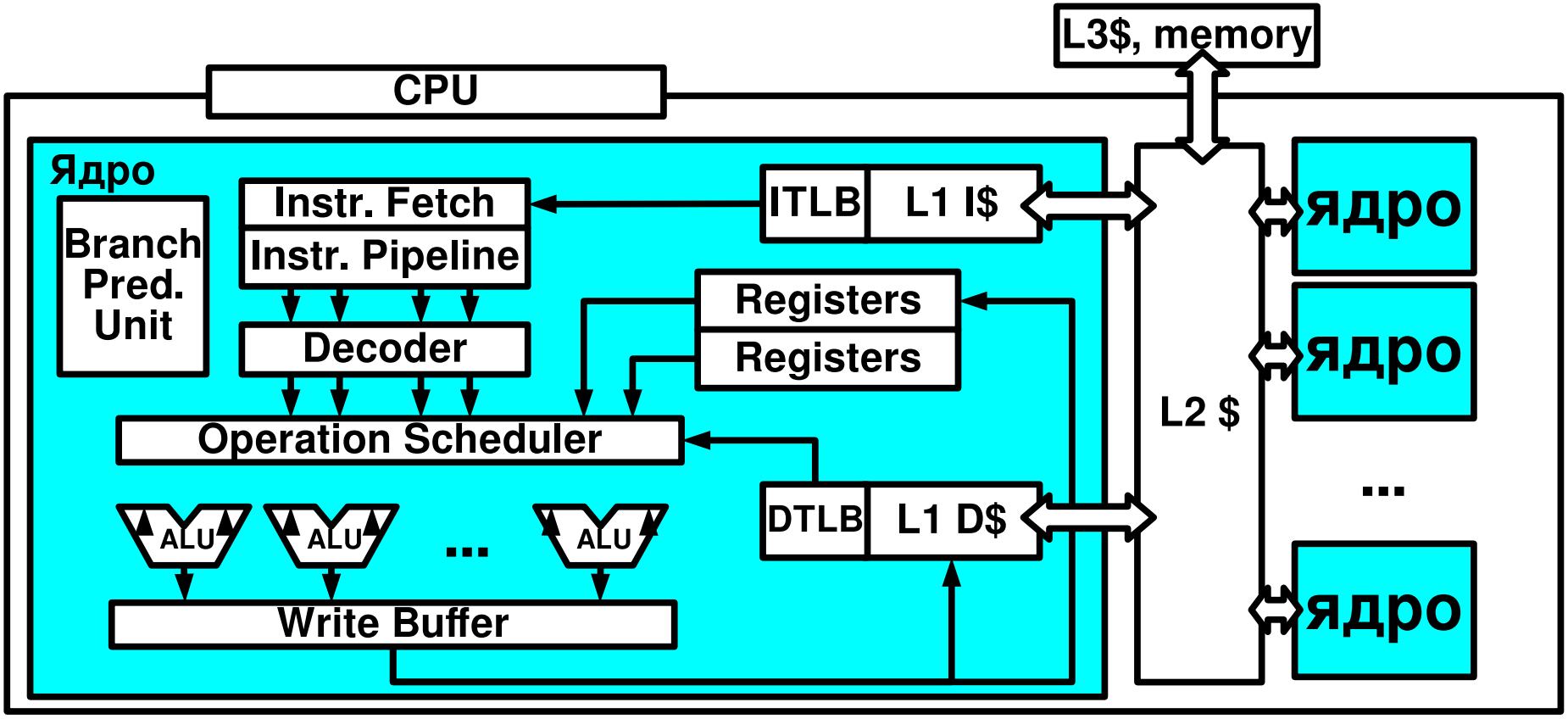


Обзор элементов компьютерных систем

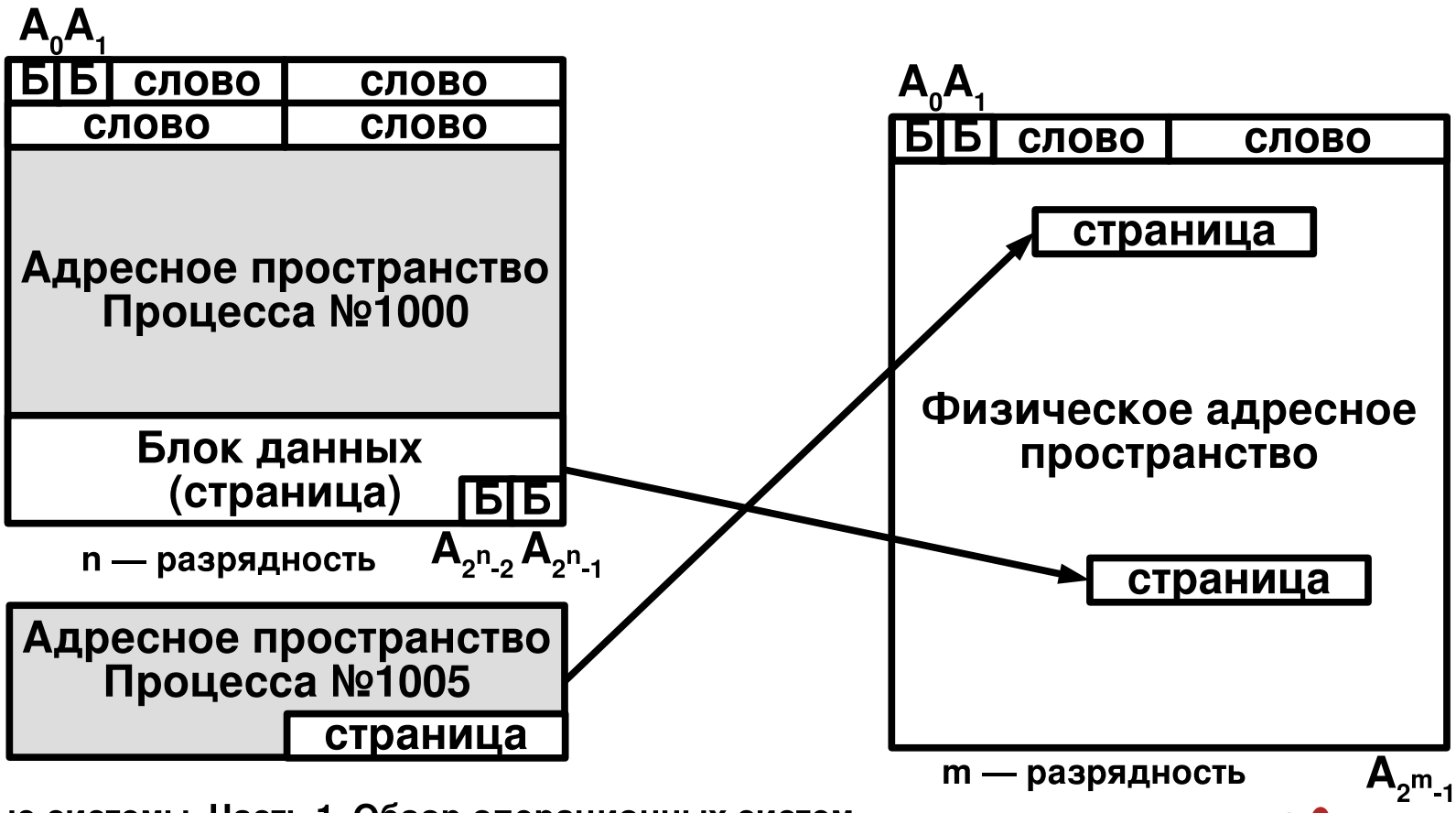
1.2

- Процессор
- Организация памяти
- Вычисления
- Прерывания

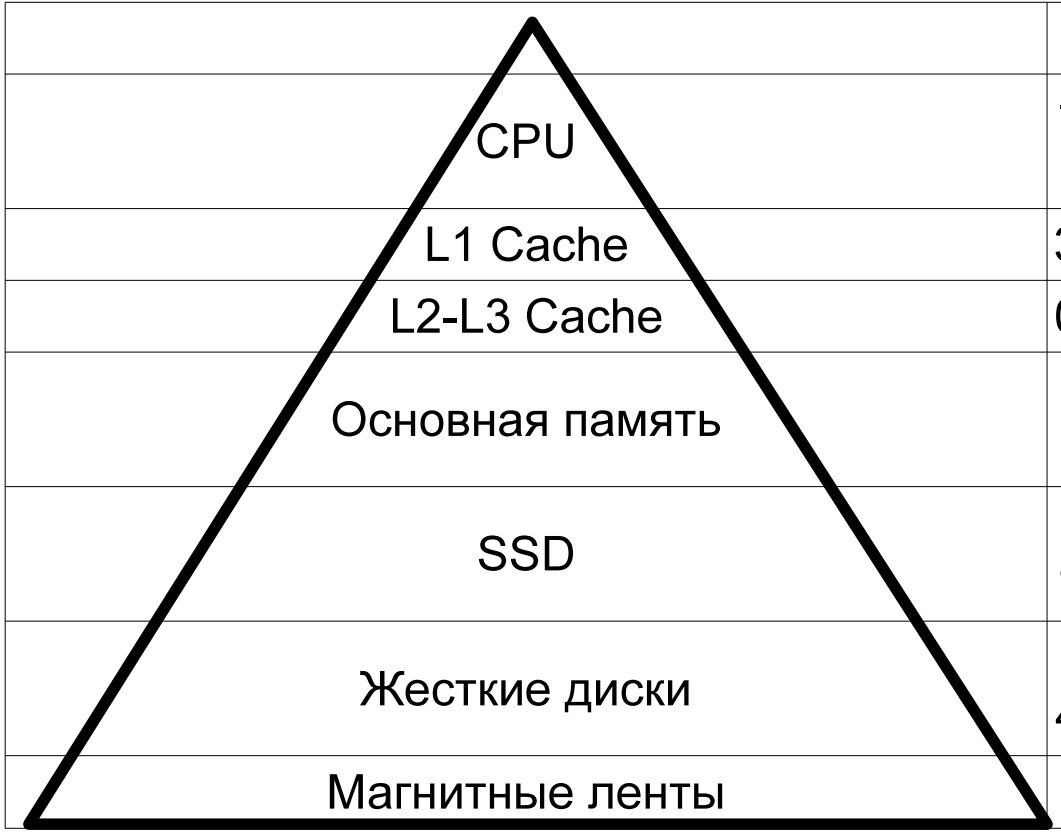
Процессор



Организация памяти



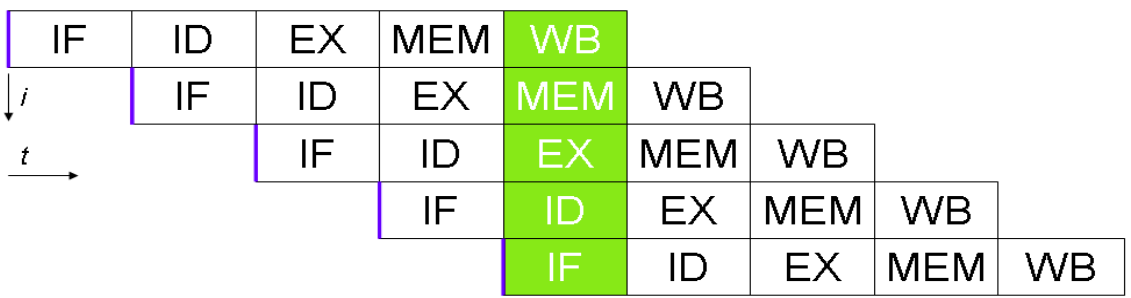
Пирамида памяти



	Объем	Тд	*	Тип	Управл.
CPU	100-1000 б.	<1нс	1с	Регистр	компилятор
L1 Cache	32-128Кб	1-4нс	2с	Ассоц.	аппаратура
L2-L3 Cache	0.5-32Мб	8-20нс	19с	Ассоц.	аппаратура
Основная память	0.5Гб-4ТБ	60-200нс	50-300с	Адресная	программно
SSD	128Гб-1Тб/drive	25-250мкс	5д	Блочн.	программно
Жесткие диски	0.5Тб-4Тб/drive	5-20мс	4м	Блочн.	программно
Магнитные ленты	1-6Тб/к	1-240с	200л	Последов.	программно

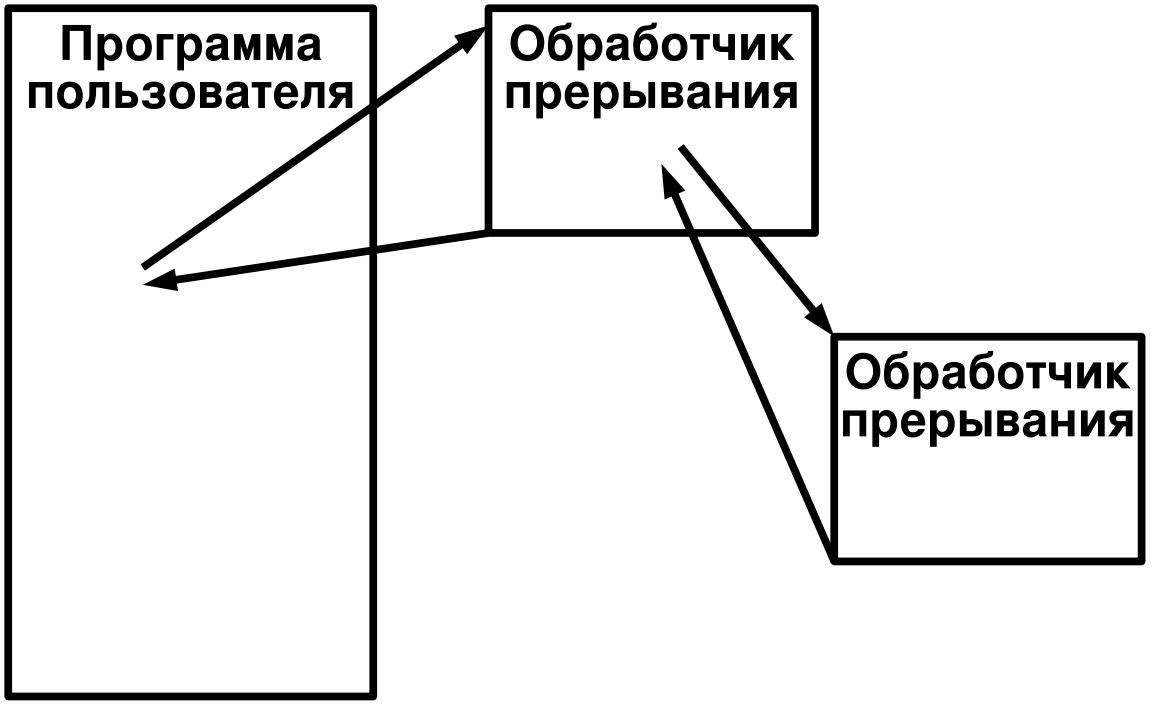
Организация вычислений

- Ядро выполняет каждую команду последовательно. Цикл команды:
  - Выборка команды (Instruction Fetch)
  - Декодирование инструкций (Instruction Decode)
  - Исполнение (Exectution)
  - Чтение памяти (MEM)
  - Запись (Write Back)

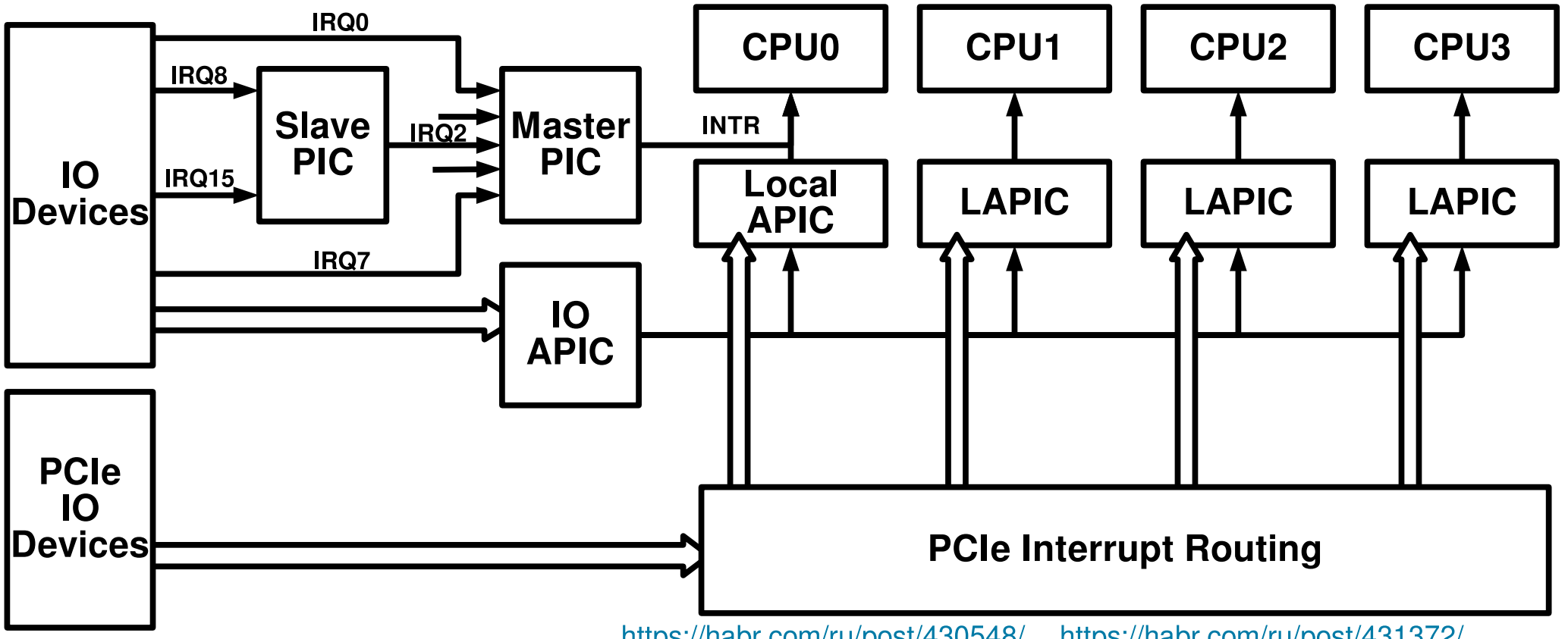


Прерывания

- Выполняются в конце цикла команды
- Могут быть вложенными
- Приоритеты

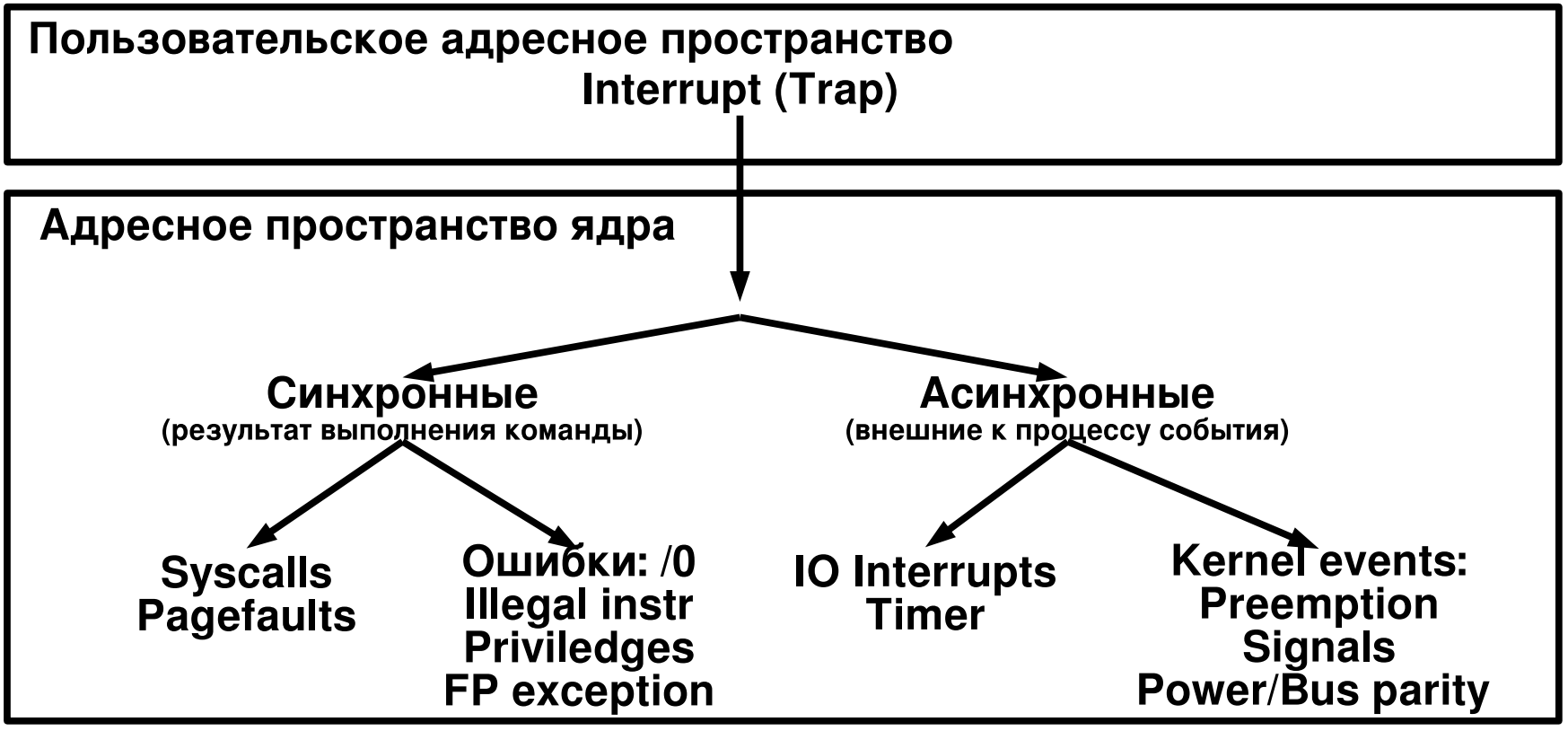


Контроллер прерываний x64



https://habr.com/ru/post/430548/ https://habr.com/ru/post/431372/

Типы прерываний



Общие сведения об ОС

1.3

- Функции ОС
- Оператор ЭВМ
- Пакетная обработка
- Многозадачность
- Разделение времени

Типичные функции ОС

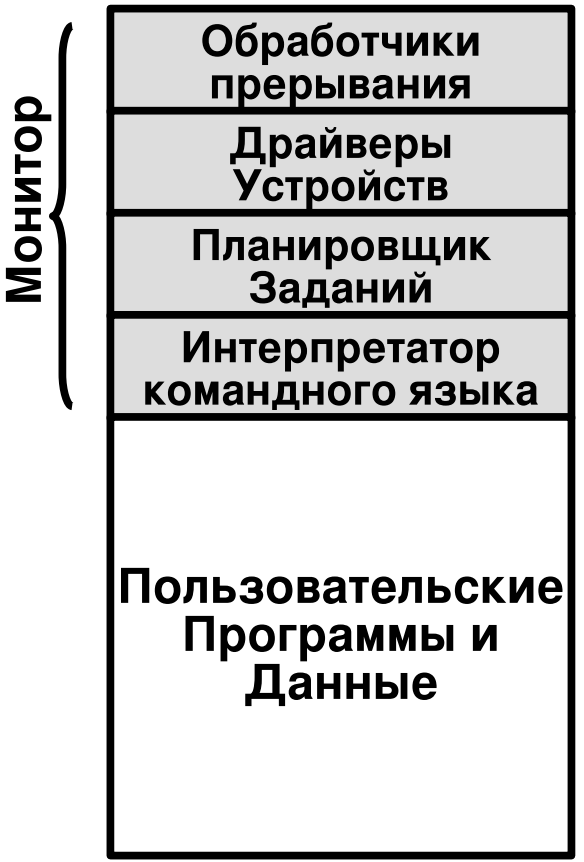
- Разработка программ
- Выполнение программ
- Доступ к устройствам ввода-вывода
- Контролируемый доступ к файлам
- Доступ к системе и системным ресурсам
- Обнаружение и обработка ошибок
- Учет использования и диспетчеризация ресурсов
- Предоставление ключевых интерфесов ОС:
  - ISA (Instruction Set Architecture) — Набор команд
  - ABI (Application Binary Interface) — Бинарный интерфейс приложения
  - API (Application Programming Interface) — Интерфейс прикладных программ

Последовательная обработка. Оператор

- В первых ЭВМ был только пульт управления
- Оператор должен был:
  - получить программу с данными от программиста;
  - подготовить программу к загрузке (н-р, с перфокарт);
  - загрузить программу и компилятор;
  - запустить программу на вычисление;
  - распечатку с результатами передать программисту.
- Минусы:
  - Наличие расписания машинного времени
  - Долгое время подготовки к работе

Пакетная обработка. Системный Монитор

- Машинное время дорогое, его простои необходимо минимизировать
- 1950 г., General Motors, IBM 701
- Наборы программ и данных передавались оператору и запускались



Обще-системная эффективность

- Одно задание плохо загружает CPU:
  - Read (15 мкс) → Compute 100 instr (1 мкс) → Write (15 мкс)
  - Общая загрука CPU ~ 3.2%
- Давайте запустим много задач, и пока одни занимаются вводом-выводом, другие будут производить вычисления

Многозадачность

Системы разделения времени (Time Sharing)

- Хорошо бы исключить оператора и добавить пользователей!
  - Посадим юзеров за терминалы, пусть сами работают
  - Будем выдавать им часть времени процессора с использованием квантования времени (time slices)
- CTSS (CompatibleTime-Sharing System), MIT 1961, IBM 709
  - Выгрузка и загрузка задач
  - 32 пользователя
- Появились проблемы разделения ресурсов и защиты одних программ от других

Основные задачи операционных систем

1.4

- Управление процессами
- Виртуальная память
- Диспетчеризация и планирование ресурсов
- Безопастность

Процессы

- Multics, 1965 г. General Electric, Bell Labs.
- Процесс — совокупность взаимосвязанных и взаимодействующих операций, преобразующих входящие данные в исходящие. (ISO 9000:2000)
- Процесс — экземпляр программы во время ее исполнения
- Процесс — единица активности ОС, в которой существуют последовательные действия, текущее состояние и набор связанных ресурсов.

Структура процесса

- Исполняемая программа
- Набор потоков исполнения
- Связанные структуры ядра
- Адресное пространство
  - Код, данные, стек, куча
- Контекст исполнения
- Контекст безопасности
- Ресурсы (файлы и пр.)
- Динамические библиотеки

Структура процесса

- Исполняемая программа
- Набор потоков исполнения
- Связанные структуры ядра
- Адресное пространство
  - Код, данные, стек, куча
- Контекст исполнения
- Контекст безопасности
- Ресурсы (файлы и пр.)
- Динамические библиотеки

Проблемы современных процессов

- Защита памяти процессов
  - Недетерминированное поведение программы
- Взаимные блокировки
  - deadlocks, starvation, livelocks
- Проблемы синхронизации
- Взаимное исключение доступа к ресурсам
- ...

Управление памятью

- Изоляция процессов
- Управление выделением и освобождением памяти
  - Heap allocator, Kernel allocator, mapping files
- Поддержка модулей
  - Динамическая загрузка модулей
- Защита и контроль доступа
  - Права на сегменты памяти (H-p: noexec data, stack)
- Долговременное хранение
- Страничный обмен
  - Paging, swapping

Виртуальная память

- Отдельное виртуальное адресное пространство для каждого процесса и ядра
- Использование подкачки страниц с диска для эффективного использования памяти
  - «Увеличение доступной памяти»
- Управление MMU и TLB
- Невыгружаемые страницы

Защита информации и безопастность ОС

- Доступ к системе
  - Защита от несанкционированного доступа
- Конфиденциальность
  - Невозможность неавторизованного доступа к данным
- Целостность данных
  - Защита данных от неавторизованного и нецелостного изменения
- Аутентификация и авторизация

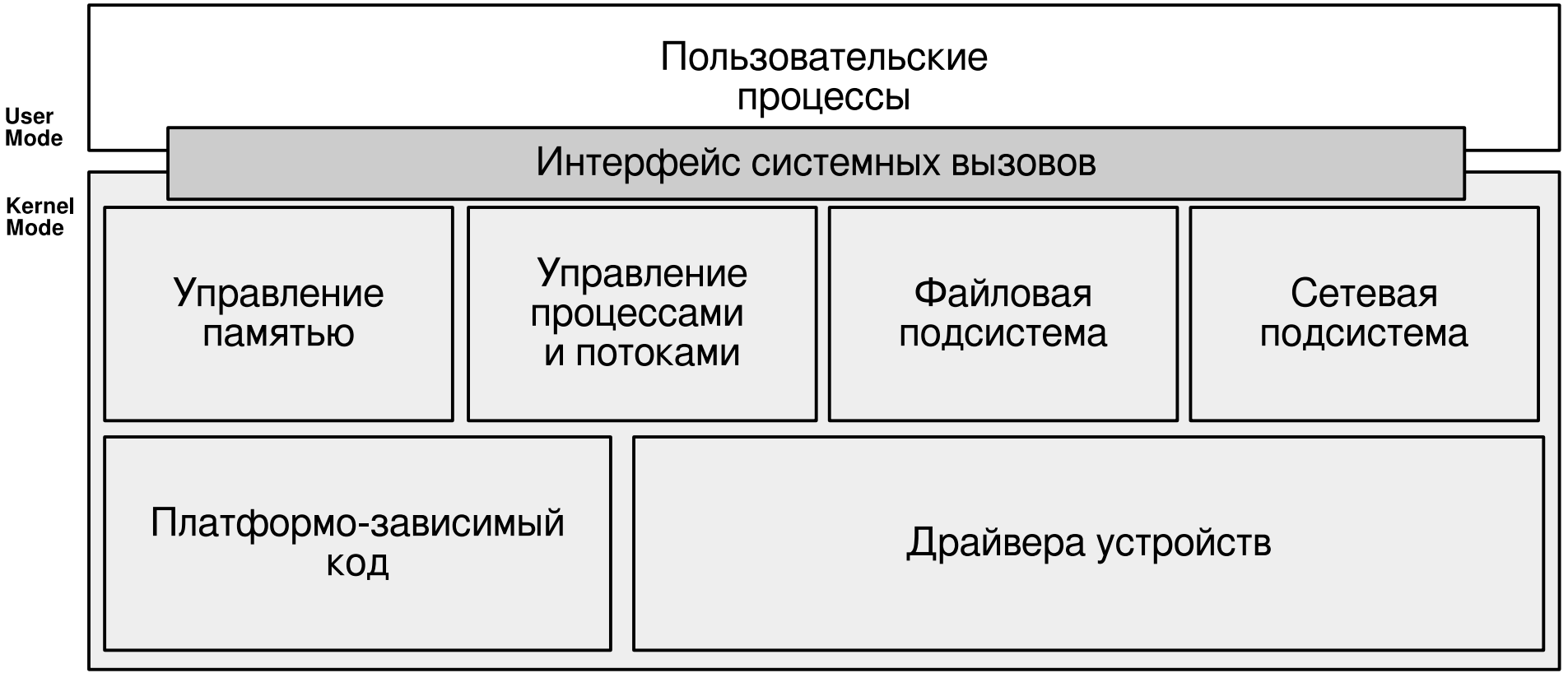
Планирование выполнения процессов и управление ресурсами

- Равноправие
  - Пользователи должны получать ресурсы равноправно
- Дифференциация отклика
  - В некоторых задачах нужно понизить время отклика
- Общесистемная эффективность
- Планировщики процессов, дисков и пр.
  - Разные классы диспетчеризации (Time Sharing, Interactive, Real Time, System, Fair Share, Fixed...)
  - https://en.wikipedia.org/wiki/I/O_scheduling

Современные архитектурные концепции операционных систем

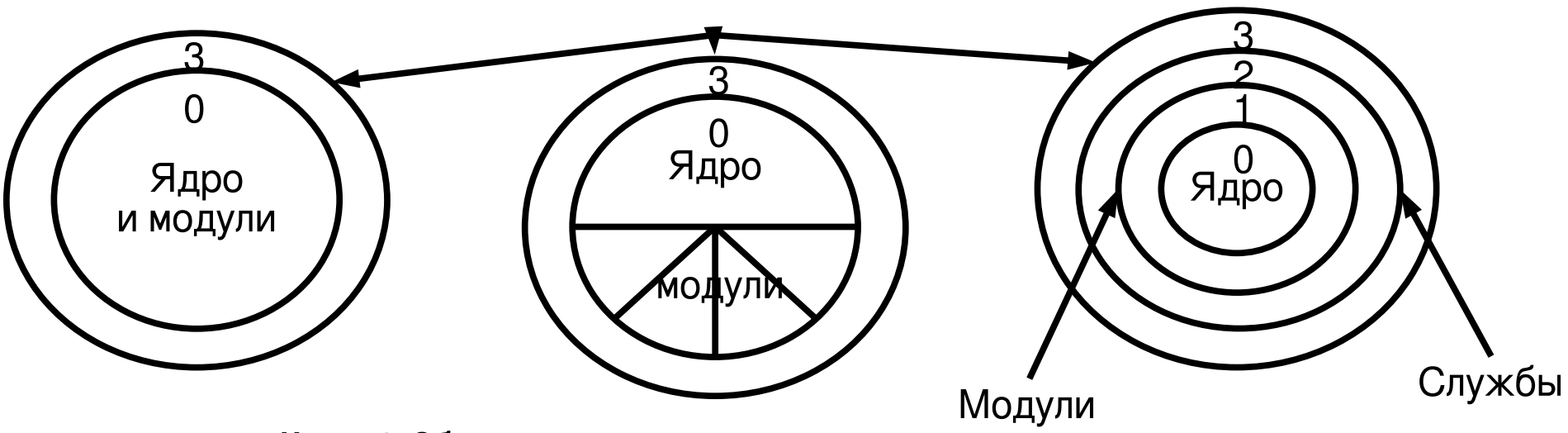
1.5

- Архитектура ядер
- Многопоточность
- SMP и ASMP
- Виртуализация



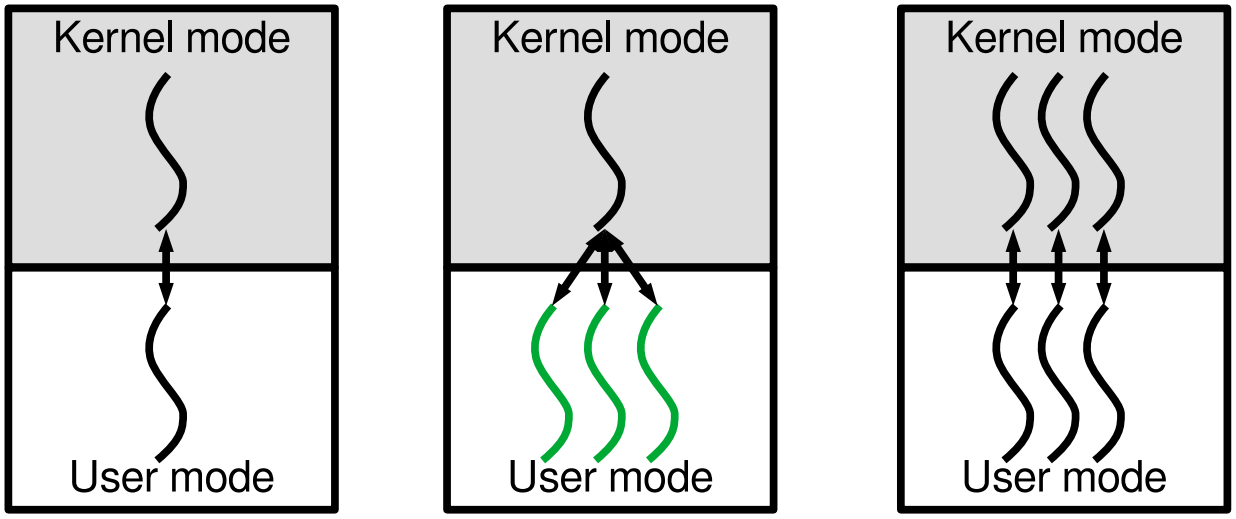
Архитектуры ядер

- Монолитное ядро
- Ядро с динамически загружаемыми модулями
- Микроядро



Многопоточность

- Поток (нить выполнения, thread) — единица диспетчеризации и выполнения ОС
- Posix Threads



SMP vs ASMP

- Asymmetric Multiprocessing — есть Master CPU, он управляет Slaves CPU
  - CPU — GPU
- Symmetric Multiprocessing – процессоры равны, процесс выполняется на нескольких процессорах одновременно
  - «Простота» разработки и производительность
  - Более высокая надежность. При отказе одного выполнять процессы могут другие
  - Масштабируемость приложений
  - Динамическое добавление ресурсов процессора
- Многопоточность ≠ Многопроцессорность

Виртуализация

- Виртуальные машины (интерпретаторы)
  - Java VM, JavaScript в браузере, Python
- Контейнеры приложений
  - Docker, Solaris containers, Linux Containers (LXC), ...
- Аппаратная виртуализация
  - KVM, Hyper-V, VMWare, Virtual Box, ...
  - Виртуализация аппаратных устройств
- Облачные технологии
  - Построены на базе аппаратной виртуализации
  - Дополнительно включают provisioning и общий мониторинг

Основные понятия надежности операционных систем

1.6

- Надежность
- Сбои
- Отказоустойчивость и резервирование

Отказоустойчивость

- Способность системы продолжать работу при аппаратных или программных ошибках
  - Избыточность аппаратуры (двойное, тройное резервирование)
  - Аппаратная «горячая» замена компонентов (диски, контроллеры, процессоры, системные платы)
  - Программная поддержка ОС выведения компонентов из системы и их подключения
  - Организация уровней хранения RAID (Redundant Array of Inexpensive Disks) в дисковой подсистеме

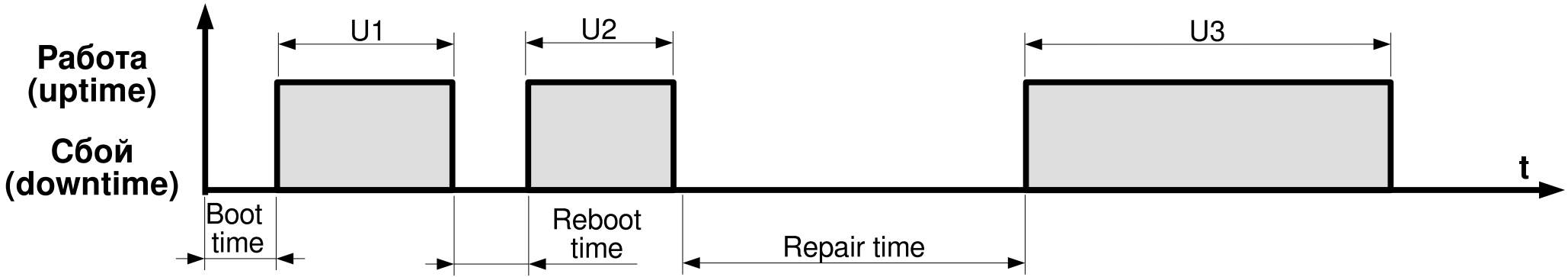
Надежность (Reliability)

- R(t) - Вероятность бесперебойной работы системы до времени t, при условии ее корректной работы в t=0
- Бесперебойная работа — корректная работа и защита данных
- Среднее время наработки на отказ (Mean Time To Failure

$$MTTF = \int_0^x R(t)dt$$

Среднее время восстановления (Mean Time To Recover)

- Обычно время для перезагрузки, ремонта или замены неисправного компонента, установки (или переустановки) ОС и ПО



$$MTTF = \frac{U1 + U2 + U3}{3} \qquad MTTR = \frac{Boot\ time + Reboot\ time + Repair\ time}{3}$$

Коэффициент доступности (Availability)

- Доля времени (%), когда система или служба доступна для запросов пользователей
- Простой (downtime) — время, в течении которого система недоступна
- Безотказная работа (uptime) — время, когда она находится в продуктивной работе

$$Availability = \frac{MTTF}{MTTF + MTTR}$$

Классы доступности систем

Класс	Коэф. доступности	Время простоя в год
Непрерывная работа	1,0	0
Выскоотказоустойчивый	0,999999	32 секунды
Отказоустойчивый	0,99999	5 минут
Восстанавливаемый	0,9999	53 минуты
Высокодоступный	0,999	8,3 часа
Обычный	0,99-0,995	44-87 часов

Отказы (faults) в системах

- Ошибочное состояние аппаратуры или ПО в результате сбоя компонентов
- Ошибки оператора (**мем не исполнять!**)

```
perl -e '$??s:;s:s;;$?::s;;=]=>%-{<-|}<&|`{;;y; -/:-@[-`{-};`-{/" -;;s;;$_;see'
```

- Физические помехи окружающей среды
- Ошибки проектирования, программирования, структур данных и пр.
- Могут быть: постоянные, временные (однократные или периодические)

Резервирование и отказоустойчивость

- Методы резервирования
  - Физическая избыточность (компонентов, серверов)
  - Временная избыточность (повтор вычислений
  - Информационная избыточность (ECC, RAID)
- Методы повышения отказоустойчивости ОС
  - Изоляция процессов
  - Разрешение блокировок при параллелизме
  - Виртуализация
  - Точки восстановления и откаты

Общая архитектура Unix/Linux

1.7

- История
- POSIX и SUS
- Генеалогия UNIX-like
- Linux
- Kernel map
- Основные подсистемы

«Отцы-основатели»

- MIT
  - Compatible Time-Sharing System (CTSS), IBM 709, 1961
  - Incompatible Timesharing System (ITS), PDP-10, 1967
- Манчестерский университет: Супервизор Atlas и экстракоды, Atlas, 1962
- MIT, GE, Bell Labs: Multics, 1965
- IBM: OS/360: Мейнфреймы System/360,1966
- Technische Hogeschool Eindhoven: THE, Electrologica X8, 1968

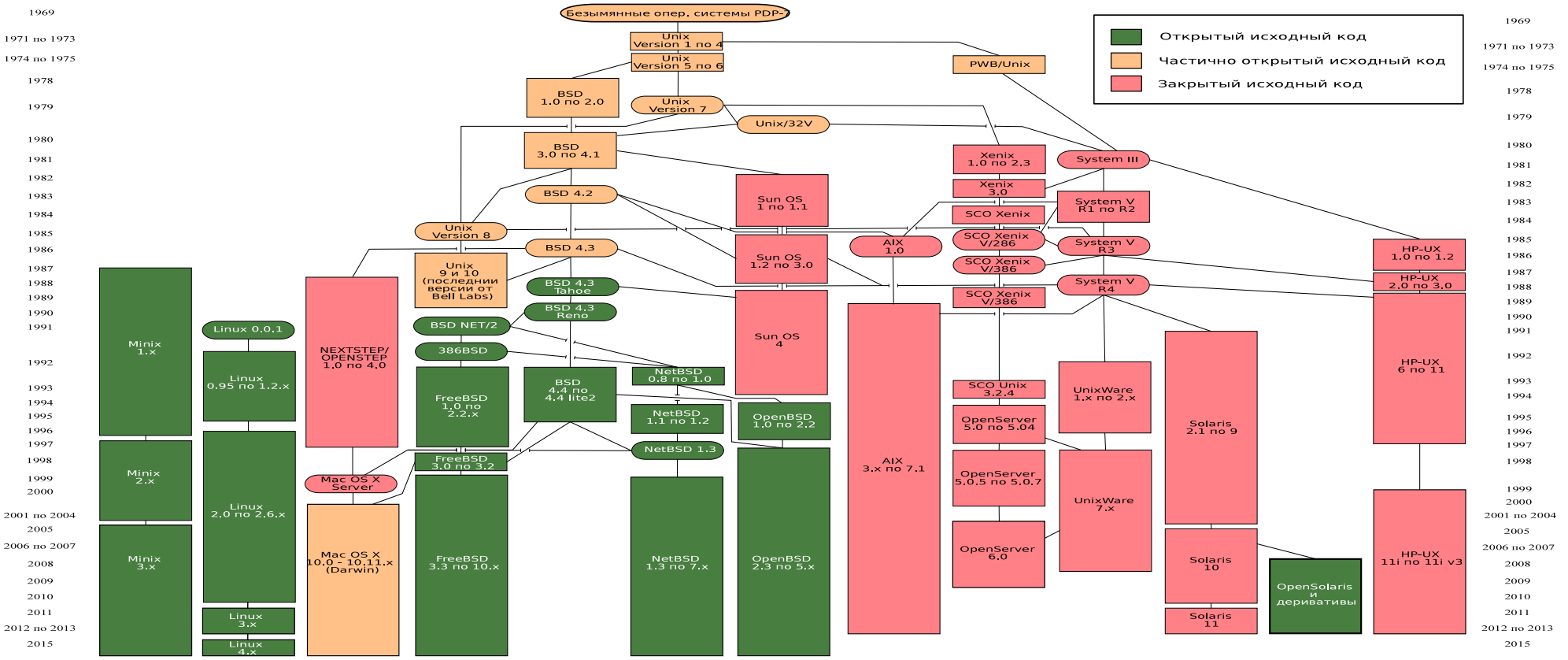
UNIX

- UNICS: Bell Labs, 1969, Кен Томпсон, Деннис Ритчи и Брайан Керниган
  - Ключевые понятия: вычислительный процесс и файл
  - Компонентная архитектура: принцип «одна программа — одна функция»
  - Минимизация ядра
  - Независимость от аппаратной архитектуры и реализация на C
  - Унификация файлов

Single UNIX Specification. POSIX

- Unix SUS: The Open Group, Austin Group
  - Основные определения
  - Системные интерфейсы
  - Командная оболочка и утилиты
  - Пояснения
  - X/Open Curses
  - UNIX: AIX, HP-UX, IRIX, Mac OS X, SCO OpenServer, Solaris, Tru64 и z/OS.
  - UNIX-like: FreeBSD, OpenBSD, NetBSD, OpenSolaris, BeleniX, Nexenta OS) и Linux
- POSIX (Portable Operating System Interface) ISO/IEC 9945

Генеалогическое дерево



https://commons.wikimedia.org/w/index.php?curid=48637753

GNU/Linux

- Ли́нус Бенедикт То́рвальдс, 1991 г., ну вы знаете, GNU GPL, ядро, общее руководство
- Ричард Столлман, Свободное Программное Обеспечение с 1983 (GNU is Not Unix) — библиотеки и обвязка — 1992

«Дистрибутив»

- Ядро
- Окружение
- Менеджер пакетов и обновлений
- Графическая подсистема
- Прикладные программы
- Поддержка

Дистрибутивы

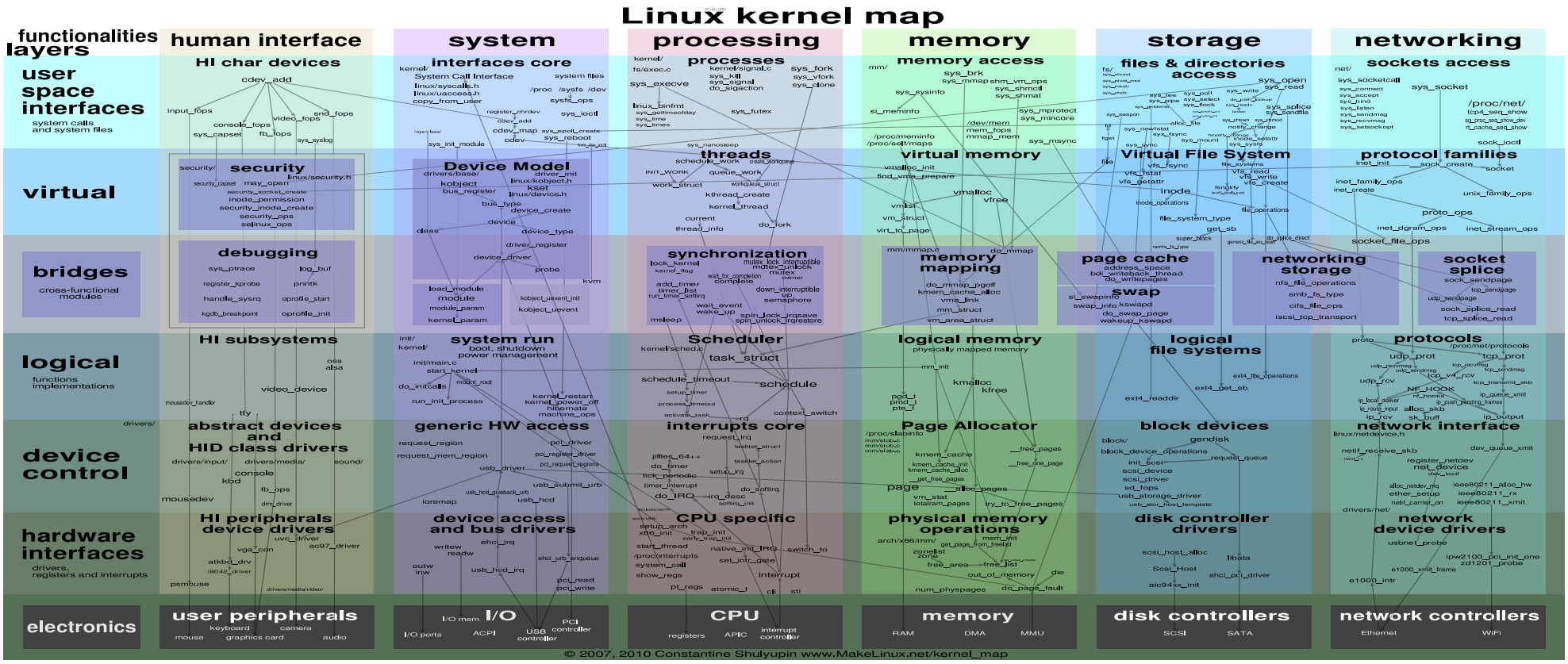
- Коммерческие и «Community»

- Ubuntu/Canonical
- Red Hat Enterprise
- SUSE
- Oracle Enterprise
- Astra Linux

- Debian/Ubuntu
- Fedora/CentOS
- Open SUSE
- ArchLinux
- Gentoo

https://ru.wikipedia.org/wiki/Список_дистрибутивов_Linux

Linux Kernel Map



https://makelinux.github.io/kernel/map/

Основные подсистемы Unix/Linux

- Процессы и планировщик.
  - Создает, управляет и планирует процессы.
- Виртуальная память.
  - Выделяет виртуальную память для процессов и управляет ею.
- Физическая память.
  - Управляет пулом кадров страниц и выделяет страницы для виртуальной памяти.
- Файловая система.
  - Предоставляет глобальное иерархическое пространство имен для файлов, каталогов и других объектов, связанных с файлами и функциями файловой системы.
- Драйверы символьных устройств.
  - Управление устройствами, которые требуют от ядра отправки или получения данных по одному байту, например терминалами, принтерами или модемами.
- Драйверы блочных устройств.
  - Управление устройствами, которые читают и записывают данные блоками, как, например, различные виды вторичной памяти (магнитные диски, CD-ROM и т.п.).

Основные подсистемы Unix/Linux (2)

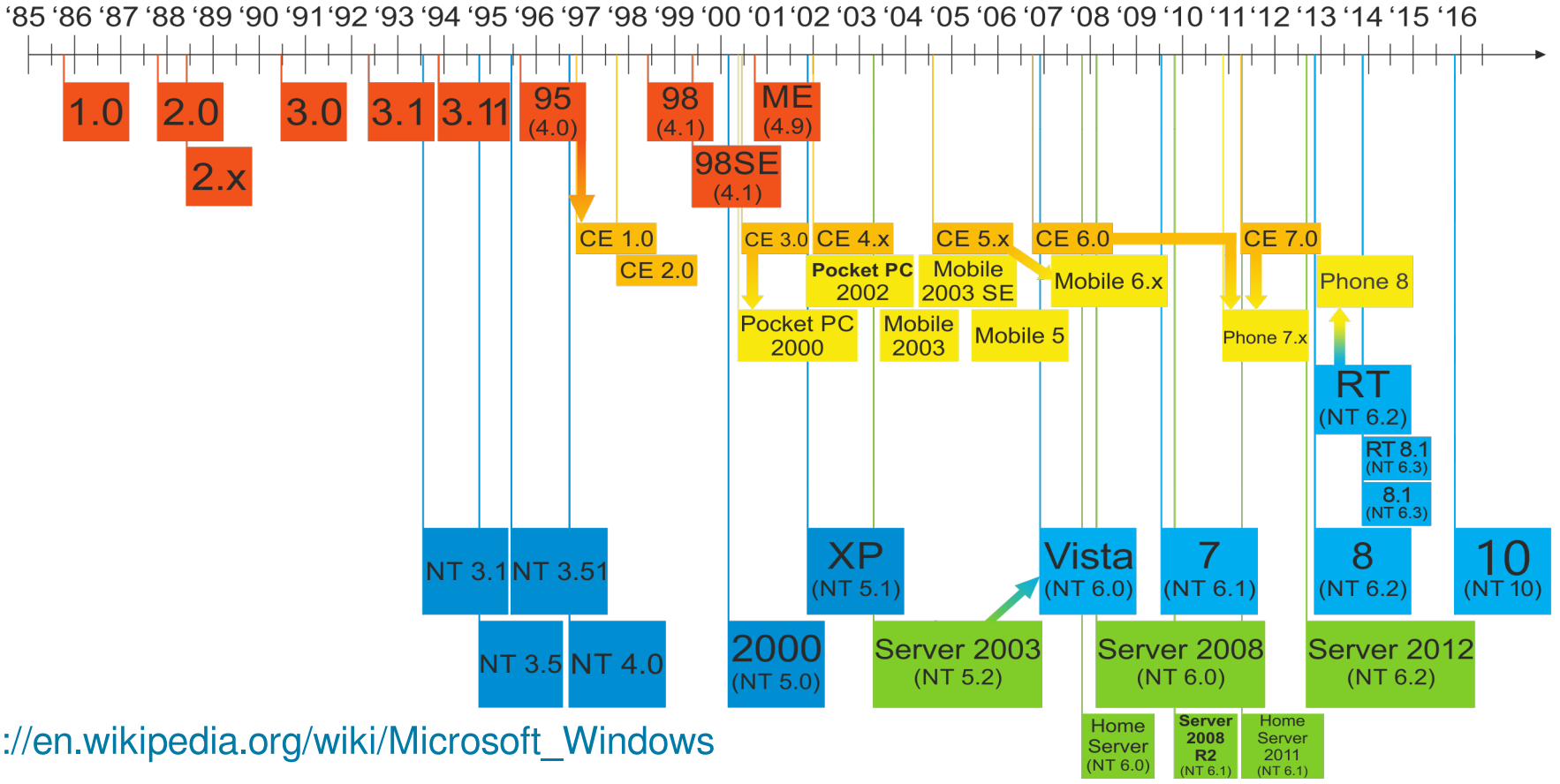
- Сетевые протоколы. TCP/IP.
  - Поддержка пользовательского интерфейса сокетов для набора протоколов
- Драйверы сетевых устройств.
  - Управление картами сетевых интерфейсов и коммуникационными портами, которые подключаются к сетевым устройствам, такими как мосты или роутеры.
- Ловушки и отказы.
  - Обработка генерируемых процессором прерываний, как, например, при сбое памяти.
- Прерывания.
  - Обработка прерываний от периферийных устройств.
- Сигналы и IPC
  - Управляет межпроцессным взаимодействием

Общая Архитектура Windows

1.8

- История Windows
- Windows 10
- Архитектура
- Windows API
- Сервисы

Версии Windows



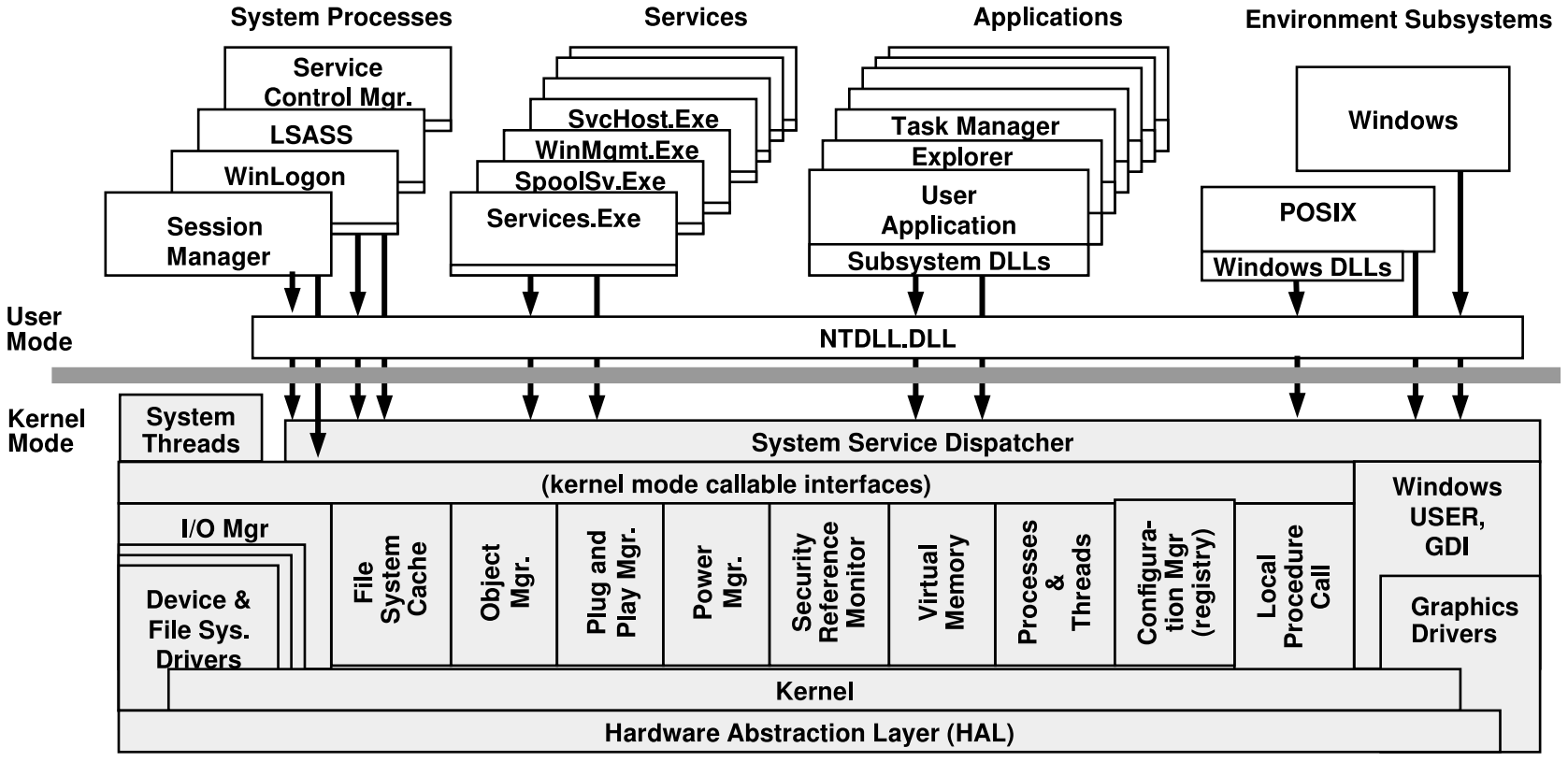
https://en.wikipedia.org/wiki/Microsoft_Windows

Windows 10 builds

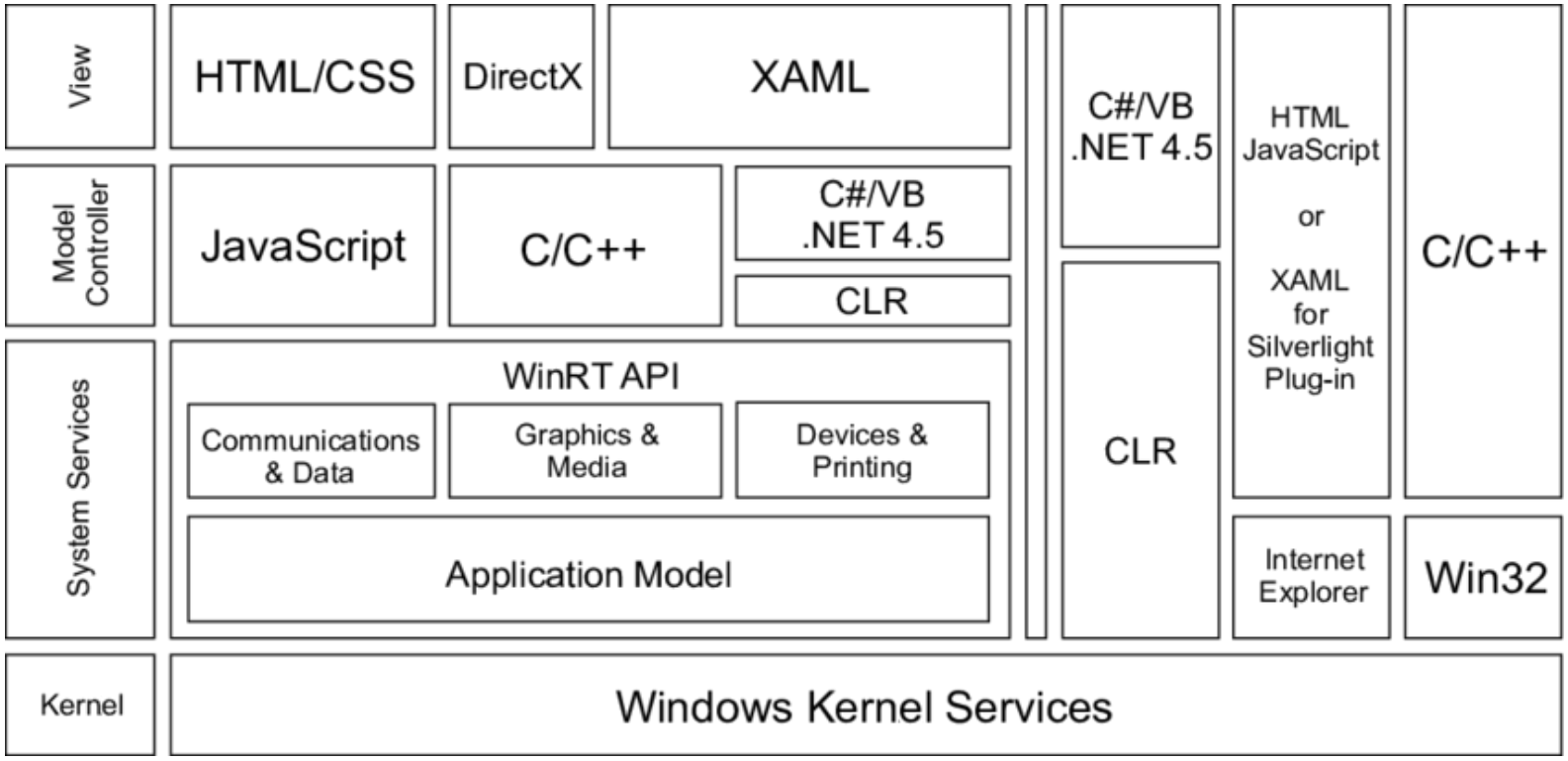
- 1507 (Threshold 1, 10.0.10240): Updated start menu, Cortana, Continium, Action Center, Microsoft Edge.
- 1511 (Threshold 2, 10.0.10586): Integrated Skype, Edge Tabs preview
- 1607 (Redstone 1, 10.0.14393): Bash command prompts, Edge, Cortana impr.
- 1703 (Redstone 2, 10.0.15063): много «small fixes»
- 1709 (Redstone 3, 10.0.16299): Windows defender, Edge, .. improvement
- 1803 (Redstone 4, 10.0.17134): много «small fixes»
- 1809 (Redstone 5, 10.0.17763): много «small fixes»
- 1903 (19H1, 10.0.18362): новая «ligth theme», Sandbox, pause update 35d,...
- 1909 (19H2, 10.0.18363): OneDrive search integration, ..
- 2004 (20H1, 10.0.19041): DirectX 12 Ultimate, upd. Android integration, Windows Subsystem for Linux 2

https://en.wikipedia.org/wiki/Windows_10_version_history

Общая архитектура Windows



Windows API (WinAPI, Win32API, WinRT)



Plant, Richard R., and Philip T. Quinlan. "Could millisecond timing errors in commonly used equipment be a cause of replication failure in some neuroscience studies?." Cognitive, Affective, & Behavioral Neuroscience 13.3 (2013): 598-614.

Сервисы и функции

- Windows API functions:
  - CreateProcess, CreateFile
- System calls (Native System Services)
  - NtCreateUserProcess
- Kernel support functions
  - ExAllocatePoolWithTag
- Windows services
  - Управляются Service Control Manager
- Dynamic link libraries (DLL)
  - msvcrt.dll, kernel32.dll

Другие важные компоненты системы

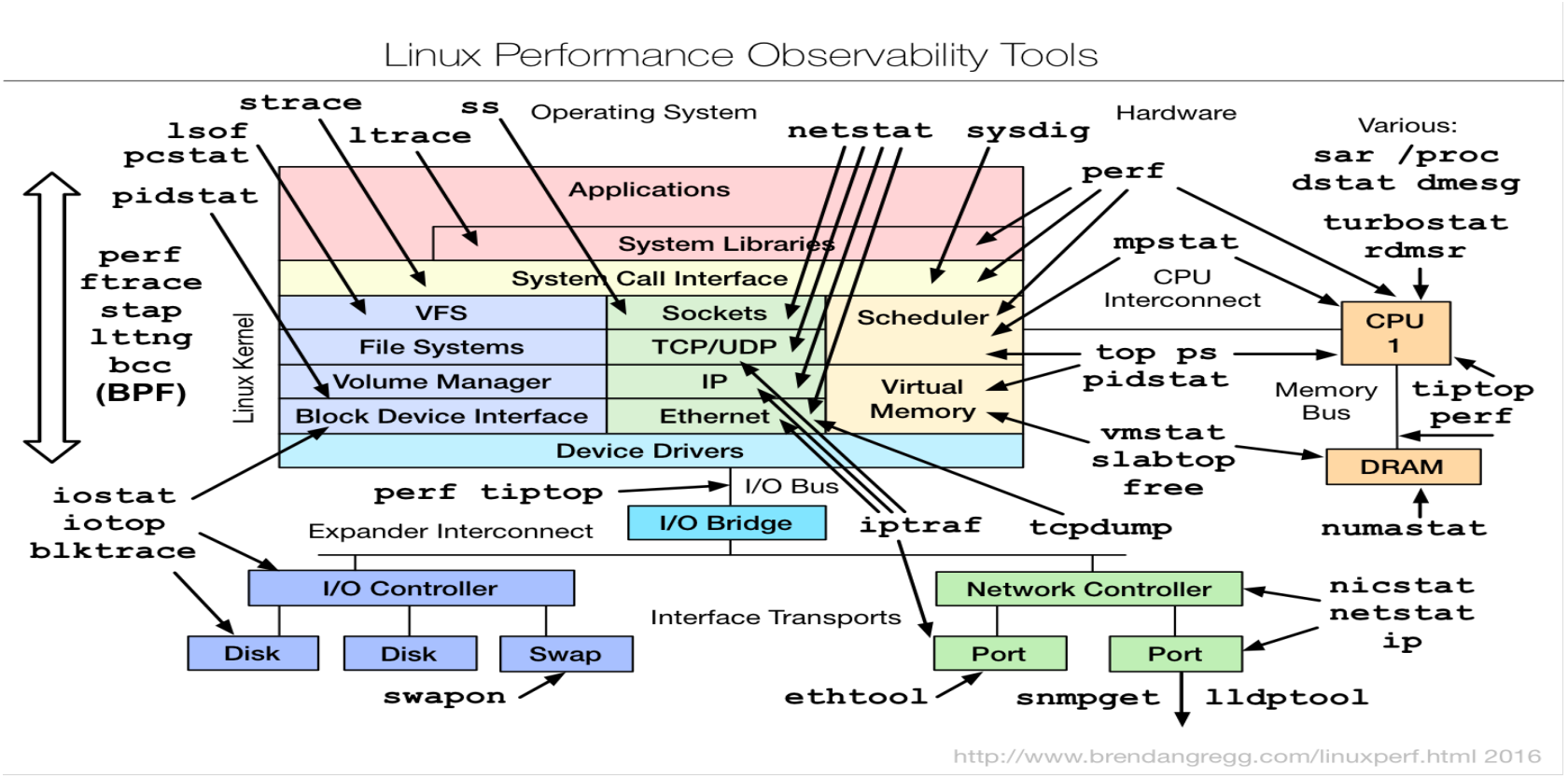
- Гипервизор Hyper-V
  - Запуск гостевых ОС, Device Guard, Hyper Guard, Credentials Guard, Application Guard, ...
- Firmware (в том числе и для устройств)
- Terminal Servers
- Объекты и безопасность
- Registry (Реестр)
- Оснастки

Средства для отладки Linux

1.9

- «Стандартные» средства
- /proc
- Трассировщики
- perf
- SystemTap
- Kernel debugger

Общесистемный мониторинг



http://www.brendangregg.com/SIides/LinuxConNA2014_LinuxPerfTools.pdf

Стандартные средства для наблюдения счетчиков ядра

- sar (system activity reporter): общесистемное средство, собирающая статистику по пейджингу (-B) и свопингу (-W), вводу-выводу (-b,-d), смонтированным системам (-F), прерываниям (-I), управлению питанием (-m), сети (-n), процессорам (-P,-u), очереди процессов и загрузке (-q,-w), памяти (-r), области подкачки (-s), терминалам (-y)
- Можно настроить сбор исторических результатов (crontab)
- Пример: sar -q 1 1 (*одно измерение за одну секунду*)

```
Linux 5.4.0-47-generic (ra)   01.10.2020   _x86_64_ (4 CPU)

14:35:36      runq-sz  plist-sz   ldavg-1   ldavg-5  ldavg-15   blocked
14:35:37            0      1034      1,44      1,70      1,75         0
Average:            0      1034      1,44      1,70      1,75         0
```

Стандартные средства для наблюдения счетчиков ядра (2)

- Процессор: ps, top, tiptop, turbostat, rdmsr, numastat, uptime
- Виртуальная память: vmstat, slabtop, pidstat, free
- Дисковая подсистема: iostat, iotop, blktrace
- Сеть: netstat, tcpdump, iptraf, ethtool, nicstat, ip
- Интерактивные (типа top) или с указанием количества запуска и интервала (типа sar)
- Некоторые работают только с правами root!

/proc

- Виртуальная файловая система, содержащая файлы статистики и управляющая модулями ядра
- find /proc |wc -l
  - 398401(over 9000!)
- Для начала:
  - /proc/cpuinfo - информация о процессоре (модель, семейство, размер кэша и т.д.)
  - /proc/meminfo - информация о памяти, размере области подкачки и т.д.
  - /proc/mounts - список смонтированных файловых систем.
  - /proc/devices - список устройств.
  - /proc/filesystems - поддерживаемые файловые системы.
  - /proc/modules - список загружаемых модулей.
  - /proc/version - версия ядра.
  - /proc/cmdline - список параметров, передаваемых ядру при загрузке.

Более подробно смотри:
kernel.org -- ver-- Documentation/filesystems/proc.rst

/proc (2)

- Можно получить много полезной информации о выполняющихся процессах

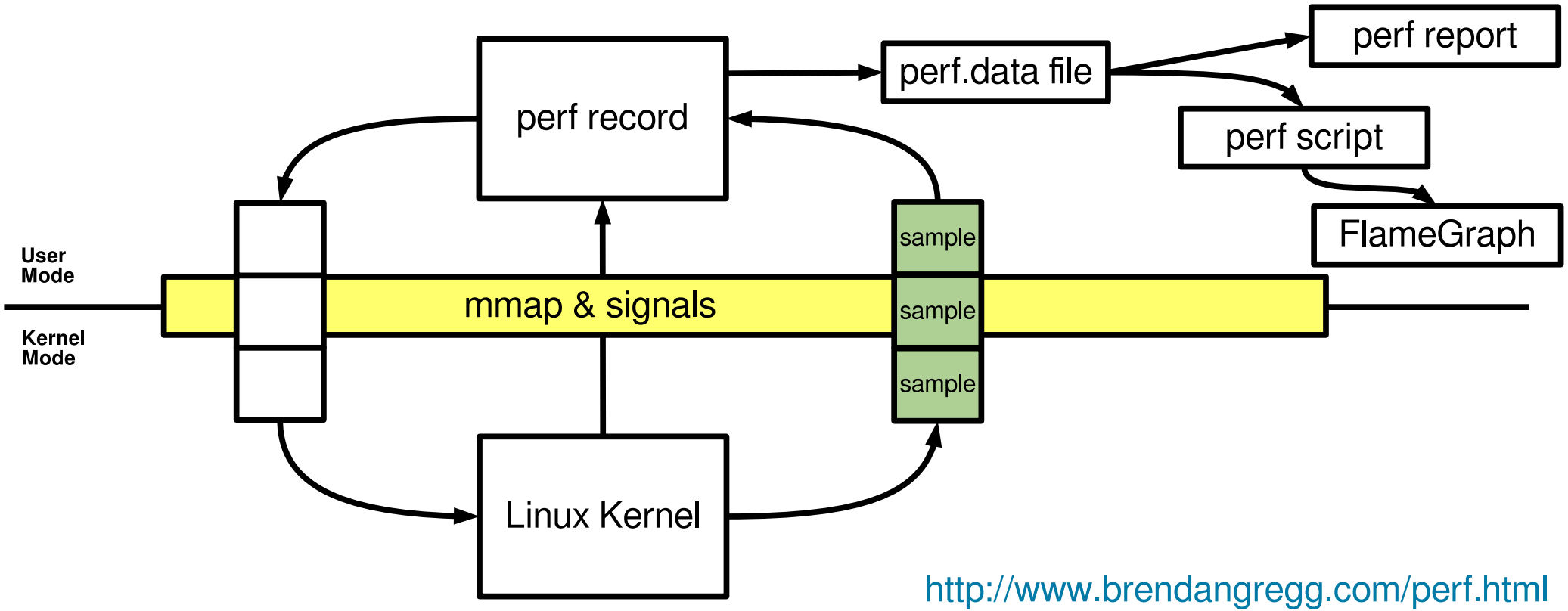
```
serge@ra:~$ echo $$
13318
serge@ra:~$ cd /proc/13318
serge@ra:/proc/13318$ ls -F
arch_status       cpuset   loginuid    numa_maps       sched          status
attr/             cwd@     map_files/  oom_adj         schedstat      syscall
autogroup         environ  maps        oom_score       sessionid      task/
auxv              exe@     mem         oom_score_adj   setgroups      timers
cgroup            fd/      mountinfo   pagemap         smaps          timerslack_ns
clear_refs        fdinfo/  mounts      patch_state     smaps_rollup   uid_map
cmdline           gid_map  mountstats  personality     stack          wchan
comm              io       net/        projid_map      stat
coredump_filter   limits   ns/         root@           statm
serge@ra:/proc/13318$ cat wchan
do_wait
```

Трассировщики

- Трассировка системных вызовов: strace
- Трассировка вызовов библиотек: ltrace
- Трассировка lock -оф(ф): bpftrace

```
serge@ra:/$ strace ls  #можно трассировать процесс с помощью -p pid_number
execve("/usr/bin/ls", ["ls"], 0x7fff46bdb930 /* 61 vars */) = 0
brk(NULL)                                       = 0x55d3d67f4000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffec23f8720) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=93197, ...}) = 0
mmap(NULL, 93197, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f54060f4000
close(3)                                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0@p\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=163200, ...}) = 0
```

Профилировщик perf (perf_events)



http://www.brendangregg.com/perf.html

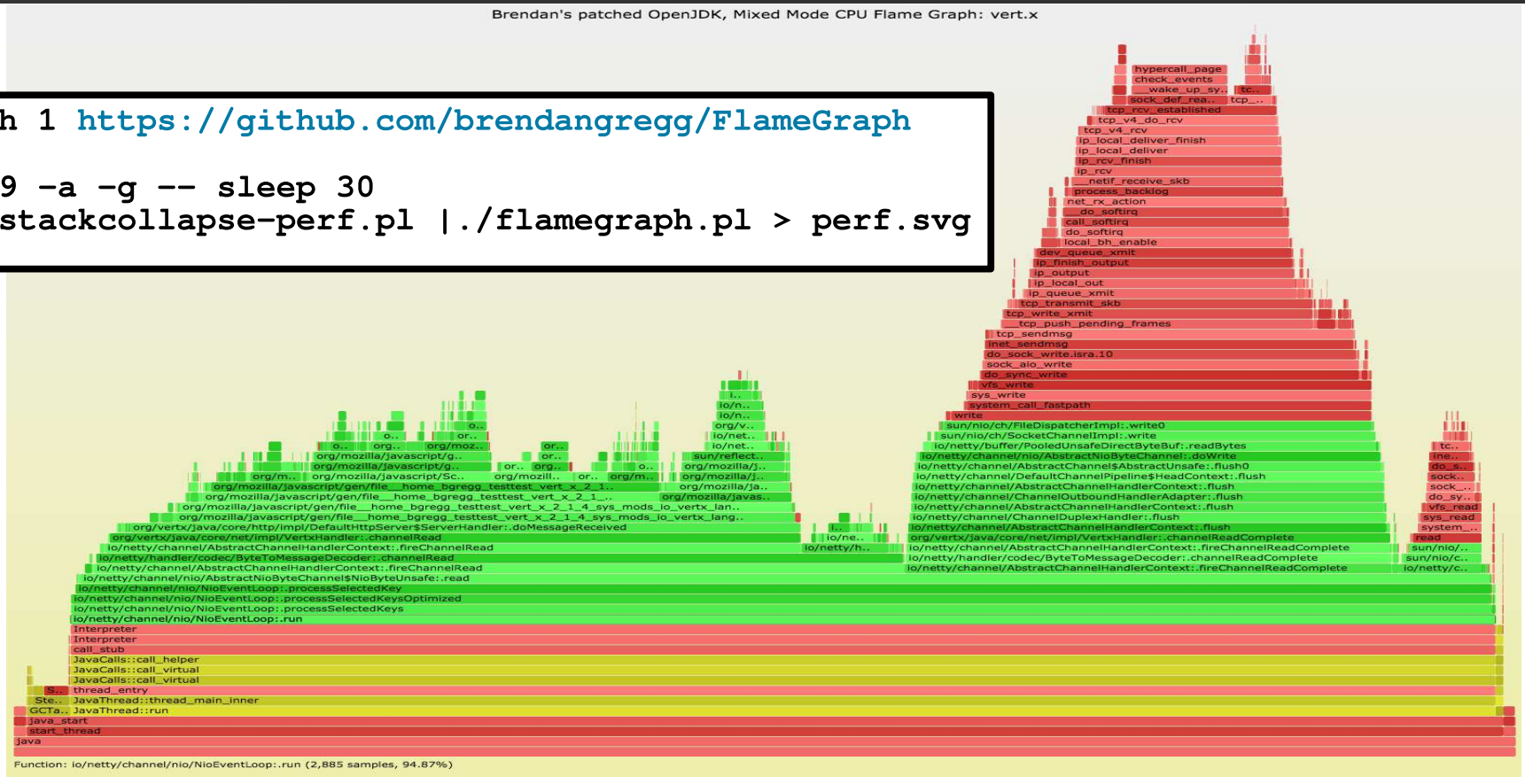
Профилировщик perf (user, kernel, h/w params)

- usage: perf [--version] [--help] [OPTIONS] COMMAND [ARGS]
- The most commonly used perf commands are:
- bench General framework for benchmark suites
- c2c Shared Data C2C/HITM Analyzer.
- config Get and set variables in a configuration file.
- data Data file related processing
- diff Read perf.data files and display the differential profile
- evlist List the event names in a perf.data file
- ftrace simple wrapper for kernel's ftrace functionality
- kallsyms Searches running kernel for symbols
- kmem Tool to trace/measure kernel memory properties
- list List all symbolic event types
- lock Analyze lock events
- mem Profile memory accesses
- record Run a command and record its profile into perf.data
- report Read perf.data and display the profile
- sched Tool to trace/measure scheduler properties (latencies)
- script Read perf.data and display trace output
- stat Gather performance counter statistics on command
- timechart Tool to visualize total system behavior
- top System profiling tool.
- probe Define new dynamic tracepoints
- trace strace inspired tool

FlameGraph

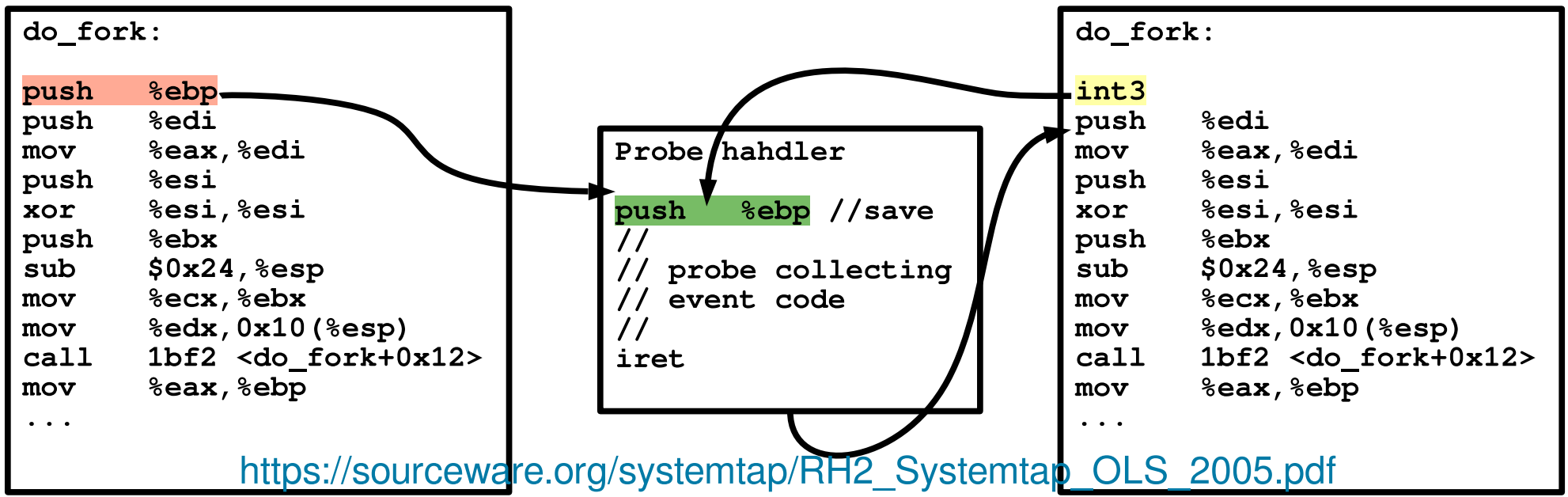
http://www.brendangregg.com/FlameGraphs/cpuflamegraphs.html

```
git clone --depth 1 https://github.com/brendangregg/FlameGraph
cd FlameGraph
perf record -F 99 -a -g -- sleep 30
perf script | ./stackcollapse-perf.pl |./flamegraph.pl > perf.svg
```



System tap

- Средства сбора статистки по kprobes и uprobes
- «Минимальное» воздействие на систему



https://sourceware.org/systemtap/RH2_Systemtap_OLS_2005.pdf

Использование SystemTap

- Скриптовый язык, похожий на AWK
- probe <event> { handler }
  - Event может быть: syscall.function, process.statement, timer.ms, begin, end, (tapset) aliases
  - Handler: управляющие структуры, переменные (числа, строки), ассоциативные массивы или статистические агрегаторы
  - Вспомогательные функции: log, printf, gettimeofday, pid
- Множество готовых скриптов для анализа

https://sourceware.org/systemtap/examples/keyword-index.html

stap пример

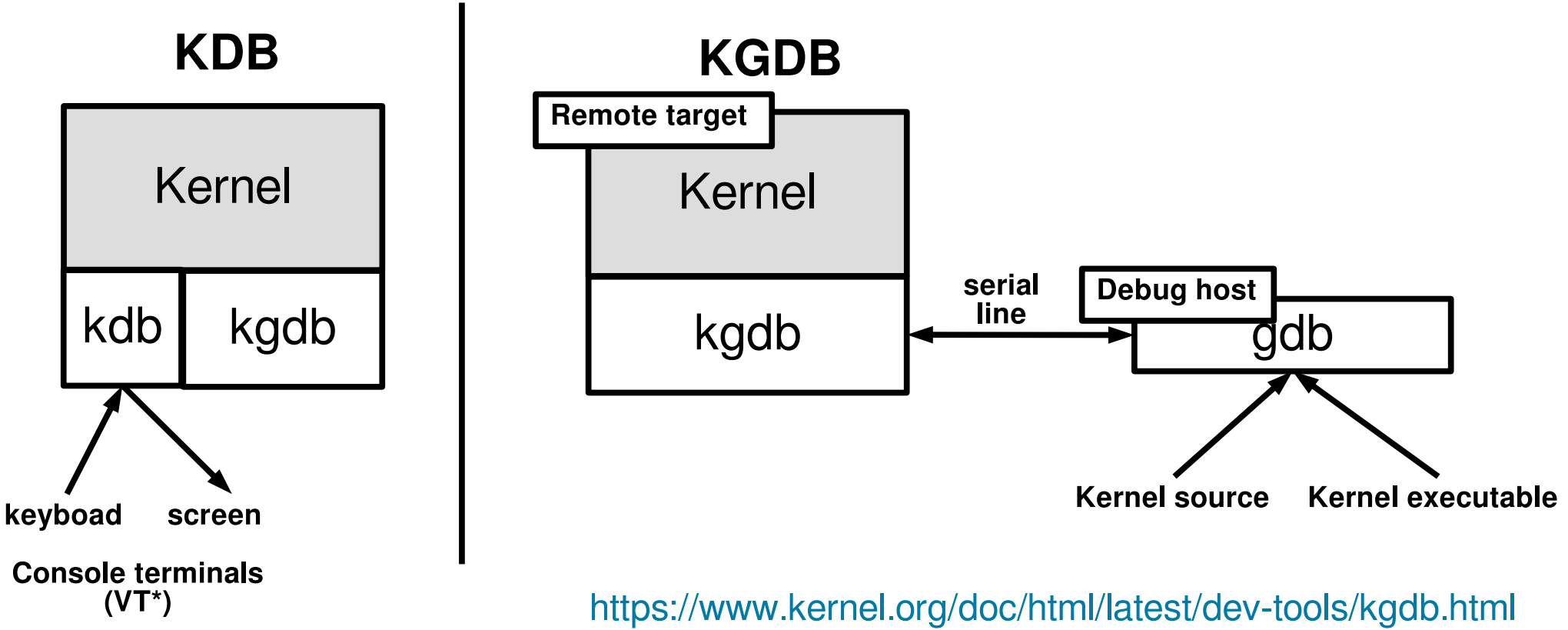
```
#! /usr/bin/env stap
global opens, reads, writes, totals

probe begin {
     printf("starting probe\n")
}

probe syscall.open {
  opens[execname()] <<< 1
}

probe syscall.read.return {
  count = retval
  if ( count >= 0 ) {
    reads[execname()] <<< count
    totals[execname()] <<< count
  }
}

probe syscall.write.return {
  count = retval
  if (count >= 0 ) {
    writes[execname()] <<< count
    totals[execname()] <<< count
  }
}
```

```
probe end {
  printf("\n%16s %8s %8s %8s %8s %8s %8s %8s\n",
    "", "", "", "read", "read", "", "write", "write")
  printf("%16s %8s %8s %8s %8s %8s %8s %8s\n",
    "name", "open", "read", "KB tot", "B avg",
    "write", "KB tot", "B avg")
  # sort by total io
  foreach (name in totals @sum- limit 20) {
    printf("%16s %8d %8d %8d %8d %8d %8d %8d\n",
      name, @count(opens[name]),
      @count(reads[name]),
      (@count(reads[name]) ? @sum(reads[name])>>10 : 0 ),
      (@count(reads[name]) ? @avg(reads[name]) : 0 ),
      @count(writes[name]),
      (@count(writes[name]) ? @sum(writes[name])>>10 : 0 ),
      (@count(writes[name]) ? @avg(writes[name]) : 0 ))
  }
}
```

stap запуск примера

```
serge@ra:/tmp$ sudo stap file.stap
starting probe
^C
                                          read      read                write     write
              name      open      read   KB tot    B avg     write   KB tot    B avg
 localStorage DB         0         0        0        0         7       96    14053
 IndexedDB #1923         0         8       25     3212         6        8     1375
     gnome-shell         0       166       13       86       133        1        8
     Web Content         0      7581        7        1      4769        4        1
      irqbalance         0        18        4      256         1        0        8
            zoom         0         7        4      635         0        0        0
 pool-gnome-shel         0        16        3      196        40        0        8
     GraphRunner         0         0        0        0      1924        1        1
      MainThread         0       958        0        1       519        0        1
     InputThread         0        19        1       72        19        0        1
            acpid         0        57        1       24         0        0        0
            gdbus         0        48        0        8        95        0        8
            Timer         0         0        0        0       969        0        1
 Chrome_~dThread         0       449        0        1       415        0        1
             Xorg         0        39        0       16         0        0        0
  Gecko_IOThread         0       440        0        1       143        0        1
 IPDL Background         0         0        0        0       434        0        1
     threaded-ml         0        32        0        5        72        0        2
```

Отладчик ядра



https://www.kernel.org/doc/html/latest/dev-tools/kgdb.html

Вызов KDB

- Загрузить ядро с параметром kgdboc=kbd (через /etc/default/grub или меню загрузки)

```
root@ra:~# cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.4.0-47-generic
root=UUID=cb68f380-c1a3-4757-a4f2-73b76b9e0934
ro ipv6.disable=1 kgdboc=kbd
```

- Запустить с виртуального терминала триггер переключения

Система встанет!

```
root@ra:~# echo g > /proc/sysrq-trigger
Entering kdb ..
[1]kdb> ?
[1]kdb> go
```

Средства для отладки Windows

1.10

- Встроенные средства
- SysInternals
- Отладчик ядра

Отладочные стредства Windows

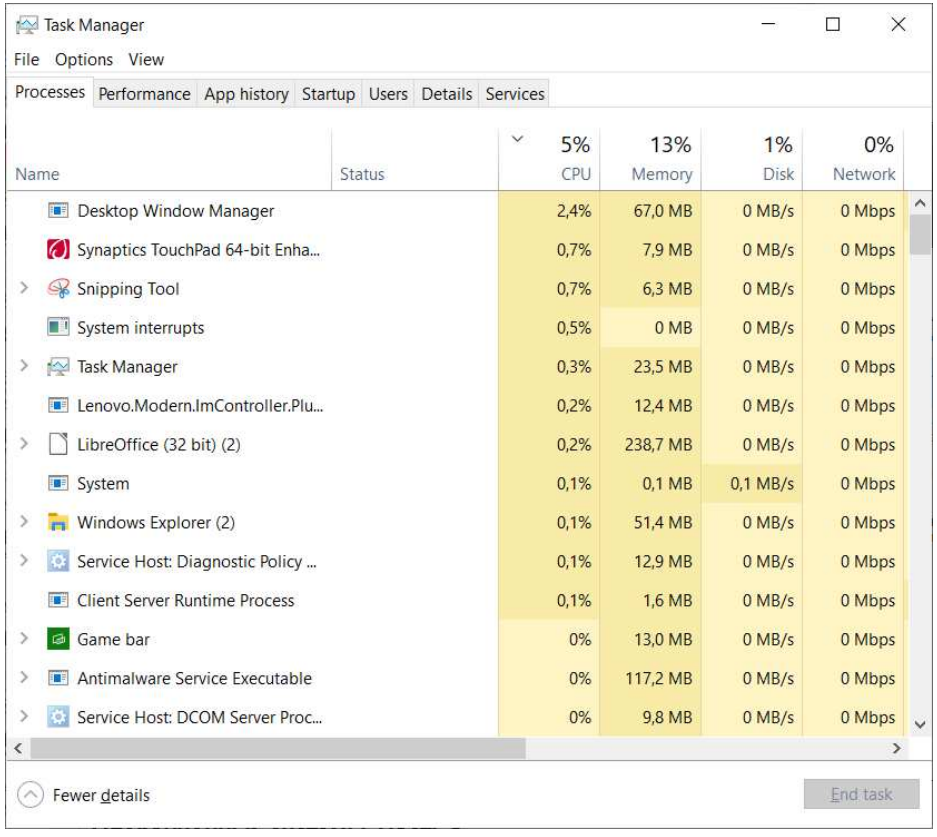
- Встроенные стандартные средства
- Windows SDK
  - Отладчики, множество утилит, поддерживающих сборку приложений
  - DTrace on Windows

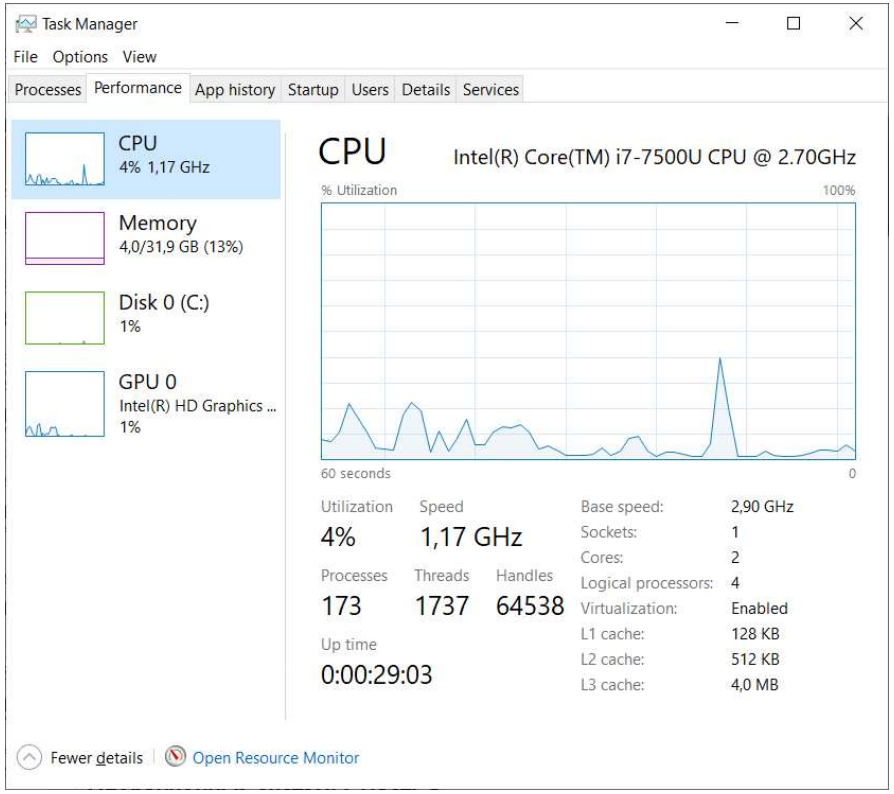
    https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest/dtrace
- SysInternals
- Сторонние средства
  - https://checkpanel.com/resources/windows-server-performance-monitoring-tools

Стандартные средства Windows

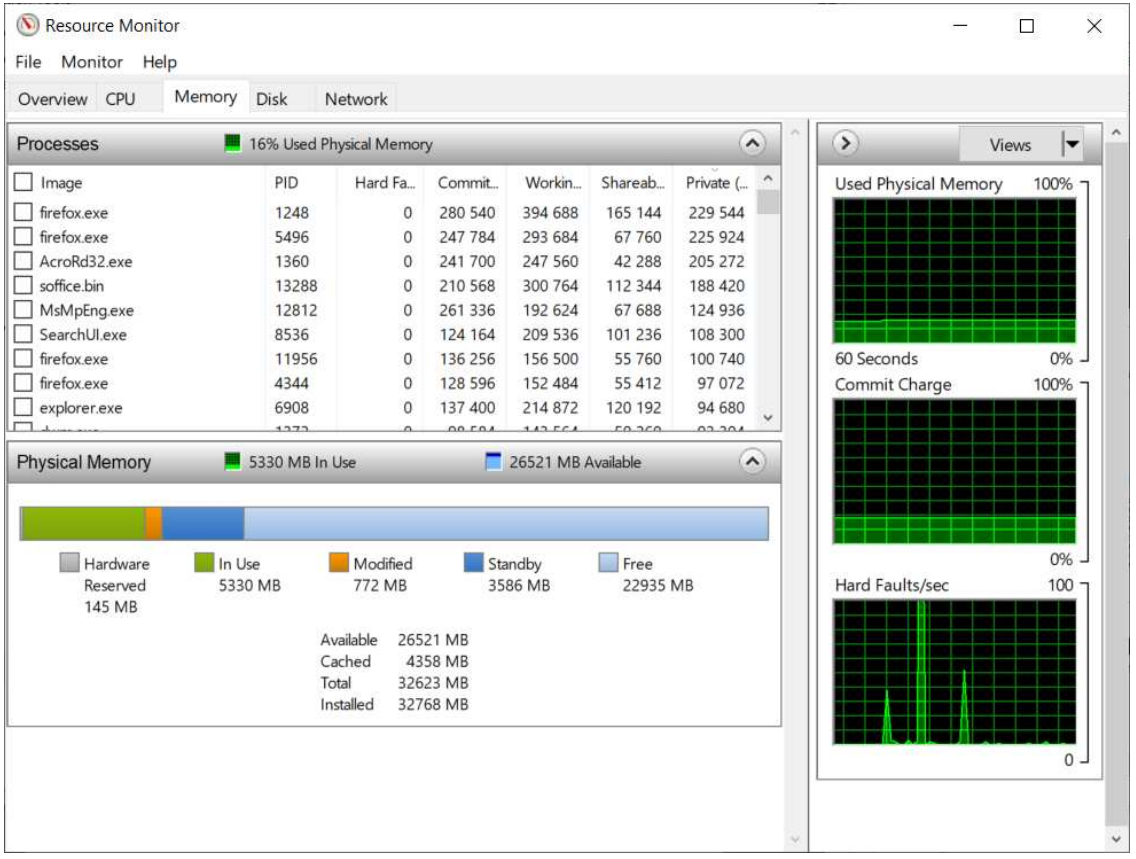
- Находятся в ControlPanel
- В большинстве случаев — средства изменения конфигурации системы и наблюдения за ее поведением
- Control Panel → System and Security → Administrative Tools
- Disk Cleanup, Performance Monitor, Resource Monitor, Registry Editor, Services, System Configuration, ...

Task Manager



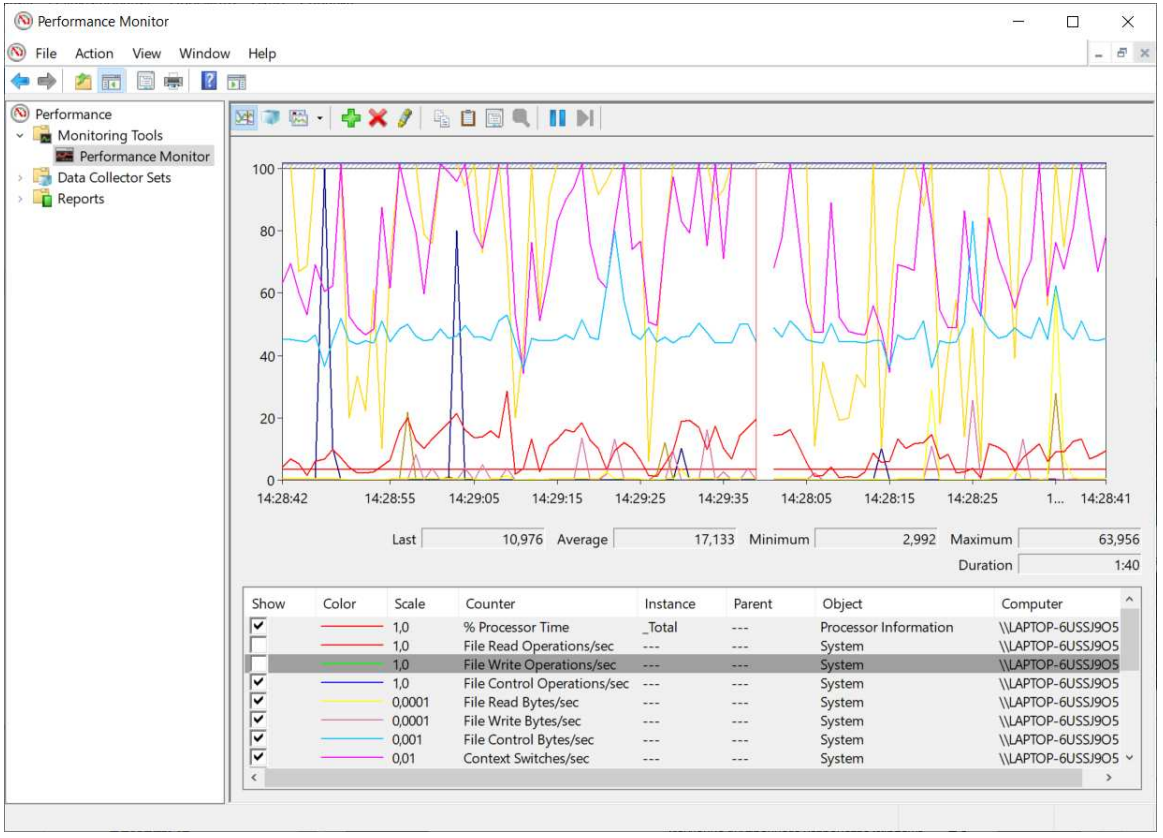


Resource Monitor



Операционные системы. Часть 1. Обзор операционных систем

Performance Monitor



SysInternals

- Множество скриптов и программ для получения информации о системе
- Автор — Марк Руссинович, в настоящее время сотрудник Microsoft (соавтор книги Windows Internals)
- Отдельно загружается и устанавливается с сайта Microsoft

https://docs.microsoft.com/ru-ru/sysinternals/

Top SysInternals utils

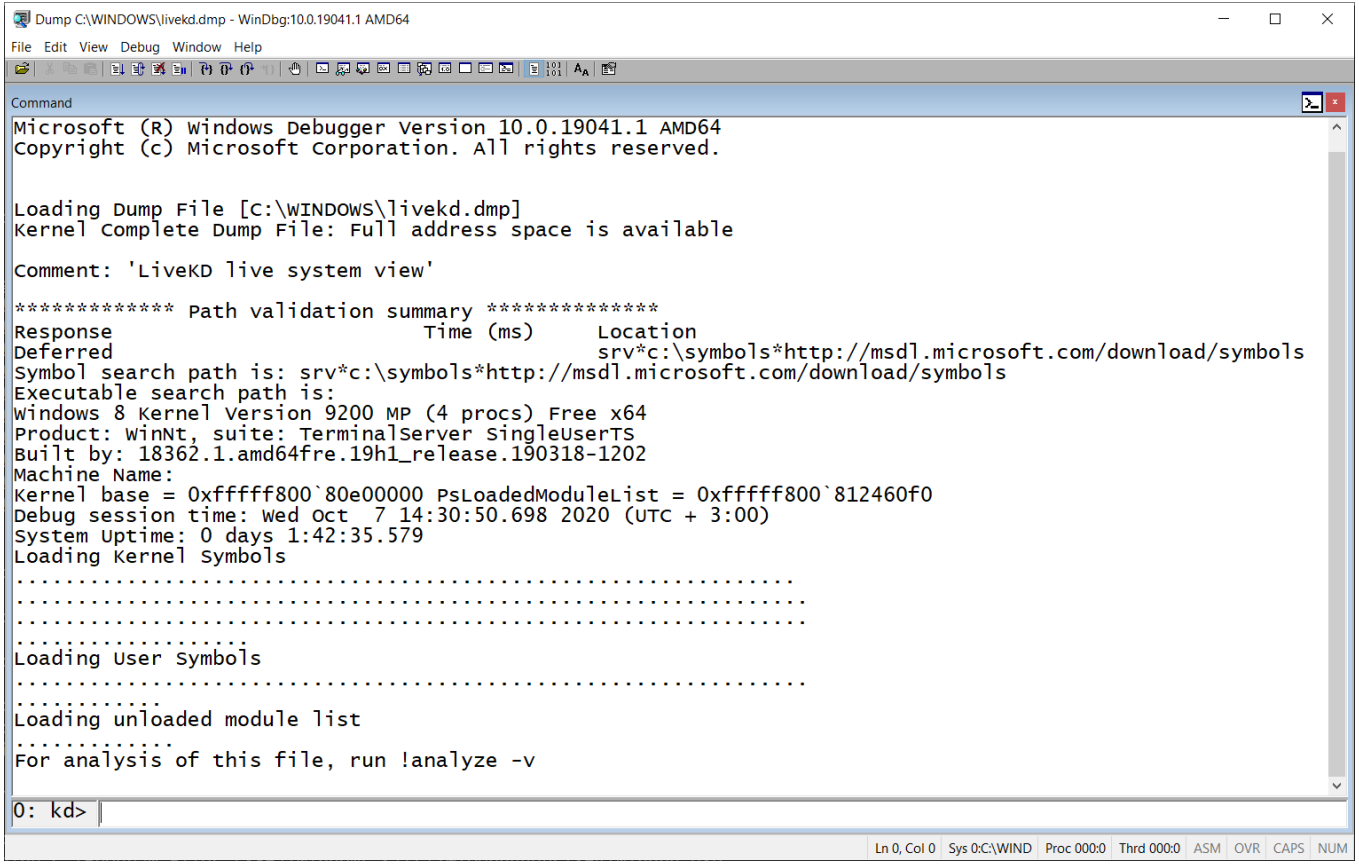
- PsList and PsKill — просмотр и остановка процессов (в том числе и удаленно)
- Process Explorer — просмотр ресурсов процесса, замена Task Manager
- Process Monitor — просмотр связанных с процессом ресурсов реестра
- Autoruns — поиск автозапускаемых программ
- Contig — дефрагментирует конкретный файл
- PSFile — позволяет показать открытые файлы, в том числе и удаленно
- MoveFile — перемещает заблокированные файлы во время перезагрузки.
- Sync — синхронизация файловой системы
- TCPview — информация о открытых сетевых соединениях
- SDelete — удалить файлы и папки без возможности восстановления

Отладчик WinDbg и KD

- Требуют загрузки символьной информации о ядре
- Без livekd (SysInternals) требуют перезагрузки ядра в отладочном режиме
- Требуют привилегий администратора

```
set _NT_SYMBOL_PATH=srv*c:\symbols*http://msdl.microsoft.com/download/symbols

C:\Program Files (x86)\SysinternalsSuite>livekd64.exe -w -k "C:\Program Files
(x86)\Windows Kits\10\Debuggers\x64\windbg.exe"
```

WinDBG.exe view



Конец первой части

?