

Справочная информация по системе Linux

Алексей Полухин

2023

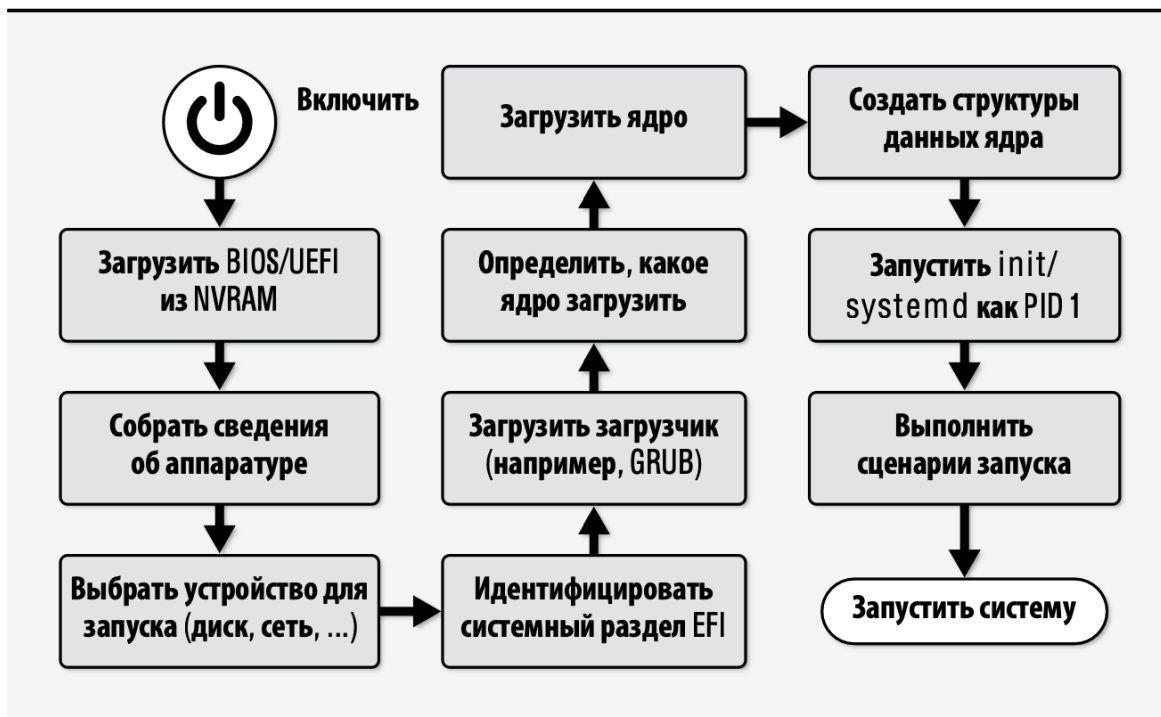


Рис. 1: Процессы загрузки Linux и Unix

1 Справочная информация. Полезные сведения

Демон в Linux – фоновый процесс. Программа на уровне пользователя

Киби-, меби-, гибби- байты

2 Операционные системы

2.1 Ядро ОС

Ядро ОС – центральная часть ОС, обеспечивающая приложениям координированный доступ к ресурсам компьютера Рис. 2. В современных ОС приложение не может напрямую обратиться к ресурсам компьютера, поэтому приложения обращаются к **ядру**.

Архитектура ядра операционной системы — это структура и дизайн основной части операционной системы, которая обеспечивает основные функции управле-



Рис. 2: Ядро ОС

ния ресурсами компьютера, планирование выполнения задач, обработку прерываний и обеспечивает взаимодействие между аппаратными устройствами и пользовательскими процессами.

В ОС, основанных на ядре, приложения имеют собственные независимые окружения. То есть свои участки памяти, своё процессорное время, свой доступ к устройствам ввода и вывода.

Современные ОС имеют пространство ядра (где идёт работа с оборудованием) и пространство пользователя.

Архитектуры ядер

- Монолитное ядро (самое быстрое) — Linux
- Микроядро (самое отказоустойчивое)
- Гибридное ядро — Windows

Ближе к оборудованию — быстрее, ближе к пространству пользователя — стабильнее. (Интересная аналогия: Python, C++, assembler)

Создание ОС с ядром возможно, когда на аппаратном уровне появляются кольца защиты. (методы разграничения ресурсов компьютера)

Кольца защиты — аппаратная реализация механизма разграничения ресурсов компьютера

Интерпретатор команд — это обычное приложение. В Linux они бывают разные: *shell, bash, ksh, csh, psh* . . . Он не является частью ядра ОС.

Для Linux не существует расширений файлов, они сделаны исключительно для удобства пользователя и для программ, которые взаимодействуют с этими файлами

Всё в Unix/Linux — это файл, просто поток байт

2.2 *UNIX*

Философия UNIX

1. Пишите программы, которые делают что-то одно, но хорошо
2. Пишите программы, которые работают вместе
3. Пишите программы, которые поддерживали бы текстовые потоки, так как это универсальный интерфейс

POSIX — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (в *UNIX*). (Средство общения с ядром ОС). Обеспечивает совместимость UNIX-подобных ОС.

Когда разработчики создают программы, используя стандарт POSIX, эти программы могут работать на разных операционных системах без необходимости изменять их код. Это облегчает перенос программного обеспечения с одной системы на другую, так как программы, сделанные в соответствии с POSIX, будут использовать одни и те же команды и функции, доступные во всех системах, поддерживающих этот стандарт.

POSIX описывает работу в пространстве пользователя. (видимо, именно из-за этого команды в терминалах MacOS, Unix и Linux совпадают)

Команды вроде `ls` (список файлов и директорий) и `pwd` (текущая рабочая директория) являются стандартными командами, определенными в стандарте POSIX.

POSIX определяет интерфейс и поведение для командной оболочки и других системных команд в операционных системах, таких как UNIX, Linux, macOS и другие, чтобы обеспечить переносимость программ между различными системами.

Сейчас UNIX используется в серверах и мейнфреймах

Мейнфрейм (Mainframe) — это тип большого и мощного компьютера, который обычно используется в корпоративных средах для обработки больших объемов данных и критически важных бизнес-приложений. (Например, используется в центрах управления (космическими) полётами, в банках, на биржах — в отраслях, где важна каждая *наносекунда*). (Используется в критически важных задачах, где важна скорость и бесперебойность)

На всех суперкомпьютерах установлен Linux, потому что они решают сложные математические задачи, если они вдруг остановятся, ничего критического не произойдёт.

2.3 Структура каталогов в Linux

См. Рис. 3

Основные каталоги

- `/` — root
- `/bin` — Необходимые утилиты, необходимые при работе всем пользователям (и в однопользовательском режиме)
- `/boot` — загрузочные файлы (файлы загрузчика, ядро, `initrd`, `System.map`)

- `/dev` — основные файлы устройств
- `/etc` — общесистемные конфигурационные файлы (настройки ОС и служб ОС)
- `/home` — домашние каталоги пользователей (их персональные настройки и данные)
- `/lib` — Основные библиотеки, необходимые для работы программ из `/bin` и `/sbin`
- `/media` — Каталог, где производится монтирование сменных носителей (USB, CD-ROM)
- `/mnt` — каталог содержит временно монтирование файловые системы
- `/opt` — Дополнительные программное обеспечение
- `/proc` — каталог, которые содержит информацию о всех процессах в нашей ОС
- `/root` — домашний каталог пользователя *root*
- `/run` — информация о системе с момента её загрузки (что запущено и чем это работает)
- `/sbin` — основные системные исполняемые файлы (основные программы для настройки и администрирования системы, *init*, *ifcondif*, *iptables*)
- `/srv` — данные для сервисов, представляемых системой (*www* или *ftp*)
- `/sys` — содержит информацию об устройствах, драйверах и некоторых свойствах ядра
- `/tmp` — временные файлы

- /usr — Большинство пользовательских приложений и утилит (используемых в многопользовательском режиме)
- /var — изменяемые файлы. (файлы регистрации, временные почтовые файлы, файлы спулера)
- /var/log — логи
- /home/username — домашний каталог пользователя

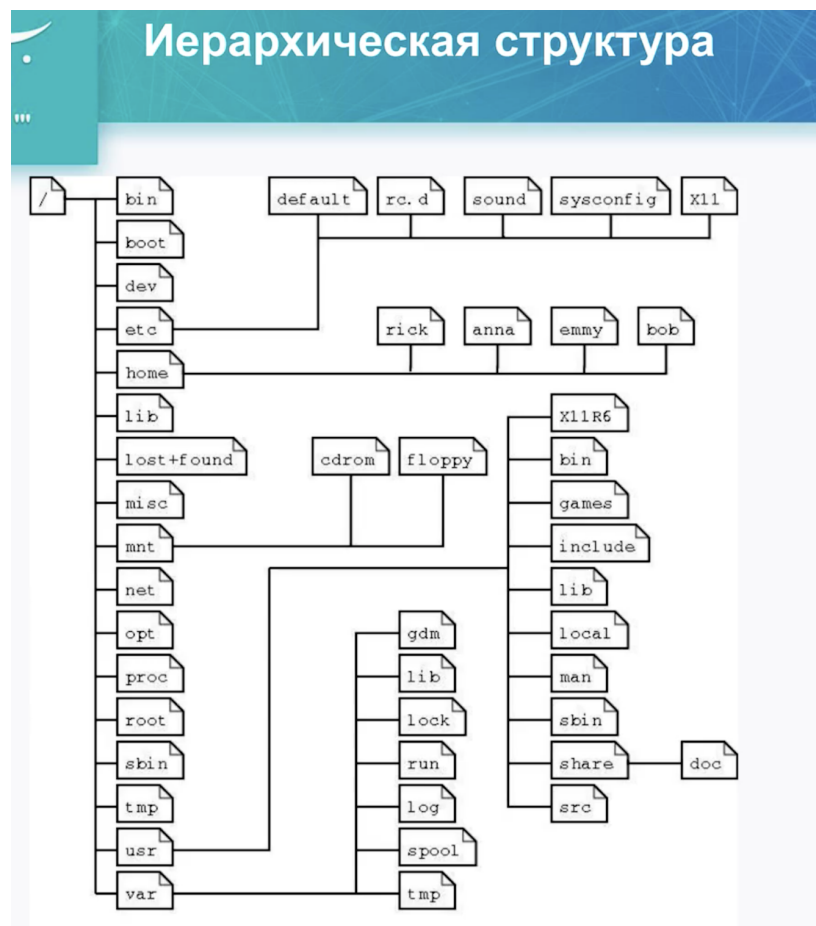


Рис. 3: Структура каталогов в Linux

3 Полезные bash-команды

\$ man -k word — ищет в документации ключевое слово word

3.1 Пример сборки кода из исходников

```
$ git clone https://github.com/the-tcpdump-group/tcpdump.git
```

```
$ cd tcpdump
```

```
$ ./configure
```

```
$ make
```

```
$ sudo make install
```

3.2 Зависимости

Wants	Модули, которые должны быть активированы одновременно, если это возможно, но не обязательно
Requires	Строгие зависимости; отказ от каких-либо предварительных условий прекращает работу этой службы
Requisite	Аналогично <i>Requires</i> , но модуль должен быть активным
BindsTo	Аналогично <i>Requires</i> , но модуль должен быть связан еще более тесно
PartOf	Аналогично <i>Requires</i> , но влияет только на запуск и остановку
Conflicts	Отрицательные зависимости; не может взаимодействовать с этими единицами

Таблица 1: Явные зависимости

4 Полезные слайды

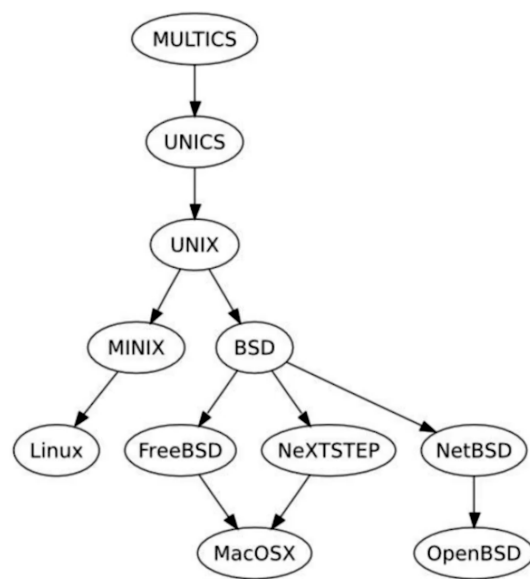


Рис. 4: Развитие операционных систем