101.1 Определение и настройка аппаратной части

Студент должен разбираться в управлении основными комплектующими ПК.

Изучаем:

- получение информации об оборудовании;
- управление устройствами;
- понятия sysfs, udev, dbus, udevadm.

Термины и утилиты:

- /sys/
- /proc/
- /dev/
- modprobe
- lsmod
- lspci
- lsusb
- udevadm

Аппаратная часть современных компьютерных устройств — это отдельный полноценный курс. В данном уроке поверхностно рассматриваются базовые манипуляции с комплектующими ПК из ОС Linux.

При работе с устройствами Linux можно столкнуться со следующими понятиями:

- **HAL** – демон, представляющий слой абстрагирования от конкретной АЧ, благодаря которому ОС может обращаться к устройствам через их абстрактные представления (конкретную инструкцию устройству будет передавать драйвер), на данный момент является устаревшим;
- **Dbus** - шина, через которую процессы обмениваются информацией, в частности приложения получают через нее информацию об оборудовании из ядра;
- **udev** - менеджер устройств, пришедший на смену HAL.

Преимущества udev:

- работает на уровне пользователя (больше свободы действий);
- событийно-управляемый (а не опрашивает ядро по расписанию как HAL);
- удобные файлы конфигурации (а не xml как у HAL);
- содержит в системе только файлы активных устройств (а не всех устройств как HAL);
- содержит имена устройств неизменными (в HAL имя зависит от порядка подключения).

В директории **/dev** находится информация об интерфейсах работы с драйверами ядра, например:

/dev/sda *- первый жесткий диск;*

/dev/sr0 *- CD-ROM;*

Примечание: в каталоге /dev, как правило, отсутствует информация о сетевых картах и видеокартах, так как работа с ними немного сложнее чем простые операции чтения-записи.

В /dev лежат специальные файлы устройств, которые можно считать указателями на драйверы. При помощи этих файлов происходит обращение к устройствам.

База данных подключенных устройств хранится в директории **/sys,** где они расположены по каталогам и именуются автоматически согласно идентификаторам, например:

/sys/block *- перечень блочных устройств;*

/sys/bus *- перечень шин;*

Если /dev содержит файлы устройств для работы с приложениями, то /sys содержит информацию об этих устройствах. Метафора: /sys это упаковка, а /dev – содержимое.

Информация о состоянии ОС и всех запущенных процессах находится в директории **/proc**, например:

/proc/cpuinfo *- информация о процессоре;*

/proc/mounts *- подключенные файловые системы;*

Самая интересная директория - /proc/sys, так как она не только содержит информацию о системе, но и позволяет ее редактировать «на лету».

Для работы с модулями ядра, например, с драйверами устройств, предназначены следующие команды:

- **lsmod** – информация о модулях ядра;
- **lspci** – информация об устройствах PCI;
- **lspcmcia** – информация об устройствах PCMCIA;
- **lsusb** – информация о шине USB;
- **rmmod** – удалить модуль;
- **insmod** – установить модуль;
- **modprobe** – деликатно выполнить действия с модулем.

Отдельного внимания заслуживает инструменты работы с udev – udevadm, который может запускаться со следующими опциями:

- **info** – получение информации из БД;
- **trigger** – запросить события для устройства;
- **settle** – дождаться завершения обработки;
- **control** – управление демоном;
- **monitor** – следить за событиями;
- **test** – симулировать запуск события.

Примечание: мы не рассматриваем на данном этапе такие продвинутые действия, как создания собственных правил обработки событий оборудования, тем не менее вы сам можете их изучить.

101.2 Загрузка системы

Студент должен разбираться в процессе загрузки системы.

Изучаем:

- команды и опции загрузчика;
- настройку последовательности загрузки в BIOS;
- включение и выключение ПК;
- стили инициализации системы;
- журналы загрузки системы.

Термины и утилиты:

- dmesg
- BIOS
- bootloader
- kernel
- initramfs
- init
- SysVinit
- systemd

Процесс загрузки любой операционной системы, представляющий собой последовательную работу ряда программных и аппаратных компонентов ПК, будет рассмотрен в этом уроке.

Вкратце процесс загрузки выглядит так:

- при старте ПК процессор переходит на адрес BIOS (UEFI) и загружает его;
- BIOS (или современный UEFI) проводит необходимые проверки, выбирает согласно своим настройкам носитель информации;
- на носителе находит MBR (или GPT для UEFI) в которой находится загрузчик;
- дальше по обстоятельствам: загрузчик может загружать ОС, может передать управление следующему загрузчику по цепочке;
- в любом случае загрузчик знает где лежит ядро ОС, грузит его и Initial Ram Disk (там конфигурационные файлы и модули необходимые для загрузки ядра) в оперативную память;
- загруженное ядро берет дальнейший процесс запуска на себя (инициализация устройств, конфигурирование процессора, памяти и т.д.)
- после всех инициализационных процедур ядро запускает процедуры инициализации всех необходимых служб ОС.

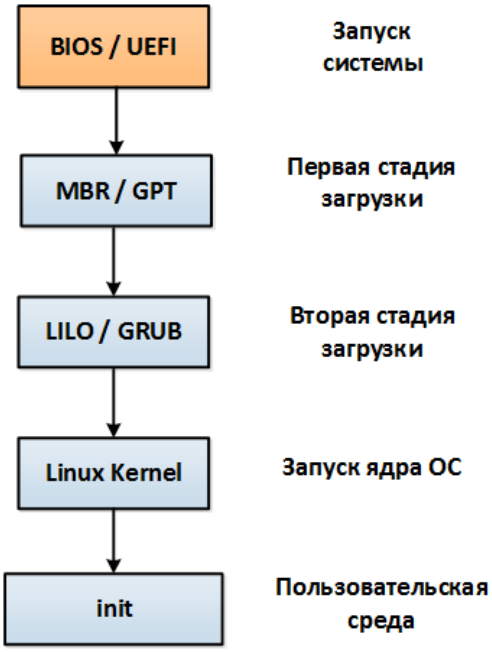

Рисунок 1. Порядок загрузки ОС

Существуют различные загрузчик ОС, например Lilo и Grub для Linux, NTLDR для Windows и т.д. Все они, как правило, имеют файл конфигурации самого загрузчика (*формат диалога загрузки, таймауты для выбора варианта загрузки, поле для ввода дополнительных опций и т.д.*) и секцию доступных для загрузки образов ОС.

Подробно стили инициализации и загрузчики рассматриваются в дальнейших уроках. Для успешного освоения данной темы необходимо знать, что в момент загрузки можно нажать выделенные клавиши (*как, правило, отличающиеся у разных загрузчиков*), и выбрать ОС для запуска или указать дополнительные опции загрузки.

При работе с загрузчиком можно увидеть следующие понятия:

- **uuid** – идентификатор носителя информации с установленной ОС;
- **kernel** – путь к ядру ОС;
- **initrd** – путь к минимальному образу ядра для загрузки в оперативную память;
- **quiet** – не показывать процесс загрузки;
- **splash** – показать заставку при загрузке.

Вся информация о загрузке хранится в журнале ***/var/log/dmesg*** *(журнал событий ядра)*, при этом данные о последней загрузке можно получить, выполнив команду **dmesg.** Также события загрузки могут располагаться в журнале ***/var/log/boot.log***. В современных ОС события загрузки можно увидеть при помощи утилиты **journalctl**, например командой ***journalctl -b0 SYSLOG_PID=1***

Процесс инициализации системы будет рассмотрен подробно в дальнейших уроках, на данном этапе нужно знать о существование трех стилей инициализации:

- **SysV** – родительский процесс инициализации системы на одном из заданных уровней запуска (runlevel);

- **systemd** – родительский процесс инициализацию системы в ускоренном режиме, за счёт параллельного запуска задач;
- **upstart** – родительский процесс инициализации системы на основе отслеживания событий.

101.3 Инициализация системы

Студент должен уметь управлять уровнями выполнения SysV (runlevels) или целями загрузки systemd (boot target)**.** Также в теме рассматриваются переключение в однопользовательский режим, выключение и перезагрузка системы, предупреждение пользователей, настройки уровня запуска ОС по умолчанию. Также нужно быть знакомыми с инициализацией в стиле upstart.

Изучаем:

- установка варианта загрузки по умолчанию;
- переключение между режимами работы;
- включение и выключение ПК;
- уведомление пользователей системы;
- корректное завершение процессов.

Термины и утилиты:

- /etc/inittab
- shutdown
- init
- /etc/init.d/
- telinit
- systemd
- systemctl
- /etc/systemd/
- /usr/lib/systemd/
- upstart
- inictl
- wall

Инициализация системы это процесс запуска скриптов, подготавливающих ОС к работе. Существует несколько различных стилей инициализации системы, использующиеся в разных семействах и даже в разных релизах ОС.

Классическим методом инициализации ОС является инициализация в стиле **SysV** (в современных ОС Linux практически не используется). Ключевым демоном является **init** (/sbin/init), являющийся родительским процессом, запускающим все остальные. Посмотреть дерево процессов и увидеть родительский можно командой pstree (для в centos нужно установить пакет psmisc)

Инициализация в стиле **SysV** оперирует с понятием уровня выполнения (***runlevel***), представляющего собой следующие режимы загрузки ОС:

0 *– выключение;*
1 *– однопользовательский режим;*
2 *– Debian/Ubuntu по умолчанию (GUI или CUI);*
3 *– RedHat/Suse по умолчанию (режим CUI);*
4 *– WildCard (программируемый режим);*

5 *– RedHat/Suse по умолчанию (режим GUI);*
6 *– перезагрузка.*

Настройки загрузки по умолчанию указываются в файле **/etc/inittab** (конфигурационный файл инициализации системы), например:

id:3:initdefault: *(уровень загрузки по умолчанию - третий);*

Все скрипты, использующиеся для запуска служб, располагаются в директории **/etc/init.d**, например:

/etc/init.d/network restart *(перезапустить службу сети);*

В каталоге **/etc** находятся директории **rc0.d, rc1.d** (и т.д.), содержащие в себе наборы скриптов (точнее ссылки на скрипты), использующиеся при переключении в разные режимы работы, например в ***rc3.d*** находятся скрипты выполняющиеся на ***runlevel3***.

Некоторые скрипты (имя начинается с “S”) запускают демоны, а некоторые (имя начинается с “K”) – останавливают.

Для работы с уровнями выполнения используют следующие команды:

- **init** или **telinit** - переключение в режима запуска;
- **runlevel** - узнать текущий режим работы;
- **halt** - выключить ОС;
- **reboot** - перезагрузить ПК;
- **shutdown** - завершить работу ПК.

Для управления демонами используется команда **service *имя_демона*** с ключами (не у всех демонов в конфиге могут присутствовать все перечисленные команды, зачастую можно увидеть ленивый скрипт только с командами start и stop):

- **start** - запустить;
- **status** - показать состояние;
- **stop** - остановить;
- **restart** - перезапустить;
- **reload** - перезагрузить конфигурационный файл службы.

___

Более современный стилем инициализации является **systemd**. Сейчас он используется на большинстве современных дистрибутивов Linux (Centos 7.0 и выше, Ubuntu 15.10 и выше), засчет ускорения загрузки (распараллеливание запуска демонов) и автоматической отказоустойчивости (отслеживание состояния демонов). Использует понятие модулей (units), которыми могу быть службы (.*service*), точки монтирования (.*mount*), устройства (.*device*) или сокеты (.*socket*).

Модули (юниты) создаваемые автоматически после установки пакетов ПО располагаются в директории **/usr/lib/systemd/.** Также можно располагать юниты в директориях **/etc/systemd/system/** (для ОС в целом) или **/etc/systemd/user/** (для пользователей).

Для управления юнитами используется утилита **systemctl**, например:

systemctl list-units *(показать запущенные юниты);*

systemctl start network.service *(запустить демон сети);*

systemctl status crond *(показать статус демона планировщика).*

Вместо ***runlevel*** в ***systemd*** используется понятие **target** (цели), только в отличии от уровней выполнения они не пронумерованы, некоторые из них могут быть запущены одновременно. Target обратно совместимы с инициализацией sysV, поэтому можно использовать команду **telinit** для переключения в другой режим выполнения.

Runlevel	Target	Описание
0	poweroff.target	выключение
1	rescue.target	однопользовательский режим
2,4	multi-user.target	настраиваемые режимы
3	multi-user.target	многопользовательский режим
5	graphical.target	графический режим
6	reboot.target	перезагрузка

Рисунок 1. Таргеты инициализации

Для управления режимами работы также используется утилита **systemctl**, например:

systemctl isolate reboot.target *(выполнить таргет reboot);*

systemctl set-default -f multi-user.target *(установить таргет multi-user в качестве режима загрузки по умолчанию);*

Для управления питаем также можно использовать **systemctl**, например:

systemctl reboot *(перезагрузить ПК);*

systemctl poweroff *(выключить ПК).*

Важная особенность systemd – гибкая система журналирования **journald**, собирающая информацию из различных источников и привязывающая ее к различным юнитам. Примеры ее использования:

journalctl –f *(просмотр сообщений в режиме реального времени);*

journalctl -n10 *(просмотр 10 последних сообщений);*

journalctl _UID=70 *(вывод всех сообщений включающих пользователя с ID=70);*

___

В исторической перспективе отмечаем систему инициализации системы является **upstart**, опирающуюся в своей работе на события, происходящие в ОС. Она использовалась в ubuntu с версии 6.10 по 15.04, и во многих других дистрибутивах, которые сейчас уже используют systemd.

Upstart оперирует понятиями **служба** (*service*), поддерживаемая в постоянном режиме работы, и **задача** (*task*), выполняющаяся разово. В процессе инициализации upstart считывает настройки из файлов конфигурации (*заданий - jobs*) в каталоге **/etc/init/.**

Каждое задание представляет собой сценарии запуска демонов с различными критериями и условиями выполнения.

Уровни инициализации или режимы работы используется такие же, как и в классическом sysV, так что команды **runlevel** и **telinit** продолжают работать. Синтаксис управления питанием и службами также схож с классическим.

Уровень инициализации по умолчанию указывается в файле ***/etc/init/rc-sysinit.conf***

Для управления инициализацией в стиле upstart используется утилита **initctl**, например:

initctl start networking *(запустить службу сети);*

initctl list *(вывести перечень служб);*

Для возможности извещения в любых дистрибутивах Linux всех пользователей, работающих в системе, о каких-либо действиях можно воспользоваться командой **wall *“текст_сообщения”***.

102.1 Разбиение жесткого диска

Студент должен уметь разбивать жесткий диск на разделы согласно требованиям Linux.

Изучаем:

- расположение файловых систем на разных разделах;
- создание разделов на диске;
- требования к разделу /boot;
- основные возможности LVM.

Термины и утилиты:

- / (корень файловой системы)
- /var
- /home
- /boot
- раздел подкачки
- точки монтирования
- разделы

При включении ПК инициализируется ПО материнской платы, которое после всех проверок передает управление первым секторам основного жесткого диска. В этих секторах находятся файлы загрузчика – специального ПО, позволяющего загрузить ядро ОС в оперативную память.

Традиционно в Linux существует единственный корень всей файловой системы (верхняя точка дерева каталогов, обозначаемая символом “/”), а все дополнительные разделы с различных жестких дисков (в том числе сетевые папки, флешки и т.д.) подключаются в ее пустые (можно монтировать и в каталоги с данными «поверх» них) каталоги (точки монтирования).

В случае наличия большого количества носителей (или исходя из требований системы) традиционно можно выделить отдельные разделы жесткого диска под следующие каталоги (в современном мире виртуалок отдельно монтируются директории с данными, а все остальное лежит на едином виртуальном диске):

- ***/*** *– корневая файловая система, самый большой раздел;*
- ***/boot*** *– загрузочный раздел;*
- ***/home*** *– домашние папки;*
- ***/root*** *– домашняя папка суперпользователя;*
- ***/etc*** *– конфигурация системы и ее компонентов;*
- ***/opt/*** *– папка для ПО от третьих поставщиков;*
- ***/var*** *– часто изменяемые данные;*
- ***/usr*** *– все установленные пакеты программ, документация, исходный код ядра;*
- ***/tmp*** *– временные файлы;*
- ***swap*** *– раздел подкачки, никуда не монтируется.*

Отдельно внимание уделяется каталогу /boot, часто монтируемому как отдельный раздел жесткого диска. На нем находятся следующие файлы:

- ***abi-..*** функции и библиотеки, через которые к ядру обращаются приложения;
- ***config-..*** файл параметров, при которых создано текущее ядро;
- ***initrd.img***-… образ стартовой корневой системы, загружающийся в ОЗУ;
- ***memtest…*** файлы ПО проверки ОЗУ;
- ***system.map..*** карта аппаратных адресов системы;
- ***vmlinuz..*** образ ядра системы.

На каждом жестком диске можно создать не более четырех разделов. Если вдруг потребуется больше разделов, то вместо основного раздела создается расширенный, который в свою очередь может содержать не более четырех логических разделов.

Жесткие диски именуются по порядку подключения: ***sda, sdb, sdc*** и т.д.

Разделы на каждом жестком диске нумеруются по порядку: ***sda1, sda2*** и т.д. При этом первые четыре цифры зарезервированы под основные и расширенные разделы, поэтому нумерация логических разделов начинается с пяти.

Рисунок 1. Разделы жесткого диска

Для создания, изменения, удаления и прочих действий с разделами используется утилита **fdisk**, которая в интерактивном режиме позволяет управлять разбиением жесткого диска *(будет подробно изучена в дальнейших уроках)*.

Для создания файловой системы на разделах используется утилита **mkfs** *(будет подробно изучена в дальнейших уроках)*.

Подключать созданные разделы можно как вручную при помощи команды **mount** ***имя_раздела пустой_каталог***, так и автоматически с использованием файла настроек **/etc/fstab** *(действия будут подробно изучены в дальнейших уроках).*

Раздел подкачки (**swap**) создается на жестком диске для временного хранения на нем данных, для которых не хватает места в оперативной памяти. Для форматирования раздела как раздела подкачки используется команда **mkswap** *(действия будут подробно изучены в дальнейших уроках)*.

В современных дистрибутивах Linux вместо классических разделов зачастую используются логические тома (раздел – часть диска, а том может располагаться на нескольких дисках одновременно), как более гибкий и надежный способ разбиения дискового пространства. Для этой цели используется LVM – менеджер логических томов.

При работе с LVM используются следующие понятия:

- физический носитель – жесткий диск (/dev/sda);
- физический том (PV) – носитель с некой системной инфой, которую понимает LVM;
- физическое пространство (PE – physical extent) – блоки дискового пространства;
- группа томов (VG) – набор PE;
- логический том – результирующее разбиение VG;

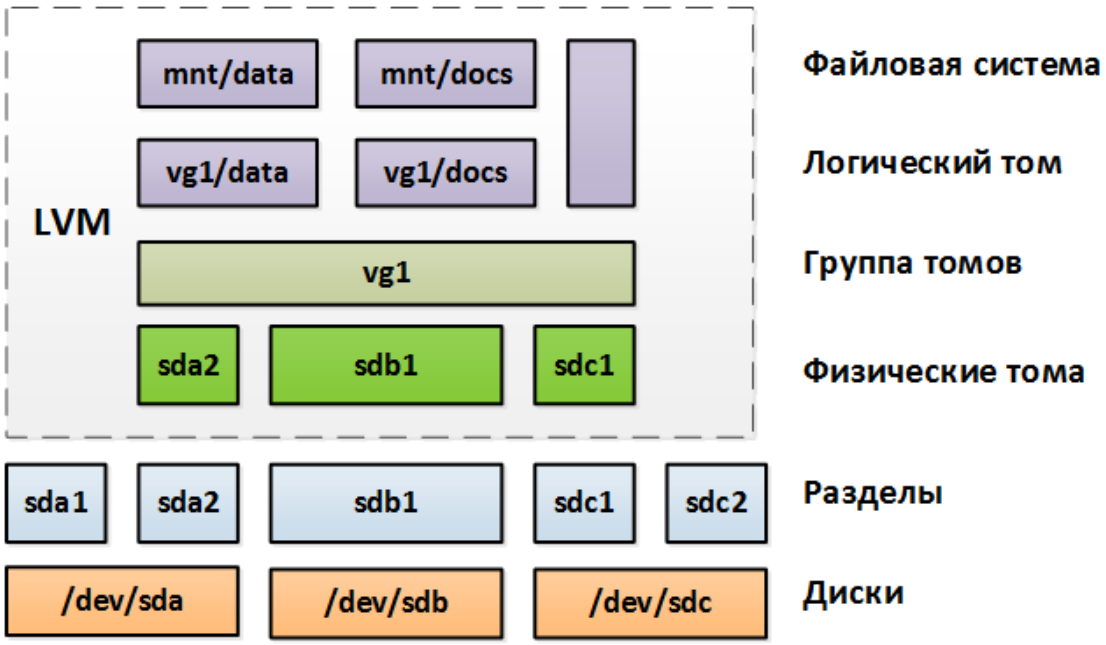

Рисунок 2. LVM

Для управления LVM доступен отдельный набор инструментов, например:

pvdisplay *(отобразить физические тома);*
pvcreate /dev/sdb2 *(создать физический том /dev/sdb2);*
vgcreate vg5 /dev/sdb1 /dev/sdb2 *(создать группу томов vg5 из двух физических томов);*
lvcreate -n lv2 -L 30G vg1 *(создать в группе vg1 логический том lv2 размером 30 Гб);*
lvresize -L 40G vg1/lv2 *(изменить размер тома lv2 в группе vg1 до 40 Гб).*

102.2 Установка загрузчика

Студент должен уметь выбирать, устанавливать и настраивать загрузчик ОС.

Изучаем:

- доступные загрузочные разделы;
- установку и настройку GRUB;
- настройку GRUB 2;
- взаимодействие с загрузчиком.

Термины и утилиты:

- menu.lst, grub.cfg и grub.conf
- grub-install
- grub-mkconfig
- MBR

При включении ПК инициализируется ПО материнской платы, которое после всех проверок передает управление первым секторам основного жесткого диска. В этих секторах находятся файлы загрузчика – специального ПО, позволяющего загрузить ядро ОС в оперативную память.

При первичной инициализации жесткого диска происходит выбор формата таблицы разделов: MBR (с ним работает BIOS) или более современный GUID (с ним работает UEFI).

Рисунок 1. MBR и GPT

Существует три самых популярных загрузчика Linux систем: LiLo, Grub и Grub2. Большинство современных дистрибутивов Linux используют Grub2.

Для настройки загрузчика Lilo используется команда **liloconfig**. Текущую конфигурацию можно увидеть в файле ***/etc/lilo.conf***. Конфигурационный файл прост для понимания, в нем указаны дисковые устройства, образы ядер, выводимый на экран при загрузке текст и т.д.

Настройки Grub хранятся в файле /***boot/grub/grub.conf*** или /***boot/grub/menu.lst.*** В нем, помимо основных опций загрузки, указаны все доступные для загрузки ОС и порядок обращения к ним.

Grub2 является отдельным проектом, полностью созданным с ноля. Для его установки можно воспользоваться командой **grub-install**, например:

grub-install /dev/sda *(установить загрузчик на устройство /dev/sda);*

Основным файлом конфигурации выбора ОС является файл ***/boot/grub/grub.cfg***, генерируемый автоматически. Общие настройки загрузчика можно увидеть в файле ***/etc/default/grub***. Скрипты, используемые загрузчиком находятся в директории ***/etc/grub.d***.

Изменения, внесенные в настройки загрузчика можно загрузить в файл конфигурации командой ***update-grub.*** Для автоматического создания файла конфигурации *(/boot/grub/grub.cfg)* с типовым меню можно воспользоваться утилитой **grub-mkconfig.**

102.3 Управление библиотеками

Студент должен уметь управлять определять необходимые для работы библиотеки и устанавливать их.

Изучаем:

- типы библиотек;
- месторасположение библиотек;
- загрузку библиотек.

Термины и утилиты:

- ldd
- ldconfig
- /etc/ld.so.conf
- LD_LIBRARY_PATH

Библиотеки — это набор функций, используемый ПО при работе. Библиотеки могут входить в состав программного обеспечения, а могут храниться отдельно, загружаясь в оперативную память по мере необходимости. Как правило процесс установки библиотеки - это обычный процесс установки пакета ПО. Но иногда бывает нужно прописать путь к библиотеке, посмотреть используемые файлы и т.д.

В Linux библиотеки располагаются тут:

- ***/lib*** - библиотеки для ПО, располагаемого в /bin;
- ***/usr/lib*** - библиотеки для ПО, располагаемого в /usr/bin;
- прочие пути, указанные в файле ***/etc/ld.so.conf*** *(по умолчанию это* ***/etc/ld.so.conf.d****)*;

Кроме того, набор возможных расположений библиотек определяется значением переменной ***LD_LIBRARY_PATH***.

Активные библиотеки располагаются в кэше (*файл* ***/etc/ld.so.cache***), для обновления которого необходимо выполнить команду **ldconfig** (заново загрузит библиотеки в кэш).

Для определения используемых двоичным файлов программы библиотек необходимо выполнение команды ldd, например:

ldd /bin/ls *(показать библиотеки, используемые командой /bin/ls);*

102.4 Управление пакетами ПО в Debian

Студент должен уметь управлять ПО в Debian системах.

Изучаем:

- установку, обновление и удаление пакетов ПО;
- получение информации о пакете ПО;
- состав и расположение файлов в пакетах ПО.

Термины и утилиты:

- /etc/apt/sources.list
- dpkg
- dpkg-reconfigure
- apt-get
- apt-cache
- aptitude

Семейство Debian обладает своими инструментами для работы с пакетами программного обеспечения (и свой формат этих пакетов: deb). Как и у прочих Linux систем, Debian предлагает мощный инструмент для работы с пакетами и несколько облегченных, предлагающих простые варианты решения наиболее частых задач.

Для работы с пакетами ПО в Debian системах предназначена утилита **dpkg**, являющая громоздким инструментом автоматизации работы с пакетами. Файл конфигурации менеджера пакетов располагается в файле ***/etc/dpkg/dpkg.cfg***. Примеры использования:

dpkg –l *(вывести информацию об установленных пакетах);*

dpkg –s firefox *(статус пакета firefox);*

dpkg –i webmin *(установить пакет webmin);*

dpkg –r webmin *(удалить пакет webmin);*

dpkg-reconfigure webmin *(переустановить пакет webmin);*

Большей популярностью, чем dpkg, при работе с пакетами ПО в Debian пользуется **apt**. Он умеет автоматически разрешать зависимости и работать с сетевыми репозиториями, в отличии от низкоуровневого dpkg. Примеры ее использования:

apt-get update *(обновить информацию о репозиториях);*

apt-get upgrade *(обновить пакеты);*

apt-get install webmin *(установить пакет webmin);*

apt-get remove webmin *(удалить пакет webmin);*

Для поиска пакетов используется команда apt-cache, например:

apt-cache search webmin *(найти пакет webmin);*

apt-cache show webmin *(показать информацию о пакете webmin);*

apt-cache depends webmin *(показать зависимости пакета webmin);*

Настройки менеджера пакетов apt находятся в файле ***/etc/apt/apt.conf***, а перечень репозиториев в файле ***/etc/apt/sources.list.***

Также для работы с пакетами в Debian системах можно использовать менеджер **aptitude**, имеющий диалоговой псевдографический интерфейс.

102.5 Управление пакетами ПО в RedHat

Студент должен уметь управлять ПО в RedHat системах.

Изучаем:

- установку, обновление и удаление пакетов ПО;
- получение информации о пакете ПО;
- состав и расположение файлов в пакетах ПО.

Термины и утилиты:

- rpm
- rpm2cpio
- /etc/yum.conf
- /etc/yum.repos.d/
- yum
- yumdownloader

Семейство RedHat обладает своими инструментами для работы с пакетами программного обеспечения (и свой формат этих пакетов: rpm). Как и у прочих Linux систем, RedHat предлагает мощный инструмент для работы с пакетами и несколько облегченных, предлагающих простые варианты решения наиболее частых задач.

Для работы с пакетами ПО в RedHat системах предназначена утилита **rpm**, являющая громоздким инструментом автоматизации работы с пакетами, например:

rpm –i webmin *(установить пакет webmin);*

rpm –e webmin *(удалить пакет webmin);*

rpm –V openssh-client *(проверить пакет openssh-client);*

rpm –qa *(показать установленные пакеты);*

Для конвертации пакета в набор исходных двоичных файлов можно воспользоваться командой **rpm2cpio**, например:

rpm2cpio ssh > bin.cpio *(достать исходники пакета ssh в архив bin.cpio);*

Большей популярностью, чем rpm, при работе с пакетами ПО в RedHat пользуется **yum**. Он умеет автоматически разрешать зависимости и работать с сетевыми репозиториями, в отличии от низкоуровневого rpm. Примеры его использования:

yum install vim *(установить vim);*

yum remove vim *(удалить vim);*

yum search vim *(найти vim);*

yum upgrade *(обновить пакеты ПО);*

Для поиска информации о пакетах yum пользуется репозиториями, конфигурация которых расположена в виде отдельных файлов в директории ***/etc/yum.repos.d/,*** например файл ***CentOS-Base.repo*** содержит несколько абзацев, указывающих на имя хранилища пакетов, его адрес и ключи для проверки.

Настройки самого менеджера пакетов yum хранятся в файле ***/etc/yum.conf***

Для скачивания пакетов можно воспользоваться утилитой **yumdownloader**, например:

yumdownloader resolve openssh *(скачать пакет openssh и все его зависимости).*

103.1 Работа в командной строке

Студент должен уметь работать с командными оболочками и командами в командной строке

Изучаем:

- работу с оболочками и командами для выполнения основных задач;
- использование и изменение среды оболочки, в том числе переменных среды;
- работу с историей выполненных команд;
- работу с командами исходя из текущего месторасположения.

Термины и утилиты:

- bash
- echo
- env
- export
- pwd
- set
- unset
- man
- uname
- history
- .bash_history

Работа администратора с Linux заключается прежде всего в оперировании в командной строке. Командная строка, или консоль — это отдельная программируемая среда со своими настройками, возможностями и инструментами. Существует множество различных оболочек, в данном уроке рассматривается bash, используемая в большинстве современных ОС.

Увидеть название оболочки, используемой пользователем по умолчанию, можно в конце каждой строки файла **/etc/passwd**. Глобальные настройки командной строки находятся в файле ***/etc/profile***, а настройки для каждого пользователя в файлах его домашней директории.

При этом существует несколько вариантов расположения настроек, в зависимости от семейства ОС. В общем виде поиск настроек осуществляется в следующем порядке: ***~/.bash_profile, ~/.bash_login***, и ***~/.profile*** (настройки берутся из первого обнаруженного файла).

При работе в консоли используются следующие распространенные команды (их больше, но в данной теме LPI почему-то заостряет внимание на этих):

- **cat** – вывод содержимого файла в консоль;
- **cd** – переход в каталог;
- **ls** – вывод содержимого каталога;

- **echo** – вывод текста в консоль;
- **touch** – обновление времени редактирования файла или создание нового пустого файла;
- **uname** – вывод имени ОС;

Консоль, как отдельная рабочая среда содержит свои переменные: переменные среды (глобальные переменные, использующиеся в ОС) и обычные переменные (работают в пределах запущенного сеанса консоли). Для просмотра переменных доступны команды:

- **env** – вывод переменных среды;
- **export** – превращение переменной в переменную среды;
- **unset** – отключение переменной;

Для создания переменной используется простой синтаксис ***имя_переменной=значение_переменной***. Для обращения к переменной указывается знак **$**, например:

X=12 *(задать переменной X значение 12);*

echo $X *(вывести значение переменной X);*

Для выполнения команды в текущем каталоге (его можно узнать командой **pwd**), нужно указывать полный путь к команде (*например, /home/semaev/script*), если текущий каталог не перечислен в значениях переменной среды **PATH**.

Для получения справочной информации доступны команды:

- **man** – справка по команде;
- **file** – справка по файлу;
- **whatis** – справка по названию;
- **history** – вывод истории команд (список хранится в .bash_history);

Отдельно следует упомянуть команду **exec**, позволяющую выполнять команду за пределами текущей оболочки, сбрасывая права суперпользователя.

103.2 Управление текстовым потоком

Студент должен уметь управлять фильтровать текстовый поток

Изучаем:

- применение текстовых фильтров с использованием встроенных команд UNIX

Термины и утилиты:

- cat
- cut
- expand
- fmt
- head
- join
- less
- nl
- od
- paste
- pr
- sed
- sort
- split
- tail
- tr
- unexpand
- uniq
- wc

Множество файлов в Linux неудобны для восприятия без предварительного форматирования, кроме того процесс обработки текстовой информации часто требует ее форматирования, обрезания, сортировки и т.д. Linux предлагает широкий набор инструментов редактирования текстового вывода, при этом следует заметить что указанные инструменты не затрагивают исходный файл (или текстовый поток), а форматируют его, выдавая полученный результат в консоль или другой файл.

Простой вывод текста (без корректировки содержимого):

Для вывода содержимого файла или объединения нескольких файлов на выводе используется команда **cat**, например:

cat 1.txt 2.txt > 3.txt *(передать содержимое двух файлов в третий);*

Команда **join** объединяет строки нескольких файлов по общему полю и отправляет их на стандартный вывод.

Команда **less** загружает файл постепенно и также отправляет его на вывод (удобно для больших файлов).

Команда **nl** нумерует строки.

Команда **pr** показывает, как содержимое файла будет выглядеть при выводе его на печать.

Команда **paste** вставляет строки из одного файла в другой и отправляет результат на стандартный вывод.

Команда **head** показывает первые строки файла, а **tail** – последние.

Например:

head –n 2 /var/log/syslog *(вывести первые две строчки файла);*

tail –f /var/log/syslog *(показывать последние строки файла, отображая содержимое в реальный момент времени);*

Команда **sort** может отсортировать содержимое файла по какому-нибудь признаку, например:

sort 1.txt *(отсортировать строки по алфавиту);*

Команда **wc** позволяет посчитать количество символов и т.д. в текстовом потоке, например:

wc 1.txt *(посчитать количество строк, символов, слов);*

Вывод форматированного текста (с корректировкой содержимого):

Команда **cut** используется для обрезания содержимого файлов на выводе, например:

cut –c 2,3,4,5,10 1.txt *(показывать только символы на указанных позициях: 2,3,4,5,10);*

Команда **expand** превращает символы табуляции в несколько пробелов, а **unexpand** наоборот – пробелы в символы табуляции.

Команда **fmt** предназначена для форматирования текста различными способами, например:

fmt 1.txt *(вывести содержимое файла в одну строку);*

fmt –w 10 1.txt *(вывести содержимое файла в строки не больше 10 символов каждая);*

Команда **od** предназначена для конвертации текста в другой формат, например:

od –c 1.txt *(отобразить в кодировке ASCI);*

Команда **tr** предназначена для замены и удаления символов в текстовом потоке, например:

echo “HELLO” | tr –t L l *(заменить “L” на “l” в полученном на вводе тексте);*

Команда **split** может разбить файла по каким-либо критериям, например:

split –l 2 1.txt *(разбить содержимое файла по две строчки);*

Команда **uniq** предназначена для поиска повторяющихся строк, например:

uniq –u 1.txt *(показать только уникальные строки);*

Sed – отдельный мощный инструмент, понимающий групповые символы, шаблоны и т.д. Будет рассмотрен немного подробнее в дальнейших урока. Работает, например, так:

sed –e ‘s/socks/people/’ 1.txt *(заменить в файле слово “socks” на “people”);*

103.3 Основные операции с файлами

Студент должен уметь управлять файлами и каталогами при помощи основных команд Linux

Изучаем:

- копирование, перенос и удаление файлов и каталогов;
- копирование нескольких файлов и каталогов, в том числе рекурсивно;
- удаление файлов и каталогов, в том числе рекурсивно;
- групповые символы (wildcard);
- отбор файлов по типу, размеру, датам и т.д.;
- работу с tar, cpio и dd.

Термины и утилиты:

- cp
- find
- mkdir
- mv
- ls
- rm
- rmdir
- touch
- tar
- cpio
- dd
- file
- gzip
- gunzip
- bzip2
- xz
- групповые символы

Частой задачей администратора в Linux является работа с файлами и каталогами: создание, копирование, перемещение и удаление. Использование групповых символов вместе с инструментами консоли позволяют осуществлять эти действия с максимальной эффективностью.

Для создания директорий используется команда **mkdir *имя_каталога***, для удаления пустого каталога - **rmdir *имя_каталога***. Просмотр содержимого текущей директории – **ls**. Например:

ls –l *(просмотреть содержимое текущего каталога);*

ls fol* *(посмотреть содержимого каталогов, имена которых начинаются с “fol”);*

Для копирования используется команда **cp *что_копировать куда_копировать***, например:

cp –R Folder NewFolder *(скопировать директорию “Folder” со всем содержимым в текущий каталог с именем новой директории “NewFolder”);*

Для переноса используется команда **mv *что_переносить куда_переносить***, например:

mv file.txt folder1/ *(перенести “file.txt” в каталог “folder1” в текущей директории);*

Для удаления используется команда **rm *что_удалить***, например:

rm –rf folder1 *(удалить каталог folder1 вместе с содержимым);*

Для создания пустого файла, или обновления метки доступа существующего, используется команда **touch *имя_файла,*** а для получения информации о файле – команда **file *имя_файла***.

Групповые символы, использующиеся при работе с текстовыми данными:

- ***** *– любое количество любых символов;*
- ***?*** *– любой символ;*
- ***!*** *– не (любой символ кроме указанных);*
- ***[ac]*** *– a или c;*
- ***[a-c]*** *– a, b, c.*

Например:

***ls ** *(вывести содержимое всех каталогов);*

ls ?at.txt *(вывести содержимое файла с именем: любая первая буква, остальная часть имени “at.txt”);*

ls *at.txt *(вывести содержимое файла с именем, оканчивающимся на “at.txt”);*

ls ![ab]* *(вывести содержимое файла или каталога с именем, начинающимся с любых символов, кроме “a” и “b”);*

Для поиска информации используется команда **find**, например:

find . –name “*A*” *(искать в текущей директории объекты, имеющие в имени символ “A”);*

find /etc –size -5M *(искать в директории “/etc” объекты размером менее 5 Мб);*

find . –type l *(искать в текущей директории ссылки);*

В среде Linux существует несколько распространённых архиваторов, использующихся для распаковки и упаковки объектов в единый файл:

- **cpio** - двоичный архиватор, копирующий информацию в архив и обратно;
- **dd** - копирование по блокам;
- **gzip** и **gunzip** - утилита сжатия и восстановления файлов (контейнер одного файла);

- bzip2 и bunzip2 - утилита сжатия и восстановления файлов (контейнер одного файла);
- xz и unxz - утилита сжатия и восстановления файлов (контейнер одного файла);
- tar - архиватор (работает с набором файлов и каталогов).

Например:

find . –name "*.txt" | cpio –o > ../test.cpio *(найти в текущей папке все файлы с расширением ".txt" и упаковать их в архив test.cpio, расположив его на уровень выше текущей директории);*

cpio -id < test.cpio *(распаковать в текущий каталог архив "test.cpio", создавая при необходимости структуру директорий);*

dd if=/dev/sdb of=drive.img *(скопировать носитель /dev/sdb в образ drive.img, располагающийся в текущей директории);*

gzip drive.img *(сжать образ "drive.img" до "drive.img.gz", удалив исходный файл);*

bunzip2 drive.img.bz2 *(распаковать архив "drive.img.bz2" в текущий каталог);*

unxz drive.img.xz *(распаковать архив "drive.img.xz" в текущий каталог);*

tar cvf arch.tar folder *(упаковать каталог "folder" в архив "arch.tar").*

tar –xvf archive.tar *(распаковать архив "archive.tar").*

103.4 Потоки, конвейеры и перенаправления

Студент должен уметь управлять потоками для обработки текстовых данных

Изучаем:

- перенаправление стандартных ввода, вывода и ошибок;
- передачу вывода одной команды на ввод другой;
- использование вывода одной команды в качестве аргументов другой;
- получение выходных данных в файл и на стандартный вывод.

Термины и утилиты:

- tee
- xargs
- stdin
- stout
- stderr

Linux традиционно использует потоки для ввода, вывода и ошибок. Обычно ввод – это клавиатура или файлы, а вывод ошибок и результатов происходит в консоль. Но часто бывает нужно объединить последовательность команд в конвейер, или отправить результат в какой-нибудь файл.

Для того чтобы послать на ввод программы некоторые данные используется знак ”**<**”, например:

wc < file1.txt *(использовать в качестве ввода для программы wc файл file.txt);*

Для того чтобы послать вывод программы куда-нибудь, кроме стандартного направления, используется знак ”**>**”, например:

ls > list.txt *(перезаписать содержимое файла list.txt результатом команды ls);*

ls >> list.txt *(дописать в файл list.txt результат команды ls);*

Для ввода, вывода и ошибок используются следующие обозначения:

- ***Stdin*** *– стандартный ввод программы (обозначение “0”);*
- ***Stdout*** *– стандартный вывод программы (обозначение “1”);*
- ***Stderr*** *– стандартные ошибки программы (обозначение “2”);*

Например:

l*s bob 2> error.txt*

ls bob > result.txt 2> error.txt

103.5 Управление процессами в Linux

Студент должен уметь создавать, отслеживать и завершать процессы

Изучаем:

- запуск задач в активном и фоновом режимах;
- настройку выполнения задачи после выхода из системы;
- способы отслеживания и сортировки активных процессов;
- завершение процессов.

Термины и утилиты:

- &
- bg
- fg
- jobs
- kill
- nohup
- ps
- top
- free
- uptime
- pgrep
- pkill
- killall
- screen

Linux содержит в себе множество инструментов для управления запущенными процессами: от сортировки, отслеживания состояния и загрузки системы до перевода процессов в фоновый режим и обратно.

Посмотреть запущенные процессы в текущей консоли можно при помощи команды **jobs**. Для вывода всех запущенных в данный момент процессов (независимо от консоли) используется команда ps с различным набором ключей, например:

Ps aux *(вывести процессы всех пользователей);*

Можно вывести процессы в консоль, отображая иерархию из взаимосвязи командой **pstree**.

Для отображения идентификаторов процессов по определенному критерию можно использовать команду **pgrep**, например:

pgrep -l -u root *(отобразить все процессы пользователя root)*

Чтобы запустить процесс в фоновом режиме необходимо указать ключ **&**, например:

Sleep 999 & *(запустить задачу sleep 999 в фоновом режиме);*

Для перевода процесса из фонового режима в активный используется команда **fg**, обратно – **bg**. В качестве аргумента команды следует указать номер процесса (*его можно увидеть на выводе команды jobs*).

Для того чтобы процесс продолжал выполняться после выхода пользователя из системы, его следует запускать командой **nohup**, например:

nohup sleep 1000 & *(запустить процесс "sleep 1000" в фоновом режиме, и выполнять его даже в случае выхода пользователя из системы);*

Для остановки процесса используется команда **kill**, например:

kill 2625 *(выключить процесс с PID=2625);*

killall sleep *(выключить все процессы с именем sleep для текущего пользователя);*

Для остановки процессов по указанному критерию можно использовать команду **pkill**, например:

pkill refox *(остановить процесс, в имени которого есть символы "refox");*

Для просмотра информации о работающих процессах в режиме реального времени используется диспетчер задача **top**. Используя его можно сортировать запущенные задачи по различным критериям и останавливать их *(клавиша K)* мягко *(sigterm)* и жестко *(sigkill)*.

Для отображения информации о времени работы системы и среднем времени загрузки АЧ используется команда **uptime**.

Для отображения информации о загрузке ОЗУ и раздела подкачки используется команда **free**.

Кроме того, Linux поддерживает работу с несколькими «экранами» при помощи команды **screen**. Она позволяет, в числе всего прочего, сворачивать запущенные приложения в консоли, в рамках той же консоли, например:

screen -S yandex ping ya.ru *(создать экран с именем yandex для команды ping ya.ru);*

screen –ls *(показать свернутые экраны);*

screen -r yandex *(развернуть экран с именем yandex).*

103.6 Изменение приоритетов процессов

Студент должен уметь менять приоритеты выполняющихся процессов

Изучаем:

- приоритеты по умолчанию создаваемых процессов;
- запуск программ с повышенным и пониженным приоритетом;
- изменение приоритета уже запущенной программы.

Термины и утилиты:

- nice
- ps
- renice
- top

Несмотря на то, что Linux является довольно стабильной системой, которая умеет автоматически подстраиваться под текущие условия работы, иногда приходится вручную управлять приоритетами процессов, для контроля над ресурсами ПК.

Для просмотра текущих процессов используется команда **ps**, например:

ps aux *(отобразить процессы по всем пользователям);*

ps –eo user,pid,pcpu,nice,comm *(отобразить процессы, выводя указанные данные);*

Приоритет процесса определяется параметром **nice** (в Ubuntu может принимать значения от -20 до +19): чем ниже значение, тем выше приоритет. По умолчанию значение nice для каждого процесса равно “-10”.

Для запуска процесса с указанным приоритетом используется команда **nice**, например:

nice –n -20 ping ya.ru *(запустить процесс с приоритетом “-20”);*

Для изменения приоритета запущенного процесса используется команда **renice**, например:

renice 15 –p 10241 *(изменить приоритет процесса с PID “10421”);*

renice -15 –u semaev *(изменить приоритет процессов пользователя “semaev”);*

Также для использования доступен диспетчер задач **top**, выводящий сведения о запущенных процессах в реальном режиме в консоль. Для изменения критерия сортировки процессов используются символы “<” и “>”.

103.7 Поиск информации при помощи регулярных выражений

Студент должен уметь осуществлять поиск необходимых файлов и информации внутри них.

Изучаем:

- создание простых регулярных выражений;
- инструменты, опирающиеся на эти выражения, для поиска файлов.

Термины и утилиты:

- grep
- egrep
- fgrep
- rgrep
- sed
- регулярные выражения.

Частой задачей в Linux является поиск необходимых файлов и сортировка информации внутри них. Для этой задачи служат регулярные выражения – специальный язык поиска и изменения информации, обладающий своим синтаксисом.

При отборе информации можно использовать стандартные групповые символы:

- ***^*** *- начало строки;*
- ***$*** *- конец строки;*
- ***.*** *- любой символ;*
- ***\<a*** *- слово, начинающееся с “a”;*
- ***\>a*** *- слово, заканчивающееся на “a”;*
- ***a-z]*** *- диапазон от “a” до ”z”;*
- ***[^t]*** *- не буква ”t”;*
- ***a|z*** *- ”a” или ”z”;*
- и т.д.;

Для простой сортировки набора строк используется команда **grep**, например:

grep oo file.txt *(отобразить в файле текст с двумя буквы «o» подряд);*

ls | grep ile *(вывести названия содержимого каталога с набором символов ‘ile’);*

grep ple$ file.txt *(отобразить в файле file.txt строки, заканчивающиеся на ‘ple’);*

Для расширенной сортировки и поиска по сложным регулярным выражениям используется команда **egrep** (также можно использовать ***grep -E***), например:

egrep ‘^(b|d)’ file.txt *(все строки, что начинаются с ”b” или ”d”);*

egrep ‘^[a-k]’ file.txt *(все строки, что начинаются с ”a” по ”k”);*

Для быстрой сортировки и поиска по набору символов без регулярных выражений используется команда **fgrep** (также можно использовать ***grep -F***), например:

fgrep c$ file.txt *(отобразить в file.txt последовательность символов “c$”);*

Для рекурсивной (включая вложенные каталоги и файлы) сортировки и поиска по набору символов используется команда **rgrep** (также можно использовать ***grep -R***), например:

rgrep word *(вывести позиции содержащие «word» в текущем и всех вложенных каталогах);*

Для изменения текстового потока согласно заданным правилам используется редактор sed, например:

sed –e ‘s/oo/aa/’ file.txt *(заменить в file.txt все “oo” на “aa”);*

sed –re ‘s/^(B|b)/C’ file.txt > newfile.txt *(в строках, начинающихся с «B» или «b», заменить первую букву на «C» и сохранить результат в newfile.txt);*

103.8 Текстовый редактор Vi.

Студент должен уметь редактировать файлы в vi, и ориентироваться в нем.

Изучаем:

- навигацию по документу с использованием vi;
- использование основных режимов работы vi;
- вставку, редактирование, удаление, копирование и поиск текста.

Термины и утилиты:

- vi
- /, ?
- h,j,k,l
- i, o, a
- c, d, p, y, dd, yy
- ZZ, :w!, :q!, :e!

Так как все основные настройки в Linux выполняются редактированием файлов, все журналы, устройства, демоны и т.д. тоже по сути являются файлами, то текстовый редактор является одним из важнейших инструментов управления ОС. Существует множество редакторов с различными возможностями, здесь же будет рассмотрен редактор Vi.

Vi имеет ***модальный*** интерфейс и одни и те же клавиши в разных режимах выполняют разные действия. По умолчанию работа начинается в ***командном режиме***.

Для перемещения по документу используются следующие символы:

- ***h*** *или* ← *(перемещение на символ влево);*
- ***j*** *или* ↓ *(перемещение на символ вниз);*
- ***k*** *или* ↑ *(перемещение на символ вверх);*
- ***l*** *или* → *(перемещение на символ вправо);*
- ***b*** *(перемещение в начало слова);*
- ***e*** *(перемещение в конец слова);*
- ***(*** *(перемещение в начало предложения);*
- ***)*** *(перемещение в конец предложения);*
- ***{*** *(перемещение в начало абзаца);*
- ***}*** *(перемещение в конец абзаца);*
- ***^*** *(перемещение в начало строки);*
- ***$*** *(перемещение в конец строки);*
- ***1G*** *(перемещение в начало файла);*
- ***G*** *(перемещение в конец файла).*

Для перехода в режим вставки можно воспользоваться следующими символами:

- ***I*** *(режим вставки перед текущим символом);*
- ***a*** *(режим вставки после текущего символа);*
- ***o*** *(вставить новую строчку);*
- ***s*** *(стереть символ и вставить).*

Для возврата в командный режим можно использовать клавишу ***Esc*** или сочетание ***Ctrl+C***

Находясь в режиме редактирования можно использовать буфер обмена:

- ***y*** *(скопировать);*
- ***d*** *(вырезать);*
- ***c*** *(изменить);*
- ***p*** *(вставить).*

Также возможны комбинации с символами перемещения по тексту и обозначениями **d** (строка), **w** (слово) и **l** (символ); например

yw *(скопировать слово);*

dd *(вырезать строку);*

cl *(изменить символ);*

d$ *(удалить с текущего места до конца строки);*

y} *(скопировать с текущего места до конца абазца);*

Для поиска информации доступны символы:

- ***/*** *(искать ниже по тексту набор символов, указанных после «/»);*
- ***?*** *(искать выше по тексту набор символов, указанных после «?»);*
- ***n*** *(показать следующий результат поиска);*
- ***N*** *(показать предыдущий результат поиска).*

Для завершения работы с файлом:

- ***:e!*** *(отменить все изменения, где «!» - игнорировать все предупреждения);*
- ***:w!*** *(записать изменения);*
- ***:q*** *(выйти из файла);*

Как всегда, возможны комбинации, например:

:wq! newfile.txt *(сохранить под именем newfile.txt и выйти);*

Также доступны сокращенные команды:

- ***ZZ*** *(сохранить и выйти);*
- ***ZQ*** *(не сохранять и выйти);*

104.1 Создание файловых систем.

Студент должен научиться разбивать диск на разделы, создавать на них файловые системы и управлять разделом подкачки.

Изучаем:

- управление таблицами разделов MBR;
- создание файловых систем mkfs (ext, XFS, VFAT);
- знакомство с ReiserFS и Btrfs;
- базовые навыки работы с GPT.

Термины и утилиты:

- fdisk
- gdisk
- parted
- mkfs
- mkswap

Linux поддерживает работу с различными файловыми системами. Для возможности работы с файловыми системами, кроме классической ext, необходим набор соответствующих инструментов, устанавливаемых вместе с набором ПО для каждой файловой системы.

Утилита **fdisk** предназначена для работы с носителями информации, использующими классическую таблицу разделов MBR (последние версии умеют работать и с GUID). Используя эту утилиту можно в интерактивном режиме создавать, удалять и изменять разделы жесткого диска, например:

fdisk –l *(вывести информацию о текущих разделах дисков);*

Утилита **gdisk** предназначена для работы с носителями информации, использующими таблицу разделов GUID. Используя эту утилиту можно в интерактивном режиме создавать, удалять, изменять и конвертировать разделы жесткого диска, например:

gdisk /dev/sdb *(начать работу с носителем /dev.sdb);*

Для создания файловой системы используется утилита **mkfs**, вызывающая специализированные инструменты для каждой отдельной файловой системы, например:

mkfs –t ext2 /dev/sdb1 *(отформатировать раздел /dev/sdb1 в ext2);*

mkfs.ext2 /dev/sdb1 *(отформатировать раздел /dev/sdb1 в ext2);*

Для расширенных возможностей управления различными файловыми системами (сжатие, расширение, перенос, копирование и т.д.) используется утилита **parted**, или ее релиз с графическим интерфейсом – **gparted**.

Для работы с нестандартными файловыми системами нужно ставить соответствующие им наборы ПО, например **xfsprogs** (для файловой системы XFS), **reiserfsprogs** (для файловой системы ReiserFS) или **btrfs-tools** (для файловой системы BTRFS). После установки этих пакетов ПО стандартные инструменты Linux смогут создавать и редактировать разделы с указанными файловыми системами.

Также следует отметить, что в Linux используется отдельный раздел подкачки, используемый в качестве временного хранилища информации в том случае, если оперативная память ПК заполнена. Для работы с ним используются следующие инструменты:

mkswap *(создать файловую систему для раздела подкачки);*

swapon *(включить раздел подкачки);*

swapoff *(выключить раздел подкачки).*

104.2 Проверка целостности файловых систем.

Студент должен умет управлять стандартной файловой системой, и понимать принципы работы с журналируемой файловой системой.

Изучаем:

- проверку целостности файловой системы;
- отслеживание айнодов и свободного пространства;
- исправление простых проблем с файловой системой.

Термины и утилиты:

- du
- df
- fsck
- e2fsck
- mke2fs
- debugfs
- dumpe2fs
- tune2fs
- инструменты XFS

Linux использует умеет работать с различными файловыми системами. Для поддержки всех возможностей отдельных файловых систем необходимо устанавливать соответствующие наборы ПО, содержащие дополнительные инструменты.

Утилита **df** (disk free) показывает свободное место в файловых системах, например:

df *(показывает в блоках килобайт)*

df –h *(показывает в мегабайтах и т.д.)*

df –i *(показывает в inode)*

Inode (индексный дескриптор) – идентификатор файла, содержащий о нем всю необходимую информацию.

Утилита **du** (disk usage) показывает занятое место в файловых системах, например:

du *(показывает размер текущей директории в блоках килобайт)*

du –h *(показывает в мегабайтах и т.д., включая вложенные папки)*

du –h /home/* *(показывает размер директории /home, отображая размер всех подкаталогов)*

du –h –summarize /home/* *(показывает размер директории /home, включая все подкаталоги)*

Утилита **fsck** предназначена для проверки файловых систем. Для работы с разными файловыми системами утилита запускает соответствующие им инструменты проверки. Проверку осуществляют для демонтированных файловых систем, неактивных в данный момент, например:

fsck /dev/sdb1 *(будет вызван инструмент проверки файловой системы, например для ext это будет инструмент e2fsck);*

fsck –t ext4 /dev/sdb1 *(можно сразу указать формат файловой системы, если он известен)*

Утилита **mkfs** предназначена для создания файловых систем. Для работы с разными файловыми системами утилита запускает соответствующие им инструменты, например:

mkfs –t xfs –f /dev/sdb1 *(создать файловую систему xfs на устройстве /dev/sdb1);*

mke2fs –t ext2 /dev/sdb1 *(создать файловую систему ext2 на устройстве /dev/sdb1);*

Для работы с файловой системой **XFS** (сейчас centos7 использует ее в качестве файловой системы по умолчанию), к примеру, используются следующие инструменты:

- ***xfs_check*** *для проверки;*
- ***xfs_repair*** *для восстановления;*
- ***xfs_info*** *для получения информации**;***
- ***xfs_metadump*** *для создания дампа.*

Для отладки классической файловой системы используется утилита **debugfs**. Этот инструмент в интерактивном режиме позволяет работать с айнодами файловой системы, например, в нем доступны следующие инструменты:

- ***ls*** *– просмотреть данные в системе;*
- ***lsdel*** *– показать удаленные файлы;*
- ***undel*** *– отменить удаление.*

Для вывода детальной информации о файловой системе (суперблоки, цилиндры, размер блока и т.д.) используется утилита **dumpfs**. Для работы с разными файловыми системами утилита запускает соответствующие им инструменты проверки, например:

dumpe2fs /dev/sdb1 > output.txt *(вывести информацию о файловой системе ext в файл output.txt)*

Утилита **tunefs** предназначена для настройки изменяемых параметров файловых систем. Для работы с разными файловыми системами утилита запускает соответствующие им инструменты настройки. Настройку осуществляют для демонтированных файловых систем, неактивных в данный момент, например:

tune2fs –O has_journal /dev/sdb1 *(включить журналирование на устройстве /dev/sdb1)*

Журналирование: опция введения журнала изменений для возможности их отката.

104.3 Монтирование файловых систем.

Студент должен уметь настраивать подключение файловых систем.

Изучаем:

- ручное подключение и отключение файловых систем;
- автоматическое монтирование при загрузке;
- настройку подключаемых портативных файловых систем.

Термины и утилиты:

- /etc/fstab
- /media
- mount
- umount

Linux использует файловую систему с единым корнем (в отличии от Windows, где каждый носитель информации имеет свой корень). Таким образом каждое отдельно устройство хранения информации подключается в единую корневую файловую систему через точки монтирования в иерархической древовидной структуре.

Для подключения устройства необходимо создать точку монтирования – любой пустой каталог. Традиционно такие точки монтирования создаются в каталоге **/mnt** (для временного монтирования пользовательских файловых систем) или **/media** (для подключения портативных устройств), например:

mkdir /mnt/hard_drive *(создать каталог /mnt/hard_drive)*

Для подключения файловой системы с носителя информации используется команда **mount**, например:

mount –t ext3 /dev/sdb1 /mnt/hard_drive *(подключить устройство /dev/sdb1 с типом файловой системы ext3 в каталог /mnt/hard_drive)*

Команда **mount**, используемая без аргументов, выводит перечень смонтированных файловых систем (в том числе и виртуальных).

Для отключения файловой системы используется команда **umount**, например:

umount /mnt/hard_drive *(в качестве аргумента можно указывать имя устройства или точку монтирования)*

___

Для автоматического подключения файловых систем используется файл /etc/fstab, в котором на каждой строке указаны id или имя устройства, точка монтирования, файловая система и дополнительные опции, например:

/dev/sdb1 *(имя устройства)* ***/mnt/hard_drive*** *(точка подключения)* ***ext3*** *(файловая система)* ***defaults*** *(опции по умолчанию)* ***0*** *(выключение дампа)* ***2*** *(порядок проверки)*

Часто используемые опции монтирования:

- auto / noauto *(подключение при загрузке ОС);*
- exec / noexec *(разрешение выполнения двоичных файлов с устройства);*
- ro *(только для чтения);*
- rw *(чтение и запись);*
- user / nouser *(разрешение подключения всем пользователям).*

Набор опций **defaults***: **rw,suid,dev,exec,auto,nouser,async**.*

Дамп (значения 1 и 0), говорит о включенном или выключенном резервном копировании устройства при помощи команды dump.

Порядок проверки указывает последовательность проверки файловых систем (0 – не проверять, 1 – корневая файловая система, 2 – все остальные).

После добавление записи в файл **/etc/fstab** указанное устройство можно монтировать при помощи команды **mount**, указывая только один аргумент – устройство или точку монтирования (так как все остальные опции уже указаны в /etc/fstab).

Часто вместо указания тома по адресу подключения (/dev/sda1 и т.д.) используется указание на том по UUID. Адресация /dev/sda1 и т.д. зависит от того, на какой порт какого контроллера жестких дисков подключен носитель информации, соответственно при переподключении дисков на другие порты – все может слететь. UUID же уникален, узнать его можно при помощи команды **blkid**

104.4 Управление квотами дисков.

Студент должен уметь управлять квотами дисков для пользователей.

Изучаем:

- установку квоты;
- работу с отчетами по квотам.

Термины и утилиты:

- quota
- edquota
- repquota
- quotaon

Использование квот позволяет управлять использованием дискового пространства как отдельных пользователей и групп, так и всех в целом. В данном руке рассматривается квотирование классической файловой системы ext.

Для возможности работы с квотами необходим пакет ПО **quota.**

Для того чтобы начать использовать квоты на устройстве, необходимо в опциях его монтирования в файле **/etc/fstab** дописать опции **usrquota** и **grpquota**.

Включить и выключить квоту можно командами **quotaon** и **quotaoff** , например:

quotaon /mnt/disk1 *(включить квоту для устройства подключенного к /mnt/disk1);*

quotaoff /mnt/disk1 *(выключить квоту для устройства подключенного к /mnt/disk1);*

Для редактирования квоты используется команда **edquota**, например:

edquota –u semaev *(изменить настройки квоты для пользователя semaev);*

edquota –g users *(изменить настройки квоты для группы users);*

При этом можно указать ограничение по объему занятого места, или по количеству файлов и папок. Квоты бывают следующих видов:

- жесткая – невозможно превысить;
- мягкая – можно превышать в течении недели.

Для получения отчета об использовании квот предназначена команда repquota, например:

repquota /mnt/disk1 *(посмотреть отчет по квоте для устройства подключенного к /mnt/disk1);*

104.5 Права доступа и владельцы файлов.

Студент должен уметь управлять доступом к объектам файловой системы, опираясь на механизмы разрешений и владельцев.

Изучаем:

- управление доступом к файлам и папкам;
- использование специальных битов;
- способы работы с масками создания файлов;
- работы с разрешением доступа для групп.

Термины и утилиты:

- chmod
- umask
- chown
- chgrp

В классической линуксовой файловой системе можно задавать права доступа на основе трех принадлежностей: владелец объекта, группа владельцев, все остальные.

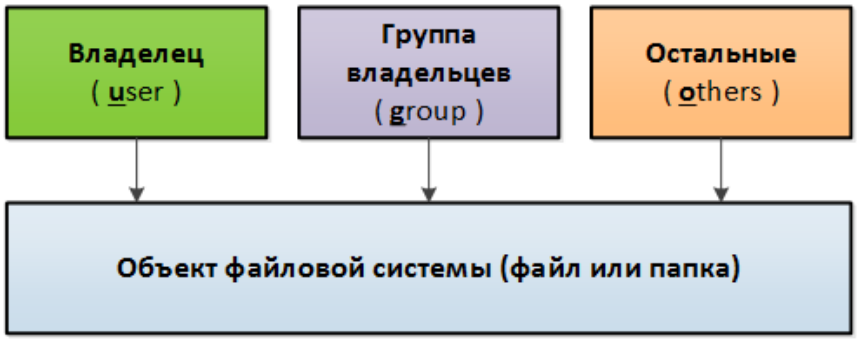

Рисунок 1. Владельцы объектов

Посмотреть текущие права доступа, равно как и владельце можно при помощи команды **stat**, например:

stat file.txt

Изменить владельца объекта можно при помощи команды **chown**, например:

chown root file.txt (сделать суперпользователя владельцем файла file.txt);

Изменить группу владельцев объекта можно при помощи команды **chgrp**, например:

chgrp users file.txt (сделать группу users группой владельцев файла file.txt);

Команда chgrp редко используется, так как возможны следующие варианты использования команды chown:

chown :users file.txt (сделать группу users группой владельцев файла file.txt);

chown root:users file.txt (сделать суперпользователя владельцем файла file.txt и группу users группой владельцев файла file.txt);

___

Для установки разрешений на доступ к объекту файловой системы Linux использует комбинации трех стандартных прав: чтение, запись и выполнение.

Права можно устанавливать, как указывая буквенные комбинации (r- read, w – write и x - execute), так и числовые (4 – read, 2 – write, 1 - execute), значения которых суммируются.

Число	Разрешение	Символ
0	Нет прав	---
1	Execute	--x
2	Write	-w-
3	Write, execute	-wx
4	Read	r--
5	Read, Execute	r-x
6	Read, Write	rw-
7	Read, Write, Execute	rwx

Рисунок 2. Права доступа к объектам

Права доступа назначаются отдельно для владельца, группы владельцев и остальных.

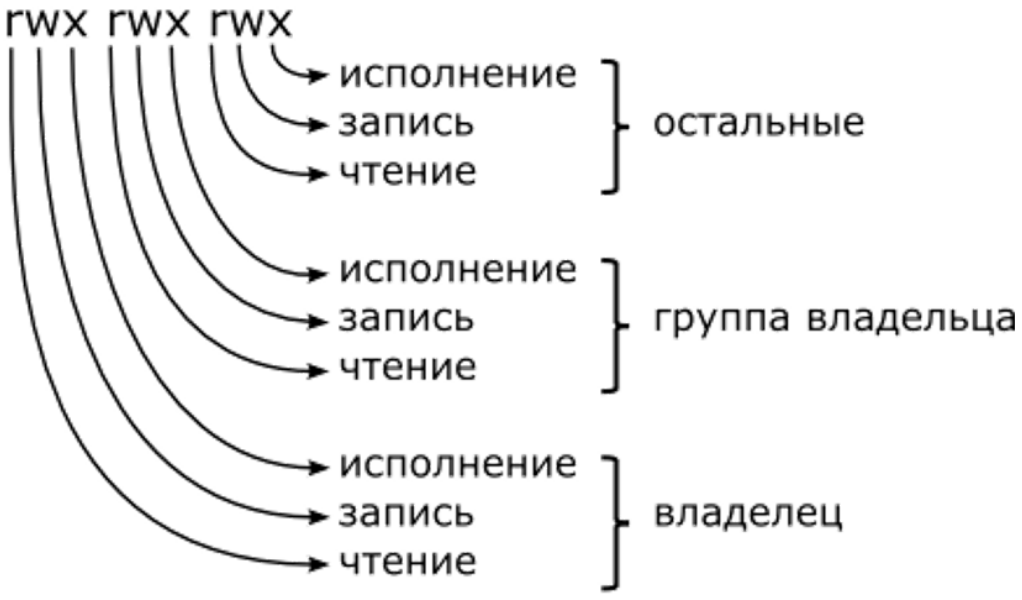

Рисунок 3. Права доступа к объектам

Для установки прав доступ используется команда chmod, например:

chmod 750 script (полные права владельцу, чтение и выполнение группе, ничего остальным);

chmod u+w script (дать право записи владельцу);

chmod ugo-x script (отобрать у всех право выполнения файла);

Просмотр прав доступа при помощи команды **stat** или полного вывода информации о содержимом текущего каталога **ls –l** показывает права доступа в буквенном виде, например:

rwxrw-r-- script (полные права владельцу, чтение и запись группе, чтение остальным);

Право выполнения, установленное на каталог говорит о возможности заходить в эту папку указанному пользователю или группе.

___

При работе с объектами файловой системы можно указывать маски создания файлов и папок, т.е. разрешения доступа по умолчанию для вновь создаваемых объектов.

Текущую маску можно посмотреть выполнив команду **umask** (в Ubuntu по умолчанию 022):

umask (отобразить числовое значение маски);

umask –s (отобразить символное значение маски).

Umask рассчитывается в восьмеричной системе счисления, и показывает какие биты следует сбросить в создаваемом объекте (побитовое И между побитовым НЕ маски и полным доступом). Напомни что полный доступ на директории – 777, а на файлы – 666. Что все это значит (https://ru.wikipedia.org/wiki/Umask):

Максимальные права установленные на директорию могут быть равны: 111 111 111

Маска 022 в восмеричной системе счисления выглядит как: 000 010 010

Ее побитовое НЕ (маски) выглядит как: 111 101 101

Располагаем их в столбик и выполняем операцию И:

111 111 111

111 101 101

---

111 101 101 или в двоичной системе счисления 755 (rwx r-x r-x)

Установить маску можно также командой **umask**, например:

umask 072 (полный доступ владельцу, нет доступа группе владельцев, чтение всем остальным).

___

Помимо группировки стандартных прав доступа (чтение, запись, выполнение), в Linux присутствуют специальные биты, устанавливающие дополнительные права доступа.

Доступны для использования следующие виды битов:

- **suid** - бит запуска от имени владельца;
- **sgid** - бит запуска от имени группы владельцев;
- **sticky** - бит защиты содержимого.

SUID (значение s или 4) применяется для запуска файла от имени владельца файла. Его можно установить командами:

chmod u+s script
chmod 4755 script

SGID (значение s или 2) применяется для доступа к файлам и папкам от имени группы владельцев, его можно установить командами:

chmod g+s script
chmod 2755 script

sticky-bit (значение t или 1) применяется для возможности изменения структуры каталогов только владельцем папки, его можно установить командами:

chmod o+t folder
chmod 1755 folder

Эти биты можно устанавливать одновременно, например:

chmod ug+s script
chmod 3755 folder

104.6 Жесткие и мягкие (символические) ссылки.

Студент должен уметь создавать и управлять ссылками на файлы.

Изучаем:

- создание ссылок;
- разницу между мягкими и жесткими ссылками;
- разницу между копированием и созданием ссылок;
- использование ссылок для выполнения административных задач.

Термины и утилиты:

- ln
- ls

Существует множество файловых систем, с различными принципами работы. В данном уроке будет рассмотрена классическая линуксовая файловая система ext, позволяющая прозрачно работать с адресацией диска.

Ключевым понятием при работе с ссылками является **айнод** (inode – индексный дескриптор): набор информации об объекте файловой системы. Айнод может содержать информацию о:

- размер объекта;
- id устройства, на котором расположен объект;
- id владельца объекта;
- id группы владельцев;
- права доступа;
- временные метки доступа;
- указатель на месторасположение на диске;
- размер блока;
- количество блоков;
- счетчик ссылок на объект и т.д.

Посмотреть айноды в текущем каталоге можно при помощи команды ***ls –li***

На объекты файловой системы можно создавать ссылки:

- **жесткие** – указывают на расположение файла на физическом устройстве (работают в пределах одного раздела носителя информации, но всегда указывают на файл, даже если он переименован, или перемещен внутри файловой системы носителя в другие каталоги);
- **мягкие** – указывают на расположение файла в файловой системе (работают между файловыми системами, но если исходный объект переместить или переименовать, ссылка работать перестает).

Создать жесткую ссылку можно командой **ln**, например:

ln file.txt hard.txt

Создать мягкую ссылку можно командой **ln -s**, например:

ln –s file.txt soft.txt

Жесткие ссылки нельзя создавать для каталогов.

Если создать жесткую ссылку на файл, и удалить файл, то к нему все еще можно получить доступ по созданной жесткой ссылке. То есть для удаления файла необходимо удаление всех жестких ссылок на него.

Созданная жесткая ссылка на файл выглядит в файловой системе как копия исходного файла. Чтобы увидеть разницу между скопированным файлом и жесткой ссылкой на него необходимо посмотреть их айноды (жесткая ссылка и оригинал будут иметь одинаковые).

104.7 Поиск и стандартное расположение системных файлов.

Студент должен ознакомиться с FHS (стандартная иерархия файловой системы), включая стандартное расположение файлов и предназначение папок.

Изучаем:

- стандартную структуру файловой системы;
- способы поиска файлов и команд.

Термины и утилиты:

- find
- locate
- updatedb
- whereis
- which
- type
- /etc/updatedb.conf

Все UNIX-подобные системы имеют схожее название, расположение и предназначение директорий в файловой системе. Стандарт иерархии файловой системы (FHS) – позволяет пользователям и разработчикам ПО ориентироваться в различных дистрибутивах Linux.

Не все каталоги можно найти в каждом дистрибутиве Linux, но в целом можно говорить о следующих стандартных расположениях:

/bin - базовые двоичные файлы команд;

/boot - файлы загрузчика;

/dev - устройства;

/etc - конфигурация ПК;

/home - домашние папки;

/lib - библиотеки и модули ядра;

/proc - информация о работающей системе;

/media - монтирование носителей;

/mnt - монтирование носителей;

/opt - дополнительное ПО;

/root - домашняя папка админа;

/sbin - основные программы настройки системы;

/srv - данные для системных служб;

/tmp - временные файлы;

/usr - бинарники файлы пользователей;

/var - переменные.

Для поиска информации в текущий момент используется команда **find**. Для поиска проиндексированной информации (поиск осуществляется намного быстрее) используется команда **locate**.

Индексацию информации для быстрого поиска можно выполнить принудительно командой **updatedb**. Выбрать каталоги для индексации можно редактированием конфигурационного файла **/etc/updatedb.conf**

Для поиска информации и командах и утилитах пользуются следующими командами:

- **which** – выводит путь к исполняемым файлам команды;
- **type** – показывает исполняемые файлы, псевдонимы, функции и т.д.;
- **whereis** – показывает исполняемые файлы, исходники, мануалы.