

Тема 1.4 Ядро операционной системы, загрузка ОС система инициализации

План занятия

1. Архитектура ядра операционной системы.
2. Виды ядра
2. Загрузка ОС
3. Инициализация системы
4. Итоги
5. Домашнее задание

По итогу занятия вы получите представление о внутреннем устройстве ядра Linux и научитесь работать с модулями ядра. Как происходит загрузка ОС и ее инициализация.

Виды архитектуры ядер ОС

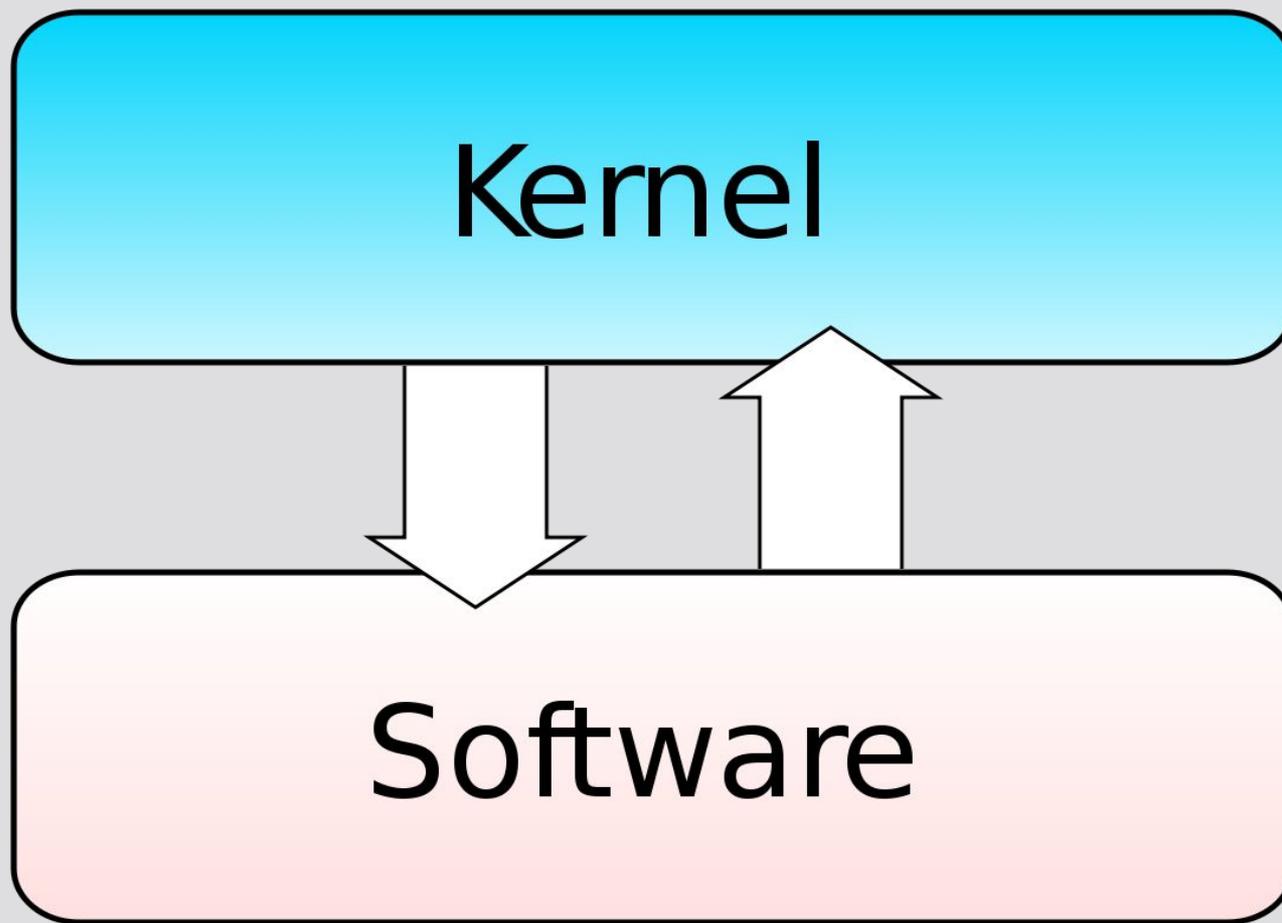
1. Монолитное ядро предоставляет богатый набор абстракций оборудования. Все части монолитного ядра работают в одном адресном пространстве. Это такая схема операционной системы, при которой все компоненты её ядра являются составными частями одной программы, используют общие структуры данных и взаимодействуют друг с другом путём непосредственного вызова процедур. Монолитное ядро — старейший способ организации операционных систем. Примером систем с монолитным ядром является большинство UNIX-систем.

Достоинства: Скорость работы, упрощённая разработка модулей. **Недостатки:** Поскольку всё ядро работает в одном адресном пространстве, сбой в одном из компонентов может нарушить работоспособность всей системы.

Примеры: Традиционные ядра UNIX (такие как BSD), Linux; ядро MS-DOS, ядро KolibriOS.

Монолитные ядра имеют долгую историю развития и усовершенствования и, на данный момент, являются наиболее архитектурно зрелыми и пригодными к эксплуатации. Вместе с тем, монолитность ядер усложняет их отладку, понимание кода ядра, добавление новых функций и возможностей, удаление «мёртвого», ненужного, унаследованного от предыдущих версий кода. «Разбухание» кода монолитных ядер также повышает требования к объёму оперативной памяти, требуемому для функционирования ядра ОС. Это делает монолитные ядерные архитектуры малоприспособными к эксплуатации в системах, сильно ограниченных по объёму ОЗУ, например, встраиваемых системах, производственных микроконтроллерах и т. д.

Виды архитектуры ядер ОС



Архитектуры ядра (что хранить в ядре; какие функции необходимо хранить в ядре):



Архитектуры ядра

2. Многослойная монолитная архитектура (модульная).

Модульное ядро — современная, усовершенствованная модификация архитектуры монолитных ядер операционных систем. В отличие от «классических» монолитных ядер, модульные ядра, как правило, не требуют полной перекомпиляции ядра при изменении состава аппаратного обеспечения компьютера. Вместо этого модульные ядра предоставляют тот или иной механизм подгрузки модулей ядра, поддерживающих то или иное аппаратное обеспечение (например, драйверов). При этом подгрузка модулей может быть как динамической (выполняемой «на лету», без перезагрузки ОС, в работающей системе), так и статической (выполняемой при перезагрузке ОС после переконфигурирования системы на загрузку тех или иных модулей).

Все модули ядра работают в адресном пространстве ядра и могут пользоваться всеми функциями, предоставляемыми ядром. Поэтому модульные ядра продолжают оставаться монолитными. Модульность ядра осуществляется на уровне бинарного образа, а не на архитектурном уровне ядра, так как динамически подгружаемые модули загружаются в адресное пространство ядра и в дальнейшем работают как интегральная часть ядра. Модульные монолитные ядра не следует путать с архитектурным уровнем модульности, присущий микроядрам и гибридным ядрам. Практически, динамичная загрузка модулей, это просто более гибкий способ изменения образа ядра во время выполнения — в отличие от перезагрузки с другим ядром. Модули позволяют легко расширить возможности ядра по мере необходимости.

Модульные ядра удобнее для разработки, чем традиционные монолитные ядра, не поддерживающие динамическую загрузку модулей, так как от разработчика не требуется многократная полная перекомпиляция ядра при работе над какой-либо его подсистемой или драйвером. Выявление, локализация, отладка и устранение ошибок при тестировании также облегчаются. Примером может служить VFS — «виртуальная файловая система», совместно используемая многими модулями файловых систем в ядре Linux.

Архитектуры ядра

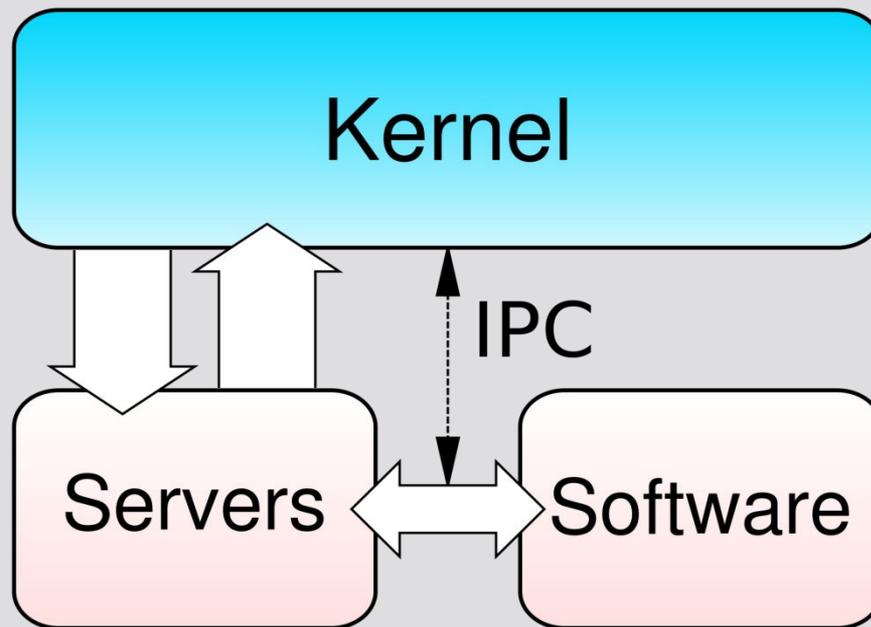


Архитектуры ядра (что хранить в ядре; какие функции необходимо хранить в ядре):

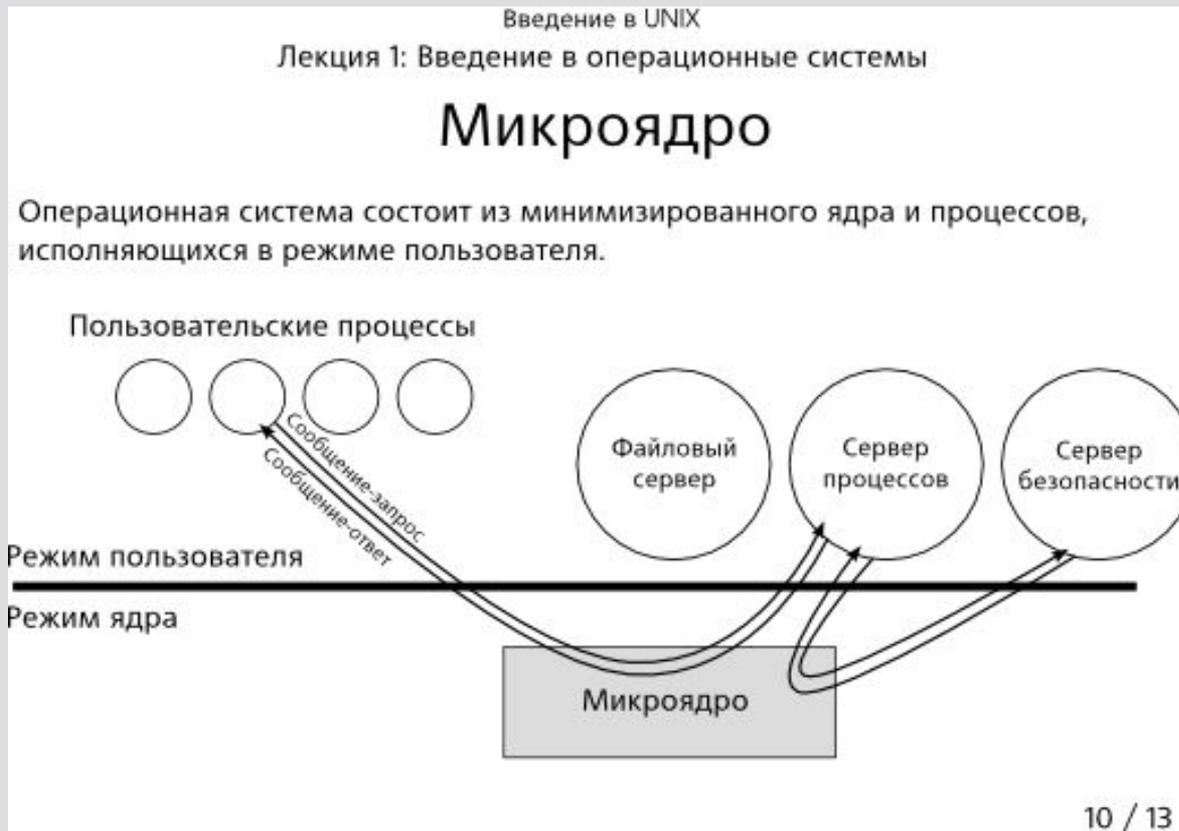
3. Микроядерная архитектура - в этом типе архитектуры ядра основные пользовательские службы, такие как управление драйверами устройств, управление стеком протоколов, управление файловой системой и управление графикой, присутствуют в пространстве пользователя, а остальные функции управление памятью, управление процессами присутствует внутри пространства ядра. Таким образом, всякий раз, когда система имеет потребность в услугах, присутствующих в пространстве ядра, ОС переключается в режим ядра, а для служб пользовательского уровня она переключается в режим пользователя. Этот тип архитектуры ядра уменьшает размер ядра, но скорость выполнения процессов и предоставления других услуг значительно ниже, чем у монолитных ядер.

Архитектуры ядра (что хранить в ядре; какие функции необходимо хранить в ядре):

3. Микроядерная архитектура



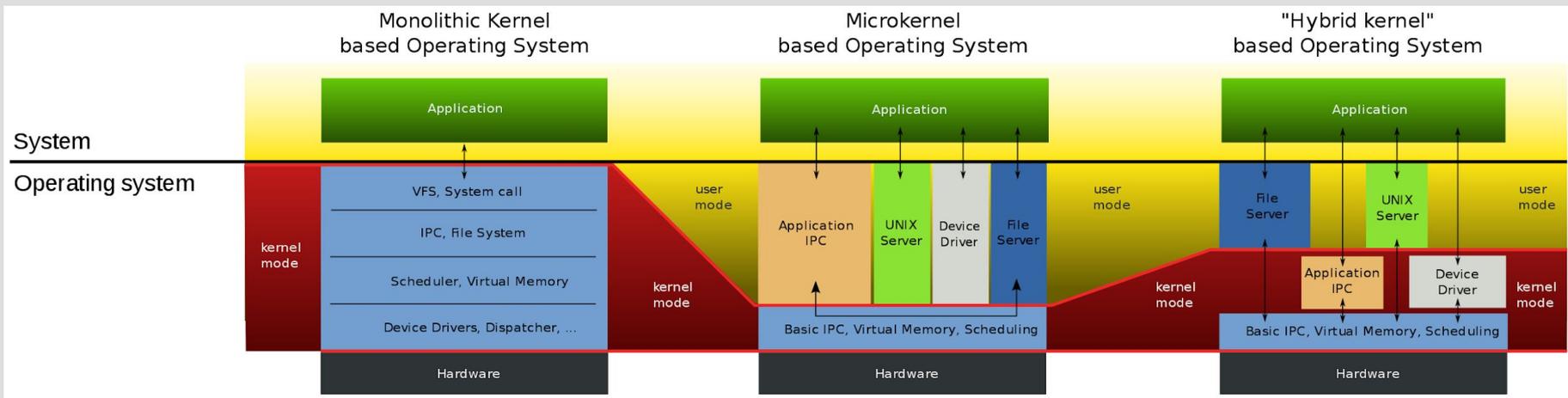
Архитектуры ядра (что хранить в ядре; какие функции необходимо хранить в ядре):



Архитектуры ядра (что хранить в ядре; какие функции необходимо хранить в ядре):

4. **Нано ядерная архитектура** - в ядро входят только машинно-зависимые модули. Данная архитектура используется в виртуализированной ОС
 5. **Экза ядерная архитектура** (в режиме ядра работают только менеджеры ресурсов). Идея применяются в гетерогенных вычислительных системах
 6. **Гибридное ядро** ([англ. Hybrid kernel](#)) — модифицированные [микроядра](#), позволяющие для ускорения работы запускать модули [ОС](#) в пространстве [ядра](#). Для наилучшей производительности системы нам требуется как высокая скорость, так и малый размер ядра, чтобы наша система могла иметь максимальную эффективность. Поэтому для решения этой задачи был разработан новый тип ядра, который представлял собой комбинацию монолитного ядра и микроядра. Этот тип ядра известен как гибридное ядро. Такой тип архитектуры используется практически во всех системах, которые производятся в настоящее время.
-

Архитектуры ядра (что хранить в ядре; какие функции необходимо хранить в ядре):



Краткое заключение по ядрам ОС

Монолитные ядра – все функции операционной системы загружены в **единый файл** и выполняются в качестве одного процесса в одном адресном пространстве.

Микроядра – все функции ядра разделяются на **несколько процессов**, которые обычно называют серверами. Все серверы поддерживаются независимыми друг от друга и выполняются каждый в своем адресном пространстве.

Гибридные ядра – различные **комбинации**, совмещающие оба вышеуказанных подхода.

Краткое заключение по ядрам ОС

	Плюсы	Минусы
Микроядра	<ul style="list-style-type: none">● При ошибке одного из серверов можно избежать падения всей системы● Запуск процессов в режиме ядра только тех серверов, которым это необходимо● Низкое потребление памяти	<ul style="list-style-type: none">● Много накладных расходов на обеспечение взаимодействия между серверами● Процессы должны ждать свою очередь, чтобы получить информацию
Монолитные	<ul style="list-style-type: none">● Прямой запуск функций● Производительность	<ul style="list-style-type: none">● При любом изменении требуется пересборка всего ядра● Любая ошибка в любой подсистеме приводит к kernel panic

Краткое заключение по ядрам ОС

Ядро Linux

На сегодняшний день Linux — **монолитное ядро с поддержкой загружаемых модулей.**

- Драйверы устройств и расширения ядра обычно запускаются в нулевом кольце защиты, с полным доступом к оборудованию.
 - Все драйверы и подсистемы работают в своем адресном пространстве, отделенном от пользовательского.
 - В **отличие от обычных монолитных ядер, драйверы** устройств легко собираются в виде модулей и загружаются или выгружаются во время работы системы.
-

Модули ядра Linux

- любой дополнительный функционал может быть загружен в виде модулей;
- Модули хранятся в системе в директории **/lib/modules/**;
- При работе с модулями используются утилиты **lsmod, insmod, modprobe и modinfo**.

Работы с модулями

- `lsmod` — информация обо всех загруженных модулях
 - `modinfo <MODULE-NAME>` — просмотр информации о модуле
 - `modprobe <MODULE-NAME>` — загрузка модуля
 - `insmod /lib/modules/.../<MODULE-NAME>.ko` — загрузка модуля с помощью `insmod`
 - `rmmod <MODULE-NAME>` — выгрузка модуля
-

Подсистемы ядра

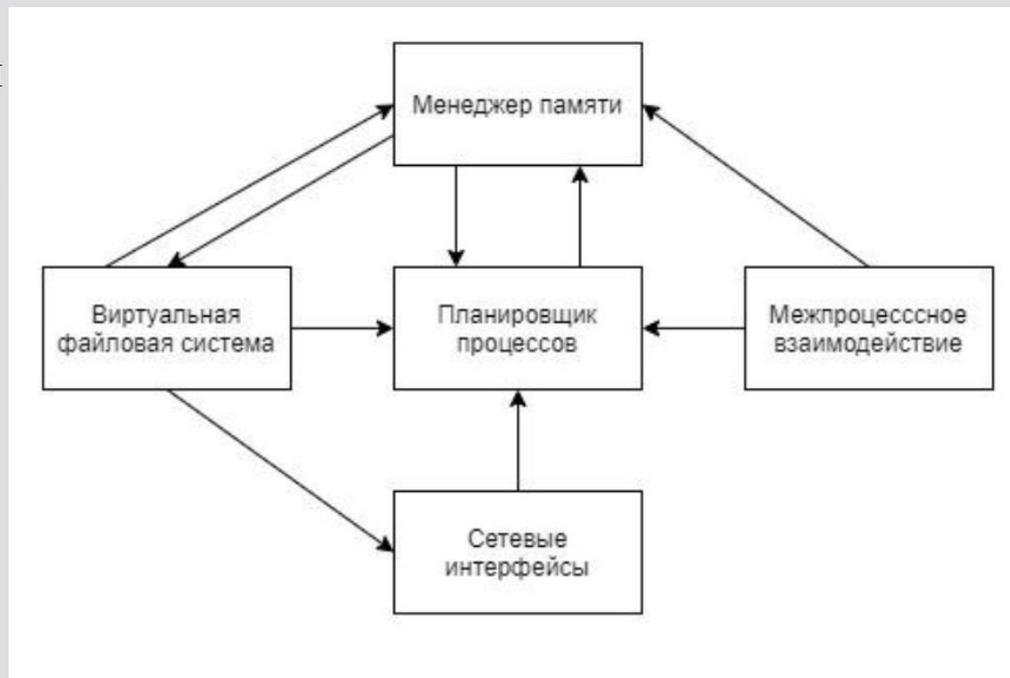
- **Process Scheduler (SCHED)** — планировщик процессов, отвечает за контроль над доступом процессов к CPU;
 - **Memory Manager (MM)** — менеджер памяти, обеспечивает различным процессам безопасный доступ к основной памяти системы;
 - **Virtual File System (VFS)** — виртуальная файловая система, создает абстрактный слой, скрывая детали оборудования, предоставляя общий файловый интерфейс для всех устройств;
 - **Network Interface (NET)** — сетевые интерфейсы, обеспечивает работу с различными сетевыми стандартами и сетевым оборудованием;
 - **Inter-Process Communication (IPC)** — межпроцессная подсистема, поддерживающая несколько механизмов для process-to-process связей в единой Linux-системе.
-

Зависимости подсистем друг от друга

Планировщик процессов — основная подсистема.

Все остальные зависят от нее, так как всем им необходимо приостанавливать и возобновлять выполнение процессов.

Например, когда процесс пытается отправить некое сообщение по сети, сетевой интерфейс может приостановить выполнение процесса, пока сетевое оборудование выполняет отправку сообщения.



Выполнение в режиме ядра

Передача ядру контроля над процессом для выполнения необходимой задачи.

Например, открытие файла или передачи данных по сети.

Существует 2 события, при которых выполнение переходит в режим ядра:

- *системные вызовы;*

Поступают от пользовательских программ и касаются работы с устройствами, памятью и т.п.

- *аппаратные прерывания.*

Сигнализируют об окончании какого-либо действия со стороны устройства или о возникшей на устройстве ошибке.

СИСТЕМНЫЕ ВЫЗОВЫ

Системный вызов — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.

Примеры системных вызовов:

open, read, write, close.

Документация доступна во втором разделе man:

man 2 <syscall-NAME> — поиск документации по системному вызову.

Strace — утилита для отслеживания системных вызовов при выполнении процесса.

strace <команда> — поиск документации по системному вызову..

Аппаратные прерывания

1. Когда ЦПУ получает прерывание, он останавливает любые Процессы

○ Если это не более приоритетное прерывание, тогда обработка пришедшего прерывания произойдет только тогда, когда более приоритетное будет завершено.

2. Сохраняет некоторые параметры в стеке

3. Вызывает обработчик прерывания.

➔ Это означает, что не все действия допустимы внутри обработчика прерывания, потому что система находится в неизвестном состоянии.

ВЫВОДЫ ПО ПОДТЕМЕ

- рассмотрели архитектуру ядра;
 - рассмотрели основные виды ядер;
 - рассмотрели модули ядра;
 - рассмотрели планировщик процессов и что такое прерывания.
-

Загрузка ОС

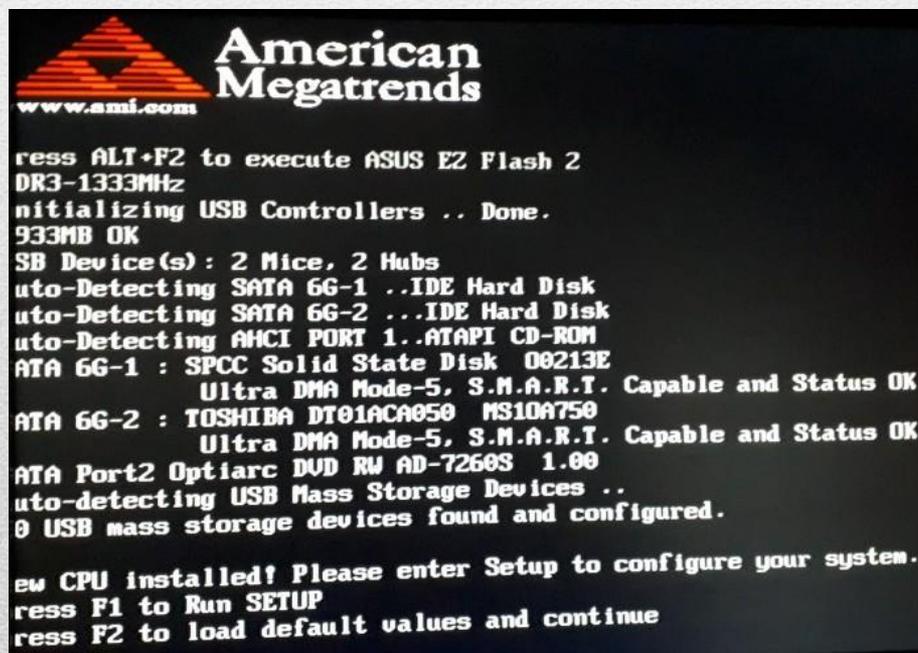
План темы:

- процесс загрузки ОС Linux;
 - основные функции BIOS/UEFI;
 - работу загрузчика GRUB;
 - начнем изучение процесса инициализации init.
-

Загрузка

Загрузка (boot, booting) — процесс запуска устройства и загрузки ОС.

В этот момент происходит обнаружение и (само)настройка всех компонентов системы.

A screenshot of a BIOS boot screen from American Megatrends. The screen is black with white text. At the top left is the American Megatrends logo, a stylized orange and red triangle, with the text 'American Megatrends' and 'www.amt.com' below it. The main text displays system initialization progress: 'Press ALT+F2 to execute ASUS EZ Flash 2', 'DDR3-1333MHz', 'initializing USB Controllers .. Done.', '933MB OK', 'SB Device(s) : 2 Mice, 2 Hubs', 'Auto-Detecting SATA 6G-1 ..IDE Hard Disk', 'Auto-Detecting SATA 6G-2 ...IDE Hard Disk', 'Auto-Detecting AHCI PORT 1..ATAPI CD-ROM', 'ATA 6G-1 : SPCC Solid State Disk 00213E', 'Ultra DMA Mode-5, S.M.A.R.T. Capable and Status OK', 'ATA 6G-2 : TOSHIBA DT01ACA050 MS10A750', 'Ultra DMA Mode-5, S.M.A.R.T. Capable and Status OK', 'ATA Port2 Optiarc DVD RW AD-7260S 1.00', 'Auto-detecting USB Mass Storage Devices ..', '0 USB mass storage devices found and configured.', 'New CPU installed! Please enter Setup to configure your system.', 'Press F1 to Run SETUP', and 'Press F2 to load default values and continue'.

Этапы загрузки

1. Загрузка BIOS/UEFI (MBR/GPT).
2. Поиск, загрузка в память и запуск загрузчика (GRUB).
3. Поиск, загрузка в память и запуск ядра ОС.
4. Запуск системных служб.
5. Запуск пользовательских служб.

1. После нажатия кнопки включения или перезагрузки управление берет на себя Базовая система ввода/вывода (Basic Input/Output System - BIOS).

2. Главный загрузчик определяет, откуда следует загружать ОС. В зависимости от типа загрузчика управление будет передано либо загрузочному коду, находящемуся в активном разделе жесткого диска, либо менеджеру загрузки, либо сам загрузчик поместит ядро ОС в оперативную память и передаст ему управление.

3. Получивший управление загрузчик операционной системы инициирует загрузку ядра ОС в память.

4. Запускается сама операционная система.

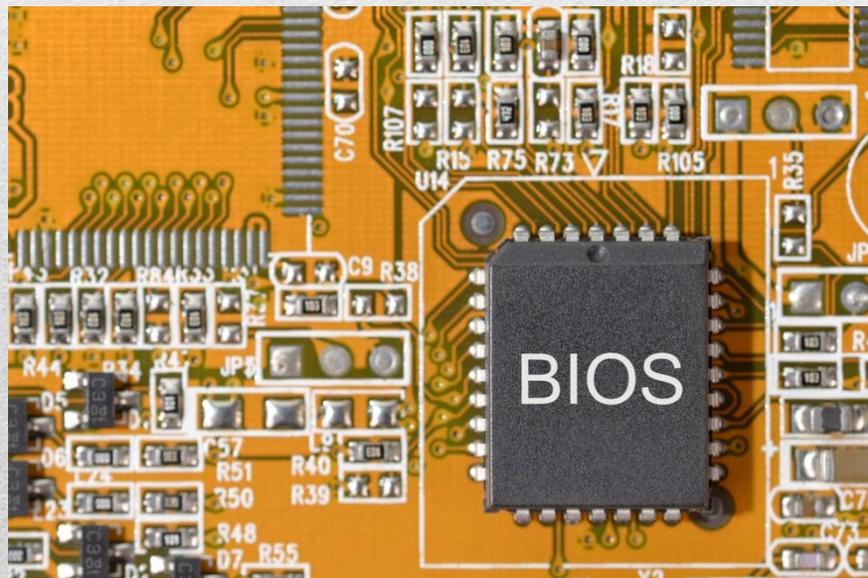
Как видно из вышесказанного, при загрузке компьютера используются абсолютные адреса, т.е. номера жестких дисков, цилиндров, головок, секторов, блоков. Выполнение этапов загрузки происходит по жестко заданной цепочке.

Для успешной загрузки ОС необходимо, чтобы все участники цепочки находились на своих строго определенных местах: главный загрузчик и основная таблица разделов в загрузочном секторе диска, загрузчик(и) ОС и таблицы разделов в остальных разделах диска.

Этапы загрузки

BIOS (basic input/output system — «базовая система ввода-вывода») - это программа, записанная в постоянной энергонезависимой памяти компьютера - ПЗУ (английская аббревиатура - CMOS).

BIOS производит тестирование и инициализацию всех устройств и, если они прошли успешно, считывает MBR по абсолютному адресу. Затем помещает считанный код главного загрузчика в оперативную память и передает ему управление.



BIOS

Изначально BIOS была собственностью IBM и применялась только IBM PC. После реверс-инжиниринга стала применяться во всех PC-совместимых компьютерах.

Ранее, помимо загрузки, BIOS использовалась для обработки прерываний от устройств ввода-вывода. Сейчас данный механизм полностью реализован на уровне операционной системы.

Также, в первых компьютерах BIOS была аппаратно реализована в виде ROM (ПЗУ), в последних — это flash-память с возможностью перезаписи.



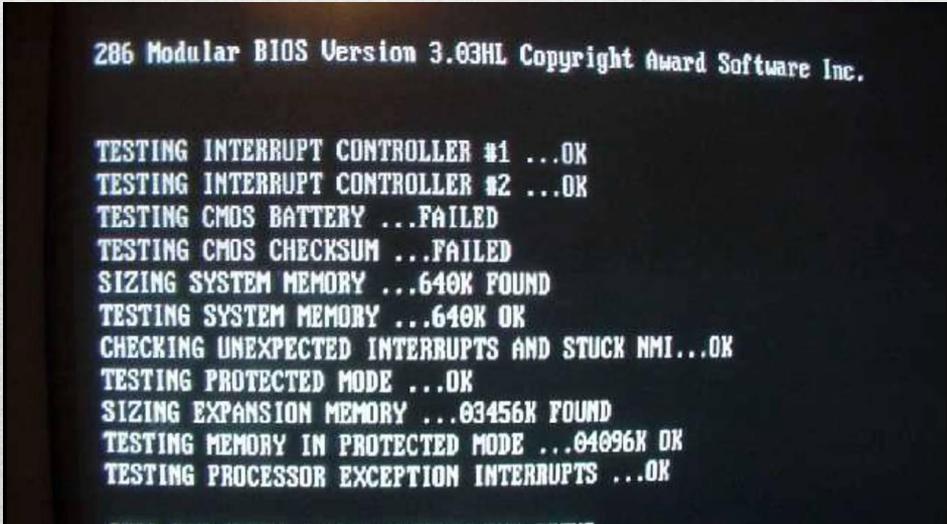
POST

POST (power-on self-test) — процесс первоначальной проверки оборудования сразу после его включения.

- Если во время POST обнаруживаются ошибки, то они выдаются пользователю как набор звуковых сигналов и номер кода ошибки, доступный через специальные отладочные платы.

- POST запускается после включения компьютера или нажатия кнопки « Reset».

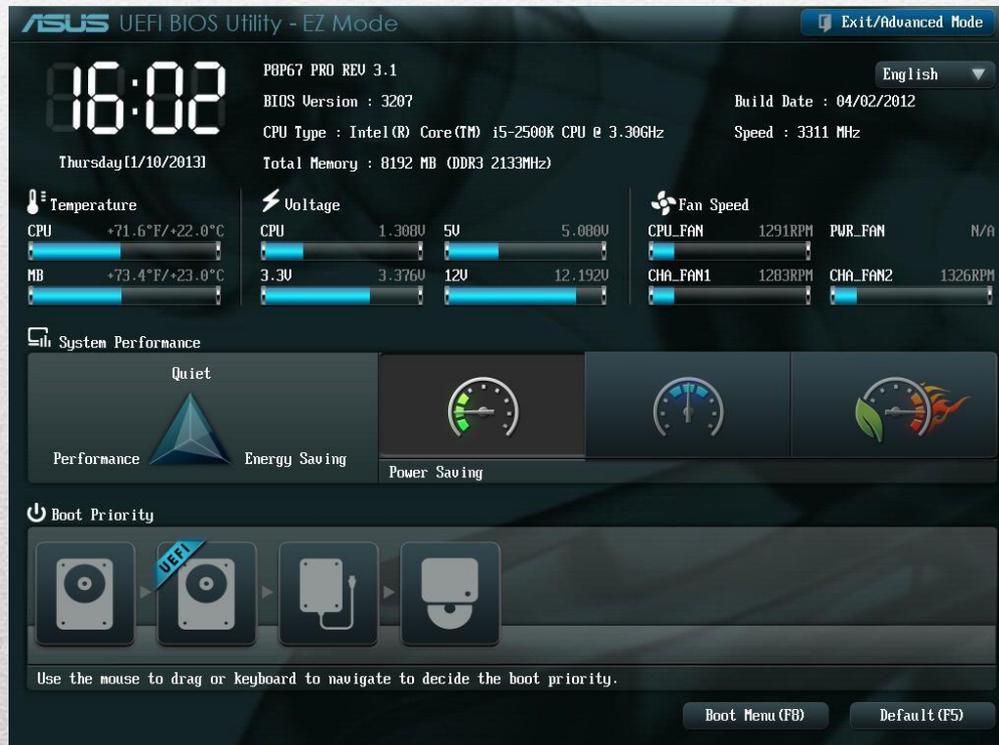
- Если перезагрузка происходит по нажатию *Alt-Ctrl-Del*, то процедура POST не запускается.



```
286 Modular BIOS Version 3.03HL Copyright Award Software Inc.  
TESTING INTERRUPT CONTROLLER #1 ...OK  
TESTING INTERRUPT CONTROLLER #2 ...OK  
TESTING CMOS BATTERY ...FAILED  
TESTING CMOS CHECKSUM ...FAILED  
SIZING SYSTEM MEMORY ...640K FOUND  
TESTING SYSTEM MEMORY ...640K OK  
CHECKING UNEXPECTED INTERRUPTS AND STUCK NMI...OK  
TESTING PROTECTED MODE ...OK  
SIZING EXPANSION MEMORY ...83456K FOUND  
TESTING MEMORY IN PROTECTED MODE ...04096K OK  
TESTING PROCESSOR EXCEPTION INTERRUPTS ...OK
```

UEFI

UEFI, Unified Extensible Firmware Interface (интерфейс расширяемой прошивки) — спецификация интерфейса между ОС и аппаратными прошивками (firmware).



UEFI

Преимущества UEFI

- возможность использовать разделы больше 2 ТБ GUID Partition Table (GPT);
- работа напрямую с ФС;
- работа с сетью;
- удобное и функциональное графическое окружение;
- 32- или 64-битное окружение;
- разработка на языке C;
- модульная архитектура;
- большие возможности по совместимости и модернизации.

Недостатки UEFI

- бóльшая возможность для внедрения вредоносных программ;
 - проблемы с гибкостью, т.к. каждая ОС должна иметь свой собственный драйвер в составе UEFI;
 - плохая поддержка старых ОС;
 - усложнение архитектуры усложняет разработку;
 - возможна ситуация, когда устанавливается только определенная ОС (например, Windows).
-

UEFI Boot Manager

UEFI, в отличие от BIOS, содержит свой собственный загрузчик — **UEFI Boot Manager**.

Boot Manager осуществляет загрузку:

- UEFI-загрузчиков ОС;
- UEFI-драйверов;
- UEFI-приложений.

Boot Manager имеет свою собственную конфигурацию, записанную в NVRAM (Non-volatile RAM).

Фазы UEFI

1. SEC (Security) — проверяет цифровые подписи и передает управление доверенному коду.
 2. PEI (Pre EFI Initialization) — инициализация устройств.
 3. DXE (Driver eXecution Environment) — загрузка сервисов UEFI.
 4. BDS (Boot Device Select) — поиск устройств загрузки.
 5. RT (Run Time) — GRUB.
-

Загрузчик

Загрузчик операционной системы, в зависимости от типа может выполнять такие функции как:

организация диалога с пользователем для выбора ОС;
выполнение подготовительных операций для запуска ядра ОС;

формирование параметров, передаваемых ядру ОС;
загрузка ядра (с диска или по сети) и передача ему управления.

Наиболее часто используются загрузчики ОС:

NTLDR -Windows NT;

Windows Boot Manager -Windows Vista, [Windows 7](#);

LILO (Linux LOader) - Linux;

GRUB (Grand Unified Bootloader);

OS/2 Boot Manager - OS/2;

BootX -Mac OS X;

Acronis OS selector - менеджер загрузки разных ОС.

Редактирование GRUB

Отредактируем /etc/default/grub

Открываем файл для редактирования редактором Nano

sudo nano /etc/default/grub

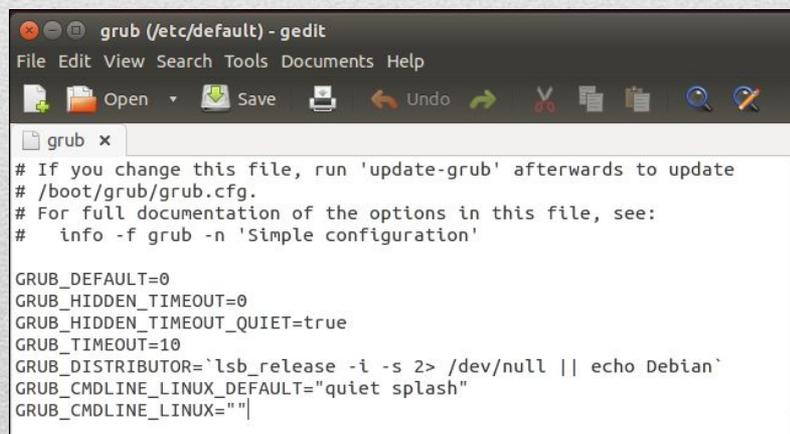
- закомментируем ожидание окна загрузки

Выполняем системную команду обновления данных загрузчика

sudo update-grub

Перезагружаем ОС «сразу»

shutdown -r now



```
grub (/etc/default) - gedit
File Edit View Search Tools Documents Help
Open Save Print Undo
grub x
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
```

Командная строка Grub

Выполнять команды grub возможно и до загрузки ОС (по аналогии в Windows – кнопка **F8**).

Для этого необходимо нажать «с» в меню GRUB (при использовании виртуальной машины может потребоваться при загрузке нажать «Shift и держать, пока не появится меню GRUB).

```
Minimal BASH-like line editing is supported. For the first word, TAB
lists possible command completions. Anywhere else TAB lists possible
device or file completions.

grub> ls
(proc) (hd0) (hd0,gpt2) (hd0,gpt1)
grub> ls hd0
error: invalid file name `hd0'.
grub> ls (hd0)
Device hd0: No known filesystem detected - Sector size 512B - Total size
15728640KiB
grub> ls (hd0,gpt1)
Partition hd0,gpt1: Filesystem type fat, UUID 6B33-696E - Partition
start at 1024KiB - Total size 248832KiB
grub> ls (hd0,gpt1)/
efi/
grub> ls (hd0,gpt1)/efi/
ubuntu/ boot/
grub> ls (hd0,gpt1)/efi/ubuntu/
grubx64.efi shimx64.efi mmx64.efi bootx64.csv grub.cfg
grub> ls (hd0,gpt2)
Partition hd0,gpt2: No known filesystem detected - Partition start at
249856KiB - Total size 15477760KiB
grub> _
```

Пример:

Параметры ядра при работе с Grub

- *Просмотр параметров ядра:*

```
sysctl -a
```

- *Просмотр конфигурационного файла:*

```
cat /etc/sysctl.conf | more
```

- *Просмотр сетевых настроек IPv4:*

```
cat /etc/sysctl.conf | grep ipv4
```

- *Включение пересылки пакетов (функций шлюза):*

```
sudo sysctl net.ipv4.ip_forward=1
```

Система инициализации

Init (инициализация)

init — специальный процесс (демон) управления системой и службами.

Расположение:

/sbin/init

Режимы работы init:

- однопользовательский (службы не запускаются);
- многопользовательский (режим запуска по умолчанию);
- сервер (аналогичен многопользовательскому, но без GUI).

Виды систем инициализации:

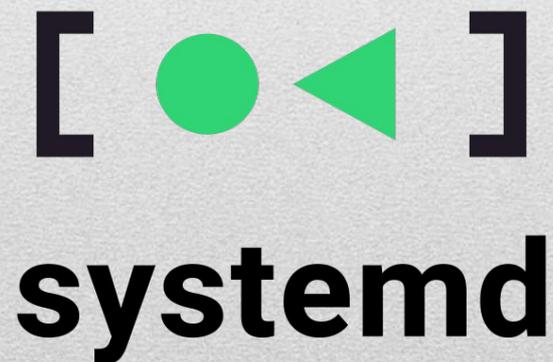
- **System V:**
 - Все службы запускаются последовательно.
 - **BSD init:**
 - FreeBSD, NetBSD, OpenBSD.
 - **systemd:**
 - упрощенный процесс загрузки;
 - параллельный запуск служб;
 - запись событий в системный журнал.
-

Рассмотрим systemd

systemd — подсистема инициализации и управления службами в Linux, фактически вытеснившая в 2010-е годы традиционную подсистему `init`. Основная особенность — интенсивное распараллеливание запуска служб в процессе загрузки системы, что позволяет существенно ускорить запуск операционной системы. Основная единица управления — модуль, одним из типов модулей являются «службы» — аналог демонов — наборы процессов, запускаемые и управляемые средствами подсистемы и изолируемые контрольными группами.

Позволяет выполнять следующие действия со службами:

- запускать/останавливать;
- добавлять/удалять;
- редактировать;
- собирать логи.



Рассмотрим systemd

systemd Utilities

systemctl journalctl notify analyze cglc cgtop loginctl nspawn

systemd Daemons

systemd
journald networkd
logind user session

systemd Targets

bootmode basic multi-user graphical user-session
dbus telephony display service
shutdown reboot dlog logind user-session tizen service

systemd Core

manager unit login namespace log
systemd service timer mount target multiseat inhibit
snapshot path socket swap session pam cgroup dbus

systemd Libraries

dbus-1 libpam libcap libcryptsetup tcpwrapper libaudit libnotify

Linux Kernel

cgroups autofs kdbus

Рассмотрим systemd

Введите команду

Top

```
cpu(s): 33,0 us, 3,0 sy, 0,0 ni, 33,7 id, 0,3 wa, 0,0 hi, 0,0 st, 0,0 st
MiB Mem : 2531,0 total, 798,4 free, 976,6 used, 755,9 buff/cache
MiB Swap: 976,0 total, 976,0 free, 0,0 used, 1366,2 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 1303 root        20   0 511508 31236 18408 S  40,4   1,2   0:02.74 packagekitd
   702 root        20   0 604532 104644 58064 S   3,6   4,0   0:03.08 Xorg
  1267 gss         20   0 1443456 339224 150648 S   3,6  13,1   0:15.04 plasmashell
  1208 gss         20   0 1048820 219144 127132 S   3,3   8,5   0:06.19 kwin_x11
  1203 gss         20   0 703436 77600 63860 R   3,0   3,0   0:01.33 kded5
   454 message+ 20   0  9640  5908  4004 S   2,6   0,2   0:01.37 dbus-daemon
  1288 gss         20   0 370796 50236 44032 S   1,0   1,9   0:00.50 DiscoverNotifie
  1044 gss         20   0 152640  2580  2204 S   0,7   0,1   0:00.21 VBoxClient
  1363 gss         20   0 821960 106540 84152 S   0,7   4,1   0:01.88 konsole
  1583 root        20   0  10100  3664  3164 R   0,7   0,1   0:00.06 top
   228 root        20   0  56664 29708 26824 S   0,3   1,1   0:00.69 systemd-journal
  1550 gss         20   0 158640 21472 19596 S   0,3   0,8   0:00.13 kio_http_cache_
    1 root        20   0  98576 10396  7704 S   0,0   0,4   0:03.40 systemd
    2 root        20   0  0 0 0 S  0,0  0,0  0:00.00 kthreadd
    3 root        0 -20  0 0 0 I  0,0  0,0  0:00.00 rcu_gp
    4 root        0 -20  0 0 0 I  0,0  0,0  0:00.00 rcu_par_gp
    5 root        20   0  0 0 0 I  0,0  0,0  0:00.10 kworker/0:0-events
    6 root        0 -20  0 0 0 I  0,0  0,0  0:00.00 kworker/0:0H-events_highpri
    7 root        20   0  0 0 0 I  0,0  0,0  0:00.01 kworker/u2:0-flush-254:0
```

```
State: running
Jobs: 0 queued
Failed: 0 units
Since: Thu 2022-10-13 12:51:19 +09; 3min 28s ago
CGroup: /
├─user.slice
│ └─user-1000.slice
│   └─user@1000.service ──
│     └─background.slice
│       └─plasma-kglobalaccel.service
│         └─1249 /usr/bin/kglobalaccel5
│           └─app.slice
│             └─gvfs-goa-volume-monitor.service
│               └─930 /usr/libexec/gvfs-goa-volume-monitor
│                 └─xdg-permission-store.service
│                   └─1156 /usr/libexec/xdg-permission-store
│                     └─xdg-document-portal.service
│                       └─1151 /usr/libexec/xdg-document-portal
│                         └─1162 fusermount -o rw,nosuid,nodev,fsname=portal,auto_unmount,subtype=portal -- /run/u
│                           └─app-konqueror-84aaee42f49340c2b7b35d538092c094.scope
│                             └─1349 /usr/bin/konqueror -session 10c5d36269000166363875700000012800017_1663640103_7431
│                               └─1478 /usr/lib/x86_64-linux-gnu/qt5/Libexec/QtWebEngineProcess --type=zygote --no-zygot
│                                 └─1479 /usr/lib/x86_64-linux-gnu/qt5/Libexec/QtWebEngineProcess --type=zygote --applicat
│                                   └─1489 /usr/lib/x86_64-linux-gnu/qt5/Libexec/QtWebEngineProcess --type=zygote --applicat
│                                     └─1514 /usr/lib/x86_64-linux-gnu/qt5/Libexec/QtWebEngineProcess --type=utility --enable-
│                                       └─xdg-desktop-portal.service
│                                         └─1144 /usr/libexec/xdg-desktop-portal
│                                           └─app-x2fusr\x2fbin\x2fkorgac-884899cbebd74fc91a4f6df23a532b5.scope
│                                             └─1328 /usr/bin/korgac -session 10c5d36269000166363775000000012800009_1663640103_741641
│                                               └─pulseaudio.service
│                                                 └─869 /usr/bin/pulseaudio --daemonize=no --log-target=journal
│                                                   └─909 /usr/libexec/pulse/gsettings-helper
```

Введите команду
systemctl status

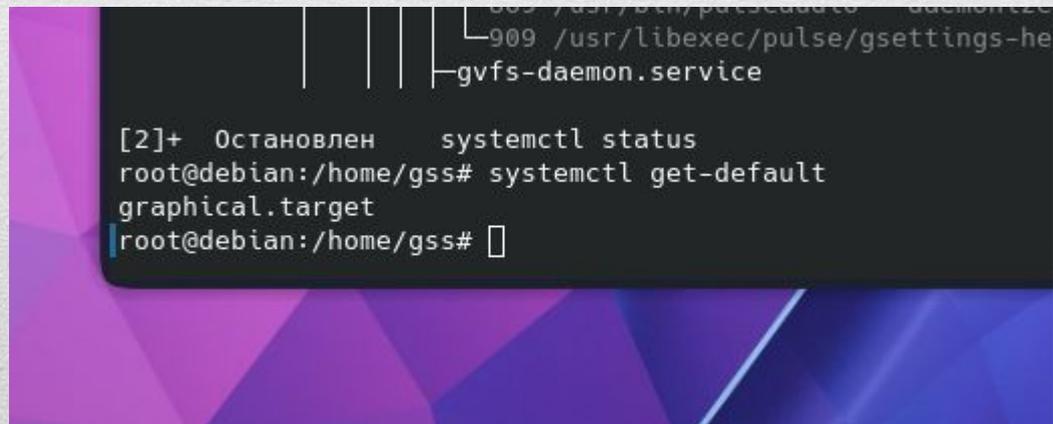
systemd - target

Цель (target) — нужное состояние системы; ссылка на файл, содержащий зависимости (службы).

➔ Systemd запускает все зависимости из соответствующего target-файла.

➔ Когда все зависимости будут запущены, то система будет работать на соответствующем target-уровне.

systemctl get-default
graphical.target



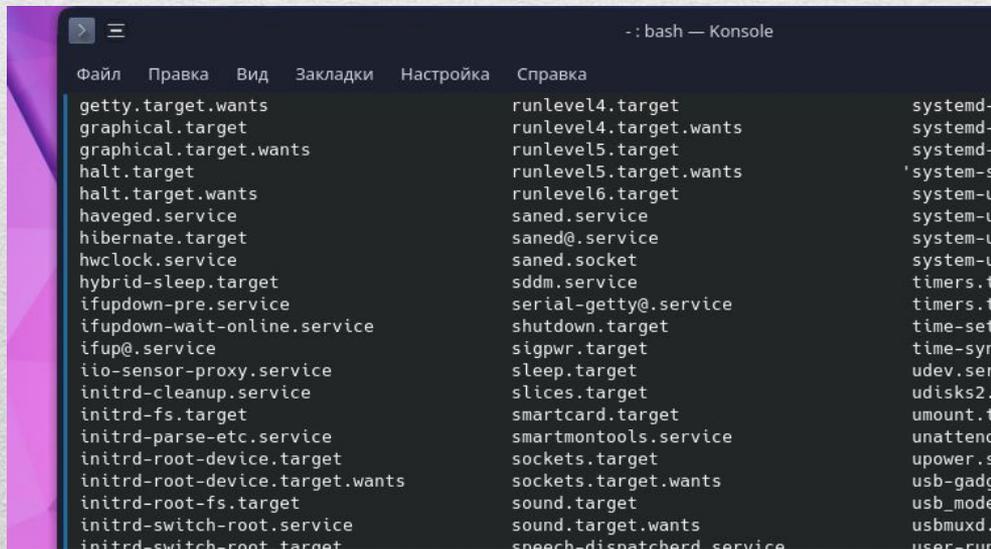
```
809 /usr/bin/pulseaudio - daemonize
909 /usr/libexec/pulse/gsettings-he
-gvfs-daemon.service

[2]+  Остановлен      systemctl status
root@debian:/home/gss# systemctl get-default
graphical.target
root@debian:/home/gss#
```

systemd - unit

Модуль (unit) — описывает запускаемую службу, устройство и т.п.
Каждый модуль описан в своем файле (unit file):

- /usr/lib/systemd/system/ — модули из пакетов (Nginx, Apache, MySQL);
- /run/systemd/system/ — модули, созданные во время работы ОС;
- /etc/systemd/system/ — модули, созданные пользователем.



The image shows a terminal window titled ': bash — Konsole'. The terminal displays a list of systemd unit files, including targets, services, sockets, and timers. The list is as follows:

```
getty.target.wants
graphical.target
graphical.target.wants
halt.target
halt.target.wants
haveged.service
hibernate.target
hwclock.service
hybrid-sleep.target
ifupdown-pre.service
ifupdown-wait-online.service
ifup@.service
iio-sensor-proxy.service
initrd-cleanup.service
initrd-fs.target
initrd-parse-etc.service
initrd-root-device.target
initrd-root-device.target.wants
initrd-root-fs.target
initrd-switch-root.service
initrd-switch-root.target
runlevel4.target
runlevel4.target.wants
runlevel5.target
runlevel5.target.wants
runlevel6.target
saned.service
saned@.service
saned.socket
sddm.service
serial-getty@.service
shutdown.target
sigpwr.target
sleep.target
slices.target
smartcard.target
smartmontools.service
sockets.target
sockets.target.wants
sound.target
sound.target.wants
speech-dispatcherd.service
systemd-
systemd-u
systemd-v
'system-sy
system-up
system-up
system-up
system-up
system-up
timers.ta
timers.ta
time-set.
time-sync
udev.serv
udisks2.s
umount.ta
unattende
upower.se
usb-gadge
usb_modes
usbmuxd.s
user-runt
```

systemd – unit. Типы модулей (юнитов)

- модули служб — обычные службы ОС;
- модули монтирования — монтируют ФС;
- целевой модули/цели — группируют другие модули;

systemd-analyze plot > test.svg
sudo systemctl list-dependencies

systemctl list-units — список модулей

systemctl list-units --type=service — список модулей-служб

systemctl status <модуль> — состояние выбранного модуля

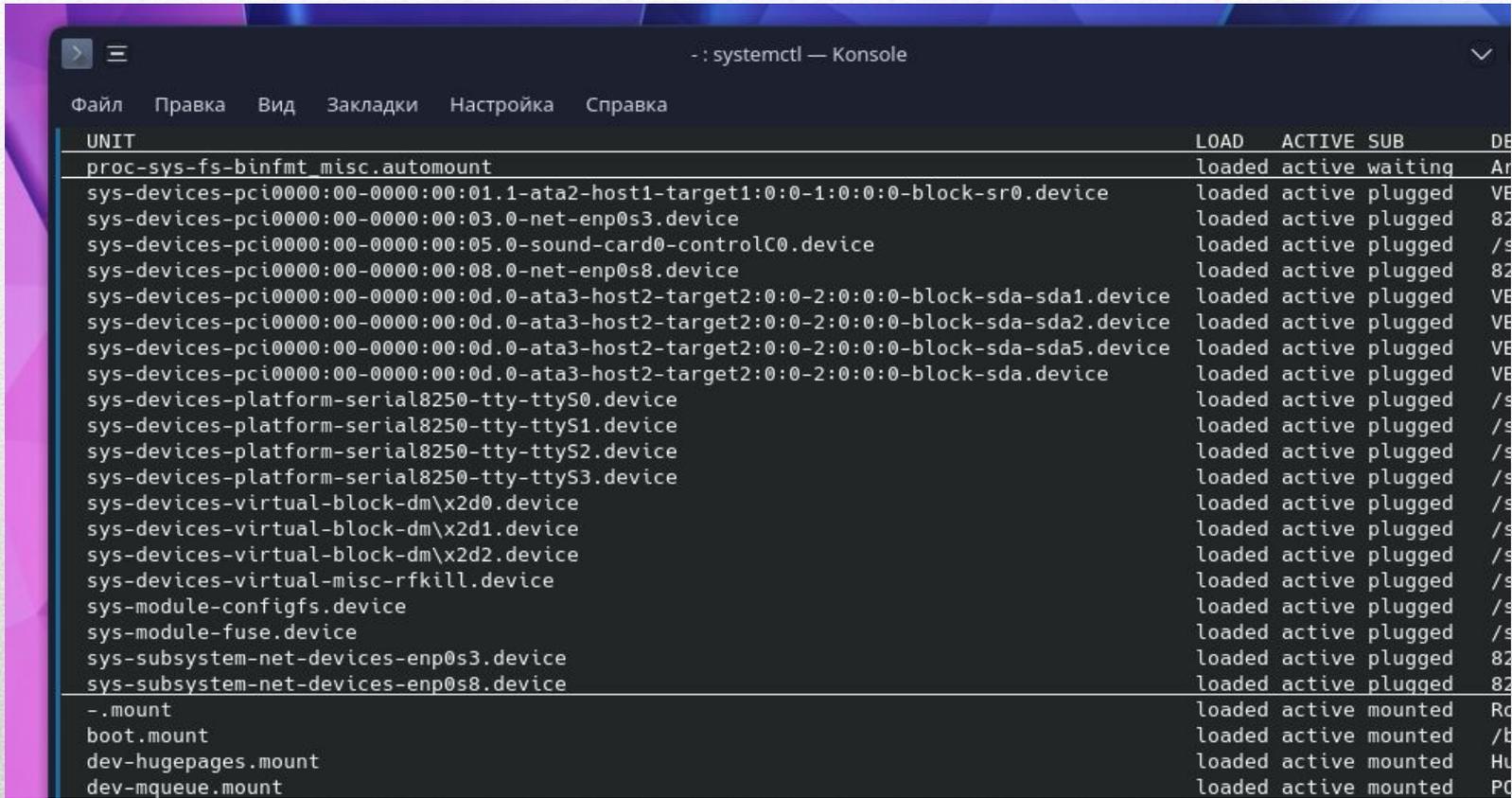
systemctl enable\disable <модуль> — разрешить/запретить модуль

systemctl start\stop\restart <модуль> — запустить/остановить/...

модуль

systemctl daemon-reload — перезапуск конфигурации systemd

systemd – unit. Типы модулей (юнитов)



```
- : systemctl — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
UNIT                                LOAD  ACTIVE SUB    DE
proc-sys-fs-binfmt_misc.automount  loaded active waiting Ar
sys-devices-pci0000:00-0000:00:01.1-ata2-host1-target1:0:0-1:0:0:0-block-sr0.device loaded active plugged VE
sys-devices-pci0000:00-0000:00:03.0-net-enp0s3.device loaded active plugged 82
sys-devices-pci0000:00-0000:00:05.0-sound-card0-controlC0.device loaded active plugged /s
sys-devices-pci0000:00-0000:00:08.0-net-enp0s8.device loaded active plugged 82
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-sda1.device loaded active plugged VE
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-sda2.device loaded active plugged VE
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-sda5.device loaded active plugged VE
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda.device loaded active plugged VE
sys-devices-platform-serial8250-tty-ttyS0.device loaded active plugged /s
sys-devices-platform-serial8250-tty-ttyS1.device loaded active plugged /s
sys-devices-platform-serial8250-tty-ttyS2.device loaded active plugged /s
sys-devices-platform-serial8250-tty-ttyS3.device loaded active plugged /s
sys-devices-virtual-block-dm\x2d0.device loaded active plugged /s
sys-devices-virtual-block-dm\x2d1.device loaded active plugged /s
sys-devices-virtual-block-dm\x2d2.device loaded active plugged /s
sys-devices-virtual-misc-rfkill.device loaded active plugged /s
sys-module-configfs.device loaded active plugged /s
sys-module-fuse.device loaded active plugged /s
sys-subsystem-net-devices-enp0s3.device loaded active plugged 82
sys-subsystem-net-devices-enp0s8.device loaded active plugged 82
-.mount loaded active mounted Rc
boot.mount loaded active mounted /b
dev-hugepages.mount loaded active mounted Hu
dev-mqueue.mount loaded active mounted PC
```

journalctl

Журнал — база данных, в которой хранятся сообщения ядра и служб, начиная с загрузки и заканчивая завершением работы.

journalctl

Настройки журнала:

nano /etc/systemd/journald.conf

journalctl -u=sshd — сообщения для модуля ssh

journalctl -b 0 -u ssh — ... ТОЛЬКО в текущем сеансе

journalctl -n 100 /usr/sbin/sshd — показать внешний лог

journalctl --since=yesterday --until=now — временной период

```
Файл  Правка  Вид  Закладки  Настройка  Справка
-- Journal begins at Thu 2022-10-13 12:58:00 +09, ends at Thu 2022-10-13 12:58:00 +09, --
ИМН 23 11:35:37 debian kernel: Linux version 5.10.0-15-amd64 (debian-kernel@lists.debian.org) (gcc-10 (Debian 1
ИМН 23 11:35:37 debian kernel: Command line: BOOT_IMAGE=/vmlinuz-5.10.0-15-amd64 root=/dev/mapper/debian--vg-ro
ИМН 23 11:35:37 debian kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
ИМН 23 11:35:37 debian kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
ИМН 23 11:35:37 debian kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
ИМН 23 11:35:37 debian kernel: x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]: 256
ИМН 23 11:35:37 debian kernel: x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard
ИМН 23 11:35:37 debian kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
ИМН 23 11:35:37 debian kernel: BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
ИМН 23 11:35:37 debian kernel: BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
ИМН 23 11:35:37 debian kernel: BIOS-e820: [mem 0x0000000001000000-0x000000000a3befff] usable
ИМН 23 11:35:37 debian kernel: BIOS-e820: [mem 0x000000000a3bf0000-0x000000000a3bffff] ACPI data
ИМН 23 11:35:37 debian kernel: BIOS-e820: [mem 0x000000000fc00000-0x000000000fc00fff] reserved
ИМН 23 11:35:37 debian kernel: BIOS-e820: [mem 0x000000000fee00000-0x000000000fee00fff] reserved
ИМН 23 11:35:37 debian kernel: BIOS-e820: [mem 0x000000000fffc0000-0x000000000fffcfffff] reserved
ИМН 23 11:35:37 debian kernel: NX (Execute Disable) protection: active
ИМН 23 11:35:37 debian kernel: SMBIOS 2.5 present.
ИМН 23 11:35:37 debian kernel: DMI: Innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
ИМН 23 11:35:37 debian kernel: Hypervisor detected: KVM
ИМН 23 11:35:37 debian kernel: kvm-clock: Using msrs 4b564d01 and 4b564d09
ИМН 23 11:35:37 debian kernel: kvm-clock: cpu 0, msr 24eb8001, primary cpu clock
ИМН 23 11:35:37 debian kernel: kvm-clock: using sched offset of 13770988775 cycles
ИМН 23 11:35:37 debian kernel: clocksource: kvm-clock: mask: 0xffffffffffffffff max_cycles: 0x1cd42e4dffb, max_
ИМН 23 11:35:37 debian kernel: tsc: Detected 3192.616 MHz processor
ИМН 23 11:35:37 debian kernel: e820: update [mem 0x00000000-0x000000fff] usable => reserved
ИМН 23 11:35:37 debian kernel: e820: remove [mem 0x000a0000-0x0000ffff] usable
ИМН 23 11:35:37 debian kernel: last_pfn = 0xa30f0 max_arch_pfn = 0x400000000
ИМН 23 11:35:37 debian kernel: MTRR default type: uncachable
ИМН 23 11:35:37 debian kernel: MTRR variable ranges disabled:
ИМН 23 11:35:37 debian kernel: Disabled
```

ИТОГИ

- рассмотрели как происходит загрузка современного ПК с ОС Linux;
 - получили представление о работе загрузчика, в том числе GRUB;
 - рассмотрели что такое инициализация системы и их основные виды;
 - начали работу с менеджером системы инициализации `systemd`, `journalctl`
-