Администрирование ОС Linux

2-е издание, исправленное

Гончарук С.В.

Национальный Открытый Университет "ИНТУИТ"
2016

Администрирование ОС Linux/ С.В. Гончарук - М.: Национальный Открытый Университет "ИНТУИТ", 2016

Курс рассчитан на получение начальных знаний о системном и сетевом администрировании ОС Linux. Курс дает возможность пользователю получить твёрдые навыки при работе с операционной системой Linux, познакомиться со строением операционных систем семейства Unix, научиться эффективно ее использовать. Основной целью курса является получение обучаемым фундаментальных навыков администрирования Linux.

В процессе обучения вы познакомитесь с файловой системой Linux (основными понятиями, командами навигации и управления файлами, структурой файловой системы); системой распределения прав (учетные записи, группы, права доступа на файлы, в том числе и специальные права доступа); методам эффективного контроля и управления процессами; методам работы с командными оболочками и основам создания сценариев; планированием заданий и средствами для управления запланированными заданиями; принципам инициализации системы SVR4; познакомитесь с основами стека TCP/IP и базовыми инструментами для работы с сетью в Linux.

Введение в операционную систему Unix

Познакомить обучаемых с предметом обучения: Unix-way, история Юникс, разновидности Юникс. Линукс, что такое Linux, средства просмотра системной информации.

Ход занятия

1. Сегодня мы с вами начинаем изучать Linux, как одну из составных частей мира информационных технологий под названием Unix. Все вы, ну или практически все раньше сталкивались с ОС семейства Windows. Я хочу сделать небольшое примечание – Unix не Windows, он не похож на него (не считая внешнего сходства) и приемы работы в unix отличаются от приемов работы в Windows. Unix – это не только операционная система, это еще и идеология работы с компьютером. Те правила, о которых мы с вами будем говорить сейчас лежат в основе изучения Linux, да и Unix вообще. Общий термин для них – Unix Way:

- одна задача – одна программа . В Unix не принято делать комбайны для выполнения "сразу всего". Программа делается таким образом, чтобы она могла выполнять одно простое действие, но выполняла его хорошо.
- есть множество путей решения.Для решений той или иной комплексной задачи каждый может выбирать свой набор простых компонентов для ее решения.
- все есть файл . Самая замечательная концепция в unix. Действительно, в Unix все представлено в виде файлов – программы, настройки, системные данные и даже устройства. И с устройствами можно работать как с простыми файлами.

Остальную часть Unix way мы с Вами будем изучать в течение этого курса. Будьте готовы к изучению не просто новых программ, а новых методов работы на компьютере.

2. В 1969 году Кен Томпсон и Денис Ритчи, работники корпорации AT&T, создали небольшую операционную систему для компьютера PDP-7. Эта операционная система получила название Unix. Однако в планы компании AT&T не входило распространение этой операционной

системы, и она предоставила ее за символическую плату учебным заведениям США, не организовав при этом службы технического сопровождения, исправления ошибок и вообще не дав никаких гарантий.

Вследствие этого пользователи, почти все являвшиеся представителями университетских вычислительных центров, были вынуждены сотрудничать друг с другом. Они сами устраняли ошибки, создавали полезные программы и утилиты и совместно их использовали. Результатом их работы стала целая серия версий Unix, распространяемых под эгидой компании Bell Labs вплоть до 1990 года (Последней версией была Unix System V Release 4 – SVR4).

Одна из групп пользователей Unix находилась в калифорнийском университете в Беркли. В 1977 году специалисты этого учебного заведения сделали следующий шаг в истории Unix и приступили к распространению магнитных лент с операционной системой 2BSD (Berkeley Software Distribution). С тех пор было продано 75 копий.

На основании Unix SVR4 и BSD были созданы все современные разновидности Unix.

3. Существует множество разновидностей Unix и Unix-подобных систем. К наиболее известным из них относятся Solaris (ранее SunOS) корпорации SUN Microsystems, AIX компании IBM, DEC Unix фирмы DEC, SCO UnixWare и прочие. Все вышеназванные системы являются коммерческими, и многие из них имеют высокую цену. Они работают на различных архитектурах (Intel, Sparc, Alpha, PowerPC и т.д.). Однако наибольший интерес сегодня в мире Unix приобрели операционные системы, построенные на модели открытого кода, такие как Linux.

Linux изначально была разработана как свободно распространяемая версия Unix. В 1991 году студент Хельсинского университета Линус Торвальдс выпустил первую версию Linux. Она была основана на операционной системе Minix – ограниченном аналоге Unix для ПК. После выпуска первого "почти безошибочного" релиза в марте 1992 года, многие программисты мира подключились к разработке этой операционной системы, и она стала расти.

На сегодняшний день Linux является полнофункциональным, открытым

и, зачастую, бесплатным, аналогом Unix. Но этого бы не произошло, не будь программного обеспечения в рамках проекта GNU (GNU's not Unix, GNU – это не Unix). Linux содержит много утилит GNU, включая трансляторы многих языков программирования (C, C++, Fortran, Pascal, LISP, Ada, BASIC, SmallTallk, Perl, PHP, Tcl/Tk и др.), отладчики, текстовые редакторы, утилиты печати и многое другое. Проект GNU развивается под эгидой фонда свободно распространяемого программного обеспечения – Free Software Foundation (FSF).

4. Linux является свободно распространяемой многозадачной многопользовательской операционной системой, похожей на Unix. Linux была разработана специально для платформы ПК (с процессором Intel) и благодаря преимуществам архитектуры позволяет достичь производительности, сравнимой с мощными рабочими станциями Unix. Linux также переносилась и на другие платформы, но все эти версии сходны с версией для ПК.

Давайте рассмотрим операционную систему как единый комплекс. Ниже приведен список того, что мы получим, установив ее:

Ядро Linux:

Ядро - это основная часть операционной системы. Оно отвечает за распределение памяти, управление процессами и периферийными устройствами. Для поддержки большего объема оперативной памяти по сравнению с физически установленной на компьютере, ядро позволяет использовать область подкачки, размещая страницы оперативной памяти на жестком диске.

Ядро Linux поддерживает множество файловых систем, включая FAT, FAT32. Собственные файловые системы Linux (ext2fs и ext3fs) разработаны для оптимального использования дискового пространства.

Утилиты GNU:

Linux содержит множество утилит GNU, без которых была бы невозможна работа с операционной системой.

X Window:

Графический интерфейс пользователя представлен в Linux средой X Window. Различные оконные менеджеры (IceWM, WindowMaker, Fluxbox и прочие) и графические среды такие как KDE и GNOME, обеспечивают удобный интерфейс и работу со средствами мультимедиа.

Интерфейсы DOS и Windows:

Поскольку Linux была создана для компьютеров класса ПК, разработчики посчитали необходимым обеспечить совместимость с программами MS-DOS. В Linux предлагается эмулятор DOS как часть дистрибутива. Он позволяет исполнять DOS-приложения непосредственно из-под Linux. Для запуска программ Microsoft Windows было разработано несколько средств. Наиболее известное из них – WINE – свободная реализация Windows API. Wine также входит в большинство дистрибутивов Linux.

Linux позволяет без проблем переносить файлы между файловыми системами DOS и Windows, напрямую обращаясь к соответствующим разделам на диске, хотя это и требует некоторой настройки.

Сетевая поддержка:

TCP/IP – основная сетевая система используемая Unix и Linux. TCP/IP – это целый набор протоколов, разработанных для Internet. Однако для объединения в локальные сети машин Unix тоже используется TCP/IP. Также Linux поддерживает другие протоколы, такие как IPX/SPX, AppleTalk и т.д.

5. Встает вопрос – как же узнать тип операционной системы, установленной у вас на компьютере. Для получения такой информации существует утилита `uname (Unix NAME)` .

`uname` , запущенная без параметров, покажет базовое имя системы:

```
gserg@ADM:~$ uname
  Linux
```

Также она может принимать следующие параметры:

`-s` – показывает название ядра системы

`-r` – имя релиза ядра системы

`-v` – имя версии, а также дату компиляции ядра

`-o` – операционную систему

`-p` – тип процессора

`-m` – тип оборудования (i386, i686, Alpha)

`-a` – всю информацию сразу

Это не все параметры uname. О справке Linux мы поговорим с вами на лекции 5.

Команда `free` показывает объем памяти и объем ее использования, а также использование `swap` :

```
gserg@ADM:~$ free
             total       used       free     shared    buffers     cached
Mem:        498916     483332      15584          0       4392     112924
-/+ buffers/cache:     366016     132900
Swap:      1453840     412532    1041308
```

Обратите внимание, что практически вся свободная память резервируется системой под дисковые буферы и дисковый кэш, что позволяет Linux более эффективно работать с дисками.

Состояние системы в данный момент, степень ее загруженности и время без перезагрузок показывает команда *`uptime`* :

```
gserg@ADM:~$ uptime
 14:24:08 up 1 day,  6:01,  2 users,  load average: 0.08, 0.19, 0.16
```

Первым идет текущее время, потом, после слова `up` – время, прошедшее с момента включения компьютера, потом показано сколько пользователей зарегистрировано сейчас в системе (это может быть и несколько регистраций одного и того же пользователя) и загрузка системы. Загрузка системы показывается в количестве процессов, одновременно работающих в системе, среднее значение за 1-ну, 5 и 15

минут. Система считается нагруженной, если это значение превышает 1 в расчете на 1 процессор.

Другим средством мониторинга производительности является команда `vmstat` :

```
[gserg@admin ~]$ vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu--
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 268928 776168  15072 203316    1    2    10    14  207  225 13  3 84  0
```

Эта команда выдает за раз достаточно большой объем информации.

Раздел `procs` :

`r` — количество ожидающих процессов

`b` — количество спящих процессов

Раздел `memory` :

`swpd` — объем используемой виртуальной памяти

`free` — объем свободной виртуальной памяти

`buff` — объем памяти, занятой под дисковые буферы

`cache` - объем памяти, занятой под дисковый кэш

Раздел `swap` :

`si` — объем памяти, подкачанной с диска

`so` — объем памяти, выгруженной на диск

Раздел `io` :

`bi` — количество блоков, отправленных на блочное устройство

`bo` — количество блоков, прочитанных с блочного устройства

Раздел `system` :

`in` — количество прерываний в секунду

`cs` — количество переключений контекста в секунду

Раздел `cpu` :

`us` — время выполнения кода уровня пользователя (в процентах от общего времени)

`sy` — время выполнения кода уровня системы (в процентах от общего времени)

`id` — время простоя процессора (в процентах от общего времени)

`wa` — время ожидания ввода/вывода

`st` — время работы виртуальной машины уровня ядра

`vmstat` показывает при простом запуске усредненные показатели за все время с момента запуска системы. Но можно попросить `vmstat` вывести показатели за заданное количество времени:

```
[gserg@admin ~]$ vmstat 1 5
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu--
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0 268844 742148  16620 212452    1    2    10    14  216  230 13  3 84  0
 0  0 268844 742140  16628 212436    0    0     0    48 1097  414  6  2 92  0
 0  0 268844 742140  16628 212436    0    0     0     0 1105  392  5  1 94  0
 0  0 268844 742172  16628 212436    0    0     0     0 1090  345  4  1 95  0
 0  0 268844 742172  16628 212436    0    0     0     0 1107  403  6  1 93  0
```

В примере выведена информация за каждую секунду на протяжении 5 секунд. Если второй параметр (`5`) не указывать, то `vmstat` будет выводить информацию каждую секунду до нажатия Ctrl+C :

```
[gserg@admin ~]$ vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu--
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
```

```
 0  0 268844 740824  16824 212720    1    2    10    14  217  230 13  3 84  0
 0  0 268844 740856  16824 212720    0    0     0     0 1088  488  8  2 90  0
 0  0 268844 740856  16824 212720    0    0     0     0 1392  873 14  4 82  0
^C
```

Для просмотра размеров файловых систем используется команда `df`:

```
[gserg@admin ~]$ df
Файловая система     1K-блоков      Исп  Доступно  Исп% смонтирован
/dev/hdb2             36733400  10074596  24762736  29% /
/dev/hdb1               101086     16228     79639  17% /boot
tmpfs                   647688         0    647688   0% /dev/shm
```

Без параметров команда выводит данные в виде количества блоков по 1 килобайту. Для человека это не очень удобная подача информации. У `df` существует ключ `-h` (или `--human`), позволяющий увидеть объемы в привычных нам единицах измерения:

```
[gserg@admin ~]$ df --human
Файловая система      Разм  Исп  Дост  Исп% смонтирована на
/dev/hdb2              36G  9,7G   24G  29% /
/dev/hdb1              99M   16M   78M  17% /boot
tmpfs                 633M     0  633M   0% /dev/shm
```

Файловая система Linux

Основные понятия: корневой каталог, точка монтирования, домашний каталог, типы файлов. Обычные файлы. Каталоги. Файлы устройств. Команды. Навигация по файловой системе: команды cd, pushd, popd, pwd. Создание, удаление и копирование файлов. Команды touch, rm, cp. Операции с каталогами. Команды mkdir и rmdir. Важнейшие каталоги файловой системы Linux.

Ход занятия

1. Файловая система Linux, в отличие от операционных систем семейства Windows не разделена по томам (дискам, устройствам), а имеет единую древовидную структуру, в основе которой лежит корневой каталог. Корневой каталог - это уровень файловой системы, выше которого по дереву каталогов подняться невозможно. В Linux корневой каталог обозначается как / (именно / - слэш, а не \ - обратный слэш). Система позволяет устанавливать много корневых каталогов. Так например для некоторого пользователя `ftp /home` будет корневым каталогом и при обращении к клиенту `ftp` на смену каталога на корневой пользователь будет попадать в `/home`.

Возникает вопрос, а как тогда разные физические устройства участвуют в формировании единой файловой системы? Сделаем небольшой экскурс в историю. В то время, когда создавалась ОС Unix, устройства – накопители информации представляли собой ящик размером с письменный стол и назывались магнитными барабанами. В то время не было необходимости подключать и отключать его по нескольку раз в час. Поэтому не был выработан и механизм быстрой смены. Для подключения любого устройства к файловой системе используется так называемая точка монтирования – каталог, все вложенные уровни которого являются файловой системой на устройстве-носителе. Например, при монтировании дискеты обычно используется каталог `/media/floppy` . То есть, все каталоги и файлы, находящиеся внутри `/media/floppy` на самом деле содержатся на дискете, вставленной в дисковод компьютера. Для подключения, или монтирования, устройств используется специальная команда, которую мы изучим на следующих занятиях. Таким образом подключаются и сетевые файловые

системы, то есть такие системы, которые реально находятся где-то на сервере сети, однако различий в работе с ними пользователь не ощущает и видит сетевые файлы и каталоги, как если бы они были расположены на локальном компьютере.

Есть у файловой системы Linux и еще одна особенность: пользователям в ней выделяется домашний каталог – специальный каталог, необходимый для хранения пользователем своих личных данных. При входе пользователя в систему, он сразу оказывается в своем домашнем каталоге. Обычно права доступа к домашнему каталогу с консоли пользователя выставлены таким образом, что доступ к каталогу запрещен всем кроме владельца и администратора.

2. В файловой системе Линукс различают несколько типов файлов. Понятие "файл" включает в себя также и интерфейсы работы с периферийными устройствами, и каналы, позволяющие разным процессам в системе обмениваться данными.

```
[student@ns lesson_2]$ ls -l
total 40
-rwxr-xr-x    1 root     root         2872 Aug 27  2001 arch
-rw-rw-rw-    1 root     root          612 Jun 25  2001 chain.b
brw-rw----    1 root     disk        3,   1 Feb  3 15:38 hda1
drwxrwxrwx    2 root     root        32768 Feb  3 15:38 ida
```

Основные типы устройств:

`-` простой файл

`d` каталог

`l` ссылка

`b` блочное устройство

`c` символьное устройство

3. Навигация по файловой системе является одним из самых важных навыков при работе с операционной системой Linux. Основными командами, используемыми при навигации по файловой системе,

являются:

`pwd` – показывает полное имя каталога, в котором находится пользователь.

```
[student@ns student]$ pwd
/home/student
[student@ns student]$
```

`cd` – изменяет текущий каталог на указанный. `cd` без параметров или с параметром `~` изменяет текущий каталог на домашний. `cd` с параметром `..` изменяет каталог на тот, который находится на один уровень выше по дереву каталогов.

```
[student@ns student]$ pwd
/home/student
[student@ns student]$ cd primer
[student@ns primer]$ pwd
/home/student/primer
[student@ns primer]$ cd ..
[student@ns student]$ pwd
/home/student
[student@ns student]$ cd /home/student/primer
[student@ns primer]$ pwd
/home/student/primer
[student@ns primer]$ cd
[student@ns student]$ pwd
/home/student
[student@ns student]$ cd /bin
[student@ns bin]$ pwd
/bin
[student@ns bin]$ cd ~
[student@ns student]$ pwd
/home/student
[student@ns student]$_
```

`pushd,popd` – эти команды работают в связке. Команда `pushd` изменяет каталог на указанный. `pushd` с параметром .. изменяет каталог на тот, который находится на один уровень выше по дереву

каталогов. Основное отличие этой команды от `cd` в том, что вся история смены каталогов запоминается в стек и потом может быть использована для быстрой обратной навигации с помощью команды `popd`.

```
[student@ns student]$ pushd /var
/var ~
[student@ns var]$ pushd log
/var/log /var ~
[student@ns log]$ popd
/var ~
[student@ns var]$ popd
~
[student@ns student]$
```

4. Пользователю Linux ежедневно приходится создавать, копировать и удалять файлы. Эти операции являются такими же важными, как перемещение по файловой системе.

Команда `cp` используется для копирования файлов. Её синтаксис таков:

```
cp [параметры] <имя файла источника> <имя каталога приемника>
```

Наиболее часто используемым параметром является параметр `-R`, позволяющий рекурсивно копировать каталоги, т.е со всем их содержимым.

```
[student@ns primer_3]$ cd ../primer_1/in_primer_1
[student@ns in_primer_1]$ ls
[student@ns primer_3]$ cd ../primer_3
[student@ns primer_3]$ cp in_primer_3 ../primer_1/in_primer_1/
[student@ns primer_3]$ cd ../primer_1/in_primer_1
[student@ns in_primer_1]$ ls
in_primer_3
[student@ns primer_2]$ cd ../primer_2
[student@ns primer_2]$ ls
in_primer_2  in_primer_2_2
[student@ns primer_3]$ cp -R * ../primer_2
[student@ns primer_3]$ cd ../primer_2
```

```
[student@ns primer_2]$ ls
in_primer_2  in_primer_3
[student@ns primer_2]$ cd in_primer_3
```

Команда `touch` позволяет создавать файлы. Её применение наиболее просто: `touch <имя файла>`.Если файл с заданным именем существует в текущей директории, команда `touch` обновит его время создания на текущее.

```
[student@ns lesson_3]$ ls
primer_1  primer_2  primer_3
[student@ns lesson_3]$ touch file
[student@ns lesson_3]$ ls
file  primer_1  primer_2  primer_3
[student@ns lesson_3]$_
```

Команда `rm` используется для удаления файлов. Основные параметры, используемые с командой `rm` это `-i` (удаление с подтверждением удаления), `-r` (рекурсивное удаление) и `-f` (удаление всех файлов без подтверждения), `-v` (подробное описание производимых действий). Параметры `-r` и `-f` используются для удаления большого количества файлов. Но при их использовании необходимо быть предельно осторожным, т.к. с помощью этих параметров можно уничтожить систему.

```
[student@ns lesson_3]$ rm -iv ./file
rm: remove `./file'? y
removing `./file'
[student@ns lesson_3]$_
```

5. Операции с каталогами также важны для пользователя Linux , как и основные операции с файлами. Основные команды, используемые при работе с каталогами это – `rmdir` и `mkdir` .

Команда `mkdir` позволяет создать каталог:

```
[student@ns student]$ ls
file  primer_1  primer_2  primer_3
[student@ns student]$ mkdir catalog
```

```
[student@ns student]$ ls
catalog file primer_1 primer_2 primer_3
[student@ns student]$_
```

`rmdir`, наоборот, позволяет удалить каталог:

```
[student@ns student]$ ls
catalog file primer_1 primer_2 primer_3
[student@ns student]$ rmdir catalog
[student@ns student]$ ls
file primer_1 primer_2 primer_3
[student@ns student]$_
```

Обращаю ваше внимание на то, что команда `rmdir` без использования дополнительных параметров, может удалять ТОЛЬКО ПУСТЫЕ КАТАЛОГИ.

6. Файловая система Linux, как и любой другой unix-подобной операционной системы, имеет строгую структуру каталогов. Каждый дистрибутив Linux может несколько изменять структуру в зависимости от предпочтений разработчиков. Мы рассмотрим те каталоги, которые используются в каждом дистрибутиве:

Имя каталога	Описание
`/bin`	в этом каталоге находятся основные исполняемые файлы, жизненно необходимые для функционирования системы
`/boot`	содержит ядро операционной системы и карты загрузки, а также конфигурационные файлы загрузчиков (*`lilo`*, *`grub`*)
`/dev`	содержит файлы, которые являются интерфейсом с периферийными устройствами
`/etc`	содержит основные файлы настроек приложений Linux
`/home`	содержит домашние папки пользователей
`/lib`	содержит основные библиотеки, необходимые для нормальной работы системы
`/lost+found`	информация, восстановленная при проверке файловой

`/lost+found`	системы на наличие ошибок
`/media`	точки монтирования отключаемых устройств (usb-диски, CD, floppy)
`/mnt`	точки монтирования ISO-образов, сетевых файловых систем, других постоянных файловых систем
`/opt`	альтернатива usr, для коммерческого ПО или ПО, не входящего в основной дистрибутив
`/proc`	внутри этого каталога находится виртуальная файловая система proc, создаваемая ядром Linux "на лету". Содержит общую информацию о системе и подробную о процессах.
`/root`	домашний каталог пользователя root
`/sbin`	утилиты суперпользователя
`/srv`	файлы, выкладываемые для доступа всевозможных внешних служб (например , `tftp`)
`/sys`	внутри этого каталога также находится виртуальная файловая система, только она содержит подробную информацию о процессах
`/tmp`	в этом каталоге находятся временные файлы, используемые запущенными в данный момент процессами
`/usr`	программы, библиотеки и другие данные пользовательских приложений
`/var/log`	содержит файлы журналов

Учетные записи в Linux

Понятие учетной записи и аутентификации. Файлы /etc/passwd и /etc/group, /etc/shadow и /etc/gshadow. Учетная запись root. Пароли в Linux. Команды login, su, newgrp, passwd, gpasswd, chage.

Ход занятия

1. Linux, как и любая unix-подобная система, является не только многозадачной, но и многопользовательской, т.е. эта операционная система позволяет одновременно нескольким пользователям работать с ней. Но система должна как-то узнавать, какой или какие из пользователей работают в данный момент. Именно для этих целей в Linux существует два понятия – учетные записи и аутентификация, которые являются частями одного механизма.

Учетная запись пользователя – это необходимая для системы информация о пользователе, хранящаяся в специальных файлах. Информация используется Linux для аутентификации пользователя и назначения ему прав доступа.

Аутентификация – системная процедура, позволяющая Linux определить, какой именно пользователь осуществляет вход.

Вся информация о пользователе обычно хранится в файлах /etc/passwd и /etc/group.

/etc/passwd – этот файл содержит информацию о пользователях. Запись для каждого пользователя занимает одну строку:

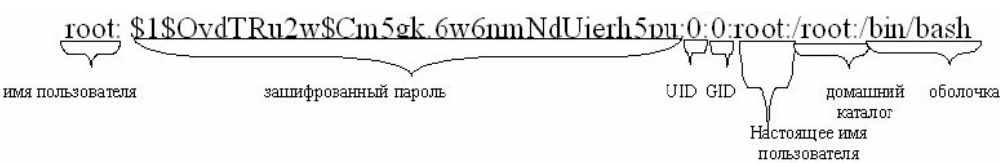


Рис. 3.1.

имя пользователя – имя, используемое пользователем на все приглашения типа `login` при аутентификации в системе.

зашифрованный пароль – обычно хешированный по необратимому алгоритму MD5 пароль пользователя или символ '!', в случаях, когда интерактивный вход пользователя в систему запрещен.

UID – числовой идентификатор пользователя. Система использует его для распределения прав файлам и процессам.

GID – числовой идентификатор группы. Имена групп расположены в файле /etc/group. Система использует его для распределения прав файлам и процессам.

Настоящее имя пользователя – используется в административных целях, а также командами типа `finger` (получение информации о пользователе через сеть).

Домашний каталог – полный путь к домашнему каталогу пользователя.

Оболочка – командная оболочка, которую использует пользователь при сеансе. Для нормальной работы она должна быть указана в файле регистрации оболочек /etc/shells .

/etc/group – этот файл содержит информацию о группах, к которым принадлежат пользователи:

Рис. 3.2.

Имя группы – имя, применяемое для удобства использования таких программ, как `newgrp` .

Шифрованный пароль – используется при смене группы командой `newgrp`. Пароль для групп может отсутствовать.

GID – числовой идентификатор группы. Система использует его для распределения прав файлам и процессам.

Пользователи, включенные в несколько групп – В этом поле через запятую отображаются те пользователи, у которых по умолчанию (в

файле /etc/passwd) назначена другая группа.

На сегодняшний день хранение паролей в файлах passwd и group считается ненадежным. В новых версиях Linux применяются так называемые теневые файлы паролей – shadow и gshadow. Права на них назначены таким образом, что даже чтение этих файлов без прав суперпользователя невозможно. Нужно учесть, что нормальное функционирование системы при использовании теневых файлов подразумевает одновременно и наличие файлов passwd и group. При использовании теневых паролей в /etc/passwd и /etc/group вместо самого пароля устанавливается символ 'x', что и является указанием на хранение пароля в /etc/shadow или /etc/gshadow.

Файл shadow хранит защищенную информацию о пользователях, а также обеспечивает механизмы устаревания паролей и учетных записей. Вот структура файла shadow :

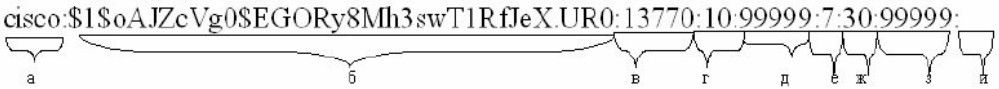


Рис. 3.3.

- а - имя пользователя ;
- б - шифрованный пароль – применяются алгоритмы хеширования, как правило MD5 или символ '!', в случаях, когда интерактивный вход пользователя в систему запрещен;
- в - число дней с последнего изменения пароля, начиная с 1 января 1970 года;
- г - число дней, перед тем как пароль может быть изменен;
- д - число дней, после которых пароль должен быть изменен;
- е - число дней, за сколько пользователя начнут предупреждать, что пароль устаревает;
- ж - число дней, после устаревания пароля для блокировки учетной записи;
- з -дней, отсчитывая с 1 января 1970 года, когда учетная запись будет заблокирована;
- и - зарезервированное поле;

Файл gshadow так же накладывает дополнительную функциональность,

вкупе с защищенным хранением паролей групп. Он имеет следующую структуру:

Рис. 3.4.

Имя группы – имя, используемое для удобства использования таких программ, как newgrp.

Шифрованный пароль – используется при смене группы командой `newgrp` . Пароль для групп может отсутствовать.

Администратор группы – пользователь, имеющий право изменять пароль с помощью `gpasswd`.

Список пользователей – В этом поле через запятую отображаются те пользователи, у которых по умолчанию (в файле /etc/passwd) назначена другая группа.

2. В Linux, кроме обычных пользователей, существует один (и только один) пользователь с неограниченными правами. Идентификаторы `UID` и *`GID`* такого пользователя всегда `0` . Его имя, как правило, `root` , однако оно может быть легко изменено (или создано несколько символьных имен с одинаковым GID и UID), так как значение для применения неограниченных прав доступа имеет только GID 0 . Для пользователя `root` права доступа к файлам и процессам не проверяются системой. При работе с использованием учетной записи `root` необходимо быть предельно осторожным, т.к. всегда существует возможность уничтожить систему.

3. В Linux используется развитая система распределения прав пользователям. Но для точного опознания пользователя одного имени недостаточно с точки зрения безопасности. Именно поэтому используется и пароль – произвольный набор символов произвольной длины, обычно ограниченной лишь используемыми методами шифрования.

Сегодня в большинстве версий Linux пароли шифруются по алгоритмам 3DES и MD5. Когда алгоритм 3DES является обратимым, то есть такой пароль можно расшифровать, MD5 – это необратимое преобразование. Пароли, зашифрованные по алгоритму 3DES не применяются при использовании теневых файлов для хранения паролей.

При аутентификации, пароль, введенный пользователем, шифруется тем же методом, что и исходный, а потом сравниваются уже зашифрованные копии. Если они одинаковые, то аутентификация считается успешной.

Учитывая ежедневно увеличивающиеся требования к безопасности, в Linux есть возможность использовать скрытые пароли. Файлы /etc/passwd и /etc/group доступны для чтения всем пользователям, что является довольно большой брешью в безопасности системы. Именно поэтому в современных версиях Linux предпочтительнее использовать скрытые пароли. Такие пароли располагаются в файлах /etc/shadow и /etc/gshadow, для паролей пользователей и групп соответственно.

4. Команда `login` запускает сеанс интерактивной работы в системе. Она проверяет правильность ввода имени и пароля пользователя, меняет каталог на домашний, выстраивает окружение и запускает командный интерпретатор. Команду `login` как правило не запускают из командной строки — это обычно за пользователя делают менеджеры консоли — например `getty` или `mgetty`.

Команда `su` (switch user) позволяет сменить идентификатор пользователя уже в процессе сеанса. Синтаксис ее прост: `su username`, где `username` – имя пользователя, которое будет использоваться. После этого программа запросит пароль. При правильно введенном пароле, `su` запустит новый командный интерпретатор с правами пользователя, указанного `su` и присвоит сеансу его идентификаторы. Если имя пользователя опущено, то команда `su` использует имя `root`.

```
[student@ns student]$ su root
Password:
[root@ns student]#_
```

При использовании команды `su` пользователем `root` она, как правило, не запрашивает пароль.

Команда `newgrp` аналогична по своим возможностям `su` с той разницей, что происходит смена группы. Пользователь должен быть включен в группу, которая указывается в командной строке `newgrp` . При использовании команды `newgrp` пользователем `root` она никогда не запрашивает пароль. Синтаксис команды аналогичен синтаксису команды `su: newgrp groupname`, где `groupname` – имя группы, на которую пользователь меняет текущую.

Команда `passwd` является инструментом для смены пароля в Linux. Для смены своего пароля достаточно набрать в командной строке `passwd`:

```
[student@ns student]$ passwd
Changing password for student
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully
[student@ns student]$_
```

Для смены пароля группы и управления группой используется команда `gpasswd` Для смены пароля достаточно набрать в командной строке `gpasswd GROUPNAME` . Сменить пароль вам удастся только если Вы являетесь администратором группы. Если пароль не пустой,то для членов группы вызов `newgrp` пароля не требует, а не члены группы должны ввести пароль. Администратор группы может добавлять и удалять пользователей с помощью параметров `-a` и `-d` соответственно. Администраторы могут использовать параметр `-r` для удаления пароля группы. Если пароль не задан, то только члены группы с помощью команды `newgrp` могут войти в группу. Указав параметр `-R` можно запретить доступ в группу по паролю с помощью команды newgrp (однако на членов группы это не распространяется). Системный администратор (`root`) может использовать параметр `-A`, чтобы назначить группе администратора.

Команда `chage` управляет информацией об устаревании пароля и

учетной записи. Обычный пользователь (не `root`) может использовать команду только для просмотра своих параметров устаревания пароля:

```
gserg@ADM:/$ chage -l gserg
Last password change                                    : Май 03, 2007
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

Суперпользователь же может использовать также иные параметры, такие как:

`-d дата` (в формате системной даты, например ДД.ММ.ГГГГ) – устанавливает дату последней смены пароля пользователем.

`-E дата` – установить дату устаревания учетной записи пользователя

`-I N` – установить количество дней неактивности N с момента устаревания пароля перед тем как учетная запись будет заблокирована

`-m N` – задает минимальное количество дней (N) между сменами пароля

`-M N` – задает максимальное количество дней (N) между сменами пароля

`-W N` – задает количество дней, за которые будет выдаваться предупреждение об устаревании пароля.

Права доступа

Распределение прав доступа в Linux. Чтение. Запись. Выполнение. Особенности прав у каталогов. Назначение прав доступа.Команды chmod, chown, chgrp. Sticky bit

Ход занятия

1. Для каждого объекта в файловой системе Linux существует набор прав доступа, определяющий взаимодействие пользователя с этим объектом. Такими объектами могут быть файлы, каталоги, а также специальные файлы (например, устройства) — то есть по сути любой объект файловой системы. Так как у каждого объекта в Linux имеется владелец, то права доступа применяются относительно владельца файла. Они состоят из набора 3 групп по три атрибута:

- чтение(`r`), запись(`w`), выполнение(`x`) для владельца;
- чтение, запись, выполнение для группы владельца;
- чтение, запись, выполнение для всех остальных.

Такие права можно представить краткой записью:

`rwxrwxrwx` – разрешено чтение, запись и выполнение для всех

`rwxr-xr-x` – запись разрешена только для владельца файла, а чтение и выполнение для всех.

`rw-rw-r--` – запись разрешена для владельца файла и группы владельца файла, а чтение – для всех.

Такое распределение прав позволяет гибко управлять ресурсами, доступными пользователям.

2. Права доступа распространяются и на каталоги. Они означают:

`r` – если установлено право на чтение из каталога, то можно увидеть его содержимое командой `ls` .

`w` – если установлено право записи в каталог, то пользователь может

создавать и удалять файлы из текущего каталога. Причем удалить файл из каталога пользователь может даже если у него нет прав на запись в файл. Есть возможность исправить эту ситуацию. Об этом я скажу позже.

`x` – если установлено право исполнения на каталог, то пользователь имеет право перейти в такой каталог командами наподобие `cd` .

Таким образом появляется возможность создания так называемых "скрытых" каталогов, когда невозможно получить список файлов, но пользователь точно знающий имя файла может скопировать его из "скрытого" каталога.

3. Для распределения прав доступа в Linux существует множество команд. Основные из них – это `chmod`, `chown` и `chgrp`.

Команда `chmod` (Change MODe – сменить режим) – изменяет права доступа к файлу. Для использования этой команды также необходимо иметь права владельца файла или права `root` . Синтаксис команды таков:

`chmod mode filename` , где

`filename` – имя файла, у которого изменяются права доступа;

`mode` – права доступа, устанавливаемые на файл. Права доступа можно записать в 2 вариантах – символьном и абсолютном.

В символьном виде использование команды `chmod` будет выглядеть следующим образом:

```
                |r|
       |u|      |w|
       |g|  |+|  |x|
chmod  |o|  |-|  |X|  filename,
       |a|  |=|  |u|
                |g|
                |o|
```

Рис. 4.1.

где:

`u,g,o,a` – установка прав для пользователя, группы, остальных пользователей, всех групп прав доступа соответственно.

`+,-,=` – добавить, удалить, установить разрешение соответственно.

`r,w,x,X,u,g,o` – право чтения, записи, выполнения, выполнения если есть такое право еще у какой либо из групп доступа, такие же как у владельца, такие же как у группы, такие же как у остальных пользователей.

`filename` - Имя файла, у которого изменяются права.

Просмотр разрешений, установленных на файл осуществляется командой `ls` с ключом `-l`:

```
[student@ns student]$ ls -l lesson5.txt
-rw-------  1  student student  39 Nov 19 15:17  lesson5.txt
[student@ns student]$ chmod g+rw lesson5.txt
[student@ns student]$ ls -l lesson5.txt
-rw-rw----  1  student student  39 Nov 19 15:18  lesson5.txt
[student@ns student]$ chmod o=u lesson5.txt
[student@ns student]$ ls -l lesson5.txt
-rw-rw-rw-  1  student student  39 Nov 19 15:18  lesson5.txt
[student@ns student]$ chmod o-w lesson5.txt
[student@ns student]$ ls -l lesson5.txt
-rw-rw-r--  1  student student  39 Nov 19 15:19  lesson5.txt
[student@ns student]$_
```

Для использования абсолютного режима необходимо представить права доступа к файлу в виде 3-х двоичных групп. Так например:

`rwx r-x r--` будет выглядеть как: `111 101 100`

Теперь каждую двоичную группу перевести в 8-ричное число: `111 – 7, 101 – 5, 100 – 4 .`

Чтобы задать файлу такие права необходимо выполнить команду:

```
[student@ns student]$ ls -l lesson5.txt
-rw-rw-r--  1  student student  39 Nov 19 15:19  lesson5.txt
[student@ns student]$chmod 754 lesson5.txt
[student@ns student]$ ls -l lesson5.txt
-rwxr-xr--  1  student student  39 Nov 19 15:19  lesson5.txt
[student@ns student]$_
```

Задание для обучаемых: попробовать изменить права файлу lesson5.txt и задать следующие: `r w x r- - r - - (744), r - - - w- - - x(421), - - x - w - r - -(124).`

Также предложить им проделать то же самое в символьном виде.

Команда `chown` (CHange OWNer – сменить владельца) – позволяет сменить владельца файла. Для использования этой команды необходимо либо иметь права владельца текущего файла или права `root` . Синтаксис команды прост:

`chown username:groupname filename` , где

`username` – имя пользователя – нового владельца файла;

`groupname` – имя группы – нового владельца файла;

`filename` – имя файла, у которого сменяется владелец.

Имя группы в синтаксисе команды можно не указывать, тогда будет изменен только владелец файла.

Команда `chgrp` используется для изменения владельца-группы файла. Синтаксис ее таков:

`chgrp groupname filename`,

где:

`groupname` – имя группы, которой будет принадлежать файл

`filename` – имя изменяемого файла

Имейте в виду, что использовать команды `chown` и `chmod` может только пользователь-владелец файла и `root`, а команду `chgrp` - пользователь-владелец файла, группа-владелец файла и `root`.

4. Существуют еще несколько особых прав, которые могут устанавливаться на файлы и каталоги. О некоторых из них мы поговорим при изучении темы "процессы". Но один рассмотрим сейчас. Это так называемый *`sticky`* `bit` (бит прикрепления).

В первых версиях Юникс этот бит использовался для того, чтобы заставить систему при работе программы оставлять образ ее кода в памяти. Тогда при следующем обращении к программе на ее запуск тратилось намного меньше времени так как чтение кода с устройства более не требовалось. Для файлов и сегодня в Linux осталось прежнее значение этого бита. А вот для каталогов этот атрибут приобрел новое значение. Если *`sticky`* `bit` установлен на каталог, то удалить файлы из такого каталога может только пользователь-владелец файла, и то только если у него есть право на запись в файл. Группа-владелец и остальные пользователи даже при наличии прав на запись в файл не смогут удалить его при установленном на каталог *`sticky`* `bit`.

Бит прикрепления устанавливается командой `chmod` в символьном виде:

```
chmod +t filename
```

Работа с файлами

Цель: Привить обучаемым твердые навыки по выводу информации из файлов на экран консоли. Вывод текстовой информации на экран. Команды cat, tac, more, less, head, tail, od. Программа просмотра справочного руководства man . Перенаправление вывода. Понятие stdin, stdout, stderr . Каналы. Операторы | и <, >, >> . Фильтрование информации. Регулярные выражения. Команда grep . Архивирование. Утилиты tar и gzip .

Ход занятия

1. Работа с файлами в Линукс является одним из базовых навыков. На этом занятии мы с вами узнаем способы вывода информации на экран из файлов. Оговорюсь сразу – сейчас я подразумеваю только обычные (регулярные) файлы.

Информация внутри них может содержаться в 2 видах – текстовом ASCII или бинарном. Для каждого из этих типов информации существуют программы, способные ее выводить на экран.

Для текстовых файлов наиболее часто используются команды `cat`, `tac`, `more`, `less`, `head` и `tail` .

Рассмотрим их поподробнее:

`cat filename` – команда используется для вывода текстовой информации из файла на экран. Например:

```
[student@localhost student]$ cat lesson6_st.txt
Это первый текстовый файл, который Вам удалось отобразить на экран
В нем содержится 12 строк текстовой информации.
Такой файл можно отобразить командой cat.
В обратном порядке такой файл можно отобразить командой tac.
Посчитать строки в нем можно командой nl.


This a first text file, where You see into your screen.
His lenght is 12 lines.
```

```
This file printing with command cat.
Inverse output make command tac.
Calculate lines in file with help command nl.
[student@localhost student]$_
```

Команда `cat` имеет несколько параметров. Опишем наиболее используемые из них:

`-b (--number-nonblank)` – пронумеровать все непустые строки;

`-n (--number)` – пронумеровать все строки;

`-s (--squeeze-blank)` – отобразить несколько подряд идущих пустых строк в виде одной пустой строки;

`-T (--show-tabs)` – показать символы табуляции, отобразив их как "^|";

`-E (--show-ends)` – показать символы конца строки как `$` .

Попробуем использовать некоторые из этих параметров:

```
[student@localhost student]$ cat -b lesson6_st.txt
 1  Это первый текстовый файл, который Вам удалось отобразить на экр
 2  В нем содержится 12 строк текстовой информации.
 3  Такой файл можно отобразить командой cat.
 4  В обратном порядке такой файл можно отобразить командой tac.
 5  Посчитать строки в нем можно командой nl.


 6  This a first text file, where You see into your screen.
 7  His lenght is 12 lines.
 8  This file printing with command cat.
 9  Inverse output make command tac.
 10  Calculate lines in file with help command nl.
[ student@localhost student]$ cat -n lesson6_st.txt
 1  Это первый текстовый файл, который Вам удалось отобразить на экр
 2  В нем содержится 12 строк текстовой информации.
 3  Такой файл можно отобразить командой cat.
 4  В обратном порядке такой файл можно отобразить командой tac.
```

```
 5  Посчитать строки в нем можно командой nl.
 6
 7
 8  This a first text file, where You see into your screen.
 9  His lenght is 12 lines.
 10  This file printing with command cat.
 11  Inverse output make command tac.
 12  Calculate lines in file with help command nl.
 [ student@localhost student]$ cat -E lesson6_st.txt
 Это первый текстовый файл, который Вам удалось отобразить на экран
 В нем содержится 12 строк текстовой информации.$
 Такой файл можно отобразить командой cat.$
 В обратном порядке такой файл можно отобразить командой tac.$
 Посчитать строки в нем можно командой nl.$
 $
 $
 This a first text file, where You see into your screen.$
 His lenght is 12 lines.$
 This file printing with command cat.$
 Inverse output make command tac.$
 Calculate lines in file with help command nl.$
 [student@localhost student]$_
```

tac `filename` – эта команда используется для вывода на экран информации из файла в обратном порядке.

```
 [student@localhost student]$ tac lesson6_st.txt
 Calculate lines in file with help command nl.
 Inverse output make command tac.
 This file printing with command cat.
 His lenght is 12 lines.
 This a first text file, where You see into your screen.


 Посчитать строки в нем можно командой  nl.
 В обратном порядке такой файл можно отобразить командой tac.
 Такой файл можно отобразить командой cat.
 В нем содержится 12 строк текстовой информации.
 Это первый текстовый файл, который Вам удалось отобразить на экран
```

```
[student@localhost student]$_
```

По умолчанию разделителем записей для этой команды является символ конца строки (`\n`). Но такое положение дел можно изменить, используя параметр

`-s (--separator=):`

```
[student@localhost student]$ tac --separator=" " lesson6_st.txt
nl.
command help with file in lines tac.
Calculate command make output cat.
Inverse command with printing file lines.
This 12 is lenght screen.
His your into see You where file, text first a nl.


This командой можно нем в строки tac.
Посчитать командой отобразить можно файл такой порядке обратном с
В командой отобразить можно файл информации.
Такой текстовой строк 12 содержится нем экране.
В на отобразить удалось Вам который файл, текстовый первый Это
[ student@localhost student]$
```

Но может возникнуть ситуация, когда размер файла намного превосходит количество строк, способных уместиться на экране. В таком случае при использовании утилиты `cat` вы не сможете комфортно прочесть файл, так как он будет очень быстро выведен. Справиться с этой ситуацией помогут нам команды `more` и `less`.

`more filename` – эта команда позволяет просматривать длинные файлы по частям. Так например:

```
[student@localhost student]$ more lesson6_bt.txt
KDE -- это интерактивная рабочая среда.
Другими словами, это набор программ,
технологий и документации, которые призваны
облегчить жизнь пользователей персональных
компьютеров. KDE нацелена на рабочие станции
```

```
под управлением Unix. Среди ее отличительных
особенностей -- "прозрачная" работа в сети и
современная философия работы.

Создатели этой рабочей среды - сообщество
разбросанных по всему миру программистов. Эти
люди - разработчики свободного программного
обеспечения - основной своей задачей считают
выпуск высококачественного программного продукта,
который позволит пользователю с легкостью
задействовать всю вычислительную мощь своего
компьютера.

KDE возникла как ответ на потребность в удобной в
использовании рабочей среде для рабочих станций под
Unix, аналогичной уже существующим системам на базе
MacOS или Windows. Основные инструменты для достижения
этой цели - это улучшенные средства межпрограммных
связей, повторное использование компонентов, технология
"drag and drop", единый внешний вид и многое другое. Таким
образом, KDE предлагает нечто большее, нежели
традиционные менеджеры окон Unix.

Стабильность, масштабируемость и открытость - вот те
--More--(56%)
```

Команда `more` использует для прокрутки две клавиши – пробел (показать следующий экран) и Enter (показать следующую строку). Но у `more` есть один недостаток – она способна прокручивать текст только вперед. То есть если вы уже смотрите второй экран, то к первому никак вернуться будет нельзя. Эту проблему с легкостью решает команда `less` .

`less filename` – позволяет просматривать файлы любой длины, прокручивая их в любую сторону. Например:

```
[student@localhost student]$ less lesson6_bt.txt
KDE -- это интерактивная рабочая среда.
Другими словами, это набор программ,
```

технологий и документации, которые призваны облегчить жизнь пользователей персональных компьютеров. KDE нацелена на рабочие станции под управлением Unix. Среди ее отличительных особенностей -- "прозрачная" работа в сети и современная философия работы.

Создатели этой рабочей среды - сообщество разбросанных по всему миру программистов. Эти люди - разработчики свободного программного обеспечения - основной своей задачей считают выпуск высококачественного программного продукта, который позволит пользователю с легкостью задействовать всю вычислительную мощь своего компьютера.

KDE возникла как ответ на потребность в удобной в использовании рабочей среде для рабочих станций под Unix, аналогичной уже существующим системам на базе MacOS или Windows. Основные инструменты для достижения этой цели - это улучшенные средства межпрограммных связей, повторное использование компонентов, технология "drag and drop", единый внешний вид и многое другое. Таким образом, KDE предлагает нечто большее, нежели традиционные менеджеры окон Unix.

Стабильность, масштабируемость и открытость - вот те качества,которые сделали Unix бесспорным выбором
lesson6_bt.txt lines 1-30/47 59%

Команда `less` понимает следующие комбинации клавиш:

Enter (стрелка вниз) – перейти на одну строку вниз;

Пробел (Page Down) – перейти на страницу вниз по тексту;

Page Up – перейти на страницу вверх по тексту;

стрелка вверх – перейти на строку вверх по тексту;

/ - поиск;

Home – перейти к началу текста;

End – перейти к концу текста;

q – выйти из программы `less` .

Но если нам не нужно отображать весь файл а только необходимо убедиться что это именно то, что мы ищем, то нам помогут команды `head` и `tail`.

`head filename1 [filename2 ...]` – по умолчанию выводит 10 первых строк файла. Команде `head` можно задавать параметры, вот самые используемые из них:

`-n lines` – вывести не 10, а `lines` строк от начала файла;

`-c bytes` – -вывести `bytes` байт с начала файла;

- `q` – не печатать заголовки файлов перед выводом (при выводе нескольких файлов сразу).

```
[student@localhost student]$ head lesson6_bt.txt
KDE -- это интерактивная рабочая среда.
Другими словами, это набор программ,
технологий и документации, которые призваны
облегчить жизнь пользователей персональных
компьютеров. KDE нацелена на рабочие станции
под управлением Unix. Среди ее отличительных
особенностей -- "прозрачная" работа в сети и
современная философия работы.

Создатели этой рабочей среды - сообщество
[student@localhost student]$ head -n 2 lesson6_bt.txt
KDE -- это интерактивная рабочая среда.
Другими словами, это набор программ,
[student@localhost student]$_
```

`tail filename` – по умолчанию выводит 10 последних строк

файла. Вот наиболее используемые ее параметры:

`-n lines` – вывести не 10, а `lines` последних строк;

`-c bytes` – вывести `bytes` последних байт файла;

`-f` – войти в постоянный цикл по считыванию конца файла. Таким образом при поступлении в файл новой информации пользователь может вести мониторинг ее в реальном времени. Выход из этого режима осуществляется комбинацией клавиш <Ctrl>+<C> .

```
[student@localhost student]$ tail lesson6_bt.txt
серверных применений и в научных учреждениях, но не привлекала
обычных пользователей.

Без Unix не было бы и Интернета, по крайней мере, в нашем
теперешнем представлении. Однако до недавнего времени Unix
не отвечал запросам рядового пользователя. Этот факт особенно
неприятен, поскольку существуют такие варианты Unix, как Linux и
FreeBSD, NetBSD и т.д., которые свободно доступны в Интернете и
славятся исключительным качеством и стабильностью.

[student@localhost student]$ tail -c 100 lesson6_bt.txt
 т.д., которые свободно доступны в Интернете и
славятся исключительным качеством и стабильностью.

[student@localhost student]$
```

Вернемся к началу нашего занятия и вспомним что есть еще и второй тип содержимого файлов – бинарные данные. Для просмотра такого типа файлов нам не подойдут команды типа `more` или `cat` :

```
RIFFWAVEfmt
D Ddata~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~
[student@localhost student]$_
```

Рис. 5.1.

Но вот команда `od` предназначена как раз для просмотра файлов такого типа.

`od filename` – просматривать бинарные файлы в виде дампа байтов. Наиболее популярные параметры команды `od`:

`-A radix` – указать систему счисления, в которой будут указываться адреса. Параметр `radix` может принимать следующие значения:

`d` – десятичная, `o` – восьмеричная (по умолчанию),

`x` – шестнадцатиричная, `n` – не выводить адресов.

`-t radix` – указать систему счисления, в которой будут указываться данные. Параметр `-t radix` может принимать следующие значения: `d` – десятичная, `o` – восьмеричная (по умолчанию), `x` – шестнадцатиричная, `n` – не выводить адресов.

`-j bytes` – пропустить `bytes` байт от начала файла.

`-v` – включить вывод последовательных одинаковых строк. По умолчанию такой вывод отключён.

`od` выводит только первую строку из множества, а вместо остальных – символ `"*"` .

```
[student@localhost student]$ od -x -j 100 lesson6.bin
0000144 8080 8080 8080 8080 8080 8080 8080 8080
*
0000224 8080 7f80 7f7f 807f 7f80 7f7f 7f7f 807f
0000244 8080 8080 7f80 7f7f 7f7f 7f7f 7f7f 7f7f
0000264 7f7f 7f7f 807f 7f80 7f7f 7f7f 7f7f 7f7f
0000304 7f7f 7f7f 807f 8080 8080 8080 8080 7f80
0000324 7f7f 7f7f 7f7f 7f7f 807f 8080 8080 7f80
0000344 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f
*
0000404 7f7f 7f7f 7f7f 7f7f 7f7f 807f 8080 8080
0000424 7f80 7f7f 7f7f 7f7f 807f 8080 7f80 7f7f
0000444 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f
```

```
*
0000504 7f7f 7f7f 807f 8080 7f80 7f7f 7f7f 7f7f
0000664 7e7e 7e7e 7e7e 7f7e 7f7f 7f7f 7f7f 7f7f
0000704 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7e7f 7e7e
0000724 7e7e 7f7e 7f7f 7f7f 7f7f 7f7f 7e7f 7e7e
0000744 7e7e 7e7e 7e7e 7f7e 7f7f 7f7f 7f7f 7f7f
0000764 7e7f 7e7e 7e7e 7e7e 7e7e 7e7e 7f7e 7f7f
0001004 7f7f 7f7f 7f7f 7f7f 7e7f 7e7e 7e7e 0a7e
0001024
[student@localhost student]$ od -x -j 100 -v lesson6.bin
0000144 8080 8080 8080 8080 8080 8080 8080 8080
0000164 8080 8080 8080 8080 8080 8080 8080 8080
0000204 8080 8080 8080 8080 8080 8080 8080 8080
0000224 8080 7f80 7f7f 807f 7f80 7f7f 7f7f 807f
0000244 8080 8080 7f80 7f7f 7f7f 7f7f 7f7f 7f7f
0000264 7f7f 7f7f 807f 7f80 7f7f 7f7f 7f7f 7f7f
0000304 7f7f 7f7f 807f 8080 8080 8080 8080 7f80
0000324 7f7f 7f7f 7f7f 7f7f 807f 8080 8080 7f80
0000344 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f
0000364 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f
0000404 7f7f 7f7f 7f7f 7f7f 7f7f 807f 8080 8080
0000424 7f80 7f7f 7f7f 7f7f 807f 8080 7f80 7f7f
0000444 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f
0000464 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f
0000504 7f7f 7f7f 807f 8080 7f80 7f7f 7f7f 7f7f
0000524 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f
0000544 7f7f 7f7f 7f7f 7e7f 7e7e 7e7e 7e7e 7e7e
0000564 7e7e 7e7e 7e7e 7f7e 7f7f 7f7f 7f7f 7f7f
0000604 7f7f 7e7f 7e7e 7e7e 7e7e 7e7e 7e7e 7e7e
0000624 7f7e 7f7f 7f7f 7f7f 7f7f 7f7f 7e7f 7e7e
0000644 7e7e 7e7e 7f7e 7f7f 7e7f 7e7e 7e7e 7e7e
0000664 7e7e 7e7e 7e7e 7f7e 7f7f 7f7f 7f7f 7f7f
0000704 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7e7f 7e7e
0000724 7e7e 7f7e 7f7f 7f7f 7f7f 7f7f 7e7f 7e7e
0000744 7e7e 7e7e 7e7e 7f7e 7f7f 7f7f 7f7f 7f7f
0000764 7e7f 7e7e 7e7e 7e7e 7e7e 7e7e 7f7e 7f7f
0001004 7f7f 7f7f 7f7f 7f7f 7e7f 7e7e 7e7e 0a7e
0001024
[student@localhost student]$
```

2. Если вам понадобилось в Линукс получить справку по какой либо команде, то стоит воспользоваться справочной системой `man`. В Линукс при выводе справки `man` ведет себя аналогично `less`.

```
man command
```

`man` поддерживает несколько параметров. Наиболее часто используются следующие:

`-k keyword` – показывает все страницы руководства, на которых встречается слово `keyword`;

`<число>` - указывает брать руководство из раздела с номером `<число>`

Имеются следующие разделы `man`:

Таблица 5.1.

Номер	Значение
1	Команды и приложения пользовательского уровня
2	Системные вызовы и коды ошибок ядра
3	Библиотечные функции
4	Драйверы устройств и сетевые протоколы
5	Стандартные форматы файлов
6	Игры и демонстрационные программы
7	Различные файлы и документы
8	Команды системного администрирования
9	Внутренние интерфейсы и спецификации ядра

Подробное руководство по команде `man` можно получить используя команду `man man` .

3. Во многих случаях при работе с файлами нам необходимо будет использовать вывод команды. Решением проблемы перенаправления вывода занимаются командные оболочки, которые мы с вами еще обсудим на занятиях.

В работе с командной строкой Linux есть понятия стандартных устройств ввода, вывода и вывода ошибок.

`stdin` – стандартное устройство ввода. Имеет файловый указатель №0. Автоматически открывается всеми процессами.

`stdout` – стандартное устройство вывода. Имеет файловый указатель №1. Автоматически открывается всеми процессами.

`stderr` – стандартный поток ошибок (специальное устройство вывода для сообщений об ошибках. Имеет файловый указатель №2. Автоматически открывается всеми процессами.

По умолчанию практически все команды Linux используют для ввода информации `stdin` , а для вывода `stdout` и `stderr`,если их параметрами не указано обратное.

Операторы перенаправления способны изменять направление вывода и ввода информации. Так оператор:

`>` - перенаправляет стандартный поток в файл (другой поток). При этом если файл существует, то он перезаписывается, если не существует – создается.

`>>` - перенаправляет стандартный поток в файл. При этом если файл существует, то информация добавляется в конец, если не существует – файл создается.

`<` - перенаправляет содержимое указанного файла на стандартный ввод программе.

`>&` - перенаправляет стандартные потоки вывода и ошибок друг в друга.

Пример1:

```
[gserg@WEBMEDIA Занятие 7]$ ls
lesson7_1.txt
[gserg@WEBMEDIA Занятие 7]$ head -n 5 lesson7_1.txt > fivelines.txt
[gserg@WEBMEDIA Занятие 7]$ ls
```

```
fivelines.txt  lesson7_1.txt
[gserg@WEBMEDIA Занятие 7]$ cat fivelines.txt
В сочетании со свободными версиями Unix, KDE дарит
миру открытую и абсолютно свободную от всех
ограничений рабочую среду для домашнего или
профессионального применения.
Эта платформа доступна всем желающим бесплатно,
[gserg@WEBMEDIA Занятие 7]$_
```

Пример2:

```
[gserg@WEBMEDIA Занятие 7]$ tail -n 1 fivelines.txt >> fivelines.txt
[gserg@WEBMEDIA Занятие 7]$ cat fivelines.txt
В сочетании со свободными версиями Unix, KDE дарит
миру открытую и абсолютно свободную от всех
ограничений рабочую среду для домашнего или
профессионального применения.
Эта платформа доступна всем желающим бесплатно,
Эта платформа доступна всем желающим бесплатно,
[gserg@WEBMEDIA Занятие 7]$
```

Пример3:

```
[gserg@WEBMEDIA Занятие 7]$ cat < fivelines.txt
В сочетании со свободными версиями Unix, KDE дарит
миру открытую и абсолютно свободную от всех
ограничений рабочую среду для домашнего или
профессионального применения.
Эта платформа доступна всем желающим бесплатно,
Эта платформа доступна всем желающим бесплатно,
[gserg@WEBMEDIA Занятие 7]$_
```

Пример4:

```
[gserg@WEBMEDIA Занятие 7]$ cat nofile.txt > result.out 2>&1
[gserg@WEBMEDIA Занятие 7]$ cat result.out
cat: nofile.txt: No such file or directory
[gserg@WEBMEDIA Занятие 7]$_
```

Такой способ перенаправления вывода хорош при работе с файлами. Но

как быть, если вывод какой-либо программы нам необходимо перенаправить на вход другой? Для этого в Linux существует такое понятие как каналы.

Канал – программный интерфейс, позволяющий процессам обмениваться данными (односторонний поток).

Организацией канала занимается `shell` . Для управления каналом существует оператор `|` . Приведу пример:

Пример5:

```
[gserg@WEBMEDIA Занятие 7]$ cat lesson7_1.txt | tail -n 3 | less
в течение многих лет с удовольствием используют ученые и
профессионалы-компьютерщики во всем мире.

lines 1-3/3 (END)
```

Рассмотрим поподробнее все, что произошло при выполнении данной нами группы команд:

- Команда `cat` прочитала файл lesson7_1.txt и передала его содержание на стандартный ввод команды `tail`
- Команда `tail` исходя из заданных ей параметров взяла 3 последние строки стандартного потока ввода и передала их на стандартный ввод команде `less`
- Команда `less` вывела информацию со стандартного ввода на экран и стала ожидать действий пользователя.

Таким образом одна команда передавала по каналу информацию другой команде.

4. Иногда необходимо вывести информацию, содержание которой вы знаете, а вот расположение – нет. Именно для таких случаев существуют регулярные выражения. Перечислю наиболее используемые из них, хотя оговорюсь сразу, что на самом деле регулярных выражений намного больше.

Регулярное выражение – средство указания шаблона для поиска его в

тексте.

^ - начало строки

$ - конец строки

[] - любой символ из заключённых в скобки. Поддерживает диапазоны, [0-9] – цифры, [a-zA-Z] -все буквы латинского алфавита

[^] - любой символ за исключением заключенных в скобки

\ - отменяет действие любого метасимвола. Например

\$ - обозначает символ $, а не \ в конце строки, а \\$ - символ \ в конце строки

.- любой один символ.

* - 0 или более раз в тексте встречается предыдущий шаблон. Так например выражение .* означает любой набор символов.

Регулярные выражения поддерживаются практически всеми текстовыми редакторами Linux. Существует также программа фильтрации текста grep . Она также использует регулярные выражения. Её мы с Вами и рассмотрим.

grep *regexp* file – утилита фильтрации текста. Ищет в файле file строки, в которых встречается выражение, соответствующее шаблону *regexp* и выводит их на стандартный вывод.

Пример6:

```
[gserg@WEBMEDIA Занятие 7]$ grep KDE lesson7_1.txt
В сочетании со свободными версиями Unix, KDE дарит
что комбинация Unix и KDE наконец подарит пользователю
[gserg@WEBMEDIA Занятие 7]$_
```

Пример7:

```
[gserg@WEBMEDIA Занятие 7]$ grep ^Э.* lesson7_1.txt
Эта платформа доступна всем желающим бесплатно,
[gserg@WEBMEDIA Занятие 7]$_
```

Как правило, утилиту `grep` используют не только для фильтрации текстовых файлов, но и, например, для фильтрации вывода каких-либо команд.

В примере ниже мы с вами попытаемся найти все файлы, начинающиеся на букву `f` в каталоге bin :

Пример8:

```
[gserg@WEBMEDIA Занятие 7]$ ls /bin | grep ^f.*
false*
fbresolution*
fgrep@
find*
[gserg@WEBMEDIA Занятие 7]$_
```

Подробнее о команде `grep` можно узнать из страницы справочного руководства `man (man grep)`.

5. Как известно, при работе в любой операционной системе часто возникает необходимость создавать резервные копии важной информации. Для такой работы необходимы программы-архиваторы и программы сжатия.

В комплект поставки ОС Linux входят как правило сразу несколько программ архивирования и/или сжатия. Но стандартом de facto для unix-like ОС стали архиватор `tar` и программа сжатия `gzip` (GNU zip).

`tar (GNU tar – GNU tape archiver)` – программа для создания архивов. Современный `tar` поддерживает сжатие, но для обеспечения совместимости с более ранними версиями Linux и Unix советуется использовать несжатый архив `tar` , либо сжимать его после создания одной из утилит сжатия.

`tar` имеет множество параметров и опций. Мы с вами рассмотрим наиболее употребительные из них:

`-c` – создать архив

`-r` – добавить файлы в архив

`-A` – добавить содержимое tar-файлов в архив

`--delete` – удалить файлы из архива (невозможно использование на архивных лентах)

`-t` – вывести список файлов в архиве

`-x` – извлечь файлы из архива

`-f` – информация будет извлекаться из файла

`-v` – расширенный вывод информации о выполняемых действиях

Пример9:

```
[gserg@WEBMEDIA Занятие 7]$ tar -xvf lesson_2.tar
./lesson_2/
./lesson_2/.quota
./lesson_2/arch
./lesson_2/chain.b
./lesson_2/ida/
./lesson_2/hda1
tar: ./lesson_2/hda1: Cannot mknod: Operation not permitted
tar: Выход, отложенный по результатам предыдущих ошибок
[gserg@WEBMEDIA Занятие 7]$_
```

Пример10:

```
[gserg@WEBMEDIA Занятие 7]$ tar -tf lesson_2.tar
./lesson_2/
./lesson_2/.quota
./lesson_2/arch
./lesson_2/chain.b
./lesson_2/ida/
./lesson_2/hda1
[gserg@WEBMEDIA Занятие 7]$_
```

Пример11:

```
[gserg@WEBMEDIA Занятие 7]$ ls
lesson_2/  lesson_2.tar  lesson7_1.txt
[gserg@WEBMEDIA Занятие 7]$ tar -cf lesson7.tar lesson7_1.txt
[gserg@WEBMEDIA Занятие 7]$ ls
lesson_2/  lesson_2.tar  lesson7_1.txt  lesson7.tar
[gserg@WEBMEDIA Занятие 7]$ tar -tf lesson7.tar
lesson7_1.txt
[gserg@WEBMEDIA Занятие 7]$_
```

`gzip options filename` – утилита компрессии (сжатия) файлов. Производит сжатие|декомпрессию файла filename или другие сопутствующие действия в соответствии с опциями options :

только имя файла – сжимает этот файл.

`-d` – декомпрессия файла

`-t` – проверяет целостность архива

`-v` – расширенный вывод информации о выполняемых действиях

Пример12:

```
[gserg@WEBMEDIA Занятие 7]$ ls
lesson_2.tar  lesson7_1.txt  pict.png.gz
[gserg@WEBMEDIA Занятие 7]$ gzip -dv pict.png.gz
pict.png.gz:         0.3% -- replaced with pict.png
[gserg@WEBMEDIA Занятие 7]$ ls
lesson_2.tar  lesson7_1.txt  pict.png
[gserg@WEBMEDIA Занятие 7]$_
```

Пример13:

```
[gserg@WEBMEDIA Занятие 7]$ ls
lesson_2.tar  lesson7_1.txt  pict.png
[gserg@WEBMEDIA Занятие 7]$ gzip pict.png
[gserg@WEBMEDIA Занятие 7]$ ls
lesson_2.tar  lesson7_1.txt  pict.png.gz
```

```
[gserg@WEBMEDIA Занятие 7]$_
```

Пример14:

```
[gserg@WEBMEDIA Занятие 7]$ gzip -tv pict.png.gz
pict.png.gz:            OK
[gserg@WEBMEDIA Занятие 7]$_
```

Для создания сжатого архива из множества файлов используйте сразу обе эти утилиты путем передачи информации каналами или последовательно запуская их из командной строки.

Процессы

Процессы в Linux. Идентификаторы процессов. Демоны. Команда ps . Права доступа процессов. Реальный и эффективный идентификаторы. Биты SUID и SGID. Управление процессами. Сигналы. Команды nice, nohup, kill, killall.

Ход занятия

1. При рассмотрении предыдущих тем мы с вами часто употребляли определение "процесс". Но что такое процесс?

Процесс – понятие совокупности программного кода и данных, загруженных в память ЭВМ. Процесс - это не запущенная программа (приложение) или команда, так как приложение может создавать несколько процессов одновременно. Код процесса не обязательно должен выполняться в текущий момент времени, так как процесс может находиться в состоянии спящего. В этом случае выполнение кода такого процесса приостановлено. Существует всего 3 состояния, в которых может находиться процесс:

Работающий процесс – в данный момент код этого процесса выполняется.

Спящий процесс – в данный момент код процесса не выполняется в ожидании какого либо события (нажатия клавиши на клавиатуре, поступление данных из сети и т.д.)

Процесс-зомби – сам процесс уже не существует, его код и данные выгружены из оперативной памяти, но запись в таблице процессов остается по тем или иным причинам.

Каждому процессу в системе назначаются числовые идентификаторы (личные номера) в диапазоне от 1 до 65535 (PID – Process Identifier) и идентификаторы родительского процесса (PPID – Parent Process Identifier). PID является именем процесса, по которому мы можем адресовать процесс в операционной системе при использовании различных средств просмотра и управления процессами. PPID определяет родственные отношения между процессами, которые в

значительной степени определяют его свойства и возможности. Другие параметры, которые необходимы для работы программы, называют "окружение процесса". Один из таких параметров – управляющий терминал – имеют далеко не все процессы. Процессы, не привязанные к какому-то конкретному терминалу называются "демонами" (daemons). Такие процессы, будучи запущенными пользователем, не завершают свою работу по окончании сеанса, а продолжают работать, т.к. они не связаны никак с текущим сеансом и не могут быть автоматически завершены. Как правило, с помощью демонов реализуются серверные службы, так например сервер печати реализован процессом-демоном cupsd , а сервер журналирования – syslogd .

2. Для просмотра списка процессов в Linux существует команда `ps` .

`ps options [PID]` – просмотр списка процессов. Без параметров `ps` показывает все процессы, которые были запущены в течение текущей сессии, за исключением демонов. Для `Options` рассмотрим следующие значения или их комбинации:

`-A` или `-e` – показать все процессы

`-f` – полноформатный вывод

`w` – показать полные строки описания процессов. Если они превосходят длину экрана, то перенести описание на следующую строку. Параметр `-ww` позволить убрать вообще все ограничения на длину отображаемой строки и делать вывод в 2 и более строк при необходимости.

Пример1:

```
[gserg@WEBMEDIA gserg]$ ps
  PID TTY          TIME CMD
 3126 pts/2    00:00:00 bash
 3158 pts/2    00:00:00 ps
[gserg@WEBMEDIA gserg]$_
```

Пример2:

```
[gserg@WEBMEDIA gserg]$ ps 3126
```

```
  PID TTY      STAT   TIME COMMAND
 3126 pts/2    S      0:00 /bin/bash
[gserg@WEBMEDIA gserg]$_
```

Пример3:

```
[gserg@WEBMEDIA gserg]$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 10:01 ?        00:00:03 init [5]
root         2     1  0 10:01 ?        00:00:00 [keventd]
root         3     1  0 10:01 ?        00:00:00 [kapmd]
root         4     1  0 10:01 ?        00:00:00 [ksoftirqd_CPU0]
root         5     1  0 10:01 ?        00:00:24 [kswapd]
root         6     1  0 10:01 ?        00:00:00 [bdflush]
...
gserg     3126  3124  0 17:56 pts/2    00:00:00 /bin/bash
gserg     3160  3126  0 17:59 pts/2    00:00:00 ps -ef
[gserg@WEBMEDIA gserg]$_
```

Пример4:

```
[gserg@WEBMEDIA gserg]$ ps -efw
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 10:01 ?        00:00:03 init [5]
root         2     1  0 10:01 ?        00:00:00 [keventd]
root         3     1  0 10:01 ?        00:00:00 [kapmd]
root         4     1  0 10:01 ?        00:00:00 [ksoftirqd_CPU0]
root         5     1  0 10:01 ?        00:00:24 [kswapd]
root         6     1  0 10:01 ?        00:00:00 [bdflush]
root         7     1  0 10:01 ?        00:00:00 [kupdated]
root         8     1  0 10:01 ?        00:00:00 [mdrecoveryd]
root        12     1  0 10:01 ?        00:00:00 [kjournald]
root       115     1  0 10:01 ?        00:00:00 devfsd /dev
root       211     1  0 10:01 ?        00:00:00 [khubd]
root       334     1  0 10:01 ?        00:00:00 [kjournald]
root       594     1  0 10:01 ?        00:00:00 [eth0]
root       730     1  0 10:01 ?        00:00:00 /sbin/dhcpcd -h WEBMEDIA -Y -N
root       772     1  0 10:02 ?        00:00:00 /sbin/dhcpcd -h WEBMEDIA -Y -N
rpc        820     1  0 10:02 ?        00:00:00 portmap
```

```
root      836     1  0 10:02 ?        00:00:00 syslogd -m 0
root      844     1  0 10:02 ?        00:00:00 klogd -2
root      879     1  0 10:02 ?        00:00:00 gpm -t ps/2 -m /dev/psaux
xfs      1074     1  0 10:02 ?        00:00:07 xfs -port -1 -daemon -droppriv -user
root     1130     1  0 10:02 ?        00:00:00 /usr/sbin/apmd -p 10 -w 5 -W -P /et
......
gserg    3122  2072  0 17:56 ?        00:00:00 xmms
gserg    3123  2072  1 17:56 ?        00:00:03 xmms
gserg    3124  1914  0 17:56 ?        00:00:02 kdeinit: konsole -icon konsole.png
gserg    3126  3124  0 17:56 pts/2    00:00:00 /bin/bash
gserg    3172  3126  0 18:01 pts/2    00:00:00 ps -efw
[gserg@WEBMEDIA gserg]$_
```

3. Процессы в ОС Linux обладают теми же правами, которыми обладает пользователь, от чьего имени был запущен процесс.

Для определения имени пользователя, запустившего процесс, операционная система использует реальные идентификаторы пользователя и группы, назначаемые процессу. Но эти идентификаторы не являются решающими при определении прав доступа. Для этого у каждого процесса существует другая группа идентификаторов – эффективные.

Как правило, реальные и эффективные идентификаторы процессов одинаковые, но есть и исключения. Например, для работы утилиты `passwd` необходимо использовать идентификатор суперпользователя, так как только суперпользователь имеет права на запись в файлы паролей. В этом случае эффективные идентификаторы процесса будут отличаться от реальных. Возникает резонный вопрос – как это было реализовано?

У каждого файла есть еще один набор прав доступа – биты `SUID` и `SGID` . Эти биты позволяют при запуске программы присвоить ей эффективные идентификаторы владельца и группы-владельца соответственно и выполнять процесс с правами доступа другого пользователя. Так как файл `passwd` принадлежит пользователю `root` и у него установлен бит `SUID` , то при запуске процесс passwd будет обладать правами пользователя `root` .

Устанавливаются биты SGID и SUID программой chmod:

chmod u+s filename – установка бита SUID

chmod g+s filename – установка бита SGID

Для установки этих битов в абсолютном режиме их стоит представить в виде трех бит: SUID , SGID , *Sticky* bit соответственно.

После выставления необходимых прав добавьте в начало числа цифру для установки специальных бит:

Пример5:

```
[gserg@WEBMEDIA gserg]$chmod 7777 filename
[gserg@WEBMEDIA gserg]$ls filename
-rwsrwsrwt  1 gserg  gserg   23811 Aug 29 11:00 filename
[gserg@WEBMEDIA gserg]$
```

4. Мы с вами рассмотрели понятие процесса, способы отображения процессов и права доступа. Но для комфортной работы в операционной системе этого, согласитесь, мало. Необходимо еще эффективно управлять процессами. Но для реализации управления мы сначала рассмотрим строение таблицы процессов:

Родителем всех процессов в системе является процесс init . Его PID всегда 1 , *PPID* – 0 . Всю таблицу процессов можно представить себе в виде дерева, в котором корнем будет процесс init . Этот процесс хоть и не является частью ядра, но выполняет в системе очень важную роль, о которой мы с вами поговорим на 16-ом занятии.

Процессы, имена которых заключены в квадратные скобки, например " [keventd]" - это процессы ядра. Эти процессы управляют работой системы, а точнее такими ее частями, как менеджер памяти, планировщик времени процессора, менеджеры внешних устройств и так далее.

Остальные процессы являются пользовательскими, запущенными либо из командной строки, либо во время инициализации системы.

Жизнь каждого процесса представлена следующими фазами:

Создание процесса – на этом этапе создается полная копия того процесса, который создает новый. Например, вы запустили из интерпретатора на выполнение команду `ls` . Командный интерпретатор создает свою полную копию.

Загрузка кода процесса и подготовка к запуску – копия, созданная на первом этапе заменяется кодом задачи, которую необходимо выполнить и создается ее окружение – устанавливаются необходимые переменные и т.п.

Выполнение процесса

Состояние зомби – на этом этапе выполнение процесса закончилось, его код выгружается из памяти, окружение уничтожается, но запись в таблице процессов еще остается.

Умирание процесса – после всех завершающих стадий удаляется запись из таблицы процессов – процесс завершил свою работу.

Во время работы процесса, ядро контролирует его состояние, и в случае возникновения непредвиденной ситуации управляет процессом с помощью посылки ему сигнала. Процесс может воспользоваться действием по умолчанию, или, если у него есть обработчик сигнала, то он может перехватить или игнорировать сигнал. Сигналы `SIGKILL` и `SIGSTOP` невозможно ни перехватить, ни игнорировать.

По умолчанию возможны несколько действий:

игнорировать – продолжать работу, несмотря на то, что получен сигнал.

завершить – завершить работу процесса.

завершить + core – завершить работу процесса и создать файл в текущем каталоге с именем core, содержащий образ памяти процесса (код и данные).

остановить – приостановить выполнение процесса, но не завершать его работу и не выгружать код из памяти.

Вот список всех сигналов, существующих в системе:

Таблица 1.1.

Название	Действие по умолчанию	Значение
SIGABRT	Завершить + core	Сигнал отправляется, если процесс вызывает системный вызов abort()
SIGALRM	Завершить	Сигнал отправляется, когда срабатывает таймер, ранее установленный.
SIGBUS	Завершить + core	Сигнал свидетельствует о некоторой аппаратной ошибке. Обычно этот сигнал отправляется при обращении к недопустимому виртуальному адресу, для которого отсутствует соответствующая физическая страница.
SIGCHLD	Игнорировать	Сигнал, посылаемый родительскому процессу при завершении его потомка.
SIGSEGV	Завершить + core	Сигнал свидетельствует об обращении процесса к недопустимому адресу или области памяти, для которой у процесса недостаточно привилегий доступа.
SIGFPE	Завершить + core	Сигнал свидетельствует о возникновении особых ситуаций, таких как деление на 0 или переполнение операции с плавающей точкой.
SIGHUP	Завершить	Сигнал посылается лидеру сеанса, связанному с управляющим терминалом, что терминал отсоединился (потеря линии). Сигнал также посылается всем процессам текущей группы при завершении выполнения лидера. Этот сигнал иногда используют в качестве простейшего средства межпроцессного взаимодействия. В частности, он применяется для сообщения демонам о необходимости обновить конфигурационную информацию. Причина выбора именно сигнала SIGHUP

		заключается в том, что демон по определению не имеет управляющего терминала и, соответственно, обычно не получает этого сигнала.
SIGILL	Завершить + core	Сигнал посылается ядром, если процесс попытается выполнить недопустимую инструкцию.
SIGINT	Завершить	Сигнал посылается ядром всем процессам при нажатии клавиши прерывания (<CTRL>+<C>)
SIGKILL	Завершить	Сигнал, при получении которого выполнение процесса прекращается. Этот сигнал нельзя ни перехватить, ни проигнорировать.
SIGPIPE	Завершить	Сигнал посылается при попытке записи в сокет, получатель данных которого завершил выполнение или закрыл файловый указатель на сокет.
SIGPOLL	Завершить	Сигнал отправляется при наступлении определенного события для устройства, которое является опрашиваемым (например, получен пакет по сети)
SIGPWR	Игнорировать	Сигнал генерируется при угрозе потери питания. Обычно он отправляется, когда питание системы переключается на источник бесперебойного питания (UPS).
SIGQUIT	Завершить	Сигнал посылается всем процессам текущей группы при нажатии клавиш <CTRL>+<\> .
SIGSTOP	Остановить	Сигнал отправляется всем процессам текущей группы при нажатии пользователем клавиш <CTRL>+<Z> . Получение сигнала вызывает останов выполнения процесса.
SIGSYS	Завершить + core	Сигнал отправляется ядром при попытке осуществления процессом недопустимого системного вызова.
		Сигнал обычно представляет своего рода

SIGTERM	Завершить	предупреждение, что процесс вскоре будет уничтожен. Этот сигнал позволяет процессу соответствующим образом "подготовиться к смерти" - удалить временные файлы, завершить необходимые транзакции и т.д. Команда kill по умолчанию отправляет именно этот сигнал.
SIGTTIN	Остановить	Сигнал генерируется ядром (драйвером управляющего терминала) при попытке процесса фоновой группы осуществить чтение с управляющего терминала.
SIGTTOU	Остановить	Сигнал генерируется ядром (драйвером терминала) при попытке процесса фоновой группы осуществить запись на управляющий терминал.
SIGUSR1	Завершить	Сигнал предназначен для прикладных задач как простейшее средство межпроцессного взаимодействия.
SIGUSR2	Завершить	Сигнал предназначен для прикладных задач как простейшее средство межпроцессного взаимодействия.

Немаловажную роль в жизни процессов играет также планировщик – это часть ядра, ответственная за многозадачность системы. Ведь в единицу времени на одном процессоре может выполняться только одна задача. Именно планировщик определяет, какой из запущенных процессов первым будет выполняться, какой вторым. Для этого у каждого процесса существует еще один параметр, называемый приоритетом. Для того, чтобы посмотреть приоритет процессов, нам необходимо использовать уже знакомую команду ps с параметром -l (long – расширенный вывод):

```
[gserg@WebMedia gserg]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR    SZ WCHAN  TTY          TIME C
0 S   500  1554  1553  0  75   0    -  1135 wait4  pts/1    00:00:00 bash
0 R   500  1648  1554  0  81   0    -   794 -      pts/1    00:00:00 ps
[gserg@WebMedia gserg]$
```

Во время своей работы, планировщик в первую очередь ставит на выполнение задачи с меньшим приоритетом. Так, приоритетом `0` , обладают только критические системные задачи, а отрицательным приоритетом – процессы ядра. Задачам с большим приоритетом достается меньше процессорного времени и потому, работают они как правило, медленнее, и потребляют намного меньше системных ресурсов.

5. Остается только решить вопрос, а может ли пользователь управлять процессами и системными параметрами? Конечно может! Для этого в Linux есть набор инструментов, позволяющих изменять приоритет процесса, посылать процессам сигналы. О них мы с вами сейчас и поговорим.

`nice -n command` - позволяет изменять приоритет, с которым будет выполняться процесс после запуска. Без указания команды `command` выдает текущий приоритет работы. `n` по умолчанию равен 10. Диапазон приоритетов расположен от -20 (наивысший приоритет) до 19 (наименьший).

```
[gserg@WebMedia gserg]$ less .bashrc &
[1] 3070
[gserg@WebMedia gserg]$ ps -efl | grep less
0 T gserg  3070 3018 0 80 0 - 1004 finish 17:56 pts/3    00:00:00 less .bashrc
0 S gserg  3072 3018 0 80 0 - 949  pipe_w 17:57 pts/3    00:00:00 grep less
[gserg@WebMedia gserg]$ nice -n 20 less .bashrc &
[1] 3081
[gserg@WebMedia gserg]$ ps -efl | grep less
0 T gserg  3081 3018 0 99 19 - 1003 finish 18:01 pts/3   00:00:00 less .bashrc
0 S gserg  3083 3018 0 81 0  - 950  pipe_w 18:01 pts/3   00:00:00 grep less
[gserg@WebMedia gserg]$
```

Сравнивая цифры приоритета, заметим, что команда `less` в первом случае выполнялась с приоритетом `80` , а во втором – `99` . Таким образом, команда `nice` сделала свое дело – понизила приоритет задачи.

`nohup command` – позволяет процессу продолжить выполнение даже при потере управляющего терминала (SIGHUP). Эту команду

выгодно использовать когда необходимо выполнить команду продолжительного действия. Вы запускаете команду и закрываете терминальный сеанс, а она при этом продолжает выполняться.

`kill -SIGKILL pid` – посылает сигнал процессу с идентификатором `pid` . Если сигнал не указан, команда посылает процессу сигнал `SIGTERM` .

```
[gserg@WebMedia gserg]$ less &
[1] 1352
[gserg@WebMedia gserg]$ ps
  PID TTY          TIME CMD
 1322 pts/2    00:00:00 bash
 1352 pts/2    00:00:00 less
 1353 pts/2    00:00:00 ps
[gserg@WebMedia gserg]$ kill -SIGKILL 1352
[gserg@WebMedia gserg]$ ps
  PID TTY          TIME CMD
 1322 pts/2    00:00:00 bash
 1355 pts/2    00:00:00 ps
[1]+  Killed                  less
[gserg@WebMedia gserg]$_
```

`killall -s SIGKILL` процесс – посылает сигнал всем процессам с именем процесс. Если сигнал не указан, посылает `SIGTERM` .

Сигнал для этой команды необходимо указывать без приставки *`SIG`* . Для получения соответствия цифрового вида и имени сигнала используется опция `-l` команды `killall` .

```
[gserg@WebMedia gserg]$ less ./.bashrc&
[1] 1374
[gserg@WebMedia gserg]$ less ./.bashrc&
[2] 1375

[1]+  Stopped                 less ./.bashrc
[gserg@WebMedia gserg]$ less ./.bashrc&
[3] 1376
```

```
[2]+  Stopped                 less ./.bashrc
[gserg@WebMedia gserg]$ ps
  PID TTY          TIME CMD
 1322 pts/2    00:00:00 bash
 1374 pts/2    00:00:00 less
 1375 pts/2    00:00:00 less
 1376 pts/2    00:00:00 less
 1377 pts/2    00:00:00 ps

[3]+  Stopped                 less ./.bashrc
[gserg@WebMedia gserg]$ killall -s KILL less
[1]   Killed                  less ./.bashrc
[2]-  Killed                  less ./.bashrc
[3]+  Killed                  less ./.bashrc
[gserg@WebMedia gserg]$
```

Командные оболочки. Занятие первое

Понятие командной оболочки. Обзор командных оболочек. Командная оболочка bash . Особенности работы (история команд, оператор "!" , действия по нажатию клавиши <tab>). Многозадачность в консоли. Задания. Управление заданиями. Переменные среды Midnight commander

Ход занятия

1. В мире Linux и Unix работа на компьютере неразрывно связана с понятием командная оболочка (shell) – программа, позволяющая пользователю взаимодействовать с системой посредством ввода и выполнения команд. Тем не менее, командная оболочка является обычной программой. Доказать это можно, установив в качестве оболочки по умолчанию в файле passwd для пользователя другую программу. Но для того, чтобы система знала ее как оболочку, необходимо добавить абсолютное имя файла в /etc/shells .

В составе Linux идет несколько командных оболочек, их состав может меняться в зависимости от дистрибутива, но всегда вы сможете обнаружить:

`Bourne Shell (sh)` – самая старая и самая распространенная командная оболочка для Unix-систем. Нет ни одной системы Unix, где она бы не применялась.

`Bourne Again Shell (bash)` – расширенная `Bourne Shell` . Обладает массой приятных преимуществ, поэтому стала так популярна в последнее время. Является оболочкой "по умолчанию" практически для всех дистрибутивов Linux.

Также популярными оболочками являются:

`csh` – оболочка, система команд которой близка к языку программирования C

`tclsh` – оболочка, система команд которой близка к языку программирования Tcl.

zsh – оболочка, обладающая, наверное, самыми широкими возможностями. Является расширением sh (bourne shell).

Поскольку в Linux "по умолчанию" используется bash , то о ней мы и поговорим.

2. Командная оболочка bash изначально являлась свободно-распространяемым аналогом Bourne Shell. В последствии, когда ее возможности выросли, тогда ее стали считать самостоятельным продуктом. Основными возможностями bash можно назвать следующие:

Таблица 1.1.

№	Возможность	Комментарий
1	Редактирование строки	Возможность отредактировать введенную команду вместо того чтобы переписывать ее заново
2	Организация каналов	Возможность перенаправления ввода-вывода, организации каналов между выполняемыми задачами
3	Удобство в работе	Использование псевдонимов команд, истории команд, автодополнения
4	Управление заданиями	Возможность создания фоновых заданий и управления ими
5	Гибкость настройки	Использование файлов-сценариев для входа для каждого пользователя отдельно, переменные среды

С перенаправлением ввода-вывода и каналами мы познакомимся позднее, на 8-м занятии. Поэтому этот пункт мы опустим. А вот об особенностях работы в bash, о предоставляемых им преимуществах поговорим подробнее.

Bash автоматически записывает все команды, набранные пользователем в файл ~/.bash_history . Для управления этим файлом служит команда history. history - это встроенная команда bash . То есть, исполняемого файла, соответствующего этой команде не существует. Сама командная оболочка выполняет все действия. Введенная без параметров, она просто выводит список всех команд, сохраненных в этом файле и идентична команде cat

`~/.bash_history .`

История команд существует для упрощения набора часто используемых команд. Историю команд можно перебирать по списку клавишами <вверх> и <вниз> .

Другой способ – набрать в командной строке ! и начало команды и нажать <Enter> . Последняя команда из истории, первые буквы которой совпадают с набранными, будет выполнена. Например:

Пример1:

```
[gserg@WebMedia gserg]$ !/usr
/usr/bin/perl ./ptest.pl
OK
[gserg@WebMedia gserg]$  !xfonts
bash: !xfonts: event not found
[gserg@WebMedia gserg]$
```

Но как ускорить ввод, если в истории еще нет необходимой нам команды? В этом случае нам поможет клавиша <Tab> . Набрав несколько первых букв команды (или пути к файлу), нажмите <Tab> и Bash автоматически дополнит вашу команду (или элемент пути). В случае, когда подходит несколько файлов или ни один файл не подходит, система выдаст звуковой сигнал. Если кнопку <Tab> нажать повторно, то когда подходит несколько файлов – система выведет список, а когда ни одного – повторит звуковой сигнал

3. С первого занятия вы должны были помнить, что Linux – многозадачная среда. Однако, до сих пор Вы еще так и не смогли воспользоваться его многозадачностью. На первый взгляд кажется, что консоль не позволяет использовать возможности многозадачности системы, и только в графической среде можно запустить одновременно две или более программы. Но это не так! Консоль тоже многозадачная.

Во-первых, вы можете открыть несколько консолей, открыв в каждой из них по программе. Переключение между консолями будет производиться с помощью клавиш Ctrl+ <Alt+Fx> , где x – номер консоли.

И даже в одной консоли с помощью команд управления заданиями Вы можете в полной мере использовать все преимущества многозадачной системы.

<Ctrl+Z> – комбинация клавиш, посылающая процессу неперехватываемый сигнал sigstop . Позволяет остановить выполнение процесса для передачи управления командной строке.

`команда &` – символ `&` после `команды` позволяет запустить ее в фоновом режиме.

`jobs` – выводит список текущих заданий командного интерпретатора.

`bg <#j>` – переводит задание `#j` в фоновый режим. Перед этим задание должно быть остановлено комбинацией клавиш <Ctrl+z> . Если на данный момент у интерпретатора есть только одно задание, то номер можно не указывать.

`fg <#j>` – переводит задание `#j` в режим выполнения на переднем плане. Задание должно быть остановлено комбинацией клавиш <Ctrl+Z> или находиться в фоновом режиме. Если на данный момент у интерпретатора есть только одно задание, то номер можно не указывать.

Пример2:

```
[gserg@WebMedia gserg]$ man bash
^Z
[1]+  Stopped                 man bash
[gserg@WebMedia gserg]$ vim
^Z
vim
[2]+  Stopped                 vim
[gserg@WebMedia gserg]$ bg 1
[1]+ man bash &
[gserg@WebMedia gserg]$ jobs
[1]+  Stopped                 man bash
[2]+  Stopped                 vim
[gserg@WebMedia gserg]$ fg 2
[2]+ vim
[gserg@WebMedia gserg]$ fg
```

```
[1]+ man bash
[gserg@WebMedia gserg]$
```

4. Переменные среды – системная информация, указывающая Ваши предпочтения, такие как текстовый редактор по умолчанию, пути поиска исполняемых файлов и т.п., а также идентификационные данные пользователя, системы и командной оболочки, такие как имя пользователя, версия Linux и прочее, используемая командным интерпретатором и другими программами.

Часто используемые пользователем переменные это:

`PATH` – переменная содержит пути, в которых системе следует искать исполняемые файлы, если в командной строке не набирается полный или относительный путь к ним.

`PWD` – переменная содержит полное имя текущей директории.

`HOME` – переменная содержит полный путь домашнего каталога пользователя.

`HOSTNAME` – переменная содержит имя компьютера.

`LOGNAME` – содержит имя пользователя, сеанс которого открыт сейчас.

`SHELL` – содержит имя командной оболочки, запущенной в текущем сеансе.

`USER` - содержит имя пользователя, сеанс которого открыт сейчас.

Список переменных, установленных в системе можно увидеть с помощью команды `export` , введенной без параметров.

Командный интерпретатор `bash` имеет и свои переменные. Чтобы локальные переменные стали системными их необходимо экспортировать с помощью все той же команды `export`.Например:

Пример3:

```
[gserg@WebMedia gserg]$ export
```

```
declare -x HOME="/home/gserg"
declare -x HOSTNAME="WebMedia"
declare -x LANG="ru_RU.KOI8-R"
declare -x LOGNAME="gserg"
declare -x PATH="/bin:/usr/bin:/usr/local/bin:/home/gserg/bin"
declare -x PWD="/home/gserg"
declare -x SHELL="/bin/bash"
declare -x TERM="Eterm"
declare -x USER="gserg"
[gserg@WebMedia gserg]$ EDITOR=/bin/vim
[gserg@WebMedia gserg]$ export EDITOR
[gserg@WebMedia gserg]$ export
declare -x EDITOR="/bin/vim"
declare -x HOME="/home/gserg"
declare -x HOSTNAME="WebMedia"
declare -x LANG="ru_RU.KOI8-R"
declare -x LOGNAME="gserg"
declare -x PATH="/bin:/usr/bin:/usr/local/bin:/home/gserg/bin:"
declare -x PWD="/home/gserg"
declare -x SHELL="/bin/bash"
declare -x TERM="Eterm"
declare -x USER="gserg"
[gserg@WebMedia gserg]$
```

Команда `unset` удаляет системную переменную. Например:

Пример4:

```
[gserg@WebMedia gserg]$ unset EDITOR
[gserg@WebMedia gserg]$ export
declare -x HOME="/home/gserg"
declare -x HOSTNAME="WebMedia"
declare -x LANG="ru_RU.KOI8-R"
declare -x LOGNAME="gserg"
declare -x PATH="/bin:/usr/bin:/usr/local/bin:/home/gserg/bin:"
declare -x PWD="/home/gserg"
declare -x SHELL="/bin/bash"
declare -x TERM="Eterm"
declare -x USER="gserg"
```

[gserg@WebMedia gserg]$

5. На сегодняшнем занятии мы познакомимся с вами с еще одной, немного нестандартной, командной оболочкой Midnight commander . Это не командная оболочка в обычном понимании. Это текстовый файловый менеджер – аналог Norton Commander или Far . Midnight commander запускается командой `mc` . Мы поговорим о его возможностях.

Экран Midnight commander'a разделен на две части. Практически все пространство экрана занимают две панели со списком каталогов и файлов. По умолчанию, вторая снизу экрана линия представляет собой командную строку, в которой Вы можете выполнять обычные команды оболочки, а на самой нижней линии изображены подсказки для функциональных клавиш (F1-F10). Верхняя линия символов содержит меню, с помощью которого возможно выполнение множества функций. Для использования меню Вы можете кликнуть мышью в необходимый пункт или нажать клавишу F9 и с помощью клавиш управления курсором выбрать нужный пункт.

Панели Midnight commander обеспечивают просмотр одновременно двух каталогов. Одна из панелей является активной (в том смысле, что пользователь может выполнять те или иные действия с находящимися в ней файлами и каталогами). В активной панели подсвечено имя одного из файлов или каталога, а также выделен цветом заголовок панели в верхней строке. Имя заголовка совпадает с названием каталога, отображаемого в данный момент. Почти все операции выполняются в активной панели. Некоторые операции, такие как перенос или копирование файлов используют пассивную панель в качестве места, куда производится копирование, перенос и т.д.

Теперь поговорим об основных сочетаниях клавиш, которые помогут Вам работать с Midnight commander'ом .

для смены активной панели применяются клавиши <TAB> или <Ctrl>+<I>

для того, чтобы отметить файл, нажмите <Insert> или <Ctrl>+<T>

нажмите <F1> для получения справки

<F3> вызовет просмотрщик файлов

с помощью <F4> вы отредактируете файл

<F5> позволит Вам скопировать файл.

<F6> перенести или переименовать файл

<F7> создать каталог

Клавиша <F8> позволит удалить файл и/или каталог

<F9> , как уже говорилось, открывает доступ к меню.

<F10> – позволит выйти из Midnight commander .

<Home> переведет указатель в начало списка файлов,

<End> - напротив – в конец списка.

<PageUp> и <PageDown> изменят положение показателя на страницу вверх и вниз соответственно.

Клавиша <*> на дополнительной клавиатуре позволит инвертировать выделение файлов (она не действует на каталоги)

Клавиша <+> на дополнительной клавиатуре позволит отметить файлы по маске, а <-> снять отметку с файлов по маске.

<Ctrl>+<R> - обновить содержание директории (перечитав с диска или из сети)

<Ctrl>+<U> - поменять местами правую и левую панели.

<Ctrl>+<O> - убрать/вернуть панели.

Хотя сочетания быстрых клавиш являются оптимальным инструментом для того, чтобы сделать работу с Midnight commander'ом максимально быстрой и удобной, новичкам довольно тяжело выучить их сразу все. Чтобы восполнить этот пробел и добавить другие возможности, для которых отсутствуют комбинации клавиш, Midnight commander имеет

меню (вызываемое по F9).

Меню состоит из пунктов: "Левая панель", "Файл", "Команда", "Настройки", "Правая панель" .

"Левая/правая панель" - эти пункты меню совершенно одинаковые. Различия между ними заключаются только в том, что выполняемые действия будут адресованы в левую или правую панель.

"Формат списка" - открывает диалоговое окно, в котором можно выбрать вид, в котором будет отображаться список файлов/каталогов. На выбор предлагается стандартный, укороченный и расширенный форматы. Однако пользователь в этом окне может и сам определить вид панели таким, какой его будет устраивать, выбрав переключателем "Определенный пользователем".

"Быстрый просмотр" - переводит панель в режим автоматического просмотра файлов, выбираемых на соседней панели. Фокус автоматически переключается на противоположную панель.

"Информация" - переводит панель в режим просмотра информации о файле, подсвеченном в соседней панели, такой как положение, права доступа и владелец, файловая система и устройство, на котором он расположен, количестве жестких ссылок, связанных с этим файлом, а также информации об устройстве, на котором расположен файл,

"Дерево" - переводит Midnight commander в режим, похожий на режим работы Проводника из ОС Windows. В панели, к которой применена команда "Дерево" , строится дерево каталогов, по которому можно перемещаться с помощью стрелок управления курсором, клавиш PageUp, PageDown, Home, End . В соседней панели высвечивается содержимое каталога, подсвеченного в дереве.

"Порядок сортировки" - открывает диалоговое окно, в котором вы можете выбрать атрибут, по которому будет производиться сортировка файлов и каталогов в списке из таких как имя, расширение, время правки, время доступа, время изменения атрибутов, размер, узел (на котором расположен файл). Также можно оставить файлы без сортировки, сортировать с учетом регистра или в обратном порядке.

"Фильтр" - позволяет выбрать имена файлов, которые будут отображаться в панели с помощью регулярного выражения, введенного в диалоговом окне.

"FTP-соединение" - с помощью этой команды Вы можете установить соединение с удаленным (или даже локальным) компьютером по протоколу ftp . Если введен только адрес удаленного сервера, то Midnight commander попытается установить анонимное соединение. Полная же строка, с помощью которой задается узел такова:

ftp:имя_пользователя:пароль@адрес_сервера:порт/каталог_на_сервере

После установки соединения, работа с удаленной файловой системой происходит аналогично работе с локальной ФС.

"Shell-соединение" - позволяет открыть сетевое соединение по протоколу FISH (FIle transfer over SHell – передача файлов посредством оболочки). FISH использует протоколы RSH (Remote SHell – удаленная оболочка) или SSH (Secure SHell – защищенная оболочка, аналог RSH , но с поддержкой шифрования передаваемых данных). Полная строка, с помощью которой пользователь может задать удаленный узел такова:

sh:имя_пользователя@адрес_сервера:опции/каталог_на_сервере

Параметр `имя_пользователя` , `опции` и `каталог_на_сервере` не обязательны. Если имя пользователя не указано, то Midnight commander будет пытаться зарегистрироваться на удаленной системе с именем пользователя, используемым на локальном компьютере.

"Пересмотреть" - аналог комбинации клавиш <Ctrl>+<R> - вызывает обновление списка файлов и каталогов в текущей панели, перечитав их с диска или по сети.

"Файл" - раздел меню, пункты которого обеспечивают основные функции обработки файлов и каталогов, такие как:

"Меню пользователя" - позволяет вызвать меню, которое устанавливает сам пользователь. Также вызывается клавишей <F2> .

"Просмотр файла" - аналог функции, выполняемой по нажатию <F3>. Позволяет просмотреть подсвеченный файл (или зайти в каталог). Поддерживает множество форматов, таких как текстовые форматы, архива, Winword DOC, исполняемые файлы Linux и т.д.

"Просмотр файла..." - то же, что и предыдущий пункт, но действует не на подсвеченный файл, а на тот, имя и путь к которому будет введен в диалоговом окне.

"Просмотр команды" - позволяет выполнить команду и просмотреть ее стандартный вывод в режиме просмотра файла.

"Редактирование" - открывает файл для правки. Простой встроенный текстовый редактор имеет достаточный набор встроенных функций для редактирования файлов конфигурации, исходных текстов программ и т.д., а встроенная автоматическая подсветка синтаксиса делает редактирование более удобным, а редактируемые тексты более удобочитаемыми.

"Копирование" - копирует файл из активной панели в пассивную. Аналог функции, вызываемой по <F5> . По умолчанию, копируемым считается подсвеченный в активной панели файл (или группа файлов), а местом назначения- каталог, открытый в пассивной панели. Это можно изменить, поправив значения полей, в открывающемся после вызова этой команды, диалоге.

"Права доступа" - позволяет изменить права доступа к файлу (или группе файлов) в диалоговом окне.

"Жесткая ссылка" - создает жесткую ссылку для файла, подсвеченного в активной панели. Путь к ссылке необходимо указать в диалоговом окне.

"Символич. ссылка" - создает символическую ссылку. По умолчанию за адресуемый принимается файл, подсвеченный в активной панели, а создаваемая ссылка будет иметь то же имя и располагаться в каталоге, открытом в пассивной панели. Пользователь может изменить это в открывающемся диалоговом окне.

"Права ссылки" - команда позволяет изменить права доступа к символической ссылке (а не адресуемому ею файлу).

"Владелец/группа" - изменяет владельца и/или группу, к которой принадлежит файл/каталог.

"Права (расширенные)" - позволяет изменять одновременно права доступа к файлу и его владельца и/или группу. Права доступа представлены в виде трех последовательностей `rwx` для владельца, группы и всех пользователей.

"Переименование" - позволяет переименовать/переместить файл. Аналог функции, вызываемой по <F6> . По умолчанию, перемещаемым/переименуемым считается подсвеченный в активной панели файл (или группа файлов), а местом назначения- каталог, открытый в пассивной панели. Это можно изменить, поправив значения полей, в открывающемся после вызова этой команды, диалоге.

"Создание каталога" - создает каталог. Аналог функции, вызываемой по <F7> . По умолчанию, каталог создается в каталоге, открытом в активной панели. Это можно изменить, если в открывшемся диалоге указать полный путь к создаваемому каталогу.

"Удаление" - удаляет файл/группу файлов/каталог. Аналог функции, вызываемой по <F8> .

"Смена каталога" - меняет текущий каталог. Аналогична команде `cd` командного интерпретатора `Bash` . Необходимый каталог вводится в диалоговом окне.

"Отметить группу" - Отмечает группу файлов по маске в каталоге, открытом в активной панели. Аналог функции, вызываемой с помощью <+> на дополнительной клавиатуре.

"uNselect group" - Снимает отметку с группы файлов по маске в каталоге, открытом в активной панели. Аналог функции, вызываемой с помощью <-> на дополнительной клавиатуре.

"Инвертировать отметку" - Изменяет значение отметки для всех файлов в каталоге, открытом в активной панели. Аналог функции, вызываемой с помощью <*> на дополнительной клавиатуре.

"Выход" - завершает работу Midnight commander'a . Аналог функции,

вызываемой по <F10> .

"Команда" - раздел меню, пункты которого вызывает дополнения и расширения Midnight commander'a, такие как:

"Дерево каталогов" - выводит на экран диалоговое окно с построенным деревом каталогов файловой системы. Выбрав каталог в дереве и нажав <Enter> Вы сможете сменить каталог в активной панели на выбранный.

"Поиск файла" - выводит на экран диалог, в котором можно указать параметры искомого файла:

От каталога - в каком каталоге искать файлы. Каталог можно ввести вручную или выбрать из дерева с помощью кнопки "Дерево".

Шаблон имени - в этой строке задается регулярное выражение, описывающее имя искомого файла.

Содержит текст – в эту строку вписывается последовательность символов, которая содержится внутри искомого файла. Установив галочку "Учет регистра" , Вы сможете учитывать регистр букв для введенного Вами значения.

После нажатия на <Enter> или кнопку "Дальше" начнется поиск файлов. Во время поиска и после его окончания Вы можете приостановить/ продолжить или отменить поиск, просмотреть, отредактировать найденные файлы, вывести их в отдельную панель или перейти в каталог, содержащий подсвеченный файл.

"Переставить панели" - команда меняет местами правую и левую панели. Аналог функции, выполняемой по <Ctrl> + <U> .

"Отключить панели" - команда убирает/возвращает панели, показывая или скрывая экран `shell`.Аналог функции, выполняемой по <Ctrl> + <O> .

"Сравнить каталоги" - команда вызывает диалоговое окно. Скорее всего она не доработана разработчиками, и в существующей версии выполняет только выделение всех файлов в обоих панелях для того, чтобы увидеть их общий размер.

"Критерий панелизации" - этот пункт меню было бы правильнее назвать "перенаправление вывода на панель" позволяет Вам выполнить программу и отобразить вывод этой программы в активной панели. Например, если вы хотите отобразить в панели только символические ссылки, то Вы можете использовать пункт "Критерий панелизации" с командой `find . -type l -print` . На панели в результате поиска окажутся только символические ссылки. Проще говоря, это расширенный инструмент фильтра и поиска файлов.

"Размеры каталогов" - функция, позволяющая увидеть размеры каталогов (включая подкаталоги и входящие в них файлы). Функция довольно требовательная к ресурсам системы и занимающая относительно долгое время. После смены каталога информация о его размере потеряется.

"Burn to CD this dir" - записывает на CD текущий каталог.

"История команд" - выводит в диалоговом окне историю команд пользователя. Выбранная команда будет автоматически помещена в командную строку.

"Справочник каталогов" - служит для вывода списка условных меток для наиболее часто используемых каталогов. Этот список можно использовать для быстрого перехода в каталог. Пользуясь диалоговым окном, Вы также можете добавить, удалить или отредактировать метку. С помощью комбинации клавиш <Ctrl>+<x> <h> вы можете быстро добавить текущий каталог в список.

"Список активных ВФС" - показывает список открытых в данный момент виртуальных файловых систем (FTP, NFS, Arhive и т.д.).

"Освободить ВФС сейчас" - завершить работу со всеми активными на текущий момент ВФС.

"Фоновые задания" - открывает диалоговое окно, с помощью которого пользователь может управлять (остановить, возобновить, снять) фоновыми заданиями.

"Восстановление файлов" - эта команда позволяет восстановить удаленные ранее файлы, но только в том случае если Вы используете

файловую систему Ext2 с установленными свойствами по восстановлению файлов.

"Файл расширений" - позволяет просмотреть и отредактировать действия, выполняемые Midnight commander'ом при выборе и активации файла в зависимости от его расширения.

"Файл меню" - позволяет редактировать меню пользователя, вызываемое по нажатию <F2> .

"Правка меню редактора" - редактирование функционального меню встроенного редактора.

"Файл синтаксиса" - позволяет редактировать файл, отвечающий за подсветку синтаксиса в редактируемом файле.

"Настройки" - раздел меню, с помощью которого Вы сможете изменить поведение Midnight commander'a , задав соответствующие настройки.

"Конфигурация" - Данный пункт меню позволяет изменить наиболее важные, основные параметры работы программы, такие как основные параметры панелей, выполнения программ и др.

"CD Burning config" - настройки записи на CD, такие как скорость записи, поддержка мультисессий, используемая ФС.

"Внешний вид" - позволяет настроить внешний вид панелей, меню, строки подсказки, статуса, подсветку файлов в зависимости от типа и т.д.

"Подтверждение" - выводит диалог, в котором пользователь может задать, на какое действие выводить окно запроса подтверждения, а на какое – нет.

"Биты символов" - позволяет выбрать кодировку, используемую для вывода символов, и также разрядность (7 или 8 бит) для таблицы символов.

"Распознавание клавиш" - позволяет определить, какая escape-последовательность для вашего терминала какой клавише соответствует.

"Виртуальные ФС" - настройка параметров виртуальных файловых систем, таких как таймауты, прокси и т.д.

"Сохранить настройки" - сохраняет сделанные вами изменения в настройках в файле конфигурации, для того чтобы все они остались в действии при следующем запуске программы.

Командные оболочки. Занятие второе

Программирование для Bash.

Ход занятия

Сценарии командных оболочек играют в Linux огромную роль. Кроме того, что каждый пользователь стремится по максимуму автоматизировать выполняемую им работу с помощью сценариев shell , так и сценарии инициализации системы и множества приложений есть ничто иное как просто сценарии shell . Как и на позапрошлом занятии, мы с Вами будем рассматривать программирование в shell на базе Bourne Shell .

1. Любой сценарий для bash начинается с указания в первой строке редактируемого файла комбинации:

```
#!/bin/bash
```

Эта последовательность указывает на программу, которую следует использовать для обработки данного сценария – в нашем случае командную оболочку bash .

Язык bash довольно мощный. Важную часть представляют собой объявления переменных. Принято называть переменные буквами верхнего регистра, например:

```
#!/bin/bash
TERM=vt100
COUNTER=0
```

Для извлечения значения переменной используется следующий синтаксис:

```
ALFA=$BETA+$GAMMA
```

Значок `$` означает, что нужно извлечь значение переменной `BETA` , а не использовать строку `"BETA"` . Переменной можно присвоить вывод какой-либо команды. При этом переменная станет массивом,

разделителем записей в котором будут служить символы пробела, табуляции и перевода строки. Для этого выполняемую команду необходимо заключить в обратные апострофы и присвоить переменной:

```
FILES=`/bin/ls -a /home/student`
```

Вывод на экран значений переменных, или просто фраз производится с помощью оператора `echo`:

```
echo $ALFA это строки BETA и GAMMA с плюсом посредине
echo Done
```

При выводе выводимую информацию можно заключать в одинарные(') и двойные(")кавычки. Одинарные полностью выводят текст, написанный внутри них, а в двойных указанные переменные заменяются их значениями:

```
[gserg@WebMedia serial]$ echo '$USER'
$USER
[gserg@WebMedia serial]$ echo "$USER"
student
[gserg@WebMedia serial]$
```

В качестве комментариев применяется знак `#` . Он может быть использован как в начале, так и в середине строки:

```
#Временный каталог
TMP='/tmp'     #используем кавычки
```

Как часть сценария может быть использована любая доступная команда операционной системы:

```
#очистим экран
clear
```

Bash поддерживает несколько операций сравнения. В случае, если условие верно, то оператор сравнения возвращает `0 (true)` . Проверить возвращаемое значение можно с помощью специального значения `$?`. Перечислю их:

```
#проверка кода возврата последней команды
[ -d abcd ]
echo $?
#проверка прав доступа к файлу:
[ -d abcd ] #является ли файл abcd каталогом?
[ -f abcd ] #является ли файл abcd обычным файлом?
[ -L abcd ] #является ли файл abcd символической ссылкой?
[ -r abcd ] #есть ли доступ на чтение к файлу abcd?
[ -w abcd ] #есть ли доступ на запись к файлу abcd?
[ -s abcd ] #файл abcd имеет ненулевой размер (он не пуст)?
[ -u abcd ] #имеет ли файл abcd установленный бит SUID?
[ -x abcd ] #является ли файл abcd исполняемым?
#проверка строк
[ -z $STRING ] #пуста ли строка STRING?
[ -n $STRING ] #строка STRING не пуста?
[ $STRING = $STRING1 ] #равны ли строки STRING и STRING1?
[ $STRING != $STRING1 ] #строки STRING и STRING1 не равны?
#проверка чисел
#при проверке чисел в условии их обязательно необходимо
#заключить в двойные кавычки
[ $DIGIT -eq $DIGIT1 ] #равны ли числа DIGIT и DIGIT1?
[ $DIGIT -ne $DIGIT1 ] #числа DIGIT и DIGIT1 не равны?
[ $DIGIT -gt $DIGIT1 ] #число DIGIT больше DIGIT1?
[ $DIGIT -lt $DIGIT1 ] #число DIGIT меньше DIGIT1?
[ $DIGIT -ge $DIGIT1 ] #число DIGIT больше или равно DIGIT1?
[ $DIGIT -le $DIGIT1 ] #число DIGIT меньше или равно DIGIT1?
```

Сценарию на bash могут быть переданы параметры командной строки. Значения их можно получить с помощью переменных `$0, $1, $2 .. $n-1, $n` . При этом `$0` – имя файла сценария, `$1` – первый параметр, `$2` – второй, ... `$n` – последний.

Но операторы сравнения без операторов ветвления бесполезны. Простейшим оператором ветвления в языке сценариев `bash` является оператор `if-then-else - fi`.Полная структура для оператора выглядит следующим образом:

```
if <оператор сравнения1>; then
  <действия 1>
```

```
elif <оператор сравнения2>; then
  <действия 2>
else
 <действие 3>
fi
```

Оператор `if` позволяет упрощать конструкцию. Разделы `elif` и `else` не являются обязательными. Для правильной работы достаточно использования только конструкции `if-fi`.

Для примера приведу скрипт:

```
#!/bin/bash
if [ -f /etc/passwd ]; then
  echo "/etc/passwd – обычный файл"
else
 echo "/etc/passwd – не обычный файл"
fi
exit
```

В этом сценарии проверяется, обычный ли файл /etc/passwd и выводится соответствующее сообщение на экран.

Естественно мы не сможем обойтись и без таких конструкций, как циклы.

Простейшим циклом является цикл перебора `for` .

```
for <ЭЛЕМЕНТ> in $<СПИСОК>
 do
  <действия>
 done
```

Циклу передается переменная со списком параметров, разделенных пробелом, знаком перевода строки или табуляции. Дальше цикл выполняет действия для каждого элемента из списка, например:

```
#!/bin/bash
echo "Поиск каталогов в $HOME"
echo
```

```
#перейдем в домашний каталог
pushd $HOME
#получим список файлов
LIST=$(ls -a $HOME)
#организуем цикл для каждого из
#значений в переменной LIST
for ITEM in $LIST
 do
  #если вхождение – каталог, выведем его на экран
  if [ -d $ITEM ]; then
   echo "$ITEM"
  fi
done
#вернемся в прежнюю директорию
popd
#выход
exit
```

Однако цикл перебора подходит нам не во всех случаях. Во многих случаях нам необходим цикл с условием. Это цикл `while`.

```
while <условие>
 do
  <действия>
 done
```

Цикл `while` выполняется до тех пор, пока истинно `<условие>`.Попробую показать на примере организации цикла со счетчиком. Для вычислений я буду использовать команду `expr` , позволяющую выполнять математические операции с переменными `bash` :

```
#!/bin/bash
#определяем наличие первого параметра
#командной строки
if [ -z $1 ]; then
 echo "Использование: $0 количество_проверок"
 exit
fi
```

```
#очистим экран
clear
#определим переменную счетчика цикла
COUNTER=0
#организуем цикл на $1 итераций
while [ $COUNTER -lt $1 ]
 do
  echo "Загрузка системы (за $1 секунд)"
  echo   echo "секунда $COUNTER"
  echo "-- загрузка системы --"
  #команда определения средней загрузки
  /usr/bin/uptime
  echo "-- место на дисках --"
  #определение свободного места на винчестере
  df -h
  echo "----------------"
  echo
  #пауза 1 секунду
  /bin/sleep 1
  #очистим экран
  clear
  #приращение переменной цикла
  COUNTER=`expr $COUNTER + 1`
 done
#выход
exit
```

Организация вывода имени программы не очень удачно выполнено в данном случае, потому что будет выводиться не только имя запущенной программы, а и путь, набранный нами в командной строке. Исправить это нам поможет директива `basename`, выделяющая из пути имя файла. Вот как будет выглядеть фрагмент приведенного выше сценария с ее применением:

```
#!/bin/bash
#определяем наличие первого параметра
#командной строки
if [ -z $1 ]; then
 echo Использование: `basename $0` количество_проверок
```

```
    exit
  fi
  ...
```

Планирование заданий. Работа с дисковыми накопителями

Понятие планирования заданий. Команда at . Демон cron . Команда crontab. Работа с дисковыми накопителями, команда mount.

Ход занятия

1. Очень часто в Linux администратор встречается с проблемой, когда выполнение какой-либо программы (или shell -сценария) может происходить и без его присутствия, но необходим инструмент, реализующий эту возможность.

В современных Linux-системах для этой цели принято использовать механизмы планирования заданий. Реализованы эти механизмы с помощью демонов планирования заданий – `at` и `cron` .

С помощью этих программ появляется возможность установить выполнение программы на заранее известное время. Команда `at` используется в тех случаях, когда выполнение задания - разовая процедура. Если же задание предполагается выполнять с какой-либо периодичностью, то лучше всего использовать демон `cron` и команду `crontab`.

Методика планирования представляет из себя понимание процессов, происходящих с сервером (или персональным компьютером) в каждый момент времени. Для планирования применяется форма, аналогичная приведенной ниже с дискретизацией в 5 минут:

01 число	0м	5м	10м	15м	20м	25м	30м	35м	40м	45м	50м	55м
0ч	Резервная копия											
1ч												
2ч				Обновления Linux			Установка обновлений Linux					
3ч	Смотри недельную форму											
4ч												

5ч												
6ч												
7ч												
8ч	Резерв для непредвиденных случаев											
9ч												
10ч												
11ч												
12ч												
13ч	Mozilla											
14ч												
15ч												
16ч												
17ч												
18ч											Проверка >	
19ч	с помощью tripware целостности файловой >											
20ч	системы											
21ч												
22ч	Самая высокая загрузка системы в это время											
23ч												

Составляется расписание на каждый день месяца. Совместно с ней составляется форма по дням недели, которая позволяет планировать выделенные 2 часа (или больше, если это потребуется). Пример такой формы для понедельника:

понедельник	0м	5м	10м	15м	20м	25м	30м	35м	40м	45м	50м	55м
4ч	Отчет SARG											
5ч												

Эти формы позволят оптимально использовать время работы сервера и грамотно планировать задания.

После того, как будут выписаны все задания, стоящие в текущий момент, нужно будет найти подходящее место для вновь вставляемого задания. Опытные системные администраторы считают, что стоит выделять около часа в сутки в расписании заданий для того, чтобы всегда можно было вставить непредвиденное разовое задание, а также освобождать от выполнения заданий время наивысшей загрузки системы. Не рекомендуется планировать несколько заданий на одно и то же время.

2. Семейство команд `at (at, atq, atrm)` представляет собой инструменты для выполнения задания в определенное время по таймеру. Для правильного функционирования данной команды в системе должен быть запущен демон `atd` . Демон `atd` поддерживает очередь заданий, которые должны быть выполнены в то или иное время.

Для постановки задания (или нескольких заданий в очередь на одно и то же время) вам необходимо выполнить команду `at`:

Пример 1.

```
[student@Klass801 student]$ at 19:00
at> /home/student/bin/first.sh
>Control-D>
job 1 at 2004-12-01 13:01
[student@Klass801 student]$_
```

Результат выполнения команды, указанной демоном `at` будет записан в Ваш почтовый ящик `linux (/var/spool/mail/student)`.Но система планирования `at` позволяет и управлять поставленными в очередь заданиями. Просмотреть очередь заданий можно используя команду `atq`:

Пример 2.

```
[student@Klass801 student]$ atq
1       2004-12-01 13:01 a student
[student@Klass801 student]$
```

Команда `at` позволяет поставить в очередь несколько заданий, которые будут последовательно выполнены друг за другом. Сделать это можно

как в интерактивном режиме (набрав команду `at` в командной строке), так и указав команде `at` параметр `-f`:

```
[student@Klass801 student]$ at -f commands 14:50
job 8 at 2004-12-05 14:50
[student@Klass801 student]$_
```

Команда `atrm` удаляет из очереди задание:

```
[root@ns root]# atq
7       2004-12-05 14:48 a root
8       2004-12-05 14:50 a root
[root@ns root]# atrm 7
[root@ns root]# atq
8       2004-12-05 14:50 a root
[root@ns root]#
```

При планировании заданий с помощью команды `at` стоит уделять особое внимание времени выполнения задания, чтобы средняя загрузка системы не превысила предельно допустимое значение.

3. В отличие от команд `at`,демон `cron` и команда управления планированием `crontab` позволят Вам точно планировать задания. Как в случае с `at`, задания запускает программа-демон `crond`. Команда `crontab` служит лишь для управления заданиями. Перед использованием команды необходимо создать файл, описывающий таблицу заданий. Формат файла таков:

```
минуты часы дни_месяца месяц дни_недели команда
минуты – числа от 0 до 59, или *
часы – числа от 0 до 23, или *
дни_месяца – числа от 1 до 31, или *
месяц – числа от 1 до 12, или *
дни_недели – числа от 0 до 7, причем 0 или 7 – воскресенье, или *; наг
0 10 * * *  /home/student/bin/script   #запуск в 10:00 ежедневно
15 * * * 1  /home/student/bin/script2  #в 15 минут каждого часа в понедель
```

Команда `crontab` позволяет использовать периоды:

```
10-15 * /2 * * /home/student/bin/script3 #каждую минуту, с 10 до 15 минут
```

```
#четным дням каждого месяца
```

После создания файла заданий необходимо вызвать команду `crontab` и указать ей в качестве параметра имя файла с заданиями :

```
[student@Klass801 student]$ crontab jobs
```

Просмотреть список заданий, установленных Вами можно с помощью параметра `-l` :

```
[student@Klass801 student]$ crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.1333 installed on Sat Dec  4 15:56:57 2004)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
10 * * * * /home/student/bin/script
[student@Klass801 student]$
```

Очистить список заданий можно с помощью параметра `-r` :

```
[student@Klass801 student]$ crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.1333 installed on Sat Dec  4 15:56:57 2004)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
10 * * * * /home/student/bin/script
[student@Klass801 student]$ crontab -r
[student@Klass801 student]$ crontab -l
no crontab for student
[student@Klass801 student]$
```

Команда `crontab` позволяет также и редактировать список заданий с помощью параметра `-e`.В качестве редактора будет использоваться редактор, указанный (в порядке очередности) в переменной окружения `$VISUAL`, `$EDITOR` или `/bin/vi` . После сохранения файла, `crontab` автоматически переинициализирует таблицу заданий.

Параметр `-u User` позволяет управлять заданиями других пользователей. При использовании этого параметра не из под суперпользователя, вам придется ввести пароль.

4. Очень часто планирование заданий связано с созданием резервных

копий. Как правило, резервные копии создаются на внешние носители (магнитные ленты, диски и прочее). Сегодня мы с вами попробуем научиться монтировать различные внешние устройства. Монтирование – подключение файловых систем внешних накопителей в один из каталогов корневой ФС (точку монтирования). Для монтирования устройств в Unix применяется команда `mount`.При запуске без параметров `mount` покажет все ФС, смонтированные на текущий момент, а также параметры, используемые при монтировании.

```
root@ADM:/home/oem# mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
/sys on /sys type sysfs (rw,noexec,nosuid,nodev)
varrun on /var/run type tmpfs (rw,noexec,nosuid,nodev,mode=0755)
varlock on /var/lock type tmpfs (rw,noexec,nosuid,nodev,mode=1777)
procbususb on /proc/bus/usb type usbfs (rw)
udev on /dev type tmpfs (rw,mode=0755)
devshm on /dev/shm type tmpfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
lrm on /lib/modules/2.6.20-15-generic/volatile type tmpfs (rw)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

Эта команда принимает много параметров, но мы с вами поговорим о следующих:

`-t FS` – указывает файловую систему, которая используется на внешнем устройстве

`-r` – смонтировать ФС в режиме "только чтение"

`-w` – смонтировать ФС в режиме "чтение/запись"

`-o` – указать дополнительные опции монтирования, такие как кодировку, набор символов, под каким пользователем монтировать и прочие. Параметры зависят от типа файловой системы, и их стоит уточнять в `man`. Общий синтаксис `mount` выглядит как:

```
mount -t FS -w/-r device mountpoint -o external_fs_options
```

Например:

```
root@ADM:/home/oem# mount /dev/sdb1 /mnt/flash
```

Для отмонтирования устройств в Unix применяется команда `umount`. В качестве параметров `umount` принимает либо точку монтирования, либо смонтированное устройство, либо и то и другое:

```
root@ADM:/home/oem# umount /dev/sdb1 /mnt/flash
root@ADM:/home/oem# umount /mnt/flash1
root@ADM:/home/oem# umount /dev/sdc1
```

Текстовые редакторы. Редактор vi

Редактор vi.

Ход занятия

В Linux широко используются около десятка текстовых редакторов. Каждый из них хорош для своих целей. Но наиболее часто используемыми все же являются редакторы vi и emacs .

1. Редактор vi – один из старейших и мощнейших редакторов в Linux. Он работает даже на таких аппаратных терминалах как VT100 и XTERM. Современный вариант этого редактора называется vim (Vi IMproved – улучшенный vi). vim эмулирует все команды vi,и в то же время добавляет много дополнительных возможностей, таких как:

- многоуровневый процесс отмены ранее выполненных действий (`undo`);
- использование нескольких окон редактирования;
- редактирование командной строки;
- встроенная справочная система (команда `:help`) и многое другое.

Мы с вами будем обсуждать именно vim – расширенный вариант vi.

Запуск редактора производится командой `vim`:

```
vim <имя файла>
```

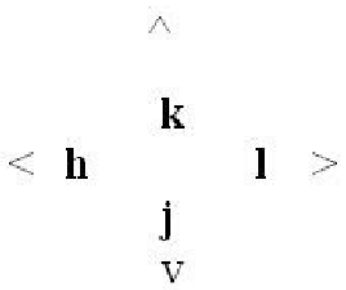
Если вы запустили `vim` без указания имени файла, то Вы можете или начать редактирование, или открыть файл командой `:edit полное_имя_файла<Enter>`.

Сохранить текущий файл можно с помощью команды `:w <имя_файла><Enter>`.Параметр `имя_файла` в команде сохранения необязателен. Если он не указан, то файл будет сохранен с тем же именем, с которым и был открыт. Если же вы редактируете новый файл, то имя указать обязательно.

Выйти из `vim` можно по команде `:q <Enter>` . Для того, чтобы выйти из редактора, сохранив все сделанные изменения, используйте комбинацию команд `:wq <Enter>` , а для отмены сохранения всех сделанных изменений - `:q!<Enter>`.

`vim` может работать в нескольких режимах: просмотра и ввода команд(по умолчанию), редактирования и выделения.

Из-за того, что редактор обеспечивает совместимость с разными типами терминалов, кроме стандартных клавиш управления, он поддерживает клавиши буквенной клавиатуры для управления курсором:



Такой режим перемещения работает только в режиме просмотра и режиме выделения.

Для того, чтобы узнать, в каком месте файла Вы сейчас находитесь, нажмите <Ctrl+g> . Внизу экрана появится строка состояния, в которой будет указано имя редактируемого файла и текущая позиция в нем.

Для перехода к концу файла, Вам необходимо нажать комбинацию <Shift+G> . Если же нужно перейти к какой-то определенной строке в файле, нажмите <№строки Shift+G> , например для перехода к 35 строке: <3><5><Shift+G>

Поиск по файлу производится с помощью команды `/` (аналогично поиску в `less`). По умолчанию поиск регистрозависимый (то есть маленькие и большие буквы считаются разными). Изменить это можно с помощью команды `:set ic <Enter>` . Подсветить цветом все искомые совпадения можно командой `:set hls <Enter>` .

Продолжить поиск до следующего вхождения вниз по тексту можно командой <n> , а вверх по тексту - <Shift+N> .

Для удаления символов в режиме ввода команд, Вам необходимо нажать <Del> или <x> , а для удаления сразу целой строки – <dd > . Если понадобилось удалить сразу несколько строк, то перед вводом команды <dd> , необходимо ввести число удаляемых строк (например: <5dd> удалит 5 строк, включая текущую вниз по тексту). Для удаления всех символов до конца слова необходимо набрать команду <dw> , а до конца строки – <d$> . Если же необходимо удалить строку до ее начала – используйте комбинацию <d^> (Если в начале строки есть пустые символы, то удалить до начала строки вместе с пустыми символами поможет команда <d0>).

Для начала редактирования (вставки текста) необходимо нажать <Insert> или <i> . Перевести vim в режим замены символов в режиме редактирования также может повторно нажатая клавиша <Insert> , а в обычном режиме - <Shift+R> . Выйти из режима редактирования можно с помощью нажатия <Esc> . Если в процессе редактирования Вы сделали ошибку, и Вам необходимо выполнить откат выполненных действий, то:

- перейти в режим ввода команд;
- нажать <u>.

Откат выполненной операции также действует на операции удаления символов, строк, слов и частей строк. Для отката (возврата выполненных команд) используется комбинация <Ctrl+r>.

При выполнении операций удаления, удаленный текст помещается в буфер. Этот буфер можно вставить потом в любое место в редактируемом тексте. Делается это нажатием клавиши <p> . Вставляемый текст будет помещен после текущей строки.

Для копирования текста в буфер используются команды <y> (1 символ), <y$> (до конца строки) , <y^> (до начала строки) , <yw> (слово), <yy> (вся строка).

Для изменения части слова используется команда <cw> . Встав над

словом, которое необходимо исправить, нажмите <cw> . После этого все знаки слова, начиная с позиции курсора будут удалены и vim перейдет в режим редактирования.

Команда <c> используется с теми же параметрами, что и команда удаления, за исключением начала строки:

<cw> – редактировать до конца слова

<c$> - редактировать до конца строки

Можно применять цифры, например:

< 2cw> - редактировать до конца второго слова, включая текущее

<5c$> - редактировать до конца пятой строки, включая текущую.

Vim позволяет производить замену:

:s/было/стало/g – сделать замену в текущей строке

:1,20s/было/стало/g – сделать замену в строках с 1 по 20-ю

:%s/было/стало/g – сделать замену во всем файле

: 30,100s/было/стало/gc – сделать замену в строках с 30 по 100 и спрашивать при каждой замене разрешения сделать ее.

:/слово/s/было/стало/g – сделать замену в строках содержащих "слово"

Для вставки файла (слияния) используется команда :r имя_файла <Enter> . В результате выполнения этой команды файл имя_файла будет прочитан и вставлен после текущей строки.

Во время работы с редактором часто возникает необходимость выполнить команду оболочки без выхода из редактора . Vim предоставляет такую возможность:

:!команда – выполнить команду "команда"

`:%!команда` – пропустить редактируемый текст через внешнюю команду "`команда`"

Во время выполнения команд нужно учитывать, что все команды действуют в пределах вновь вызванного командного интерпретатора, и потому не меняют системного окружения текущего, т.е. в результате выполнения команд не изменится рабочий каталог редактора или какие-либо переменные окружения.

Помните, что все команды, начинающиеся с двоеточия, необходимо заканчивать нажатием `<Enter>` .

Текстовые редакторы. Редактор Emacs

Редактор Emacs.

Ход занятия

Emacs представляет собой мощный экранный редактор текста с массой возможностей. Его можно считать средой программирования с большими возможностями редактирования текста, т.к. Emacs предлагает массу встроенных функций по компиляции и поиску ошибок в коде. Есть несколько разновидностей Emacs . Мы с вами будем рассматривать GNU Emacs .

Emacs написан на языке высокого уровня Elisp . Хотя для конфигурации и работы с редактором знание Elisp не требуется, но если Вы возьметесь за создание специфических функций обработки текста, то без знания языка Вам не обойтись.

Для работы с emacs используется система меню и комбинаций клавиш. Используются сочетания с клавишами <ctrl> и <meta>.Так как на клавиатуре для IBM PC совместимых ПК клавиши <meta> нет, то вместо нее можно использовать <alt> или <esc>.Для доступа к системе меню используйте клавишу F10.

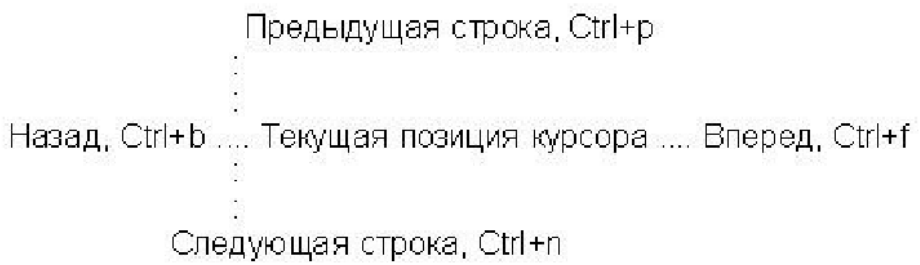
Для запуска Emacs наберите в командной строке `emacs <имя_файла>` . Параметр имя_файла необязателен. Если он не указан, то будет открыт для редактирования новый файл. Если установлена переменная `$DISPLAY` (то есть открыт графический сеанс X), то Emacs попытается открыть графическое окно программы на экране, указанном в переменной `$DISPLAY` , в противном случае будет работать в текстовом режиме.

Чтобы открыть файл для редактирования, наберите команду <Ctrl+x> <Ctrl+f> и введите в появившейся строке имя файла, который Вы собираетесь открыть.

Для отмены набранной команды открытия или сохранения файла, нажмите <Ctrl+g>.

Для завершения работы с редактором используйте комбинацию клавиш <Ctrl+x> <Ctrl+c> или пункт меню F10-File-Exit Emacs . Перед выходом из редактора сохраните редактируемый документ. Если вы не используете систему меню (F10), то редактор не запросит подтверждения сохранения данных, и все новые данные будут потеряны.

Для перемещения курсора используйте схему:



Современные терминалы понимают также перемещение курсора более привычным для нас способом – с помощью клавиш управления курсором.

Если есть необходимость более быстрого перемещения, используйте <alt+f> для перемещения на слово вперед и <alt+b> для перемещения на слово назад, <Ctrl+a> в начало строки и <Ctrl+e> в конец строки, <alt+a> в начало предложения и <alt+e> - в конец предложения, Alt+< - в начало текста и Alt+> - в конец текста (для набора знаков < и > используйте <Shift>).

Если Вы хотите вставить текст, просто наберите символы, и они отобразятся в редактируемом документе. Удаляется текст с помощью клавиш <Del> и <Backspace>, однако клавиша <Backspace> есть не на всех терминалах (в отличие от <Del>). Если же Ваш терминал не поддерживает этих клавиш, то Вы можете воспользоваться комбинациями: <Ctrl+d> удалит символ под курсором (аналог <Del>), <alt+d> удалит слово следующее, за курсором, <Ctrl+k> - все от курсора до конца строки, <alt+k > - до конца предложения.

Для вставки только что удаленного текста воспользуйтесь командой <Ctrl+y>.Для вставки сохраняется только текст, удаленный командами группового удаления (то есть теми, которые удаляют сразу большую

порцию текста, а не один символ).

Существует в Emacs и команда отката. Это комбинация <Ctrl+x> <u>.

Для повторения команд редактирования может использоваться счетчик повторения – команда <Ctrl+u>.Действует она следующим образом:

- <Ctrl+u> 8 <u> - произведет откат на 8 действий.
- <Ctrl+u>7 <Ctrl+d> - удалит 7 символов

Для сохранения сделанных изменений, используйте <Ctrl+x> <Ctrl+s>.Если Вы редактируете новый файл, то Emacs запросит имя файла для сохранения. Если Вам необходимо сохранить файл под другим именем, наберите <Ctrl+x> <Ctrl+w>.

Вы можете открыть несколько файлов с помощью команды <Ctrl+x> <Ctrl+f> и каждый из них будет помещен во внутренний буфер Emacs . Для переключения между файлами используйте всё ту же команду <Ctrl+x> <Ctrl+f>.При вводе имени уже открытого файла он не будет перечитан с диска, а просто будет переключен активный буфер. Второй способ переключения между буферами в комбинации клавиш <Ctrl+x> <b>.При этом вам нужно будет ввести имя редактируемого в буфере файла в минибуфер внизу экрана без указания полного пути к нему. Для закрытия текущего буфера используйте <F10> <f> <c>.Для очистки буфера (удаления всего содержимого) <Ctrl+x> <k>

Для отображения списка всех открытых буферов используйте команду <Ctrl+x> <Ctrl+b>. Emacs позволяет одновременно использовать несколько окон редактирования. Для открытия второго окна нажмите <Ctrl+2>.При этом экран разделится на 2 части и обеих будет загружен текущий буфер. Нажмите <Ctrl+x> <o> для перехода в соседнее окно и выполните там команду <Ctrl+x> <Ctrl+f> или <Ctrl+x> <b> для изменения активного буфера. Теперь Вы можете редактировать два разных текста одновременно видя их на экране. Для возврата к одному окну нажмите <Ctrl+x>1 (1 – это единица), а для закрытия текущего окна - <Ctrl+x>0.

Для сохранения нескольких буферов наберите <Ctrl+x><s>. Emacs будет спрашивать Вас перед сохранением каждого буфера, в котором есть

измененные данные. Для того чтобы закрыть буфер без сохранения (убить буфер) <Ctrl+x><Ctrl+k>

Поиск по файлу осуществляется командой <Ctrl+s>.После набора команды в строке минибуфера вы введете искомое слово. При этом Emacs уже найдет его первое вхождение в тексте. Для поиска следующего вхождения нажмите еще <Ctrl+s> раз. Для поиска в обратном направлении, используйте <Ctrl+r>.Когда Emacs достигнет конца (или начала) документа, он подаст звуковой сигнал. Если Вы повторите команду <Ctrl+s> или <Ctrl+r>,то Emacs продолжит поиск с начала (или с конца) документа. Необходимо знать, что Emacs начинает поиск с текущей позиции курсора в буфере. Если необходимо найти все вхождения, то Вам стоит либо перейти к началу документа, либо позволить Emacs'у поиск "по кругу", проигнорировав звуковой сигнал конца (начала) файла.

Замена осуществляется по команде <alt+%> (при этом `%` стоит набирать с шифтом). Emacs будет искать вхождения и спрашивать – пробел является подтверждением, <Del> или <esc> – служат прерыванием.

Встроенная справка Emacs позволяет получать помощь по тем или иным командам. Для получения краткой справки по команде введите: <Ctrl+h> <c> <команда>,так например для получения справки по команде <Ctrl+2> необходимо ввести: <Ctrl+h> <c> <Ctrl+2>.

Для отображения полной справки по команде нужно использовать параметр `<k>` для команды <Ctrl+h>,например <Ctrl+h> <k> <Ctrl+p> открывает в отдельном окне справку по команде <Ctlr+p>.

Уровни инициализации SVR4

Процесс init. Уровни инициализации. >Файл /etc/inittab. Каталог /etc/rc.d.

Ход занятия

Каждый раз включая компьютер, Вы наблюдаете процесс загрузки системы. Сегодня мы с Вами попробуем рассмотреть этот процесс с установленной операционной системой Linux подробнее.

Итак, Вы нажали кнопку питания на системном блоке ПК. Устройства Вашего компьютера проходят автотестирование. Процессор автоматически стартует программу, находящуюся в нулевой ячейке памяти – BIOS (Basic Input Output System - Базовая Система Ввода Вывода), который в первую очередь проверяет исправность всех необходимых (именно необходимых) для работы ПК устройств.

После инициализации BIOS, определения таких устройств как жесткие диски, CDROM и т.п., BIOS передает управление программе загрузчика, находящегося в MBR (Master Boot Record - основная загрузочная запись) на устройстве согласно порядку в настройках CMOS (обычно это CD, HD, USB-устройство или сеть).

В IBM PC совместимых компьютерах размер загрузочной записи ограничен 512 байтами. Это очень мало для полноценной инициализации системы, поэтому обычно загрузчики Linux (LILO или GRUB, как в нашем случае) разделяются на 2 части (stage1 и stage2) и первая из них располагается в MBR. Её задача состоит в том, чтобы инициализировать вторую часть загрузчика.

Вторая часть загрузчика, как правило, предоставляет меню для выбора операционной системы, которая будет загружена, или версии ядра, с которым будет загружен Linux. Но основная задача второй части загрузчика состоит в том, чтобы корректно загрузить в память ПК ядро Linux и передать ему управление.

Загрузившись в память, ядро инициализирует устройства, подключает корневую файловую систему, а также файловую систему /dev , запускает процесс /sbin/init . Так как исполняемый файл init находится в каталоге

/sbin , то не рекомендуется выделять этот каталог в отдельный раздел. Как уже говорилось выше, на момент инициализации ядра примонтирована только корневая ФС. Если доступа к каталогу /sbin не будет, то загрузка системы потерпит фиаско.

Процесс init запускается и анализирует файл /etc/inittab . С этого момента необходимо пояснить, что мы с Вами рассматриваем вариант загрузки Unix SysV. Она отличается от загрузки BSD-like систем. И хотя Linux может работать как с первым, так и со вторым вариантом загрузки, подавляющее большинство дистрибутивов используют инициализацию системы SysV (исключение, например, составляет дистрибутив Gentoo). Хотя BSD-like стиль загрузки проще, загрузка в стиле SysV дает на мой взгляд больше возможностей для точной настройки системы.

В SysV стиле загрузки существует понятие уровней запуска или, правильнее, уровней инициализации системы. По умолчанию, в системе использовано 7 уровней инициализации:

1. 0 - останов системы.
2. 1 - загрузка в однопользовательском режиме
3. 2 - загрузка в многопользовательском режиме без поддержки сети
4. 3 - загрузка в многопользовательском режиме с поддержкой сети
5. 4 - не используется
6. 5 - загрузка в многопользовательском режиме с поддержкой сети и графического входа систему
7. 6 - перезагрузка

Открыв для просмотра файл /etc/inittab , Вы увидите примерно следующий текст (только снабженный комментариями):

```
id:5:initdefault:

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

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
```

```
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Инициализировав основные параметры системы, init запускает скрипты инициализации из каталога /etc/rc.d/

В каталоге /etc/rc.d располагается дерево каталогов, имена которых отличаются лишь одним символом - `rc*.d` , где `*` - номер уровня инициализации, а также каталога init.d , внутри которого находятся управляющие скрипты для системных сервисов. Каждый скрипт принимает минимум 2 параметра – `start` (запуск сервиса) и `stop` (останов сервиса).

Внутри каталогов /etc/rc.d/rc*.d находятся ссылки на скрипты из каталога /etc/rc.d/init.d в виде `S|K#name` . Буква `S` означает что при вызове скрипта процессу init необходимо вызвать скрипт с параметром `"start"` (от слова Start), а буква `K` - с параметром `stop` (от слова Kill). Дальше идет `число` . Все скрипты выполняются в порядке возрастания чисел. Если у двух скриптов одинаковые числа, то такие скрипты будут выполняться в прямом алфавитном порядке.

Далее начинается инициализация виртуальных консолей в таком же порядке, как они описаны в файле /etc/inittab .

- `1` - порядковый номер консоли
- `2345` - номера уровней инициализации, для которых консоль инициализируется
- `respawn` - этот параметр означает, что init должен перезапустить обслуживающий консоль процесс после выхода из сеанса или в случае краха.
- `/sbin/mingetty tty6` – программа (с указанием

параметров), которая будет обслуживать консоль.

Таким образом, Вы легко можете создать свой уровень инициализации (под номером 4 или 7, 8...), просто исправив файл /etc/inittab и создав необходимые ссылки в каталоге /etc/rc.d/rc*.d

Система X Window

Система X Window. Демон X. Запуск X. Скрипт startx . 5-й уровень инициализации.

Ход занятия

1. Давайте для начала окунемся в само понятие X Window (или как его еще называют – X11, X). X11 – это протокол графического интерфейса, основанный на сетевой модели "клиент-сервер".

В качестве сервера может выступать несколько программ, как коммерческих, так и открытых. Долгое время стандартом X-сервера для Linux был XFree86 (о котором мы с вами и поговорим), однако сейчас довольно популярным X-сервером становится XOrg, обладающим намного большими возможностями.

Программы с графическим интерфейсом выступают в качестве клиентов для X-сервера.

Сервер X умеет рисовать на экране точки и простейшие геометрические фигуры, опрашивать устройства ввода (клавиатура, мышь и т.п.) и сообщать об этом клиенту в виде "пользователь нажал на правую кнопку мыши в точке экрана такой-то", а клиенты давать ему команды, типа "нарисуй прямоугольник с координатами такими-то", "Назови-ка мне текущие координаты мыши", причем все эти команды способны передаваться не только на локальной машине, но и по сети! Таким образом, для работы самой программы, которая в большинстве случаев выполняет вычисления (или другую полезную работу, в отличие от сервера), можно использовать мощную рабочую станцию, а вот вывод на экран производить на совсем другом ПК, с которого будет также происходить и управление программой (ввод с клавиатуры, мыши, светового пера и т.п.).

X-сервер может обслуживать несколько клиентов одновременно. Так каждое открытое на вашем экране окно – отдельный X-клиент.

Однако X-сервер предоставляет окружению верхнего уровня полную свободу в отношении оформления окон, точнее сказать X-сервер просто

их не прорисовывает. Всем этим управляют особые программы - оконные менеджеры, которых существует бесчисленное множество. Наиболее популярными оконными менеджерами для Linux являются `kwm` (используется в KDE), `metacity` (используется в GNOME), `twm` (стандартный для XFree86), `icewm`, `fluxbox`, `openbox`, `fvwm` , `WindowMaker`, `Enlightenment` и другие. Каждая из этих программ предоставляет различные внешние виды, удобства использования и конфигурации, и каждый из пользователей волен выбирать их на свой вкус.

Основными свойствами оконного менеджера можно считать:

- возможность изменения размера окна
- перемещения окон
- переключение между приложениями
- сворачивание приложения (в заголовок, значок или на какую-либо панель)

Другой важной возможностью оконного менеджера является управление фокусировкой мыши – каждый из них должен иметь способ выбора активного окна, а также визуально выделить такое окно. Самым популярным является метод "фокус по щелчку". При использовании этого метода фокус получает то окно, по которому был произведен щелчок мыши. Подобный метод используется в Microsoft® Windows®. Его так же поддерживают большинство оконных менеджеров для X.

Другим распространенным методом является "фокус, следующий за мышью". При использовании этого метода, активным считается то окно, над которым находится мышь. При этом, если мышь попадает на пустое место на экране, то фокус теряют все окна.

Еще один из методов называется "нечеткий фокус". Он аналогичен "фокусу, следующему за мышью" за исключением того, что окно потеряет фокус только тогда, когда фокус получит другое окно.

Однако оконными менеджерами не обеспечивается инструментальная база для построения приложений. В X эта инструментальная база называется "widget". Это непереводимый термин X, который составлен из двух слов – window (окно) и gadget (приспособление). Виджеты – это

кнопки, менюшки, выпадающие списки и т.п. Обычно виджеты собраны в библиотеки. Наиболее часто используемые из них такие:

- `OpenMotif` – библиотека, аналог известнейшей библиотеки Motif® на которой построена культовая графическая среда для Unix. Набор виджетов из библиотеки *`Motif®`* стал прародителем интерфейса для Microsoft® Windows®.
- `Xlib` – базовая библиотека виджетов для X. Примером приложений для этой библиотеки могут служить приложения xcalc, xterm, xfontsel.
- `GTK/GTK2/GTK+(GIMP ToolKit)` – набор библиотек для интерфейса среды окружения рабочего стола GNOME. Эти библиотеки созданы и поддерживаются проектом GNU. Примером приложений, написанных для GTK могут служить Xchat, Gimp и многие другие.
- Qt(v1, 2, 3, 4) – кроссплатформенная (Linux/FreeBSD/OpenBSD/MacOS/Windows, а также большинство коммерческих версий Unix) графическая библиотека, написанная компанией Trolltech. Сейчас библиотека Qt распространяется на правах GPL. На основе этой библиотеки построена среда окружения рабочего стола KDE, а также множество независимых приложений. Примером приложений на Qt могут служить QtDesigner, kcalc, kate и другие.

  В Linux-системе XWindow обслуживает демон X (/usr/bin/X11/X). В случае с XFree86 он использует конфигурационный файл /etc/X11/XF86Config.

Формат файла имеет следующий вид:

```
Section "ServerLayout"
        Identifier     "XFree86 Configured"
        Screen      0  "Screen0" 0 0
        InputDevice    "Mouse0" "CorePointer"
       InputDevice    "Keyboard0" "CoreKeyboard"
EndSection
```

Эта часть конфигурационного файла описывает, дисплей с каким номером должен использовать сервер, а также используемые

устройства ввода.

```
Section "Files"
        RgbPath      "/usr/X11R6/lib/X11/rgb"
        FontPath     "unix/:7100"
EndSection
```

Здесь описывается путь к набору палитр для вывода информации (RgbPath), а также путь к каталогам со шрифтами (FontPath). Строка "unix/:7100" говорит о том, что запущен сервер шрифтов X (xfs) и он будет доступен по локальному unix-сокету с номером 7100.

```
Section "Module"
        Load  "GLcore"
        Load  "dbe"
        Load  "dri"
        Load  "extmod"
        Load  "fbdevhw"
        Load  "glx"
        Load  "pex5"
        Load  "record"
        Load  "xie"
        Load  "v4l"
        Load  "type1"
EndSection
```

В разделе Module описываются расширения, которые будет использовать сервер.

```
Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "keyboard"
        Option       "Protocol" "Standard"
        Option       "XkbModel" "pc104"
        Option       "XkbRules" "xfree86"
        Option       "XkbLayout" "us,ru(winkeys)"
        Option       "XkbOptions" "grp:ctrl_shift_toggle,grp_led:scroll"
EndSection
Section "InputDevice"
```

```
        Identifier  "DevInputMice"
        Driver      "mouse"
        Option      "Protocol" "IMPS/2"
        Option      "Device" "/dev/input/mice"
        Option      "ZAxisMapping" "4 5"
        Option      "Emulate3Buttons" "no"
EndSection
```

Секция InputDevice (устройство ввода) описывает клавиатуру, раскладку, а также тип и свойства мыши. Мышь и клавиатуру необходимо описывать в разных секциях.

```
Section "Monitor"
#1024x768 @ 70.1 Hz, 56.5 kHz hsync
 Identifier   "Monitor"
 VendorName   "SAM"
 ModelName    "Samsung SyncMaster 550(M)s"
 HorizSync    30.0 - 62.0
 VertRefresh  50.0 - 121.0
 ModeLine    "1024x768" 75.0 1024 1048 1184 1328 768 771 777 806 -hsync
 ModeLine     "800x600" 50.0 800 856 976 1040 600 637 643 666 +hsync +vs
 ModeLine     "640x480" 31.5 640 680 720 864 480 488 491 521
EndSection
```

В разделе описания свойств монитора описываются поддерживаемые монитором режимы, а также название монитора и производитель.

```
Section "Device"
        Identifier  "nVidia Corporation|NV4 [RIVA TNT]"
        Driver      "nvidia"
        BoardName   "RIVA TNT"
        BusID       "PCI:1:0:0"
EndSection
```

В разделе Device описывается используемый видеоадаптер, а также параметры видеовывода, которые он поддерживает, обслуживающий драйвер.

```
Section "Screen"
        Identifier "Screen0"
```

```
        Device     "nVidia Corporation|NV4 [RIVA TNT]"
        Monitor    "Monitor"
        DefaultDepth     24
        SubSection "Display"
            Depth     16
            Modes    "1024x768" "800x600" "640x480"
        EndSubSection
        SubSection "Display"
            Depth     24
            Modes    "1024x768" "800x600" "640x480"
        EndSubSection
  EndSection
```

А вот секция Screen как раз описывает выбранный пользователем режим. Сначала указываются используемые видеоадаптер и монитор, а также глубину цветовой палитры (DefaultDepth). При запуске сервер X будет пытаться инициализировать работу в режимах (Modes), указанных в подсекциях (subsection) Display начиная с наибольшего.

2. Для запуска X в Linux могут быть использованы два разных метода. Первый из них используется в пятом уровне инициализации – скрипт prefdm (/etc/X11/prefdm). При этом будет использоваться какой-либо менеджер входа в систему (xdm, kdm, gdm или другой), с помощью которого вы сможете выбрать используемую рабочую среду или оконный менеджер. Однако существует и другой способ запуска системы. Если Вы загрузитесь с уровнем инициализации, не использующим графический вход в систему (например 3 или 2), то запустить X можно будет скриптом startx (/usr/bin/X11/startx). Используемый по умолчанию менеджер окон (или среда окружения рабочего стола) описываются в файле Xclients (/etc/X11/xinit/Xclients). Для установки своего клиента, установите переменную скрипта `PREFERRED` в значение, которое описывает строку запуска оконного менеджера (среды окружения рабочего стола), например:

```
  PREFERRED=startkde          #запуск KDE
```

или

```
  PREFERRED=gnome-session          #запуск GNOME
```

или

```
PREFERRED=icewm                #запуск оконного менеджера icewm
```

Сетевое администрирование Linux. Сетевая модель OSI

Структура модели OSI. Семейство протоколов TCP/IP.

Ход занятия

1. Идея передачи данных по сети на большие расстояния возникла очень давно. Первые компьютерные сети работали в режиме разделения времени, когда подключения одного компьютера к другому было возможно только в определенный временной интервал. Такие сети были основаны на технологиях компании IBM, выпускающей в то время большие компьютеры – мэйнфрэймы.

Появление локальных сетей в начале 80-х годов прошлого века с появлением в конце 80-х интернет произвело поистине революцию в мире информации. Но вместе с появлением новых возможностей, появилась и задача управления сетями. Unix-системы, традиционно занимавшие роль систем, использующих возможности сети оказались в выигрышном положении, а специалисты, знакомые с их возможностями – высокооплачиваемыми и всегда требуемыми кадрами.

В 1984 году международная организация по стандартизации (ISO) создала эталонную модель взаимодействия открытых систем (или OSI – Open System Interconnection). Модель решает задачу перемещения данных по сети путем распределения ее по 7 уровням, которыми управлять легче, нежели единой целостной системой.

Уровни более или менее независимы друг от друга, так что задачи, связанные с каждым из них тоже могут выполняться независимо. К этому тезису мы еще вернемся.

Итак, вот эти уровни:

7 – уровень приложений: это ближайший к пользователю уровень OSI . В задачи, выполняемые на этом уровне входит определение доступности ресурсов, аутентификации (определение подлинности) пользователя, отображения информации и т.п. В качестве примеров можно привести протоколы FTP(file transfer protocol), SMTP (simple mail transfer protocol), HTTP (hyper text transfer protocol) .

6 – уровень представлений: обеспечивает различные кодирования и преобразования, которым подвергаются данные приложения. Популярные протоколы уровня представлений это MPEG – стандарт сжатия и кодирования видео , GIF, JPEG, PNG – стандарты сжатия и кодирования графических изображений, SSL – защищенные соединения.

5 – сеансовый уровень: на этом уровне устанавливаются сеансы обмена данными, происходит их управление и завершение. Наиболее известным протоколом этого уровня являемся протокол SMB (server message block) – передача файлов в сетях Windows.

4 – транспортный уровень: принимает данные от более высокого уровня и разбивает их на части для передачи по сети. Как правило, транспортный уровень "отвечает" за доставку и правильную сборку данных. Именно на этом уровне происходит управление потоками данных, передаваемых по сети.

Наиболее известные протоколы транспортного уровня это TCP, UDP, SPX .

3 – сетевой уровень определяет сетевой адрес, отвечает за маршрутизацию пакетов. Сетевой уровень определяет логическое устройство сети.

Известнейшие протоколы сетевого уровня: IP, X25, IPX .

2 – канальный уровень: обеспечивает надежную передачу данных в физической сети. Спецификации канального уровня определяют важнейшие характеристики сети, такие как размер пакета, пропускная способность и т.п. Также на канальном уровне определяется физическое (MAC - управление доступом к носителю) устройство сети посредством присвоения ему уникального MAC-адреса (или без таковых в соединениях точка-точка).

Наиболее известные протоколы канального уровня: Ethernet, Token Ring, PPP, DSL, ATM и др.

1 – физический уровень модели OSI : регламентирует физические, механические процедурные спецификации. Проще говоря, физический уровень определяет среду передачи – витая пара, медный провод,

оптоволоконный кабель и др.

Говоря ранее о независимости разных уровней друг от друга я имел в виду следующее: протокол Ethernet (канальный уровень) может работать как на витой паре, так и на оптоволоконном или коаксиальном кабеле, и вместе с тем на базе Ethernet может быть построена сеть IP, IPX или, например, AppleTalk .

2. Поскольку в современном мире профессия сетевого администратора де-факто связана с работой в сети Интернет, то и изучать основы сетевого администрирования мы будем с вами на примере семейства протоколов TCP/IP версии 4 (в разработке находится протокол IPv6 , переход на который уже начался в странах восточной и юго-восточной азии в качестве эксперимента). Наша сегодняшняя задача состоит в том, чтобы познакомиться с протоколами, входящими в семейство TCP/IP и провести соответствие между ними и эталонной моделью. Полученные сегодня знания будут являться для Вас основным фундаментом для дальнейшего изучения сети.

В состав семейства протоколов TCP/IP входят 8 протоколов, не считая сторонних протоколов маршрутизации. Лояльность разработчиков этого семейства протоколов позволила использовать любой протокол маршрутизации, однако в стандарте TCP/IP определен и собственный.

Начнем с нижнего уровня и перечислим их:

- ARP – Address Resolution Protocol – протокол преобразования адресов. Обеспечивает преобразование сетевых адресов в адреса физических устройств MAC. Работает одновременно на трех уровнях – канальном, сетевом и уровне приложений.
- RARP – Reverse Address Resolution Protocol – протокол обратного преобразования адресов. Обеспечивает преобразования MAC-адреса в IP-адрес . Для работы требует наличие сервера RARP с таблицей преобразования. Чаще всего используется для загрузки бездисковых рабочих станции, которые при запуске не знают своего IP-адреса . Работает одновременно на двух уровнях – канальном и сетевом.
- - DHCP – Dynamic Host Configure Protocol – протокол динамической конфигурации хоста. Позволяет присваивать IP-адреса , маршрут

по умолчанию и некоторую другую сетевую информацию о сети IP-устройствам . Работает одновременно на двух уровнях – канальном и сетевом.

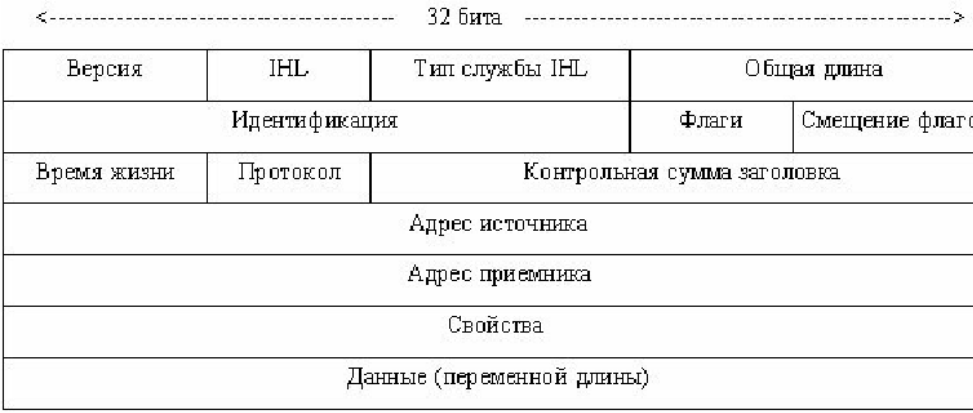
- - IP – Internet Protocol – протокол сетевого уровня, который содержит информацию об адресе логического устройства сети и некоторую информацию о маршрутизации пакетов в сети. Является основным сетевым протоколом в наборе протоколов TCP/IP . Имеет две основные функции – передачу дейтаграмм (блоков данных) по сети с наименьшими затратами без подтверждения соединения и обеспечение фрагментации (разбивки) пакетов и последующей сборки для поддержки передачи протоколу канального уровня с различным максимальным размером блоков передаваемых данных.
- ICMP - Internet Control Message Protocol – протокол контроля сообщений в сети Internet – обеспечивает создание и отправку пакетов с отчетами об ошибках и другой информации об обработке IP-пакетов , а также контроля доступности узлов в сети. Работает на сетевом уровне модели OSI .
- IRDP – ICMP Router-Discovery Protocol – ICMP-протокол обнаружения маршрутизатора: использует объявления и запросы маршрутизаторов, чтобы определить адреса маршрутизаторов соседних сетей. Работает на сетевом уровне.
- UDP – User Datagram Protocol – протокол передачи блоков данных пользователя: протокол транспортного уровня, не требующий подтверждения соединения. Имеет систему портов, позволяющих различать приложения, работающие на одном устройстве.
- TCP – Transfer Control Protocol – протокол управления передачей: протокол транспортного уровня модели OSI с подтверждением соединения. Кроме системы портов вводит также понятие соединения, позволяющего одновременную работу портов в режимах "один ко многим".

Сетевое администрирование Linux. Протокол IP

Структура пакета IP. Структура IP-адреса. Подсети. ifconfig и настройка протокола IP. Маршрутизация. Автономные области. M9. Команда route.

Ход занятия

1. IP (Internet Protocol) – протокол сетевого уровня, который содержит информацию об адресации и некоторую управляющую информацию для маршрутизации пакетов. Протокол описан в запросе на комментарий 791 (RFC 791).



Версия – Версия используемого протокола IP

IHL (IP header length) –длина IP-заголовка. Длина заголовка в 32-разрядных блоках

Тип службы – определяет управление протоколом верхнего уровня (TCP или UDP) и присваивает важность пакету.

Общая длина – Длина всего ip-пакета в байтах, включая данные и заголовок.

Идентификация – целое уникальное число, определяющее пакет. Используется при сборке фрагментированных пакетов.

Флаги – Состоит из 3-х бит. Первый бит определяет, может ли пакет

быть фрагментирован, а второй – является ли пакет последним в серии фрагментированных. Третий бит не используется.

Смещение флагов – содержит значение позиции данных фрагмента относительно начала данных. Используется только в фрагментированных пакетах.

Время жизни – Счетчик, который постепенно уменьшается до нуля (на единицу при прохождении каждого маршрутизатора), после чего пакет уничтожается во избежание бесконечной передачи по сети.

Протокол – Протокол верхнего уровня (TCP или UDP)

Контрольная сумма заголовка – Помогает убедиться в целостности пакета.

Адрес источника – определяет узел-отправитель

Адрес приемника – определяет узел-получатель

Свойства – Позволяет IP определять различные свойства, например безопасность

Данные – Информация верхнего уровня.

Протокол IP как протокол сетевого уровня неразрывно связан с понятием адресации. Адрес IP – это 32-разрядный адрес, который содержит 4 группы по одному байту, обычно записываемых в 10-тичном виде через точку. Каждая группа называется октетом. Минимальное значение октета – 0, максимальное – 255.

Протокол IP определяет также понятие подсети. Это группа ip-адресов, имеющая общую маршрутизацию.

Подсети определяются масками. Маска – это часть сетевого адреса, определяющая какие биты адреса относятся к сети, а какие – к хосту. Биты маски, установленные в 1 определяют сеть, а в 0 – хост.

Например:

```
192.168.2.31/255.255.255.0
```

Маска подсети 255.255.255.0 в двоичном виде будет выглядеть:

```
11111111 11111111 11111111 00000000
```

Отсюда можно сделать вывод, что для того чтобы найти адрес 192.168.2.31 нужно найти сеть 192.168.2.0, а в ней хост 31.

IP-адреса делятся на пять классов – A, B, C, D и E. Для коммерческого использования предназначены только первые 3.

Таблица 5.1.

Класс	Маска	Количество битов, сеть/хост	Максимально количество хостов
A	255.0.0.0	8 бит на сеть/24 бита на хост	16777214 (2^{24}-2)*
B	255.255.0.0	16 бит на сеть/16 бит на хост	65534 (2^{16}-2)
C	255.255.255.0	24 бит на сеть/8 бит на хост	254 (2^{8}-2)

* - один адрес зарезервирован как широковещательный, и один – для сети.

Существуют также специально выделенные диапазоны сетей для использования в локальных сетях. Это так называемые фэйковые сети (от англ. fake – обманывать, также их называют приватными, серыми адресами). Эти сети не маршрутизируются в сети интернет. Выделены диапазоны для 3 классов сетей:

Таблица 5.2.

Класс	Диапазон
A	10.0.0.0 – 10.255.255.255 (1 сеть класса A)
B	172.16.0.0 – 172.31.255.255 (16 сетей класса B)
C	192.168.0.0 – 192.168.255.255 (255 сетей класса C)

2. Настройка протокола IP в Linux может выполняться как с помощью встроенных средств, таких как netconf от RedHat, так и вручную.

Для отображения параметров протокола IP используется команда /sbin/ifconfig. С помощью этой же команды можно настроить устройство или добавить второй ip для карты:

```
bash-2.05b#  /sbin/ifconfig eth0 192.168.2.31 netmask 255.255.255.0
bash-2.05b# /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:C0:26:2C:AC:D1
          inet addr:192.168.2.31  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:147329 errors:0 dropped:0 overruns:0 frame:0
          TX packets:47207 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          RX bytes:96478376 (92.0 Mb)  TX bytes:8043931 (7.6 Mb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:10514 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10514 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          RX bytes:9712961 (9.2 Mb)  TX bytes:9712961 (9.2 Mb)
bash-2.05b# /sbin/ifconfig eth0 add 192.168.2.253 netmask 255.255.255.0
bash-2.05b# /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:C0:26:2C:AC:D1
          inet addr:192.168.2.31  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:148126 errors:0 dropped:0 overruns:0 frame:0
          TX packets:47781 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          RX bytes:96843116 (92.3 Mb)  TX bytes:8103494 (7.7 Mb)

eth0:0    Link encap:Ethernet  HWaddr 00:C0:26:2C:AC:D1
          inet addr:192.168.2.253  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:148126 errors:0 dropped:0 overruns:0 frame:0
          TX packets:47781 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          RX bytes:96843116 (92.3 Mb)  TX bytes:8103494 (7.7 Mb)
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:10514 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10514 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          RX bytes:9712961 (9.2 Mb)  TX bytes:9712961 (9.2 Mb)
```

Однако в RedHat-системах принято настраивать протокол IP с помощью системы стартовых скриптов на основе файлов настроек:

/etc/sysconfig/network – основной конфигурационный файл сети. В нем описан сам факт использования сети, имя хоста, маршрут по умолчанию и адреса DNS:

```
bash-2.05b# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=WebMedia
GATEWAY=192.168.2.2
DNS1=192.168.2.2
DNS2=192.168.2.4
```

/etc/sysconfig/network-scripts/ifcfg-<псевдоним устройства> - описывает параметры сетевого устройства:

```
bash-2.05b# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
ONBOOT=yes
IPADDR=192.168.2.31
NETMASK=255.255.255.0
NETWORK=192.168.2.0
BROADCAST=192.168.2.255
```

или, при использовании протокола динамической конфигурации:

```
bash-2.05b# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Нужно учесть, что в выводе команды ifconfig вы будете видеть название устройства, указанного в названии файла, а реально будет использовано

устройство, указанное в файле. Причем эти названия не обязательно должны совпадать. Например, в вывод ifconfig можно установить ppp0, хотя реально будет использоваться eth0.

3. Маршрутизация в интернет построена на данных о IP-подсетях, а также о группах подсетей, принадлежащих крупным магистральным провайдерам. Такие группы адресов называются автономными областями. Маршрутизация может быть статической, на основе заранее созданных человеком маршрутов и динамической, когда маршруты создаются протоколами маршрутизации, такими как RIP (Routing Information Protocol – протокол информации и маршрутизации), OSPF (Open Short Path First – использовать короткий путь сначала), BGP (Border Gateway Protocol – протокол граничного шлюза) и др. Их рассматривать мы не будем. У маршрутизатора есть набор правил, определяющих сетевой интерфейс, на который может быть отправлен пакет в зависимости от адреса получателя, а также маршрут по умолчанию, куда отправляются пакеты, не соответствующие общим правилам, например:

- для сети 192.168.2.0/255.255.255.0 отправить в eth0
- для сети 214.54.0.0/255.255.0.0 отправить в eth1
- маршрут по умолчанию wan0

Таким образом, когда маршрутизатор получит пакет с адресом получателя 192.168.2.31, то он отправит его в интерфейс eth0, а если он получит пакет для 80.92.30.1, то отправит его в интерфейс wan0. Может быть также, что вместо маршрута-устройства задан адрес сетевого шлюза. В этом случае маршрутизация пакетов будет осуществляться с помощью шлюза, а хост, у которого указан шлюз сможет напрямую адресовать только хосты собственной подсети. Совсем другой тип маршрутизации применяется в автономных областях. В этом случае на магистральных развязках пакеты для всех подсетей области отправляются на граничный маршрутизатор области, а он уже выполняет внутриобластную маршрутизацию. Как центральная магистральная развязка в России используется группа маршрутизаторов магистральных провайдеров, таких как РосТелеКом, ТрансТелеКом, МТУ Интел, географически расположенная на международной телефонной станции М9 в г. Москве. Именно из этой точки осуществляется трансляция потоков данных между крупными российскими магистральными провайдерами, а также зарубеж. Для

настройки статической маршрутизации в Linux используется команда `/sbin/route` .

```
[gserg@WebMedia gserg]$ /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0     *               255.255.255.0   U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         ns.edu.vologda. 0.0.0.0         UG    0      0        0 eth0
```

Введенная без параметров она показывает таблицу маршрутизации, используемую на ПК/сервере. Использование команды описано подробно в man-странице. Я же приведу несколько примеров:

1) Просмотр установленных маршрутов

```
bash-2.05b# /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0     *               255.255.255.0   U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         ns.edu.vologda. 0.0.0.0         UG    0      0        0 eth0
```

2) Добавление маршрутов

```
bash-2.05b# /sbin/route add 192.168.1.0 gw 192.168.2.1
bash-2.05b# /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     192.168.2.1     255.255.255.255 UGH   0      0        0 eth0
192.168.2.0     *               255.255.255.0   U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         ns.edu.vologda. 0.0.0.0         UG    0      0        0 eth0
```

3)Удаление маршрутов

```
bash-2.05b# /sbin/route del 192.168.1.0
```

```
bash-2.05b# /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0     *               255.255.255.0   U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         ns.edu.vologda. 0.0.0.0         UG    0      0        0 eth0
bash-2.05b#_
```

Для создания маршрутов, которые впоследствии будут использоваться при загрузке Вы можете использовать в RedHat-based системах файл /etc/sysconfig/static-routes:

```
eth0 net 192.168.3.0 netmask 255.255.255.0 gw 192.168.2.4
 eth0 net 192.168.1.0 netmask 255.255.255.0 gw 192.168.2.1
```

Формат файла следующий:

```
интерфейс пробел параметры_команды _/sbin/route
```

Сетевое администрирование Linux. Протокол UDP

Структура полного адреса в протоколе UDP. Понятие UDP-портов. Структура UDP-пакета. Распространенные UDP-сервисы. DNS. Bind.

Ход занятия

1. UDP (User Datagram Protocol , протокол передачи блоков данных пользователя) – протокол транспортного уровня, не требующий подтверждения соединения, принадлежащий семейству протоколов интернет TCP/IP.

<---------- 32 бита ---------->

Порт источника	Порт приемника
Длина	Контрольная сумма
Данные	

По сути, протокол UDP является прослойкой между IP и протоколами верхнего уровня, предоставляя возможность нескольким приложениям работать на одном компьютере с помощью системы портов.

Порты представляют собой программные интерфейсы, которые можно описать на примере работы почтового отделения.

Каждое почтовое отделение имеет индекс – в нашем примере IP-адрес , а каждый конкретный получатель – улицу, квартиру и дом – это порт. Если представить пересылку письма, то сначала письмо доставляется от отправителя в почтовое отделение (инкапсулируется, т.е. встраивается в пакет IP). После этого письмо доставляется на почтовое отделение получателя по индексу (доставляется IP-пакет на хост-получатель). Почтальон разносит письма в почтовые ящики и доставляет наше письмо получателю (подсистема UDP хоста-получателя декапсулирует UDP-пакет из IP и отправляет приложению-получателю).

2. UDP – транспортный протокол для нескольких известных служб, таких как:

NFS (Network FileSystem – сетевая файловая система) – протокол уровня приложений для передачи файлов по локальной IP-сети . Каталоги NFS

, открытые для общего доступа могут быть смонтированы (как в UNIX системах, так и в системах на базе Windows NT/2000/XP/2003) как локальные каталоги, с которыми можно работать.

SMB (Server Message Block – блок сообщений сервера) – протокол уровня приложений для передачи файлов в сетях Windows.

SNMP (Simple Network Managment Protocol – простой протокол управления сетью) – протокол уровня приложений, использующийся для сбора информации о состоянии и управлении активными сетевыми устройствами, такими как маршрутизаторы, программируемые коммутаторы, серверы и др.

TFTP (Trivial File Transfer Protocol – простейший протокол передачи файлов) – протокол уровня приложений, использующийся, чаще всего, для приема/передачи конфигурационной и другой информации с активных сетевых устройств.

3. DNS (Domain Name Service – служба доменных имен) – протокол уровня приложений, позволяющий преобразовывать символьные имена интернет-хостов в IP-адреса .

Наиболее распространенным DNS-сервером на сегодняшний день признан BIND (Berkley Internet Name Daemon – Демон интернет-имен университета Беркли). Демон предоставляет стандартный DNS-сервис на портах UDP-53 (обработка запросов клиентов), TCP-53 (пересылка информации о зонах между серверами имен).

Служба имен – это распределенная иерархическая (поделенная на уровни) база данных в интернет.

Имена DNS подразделяются на имена хостов и имена доменов разного уровня:

www.volnet.ru – здесь www – имя компьютера, volnet – домен второго уровня, ru – домен первого уровня.

Домен – обозначение условной группы компьютеров, объединенных по признакам принадлежности к организации, сети, географическому положению или сфере деятельности.

Существуют несколько международных доменов первого уровня:

- com – коммерческие организации
- edu – образовательные учреждения
- org – некоммерческие организации
- mil – военные организации
- gov – государственные организации
- biz – бизнес
- int – международные организации и проекты.

Рассмотрим работу DNS на примере поиска доменного имени ссылка: http://www.volnet.ru.

Клиент сервера Dialup.mtu.com отправляет запрос на поиск вышеуказанного сервера. Сервер DNS , принимает запрос и ищет соответствие имени и ip-адреса у себя в кэше. Если не находит, то отправляет запрос в головной сервер домена верхнего уровня (в данном случае – серверу домена com). И в этом случае сервер пытается сначала найти соответствие у себя в кэше. Если не найдено, то сервер пересылает запрос головному вверх по иерархии, пока не дойдет до домена первого уровня. Там возможна пересылка между разными доменами первого уровня, между com и ru и потом происходит спуск вниз по иерархии вплоть до сервера запрашиваемого домена. На всех уровнях результат запроса запоминается в кэше, чтобы в следующий раз не перезапрашивать его заново.

За настройку DNS отвечают в системе несколько файлов.

/etc/hosts – файл, в котором прописываюся статические имена хостов. Эти имена не передаются по сети и действуют только в пределах данного хоста.

```
[root@ns root]# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost
80.92.3.2       ns.edu.vologda.ru       ns
80.92.3.3       meson-vol-c0.volnet.ru
192.168.2.4     email.edu.vologda.ru
```

```
192.168.2.4     server.schoolm.edu.vologda.ru
192.168.2.130   server-2003
192.168.2.31    webmedia
```

/etc/resolv.conf – файл конфигурации клиентов DNS . Именно в этом файле прописываются адреса серверов DNS , с которыми работают сетевые клиенты.

```
[root@ns root]# cat /etc/resolv.conf
search volnet.ru edu.vologda.ru
nameserver 80.92.3.2
nameserver 192.168.2.2
nameserver 80.92.0.10
```

/etc/named.conf – файл конфигурации сервера BIND . В нем описываются основные параметры сервера, такие как каталоги хранения файлов зон, каталоги журналирования, а также определяются сами зоны DNS и узлы, имеющие право на получение и изменение зон.

```
[root@ns root]# cat /etc/named.conf
options {
        directory "/var/named";
        allow-transfer{
                213.24.34.4;
                213.24.35.193;
                213.24.34.14;
                213.247.150.121;
                213.158.26.84;
                192.168.2.4;
        };
};
logging{
        channel a_main{
                file "/var/log/named/named.log";
                print-time yes;
                print-category yes;
                print-severity yes;
        };
        category default{
```

```
                a_main;
                default_debug;
        };
};
zone "." {
        type hint;
        file "named.ca";
};
zone "edu.vologda.ru"{
        type master;
        file "edu.vologda.ru";
        allow-transfer{
                213.24.34.4;
                213.24.34.14;
                80.92.0.10;
                213.158.26.84;
                213.247.150.119;
                192.168.2.4;
        };
};
zone "volnet.ru"{
        type master;
        file "volnet.ru";
        allow-transfer{
                213.24.35.193;
                213.158.26.84;
                80.92.0.10;
                192.168.2.4;
        };
        allow-update{
                213.24.35.193;
        };
};
```

Файлы зон выглядят следующим образом:

```
@       IN      SOA     ns.edu.vologda.ru.      hostmaster.edu.vologda.ru. (
                        2002031128 ; serial
                        14400 ; refresh
```

```
                900 ; retry
                1209600 ; expire
                86400 ; default_ttl
                )
@               IN      MX      5       ns.edu.vologda.ru.
@               IN      NS      ns.edu.vologda.ru.
@               IN      NS      ns.icb.vologda.ru.
mail            IN      CNAME   ns.edu.vologda.ru.
proxy           IN      CNAME   ns.edu.vologda.ru.
proxy2          IN      A       80.92.3.134
meson           IN      CNAME   ns.edu.vologda.ru.
www             IN      CNAME   ns.edu.vologda.ru.
meson-vol-c0    IN      A       80.92.3.131
```

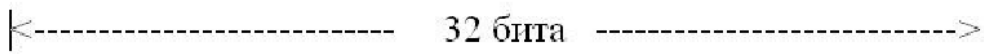
Сетевое администрирование Linux. Протокол TCP. Занятие первое

Структура полного адреса в протоколе TCP. Понятие TCP-соединения. Структура TCP-пакета. Распространенные TCP-сервисы. Telnet: xinetd, in.telnetd. SSH: sshd.

Ход занятия

1. TCP (Transmission Control Protocol , протокол управления передачей) обеспечивает надежную передачу данных в среде IP. TCP относится к транспортному уровню OSI. Этот протокол предоставляет такие услуги как потоковая передача данных, надежность, эффективное управление потоком, дуплексный режим.

Формат пакета TCP:



Порт источника			Порт приемника
Порядковый номер			
Номер подтверждения			
Сдвиг данных	Резервные	Флаги	Окно
Контрольная сумма			Указатель срочности
Параметры			
Данные			

Порт источника и порт приемника – точки, в которых процессы верхнего уровня принимают услуги TCP.

Порядковый номер – Обычно номер первого байта в сообщении. Может также использоваться для обозначения исходного порядкового номера в передаче.

Номер подтверждения – порядковый номер следующего байта данных, который ожидает получить приемник.

Сдвиг данных – Число 32-разрядных слов в заголовке TCP.

Резервные – Область, зарезервированная для использования в будущем.

Флаги – Различная управляющая информация, в том числе биты SYN, ACK и FIN.

Окно – Размер приемного окна (буфера памяти) приемника.

Контрольная сумма – Показывает, не был ли заголовок поврежден при передаче.

Указатель срочности – Указывает на первый байт срочных данных в пакете.

Параметры – Различные дополнительные параметры TCP.

Данные – Информация верхнего уровня.

В протоколе TCP принято понятие потока – последовательности битов переменной длины, передающихся между двумя объектами. Именно поток является единой и неделимой частью TCP-соединения. Одним словом, понимать под одним целым необходимо связку IP:TCPпорт-IP2:TCPпорт2, а не IP:TCPпорт, как в протоколе UDP. Это обозначает, что несколько клиентов могут одновременно работать с одной службой на одном сервере и одном порту, так как IP1:TCPпорт1-IP2:TCPпорт2 и IP1:TCPпорт1-IP3:TCPпорт3, будут различными потоками, не пересекающимися друг с другом.

Установка соединения в протоколе TCP происходит по механизму трёхэтапной синхронизации (three-way handshake) . Этот механизм синхронизирует обе стороны, позволяя им согласовать начальные порядковые номера. Он также обеспечивает готовность сторон к приему/передаче, чтобы избежать передачи лишних пакетов при установке и после разрыва соединения.

Первый хост (А) открывает соединение, посылая второму хосту (Б) пакет с начальным номером соединения и установленным флагом SYN.

Хост Б получает SYN-пакет, записывает номер (X) отвечает пакетом с порядковым номером X+1 и установленными битами SYN и ACK. Также хост Б указывает номер подтверждения (Y). Если этот номер равен, например, 40, то это означает что хост принял 39 байт и ожидает 40-й байт. Эта технология называется подтверждением передачи данных. Затем хост А подтверждает прием всех байтов, посланных хостом Б, указывая номер подтверждения Y+1 и устанавливая флаг ACK. Только после этого начинается передача данных.

Протокол TCP обеспечивает надежную передачу данных за счет следующей технологии: если после отправки пакета через заданный промежуток времени (таймаут) не придет подтверждения передачи (пакета с установленным флагом ACK), то пакет будет отправлен снова. Эта технология называется подтверждением приема и повторной передачей (Positive Acknowledgment and Retransmission, PAR).

Присваивая каждому пакету порядковый номер, PAR позволяет хостам отслеживать пакеты, потерянные или дублированные вследствие сетевых задержек или сбоев.

Скользящее окно TCP позволяет использовать пропускную способность сети более эффективно, поскольку с его помощью хосты могут отправлять несколько пакетов не дожидаясь подтверждения. Окно измеряется в байтах, которые может послать отправитель во время ожидания подтверждения передачи данных. Размер окна определяется во время установки соединения, но может изменяться во время передачи. Если размер окна равен 0, то передача данных запрещена.

Предположим, что TCP-отправителю надо переслать с помощью скользящего окна последовательность байт (пронумерованных от 1 до 10) получателю с размером окна 5. Отправитель помещает в окно первые 5 байт, передает и ждет подтверждения. После того, как он получит пакет с флагом ACK и номером подтверждения, равным 6, то отправитель передает байты с 6 по 10. Получатель отправляет ACK с номером подтверждения равным 11, и указывает что размер окна равен 0. В этом случае отправитель будет ожидать пакета ACK с номером подтверждения 11 и ненулевым размером окна перед началом передачи.

2. Распространенные TCP-сервисы, это практически все, что средний пользователь знает сегодня об интернет:

WWW (World Wide Web, всемирная паутина) – самый распространенный сервис в Интернет, построенный на протоколе уровня приложений HTTP. Использует порт TCP80.

FTP (File Transfer Protocol – протокол передачи файлов) – используется для передачи файлов по сети и организации интернет-файлархивов. Использует порты TCP20-21

SMTP (Simple Mail Transfer Protocol – простой протокол передачи почты) – используется для отправки сообщений локального пользователя и передачи электронной почты между серверами. Работает на порту TCP25.

POP3 (Post Office Protocol v.3 – Протокол офисной почты версии 3) – используется для получения почты конечными пользователями. Использует порт TCP110

IRC (Internet Relayed Chat – Разговор через интернет в реальном времени) – чат-протокол, один из самых распространенных сервисов в интернет. Использует порты TCP194, TCP6660-6667.

3. Немного подробнее хотелось бы поговорить о нескольких сервисах интерактивного доступа на базе TCP, и о структуре запуска интернет-сервисов в Linux. На сегодняшний день широко распространены 3 сервиса интерактивного доступа к командной строке: RSH (Remote SHell - удаленная оболочка), Telnet и SSH (Secure SHell – защищенная оболочка). SSH является развитием RSH. Протоколы очень похожи, за исключением того, что в SSH принято шифрование как пароля и имени пользователя, так и передаваемых данных, что позволило ему практически повсеместно вытеснить RSH. Сервисы RSH и SSH – это Unix-ориентированные протоколы, которые сравнительно редко встречаются в других системах. В отличие от них сервис Telnet применяется на многих платформах, в том числе и на Windows.

В большинстве случаев TCP-службы в Linux работают как демоны и сами осуществляют передачу данных по сети. Однако в некоторых случаях используется демон интернет xinetd, который обеспечивает работу по сети, а программы, функционирущие с ним обмениваются данными по каналам, образованным дескрипторами файлов 0 (стандартный ввод), 1 (стандартный вывод) и 2 (стандартный поток

ошибок). Обычно, сервисы, работающие с xinetd имеют в своем имени приставку in. Так, например, файл службы Telnet называется in.telnetd.

Практическая часть: настройка сервисов `xinetd+in.telnet`, `sshd`.

Сетевое администрирование Linux. Протокол TCP. Занятие второе

FTP. vsftpd. HTTP. Apache.

Ход занятия

1. Протокол FTP (File Transfer Protocol) – один из старейших в интернет. Это протокол седьмого уровня (уровня приложений) модели OSI , основанный на надежной передаче данных (протоколе TCP , порты 21, 20).

Работа с ftp-ресурсами сходна с работой в локальной файловой системе. Существует множество специализированных клиентов для работы с протоколом FTP . Однако практически все современные браузеры способны обрабатывать информацию с FTP-ресурсов . Их мы и будем использовать в практической части сегодняшнего занятия.

FTP-серверов значительно меньше чем клиентов, но тоже достаточно много, чтобы можно было в них запутаться. Мы с Вами будем настраивать FTP-сервер vsftpd (Very Secure FTP Daemon – очень надежный FTP-демон), входящий в поставку ASPLinux.

Перед рассмотрением его настройки, я хочу немного рассказать о работе протокола FTP . Этот протокол изначально является протоколом с аутентификацией по имени пользователя. Однако для протокола FTP существует и так называемый анонимный режим. В случае его использования (браузеры используют этот режим по умолчанию) вместо имени вводится `anonymous` , а вместо пароля – `адрес электронной почты` (браузер использует или настоящий из адресной книги, или случайно подобранный).

Другой особенностью протокола является использование разных режимов – пассивного и активного. В пассивном режиме клиент открывает соединения для передачи данных и управляет потоком, а сервер пассивно отдает данные. Этот режим хорошо работает из внутренних сетей с фэйковой адресацией. В активном режиме поток для передачи данных открывает и контролирует сервер. Однако, если

клиент находится за файрволом с преобразованием адресов (маскарадингом), то он не сможет работать с сервером в этом режиме.

Последней особенностью, обсуждаемой нами, будет передача данных в бинарном или текстовом (ascii) виде. Некоторые бинарные данные в первых версиях серверов/клиентов не могли быть переданы в прямом виде по сети в связи с особенностями реализации FTP . Для этой цели были придуманы несколько режимов – бинарный, когда передающиеся данные никак не преобразовываются для передачи по сети, и ascii, когда данные с помощью определенного кодирования преобразуются в последовательность символов ascii (с кодами от 0 до 127). Формат ascii увеличивал размер файла, а соответственно и время загрузки из сети, но позволял преодолеть возникающие проблемы. Современные FTP-серверы/клиенты самостоятельно способны выбирать нужный режим передачи, поэтому пользователь в большинстве случаев не задумывается об этом.

Vsftpd представляет из себя программу-демон. В ASPLinux для него предварительно создан скрипт запуска, помещенный в /etc/rc.d/init.d с именем vsftpd . Сам исполняемый файл адресуется как /usr/sbin/vsftpd .

За настройку сервера отвечают несколько файлов:

/etc/vsftpd.busy_banner – файл, в котором записывается сообщение, выдаваемое клиенту, когда сервер не может ответить на запрос.

/etc/vsftpd.ftpusers – содержит список пользователей, которым запрещен доступ по ftp . Как правило этот файл содержит список критически важных для системы пользователей, таких как `root` , `bin` , `lpd` и другие.

/etc/vsftpd.user_list – список пользователей, значение которого меняется в зависимости от того, как установлен параметр `userlist_deny` в основном файле конфигурации. Если Этот параметр установлен в `NO` , то на FTP смогут попасть только пользователи, указанные в этом файле, если в `YES` , то все, кроме этих пользователей. Нужно иметь в виду, что если пользователь запрещен в файле /etc/vsftpd.ftpusers , то он не сможет иметь доступ, даже будучи разрешенным в этом файле.

/etc/vsftpd/vsftpd.conf -основной файл конфигурации сервера. Рассмотрим

его поподробнее:

`anonymous_enable=YES` – параметр-переключатель, разрешающий или запрещающий доступ к анонимному FTP-каталогу (/var/ftp/pub)

`local_enable=YES` – если этот параметр стоит в `NO` , то пользователи той же системы, где запущен сервер, не могут получить к нему доступ.

`write_enable=YES` – Если этот параметр установлен в `NO` , то ни один из пользователей не имеет права закачать файлы на FTP-ресурс .

`local_umask=022` – Определяет права доступа к файлам для всех пользователей, кроме `anonymous` .

`anon_upload_enable=YES` – Если параметр установлен в `YES` , то анонимные пользователи могут изменять существующие в каталоге /var/ftp/pub файлы.

`anon_mkdir_write_enable=YES` – Позволяет анонимным пользователям закачивать файлы на FTP-ресурс и создавать каталоги.

`dirmessage_enable=YES` – Если параметр включен, то FTP-сервер будет выдавать сообщение из файла при переходе в каталог. Это сообщение может содержаться в файле с именем .message внутри каталога, однако имя файла может быть дополнительно указано с помощью опции `message_file` .

`xferlog_enable=YES` – разрешает серверу записывать журнал с именами переданных файлов. По умолчанию, журнал пишется в /var/log/vsftpd.log , однако имя файла может быть дополнительно указано опцией `xferlog_file` . Формат файла может быть указан стандарной (`YES`) или нестандартной опцией `xferlog_std_format` .

`connect_from_port_20=YES` – позволяет использовать для передачи данных 20-й порт.

`chown_uploads=YES` – изменяет владельца файла, закачанного анонимным пользователем на того, который указан в параметре

`chown_username .`

`idle_session_timeout` – указывает время в секундах, через которое будет разорвано соединение с клиентом, не выполняющим никаких действий.

`data_connection_timeout` – указывает время в секундах, через которое будет разорвано соединение в случае перерыва в передаче данных.

Дополнительную информацию по конфигурации этого сервера можно получить в руководстве man (страницы vsftpd, vsftpd.conf).

2. Протокол HTTP – на сегодняшний день самый распространенный протокол интернет, базирующийся на TCP (порт 80). Множество смежных стандартов, связанных с публикацией, обработкой и отображением HTML-страниц , настолько усложнили вэб-серверы, что их конфигурации подчас является непростым делом для начинающего администратора. Мы с вами попробуем настроить сервер на базовую работу и посмотреть основные параметры его конфигурации.

В качестве пособия мы будем использовать один из самых мощнейших и самый распространенный в интернет сервер Apache в версии 2.0.40. Конфигурирование Apache 2.x сильно отличается от конфигурирования его предшественника – линейки 1.3

Основным файлом конфигурации является httpd.conf , который в RedHat-совместимых системах (в том числе в ASPLinux) расположен в /etc/httpd/conf .

`ServerTokens OS` – этот параметр позволяет скрыть информацию об используемых модулях.

`ServerRoot "/etc/httpd"` – указывает на каталог, где расположены конфигурационные файлы, файлы журналов и другая необходимая информация для работы сервера.

`Timeout 300` – указывает время в секундах, через которое будет разорвано соединение с клиентом, не проявляющим активности.

`KeepAlive Off` – разрешает или запрещает поддерживать соединение с клиентом пакетами типа PING-PONG .

`Listen 80` – указывает порт (или ip-адрес и порт в формате `192.168.2.2:80`) на котором будет функционировать вэб-сервер.

`LoadModule` – позволяет загрузить и использовать модуль расширения для поддержки дополнительных возможностей у сервера.

`User apache, Group apache` – параметры указывают пользователя и группу, от имени которого должен будет работать сервер.

`ServerAdmin root@localhost` – указывает адрес электронной почты администратора сервера, который будет указан на страницах с сообщениями об ошибках.

`ServerName hosters.volnet.ru` – указывает доменное имя, с которым работает сервер. Если для компьютера доменное имя не определено, в эту позицию вписывается IP-адрес .

`DocumentRoot "/var/www/html"` – указывает каталог, в котором располагаются документы HTML.

Раздел Directory описывает каталог, в котором расположены документы.

`<Directory "/var/www/html">` -начало описания

`Options` – опции каталога с документами, указываемые через пробел. В качестве опций могут использоваться : `Includes` – включаемые файлы, `FollowSymLinks` – переход по символьным ссылкам, `SymLinksifOwnerMatch` – переход по символьным ссылкам, если владелец совпадает , `ExecCGI` – выполнять файлы CGI и передавать вывод файлов клиенту и другие.

`AllowOverride` - позволяет изменять параметры вложенных директорий с помощью файлов .htaccess полностью (`All`), не изменять (`None`) или отдельные параметры.

`Order` – определяет допустимые значения при конфигурации безопасности. Может включать параметры `Deny, Allow` или то и

другое через запятую.

`Deny/Allow from all/192.168.3.11` – определяет доступ к каталогу.

`</Directory>` - закрывает описание каталога.

Сетевое администрирование Linux. ICMP

Протокол ICMP. Типы пакетов. Утилиты ping, traceroute, tcptraceroute. Утилиты управления сетью. Nmap. NetCat. Netstat.

Ход занятия

1. Протокол ICMP (Internet Control Message Protocol , протокол контроля сообщений в интернет) – это Internet-протокол третьего (сетевого) уровня модели OSI , создающий пакеты с сообщениями об ошибках и другой информацией об обработке IP-пакетов .

В протоколе ICMP описаны несколько типов пакетов, таких как:

- эхо-запрос – пакет, отправляемый для проверки связи с удаленным узлом
- эхо-ответ – пакет, получаемый источником в ответ на пакет типа эхо-запрос
- назначение недоступно – получатель такого пакета информируется о недоступности сети/узла/порта назначения.
- Редирект – пакет информирует узел об изменении маршрута до назначения. В современных ОС данный тип пакета игнорируется.
- Время вышло – узел-источник получает этот тип пакета, если IP-пакет был уничтожен в связи с обнулением поля TimeToLive.

Остальные типы (а также подтипы) пакетов ICMP можно посмотреть в заголовочном файле /usr/include/linux/icmp.h.

2. Утилита ping служит для проверки факта наличия связи с удаленным узлом, а также надежности и скорости связи. Иногда утилита используется для раскрытия ip-адресов хоста по его имени.

Синтаксис ее таков:

```
ping [параметры] имя_или_ip-адрес_хоста
```

С помощью параметра `-c` вы можете указать количество запросов, которые выполнит ping . Если параметр `-c` опущен, то ping будет подавать запросы до тех пор, пока не получит сигнал или пользователь

не нажмет <Ctrl-c> .

```
[gserg@WebMedia linux]$ ping -c3 www.ya.ru
PING ya.ru (213.180.204.8) 56(84) bytes of data.
64 bytes from ya.ru (213.180.204.8): icmp_seq=0 ttl=49 time=26.7 ms
64 bytes from ya.ru (213.180.204.8): icmp_seq=1 ttl=49 time=25.6 ms
64 bytes from ya.ru (213.180.204.8): icmp_seq=2 ttl=49 time=24.7 ms

--- ya.ru ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2019ms
rtt min/avg/max/mdev = 24.741/25.718/26.724/0.809 ms, pipe 2
[gserg@WebMedia linux]$
```

С помощью параметра `-f` можно проверить быстродействие сети (работает только из под root). При использовании этого параметра ping посылает около 1000 пакетов в секунду.

```
[gserg@WebMedia gserg]$ ping -f 192.168.2.254
PING 192.168.2.254 (192.168.2.254) 56(84) bytes of data.
ping: cannot flood; minimal interval, allowed for user, is 200ms
[gserg@WebMedia gserg]$ su -c "ping -f 192.168.2.254"
Password:
PING 192.168.2.254 (192.168.2.254) 56(84) bytes of data.

--- 192.168.2.254 ping statistics ---
238 packets transmitted, 238 received, 0% packet loss, time 4379ms
rtt min/avg/max/mdev = 0.147/0.177/0.478/0.031 ms, pipe 2, ipg/ewma 18.478/
[gserg@WebMedia gserg]$
```

Параметром `-s` может быть указан размер пакета в байтах (не менее `64` и не более `65000`), в ином случае размер пакета составит `64` байта.

```
[gserg@WebMedia linux]$ ping -s 65000 192.168.2.254
PING 192.168.2.254 (192.168.2.254) 65000(65028) bytes of data.
65008 bytes from 192.168.2.254: icmp_seq=0 ttl=64 time=12.0 ms
65008 bytes from 192.168.2.254: icmp_seq=1 ttl=64 time=12.0 ms
65008 bytes from 192.168.2.254: icmp_seq=2 ttl=64 time=11.9 ms
65008 bytes from 192.168.2.254: icmp_seq=3 ttl=64 time=11.9 ms
```

```
--- 192.168.2.254 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3026ms
rtt min/avg/max/mdev = 11.988/12.029/12.083/0.086 ms, pipe 2
[gserg@WebMedia linux]$
```

Дополнительную информацию можно получить из справочного руководства man .

Команда traceroute поможет Вам проверить путь до удаленного хоста, выдавая все промежуточные маршрутизаторы. С помощью этой команды становится просто диагностировать проблему в большой сети, быстро выявляя сбойный участок. Синтаксис команды такой же, как и у ping , за исключением параметров командной строки.

```
[gserg@WebMedia linux]$ traceroute www.ya.ru
traceroute to ya.ru (213.180.204.8), 30 hops max, 38 byte packets
 1  ns.edu.vologda.ru (192.168.2.2)  0.281 ms  0.192 ms  0.162 ms
 2  80.92.3.3 (80.92.3.3)  1.169 ms  1.257 ms  1.103 ms
 3  sv-vol00ra-s2-0.severttk.ru (80.92.3.1)  3.140 ms  3.250 ms  3.111 ms
 4  sv-yar00rb-s1-0-0.severttk.ru (80.92.0.69)  34.511 ms sv-yar00rb-s2-5-0.
severttk.ru (80.92.0.97)  25.161 ms sv-yar00rb-s1-0-0.severttk.ru (80.92.0.69)
 5  MSK41-F000.145.gw.transtelecom.net (217.150.38.142)  28.066 ms  21.43
 6  MSK41.TRANSTELECOM.NET (193.232.244.211)  25.680 ms  18.921 n
     MPLS Label=350 CoS=6 TTL=255 S=1
 7  ix2-m9.yandex.net (193.232.244.93)  23.085 ms  21.548 ms  35.076 ms
 8  c3-vlan3.yandex.net (213.180.192.171)  25.177 ms  23.104 ms  22.918 ms
 9  ya.ru (213.180.204.8)  33.519 ms  23.189 ms  32.496 ms
```

Команда tcptraceroute практически аналогична предыдущей, однако в качестве базового протокола она использует протокол tcp . Это позволяет проверять доступ к службам tcp и путь до них.

```
gserg@ADM:~$ tcptraceroute smtp.mail.ru 25
Selected device eth0, address 10.52.2.1, port 45217 for outgoing packets
Tracing the path to smtp.mail.ru (194.67.23.111) on TCP port 25 (smtp), 30 hop
 1  10.52.0.5  0.383 ms  0.231 ms  0.219 ms
 2  b229.vologda.ru (193.19.67.229)  1248.475 ms  345.180 ms  106.562 ms
 3  spb-vgld.ptn.ru (212.48.195.9)  78.138 ms  65.802 ms  157.785 ms
 4  spb-bbn1-ge-2-1-0-88.rt-comm.ru (213.59.5.141)  120.521 ms  114.442 m
```

```
 5  msk-bgw1-ge1-0-0-0.rt-comm.ru (217.106.0.14)  76.536 ms  177.440 ms
 6  msk-bgw1-ge1-0-0-0.rt-comm.ru (217.106.0.14)  143.422 ms  90.071 ms
 7  195.161.165.246  172.163 ms  133.969 ms  110.668 ms
 8  cat03.Moscow.gldn.net (195.239.10.189)  79.129 ms  77.774 ms  93.654 n
 9  cat12.Moscow.gldn.net (194.186.159.238)  76.082 ms  162.883 ms  77.04
10  mailru-KK12-2-gw.Moscow.gldn.net (195.239.8.90)  105.042 ms  75.285
11  * * *
12  smtp.mail.ru (194.67.23.111) [open]  78.527 ms  81.314 ms  80.269 ms
```

В большинстве случаев, параметры командной строки не приходится использовать, однако их все же следует изучить с помощью справочного руководства man .

3. Утилиты управления сетью существуют для диагностистики как сетевых проблем, так и диагностики проблем каждого конкретного хоста.

Наиболее популярной утилитой, с помощь которой диагностируются проблемы хостов является nmap , который входит в поставку практически всех современных дистрибутивов операционной системы Linux.

Nmap обладает множеством способностей, но нас в первую очередь интересует его возможности определения активных сервисов удаленного хоста, а также операционной системы.

Полноценно использовать nmap можно только при наличии прав суперпользователя. Вот те параметры nmap , которые нас будут интересовать:

- `-sS` – скрытое сканирование TCP-портов
- `-sT` – открытое сканирование TCP-портов
- `-sU` – открытое сканирование UDP-портов
- `-sP` – ping-сканирование (поиск активных хостов в сети)
- `-sF, -sN, -sX` – разные скрытые типы сканирования TCP-портов.
- `-O` – определение типа операционной системы (не работает с `-sP`)
- `-p` – указание диапазона портов для сканирования (`12345` ,

12345-23333)

- -v – подробный вывод.

```
[root@WebMedia linux]# nmap -v -sS -O 192.168.2.84

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2005-04-23 17:24 MSI
Host 192.168.2.84 appears to be up ... good.
Initiating SYN Stealth Scan against 192.168.2.84 at 17:24
Adding open port 135/tcp
Adding open port 139/tcp
Adding open port 445/tcp
The SYN Stealth Scan took 0 seconds to scan 1657 ports.
For OSScan assuming that port 135 is open and port 1 is closed and neither are
Interesting ports on 192.168.2.84:
(The 1654 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
Device type: general purpose
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows 2000 Profes
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=9641 (Worthy challenge)
IPID Sequence Generation: Incremental

Nmap run completed -- 1 IP address (1 host up) scanned in 2.391 seconds
```

Команда NetCat (nc) – это еще более широко используемая команда. Ее предназначение, в первую очередь, это проверка работоспособности сети на конкретном узле и выявление ошибок при передаче. Команда nc может открыть порт и выводить информацию из него на экран (или в файл, другое приложение по каналу). Эту возможность часто используют для проверки работы TCP . Для открытия порта служит ключ -l , а порт можно указать параметром -p:

```
1-я консоль
[root@WebMedia linux]# nc -l -p 25 127.0.0.1
test
```

```
test double
 punt!

2-я консоль
[gserg@WebMedia gserg]$ telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
test
test double
Connection closed by foreign host.
[gserg@WebMedia gserg]$
```

Однако и с помощью nc можно просканировать открытые порты. Правда, nc не обладает таким широким набором возможностей как nmap , и делает сканирование намного медленнее. Для этого используется параметр `-z` . Параметр `-v` позволяет нам получить дополнительную информацию:

```
[root@WebMedia linux]# nc -z -v 127.0.0.1 600-700
localhost.localdomain [127.0.0.1] 631 (ipp) open
```

Команда netstat позволяет просмотреть открытые соединения на текущем компьютере. Введенная без параметров, она покажет все соединения, включая открытые сокеты unix. Команда принимает параметры, наиболее часто используемыми являются:

- `-n` – показывает числовые значения номеров портов вместо имени из /etc/services и числовые ip-адреса вместо доменных имен;
- `-p` – показывает pid процесса, использующего соединение;
- `-t` – показывать только TCP-сокеты
- `-u` – показывать только UDP-сокеты
- `-i` – показать список интерфейсов и статистику трафика на них
- `-l` – только слушающие порты

Таким образом, просмотр всех программ, слушающих TCP-порты :

```
root@ADM:/var/mail# netstat  -tpl
```

```
Активные соединения с интернетом (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State       PID/Program
tcp        0      0 localhost:2208          *:*               LISTEN      4785/hpiod
tcp        0      0 *:print01               *:*               LISTEN      4864/inetd
tcp        0      0 *:51762                 *:*               LISTEN      26729/sim
tcp        0      0 *:telnet                *:*               LISTEN      4864/inetd
tcp        0      0 localhost:ipp           *:*               LISTEN      4761/cupsd
tcp        0      0 localhost:2207          *:*               LISTEN      4788/python
tcp6       0      0 *:5800                  *:*               LISTEN      9066/kded [kde
tcp6       0      0 *:5900                  *:*               LISTEN      9066/kded [kde
tcp6       0      0 *:65005                 *:*               LISTEN      9519/notes2w
tcp6       0      0 *:57177                 *:*               LISTEN      9519/notes2w
```

Просмотр всех интерфейсов:

```
root@ADM:/var/mail# netstat  -i
```

Таблица интерфейсов ядра

```
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-D
eth0  1500 0    238445      0      0      0   200095      0      0      0 BMRU
lo    16436 0       282      0      0      0      282      0      0      0 LRU
```

Сетевое администрирование Linux. Iptables

Таблицы. Цепочки. Прохождение трафика. Механизм определения состояний. Базовый синтаксис и команды.

Ход занятия

1. Iptables — это программный интерфейс к файрволу ядра Netfilter . Iptables представляет собой утилиту командной строки, а Netfilter загружается в ядро в качестве модулей (основным из которых является ip_tables).

Ядро обрабатывает трафик в определенном порядке. Модули для обработки трафика называются таблицами, в которых существуют по умолчанию несколько цепочек для обработки пакетов. Вот список существующих таблиц и их назначение:

Таблица 20.1.

Название таблицы	Цепочки по умолчанию	Назначение
RAW	PREROUTING, OUTPUT	Выбор пакетов, не обрабатываемых системами контроля соединений и nat. Возможные действия: NOTRACK
MANGLE	PREROUTING, INPUT, OUTPUT, FORWARD, POSTROUTING	Внесение изменений в заголовки пакетов. Основные действия: TOS, TTL, MARK
NAT	PREROUTING, OUTPUT, POSTROUTING	Преобразование сетевых адресов. Основные действия: DNAT, SNAT, MASQUERADE,
FILTER	INPUT, OUTPUT, FORWARD	Фильтрация пакетов. Основные действия: ACCEPT, DROP

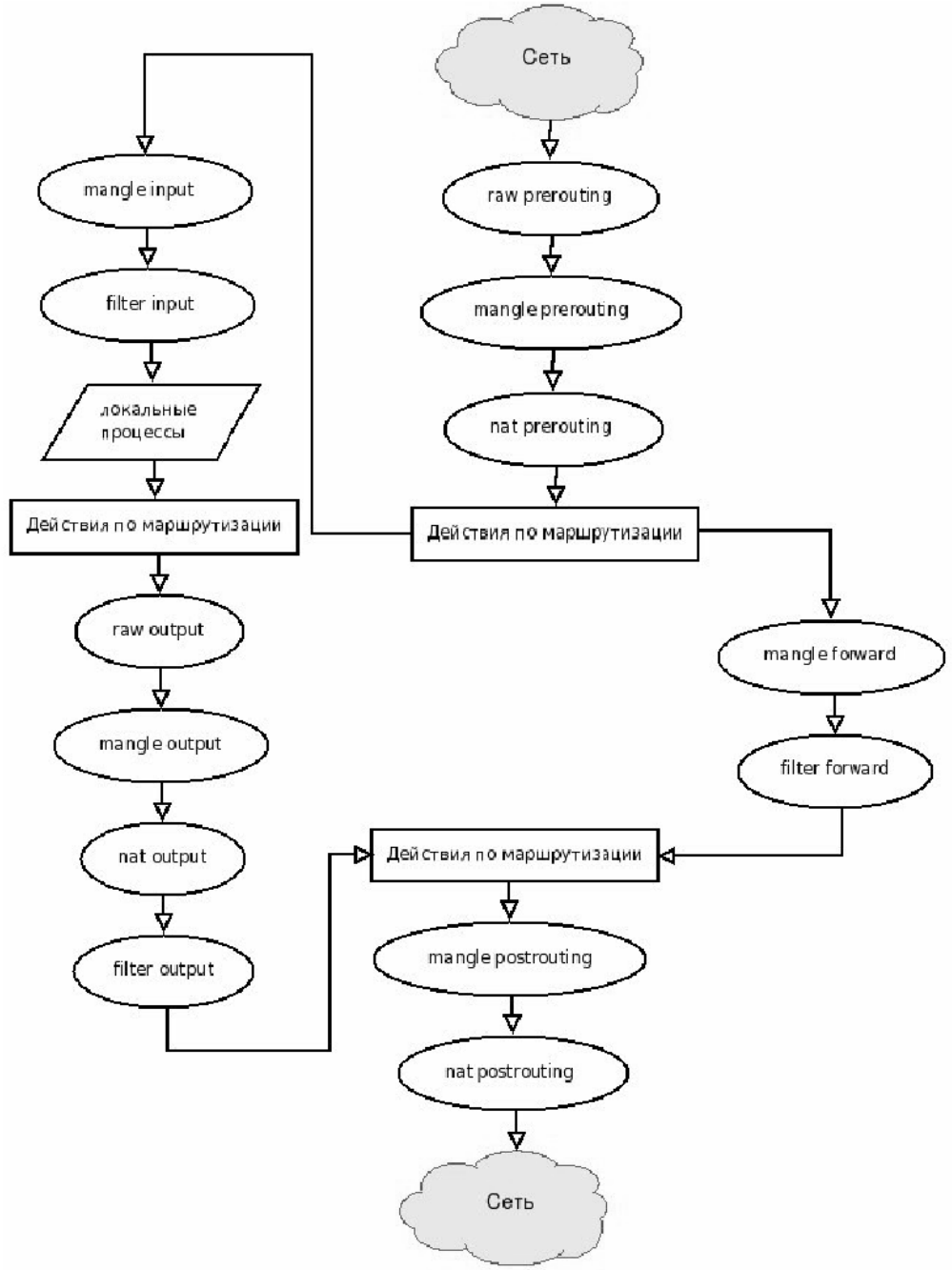
Синтаксис Iptables позволяет создавать свои цепочки, куда перенаправлять трафик по определенным критериям. Общая схема прохождения трафика такова, что в одной таблице пакету может быть назначено только одно действие, однако пакет может быть обработан в

других таблицах. Наиболее используемые действия:

Таблица 20.2.

Действие	Значение
`Переход`	При соответствии пакета указанным критериям, он передается на обработку в свою созданную цепочку, после прохождения которой, пакет будет возвращен в цепочку, откуда вызван переход, если в своей цепочке он не был обработан
`ACCEPT`	Пакет принят в этой таблице. Передается на обработку в следующую.
`DROP`	Пакет отброшен. Движение пакета прекращается.
`DNAT`	Destination NAT. Имеет дополнительный параметр `--to-destination IP`
`SNAT`	Source NAT. Имеет дополнительный параметр `--to-source IP`
`REJECT`	Пакет не принят. Хост-источник получит пакет Destination unreachable
`LOG, ULOG`	Позволяют журналировать информацию о пакете. Имеют много дополнительных параметров
`MASQUERADE`	По сути это `SNAT` , но динамически определяющий IP на исходящем интерфейсе, что позволяет его использовать, например, с DHCP . Имеет параметр `--to-ports` .
`REDIRECT`	Позволяет перенаправлять пакеты с одного порта на другой. Имеет параметр `--to-ports`

Вот схема прохождения трафика по цепочкам:



Кроме основного функционала, Iptables способен подгружать дополнительные модули, для дополнительной обработки. Одним из таких модулей является модуль обработки состояний пакета state . Он

позволяет указывать состояние пакета:

Таблица 20.3.

Состояние	Описание
NEW	Пакет открывает новое соединение (TCP) или принадлежит однонаправленному потоку
RELATED	Показывает, что пакет принадлежит уже имеющемуся соединению, но открывает новое. Например, открывается сессия передачи данных в FTP
ESTABLISHED	Соединение установлено, пакеты идут в обоих направлениях
INVALID	Пакет связан с неизвестным потоком или соединением или имеет ошибки в заголовке

2. Консольная команда iptables принимает в качестве параметров описание одного правила. В базовом варианте для загрузки правил подразумевается создание скрипта вида:

```
#!/bin/bash
 #rc.firewall — load iptables rules
 PROG='/sbin/iptables'
 #clear all chains tables
 $PROG -F

 #filter table
 $PROG -A INPUT -t filter -p TCP -s 80.32.5.7/32 -d 0.0.0.0 —dport 80 -j A
 …
```

Команда принимает следующие основные параметры для работы с цепочками:

Таблица 20.4.

параметр	назначение
-A CHAIN	добавить правило в конец цепочки CHAIN
-F CHAIN	обнулить цепочку CHAIN
-I CHAIN N	вставить правило с номером N в цепочку CHAIN
-D CHAIN	удалить правило из цепочки CHAIN

`-R CHAIN N`	заменить правило N в цепочке CHAIN
`-L CHAIN`	показать список правил в цепочке CHAIN
`-N CHAIN`	создать цепочку CHAIN
`-X CHAIN`	удалить цепочку CHAIN
`-E CHAIN1 CHAIN2`	переименовать цепочку CHAIN1 в CHAIN2
`-P CHAIN policy`	задать политику по умолчанию (ACCEPT или DROP)
`-Z CHAIN`	обнулить все счетчики внутри цепочки

Для параметров используются критерии:

Таблица 20.5.

критерии	назначение
`-t TABLE`	указывает таблицу для цепочки
`-p`	протокол (IP, TCP, UDP, ALL и т.д.)
`-s`	адрес источника с маской
`-d`	адрес приемника с маской
`-i`	входящий интерфейс
`-o`	исходящий интерфейс
`--sport`	порт на источнике (только для TCP и UDP)
`--dport`	порт на приемнике (только для TCP и UDP)
`--tcp-flags`	флаги (только для TCP). Принимает значения `SYN, ACK, RST, FIN, URG, PSH`
`--syn`	соответствует пакетам с установленным флагом SYN и сброшенными флагами ACK и FIN (только для TCP)
`--icmp-type`	указывает тип ICMP-пакета (только для ICMP)
`-m модуль`	загружает дополнительный модуль
`-m mac --mac-source`	позволяет указать MAC-адрес источника
`-m state --state`	позволяет определить состояние пакета

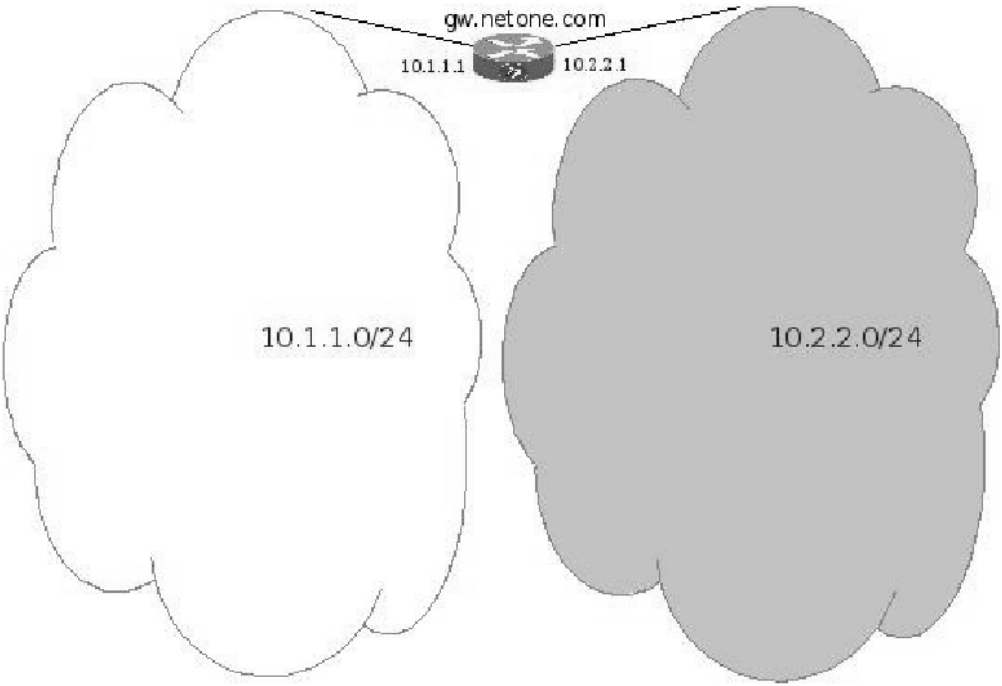
`-j`	указывает действие

Примеры:

```
iptables -P INPUT ACCEPT -t filter
iptables -A INPUT -t filter -p TCP -s 192.168.1.1/32 -d 192.168.2.0/24 --dpc
iptables -A POSTROUTING -t nat -p ALL -s 192.168.1.0/24 -j SNAT --to-sc
iptables -N mychain -t filter
iptables -A OUTPUT -t filter -d 193.19.64.11 --dport 21 -j mychain
iptables -A mychain -t filter -s 192.168.1.0/24 -j ACCEPT
iptables -A mychain -t filter -s 192.168.2.0/24 -j REJECT
iptables -A FORWARD -t filter -m mac --mac-source 11:22:33:44:55:66 -j AC
iptables -A INPUT -t filter -p TCP -d 192.168.1.0/32 --dport 21 -m state --sta
```

Карты практических занятий

Роль: администратор сети, администратор шлюза



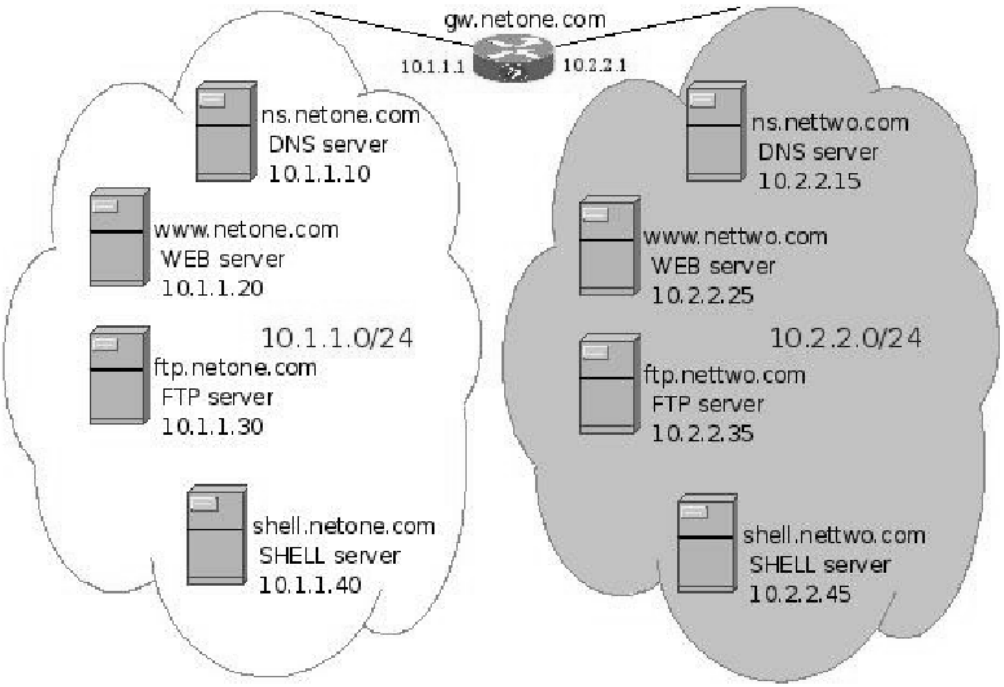
Задачи:

1. Настроить сетевые интерфейсы eth0 и eth0:0 с адресами, указанными на схеме сети.
2. Разрешить маршрутизацию (записать 1 в файл /proc/sys/net/ipv4/ip_forward) и сделать настройку таким образом, чтобы при загрузке происходила такая запись автоматически
3. Настроить маршрутизацию с помощью route
4. С помощью средств управления сетью создать карты обеих сетей, в которые включить ip компьютера, запущенные службы, предполагаемая роль.
5. Создать с помощью xinetd собственную echo-службу, позволяющую проверять работу сети.

Написать скрипт, проверяющих доступность всех хостов в сети.

Внимание! Все настройки необходимо выполнить таким образом, чтобы после перезагрузки все необходимые сервисы поднимались автоматически.

Роль: администратор DNS-сервера сети 1

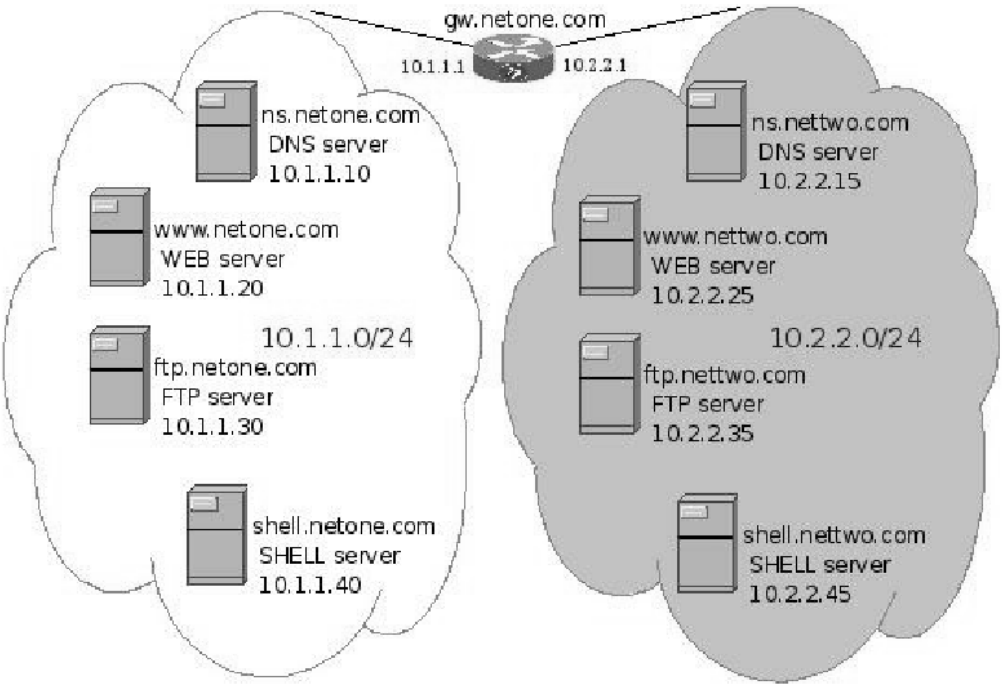


Задачи:

1. Настроить сетевой интерфейс eth0 с адресом, указанным на схеме сети. Также настроить имя хоста.
2. Создать master-зону netone.com и прописать в нее все хосты вашей сети. Указать в параметре allow-transfer ip-адрес сервера ns.nettwo.com
3. Создать файл зон для зоны netone.com
4. Создать slave-зону, в разделе masters которой указать ip-адрес сервера ns.nettwo.com. Проверить, скопировался ли файл slave-зоны с ns.nettwo.com.
5. Проверить правильность преобразования имен.

Внимание! Все настройки необходимо выполнить таким образом, чтобы после перезагрузки все необходимые сервисы поднимались автоматически.

Роль: администратор DNS-сервера сети 2

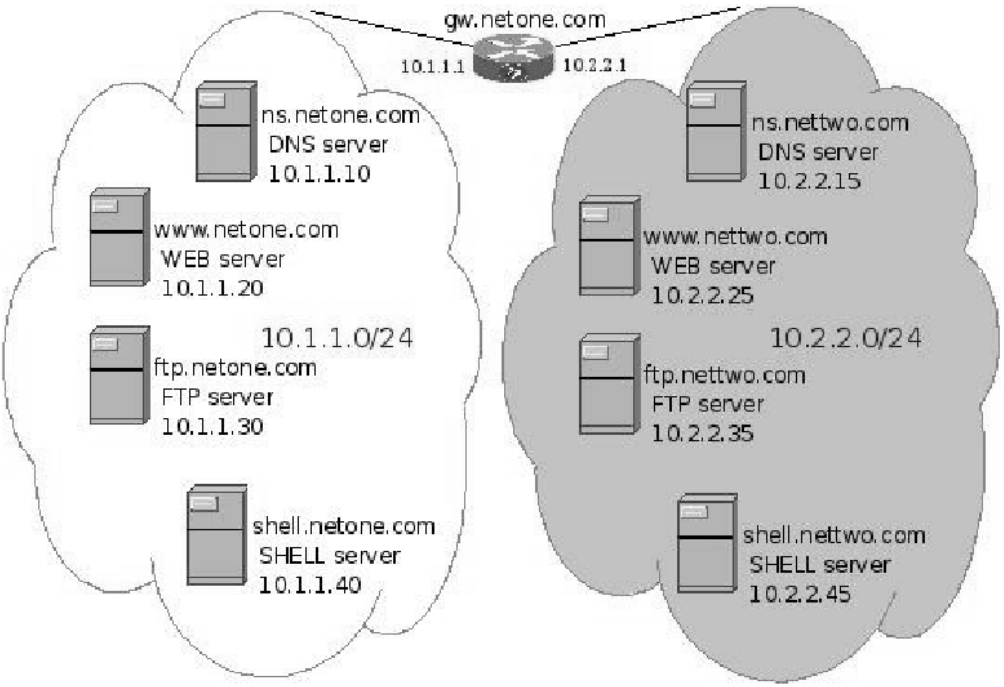


Задачи:

1. Настроить сетевой интерфейс eth0 с адресом, указанным на схеме сети. Также настроить имя хоста.
2. Создать master-зону nettwo.com и прописать в нее все хосты вашей сети. Указать в параметре allow-transfer ip-адрес сервера ns.netone.com
3. Создать файл зон для зоны netone.com
4. Создать slave-зону, в разделе masters которой указать ip-адрес сервера ns.netone.com. Проверить, скопировался ли файл slave-зоны с ns.netone.com.
5. Проверить правильность преобразования имен.

Внимание! Все настройки необходимо выполнить таким образом, чтобы после перезагрузки все необходимые сервисы поднимались автоматически.

Роль: администратор FTP-сервера сети 1



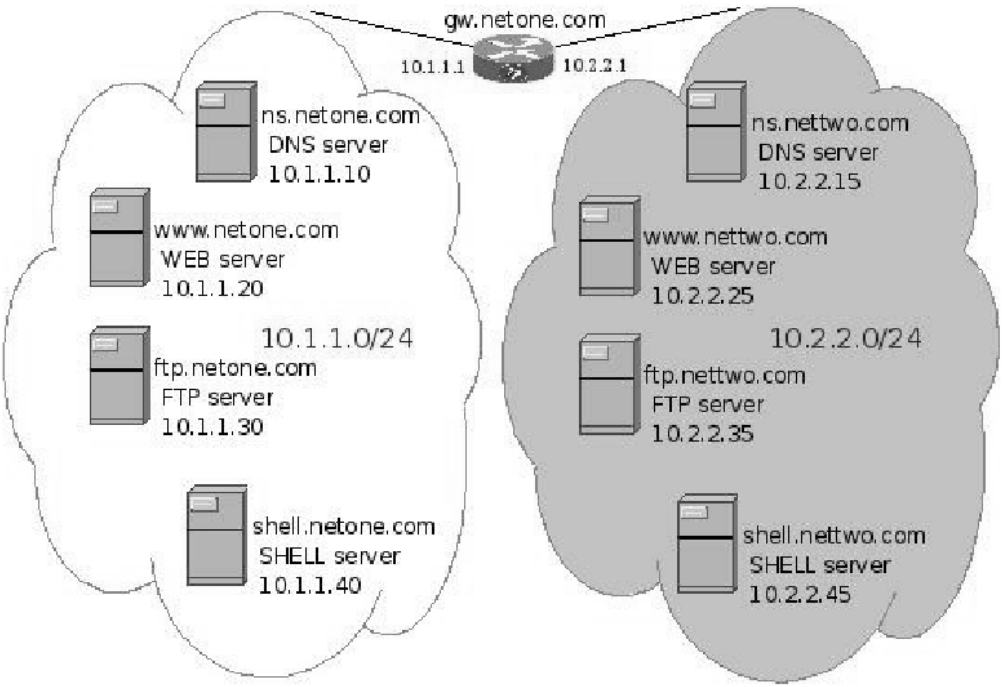
Задачи:

1. Настроить сетевой интерфейс eth0 с адресом, указанным на схеме сети. Также настроить имя хоста.
2. Настроить vsftpd, чтобы при обращении по имени из браузера на адрес ftp://ftp.netone.com вы попадали в /var/ftp/. Анонимный пользователь не должен иметь прав записи в каталог!!!
3. Необходимо создать файл, выводящий приветствие при посещении ftp-ресурса.
4. Создать пользователя, который может писать в каталог /var/ftp/pub, подключившись по ftp.

Внимание! Все настройки необходимо выполнить таким образом, чтобы

после перезагрузки все необходимые сервисы поднимались автоматически.

Роль: администратор FTP-сервера сети 2



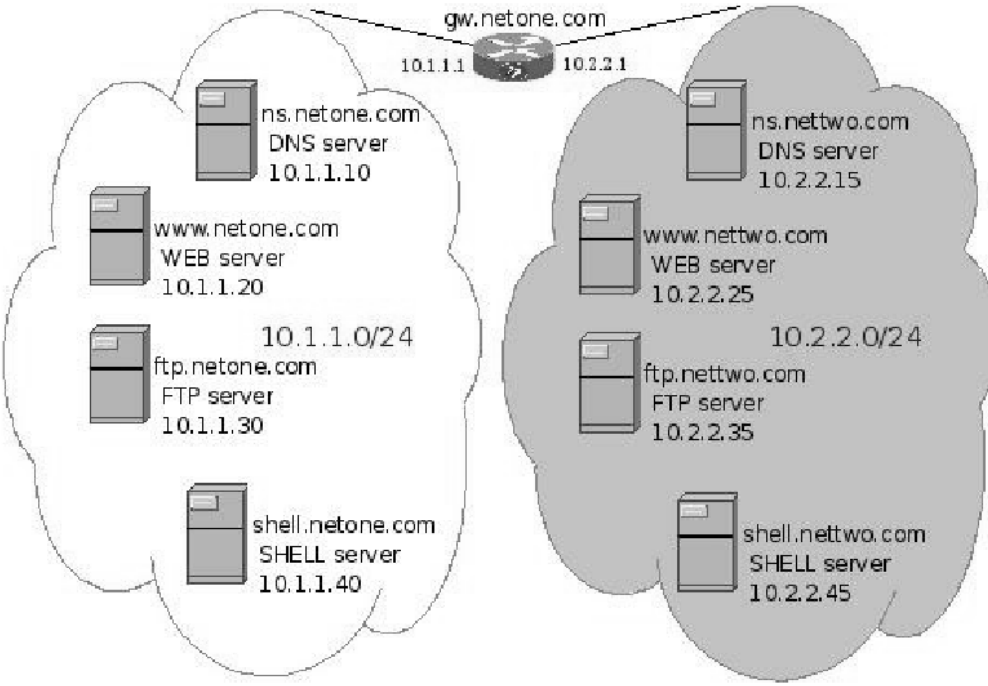
Задачи:

1. Настроить сетевой интерфейс eth0 с адресом, указанным на схеме сети. Также настроить имя хоста.
2. Настроить vsftpd, чтобы при обращении по имени из браузера на адрес ftp://ftp.nettwo.com вы попадали в /var/ftp/. Анонимный пользователь не должен иметь прав записи в каталог!!!
3. Необходимо создать файл, выводящий приветствие при посещении ftp-ресурса.
4. Создать пользователя, который может писать в каталог /var/ftp/pub, подключившись по ftp.

Внимание! Все настройки необходимо выполнить таким образом, чтобы после перезагрузки все необходимые сервисы поднимались

автоматически.

Роль: администратор SHELL-сервера сети 1

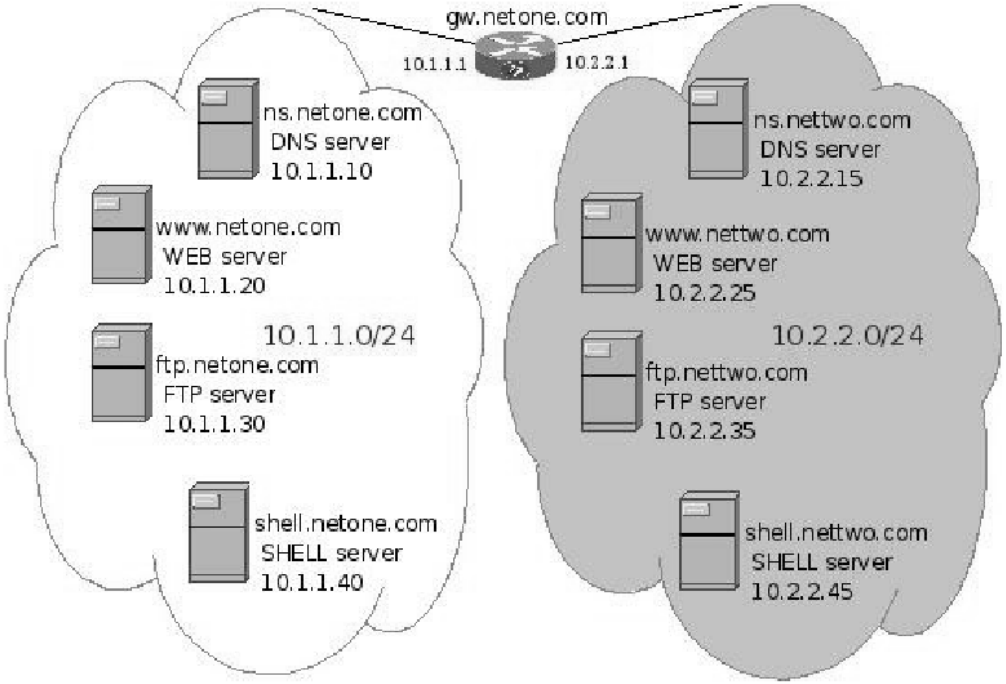


Задачи:

1. Настроить сетевой интерфейс eth0 с адресом, указанным на схеме сети. Также настроить имя хоста.
2. Настроить доступ к хосту по протоколам telnet и ssh.
3. Создать файл, выводящий приветствие при входе пользователя, сообщающий имя компьютера, правила работы
4. Создать пользователя с пустым паролем и разрешить его вход по протоколу ssh.

Внимание! Все настройки необходимо выполнить таким образом, чтобы после перезагрузки все необходимые сервисы поднимались автоматически.

Роль: администратор SHELL-сервера сети 2

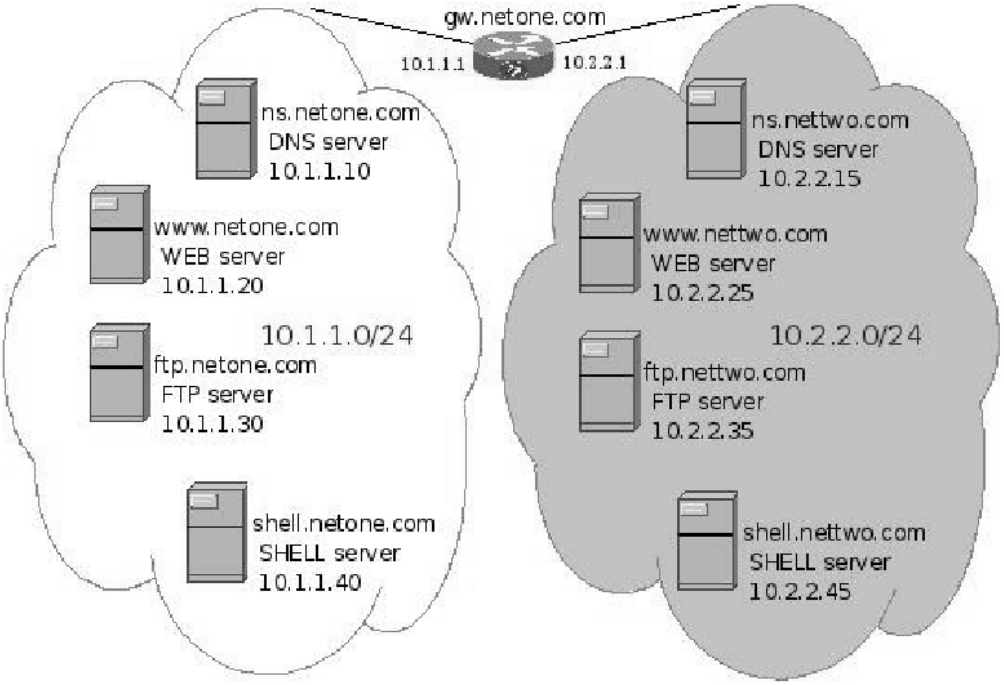


Задачи:

1. Настроить сетевой интерфейс eth0 с адресом, указанным на схеме сети. Также настроить имя хоста.
2. Настроить доступ к хосту по протоколам telnet и ssh.
3. Создать файл, выводящий приветствие при входе пользователя, сообщающий имя компьютера, правила работы
4. Создать пользователя с пустым паролем и разрешить его вход по протоколу ssh.

Внимание! Все настройки необходимо выполнить таким образом, чтобы после перезагрузки все необходимые сервисы поднимались автоматически.

Роль: администратор WEB-сервера сети 1

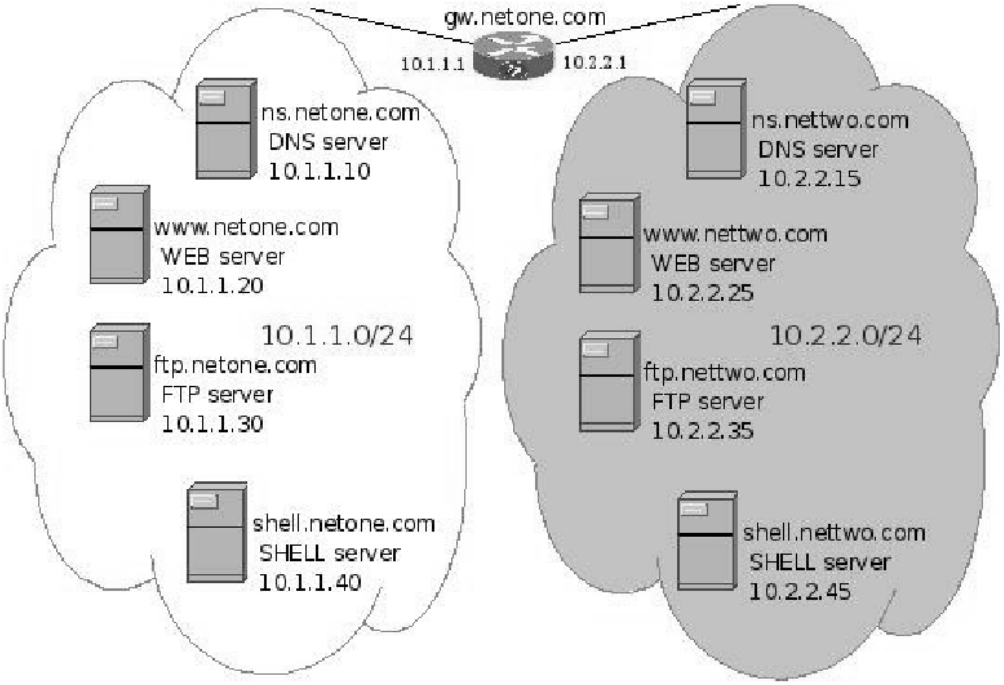


Задачи:

1. Настроить сетевой интерфейс eth0 с адресом, указанным на схеме сети. Также настроить имя хоста.
2. Настроить Apache web-сервер, чтобы при обращении по имени из браузера ссылка: http://www.netone.com он выводил содержимое файла /var/www/html/index.html.
3. Написать файл /var/www/html/index.html, содержащий информацию о вашей компании.
4. Написать скрипт, выполняющий роль CGI и выводящий на экран текущее время и список всех символических ссылок в каталоге /etc

Внимание! Все настройки необходимо выполнить таким образом, чтобы после перезагрузки все необходимые сервисы поднимались автоматически.

Роль: администратор WEB-сервера сети 2



Задачи:

1. Настроить сетевой интерфейс eth0 с адресом, указанным на схеме сети. Также настроить имя хоста.
2. Настроить Apache web-сервер, чтобы при обращении по имени из браузера ссылка: http://www.nettwo.com он выводил содержимое файла /var/www/html/index.html.
3. Написать файл /var/www/html/index.html, содержащий информацию о вашей компании.
4. Написать скрипт, выполняющий роль CGI и выводящий на экран текущее время и список всех символических ссылок в каталоге /etc

Внимание! Все настройки необходимо выполнить таким образом, чтобы после перезагрузки все необходимые сервисы поднимались автоматически.

Список литературы

1. Робачевский А.М., "Операционная система Unix®", СПб. БВХ – Санкт-Петербург, 1999

2. Армстронг (мл.) Джеймс, "Секреты Unix®" : 2-е изд, М.: Издательский дом "Вильямс", 2000

3. Паркер Тим, "Linux 5.2. Энциклопедия пользователя», К.: Издательство "ДиаСофт", 1999

4. Oscar Anderson, Iptables Tutorial, URL: http://www.opennet.ru/docs/RUS/iptables/

5. Таблицы Iptables, URL: http://www.protocols.ru/modules.php?name=News&file=article&sid=100 , В переводе Андрея Киселева

6. Шевель А, "Linux. Обработка текстов. Специальный справочник", Спб.: Питер, 2001

7. Системная справочная служба Linux Man

8. Д. Тейнсли, “Linux и Unix: программирование в shell Руководство разработчика” , К.: Издательская группа BHV, 2001

9. Справочная система Midnight commander

10. Команда vimtutor ru

11. Графическая оболочка X-Windows System, URL: http://www.unix.org.ua/solaris/xwindows.htm

12. The xfree86, URL: http://www.xfree.org

13. X.Org Foundation, URL: http://www.x.org

14. Cisco systems и др. - Руководство по технологиям объедененных сетей, 3-е издание, URL: http://lunix.net.ru , М. : Издательский дом “Вильямс”, 2002

15. Кирх. О, Доусон Т, Linux для профессионалов. Руководство администратора сети, 2-е издание, СПб.: Питер, 2001

Содержание