МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

А.И. Спивак, О.И. Спивак, И.С. Лебедев

СЕТЕВЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Учебное пособие

Санкт-Петербург

2016

Спивак А.И., Спивак О.И., Лебедев И.С. Сетевые операционные системы. Лабораторный практикум. – СПб: Университет ИТМО, 2016. – 56 с.

Лабораторный практикум посвящен вопросам функционирования операционных систем, а также основным системным программам. Практикум предназначен для формирования у студентов навыков по эффективному применению операционных систем для решения прикладных задач. Полученные знания позволят полноценно использовать операционные системы в области обеспечения информационной безопасности.

В полном объёме излагаемый материал рассчитан для подготовки студентов технических университетов по направлению: 090900 – «Информационная безопасность» и 090103 – «Организация и технология защиты информации».

Рекомендовано к печати учёным советом факультета ИБиКТ от 30 сентября 2016 г., протокол №7.

УНИВЕРСИТЕТ ИТМО

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Содержание

Введение 4

Структура пособия 4

Лабораторная работа №1. Базовые примитивы работы в операционной системе семейства GNU Linux. 5

Лабораторная работа №2. Мониторинг основных показателей функционирования операционной системы. 10

Лабораторная работа №3. Использование команд для управления основными параметрами операционной системы 15

Лабораторная работа №4. Управление внешними устройствами в операционной системе Linux 19

Лабораторная работа №5. Начальная загрузка операционной системы и периодические процессы 22

Лабораторная работа №6. Сетевые средства мониторинга операционной системы Linux 25

Лабораторная работа №7. Файловые системы операционных систем 31

Лабораторная работа №8. Менеджер томов в GNU Linux и программный RAID 36

Лабораторная работа №9. Обеспечение безопасности в среде операционной системы GNU Linux 39

Лабораторная работа №10. Виртуализация операционных систем в среде GNU Linux 42

Лабораторная работа №11. Написание Shell скриптов 43

Лабораторная работа №12. Утилиты мониторинга производительности в среде операционной системы GNU Linux 45

Лабораторная работа №13. Работа с системой учета и регистрации событий операционной системы GNU Linux 47

Лабораторная работа №14. Использование сетевых файловых систем в операционной системе GNU Linux. 50

Список литературы 54

Введение

Данное учебное пособие предназначено для бакалавров по курсу дисциплины «Сетевые операционные системы». Знакомство с операционной системой (ОС) предполагает получение навыков для работы с основными системными утилитами и знаний о возможностях их использования. Цель данного пособия показать в ходе выполнения комплекса лабораторных и практических работ как корректно и эффективно использовать различные системные программы операционных систем. Их изучение позволит эффективно использовать ОС для решение прикладных задач, а также получить дополнительных навыки по определению и устранению сбоев и неполадок в работе ОС. Лабораторные работы охватывают широкий спектр вопросов работы операционной системы, начиная от начальной загрузки, работы с процессами, памятью, дисковой подсистемой и заканчивая особенностями функционирования прикладных системных процессов таких как регистрация событий. Отдельное внимание посвящено работе операционной системы в сетевом окружении – этот вопрос охватывается аспектами настройки сетевых файловых систем и мониторингом сетевой активности.

Учебное пособие будет полезно студентам для получения практических навыков работы в операционных системах Windows и GNU Linux.

Полученные знания помогут студентам в освоении последующих дисциплин направления «Информационная безопасность».

Структура пособия

Пособие представляет собой набор лабораторных работ, которые дополняют лекционный курс по дисциплине «Сетевые операционные системы». Порядок лабораторных работ совпадает с очередностью рассмотрения тем в лекционном курсе.

Каждая лабораторная работа включает в себя цель выполнения работы, теоретические сведения, охватывающие основные вопросы, знание которых необходимо для выполнения заданий лабораторной работы. После описания следует порядок выполнения лабораторной работы, где пошагово указываются действия, которые необходимо осуществить студенту. Заключительная часть лабораторной работы описывает порядок предоставления отчета преподавателю для защиты.

Лабораторная работа №1. Базовые примитивы работы в операционной системе семейства GNU Linux.

Цель работы: Знакомство с основными принципами работы в операционной системе Linux.

Теоретические сведения

Операционная система Linux впервые появилась в 1991 как собственная разработка финского студента Линуса Торвальдса. Для получения более широкой сферы применения Линус изменил лицензию на GNU GPL и опубликовал исходный код, что позволяло достаточно свободно использовать операционную систему для коммерческих целей. После этого к разработке операционной системы Linux подключилось значительное число разработчиков. Торвальдс до сих пор остается координатором в разработке ядра данной операционной системы. Благодаря свободной лицензии разработчиками со всего мира написано большое число приложений, которые позволяют использовать ОС и для серверных платформ и настольных компьютеров.

Некоторые особенности Linux

При разработке операционной системы уделялось особое внимание ее функционированию в качестве полноценной многопользовательской системы, в которой одновременно может работать множество пользователей.

Базовые функции ОС выделяются в ядро Linux, они определяют возможности по взаимодействию с аппаратным обеспечением, распределению оперативной памяти, управлению процессами, по работе с файловыми системами и т.п.

Архитектура ядра на основе выделения функциональных модулей является одной из отличительных черт ОС Linux. Одновременно с доступностью исходного кода ядра это позволяет скомпилировать и скомпоновать исполняемый код ядра с включением только того функционала, который необходим в каждом конкретном случае. Доступность возможности загрузки модулей уже в процессе работы ядра обеспечивает динамическое расширение функционала ядра при такой необходимости, а также экономию памяти, когда необходимость в функции, обеспечиваемой модулем, пропала.

Так как разработка ядра ведется сообществом разработчиков, необходима строгая организация процесса внесения изменений в код ядра. Весь процесс замкнут на Линуса Торвальдса, который оставляет за собой право одобрения или отклонения всех правок кода ядра ОС Linux.

Основные сведения. Командный интерпретатор

В данный момент времени широко распространены два способа взаимодействия с ОС: при помощи передачи команд и получения на них

ответов и использование графического интерфейса, где передача команд осуществляется преимущественно манипулятором-мышью, а ответы представляют собой визуальное изменение интерфейса. Изначально принятым в среде ОС Linux является использование первого из описанных методов взаимодействия. Для осуществления ОС возможности принятия команд используется специальная программа - командный интерпретатор, цель которого интерпретировать команды пользователя в конкретные действия в системе и отображать полученный результат. Экземпляр командного интерпретатора запускается для каждого пользователя и взаимодействует только с ним. В ОС Linux доступно для использования пользователями в качестве интерпретаторов несколько программ, которые имеют несущественные различия в поведении: sh, bash, ksh, zsh.

Умение работать в командном интерпретаторе позволяет достаточно эффективно управлять ОС, а также использовать все ее возможности для решения прикладных задач.

Команда man

Интерактивная справка о системах ОС, модулях, командах и утилитах осуществляется с использованием команды man.

Команды для работы в файловой системе

ls - вывод на экран списка файлов и каталогов в текущем каталоге
pwd - вывод имени текущего каталога
cd - смена текущего каталога
cp - копирование файлов и каталогов
mv - перемещение и переименовать файлов и каталогов
ln - создание ссылки для файла или каталога
rm - удаление файлов и каталогов
mkdir - создание нового каталога
rmdir - удаление пустого каталога
tar - утилита архивирования без сжатия
find - поиск файлов и каталогов по заданным критериям
gzip - утилита сжатия файлов

Работа с процессами

ps - отображение информации о процессах
kill - посылка сигнала процессу
killall - посылка сигнала всем процессам по имени программы
pstree - построение дерева процессов

Средства работы с сетью

ping - проверка работоспособности сетевого соединения при помощи пакетов ICMP и UDP
ifconfig - конфигурирование сетевых интерфейсов
arp – работа с таблицей ARP

tcpdump - программа-сетевой анализатор

traceroute - выполнение трассировки пути следования до заданного хоста

netstat - информация о портах и соединениях

route - работа с таблицей маршрутизации

Работа с текстовыми данными

cat - вывод содержимого файла на стандартный вывод

more - просмотр содержимого текстового файла постранично

less - просмотр содержимого текстового файла с возможностью вернуться к предыдущим страницам

head/tail — отображение первых/последних несколько строк файла

vi - текстовый редактор

wc - подсчет количества строк, слов и символов в текстовом файле

file - вывод типа данных файла

diff - сравнение двух текстовых файла

Работа с регулярными выражениями

sed - потоковый текстовый редактор, который способен применять преобразования к последовательному потоку текстовых данных

awk - обработка входного потока по заданным шаблонам

grep - поиск строк согласно заданным регулярным выражениям

Работа с командным интерпретатором

export - определение переменных окружения

set - вывод переменных окружения

alias - установка псевдонима для команды

<> - перенаправления ввода или вывода программы

| - организация конвейера, когда вывод одной команды подается на вход другой

& - запуск программы в фоновом режиме

Системные утилиты

uname - вывод информации о хосте

uptime - время работы системы

init # - переход на заданный уровень выполнения

who - список активных пользователей системы

reboot - перезагрузка системы

shutdown - выключение системы

passwd - установка/изменение пароля

groupadd - добавление группы

useradd - добавление учетной записи пользователя

id - вывод идентификатора учетной записи пользователя

mount - монтирование файловых систем

dmesg - вывод сообщений от ядра ОС

Команды для выполнения диагностики системы
vmstat - статистика использования системной памяти
free - информация об используемой и свободной памяти
mpstat - статистика использования процессора
sar - статистика использования системных ресурсов
iostat - статистика использования подсистемы ввода-вывода

Работа с правами доступа
chmod - изменение прав доступа к файлам и каталогам
chgrp - изменения группы-владельца для файлов и каталогов
chown - изменение владельца для файлов и каталогов

Порядок выполнения работы

В соответствии с вариантом выполнить перечисленные ниже задания. Сохранить выполненные команды и их вывод. В отчете описать команды, использованные в ходе работе и дать для них исчерпывающие объяснения.

Вариант 1

1. Вывести на экран таблицу маршрутизации хоста.
2. Выяснить имя компьютера и его архитектуру.
3. Создать текстовый файл. Написать в нем три цифры 1, каждая в новой строке. Затем используя команду sed заменить цифры 1 на 3.
4. Определить шлюз по умолчанию для хоста.
5. Вывести на экран информацию о запущенных процессах в структурированном виде.

Вариант 2

1. Вывести статистическую информацию о системном вводе-выводе за последние 25 секунд.
2. Выполнить перехват сетевых пакетов, имеющих тип ICMP.
3. Добавить к переменной окружения PATH путь /usr/local/bin.
4. Вывести на экран список файлов и каталогов в каталоге /var/log. упорядоченных по последней дате изменения.
5. Вывести на экран все имена пользователей из файла /etc/passwd, имена которых начинаются с ad.

Вариант 3

1. Вывести рекурсивно на экран список файлов и каталогов в директории /var/log.
2. Вывести дерево процесса init.
3. Посчитать количество слов какого-либо файла.

4. Вывести на экран информацию о сетевых соединениях в режиме established.
5. Вывести информацию о версии ядра ОС.

Вариант 4

1. Выяснить идентификатор пользователя mail.
2. Узнать список смонтированных файловых систем.
3. Вывести на экран информацию об одном из сетевых интерфейсов и узнать на какой скорости работает сетевой адаптер.
4. Установить права доступа для файла так, чтобы только владелец мог его читать.
5. Найти в сообщениях от ядра ОС упоминание о сетевом интерфейсе.

Вариант 5

1. Перенаправить вывод команды, получающей список переменных окружения в файл. После этого используя команду awk вывести на экран только названия переменных.
2. Вывести содержимое таблицы ARP.
3. Определить количество запущенных процессов bash.
4. Найти все файлы в директории /var/log, имеющие расширение log.
5. Вывести на экран последние 32 строки файла.

Лабораторная работа №2. Мониторинг основных показателей функционирования операционной системы.

Цель работы: Получение сведений об основных показателях функционирования операционных систем Windows и Linux

Теоретические сведения

Описание используемых в работе утилит.

Для ОС Windows:

Сведения о системе – Программа для отображения данных о конфигурации системы. Данные отображаются как для локального компьютера, так и для удаленного компьютера, с которым предварительно необходимо установить соединение. Запуск программы осуществляется запуском **msinfo32** в «Пуск» → «Выполнить» или выполнив переход «Пуск» → «Все программы» → «Стандартные» → «Служебные» → «Сведения о системе».

Диспетчер задач – программа мониторинга за активностью в ОС Windows, отображает сведения о выполняющихся процессах, используемых ресурсах и загруженности компьютера. Запуск диспетчера осуществляется при помощи одновременного нажатия комбинации клавиш «Ctrl+Shift+Esc».

Системный монитор – компонент, позволяющий производить оценку производительности локального компьютера. Также можно установить соединение с компонентом, выполняющимся на удаленном компьютере и отображать данные с него. К основным возможностям можно отнести:

- сбор и просмотр в реальном времени информации, отражающей производительность локального или удаленного компьютеров;
- просмотр ранее сохраненных в журналах счетчиков данных;
- визуализация данных в виде графиков, диаграмм и табличного отчета;
- интеграция функциональных возможностей в графические объекты, поддерживающие элементы управления ActiveX, например, веб-страницы, Microsoft Word, а также другие приложения, входящие в состав набора Microsoft Office;
- создание страниц HTML с отчетами о производительности;
- создание и сохранение конфигураций параметров проведения диагностики, которые можно впоследствии использовать на других компьютерах в консоли управления MMC.

Данный компонент операционной системы Windows позволяет производить оценку производительности системы и выявлять наиболее

загруженные элементы. Справка Windows содержит таблицу пороговых значений, которые можно использовать в процессе диагностики системы.

Кроме того, начиная с операционной системы Windows 7, в состав этого компонента вошла самостоятельная ранее программа **«Оповещения и журналы производительности»**. Это средство использует так называемые группы сборщиков данных. Оно позволяет собирать данные о производительности компьютера в автоматическом режиме. Их можно сохранять в файл, а затем просматривать с помощью системного монитора или в программе работы с электронными таблицами для анализа и создания отчета.

Для получения доступа к описанным компонентам ОС Windows необходимо запустить оснастку **perfmon.msc**. Для выполнения запуска набрать **perfmon** в «Пуск» → «Выполнить» или найти и запустить файл perfmon.exe в папке C:\Windows\System32.

Для ОС Linux:

Для доступа к данным о производительности системы используется утилита **sysctl**, она осуществляет изменение параметров, хранящихся в виртуальной файловой системе **/proc**. Некоторые из них указаны ниже:

/proc/cmdline: Содержит информацию о параметрах ядра, полученных при загрузке системы.

/proc/cpuinfo: Содержит информацию о процессоре компьютера, на котором выполняется ядро.

/proc/meminfo: Содержит информацию об оперативной памяти компьютера.

/proc/loadavg: Содержит данные о средней загрузке процессора, включая информацию за разные промежутки времени (1 минута, 5 минут, 10 минут).

proc/devices: Содержит данные о символьных и блочных устройствах, имеющихся в системе.

/proc/filesystems: Содержит список поддерживаемых типов файловых систем.

Системный журнал, как правило, выводит события в файл **/var/log/syslog**, события от ядра системы содержатся в файле **/var/log/dmesg**.

Наиболее часто используемые утилиты для мониторинга быстродействия системы и используемых ресурсов объединены в пакете утилит **sysstat**. В данный пакет входят утилита **iostat**, отображающая информацию об использовании системой центрального процессора и устройств хранения, и утилита **sar** для мониторинга активности операционной системы.

Утилита **sar** позволяет опрашивать состояние системы с некоторым интервалом. Например, команда **sar -uR 1 5** будет опрашивать загрузку процессора и состояние памяти 5 раз с интервалом в 1 секунду. По умолчанию, **sar** обновляет статистику системы каждые 10 минут, однако, чтобы обновление работало корректно в ОС Ubuntu Linux необходимо в файле **/etc/defaults/sysstat** поменять параметр **ENABLED** на **true**. После этого, с помощью вызова **sar –uR** (к примеру) можно просмотреть статистику использования процессора и памяти в интервалах по 10 минут. В конце файла со статистикой также будут показаны усредненные показатели.

Описание установки для проведения лабораторной работы

Установка для выполнения лабораторной работы представляет собой рабочую станцию, функционирующую под управлением ОС MS Windows 7. На каждой рабочей станции установлен пакет VirtualBox, в котором могут быть запущены операционные системы Windows 7 и Linux.

Порядок выполнения работы

1 Запуск установки

1.1 Войти в систему. Запустить в среде VirtualBox операционную систему Windows.

1.2 Выполнить вход в операционную систему Windows.

2 Мониторинг основных показателей работы ОС Windows

2.1 Запустите программу «Сведения о системе» и ознакомьтесь с её возможностями.

2.2 Сохраните данные о конфигурации в файл, выполнив «Файл» → «Экспорт». Включите в отчет общие сведения о системе, данные об адаптерах сети, связанные с сетевыми подключениями, а также иные сведения на ваш выбор.

2.3 Откройте программу «Диспетчер задач» и ознакомьтесь с её возможностями.

2.4 Перейдите во вкладку «Процессы». Выполните «Вид» → «Выбрать столбцы», ознакомьтесь с дополнительными параметрами и включите отображение каких-либо из них. Включите в отчет скриншот со списком процессов (недоступно в более поздних версиях ОС Windows).

2.5 Перейдите во вкладку «Быстродействие», ознакомьтесь с отображаемыми показателями. Включите в отчет скриншот содержимого вкладки.

2.6 Перейдите во вкладку «Сеть», ознакомьтесь с отображаемыми показателями. Наблюдайте изменение графика загрузки при передаче каких-либо данных по сети. Для этого можно открыть какой-либо сайт или начать передачу данных на какой-либо сетевой узел. Включите в отчет скриншот содержимого вкладки.

2.7 Откройте компонент «Системный монитор» и ознакомьтесь с его возможностями.

2.8 На панели инструментов нажмите кнопку «Добавить» (зеленый знак «+»), добавляющую счетчики. Ознакомьтесь с объектами и счетчиками, которые можно добавить.

2.9 Выберите группу «Сетевой интерфейс». Выберите экземпляры выбранного объекта. Включите счетчик сетевой активности для уровня каналов передачи данных. Для этого для объекта «Сетевой интерфейс» добавьте счетчик «Всего байт/сек». После добавления откройте какой-либо сайт или начните передачу данных на какой-либо сетевой узел. Дождитесь появления изменений на диаграмме.

2.10 Добавьте дополнительные счетчики и протестируйте их.

2.11 Сделайте скриншот системного монитора и включите в отчет.

2.12 Сохраните параметры в файл, вызвав контекстное меню и выбрав «Сохранить как».

2.13 Откройте папку «Группы сборщиков данных». Ознакомьтесь с содержимым.

2.13.1 Создайте во вкладке «Особый» группу сборщиков данных. Для этого откройте вкладку. Затем щелкните правой кнопкой мыши в основном поле окна. В контекстном меню выберите «Создать – Группа сборщиков данных». Затем, следуя инструкциям, создайте сначала группу сборщиков данных из шаблона, а затем – вручную.

2.13.2 Откройте одну из групп сборщиков данных. Добавьте пару счетчиков данных на свое усмотрение. Затем откройте свойства одного из сборщиков. Выберите двоичный формат журнала. Щелкните в дереве слева по группе сборщиков данных. Выберите в свойствах (из контекстного меню) папку, в которую будет собираться протоколируемая информация. Затем запустите группу с помощью кнопки быстрого действия в верхней части окна. Откройте папку, выбранную ранее, откройте нужный файл и посмотрите результаты. По ходу работы занесите несколько скриншотов в отчет.

2.13.3 Повторите то же самое для других типов файлов журнала.

2.13.4 Сохраните шаблон группы сборщиков данных, выбрав соответствующий пункт контекстного меню (щелчок правой кнопкой мыши по группе). Затем этот шаблон можно будет использовать при создании новых групп сборщиков данных. Сделайте скриншот и занесите его в отчет.

3 Мониторинг основных показателей работы ОС Linux.

3.1 Ознакомиться с параметрами системы, хранящимися в виртуальной файловой системе **/proc**. Добавить в отчет информацию из **/proc/meminfo** и **/proc/cpuinfo.**

3.2 Ознакомиться с журналом ядра, выполнив команду **dmesg**. Отфильтровать только сообщения о процессоре с помощью **dmesg | grep CPU**. Полученные сообщения включить в отчет.

3.3 Ознакомиться с возможностями команды **iostat**.

3.4 Используя команду **iostat**, получить информацию о состоянии процессора и блочных устройств, включить ее в отчет.

3.5 Ознакомиться с возможностями команды **sar**.

3.6 Используя команду **sar**, выполнить мониторинг состояния памяти системы, центрального процессора, блочных устройств и сетевых интерфейсов. Мониторинг выполнить 1 раз, информацию включить в отчет.

Требования к оформлению отчета и защите

Отчет должен содержать описание порядка выполнения всех команд и содержание указанных при выполнении работы файлов. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся исследованного функционала.

Лабораторная работа №3. Использование команд для управления основными параметрами операционной системы

Цель работы: Получение сведений о средствах управления основными параметрами операционных систем Windows и Linux.

Теоретические сведения

Утилиты, используемые в работе:

Для изучения средств управления ОС Windows необходимо ознакомиться с основными программами:

Оснастки (snap-ins) – программы для управления различными подсистемами ОС Windows. Ниже перечислены некоторые из оснасток (в скобках указаны названия исполняемых файлов, предоставляющих доступ к оснасткам).

Редактор объекта групповой политики (gpedit.msc) — Управление политиками и конфигурацией операционной системы.

Службы (services.msc) — Управление службами Windows. Получение информации о службах, запуск и остановка служб.

Просмотр событий (eventvwr.msc) — Работа с журналами событий Windows.

Параметры безопасности (secpol.msc) – Управление политиками безопасности. Оснастка может быть включена как расширение оснастки **gpedit.msc**.

Управление компьютером (compmgmt.msc) — Управление компьютером и соответствующие служебные программы. Включает в себя некоторые из других оснасток, в том числе **services.msc** и **eventvwr.msc.** Данную оснастку можно открыть, вызвав контекстное меню у «ярлыка» «Мой компьютер» и нажав «Управление».

Получить доступ к оснасткам можно, набрав название соответствующего файла в «Пуск» → «Выполнить», набрав название в командной строке, или, запустив соответствующий файл в «[папка ОС Windows]\system32\[название файла]», где [папка ОС Windows] — путь к папке операционной системы, а [название файла] – название файла, запускающего оснастку.

Консоль управления Microsoft (Microsoft Management Console или MMC) – Программа, группирующая средства администрирования, которые используются для администрирования компьютеров, служб, других системных компонентов и сетей. Позволяет формировать произвольный набор оснасток.

Консоль управления Microsoft можно открыть, набрав **mmc** в «Пуск» → «Выполнить», набрав **mmc** в командной строке, или, запустив **mmc.exe** в «[папка ОС Windows]\system32\», где [папка ОС Windows] — путь к папке операционной системы.

MSConfig – утилита, позволяющая управлять параметрами автозагрузки, автозагрузкой приложений, драйверов и служб.

Утилиту **MSConfig** можно открыть, набрав её название в «Пуск» → «Выполнить».

Основные сведения для изучения средств управления ОС Linux.

proc - Виртуальная файловая система, использующаяся как интерфейс к ядру, в ней содержатся текущие параметры работы ядра и показатели функционирования. Большая часть параметров недоступно для изменения и может быть только прочтено. Информация распределена по директориям, отвечающим каждая за отдельный компонент функционирования системы.

/proc/sys/fs – Файлы, содержащие настройки файловой системы.

/proc/sys/kernel – Файлы, содержащие настройки ядра.

/proc/sys/net – Файлы, содержащие параметры сетевого взаимодействия.

/proc/sys/vm - Файлы, необходимые для управления памятью.

/proc/scsi/scsi - Информация о SCSI устройствах, доступных ядру ОС.

/proc/sys/fs/file-max

Содержит максимальное количество одновременно открытых дескрипторов файлов. Требует увеличение при получении сообщений о том, что достигнуто максимальное количество открытых файлов. По умолчанию значение – 4096.

/proc/sys/kernel/domainname
/proc/sys/kernel/hostname

Содержат сетевое и доменное имена компьютера.

/proc/sys/kernel/msgmax

Содержит максимальный размер сообщения, отправляемого между процессами в рамках межпроцессного взаимодействия. По умолчанию – 8192.

/proc/sys/kernel/ctrl-alt-del

Содержит параметр, отвечающий за реакции системы на нажатие комбинации клавиш Ctrl-Alt-Del. Значение 0 означает отправку этого значения программе init, для осуществления корректной остановки системы и перезагрузку. В случае значения больше 0, происходит немедленная перезагрузка. По умолчанию значение установлено как 0.

proc/sys/vm/buffermem

Содержит данные об использовании системной памяти в качестве буферной. Три значения, указанные через пробел, означают процент использования памяти под буфер и могут интерпретироваться следующим образом:

- Минимальный процент системной памяти для использования под буфер;
- Значения, которые система устанавливает для буфера в случае уменьшения количества доступной памяти;
- Максимальный процент системной памяти для использования под буфер.

По умолчанию используются следующие значения- 2 10 60.

proc/sys/net/core/message_burst

Содержит значение в десятых долях секунды, которое используется при записи нового предупреждения от ядра. В случае получения сообщения быстрее этого интервала они игнорируются. Используется для защиты от атак отказа в обслуживании, заключающихся в наводнении системы сообщениями. По умолчанию используется значение - 50 (5 секунд).

/proc/sys/net/core/netdev_max_backlog

Содержит параметр максимального количества сетевых пакетов в очереди на обработку ядром. Для случая, когда сетевой интерфейс получает их быстрее, чем ядро обрабатывает. По умолчанию используется значение - 300.

sysctl – системная утилита ОС Linux, которая позволяет менять параметры ядра в **/proc/sys** в реальном времени.

Описание установки для проведения лабораторной работы

Установка для выполнения лабораторной работы представляет собой рабочую станцию, функционирующую под управлением ОС Linux. На каждой рабочей станции установлен пакет VirtualBox, в котором могут быть запущены операционные системы Windows 7 и Linux.

Порядок выполнения работы

1. Запуск установки

1.1 Войдите в систему. Запустите в среде VirtualBox операционную систему Windows.

1.2 Выполните вход в операционную систему Windows.

2. Изучение средств управления в ОС Windows.

2.1 Запустите некоторые из оснасток и ознакомьтесь с их содержимым.

2.2 Запустите консоль управления **mmc** и ознакомьтесь с возможностями интерфейса.

2.3 Добавьте несколько оснасток. Для этого в меню нажмите «Консоль», далее «Добавить и удалить оснастку». В появившемся окне во вкладке «Изолированная оснастка» нажмите «Добавить» и выберите соответствующую оснастку. Попробуйте добавить ссылку на веб-ресурс, а также, с помощью добавления пустых папок, расширить древовидную структуру консоли оснасток.

2.4 Настройте расширения оснасток. Для этого в меню нажмите «Консоль», выберите «Добавить и удалить оснастку». В появившемся окне во вкладке «Расширения» будут доступны опции управления расширения оснастками. Ознакомьтесь с предоставляемыми возможностями и отключите часть расширений добавленных вами оснасток.

2.5 Сделайте скриншот дерева консоли mmc в раскрытом виде. Включите его в отчет.

2.6 Сохраните консоль в файл. Для этого в меню нажмите «Консоль», далее «Сохранить как». После этого, запуская сохраненный файл, будет происходить открытие консоли.

2.7 Добавьте сохраненную консоль в автозагрузку. Для этого создайте ярлык для файла консоли. Скопируйте ярлык в папку «Автозагрузка» (чтобы открыть папку, выполните «Пуск» -> «Все программы», вызовите контекстное меню папки «Автозагрузка», нажмите «Открыть»).

2.8 Перезагрузитесь, убедитесь в автозагрузке сохраненной консоли.

2.9 Откройте утилиту **msconfig**, ознакомьтесь с возможностями вариантов запуска, автозагрузки программ и служб.

2.10 Отключите автозагрузку сохраненной консоли. Сделайте скриншот списка автозагрузки приложений.

2.11 Перезагрузитесь, убедитесь в отсутствии загрузки сохраненной консоли. Включите автозагрузку консоли обратно. Перезагрузитесь, удалите ярлык запуска консоли из папки «Автозагрузка».

3. Изучение средств управления в ОС Linux.

3.1 Выполните вход в операционную систему Linux.

3.2 Вставьте какой-либо носитель. Откройте файл **/proc/scsi/scsi** и убедитесь в наличии этого носителя. Включите список носителей в отчет.

3.3 С помощью **sysctl** выведите список и значения параметров, относящихся к **/proc/sys/fs**, **/proc/sys/kernel**, **/proc/sys/net** и **/proc/sys/vm**. Включите в отчет параметры, относящиеся к **/proc/sys/kernel**.

Требования к оформлению отчета и защите

Отчет должен содержать описание порядка выполнения всех команд и содержание указанных при выполнении работы файлов. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся исследованного функционала.

Лабораторная работа №4. Управление внешними устройствами и модулями ядра в операционной системе GNU Linux.

Цель работы: Получение сведений об управлении внешними устройствами в операционной системе GNU Linux.

Теоретические сведения

Утилиты, используемые в работе:

Изначально задачи по управлению подключением и конфигурированием внешних устройств возлагались на пользователя, который самостоятельно должен был определять и подгружать модули для поддержки в ОС новых устройств созданием соответствующих файлов для работы с ними. Впоследствии данные функции были автоматизированы и объединены в менеджере устройств **udev**. **Udev** представляет собой менеджер устройств, который выполняет задачи по управлению файлами устройств в каталоге **/dev** и обработкой событий, возникающих при добавлении и отключении внешних устройств. В настоящий момент **udev** стало частью кода системы управления первоначальной загрузки ОС Linux - **systemd**.

Udev поддерживает в актуальном состоянии содержимое каталога **/dev**, где каждое устройство, подключенное к системе, имеет соответствующий ему файл. При подключении устройства файл создается, а при его отключении удаляется. Сам каталог находятся в виртуальной файловой системе, содержимое которой формируется каждый раз при старте системы.

Поддержка правил обработки событий от устройств позволяет **udev** расширить возможности путем создания гибких сценариев реагирования на сообщения, включая использование дополнительных скриптов и внешних программ обработки и задействование экспорта и импорта дополнительных данных для обработки событий ядром.

В составе системы **udev** можно выделить три составные части:

libudev – библиотека, использующаяся для получения доступа к сведениям об устройствах.

udevd – демон в пространстве пользователя, который выполняет основные задачи системы, а именно управление файлами в директории **/dev**.

udevadm – утилита для взаимодействия с функциями **udev**.

Утилита **udevadm** принимает следующие параметры вызова:

udevadm info – отображает информацию о текущих устройствах, зарегистрированных в базе данных **udev**.

udevadm trigger – вывод сообщений, отражающих события ядра ассоциированные с устройством.

udevadm settle – вывод очереди событий **udev**.

udevadm control – управление состоянием демона **udev**.

udevadm monitor – вывод сообщений ядра **uevent** и сообщений, инициируемых выполняющимися правилами **udev** с указанием путей к устройствам. Часто используется для анализа временных интервалов между событиями ядра **uevent** и событиями, переданными **udev**.

udevadm test – имитация заданных событий **udev** для выбранного устройства и вывод отладочной информации.

Модули ядра ОС Linux

Ядро Linux построено по модульному принципу, когда отдельный функционал ядра может динамически добавляться в текущее загруженное ядро путем загрузки соответствующего модуля. Для выполнения задач загрузки и выгрузки модулей ядра задействуются соответствующие утилиты.

modprobe – утилита для загрузки и выгрузки модуля из ядра Linux.

lsmod – утилита, которая извлекает информацию из файла **/proc/modules**, и отображает загруженные в данный момент модули ядра.

modinfo – утилита отображения сведений об указанном модуле ядра Linux.

Описание установки для проведения лабораторной работы

Установка для выполнения лабораторной работы представляет собой рабочую станцию, функционирующую под управлением ОС Linux. На каждой рабочей станции установлен пакет VirtualBox, в котором может быть запущена операционная система Linux.

Порядок выполнения работы

1. Запуск установки.

1.2 Войдите в систему. Запустите в среде VirtualBox операционную систему Linux.

1.3 Выполните вход в операционную систему Linux.

2. Изучение менеджера устройств **udev** и утилит работы с модулями ядра ОС Linux.

2.1 Ознакомьтесь с руководством программы **udev**.

2.2 Ознакомьтесь с возможностями утилиты **udevadm**.

2.3 Включите **udevadm** monitor и вставьте какой-либо носитель данных.

2.4 Ознакомьтесь со списком событий, включите скриншот списка в отчет.

2.5 С помощью данного списка событий или любым иным способом (например, командой **fdisk –l**) запомните название файла устройства (например, **sdb1**).

2.6 Используя название, получите информацию об устройстве, выполнив команду **udevadm info --query=all --name**=[название файла устройства]. Включите полученную информацию в отчет.

2.7 Перейдите в каталог **/lib/udev/rules.d** и ознакомьтесь с правилами, используемыми **udev**.

2.8 Перейдите в **/etc/udev/rules.d**. В этом каталоге содержатся пользовательские правила, а также правила, необходимые для замены «оригинальных» правил. Создайте файл правила с низким приоритетом (с числом в названии больше 90).

2.9 Включите в файл правило, срабатывающее при вставке носителя: KERNEL=="[название файла устройства]", ACTION=="add", RUN+="[команда]". команда – выполняемое действие, например **/bin/mkdir /home/administrator/new_dir**.

2.10 Перезагрузите выполняемые правила, выполнив команду **sudo udevadm control –reload-rules**. Вставьте носитель, убедитесь в выполнении указанного действия. Включите скриншот содержимого файла с правилом в отчет.

2.11 Попробуйте добавить иные правила.

2.12 Ознакомьтесь с возможностями утилит **modprobe**, **lsmod** и **modinfo**.

2.13 Выведите список загруженных модулей ядра. Включите его в отчет.

2.14 Выберите любой модуль из полученного ранее списка и выведите информацию о нем с помощью **modinfo**. Включите ее в отчет.

2.15 Выберите любой модуль из набора модулей, хранящихся в **/lib/modules**, и загрузите его в память. Проверьте, что модуль действительно загружен: для этого снова выведите список работающих модулей. Включите эту информацию в отчет.

Требования к оформлению отчета и защите.

Отчет должен содержать описание порядка выполнения всех команд и содержание указанных при выполнении работы файлов. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся исследованного функционала.

Лабораторная работа №5. Начальная загрузка операционной системы GNU Linux и периодические процессы.

Цель работы: Получение навыков написание стартовых скриптов для управления процессами, старт и завершение которых происходит вместе с ОС Linux. Использование системы cron.

Теоретические сведения

В системе Unix для обеспечения запуска системных и прикладных процессов при старте системы используются системы инициализации типа System V, также возможно использования сходных по функционалу других реализаций. Для управления процессом запуска в различных режимах имеется несколько уровней запуска, при переходе в каждый из уровней осуществляется запуск скриптов из соответствующей директории, с действием, которое происходит (старт или стоп). Для упрощения написания стартовых скриптов имеется возможность использовать пустой файл с уже заданной структурой и форматом имени файла. Такой файл-шаблон называется skeleton, его пример содержится на рисунке 1 ниже:

```
### BEGIN INIT INFO
# Provides:          skeleton
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Example initscript
# Description:       This file should be used to construct scripts to be
#                    placed in /etc/init.d.
### END INIT INFO
```

Рисунок 1 - Файл-шаблон для создания стартового скрипта

Описанные в заголовке параметры имеют специальное назначение [skeleton] и указывают на условия, при которых происходит запуск этого стартового скрипта. При написании своего собственного скрипта нужно его логические элементы вставить в соответствующие разделы файла. Таким образом получается единообразная и упорядоченная структура каждого файла запуска и обработка демоном **init** происходит в нужном порядке и с нужными параметрами.

После этого необходимо поместить файл в директорию **/etc/init.d** и при помощи команды (рисунок 2), добавить в автозагрузку, которая создаст соответствующие ссылки на стартовый скрипт из директорий соответствующих уровней запуска.

```
update-rc.d <имя_скрипта> defaults
```

Рисунок 2 - Команда для обновления ссылок на стартовые скрипты

В последних версиях операционных систем Linux все чаще встречается реализация с использованием демона **systemd**. Его концепция несколько отличается от уровней запуска, хотя и предполагается обратная совместимость скриптовой базы.

Первое отличие **systemd** от традиционного подхода заключается в использовании концепции юнитов – отдельных конфигурационных файлов для каждого аспекта работы сервиса. Типы юнитов бывают: системный сервис, точка автомонтирования, файл устройства и т.д.

Сервисные юниты типа **service** являются аналогами стартовых скриптов System V. Первое различие заключается в расширении системы уровней запуска до целей запуска - **targets**. С целью обратной совместимости существуют цели, соответствующие семи уровням запуска с 0 до 6. Файлы целей позволяют группировать вместе юниты, используя цепочки зависимостей. Такой подход является гораздо более гибким по сравнению с уровнями запуска.

Демон **systemd** также включает в себя возможности по управлению удаленным узлом посредством протокола ssh.

Пример описания сервисного юнита системы регистрации событий представлен на рисунке 3.

```
[Unit]
Description=System Logging Service
Requires=syslog.socket

[Service]
Type=notify
ExecStart=/usr/sbin/rsyslogd -n
StandardOutput=null

[Install]
WantedBy=multi-user.target
Alias=syslog.service
```

Рисунок 3 - Описание сервисного юнита для системы регистрации событий

Для запуска различных процессов по расписанию используется системный демон **cron**. Он позволяет задавать расписание для запуска пользовательских и системных программ. Конфигурационный файл представляет собой набор команд с указанием времени их периодического запуска. Каждая строка имеет вид, показанный на рисунке 4.

```
0 5 * * 1  cd /var/tmp/updater;./run_update.sh
```

Рисунок 4 - Конфигурационный файл демона cron

В начале строки указывается действующее расписание для запуска скрипта, во второй части - строка запуска команды. Расписание имеет вид минуты, часы, день, месяц, день недели. Первые пять столбцов имеют в качестве разделителя пробел, в то время как в команде в качестве

разделителей используются стандартные разделители аргумента. На рисунке 4 расписание интерпретируется как запуск скрипта обновления в 05:00 каждый понедельник.

Для просмотра конфигурационного файла **cron** можно воспользоваться командой **crontab -l**, она покажет расписание для текущего пользователя. Для изменения необходимо запустить команду **crontab** с ключом **e**, в таком случае будет запущен текстовый редактор по умолчанию для внесения изменений. Командные строки обрабатываются с использование командного интерпретатора **sh**, поэтому их вид должен быть соответствующим.

Предназначение демона **cron** обычно связывается с необходимостью запуска периодических задач, необходимых для выполнения обслуживания системы, например: чистка файловой системы, обновление системы, циклическое использование файлов журналов, синхронизация версий файлов, резервное копирование и т.п.

Порядок выполнения работы

1. Написать стартовый сценарий, который запускается последним при переходе на режим выполнения в однопользовательском режиме. Стартовый сценарий обязан поддерживать параметры остановки и запуска.
2. В среде, содержащей систему **systemd**, описать новый тестовый системный юнит, который запускается после монтирования всех файловых систем и сохраняет список смонтированных систем и время в файл журнала.
3. Создать тестовый скрипт и обеспечить его выполнения по расписанию каждую пятницу 2 недели каждого месяца в 01 часов 12 минут.

Требования к оформлению отчета и защите

Отчет должен содержать описание порядка выполнения всех команд и содержание указанных при выполнении работы файлов. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся написанного стартового сценария и установленного расписания.

Лабораторная работа №6. Сетевые средства мониторинга операционной системы Linux.

Цель работы: Получение навыков использования утилит мониторинга сети, сбор статистической информации и представление ее в графическом виде.

Теоретические сведения

Мониторинг за производительностью сетевых средств ОС крайне важен для обеспечения работоспособности целого класса системных и прикладных программ. Мониторинг за сетью в ОС разделяется на три основных направления:

- Мониторинг работоспособности подключения к сети;
- Статистика сетевого взаимодействия;
- Графическое представление графической информации.

Необходимость проверки работоспособности возникает при устранении неполадок при подключении компьютера к сетям и во время его эксплуатации. В этом случае характерно применение утилит, позволяющих диагностировать различного рода проблемы при работе сетевых служб. Утилиты различаются по своему назначению от самых простых до достаточно сложных, способных выявлять комплексные ошибки. В любом случае все эти утилиты предоставляют разностороннюю информацию о текущем состояния подключения к сети, для того чтобы администратор вычислительной системы мог самостоятельно, принимая во внимание различные аспекты выявить и устранить ошибки подключения.

Самой простой утилитой проверки соединения является программа **ping**, рисунок 5. Она служит для проверки качества соединения и достижимости сетевыми пакетами целевого узла. Для работы программа использует запросы протокола ICMP и, получая ICMP ответы, показывает статистику доставки пакетов и скорость непосредственно канала между источником и адресатом.

Обычный размер запроса составляет 64 байта, но может быть изменен при помощи соответствующего ключа.

```
ping 10.1.5.1 -c 6
PING 10.1.5.1 (10.1.5.1) 56(84) bytes of data.
64 bytes from 10.1.5.1: icmp_seq=1 ttl=64 time=0.223 ms
64 bytes from 10.1.5.1: icmp_seq=2 ttl=64 time=0.178 ms
64 bytes from 10.1.5.1: icmp_seq=3 ttl=64 time=0.256 ms
64 bytes from 10.1.5.1: icmp_seq=4 ttl=64 time=0.241 ms
64 bytes from 10.1.5.1: icmp_seq=5 ttl=64 time=0.237 ms
64 bytes from 10.1.5.1: icmp_seq=6 ttl=64 time=0.245 ms

--- 10.1.5.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4998ms
```

```
rtt min/avg/max/mdev = 0.178/0.230/0.256/0.025 ms
```

Рисунок 5 - Определение доступности узла при помощи утилиты **ping**

Второй утилитой проверки работоспособности соединения можно назвать **traceroute**. Особенностью является способность не только определить достижимость по сети целевого узла, но и определить маршрут передачи пакетов. Для отправки запросов утилита может использовать не только ICMP, но и UDP, TCP, GRE. В ходе посылки пакета для определения промежуточных узлов параметр TTL последовательно увеличивается для возможности получения пакета от узла маршрутизатора о том, что время жизни истекло. Этот факт и используется для фиксации очередного узла маршрута следования. Пример трассировки, рисунок 6.

```
traceroute to ya.ru (213.180.193.3), 30 hops max, 60 byte packets
 1  192.168.18.1 (192.168.18.1)  0.331 ms  0.430 ms  0.537 ms
 2  217-79-5-241.obit.ru (217.79.5.241)  2.529 ms  2.520 ms  2.805 ms
 3  vi-xx-1909.kant7-home.255.92.spb.obit.ru (178.16.151.46)  2.486 ms  2.777 ms  2.889 ms
 4  vi-xx-1900.ohr1.spb.obit.ru (178.16.151.104)  2.539 ms  2.626 ms  2.615 ms
 5  vi-xx-1535.shr2.spb.obit.ru (95.161.159.245)  2.822 ms  2.811 ms  2.935 ms
 6  te-0-0-xxx.hnt2.spb.obit.ru (95.161.159.237)  2.921 ms  1.180 ms  2.283 ms
 7  vi-xx-1561.ohr1.spb.obit.ru (95.161.159.240)  2.273 ms  2.369 ms  2.359 ms
 8  vi-xx-0302.brc1.spb.obit.ru (85.114.1.112)  2.469 ms vi-xx-0303.brc1.spb.obit.ru
(85.114.1.114)  2.448 ms  2.442 ms
 9  vi-xx-1692.zs33.0.196.spb.obit.ru (79.142.92.103)  2.522 ms  2.610 ms  2.598 ms
10  diana-spb-ix.yandex.net (194.226.100.90)  16.139 ms  16.128 ms  16.116 ms
11  m9-p2-100ge-2-0-3.yndx.net (213.180.213.16)  16.832 ms  17.268 ms  17.399 ms
12  fol5-c2-ae7.yndx.net (87.250.239.84)  10.286 ms  9.997 ms  9.865 ms
13  * * *
14  www.yandex.ru (213.180.193.3)  12.329 ms  16.766 ms  11.709 ms
15  * * *
16  www.yandex.ru (213.180.193.3)  11.494 ms  11.467 ms  10.995 ms
```

Рисунок 6 - Трассировка маршрута утилитой traceroute

Использование данной утилиты позволяет не только выяснить доступно ли соединение и в случае отсутствия связи выяснить узел, на котором происходит сбой соединения. В гетерогенных сетях, обладающих сложной конфигурацией соединений данная способность часто оказывается востребованной.

Еще одной утилитой, выполняющей трассировку маршрута, является **mtr**, рисунок 7. Ее отличие заключается в более наглядном представлении результатов и определением статистических параметров скорости и надежности доставки не только по всему маршруту, но и по отдельным его участкам.

```
                My traceroute  [v0.85]
kvm (0.0.0.0)
Wed Dec 14 01:55:05 2016
Keys:  Help   Display mode   Restart statistics   Order of fields quit

Packets               Pings Host        Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 192.168.18.1 0.0%    22    0.3   0.3   0.3   0.4   0.0
```

```
 2. 217-79-5-241.obit.ru 0.0%    22    1.1   1.1   0.9   2.5   0.2
 3. vi-xx-1909.kant7-home.255.92.spb.obit.ru 0.0%    21    1.2   6.8   1.0  66.2  17.6
 4. vi-xx-1900.ohr1.spb.obit.ru 0.0%    21    1.0   1.1   0.9   1.9   0.0
 5. vi-xx-1535.shr2.spb.obit.ru 0.0%    21    1.0   1.1   1.0   1.2   0.0
 6. te-0-0-xxx.hnt2.spb.obit.ru 0.0%    21    1.2   1.1   1.0   1.3   0.0
 7. vi-xx-1561.ohr1.spb.obit.ru 0.0%    21    1.2   1.7   1.1  11.8   2.2
 8. vi-xx-0302.brc1.spb.obit.ru 0.0%    21    1.4   1.4   1.2   2.1   0.0
 9. vi-xx-1692.zs33.0.196.spb.obit.ru 0.0%    21    1.5   1.4   1.2   2.2   0.0
10. diana-spb-ix.yandex.net 0.0%    21    1.5   1.5   1.4   1.6   0.0
11. m9-p2-100ge-2-0-3.yndx.net 0.0%    21   11.0  11.8  10.6  15.3   0.8
12. fol5-c2-ae7.yndx.net 0.0%    21   10.0  10.4   9.8  20.4   2.2
13. www.yandex.ru 0.0%    21    9.5   9.6   9.5   9.8   0.0
```

Рисунок 7 – Диагностика сетевого маршрута при помощи утилиты mtr

Статистические показатели сетевых взаимодействий нужны для целостного представления о среде работы системы. Мониторинг объема трафика может быть необходим для выполнения финансовых расчетов, а также для целей информационной безопасности. Прогнозирование и адекватная оценка максимально возможной производительности узлов невозможно реализовать без исторической информации о нагрузке сетевых интерфейсов высоконагруженных сервисов.

Для этих целей можно использовать следующие утилиты: **iptraf**, **dstat**, **iftop**.

Iptraf, являясь консольной программой, имеет интерфейс, который позволяет выполнить мониторинг нагрузки на разных интерфейсах, а также с различными опциями, для обеспечения детального анализа нагрузки на сетевом канале.

Dstat является модульной утилитой для мониторинга за показателями производительности системы, в случае анализа сетевой активности необходимо использовать с ключом **-net**. Утилита может работать в интерактивном режиме и показывать текущие значения в активной консоли. Также поддерживается возможность сохранения в файл для последующего анализа.

Для визуализации информации о загрузке сетевого интерфейса лучше всего подходят следующие утилиты: **mrtg** и **rrdtool**.

Наиболее старой из указанных утилит визуализации является **mrtg**. Утилита позволяет отображать в виде графика информацию не только по загруженности канала, но и другие данные, которые необходимо визуализировать в независимости от их происхождения (использование оперативной памяти, потребление электроэнергии и т.п.), хотя изначально разрабатывалась для мониторинга сетевого трафика.

Для работы с **mrtg** необходимо описать способ представления данных в файле **mrtg.cfg** (рисунок 8) после запуска команды будет сгенерирован html-файл, содержащий данные для отображения веб-браузером.

```
WriteExpires: Yes
Refresh: 300
```

```
WithPeak[^]: wym
Suppress[^]: y
Target[eth0]: `/opt/lab/parse.sh eth0`
WorkDir: /opt/lab/html/mrtg
Options[eth0]: growright
Title[eth0]: eth0 Traffic
PageTop[eth0]:eth0 Traffic
MaxBytes[eth0]:99999999
kilo[eth0]: 1024
YLegend[eth0]: bytes per Second
ShortLegend[eth0]: bytes per Second
LegendO[eth0]:   eth0 In Traffic :
LegendI[eth0]:   eth0 Out Traffic :
```

Рисунок 8 - Пример конфигурационного файла для программы mrtg

Для работы необходимо указать либо community и сервер SNMP для получения данных, либо внешнюю программу. Статистику сетевого взаимодействия можно получить при помощи тривиального скрипта, показанного на рисунке 9, который возвращает текущий счетчик байт, переданных через заданный сетевой интерфейс.

```
#!/bin/bash
grep "$1" /proc/net/dev | awk -F ":" '{print $2}' | awk '{print $1 " \n" $9 ;}'
```

Рисунок 9 - Скрипт для получения статистики с сетевого интерфейса

Для периодического обновления графика можно задействовать демон **cron**, указав строку для запуска программы **mrtg**, рисунок 10.

```
0-59/5 * * * * env LANG=C /usr/bin/mrtg /opt/lab/mrtg.cfg
```

Рисунок 10 - Запись в конфигурации программы cron для периодического обновления данных

Пример результата изображен на рисунке 11.

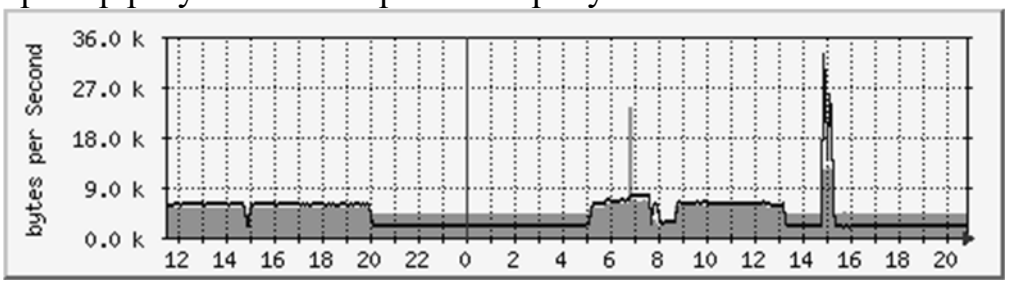

Рисунок 11 - Пример полученного программой mrtg графика

Усовершенствованным инструментом визуализации сетевого взаимодействия является **rrdtool**.

Набор утилит предназначен для сбора, хранения и визуализации данных, но с акцентом на их изменяемость во времени. Для учета этой особенности используется кольцевая база данных RRD, размер которой не меняется с момента создания, а данные при их поступлении агрегируются на основе временного промежутка.

Для реализации простейшего сценария мониторинга сетевого трафика нужно выполнить последовательность действий. Сначала создать кольцевую базу данных при помощи утилиты **rrdtool create**, рисунок 12.

```
rrdtool create eth0.rrd --step 300 DS:input:COUNTER:600:U:U
DS:output:COUNTER:600:U:U RRA:AVERAGE:0.5:1:576 RRA:MAX:0.5:1:576
RRA:AVERAGE:0.5:6:672 RRA:MAX:0.5:6:672 RRA:AVERAGE:0.5:24:732
RRA:MAX:0.5:24:732 RRA:AVERAGE:0.5:144:1460 RRA:MAX:0.5:144:1460
```

Рисунок 12 - Пример команды создания кольцевой базы данных rrd

После этого созданную базу необходимо периодически обновлять соответствующими счетчиками командой **rrdtool update** (рисунок 13), удобнее всего это реализовать с задействованием демона запуска периодических заданий **cron**.

```
#!/bin/bash
INPUT=`/sbin/ifconfig $1 |grep bytes|cut -d":" -f2|cut -d" " -f1`
OUTPUT=`/sbin/ifconfig $1 |grep bytes|cut -d":" -f3|cut -d" " -f1`
rrdtool update /var/tmp/rrdtool/eth0.rrd -t "input:output" N:$INPUT:$OUTPUT
```

Рисунок 13 - Пример команды обновления кольцевой базы данных rrd

В завершении можно воспользоваться командой **rrdtool graph** (рисунок 14), которая сгенерирует график на основе имеющейся в базе информации.

```
rrdtool graph net.png -v bytes/sec --slope-mode --imgformat PNG
DEF:input=eth0.rrd:input:AVERAGE DEF:output=eth0.rrd:output:AVERAGE
CDEF:output_neg=output,-1,* AREA:input#32CD32:"In " "GPRINT:input:MAX: Max\\: %6.1lf
%s" "GPRINT:input:AVERAGE:Average\\: %6.1lf %S" "GPRINT:input:LAST:Current\\: %6.1lf
%S\\n" HRULE:0#000000 AREA:output_neg#0033CC:"Out" "GPRINT:output:MAX: Max\\:
%6.1lf %S" "GPRINT:output:AVERAGE:Average\\: %6.1lf %S"
"GPRINT:output:LAST:Current\\: %6.1lf %S\\n"
```

Рисунок 14 - Пример команды генерации графика на основе базы данных rrd

В результате получается график вида - рисунок 15

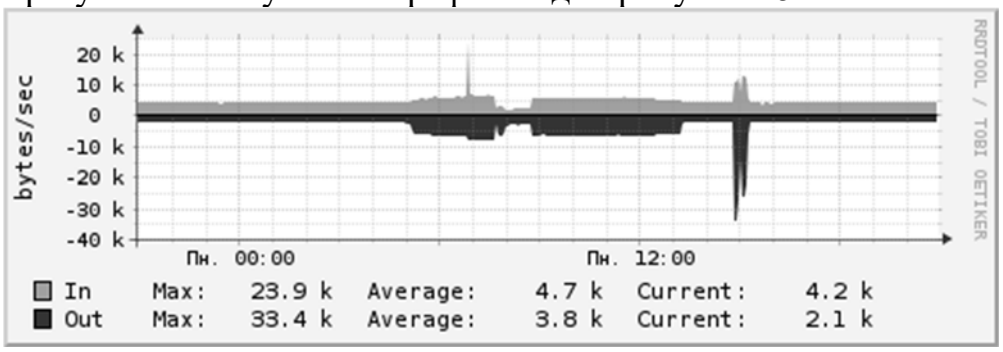

Рисунок 15 - График, сформированный командой rrdtool с параметром graph

Порядок выполнения работы

1. Написать скрипт для сбора статистики с интерфейса. Обеспечить его постоянной загрузкой активностью: скачивание файла, открытие веб-страниц и т.п.
2. Обеспечить сбор данных для формирования графика активности с использованием утилиты mrtg.
3. Сформировать график сетевой активности.
4. Создать кольцевую БД rrd и обеспечить ее постоянное обновление данными сетевой активности.
5. Сгенерировать график сетевой активности за заданный промежуток времени (на одном графике должен быть и входящий и исходящий трафик).
6. Проверить работоспособность при обращении к некоторым общедоступным веб-ресурсам при помощи утилит трассировки маршрута. Наложить путь на географическую карту. В качестве веб-ресурса можно использовать сайты google.com, kernel.org, ubuntu.com

Требования к оформлению отчета и защите

Отчет должен содержать описание всех команд и скриншоты полученных графических представлений. При защите отчета, исполнитель должен быть готов ответить на вопросы по ходу выполнения лабораторной работы.

Лабораторная работа №7. Файловые системы операционных систем.

Цель работы: Получение навыков управления различными файловыми системами в среде операционной системы GNU Linux.

Теоретические сведения

Файловая система является элементом для организации хранения данных, необходимых для функционирования ОС, а также данных прикладных программ и пользователей. Структурированное хранение данных жизненно важно для эффективного поиска, получения и сохранение информации на энергонезависимых носителях информации. Файловая система включает в себя структуры для организации хранения данных, а также программное обеспечения для организации доступа к ним. Обычно все данные хранятся в логически разделенных порциях – файлах, которые по сути являются меткой, которая присваивается порции пользователем либо процессом. Директории являются специальными файлами, которые точно также объединяют назначением меткой совокупность файлов, это и является именем директории. Лучше всего, когда метка имеет какое-то смысловое значения для возможности быстрого поиска пользователем интересующей его информации. Организация файлов и директорий выполняется в виде иерархической структуры, началом которой является корневая директория **/**, от которой берут начало все остальные директории. В операционных системах Windows корневая директория также существует, но является скрытой и от нее непосредственно берут начало логические диски. Работа с иерархией позволяет расширять логическую структуру файловой системы подсоединением других файловых систем в точках монтирования, не изменяя текущую организацию файловой системы. Процесс работы с данными в файловой системе организуется при помощи системных вызовов, которые передаются в ядро для выполнения и взаимодействуют непосредственно со структурами файловой системы для нахождения необходимых блоков данных, а затем производят операции на устройстве хранения. Для прикладных процессов взаимодействие складывается из высокоуровневых операций: открытие файла, позиционирования в файле, чтения данных из файла, запись в файл, закрытие файла, создание директории, удаление директории и т.п.

Операционная система Linux поддерживает широкий спектр файловых систем. Каждая из них обладает своими преимуществами и недостатками, которые определяют их использование в каждом конкретном случае. Способ управления файловыми системами одинаков вне зависимости от ее типа.

Для поддержки файловой системы ОС должна содержать программный код для работы с ней.

К числу поддерживаемых файловых систем в операционных системах GNU Linux относятся:

- Ext2
- Ext3
- Ext4
- Xfs
- Btrfs
- Zfs
- Reiserfs
- Fat32

Ext2 – файловая система впервые представленная в 1992 году, основными особенностями является использование индексных дескрипторов для работы с файлами, включая хранение метаинформации и адресации блоков данных. Блоки объединены в группы блоков для обеспечения близости хранения связанных данных.

Ext3 – файловая система, устранившая основной недостаток ext2, – отсутствие возможности восстановления после сбоев. Ext3 дополнена возможностью ведения журналов операций, которые могут быть задействованы при сбое, что значительно повысило надежность хранения данных. Также многие лимиты хранения структур файловой системы были расширены по сравнению с ext2.

Ext4 – следующая версия файловой системы ext3. Основные улучшения направлены на повышение производительности работы. Такие как введение экстентов - последовательных записанных блоков данных для эффективного управления большими файлами, в этом случае нет необходимости хранения адресной информации для каждого блока, только начало экстента и его длину в блоках. К другими улучшениям относятся еще большее увеличение лимитов для структур файловой системы, например, максимальный размер файла составляет 16 Тб, снятие ограничения на количество подкаталогов (в ext3 32000). Отложенное распределение блоков позволяет повысить производительность путем последовательной записи большого числа кэшированных данных, уменьшая тем самым фрагментацию данных.

XFS – журналируемая файловая система, разработанная специально для ОС IRIX, позднее была опубликована под лицензией GNU GPL, вследствие чего получила распространение в среди Linux дистрибутивов. Изначально XFS проектировалась для использования в дисках большого размера поэтому используется 64 битная адресация. Принципы работы сходны с уже описанной ext4, к основным из которых относится использование индексных дескрипторов, учет пространства экстентами, широкое использование бинарных деревьев для хранения структур файловой системы. В качестве основного недостатка эксперты выделяют слишком активное использование кеширования, что может приводить к

потерям большого числа активных, но не законченных операций при сбоях электропитания.

Btrfs – файловая система разрабатываемая компанией Oracle. Принцип работы заключается в активном использовании Б-деревьев и механизма копирования при записи. Обладает рядом преимуществ по сравнению с уже рассмотренными файловыми системами. Среди них выделяется поддержка на уровне файловой системы программной реализации RAID, создание моментальных снимков, возможности по клонированию, сжатие данных.

Zfs – файловая система первоначально разработанная и используемая в операционной системе Solaris. Является проектом с открытым исходным кодом. К основным особенностям можно отнести совмещенность файловой системы и менеджера томов. Является 128 битной файловой системой, что дает возможность значительно расширить лимиты внутренних структур, таких как: размер файловой системы, количество файлов, размер одного файла, размер атрибута и т.п. При работе используются пулы хранения, которые позволяют объединять несколько физических устройств в один том и уже на его основе создавать файловую систему. Zfs поддерживает свой собственных вариант построения отказоустойчивого хранилища на основе технологии RAIDZ с возможностью горячего резерва и двойной, а в последних версиях тройной, четности. Имеется возможность выделения отдельных физических устройств для выполнения кеширования, что актуально в настоящее время в связи с широким развитием твердотельных накопителей. Широкое распространение в ZFS имеет технология создания моментальных снимков и клонов файловой системы. Благодаря использованию модели транзакций на основе копирования при записи, снимки и клоны создаются практически мгновенно и ими удобно пользоваться для целей резервного копирования и управления файловыми системами. Сквозной контроль целостности данных позволяет определять корректность данных каждый раз при их считывании. Использование механизма дедупликации позволяет на уровне файловой системы определять дублирующиеся данные и хранить только один их экземпляр, сохраняя тем самым дисковое пространство.

ReiserFS

Была первой журналируемой файловой система, поддержка которой включена в ядро ОС Linux. Основными особенностями является возможность изменения размера файловой системе без необходимости ее размонтирования. Обладает склонностью к фрагментации, при этом существует способы дефрагментации, но они крайне неэффективны. В данный момент считается устаревшей файловой системой даже при условии разработки новой версии, имеющей название Reiser4. В тоже время имеет достаточно высокую производительность операций над файлами малого размера.

Fat32

Файловая система, имеющая широкое распространение в среде Windows. Является дальнейшим развитием файловой системы Fat, которую усовершенствовали для увеличения размера тома путем смены разрядности адресации блоков данных (в терминах Fat - кластеров) с 16 до 32. При этом все остальные принципы работы остались неизменными для обратной совместимости. Раздел при форматировании его под файловую систему fat32 разделяется на две области: системную и пространство для хранения данных. Системная область содержит структуры, необходимые для управления размещением данных файлов и каталогов. Также здесь содержится корневой каталог и загрузочный сектор для загрузки ОС. Кластера файла содержатся в пространстве для хранения данных. Информация о занятых файлом кластерах содержится в таблице размещения файлов (File Allocation Table - FAT) благодаря этой структуре файловая система и получила свое название. Каждому кластеру пространства данных соответствует запись в FAT. Для адресации всего набора кластеров файла используется цепочка, в которой в записи соответствующей занятому кластеру содержится адрес следующего кластера файла, в случае свободного кластера содержится нулевое значение. Таблица FAT содержится в двух экземплярах вследствие особого значения для работы файловой системы, ввиду интенсивного использования она также загружается в оперативную память и хранится там максимально долго. Основным недостатком является достаточно медленная работы с файлами из-за выделения первого свободного кластера, что приводит к созданию сложных цепочек кластеров и фрагментации. Также отмечается низкая отказоустойчивость, так как журнала файловая система не ведет.

NTFS

Современная файловая система, распространенная в среде Windows. Спецификация файловой системы является закрытой, что затруднило создание программного кода для доступа для других ОС. Тем не менее методом обратной разработки специалистами написаны драйвера для доступа из других операционных систем. Пришла на смену файловой системе Fat32. Имеет следующие улучшения: хранение метаданных, журналирование, списки контроля доступа, квотирование. Основной структурой для хранения информации о файлах является MFT (Master File Table). Под хранение этой таблице выделяется целая область для предотвращения ее фрагментирования в процессе работы файловой системы. Каждая запись MFT имеет фиксированный размер и включает в себя информацию в виде имени файла, размера, набора атрибутов, а также положения на диске отдельных фрагментов. Если для хранения информации о файле недостаточно одной записи MFT, то может задействоваться несколько записей, которые могут располагаться непоследовательно. Атрибуты могут быть резидентными, то есть храниться целиков в записи MFT, либо нерезидентными хранящимися за

пределами области MFT. Для адресации кластеров возможно использование 64 разрядных идентификаторов, но в текущих версиях используются только 32.

Порядок выполнения работы

1. Создать в виртуальной машине неразмеченный диск, который затем использовать для создания различных файловых систем.
2. Создать раздел и на нем файловую систему ext2.
3. Создать раздел и на нем файловую систему ext3.
4. Создать раздел и на нем файловую систему ext4.
5. Создать раздел и на нем файловую систему xfs.
6. Создать раздел и на нем файловую систему btrfs.
7. Создать раздел и на нем файловую систему zfs.
8. Создать раздел и на нем файловую систему reisers.
9. Создать раздел и на нем файловую систему fat32.
10. Создать раздел и на нем файловую систему ntfs
11. Активировать опцию журналирования для файловых систем, которые ее поддерживают.
12. Настроить ОС, чтобы все файловые системы монтировались при старте системы.
13. Создать раздел swap и активировать его в системе.
14. Создать файловую систему типа swap в файле и активировать в системе.
15. Протестировать все разделы для операций чтения маленьких файлов (размер 16Кб), цикл тестирования не менее 100 раз.
16. Протестировать все файловые системы для операций записи больших файлов (500Мб).
17. Протестировать операции записи для маленьких файлов.
18. Протестировать операции записи для больших файлов.
19. Создать структуру каталогов с не менее чем 1000 подкаталогов в каждой файловой системе, замерить время создания.
20. Замерить время поиска по созданной структуре для каждой файловой системы.
21. Представить результаты каждого эксперимента по замеру производительности операции в файловой системе в графическом виде.

Требования к оформлению отчета и защите

Отчет должен содержать все выполненные команды по заданию и вывод при их исполнении. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся выполненных действий с файловыми системами.

Лабораторная работа №8. Менеджер томов в GNU Linux и программный RAID.

Цель работы: Получение навыков использования менеджера томов для управления устройствами ввода-вывода и дисковым пространством. Знакомство с программным RAID в среде Linux.

Теоретические сведения

В операционной системе Linux для эффективного управления хранением данных имеется инструмент под названием LVM (Logical Volume Manager). Данное программное обеспечение позволяет управлять томами, созданными поверх физических устройств ввода-вывода. Такие тома воспринимаются в системе как обычные блочные устройства с возможностью создания файловых систем и т.п. LVM позволяет добавлять гибкость в использование физических устройств ввода-вывода: изменять размер логического тома, зеркалировать тома, создавать моментальные снимки. Моментальные снимки мощный инструмент для выполнения функций резервного копирования на уровне устройств, а не на уровне файловой системы.

На нижнем уровне структуры LVM находится физический том pv (Physical volume), который представляет собой либо отдельное физическое устройство хранения, например, накопитель на жестких магнитных дисках, либо устройство аппаратного RAID контроллера. Чуть выше по иерархии находится группа томов vg (Volume Group), которая включает в себя один или несколько физических томов. Таким образом группа томов представляет собой логическую единицу, которая объединяет несколько физических накопителей. При этом в группу томов можно добавить в последствии новые физические тома, тем самым расширяя пространство группы томов. На базе группы томов создаются логические тома lv (Logical Group), являющиеся эквивалентом разделов в системе без использования LVM. На базе логических томов уже непосредственно создаются файловые системы.

Для работы с физическими томами применяются утилиты:

pvcreate – инициализация физического устройства как физического тома LVM.

pvdisplay – отображение справочной информации о состоянии и использовании физического тома LVM.

pvscan – сканирование физических устройств на наличие физических томов LVM.

pvmov – перемещение данных с одного физического тома на другой.

Для работы с группами томов предназначены следующие утилиты:

vgcreate – создание группы томов.

vgchange – активация и деактивация LVM в системе.

vgremove – удаление группы томов. Перед удалением удаляемую группу томов нужно деактивировать командой **vgchange – a n VG_NAME**.

vgextend - добавление в группу томов еще одного физического тома с увеличением емкости группы томов.

vgreduce – удаление физического тома из группы томов.

vgexport – подготовка группы томов к экспортированию на другую систему.

vgimport – импортирование группы томов с другой системы.

vgscan – сканирование дисков для нахождения групп томов и их активация в системе.

vgdisplay – отображение справочной информации по указанной группе томов.

lvcreate – создание логического тома. При указании опции -s создаваемый логический том будет представлять собой моментальный снимок другого логического тома.

lvremove – удаление логического тома. Перед удалением он должен быть размонтирован.

lvextend – увеличение размера логического тома. Размер файловой системы на нем также должен быть увеличен соответствующими утилитами файловых систем. Например, для ext3/ext4 – resize2fs.

lvreduce – уменьшение размера логического тома. Перед выполнением операции нужно уменьшить размер файловой системы.

Для создания RAID уровней 1,0,4,5 и 6 в ОС Linux можно воспользоваться набором утилит **mdadm**, которые дают возможность на программном уровне реализовать массив заданного типа. Все основные действия, такие как создание массива заданного типа, добавление устройств, удаление устройств, осуществляются при помощи утилиты **mdadm**. Для проверки статуса устройств в массиве, его состояния и режима работы необходимо открыть файл в виртуальной файловой системе **/proc/mdstat**.

Порядок выполнения работы

1. Создать физические тома на существующих устройствах ввода-вывода.
2. Создать группу томов.
3. Добавить в группу несколько логических томов различного размера.
4. Создать файловую систему на логическом томе.
5. Добавить еще один физический том в группу томов.
6. Расширить существующий логический том за счёт добавленного физического тома.
7. Расширить файловую систему на логическом томе.
8. Создать моментальный снимок логического тома.

9. Выполнить резервную копию логического тома при помощи моментального снимка.

10. Создать программный RAID массив 0 типа из двух логических томов, создать на полученном устройстве файловую систему.

11. Создать программный RAID массив 1 типа из двух логических устройств. Создать файловую систему, проверить статус массива.

12. Обеспечить автоматическое монтирование массивов при старте системы.

Требования к оформлению отчета и защите

Отчет должен содержать описание порядка выполнения всех команд и вывод, полученный при их исполнении. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся выполненных действий с логическими томами и утилитами для работы с RAID массивами.

Лабораторная работа №9. Обеспечение безопасности в среде операционной системы GNU Linux.

Цель работы: Получение навыков конфигурирования параметров безопасности ОС Linux.

Теоретические сведения

В операционной системе Linux для выполнения части системных задач необходимо обладать привилегиями суперпользователя. Для выделения права на запуск определенных команд обычным пользователям применяется файл **sudoers** с описанием команд, доступных с командой **sudo**. С ее помощью можно определить команды, требующие повышенных полномочий обычным пользователям без установки битов SUID. Обычной практикой является использование даже для выполнения действий, связанных с управлением ОС, обычной, ограниченной учетной записи, с задействованием команды **sudo** для команд, требующих повышенных привилегий. Такое поведение позволяет минимизировать риск взлома системы, используя пользовательское окружение и уязвимости в пользовательском программном обеспечении. В случае работы с использованием прав суперпользователя недостатки, например, в браузере могут привести к получению злоумышленником прав на атакуемой системе.

Парольная защита загрузчика ОС. В ОС Linux стандартным загрузчиком является **grub**, обладающий широкими возможностями по управлению загрузкой ОС. Одним из способов повышения безопасности является установка пароля на загрузку ОС. Таким образом затрудняются запуск системы даже при условии физического доступа злоумышленника к системе и его возможности по извлечению системного диска и подключению к своему оборудованию для старта системы.

SELinux – система гибкого принудительного контроля доступа, призванная расширить возможности ОС по управлению доступом к ресурсам. При помощи специальных политик происходит определение возможностей по доступу к ресурсам со стороны пользователей и процессов. Классическая модель разграничения доступа, существующая в Linux, основана на избирательном доступе и списках контроля доступа. Основным ее недостатком является возможность самостоятельного назначения прав ресурсам пользователями, владельцами которых они являются, и отсутствие механизма назначения минимально возможных привилегий для пользователей в соответствии с принципом минимизации прав. Это может приводить к тому что пользователи могут назначать некорректные права на свои данные так, что к ним могут получить доступ другие пользователи, а также в случае проникновения в систему через приложения злоумышленник может изменить права на данные пользователя. В рамках системы SELinux выделяются следующие понятия:

Домен – минимально разрешенные процессу действия в системе.

Роль – набор доменов. По сути разные комбинации доменов определяют разные по разрешенному функционалу типы выполняющихся процессов.

Тип – тоже самое что домен, но имеющий отношение к статическим ресурсам, файлам, директориям.

Контекст безопасности - совокупность атрибутов, при которых происходит выполнение процесса.

Политика безопасности – набор правил, которые определяют допустимые и запрещенные действия субъектов, оперируя терминами ролей, доменов и типов.

SELinux характеризуется режимом работы, который определяет его влияние на работу системы. Режим Enforcing означает что все действия, которые будут нарушать действующую политику безопасности, будут блокироваться, и информация о них регистрироваться в журнале. Permissive режим предполагает регистрацию нарушений политики безопасности без непосредственной блокировки. Этот режим удобен на этапе настройки системы для определения перечня необходимых для функционирования сервиса действий в системе. Disabled режим полностью отключает систему SELinux.

Важным аспектом обеспечения безопасности является аудит журнала событий на предмет подозрительной активности. Лучше всего такие действия выполнять на периодической основе в автоматизированном режиме. Особое значение имеет возможность отсылки сообщения, так как данные при проникновении и получение злоумышленником прав суперпользователя могут быть стерты из журнала. Часть задач по анализу журналов безопасности может выполнять утилита **logwatch** с опцией отсылки уведомлений по какому-либо каналу коммуникации.

Для удаленного доступа к Linux узлу обычно используется программа ssh, обеспечивающая защищенный обмен данными. Ее возможности позволяют организовать взаимодействие различными способами. К основным преимуществам можно отнести аутентификацию не только по паролю, но и по ключевой паре, которая должна быть предварительно сформирована и распространена на узлы для последующей аутентификации. Другим важным аспектом является возможность организации сетевых туннелей для инкапсуляции взаимодействия по любому протоколу, основанному на транспорте TCP, внутри SSH соединения.

Порядок выполнения работы

1. Определить для обычного пользователя возможность для запуска команды **tcpdump** через команду **sudo**.
2. Установить пароль на загрузчик операционной системы.
3. Отредактировать существующую политику для SELinux с сервисом **Samba** таким образом, чтобы можно было настроить работу с

разделяемым ресурсом, находящимся в произвольном месте файловой системы.

4. Настроить программу **logwatch** на отсылку оповещений по почте о неудачных попытках входа в систему.

5. Настроить ограничения для работы программы ssh путем редактирования файла конфигурации. Запретить удаленный доступ к системе суперпользователю, изменить порт для подключения с 22 на иной (например 6622),

6. Настроить аутентификацию программы SSH по ключевой паре вместо паролей.

Требования к оформлению отчета и защите

Отчет должен содержать описание порядка выполнения всех команд и вывод, полученный при их исполнении. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся выполненных настроек для повышения безопасности ОС Linux.

Лабораторная работа №10. Виртуализация операционных систем в среде GNU Linux.

Цель работы: Получение навыков использования систем виртуализации в среде ОС Linux.

Теоретические сведения

В операционной системе Linux доступно для использования несколько технологий виртуализации различного уровня: уровня гипервизора (Xen, KVM), уровня приложения (VirtualBox, VMware), на уровне операционной системы (LXC). В зависимости от задачи отдается предпочтение соответствующей технологии виртуализации. Наибольшую популярность в данный момент имеет технология, встроенная в ядро ОС Linux - kvm. Функционал обеспечивается при помощи модуля ядра kvm (kvm-amd.ko и kvm-intel.ko в зависимости от типа процессора). Виртуальная машина реализована как Linux-процесс и подчиняется стандартным способам изоляции и квотирования ресурсов (на основе cgroups). Процесс взаимодействует с ядром операционной системы посредством интерфейса /dev/kvm для управления виртуальной машиной. Для эмуляции устройств используется адаптированная версия qemu для обеспечения возможности работать с базовым набором устройств.

Для удобного взаимодействия с гостевыми виртуальными машинами, их созданием и удалением существует набор утилит **virsh**.

Учитывая богатый спектр возможностей по динамическому управлению виртуальными машинами, а также высокие показатели производительности при использовании технологии KVM, в настоящий момент ее применение широко представлено в отрасли.

Порядок выполнения работы

1. Ознакомиться с утилитами управления виртуальными машинами virsh.
2. Создать виртуальную машину Linux.
3. Создать виртуальную машину Windows.
4. Настроить сетевой мост для сетевого интерфейса гостевой ОС для прямой работы с сетью.
5. Создать виртуальную машину с хранением образа жесткого диска на логическом томе.
6. Активировать технологию memory ballooning и уменьшить количество потребляемой памяти гостевых ОС Linux и Windows, не выключая их.

Требования к оформлению отчета и защите

Отчет должен содержать описание порядка выполнения всех команд и вывод, полученный при их исполнении. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся выполненных действий с виртуальными машинами.

Лабораторная работа №11. Написание Shell скриптов

Цель работы: Получение навыков написания простейших shell скриптов для автоматизации выполнения системных и прикладных задач.

Теоретические сведения

Shell скрипт - это программа, написанная на скриптовом языке, которая может быть интерпретирована командной оболочкой операционной системы. В зависимости от дистрибутива Linux системы такими интерпретаторами могут быть bash, sh, ksh, zsh и др. Скрипты существуют не только для данного вида интерпретаторов, также возможно их написание для интерпретаторов языков Perl и Python. В операционных системах Unix для выполнения системных и прикладных задач используют скрипты, написанные на всех трех указанных скриптовых языках. Основной целью написания скриптов является автоматизация рутинных операций в операционных системах с использованием командной строки. Скрипты позволяют при помощи конструкций условий, циклов и объявления переменных решать задачи, которые администратор может запрограммировать, тем самым значительно упростить себе работу по управлению ОС.

Для ознакомления с синтаксисом каждого из языков необходимо воспользоваться официальной документацией, ссылку на которую можно найти в конце пособия в списке литературы.

Основные сведения, необходимые для написания shell скриптов:

- Для запуска написанного скрипта он должен обладать правами выполнения в ОС.
- Для написания скрипта могут использоваться встроенные языковые конструкции shell интерпретатора, а также внешние программы.
- Начало скрипта должно содержать информацию о том, в каком интерпретаторе будет выполняться скрипт, например, #!/bin/bash.

Порядок выполнения работы

В соответствии с заданием написать скрипт и убедиться в корректности его работы.

Список вариантов:

1. Написать скрипт, осуществляющий поиск в заданной директории файлов log и собирающий информацию об ошибках подключения по ssh (обычно файл syslog.log).
2. Написать скрипт, сравнивающий текущие открытые порты с заданным списком и выдающий оповещение на все консоли при открытии

нового порта. Заданный список формируется при первом запуске, затем с ним сравнивается.

3. Написать скрипт, который проверяет работоспособность некого узла по нескольким сетевым портам и в случае невозможности получения ответа отправляет сообщение на все консоли сервера.

4. Написать скрипт вывода статистики по смонтированным системам по их общему объему и заполнению, количеству директорий в них и файлов, вывод названия самого большого файла.

5. Написать скрипт, определяющий среднюю загрузку узла и при превышении заданного порога выдающий сообщение в системный журнал.

6. Написать скрипт, выполняющий архивирование каждого файла в директории, если размер каталога превышает заданную величину.

7. Написать скрипт, который выводит список доступных версий ядра для загрузки ОС, а также версию в настоящий момент загруженного ядра. Информация доступна в конфигурационном файле загрузчика ОС.

8. Написать скрипт проверки корректности настройки DNS при помощи разрешения какого-либо символьного имени в цифровой и вывода результата в виде успех/не успех.

9. Написать скрипт для вывода списка модулей ядра, доступных для загрузки, а также в настоящий момент уже загруженных.

10. Написать скрипт, осуществляющий проверку содержания web-ресурса определенной строки в выводе на запрос GET по протоколу http.

Требования к оформлению отчета и защите

Отчет должен содержать описание порядка выполнения всех команд и содержание указанных при выполнении работы файлов. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся написанного shell скрипта.

Лабораторная работа №12. Утилиты мониторинга производительности в среде операционный системы GNU Linux

Цель работы: Получение практических навыков работы с утилитами диагностирования состояния памяти и выполнения процессов в операционной системе Linux.

Теоретические сведения

Обзор основных утилит для работы с памятью в операционной системе Linux:

free

Утилита free показывает общее количество свободной и используемой физической памяти. В зависимости от параметров вывод может быть представлен в байтах, килобайтах, мегабайтах, гигабайтах и терабайтах. Возможно обеспечение вывода с обновлением через заданное количество в несколько секунд. Одним из преимуществ является вывод с учетом и без учета буфера и кэша, это позволяет сразу же узнать доступное количество памяти для запускаемого приложения с учетом того, что память в буферах и кэше будет при необходимости освобождена операционной системой.

vmstat

Программа, показывающая статистическую информацию о процессах, памяти, активности процессоров и т.д. Особенностью является структурированный вывод в одной таблице указанных данных работы системы. Также, как и **free**, поддерживает вывод информации с определенным интервалом, что обеспечивает возможность непрерывного мониторинга за показателями операционной системы.

top

Программа предоставляет динамический вывод информации о системе, о текущих активных задачах и их параметрах. Вывод представлен в виде таблицы, интерактивное управление позволяет онлайн менять порядок сортировки и выводимую информацию о процессах.

htop

Аналогичная по функционалу программе **top**, но вывод дополнен блоком загрузки отдельно по ядрам, памяти и разделу подкачки с градацией по цветам от 100% потребления ресурса.

ps

Программа предоставляет информацию об активных в данный момент процессах. Обладает широким спектром параметров для отображения различных характеристик запущенных процессов.

dstat

Также является утилитой для мониторинга работы системы с расширенным набором доступных счетчиков. Позволяет сочетать сбор различных характеристик в зависимости от целей наблюдения. В том

числе обладает возможностью подключения плагинов для обработки своих собственных счетчиков.

iostat

Утилита для мониторинга производительности устройств ввода-вывода. Поддерживается периодический вывод информации с округлением среднего за заданный промежуток времени. Стандартный вывод содержит информацию о количестве запросов в секунду (tps), средних пропускных способностях устройств за промежуток времени, прочитанном и записанном количестве данных в килобайтах.

Описание установки для проведения практической работы

Установка для выполнения практической работы представляет собой рабочую станцию, функционирующую под управлением ОС MS Windows 7. На каждой рабочей станции установлен пакет VirtualBox, в котором запущена операционная система Linux.

Порядок выполнения работы

1. Запуск установки.

1.1 Войти в систему. Запустить в среде VirtualBox операционную систему Linux.

1.2 Выполнить вход в операционную систему Linux.

2. Изучение предложенных утилит.

2.1 Войти в систему, используя учетную запись с правами суперпользователя.

2.2 Запустить поочередно каждую из утилит, ознакомиться со справочной документацией, имеющейся для каждой утилиты.

2.3 Сделать скриншоты запуска утилит с выводом максимально возможного количества информации о параметрах, связанных с использованием памяти в системе.

2.4 В соответствии с документацией определить, что обозначают все значения параметров, предоставляемых в выводе утилит.

2.5 Подготовить ответы на вопросы относительно каждого параметра для защиты отчета по данной лабораторной работе.

Требования к оформлению отчета и защите

Отчет предоставляется в электронном или бумажном виде, содержит полученные в ходе выполнения лабораторной работы результаты.

Лабораторная работа №13. Работа с системой учета и регистрации событий операционной системы GNU Linux

Цель работы: получение навыков по самостоятельному конфигурированию подсистемы регистрации и учета событий в операционной системе Linux.

Теоретические сведения

В процессе функционирования операционной системы различные выполняющиеся программы генерируют сообщения, имеющие общесистемный характер. К ним относятся оповещения об успешном запуске или завершении какой-либо задачи, журналирование нормального хода выполнения программы, а также различные сообщения об ошибках. Отдельно можно выделить информацию, связанную с обеспечением безопасности при работе операционной системы. Данные о регистрации пользователей, смены паролей, выполнении доступа к ресурсам ОС могут понадобиться при последующем расследовании в случае возникновения инцидента информационной безопасности. Гораздо эффективней и удобней использовать единый в рамках операционной системы способ сбора и управления событиями выполняющихся программ. Это позволяет хранить информацию централизованно и применять различные методы обработки данных в журналах для оперативного выявления неисправностей и предоставлять быстрый доступ к журналам пользователям, а также управлять пространством для хранения. Следствием группировки является нахождение всех файлов в одной директории либо наборе поддиректорий, где пользователь легко может найти журнал интересующей его программы. Управление пространством подразумевает возможность сокращения занимаемого дискового места файлами журналами. Для этого периодически происходит удаление устаревших журналов, либо их перезапись по истечении времени хранения.

Наиболее популярной системой регистрации событий является программа **syslog**. Она позволяет определить правила, по которым данные от программ попадают по различным назначениям, которыми могут являться журнальные файлы, пользовательские терминалы и другие узлы с установленной версии **syslog**. Принятие решения осуществляется в соответствии с источником сообщения и степени его важности. Одной из отличительных черт является использование этой системы для событий ядра и системных утилит, поэтому запуск системного демона **syslog** происходит сразу после загрузки ядра для возможности протоколирования процесса загрузки.

При работе с **syslog** любой программой производится вызов функции syslog(), которая в свою очередь производит запись в файл специального назначения **/dev/log**, из которого демон осуществляет чтение и обработку в соответствии со своим конфигурационным файлом.

В настоящее время широкое распространения получили улучшенные версии **syslog**, а именно **rsyslog** и **syslog-ng**. Из их возможностей можно отметить способность передачи сообщений по сетевому протоколу TCP вместо UDP, поддержка SSL шифрования, использование в качестве хранилищ сообщений базы данных.

Рассмотрим используемую конфигурацию демоном **rsyslog**. Как и большинство в данный момент используемых конфигураций системных демонов операционной системы Linux rsyslog использует разделение одного единственного файла с конфигурацией на подгружаемые файлы для структурированного хранения. Основной файл находится по пути **/etc/rsyslog.conf**, а подгружаемые файлы по пути **/etc/rsyslog.d/*.conf**. Фрагмент файла с настройками по умолчанию представлен на рисунке 16 ниже:

```
auth,authpriv.*              /var/log/auth.log
*.*;auth,authpriv.none       -/var/log/syslog
#cron.*                      /var/log/cron.log
#daemon.*                    -/var/log/daemon.log
kern.*                       -/var/log/kern.log
#lpr.*                       -/var/log/lpr.log
mail.*                       -/var/log/mail.log
#user.*                      -/var/log/user.log
```

Рисунок 16 - Конфигурационный файл системы регистрации событий

Основная часть конфигурационного файла направлена на описание источников и назначений для хранения файлов с указанием минимальной важности.

Для хранения данных в базе данных необходимо подключить соответствующий модуль, например, для базы данных PostgreSQL, рисунок 17.

```
$ModLoad ompgsql
```

Рисунок 17 - Параметр для хранения сообщений в базе данные PostgreSQL

Для сохранения сообщений в базе данных необходимо указать строку по примеру, изображенному на рисунке 18.

```
mail.*      :ommysql:127.0.0.1,syslog,syslogwriter,topsecret
```

Рисунок 18 - Указание назначения для хранения в базе данных MySQL

Для пересылки сообщений на другой узел сначала нужно активировать прием сообщений на целевом узле **rsyslog** (рисунок 19).

```
$ModLoad imtcp
$InputTCPServerRun 514
```

Рисунок 19 – Параметр для приема событий по сети

Для отправки используйте строку, по примеру рисунка 20

```
*.*   @@192.0.2.1:514
```

Рисунок 20 - Указание назначения на удаленном сервере rsyslog

Для проверки работы системы регистрации событий обычно используется утилита **logger**, которая позволяет, используя стандартный способ доставки сообщения, проверять различные источники их поступления. Пример использования утилиты показан на рисунке 21.

```
logger -p local0.error "Test rsyslog message"
```

Рисунок 21 - Запуск утилиты для передачи сообщения системе регистрации событий

Порядок выполнения работы

1. Проверить наличие в используемом дистрибутиве Linux **rsyslog** демона. При его отсутствии установить.
2. Осуществить настройку сохранения сообщений от источника local8 в отдельный файл.
3. Проверить, что выполненные настройки корректны выполнением тестовой посылки сообщения утилитой **logger**.
4. Настроить хранение данных для источника local7 в базе данных PostgreSQL.
5. Проверить что данные теперь сохраняются в таблице базы данных.
6. Обеспечить отправку сообщений с одного узла на другой, на котором хранение осуществляется в базе данных PostgreSQL.

Требования к оформлению отчета и защите

Отчет должен содержать описание порядка выполнения всех команд и сделанных настроек системы регистрации событий. При защите отчета, исполнитель должен быть готов ответить на вопросы, касающиеся выполненных настроек и предоставить подтверждение о корректном выполнении каждого пункта задания.

Лабораторная работа №14. Использование сетевых файловых систем в операционной системе GNU Linux.

Цель работы: получение навыков по настройке и обслуживанию распределенных файловых систем, функционирующих по протоколам NFS и CIFS.

Теоретические сведения

NFS

В гетерогенных информационных системах для эффективной организации работы возникает необходимость экспорта и импорта файловых систем с удаленных сетевых узлов. В UNIX сетях основным инструментом для реализации функционала совместного использования файловых систем с удаленных узлов является сетевая файловая система, функционирующая по протоколу NFS. Данный протокол был разработан компанией Sun Microsystems в 1984 году и до сих пор имеет широкое распространение в UNIX-совместимых сетях. Первоначально NFS разрабатывалась для замены файловой системы для бездисковых станций, но разработка была настолько успешной, что ее начали использовать и в обычных локальных сетях. Первые версии использовали в качестве транспорта протокол UDP, последние версии начиная с 3 уже могут работать по протоколу TCP. Текущей версией NFS является 4.1 с поддержкой параллельного доступа к распределенной сети NFS серверов.

Сетевая файловая система функционирует на основе клиент-серверной технологии. Предоставление доступа к локальной файловой системе на стороне сервера происходит при помощи демонов **nfsd** и **mountd**. Их запуск, как правило, осуществляется автоматически при старте системы. Данные демоны для своей работы используют файл базы данных, содержащий информацию об экспортируемых файловых системах и клиентах, имеющих к ним доступ. Формат файла базы данных двоичный, поэтому существует текстовый конфигурационный файл **/etc/exports**, данные которого при помощи утилиты **exportfs** преобразуются в бинарный вид, уже в свою очередь используемый демонами NFS.

NFS позволяет экспортировать не только целиком файловые системы, но и отдельные каталоги, при этом переход от одной файловой системы к другой не допускается. В этом случае нужно отдельно создавать записи в конфигурационном файле для разных файловых систем.

Возможно экспортирование файловых систем, используя команду **share** с соответствующими ключами, рисунок 22.

```
share -F nfs -o rw=192.168.18.0/24 /home/users
```

Рисунок 22 - Экспортирование локальной файловой системы по протоколу NFS

В данном случае производится экспортирование каталога **/home/user** с правами на выполнение операций чтения и записи всем клиентам с адресами из сети 192.168.18.0/24.

Пример записи в файле конфигурации для выполнения экспортирования на рисунке 23.

```
/opt/exported 192.168.18.3(rw,sync)
```

Рисунок 23 - Файл конфигурации серверной части NFS

Для монтирования удаленных файловых систем с узлов-клиентов необходимо задействовать команду **mount**. Благодаря унификации возможно включение записи об автоматическом монтировании удаленных систем в стандартный файл с указанием какие файловые системы нужно монтировать при старте системы **/etc/fstab**.

При ручном монтировании можно использовать команду вида (рисунок 24):

```
mount -t nfs 192.168.18.1:/home/users /mnt/fs
```

Рисунок 24 - Команда монтирования удаленной файловой системы по протоколу NFS

При монтировании можно использовать различные флаги, определяющие действие клиента в случае возникновения сбоев при работе с удаленной файловой системой. Например, использование флага soft позволяет предотвратить зависание процессов, обращающихся к недоступной в данный момент удаленной файловой системы путем завершение активных операций с возвращением статуса об ошибке.

Для целей мониторинга за работой удаленной файловой системы полезно использовать программу **nfsstat**, отображающую статистику работы собираемую демонами NFS. Вывод информации по статистике процессов (-s) и об операциях на стороне клиента (-c). Анализ информации позволяет выявить проблемы в работе сервисов NFS, рисунок 25.

```
nfsstat -c
nfsstat -s
```

Рисунок 25 - Утилиты для вывода статистики работы протокола NFS

SMB

Особым значением в гетерогенных сетях обладает возможность использования общих файловых систем различными типами операционных систем. Для обеспечения возможности экспортирования и работы с файловыми системами ОС Windows в состав Linux операционных систем включаются пакеты поддержки протокола SMB.

SMB (Server Message Block) протокол для удаленного доступа к сетевым ресурсам (файлам, принтерам и т.п.) Первая версия протокола

называлась CIFS (Common Internet File System) и разрабатывалась несколькими крупными коммерческими компаниями в числе которых IBM, Microsoft, Intel. Последующие версии протокола связаны с разработками компании Microsoft для использования в операционных системах Windows для сетевого взаимодействия.

Для обеспечения возможности работы операционной системы Linux с ресурсами, находящимися на Windows узлах, используется пакет программ Samba. Он доступен для установки на большинстве UNIX-подобных систем в том числе дистрибутивах Linux.

Наряду с возможностью работы в качестве клиента для Windows сети, Linux система может быть сервером, предоставляющим свои ресурсы Windows клиентам.

Файлом, содержащим настройки работы Samba в Linux, является **/etc/samba/smb.conf**.

В нем описываются режимы работы сервера, настройки аутентификации клиентов, описываются ресурсы, предоставляемые для использования клиентами.

Пример файла на рисунке 26.

```
; Comments
[global]
; General server settings
netbios name = testsrv
server string =
workgroup = WORKGROUP
announce version = 5.0
socket options = TCP_NODELAY IPTOS_LOWDELAY SO_KEEPALIVE
SO_RCVBUF=8192 SO_SNDBUF=8192
passdb backend = tdbsam
security = user
[TEST-SHARE]
path = /opt/lab
browseable = yes
read only = no
guest ok = no
create mask = 0644
directory mask = 0755
```

Рисунок 26 - Пример файла конфигурации программы Samba

Сервер Samba способен использовать различные способы аутентификации пользователей. Параметр “user” означает использование отдельной базы пользователей для аутентификации при доступе к ресурсу. Опция “share” дает возможность использовать устаревшую схему доступа к ресурсу по паролю. Параметр “ads” позволяет серверу Samba являться полноценным членом домена Active Directory.

Для выполнения команды монтирования удаленного ресурса нужно воспользоваться командой mount, ее пример на рисунке 27.

```
mount -t cifs //192.168.18.3/public /mnt/shr -o
user=testuser,password=pwd,workgroup=WORKGROUP,ip=192.168.18.3,utf8
```

Рисунок 27 - Команда монтирования удаленной файловой системы по протоколу SMB

Порядок выполнения работы

1. Настроить экспортирование директории для выполнения операций чтения и записи по протоколу NFS для определенного узла в сети.
2. Подключить удаленную файловую систему по протоколу NFS и проверить возможность чтения и записи файлов.
3. На сервере NFS проанализировать статистику работы.
4. Настроить сервер для предоставления доступа по протоколу SMB, применяя опцию аутентификации “user”.
5. Произвести доступ к удаленному ресурсу с Windows узла.
6. Произвести монтирование удаленного ресурса по протоколу SMB на Linux узле.

Требования к оформлению отчета и защите

Отчет должен содержать описание всех выполненных команд по работе с сетевыми файловыми системами. При защите отчета, исполнитель должен быть готов ответить на вопросы по ходу выполнения лабораторной работы.

Список литературы

1. Rsyslog. Конфигурация для отправки сообщений на удаленный узел регистрации событий [Электронный ресурс] – Режим доступа: http://www.rsyslog.com/sending-messages-to-a-remote-syslog-server/, свободный. – Загл. с экрана. – Яз. англ.
2. Rsyslog. Конфигурация для хранения сообщений rsyslog в базе данных [Электронный ресурс] – Режим доступа: http://www.rsyslog.com/doc/v8-stable/tutorials/database.html, свободный. – Загл. с экрана. – Яз. англ.
3. Skeleton Init Script [Электронный ресурс] – Режим доступа: http://wiki.debian.org/LSBInitScripts, свободный. – Загл. с экрана. – Яз. англ.
4. MRTG web site [Электронный ресурс] – Режим доступа: http://oss.oetiker.ch/mrtg/, свободный. – Загл. с экрана. – Яз. англ.
5. LVM [Электронный ресурс] – Режим доступа: http://help.ubuntu.ru/wiki/lvm, свободный. – Загл. с экрана. – Яз. англ.
6. Технология виртуализации KVM [Электронный ресурс] – Режим доступа: http://help.ubuntu.ru/wiki/kvm, свободный. – Загл. с экрана. – Яз. англ.
7. Программирование на языке сценариев командной оболочки Bash [Электронный ресурс] – Режим доступа: http://www.opennet.ru/docs/RUS/bash_scripting_guide/, свободный. – Загл. с экрана. – Яз. англ.
8. Руководство по программе iptraf [Электронный ресурс] – Режим доступа: http://iptraf.seul.org/2.7/manual.html, свободный. – Загл. с экрана. – Яз. англ.
9. Уровни запуска операционной системы Linux [Электронный ресурс] – Режим доступа: https://www.centos.org/docs/rhel-rg-en-3/s1-boot-init-shutdown-sysv.html, свободный. – Загл. с экрана. – Яз. англ.
10. Cron. Запуск периодических процессов. [Электронный ресурс] – Режим доступа: https://help.ubuntu.com/community/CronHowto, свободный. – Загл. с экрана. – Яз. англ.
11. Udev. [Электронный ресурс] – Режим доступа: http://www.opennet.ru/base/sys/udev_dynamic.txt.html, свободный. – Загл. с экрана. – Яз. англ.
12. Сетевая файловая система NFS. [Электронный ресурс] – Режим доступа: http://www.ietf.org/html.charters/nfsv4-charter.html, свободный. – Загл. с экрана. – Яз. англ.
13. Samba. [Электронный ресурс] – Режим доступа: http://help.ubuntu.ru/wiki/samba, свободный. – Загл. с экрана. – Яз. англ.
14. Методы обеспечения безопасности в Linux. [Электронный ресурс] – Режим доступа: https://wiki.ubuntu.com/Security/, свободный. – Загл. с экрана. – Яз. англ.
15. Технология SeLinux. [Электронный ресурс] – Режим доступа: https://access.redhat.com/documentation/en-

US/Red_Hat_Enterprise_Linux/7/html/SELinux_Users_and_Administrators_Guide/, свободный. – Загл. с экрана. – Яз. англ.

16. Виртуальная файловая система procfs [Электронный ресурс] – Режим доступа: http://www.linuxcenter.ru/lib/articles/system/procfs.phtml, свободный. – Загл. с экрана. – Яз. англ.

17. Документация по операционной системе Windows [Электронный ресурс] – Режим доступа: http://technet.microsoft.com, свободный. – Загл. с экрана. – Яз. англ.

18. Немет Э. и др. Руководство администратора Linux. – М. и др. : Вильямс, 2007.

19. Курячий Г. В. Операционная система Linux: курс лекций: учебное пособие для студентов вузов //М.: Интернет-ун-т информ. технологий. – 2011.

20. Документация по языку Perl [Электронный ресурс] – Режим доступа: https://www.perl.org/docs.html, свободный. – Загл. с экрана. – Яз. англ.

21. Документация по языку Python [Электронный ресурс] – Режим доступа: https://docs.python.org/3/, свободный. – Загл. с экрана. – Яз. англ.

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА БЕЗОПАСНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Кафедра «Безопасные информационные технологии» (БИТ) осуществляет подготовку специалистов по специальности 090103 «Организация и технология защиты информации», бакалавров и магистров по направлению 090900 «Информационная безопасность». Широкий профиль подготовки, знание методов обеспечения информационной безопасности и средств защиты информации, практические навыки работы с современными техническими, программными и программно-аппаратными средствами защиты информации - все это позволяет выпускникам кафедры найти работу на производственных предприятиях, в подразделениях информационной безопасности, научно-исследовательских и инновационных организациях, а также в коммерческих структурах. Выпускники кафедры последних лет работают в Федеральной службе технического и экспортного контроля (ФСТЭК), Лаборатории Касперского, компаниях Dr.Web, специализированных предприятиях в сфере разработки и применения комплексных систем защиты информации «Эврика», ГазИнформСервис, и т.д. Партнерами кафедры являются ОАО «Воентелеком», Санкт-Петербургский институт информатики и автоматизации РАН, Военная академия Генерального штаба ВС РФ, Военно-космическая академия им. А.Ф. Можайского, Бостонский университет (США), Комитет по управлению городским имуществом администрации Санкт-Петербурга и другие научные организации и вузы.

Спивак Антон Игоревич
Спивак Ольга Игоревна
Лебедев Илья Сергеевич

Сетевые операционные системы
Лабораторный практикум

Учебное пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49