
Предисловие

На этом занятии мы поговорим о:

- видах ядер ОС;
- архитектуре ядра;
- работе с модулями ядра;
- автозагрузке модулей.

По итогу занятия вы получите представление о внутреннем устройстве ядра Linux и научитесь работать с модулями ядра.

План занятия

1. [Предисловие](#)
2. [Ядра операционных систем](#)
3. [Модули ядра](#)
4. [Подсистемы ядра](#)
5. [Выполнение в режиме ядра](#)
6. [Автозагрузка и автосборка модулей](#)
7. [Итоги](#)
8. [Домашнее задание](#)



Ядра операционных систем

Ядра операционных систем

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.

Сравнение типов ядер

	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

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

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](#).

Работа 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

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

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>` — отслеживание системных вызовов приложения.

Аппаратные прерывания

1. Когда CPU 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.



Автозагрузка и автосборка модулей

DKMS

Dynamic Kernel Module Support (DKMS) — это фреймворк, который используется для генерации тех модулей ядра Linux, которые в общем случае не включены в дерево исходного кода.

DKMS позволяет драйверам устройств автоматически пересобираться, когда ядро уже установлено.

➡ Пользователь может не ждать, пока какая-то компания, проект или сопровождающий пакета выпустит новую версию модуля.

Значительное число модулей, не включенных в ядро, имеют DKMS вариант; некоторые из них размещаются в официальных репозиториях.

Работа с dkms

- `dkms add -m <MODULE-NAME>` — добавление модуля в dkms;
- `dkms status` — проверка статуса модуля в dkms;
- `dkms build -m <MODULE-NAME> -v <MODULE-VERSION>` — сборка модуля с помощью dkms;
- `dkms install -m <MODULE-NAME> -v <MODULE-VERSION>` — установка модуля с помощью dkms.
- `dkms autoinstall` — сборка и установка всех модулей.

Автозагрузка модулей

Для автоматической загрузки модулей в разных дистрибутивах предусмотрены разные механизмы.

Ubuntu: файл `/etc/modules`

CentOS: `/etc/modules-load.d/*.conf`

В этих файлах преимущественно перечисляются альтернативные имена модулей, их параметры, применяемые при их загрузке, а также черные списки, запрещенные для загрузки.



Итоги

Итоги

Сегодня мы рассмотрели ядро Linux:

- Ядро — такая же программа, только больше;
- Работа с модулями;
- Работа DKMS.