**Модули ядра линукс**

Модуль ядра – фрагменты кода, которые могут быть загружены и выгружены в ядро по требованию. Они расширяют функциональность ядра без необходимости перезагрузки системы Модуль может быть настроен как встроенный в ядро, так и загружаемый во время работы ОС.

Пример модуля – драйвера устройств.

Модули ядра линукс находятся в папке lib/modules в зависимости от версии ядра создаются соответствующие папки.

Основные команды для управления модулями:

* lsmod – посмотреть загруженные модули
* modinfo – информация о модуле
* insmod – загрузить модуль
* rmmod – удалить модуль
* dpkg -S \*.ko | grep /lib/modules – вывод всех модулей в системе
* find /lib/modules/$(uname -r) -name \*.ko – для текущего ядра
* sudo modprobe modname – загрузить модуля ядра
* sudo insmod /lib/modules/kernel\_v/…/\*.ko – Загрузка модуля с помощью пути
* sudo rmmod modulename – удаление модуля

**Debugfs**

В ядре линукс многие драйверы и подсистемы поддерживают специальную файловую систему, которая называется debugfs. Она очень похожа на другие виртуальные файловые системы (например procfs), но у неё нет никакого функционального назначения, кроме предоставляемой отладочной информации. Файлы тут генерируются ядром, и их содержимое зависит от драйвера подсистемы.

Корневой каталог debugfs доступен только для root

Для того, чтобы в системе присутствовала данная директория, при сборке ядра необходимо указать:

CONFIG\_DEBUG\_FS=y

API debugfs документирован в исходных файлах ядра fs/debugfs/inode.c fs/debugfs/file.c. Для добавления драйвера необходимо сделать 3 шага:

* Определить структуру каталога
* Создать ф-и работы с файлами
* Зарегистрировать файлы в debugfs

debugfs обычно монтируется с помощью команды mount -t debugfs none /sys/kernel/debug

Код, использующий debugfs должен включать в себя «linux/debugfs.h»

Для того, чтобы создать каталог хранения файлов в debugfs необходимо создать структуру. В случае успеха создастся каталог с именем name под указанным родительским каталогом. Если parent = null то каталог будет создан от корня

struct dentry \*debugfs\_create\_dir(const char \*name, struct dentry \*parent);

Для создания файла, используем структуру. Здесь name – имя файла, mode – описание прав доступа, parent – каталог в котором должен хранится файл, fops – набор операций которые реализуют поведение файла (хотя бы чтение/запись).

struct dentry \*debugfs\_create\_file(const char \*name, umode\_t mode, struct dentry \*parent, void \*data, const struct file\_operations \*fops);

В ряде случаев создание набора файловых операций на самом деле не требуется. Debugfs предоставляет набор вспомогательных ф-й для простых ситуаций. Файлы, содержащие одно целое значение могут быть созданы с помощью любого из них.

* Значение в десятичной сс void debugfs\_create\_u64(const char \*name, umode\_t mode, struct dentry \*parent, u64 \*value);
* Значение в шестнадцатеричной сс void debugfs\_create\_x32(const char \*name, umode\_t mode, struct dentry \*parent, u32 \*value);
* Неизвестный размер значения void debugfs\_create\_size\_t(const char \*name, umode\_t mode, struct dentry \*parent, size\_t \*value);
* Создание переменной struct dentry \*debugfs\_create\_ulong(const char \*name, umode\_t mode, struct dentry \*parent, unsigned long \*value);
* Логические значения void debugfs\_create\_bool(const char \*name, umode\_t mode, struct dentry \*parent, bool \*value);

**PPP channel**

**PCI dev**