День 2. Лекция 1. Процесс загрузки Linux.

Разработал: Максимов А.Н.

Содержание

Функции загрузчика

Загрузчик

Загрузчик отвечает за:

- Первоначальную инициализацию устройств
- Загрузку двоичный код приложения, обычно ядро операционной системы из флеш-памяти, по сети или из другой памяти
- Возможную распаковку приложения
- Запуск приложения

Кроме перечисленных основных функций, загрузчик, как правило, предоставляет оболочку с командами, реализующие различные операции

- Загрузка данных по сети или с устройств хранения информации, тестирование памяти, диагностика устройств, и т.д.)

Загрузчик на x86

- Процессор как правило имеет постоянную память, содержащую программу, BIOS
- BIOS инициализирует устройства и загружает, из первых 512 байт устройства хранения, Stage 1 загрузчик
- Stage 1 загрузчик в свою очередь загружает полноценный загрузчик, Stage 2 загрузчик, который как правило уже знает о форматах файловых систем и способен загрузить ядро ОС напрямую с устройства хранения информации

Шаги загрузки (1)

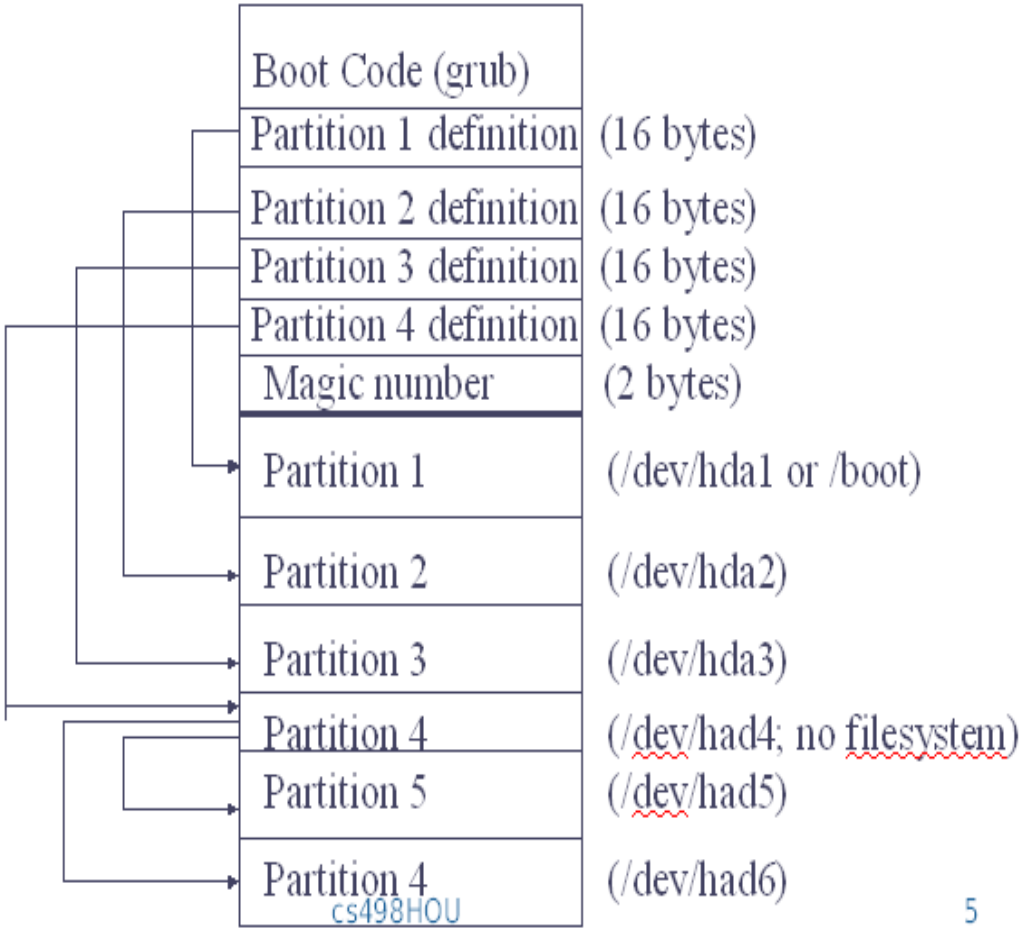
BIOS стартует и производит диагностику аппаратных устройств.

Находится аппаратное устройство с которого будет осуществляться загрузка

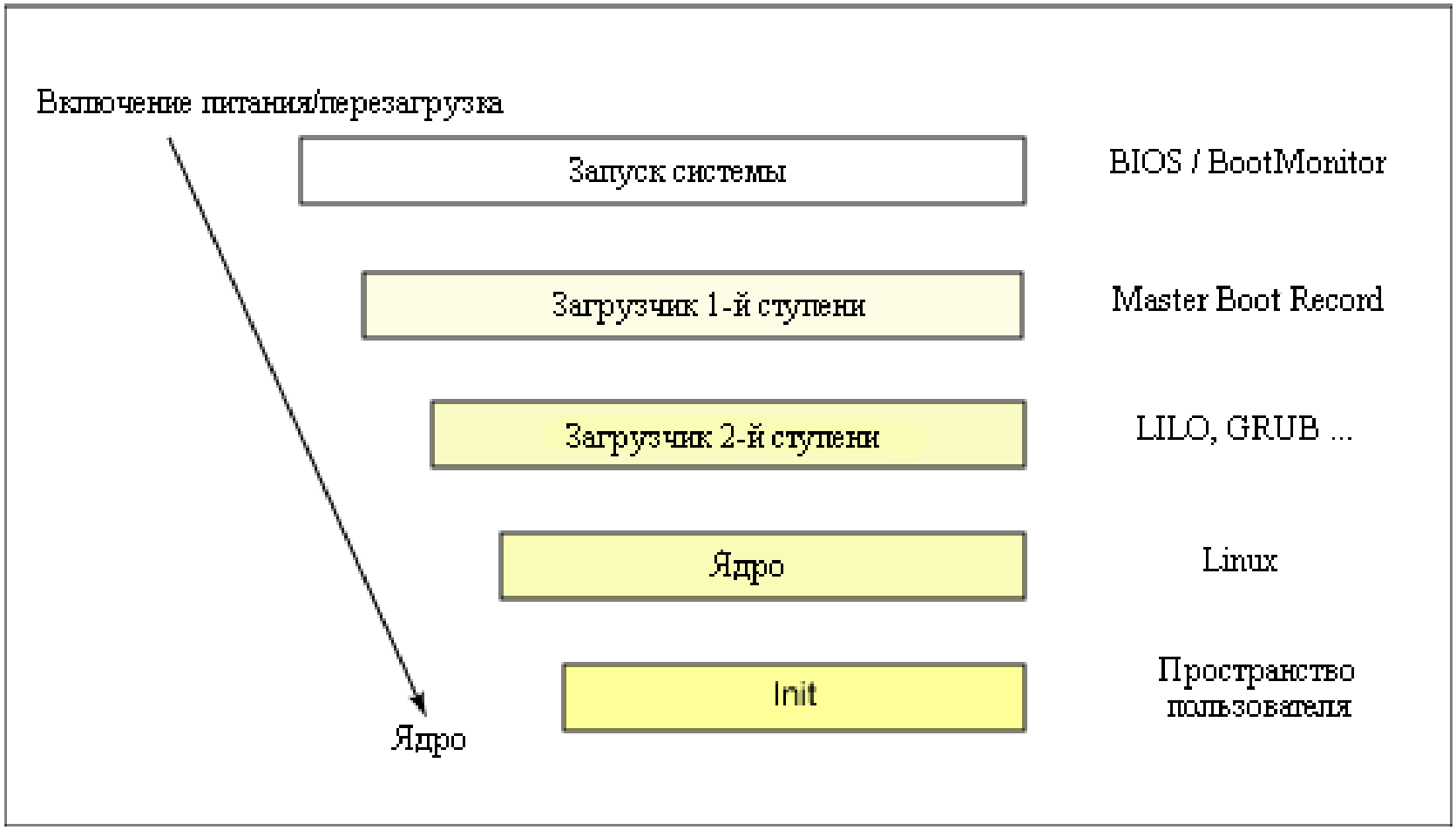
(diskette drives, CD-ROM drives, and hard drives)

Загружается начальная программа хранящаяся в Master Boot Record (MBR, находится в первом секторе устройства) и передает управление загрузчику.

Расположение партитионов загрузчика



Шаги загрузки (2)



Вторичный загрузчик (1)

Наиболее распространены два загрузчика: Linux Loader (lilo) and Grand Unified Bootloader (grub)

Первичный ищет вторичный загрузчик в партитионе, которая сконфигурирована как загрузочная

(/boot partition).

Запускает вторичный загрузчик.

Вторичный загрузчик (2)

Вторичный загузчик отображает различные варианту ядра ОС, которые могут быть загружены.

Находит образ ядра в /boot.

Образ ядра называется /boot/vmlinuz-<kernel-version>

Помещает начальный initial RAM disk image, называемый initrd, в память. initrd используется ядром для загрузки драйвероа, необходимых для загрузки системы.

Передает управление ядру.

Ядро

Инициализирует и конфигурирует память компьютера и конфигурирует аппаратное обеспечение входящие в систему (CPU , подсистему в.в., подсистему хранения).

Разархивирует и монтирует initrd для загрузки необходимых драйверов.

Монтирует корневую файловую систему в режиме read-only mode и освобождает неиспозуемую память.

Запускает процесс init путем запуска /sbin/init.

Процесс инициализации

init разбирает файл /etc/inittab для того чтобы определить какие программы запускать и на каком уровне.

0 используется для того, чтобы остановить систему. Система выполняет командуinit 0 и halted.

1 Переводит систему во однопользовательский режим.

2 Переводит систему в многопользовательский режим без поддержки сети.

3 Переводит систему в стандартный режим без запуска X.

4 Не используется.

5 (X-based) login.

Inittab

id:5:initdefault:

Говорит программе init какой run level использовать после перезагрузки.

si::sysinit:/etc/rc.d/rc.sysinit

Говорит программе init запустить скрипт rc.sysinit script.

Т.к. Второе поле пустое скрипт запускается во время загрузки на всех run levels.

rc.systinit

Устанавливает пути и имя hostname, и прверяет активна ли сеть.

Монтирует /proc file system

Устанавливает параметры ядра (kernel parameters)

Setting the system clock

Загружает keymaps и fonts

Запускает swapping

Инициализирует USB controller вместе с присоединенными устройствами.

Проверяет root file system.

Перемонтирует root file system как read-write.

Загружает модули.

/etc/rc.d/rc3.d

```
K01yum          K35vncserver  K74ypserv           S12syslog      S28autofs      S90xfs
K05saslauthd    K36lisa       K74ypxfrd           S13irqbalance  S40smartd      S95anacron
K10dc_server    K45named      K89netplugd         S13portmap     S44acpid       S95atd
K10psacct       K50netdump    K99readahead        S14nfslock     S55cups
S97messagebus
K12dc_client    K50snmpd      K99readahead_early  S18rpcgssd     S55sshd        S97rhnsd
K15httpd        K50snmptrapd  S00microcode_ctl    S19rpcidmapd   S56rawdevices  S99local
K20nfs          K50tux        S05kudzu            S19rpcsvcgssd  S56xinetd
S99mdmonitor
K24irda         K50vsftpd     S06cpuspeed         S20random      S80sendmail    S99mdmpd
K25squid        K70aep1000    S08iptables         S24pcmcia      S85gpm
K34yppasswdd    K70bcm5820    S09isdn             S25netfs       S87IIim
K35smb          K74ntpd       S10network          S26apmd        S90crond
```

Все файлы здесь являются символическими ссылками на скрипты, которые находятся в /etc/rc.d/init.d.

Система сначала запускает скрипты имена которых начинаются на K для того, чтобы убить связанные с ними процессы

→ /etc/rc.d/init.d/<command> stop

Система запускает скрипты начинающиеся с S для того чтобы запустить процессы

→ /etc/rc.d/init.d/<command> start

Изменение символа в имени с K на S (e.g., K20nfs → S20nfs) указывает Linux запустить процесс вместо того, чтобы убить.

inittab (продолжение)

ca::ctrlaltdel:/sbin/shutdown -t3 -r now

Указывает Ctrl+Alt+Delete для указания перезагрузки системы.

-t указывает, что init process ждет 3 секунды после предупреждения до посыки сигнала kill.

pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

3:2345:respawn:/sbin/mingetty tty3

Инициализировать ttys, обеспечивает login и запрашивает ввод пользователя, после этого начинает вход в систему для пользователя.

Upstart

- Первое появление в Ubuntu 6.10 (2006)
- Уход от последовательного выполнения скриптов
- Работа основана на обработке событий
- Возможность распаралелить обработку событий => ускорение загрузки ОС
- Оставлен /etc/init/rc.conf для обратной совмесимости с System V
- Есть службы (service) и задачи (task), службы могут автоматически перезапускаться
- $ /etc/init/sshd start — старый способ работы с сервисами
- $ service start sshd — актуальный способ работы с сервисами

Systemd

- Сокет-активные и шина-активные сервисы, которые иногда приводят к лучшему распараллеливанию взаимозависимых сервисов.

- cgroups используется для отслеживания сервисных процессов, вместо идентификаторов процессов (PID). Это означает, что демоны не будут потеряны даже после разветвления в другие процессы.

- В апреле 2012 исходные коды менеджера устройств udev были объединены с systemd.

- в Ubuntu перешел к Systemd

- актуальный способ работы с сервисами:

systemctl start sshd.service

systemd

Systemd - подсистема инициализации и управления службами в Linux. В современных дистрибутивах заменила классический init.d.

Основные особенности:

- 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.

Пример программы ☺

Предположим, что у нас есть программа foo_service.sh:

```
DATE=`date '+%Y-%m-%d %H:%M:%S'`
echo "Example service started at ${DATE}" | systemd-cat -p info

while :
do
echo “Work loop ...";
sleep 10;
done
```

Unit файл fooservice.service

```
[Unit]
Description=Example systemd service.

[Service]
Type=simple
ExecStart=/bin/bash /usr/bin/foo_service.sh

[Install]
WantedBy=multi-user.target
```

sudo cp fooservice.service /etc/systemd/system/fooservice.service
sudo chmod 644 /etc/systemd/system/fooservice.service

Управление из командной строки

sudo systemctl start myservice

sudo systemctl status myservice

sudo systemctl stop myservice

sudo systemctl restart myservice

sudo systemctl enable myservice

https://www.linode.com/docs/quick-answers/linux/start-service-at-boot/

Загрузчик на встраиваемых устройствах

- Процессор как правило имеет встроенный код для загрузки
- Этот код способен загрузить Stage 1 загрузчик в SRAM из MMC, NAND, SPI flash, UART (transmitting data over the serial line), etc.
- Stage 1 загрузчик инициализирует DRAM и устройства и загружает Stage 2 загрузчик в память

Один из основных универсальных загрузчиков для встраиваемых систем это U-Boot

U-Boot

- Лицензия GPLv2, как и Linux
- В свободном доступе на http://www.denx.de/wiki/U-Boot
- Последняя версия исходного кода доступна в Git репозитории: http://git.denx.de/?p=u-boot.git;a=summary
- С 2008 года релизы выходят каждые 3 месяца, версии именуются YYYY.MM

U-Boot

Обычно устанавливается на флеш-память для запуска устройством. В зависимости от устройства, возможны различные варианты установки:

- CPU предоставляет монитор загрузки, с которым возможно взаимодействовать через UART или USB порт
- CPU загружает со съемного накопителя (MMC) перед загрузкой с несъемного (NAND)
- U-Boot уже предустановлен и его возможно использовать для установки новой версии
- JTAG интерфейс, для записи в флеш-память без запущенной какой-либо системы

Пример:

```
U-Boot 2013.04 (May 29 2013 - 10:30:21)
OMAP36XX/37XX-GP ES1.2, CPU-OPP2, L3-165MHz, Max CPU Clock 1 Ghz
IGEPv2 + LPDDR/NAND
I2C:
ready
DRAM: 512 MiB
NAND: 512 MiB
MMC:
OMAP SD/MMC: 0
Die ID #255000029ff800000168580212029011
Net:
smc911x-0
U-Boot #
```

U-Boot имеет оболочку, предоставляющую достаточно богатый набор команд для управления загрузкой системы. Для подробной справки, введите ***help*** в командной сроке U-Boot

U-Boot в основном используется для загрузки и запуска ядра, но также он может использоваться для замены ядра и корневой ФС сохраненной во флеш-памяти.
Для передачи файлов между устройством и рабочей станции могут быть использованы:

- Сеть, U-Boot как правило имеет драйвер сетевого чипа
- USB, если U-Boot содержит драйвер USB контроллера вашей платформы
- SD или microSD карты, если U-Boot содержит драйвер SD контроллера вашей платформы
- Последовательный порт

Сборка для beagle bone

Установка кросс компилятора:
yum install gcc-arm-linux-gnu или
apt-get install gcc-arm-linux-gnueabi

Добыть ядро :)
git clone git://github.com/beagleboard/kernel.git
git checkout 3.8

Собрать ядро
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- beaglebone_defconfig
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- uImage dtbs
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- uImage-dtb.am335x-boneblack
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- modules

Ядро тут:
….. kernel/arch/arm/boot

Детально см. http://elinux.org/Building_BBB_Kernel

Загрузка по tftp

Команды для загрузки по tftp из uboot

dhcp

setenv serverip 192.168.1.76

tftp 0x80200000 uImage-BBB

setenv bootargs console=ttyO0,115200n8 quiet root=/dev/mmcblk0p2 ro rootfstype=ext4 rootwait

bootm 0x80200000