А.В. Аникин, И.Г. Жукова
Д.В. Литовкин, И.С. Гурьянов

Лабораторный практикум по дисциплине «Администрирование операционных систем»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.В. Аникин, И.Г. Жукова
Д.В. Литовкин, И.С. Гурьянов

Лабораторный практикум по дисциплине «Администрирование операционных систем»

Учебное пособие

Волгоград
2015

УДК 004.451 (075)

Рецензенты:
зав. кафедрой «Фундаментальная информатика и оптимальное управление» Волгоградского государственного университета,
д-р. физ.-мат. наук, профессор *А. А. Воронин*;

профессор кафедры «Теория и методика обучения математике и информатике» Волгоградского государственного социально-педагогического университета,
д-р пед. наук, канд. физ.-мат. наук, профессор *Т. М. Петрова*

Печатается по решению редакционно-издательского совета
Волгоградского государственного технического университета

Аникин, А. В.
Лабораторный практикум по дисциплине «Администрирование операционных систем»: учеб. пособие / А. В. Аникин, И. Г. Жукова, Д. В. Литовкин, И. С. Гурьянов; ВолгГТУ. – Волгоград, 2015. – 128 с.

ISBN 978-5-9948-1959-3

В пособии изложены цель, содержание и порядок выполнения лабораторных работ по курсу «Администрирование операционных систем».

Предназначено для студентов всех форм обучения по направлению 09.03.04 «Программная инженерия».

Ил.9. Табл. 40. Библиогр.: 6 назв.

ISBN 978-5-9948-1959-3

Оглавление

Введение

Практикум разработан для студентов, обучающихся в бакалавриате по направлению «Программная инженерия». Он ориентирован на получение знаний и навыков по работе, настройке, администрированию операционных систем семейства UNIX/Linux.

В рамках лабораторных работ рассматриваются вопросы работы с командной оболочкой bash, работы с файловой системой, администрирование пользователей, управление процессами и задачами, написание собственных сценариев командной оболочки Bash, сетевое администрирование.

Задания ориентированы на использование дистрибутивов Debian, Ubuntu.

Тема 1. Командная оболочка bash

Цель: изучить интерфейс командной строки ОС Linux, приобрести основные навыки по работе с терминалом командной строки оболочки bash.

Задачи:

1. Изучить оболочки командной строки ОС Linux.
2. Изучить основы работы с командной строкой.
3. Понять структуру командной строки.
4. Изучить файловое и переменное окружение ОС Linux.
5. Изучить возможности терминала.
6. Изучить основные типовые команды командной строки.

Командная оболочка — это программа, взаимодействующая с пользователем с помощью текстового интерфейса. Он называется интерфейсом командной строки (CLI). Оболочка позволяет пользователю запускать программы и выполнять команды операционной системы. Оболочка интерпретирует введенные пользователем команды, преобразуя их в инструкции операционной системы.

Существует несколько разновидностей командной оболочки в ОС Linux:

1. Bourne shell (sh) — оригинальная командная оболочка, является самой ранней оболочкой UNIX. sh является стандартной и доступна почти в любом дистрибутиве *nix. Существует много командных оболочек, основанных (идейно или напрямую) на Bourne shell;

- ksh (KornShell) — клон шелла Борна, разработанный Дэвидом Корном из AT&T Labs. Синтаксис совместим, функциональность интерактивности увеличена.
- pdksh (public domain ksh) — открытая реализация ksh.
- bash (bourne again shell) (эмуляция совместимости POSIX) расширенная свободная оболочка ash, сходная с pdksh, стандартная

оболочка в Linux. В дальнейшем мы будем использовать только эту оболочку.

2. C shell — (несовместима с POSIX shell) оболочка, с синтаксисом на основе Си.

- csh (C-Shell) — оболочка из состава дистрибутива BSD, имеет Си-образный синтаксис и не является POSIX-совместимой.
- tcsh (csh) — реализация csh с интерактивными возможностями, не уступающими bash. Удобна для интерактивной работы, совместима с csh.

3. ash (Almquist shell) — одна из самых маленьких оболочек, доступных для UNIX. ash используется при загрузке Linux в однопользовательском режиме, в защищённом режиме или при загрузке дискетных версий Linux.

Оболочки обладают различной функциональностью и даже различными командами. При написании сценария оболочки (скрипта) необходимо проверять, в какой оболочке он запускается. Команда */etc/shells* позволяет посмотреть список доступных оболочек.

Встроенные системные команды Linux-систем состоят из процедур оболочки и исполняемых файлов.

- Встроенные команды в разных оболочках могут выполняться по-разному;
- встроенные команды выполняются быстрее, чем системные;
- для многих встроенных команд есть системные аналоги.

Места расположения системных команд pwd и /bin/pwd:

- /bin;
- /sbin;
- /usr/bin;
- /usr/sbin;
- /usr/local/bin;

- /usr/local/sbin.

Запустить терминал можно сочетание клавиш Ctrl+Alt+T. Закрыть терминал можно введя команду exit или сочетанием клавиш Ctrl+D. Переключиться в виртуальный терминал можно нажав комбинацию клавиш Ctrl+Alt+F1, выйти из виртуального терминала: Ctrl+Alt+F7.

Другие клавиатурные сочетания bash:

- <Ctrl> + <B> курсор влево;
- <Ctrl> + <F> курсор вправо;
- <Alt> + <B> курсор на слово влево;
- <Alt> + <F> курсор на слово вправо;
- <Ctrl> + <A> курсор в начало строки;
- <Ctrl> + <E> курсор в конец строки;
- <Ctrl> + <H> удаление символа перед курсором;
- <Ctrl> + <D> удаление символа в позиции курсора;
- <Alt> + <D> удаление слова;
- <Ctrl> + <L> очистка экрана;
- <Alt> + <T> перемена мест аргументов;
- <Alt> + <U> перевод слова в верхний регистр;
- <Alt> + <L> перевод слова в нижний регистр;
- <Ctrl> + <C> остановка выполнения команды;
- <Ctrl> + <Z> приостановка выполнения задания (bg, kill);
- <Ctrl> + <R> поиск команды в истории.

Выполнение множества команд доступны только root пользователям. Root это специальный аккаунт в UNIX-подобных системах с UID (User IDentifier) 0, владелец, которого имеет право на выполнение всех без исключения операций. Для того, что бы войти под root сначала его необходимо активировать, а затем в командной строке ввести команду su, после прохождения авторизации ваша работа не будет ограничивается в

правах доступа.

В основном, команды, запускаемые из командной строки, имеют следующий формат: command -options <filename> .

- options, и <filename> являются необязательными параметрами: существуют команды, не требующие ввода ни одного из них, и команды, требующие ввода нескольких опций и имен файлов. Если используются несколько опций одновременно, их можно сгруппировать. Например, для просмотра подробного списка (-l) всех файлов текущего каталога, включая скрытые файлы (-a), воспользуйтесь командой : *ls –al*.

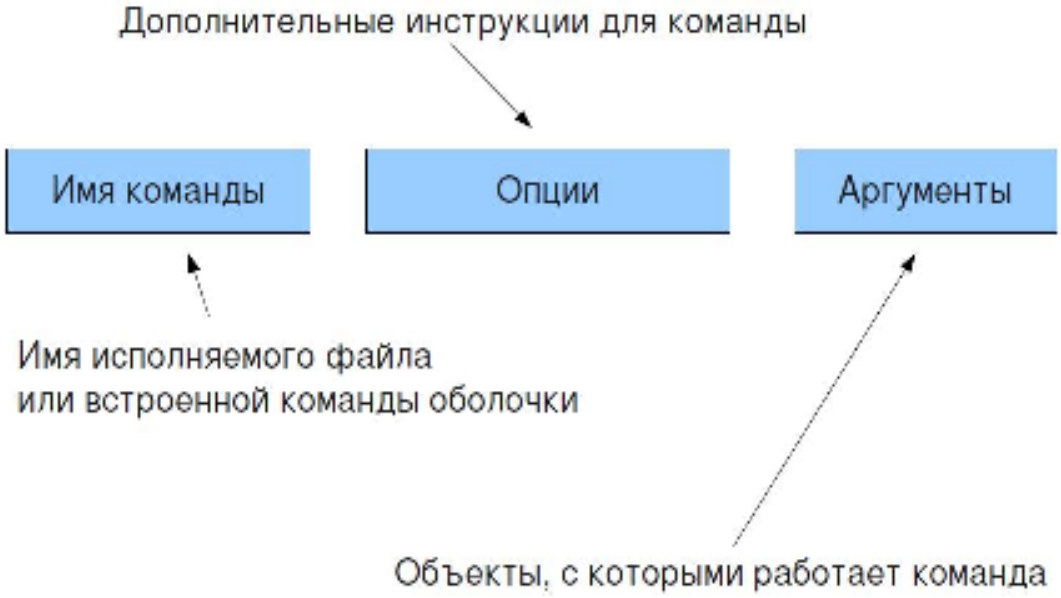

Рисунок 1- Структура командной строки

Для ввода длинной команды используется \ (перевод строки). Для ввода нескольких команд в одной строки используется «;». Команда1 && Команда2 — команда 2 выполняется только в случае удачного выполнения команды1. Команда1 || Команда2 — команда 2 выполняется только в случае неудачного выполнения команды 1.

Получить помощь по командам в Unix-системе можно несколькими способами.

1. help – встроенная помощь оболочки. Большинство команд Linux могут быть запущены с параметром *--help* . Например, эта команда даст

Вам краткую помощь по команде cp (копирование): *cp --help | less*.

2. man — система помощи в любой Unix системе. Более расширенная информация доступна из командной строки с использованием так называемых страниц руководства (manual pages). Например: *man cp* выведет на экран страницу руководства для команды *cp* (копирование). Страницы руководства являются стандартными системами помощи в Linux, и содержат много детальной технической информации. Система man не занимается отображением страниц, а только находит их и форматирует, передавая программе просмотра (по умолчанию less): PgUp,PgDn – перемещение по тексту, <пробел> следующая страница, </> строка – поиск подстроки вниз, <?>строка – поиск подстроки вверх, <n> следующее вхождение искомой строки, <q> выход. Команда *manpath* позволяет узнать путь поиска страниц man. Страницы man состоят из стандартных разделов:

- NAME – информация, которая будет использована при поиске по ключевому слову;
- SYNOPSIS – формат вызова, опции и аргументы;
- DESCRIPTION – описание объекта (программы, файла, библиотеки);
- OPTIONS – подробное описание опций;
- FILES – файлы, связанные с командой;
- AUTHOR – имя автора с указанием электронной почты;
- SEEALSO– указатели на другие страницы man;
- COPYRIGHT – права собственности, политика распространения.

3. info— гипертекстовая иерархическая система Gnu TexInfo. Например: *info cp* выведет Вам информацию о команде "cp" (копирование). Часто info содержит информацию схожую с man, но более свежую. Команды: <n> следующий узел, <p> предыдущий узел, <u> родительский узел, <l> предыдущая страница, <s> поиск строки на странице, <q> выход.

4. /usr/share/doc — документация программ.

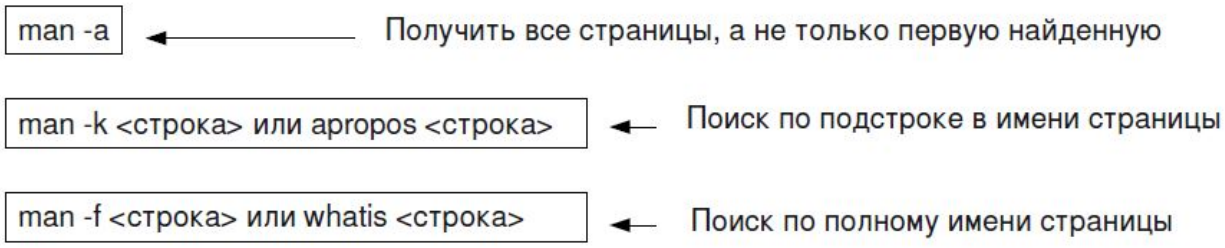

Рисунок 2 - Работа с системой man

Страницы руководства man в Linux делятся на следующие секции:

1. Команды пользовательского уровня и приложения;
2. Системные вызовы и коды ошибок ядра;
3. Библиотечные функции;
4. Информация о файлах устройств и других специальных файлах;
5. Форматы конфигурационных файлов;
6. Помощь по играм;
7. Макросы, кодировки, интерфейсы;
8. Команды системного администрирования;
9. Внутренние интерфейсы и спецификации ядра.

Порядок перечисления секций в этой таблице не случаен. Дело в том, что файлы с информацией расположены в подкаталогах каталога /usr/man и команда man ищет нужную информацию, просматривая эти подкаталоги именно в том порядке, который приведен. Если вы, например, дадите команду «[user]$ man swapon» то получите справку о команде swapon из секции 8. Поэтому если вы хотите получить справку по системному вызову swapon, надо дать команду «[user]$ man 2 swapon» указывая номер секции, в которой надо искать информацию.

В Linux формирование файловой системы и каталогов осуществляется по таким правилам:

- логически файловая структура организована в виде иерархии;
- каждый каталог может иметь множество подкаталогов, но у каждого;

- подкаталога имеется только один родительский каталог;
- имя корневого каталога /(он сам для себя является родительским);
- прописные и строчные буквы в именах файлов различаются:TheFile и thefile — разные файлы;
- абсолютные имена файлов показывают путь к файлу от корневого каталога (имена начинаются с /)/home/user1/homework/lab1.html;
- относительные имена показывают путь к файлу от текущего каталога;
- имена файлов могут содержать точки: archive.tar.gz;
- имя файла не может содержать символов / и \0 (null).

Типы файлов в Linux:

- --Обычные файлы;
- d – каталоги;
- l – символические ссылки (указатель на другой файл);
- b – блочные устройства (специальные файлы для обращения к стройствам, например, жесткому диску);
- c – символьные устройства (специальные файлы, предназначенные для ввода/вывода с таких устройств, как терминал или мышь);
- p – именованный канал (один из вариантов организации взаимодействия между процессами);
- s – сокеты (предназначены для организации сетевого межпроцессного взаимодействия).

Понятие параметра в оболочке bash подобно понятию переменной в обычных языках программирования. Именем (или идентификатором) параметра может быть слово, состоящее из алфавитных символов, цифр и знаков подчеркивания (только первый символ этого слова не может быть

цифрой), а также число или один из следующих специальных символов: *, @, #, ?, - (дефис), $, !, 0, _ (подчеркивание). Чтобы отличать команды от переменных, переменные лучше обозначать большими буквами (пример-HOSTNAME).

Таблица 1. Переменные оболочки и окружения

Экранирование строки	VAR1='Bolshoy Privet!'
	VAR1='Vam vsem '$VAR1
Изменение значения	VAR1=${VAR1}ZZ
Список всех переменных оболочки	set
Уничтожить переменную	unset имя
Список переменных окружения	env

Говорят, что параметр задан или установлен, если ему присвоено значение. Значением может быть и пустая строка. Чтобы вывести значение параметра, используют символ $ перед его именем. Так, команда «[user]$ echo name» выдаст на экран слово name, а команда «[user]$ echo $name» выдаст значение переменной name (если таковое, конечно, задано). Переменные оболочки доступны только в той оболочке, в которой были описаны! Перевод переменной оболочки в переменную окружения (доступна для дочерних процессов): *export VAR1*.

В ОС Linux важнейшими переменными окружения являются:

- HOME — путь к домашнему каталогу;
- LOGNAME и USER — имя пользователя;
- MAIL — путь к почтовому ящику;
- PATH — путь поиска исполняемых файлов;
- PS1 — вид приглашения оболочки;
- PWD — имя текущего каталога;
- OLDPWD — имя предыдущего каталога;
- SHELL — имя исполняемого файла оболочки;
- TERM — тип терминала;
- HOSTNAME — имя хоста;
- SHLVL — номер загруженной оболочки.

Таблица 2. Файлы настроек хранящие переменные окружения

Переменные, общие для всех пользователей	/etc/profile
Настройки пользователя (профиль)	~/.bash_profile
	~/.bash_login
	~/.profile
Выполняется при ручном запуске оболочки	~/.bashrc

В командных оболочках, используемых в Linux, есть масса способов экономии усилий (нажатий на клавиши) при выполнении наиболее распространённых действий: автоматическое дополнение длинных названий команд или имён файлов, поиск и повторное выполнение команды, уже когда-то исполнявшейся раньше, подстановка списков имён файлов по некоторому шаблону и многое другое.

Хорошую возможность не тратить время на набор одних и тех же команд подарили нам разработчики оболочки bash. Они предоставили возможность использовать историю команд, чтобы сократить время набора команд и сделать работу в командной строке более эффективной.

По умолчанию, история команд включена, и все команды, которые вы выполняете в командной строке, могут быть использованы вами повторно без особых затрат времени на их набор.

Историю команд в Linux можно отключить, выполнив в командной строке команду: $ *set +o history*

Если, после отключения истории команд, вы вновь захотите использовать ее в своей работе, просто выполните команду: $ *set -o history*

Одна из переменных окружения, имеющая название HISTSIZE, хранит в себе количество выполненных команд. Посмотреть значение переменной HISTSIZE можно выполнив в командной строке следующую команду*: $ echo $HISTSIZE*.

Таблица 3. Команды для работы с историей

Описание	**Команда**
посмотреть содержимое истории	$ history
12 выполненных команд	$ history 12
удалить 257 строку в истории	$ history -d257
повторить последнюю выполненную	$!!

команду	
повторить предпоследнюю выполненную команду	$!-2
выполненную ранее и начинающуюся с букв sud	$!sud
поиск недавно введенной команды, начинающейся на <префикс>	!<префикс>
поиск и редактирование недавней команды	fc <префикс>
поиск недавней команды по подстроке	<Ctrl>+<R>
последние команды	<вверх>, <вниз>

Автоматически попытаться дополнить командную строку именами файлов или команд клавишей <Tab>, если вариантов несколько то повторным нажатием <Tab> можно вывести список. Если строка начинается с $ - дополняется имя переменной оболочки, ~ дополняется имя пользователя, @ дополняется имя хоста.

Псевдонимы команд служат для ускорения набора длинных, часто используемых команд.

Таблица 4. Псевдонимы команд

Описание	**Команда**
вывести список псевдонимов	alias
создать новый псевдоним	alias lls='ls ld'
удалить псевдоним	unalias lls
удалить все псевдонимы	unalias a

Командная подстановка (command substituion) — результат выполнения одной команды автоматически передается в качестве аргументов другой команде. Синтаксис: внешняя_команда `внутренняя_команда` и внешняя_команда $(внутренняя_команда). Пример: ls l `which rpm` ls l $(cat /etc/shells).Можно присвоить результат выполнения переменной оболочки: ID=`id';echo $ID.

В командной строке можно вычислять выражения, заключенные в квадратные скобки или в двойные круглые скобки. Перед скобками должен стоять символ $, а результаты выражений можно передавать как аргумент какой-либо команде или назначать переменной. Арифметика выполняющаяся в командной строке является целочисленной. Пример: echo $((1+2)),echo $((7%3)),echo $((5*6)),echo $((7/3)) .

Шаблоны подстановки и перечисление. Символ «звездочка» * является шаблоном для любого количества любых символов в именах файлов, и даже для их отсутсвия. echo * (выведет все имена файлов в данной директории);

- Единственный символ, который не удовлетворяет этому шаблону - лидирующая точка в именах скрытых файлов echo .* (выведет все имена файлов в данной директории, включая те, которые начинаются с точки);
- Символ ? заменяет один символ в имени файла, который должен находится в той позиции, где находится знак вопроса;
- Шаблон диапозона - квадратные скобки [0-9] для любых цифр [a-zA-Z] для букв английского [!abc] множество любых символов кроме a, b, c;
- Механизм перечисления - фигурные скобки {rc,_profile} echo .bash{rc,_profile} обращение к двум файлам, .bashrc и .bash_profile. Имена этих файлов имеют общую подстроку .bash, которая вынесена за фигурные скобки. В фигурных скобках через запятую перечислены варианты продолжения: rc и _profile/.

Таблица 5. Базовые команды

Описание	**Команда**	**Пример**
очистить экран	clear	
вывести имя текущего каталога	pwd	
вывести содержимое текущего каталога	ls	
вывести содержимое произвольного каталога	ls <каталог>	
вывести подробную информацию	ls –l	
переместиться в домашний каталог	cd	
переместиться в заданный каталог	cd <каталог>	
cd переместиться в предыдущий каталог	cd-	
простейший способ создать	<filename>	

файл		
создание файла или изменение даты модификации файла	touch <filename(s)>	date > f1; cat f1; ls –l f*; sleep 60; touch f1 f2; date; cat f1; ls –l f*
удаление файла(ов) и каталогов. Опции: f– не спрашивать подтверждения; i– спрашивать подтверждения; r– рекурсивно удалить каталог и его содержимое	rm <filename(s)>	
Проверка имен файлов перед удалением по шаблону	touch f{1,2,3} ls f*[1,2] rm f*[1,2]	
создание каталога	mkdir <dir>	
создание дерева	mkdir –p dir1/dir2/dir3	
удаление пустого каталога	rmdir <dir>	
удаление дерева пустых каталогов	rmdir –p	
Копирование	cp <source> <destination>	
Рекурсивное копирование	cp R <dir1> <dir2>	
Перемещение (переименование)	mv <source> <destination>	
Поиск файлов. Критерии: name (по имени); iname (по имени игнорируя регистр); type (по типу); size (по размеру); empty (пустые); mtime (по дате модификации); perm (по правам доступа); user (по принадлежности); group (по принадлежности).	Find <места_поиска> <критерии>	Если используются шаблоны подстановки, то необходимы кавычки: find ~ -name "dom*". Два критерия, объединенных условием ИЛИ: find ~ -name "dom*" -o –empty. Подставляются имена файлов: find ~ -name "*core*" –exec rm –f {} \;
Поиск файлов в базе данных командой locate	locate <подстрока имени файла>	
Поиск файла в каталогах, входящих в переменную PATH (строку поиска)	which <что-ищем>	
Поиск файла в системных каталогах (не смотря на строку поиска)	whereis <что-ищем>	
Определение типа файлов	file /etc/passwd file /bin/ls	
Свободное место	df –h	
Сколько занимает папка	du –sh <имя-папки>	

Практическое задание по теме

Выполнить установку Ubuntu Linux LTS.

1. Перейти в корневую директорию (папку). Проверьте, в какой директории находитесь.

2. Вывести пронумерованный список директорий далее работать с директорией # (ваш –номер по списку);

3. Вывести содержимое директории:

- в формате по умолчанию;
- в обратном порядке;
- содержимое поддиректорий;
- вывести все файлы включая скрытые;
- вывести файлы с указанием их размера в КБ/МБ/ГБ;
- вывести файлы отсортированные по размеру, с указанием размера в КБ/МБ/ГБ;
- только имена вложенных директорий, расположенных в текущей директории;
- отсортированное по дате создания файла;
- отсортированное по дате обращения к файлу;
- только файлы, вторая буква имени которых – гласная англ. Алфавита
- записать список файлов и папок в текущей директории (с полной информацией о них) в файл dirlist.txt , который лежит в домашней директории.

4. перейти домашнюю папку с помощью короткой команды;

5. вернуться в предыдущую директорию;

6. вернуться обратно в домашнюю;

7. перейти на уровень выше;

8. Перейдите в каталог /tmp. С помощью одной команды перейдите в подкаталог local/bin каталога /usr.

9. вывести содержимое файла dirlist.txt:

- просто;
- в обратном порядке;
- с нумерацией непустых строк строк;
- с нумерацией всех строк;
- схлопывая подряд идущие пустые строки в одну;

14. Создать в домашней директории папку linux_lab1

15. Войти в директорию linux_lab1

16. Скопировать в нее файл dirlist.txt из домашней директории

17. Удалить файл dirlist.txt из домашней директории

18. Создать директорию manyfiles

19. Создать в ней 100 файлов с именами a1, a2, a3, …. a100.

20. Удалить только файлы с четными номерами.

21. Вывести строки файла dirlist.txt, содержащие файлы с определенным месяцем (в зависимости от номер варианта 1-январь,..12 – декабрь, 13 – опять январь) и записать их в файл grep_month_name.txt

22. Записать строки, не содержащие этот месяц, в файл grep_other_monthes.txt.

23. Создать папку grep, переместить в нее файлы созданные в пунктах 31 и 32.

24. Находясь в папке linux_lab1 найти все файлы в этой директории и ее поддиректориях в которых встречается подстрока root, вывести строки с указанием их номеров.

25. Найти все файлы в системе, содержащие в имени bash;

26. Найти файлы, измененные за последний час.

27. Найти символические ссылки в каталоге / (но не глубже), вывести, на что они указывают.

28. Посмотрите, какие переменные окружения заданы в вашей системе;

29. Поменять приглашение командной строки, добавить текущее время.

30. Удалить весь каталог manyfiles со всеми файлами;

31. Создайте текстовый файл следующего содержания:

- 1+2
- 6*4
- 97%12
- 43215/43*100

Посчитайте все примеры из файла с помощью одной команды. (Вариантов команды существует несколько, засчитывается каждый из них).

Тема 2. Работа с файлами и каталогами. Управление пользователями

Цель: Познакомиться с принципами аутентификации, форматами файлов для хранения учетных записей и изучить команды для управления учетными записями.

Задачи:

1. Понять где хранится информация о пользователе, в каком формате;
2. Научиться регистрировать, удалять, блокировать, учетные записи. Управлять паролями учётных записей;
3. Научиться управлять группами пользователей;
4. Понять, что такое профили пользователя, для чего они нужны, как ими управлять;
5. Научиться получать отчёты об активности пользователей.

Linux — многопользовательская система, учётная запись — это информация о пользователе а системная политика представляет собой правила работы в системе.

При создании нового пользователя следует соблюдать правила при выборе имени пользователю. Имя может содержать символы латинского алфавита, цифры, _, $,[a-z_][a-z0-9_]*[$].Нельзя использовать пробел, : @,#, т.д. и специальные символы (табуляция, конец строки).

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

Аутентификация – системная процедура, позволяющая операционной системе определить, какой именно пользователь осуществляет вход. Вся информация о пользователе обычно хранится в файлах /etc/passwd и /etc/group. В файле /etc/passwd содержится

информация о пользователях. Запись для каждого пользователя занимает одну строку.

Рисунок 3 - Запись информации о пользователе в файл /etc/passwd

На рисунке 3 структура записи о пользователя, в которую входит следующая информация:

- 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;

При добавлении нового пользователя происходит новая запись в /etc/passwd и /etc/shadow, создаётся домашний каталог. Каталог /etc/skel содержит файлы, которые копируются в создаваемый домашний каталог пользователя.

Добавляя нового пользователя чаще всего используются опции команды useradd такие как:

- s — файл оболочки по умолчанию;
- d — путь к домашнему каталогу;
- m — необходимо создавать домашний каталог;
- M — не создавать домашний каталог;
- k — путь к альтернативному каталогу скелета;
- u — назначить UID;
- g — назначить GID (первичную группу);
- G — список групп пользователя;
- e — дата блокировки учетной записи;
- f — срок после устаревания пароля до блокировки учетной записи.

Если пользователь не имеет право входить в сеанс, то ему в качестве оболочки нужно указать:

- /bin/false — команда, всегда возвращающая код ошибки;
- /dev/null — специальный файл-поглотитель;
- /sbin/nologin — возвращает код ошибки и сообщение о невозможности входа в сеанс.

Управление паролями в Linux часть системной политики. Управляя паролями требуется определить категорий пользователей, которые могут сами выбирать пароль, правила выбора паролей и требования к их уровню сложности, сроки устаревания паролей, длительность периода запрета на изменение пароля. Это является типичными правилами при управлении паролями.

Опции при управлении паролями:

- -l — блокировка;
- -u — разблокирование;
- -S — текущее состояние пароля;
- -d — удаление пароля;

- -n — период запрета смены пароля;
- -x — максимальный срок использования пароля;
- -w — период предупреждений;
- -i — период после устаревания пароля до блокировки.

Группы пользователей очень удобный механизм администрирования пользователей в ОС Linux. Информация о группах пользователей хранится в файле /etc/group. Запись для каждой группы пользователей так же занимает одну строку как и в /etc/passwd. Строка в /etc/group имеет следующую структуру: <имя группы>:<пароль группы>:<GID группы>:<список пользователей>.

В файлах профиля пользователя устанавливаются переменные окружения. PATH- путь поиска исполняемых файлов, USER- имя пользователя, HOME- путь к домашнему каталогу. Так же в профиле пользователя хранятся настройки пользователя, файл сценария оболочки, который выполняется автоматически при входе в сеанс и umask. Umask- это права на доступ к файлам и каталогам по умолчанию.

Файл ~/.bashrc — это ресурс оболочки (для пользователя), выполняется только при запуске оболочки из командной строки (в отличие от файлов профиля), содержит дополнительные настройки оболочки /etc/bashrc — общесистемный файл ресурсов оболочки (удобно задавать псевдонимы для команд).

При входе в сеанс bash исполняются:

1. /etc/profile;
2. ~/.bash_profile;
3. ~/.bashrc;
4. /etc/bashrc.

Если выполняется запуск оболочки из командной строки выполняются 3) и 4).

Отчёты об активности пользователей хранятся в файлах как и прочая

информация в ОС Linux. В файле /var/run/utmp хранится список пользователей, находящихся в сеансе, в файле /var/log/wtmp хранится информация об открытых и законченных сеансах пользователей, файл /var/log/lastlog хранит информацию о последних входах в сеанс.

Опции команды who:

- -b — время последней загрузки системы;
- -H — печать заголовка;
- -login — информация о системных процессах, контролирующих вход в сеанс;
- -q — имена всех пользователей в системе и их количество;
- -u — подробная информация о сеансах;
- -a — полная информация о статусе процессов, контролирующих вход в сеанс.

Опции команды lastlog:

- -b — входы в сеанс, ранее указанного количества дней;
- -t — входы в сеанс, за указанный период;
- -u — входы в сеанс пользователя.

Таблица 6. Команды для администрирования пользователей и учетных пользователей

Описание	**Команда**
регистрация нового пользователя	useradd <username>
получить информацию о пользователе	id <username>
выдать настройки команды по умолчанию	useradd –D
изменение учетной записи	usermod <options> <username>
удаление учетной записи	userdel <username>
управление паролями	passwd <options> <username>
создание группы	groupadd <groupname>
удаление группы	groupdel <groupname>
назначение администратора группы	gpasswd –A <username> <groupname>
добавление пользователя к группе, удаление пользователя из группы (администратором группы)	gpasswd –a <username> <groupname>, gpasswd –d <username> <groupname>
задать пароль на вход в группу	gpasswd <password>

для не членов группы	
изменить текущую группу	newgrp <groupname>
выполнить файл профиля	source <profile_file>
список пользователей, находящихся в сеансе	who
информация об открытых и законченных сеансах	last
информация о последних входах в сеанс	lastlog

Практическое задание по теме

1. Ознакомиться с содержимым файлов:
   - /etc/passwd;
   - /etc/shadow;
   - /etc/group.
2. Создать следующие группы:
   - Workers;
   - Teachers;
   - Students.

3. Создать пользователей user_[номер варианта]_ N, где N =1, 2, .., 5, uid учетной записи должен быть равен 1000+N. Пользователей с N равным 1 и 2 добавить в группу workers вручную внеся изменения в конфигурационный файл. После добавления пользователей осуществить проверку файла /etc/group на ошибки. Пользователей с N равным 3, 4 и 5 добавить в группу students при помощи команд администрирования. Проверьте результат, выполнив действия п.1.

4. Создать пользователя teacher_[номер варианта]. В комментарии к учетной записи должны быть Ваше имя и фамилия. uid учетной записи должен быть равен 3000. Пользователя добавить в группу teachers.

5. Для всех пользователей задайте пароли, используя команду passwd.

6. Создать директорию labs в корневом каталоге. В нем создать каталоги library и tests.

7. Создать файлы book_[фамилия студента]_N и поместить их в library.

8. Создать текстовый файл test_[имя студента], и поместить в tests. Файлы должны содержать скрипт на создание пользователя user[номер варианта] и задание ему пароля pass[номер варианта]. Сделайте эти файлы исполняемыми для пользователей группы students.

9. В директории labs создать файл list, который должен содержать список файлов директории /etc.

10. Дать право на изменение файла только пользователю teacher_[номер варианта], а на чтение пользователям группы workers.

11. Настроить права доступа к каталогу library и tests, таким образом, чтобы пользователи группы teachers могли изменять и создавать там файлы, а пользователи группы students имели доступ на чтение.

12. Просмотрите файл /etc/shadow (с правами root). У всех ли пользователей содержимое второго поля выглядит приблизительно одинаково?

13. Какие символы могут содержаться в шифрованной строке пароля в /etc/shadow?

14. Зарегистрируйте пользователя test1, для которого запрещен вход в сеанс, имеющего домашний каталог /home/nouser и являющегося членом групп user и mail. Пользователь должен иметь UID равный 2000.

15. Создайте учетную запись для пользователя test2 с настройками по умолчанию. Проверьте, создался ли домашний каталог пользователя, наполнен ли он файлами и какому пользователю он принадлежит?

16. Измените имя пользователя test2 на test3.

17. Удалите пользователя test3.

18. Помимо файла /etc/default/useradd имеется еще один конфигурационный файл, влияющий на поведение команды useradd. Найдите его и изучите его содержание. Какая настройка позволяет изменять минимальный UID для новых пользователей?

19. Зарегистрируйте пользователя test4 с настройками по умолчанию и установите для него пароль. Изучите содержимое соответствующей записи в /etc/shadow.

20. Установите дату устаревания пароля для пользователя на 31 декабря текущего года. Проверьте, что изменилось в /etc/shadow.

21. Удалите пароль пользователя и проверьте изменения в /etc/shadow.

22. Заблокируйте учётную запись test4.

23. Создайте группу пользователей xusers с GID, равным 1010.

24. Зарегистрируйте себя в качестве участника группы xusers. Проверьте результат выполненного действия.

25. Как изменить имена и GID групп? Измените имя группы на yusers.

26. Сделайте так, чтобы при запуске оболочки из командной строки выдавалось приветствие.

Получение отчётов об активности пользователей

27. Определите, когда последний раз была загружена система.

28. Кто входил в сеанс за последние 2 недели?

Контрольные вопросы

29. Почему в конфигурационных файлах пароли не хранятся в явном виде?

30. Почему не рекомендуется выполнять повседневные операции, используя учётную запись root?

31. В чем отличие механизмов получения особых привилегий su и sudo?

Тема 3. Процессы.

Отложенное и регулярное выполнение заданий

Цель: Познакомиться с понятиями задачи, процесса и потока. Научиться получать информацию о процессах в системе и управлять ими.

Задачи:

1. Изучить процессы и задания.
2. Научиться отложенному выполнению заданий.
3. Научиться автоматизации выполнения регулярных задач.

Процесс — это экземпляр программы, исполняемый процессором или ожидающий этого момента в очереди. Задание — это команда, запущенная на исполнение пользователем с помощью оболочки. Задание может порождать несколько процессов. Каждый процесс работает в своем виртуальном адресном пространстве. Linux является многозадачной ОС. В ней может одновременно выполнятся множество процессов. Процесс обладает атрибутами. Сразу следует отметить, что любой процесс, системный ли, пользовательский, интерактивный или не интерактивный имеет атрибуты, которые делятся на адресное пространство (в памяти) и набор структур, содержащихся в ядре.

Таблица 7. Структура атрибутов процесса

Атрибуты адресного пространства	Атрибуты содержащиеся в ядре
заголовок	таблица распределения памяти
код процесса	текущий статус
данные (data)	приоритет
куча (heap) — область памяти, предназначенная для выделения памяти динамическим переменным	информация об используемых ресурсах
стек (stack) — структура типа LIFO, предназначенная для вызова подпрограмм и хранения некоторых переменных	информация об открытых файлах и сетевых портах
	запись о том, какие сигналы блокируются
	владелец процесса

При создании процесса с ним ассоциируется 3 потока ввода/вывода. Первый поток stdin-стандартный поток вывода, второй stdout так же

является стандартным потоком вывода только имеет другой дескриптор и третий stderr этот поток так же стандартный и используется для вывода ошибок. Процессы создаются другими процессами при помощи системного вызова fork(). Между процессами устанавливается отношения наследства.

Процессы имеют набор идентификаторов: уникальный порядковый номер процесса(PID), родительский процесс (PPID), ID пользователя от имени, которого выполняется процесс (UID), ID группы пользователей от имени, которой выполняется процесс (GID).

Процессы делятся на три категории. Первая категория это процессы ядра, вторая службы и третья прикладные процессы. Часто возникает необходимость выполнить какую-либо команду в заданный момент будущего или же в тот момент, когда загрузка системы минимальна.

Команда at позволяет указать момент времени, в который должна быть исполнена команда. Команда at ставит задания в очередь, которая обслуживается службой atd. Утилита cron позволяет выполнять задание на регулярной основе с заданной периодичностью. Основой регулярного выполнения заданий cron являются таблицы, в которых кодируется периодичность выполнения заданий.

Таблица 8. Типовые команды управления процессами

Описание	**Команда**
Определить общее время работы, время работы на стороне пользователя и на стороне ядра	time <программа>
запуск задания в фоновом режиме	<команда> &
статус фоновых заданий	jobs
перевести задание в интерактивный режим	fg %<номер задания>
	fg %<первые буквы команды>
перевести задание в фоновый режим	bg %<номер задания>
завершение работы фонового задания	kill %<номер задания>
создает копию текущего процесса	fork()

подменяет код текущего процесса кодом исполняемой программы	exec()
ожидает завершения дочернего процесса	wait()
вывести информацию о процессах, связанных с текущим терминалом	ps
вывести более подробную информацию	ps -f
вывести еще более подробную информацию	ps -l
вывести информацию о всех процессах в системе	ps -e или ps -A
получить список PID процессов по имени команды	pgrep <имя команды>
выводится не только PID, но и имя процесса	Pgrep -l <имя команды>
получить список потоков	ps -fLC soffice.bin
постоянный мониторинг процессов	top, htop
данные об активности процесса	sudo strace –p <PID>
выполнить команду в определенный момент времени	at
указать команды в файле	at <время> -f <filename>
получить список заданий	atq
удалить задание из списка	atrm <номер задания>
операции с таблицами заданий	crontab
управление службами	service <имя службы> start | stop | restart

Cron — это программа, выполняющая задания по расписанию. Позволяет неоднократный запуск заданий. Т.е. задание можно запустить в определенное время или через определенный промежуток времени. Формат и значения полей, для пользовательских файлов crontab, будут приведены в конце статьи.

При загрузке системы, запускается демон cron и проверяет очередь заданий at и заданий пользователей в файлах crontab. При запуске, демон cron сначала проверяет каталог /var/spool/cron на наличие файлов crontab, файлы crontab имеют имена пользователей, соответствующие именам пользователей из /etc/passwd Каждый пользователь может иметь только

один файл crontab, записей в файле может быть несколько.

Другими словами - файлы crontab содержат инструкции для демона cron, который запустит задание(я) описаное в файле crontab. Все файлы crontab из каталога /var/spool/cron загружаются в память, одновременно с ними загружаются файлы из /etc/cron.d После этого демон cron загружает содержимое файла /etc/crontab При стандартных настройках, содержимое /etc/crontab имеет следующий вид:

SHELL=/bin/bash

PATH=/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=root

HOME=/

run-parts

01 * * * * root run-parts /etc/cron.hourly

02 4 * * * root run-parts /etc/cron.daily

22 4 * * 0 root run-parts /etc/cron.weekly

42 4 1 * * root run-parts /etc/cron.monthly

Информация файла указывает, что:

- содержимое каталога /etc/cron.hourly будет запускаться каждый час на первой минуте часа.

- содержимое каталога /etc/cron.daily будет запускаться каждый день на второй минуте четвертого часа

- содержимое каталога /etc/cron.weekly будет запускаться каждое воскресенье на 22-ой минуте 4-го часа.

- содержимое каталога /etc/cron.monthly будет запускаться каждый первый день месяца на 42-ой минуте 4-го часа.

SHELL=/bin/bash означает использовать для запуска команд /bin/bash, если переменная не указана, то значение будет взято из /etc/passwd для пользователя являющимся владельцем файла.

HOME=/ корневой каталог для пользователя (параметр не

обязательный) При необходимости доступа к специальным свойствам интерпретатора, значения переменных SHELL и HOME можно изменить, не зависимо от того, что прописано в /etc/passwd

MAILTO=root означает кому отсылать сообщение о результате работы команд.

Все содержимое из этих каталогов будет запускаться с правами доступа пользователя root и файлы должны иметь права доступа на выполнение. Поэтому перед размещением файлов в одном из этих каталогов необходимо убедиться, что сценарии не насесут вред системе.

После того, как демон cron запущен и прочёл содержимое всех файлов crontab, он бездействует, просыпаясь каждую минуту и проверяя не требуется ли запуск какой-либо команды в данную минуту, или не появился ли новый файл crontab который необходимо обработать. Демон cron определяет изменения по времени модификации файлов или каталогов, такое его свойство избавляет от необходимости перезапуска демона.

Как отмечалось выше, размещение файлов для cron в каталогах

/etc/cron.hourly

/etc/cron.daily

/etc/cron.weekly

/etc/cron.monthly

доступно только пользователю root, для использования файлов crontab пользователями, нужно использовать команду crontab. Команда служит для создания, изменения и добавления файла для демона cron.

Рассмотрим пример создания файла crontab для пользователя user, домашняя директория пользователя - /home/user

Задача: запускать каждую минуту файл /home/user/mail, который будет отправлять почту

#содержимое файла mail (файл должен быть с правами на запуск!

например -rwxr-xr-x)

#!/bin/bash

mess="test cron"

echo "$mess" |mutt -s "subj" -m application/octet-stream bob@server.ru

1. Создаем временный файл /home/user/test со следующим содержимым:

SHELL=/bin/bash

MAILTO=user

0-59 * * * * /home/user/mail

2. Запускаем в терминале команду crontab /home/user/test

После этого в каталоге /var/spool/cron будет создан файл "user" примерно с таким содержимым:

DO NOT EDIT THIS FILE - edit the master and reinstall.

(/home/user/test installed on Mon Mar 29 02:31:34 2004)

(Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)

SHELL=/bin/bash

MAILTO=user

0-59 * * * * /home/user/mail

и файл /home/user/mail будет запускаться демоном cron каждую минуту.

Доступ в каталог /var/spool/cron непривилегированому пользователю закрыт, что бы посмотреть юзером "user" есть ли у него файл crontab, достаточно набрать команду crontab -l , если файл существует-будет показано его содержимое.

Для удаления файла используется команда crontab -r

Для редактирования crontab -e

Для управления файлами crontab пользователем root используется синтаксис:

crontab -u user filename

где user — имя пользователя (если опция u не задана - будет обработан crontab того пользователя, который запустил команду crontab).

Каждая команда в пользовательском файле crontab занимает одну строку и состоит из шести полей. Пользовательские файлы crontab находятся в каталоге /var/spool/cron

Общий формат команды:

минута час день_месяца месяц день_недели команда

Допустимые значения:

минута: от 0 до 59

час: от 0 до 23

день_месяца: от 1 до 31

месяц: от 1 до 12 (можно три буквы из названия месяца, регистр не имеет значения, от jan до dec)

день_недели: от 0 до 6 (0 это воскресенье, можно указывать от sun до sat).

Каждое из полей даты и времени может быть обозначено символом *, что будет соответствовать любому возможному значению. Для этих полей можно указывать диапазоны значений, разделенных дефисом, например:

* 5 4-10 0-3 * echo "HELLO" - печать HELLO в 5:00 на 4,5,6,7,8,9,10 дни января, февраля, марта и апреля.

Пошаговая запись:

* */2 * * sat echo "HELLO" - печать HELLO каждый четный час, каждую субботу.

Равнозначная предыдущему примеру запись (списком):

* 0,2,4,6,8,10,12,14,16,18,20,22 * * sat echo "HELLO" - печать HELLO каждый четный час, каждую субботу.

То же самое с указанием диапазона:

* 0-23/2 * * sat echo "HELLO" -печать HELLO каждый четный

час, каждую субботу.

Для отладки задания cron, можно перенаправить результат в файл.

Пример:

0-59 * * * * /home/user/mail 2>/tmp/tmp.cron

Если при запуске команды /home/user/mail возникнут ошибки, то они будут записаны в файл /tmp/tmp.cron и вы всегда сможете узнать причину. В случае перенаправления вывода в файл, письмо, юзеру указаному в переменной MAILTO отправлено не будет.

Посмотреть информацию о всех командах запускаемых демоном cron можно в файле /var/log/cron. В нем записано время запуска всех заданий cron за предыдущий день:

Mar 29 04:03:00 rst CROND[4434]: (user) CMD (/home/user/mail)

Mar 29 04:03:59 rst CROND[4493]: (user) CMD (/home/user/mail)

Mar 29 04:05:00 rst CROND[4507]: (user) CMD (/home/user/mail)

Mar 29 04:06:00 rst CROND[4549]: (user) CMD (/home/user/mail)

Практическое задание по теме

Процессы, сигналы и приоритеты

1.Найдите пустые файлы в домашнем каталоге в фоновом режиме.

2.Запустите в фоновом режиме два задания: sleep 200 и sleep 2000, выведите информацию о состоянии заданий.

3.Снимите с выполнения второе задание, выведите информацию о заданиях.

4.Выполните команду exec ls R /etc. Изучите её поведение.

5.Запустите порождённую оболочку bash. Исследуйте, посылая родительской оболочке сигналы TERM, INT, QUIT и HUP, что при этом происходит?

6.От имени обычного пользователя пошлите сигнал KILL любому процессу, запущенному от имени другого пользователя. Что произойдет?

7.Запустите в фоновом режиме команду sleep 1000. Проверьте, на какие сигналы из следующих: TERM, INT, QUIT и HUP, реагирует эта команда.

8.Запрограммируйте оболочку так, чтобы при получении ей сигнала TERM создавался файл pwd.txt, содержащий информацию о текущем каталоге.

9. Запустите порожденную оболочку. Работает ли в ней созданный обработчик?

10. От имени обычного пользователя попытайтесь запустить оболочку bash со значением nice number, равным 1. Какое сообщение выводится?

11.От имени супер-пользователя запустите команду индексирования базы данных поиска в следующем виде: time nice n 19 updatedb. Затем выполните такую же команду, в которой значение nice number для updatedb будет 5. Сравните полученные результаты.

Отложенное и регулярное выполнение заданий (at, cron)

12. Проверьте, запущены ли какие-нибудь задания в cron или at для пользователя root.

13. Сделайте при помощи cron так, чтобы буферы ядра на диске очищались каждый час.

14. Сделайте при помощи cron запуск команды updatedb раз в сутки.

15. Сделайте сигнал окончания пары при помощи команды at – чтобы прозвенел звонок (воспроизвести звуковой файл или использовать ASCII символ beep).

16. Что произойдет, если в файл temp, используя at, одновременно записать разные данные?

17. Выясните приоритет выполнения одновременных заданий для at. Одновременный запуск организуйте разными способами, например, через

now+1 minutes и HH:MM

18. С помощью cron сделайте так, чтобы каждые 10 минут убивался браузер firefox.

19. Очистите список заданий для cron и at.

20. Узнайте ip адрес компьютера вашего соседа, зайдите на него по ssh и сделайте при помощи cron так, чтобы каждую минуту раздавался сигнал. Как от этого защититься?

Текстовые файлы и потоки

21. Получите список всех процессов и перенаправьте вывод в файл ps.txt

22. Выполните команду поиска всех обычных файлов в каталоге /usr/share так, чтобы найденные имена файлов были записаны в файл SysComm в домашнем каталоге, а поток ошибок был записан в нуль-устройство /dev/null.

23. Выполните ту же команду, но так, чтобы потоки вывода и ошибок были записаны в файл SysComm.

24. Допишите в конец файла SysComm информацию о текущей дате и времени, выводимую командой date.

25. Выведите список всех процессов в файл ps.txt и на экран одновременно.

26. Определите из man ascii восьмеричный код системного звукового сигнала (bell) и выведите его с помощью команды echo.

27. Получите столбец относительных приоритетов всех процессов в системе.

28. С помощью sed получите список только тех процессов, которые связаны с каким-либо терминалом (фильтр по строке tty).

29. В полученном списке процессов с помощью sed замените все строки tty на terminal.

30. При помощи awk удалите из полученного списка второй столбец.

31. При помощи awk пронумеруйте полученный список.

32.Определите, какая опция diff рекурсивно сравнивает каталоги.

33.Сравните содержимое домашних каталогов двух пользователей.

34.Создайте два файла ps1.txt и ps2.txt, содержащие полный список текущих процессов. Получите файл patch.txt с отличиями ps1.txt от ps2.txt. Удалите ps2.txt и восстановите его содержимое по файлам ps1.txt и patch.txt.

35.Выведите содержимое файла .bashrc, заменив каждый символ табуляции двумя пробелами.

36.Получите список групп пользователей в системе, отсортированный по GID в обратном числовом порядке.

37. С помощью find, head и sort получите список из десяти файлов в домашнем каталоге, занимающих наибольшее дисковое пространство. (2)

38. Используя файл /etc/passwd, содержащий данные об учетных записях пользователей, сколько пользователей зарегистрировано в системе. Команда должна выводить число пользователей.

39.Определите, для скольких пользователей оболочка по умолчанию — bash. Вывести число таких пользователей.

40.Сколько имеется пользователей, UID которых больше 100?

41.Выведите пронумерованный список файлов в текущем каталоге.

42.Задайте шаг нумерации, равный двум.

43.Выведите последние три строки файла /etc/passwd, заменив разделители — двоеточия на разделители — вертикальные черты (|).

44.Разделите файл /etc/passwd на части по 10 строк, находящиеся в текущем каталоге. Имена файлов должны начинаться с passwd. Выведите пронумерованный список файлов в текущем каталоге.

45.Создайте в домашнем каталоге несколько нескрытных пустых каталогов. Удалите их, используя команду xargs.

Тема 4. Написание сценариев Bash

Цель: научиться составлять простые сценарии Bash и познакомиться с основными конструкциями языка сценариев. Изучить команды ветвления, научиться использованию циклов и функций.

Задачи:

1. Изучить синтаксис написание скриптов;
2. Научиться вызывать и писать сценарии.

Сценарий оболочки — это текстовый файл, содержащий программу, состоящую из системных и встроенных команд, для сценариев принято устанавливать расширение .sh. Вызвать сценарий можно явно либо нет (см. таблицу 7). Bash-скрипты могут пригодится для установления правил файервола при загрузке системы, выполнять резервное копирование настроек и данных, добавлять почтовые ящики в почтовый сервер (точнее в базу mysql), запускать в определенное время программу, которая сканирует логи прокси-сервера и выдает удобный web-отчет по количеству скачанного трафика, отправлять на почту информацию.

Любой bash-скрипт должен начинаться со строки: #!/bin/bash. В этой строке после #! указывается путь к bash-интерпретатору, поэтому если он у вас установлен в другом месте(где, вы можете узнать набрав whereis bash). Комментарии начинаются с символа # (кроме первой строки). При написании скриптов используется экранирование. Экранирование –это механизм защиты специальных символов оболочки от интерпретации. Способы экранирование:

- 'все отлично'; -кроме";
- "даже лучше"; -кроме" ",$,\,`;
- без\ кавычек.

В скриптах bash переменные не имеют типа и в большинстве случаев интерпретируются как строки. Что бы объявить типизированные переменные следует воспользоваться встроенной функцией declare.

Функция declare обладает следующими опциями:

- -a — массив;
- -i — целое число;
- -r— только для чтения;
- -x— переменная на экспорт.

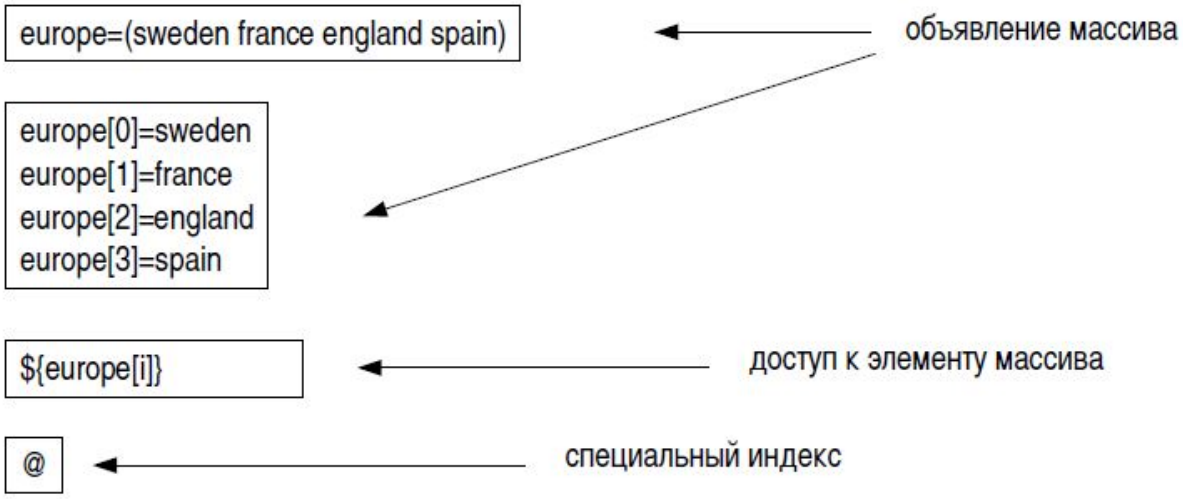

Рисунок 4 - Работа с массивами в скриптах bash

Позиционные параметры в скриптах bash позволяют получить имя сценария и переданные ему в командной строке параметры: имя команды ($0), значения девяти аргументов командной строки ($1,…,$9). Специальные параметры:

- $* -строка, составленная из значений всех аргументов командной строки;
- $@ -тоже, что и $*, только значения разделены пробелами;
- $# -количество аргументов командной строки;
- $? -код возврата предыдущей команды;
- $$ -PID оболочки.

Команда test позволяет проверить заданные условия. При помощи команды test можно выполнить такие проверки как: проверка файлов на предмет выполнения заданных условий, сравнение файлов, проверить установки опций оболочки, сравнить строки, сравнить целые числа. Если

тест выполнен успешно, команда test передает нулевой код возврата. Так, используя команду test -e, можно проверить существование файла.

Часто используемые опции команды test:

- -e — файл существует;
- -f — файл является обычным файлом;
- -d — файл является каталогом;
- h или L — файл является символической ссылкой;
- r — файл доступен для чтения;
- w — файл доступен для записи;
- x — файл доступен для исполнения;
- s — файл не пуст.

Операторы условного исполнения команд && и || могут быть применены для управления потоком исполнения команд. Они позволяют исполнять команды в зависимости от успешности выполнения предыдущих команд. Если оператор && установлен между двумя командами, то вторая из них будет исполнена только в случае успеха предыдущей. Оболочка предоставляет также специальные команды if, case, for, while, util,function.

Таблица 9. Синтаксис некоторых специальных команд bash скриптов

Команда	**Синтаксис**
if	if [<условие>] then команды elif [<условие>] then команды else команды fi
case	case слово in шаблон1) команды ;;

	шаблон2) команды ;; esac
for-цикл перебора значений	for <имя> in <список> do команды, которые используют переменную $<имя> done
while-выполняется, пока истинно условие	while [условие]; do команды done
until-выполняется, пока условие ложно	until [условие]; do команды done

function	function имя()
	{
	команды
	}

Функции должны быть описаны в файле до первого вызова. Часто функции описывают в отдельных файлах и пользуются inline-подстановкой.

Рисунок 5 - Передача параметров в функции

Отладка сценариев bash возможна несколькими способами:

1. Вывод при помощи echo;
2. Использовать опции оболочки;
3. noexec — не выполнять команды, только проверять синтаксис (n);
4. verbose — выводить команды перед запуском (v);
5. xtrace — выводить команды после обработки (x).

Таблица 10. Команды для работы со сценариями

Описание	Команда
явный вызов оболочки	bash program.sh
не явный вызов оболочки	./program.sh
сделать файл программы доступным для чтения и исполнения для всех пользователей	chmod a+rx program.sh
сдвигает позиционные параметры на n (по умолчанию на 1 вправо). Сдвинуть параметры обратно нельзя!	shift <n>
устанавливает значения позиционных параметров	set
изменить только третий параметр	set $1 $2 newvalue $4
проверка существования файла	test e <filename>
выполнение второй команды только в случае успеха первой	<команда1> && <команда2>
выполнение второй команды только в случае неудачи первой	<команда1> || <команда2>

Практическое задание по теме

Сценарии bash

1. Вывести любое сообщение с помощью команды echo перенаправив вывод:

- в несуществующий файл с помощью символа >;
- в несуществующий файл с помощью символа >>;
- в существующий файл с помощью символа >;
- в существующий файл с помощью символа >>;

Объяснить результаты.

2. Переадресовать стандартный ввод для команды cat на файл.

3. Вывести сообщение с помощью команды echo в канал ошибок. Создать файл myscript:

```
#!/bin/sh
echo stdout
```

```
echo stderr>&2
exit 0
```

Запустить его:

- без перенаправления (sh myscript);

- перенаправив стандартный вывод в файл, просмотреть содержимое файла (sh myscript > file1);

- перенаправить стандартный канал ошибок в существующий и несуществующий файлы с помощью символов > и >> (а тут и дальше уже сами :));

- перенаправив стандартный вывод в файл 1, стандартный канал ошибок - в файл 2;

- перенаправив стандартный вывод и стандартный канал ошибок в файл 3;

- перенаправив стандартный вывод в файл 4 с помощью символа >, а стандартный канал ошибок в файл 4 с помощью символа >>;

Объяснить результаты.

4. Вывести третью строку из последних десяти строк отсортированного в обратном порядке файла /etc/group.

5. Подсчитать при помощи конвейера команд количество блочных устройств ввода-вывода, доступных в системе.

6. Написать скрипт, выводящий на консоль все аргументы командной строки, переданные данному скрипту. Привести различные варианты запуска данного скрипта, в том числе без непосредственного вызова интерпретатора в командной строке.

7. Напишите сценарий, проверяющий имя текущего каталога и выводящий сообщение об ошибке, если оно короче пяти символов.

8. Требуется проверить, является ли файл обычным или он является каталогом. Если это обычный файл, то сценарий должен выводить имя файла и его размер. В случае, если размер файла превышает килобайт, то

размер должен выводиться в килобайтах. Если размер превышает мегабайт — в мегабайтах.

9. Напишите сценарий, выводящий посекундно в цикле имена файлов текущего каталога и их порядковый номер.

10. Напишите сценарий, который генерирует тысячу файлов 1.txt .... 1000.txt, и в каждый файл записывает подряд 100 чисел N, где N = порядковый номер файла. Затем скрипт должен соединить в один файл все файлы с четными номерами (even.txt) и в другой файл — все файлы с нечетными номерами (odd.txt).

11.Написать командный файл, реализующий меню из трех пунктов: 1-ый пункт - ввести пользователя и вывести на экран все процессы, запущенные данным пользователем; 2-ой пункт - показать всех пользователей, в настоящий момент, находящихся в системе; 3-ий пункт – завершение работы.

12. Написать командный файл, реализующий меню из трех пунктов: 1-ый пункт - вывести всех пользователей, в настоящее время, работающих в системе; 2-ой пункт – послать сообщение пользователю, имя пользователя, терминал и сообщение вводятся с клавиатуры; 3-ий пункт – завершение работы.

13. Написать командный файл, реализующий меню из трех пунктов: 1-ый пункт - показать все процессы пользователя, запустившего данный командный файл; 2-ой пункт – послать сигнал завершения процессу текущего пользователя(ввести PID процесса); 3-ий пункт – завершение работы.

14. Написать командный файл, посылающий сигнал завершения процессам текущего пользователя. Символьная маска имени процесса вводится с клавиатуры.

15. Написать командный файл подсчитывающий количество определенных процессов пользователя (Ввести имя пользователя и

название процесса).

16. Реализовать меню из двух пунктов: 1-ый пункт – определить количество запущенных данным пользователем процессов bash (предусмотреть ввод имени пользователя); 2-ой пункт – завершить все процессы bash данного пользователя.

17. Реализовать Меню из трех пунктов: 1-ый пункт поиск файла в каталоге <Имя файла> и <Имя каталога> вводятся пользователем; 2-ой пункт – копирование одного файла в другой каталог - <Имя файла> и <Имя каталога> вводятся; 3-ий пункт – завершение командного файла.

18. Написать командный файл который в цикле по нажатию клавиши выводит информацию о системе, активных пользователях в системе, а для введенного имени пользователя выводит список активных процессов данного пользователя.

19. Реализовать командный файл который при старте выводит информацию о системе, информацию о пользователе, запустившем данный командный файл, далее в цикле выводит список активных пользователей в системе – запрашивает имя пользователя и выводят список всех процессов bash запущенных данным пользователем.

20. Реализовать командный файл, позволяющий в цикле посылать всем активным пользователям сообщение – сообщение вводится с клавиатуры. Командный файл при старте выводит имя компьютера, имя запустившего командный файл пользователя, тип операционной системы, IP-адрес машины.

21. Реализовать командный файл, позволяющий в цикле посылать всем активным пользователям (исключая пользователя, запустившего данный командный файл) сообщение – сообщение вводится с клавиатуры. Командный файл при старте выводит имя компьютера, имя запустившего командный файл пользователя, тип операционной системы, список загруженных модулей.

22. Реализовать командный файл который при старте выводит информацию о системе, информацию о пользователе, запустившем данный командный файл, далее в цикле выводит список активных пользователей в системе – запрашивает имя пользователя и выводят список всех терминалов, на которых зарегистрирован этот пользователь.

23. Реализовать командный файл, который выводит: дату, информацию о системе, текущий каталог, текущего пользователя, настройки домашнего каталога текущего пользователя, далее в цикле выводит список активных пользователей – запрашивает имя пользователя и выводит информацию об активности данного пользователя.

24. Реализовать командный файл, который выводит: дату в формате день – месяц – год – время, информацию о системе в формате: имя компьютера : версия ОС : IP адрес : имя текущего пользователя : текущий каталог, выводит настройки домашнего каталога текущего пользователя и основные переменные окружения. Далее в цикле выводит список активных пользователей – запрашивает имя пользователя и выводит информацию об активности введенного пользователя.

25.Реализовать командный файл, реализующий символьное меню (в цикле):

1 Пункт: Вывод полной информации о файлах каталога: Ввести имя каталога для отображения.

2 Пункт изменить атрибуты файла: файл вводится с клавиатуры по запросу, атрибуты, которые требуются установить тоже вводятся. После изменения атрибутов вывести на экран расширенный список файлов для проверки установленных атрибутов.

3 Выход.

При старте командный файл выводит информацию об имени компьютера, IP-адреса, и список всех пользователей, зарегистрированных в данный момент на компьютере.

26. Реализовать командный файл, реализующий символьное меню (в цикле):

1 Пункт: Вывод полной информации о файлах каталога: Ввести имя каталога для отображения.

2 Пункт: создать командный файл: файл вводится с клавиатуры по запросу, далее изменяются атрибут файла на исполнение, затем вводится с клавиатуры строка которую будет исполнять командный файл. После изменения атрибутов вывести на экра расширенный список файлов для проверки установленных атрибутов и запустить созданный командный файл.

3 Выход.

При старте командный файл выводит информацию об имени компьютера, IP-адреса, и список всех пользователей зарегистрированных в данный момент на компьютере.

27. Написать командный файл реализующий символьное меню

1 Пункт: работа с информационными командами(реализовать все основные информационные команды)

2 Пункт: Копирование файлов: в этом пункте выводится информация о содержимом текущего каталога, далее предлагается интерфейс копирования файла: ввод имени файла и ввод каталога для копирования. По выполнению пункта выводится содержимое каталога, куда был скопирован файл и выводится содержимое скопированного файла.

3 Пункт: Выход.

28. Написать скрипт с использованием цикла for, выводящий на консоль размеры и права доступа для всех файлов в заданном каталоге и всех его подкаталогах (имя каталога задается пользователем в качестве первого аргумента командной строки).

29. Написать скрипт, находящий в заданном каталоге и всех его

подкаталогах все файлы, владельцем которых является заданный пользователь. Имя владельца и каталог задаются пользователем в качестве первого и второго аргумента командной строки. Скрипт выводит результаты в файл (третий аргумент командной строки) в виде: полный путь, имя файла, его размер. На консоль выводится общее число просмотренных файлов.

30. Написать скрипт поиска одинаковых по их содержимому файлов в двух каталогах, например, Dir1 и Dir2. Пользователь задаёт имена Dir1 и Dir2 в качестве первого и второго аргумента командной строки. В результате работы файлы, имеющиеся в Dir1, сравниваются с файлами в Dir2 по их содержимому. На экран выводятся число просмотренных файлов и результаты сравнения.

Тема 5. Файловая система

Цель: Познакомиться с организацией хранения данных на жёстких магнитных дисках, научиться создавать дисковые разделы, файловые системы и разделы подкачки, монтировать файловые системы и поддерживать их в рабочем состоянии.

Как уже говорилось, Linux — система многопользовательская и вопрос об организации разграничения доступа к файлам и каталогам является одним из если не главных то важных вопросов, который должна решать операционная система. Файловая система ОС Linux состоит из «Суперблока», «Массива индексных дескрипторов» и блоков. Суперблок состоит из информации необходимой для монтирования файловой системы:

- тип файловой системы;
- размер и количество;
- блоков;
- количество индексных дескрипторов;
- время последнего монтирования;
- флаг монтирования;
- счетчик числа монтированний;
- список свободных блоков и индексных дескрипторов.

Массив индексных дескрипторов содержит данные о файлах:

- владелец и группа пользователей;
- права доступа;
- тип;
- количество имен;
- дата доступа;
- дата модификации;
- дата изменения метаданных;

- количество блоков;
- указатели на блоки.

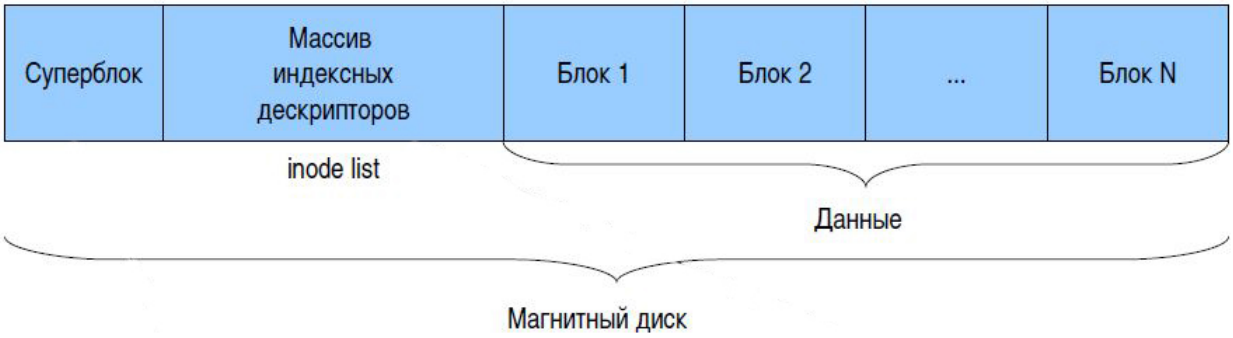

Рисунок 6 - Устройство файловой системы ОС Linux

Каталог — особый тип файлов, хранящий таблицу имен файлов и связанных с ними индексных дескрипторов. На один и тот же индексный дескриптор могут указывать несколько файлов. Если у файла несколько имен, то говорят, что между этими именами существует жесткая связь (hard link). У каталогов минимум два имени: Обычное (/etc) . - имя текущего каталога. При появлении дочернего каталога количество имен у родительского увеличивается на 1 (каждый дочерний содержит имя .. - имя родительского каталога). На один и тот же индексный дескриптор может указывать несколько имен файлов. Это фиксируется счетчиком имен файла (link counter) в индексном дескрипторе. Если у файла имеется несколько имен, то говорят, что между этими именами существует жесткая связь (hard link). Все имена файла абсолютно эквивалентны, и изменение содержимого этих файлов будут синхронным. Особые значения номеров inode:

- 0 — свободный индексный дескриптор;
- 1 — для файловых систем, порожденных ядром: /proc и /sys;
- 2 — inode корневой файловой системы и точек монтирования файловых систем.

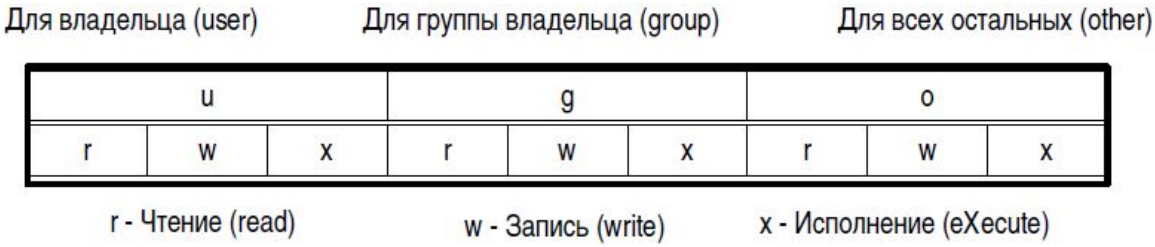

Рисунок 7 - Права доступа к файлу

x — поиск: право чтения метаданных файлов каталога, а также право входить в каталог командой cd;

r — право на чтение имен файлов в каталоге (ls), если нет права x, то нельзя получить подробную информацию о файлах;

w — право на запись в каталог, без права x нельзя осуществлять запись, т.к. требуется доступ к метаданным.

0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	1
	...			...			...	
1	1	1	1	1	1	1	1	1

Рисунок 8 - Двоичная запись прав доступа к каталогу

Права на каталоги должны быть нечетными, либо отсутствовать:

- 0 — прав нет;
- 1 (x) — можно входить в каталог и обращаться к файлам в нем;
- 3 (wx) — можно входить в каталог, разрешены операции с файлами, однако получить список файлов нельзя;
- 5 (rx) — можно входить в каталог, получать подробную информацию о файлах, обращаться к файлам, нельзя переименовывать и удалять файлы;
- 7 (rwx) — полные права.

Для изменения прав доступа к файлу используется команда chmod. Ее можно использовать в двух вариантах. В первом варианте вы должны явно указать, кому какое право даете или кого этого права лишаете: [user]$

chmod wXp имя-файла где вместо символа w подставляется:

- либо символ u (т. е. пользователь, который является владельцем);
- либо g (группа);
- либо o (все пользователи, не входящие в группу, которой принадлежит данный файл);
- либо a (все пользователи системы, т. е. и владелец, и группа, и все остальные).

Вместо X ставится:

- либо + (предоставляем право);
- либо – (лишаем соответствующего права);
- либо = (установить указанные права вместо имеющихся),
- Вместо p — символ, обозначающий соответствующее право:
- r (чтение);
- w (запись);
- x (выполнение).

Выполнять команды chown, chgrp разрешено только для пользователям root.

Жесткая связь имеет место между несколькими именами файла, указывающими на одни и те же метаданные (inode). Наличие у файла другого имени, т.е. наличие жесткой связи, можно обнаружить, изучив вывод команды ls –l. Следовательно, любые изменения с файлом, зеркально отразятся на файле, жестко связанном с ним. Жесткие связи можно устанавливать только в пределах одной файловой системы. Для создания жесткой связи с файлом применяется команда ln. Первый аргумент команды — имя файла, а второй — имя жесткой связи с ним.

Символические ссылки — это особый вид файлов, представляющий собой указатели на другие файлы. Символические ссылки создают командой ln -s, где первый аргумент — имя уже существующего файла, а

второй аргумент — либо имя символической ссылки, либо каталога, где будет образован файл символической ссылки с таким же именем, что и исходный файл. Символические ссылки можно устанавливать на каталоги. При создании символических ссылок в другом каталоге следует указывать полное имя исходного файла, если это требование не выполняется, то:

- либо ссылка будет оборванной/висящей, т. е. символическая ссылка будет указывать на несуществующий файл;
- либо ссылка будет указывать на существующий файл в целевом каталоге, имя которого (случайно или намеренно) совпадет с исходным файлом, однако ссылка будет указывать не на исходный файл.

Символическая ссылка может оказаться оборванной в случае, если:

- файл, на который она указывает, перемещен, переименован или удален;
- нет достаточных прав доступа на файл, указываемый символической ссылкой;
- файл находится в файловой системе, которая сейчас не смонтирована.

Современные жесткие диски состоят из одного или нескольких магнитных дисков и магнитных головок (head). Каждая магнитная поверхность диска разбита на дорожки (track). Дорожки одного диаметра на всех магнитных поверхностях образуют цилиндр (cyl). Количество дорожек равно произведению количества цилиндров на количество головок:

$$track = cyl \times head.$$

Каждая дорожка разбита на секторы (sect), стандартный размер которых составляет 512 байтов. Объем диска в байтах равен:

$$capacity = cyl \times head \times sect \times 512.$$

Стандартное количество секторов в дорожке равняется 63. Поскольку количество байтов в секторе и секторов в дорожке являются

постоянными величинами, то основными параметрами диска (так называемой геометрией) являются количества цилиндров cyl и головок head.

Большинству устройств (за исключением, разве что, сетевых интерфейсов) в Linux соответствуют специальные файлы устройств, размещающиеся в каталоге /dev. Эти файлы не используют блоки данных в файловой системе. Команда ls –l для файлов устройств вместо размера выводит два параметра: мажор и минор. Мажор — это номер драйвера для этого вида устройств в ядре Linux, а минор — номер экземпляра устройства, обслуживаемого данным драйвером. Файлы устройств бывают блочными (в выводе первый ls –l символ b) и символьными (в выводе первый ls –l символ c). Блочные устройства в Linux — все те, информацию на которые можно записывать исключительно блоками и считывать так же. Блочные файлы устройств — интерфейс к устройствам, имеющим файловую систему. Примерами символьных устройств являются терминал, клавиатура и мышь. Обмен информацией с такими устройствами осуществляется посимвольно. Схема именования файлов устройств фиксируется специальным соглашением, которое можно найти в каталоге с исходным кодом ядра (обычно/usr/src/linux) в файле Documentation/devices.txt. Соглашение устанавливает, что файлы устройств IDE жестких дисков называются:

- /dev/hda — Primary Master;
- /dev/hdb — Primary Slave;
- /dev/hdc — Secondary Master;
- /dev/hdd — Secondary Slave.

Имена SCSI-дисков (и SATA) начинаются с sd, первому SCSI-диску соответствует файл устройства /dev/sda, второму — /dev/sdb и т. д. в соответствии со SCSI ID данного жесткого диска. В ядре Linux для IDE-дисков предусматривается мажор 3, а минор зависит от номера раздела: 0

— для всего диска, 1 — для первого раздела и т. д. Для разделов SCSI и SATA дисков (мажор 8) предусматривается аналогичный порядок.

Интерактивная утилита fdisk позволяет оперировать дисковыми разделами жестких магнитных дисков, предоставляя специальный набор собственных команд. Имеется также команда sfdisk, которая, в отличие от fdisk, является утилитой для не интерактивного редактирования таблицы разделов на жестком диске. При неосторожном ее использовании легко можно утратить все данные на жестком диске, т. к. данные для редактирования таблицы разделов задаются в командной строке sfdisk. Весьма популярна программа cfdisk, которая позволяет редактировать таблицу разделов диска с помощью простого интерфейса меню. Работать с командой fdisk может только администратор. Для редактирования таблицы разделов на диске эту команду следует запустить в интерактивном режиме, указав файл устройства для требуемого жесткого диска.

Список основных команд утилиты fdisk:

- q — завершение работы без сохранения изменений;
- l — вывод списка возможных типов разделов;
- d — удаление раздела (для удаления будет запрошен номер удаляемого раздела);
- n — создание нового раздела;
- t — установка типа вновь созданного раздела (для установки типа необходимо ввести номер типа раздела);
- a — выбор активного раздела;
- w — запись измененной таблицы разделов.

Для возможности работы с разделом накопителя на магнитных дисках в разделе должна быть создана файловая система, т. е. должно быть произведено форматирование раздела. При этом в новой файловой системе формируется суперблок, создается массив индексных дескрипторов и выделяется пространство для блоков данных. Стандартной файловой

системой в Linux является ext2. Однако помимо ext2 широко используются следующие файловые системы:

- ext3 — современная версия ext2 с поддержкой журналирования, индексированием каталогов и с улучшенными показателями быстродействия;
- ext4 — дальнейшее развитие ext3 с возможностью создавать очень большие файловые системы (1 экзабайт = 1018 байт), ориентированная на работу с экстентами (непрерывная последовательность блоков, принадлежащих файлу) с улучшенными параметрами надежности журналирования;
- XFS — высокопроизводительная файловая система для очень больших объемов хранимой информации (8 экзабайт), разработанная в Silicon Graphics;
- JFS — высокопроизводительная файловая система от IBM, используемая при необходимости хранения больших объемов информации (32 петабайта = 32 × 1015 байт).

В Linux реализована поддержка и других файловых систем, однако основные используемые файловые системы: ext2, ext3 и ext4, две последние из которых поддерживают журналирование (journaling). Это значит, что на диске выделяется место для хранения информации о том, какие операции и с какими блоками в файловой системе выполнялись в последнее время. Наличие журналирования позволяет быстрее и надежнее обеспечить восстановление файловой системы после сбоя. Операции записи на диск в журналируемых файловых системах оформлены в виде транзакций, которые либо завершаются целиком, либо не принимаются вовсе. Использование журналирования существенно повышает целостность данных и уменьшает вероятность потери данных при сбое. Для создания файловой системы применяется команда mkfs, с аргументом — файлом устройства, соответствующим разделу жесткого диска. Команда

mkfs по умолчанию создает файловую систему ext2. Так, для создания файловой системы ext2 на втором первичном разделе SATA жесткого диска следует выполнить такую команду от имени супер пользователя.

Целостность ФС может быть нарушена в результате сбоя (отключение питания, аппаратная неисправность). Сбой характеризуется тем, что информация в кэше для данной ФС не успевает синхронизироваться с состоянием ФС. Проблемы:

- искажение или потеря информации в блоках данных;
- появление занятых блоков данных, на которые не указывает ни один дескриптор;
- наличие перекрестных ссылок на блоки данных;
- появление метаданных с ненулевым счетчиком ссылок, на которые не ссылаются никакие файлы;
- появление противоречивых записей в каталоге.

Для восстановления целостности файловых систем в Linux используется утилита fsck, которая аналогично mkfs является надстройкой над специализированными утилитами для различных файловых систем. В Linux все файловые системы на блочных устройства сведены в единую файловую систему посредством каталогов — точек монтирования.

Процесс подключения файловой системы на блочном устройстве к корневой файловой системе называется монтированием. Монтирование — процесс подключения файловой системы, существующей на дисковом или ином блочном устройстве, к корневой файловой системе. За исключением ФС, для которых имеются специальные настройки в /etc/fstab, монтирование осуществляет только root /mnt — каталог для точек монтирования файловых систем /media — каталог для точек монтирования ФС на съемных носителях.

Раздел или файл подкачки необходим при работе GNU/Linux для обеспечения временного перемещения страниц памяти из ОЗУ в этот

раздел или файл. Такое перемещение бывает крайне необходимо при недостатке физической памяти. При этом страницы памяти, временно не используемые, но, тем не менее, необходимые для работы операционной системы или приложений, временно перемещаются в раздел подкачки. Процесс обмена страницами памяти между ОЗУ и разделом подкачки называется paging, а раздел подкачки называется swap-разделом. Понятия swapping и paging отличаются. Понятие swapping обозначает полное перемещение образа процесса из ОЗУ в раздел подкачки. Получить информацию об использовании раздела подкачки можно с помощью команды swapon -s.

Файл /etc/fstab содержит список файловых систем, монтируемых автоматически при загрузке, или монтируемых по требованию пользователя. Поля fstab:

- имя;
- файла монтируемого устройства;
- точка;
- монтирования;
- тип;
- файловой системы;
- опции;
- монтирования;
- <dump>: надо ли для этой ФС производить автоматическое резервное копирование командой dump: 1 - разрешено, 0 - запрещено.

Таблица 11. Опции файла /etc/fstab

Опция	**Назначение**
defaults	rw, suid, dev, exec, auto, nouser, asynch
asynch	асинхронный режим ввода - вывода
auto/noauto	монтировать (не монтировать) вовремя загрузки
exec/noexec	разрешение /запрет на выполнение файлов с машинным кодом

suid/nosuid	можно /нельзя устанавливать бит SUID
user/nouser	можно /нельзя монтировать обычному пользователю
ro	только для чтения
rw	разрешено и чтение , и запись

Залог сохранности данных — это регулярное и неукоснительное выполнение резервного копирования данных. При планировании резервного копирования надо определить следующее:

- какие данные должны быть скопированы и где они находятся (в каких каталогах или файловых системах а также, возможно, на каких компьютерах);
- какой тип носителей данных будет использован для резервного копирования;
- с какой периодичностью будет производиться копирование данных;
- как будет осуществляться ротация резервных носителей данных;
- кто уполномочен производить резервное копирование;
- можно ли выполнять резервное копирование полностью автоматически;
- в какие моменты времени должно осуществляться резервное копирование;
- где и как будет производиться хранение носителей резервных копий;
- каков будет порядок восстановления утраченных данных из резервных копий;
- каково допустимое время, необходимое для восстановления данных.

В процессе принятия решения об организации резервного копирования надо учесть частоту изменения данных в файловой системе.

В большинстве систем содержимое следующих каталогов меняется относительно редко после установки системы:

- /boot — файлы, необходимые для загрузки ядра Linux;
- /etc — конфигурационные файлы, база данных паролей и сценарии инициализации системы и запуска демонов;
- /lib — основные жизненно важные библиотеки;
- /opt — дополнительное программное обеспечение;
- /bin и /sbin — исполняемые файлы, жизненно важные для системы;
- /usr — вторичная иерархия файловой системы, содержащая статические файлы.

Рекомендуется хранить редко изменяющиеся файлы в каталоге /usr, а часто изменяемые — в каталогах /var и /home.

Команда dd осуществляет низкоуровневое поблочное копирование из файла, указанного в командной строке после префикса if=, в файл, указанный после of=. По умолчанию используются блоки размером 512 байтов (один сектор). Обычно команда dd не применяется для резервного копирования файловых систем, однако она с успехом может быть использована для создания образов файловых систем и для восстановления файловых систем из образов. В UNIX- и GNU/Linux-системах команды архивирования отделены от утилит сжатия файлов. Все утилиты сжатия осуществляют компрессию файлов, указанных в качестве аргументов. При этом к исходным названиям файлов добавляются стандартные расширения, перечисленные далее, а права доступа и владения сохраняются. Для сжатия файлов достаточно указать их в качестве аргументов. В Linux используются следующие утилиты сжатия:

- gzip — применяется наиболее часто, сжатые файлы имеют расширения .gz;

- bzip2 — часто обеспечивает лучшую степень сжатия, чем gzip, сжатые файлы имеют расширения .bz2;
- compress — стандартная UNIX-утилита сжатия, в GNU/Linux используется реже, чем предыдущие, сжатые файлы имеют расширение .Z.

После сжатия к их названиям был добавлен суффикс .gz. Далее приведены часто используемые опции команды gzip:

- -d — декомпрессия сжатых файлов, как gunzip;
- -c — вывести сжатое содержимое файлов в поток вывода, без изменения файлов;
- -r — сжать содержимое каталога рекурсивно;
- -S — установить иной, чем .gz, суффикс;
- -t — тестировать содержимое архива;
- -v — подробный вывод информации о работе команды;
- -l — вывести информацию об уровне сжатия файлов.

Декомпрессию сжатых gzip-файлов выполняет команда gunzip.

Команда zcat распаковывает и выводит в поток содержимое сжатых gzip-файлов. Что касается команды bzip2, то в ней реализован иной алгоритм сжатия, часто обеспечивающий более высокий уровень компрессии. Многие опции команды bzip2 функционально идентичны соответствующим опциям gzip. Один из наиболее часто используемых инструментов резервного копирования — команда tar (tape archive). Аргументы команды tar — это файлы и каталоги, которые должны быть помещены в архив. Имя архива указывают после опции f команды. Часто используемые опции команды tar приведены в списке:

- -A — добавление файлов tar-архива в существующий архив (слияние);
- -c — сравнение содержимого архива с заданным каталогом;
- --delete — удаление файлов из архива;
- -r — добавление файлов в конец архива;

- -u — обновление архива версиями файлов, более новыми, чем в архиве;
- -b — указывает размер блока (n×512 байтов);
- -C — изменение каталога;
- -h — разыменовывать символические ссылки, т. е. сохранять в архиве не файлы символических ссылок, а файлы, на которые они указывают;
- -l — при создании архива оставаться в пределах текущей файловой системы и не переходить в смонтированные к подкаталогам файловые системы;
- -L — указать длину ленты (n×512 байтов);
- -m — не восстанавливать дату модификации файлов при извлечении их из архива;
- -p — сохранять при восстановлении файлов оригинальные права владения и права доступа к ним;
- -M — указывает, что архив состоит из нескольких томов;
- -P — сохранять в архиве файлы с абсолютными именами;
- -N — помещать в архив только те файлы, которые были созданы или изменены после специфицированной даты;
- -O — разархивировать файлы в стандартный поток вывода;
- -T — взять имена файлов для извлечения из архива или помещения в архив из заданного после опции файла;
- -Z — использовать утилиту сжатия compress.

Таблица 12. Основные команды для работы с файловой системы

Описание	**Команда**
получить информацию об индексном дескрипторе файла	stat <filename>
посмотреть права доступа к файлу	ls –l <filename>

посмотреть права на каталог	ls –ld <каталог>
изменить владельца/группу пользователей файла/каталога	chown
изменить группу пользователей файла/каталога	chgrp
установить права доступа к файлу/каталогу	chmod
вывести/задать маску	umask
показать подробную информацию и информацию о номере inode	ls -li
создать жесткую связь	ln <filename> <newname>
создавать жесткие связи вместо копирования	cp -l
создать символическую ссылку	ln –s <filename> <linkname>
вместо копирования создавать символические ссылки	cp -s
получить сведения о геометрии диска	/sbin/fdisk -l
редактировать таблицу разделов	fdisk <имя устройства>
монтирование	mount <опции> <устройство> <точка монтирования>
демонтирование	umount <устройство> или <точка монтирования>
вывести информацию об использовании подкачки	/sbin/swapon s
включить подкачку	swapon <файл> или <раздел>
выключить подкачку	swapoff <файл> или <раздел>
получить информацию о свободном пространстве	df <файл устройства>
получить информацию об используемом пространстве в каталоге	du <каталог>
вывести текущие параметры жесткого диска	hdparm <имя файла устройства>
установить параметры	hdparm <опции> <имя файла устройства>

жесткого диска	
сжать файлы	gzip <файлы>
сжать файлы (сжимает лучше, чем gzip)	bzip2 <файлы>
проверить целостность ФС	fsck <опции> <раздел>
копировать по блокам	dd if=<входящий файл> of=<исходящий файл>

Практическое задание по теме

1. Смонтируйте и отмонтируйте флешку из командной строки.

2. Получите информацию об использовании дискового пространства домашним каталогом.

3. Выведите список подкаталогов /usr/local и для каждого — занимаемый каталогом размер.

4. Скопируйте первые 512 байт из файла устройства жесткого диска, с которого загружается операционная система, в файл first_sect.img и определите тип его содержимого.

5. Создайте пустой файл и сожмите его gzip. Уменьшился ли размер этого файла?

6. От имени суперпользователя создайте полный сжатый архив etc.tar.gz всех файлов в каталоге /etc.

7. Создайте большой текстовый файл. Сожмите его алгоритмами gzip и bzip2. Сравните размер полученных файлов. Создайте файлы, копируя данные командой dd из устроств /dev/zero и /dev/random, сравните размер сжатых файлов.

8. Создайте небольшой каталог testarch с файлами в домашнем каталоге. Выполните резервное копирование этого каталога командой rsync. Создайте в каталоге testarch новый файл newfile. Создайте инкрементальную копию. Файл newfile случайно удален (удалите его). Восстановите файл newfile из бэкапа.

9. Изучить скорость работы дисковой подсистемы, копируя данные

командой dd. Построить графики зависимости скорости копирования от размера блока (bs) и от количества блоков при фиксированном размере блока.

10. Определите номер inode родительского каталога корневого каталога. Сравните номер inode вашего домашнего каталога ~ и имени «точка» . в нем самом.

11. Определите количество имен у файла /bin/gzip. Получите подробную информацию об индексном дескрипторе этого файла. Найдите все жесткие связи файла /bin/gzip.

12. Какая группа пользователей установлена для вашего каталога? Какая группа пользователей установлена на файл /bin/ls и кто является его владельцем? Проверьте полученную информацию при помощи команды stat.

13. Получите список файлов в домашнем каталоге в подробном формате. Какие права доступа установлены для них?

14. Переведите из восьмеричной формы записи прав доступа в символьную 641. Переведите запись прав доступа rw-r----- в восьмеричную форму.

15. Имеется файл script.sh с правами 750, владелец root, группа sys. Может ли пользователь user2, являющийся членом групп user2 и tty, чтолибо изменить в тексте файла? Может ли пользователь user2 запустить на исполнение этот файл? Какие права должны быть добавлены на файл, чтобы user2 мог читать и исполнять этот файл?

16. Определите, какие права установлены на ваш домашний каталог.

Два пользователя

17. Создайте пользователей user1 и user2, принадлежащих одной и той же группе users. В домашнем каталоге пользователя user1 создайте каталог d1 с правами drwx—x user1 users. Может ли пользователь user2

переименовать файл f1 в этом каталоге? Может ли пользователь user2 узнать, какие права установлены на файл f1? Установите на файл f1 права 640 user1 users. Может ли user2 просмотреть содержимое файла? Может ли он запустить на исполнение этот файл?

18. Установите на каталог d1 права 751. Может ли теперь пользователь user2 переименовать файл f1? На что должны быть установлены права, чтобы user2 смог удалить файл f1 из каталога d1? Какие эти права?

Пользователь root

19. В вашем домашнем каталоге создайте файл filetodel с правами 600 root root. Можете ли вы удалить этот файл?

20. Находясь в сеансе обычного пользователя, попытайтесь изменить права владения на любой файл в вашем домашнем каталоге. Что происходит?

21. Перейдите в сеанс супер-пользователя и, находясь в домашнем каталоге обычного пользователя, измените права владения на любой файл так, чтобы он принадлежал пользователю user2. Измените группу того же файла на sys. С помощью одной команды измените владельца и группу пользователей этого файла на исходные. Покиньте сеанс супер-пользователя.

22. Создайте цепочку каталогов d1/d2/d3, а в них – файлы d1/f1, d1/d2/f2, d1/d3/d3/f3. Установите на файл d1/f1 права 400. Можете ли вы изменить этот файл? Рекурсивно добавьте права на чтение и запись для каталога d1. Можете ли вы теперь изменить файл f1? Отнимите у каталога d1 права на чтение и запись. Можно ли теперь получить информацию о файлах в этом каталоге? С помощью команды find при условии установки exec измените права на каталоги и все подкаталоги d1 на 750.

23. Командами find, xargs и chmod установите права на все обычные

файлы в каталоге d1 равными rwr—r.

24. Установите значение umask 000. Проверьте, с какими правами создаются новые каталоги и файлы.

25. В символьной форме установите такое значение umask, чтобы вновь создаваемые файлы имели права доступа 644.

26. Требуется, чтобы владелец создаваемого каталога мог создавать, читать и записывать файлы в каталог, а также переходить в него. Члены группы владельца должны иметь права на создание и удаление файлов в этом каталоге. Все остальные никаких прав иметь не должны. Какое должно быть значение маски для того, чтобы удовлетворить перечисленным требованиям?

27. Где в системе хранятся разделяемые библиотеки (*.so)?

28. Где в системе хранятся заголовочные файлы (*.h)?

29. Найдите в системе все файлы именованных каналов и сокетов. Какой размер имеют файлы именованных каналов и сокетов?

30. Создайте в текущем каталоге файл f1, содержащий любой текст и жесткую связь с файлом link1.

31. Можете ли вы перенести этот файл в каталог /tmp и почему? Можно ли просмотреть содержимое файла по имени link1?

32. Перенесите файл f1 из каталога /tmp обратно. Создайте символическую ссылку на него – slink2. Распечатайте содержимое файла f1 при помощи ссылки slink2.

33. Перенесите файл f1 в /tmp. Что произошло со ссылкой slink2? Можно ли теперь с её помощью распечатать содержимое файла f1?

34. Попробуйте создать жесткую связь с домашним каталогом командой ln. Удается ли это?

35. Как можно создать жесткую связь с именем домашнего каталога? Какое может быть имя у этой связи, и с помощью какой команды она создается?

36. Создайте символическую ссылку на каталог /usr/share/doc в вашем домашнем каталоге. Попробуйте выполнить команду cd ~/doc.

37. Получите список файлов – символических ссылок, находящихся в каталоге /usr.

38. Создайте в домашнем каталоге символическую ссылку на исполняемый файл /bin/ls. Попробуйте воспользоваться новой ссылкой.

39. С помощью команды fdisk l изучите таблицу разделов вашего жесткого диска.

Тема 6. Сетевое администрирование. Netfilter/iptables

Цель: Изучение работы с файерволом. изучение особенностей фильтрации входного трафика и построение правил по критериям «отправитель», «протокол», «порт назначения», «входной интерфейс», журналирования трафика.

Netfilter — межсетевой экран (МСЭ, брандмауэр, файерволл), встроен в ядро Linux с версии 2.4. Netfilter управляется утилитой **iptables** (Для IPv6 — ip6tables). До **netfilter/iptables**был **Ipchains**, который входил в состав ядер Linux 2.2. До ipchains в Linux был так называемый **ipfw (IPV4 firewal)**, перенесенный из BSD. Утилита управления - ipfwadm. Проект netfilter/iptables был основан в 1998. Автором является Расти Расселл (он же руководил и прошлыми разработками). В 1999 г. образовалась команда Netfilter Core Team (сокращено coreteam). Разработанный межсетевой экран получил официальное название netfilter. В августе 2003 руководителем coreteam стал Харальд Вельте (Harald Welte).

Проекты *ipchains* и *ipfwadm* изменяли работу стека протоколов ядра Linux, поскольку до появления**netfilter** в архитектуре ядра не существовало возможностей для подключения дополнительных модулей управления пакетами. **iptables** сохранил основную идею *ipfwadm* — список правил, состоящих из критериев и действия, которое выполняется если пакет соответствует критериям. В *ipchains* была представлена новая концепция — возможность создавать новые цепочки правил и переход пакетов между цепочками, а в iptables концепция была расширена до четырёх таблиц (в современных netfilter - более четырех), разграничивающих цепочки правил по задачам: *фильтрация, NAT, и модификация пакетов*. Также iptables расширил возможности Linux в области определения состояний, позволяя создавать межсетевые экраны работающие на сеансовом уровне.

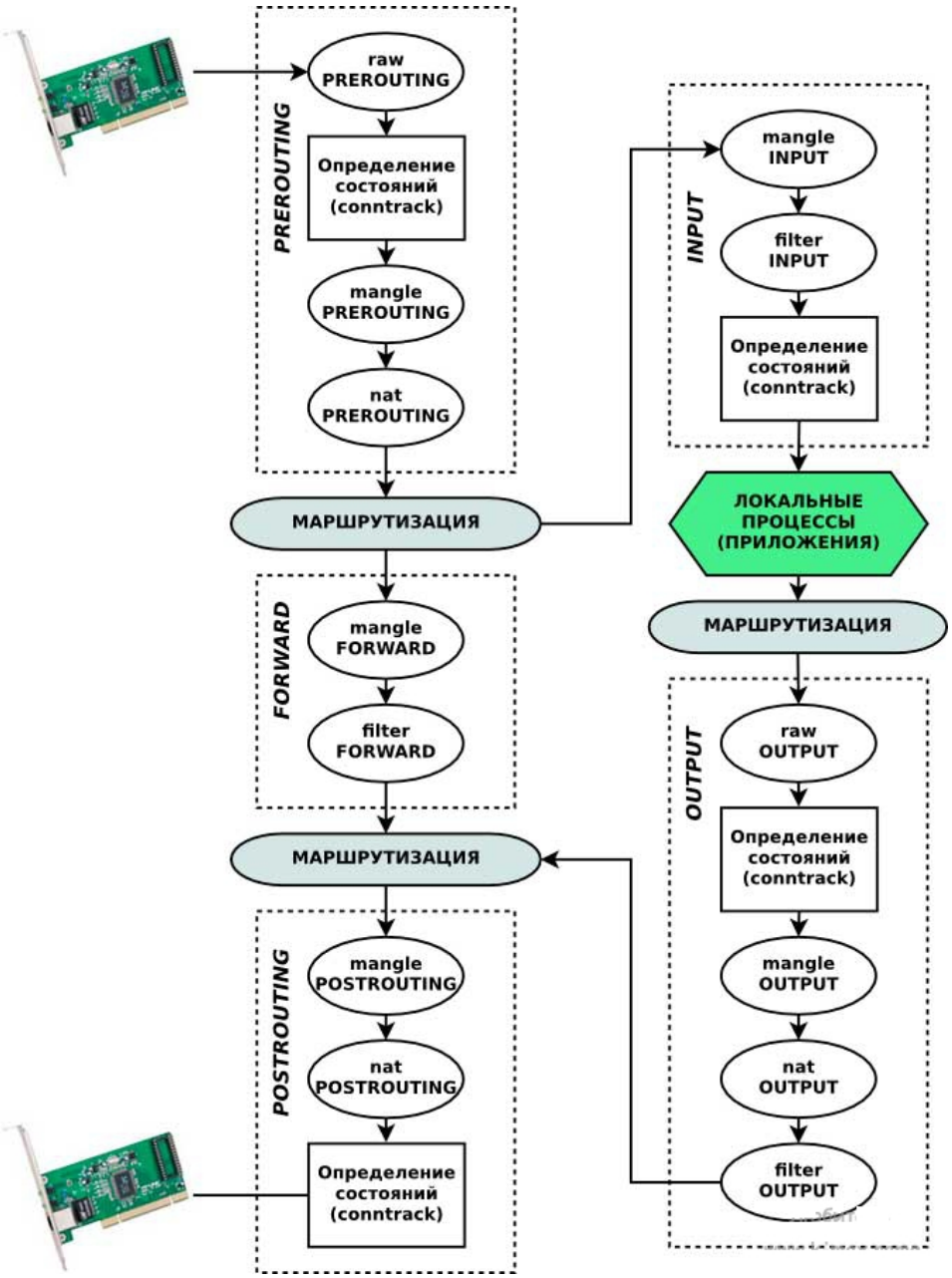

Рисунок 9 - Схема работы netfilter

Сетевые пакеты поступают в сетевой интерфейс, настроенный на стек TCP/IP и после некоторых простых проверок ядром (например, контрольная сумма) проходят последовательность **цепочек (chain)** (обозначены пунктиром). Пакет обязательно проходит первоначальную цепочку **PREROUTING**. После цепочки **PREROUTING**, в соответствии с таблицей маршрутизации, проверяется кому принадлежит пакет и, в зависимости от назначения пакета, определяется куда он дальше

попадет (в какую цепочку). Если пакет НЕ адресован (в TCP пакете поле адрес получателя - НЕ локальная система) локальной системе, то он направляется в цепочку **FORWARD**, если пакет адресован локальной системе, то направляется в цепочку **INPUT** и после прохождения **INPUT**отдается локальным демонам/процессам. После обработки локальной программой, при необходимости формируется ответ. Ответный пакет пакет отправляемый локальной системой в соответствии с правилами маршрутизации направляется на соответствующий маршрут (хост из локальной сети или адрес маршрутизатора) и направляется в цепочку **OUTPUT**. После цепочки **OUTPUT** (или **FORWARD**, если пакет был проходящий) пакет снова сверяется с правилами маршрутизации и отправляется в цепочку **POSTROUTING**.

Каждая цепочка, которую проходит пакет состоит из набора **таблиц (table)** (обозначены овалами).Таблицы в разных цепочках имеют одинаковое наименование, но тем не менее никак между собой **не связаны**. Например **таблица nat** в цепочке **PREROUTING** никак не связана с **таблицей nat** в цепочке **POSTROUTING**. Каждая таблица состоит из упорядоченного набора (списка) **правил**. Каждое правило содержи **условие**, которому должен соответствовать проходящий пакет и **действия к пакету**, подходящему данному условию.

Проходя через серию **цепочек** пакет последовательно проходит **каждую таблицу** (в указанном на иллюстрации порядке) и в **каждой таблице** последовательно сверяется с **каждым правилом** (точнее сказать - с каждым набором условий/критериев в правиле), и если пакет соответствует какому-либо **критерию**, то выполняется **заданное действие** над пакетом. При этом, в **каждой таблице** (кроме пользовательских) существует заданная **по-умолчанию политика**. Данная политика определяет действие над

пакетом, в случае, если пакет не соответствует ни одному из правил в таблице. Чаще всего - это действие **ACCEPT**, чтобы принять пакет и передать в следующую таблицу или **DROP** - чтобы отбросить пакет. В случае, если пакет не был отброшен, он завершает своё перемещение по ядру системы и отправляется в сетевую карту сетевой интерфейс, которая подходит по правилам маршрутизации. Цепочки netfilter:

PREROUTING — для изначальной обработки **входящих** пакетов;

INPUT — для входящих пакетов, адресованных непосредственно **локальному компьютеру;**

FORWARD — для **проходящих** (маршрутизируемых) пакетов;

OUTPUT — для пакетов, **создаваемых** локальным компьютером (исходящих);

POSTROUTING— для окончательной обработки **исходящих** пакетов.

Также можно создавать и уничтожать собственные цепочки при помощи утилиты iptables.

Цепочки организованны в 4 **таблицы**:

raw — пакет проходит данную таблицу до передачи системе определения состояний. Используется редко, например для маркировки пакетов, которые НЕ должны обрабатываться системой определения состояний. Для этого в правиле указывается действие *NOTRACK*. Содержится в цепочках*PREROUTING* и *OUTPUT*.

mangle — содержит правила модификации (обычно полей заголовка) IP□пакетов. Среди прочего, поддерживает действия *TTL*, *TOS*, и *MARK* (для изменения полей TTL и TOS, и для изменения маркеров пакета). Редко необходима и может быть опасна. Содержится во всех пяти стандартных цепочках.

nat — предназначена для подмены адреса отправителя или получателя. Данную таблицу проходят только первый пакет из потока,

трансляция адресов или маскировка (подмена адреса отправителя или получателя) применяются ко всем последующим пакетам в потоке **автоматически**. Поддерживает действия *DNAT*, *SNAT*, *MASQUERADE*, *REDIRECT*. Содержится в цепочках *PREROUTING*, *OUTPUT*, и *POSTROUTING*.

filter — основная таблица, используется по умолчанию если название таблицы не указано. Используется для фильтрации пакетов. Содержится в цепочках *INPUT*, *FORWARD*, и *OUTPUT*.

Как уже было отмечено, непосредственно для фильтрации пакетов используются таблицы filter. Поэтому в рамках данной темы важно понимать, что для пакетов, предназначенных данному узлу необходимо модифицировать таблицу filter цепочки INPUT, для проходящих пакетов — цепочки FORWARD, для пакетов, созданных данным узлом — OUTPUT.

Таким образом, когда пакет приходит на файервол, то он сначала попадает на сетевое устройство, перехватывается соответствующим драйвером и далее передаётся в ядро. Далее пакет проходит ряд таблиц и затем передаётся либо локальному приложению, либо переправляется на другую машину. Порядок следования пакета приводится ниже.

Таблица 13 – Порядок движения транзитных пакетов

Шаг	Таблица	Цепочка	Примечание
1			Кабель (Интернет)
2			Сетевой интерфейс (например, eth0)
3	Mangle	PREROUTING	Обычно эта цепочка используется для внесения изменений в заголовок пакета, например для изменения битов TOS и пр..

4	Nat	PREROUTING	Эта цепочка используется для трансляции сетевых адресов (Destination Network Address Translation). Source Network Address Translation выполняется позднее, в другой цепочке. Любого рода фильтрация в этой цепочке может производиться только в исключительных случаях
5			Принятие решения о дальнейшей маршрутизации, то есть в этой точке ре приложению или на другой узел сети.
6	Filter	FORWARD	В цепочку FORWARD попадают только те пакеты, которые идут на другой хост Вся фильтрация транзитного трафика должна выполняться здесь. Не забывайте, что через эту цепочку проходит трафик в обоих направлениях, обязательно учитывайте это обстоятельство при написании правил фильтрации.
7	Mangle	FORWARD	Далее пакет попадает в цепочку FORWARD таблицы mangle, которая должна использоваться только в исключительных случаях, когда необходимо внести некоторые изменения в заголовок пакета между двумя точками принятия решения о маршрутизации.

8			Принятие решения о дальнейшей маршрутизации, то есть в этой точке, к примеру, решается на какой интерфейс пойдет пакет.
9	Nat	POSTROUTING	Эта цепочка предназначена в первую очередь для Source Network Address Translation. Не используйте ее для фильтрации без особой на то необходимости. Здесь же выполняется и маскировка (Masquerading).
10	Mangle	POSTROUTING	Эта цепочка предназначена для внесения изменений в заголовок пакета уже после того как принято последнее решение о маршрутизации.
11			Выходной сетевой интерфейс (например, eth1).
12			Кабель (например, LAN).

Из данной таблицы видно, что пакет проходит несколько этапов, прежде чем он будет передан далее. На каждом из них пакет может быть остановлен, будь то цепочка iptables или что либо еще, но наибольший интерес представляет iptables. Следует заметить, что нет каких либо цепочек, специфичных для отдельных интерфейсов или чего либо подобного. Цепочку FORWARD проходят ВСЕ пакеты, которые движутся через данный МСЭ/маршрутизатор. Не следует использовать цепочку INPUT для фильтрации транзитных пакетов, так как они туда просто не попадают. Через эту цепочку движутся только те пакеты, которые предназначены данной машине.

А теперь рассмотрим порядок движения пакета, предназначенного локальному процессу/приложению.

Таблица 14 – Для локального приложения

Шаг	Таблица	Цепочка	Примечание
1			Кабель (Интернет)
2			Входной сетевой интерфейс (например, eth0)
3	Mangle	PREROUTING	Обычно используется для внесения изменений в заголовок пакета, например для установки битов TOS и пр.
4	Nat	PREROUTING	Преобразование адресов (DestinationNetwork Address Translation). Фильтрация пакетов здесь допускается только в исключительных случаях.
5			Принятие решения о маршрутизации.
6	Mangle	INPUT	Пакет попадает в цепочку INPUT таблицы mangle. Здесь вносятся изменения в заголовок пакета перед тем как он будет передан локальному приложению.
7	Filter	INPUT	Здесь производится фильтрация входящего трафика. Помните, что все входящие пакеты, адресованные нам, проходят через эту цепочку, независимо от того с какого интерфейса они поступили.
8			Локальный процесс/приложение

Важно помнить, что на этот раз пакеты идут через цепочку INPUT, а не через FORWARD. И в заключение рассмотрим порядок движения пакетов, созданных локальными процессами.

Таблица 15 – От локальных процессов

Шаг	Таблица	Цепочка	Примечание
1			Локальный процесс
2	Mangle	OUTPUT	Здесь производится внесение изменений в заголовок пакета. Фильтрация, выполняемая в этой цепочке, может иметь негативные последствия.
3	Nat	OUTPUT	Эта цепочка используется для трансляции сетевых адресов (NAT) в пакетах, исходящих от локальных процессов файервола.
4	Filter	OUTPUT	Здесь фильтруется исходящий трафик.
5			Принятие решения о маршрутизации. Здесь решается, куда пойдет пакет дальше.
6	Nat	POSTROUTING	Здесь выполняется Source Network Address Translation. Не следует в этой цепочке производить фильтрацию пакетов во избежание нежелательных побочных эффектов. Однако и здесь можно останавливать пакеты, применяя политику по умолчанию DROP.

7	Mangle	POSTROUTING	Цепочка POSTROUTING таблицы mangle в основном используется для правил, которые должны вносить изменения в заголовок пакета перед тем, как он покинет МСЭ, но уже после принятия решения о маршрутизации. В эту цепочку попадают все пакеты, как транзитные, так и созданные локальными процессами файервола.
8			Сетевой интерфейс (например, eth0)
9			Кабель (Internet)

Таблица Mangle

Как уже упоминалось выше, эта таблица предназначена, главным образом для внесения изменений в заголовки пакетов. В этой таблице можно устанавливать биты TOS (Type Of Service) а так же другие биты.

Следует помнить, что в этой таблице не следует производить любого рода фильтрацию, маскировку или преобразование адресов (DNAT, SNAT, MASQUERADE).В этой таблице допускается выполнять только следующие действия:

- TOS
- TTL
- MARK

Действие TOS выполняет установку битов поля Type of Service в пакете. Это поле используется для назначения сетевой политики обслуживания пакета, то есть задает желаемый вариант

маршрутизации. Однако, следует заметить, что данное свойство в действительности используется на незначительном количестве маршрутизаторов в Интернете. Другими словами, не следует изменять состояние этого поля для пакетов, уходящих в Интернет, потому что на маршрутизаторах, которые таки обслуживают это поле, может быть принято неправильное решение при выборе маршрута.

Действие TTL используется для установки значения поля TTL (Time To Live) пакета. Можно присваивать определенное значение этому полю, чтобы скрыть МСЭ от чересчур любопытных провайдеров (Internet Service Providers). Дело в том, что отдельные провайдеры очень не любят когда одно подключение разделяется несколькими компьютерами, и тогда они начинают проверять значение TTL приходящих пакетов и используют его как один из критериев определения того, один компьютер «сидит» на подключении или несколько.

Действие MARK устанавливает специальную метку на пакет, которая затем может быть проверена другими правилами в iptables или другими программами, например iproute2. С помощью «меток» можно управлять маршрутизацией пакетов, ограничивать трафик и т. п.

Таблица Nat

Эта таблица используется для выполнения преобразований сетевых адресов NAT (Network Address Translation) Как уже упоминалось выше, только первый пакет из потока проходит через цепочки этой таблицы, трансляция адресов или маскировка применяются ко всем последующим пакетам в потоке автоматически. Для этой таблицы характерны действия:

- DNAT
- SNAT

• MASQUERADE

Действие DNAT (Destination Network Address Translation) производит преобразование адресов назначения в заголовках пакетов. Другими словами, этим действием производится перенаправление пакетов на другие адреса, отличные от указанных в заголовках пакетов.

SNAT (Source Network Address Translation) используется для изменения исходных адресов пакетов. С помощью этого действия можно скрыть структуру локальной сети, а заодно и разделить единственный внешний IP адрес между компьютерами локальной сети для выхода в Интернет. В этом случае МСЭ, с помощью SNAT, автоматически производит прямое и обратное преобразование адресов, тем самым давая возможность выполнять подключение к серверам в Интернете с компьютеров в локальной сети.

Маскировка (MASQUERADE) применяется в тех же целях, что и SNAT, но в отличие от последней, MASQUERADE дает более сильную нагрузку на систему. Происходит это потому, что каждый раз, когда требуется выполнение этого действия - производится запрос IP адреса для указанного в действии сетевого интерфейса, в то время как для SNAT IP адрес указывается непосредственно. Однако, благодаря такому отличию, MASQUERADE может работать в случаях с динамическим IP адресом, когда подключение к сети осуществляется через PPP или SLIP.

Таблица Filter

Как следует из названия, в этой таблице должны содержаться наборы правил для выполнения фильтрации пакетов. Пакеты могут пропускаться далее, либо отвергаться, в зависимости от их содержимого. В этой таблице допускается использование большинства из существующих действий, однако ряд действий, которые были

рассмотрены выше в этой главе, должны выполняться только в присущих им таблицах.

Построение правил

Как уже говорилось выше, каждое правило -- это строка, содержащая в себе критерии определяющие, подпадает ли пакет под заданное правило, и действие, которое необходимо выполнить в случае выполнения критерия. В общем виде правила записываются примерно так:

iptables [-t table] command [match] [target/jump]

Нигде не утверждается, что описание действия (target/jump) должно стоять последним в строке, однако, будем придерживаться именно такой нотации для удобства.

Если в правило не включается спецификатор [-t table], то по умолчанию предполагается использование таблицы filter, если же предполагается использование другой таблицы, то это требуется указать явно. Спецификатор таблицы так же можно указывать в любом месте строки правила, однако более или менее стандартом считается указание таблицы в начале правила.

Далее, непосредственно за именем таблицы, должна стоять команда. Если спецификатора таблицы нет, то команда всегда должна стоять первой. Команда определяет действие iptables, например: вставить правило, или добавить правило в конец цепочки, или удалить правило и т.п.

Раздел match задает критерии проверки, по которым определяется, подпадает ли пакет под действие этого правила или нет. Здесь можно указать самые разные критерии -- и IP-адрес источника пакета или сети, и сетевой интерфейс и т.д. Существует множество критериев, которые будут рассмотрены далее.

И, наконец, target указывает, какое действие должно быть

выполнено при условии выполнения критериев в правиле. Здесь можно заставить ядро передать пакет в другую цепочку правил, «сбросить» пакет и забыть про него, выдать на источник сообщение об ошибке и т.п.

Опция -t указывает на используемую таблицу. По умолчанию используется таблица filter. С ключом -t применяются следующие опции.

Таблица 16 – Таблицы

Таблица	Описание
nat	Таблица nat используется главным образом для преобразования сетевых адресов (Network Address Translation). Через эту таблицу проходит только первый пакет из потока. Преобразования адресов автоматически применяется ко всем последующим пакетам. Это один из факторов, исходя из которых мы не должны осуществлять какую- либо фильтрацию в этой таблице. Цепочка PREROUTING используется для внесения изменений в пакеты на входе в МСЭ. Цепочка OUTPUT используется для преобразования пакетов, созданных приложениями внутри МСЭа, перед принятием решения о маршрутизации. И последняя цепочка в этой таблице -- POSTROUTING, которая используется для преобразования пакетов перед выдачей их во вне.

mangle	Эта таблица используется для внесения изменений в заголовки пакетов. Примером может служить изменение поля TTL, TOS или MARK. Важно: в действительности поле MARK не изменяется, но в памяти ядра заводится структура, которая сопровождает данный пакет все время его прохождения через машину, так что другие правила и приложения на данной машине (и только на данной машине) могут использовать это поле в своих целях. Таблица имеет пять цепочек PREROUTING, POSTROUTING, INPUT, OUTPUT и FORWARD. PREROUTING используется для внесения изменений на входе в МСЭ, перед первым принятием решения о маршрутизации. POSTROUTING используется для внесения изменений на выходе из МСЭа, после последнего принятия решения о маршрутизации. INPUT -- для внесения изменений в пакеты перед тем как они будут переданы локальному приложению внутри МСЭа. OUTPUT -- для внесения изменений в пакеты, поступающие от приложений внутри МСЭа. FORWARD -- для внесения изменений в транзитные пакеты после первого принятия решения о маршрутизации, но перед последним принятием решения о маршрутизации. Следует заметить, что таблица mangle ни в коем случае не должна использоваться для преобразования сетевых адресов или маскарадинга (Network Address Translation, Masquerading), поскольку для этих целей имеется таблица nat.

filter	Таблица filter используется главным образом для фильтрации пакетов. Для примера, здесь мы можем выполнить DROP, LOG, ACCEPT или REJECT без каких либо сложностей, как в других таблицах. Имеется три встроенных цепочки. Первая -- FORWARD, используемая для фильтрации пакетов, идущих транзитом через МСЭ. Цепочку INPUT проходят пакеты, которые предназначены локальным приложениям (МСЭу). И цепочка OUTPUT -- используется для фильтрации исходящих пакетов, сгенерированных приложениями на самом МСЭе.

Выше были рассмотрены основные отличия трех имеющихся таблиц. Каждая из них должна использоваться только в своих целях, и это следует понимать. Не целевое использование таблиц может привести к ослаблению защиты МСЭ и сети, находящейся за ним.

Ниже приводится список команд и правила их использования. Посредством команд пользователь сообщает iptables что он предполагает сделать. Обычно предполагается одно из двух действий это добавление нового правила в цепочку или удаление существующего правила из той или иной таблицы. Далее приведены команды, которые используются в iptables.

Таблица 17 – Команды

Команда	-A, --append
Пример	iptables -A INPUT ...
Пояснения	Добавляет новое правило в конец заданной цепочки.
Команда	-D, --delete
Пример	iptables -D INPUT --dport 80 -j DROP, iptables -D INPUT 1

Пояснения	Удаление правила из цепочки. Команда имеет два формата записи, первый -- когда задается критерий сравнения с опцией -D (см. первый пример), второй -- порядковый номер правила. Если задается критерий сравнения, то удаляется правило, которое имеет в себе этот критерий, если задается номер правила, то будет удалено правило с заданным номером. Счет правил в цепочках начинается с 1.
Команда	-R, --replace
Пример	iptables -R INPUT 1 -s 192.168.0.1 -j DROP
Пояснения	Данная команда заменяет одно правило другим. В основном она используется во время отладки новых правил.
Команда	-I, --insert
Пример	iptables -I INPUT 1 --dport 80 -j ACCEPT
Пояснения	Вставляет новое правило в цепочку. Число, следующее за именем цепочки, указывает номер правила, перед которым нужно вставить новое правило, другими словами число задает номер для вставляемого правила. В примере выше, указывается, что данное правило должно быть 1-м в цепочке INPUT.
Команда	-L, --list
Пример	iptables -L INPUT
Пояснения	Вывод списка правил в заданной цепочке, в данном примере предполагается вывод правил из цепочки INPUT. Если имя цепочки не указывается, то выводится список правил для всех цепочек. Формат вывода зависит от наличия дополнительных ключей в команде, например -n, -v, и пр.
Команда	-F, --flush
Пример	iptables -F INPUT

Пояснения	Сброс (удаление) всех правил из заданной цепочки (таблицы). Если имя цепочки и таблицы не указывается, то удаляются все правила, во всех цепочках.
Команда	-Z, --zero
Пример	iptables -Z INPUT
Пояснения	Обнуление всех счетчиков в заданной цепочке. Если имя цепочки не указывается, то подразумеваются все цепочки. При использовании ключа -v совместно с командой -L, на вывод будут поданы и состояния счетчиков пакетов, попавших под действие каждого правила. Допускается совместное использование команд -L и -Z. В этом случае будет выдан сначала список правил со счетчиками, а затем произойдет обнуление счетчиков.
Команда	-N, --new-chain
Пример	iptables -N allowed
Пояснения	Создается новая цепочка с заданным именем в заданной таблице В выше приведенном примере создается новая цепочка с именем allowed. Имя цепочки должно быть уникальным и не должно совпадать с зарезервированными именами цепочек и действий (DROP, REJECT и т.п.)
Команда	-X, --delete-chain
Пример	iptables -X allowed
Пояснения	Удаление заданной цепочки из заданной таблицы. Удаляемая цепочка не должна иметь правил и не должно быть ссылок из других цепочек на удаляемую цепочку. Если имя цепочки не указано, то будут удалены все цепочки, определенные командой -N в заданной таблице.
Команда	-P, --policy
Пример	iptables -P INPUT DROP

Пояснения	Определяет политику по умолчанию для заданной цепочки. Политика по умолчанию определяет действие, применяемое к пакетам, не попавшим под действие ни одного из правил в цепочке. В качестве политики по умолчанию допускается использовать DROP, ACCEPT и REJECT.
Команда	-E, --rename-chain
Пример	iptables -E allowed disallowed
Пояснения	Команда -E выполняет переименование пользовательской цепочки. В примере цепочка allowed будет переименована в цепочку disallowed. Эти переименования не изменяют порядок работы, а носят только косметический характер.

Команда должна быть указана всегда. Список доступных команд можно просмотреть с помощью команды iptables -h или, что то же самое, iptables --help. Некоторые команды могут использоваться совместно с дополнительными ключами. Ниже приводится список дополнительных ключей и описывается результат их действия. Следует заметить, что здесь не приводится дополнительных ключей, которые используются при построении критериев (matches) или действий (targets). Эти опции рассмотрим далее.

Таблица 18 – Ключи

Ключ	-v, --verbose
Команды, с которыми	--list, --append, --insert, --delete, --replace

Описание	Данный ключ используется для повышения информативности вывода и, как правило, используется совместно с командой --list. В случае использования с командой --list, в вывод этой команды включаются так же имя интерфейса, счетчики пакетов и байт для каждого правила. Формат вывода счетчиков предполагает вывод кроме цифр числа еще и символьные множители K (x1000), M (x1,000,000) и G (x1,000,000,000). Для того, чтобы заставить команду --list выводить полное число (без употребления множителей) требуется применять ключ -x, который описан ниже. Если ключ -v, --verbose используется с командами --append, --insert, --delete или --replace, то то на вывод будет выдан подробный отчет о произведенной операции.
Ключ	-x, --exact
Команды, с которыми	--list
Описание	Для всех чисел в выходных данных выводятся их точные значения без округления и без применения множителей K, M, G. Важно то, что данный ключ используется только с командой --list и не применяется с другими командами.
Ключ	-n, --numeric
Команды, с которыми	--list

Описание	Заставляет iptables выводить IP-адреса и номера портов в числовом виде предотвращая попытки преобразовать их в символические имена. Данный ключ используется только с командой --list.
Ключ	--line-numbers
Команды, с которыми	--list
Описание	Ключ --line-numbers включает режим вывода номеров строк при отображении списка правил командой --list. Номер строки соответствует позиции правила в цепочке. Этот ключ используется только с командой --list.
Ключ	-c, --set-counters
Команды, с которыми	--insert, --append, --replace
Описание	Этот ключ используется при создании нового правила для установки счетчиков пакетов и байт в заданное значение. Например, ключ --set-counters 20 4000установит счетчик пакетов = 20, а счетчик байт = 4000.

Критерии

Здесь будут подробно рассмотрены критерии выделения пакетов. Все критерии разбиты на пять групп. Первая – общие критерии, которые могут использоваться в любых правилах. Вторая - TCP критерии, которые применяются только к TCP пакетам. Третья – UDP критерии, которые применяются только к UDP пакетам. Четвертая - ICMP критерии для работы с ICMP пакетами. И, наконец, пятая – специальные критерии, такие

как state, owner, limit и пр.

Таблица 19 – Общие критерии

Критерий	-p, --protocol
Пример	iptables -A INPUT -p tcp
Описание	Этот критерий используется для указания типа протокола. Примерами протоколов могут быть TCP, UDP и ICMP. Список протоколов можно посмотреть в файле /etc/p rotocols. Прежде всего, в качестве имени протокола в данный критерий можно передавать три вышеупомянутых протокола, а также ключевое слово ALL. В качестве протокола допускается передавать число - номер протокола, так например, 255 соответствует протоколу RAW IP. Соответствия между номерами протоколов и их именами можно посмотреть в файле /etc/protocols, который уже упоминался выше. Если данному критерию передается числовое значение 0, то это эквивалентно использованию спецификатора ALL, который подразумевается по умолчанию, когда критерий --protocol не используется. Для логической инверсии критерия, перед именем протокола (списком протоколов) используется символ !, например --protocol ! tcp подразумевает пакеты любого протокола, кроме tcp.
Критерий	-s, --src, --source
Пример	iptables -A INPUT -s 192.168.1.1
Описание	IP-адрес(а) источника пакета. Адрес источника может указываться так, как показано в примере, тогда подразумевается единственный IP-адрес. А можно указать адрес в виде address/mask, например как

	192.168.0.0/255.255.255.0, или более современным способом 192.168.0.0/24, то есть фактически определяя диапазон адресов Как и ранее, символ !, установленный перед адресом, означает логическое отрицание, то есть --source ! 192.168.0.0/24 означает любой адрес кроме адресов 192.168.0.x
Критерий	-d, --dst, --destination
Пример	iptables -A INPUT -d 192.168.1.1
Описание	IP-адрес(а) получателя. Имеет синтаксис схожий с критерием --source, за исключением того, что подразумевает адрес места назначения. Точно так же может определять как единственный IP-адрес, так и диапазон адресов. Символ ! используется для логической инверсии критерия.
Критерий	-i, --in-interface
Пример	iptables -A INPUT –I eth0
Описание	Интерфейс, с которого был получен пакет. Использование этого критерия допускается только в цепочках INPUT, FORWARD и PREROUTING, в любых других случаях будет вызывать сообщение об ошибке. При отсутствии этого критерия предполагается любой интерфейс, что равносильно использованию критерия -i +. Как и прежде, символ ! инвертирует результат совпадения. Если имя интерфейса завершается символом +, то критерий задает все интерфейсы, начинающиеся с заданной строки, например -i PPP+ обозначает любой PPP интерфейс, а запись -i ! eth+ -- любой интерфейс, кроме любого eth.
Критерий	-o, --out-interface

Пример	iptables -A FORWARD -o eth0
Описание	Задает имя выходного интерфейса. Этот критерий допускается использовать только в цепочках OUTPUT, FORWARD и POSTROUTING, в противном случае будет генерироваться сообщение об ошибке. При отсутствии этого критерия предполагается любой интерфейс, что равносильно использованию критерия -o +. Как и прежде, символ ! инвертирует результат совпадения. Если имя интерфейса завершается символом +, то критерий задает все интерфейсы, начинающиеся с заданной строки, например -o eth+ обозначает любой eth интерфейс, а запись -o ! eth+ - любой интерфейс, кроме любого eth
Критерий	-f, --fragment
Пример	iptables -A INPUT -f
Описание	Правило распространяется на все фрагменты фрагментированного пакета, кроме первого, сделано это потому, что нет возможности определить исходящий/входящий порт для фрагмента пакета, а для ICMP-пакетов определить их тип. С помощью фрагментированных пакетов могут производиться атаки на ваш МСЭ, так как фрагменты пакетов могут не отлавливаться другими правилами. Как и раньше, допускается использования символа ! для инверсии результата сравнения. только в данном случае символ ! должен предшествовать критерию -f, например ! -f. Инверсия критерия трактуется как «все первые фрагменты фрагментированных пакетов и/или нефрагментированные пакеты, но не вторые и последующие фрагменты

	фрагментированных пакетов».

Неявные критерии

В этом разделе рассмотрим неявные критерии, точнее, те критерии, которые подгружаются неявно и становятся доступны, например, при указании критерия --protocol. На сегодняшний день существует три автоматически подгружаемых расширения, это TCP критерии, UDP критерии и ICMP критерии. Загрузка этих расширений может производиться и явным образом с помощью ключа -m, -match, например -m tcp.

TCP-критерии - это расширение зависит от типа протокола и работает только с TCP пакетами. Чтобы использовать эти дополнительные критерии, необходимо в правилах указывать тип протокола --protocol tcp. Важно: критерий --protocol tcp обязательно должен стоять перед специфичным критерием. Эти расширения загружаются автоматически как для tcp протокола, так и для udp и icmp протоколов.

Таблица 20 – TCP критерии

Критерий	--sport, --source-port
Пример	iptables -A INPUT -p tcp --sport 22
Описание	Исходный порт, с которого был отправлен пакет. В качестве параметра может указываться номер порта или название сетевой службы. Соответствие имен сервисов и номеров портов вы сможете найти в файле /etc/services При указании номеров портов правила отрабатывают несколько быстрее. однако это менее удобно при разборе листингов скриптов. Номера портов могут задаваться в виде интервала из минимального и максимального номеров, например --source-port 22:80. Если опускается минимальный порт, то есть когда

	критерий записывается как --source-port :80, то в качестве начала диапазона принимается число 0. Если опускается максимальный порт, то есть когда критерий записывается как --source-port 22:, то в качестве конца диапазона принимается число 65535. Допускается такая запись --source-port 80:22, в этом случае iptables поменяет числа 22 и 80 местами, то есть подобного рода запись будет преобразована в --source-port 22:80. Как и раньше, символ ! используется для инверсии. Так критерий --source-port ! 22 подразумевает любой порт, кроме 22. Инверсия может применяться и к диапазону портов, например --source-port ! 22:80.
Критерий	--dport, --destination-port
Пример	iptables -A INPUT -p tcp --dport 22
Описание	Порт, на который адресован пакет. Аргументы задаются в том же формате, что и для --source-port.
Критерий	--tcp-flags
Пример	iptables -p tcp --tcp-flags SYN,ACK,FIN SYN
Описание	Определяет маску и флаги tcp-пакета. Пакет считается удовлетворяющим критерию, если из перечисленных флагов в первом списке в единичное состояние установлены флаги из второго списка. Так для вышеуказанного примера под критерий подпадают пакеты, у которых флаг SYN установлен, а флаги FIN и ACK сброшены. В качестве аргументов критерия могут выступать флаги SYN, ACK, FIN, RST, URG, PSH, а так же зарезервированные идентификаторы ALL и NONE. ALL -- значит ВСЕ флаги и NONE - НИ ОДИН флаг. Так, критерий --tcp-flags ALL NONE

	означает, что все флаги в пакете должны быть сброшены. Как и ранее, символ ! означает инверсию критерия Важно: имена флагов в каждом списке должны разделяться запятыми, пробелы служат для разделения списков.
Критерий	--syn
Пример	iptables -p tcp --syn
Описание	Критерий --syn является по сути реликтом, перекочевавшим из ipchains. Критерию соответствуют пакеты с установленным флагом SYN и сброшенными флагами ACK и FIN. Этот критерий аналогичен критерию --tcp-flags SYN,ACK,FIN SYN. Такие пакеты используются для открытия соединения TCP. Заблокировав такие пакеты, вы надежно заблокируете все входящие запросы на соединение, однако этот критерий не способен заблокировать исходящие запросы на соединение. Как и ранее, допускается инвертирование критерия символом !. Так критерий ! --syn означает все пакеты, не являющиеся запросом на соединение, то есть все пакеты с установленными флагами FIN или ACK.
Критерий	--tcp-option
Пример	iptables -p tcp --tcp-option 16
Описание	Удовлетворяющим условию данного критерия будет будет считаться пакет, TCP параметр которого равен заданному числу. TCP Option - это часть заголовка пакета. Она состоит из 3 различных полей. Первое 8-ми битовое поле содержит информацию об опциях, используемых в данном соединении. Второе 8-ми битовое поле содержит длину поля опций. Если следовать стандартам до конца, то следовало бы реализовать

	обработку всех возможных вариантов, однако, вместо этого мы можем проверить первое поле, и в случае, если там указана неподдерживаемая нашим МСЭом опция, то просто перешагнуть через третье поле (длина которого содержится во втором поле). Пакет, который не будет иметь полного TCP заголовка, будет сброшен автоматически при попытке изучения его TCP параметра. Как и ранее, допускается использование флага инверсии условия [!].

UDP-критерии - эти расширения подгружаются автоматически при указании типа протокола --protocol UDP. Важно отметить, что пакеты UDP не ориентированы на установленное соединение, и поэтому не имеют различных флагов, которые дают возможность судить о предназначении датаграммы. Получение UDP пакетов не требует какого-либо подтверждения со стороны получателя. Если они потеряны, то они просто потеряны (не вызывая передачу ICMP сообщения об ошибке). Это предполагает наличие значительно меньшего числа дополнительных критериев, в отличие от TCP пакетов.

Таблица 21 – UDP критерии

Критерий	--sport, --source-port
Пример	iptables -A INPUT -p udp --sport 53
Описание	Исходный порт, с которого был отправлен пакет. В качестве параметра может указываться номер порта или название сетевой службы. Соответствие имен сервисов и номеров портов можно найти в файле /etc/services При указании номеров портов правила отрабатывают несколько быстрее. однако это менее удобно при разборе листингов скриптов. Номера портов могут задаваться в виде интервала из

	минимального и максимального номеров, например --source-port 22:80. Если опускается минимальный порт, то есть когда критерий записывается как --source-port :80, то в качестве начала диапазона принимается число 0. Если опускается максимальный порт, то есть когда критерий записывается как --source-port 22:, то в качестве конца диапазона принимается число 65535. Допускается такая запись --source-port 80:22, в этом случае iptables поменяет числа 22 и 80 местами, то есть подобного рода запись будет преобразована в --source-port 22:80. Как и раньше, символ ! используется для инверсии. Так критерий --source-port ! 22 подразумевает любой порт, кроме 22. Инверсия может применяться и к диапазону портов, например --source-port ! 22:80.
Критерий	--dport, --destination-port
Пример	iptables -A INPUT –p udp --dport 53
Описание	Порт, на который адресован пакет. Формат аргументов полностью аналогичен принятому в критерии --source-port.

ICMP критерии. Этот протокол используется, как правило, для передачи сообщений об ошибках и для управления соединением. Он не является подчиненным IP протоколу, но тесно с ним взаимодействует, поскольку помогает обрабатывать ошибочные ситуации. Заголовки ICMP пакетов очень похожи на IP заголовки, но имеют и отличия. Главное свойство этого протокола заключается в типе заголовка, который содержит информацию о том, что это за пакет. Например, при попытке соединиться с недоступной машиной в ответ придет сообщение ICMP host unreachable. Существует только один специфичный критерий для ICMP пакетов. Это расширение загружается автоматически, при указании критерия --protocol

ICMP. Заметьте, что для проверки ICMP пакетов могут употребляться и общие критерии, поскольку известны и адрес источника и адрес назначения и пр.

Таблица 22 – ICMP критерии

Критерий	--icmp-type
Пример	iptables -A INPUT -p icmp --icmp-type 8
Описание	Тип сообщения ICMP Тип сообщения ICMP определяется номером или именем. Числовые значения определяются в RFC 792. Чтобы получить список имен ICMP значений выполните команду iptables --protocol icmp --help. Как и ранее, символ ! инвертирует критерий, например --icmp-type!8.

Явные критерии

Перед использованием этих расширений, они должны быть загружены явно, с помощью ключа -m или --match. Все отличие между явными и неявными критериями заключается только в том, что первые нужно подгружать явно, а вторые подгружаются автоматически.

MAC критерий используется для проверки исходного MAC-адреса пакета. Модуль -m mac, на сегодняшний день, предоставляет единственный критерий, но возможно в будущем он будет расширен и станет более полезен.

Таблица 23 – MAC критерии

Модуль расширения должен подгружаться явно ключом -m mac. Критерий	--mac-source

Пример	iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01
Описание	MAC адрес сетевого узла, передавшего пакет. MAC адрес должен указываться в форме XX:XX:XX:XX:XX:XX. Как и ранее, символ ! используется для инверсии критерия, например --mac-source ! 00:00:00:00:00:01, что означает - пакет с любого узла, кроме узла, который имеет MAC адрес 00:00:00:00:00:01 Этот критерий имеет смысл только в цепочках PREROUTING, FORWARD и INPUT и нигде более.

Критерий lmit - должен подгружаться явно ключом -m limit. Прекрасно подходит для правил, производящих запись в системный журнал (logging) и т.п. Добавляя этот критерий, можно ограничить предельное число пакетов в единицу времени, которое способно пропустить правило. Можно использовать символ ! для инверсии, например -m ! limit. В этом случае подразумевается, что пакеты будут проходить правило только после превышения ограничения.

Таблица 24 – Критерий limit

Критерий	--limit
Пример	iptables -A INPUT -m limit --limit 3/hour

Описание	Устанавливается максимальное количество пакетов за единицу времени, к которому данное правило будет применено при совпадении всех прочих условий. В качестве аргумента указывается число пакетов и время. Допустимыми считаются следующие единицы измерения времени: /second /minute /hour /day. По умолчанию принято значение 3 пакета в час, или 3/hour. Использование флага инверсии условия [!] в данном критерии недопустим.
Критерий	--limit-burst
Пример	iptables -A INPUT -m limit --limit-burst 5
Описание	Устанавливает максимальное значение числа burst limit для критерия limit. Это число увеличивается на единицу, если получен пакет, подпадающий под действие данного правила, и при этом средняя скорость (задаваемая ключом --limit) поступления пакетов уже достигнута. Так происходит до тех пор, пока число burst limit не достигнет максимального значения, устанавливаемого ключом --limit-burst. После этого правило начинает пропускать пакеты со скоростью, задаваемой ключом --limit. Значение по-умолчанию принимается равным 5.

Расширение multiport позволяет указывать в тексте правила несколько портов и диапазонов портов. Недопустимо использовать стандартную проверку портов и расширение -m multiport (например --sport 1024:63353 -m multiport --dport 21,23,80) одновременно. Подобные правила будут просто отвергаться iptables.

Таблица 25 – Расширение Multiport

Критерий	--source-port
Пример	iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110
Описание	Служит для указания списка исходящих портов. С помощью данного критерия можно указать до 15 различных портов. Названия портов в списке должны отделяться друг от друга запятыми, пробелы в списке не допустимы. Данное расширение может использоваться только совместно с критериями the -p tcp или -p udp. Главным образом используется как расширенная версия обычного критерия --source-port.
Критерий	--destination-port
Пример	iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110
Описание	Служит для указания списка входных портов. Формат задания аргументов полностью аналогичен -m multiport --source-port
Критерий	--port
Пример	iptables -A INPUT -p tcp -m multiport --port 22,53,80,110
Описание	Данный критерий проверяет как исходящий, так и входящий порт пакета. Формат аргументов аналогичен критерию --source-port и --destination-port. Обратите внимание на то что данный критерий проверяет порты обеих направлений, то есть если вы пишете-multiport --port 80, то под данный критерий подпадают пакеты, идущие с порта 80 на порт 80. .

Расширение mark предоставляет возможность «пометить» пакеты специальным образом. Mark - специальное поле, которое существует только в области памяти ядра и связано с конкретным пакетом. Может использоваться в самых разнообразных целях, например, ограничение трафика и фильтрация. На сегодняшний день существует единственная возможность установки метки на пакет в Linux -- это использование действия MARK. Поле mark представляет собой беззнаковое целое число в диапазоне от 0 до 4294967296 для 32-битных систем.

Таблица 26 – Расширение mark

Критерий	--mark
Пример	iptables -t mangle -A INPUT -m mark --mark 1
Описание	Критерий производит проверку пакетов, которые были предварительно «помечены». Метки устанавливаются действием MARK, которое мы будем рассматривать ниже. Все пакеты, проходящие через netfilter, имеют специальное поле mark. Следует помнить, что нет никакой возможности передать состояние этого поля вместе с пакетом в сеть. Поле mark является целым беззнаковым, таким образом можно создать не более 65535 различных меток. Допускается использовать маску с меткам. В данном случае критерий будет выглядеть подобным образом: --mark 1/1. Если указывается маска, то выполняется логическое AND метки и маски.

Расширение owner предназначено для проверки «владельца» пакета. Изначально данное расширение было написано как пример демонстрации возможностей iptables. Допускается использовать этот критерий только в

цепочке OUTPUT. Такое ограничение наложено потому, что на сегодняшний день нет реального механизма передачи информации о «владельце» по сети. Справедливости ради следует отметить, что для некоторых пакетов невозможно определить «владельца» в этой цепочке. К пакетам такого рода относятся различные ICMP ответы. Поэтому не следует употреблять этот критерий к ICMP responses пакетам.

Таблица 27 – Расширение owner

Критерий	--uid-owner
Пример	iptables -A OUTPUT -m owner --uid-owner 500
Описание	Производится проверка «владельца» по User ID (UID). Подобного рода проверка может использоваться, к примеру, для блокировки выхода в Интернет отдельных пользователей.
Критерий	--gid-owner
Пример	iptables -A OUTPUT -m owner --gid-owner 0
Описание	Производится проверка «владельца» пакета по Group ID (GID).
Критерий	--pid-owner
Пример	iptables -A OUTPUT -m owner --pid-owner 78
Описание	Производится проверка «владельца» пакета по Process ID (PID).
Критерий	--sid-owner

Пример	iptables -A OUTPUT -m owner --sid-owner 100
Описание	Производится проверка Session ID пакета. Значение SID наследуются дочерними процессами от «родителя», так, например, все процессы HTTPD имеют один и тот же SID (примером таких процессов могут служить HTTPD Apache и Roxen).

Критерий state используется совместно с кодом трассировки соединений и позволяет получать информацию о трассировочном признаке состояния соединения, что позволяет судить о состоянии соединения, причем даже для таких протоколов как ICMP и UDP. Данное расширение необходимо загружать явно, с помощью ключа -m state.

Таблица 28 – Критерии state

Критерий	--state
Пример	iptables -A INPUT -m state --state RELATED,ESTABLISHED
Описание	Проверяется признак состояния соединения (state) На сегодняшний день можно указывать 4 состояния: INVALID, ESTABLISHED, NEW и RELATED. INVALID подразумевает, что пакет связан с неизвестным потоком или соединением и, возможно содержит ошибку в данных или в заголовке. ESTABLISHED указывает на то, что пакет принадлежит уже установленному соединению через которое пакеты идут в обеих направлениях. NEW подразумевает, что пакет открывает новое соединение или пакет принадлежит однонаправленному потоку. И наконец, RELATED указывает на то что пакет принадлежит уже существующему

	соединению, но при этом он открывает новое соединение Примером тому может служить передача данных по FTP, или выдача сообщения ICMP об ошибке, которое связано с существующим TCP или UDP соединением. Следует отметить, что признак NEW это не то же самое, что установленный бит SYN в пакетах TCP, посредством которых открывается новое соединение, и, подобного рода пакеты, могут быть потенциально опасны в случае, когда для защиты сети вы используете один сетевой экран. Более подробно эта проблема рассматривается ниже в данном документе

Критерий мусора unclean не имеет дополнительных ключей и для его использования достаточно явно загрузить модуль. Данная проверка производится для вычленения пакетов, которые имеют расхождения с принятыми стандартами, это могут быть пакеты с поврежденным заголовком или с неверной контрольной суммой и пр., однако использование этой проверки может привести к разрыву и вполне корректного соединения.

Критерий TOS предназначен для проведения проверки битов поля TOS. TOS -- Type Of Service -- представляет собой 8-ми битовое, поле в заголовке IP-пакета. Модуль должен загружаться явно, ключом -m tos.

Далее приводится описание поля TOS.

Данное поле служит для нужд маршрутизации пакета. Установка любого бита может привести к тому, что пакет будет обработан маршрутизатором не так как пакет со сброшенными битами TOS. Каждый бит поля TOS имеет свое значение. В пакете может быть установлен

только один из битов этого поля, поэтому комбинации не допустимы. Каждый бит определяет тип сетевой службы:

Минимальная задержка

Используется в ситуациях, когда время передачи пакета должно быть минимальным, то есть, если есть возможность, то маршрутизатор для такого пакета будет выбирать более скоростной канал. Например, если есть выбор между оптоволоконной линией и спутниковым каналом, то предпочтение будет отдано более скоростному оптоволокну.

Максимальная пропускная способность

Указывает, что пакет должен быть переправлен через канал с максимальной пропускной способностью. Например спутниковые каналы, обладая большей задержкой имеют высокую пропускную способность.

Максимальная надежность

Выбирается максимально надежный маршрут во избежание необходимости повторной передачи пакета. Примером могут служить PPP и SLIP соединения, которые по своей надежности уступают, к примеру, сетям X.25, поэтому, сетевой провайдер может предусмотреть специальный маршрут с повышенной надежностью.

Минимальные затраты

Применяется в случаях, когда важно минимизировать затраты (в смысле деньги) на передачу данных. Например, при передаче через океан (на другой континент) аренда спутникового канала может оказаться дешевле, чем аренда оптоволоконного кабеля. Установка данного бита вполне может привести к тому, что пакет пойдет по более «дешевому» маршруту.

Обычный сервис

В данной ситуации все биты поля TOS сброшены. Маршрутизация такого пакета полностью отдается на усмотрение провайдера.

Таблица 29 – Критерий TOS

Критерий	--tos
Пример	iptables -A INPUT –p tcp -m tos --tos 0x16
Описание	Данный критерий предназначен для проверки установленных битов TOS, которые описывались выше. Как правило поле используется для нужд маршрутизации, но вполне может быть использовано с целью «маркировки» пакетов для использования с iproute2 и дополнительной маршрутизации в linux. В качестве аргумента критерию может быть передано десятичное или шестнадцатиричное число, или мнемоническое описание бита, мнемоники и их числовое значение вы можете получить выполнив команду iptables -m tos -h. Ниже приводятся мнемоники и их значения. Minimize-Delay 16 (0x10) (Минимальная задержка), Maximize-Throughput 8 (0x08) (Максимальная пропускная способность), Maximize-Reliability 4 (0x04) (Максимальная надежность), Minimize-Cost 2 (0x02) (Минимальные затраты), Normal-Service 0 (0x00) (Обычный сервис).

Критерий TTL (Time To Live) является числовым полем в IP заголовке. При прохождении очередного маршрутизатора, это число уменьшается на 1. Если число становится равным нулю, то отправителю пакета будет передано ICMP сообщение типа 11 с кодом 0 (TTL equals 0 during transit) или с кодом 1 (TTL equals 0 during reassembly) . Для использования этого критерия необходимо явно загружать модуль ключом -m ttl.

Таблица 30 – Критерий TTL

Критерий	--ttl
Пример	iptables -A OUTPUT -m ttl --ttl 60
Описание	Производит проверку поля TTL на равенство заданному значению. Данный критерий может быть использован при наладке локальной сети, например: для случаев, когда какая либо машина локальной сети не может подключиться к серверу в Интернете, или для поиска «троянов» и пр.

Действия и переходы

Действия и переходы сообщают правилу, что необходимо выполнить, если пакет соответствует заданному критерию. Чаще всего употребляются действия ACCEPT и DROP.

Описание переходов в правилах выглядит точно так же как и описание действий, то есть ставится ключ -j и указывается название цепочки правил, на которую выполняется переход. На переходы накладывается ряд ограничений, первое - цепочка, на которую выполняется переход, должна находиться в той же таблице, что и цепочка, из которой этот переход выполняется, второе - цепочка , являющаяся целью перехода должна быть создана до того как на нее будут выполняться переходы. Например, создадим цепочку tcp_packets в таблице filter с помощью команды

iptables -N tcp_packets

Теперь можно выполнять переходы на эту цепочку подобно

iptables -A INPUT -p tcp -j tcp_packets

То есть, встретив пакет протокола tcp, iptables произведет переход на цепочку tcp_packets и продолжит движение пакета по этой цепочке. Если

пакет достиг конца цепочки, то он будет возвращен в вызывающую цепочку (в нашем случае это цепочка INPUT) и движение пакета продолжится с правила, следующего за правилом, вызвавшем переход. Если к пакету во вложенной цепочке будет применено действие ACCEPT, то автоматически пакет будет считаться принятым и в вызывающей цепочке, и уже не будет продолжать движение по вызывающим цепочкам. Однако пакет пойдет по другим цепочкам в других таблицах.

Действие - это предопределенная команда, описывающая действие, которое необходимо выполнить, если пакет совпал с заданным критерием. Например, можно применить действие DROP или ACCEPT к пакету, в зависимости от требований. Существует и ряд других действий, которые описываются ниже в этой секции. В результате выполнения одних действий, пакет прекращает свое прохождение по цепочке, например DROP и ACCEPT, в результате других, после выполнения неких операций, продолжает проверку, например, LOG, в результате работы третьих даже видоизменяется, например DNAT и SNAT, TTL и TOS, но так же продолжает продвижение по цепочке.

Действие Accept. Данная операция не имеет дополнительных ключей. Если над пакетом выполняется действие ACCEPT, то пакет прекращает движение по цепочке (и всем вызвавшим цепочкам, если текущая цепочка была вложенной) и считается ПРИНЯТЫМ, тем не менее, пакет продолжит движение по цепочкам в других таблицах и может быть отвергнут там. Действие задается с помощью ключа -j ACCEPT.

Действие Drop. Данное действие просто «сбрасывает» пакет и iptables «забывает» о его существовании. «Сброшенные» пакеты прекращают свое движение полностью, то есть они не передаются в другие таблицы, как это происходит в случае с действием ACCEPT. Следует помнить, что данное действие может иметь негативные последствия, поскольку может оставлять незакрытые «мертвые» сокеты как на стороне

сервера, так и на стороне клиента, наилучшим способом защиты будет использование действия REJECT особенно при защите от сканирования портов.

Действие QUEUE ставит пакет в очередь на обработку пользовательскому процессу. Оно может быть использовано для нужд учета, проксирования или дополнительной фильтрации пакетов.

Действие RETURN прекращает движение пакета по текущей цепочке правил и производит возврат в вызывающую цепочку, если текущая цепочка была вложенной, или, если текущая цепочка лежит на самом верхнем уровне (например INPUT), то к пакету будет применена политика по умолчанию. Обычно, в качестве политики по умолчанию назначают действия ACCEPT или DROP

Для примера, допустим, что пакет идет по цепочке INPUT и встречает правило, которое производит переход во вложенную цепочку - --jump EXAMPLE_CHAIN. Далее, в цепочке EXAMPLE_CHAIN пакет встречает правило, которое выполняет действие --jump RETURN. Тогда произойдет возврат пакета в цепочку INPUT. Другой пример, пусть пакет встречает правило, которое выполняет действие --jump RETURN в цепочке INPUT. Тогда к пакету будет применена политика по молчанию цепочки INPUT.

Действие LOG - действие, которое служит для журналирования отдельных пакетов и событий. В журнал могут заноситься заголовки IP пакетов и другая интересующая вас информация. Информация из журнала может быть затем прочитана с помощью dmesg или syslogd либо с помощью других программ.

LOG имеет пять ключей, которые перечислены ниже.

Таблица 31 – Ключи для действия LOG

Ключ	--log-level
Пример	iptables -A FORWARD -p tcp -j LOG --log-level debug
Описание	Используется для задания уровня журналирования (log level). Полный список уровней можно найти в руководстве (man) по syslog.conf. Обычно, можно задать следующие уровни: debug, info, notice, warning, warn, err, error, crit, alert, emerg и panic. Ключевое слово error означает то же самое, что и err, warn - warning и panic - emerg. Важно: в последних трех парах слов не следует использовать error, warn и panic. Приоритет определяет различия в том как будут заноситься сообщения в журнал. Все сообщения заносятся в журнал средствами ядра. Если вы установите строку kern.=info /var/log/iptables в файле syslog.conf, то все ваши сообщения из iptables, использующие уровень info, будут заноситься в файл /var/log/iptables Однако, в этот файл попадут и другие сообщения, поступающие из других подсистем, которые используют уровень info.
Ключ	--log-prefix
Пример	iptables -A INPUT -p tcp -j LOG --log-prefix «INPUT packets»
Описание	Ключ задает текст (префикс), которым будут предваряться все сообщения iptables. Сообщения со специфичным префиксом затем легко можно найти, к примеру, с помощью grep. Префикс может содержать до 29 символов, включая и пробелы.
Ключ	--log-tcp-sequence

Пример	iptables -A INPUT -p tcp -j LOG --log-tcp-sequence
Описание	Этот ключ позволяет заносить в журнал номер TCP Sequence пакета. Номер TCP Sequence идентифицирует каждый пакет в потоке и определяет порядок «сборки» потока. Этот ключ потенциально опасен для безопасности системы, если системный журнал разрешает доступ «НА ЧТЕНИЕ» всем пользователям. Как и любой другой журнал, содержащий сообщения от iptables.
Ключ	--log-tcp-options
Пример	iptables -A FORWARD -p tcp -j LOG --log-tcp-options
Описание	Этот ключ позволяет заносить в системный журнал различные сведения из заголовка TCP пакета. Такая возможность может быть полезна при отладке. Этот ключ не имеет дополнительных параметров, как и большинство ключей действия LOG.
Ключ	--log-ip-options
Пример	iptables -A FORWARD -p tcp -j LOG --log-ip-options
Описание	Этот ключ позволяет заносить в системный журнал различные сведения из заголовка IP пакета. Во многом схож с ключом --log-tcp-options, но работает только с IP заголовком.

Действие MARK используется для установки меток для определенных пакетов. Это действие может выполняться только в пределах таблицы mangle. Установка меток обычно используется для нужд

маршрутизации пакетов по различным маршрутам, для ограничения трафика и т.п. Следует помнить, что «метка» пакета существует только в период времени, пока пакет не покинул МСЭ, то есть метка не передается по сети. Если необходимо как-то пометить пакеты, чтобы использовать маркировку на другой машине, то можете попробовать манипулировать битами поля TOS.

Таблица 32 – Ключи для действия MARK

Ключ	--set-mark
Пример	iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 2
Описание	Ключ --set-mark устанавливает метку на пакет. После ключа --set-mark должно следовать целое беззнаковое число.

Действие REJECT используется, как правило, в тех же самых ситуациях, что и DROP, но в отличие от DROP, команда REJECT выдает сообщение об ошибке на машину, передавший пакет. Действие REJECT работает только в цепочках INPUT, FORWARD и OUTPUT (и во вложенных в них цепочках). Пока существует только единственный ключ, управляющий поведением команды REJECT.

Таблица 33 – Действие REJECT

Ключ	--reject-with
Пример	iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset
Описание	Указывает, какое сообщение необходимо передать в ответ, если пакет совпал с заданным критерием. При применении

	действия REJECT к пакету, сначала на машину-отправитель будет отослан указанный ответ, а затем пакет будет «сброшен». Допускается использовать следующие типы ответов: icmp-net-unreachable, icmp-host-unreachable, icmp-port-unreachable, icmp-proto-unreachable, icmp-net-prohibited и icmp-host-prohibited. По-умолчанию передается сообщение port-unreachable. Все вышеуказанные типы ответов являются ICMP error messages. В заключение укажем еще один тип ответа - tcp-reset, который используется только для протокола TCP. Если указано значение tcp-reset, то действие REJECT передаст в ответ пакет TCP RST, пакеты TCP RST используются для закрытия TCP соединений.

Действие TOS используется для установки битов в поле Type of Service IP заголовка. Поле TOS содержит 8 бит, которые используются для маршрутизации пакетов. Это один из нескольких полей, используемых iproute2. Так же важно помнить, что данное поле может обрабатываться различными маршрутизаторами с целью выбора маршрута движения пакета. Как уже указывалось выше, это поле, в отличие от MARK, сохраняет свое значение при движении по сети, а поэтому может использоваться вами для маршрутизации пакета. На сегодняшний день, большинство маршрутизаторов в Интернете никак не обрабатывают это поле, однако есть и такие, которые смотрят на него. Команда TOS имеет только один ключ, который описан ниже.

Таблица 34 – Действие TOS

Ключ	--set-tos
Пример	iptables -t mangle -A PREROUTING -p TCP --dport 22 -j TOS --

	set-tos 0x10
Описание	Ключ --set-tos определяет числовое значение в десятичном или шестнадцатиричном виде. Поскольку поле TOS является 8-битным, то можно указать число в диапазоне от 0 до 255 (0x00 - 0xFF). Однако, большинство значений этого поля никак не используются. Вполне возможно, что в будущих реализациях TCP/IP числовые значения могут быть изменены, поэтому, во избежание ошибок, лучше использовать мнемонические обозначения: Minimize-Delay (16 или 0x10), Maximize-Throughput (8 или 0x08), Maximize-Reliability (4 или 0x04), Minimize-Cost (2 или 0x02) или Normal-Service (0 или 0x00). По-умолчанию большинство пакетов имеют признак Normal-Service, или 0. Список мнемоник можно получить, выполнив команду iptables -j TOS -h.

Действие MIRROR может использоваться только для экспериментов и в демонстрационных целях, поскольку это действие может привести к «зацикливанию» пакета. В результате действия MIRROR в пакете, поля source и destination меняются местами, и пакет отправляется в сеть.

Данное действие допускается использовать только в цепочках INPUT, FORWARD и PREROUTING, и в цепочках, вызываемых из этих трех. Пакеты, отправляемые в сеть действием MIRROR, больше не подвергаются фильтрации, трассировке или NAT, избегая тем самым «зацикливания» и других неприятностей.

Действие SNAT используется для преобразования сетевых адресов (Source Network Address Translation), то есть изменение исходящего IP адреса в IP заголовке пакета. Например, это действие можно использовать

для предоставления выхода в Интернет другим компьютерам из локальной сети, имея лишь один уникальный IP адрес. Для этого. необходимо включить пересылку пакетов (forwarding) в ядре и затем создать правила, которые будут транслировать исходящие IP адреса нашей локальной сети в реальный внешний адрес. В результате, внешний мир ничего не будет знать о нашей локальной сети, он будет считать, что запросы пришли с МСЭ.

SNAT допускается выполнять только в таблице nat, в цепочке POSTROUTING. Другими словами, только здесь допускается преобразование исходящих адресов. Если первый пакет в соединении подвергся преобразованию исходящего адреса, то все последующие пакеты, из этого же соединения, будут преобразованы автоматически и не пойдут через эту цепочку правил.

Таблица 35 – Действие SNAT

Ключ	--to-source
Пример	iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000
Описание	Ключ --to-source используется для указания адреса, присваиваемого пакету. Все просто, вы указываете IP адрес, который будет подставлен в заголовок пакета в качестве исходящего. Дополнительно можно указать диапазон портов, которые будут использоваться только для нужд SNAT. Все исходящие порты будут после этого отображаться в заданный диапазон. iptables старается, по возможности избегать отображения портов, однако не всегда это возможно, и тогда производится отображение. Если диапазон портов не задан, то исходные порты ниже 512 отображаются в диапазоне 0-511,

	порты в диапазоне 512-1023 отображаются в диапазон 512-1023, и, наконец, порты из диапазона 1024-65535 отображаются в диапазон 1024-65535. Что касается портов назначения, то они не подвергаются отображению.

Действие DNAT (Destination Network Address Translation) используется для преобразования адреса места назначения в IP заголовке пакета. Если пакет подпадает под критерий правила, выполняющего DNAT, то этот пакет, и все последующие пакеты из этого же потока, будут подвергнуты преобразованию адреса назначения и переданы на требуемое устройство, хост или сеть. Данное действие может, к примеру, успешно использоваться для предоставления доступа к web-серверу, находящемуся в локальной сети, и не имеющему реального IP адреса. Для этого следует построить правило, которое перехватывает пакеты, идущие на HTTP порт МСЭ и выполняя DNAT передает их на локальный адрес web-сервера. Для этого действия так же можно указать диапазон адресов, тогда выбор адреса назначения для каждого нового потока будет производиться случайным образом.

Действие DNAT может выполняться только в цепочках PREROUTING и OUTPUT таблицы nat, и во вложенных подцепочках. Важно запомнить, что вложенные подцепочки, реализующие DNAT не должны вызываться из других цепочек, кроме PREROUTING и OUTPUT.

Таблица 36 – Действие DNAT

Ключ	--to-destination
Пример	iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10
Описание	Ключ --to-destination указывает, какой IP адрес должен быть

	подставлен в качестве адреса места назначения. В выше приведенном примере во всех пакетах, пришедших на адрес 15.45.23.67, адрес назначения будет изменен на один из диапазона от 192.168.1.1 до 192.168.1.10. Как уже указывалось выше, все пакеты из одного потока будут направляться на один и тот же адрес, а для каждого нового потока будет выбираться один из адресов в указанном диапазоне случайным образом. Можно также определить единственный IP адрес. Можно дополнительно указать порт или диапазон портов, на который (которые) будет перенаправлен трафик. Для этого после ip адреса через двоеточие укажите порт, например --to-destination 192.168.1.1:80, а указание диапазона портов выглядит так: --to-destination 192.168.1.1:80-100. Как можно видеть, синтаксис действий DNAT и SNAT во многом схож. Не следует забывайть, что указание портов допускается только при работе с протоколом TCP или UDP, при наличии опции --protocol в критерии.

Действие маскарадинга (MASQUERADE) в основе своей представляет то же самое, что и SNAT только не имеет ключа --to-source. Причиной тому то, что маскарадинг может работать, например, с dialup подключением или DHCP, то есть в тех случаях, когда IP адрес присваивается устройству динамически.

Маскарадинг подразумевает получение IP адреса от заданного сетевого интерфейса, вместо прямого его указания, как это делается с помощью ключа --to-source в действии SNAT. Действие MASQUERADE имеет хорошее свойство - «забывать» соединения при остановке сетевого интерфейса. В случае же SNAT, в этой ситуации, в таблице трассировщика остаются данные о потерянных соединениях, и эти данные могут

сохраняться до суток, поглощая ценную память. Эффект «забывчивости» связан с тем, что при остановке сетевого интерфейса с динамическим IP адресом, есть вероятность на следующем запуске получить другой IP адрес, но в этом случае любые соединения все равно будут потеряны, и было бы глупо хранить трассировочную информацию.

Действие MASQUERADE допускается указывать только в цепочке POSTROUTING таблицы nat, так же как и действие SNAT. MASQUERADE имеет ключ, описываемый ниже, использование которого необязательно.

Таблица 37 – Действие MASQUERADE

Ключ	--to-ports
Пример	iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000
Описание	Ключ --to-ports используется для указания порта источника или диапазона портов исходящего пакета. Можно указать один порт, например: --to-ports 1025, или диапазон портов как здесь: --to-ports 1024-3000. этот ключ можно использовать только в правилах, где критерий содержит явное указание на протокол TCP или UDP с помощью ключа --protocol.

Действие REDIRECT — выполняет перенаправление пакетов и потоков на другой порт той же самой машины. К примеру, можно пакеты, поступающие на HTTP порт перенаправить на порт HTTP proxy. Действие REDIRECT очень удобно для выполнения «прозрачного» проксирования (transparent proxying), когда машины в локальной сети даже не подозревают о существовании прокси.

REDIRECT может использоваться только в цепочках PREROUTING и OUTPUT таблицы nat. Для действия REDIRECT предусмотрен только

один ключ.

Таблица 38 – Действие REDIRECT

Ключ	--to-ports
Пример	iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
Описание	Ключ --to-ports определяет порт или диапазон портов назначения. Без указания ключа --to-ports, перенаправления не происходит, то есть пакет идет на тот порт, куда и был назначен. В примере, приведенном выше, --to-ports 8080 указан один порт назначения. Если нужно указать диапазон портов, то мы должны написать нечто подобное --to-ports 8080-8090. Этот ключ можно использовать только в правилах, где критерий содержит явное указание на протокол TCP или UDP с помощью ключа --protocol.

Действие TTL используется для изменения содержимого поля Time To Live в IP заголовке. Один из вариантов применения этого действия - это устанавливать значение поля Time To Live ВО ВСЕХ исходящих пакетах в одно и то же значение.

Действие TTL можно указывать только в таблице mangle и нигде больше. Для данного действия предусмотрено 3 ключа, описываемых ниже.

Таблица 39 – Действие TTL

Ключ	--ttl-set
Пример	iptables -t mangle -A PREROUTING -o eth0 -j TTL --ttl-set 64
Описание	Устанавливает поле TTL в заданное значение. Оптимальным считается значение около 64.

Ключ	--ttl-dec
Пример	iptables –t mangle -A PREROUTING -o eth0 -j TTL --ttl-dec 1
Описание	Уменьшает значение поля TTL на заданное число.
Ключ	--ttl-inc
Пример	iptables -t mangle -A PREROUTING -o eth0 -j TTL --ttl-inc 1
Описание	Увеличивает значение поля TTL на заданное число.

Действие ULOG предоставляет возможность журналирования пакетов в пользовательское пространство. Оно заменяет традиционное действие LOG, базирующееся на системном журнале. При использовании этого действия, пакет, через сокеты netlink, передается специальному демону, который может выполнять очень детальное журналирование в различных форматах (обычный текстовый файл, база данных MySQL и пр.) и к тому же поддерживает возможность добавления надстроек (плагинов) для формирования различных выходных форматов и обработки сетевых протоколов.

Таблица 40 – Действие ULOG

Ключ	--ulog-nlgroup
Пример	iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-nlgroup 2
Описание	Ключ --ulog-nlgroup сообщает ULOG, в какую группу netlink должен быть передан пакет. Всего существует 32 группы (от 1 до 32). По-умолчанию используется 1-я группа.
Ключ	--ulog-prefix
Пример	iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-prefix «SSH connection attempt: «

Описание	Ключ --ulog-prefix имеет тот же смысл, что и аналогичная опция в действии LOG. Длина строки префикса не должна превышать 32 символа.
Ключ	--ulog-cprange
Пример	iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-cprange 100
Описание	Ключ --ulog-cprange определяет, какую долю пакета, в байтах, надо передавать демону ULOG. Если указать число 100, как показано в примере, то демону будет передано только 100 байт из пакета, это означает, что демону будет передан заголовок пакета и некоторая часть области данных пакета. Если указать 0, то будет передан весь пакет, независимо от его размера. Значение по-умолчанию равно 0.
Ключ	--ulog-qthreshold
Пример	iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-qthreshold10
Описание	Ключ --ulog-qthreshold устанавливает величину буфера в области ядра. Например, если задать величину буфера равной 10, как в примере, то ядро будет накапливать журналируемые пакеты во внутреннем буфере и передавать в пользовательское пространство группами по 10 пакетов. По-умолчанию размер буфера равен 1 из-за сохранения обратной совместимости с ранними версиями ulogd, которые не могли принимать группы пакетов.

Практическое задание по теме

1. Фильтрация входящего трафика по критерию «отправитель».

В общем виде правило для фильтрации трафика по критерию «отправитель» будет создано следующим образом:

iptables -t filter -I INPUT -s source_addr_or_name -j {DROP/REJECT} [--reject-with reject_type]

Здесь в квадратные скобки заключен необязательный параметр.

Составить правила для фильтрации входящего трафика с лабораторных машин с целями DROP и REJECT. При использовании цели REJECT проверить влияние параметра --reject-with на результаты сканирования. В отчете указать различия при фильтрации трафика целями DROP и REJECT. При проверке защищенности машины использовать утилиы ping и nmap.

2. Фильтрация входящего трафика по критерию «протокол».

В общем виде правило для фильтрации трафика по критерию «протокол» будет создано следующим образом:

iptables -t filter -I INPUT -p {tcp/udp/icmp/all} -j {DROP/REJECT} [--reject-with reject_type]

Здесь в квадратные скобки заключен необязательный параметр.

Составить правила для фильтрации входящего трафика с лабораторных машин с целями DROP и REJECT по протоколам tcp,udp и icmp. При использовании цели REJECT проверить влияние параметра --reject-with на результаты сканирования. В отчете указать различия при фильтрации трафика целями DROP и REJECT.

При проверке защищенности машины использовать утилиту nmap.

3. Фильтрация входящего трафика по критерию «порт назначения»

В общем виде правило для фильтрации трафика по критерию «порт назначения» будет задано следующим образом:

iptables -t filter -I INPUT -p {tcp/udp/icmp/all} --dport port -j {DROP/REJECT} [--reject-with reject_type]

Здесь в квадратные скобки заключен необязательный параметр.

Составить правила для фильтрации входящего трафика с лабораторных машин с целями DROP и REJECT по протоколам tcp,udp и icmp. При использовании цели REJECT проверить влияние параметра --reject-with на результаты сканирования. В отчете указать различия при фильтрации трафика целями DROP и REJECT.

При проверке защищенности машины использовать утилиту nmap.

4. Фильтрация входящего трафика по критерию «входной интерфейс»

В общем виде правило для фильтрации трафика по критерию «входной интерфейс» будет задано следующим образом:

iptables -t filter -I INPUT [-p {tcp/udp/icmp/all}] [--dport port] -j {DROP/REJECT} [--reject-with reject_type] -i interface_name

Здесь в квадратные скобки заключены необязательные параметры.

Составить правила для фильтрации входящего трафика с лабораторных машин с целями DROP и REJECT. При использовании цели REJECT проверить влияние параметра --reject-with на результаты сканирования. В отчете указать различия при фильтрации трафика целями DROP и REJECT. При проверке защищенности машины использовать утилиту nmap.

5. Маскировка машины путем блокирования ICMP-трафика

В общем виде правило будет задано следующим образом:

iptables -t filter -I INPUT -p icmp --icmp-type type -j {DROP/REJECT} [--reject-with reject_type]

Здесь в квадратные скобки заключен необязательный параметр.

Составить правила для фильтрации входящего icmp-трафика с лабораторных машин с целями DROP и REJECT. При использовании цели REJECT проверить влияние параметров --rejectwith и --icmp-type на результаты сканирования. В отчете указать различия при фильтрации трафика целями DROP и REJECT.

Для получения возможных аргументов ключа --icmp-type использовать команду iptables -p icmp -h. При проверке маскировки машины использовать утилиту nmap.

6. Журналирование пакетов

В общем виде правило для журналирования пакетов по определенному критерию будет задано следующим образом:

iptables -t filter -I INPUT [other options,like proto and so] -j LOG [--log-level level --log-prefix prefix --log-tcp-sequence --log-tcpsequence --log-tcp-sequence]

Здесь в квадратные скобки заключены необязательные параметры.

Правила с целью LOG являются прозрачными, то есть пакет, обработанный по правилу с целью LOG, продолжает свое движение по цепочке правил. Написать правила для журналирования:

1) всех icmp-пакетов;

2) пакетов, приходящих на 22 порт;

3) всех пакетов, приходящих на интерфейс eth0.

Список рекомендуемой литературы

1. Гордеев, А. В. Операционные системы [Текст] : учебник / А. В. Гордеев. - 2-е изд. - СПб. : Питер, 2004. - 416 с. - ISBN 5-94723-632-X..

2. Бэкон, Д. Операционные системы. Параллельные и распределенные системы [Текст] / Д. Бэкон, Т. Харрис. - СПб. : Питер, 2004. - 800 с. - ISBN 5-94723-969-8..

3. Лабораторные работы и лекции по Linux [Электронный ресурс] // Кафедра Информационных Компьютерных Технологий РХТУ им. Д.И. Менделеева: файловое хранилище. – Режим доступа: http://ikt.muctr.ru/images/naumenko/linux/2014/ – Загл. с экрана.

4. Береснев А. Л. Администрирование GNU/Linux с нуля. — 2-е изд., перераб. и доп. — СПб.:БХВ-Петербург, 2010. — 576 с.: ил. + (Дистрибутивы на CD-ROM) — (Системный администратор) ISBN 978-5-9775-0518-5.

5. Соболь М. Linux. Администрирование и системное программирование. 2-е изд. - СПб.: Питер, 2011. - 880 с.; ил.

6. Кофлер М. Linux. Установка, настройка, администрирование. - СПб.: Питер, 2014. - 768 с.

У ч е б н о е и з д а н и е

Антон Викторович **Аникин**
Ирина Георгиевна **Жукова**
Дмитрий Васильевич **Литовкин**
Илья Сергеевич **Гурьянов**

**ЛАБОРАТОРНЫЙ ПРАКТИКУМ
ПО ДИСЦИПЛИНЕ
«АДМИНИСТРИРОВАНИЕ
ОПЕРАЦИОННЫХ СИСТЕМ»**

Учебное пособие

Выпускающий редактор Л. П. Кузнецова
Темплан 2015 г. Поз. № 47.

Подписано в печать 03.12.15. Формат 60 х 84 1/16.
Бумага газетная. Печать офсетная. Усл. печ. л. 7,44. Уч.-изд. л. 5,57.
Тираж 30 экз. Заказ № _____

Волгоградский государственный технический университет.
400005, г. Волгоград, просп. им. В. И. Ленина, 28, корп. 1.

Отпечатано в типографии ИУНЛ ВолгГТУ.
400005, г. Волгоград, просп. им. В. И. Ленина, 28, корп. 7.