Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра систем информационной безопасности

БЕЗОПАСНОСТЬ ОПЕРАЦИОННЫХ СИСТЕМ

Методические указания по выполнению лабораторных работ
для студентов специальности
5В100200 – Системы информационной безопасности

Алматы 2017

СОСТАВИТЕЛЬ: Е.А. Зуева. Безопасность операционных систем. Методические указания по выполнению лабораторных работ для студентов специальности 5В100200 – Системы информационной безопасности. - Алматы: АУЭС, 2017. – 68 с.

Методические указания содержат указания по подготовке к проведению двенадцати лабораторных работ по работе операционных систем (ОС) Linux и Windows; приведены описания каждой лабораторной работы, дана методика проведения и ход выполнения, представлен перечень рекомендуемой литературы и контрольные вопросы в конце каждой темы.

Выполнение лабораторных работ определяет практичкскую основу профессионального образования курса. Программа дисциплины «Безопасность операционных систем» предусматривает изучение методов и средств управления процессами режимами работы ВМ, систем и сетей, режимами управления вводом-выводом информации; файловой системой; изучение способов организации и защиты файлов, методов распределения и защиты памяти, рассмотрены принципы построения и защита от сбоев и несанкционированного доступа, а также характеристика средств управления многопроцессорными системами и сетями.

Профессиональная подготовка бакалавра специальности 5В100200 – «Системы информационной безопасности» требует умения грамотно работать с ОС, а также с различными операционными средами, системами и оболочками. Предметом изучения в рассматриваемой дисциплине являются ОС и средства, расширяющие их возможности.

Методические указания предназначены для студентов всех форм обучения специальности 5В100200 – Системы информационной безопасности.

Илл. 45, табл. 6, библиогр. - 16 назв.

Рецензент: Аренбаева Ж.Г., канд. экон. наук, профессор АУЭС

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2017 г.

Содержание

Введение

В настоящий сборник включены лабораторные работы, целью которых является изучение студентом принципов безопасной работы операционных систем (ОС) Linux и Windows.

Данная дисциплина предназначена для высших учебных заведений, ведущих подготовку по специальности 5В100200 - Системы информационной безопасности.

Программа дисциплины «Безопасность операционных систем» предусматривает изучение методов и средств управления процессами режимами работы ВМ, систем и сетей, режимами управления вводом-выводом информации; файловой системой; изучение способов организации и защиты файлов, методов распределения и защиты памяти, рассмотрены принципы построения и защита от сбоев и несанкционированного доступа, а также средств управления многопроцессорными системами и сетями.

В рамках данного пособия проводится укрепление знаний студента о функциональных возможностях и адаптации ОС под конкретные задачи грамотного администрирования и правильного распределения ролей участия взаимодействия всех пользователей в системе и возможности управления ограничением воздействия на определенных этапах вмешательства в систему.

В настоящий сборник включены лабораторные работы, целью которых является обучение методологическим основам принципов построения и функционирования средств реализации системного программного обеспечения вычислительных машин, систем и сетей и организация ее защиты.

Материал по каждой лабораторной работе включает в себя цель, рабочее задание, методические указания для выполнения работы и контрольные вопросы для самостоятельной подготовки.

Этапы выполнения лабораторной работы следующие: изучение теоретической части, выполнение рабочего задания, создание отчета и защита работы.

Все лабораторные работы ориентированы на проявление элементов научно-исследовательской деятельности студентов.

Выполнение каждой лабораторной работы должно завершаться оформлением отчета по «Стандарту организации учебно-методические и учебные работы СТ НАО 56023-1910-04-2014». Выполненная работа и оформленный отчет защищается у преподавателя.

Выполнение лабораторных заданий дает возможность выработки навыков и знаний у студентов.

1 Лабораторная работа №1. Запуск и конфигурирование ОС

Цели работы: изучение работы с физическими и логическими разделами диска; приобретение навыков настройки ОС.

1.1 Рабочее задание

1. Скачать и установить виртуальную машину *Sun* или *VmWare*.
2. Выбрать дистрибутив *Linux* и установить на виртуальную машину.
3. Установить *Windows* на виртуальную машину.
4. В каждой ОС сделать *snapshot* №1 (первоначальный снимок ОС).
5. Произвести в каждой ОС в виртуальной машине настройку:
- настроить сеть, обновить репозитории/источники данных;
- скачать и установить шрифты, темы, сменить оформление;
- установить в *Linux* программы *midnight commander*, *nano*, *gedit*;
- установить в *Windows* текстовой редактор с цветной подсветкой.
6. В каждой виртуальной машине сделать *snapshot* №2 (снимок ОС №2).
7. Восстановить по *snapshot* №1 состояние ОС.
8. Восстановить по *snapshot* №2 состояние ОС.

1.2 Методические указания к выполнению лабораторной работы

Установить ОС можно:

а) на виртуальную машину: выбирается и устанавливается в *Windows* виртуальная машина. Затем создается новая машина и в качестве образа ОС ставится выбранная ОС: файл вида **.iso;*

б) на раздел жесткого диска: при разбиении разделов можно использовать встроенные средства *Windows* (правая клавиша «*Компьютер*» - «*Управление дисками*»). Или программа *Partition Magic*, *Acronis disk director suite*. После создания раздела надо перезагрузиться и в начале загрузки удерживать нажатой клавишу «*Delete»* – чтобы зайти в *BIOS* и выставить первичной загрузкой *CD* (если планируется установка с *CD*-диска) или *USB* (если планируется установка с *flash*). После сохранения изменений и выхода из *BIOS* начинает считываться сменный носитель и начинается установка.

В процессе установки студенту для входа в систему нужно придумать логин (фамилия_имя), пароль и выбрать оболочку.

Зафиксируйте, каким образом у вас произошло разделение дискового пространствавы в каждой из установленных ОС.

1.3 Список контрольных вопросов

1 Какие папки находятся в корневом разделе, что в них содержится?

2 Что такое *swap*?

3 Какие файловые системы вы задействовали при установке разных дистрибутивов и на каких разделах, какое разбиение дисков вы произвели?

2 Лабораторная работа №2. Управление процессами. Изучение работы системных функций

Цель работы: приобретение навыков работы с процессами.

2.1 Рабочее задание

Выполнить задания по управлению процессами в *Linux*:

1) Запустить игру и файловый менеджер. Переназначить им приоритеты. Завершить все вышеназванные процессы в одной консольной строке.

2) Открыть вторую и третью консоль из первого терминала. Закрыть первую и третью консоли из второго терминала.

3) Вывести информацию о состоянии процессов. Отфильтровать по состоянию, перевести из одного режима в другой (*ps, pstree, top, bg, fg*).

4) Назначить приоритеты запускаемым задачам в фоновом режиме, изменить приоритеты запущенных процессов (*nice, renice*).

2.2 Методические указания к выполнению лабораторной работы

Все запущенные процессы имеют уникальные номера - *PID*. Команды *Linux*, необходимые для мониторинга работы ОС (все показания выводятся на экран в реальном времени, а число после команды означает интервал между выводом информации):

- *top* - информация в реальном времени о процессах, потребление ОЗУ;
- *ps* – показать все процессы;
- *ps aux* - показать все загруженные процессы;
- *echo $$* - показать *PID* оболочки;
- *fuser -va 22/tcp* - показать *PID* процесса, использующий порт 22;
- *fuser -va /home* - показывает *PID* процесса, имеющего доступ к /*home*;
- *lsof /home* - список процессы, которые используют /*home*;
- *kill* 1234 - «завершить» процесс с *PID* 1234;
- *killall* <имя_программы> - «убить» процессы по имени программы;
- *nice* и *renice* – назначение и переназначение приоритетов процессам.
- *ps axu | grep test* - все процессы, запущенные от пользователя *test.*

Возможность для пользователя задавать значение приоритета его собственных процессов определяет отношение к другим пользователям системы, ведь процесс - это набор правил, которым руководствуется данная программа при использовании выделенного процессорного времени, памяти и ресурсов ввода/вывода. Каждый процесс, запущенный в системе, имеет свой *ID (PID)*, по которому его можно отслеживать. Ядро позволяет собирать информацию о каждом процессе, которая включает, но не ограничивается:

- статус процесса (работает, спит, зомби или остановлен);
- приоритет выполнения процесса;
- информация об используемых ресурсах и- владелец процесса;
- сетевые порты и файлы, открытые процессом и т.д.

Создаем процесс *yes* в терминале, перенаправив вывод в */dev/null:*

yes > /dev/null &

[1] 5997

Воспользуемся командой, чтобы извлечь информацию о процессе:

ps -l

```
F S   UID   PID  PPID  C  PRI  NI  ADDR SZ WCHAN TTY    TIME      CMD
0 S  1000  5830  3283  0    80   0 -   6412  wait          pts/0    00:00:00 bash
0 R  1000  5997  5830 99  80   0 -   1757 -               pts/0    00:00:09 yes
0 R  1000  5998  5830  0    80   0 -   2399 -               pts/0    00:00:00 ps
```

Из этой таблицы можно узнать, что:

F - FLAG: процесс запущен без привилегий суперпользователя. В противном случае, мы могли бы увидеть число 4 или сумму 1 и 4.

S - STATE: процесс в настоящее время работает.

UID - ID пользователя, инициализировавшего процесс.

PID - ID процесса нашей команды *yes* 5997.

PPID - Parent Process ID. Это *ID* родительского для нашей команды yes процесса. В нашем случае это bash с *PID* 5830.

C - загрузка процессора, выражается в %.

PRI - Приоритет процесса, большее значение значит меньший приоритет.

NI - Значение *nice*, которое находится в диапазоне от -20 до 19. Большее значение означает меньший приоритет.

Принцип работы планировщика *Linux* (для ядра версии >= 2.6) вытесняющий, то есть способность ядра выбирать среди всех заданий то, которое имеет наивысший приоритет. Далее, ядро делит списки приоритета на задачи реального времени и пользовательские задания, ранжирующиеся от 1-100 и 101-140 соответственно. Далее, ядро выделяет задачам с более высоким приоритетом больший квант времени, а задачам с меньшим приоритетом - меньший квант времени, который в среднем составляет 200 и 10 мс соответственно, т.е. каждое задание допускается к выполнению только, если у него остается какая-либо часть времени. Поэтому меньший отрезок времени для выполнения означает, что процесс получает меньше времени в очереди выполнения и получает меньше ресурсов. Когда отрезок времени процесса заканчивается, он помещается в очередь выполнения с истекшим временем: затем его приоритет пересчитывается: и он снова помещается в активную очередь выполнения. Важно помнить, что обе очереди выполнения содержат списки задач, отсортированных по их приоритету.

Чтобы установить значение *nice* ниже нуля, требуются права суперпользователя. В противном случае будет установлено значение 0. Если попробовать задать значение *nice* -1 без прав *root*:

$ $ nice -n -1 yes > /dev/null &

[1] 5285

nice: cannot set niceness: Permission denied

$ ps -l

```
F S   UID   PID  PPID  C  PRI  NI ADDR SZ WCHAN  TTY      TIME   CMD
0 S  1000  3383  3380  0    80   0 -  6447 wait             pts/0    00:00:00 bash
0 R  1000  5285  3383 95  80   0 -  1757 -                pts/0    00:00:07 yes
0 R  1000  5295  3383  0   80   0 -  2399 -                pts/0    00:00:00 ps
```

Поэтому, чтобы задать значение *nice* меньше 0, необходимо запускать программу как *root* или использовать *sudo*.

 # nice -n -1 yes > /dev/null &

```
[1] 5537
```

 # ps -l

```
F S   UID   PID  PPID C  PRI  NI ADDR SZ WCHAN  TTY      TIME     CMD
4 S     0    5428  3383  0  80   0 - 14430 wait             pts/0    00:00:00 su
0 S     0    5436  5428  1   80   0 -  7351 wait             pts/0    00:00:00 bash
4 R     0    5537  5436 87 79  -1 -  1757 -                pts/0    00:00:04 yes
4 R     0    5538  5436  0  80   0 -  2399 -                pts/0    00:00:00 ps
```

Пробуем изменить значение *nice* у запущенной программы с помощью команды *renice (*есть работающая программа *yes* со значением *nice* 10):

ps -l

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY      TIME   CMD
0 S  1000  3383  3380  0  80   0 -  6447 wait             pts/0    00:00:00 bash
0 R  1000  5645  3383 99  90  10 -  1757 -                pts/0    00:00:04 yes
0 R  1000  5646  3383  0  80   0 -  2399 -                pts/0    00:00:00 ps
```

Изменим значение *nice* на 15:

renice -n 15 -p 5645

```
5645 (process ID) old priority 10, new priority 15
```

ps -l

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY      TIME   CMD
0 S  1000  3383  3380  0  80   0 -  6447 wait             pts/0    00:00:00 bash
0 R  1000  5645  3383 99  95  15 -  1757 -                pts/0    00:00:31 yes
0 R  1000  5656  3383  0  80   0 -  2399 -                pts/0    00:00:00 ps
```

Согласно правилам, обычный пользователь может только увеличивать значение *nice* (уменьшая приоритет) любого процесса. Если попробовать изменить значение *nice* с 15 до 10, получим следующее сообщение об ошибке:

renice -n 10 -p 5645

```
renice: failed to set priority for 5645 (process ID): Permission denied
```

Команда *renice* позволяет суперпользователю изменять значение *nice* процессов любого пользователя. Это делается с помощью ключа *-u*. Команда изменяет значение приоритета всех процессов пользователя на -19:

renice -n -19 -u lubos

```
1000          ID old priority 0, new priority -19.
```

2.3 Список контрольных вопросов

1 Зачем нужны приоритеты?

2 Какие режимы работы процессов существуют?

3 Лабораторная работа №3. Команды управления процессами. Сигналы, каналы. Взаимодействие процессов систем

Цель работы: приобретение навыков работы с процессами и сигналами, каналами.

3.1 Рабочее задание

1. Привести примеры в *Linux* работы функций *fork(), exec(), signal(), wait(), jobs, mkfifo, nohup, pipe*().

2. Настроить планировщик *Linux* на запуск двух задач по расписанию.

3. Выполнить задания по управлению процессами в *Windows*:

1) Просмотреть в *Windows* дерево процессов утилитой *Tlist.exe*. Запустить родительские и дочерние процессы. Сделать соответствующие выводы.

2) Просмотреть в *Windows* список установленных драйверов устройств.

3) Разобрать в *Windows* работу утилиты *Msconfig*.

3.2 Методические указания к выполнению лабораторной работы

Для выполнения пункта 2 рабочего задания надо понимать, что *cron* – демон, занимающийся планированием и выполнением команд, запускаемых по определенным датам и в определенное время. Команды, выполняемые периодически, указываются в файле */etc/crontab* (не через команду *cron*, а путем внесения строк в файл *crontab* или с 28 использованием одноименной команды *crontab*). Команды, которые должны быть запущены лишь однажды, добавляются при помощи *at*. Синтаксис строки в *crontab*: каждая команда в файле *crontab* занимает одну строку и состоит из шести полей:

минута час день_месяца месяц день_недели команда

Допустимые значения: минута от 0 до 59; час от 0 до 23; день_месяца от 1 до 31; месяц от 1 до 12 (или буквы от jan до dec, независимо от регистра); день_недели от 0 до 6 (0 это воскресенье или три буквы от sun до sat).

Если в соответствующее поле поместить символ *, это будет соответствовать любому возможному значению. Для полей можно указывать диапазоны значений, разделенных дефисом, например, вывод "*Hello World*!" в 11:00 в 6,7,8,9 дни января, февраля и марта:

*0 11 6-9 1-3 * echo "Hello World!"*.

А вывод "*Hello World*!" каждый четный час каждого понедельника:

*0 */2 * * mon echo "Hello World!"*

Для выполнения задания пункта 3 необходимо скачать установочный файл *Windbg* (*Debugging Tools for Windows*) по ссылке с сайта *Microsoft*: *http://msdl.microsoft.com/download/symbols/debuggers/dbg_x86_6.11.1.404.msi.*

В процессе установки возникает окно, показанное на рисунке 1.

Рисунок 1 – Принскрин при установке программного обеспечения

1) После установки программного обеспечения, необходимо для просмотра дерева процессов вызвать командную строку, напечатать «*cmd*» и нажать клавишу *Enter* (рисунок 2).

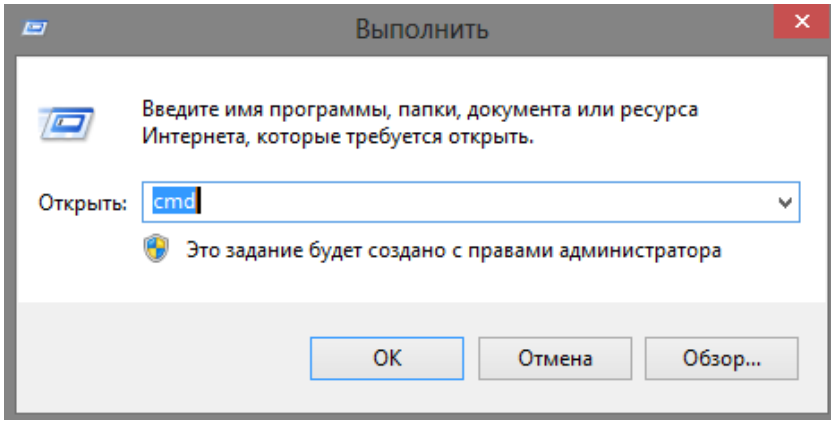

Рисунок 2 – Вызов командной строки

Открывается командная строка *Windows*. Набираем команду для перехода в только что созданную директорию (рисунок 3): *cd "C:\Program Files\Debugging Tools for Windows (x86)"*.

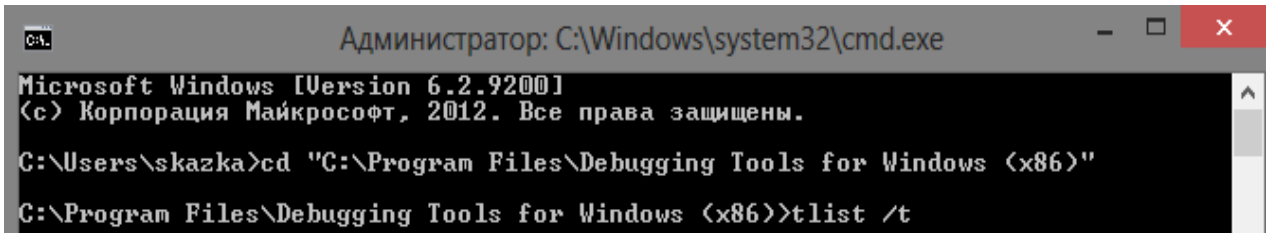

Рисунок 3 – Переход в созданную директорию

Дерево процессов показывается утилитой «*Tlist.exe*» при ключе */t.* Образец вывода команды показан на рисунке 4.

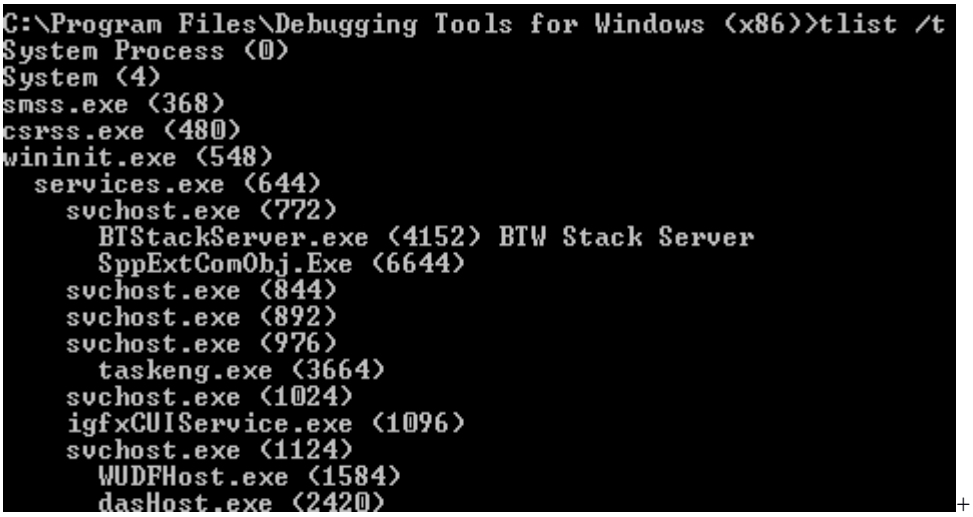

Рисунок 4 – Пример части работы команды

Взаимоотношения процессов (дочерний-родительский) *Tlist* показывает отступами. Имена процессов, родительские процессы которых на данный момент завершились, выравниваются по левому краю, потому что установить их родственные связи невозможно - даже если процессы-прапредки еще существуют. *Windows* сохраняет идентификатор только родительского процесса, так что проследить его создателя нельзя. Чтобы убедиться в этом, выполните следующие операции:

а) откройте окно командной строки. Наберите *start cmd* для запуска второго окна командной строки, расположите окна рядом (рисунок 5);

Рисунок 5 – Запуск второго окна командной строки

б) откройте диспетчер задач (*Ctrl+Alt+Del* – Диспетчер задач);

в) переключитесь на второе окно командной строки. Введите «*mspaint*» для запуска *Microsoft Paint* (рисунок 6);

Рисунок 6 – Запуск во втором окне *Microsoft Paint*

г) щелкните второе окно командной строки, введите «*exit*» и нажмите *Enter* (рисунок 7);

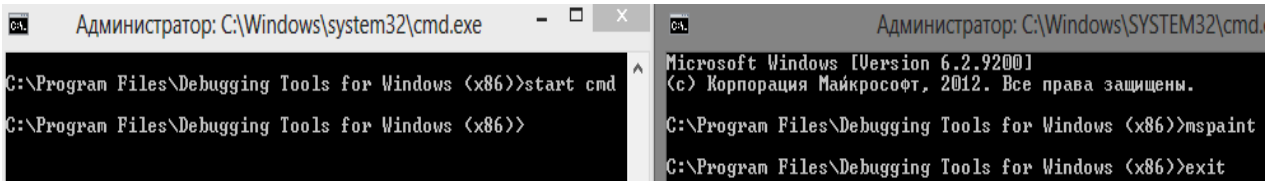

Рисунок 7 – Выход во втором окне

д) заметьте, что окно *Paint* остается, даже несмотря на то, что родительский процесс *cmd2* завершился. Переключитесь в диспетчер задач и откройте его вкладку «Процессы». Найдите задачу «Обработчик команд *Windows*», переключитесь на вкладку «Подробности» и щелкните процесс *Cmd.exe* правой кнопкой мыши и выберите команду «Завершить дерево процессов» (рисунок 8);

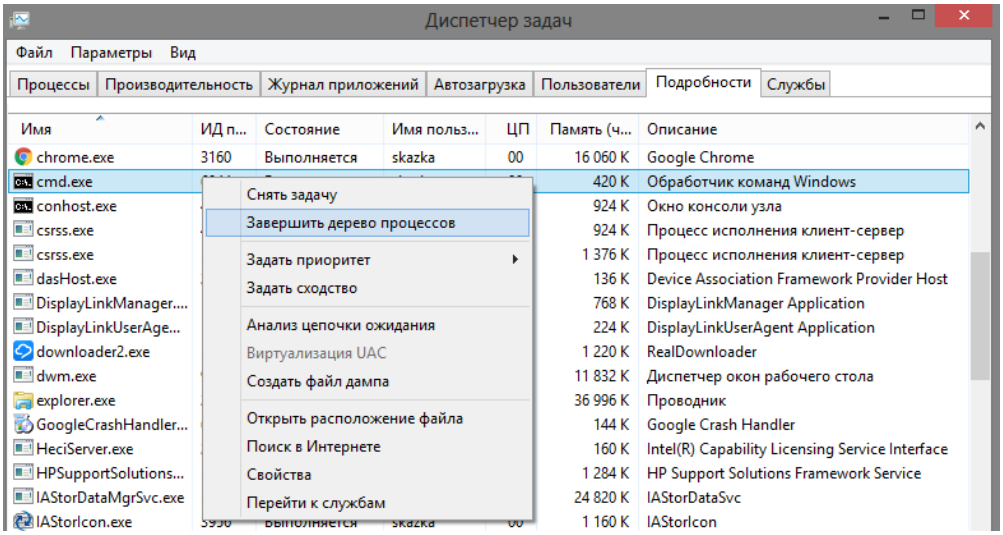

Рисунок 8 – Отслеживание процесса

е) в окне «Предупреждение диспетчера задач» (рисунок 9) щелкните «Завершить дерево процессов». Теперь и первое окно командной строки исчезнет, но вы по-прежнему сможете наблюдать окно *Paint*, так как оно является внуком первого из завершенных процессов «Командной строки». А поскольку второй (родительский процесс для *Paint*, т.е. *cmd2*) тоже завершен, связь между родителем и внуком потеряна.

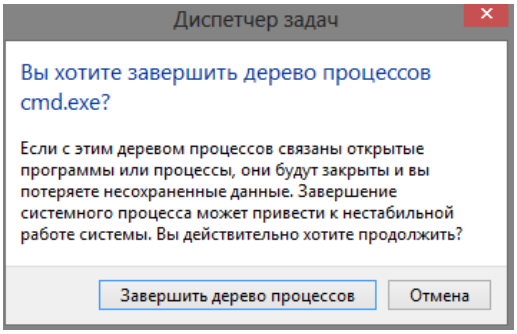

Рисунок 9 – Завершение процесса

2) Просмотр установленных драйверов устройств.

Вывести список установленных драйверов: Пуск → Выполнить в появившейся командной строке набрать *msinfo32*. Выбираем Программная среда → Системные драйверы (рисунок 10).

В этом окне выводится список драйверов, определенных в реестре, а также их тип и состояние - работает или нет. Драйверы устройств и процессы *Windows*-сервисов определяются в разделе реестра *HKLM\SYSTEM\CurrentControlSet\Services*. Однако они отличаются по коду типа. На вкладке «Сведения о системе» можно найти реальную информацию об ОС.

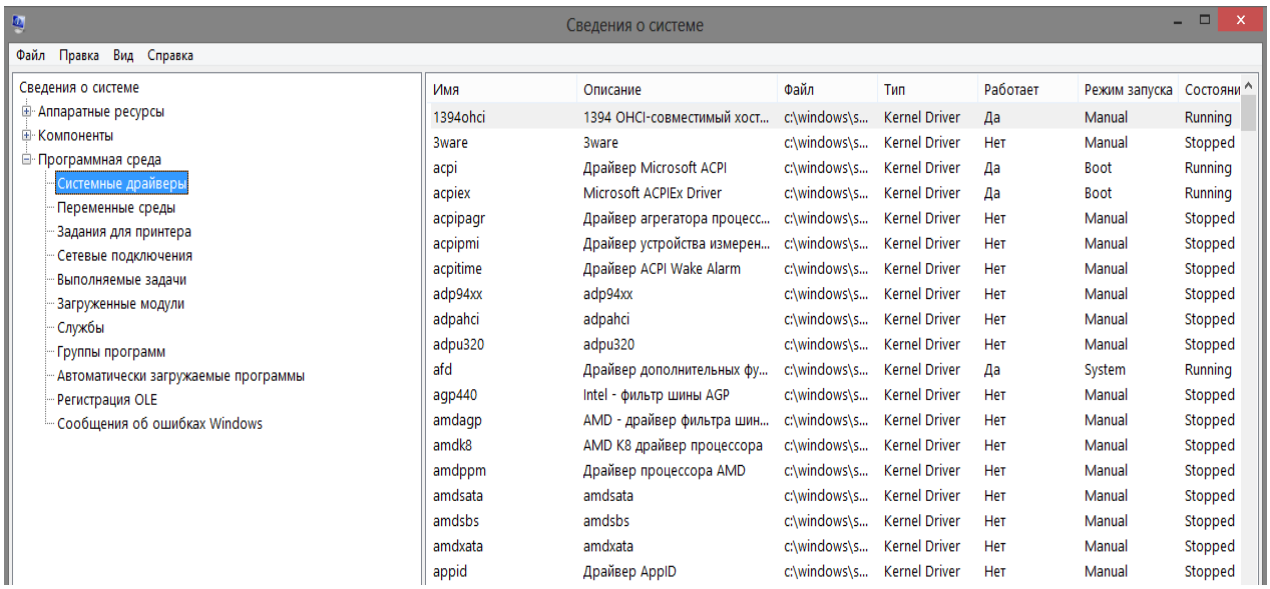

Рисунок 10 – Просмотр установленных драйверов устройств

3) Утилита *Msconfig*.

Чтобы увидеть, какие программы настроены на автоматический запуск на вашем компьютере, запустите утилиту *Msconfig* (рисунок 11). Сделайте самостоятельное исследование, перенастроив некоторые службы.

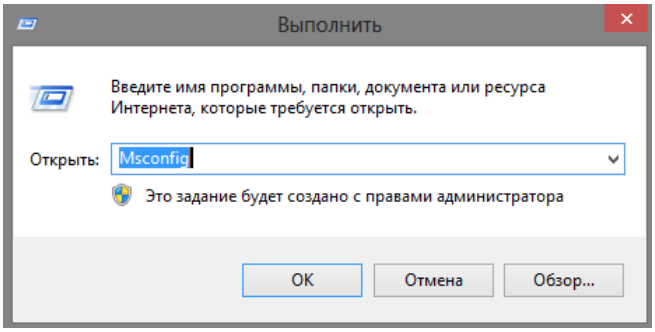

Рисунок 11 – Просмотр программ на автоматический запуск

3.3 Список контрольных вопросов

1 Какую информацию дает *Tlist.exe*?

2 Как можно просмотреть список установленных драйверов устройств?

4 Лабораторная работа №4. Способы организации ввода-вывода. Произвольный доступ к файлу. Библиотека стандартного ввода-вывода систем

Цели работы: приобретение навыков при организации ввода-вывода.

4.1 Рабочее задание

1. В *Linux* создайте каталог со своим именем в директории */home/user*/. Все файлы в этой лабораторной работе должны находиться здесь.

2. Создайте файл *file1*, в который надо записать все адреса электронной почты, встречающиеся в каком-либо на выбор студента файле из директории /*etc*. Выведите содержимое файла также и на экран.

3. Создайте файл *file2*, в который выводятся строки файла */var/log/Xorg.0.log*, содержащие предупреждения и информационные сообщения.

4. Создайте файл *file3* из файла *file2*, копируя и заменяя маркеры предупреждений и информационных сообщений на слова «*Warning*:» и «*Information*:».

5. Найти в директории /*etc* все файлы, которые имеют какое-либо одно на ваш выбор расширение, выведите на экран файл с разными атрибутами (опциями).

6. Выведите список пользователей системы с указанием их *UID*, отсортировав по *UID*. Сведения о пользователей хранятся в файле */etc/passwd*. В каждой строке этого файла первое поле – имя пользователя, третье поле – *UID*. Разделитель – двоеточие.

7. Подсчитайте общее количество строк в файлах, находящихся в директории /*var/log*/ и имеющих расширение *log*.

4.2 Методические указания к выполнению лабораторной работы

Основным интерфейсом в ОС *Linux* является консольный интерфейс с текстовым вводом и выводом данных. Это определяет подход к управлению объектами операционной системы в их текстовом отображении. Например, состояние процессов отображается в виде набора текстовых файлов в псевдофайловой системе */proc*, сведения о событиях в системе хранятся в текстовых файлах журналов, настройки отдельных пакетов в текстовых конфигурационных файлах. Это необходимо для решения задач управления ОС освоение инструментария работы с текстовыми потоками.

У любого процесса, по умолчанию, всегда открыты три файла – *stdin* (стандартный ввод, клавиатура), *stdout* (стандартный вывод, экран) и *stderr* (стандартный вывод сообщений об ошибках на экран). Эти и любые другие открытые файлы могут быть перенаправлены. В данном случае термин «перенаправление» означает: получить вывод из файла (команды, программы, сценария) и передать его на вход в другой файл (команду, программу,

сценарий). Дескрипторы файлов открытых, по умолчанию: 0 = *stdin*; 1 = *stdout;* 2 = *stderr*. При этом действует следующее:

1 команда > файл – перенаправление стандартного вывода в файл, содержимое существующего файла удаляется.

2 команда >> файл – перенаправление стандартного вывода в файл, поток дописывается в конец файла.

3 команда1 | команда2 – перенаправление стандартного вывода первой команды на стандартный ввод второй команды = образование конвейера команд.

4 команда1 $(команда2) – передача вывода команды 2 в качестве параметров при запуске команды 1.

В документации по командам говорится, что:

sort – сортирует поток текста в порядке убывания или возрастания;

uniq – удаляет повторяющиеся строки из отсортированного файла;

cut – извлекает отдельные поля из текстовых файлов (поле – последовательность символов в строке от разделителя до разделителя);

head – выводит начальные строки из файла на *stdout;*

tail – выводит последние строки из файла на *stdout*;

wc – считает количество слов/строк/символов в файле или в потоке;

tr – заменяет одни символы на другие.

Полнофункциональные многоцелевые утилиты:

grep – многоцелевая поисковая утилита, использующая выражения;

grep pattern [file...] – утилита поиска участков текста в файле(ах), соответствующих шаблону *pattern*;

Sed – неинтерактивный «потоковый редактор», принимает текст либо с устройства *stdin*, либо из текстового файла, выполняет операции над строками и выводит результат на устройство *stdout* или в файл. *Sed* определяет по заданному адресному пространству, над какими строками следует выполнить операции. Адресное пространство строк задается либо их порядковыми номерами, либо шаблоном. Например, команда *3d* заставит *sed* удалить третью строку, а команда /*windows*/*d* означает, что строки, содержащие "*windows*", должны быть удалены. Наиболее часто используются команды *p* – печать (на *stdout*), *d* – удаление и *s* – замена;

awk – утилита контекстного поиска и преобразования текста, инструмент для извлечения и/или обработки полей (колонок) в структурированных файлах. *Awk* разбивает каждую строку на отдельные поля. Поля - это последовательности символов, отделенных друг от друга пробелами, однако, можно назначить другие символы в качестве разделителя полей. *Awk* анализирует и обрабатывает каждое поле в отдельности.

4.3 Список контрольных вопросов

1 Какие варианты вывода существуют?

2 Какие команды используются для поиска и замены участков текста?

5 Лабораторная работа №5. Файловая система. Доступ к файлам. Работа с файлами и каталогами. Изучение типов файлов. Поиск системных журналов систем

Цели работы: приобретение навыков работы с файловыми системами, с ее объектами, сменой их атрибутов и форматам вывода информации по ним.

5.1 Рабочее задание

1. Выполнить задания по управлению папками и файлами в *Linux*:

- зайти в меню терминала, «Настройки» - «Управление профилями», сменить цвет текста, фона, иконку и строку с названием консоли на свою фамилию, чтобы это фигурировало в отчете;
- определить имя текущего каталога. Создать в /*home*/имя_вашего_ пользователя 2 папки, переместить одну папку в другую;
- записать календарь текущего года, месяца в файлы *year.txt* и *month*;
- проверить существование файлов *year.txt* и *yaer.txt* в данной папке;
- создать к файлу *month* все виды ссылок;
- переименовать все созданные ссылки и дозаписать имя и фамилию в файл *month;*
- устроить поиск строки, содержащей имя и фамилию в файлах папки;
- сравнить файлы в папке;
- создать каталог «отчество_студента» и переместить в него все файлы по маске *.*txt*;
- зайти в папку «отчество_студента» и удалить все файлы по маске, содержащие в названии символ на ваш выбор;
- создать пустой файл *and.txt*. Объединить *year.txt* и *month*, слить в пустой файл, дозаписать список процессов и историю команд и просмотреть его постранично. Найти в нем 2 какие-либо последовательности символов одновременно на ваш выбор;
- подсчитать количество строк в файле *and.txt*;
- изменить (назначить разные) права (доступ, владельцев, группы) на три любые объекта в вашей папке на каждый файл, сохранив таблицу первоначальных доступа, прав, групп в файле *prava.txt* и записав полученные права после смены;
- удалить рабочую папку и все объекты в ней рекурсивно.

2. В *Windows* отследить события в системных журналах событий.

5.2 Методические указания к выполнению лабораторной работы

1. Для работы в *Linux* с файлами и директориями есть команды:

- *pwd* - выводит текущий путь;
- *ls* - выводит список файлов и каталогов по порядку (*ls -l* – наиболее полный вывод с атрибутами содержимого папки);
- *cd* - переход в домашнюю директорию;
- *cd /home* - переход в директорию /*home*;
- *cd ..* - переход на уровень выше;

– *cd* - - возврат в предыдущую директорию;
– *mkdir* 0 - создание директории с именем 0;
– *rmdir* 0 - удаление директории с именем 0;
– *touch* 2 – создание пустого файла;
– *cp* 1 2 – копирование файла 1 в 2;
– *nano* 1 (или *gedit* 1) – открытие файла 1 через текстового редактора;
– *cat* 1 - показать содержимое файла 1;
– *tac* 1 - показать содержимое файла 1 строками в обратном порядке;
– *echo «*Привет» – выводит на экран сообщение «Привет»;
– *echo* «Привет» > 1 – замена содержимого файла 1 строкой «Привет»;
– *echo* «Привет» >> 1 – дозапись в файл 1 строки «Привет»;
– *echo* «Привет» | *tee -a* 1 - добавление к концу файла 1 «Привет»;
– *mv* 1 2 – переименование/перемещение 1 в 2 (для файла и папки);
– *head* 1 – выводит начало файла (по умолчанию первые 10 строк);
– *tail* 1 - выводит конец файла (по умолчанию последние 10 строк);
– *sort* 1 – сортирует строки в файле 1;
– *find* 1 – поиск файла 1 в текущей папке;
– *grep* «23» 1 – поиск в файле 1 строки, содержащей «23»;
– *ln* 1 22 – создание ссылки 22 на файл 1 (полным копированием);
– *ln –s* 1 33 – создание символьной ссылки 33 на файл 1 (ярлык);
– *locate* 1 - поиск всех файлов по машине, содержащих в названии 1;
– *du -sh* 0 - размер заданной директории 0, подходит и для файлов;
– *wc* 1- подсчет количества символов, букв, байт в файле 1;
– *rm* 1 - удаление файла 1;
– *rm* -r - рекурсивное удаление;
– *more* 1 – постраничный просмотр файла 1;
– *yes abc* – бесконечно печатает *abc;*
– *nl 1* – нумерует строки файла 1.

ls -l выдает информацию о текущей папке в полном объеме по своим атрибутам, дате и владельцу, его группе и правами. Объекту соответствует 9 битов разрешений, они формируют режим доступа и определяют, какие пользователи и группы имеют права на объект (рисунок 12).

Рисунок 12 – Распределение прав владения объектом

После одного бита определения объекта:
- первые три бита определяют права для владельца файла;
- вторые три бита определяют права для группы, к которой принадлежит владелец данного файла;

- последние определяют права для всех остальных пользователей.

Изменить режим доступа для файлов может только владелец файла или суперпользователь. Код доступа к файлу задается одним из двух режимов:

– цифровой (1- выполнение, 2 - запись, 4 - чтение);

– символьный (x - выполнение, w - запись, r - чтение).

Обозначения для символьного режима: *u – user* (владелец файла), *g – group* (группа владельца), *o – other* (остальные), *a – all* (все). В цифровом виде для смешанных прав соответствующие числа складывают (рисунок 13).

Для работы в *Linux* с правами доступа на объекты есть команды:

- *chmod* 777 1 – изменить прав доступа файла 1, 0777 – разрешение на чтение/запись/исполнение для всех групп – полный доступ;

- *chmod -R* 777 0 - рекурсивное изменение прав доступа к директории 0, 777 – разрешение на чтение/запись/исполнение для всех групп, все вложенные директории и файлы будут иметь права 777;

- *chown skazka:test 1* - изменение владельца и группы владельца только для файла 1;

- *chgrp test* 1 - изменение группы владельца файла 1;

Для передачи файла пользователю *test*: *cat file.txt | write test pts/1*. Для отправки файла всем пользователям: cat *file*.txt | wall.

- *history | tail -30* - показать последние 30 набранных команд.

Команды в терминале выводят информацию, можно использовать метасимволы для формирования сложных команд:

а) > - запись в файл. Например, существует файл *goa* с некоторым содержимым, набираем «*date>goa*» - содержимое файла стирается и в него записывается текущая дата;

б) >> - дозапись в конец существующего файла. Например, существует файл *goa* с некоторым содержимым, набираем «*date>>goa*» - содержимое файла остается + в конец файла *goa* добавляется строка с текущей датой;

в) | - программный канал - результат одной команды передается другой команде;

г) & - процесс выполняется в фоновом режиме;

д) ? – любой 1 символ;

е) * - любое количество любых символов;

ж) ; - перечисление, команды выполняются друг за другом;

и) && - при объединении команд: последующая команда выполняется только при нормальном завершении предыдущей;

к) || - при объединении команд: последующая команда выполняется только, если не выполнилась предыдущая команда;

л) () - группирование команд в скобки, как в арифметических расчетах;

м) { } - группирование команд с объединенным выводом;

н) [ ] - указание диапазона данных или перечисление данных (без запятых).

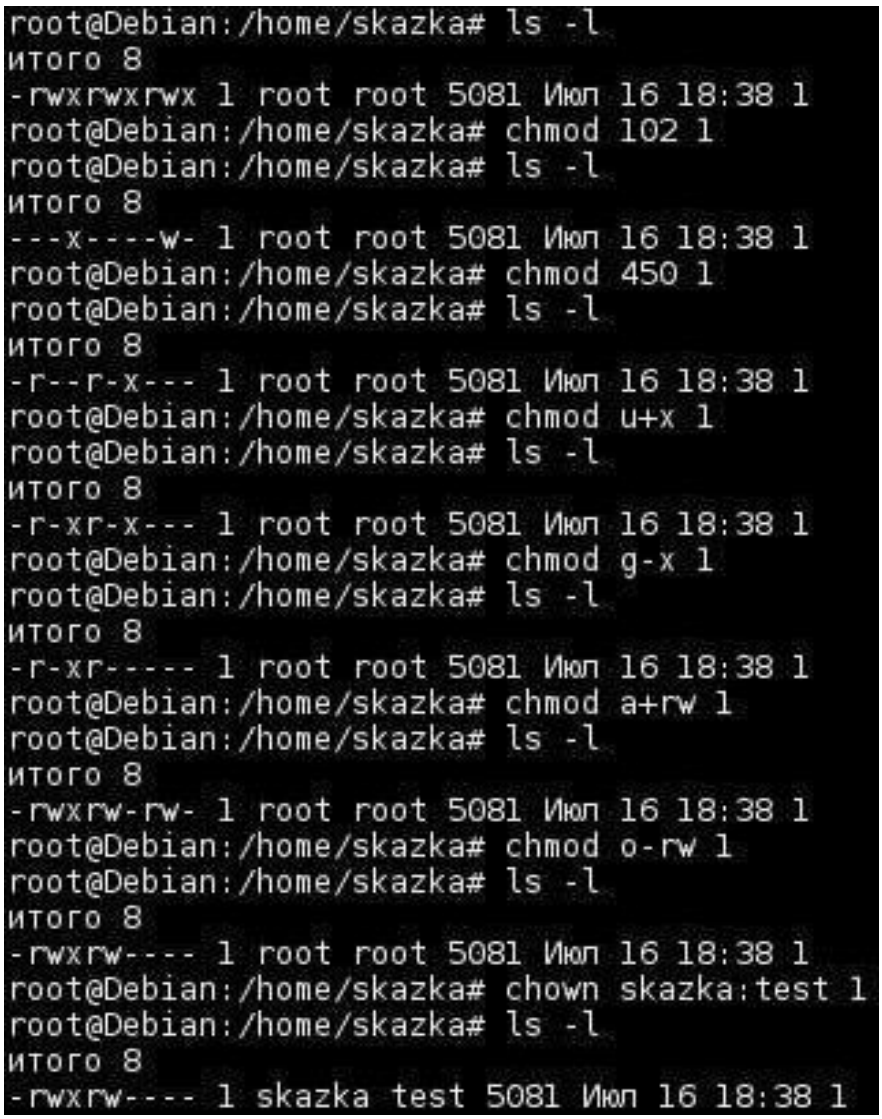

Рисунок 13 – Примеры смены прав доступа на объект

2. В операционной системе *Windows* реализована функция отслеживания важных событий, которые происходят в работе системных программ. Под понятием «события» подразумеваются любые происшествия в системе, которые фиксируются в специальном журнале и сигнализируют о себе пользователям или администраторам. Это может быть служебная программа, не желающая запускаться, сбой в работе приложений или некорректная установка устройств. Все происшествия регистрирует и сохраняет журнал событий *Windows*. Он также располагает и показывает все действия в хронологическом порядке, помогает производить системный контроль, обеспечивает безопасность операционной системы, исправляет ошибки и диагностирует всю систему.

Следует периодически просматривать этот журнал на предмет появления поступающей информации и производить настройку системы для сохранения важных данных.

Программа, отвечающая за регистрацию происшествий, запускается нажатием кнопки «Пуск» - «Панель управления» - «Администрирование» - «Просмотр событий» (рисунок 14).

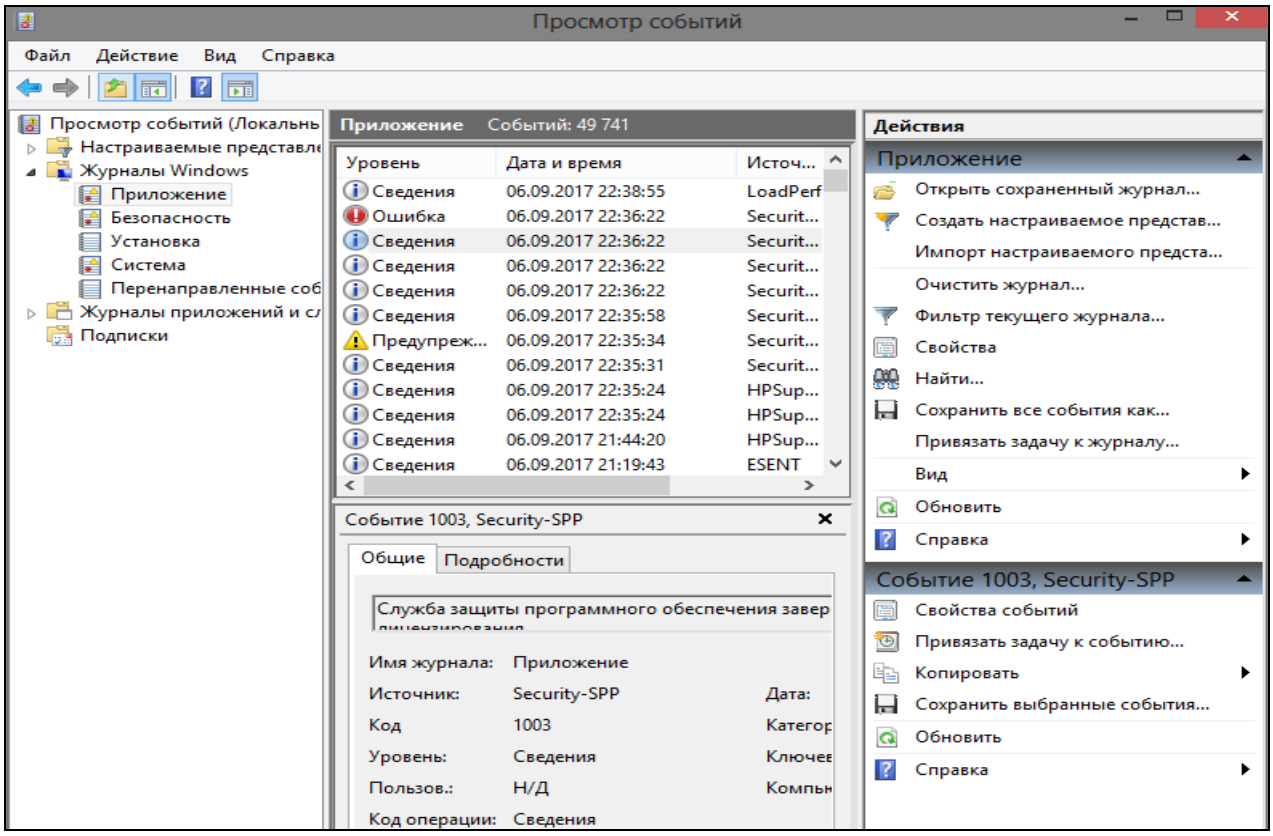

Рисунок 14 – Утилита после запуска

В *Windows* установлены два вида журналов событий: системные архивы и служебный журнал приложений. Первый для фиксации общесистемных происшествий, связанных с производительностью различных приложений, запуском и безопасностью. Второй записывает события их работы. Для контроля и управления всеми данными служба «Журнал событий» использует вкладку «Просмотра», которая подразделяется на следующие пункты:

1) Приложение – хранятся события, которые связаны с какой-то определенной программой. Например, почтовые службы хранят историю пересылки информации, различные события в почтовых ящиках.

2) Безопасность - сохраняет все данные, относящиеся к входам в систему и выходу из нее, использованию административных возможностей и обращению к ресурсам.

3) Установка – в этот журнал событий заносятся данные, которые возникают при установке и настройке системы и ее приложений.

4) Система – фиксирует все события операционки такие, как сбой при запуске служебных приложений или при установке и обновлении драйверов устройств, разнообразные сообщения, касающиеся работы всей системы.

5) Перенаправленные события – если этот пункт настроен, то в нем хранится информация, которая приходит с других серверов.

Очень важным моментом в предохранении системы от сбоев и зависаний является периодическое просматривание журнала «Приложение», в котором фиксируются сведения о происшествиях, недавних действиях с той или иной программой, а также предоставляется выбор доступных операций. Зайдя в журнал событий *Windows*, в подменю «Приложение» можно увидеть список всех программ, вызвавших различные негативные события в системе, время и дату их появления, источник, а также степень проблемности. В этой консоли можно сохранить все события за последние несколько месяцев, очистить журнал от старых записей, изменить размер таблицы.

Следует научиться применять это с этим полезным приложением «Планировщик задач». Для этого необходимо правой клавишей мыши кликнуть на любое происшествие и в открывшемся окне выбрать меню привязки задачи к событию. В следующий раз, когда произойдет такое происшествие, ОС запустит установленную задачу на обработку ошибки и ее исправление (рисунок 15).

Журнал событий – способ, позволяющий программам и системе фиксировать и сохранять все события на компьютере в одном месте. В таком журнале хранятся все операционные ошибки, сообщения и предупреждения системных приложений.

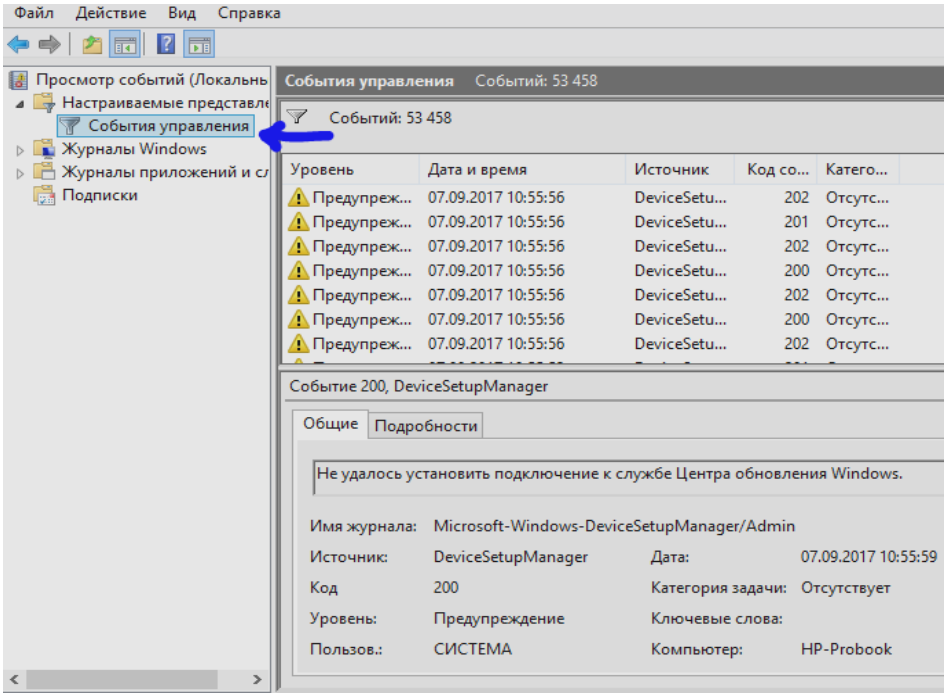

Рисунок 15 – Установка реакции на событие

5.3 Список контрольных вопросов

1 Как используются цифровые и буквенные назначения прав объектам системе в *Linux?*

2 Какие существуют виды журналов событий в *Windows*?

3 Зачем изучать работу Журнала событий?

6 Лабораторная работа №6. Структура реестра ОС. Изучение работы реестра систем

Цели работы: приобретение навыков работы по работе с реестром *Windows*; изучение структуры реестра ОС.

6.1 Рабочее задание

При выполнении этой лабораторной работы надо работать на виртуальной машине *Windows*. Перед началом работы рекомендуется сделать снимок экрана (*snapshot*), чтобы была возможность «откатиться» в случае возникновения неполадок в системе. Необходимо выполнить задания:

1) Познакомиться со сторонними утилитами для работы с реестром *Process Monitor* и *Registry Monitor (regmon*), их необходимо скачать. Установить дополнительно еще 2 программы работы с реестром и немного поработать и в них. Сделать анализ проделанной работы.

2) Познакомиться с принципами использования программ редактирования реестра *regedt.exe (regedit32.exe*) и найти информацию, чтобы:

- обеспечить ускоренную загрузку ОС;
- обеспечить более быстрое завершение ОС;
- поставить запрет и разрешение расширенного режима *CMD*;
- изменение раскладки клавиатуры, по умолчанию, с русского языка на английский;
- отключить возможность создания ярлыков;
- запретить пользователям запуска диспетчера задач;
- увеличить или уменьшить скорость вывода всплывающих меню;
- скрыть значки дисков в окне «Мой компьютер»;
- при входе в систему переключатель *Num Lock* включен;
- запретить пользователю запуск заданного им списка программ;
- обеспечить автоматическую выгрузку *DLL*-библиотек;
- разрешить ОС автоматически завершать зависшие программы;
- запретить командную строку;
- скрыть доступ к апплету «Установка/ Удаление программ» в Панели управления;
- запретить запуск апплета «Экран» в Панели управления;
- очистить файл подкачки *Windows*;
- изменить порог выдачи предупреждения о недостатке свободного места на диске.

6.2 Методические указания к выполнению лабораторной работы

Реестр *Windows* представляет собой реляционную базу данных, в которой аккумулируется вся необходимая для нормального функционирования компьютера информация о настройках операционной системы, программном обеспечении и оборудовании. Все хранящиеся в

реестре данные представлены в стандартизированной форме и четко структурированы согласно предложенной разработчиками *Windows* иерархии.

В случае установки или удаления каких-либо устройств, приложений или системных компонентов информация о подобных изменениях записывается в реестр и считывается оттуда в ходе каждой загрузки операционной системы. Реестры разных версий *Windows* имеют различия.

Редактирование реестра *Windows* позволяет:

– решать проблемы, возникающие в процессе эксплуатации прикладного программного обеспечения, гибко настраивать режимы работы приложений;

– устранять неполадки в работе оборудования, вызванные некорректным использованием ресурсов ОС или драйверов устройств;

– настраивать параметры и ограничения пользовательской среды *Windows*, изменять заданные, по умолчанию, характеристики ОС;

– управлять быстродействием компьютера;

– перераспределять ресурсы ОС по усмотрению администратора;

– управлять конфигурацией компонентов *Windows* и системных сервисов, что позволяет оптимизировать работу операционной системы, в зависимости от назначения компьютера и стоящих перед пользователем задач.

Структура реестра 64-разрядной версии ОС *Windows* несколько отличается от архитектуры реестра 32-разрядных версий ОС *Windows*. Реестр 64-*bit* имеет два независимых раздела: в одном содержатся данные, относящиеся к 32-разрядным компонентам операционной системы, в другом - все сведения по 64-разрядным компонентам, - причем ключи и ветви обоих разделов имеют практически одинаковые наименования и обозначения. В комплекте поставки ОС *Windows* 64-bit имеется две версии Редактора реестра: одна, запускаемая по умолчанию, демонстрирует только 64-разрядный раздел реестра ОС, а другая предназначена для редактирования 32-разрядного раздела. Для того чтобы запустить на компьютере, работающем под управлением 64-разрядной версии ОС *Windows,* 32-разрядную версию Редактора реестра, необходимо закрыть окно 64-разрядной версии Редактора, если эта программа была запущена ранее, поскольку оба эти приложения не могут работать одновременно.

В *Windows* реестр хранится в бинарном (двоичном) виде, поэтому для ручной работы с ним необходима специальная программа - редактор реестра - это *Regedit.exe*, в более ранних версиях *Windows* ими являются *Regedit.exe* и *Regedt32.exe*, имеющий дополнительные возможности работы с реестром (рисунок 16). Есть и другие программы, в том числе и консольные (*Reg.exe*).

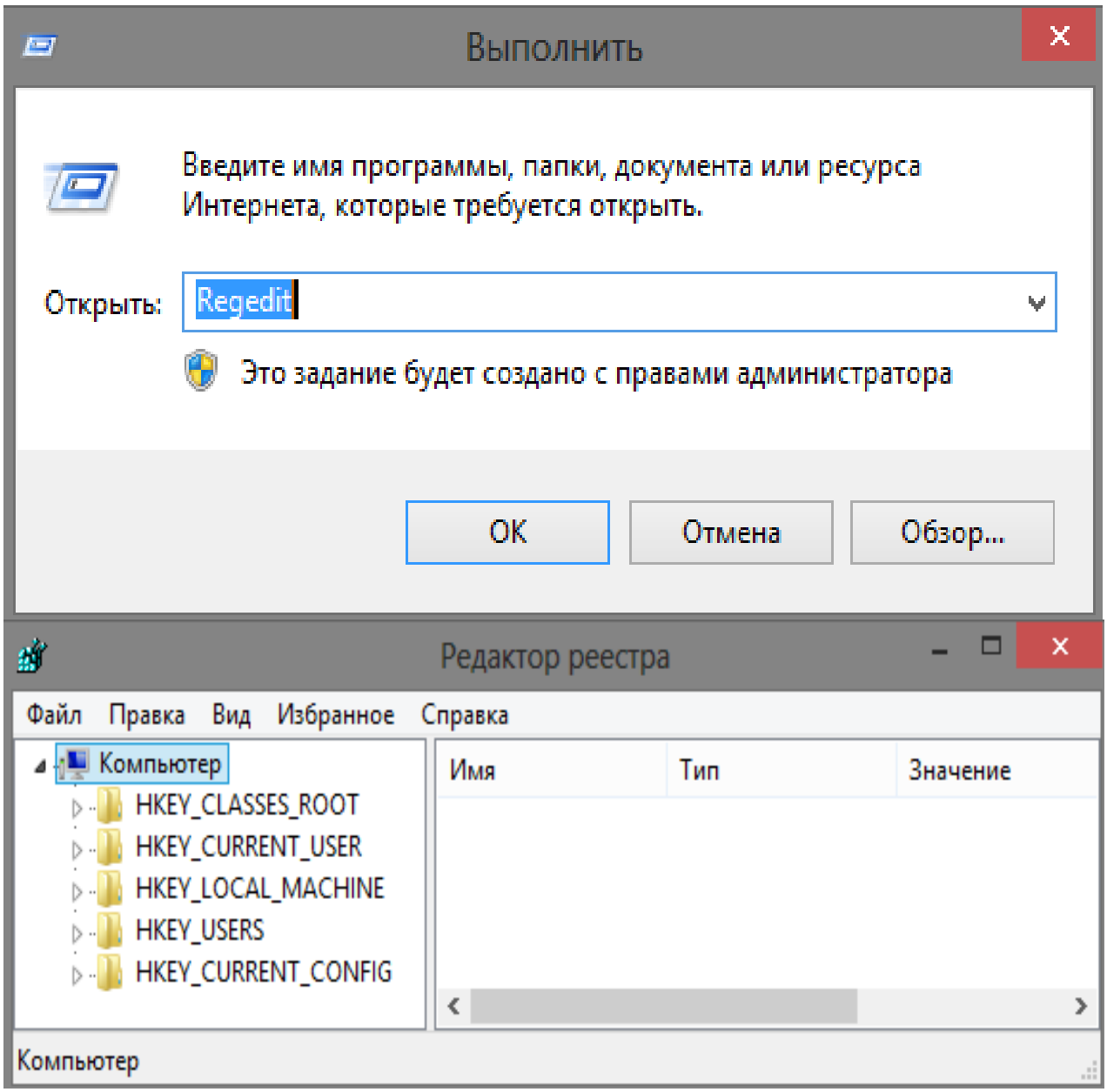

Рисунок 16 – Вызов и вид программы для работы с реестром

Реестр *Windows* имеет многоуровневую структуру, состоящую из четырех ступеней. К первой, самой верхней в иерархии реестра, ступени относятся так называемые ветви (*Hive Keys*), или ульи, которые принято обозначать по их английскому наименованию аббревиатурой *HKEY_*, где за символом подчеркивания следует обычно название самой ветви.

В реестре *Windows* существует пять ветвей:

1) HKEY_CLASSES_ROOT или *HKCR* сопоставляет расширения файлов и идентификаторы классов *OLE*. Фактически он указывает на *HKLM\Software\Classes*. Система использует эти соответствия, чтобы определить, какие приложения или компоненты нужно использовать при открытии или создании тех или иных типов файлов или объектов данных.

2) HKEY_CURRENT_USER или *HKCU* указывает на профиль текущего пользователя (вошедшего в данный момент в систему) внутри *HKU*. *Microsoft*

требует, чтобы приложения хранили все предпочтения пользователей в подразделах под *HKCU*. Например, *HKCU\Software\Microsoft\Windows\ CurrentVersion\Applets\Paint* содержит личные настройки пользователей программы *Paint*.

3) *HKEY_LOCAL_MACHINE* или *HKLM* хранит все настройки, относящиеся к локальному компьютеру. Приложения должны хранить здесь данные только в том случае, когда они относятся ко всем, кто пользуется компьютером. Например, драйвер принтера может хранить здесь набор настроек принтера, применяемых, по умолчанию, и копировать эти данные для каждого профиля пользователя при входе пользователя в систему.

4) *HKEY_USERS* или *HKU* содержит записи для каждого из пользователей, когда-либо входивших в систему. Владельцем каждой из этих записей является соответствующая пользовательская учетная запись, там содержатся настройки профиля этого пользователя. Если используются групповая политика, то задаваемые в ней настройки применяются здесь к профилям отдельных пользователей.

5) *HKEY_CURRENT_CONFIG* или *HKCC* хранит информацию о текущей загрузочной конфигурации компьютера. В частности, здесь хранится информация о текущем наборе системных служб и об устройствах, имевшихся во время загрузки. На самом деле, этот корневой раздел является указателем на раздел внутри *HKLM*.

Второй ступенью в иерархической системе реестра являются так называемые разделы, или ключи (*Keys*). В *Windows* нет какого-либо единого стандарта в обозначении ключей системного реестра, поэтому их имена были назначены разработчиками, исходя из типа данных, представленных внутри ключа. Ключи отображаются в программе Редактор реестра в виде подпапок ветвей *HKEY_*. Следует понимать, что не существует также каких-либо жестких ограничений, сопоставляющих ключам строго определенный тип данных. Иными словами, ключи в иерархии реестра служат исключительно для облегчения доступа к информации и являются одним из средств ее упорядочения. Функционально ключи можно разделить на две условные категории: определяемые системой, то есть те, имена которых назначены операционной системой, причем изменение этих имен может привести к отказу или сбоям в работе *Windows*, и определяемые пользователем - имена этих ключей могут быть изменены администратором компьютера, и такие изменения не приведут к каким-либо фатальным последствиям.

Реестр является настоящей базой данных, поэтому в нем используется технология восстановления, похожая на используемую в *NTFS*. *LOG*-файлы содержат журнал транзакций, который хранит все изменения. Благодаря этому реализуется атомарность реестра, то есть в данный момент времени в реестре могут быть либо старые значения, либо новые, даже после сбоя. В отличие от *NTFS*, здесь обеспечивается сохранность не только структуры реестра, но и данных.

Реестр или его отдельные части можно экспортировать в текстовые *reg*-файлы, редактировать в блокноте, а затем импортировать обратно. Некорректное изменение хранящейся в реестре информации способно нарушить работоспособность *Windows*. Достаточно допустить ошибку в записи значения какого-либо ключа или параметра, и пользователь больше не сможет загрузить компьютер. Тогда спасти положение может только восстановление последней работоспособной копии. Именно по этой причине разработчики *Windows* заметно ограничили доступ к реестру и редактировать параметры реестра, касающиеся безопасности, могут только пользователи, имеющие в системе учетную запись администратора.

Для создания архивной копии или восстановления реестра можно вызвать утилиту из меню «Служебные» - «Архивация данных». После запуска утилиты в окне необходимо выбрать кнопку «Диск аварийного восстановления». Будет выдано сообщение, в котором необходимо поставить галочку «Архивировать реестр». Реестр будет скопирован в папку *%systemroot%\repair\RegBack*, где *%systemroot%* - папка, куда установлена *Windows*. Будет создана резервная копия реестра.

Управление протоколированием с помощью ключей реестра:

1) Очистка файла подкачки.

Файл подкачки pagefile.sys находится в корне каждого или только системного диска. Там могут оставаться пароли к различным ресурсам и другая конфиденциальная информация. Для очистки данного файла после завершения работы устанавливается параметр типа *DWORD ClearPageFileAtShutdown*=1 в разделе *[HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\ Control\Session Manager\Memory Management*]. Установки вступят в силу после перезагрузки системы.

2) Автоматическое удаление временных файлов после работы в Интернет *[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Current Version\Internet Settings\Cache]* Значение ключа *Persistent*=0 заставит *Internet Explorer* удалять все временные файлы.

3) Отменить сохранение списка документов, с которыми вы работали *NoRecentDocsHistory*=1 в *[HKEY_LOCAL_MACHINE\Software\Microsoft\ Windows\CurrentVersion\Policies\ Explorer]*

4) Отмена сохранения информации о действиях пользователя

Ключ *NoInstrumentation*=1 запрещает записывать с какими приложениями недавно работал пользователь и к каким документам получал доступ: *[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ Policies\Explorer]*

5) Пароли и безопасность: рассматриваемые настройки хранятся в ветви *[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Network]*. Все ключи имеют тип *DWORD*, если это не обговорено отдельно; значение ключа, равное 1, включает данную опцию, 0 выключает.

1. Звездочки в паролях. При попытке доступа к защищенному паролем ресурсу *Windows* не скрывает пароль, который вы вводите. Параметр

HideSharePwds=1 определяет скрывать пароли к расшаренным ресурсам звездочками.

2. Запрет на доступ к файлам и принтерам: для запрещения доступа к файлам служит ключ *NoFileSharing*, а для запрета управления доступом к файлам - ключ *NoFileSharingControl*. Запрет доступа к принтерам устанавливается ключом *NoPrintSharing*. Ключ *NoPrintSharingControl* устанавливает запрет на управление доступом к принтерам.

3. Запрет перечисления рабочей группы

Для того чтобы запретить перечисление содержимого рабочей группы, надо установить значение ключа *NoWorkgroupContents* равным 1. Даже при запрете пользователи могут подключаться к компьютерам в своей рабочей группе или домене. Для этого необходимо набрать полное сетевое имя разделенного ресурса в формате *UNC*, в диалоговых окнах команд «Выполнить» или «Подключить сетевой диск».

4. Запрет доступа для анонимных пользователей: анонимный пользователь может получить доступ к списку пользователей и открытых ресурсов. Чтобы это запретить, можно воспользоваться данным ключом: *[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\LSA]*
RestrictAnonymous=1

5. Включить режим, при котором в режиме обзора сети другие пользователи не будут видеть компьютера: *[HKEY_LOCAL_MACHINE\ SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters] Hidden=1*

6. Автоматический вход в систему без ввода имени и пароля - создать или отредактировать в разделе реестра *[HKEY_LOCAL_MACHINE\ SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon*] строковые параметры «*DefaultDomainName», «DefaultUserName», «DefaultPassword*» в качестве их значений укажите требуемые для входа в систему имя домена, имя пользователя и пароль соответственно. Создать или отредактировать в этом же разделе строковый параметр «*AutoAdminLogon»* для автоматического входа в систему, присвоив ему значение 1. 0 отменяет автоматический вход. Пароль сохраняется как текст, поэтому любой пользователь, имеющий доступ к системному реестру, может увидеть заданный, по умолчанию, пароль, но доступ к реестру можно запретить.

7. Пароль после ждущего режима: параметр типа *DWORD PromptPasswordOnResume*=1 в [HKEY_CURRENT_USER\Software\Policies\ Microsoft\Windows\System\Power].

8. Запрет на доступ к содержимому выбранных дисков: можно не скрывать сами значки дисков, но запретить пользователю доступ к файлам заданных дисков через Проводник, Мой компьютер, Выполнить или команду *Dir*. Для этого необходимо создать параметр *NoViewOnDrive* типа *DWORD* в разделе *[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\ Policies\Explorer*], содержащий битовую маску для дисков. Например, диск *A* имеет бит 1, диск *C* - 4, диск *D* - 8. Таким образом, чтобы скрыть диски *A* и *D*, нужно сложить их значения 1 (*A*) + 8 (*D*) и установить значение 9. Список

всех дисков: *A: 1, B: 2, C: 4, D: 8, E: 16, F: 32, G: 64, H: 128, I: 256, J: 512, K: 1024, L: 2048, M: 4096, N: 8192, O: 16384, P: 32768, Q: 65536, R: 131072, S: 262144, T: 524288, U: 1048576, V: 2097152, W: 4194304, X: 8388608, Y: 16777216, Z: 33554432, Все диски: 67108863*

9. Установка способа доступа к расшаренным ресурсам компьютера из сети: позволяет другому пользователю получить информацию о доступных для общего пользования директориях и об имеющихся на компьютере локальных пользователях. Раздел: *[HKLM\System\CurrentControlSet\Control\Lsa*], параметр *restrictanonymous*, тип *DWORD*. Если значение равно 1 - запрещает анонимным пользователям просматривать удаленно учетные записи и расшаренные ресурсы, 2 - отказывает любой неявный доступ к системе (в сетевом окружении компьютер не будет виден, однако, доступ к нему можно будет получить, обратившись по его *IP*). Установки вступят в силу после перезагрузки системы.

6) Работа с реестром.

1. Размер реестра ограничивается параметром *RegistrySizeLimit* (тип *REG_DWORD*) в разделе реестра [*HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control*].

2. Запрещение запуска редактора реестра: *[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVerson\Policies\System]* параметр *DisableRegistryTools*=1 типа *DWORD*. Запуск редактора реестра будет запрещен, однако, останется возможность вносить изменения с помощью программного обеспечения сторонних разработчиков и с помощью *REG*-файла.

7) *CMD*.

1. Запрет на расширенный режим командного процессора *CMD.EXE*. Например, в расширенном режиме существуют такие команды как *del, erase, chdir, goto*. Для запрета создается параметр типа *DWORD EnableExtensions*=0 в разделе [*HKCU\Software\Microsoft\Command Processor*].

2. Запрет на режим командной строки и обработки *bat*-файлов. Для этого в *[HKCU\Software\Policies\Microsoft\Windows\System]* надо создать параметр типа *DWORD DisableCMD,* который может принимать значения:

- 0 (или отсутствие записи в реестре) - система может использовать режим командной строки и обрабатывать *bat*-файлы;
- 1 - система не может использовать режим командной строки, но может обрабатывать *bat*-файлы;
- 2 - система не может использовать режим командной строки и обрабатывать *bat*-файлы.

8) Изменение порога выдачи предупреждения о недостатке свободного места на диске. Если на диске остается свободным менее 10%, по умолчанию, места, то система информирует об этом появлением иконки в области уведомления. Можно изменить порог в процентном соотношении параметром типа *DWORD DiskSpaceThreshold*, в котором указывается значение от 0 до 99

(т.е. процент от объема диска) в разделе *[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanmanServer\Parameters]*.

9) Отключение сообщения о недостатке свободного места на диске *[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer]* ключ типа *DWORD NoLowDiskSpaceChecks*=1.

10) Отключение (запрет) *Task Manager: [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System]* создается ключ типа *DWORD* под названием *DisableTaskMgr*, значение 1. Удалив этот ключ или присвоив ему 0, можно вновь разрешить *Task Manager*.

11) Завершение задач, которые «повисли» и перестали отвечать можно настроить таким образом, чтобы они закрывались автоматически и, значит, значение ключа надо выставить соответствующим *HungAppTimeout* в разделе [*HKCU\Control Panel\Desktop*], определяющее время в миллисекундах (время, через которое не отвечающее приложение считается зависшим). Кроме этого ключа, в той же ветке есть ключи *WaitToKillServiceTimeout*, который задаёт время ожидания перед «убийством» зависшей службы, и *AutoEndTasks*, присвоив которому значение 1, можно разрешить системе убивать зависшие процессы самостоятельно. Не следует ставить очень малые значения *Timeout*, поскольку могут возникнуть проблемы с невовремя закрытыми программами и службами.

12) Автоматическое включение *NumLock* при загрузке делается строковым значением *InitialKeyboardIndicators*, равным 2 в *[HKEY_CURRENT_USER\Control Panel\Keyboard]*.

13) Ненужные записи в блоке «Установка/Удаление программ». Все программы должны включать программу *Uninstall*, иногда программа может удалиться, но запись в блоке «Установка/Удаление программ» остается: *[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall]* - под этим ключом ряд подключей, каждое из которых представляет установленную программу. Параметры *DisplayName* и *UninstallString* - названия, используемые в списке программ «Установка/Удаление» и программа, используемая для деинсталляции.

14) Запрещение запуска программ. *Windows* позволяет ограничить доступ к программам, кроме разрешенных в специальном списке. Для ограничения запускаемых программ надо открыть раздел *[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVerson\Policies\Explorer]* и создать там ключ *RestrictRun*=1 типа *DWORD*. Затем тут же надо создать подраздел с аналогичным именем *RestrictRun* и в нем перечислить список разрешенных к запуску программ для текущего пользователя. Записи в этом подразделе пронумеровываются, начиная с 1, и содержат строки с путями (необязательно) и именами приложений. Файлы должны быть с расширением. Необходимо указать файл *Regedit.exe*, иначе невозможно будет запустить редактор реестра. Для сброса ограничения на запуск программ надо установить значение ключа *RestrictRun* в 0.

15) Автоматическая выгрузка *DLL*. Оболочка *Windows* выгружает неиспользуемые *DLL* не сразу, а через некоторое время. Этот промежуток времени иногда может достигать больших интервалов (особенно при отладке программ). Для автоматической выгрузки всех *DLL* в разделе *[HKEY_LOCAL_MASHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer]* надо установить значение строкового параметра *AlwaysUnloadDLL,* равным 1.

16) Можно ускорить действие файловой системы, увеличив параметр типа *DWORD IoPageLockLimit* в разделе *[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management]* от заданных по умолчанию 512 КБ до 4 МБ и более (таблица 1). Этот параметр представляет максимальное число байт, которые могут быть блокированы для операций ввода/вывода. Когда значение параметра равно 0, то система использует встроенный алгоритм определения необходимой памяти и использует объем 512 КБ. Установка максимального значения должна основываться на объеме памяти системы. Установки вступят в силу после перезагрузки системы.

Таблица 1 – Параметры ускорения действие файловой системы

RAM (MB)	*IoPageLockLimit*
32	4096000
64	8192003
128	16384000
256+	65536000

17) Управление размером файла *SHELLICONCACHE*. *Windows* хранит некоторые значки, используемые оболочкой, в файле *SHELLICONCACHE*. При частом изменении параметров оболочки размер данного файла увеличивается, что приводит постепенно к тормозам при перерисовке значков из-за отсутствия места в кэше. Можно увеличть размер кэша этого файла, устанавливая строковый параметр *MaxCachedIcons* равным 5000 (5 Мбайт) в разделе *[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer*]. По умолчанию, этот параметр не присутствует в реестре. Рекомендуется иногда удалять файл *SHELLICONCACHE*, особенно, когда вы начинаете замечать, что значки становятся черными на Рабочем столе.

18) Скрытие «Свойств папки» делается с помощью *[HKEY_CURRENT_USER\Software\Microsoft\Windows\Current Version\Policies\Explorer]* - создать параметр типа *DWORD* с названием *NoFolderOptions* и значением 1. После перезагрузки «Свойств папки» в проводнике не будет.

19) Запреты: добавление ключа типа *DWORD* со значением 1 по адресу *[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer]* включает опцию, удаление или «0» отключает меню «Пуск»:

– *NoRecentDocsMenu* - Скрыть «Документы»;

– *NoFavoritesMenu* - Скрыть «Избранное»;

– *NoRun* Скрыть - «Выполнить»;
– *NoClose* Скрыть - «Завершение работы»;
– *NoLogoff* Скрыть - «Завершение сеанса»;
– *NoWindowsUpdate* - скрыть «*Windows Update*»;
– *NoSetFolders* - скрыть «Настройка»;
– *NoSetTaskbar* - запретить настройку Панели задач.

Меню «Пуск», подменю «Настройка»:

– *NoSetActiveDesktop* - Скрыть «Рабочий стол *Active Desktop»;*
– *NoSetFolders* - скрыть «Принтеры» и «Панель управления»;
– *NoSetTaskbar* - скрыть «Панель задач» и меню «Пуск»;
– *NoNetworkConnections* - скрыть «Удаленный доступ к сети.

Меню «Пуск», подменю «Документы»:

– *NoRecentDocsHistory* - не помнить недавно открытых документов;
– *MaxRecentDocs* - количество недавно открытых документов;
– *ClearRecentDocsOnExit* - очистить список недавно открытых документов при выходе;
– *NoSMDocs* - скрыть «Мои документы»;
– *NoSMMyPictures* - скрыть «Мои рисунки».

Меню «Пуск», подменю «Настройка», пункт «Принтеры»:

– *NoPrinterTabs* - скрыть вкладки в диалоге «Свойства: принтер»;
– *NoDeletePrinter* - запретить удаление принтера;
– *NoAddPrinter* - запретить добавление принтера.

«Рабочий стол»:

– *NoDesktop* - отключить «Рабочий стол»;
– *NoStartBanner* - отключить «Начните работу с нажатия этой кнопки»;
– *NoActiveDesktop* - отключить *Active Desktop;*
– *NoActiveDesktopChanges* - запретить изменения на *Active Desktop.*

«Разное»:

– *NoSaveSettings* - запретить сохранение установок;
– *NoDrives* - скрыть все диски в «Мой компьютер» (*FF FF FF FF*). В зависимости от значения скрываются разные буквы дисков, 00 00 00 00 - не скрыт ни один.

«Панель управления», диалог «Сеть» *[HKEY_CURRENT_USER\Software\ Microsoft\Windows\CurrentVersion\Policies\Network]:*

– *NoNetSetup* - диалог «Сеть» недоступен;
– *NoNeSetupIDPage* - вкладка «Идентификация» недоступна;
– *NoNetSetupSecurityPage*- вкладка «Управление доступом» недоступна;
– *NoEntireNetwork* - скрыть «Вся сеть»;
– *NoWorkgroupsContents* - скрыть содержимое сети.

«Панель управления», диалог «Пароли» *[HKEY_CURRENT_USER\ Software\ Microsoft\Windows\CurrentVersion\Policies\System]:*

– *NoSecCPL* - диалог «Пароли» недоступен;

– *NoPwdPAGE* - вкладка «Смена паролей» недоступна;

– *NoAdminPage* - вкладка «Удаленное администрирование» недоступна;

– *NoProfilPage* - вкладка «Профили пользователей» недоступна.

«Панель управления», диалог «Экран»:

– *NoDispCPL* - диалог «Экран» недоступен;

– *NoDispAppearancePage* - вкладка «Оформление» недоступна;

– *NoDispBackgroundPage* - вкладка «Фон» недоступна;

– *NoDispScrSavePage* - вкладка «Заставка» недоступна;

– *NoDispSettingsPage* - вкладка «Настройка» недоступна.

20) Отключение автоматической проверки апдейтов можно параметром типа *DWORD EnableBalloonTips*=0 в разделе *[HKEY_CURRENT_USER/ Software/Microsoft/Windows/CurrentVersion/Explorer/Advanced]*.

21) Связанные документы. При перемещении или удалении *html*-документа будут также перемещены или удалены и сопоставленные с этим документом файлы, которые содержатся в папке ИмяДокумента.*files*. Если необходимо отключить подобное поведение, то надо создать параметр типа *DWORD NoFileFolderConnection*=1 в разделе *[HKEY_CURRENT_USER\ Software\Microsoft\Windows\CurrentVersion\Explorer]*.

22) Проводник выступает и в качестве оболочки *Windows* и в качестве файл-менеджера. На использовании памяти это сказывается не лучшим образом. При нормальных условиях Проводник отнимает целых 8 Мбайт памяти *Windows*. Из-за проблем с выделением памяти *Windows* использует двойную квоту памяти для Проводника, считая ее используемой разными программами. Чтобы избавиться от этой проблемы, надо запустить Проводник как два отдельных процесса вместо одного. Для этого нужно изменить значение параметра типа *DWORD SeparateProcess* ("0" - один процесс, "1" - два процесса) в разделе *[HKEY_CURRENT_USER\Software\Microsoft\ Windows\CurrentVersion\Explorer\Advanced]*.

23) Изменение раскладки клавиатуры, по умолчанию. В разделе [*HKU\DEFAULT\Keyboard Layout\Preload*] на первую позицию надо поместить желаемую раскладку - 00000409 (английская раскладка) или 00000419 (русская).

6.3 Список контрольных вопросов

1 Зачем редактировать реестр *Windows*?

2 Какие отличия для 32- и 64-разрядных версий *Windows*?

3 Каким образом можно восстановить первоначальные значения реестра?

4 Какие программы дополнительно можно использовать для работы по изменению реестра?

7 Лабораторная работа №7. Изучение базовых прав доступа. Переход в режим суперпользователя. Изучение базы данных пользователей. Добавление и удаление пользователей систем

Цели работы: приобретение навыков работы с ОС через терминал при управлении учетными записями.

7.1 Рабочее задание

Выполнить задания по управлению учетными записями в *Linux*:

1) Создать 2 пользователей в системе (по имени папы и мамы) с паролями, создать группу *Roditeli*, добавить обоих пользователей в эту группу, вывести по ним всевозможную информацию.

2) Под суперпользователем переопределить пароли для обеих учетных записей.

3) Обменяться сообщениями вышеуказанным пользователям.

4) Обменяться файлами вышеуказанным пользователям.

5) Удалить всех созданных пользователей и группу.

7.2 Методические указания к выполнению лабораторной работы

Командная строка начинается приглашением - подсказкой, что система готова принимать команды пользователя (рисунок 17). Приглашение может быть оформлено по-разному, но чаще всего оно заканчивается символом «$» (права обычного пользователя) или «#» (права суперпользователя).

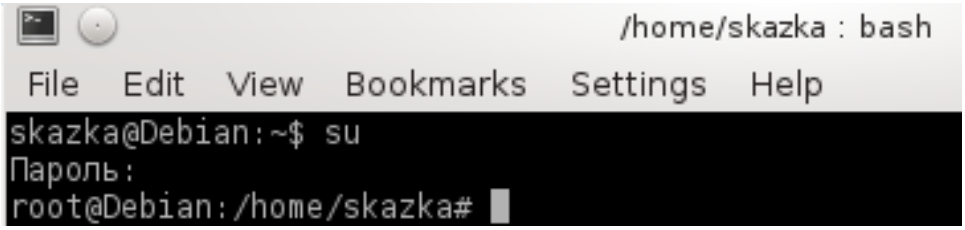

Рисунок 17 - Введенная команда su в строке приглашения позволяет залогиниться под суперпользователем

Например, в конструкции *abc@def:gh* до «@» - имя пользователя; между «@» и «:» - имя домена; после «:» - путь к папке, где находится текущий пользователь и в конце признак пользователя, например, запись *skazka@Debian:~$* означает: пользователь *skazka* сидит с доменным именем *Debian*, ~ - находится в своей домашней директории (/*home/skazka*), являясь обычным пользователем. Или *root@Debian:/ home/skazka#* означает: пользователь root сидит на машине с доменным именем *Debian*, находится в директории /*home/skazka*, являясь суперпользователем.

В консоли команды для работы с пользователями и группами пользователей:

- *id skazka* - информацию по пользователю *skazka* (логин, *UID, GID*);

- *finger skazka* - показать информацию о пользователе *skazka*;
- *last* - действия последних зарегистрированных пользователей;
- *who* - показывает имя текущего пользователя и время входа;
- *whoami* - показывает к какой группе относится текущий пользователь;
- *useradd skazka* - добавление нового пользователя *skazka* (или *adduser*);
- *groupadd pritcha* - добавление группы *pritcha*;
- *usermod -a -G pritcha skazka* - добавляет в группу *pritcha* пользователя *skazka* (для *Debian*-подобных ОС);
- *passwd skazka* - задание пароля пользователю *skazka*;
- *userdel skazka* - удаление пользователя *skazka*;
- *groupdel pritcha* - удаление группы *pritcha*;
- *login skazka* - залогиниться под пользователем *skazka*;
- *exit* - завершение сеанса текущего пользователя;
- *su* (или *sudo su*) - войти под администраторской учетной записью;
- *sudo nano* - запустить приложение *nano* от имени администратора (*sudo= substitute user and do)* позволяет пользователю выполнять указанные программы с административными привилегиями без ввода пароля суперпользователя);
- *wall* и *write* - отправляет на терминалы других пользователей сообщения. *write* позволяет общаться с другими пользователями путем копирования строк из Вашего терминала в их в режиме онлайн. Алгоритм действий: узнаем, кто в данный момент находится в системе и к какому терминалу подключен с помощью команды *who* (рисунок 18), далее отправляем сообщение пользователю *test*, например: *write test pts/2*. Когда мы нажмем *Enter*, наше сообщение будет отправлено в его терминал, *Ctrl+D*, чтобы прервать *write*.

Для отправки широковещательного сообщения всем подключенным пользователям, используется команда *wall* (*wall = write to all*), формат, как у команды *write*, сообщение будет отправлено после того, как нажмете *Ctrl+D* (рисунок 19).

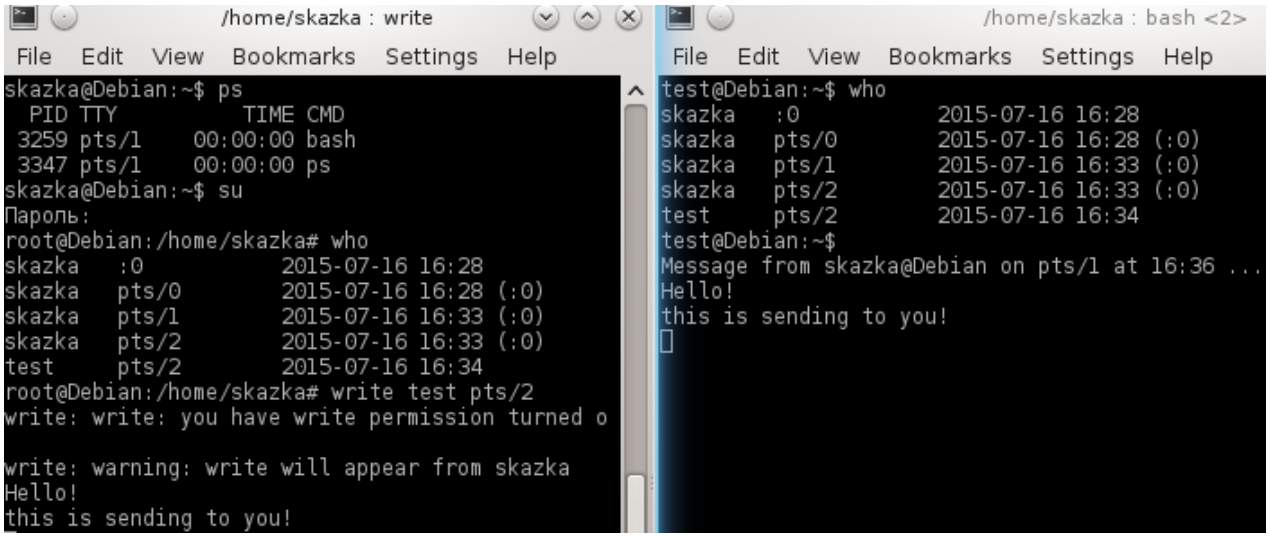

Рисунок 18 – Работа команды *write*

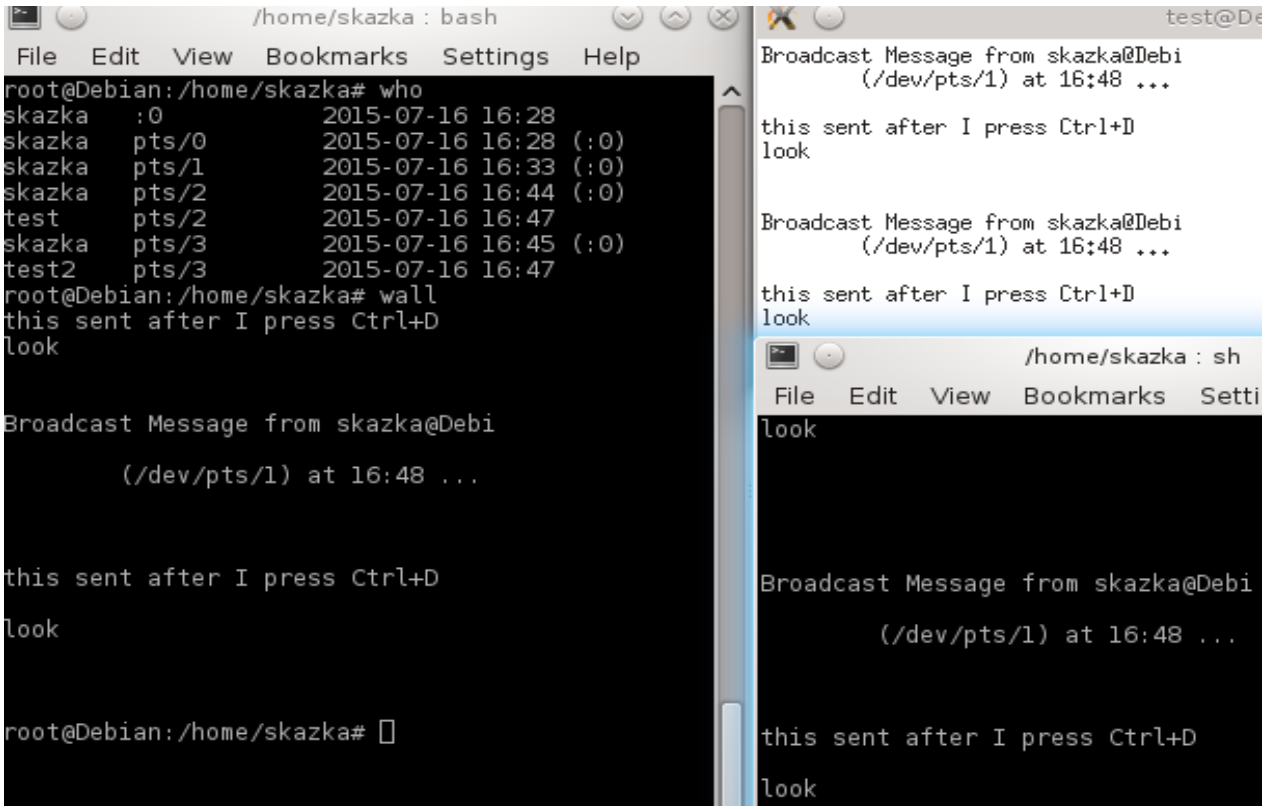

Рисунок 19 – Работа команды *wall*

Команды *write* и *wall* работают при передаче сообщений на терминалы других пользователей. При этом можно также передавать и файлы: *cat file.txt | write user pts/1* и *cat file.txt | wall.*

Все настройки пользователей и групп хранятся в текстовых файлах:

1) */etc/passwd*

Тут хранится вся информация о пользователях, кроме пароля, одна строка соответствует описанию одного пользователя. Содержание строки:

skazka:x:1000:1000:Yekaterina Zuyeva:/home/zuyeva:/bin/bash

Строка состоит из нескольких полей, каждое из которых отделено от другого двоеточием. Значение каждого поля приведено в таблице 2. Второе и последнее поля необязательные и могут не иметь значения.

Таблица 2 – Параметры файла */etc/passwd*

Поле	Описание
skazka	Имя пользователя для входа в систему
x	Необязательный зашифрованный пароль
1000	Числовой идентификатор пользователя (*UID*)
1000	Числовой идентификатор группы (*GID*)
Yekaterina Zuyeva	Поле комментария
/home/ zuyeva	Домашний каталог пользователя
/bin/bash	Оболочка пользователя

2) /etc/group

Тут хранится информация о группах, записанная в аналогичном с */etc/passwd* виде (таблица 3):

skazka:x:1000: *skazka,test*

В этом файле второе и четвертое поля могут быть пустыми.

Таблица 3 – Параметры файла */etc/group*

Поле	Описание
skazka	Название группы
x	Необязательный зашифрованный пароль
1000	Числовой идентификатор группы (*GID*)
skazka, test	Список пользователей, находящих в группу

3) /etc/shadow

Хранит в себе пароли и поэтому права, установленные на этот файл, не дают считать его обычному пользователю. Значение каждого поля приведено в таблице 4. Пример одной из записей файла:

skazka:6Yvp9VO2s$VfI0t.o754QB3HcvVbz5hlOafmO.LaHXwfavJHniHNzq/bCI3AEo562hhiWLoBSqxLy7RJJNm3fwz.sdhEhHL0:15803:0:99999:7:::

Таблица 4 – Параметры файла */etc/group*

Поле	Описание
skazka	Имя пользователя для входа в систему
6Yvp9VO2s$VfI0t.o754QB3HcvVbz5hlOafmO.LaHXwfavJHniHNzq/bCI3AEo562hhiWLoBSqxLy7RJJNm3fwz.sdhEhHL0	Необязательный зашифрованный пароль
15803	Дата последней смены пароля
0	Минимальный срок действия пароля
99999	Максимальный срок действия пароля
7	Период предупреждения о пароле
предпоследнее поле	Период неактивности пароля
последнее поле	Дата истечения срока действия записи

7.3 Список контрольных вопросов

1 Что содержат строка приглашения?
2 В каких файлах содержится информация о пользователях?
3 Какие команды для передачи на терминалы пользователей?

8 Лабораторная работа №8. Управление сетью. Сетевые службы. Изучение основных команд для работы в сети систем

Цели работы: отработка навыков работы по настройке программного обеспечения по шифрованию файлов и папок, разделов; изучение и конфигурация настроек инструментов настройки и управления сетевыми правилами.

8.1 Рабочее задание

1. Выполнить задания по шифрованию файлов, папок, раздела диска:

1) В каталоге /*home*/название_пользователя создать папку *Crypt*.

2) В ней создать 3 файла:

– *file1*-ваша_фамилия.*txt* (содержимое - текущее время и дата);

– *file2*-ваша_фамилия.*txt* (содержимое – содержимое текущей папки);

– *key*-ваша_фамилия.*txt* (пароль при шифровании файлов).

3) Зашифровать 2 файла через консоль:

– *file1*-ваша_фамилия.*txt* → *file3*-ваша_фамилия.*txt.encoded*;

– *file2*-ваша_фамилия.*txt* → *file4*-ваша_фамилия.*txt.encoded*;

– вывести содержимое папки на экран.

4) Расшифровать файлы через консоль:

– *file3*-ваша_фамилия.*txt.encoded* → *file5*-ваша_фамилия.*txt.decrypt*;

– *file4*-ваша_фамилия.*txt.encoded* → *file6*-ваша_фамилия.*txt.decrypt*;

– вывести содержимое папки на экран;

– сравнить через консоль содержимое *file1*-ваша_фамилия.txt с *file5*-ваша_фамилия.*txt.decrypt*; *file2*-ваша_фамилия.*txt* с *file6*-ваша_фамилия. *txt.decrypt*.

5) Создать внутри папки *Crypt* новую папку *CryptDir*, скопировать в нее файлы *file3*-ваша_фамилия.*txt.encoded*, *file6*-ваша_фамилия.*txt.decrypt*.

6) Написать про опции сжатия используемой утилиты.

7) Скачать любую программу для шифрования папок и зашифровать директорию *CryptDir*; расшифровать и сравнить результаты.

8) Скачав любую программу для шифрования дисковых разделов, зашифровать один из своих дисковых разделов.

2. Соединить ОС1 и ОС2 (железо с железом). Настроить между ними сеть тремя способами (с организацией общей папки и без нее):

1) Скопировать из ОС1 в ОС2 зашифрованные 3 файла:

– *file3*-ваша_фамилия.*txt.encoded*;

– *file4*-ваша_фамилия.*txt.encoded*;

– key-ваша_фамилия.*txt*.

2) Расшифровать в ОС2 полученные *file3* и *file4*. Результаты соединить в один файл *result*-ваша_фамилия.

3) Файл *result*-ваша_фамилия скопировать *result2*-ваша_фамилия, дописать в конец использованный ключ для расшифровки.

4) Скопировать из ОС2 в ОС1 файл *result*-ваша_фамилия.

5) Настроить правило в файерволе, не позволяющее копировать из ОС2 в ОС1, проверить это (пытаясь передать файл *result2*-ваша_фамилия в ОС1).

6) Убрать поставленное правило, проверить работоспособность.

3. Соединить ОС1 и ОС2 (виртуальные машины на одном компьютере). Настроить между ними сеть тремя способами (с организацией общей папки и без нее). Проделать пункты 2.1-2.6 аналогично на виртуальных машинах.

8.2 Методические указания к выполнению лабораторной работы

1. С появлением в 2.6-х ядрах низкоуровневого драйвера логических томов - *Device mapper* стала возможной работа не только с петлевыми устройствами, но и непосредственно с дисковыми разделами. Более того, благодаря модульной структуре *Device mapper*, ничто не мешает, комбинируя различные цели, организовать шифрование данных на *RAID*-массиве или создать зашифрованный своп. Цель для *Device mapper*, отвечающая за шифрование данных, называется *dm-crypt.*

Результат шифрования файла с помощью встроенной утилиты, а именно: *OpenSSL* представлен на рисунке 20. Команда в консоли, использованная при шифровании файла *file1-Zuyeva.txt*, находящегося в /*home/skazka/Crypt* будет следующей:

openssl enc -e -aes-256-cbc -in /home/skazka/Crypt/file1-Zuyeva.txt -out file3-Zuyeva.txt.encoded

После этого пароль вводится 2 раза для шифрования, опция *-enc* значит, что шифрование идет с помощью алгоритма (*AES256); -e* указывает на процедуру шифрования; *-in* указывает входящий файл; *-out* - выходящий файл.

Для дешифрования команда (параметр *-d* указывает на дешифровку):

openssl enc -d -aes-256-cbc -in /home/skazka/Crypt/file3-Zuyeva.txt.encoded -out file5-Zuyeva.txt.decrypt

Рисунок 20 – Результат шифрования *file1-Zuyeva.txt* с помощью *ssh*

2. *Соединение двух компьютеров посредством локальной сети.*

Способ №1.

Для передачи данных по локальной сети (при соединённом *Ethernet*-кабеле) нужно настроить *IP*-адресацию. В качестве примера использовался *IP*-адрес для сервера 192.168.1.203, с которого был загружен файл *shaika.txt* с зашифрованным содержимым (рисунок 21). Последовательность действий: настраивается *IP*-адрес, с клиента запускается пинг сервера: *ping* 192.168.1.203, пинг проходит успешно, значит устройство включено, и оно в сети.

```
c64 bytes from 192.168.1.203: icmp_seq=2 ttl=64 time=0.385 ms
64 bytes from 192.168.1.203: icmp_seq=3 ttl=64 time=0.388 ms
64 bytes from 192.168.1.203: icmp_seq=4 ttl=64 time=0.381 ms
ccc64 bytes from 192.168.1.203: icmp_seq=5 ttl=64 time=0.380 ms
^C
--- 192.168.1.203 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.380/0.392/0.430/0.031 ms
[root@localhost shaika]# ssh shaikenov@192.168.1.203:/home/shaikenov/shaika.txt
/home/shaika/
ssh: Could not resolve hostname 192.168.1.203:/home/shaikenov/shaika.txt: Name
r service not known
[root@localhost shaika]# scp shaikenov@192.168.1.203:/home/shaikenov/shaika.txt
/home/shaika/
The authenticity of host '192.168.1.203 (192.168.1.203)' can't be established.
ECDSA key fingerprint is SHA256:QotdDIhoPjNnXJo2BZZl3IUCYIWza7Um6Cte3Md8tjg.
ECDSA key fingerprint is MD5:08:19:c8:93:8f:9b:13:eb:f3:6c:7b:5b:45:be:86:bf.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.203' (ECDSA) to the list of known hosts.
shaikenov@192.168.1.203's password:
Permission denied, please try again.
shaikenov@192.168.1.203's password:
shaika.txt                                    100%    0     0.0KB/s   00:00
[root@localhost shaika]#
```

Рисунок 21 - Передача между двумя машинами с помощью протокола *ssh*

Способ №2.

Еще одним способом передачи файлов с одной ОС на другую является реализация *FTP* или TFTP-сервера. В качестве сервера выступает машина с *IP*-адресом 192.168.1.200. Для этой цели нужно скачать утилиту, которая позволяет обращаться к *TFTP*-серверу, например, *WinSCP* (рисунки 22, 23).

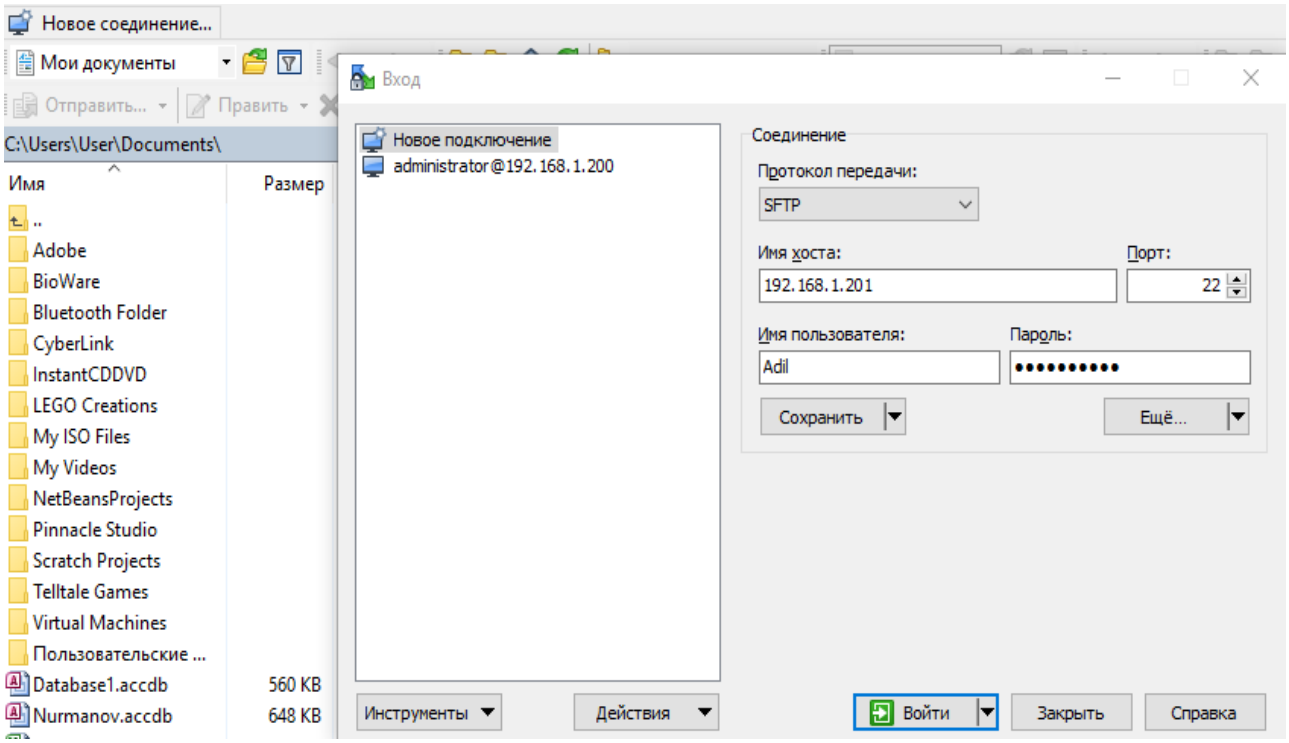

Рисунок 22 - Утилита *SFTP*

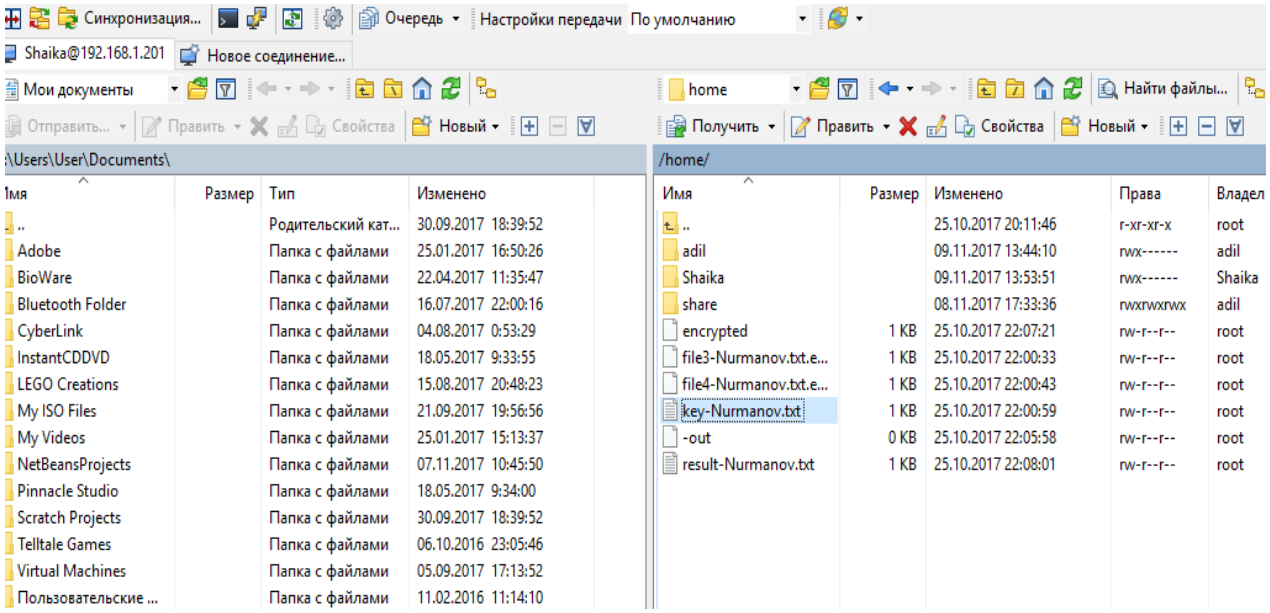

Рисунок 23 - Скачивание файлов с сервера

Способ №3.

Еще одним способом соединения является создание общей папки доступа. Осуществляется это с помощью доступа к папке (данный метод продемонстрирован благодаря возможностям Ubuntu на рисунках 24-25).

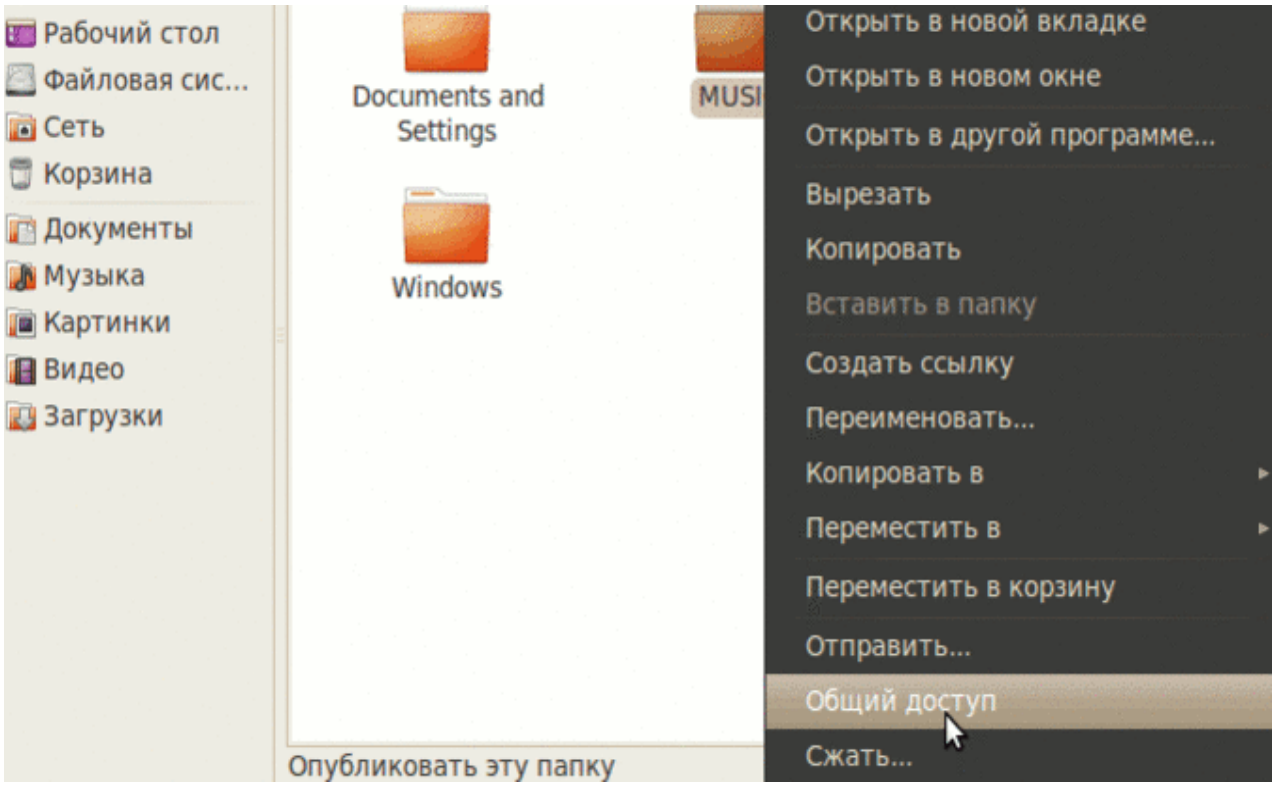

Рисунок 24 - Доступ к папке из локальной сети

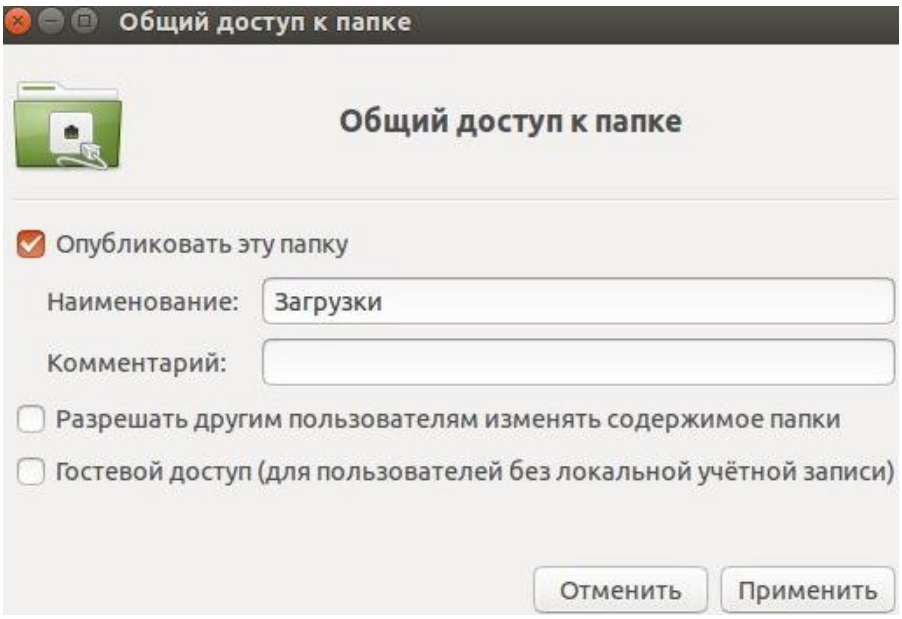

Рисунок 25 - Общий доступ к папке

Межсетевой экран, файрвол или брандмауэр - комплекс аппаратных и программных средств в компьютерной сети, осуществляющий контроль и фильтрацию проходящих через него сетевых пакетов в соответствии с заданными правилами.

Во все ядра *Linux*, начиная с 2.0, встроено средство для фильтрации сетевых пакетов. В 2.0 это *ipfwadm*, в 2.2 - *ipchains*, а в 2.4 и 2.6 - *iptables*. Принцип фильтрации такой: когда через ядро проходит пакет, он проверяется на совпадение с одним или несколькими правилами. При этом в зависимости от этих правил он может быть пропущен (*ACCEPT*), отброшен (*DROP*) или отклонен (*REJECT*). Кроме того, он может быть отправлен на проверку в следующую цепочку правил. Здесь же можно указать, что факт прохождения пакета, подходящего под определенное правило, должен быть отмечен в *syslog*. Правила могут включать в себя проверку адреса. Для изменения используемого набора правил используется программа, которая так и называется - *iptables*. Все правила хранятся в памяти ядра и при перезагрузке сбрасываются. Поэтому необходимо создать файл конфигурации, из которого правила фильтрации будут считываться при загрузке машины. Обычно это */etc/rc.d/rc.firewall*. Это обычный скрипт оболочки, который вызывает */sbin/iptables* с определенными параметрами, соответствующими составленным правилам. Поэтому в большинстве дистрибутивов для изменения конфигурации *iptables* необходимо отредактировать указанный файл и запустить его (этот файл, как правило, автоматически выполняется при загрузке машины). После чего можно посмотреть обновленную таблицу правил командой *iptables -L*.

3. Запустим еще одну виртуальную машину. Установим на ней сетевое подключение и адрес пользователя (второй виртуальной машины) 192.168.1.201, а адрес сервера 192.168.1.200. С помощью команды *scp* можно скопировать файл из директории сервера, если известно его местонахождение,

далее прописываем путь и папку, в которую мы хотим сохранить наш файл: *scp 192.168.1.200:/home/ nurmanovadil/Crypt/file3-nurmanovadil.txt.encoded /home* (результат представлен на рисунке 26). На рисунке 27 открываем файл.

```
The authenticity of host '192.168.1.200 (192.168.1.200)' can't be established
ECDSA key fingerprint is e1:0c:5f:f3:b9:b9:8a:42:cd:e8:83:29:91:6f:f8:20.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.200' (ECDSA) to the list of known hosts
root@192.168.1.200's password:
file3-Nurmanov.txt.encoded                    100%   65     0.1KB/s   00:00
[root@localhost network-scripts]# ls
ifcfg-enp0s3  ifdown-ppp       ifup-ib      ifup-Team
ifcfg-lo      ifdown-routes    ifup-ippp    ifup-TeamPort
ifdown        ifdown-sit       ifup-ipv6    ifup-tunnel
ifdown-bnep   ifdown-Team      ifup-isdn    ifup-wireless
ifdown-eth    ifdown-TeamPort  ifup-plip    init.ipv6-global
ifdown-ib     ifdown-tunnel    ifup-plusb   network-functions
ifdown-ippp   ifup             ifup-post    network-functions-ipv6
ifdown-ipv6   ifup-aliases     ifup-ppp
ifdown-isdn   ifup-bnep        ifup-routes
ifdown-post   ifup-eth         ifup-sit
[root@localhost network-scripts]# cd /home/
[root@localhost home]# ls
file3-Nurmanov.txt.encoded
[root@localhost home]# gedit file3-Nurmanov.txt.encoded
```

Рисунок 26 - Копирование файла с сервера

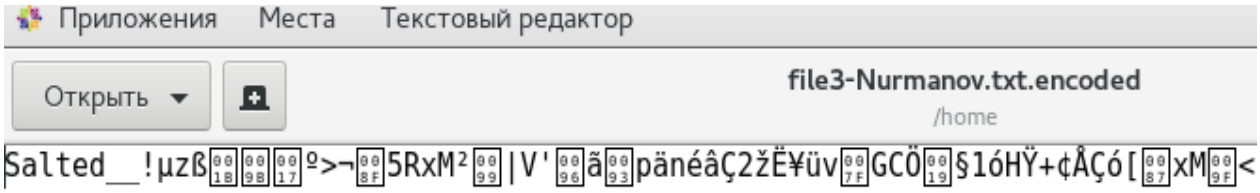

Рисунок 27 - Открытие зашифрованного файла на второй ОС

Настройка правила в файерволе, не позволяющее копировать из ОС2 в ОС1, показана на рисунке 28; проверка на рисунках 29-30.

Убираем поставленное правило, проверяем работоспособность (рисунок 31).

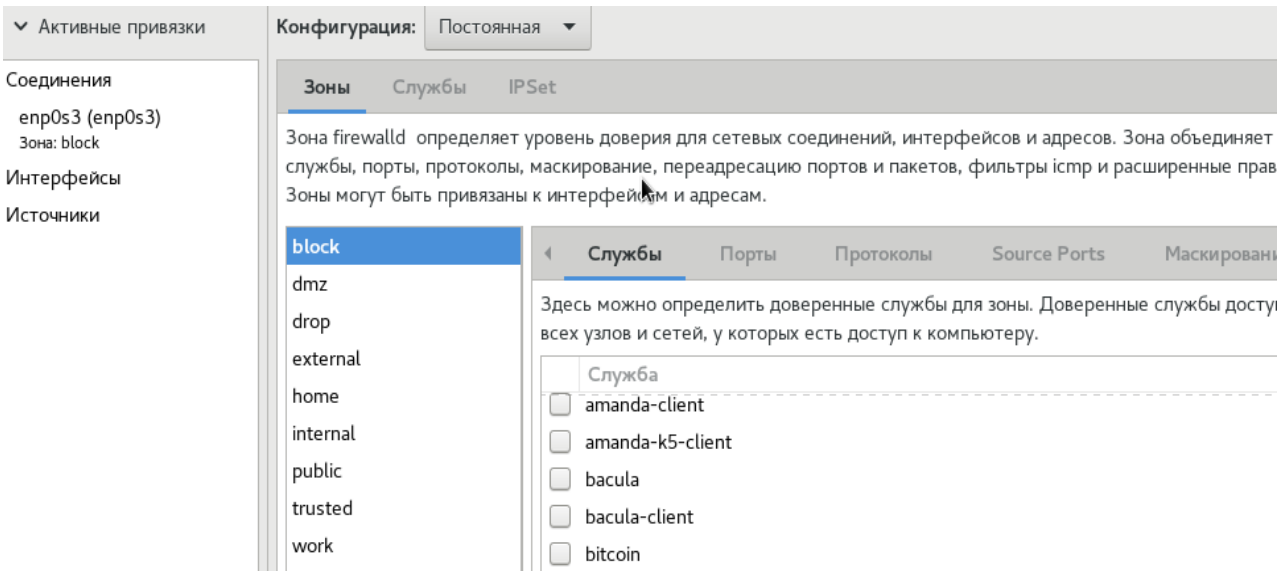

Рисунок 28 - Помещаем интерфейс в блокированную зону

```
Файл  Правка  Вид  Поиск  Терминал  Справка
[nurmanovadil@localhost Crypt]$ ping 192.168.1.201
PING 192.168.1.201 (192.168.1.201) 56(84) bytes of data.
From 192.168.1.201 icmp_seq=1 Destination Host Prohibited
From 192.168.1.201 icmp_seq=2 Destination Host Prohibited
From 192.168.1.201 icmp_seq=3 Destination Host Prohibited
```

Рисунок 29 - Запрещаем трафик

```
[nurmanovadil@localhost Crypt]$ sudo scp 192.168.1.201:/home/result-Nurmanov.txt
 /home/nurmanovadil/Crypt/
[sudo] пароль для nurmanovadil:
ssh: connect to host 192.168.1.201 port 22: No route to host
```

Рисунок 30 - Попытка скопировать файл

```
[nurmanovadil@localhost Crypt]$ ping 192.168.1.201
PING 192.168.1.201 (192.168.1.201) 56(84) bytes of data.
64 bytes from 192.168.1.201: icmp_seq=1 ttl=64 time=0.372 ms
64 bytes from 192.168.1.201: icmp_seq=2 ttl=64 time=0.362 ms
64 bytes from 192.168.1.201: icmp_seq=3 ttl=64 time=0.406 ms
64 bytes from 192.168.1.201: icmp_seq=4 ttl=64 time=0.365 ms
```

Рисунок 31 - Устанавливаем интерфейс на публичный доступ и связь снова доступна

8.3 Список контрольных вопросов

1 Какие параметры могут быть еще в *openssl enc*, кроме *-aes-256-cbc*?
2 Каким образом осуществляется настройка правил файервола?
3 Назовите 3 способа настройки общего доступа.

9 Лабораторная работа №9. Обеспечение безопасности операционной системы. Изучение программных и системных угроз и типов сетевых атак систем

Цели работы: приобретение навыков по организации безопасности при загрузке в разных режимах ОС; освоение мер деактивации и ограничения пользователей.

9.1 Рабочее задание

1. Запустить ОС в минимальном окружении: ядро и командный интерпретатор:

1) перезагрузить систему и войти в меню загрузчика *GRUB2*;

2) изменить параметры запуска ядра;

3) попробовать создать файл «*proba.txt*» в корневом каталоге*;*

4) перемонтировать корневой раздел в режиме чтения-записи;

5) попробовать создать файл «*proba.txt*» в корневом каталоге;

6) загрузиться в обычном режиме.

2. Создать учетные записи *user1, user2, user3*. Отключить учетную запись *user1* одним способом, *user2* другим способом, *user3* третьим, восстановить их работоспособность.

3. Показать смену пароля на *root*-пользователя в ОС через взлом с помощью *LiveCD* и через взлом загрузчика.

9.2 Методические указания к выполнению лабораторной работы

1. Для запуска ОС в минимальном окружении: надо выполнить следующие действия:

1) Перезагрузить систему и войти в меню загрузчика *GRUB2*.

2) Изменить параметры запуска ядра, дописав параметр *init=/bin/bash* в конец строки параметров ядра (*kernel*) (рисунок 32). Выполнить загрузку ядра с данным параметром и дождаться приглашения терминала.

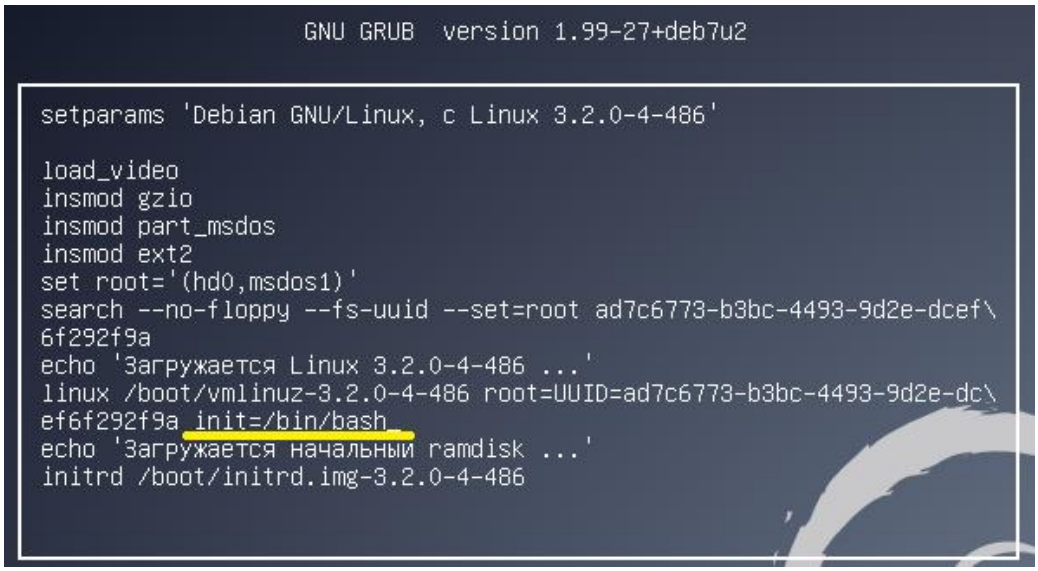

Рисунок 32 – Изменения режима загрузки текущей записи

3) Попробовать создать файл «*proba.txt*» в корневом каталоге: *«touch /proba.txt».* Ответом будет комментарий об ошибке о том, что корневая файловая система доступна только для чтения (рисунок 33).

Рисунок 33 – Сообщение об ошибке

4) Перемонтировать корневой раздел в режиме чтения-записи: *«mount -o remount, rw /»*, где *remount, rw* - опции перемонтирования в режим чтения-записи (только для чтения будет *ro - read-only*). А слеш - корневой раздел (рисунок 34).

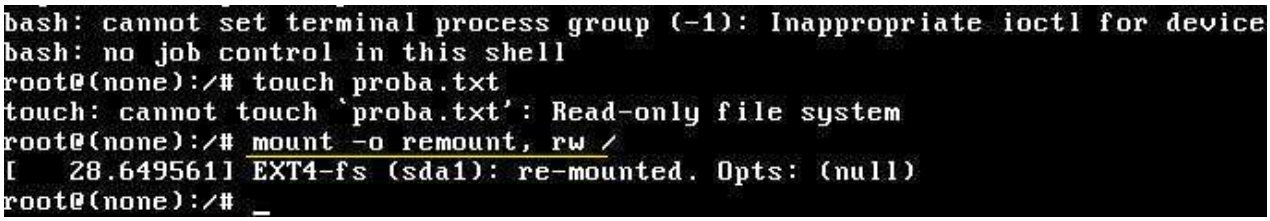

Рисунок 34 – Команда перемонтирования

5) Попробовать создать файл «*proba.txt*» в корневом каталоге: *«touch proba.txt».* Файл успешно создан (рисунок 35).

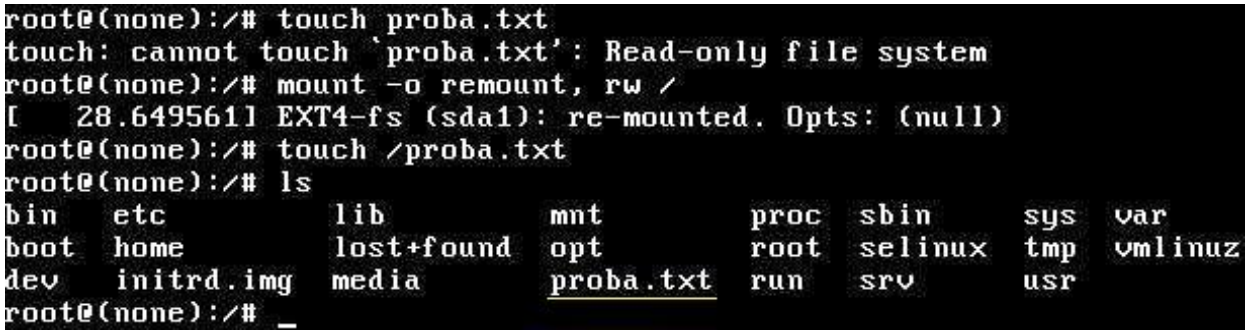

Рисунок 35 - Реакция ОС на команду создания файла *proba.txt*

6) Выполнить команду «*exec /sbin/init*» для начала нормальной загрузки ОС. Загрузка системы в минимальном окружении может быть полезна в различных случаях, особенно, если происходит восстановление удаленных случайно данных, когда любое воздействие на диск (запись) нежелательны. При желании можно перемонтировать корневой раздел в режиме чтения-записи и продолжить реанимацию системы. Как только операции по

восстановлению-диагностике будут выполнены, можно продолжить нормальную загрузку системы.

2. Способ 1. Редактирование файла */etc/shadow*. В данном файле хранятся зашифрованные пароли из /*etc/passwd*. К примеру, вот так выглядит зашифрованный пароль в файле *shadow*:

user1:6KYriHdKR$Yku3LWgJmomsynpcle9BCA:15711:0:4444:5:::

для отключения аккаунта надо перед зашифрованным паролем добавить символы ! или *:

user:!6KYriHdKR$Yku3LWgJmomsynpcle9BCA:15711:0:4444:5:::

или:

user:6KYriHdKR$Yku3LWgJmomsynpcle9BCA:15711:0:4444:5:::*

После чего пользователь *user1* не сможет себя авторизовать, так как системе не удастся расшифровать пароль. Чтобы вернуть, надо удалить из файла /*etc/shadow* символы ! и *, которые добавили раньше.

Способ 2. Оболочка *nologin*. Существует еще способ деактивации аккаунта пользователя - это пользовательская оболочка *nologin*, которая находится по адресу: /*usr/sbin/nologin*. В файле *passwd* изменить по примеру:

user2:x:1001:1001:Test,User,,:/home/user:/bin/bash

на

user2:x:1001:1001:Test,User,,:/home/user:/usr/sbin/nologin

После чего пользователь *user2* не сможет авторизоваться, несмотря на ввод правильного пароля.

Способ 3. То же самое можно сделать с помощью команды: «*usermod -L user3*». Любой метод авторизации, использующий для аутентификации пользователя файл /*etc/shadow*, больше не будет работать, так как расшифровать пароль будет невозможно:

$ su user3

Password:

su: Authentication failure

Для активации аккаунта можно ввести команду: «*usermod -U user3*».

3. Чтобы осуществить взлом с помощью *LiveCD,* необходимо загрузиться с *Live*-диска, подмонтировать корневой раздел, отредактировать /*etc/shadow*, который состоит из строк, каждая из которых имеет 9 полей разделенных двоеточиями. В первом поле хранится имя пользователя, во втором - хеш пароля. Надо удалить содержимое второго поля нужной строки, сохранить файл, отмонтировать файловую систему, перезагрузиться и, загрузившись с жесткого диска, войти в систему как *root*, без пароля.

9.3 Список контрольных вопросов

1 Найти файлы: *passwd, group, shadow u gshadow*; показать, пояснить.

2 Описать алгоритм взлома через *CD*-диск.

3 Описать алгоритм взлома через загрузчик.

10 Лабораторная работа №10. Архивирование. Восстановление

Цель работы: приобретение навыков работы с упаковщиком *TAR*; научиться упаковывать и распаковывать файлы посредством *GZIP* и *BZIP2*.

10.1 Рабочее задание

1. Выполнить задания по архивированию объектов:

- создать папку, в ней файл с календарем 1-ваша_фамилия.*txt*; в файл 2-ваша_фамилия.*txt* записать информацию о работающих пользователях (*w*); в файл 3-ваша_фамилия.*txt* записать информацию о запущенных процессах в системе; создать архив 4-ваша_фамилия.*tar* из файла 1-ваша_фамилия.*txt*, добавить в него остальные два файла; удалить три файла, оставив в папке только архив; распаковать файлы в текущую папку и на уровень выше, удалить архив, проверить содержимое полученного; показать пример архивирования с исключением объектов;

- создать вторую папку; в ней создать файл 5-ваша_фамилия.*txt* и *6*-ваша_фамилия.*txt* (в первом - информация о дистрибутиве, во втором - информация о пользователе); упаковать содержимое папки в архив *tar*;

- упаковать содержимое папки в архив tar с применением архиватора *gzip* (или *bzip2*), сравнить размеры;

- перенести архив во временный каталог, удалить содержимое первоначального каталога; восстановить содержимое из архива;

- установить архиватор *rar*, заархивировать файлы 5-ваша_фамилия. *txt* и *6*-ваша_фамилия.*txt* по паролю «123», удалить первоисточник файлов и восстановить из архива файлы.

2. Сделать резервную копию системы, показать пример удачного восстановлению из образа системы.

10.2 Методические указания к выполнению лабораторной работы

1. Архиватор нужен для уменьшения размеров файлов, упаковывания каталогов в единые объекты для более быстрой передачи по каналам связи и компактного хранения.

В *Linux* есть три различных метода упаковки: *tar*, *gzip* и *bzip2*. Команды для работы:

tar cf 2.tar 1.txt - создать *tar*-архив с именем 2.*tar*, содержащий 1.*txt*;

tar czf 2.tar.gz 1.txt - создать *tar*-архив с сжатием *Gzip* по имени 2.*tar.gz*;

tar cjf 2.tar.bz21.txt - создать *tar*-архив с сжатием *Bzip2* по имени 2.*tar.bz*;

tar xf 2.tar - распаковать архив 2.*tar* в текущую папку;

tar xzf 2.tar.gz - распаковать *tar*-архив с *Gzip*;

tar xjf 2.tar.bz - распаковать *tar*-архив с *Bzip2*.

tar --exclude='/home/user/' -cvzf archive.tar.gz /home/** - архивировать всю директорию /*home*, но исключить из архива папку /*home/user*.

После создания и просмотра содержимого файлов посмотрим содержимое каталога (рисунок 36).

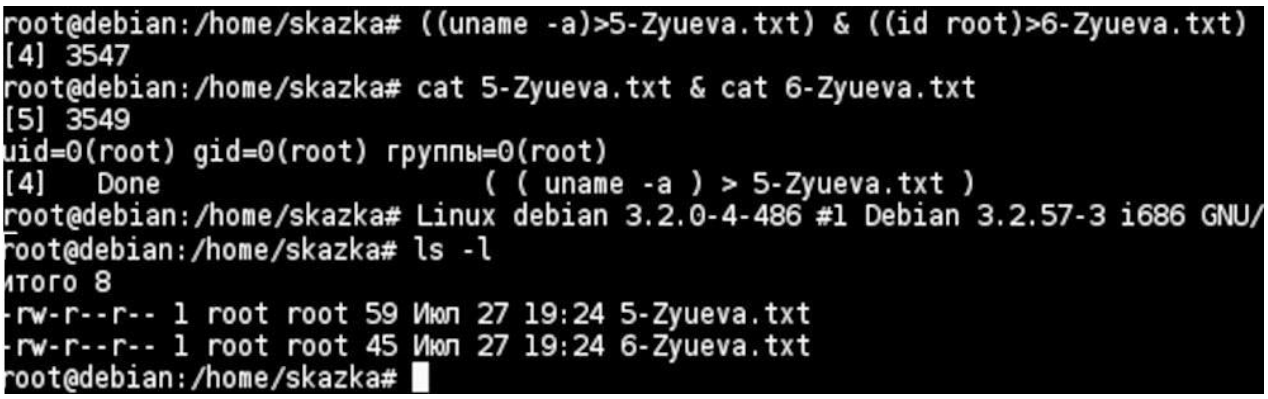

Рисунок 36 – Создание и просмотр содержимого файлов

Команда «*xxd -l 100 5-Zyueva.txt*» выводит первые 100 байт файла. Чтобы упаковать все это в архив 7.*tar,* надо воспользоваться одной из команд выше.

Далее посмотреть размер архива: *«ls -lh 7.tar*», при этом получается, что исходные файлы были даже меньше по размеру, а в архиве стали весить больше, так как сам по себе *tar* не сжимает файлы, а просто упаковывает их в своего рода контейнер, и уже этот контейнер можно «скормить» утилитам-упаковщикам *gzip* или *bzip2* (их можно использовать в *tar*, с помощью ключей *z* и *j* соответственно).

Сожмём теперь этот файл при помощи *gzip: «gzip 7.tar*». Нетрудно заметить, что к расширению файла теперь припишется *gz* автоматически и размер файла уменьшился (большой уровень сжатия).

Распаковка из резервной копии ведется похоже. Есть тут два пути:

а) последовательный: распакуем архив *.gz*: «*gzip -d 7.tar.gz*», ключ «*-d*» означает «*decompress*», в папке появился файл 7.*tar* прежнего размера. Далее: «*tar -xvf 7.tar*» - распакует содержимое этого контейнера.

Ввиду большой популярности архиваторов *gzip* (и *bzip2*), их поддержка уже включена во многие утилиты. В частности, в *tar*. Для *gzip* прибавляется опция *z*, для *bzip2 – j*;

б) можно обойтись одной командой: «*tar -xvzf ~/backup.tar.gz*» и декомпрессировать и разжать сразу.

Если архиватор не установлен: «*apt-get install rar*». Для архивации всех файлов и папок в директории */home/skazka/lab9/z1-5* и директории */home/skazka/lab9/z1-3* надо использовать «*rar a -r -m5 myarchive.rar /home/skazka/lab9/z1-5/* /home/skazka/lab9/z1-3/*»,* где *«-a*» - добавить данные (*add*); «*-m5*» - степень сжатия от 0 до 5. 0 - без сжатия; *myarchive.rar* - имя архива; остальное - архивируемые директории через пробел; «*-r*» - опция, которая добавляет информацию для восстановления. Опция «*-hp*» позволяет задать пароль на архив.

Распаковать архив: «*rar e myarchive.rar*» в текущую директорию, проверить архив*: «rar t myarchive.rar»*; восстановить *«rar myarchive.rar».*

2. Резервное копирование системы (*backup*) является одной из важных профилактических мер по поддержанию стабильности работы системы. Для резервного копирования понадобится утилита по работе с архивами – *tar*.

Для выполнения задания потребуется флеш-*LiveCD*, с которого ставили систему, и раздел, на который нужно сохранить данные. Его также нужно примонтировать.

Итак, предположим, что ОС установлена на первом разделе первого жесткого диска (*/dev/sda1*). Загружаемся с *LiveCD* и монтируем этот раздел скажем в /mnt, проверяя попутно разделы компьютера (рисунок 37).

```
root@kali:~# mount /dev/sda1 /mnt
root@kali:~# fdisk -l
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0d3d7496

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *        2048 40038399 40036352 19.1G 83 Linux
/dev/sda2       40038400 41940991  1902592  929M 82 Linux swap / Solaris


Disk /dev/sdb: 931.5 GiB, 1000204885504 bytes, 1953525167 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xa4b57300

Device     Boot      Start        End    Sectors   Size Id Type
/dev/sdb1            16065 1953520064 1953504000 931.5G  f W95 Ext'd (LBA)
/dev/sdb5            16128 1538882414 1538866287 733.8G  b W95 FAT32
```

Рисунок 37 – Монтирование и просмотр разделов

Монтируем раздел, на котором предполагается разместить резервную копию. При возникновении ошибок следует отмонтировать раздел и обратно примонтировать (рисунок 38). Далее переходим в директорию примонтированного раздела с системой и смотрим какие директории есть, в примонтированной директории будем создавать резервную копию (рисунок 39). Увидев список директорий, включаем нужные для резервной копии (рисунок 40). Процесс создания резервной копии отражен на рисунке 41.

```
root@kali:~# mkdir /backup
root@kali:~# mount /dev/sdb6 /backup
Mount is denied because the NTFS volume is already exclusively opened.
The volume may be already mounted, or another software may use it which
could be identified for example by the help of the 'fuser' command.
root@kali:~# umount /dev/sdb6
root@kali:~# mount /dev/sdb6 /backup
```

Рисунок 38 – Монтирование раздела для *backup*

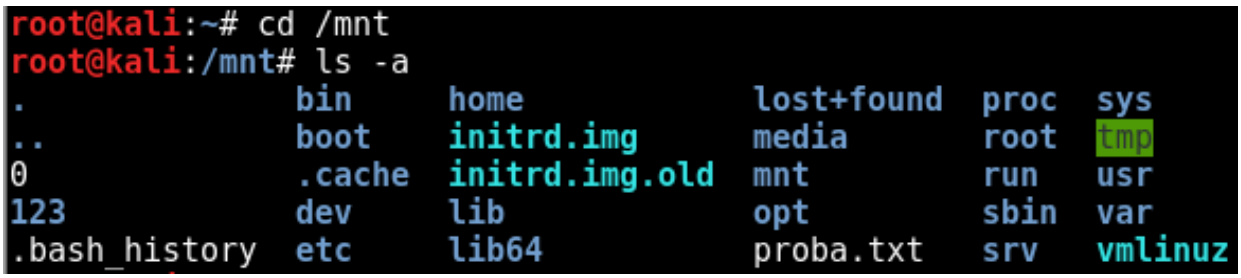

Рисунок 39 – Системные директории

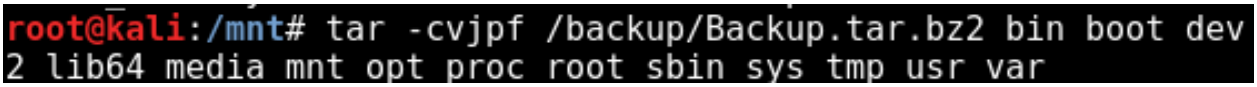

Рисунок 40 - Команда для создания архива

Рисунок 41 – Процесс создания резервной копии

Для успешного восстановления понадобится все тот же *LiveCD*, сама резервная копия и некоторое количество времени. Загружаемся с *LiveCD* и монтируем разделы по уже вышеописанной схеме. Если не переносили резервную копию на другой жесткий диск, то имеющуюся систему нужно предварительно удалить командой: *rm –rf /mnt/**. Копируем архив с резервной копией на целевой раздел: *cp /backup/Backup.tar.bz2 /mnt/*. Переходим в нашу будущую систему и разархивируем *backup* (процесс показан на рисунке 42). На этом восстановление резервной копии завершено.

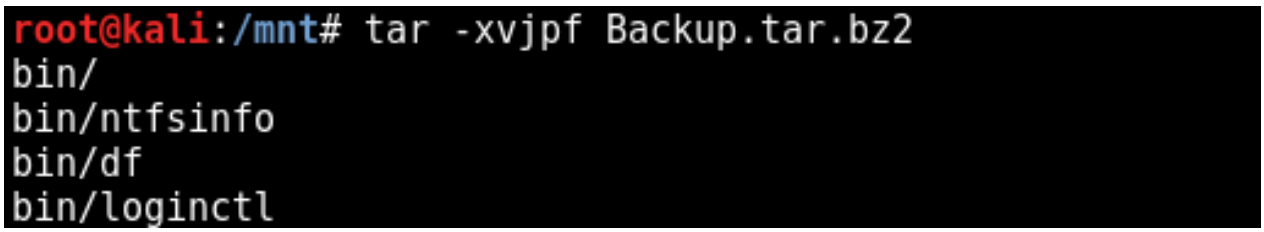

Рисунок 42 – Восстановление системы

10.3 Список контрольных вопросов

1 С какими типами файлов-архивов можно работать?
2 Зачем делать бэкап системы?
3 Расскажите про способы резервного копирования.

11 Лабораторная работа №11. Реализация простых сценариев

Цель работы: приобретение навыков работы по настройке работы ОС путем программирования несложных скриптов.

11.1 Рабочее задание

1. Задать постоянные алиасы в системе.

2. Написать и запустить скрипты, задействовав разные цвета и заливки:

- вывести на дисплей информацию обо всех разделах, смонтированных в системе в понятном для человека виде и записать ее в файл, создать к получившемуся файлу все виды ссылок;

- показать содержимое текущей папки, создать файл (с текущей датой и временем работы системы), вывести информацию об атрибутах файла, поменять их, просмотреть файл с его атрибутами и удалить файл, показать опять содержимое директории;

- посчитать количество букв в строке; найти любую последовательность букв; посчитать количество символов текущего скрипта;

- посчитать количество строк в файле;

- показать содержимое текущего скрипта, скопировать содержимое, очистить его и переписать его заново;

- вывести в скрипте значения системных переменных;

- привести пример расчета арифметических операций;

- привести пример ожидания чтения с клавиатуры;

- привести пример работы с координатами;

- показать работу команд для папок и файлов;

- привести пример работы скрипта с пятью переменными и чтением с клавиатуры;

- запустить 3 скрипта одновременно в одном скрипте и вывести сообщения.

3) На автозагрузку в системе поставить любую задачу.

4) Поставить скрипт на загрузку в определенное время (например, скрипт запуска любой службы в системе) через планировщика.

11.2 Методические указания к выполнению лабораторной работы

1. Команда *alias* позволяет пользователю создавать простые псевдонимы для команд любой сложности (вместе с опциями, аргументами, перенаправлениями и программными каналами). Псевдонимами заменяются команды или группы команд, которые долго или неудобно набирать на клавиатуре, их применение ускоряет и упрощает работу в командной строке.

Но особенность такова, что если алиясы были заданы через командную строку, то они действительны для текущей сессии текущего пользователя, после выхода или перезагрузки они исчезнут.

Для того чтобы они были действительны навсегда, рекомендуется записать их в файл *bashrc* (для удобства их можно записать в конец файла).

Универсальным является метод с использованием файла *bashrc*. Сначала нужно проверить наличие файла *bashrc* в системе: «*locate bashrc*». В зависимости от наличия файлов типа *bashrc* (*.bashrc, bash.bashrc* и т.п.) в различных директориях, возможно несколько вариантов:

а) чтобы создать постоянные псевдонимы для данного пользователя, если в домашней директории есть файл *.bashrc* (скрытый), то нужно просто вписать в конец этого файла нужные псевдонимы по одному на строку.

Например:

alias c='clear'

alias grep='grep --color'

и так далее, а если в домашней директории не имеется файла *.bashrc*, то нужно создать текстовый файл вписать туда нужные псевдонимы как показано выше. Заработают созданные псевдонимы после перезагрузки;

б) чтобы создать постоянные псевдонимы для пользователя *root*: если существует файл */root/.bashrc* (скрытый), то вписать нужные псевдонимы в этот файл. Если такового файла нет, то следует создать его и вписать псевдонимы. А вписав новые псевдонимы, не забудьте перезагрузиться.
Заработают созданные псевдонимы после перезагрузки;

в) */etc/bash.bashrc* (если используется оболочка *bash*). Все то же самое как и описано выше.

Команда *alias* необычна тем, что имеет всего одну опцию *-p*, чтобы «одним махом» создать псевдоним и просмотреть список уже созданных. Например: «*alias -p p='pwd'*» показывает созданные алиасы до этого момента и создает при этом следующий (не показывая его в общем списке).

Чтобы изменить поведение команды, по умолчанию, например, введя псевдоним: «*alias ls='ls –a'*», можно быть уверенным, что команда *ls* покажет также скрытые файлы, чего по умолчанию, она не делает.

Если взять команду *df*, которая выводит информацию обо всех разделах, смонтированных в системе, то можно увидеть, что информация не совсем доходчива, так как единицей размера раздела, по умолчанию, выбран однокилобайтный блок (это тянется с тех дней, когда килобайт считался большим количеством). Существует опция *-h*, которая использует в качестве единиц размера мегабайты и гигабайты, это намного нагляднее, поэтому имеет смысл ввести псевдоним «*alias df='df –h'*».

Полезным также будет псевдоним «*alias grep='grep –color'*», который сделает вывод команды *grep* цветным (рисунок 43).

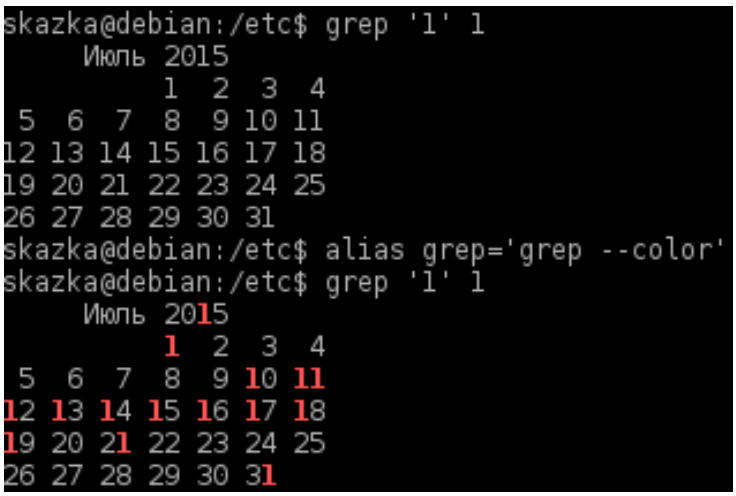

Рисунок 43 – Изменение параметров вывода команды через алиас

Чтобы избежать последствий неправильного набора команд, например, если пользователь постоянно ошибается и постоянно печатает *pdw* вместо *pwd*, он может создать псевдоним: «*alias pdw=’pwd’*» и больше не задумываться о том, правильно ли он ввел команду.

Чтобы повысить безопасность системы, сделав некоторые «опасные» команды интерактивными: это заставит пользователя подтверждать свои действия. Например, команда *rm* удаляет файлы и директории без возможности восстановления, поэтому имеет смысл создать для нее псевдоним: «*alias rm=’rm –i'*». В интерактивном варианте команда не столь опасна. Или взять команду *cp*, копирующую содержимое одного файла в другой. Если по ошибке указать в качестве аргумента существующий файл, то команда сотрет его содержимое и перезапишет новым. Избежать этого поможет псевдоним: «*alias cp=’cp -i’*», который заставит подтвердить операцию копирования, снизив тем самым риск ошибки.

Удаление псевдонимов: «*unalias* имя_псевдонима», эта команда удаляет не только созданные для текущей сессии псевдонимы, но и постоянные, прописанные в конфигурационном файле. Опция «*-a*» позволяет удалить все псевдонимы для данного пользователя и данной сессии: «*unalias -a*».

2. Хорошей идеей было бы создать директорию *~/scripts*, в которой будут находиться все скрипты. Взаимодействие с командным интерпретатором *Shell* осуществляется с помощью командной строки. Командный файл или скрипт содержит одну или несколько выполняемых команд или процедур. Скрипт целесообразно делать, когда используется одна и та же последовательность команд, записав которую можно вызывать на выполнение многократно. По правилам хорошего тона программирования в ОС скрипт должен иметь расширение *sh*, чтобы люди отличали простые файлы от исполняемых, но это правило не всегда используется.

Если скрипты в ОС не запускаются, и система выдает ошибку, то необходимо проверить в файле */etc/fstab* напротив нужного раздела системы, стоит ли параметр exec к файлам (исполняемость файлов). Перед написанием скрипта стоит зайти в */bin* и посмотреть, какая оболочка установлена (команда

«*find *sh**»), в дальнейшем, например, будем пользоваться *bash*. Работа со скриптами ведется через терминал.

Алгоритм создания скрипта:

- в нужной папке создать файл и заполнить его привычным для скрипта содержанием - командами, начиная с указания компилятора (в нашем случае это *bash*);

- изменить права к файлу-скрипту: добавить *x* – исполняемость;

- запустить файл в терминале.

Напишем несколько скриптов.

Пример 1. Вывести в сообщении название скрипта и его автора, календарь, просмотреть текущую директорию.

Выбираем папку, например, */home*, заходим, создаем (*cat>primer1.sh*) с содержимым, представленным на рисунке 44а, при этом учитывать надо, что «#» - решетка с пробелом это комментарий; а без пробела - системный символ. Готовый файл нужно сделать запускаемым в терминале (*chmod 777 primer1.sh*). Запустить на выполнение: «*./1.sh*» или «*bash primer1.sh*» (результат представлен на рисунке 44б).

а)

```
File  Edit  View  Bookmarks  Settings  Help
 GNU nano 2.2.6                                  Файл

#!/bin/bash
echo " ------------------------------------------"
echo "| this my first script, author is Zuyeva E.A. |"
echo " ------------------------------------------"
cal
ls  -l
echo "Final"
```

б)

```
skazka@debian:~$ bash primer1.sh
 ------------------------------------------
| this my first script, author is Zuyeva E.A. |
 ------------------------------------------
     Июль 2015
Вс Пн Вт Ср Чт Пт Сб
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

итого 4
-rw-r--r-- 1 skazka skazka 199 Июл 29 15:58 primer1.sh
Final
skazka@debian:~$
```

а) листинг; б) выполнение скрипта.

Рисунок 44

Пример 2. Написать скрипт, выводящий текущую дату, создать файл и показать атрибуты этого файла до и после их смены.

Через консоль создаем файл (*primer2.sh*), в который пишем:

#!/bin/bash

date
touch file_for_primer2.txt
echo “byl sozdan fail, vot ego prava:”
ls –l file_for_primer2.txt
chmod 777 file_for_primer2.txt
echo “a vot prava posle izmenenii:”
ls -l file_for_primer2.txt

Файл в консоле *primer2.sh* сделать испоняемым «*chmod primer2.sh*».

Запустить на выполнение: *./primer2.sh* или *bash primer2.sh*.

Для обозначения переменных используются правила:

– последовательность букв, цифр и символов подчеркивания;

– переменные не могут начинаться с цифры;

– присваивание значений проводится с использованием «=», например, *PS2 = '<'*;

– для обращения к значению переменной перед ее именем ставится знак $.

Переменные можно разделить на следующие группы:

- позиционные переменные вида *$n*, где *n* - целое число;

- простые переменные, значения которых может задавать пользователь или они могут устанавливаться интерпретатором;

- специальные переменные # ? - ! $ устанавливаются интерпретатором и позволяют получить информацию о числе позиционных переменных, коде завершения последней команды, идентификационном номере текущего и фонового процессов, о текущих флагах интерпретатора Shell.

Пример 3. Переменной x присвоить значение переменной z. Листинг:

z=1000
x= $z
echo $x

Пример 4. Присвоить переменным *x, y, z* значения 10, 100, 200, вывести *x2, y2, z2 (x2=x, y2=y, z2=z):*

x=10; y=100; z=200
x2=$x && y2=$y && z2=$z
echo x2=$x2,y2=$y2, z2=$z2

Пример 5. Подсчет числа символов в цепочках символов. Операция выполняется с использованием функции length в команде *expr,* а *«`»* - находится на клавише с буквой «ё»:

x="The program is written"
y=` expr length "$x"`
echo $y

Пример 6. Вывод сообщения и считывание информации с клавиатуры:

echo -n "Please write down your name:"
read x
echo x=$x

Переменные начинаются с символа $. Чтобы выполнить команду и присвоить вывод переменной, нужно заключить команду в апострофы («`»). Сравните вывод двух скриптов:

#!/bin/bash
x=ls $$ echo $x

и

#!/bin/sh
x=`ls` $$ echo $x

Чтобы обратиться к системным значениям, надо знать информацию:

- $*UID* - содержит идентификатор, который устанавливается при логине;
- $*GROUPS* - группы, к которым принадлежит текущий пользователь;
- $*HOME* – домашний каталог пользователя;
- $*HOSTNAME* – *hostname* компьютера;
- $*HOSTTYPE* – архитектура машины;
- $*PWD* – рабочий каталог;
- $*OSTYPE* – тип ОС;
- $*PATH* – путь поиска программ;
- $*PPID* – идентификатор родительского процесса;
- $*SECONDS* – время работы скрипта (в секундах);
- $# – общее количество параметров, переданных скрипту;
- $* – все аргументы, переданные скрипту (выводятся в строку);
- $@ – то же, что и предыдущий, но параметры выводятся в столбик;
- $! – *PID* последнего запущенного в фоне процесса;
- $$ – *PID* самого скрипта.

В таблице 5 даны листинги скриптов и результат их выполнения.

Таблица 5 – Листинги скриптов с выполнением

Листинг	Результат выполнения
вывод на экран #!/bin/bash echo "Hello World"	Hello World
вывод на экран через переменную #!/bin/bash peremennaya="HELLO WORLD!!!" echo $peremennaya	HELLO WORLD!!!

глобальные и локальные переменные
#!/bin/bash
VAR="global variable"
function bash {
local VAR="local variable"
echo $VAR
}
echo $VAR
bash
echo $VAR
	global variable
	local variable
	global variable

выполнение в скрипте команд оболочки #!/bin/bash echo `uname -o` echo uname -o	GNU/Linux uname -o
чтение/ввод переменных #!/bin/bash echo -e "Type 1 word: \c " read a echo "The word you entered is: $a" echo -e "Enter two words: " read b c echo "Here is your input: \"$b\" \"$c\"" echo "How do you feel about bash scripting? " read echo "You said $REPLY, I'm glad to hear that! " echo "What are your favorite colours (3 colors)?" read -a colours echo "My favorite colours are also ${colours[0]}, ${colours[1]} and ${colours[2]}:-)"	Type 1 word: eferfref The word you entered is: eferfref Enter two words: rvreverfvefverver rv Here is your input: "rvreverfvefverver" "rv" How do you feel about bash scripting? fevfeverver You said fevfeverver, I'm glad to hear that! What are your favorite colours (3 colors) ? e3 4 4 My favorite colours are also e3, 4 and 4:-)
арифметические операции a=7 b=7 r=$(($a + $b)) echo $r echo r=$((${a} + ${b})) echo $r	14 14
координаты #!/bin/bash echo -en "\E[3;3f Hello, world!" #курсора координаты (3;3), а затем выведен текст	Hello, world!
временное ожидание ввода #!/bin/bash echo "I waiting you: print me 2 simbols now" read -n 2 B echo echo "I waiting you: print me 3 simbols during 5 sec" read -t 5 -n 3 B	I waiting you: print me 2 simbols now df I waiting you: print me 3 simbols during 5 sec dff
работа с объектами, аналог ls -l для .txt файлов в текущей папке #!/bin/bash ls -l | grep "\.txt$" далее уже не в файле, а в консоле alias z=”bash 1”	-rwxr-xr-x 1 root root 35 Mar 15 18:58 1.txt -rwxr-xr-x 1 root root 35 Mar 15 18:58 2.txt

3. Организация автозагрузки. Чтобы правильно запустить/остановить сервис, необходимо описать сценарий с командами для запуска/остановки. Содержимое каталога */etc/init.d* - содержит скрипты, которые управляют загрузками/остановками сервисов в ОС. Итак, первый, но не последний пункт успешной настройки - наличие скрипта в */etc/init.d*. В скрипте не описывается, когда должен выполняться тот или иной сценарий. Он лишь может принимать параметры *start, stop, restart* и так далее. ОС знает, когда необходимо вызвать скрипт, так как в каталогах хранятся символические ссылки на скрипты из */etc/init.d /etc/rcN.d*, где *N* - это цифра от 0 до 6. Что означает каждый каталог:

- *rc0.d* - выполнение скрипта при выключении системы;
- *rc1.d* - выполнение скрипта при запуске системы в одно-пользовательском режиме;
- *rc2.d* - выполнение скрипта при запуске системы в много-пользовательском режиме;
- *rc3.d* - *rc5.d* - зарезервировано;
- *rc6.d* - выполнение скрипта при перезагрузке системы.

Например, если происходит перезагрузка, то будут выполнены все скрипты из каталога */etc/rc6.d*, при выключении из */etc/rc0.d* и так далее. Цифра в названии каталога называется уровнем запуска, т.е. есть каталог */etc/rc0.d* будет называться нулевым уровнем запуска и так далее. Есть очередность выполнения скриптов из каталогов *rcN.d*. Ведь для правильной организации запуска/остановки работы может потребоваться запускать/останавливать сервисы в определенном порядке. Этот момент решается специальным именованием файлов в каталогах уровней запуска. Файлы имеют следующие имена: [*S*|*K*]*NN*[имя], где [*S*|*K*] - это один символ («*S*» означает, что скрипт запускает сервис, «*K*» – останавливает), *NN* – порядковый номер, [имя] – имя файла. Символ «*S*» или «*K*» самостоятельно выбирать не придется, так как все скрипты в каталогах *rc1.d-rc5.d* должны начинаться с символа *«S*», а в каталогах *rc0.d* и *rc6.d* – с символа «*K*». Число «*NN*» определяет очередность запуска скриптов, который производится от меньшего к большему. Чем меньше число у скрипта для запуска, тем раньше он будет запущен при старте системы; чем больше число у скрипта остановки сервиса, тем позже он будет выполнен.

При необходимости запуска какой-либо службы или приложения до или после конкретного существующего сервиса надо посмотреть его порядковый номер в соответствующей директории *rcN.d* и учитывать при выборе порядкового номера для своего скрипта.

В каталоге */etc/init.d* находится пример скрипта для управления запуском/остановкой сервисов. Это файл */etc/init.d/skeleton*, а в примере ниже он будет упрощен. Для создания нового скрипта необходимо сделать копию примера и отредактировать его. Далее надо воспользоваться командой: «*sudo cp /etc/init.d/skeleton /etc/init.d/myscript && vi /etc/init.d/myscript*». При создании нового скрипта надо дать ему права на выполнение: «*chmod +x /etc/init.d/myscript».*

Можно воспользоваться специализированной утилитой *update-rc.d*, с её помощью можно добавить новый скрипт в любой уровень загрузки, удалить существующий и так далее. Вот пример использования: «*sudo update-rc.d myscript start 99 2 3 4 5 . stop 01 0 1 6*» - добавит новый скрипт «myscript» во все уровни загрузки. Будет выполнен запуск сервиса на уровнях со 2 по 5 с приоритетом 99 (в последнюю очередь) и остановка сервиса на 0, 1 и 6 уровнях с приоритетом 01 (самым первым). Чтобы удалить скрипт из автозагрузки: «*sudo update-rc.d -f myscript remove*».

4. *Cron* - планировщик *Linux*, он выполняет задания по расписанию.

CronTab – утилита, позволяющая автоматически запускать программы в определенное время, в том числе и периодически, например, раз в час, каждую пятницу и т.д.

Структура файла с заданиями для *CronTab*:

```
* * * * * command
— — — — —
| | | | |
| | | | +—— День недели (0 - 6) (Sunday=0)
| | | +———- Месяц года (1 - 12)
| | +———— День месяца (1 - 31)
| +————— Час дня запуска (0 - 23)
+——————- Минута часа для запуска (0 - 59)
```

command - запускаемая программа или скрипт; значок * задаёт параметр (день, год, месяц, час).

Пример:

01 * * * * *xclock* - запуск программы *xclock* каждый час в одну минуту.

0 6 * * * *script* - запуск скрипта каждый день в 6 часов утра.

Значения могут быть числом, трехбуквенным названием, а также диапазоном, например, запись «1-5» в поле day будет означать «с понедельника по пятницу». Значения могут отделяться запятыми: «1,15,31» в поле *day* будет запускать указанную команду 1-го, 15-го и 31-го числа каждого месяца.

Все пять полей времени допускают использование символа «*», который обозначает «использовать любое допустимое значение» для этого поля.

Для создания задания можно использовать команды:

- *crontab -e* - изменит *crontab* файл или создаст новый;
- *crontab -l* - отобразит содержимое существующего *crontab* файла;
- *crontab -r* - удалит *crontab* файл;
- *crontab -v* - отображает когда в последний раз изменяли свой *crontab*.

11.3 Список контрольных вопросов

1 Что такое «скрипт» и для чего он создается?

2 Как запустить скрипт по времени?

12 Лабораторная работа №12. Реализация сложных сценариев

Цель работы: приобретение навыков работы по настройке работы ОС путем программирования сложных скриптов разных структур.

12.1 Рабочее задание

Написать и запустить скрипты:

- показать работу арифметического сравнения;
- показать работу по чтению из файлов, сравнения количества строк;
- вывести в цикле на экран и в файл числа от 10 до 20;
- привести пример работы 2 циклов в 1 скрипте;
- получить системную информацию, вывести на экран и завершить работу скрипта выводом циклического сообщения;
- с использованием условных команд;
- с использованием группировки команд;
- с использованием обработки пользовательского ввода и вывода файлов;
- циклический показ с очисткой экрана и выводом календаря текущего месяца;
- вывести на дисплей полную информацию обо всех процессах, найти строки, содержащие последовательность символов «ваша_фамилия», посчитать их и записать их файл;
- в параметрах скрипта передаются две строки. Вывести сообщение о равенстве или неравенстве переданных строк;
- в параметрах при запуске скрипта передаются три целых числа. Вывести максимальное из них;
- пример работы с массивами;
- считать с клавиатуры целые числа, пока не будет введено четное число, после этого вывести количество считанных чисел;
- показать группировку операций по работе с файлами и строками;
- создать текстовое меню с четырьмя пунктами. При вводе пользователем номера пункта меню происходит запуск редактора *gedit*, редактора на ваш выбор, браузера *links* или выход из меню;
- запрашивать имя файла, если файл не существует, выводит сообщение об ошибке, но, если файл существует, выводит информацию о файле в формате: имя файла (не включая путь к файлу), тип файла, размер файла, владелец файла, права доступа, дата создания файла;
- работа с архивами, который запрашивает тип действия: разархивировать или заархивировать; для архивации: запрашивает каталог для архивации и имя архива, создаёт архив с этим именем; для распаковки: спрашивает имя файла с архивом и распаковывает его;
- выводит имя текущего каталога и запрашивает первое и второе расширение файлов, находит в текущем каталоге все файлы с первым

расширением и меняет их расширение на второе расширение. Если таких файлов не существует, выводит сообщение об ошибке и начинает сначала;

- работа с паролями, запрашивает имя пользователя; запрашивает тип действия для данного пользователя: заблокировать или разблокировать и блокирует или разблокирует данного пользователя;

- посмотреть список запущенных процессов, дать ответ запущено ли определенное приложение;

- посчитать количество процессов, запущенных пользователем user, и вывести в файл пары *PID*:команда для таких процессов;

- скрипт, по результату выполнения которого может возникнуть принскрин, представленный на рисунке 45.

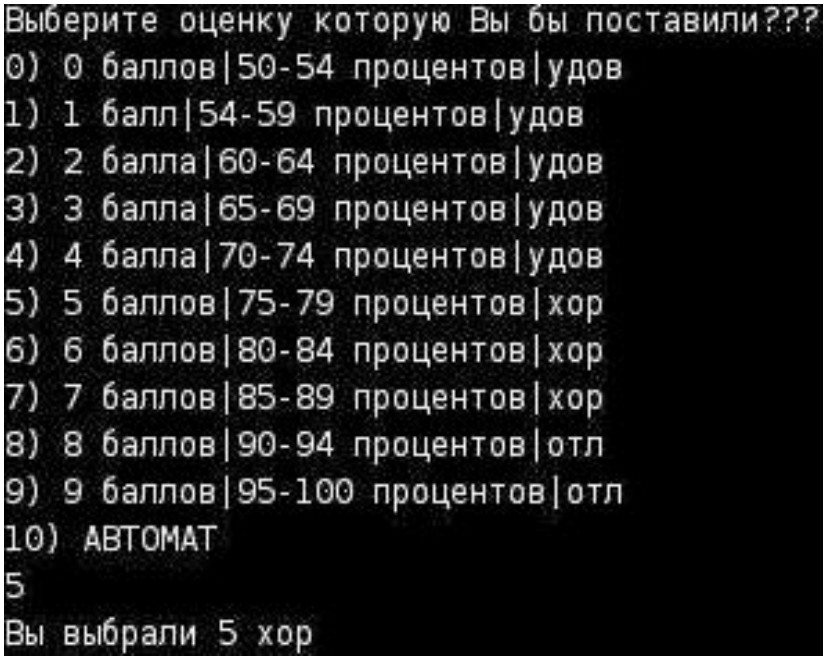

Рисунок 45 – Принскрин результата работы скрипта

12.2 Методические указания к выполнению лабораторной работы

Пример 1. Организовать циклический показ информации по процессам и очистку *экрана (Ctrl+Z – выход):*

```
while clear
do
ps -a
done
```

Пример 2. Организовать циклический просмотр списка файлов и выдачу сообщения (est) при появлении заданного имени (*primer6.sh*) в списке:

```
while ls
do
        find primer6.sh && echo est
done
```

Формат условного оператора *if*:

if <условие>
then
list1
else
list2
fi

Оператор цикла с условием *while true* и *while false*. Команда *while* (пока) формирует циклы, выполняющиеся до тех пор, пока команда *while* определяет значение следующего за ним выражения как *true* или *false*. Формат оператора цикла с условием *while true*:

while list1
do
list2
done

Здесь *list1* и *list2* - списки команд. *While* проверяет код возврата списка команд, стоящих после *while*, и если его значение равно 0, то выполняется команды, стоящие между *do* и *done*.

Оператор цикла с условием *while false* имеет формат:

until list1
do
list2
done

В отличие от предыдущего случая условием выполнения команд между *do* и *done* является ненулевое значение возврата. Программный цикл может быть размещен внутри другого цикла (вложенный цикл). Оператор *break* прерывает ближайший к нему цикл. Если в программу ввести оператор *break* с уровнем 2 (*break* 2), то это обеспечит выход за пределы двух циклов и завершение программы.

Оператор цикла с перечислением *for*:

for name in [wordlist]
do
list
done

где *name* - переменная;
wordlist - последовательность слов;
list - список команд.

Переменная *name* получает значение первого слова последовательности *wordlist*, после этого выполняется список команд, стоящий между *do* и *done*. Затем name получает значение второго слова *wordlist* и снова выполняется список *list*. Выполнение прекращается после того, как кончится список *wordlist*.

Для работы с файлами и символами в них существует некая группировка операций:

- команда > файл – перенаправление стандартного вывода в файл, содержимое существующего файла удаляется;

- команда >> файл – перенаправление стандартного вывода в файл, поток дописывается в конец файла;

- команда1 | команда2 – перенаправление стандартного вывода первой команды на стандартный ввод второй команды = образование конвейера команд;

- команда1 $(команда2) – передача вывода команды 2 в качестве параметров при запуске команды 1. Внутри скрипта конструкция $(команда2) может использоваться, например, для передачи результатов работы команды 2 в параметры цикла *for … in*.

Работа со строками (внутренние команды *bash*):

- ${*#string*} – выводит длину строки (*string* – имя переменной);

- ${*string:position:length*} – извлекает $*length* символов из $*string*, начиная с позиции $*position*. Частный случай: ${*string:position*} извлекает подстроку из $*string*, начиная с позиции $*position*;

- ${*string#substring*} – удаляет самую короткую из найденных подстрок $*substring* в строке $*string*. Поиск ведется с начала строки, $*substring* – регулярное выражение;

- ${*string##substring*} – удаляет самую длинную из найденных подстрок $*substring* в строке $*string*. Поиск ведется с начала строки, $*substring* – регулярное выражение;

- ${*string/substring/replacement*} – замещает первое вхождение $*substring* строкой $*replacement*, $*substring* – регулярное выражение;

- ${*string//substring/replacement*} – замещает все вхождения $*substring* строкой $*replacement*, $*substring* – регулярное выражение.

В таблице 6 даны листинги скриптов и результат их выполнения.

Таблица 6 – Листинги скриптов с выполнением

Листинг	Результат выполнения

циклы
for a in `seq 1 5`; do
  echo "Proshlo $a sec from Start"
  sleep 1;
done
echo "Final"
	Proshlo 1 sec from Start
	Proshlo 2 sec from Start
	Proshlo 3 sec from Start
	Proshlo 4 sec from Start
	Proshlo 5 sec from Start
	Final

массивы #!/bin/bash ARRAY=('Pervyi Element' 'Vtoroi' 'Tretii' '4-yi') ELEMENTS=${#ARRAY[@]} for ((i=0;i<$ELEMENTS;i++)); do echo ${ARRAY[${i}]} done	Pervyi Element Vtoroi Tretii 4-yi

чтение из файла 1 #!/bin/bash declare -a ARRAY # exec <filename с клавы stdin в файл exec 10<1 let count=0 while read LINE <&10; do ARRAY[$count]=$LINE ((count++)) done echo Number of elements: ${#ARRAY[@]} echo ${ARRAY[@]} # закрываем файл exec 10>&-	Number of elements: 13 #!/bin/bash declare -a ARRAY # exec <filename с клавы stdin в файл exec 10<1 let count=0 while read LINE <&10; do ARRAY[$count]=$LINE ((count++)) done echo Number of elements: ${#ARRAY[@]} echo ${ARRAY[@]} # закрываем файл exec 10>&-

условие если-иначе
#!/bin/bash
a=4
echo "1. Bash"
echo "2. Scripting"
echo "3. Tutorial"
echo -n "Please choose number [1,2 or 3] "
while [$a -eq 4]; do
read a
if [$a -eq 1] ; then
echo "You have chosen: Bash"
else
if [$a -eq 2] ; then
echo "You have chosen: Scripting"
else
if [$a -eq 3] ; then
echo "You have chosen: Tutorial"
else
echo "Please make a choice between 1-3 !"
echo "1. Bash"
echo "2. Scripting"
echo "3. Tutorial"
echo -n "Please choose a word [1,2 or 3]? "
a=4
fi
fi
fi
done
	1. Bash
	2. Scripting
	3. Tutorial
	Please choose number [1,2 or 3] 5
	Please make a choice between 1-3 !
	1. Bash
	2. Scripting
	3. Tutorial
	Please choose a word [1,2 or 3]? 3
	You have chosen: Tutorial

арфметические сравнения -lt < -gt > -le <= -ge >= -eq == -ne != #!/bin/bash a=2 $$ b=2	Both Values are equal

if [$a -eq $b]; then echo "Both Values are equal" else echo "Values are NOT equal" fi	

сравнение переменных
#!/bin/bash
a=2
b=1
if [$a -eq $b]; then
echo "Both Values are equal"
elif [$a -gt $b]; then
echo "$a is greater then $b"
else
echo "$b is greater then $a"
fi
	2 is greater then 1

символьно-текстовые сравнения
= одинаковые
!= не одинаковые
< меньще чем
> больше чем
-n s1 переменная s1 не пустая
-z s1 переменная s1 пустая
#!/bin/bash
a="Bash"
b="Scripting"
if [$a = $b]; then
echo "Both Strings are equal"
else
echo "Strings are NOT equal"
fi
	Strings are NOT equal

цикл for
#!/bin/bash
for f in $(ls /var/); do
echo $f
done
 	
или
for ((value=1 ; value<5; value++)) do
echo “Like it”
done
	backups
	cache
	lib
	local
	log
	mail
	opt
	run
	spool
	tmp

цикл while #!/bin/bash a=6 while [$a -gt 0]; do echo Value of count is: $a let a=a-1 done	Value of count is: 6 Value of count is: 5 Value of count is: 4 Value of count is: 3 Value of count is: 2 Value of count is: 1
until цикл #!/bin/bash a=0	Value of count is: 0 Value of count is: 1 Value of count is: 2

until [$a -lt 5]; do echo Value of count is: $a let a=a+1 done	Value of count is: 3 Value of count is: 4 Value of count is: 5
оператор выбора select #!/bin/bash a='Choose one word: ' select b in "linux" "bash" "scripting" "tutorial" do echo "The word you have selected is: $b" break done exit 0	1) linux 2) bash 3) scripting 4) tutorial #? 2 The word you have selected is: bash
оператор выбора case #!/bin/bash echo "What is your lovely language?" echo "1) bash" echo "2) perl" echo "3) python" echo "4) nothing" read a; case $a in 1) echo "You selected bash";; 2) echo "You selected perl";; 3) echo "You selected python";; 4) exit esac	What is your lovely language? 1) bash 2) perl 3) python 4) nothing 4 What is your lovely language? 1) bash 2) perl 3) python 4) nothing 3 You selected python
#!/bin/bash echo "Выберите редатор для запуска:" echo "1 Запуск программы nano" echo "2 Запуск программы vi" echo "3 Выход" read a # читаем в $a со стандартного ввода case $a in 1) /usr/bin/nano ;; # если $a = 1, то запустить nano 2) /usr/bin/vi ;; # если $a = 2, то запустить vi 3) exit 0 ;; *) #если введено с клавиатуры то, что в case не описывается, выполнять следующее: echo "Введено неправильное действие" esac #окончание оператора case.	Выберите редатор для запуска: 1 Запуск программы nano 2 Запуск программы vi 3 Выход После выбор цифры и нажатия Enter запуститься тот редактор, который вы выбрали (если конечно все пути указаны правильно, и у вас установлены эти редакторы)

12.3 Список контрольных вопросов

1 Что такое циклический вывод?
2 Как с клавиатуры отследить ввод данных?

Список литературы

Основная

1 Таненбаум Эндрю, Бос Х. Современные операционные системы – 4-е изд. – СПб.: Питер, 2015. – 1120 с.: ил. – (Классика computer science).

2 Мэтью Нейл. Основы программирования в Linux: руководство: пер. с англ. / Мэтью Нейл, Стоунс Ричард. – 4–е изд. – СПб.: БХВ–Петербург, 2009. –896 с.: ил.

3 Мэйволд Эрик. Безопасность сетей: Практ. пособие: Самоучитель / Мэйволд Эрик.- М.: СП ЭКОМ: БИНОМ, 2005. – 528 с.: ил. – (Шаг за шагом)

4 Олифер В. Г. Компьютерные сети. Принципы, технологии, протоколы: Учебник для Вузов. / Олифер В. Г., Олифер Н. А.- СПб.: Питер, 2010. – 943 с.: ил.

5 Басс Лен. Архитектура программного обеспечения на практике: Практ. пособие / Басс Лен, Клементс П., Кацман Р. – 2–е изд. – СПб.: Питер, 2006. – 575 с.: ил. – (Классика computer science).

6 Гордеев А.В., Молчанов А.Ю. Системное программное обеспечение. – СПб.: Питер, 2003. - 736 с.: ил.

7 Дейтел Харвин М. Операционные системы. Основы и принципы: Пер. с англ. - 3-е изд.- М.: БИНОМ, 2006. - 1024 с.: ил.

8 Олифер В.Г., Олифер Н.А. Сетевые операционные системы. - СПб.: Питер, 2009. - 672с.: ил.

9 Эви Немет, Гарт Снайдер и др. UNIX. Руководство системного администратора. – Киев: Вильямс, 2012. – 1312 с.

10 Курячий Г.В., Маслинский К.А. Операционная система Linux. – М.: Интуит.Ру, 2005. – 392 с.: ил.

11 Безбогов А.А., Мартемьянов Ю. Безопасность операционных систем. изд. Машиностроение-1, 2007. – 220 с.

Дополнительная

12 Курячий Г.В. Операционная система UNIX. – М.: Интуит.Ру, 2004. – 292 с.: ил.

13 Рейчардс К., Фостер-Джонсон Э. UNIX: справочник. – СПб.: Питер Ком, 1999. – 384 с.: ил.

14 Глушаков С.В. и др. Сетевые технологии WINDOWS NT. – Харьков: Фолио; М.: ООО «Издательство АСТ», 2001. – 501 с.

15 Стивенс У. UNIX, взаимодействие процессов. – М.: Питер, 2002. – 576 с.

16 Руссинович М.- Внутреннее устройство Microsoft Windows. Питер, 2013. - 800 с.

Сводный план 2017 г., поз. 115

Зуева Екатерина Александровна

БЕЗОПАСНОСТЬ ОПЕРАЦИОННЫХ СИСТЕМ

Методические указания по выполнению лабораторных работ
для студентов специальности
5В100200 – Системы информационной безопасности

Редактор Л.Т. Сластихина
Специалист по стандартизации Н.К. Молдабекова

Подписано в печать __ __ _____ Формат 60х84 1/17.
Тираж 20 экз. Бумага типографская №1
Объем 4,25 уч.-изд.л. Заказ № __ Цена ___ тенге

Копировально-множительное бюро
некоммерческого акционерного общества
«Алматинский университет энергетики и связи»
050013, Алматы, ул. Байтурсынова, 126