МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический
университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)
Чистопольский филиал «Восток»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

ОПЕРАЦИОННЫЕ СИСТЕМЫ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Индекс по учебному плану: **Б1.В.ДВ.05.02**
Направление подготовки: **09.03.01 Информатика и вычислительная техника**
Квалификация: **Бакалавр**
Профиль подготовки: **Вычислительные машины, комплексы, системы и сети**
Типы задач профессиональной деятельности: **проектный, производственно-технологический**

Рекомендовано УМК ЧФ КНИТУ-КАИ

Чистополь
2023 г.

Лабораторная работа №1

Работа с командной строкой

Linux

Цель работы: знакомство с командной строкой Linux, изучение основных команд (программ) Linux.

Порядок выполнения работы

Запустить Konsole. Эта программа предназначена для выполнения функций командной строки ОС Linux. Здесь в интерактивном режиме вы можете выполнять любые команды и программы, зарегистрированные в системе. Кратко перечислим основные команды (программы) ОС Linux.

1. Вывод справки по команде (man)

```
man <имя изучаемой команды>
```

man (от англ. manual — руководство) — команда Unix, предназначенная для форматирования и вывода справочных страниц.

2. Список работающих процессов (top)

```
top
```

консольная команда UNIX-совместимых операционных систем, список работающих в данный момент процессов и информацию о них. Команда top показывает список работающих в данный момент процессов и информацию о них, включая использование ими памяти и процессора. Список интерактивно формируется в реальном времени.

Чтобы выйти из программы top, нажмите клавишу [q].

3. Количество памяти (free)

```
free [-b | -k | -m] [-o] [-s delay ] [-t] [-V]
```

Показывает общее количество свободной и используемой физической памяти и памяти отведенной для свопирования в системе, так же и совместно используемую память и буфера используемые ядром.

Опции :

`-b` показывает количество памяти в байтах; опция -k (по умолчанию) поазывает количество пвмяти в килобайтах; Опция -m показывает количество памяти в мегабайтах.

`-t` показывает строки содержащие полное количество памяти.

`-o` запрещает показывать строки относящиеся к "массиву буфера" . Если не определено отнять/добавить память буферов из/в используемую/свободнуб память (соответственно!).

`-s` разрешает безостановочно выводить информацию с промежутком в *delay* секунд.
`-V` показывает информацию о версии программы.

4. Отчет о работающих процессах (ps)

```
ps [опции]
```

Команда ps выводит в стандартный вывод информацию о текущем состоянии процессов. Опции

`-a` все терминальные процессы
`-e` все процессы.
`-g` выбирать процессы по *списку* лидеров групп.
`-p` выбирать процессы по *списку* идентификаторов процессов.
`-t` выбирать процессы по *списку* терминалов.
`-u` выбирать процессы по *списку* идентификаторов пользователей.
`-f` генерировать полный листинг.
`-l` генерировать листинг в длинном формате.

5.Выдача информации о файлах или каталогах (ls)

Синтаксис команды:

```
ls [флаги] [имя ...]
```

Команда ls для каждого имени каталога распечатывает список входящих в этот каталог файлов; для файлов - повторяется имя файла и выводится дополнительная информация в соответствии с указанными флагами. По умолчанию имена файлов выводятся в алфавитном порядке. Если имена не заданы, выдается содержимое текущего каталога. Если заданы несколько аргументов, то они сортируются по алфавиту, однако сначала всегда идут файлы, а потом каталоги с их содержимым.

6. Выдача имени текущего каталога (pwd)

Синтаксис команды:

```
pwd
```

Бывает, что при ее изучении, вы попадаете в какой-то каталог, про который уже не помните, как он называется и как вы в него попали. Узнать его полное имя позволяет команда pwd.

7.Смена текущего каталога (cd)

Синтаксис команды:

```
cd [каталог]
```

Команда cd применяется для того, чтобы сделать заданный каталог текущим. Если каталог не указан, используется значение переменной окружения $HOME (обычно это каталог, в который Вы попадаете сразу после входа в систему).

Если каталог задан полным маршрутным именем, он становится текущим. По отношению к новому каталогу нужно иметь право на выполнение, которое в данном случае трактуется как разрешение на поиск.

8. Изменение режима доступа к файлам (chmod)

Синтаксис команды:

```
chmod режим файл
```

Права доступа к указанным файлам (среди которых могут быть каталоги) изменяются в соответствии с указанным режимом. Режим может быть задан в абсолютном или символьном виде.

Использование символьного вида основано на однобуквенных обозначениях, которые определяют класс доступа и права доступа для членов данного класса. Права доступа к файлу зависят от идентификатора пользователя и идентификатора группы, в которую он входит. Режим в целом описывается в терминах трех последовательностей, по три буквы в каждой:

Владелец	Группа	Прочие
(u)	(g)	(o)
rwx	rwx	rwx

Здесь владелец, члены группы и все прочие пользователи обладают правами чтения файла, записи в него и его выполнения. В примере показаны обозначения как для класса доступа, так и для прав доступа внутри класса.

Для задания режима доступа в символьном виде используется следующий синтаксис:

```
[кому] операция права
```

Часть [кому] есть комбинация букв u, g и o (владелец, члены группы и прочие пользователи соответственно). Если часть кому опущена или указано a, то это эквивалентно ugo.

Операция может быть: + (добавить право), - (лишить права), = (в пределах данного класса присвоить права абсолютно, то есть добавить указанные права и отнять неуказанные).

Права - любая осмысленная комбинация следующих букв:

r Право на чтение.

w Право на запись.

x Право на выполнение (поиск в каталоге).

s При выполнении переустанавливать действующий идентификатор пользователя или группы.

t После выполнения программы сохранять сегмент команд (бит навязчивости).

l Учет блокировки доступа.

Опустить часть права можно только если операция есть = (для лишения всех прав).

Если надо сделать более одного указания об изменении прав, то при использовании символьного вида в правах не должно быть пробелов, а указания должны разделяться запятыми. Например, команда chmod u+w,go+x f1 добавит для владельца право писать в файл f1, а для членов группы и прочих пользователей - право выполнять файл. Права устанавливаются в указанном порядке. Право s можно добавлять только для пользователя и группы, право t - только для пользователя.

Чтобы установить права, позволяющие владельцу читать и писать в файл, а членам группы и прочим пользователям только читать, надо использовать следующую запись:

```
chmod u=rw,go=r f1
```

Позволить всем выполнять файл f2

```
chmod +x f2
```

9. Копирование файлов (cp)

```
cp файл1 [файл2 ...] целевой_файл
```

Команда cp копирует файл1 в целевой_файл. Файл1 не должен совпадать с целевым_файлом (будьте внимательны при использовании метасимволов shell'a). Если целевой_файл является каталогом, то файл1, файл2, ..., копируются в него под своими именами. Только в этом случае можно указывать несколько исходных файлов.

Если целевой_файл существует и не является каталогом, его старое содержимое теряется. Режим, владелец и группа целевого_файла при этом не меняются.

Если целевой_файл не существует или является каталогом, новые файлы создаются с теми же режимами, что и исходные (кроме бита навязчивости, если Вы не суперпользователь). Время последней модификации целевого_файла (и последнего доступа, если он не существовал), а также время последнего доступа к исходным файлам устанавливается равным времени, когда выполняется копирование. Если целевой_файл был ссылкой на другой файл, все ссылки сохраняются, а содержимое файла изменяется.

10. Перемещение (переименование) файлов (mv)

Синтаксис команды:

```
mv [-f] файл1 [файл2 ...] целевой_файл
```

Команда mv перемещает (переименовывает) файл1 в целевой_файл. Файл1 не должен совпадать с целевым_файлом (будьте внимательны при использовании метасимволов shell'a). Если целевой_файл является каталогом, то файл1, файл2, ..., перемещаются в него под своими именами. Только в этом случае можно указывать несколько исходных файлов.

Если целевой_файл существует и не является каталогом, его старое содержимое теряется. Если при этом обнаруживается, что в целевой_файл не разрешена запись, то выводится режим этого файла [см. chmod] и запрашивается строка со стандартного ввода. Если эта строка начинается с символа y, то требуемые действия все же выполняются, при условии, что у пользователя достаточно прав для удаления целевого_файла. Если была указана опция -f или стандартный ввод назначен не на терминал, то требуемые действия выполняются без всяких запросов. Вместе с содержимым целевой_файл наследует режим файла1.

Если файл1 является каталогом, то он переименовывается в целевой_файл, только если у этих двух каталогов общий надкаталог; при этом все файлы, находившиеся в файле1, перемещаются под своими именами в целевой_файл. Если файл1 является файлом, а целевой_файл - ссылкой, причем не единственной, на другой файл, то все остальные ссылки сохраняются, а целевой_файл становится новым независимым файлом.

11. Удаление файлов (rm)

Синтаксис команды: `rm [-f] [-i] файл ...`
`rm -r [-f] [-i] каталог`

Команда rm служит для удаления указанных имен файлов из каталога. Если заданное имя было последней ссылкой на файл, то файл уничтожается. Для удаления пользователь должен обладать правом записи в каталог; иметь право на чтение или запись файла не обязательно. Следует заметить, что при удалении файла в Linux, он удаляется навсегда. Здесь нет возможностей вроде "мусорной корзины" в windows 95/98/NT или команды undelete в DOS. Так что, если файл удален, то он удален!

Если нет права на запись в файл и стандартный ввод назначен на терминал, то выдается (в восьмеричном виде) режим доступа к файлу и запрашивается подтверждение; если оно начинается с буквы y, то файл удаляется, иначе - нет. Если стандартный ввод назначен не на терминал, команда rm ведет себя так же, как при наличии опции -f.

Допускаются следующие три опции:

`-f` Команда не выдает сообщений, когда удаляемый файл не существует, не запрашивает подтверждения при удалении файлов, на запись в которые нет прав. Если нет права и на запись в каталог, файлы не удаляются. Сообщение об ошибке выдается лишь при попытке удалить каталог, на запись в который нет прав (см. опцию -r).

`-r` Происходит рекурсивное удаление всех каталогов и подкаталогов, перечисленных в списке аргументов. Сначала каталоги опустошаются, затем удаляются. Подтверждение при удалении файлов, на запись в которые нет прав, не запрашивается, если задана опция -f или стандартный ввод не

назначен на терминал и не задана опция -i. При удалении непустых каталогов команда rm -r предпочтительнее команды rmdir, так как последняя способна удалить только пустой каталог. Но команда rm -r может доставить немало острых впечатлений при ошибочном указании каталога!

`-i` Перед удалением каждого файла запрашивается подтверждение. Опция -i устраняет действие опции -f; она действует даже тогда, когда стандартный ввод не назначен на терминал.

ПРИМЕРЫ Опция -i часто используется совместно с -r. По команде:

```
rm -ir dirname
```

запрашивается подтверждение:

```
directory dirname: ?
```

При положительном ответе запрашиваются подтверждения на удаление всех содержащихся в каталоге файлов (для подкаталогов выполняются те же действия), а затем подтверждение на удаление самого каталога.

12.Удаление каталогов (rmdir)

Синтаксис команды: `rmdir [-p] [-s] каталог ...`

Команда rmdir удаляет указанные каталоги, которые должны быть пустыми. Для удаления каталога вместе с содержимым следует воспользоваться командой rm с опцией -r. Текущий каталог [см. pwd] не должен принадлежать поддереву иерархии файлов с корнем - удаляемым каталогом.

Для удаления каталогов нужно иметь те же права доступа, что и в случае удаления обычных файлов [см. rm].

Командой rmdir обрабатываются следующие опции:

`-p` Позволяет удалить каталог и вышележащие каталоги, оказавшиеся пустыми. На стандартный вывод выдается сообщение об удалении всех указанных в маршруте каталогов или о сохранении части из них по каким-либо причинам.

`-s` Подавление сообщения, выдаваемого при действии опции -p.

13.Создание ссылки на файл (ln)

Синтаксис команды:

`ln [-f] файл1 [файл2 ...] целевой_файл` Команда ln делает целевой_файл ссылкой на файл1. Файл1 не должен совпадать с целевым_файлом (будьте внимательны при использовании метасимволов shell'a). Если целевой_файл является каталогом, то в нем создаются ссылки на файл1, файл2, ... с теми же именами. Только в этом случае можно указывать несколько исходных файлов.

Если целевой_файл существует и не является каталогом, его старое содержимое теряется. Если при этом обнаруживается, что в целевой_файл не разрешена запись, то выводится режим доступа к этому файлу [см. chmod] и запрашивается строка со стандартного ввода. Если эта строка начинается с символа y, то требуемые действия все же выполняются, при условии что у пользователя достаточно прав для удаления целевого_файла. Если была указана опция -f или стандартный ввод назначен не на терминал, то требуемые действия выполняются без всяких запросов. Целевой_файл наследует режим доступа к файлу1.

Команда ln не создает ссылок между разными файловыми системами, поскольку они (файловые системы) могут добавляться и удаляться.

14.Создание каталога (mkdir)

`mkdir [-m режим_доступа] [-p] каталог ...`

По команде mkdir создается дин или несколько каталогов с режимом доступа 0777 [возможно измененном с учетом umask и опции -m]. Стандартные файлы (. - для самого каталога и .. - для вышележащего) создаются автоматически; их нельзя создать по имени. Для создания каталога необходимо располагать правом записи в вышележащий каталог.

Идентификаторы владельца и группы новых каталогов устанавливаются соответственно равными реальным идентификаторам владельца и группы процесса.

Командой mkdir обрабатываются две опции:

`-m` режим_доступа - (явное задание режима_доступа для создаваемых каталогов [см. chmod]).

`-p` (при указании этой опции перед созданием нового каталога предварительно создаются все несуществующие вышележащие каталоги).

15.Вывод аргументов в стандартный поток вывода(echo)

`echo [опции] [`*`string`* `...]`

`-n` не выводить завершающий символ новой строки.

`-e` раздрешить интерпретацию следующих backslashescaped последовательностей в строках:

`\a` alert (звонок)
`\b` backspace
`\c` запретить завершающий символ новой строки
`\f` перегон страницы
`\n` новая строка
`\r` перевод строки
`\t` горизонтальная табуляция

\v вертикальная табуляция
\\ обратный слэш

Команда `echo` предназначена для выдачи на стандартный вывод строки символов, которая задана ей в качестве аргумента.

Передаваемая строка может быть перенеправлена в файл с использованием оператора перенаправления вывода `> . Например:`

```
$echo "Hello, world!" > myfile
```

Задание на лабораторную работу.

1.Ознакомиться с командами Linux. Выполнить команды top, free, ps с различными опциями.

2. Войти в свой домашний каталог. Для этого нужно сделать команду `cd ~` Вы находитесь в своем рабочем каталоге. Здесь хранятся ваши пользовательские файлы и настройки программ, которые вы используете.

3. Создать следующую структуру каталогов и файлов

- в домашнем каталоге создать каталог **inform**
- Перейти в каталог и **inform** создать в нем каталог **lab1**
- Внутри каталога **lab1** создать каталог **catalog1**, файл **file1(например, используя команду echo)**, каталог **catalog2. Перейти в каталог catalog2.**
- Внутри каталога **catalog2** создать файлы **file3 и file4** , каталог **catalog3**
- Внутри каталога **catalog3** создать файл **file5,** жесткую ссылку на файл **file1,** жесткую ссылку на каталог **catalog2.**
- Создать в каталоге **lab1** символичесткую ссылку **s_link** на файл **file5**

4.Запустить программу MC (Midnight Commander):
`mc` Здесь вы можете посмотреть структуру созданных вами каталогов и просмотреть содержимое файлов.

Лабораторная работа №2

Администрирование пользователей

Цель работы: получить навыки администрирования пользователей, а так же изучить смежные вопросы.

Вступление.

В данной лабораторной работе студенты должны получить навыки управления пользователями, группами, паролями, изучить механизмы получения особых привилегий и установки дисковых квот.

Ход работы:

В ходе работы необходимо изучить теоретические моменты, связанные с администрированием пользователей, а также проделать практические задания, описанные ниже.

1. **Механизм учетных записей.**

   В ОС Suse Linux существует три типа пользователей: пользователь **root**, обычные пользователи и системные пользователи. Каждый пользователь имеет в системе учетную запись. Информация об учетных записях хранится в текстовом файле `/etc/passwd.` Зашифрованные пароли обычно хранятся `/etc/shadow.`

   Системный пользователь - это не человек, а процесс, выполняющийся на компьютере. В отличии от обычных пользователей системные пользователи не имеют начальных каталогов и паролей, поэтому в систему нельзя войти под именем системного пользователя.

2. **Идентификаторы пользователей и групп.**

   Компьютер - это машина, работающая с числами. Он идентифицирует пользователей по номерам, известным, как идентификатор пользователя (UID) и идентификатор группы (GID).

   Пользователь `root` имеет неограниченные права в системе, его UID, GID равны 0.

   Идентификаторы в диапазоне от 1 до 499 и 65534 зарезервированы для системных пользователей. Идентификаторы для людей начинаются с 500.

3. **Права доступа**

   Права доступа бывают трех видов: чтение (read), запись (write), выполнение (execute), а также каждый вид прав имеет цифровой аналог 4, 2, 1 соответственно. При наличии нескольких видов прав одновременно цифры суммируются.

   Команды изменения прав группы, пользователя или доступа к файлу или каталогу:

   - `chgrp -` изменение принадлежности файла или каталога к

определенной группе

- `chown` – изменяет владельца файла или каталога
- `chmod` – изменяет режим доступа к файлу или каталогу. Если при помощи команды `chmod g+s [имя каталога].`

4. Группы пользователей

Список групп содержится в файле `/etc/group.`

Ниже представлены наиболее часто используемые инструменты командной строки для управление группами:

- `groupadd` - создание новой группы;
- `groupde` **1** - удаление существующей группы;
- `groupmod` - модификация параметров группы (ключи: **-g, -n**)
- `gpasswd` - создание пароля группы (ключ: **-A** -создание администратора группы);
- `useradd -G` - использование команды с данным аргументом позволяет добавить пользователя к определенной группе при создании учетной записи;
- `usermod -G` - добавление пользователя к группе;
- `grpck` - проверка файла `/etc/group` на ошибки.

Система Suse Linux предоставляет администратору и графический интерфейс `system-config-users`, но опытные администраторы предпочитают использовать консоль.

5. Управление пользователями

Первым делом необходимо создать учетную запись пользователя с предоставлением UID, создать начальный каталог, поместить туда стандартный набор файлов. Во-вторых, пользователя следует отнести к определенным группам и определить, какой объем дискового пространства он может использовать.

В Suse имеется несколько инструментов командной строки для управления пользователями, такие как `useradd`, `userdel`, `passwd`, `usermod`. При создании пользователя в каталоге `/etc/skel` содержится набор файлов, который помещается в начальный каталог пользователя.

- `useradd` - создание нового пользователя;
- `userdel` - используется для удаления учетной записи пользователя, при этом будет удален и начальный каталог пользователя;
- `passwd` - задает пароль пользователя (ключ: **-l** - заблокировать учетную запись пользователя);
- `usermod` - команда изменяет атрибуты пользователя (ключи: **-s, -**

u).

- `useradd user -p password -и 1000.`

6. Механизмы получения особых привилегий

Бывают случаи, когда обычным пользователям требуется запускать команды с правами других пользователей. При помощи команды **su** существует возможность заменить пользователя на любого другого пользователя системы. Используя команду `sudo` возможно обычному пользователю выполнять команды, доступные лишь суперпользователю. Список авторизованных пользователей содержится в файле `/etc/sudoers.`

7. Дисковые квоты

В больших системах, в которых работает много пользователей, обязательно возникает необходимость контролировать дисковое пространство, которое занимают пользователи. Для управления дисковыми квотами должен быть проинсталлирован программный пакет `quota.`

Задание. Для выполнения лабораторной работы необходимо:

1. Создать группы:
   - `workers,`
   - `teachers,`
   - `students`

2. Создать пользователей `kit_[номер варианта]_` N, где N =1, 2, .., 5, `uid` учетной записи должен быть равен `1000+N.`
Пользователей с N равным 1 и 2 добавить в группу `workers .`
Пользователей с N равным 3, 4 и 5 добавить в группу `students`

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

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

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

6. Создать текстовый файл `test_[имя студента]`, и поместить в `tests.` Файлы должны содержать скрипт на создание пользователя `user [номер варианта]` и задание ему пароля `pass [номер варианта].`

Сделайте эти файлы исполняемыми для пользователей группы `students`.

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

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

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

Лабораторная работа №3

Программирование в ОС семейства Linux

Программирование на SHELL. Использование командных файлов

Цель лабораторной работы: изучение основных возможностей языка программирования Shell с целью автоматизации процесса администрирования системы за счет написания и использования командных файлов.

Командный язык Shell – язык программирования высокого уровня. На этом языке пользователь осуществляет управление компьютером. После входа в систему Вы начинаете взаимодействовать с командной оболочкой. Shell не является единственным командным языком (хотя именно он стандартизирован в рамках POSIX – стандарта мобильных систем).

Процедура языка shell

Shell – одна из многих команд Linux. Процедура языка Shell – это командный файл. Для выполнения команд необходимо текстовый файл сделать исполнимым (с помощью команды **chmod**).

Запуск осуществляется следующим образом:

sh имя_исполняемого_файла

Структура команд

Команды в Shell имеют следующий формат:

<имя команды><флаги><аргументы>

В таблице 2.1 представлены некоторые средства группировки команд, которые могут быть использованы при создании командных файлов на shell.

Таблица 3.1 – Средства группировки команд

Средства группировки	Пояснение
;	определяет последовательное выполнение команд
&	определяет асинхронное (фоновое) выполнение команд
&&	определяет выполнение последующей команды при нормальном завершении предыдущей
||	определяет выполнение последующей команды при ненормальном завершении предыдущей

Например,

k1&&k2; k3

k2 будет выполнена при успешном выполнении k1; k3 будет выполнена после любого из исходов обработки k2

k1&&{k2;k3} - k2, k3 будут выполнены при успешном выполнении k1

{k1;k2}& - в фоновой режиме будет выполняться последовательность команд k1, k2

Перенаправление данных

Символы > >> обозначают перенаправление ввода/вывода

Например,

ls>file1 // команда ls сформирует список файлов текущего каталога и поместит его в файл file1

wc –l < file1 // команда wc подсчитает число строк файла file1 и выдаст эту информацию на экран

Можно сочетать перенаправления

wc –l < file1>file2 // команда wc подсчитает число строк файла file1 и выдаст эту информацию в файл file2

Shell-переменные

Определение переменной содержит имя и значение

var = value

Доступ к переменной осуществляется по имени (со знаком $ перед именем)

fruit = apple (определение)
echo $ fruit (доступ)
apple (результат)

Возможна конкатенация строк

fruit = apple
fruit = pine$ fruit
echo $ fruit
pineapple

Переменная может быть:

1. Частью полного имени файла, например

d = /usr/bin

2. Частью команды, например,

s = –sort +b filename‖ //наличие пробелов требует кавычек

Предопределенные переменные языка Shell

В таблице 3.2 представлены наиболее употребительные переменные.

Таблица 3.2 - Наиболее употребительные переменные

Название	Пояснение
HOME	домашний каталог пользователя
PATH	множество каталогов, в которых ОС ищет команды
PS1	первичная подсказка

Изменение значения переменной PS1 осуществляется в login-файле.

Изменение значения переменной PATH:

echo path	посмотреть
:/bin :/usr/bin	значение path
cd	домой
mkdir bin	новый каталог
echo $home	посмотреть
/users/maryann	текущий каталог
$path = :$home/bin:$path	изменение path
echo path	посмотреть
:/users/maryann/bin :/bin :/usr/bin	новое значение path

Установка переменной Shell выводом из команды

Пример1
now =‘data‘
echo $now
Sun Mart 21 12:00:01 PM 2001

Пример2
menu =‘cat file‘
echo $menu
text

Пример3
Для обеспечения видимости переменной используется команда export.
1) a = b; export a
echo $a
2) d = /home/sv; export d
echo $d
3) c = _pwd‘; export c
echo $c

Структурные операторы shell

Оператор цикла **FOR**

Синтаксис:
for <переменная> in <список значений>
do <список команд>
done

Пример
Пусть имеется командный файл makelist:
cat makelist
sort +1 people | pr –h Distribution | lpr //Сортировка по второму полю, печать заголовка, распечатка файла

Если вместо одного файла people имеется несколько adminpeople, hardpeople, softpeople, то необходимо повторить выполнение процедуры с различными файлами с помощью оператора for

Пример 1
for file in adminpeople, hardpeople, softpeople
do
sort +1 $file | pr –h Distribution | lpr
done

Пример 2
for file in *people
do
sort $file
done

Условный оператор **IF**

Синтаксис:
if <если эта команда выполняется успешно, то>
then <выполнить все последующие команды до else или, если его нет, до fi>
[else <иначе выполнить следующие команды до fi>]
fi

Пример
if test $# -eg 0
then echo −You must give a filename‖
exit 1
else sort +1 $1| …| lpr
fi
// Если значение переменной # равно нулю, выводится сообщение «You must give a filename‖, выполнение программы прекращается. Иначе выполняется команда сортировки данных.

Команда **TEST**

Эта команда применяется внутри shell-процедур.
Имеется три типа проверок: оценка числовых значений; оценка типа файла; оценка строк.
Для чисел синтаксис следующий:
N op M, где N, M –числовые переменные, операция op принимает следующие значения:

-eg	равно
-ne	не равно
-gt	больше
-lt	меньше
-ge	больше и равно
-le	меньше или равно

Для файла синтаксис такой:
op filename

операция op принимает следующие значения:

-s	файл существует и не пуст
-f	файл, а не каталог
-d	файл-директория
-w	файл для записи
-r	файл для чтения

Для строк синтаксис такой:

S op R

операция op принимает следующие значения:

=	эквивалентность
!=	неэквивалентность
-z	строка нулевой длины
-n	ненулевая длина строки

Несколько проверок могут быть объединены логическими операциями **–a** (and) и **-o** (or).

Пример 1

```
if test –w $2 -a -r $1
then cat $1>> $2
else echo –can not append‖
fi
```

Пример 2

```
echo –Vvedite a‖
read a
echo –Vvedite b‖
read b
if test $a –lt $b
then
echo –Hello‖
else
echo –…‖
fi
```

Оператор цикла **WHILE**

Синтаксис:

while <команда>
do <команда>
done

Пример 1

```
n=0
while test $n -lt 5
```

```
do
cat myfile1|lpr
n=`expr$n+1`
done
```

Пример 2

```
while test $# -gt 0
do
if test –s $1
then echo –no file $1‖> $2
else sort +1 $1 …
fi
do
```

Оператор цикла **UNTIL**

Этот оператор инвертирует условие повторения по сравнению с оператором

while
until <команда>
do <команды>
done

Пока –команда‖ не выполнится успешно, выполнять –команды‖, завершаемые ключевым словом done

Пример

```
if test $# -eg 0
then echo –Usage …‖>$2
exit
fi
until test $# -eg 0
do
if test –s $1
then echo –no file $1‖>$2
else sort …
fi
shift
done
```

Оператор выбора **CASE**

Синтаксис:

case <string> in

string1) <если string = string1, то выполнить все следующие команды до ;;> ;;

string2) <если string = string2, то выполнить все следующие команды до ;;> ;; string3) и т.д.

esac.

…

Пример 1

```
together = no
case $1 in
-t) together = yes
shift;;
-?) echo –$0: no option $1‖
exit;;
esac
if test $ together = yes
then sort …
fi
```

Пример 2

```
while true
do
        echo‖Check menu your  computer‖
1) Disk space
2) Mounted file systems
3) System name
4) Who is logged in
5) Exit

        –
echo –What do you want?‖
read number
case $number in
1) df
;;
2) mount –t msdos dev/fd0 /mnt/floppy
;;
3) uname
;;
4) who
```

```
;;
5)break
;;
*) echo –you must enter a number (1 throught 5)‖
continue
;;
esac
done
exit 0
```

На экране появляется меню. При выборе определенного пункта меню выполняется соответствующая команда.

Отладка процедур языка shell

1. Имеются три средства, позволяющие вести отладку процедур.
2. Размещение в теле процедуры команды echo для выдачи сообщений, являющихся трасой выполнения процедуры.
3. Опция –v (verbose – многословный) в команде Shell приводит к печати команды на экран перед ее выполнением.
4. Опция –x (execute) в команде Shell приводит к печати команды на экране по мере ее выполнения с заменой всех переменных их значениями.

Практическое задание к выполнению лабораторной работы на тему “Программирование на SHELL в ОС семейства UNIX”

1. Используя команды ECHO, PRINTF вывести информационные сообщения на экран.

2. Присвоить переменной A целочисленное значение. Просмотреть значение переменной A.

3. Присвоить переменной B значение переменной A. Просмотреть значение переменной B.

4. Присвоить переменной C значение —путь до своего каталога‖. Перейти в этот каталог с использованием переменной.

5. Присвоить переменной D значение —имя команды‖, а именно, команды DATE. Выполнить эту команду, используя значение переменной.

6. Присвоить переменной E значение —имя команды‖, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной.

7. Присвоить переменной F значение —имя команды‖, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной.

Написать скрипты, при запуске которых выполняются следующие действия:

8. Программа запрашивает значение переменной, а затем выводит значение этой переменной.

9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной.

10. Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) BC).,

11. Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран.

12. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки.

13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается.

14. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно.

15. Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения.

16. Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран.

17. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.

18. В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc.

19. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение.

20. Если файл есть каталог и этот каталог можно читать, просматривается содержимое этого каталога. Если каталог отсутствует, он создается. Если файл не есть каталог, просматривается содержимое файла.

21. Анализируются атрибуты файла. Если первый файл существует и используется для чтения, а второй файл существует и используется для записи, то содержимое первого файла перенаправляется во второй файл. В случае несовпадений указанных атрибутов или отсутствия файлов на экран выдаются соответствующие сообщения (использовать а) имена файлов; б) позиционные параметры).

22. Если файл запуска программы найден, программа запускается (по выбору).

23. В качестве позиционного параметра задается файл, анализируется его размер. Если размер файла больше нуля, содержимое файла сортируется по первому столбцу по возрастанию, отсортированная информация помещается в другой файл, содержимое которого затем отображается на экране.

24. Командой TAR осуществляется сборка всех текстовых файлов текущего каталога в один архивный файл my.tar, после паузы просматривается содержимое файла my.tar, затем командой GZIP архивный файл my.tar сжимается.

25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных.

Лабораторная работа №4

Сетевое администрирование Suse Linux. Протокол IP.

Задание и цель работы. Целью работы является создание сети на основе протокола IP, использующей IP-маршрутизацию и тунелирование.

Для достижения поставленной цели необходимо решить следующие задачи.

– Уяснить основные понятия транспортного-протокола (Ethernet).
– Изучить основные понятия IP-протокола.
– Изучить статическое и динамическое назначение IP-адресов.
– Изучить статическое и динамическое назначение маршрутов в протоколе IP.
– Настроить вычислительную сеть в соответствии с полученным заданием.

В ходе работы студент должен получить представление о:
– основных понятия протокола IP, настройке IP-адреса, маски сети и маршрутов;
– настройке протокола DHCP для настройки IP;
– настройке протокола RIP для динамических настроек маршрутов IP.

В результате работы студента должна быть получена работающая в соответствии с заданием сеть передачи данных.

2. Краткие теоретические сведения

2.1. Сеть передачи данных и протокол ARP

В этой работе используется виртуальная сеть передачи данных Ethernet, используемая для соединения двух и более виртуальных машин одноним сегментом широковещания. Для передачи данных в пределах этой сети используется виртуальный протокол Ethernet, использующий для адресации MAC-адреса (шесть октетов, разделенные двоеточиями), которые идентифицируют конкретный сетевой адаптер.

При передаче данных в переделах Ethernet-сети компьютеры могут либо использовать широковещание (используется, в частности, в протоколе RIP) либо передачу конкретному адресату, для чего требуется узнать его MAC-адрес по известному IP-адресу. Для решения этой задачи используется протокол ARP. Отображение IP —> MAC запоминается во временных arp-таблицах *(ARP cache)* системы.

Протокол ARP, обратный ему протокол RARP, ряд другие протоколов и протоколы передачи данных типа Ethernet относят к канальному уровню *(link layer)* в стеке TCP/IP.

2.2. Протоколы IPv4 и ICMP

Протокол IPv4 *(Internet Protocol*) позволяет передавать данные между компьютерами, находящихся в разных сетях (с точки зрения нижестоящего транспортного протокола). Для работы с протоколом IP компьютер должен

иметь как минимум один IP-адрес (четыре октета) на каждый из своих сетевых интерфейсов.

Маска сети используется для определения, не является ли IP-адрес назначения принадлежащим к той же физической (точнее, к той же широковещательной) сети. Она обозначает количество бит с начала адреса, который отводятся под адрес сети[1]. Если сети отправителя и получателя совпадают, то для связи используется протокол канального уровня. Если нет, то данный IP-пакет необходимо послать маршрутизатору.

С каждым сетевым интерфейсом может быть связано, вообще говоря, несколько пар адресов и масок сети, но обычно существует только одна на каждый интерфейс.

Каждый компьютер, участвующей в передаче по протоколу IP, имеет маршрутную таблицу, в которой указано, какие маршрутизаторы (подключенные к смежным с компьютером физическим сетям) отвечают за передачу в те или иные диапазоны IP-адресов назначения. Частным случаем в этой таблице является маршрутизатор по умолчанию, использующийся во всех неописанных явно случаях.

Таблица маршрутизации может быть описана в конфигурации сетевых подключений, а так же изменяться динамически в ходе работы компьютера.

Для посылки служебных сообщений между компьютерами (в том числе от маршрутизатора к пославшему ему пакеты компьютеру) может использоваться протокол ICMP, передаваемый поверх протокола IP Наиболее хоррошо известным его применением является команда ping.

Для передачи данных с промощью протокола IP используются работающие поверх него протоколы UDP и TCP

2.3. Динамическая настройка IP-адресов

Для динамической настройки IP-адреса и всех его параметров может использоваться протокол DHCP. Он позволяет выдавать желающим IP-адреса из некоторого указанного диапазона. Кроме адреса, клиент DHCP получает от сервера маску, адреса маршрутизаторов и DNS-серверов. DHCP использует широковещательную передачу для обмена и протокол UDP

2.4. Протокол динамической маршрутизации RIP

Протокол динамической маршрутизации RIP работает поверх UDP (обычно в режиме широковещания) и представляет простейший протокол с использованием вектора расстояния. Каждый RIP-маршрутизатор имеет отдельную таблицу известных сетей, каждая строка которой включает: адрес сети, расстояние до неѐ (в сетях) и адрес следующего маршрутизатора. При появлении нового действующего маршрута он добавляется в системную таблицу маршрутизации, а если маршрут до сети стал недоступен, то соотвест-вующая запись удаляется из системной

таблицы маршрутов. Для уменьшения вероятноти зацикливания длина маршрута 16 считается признако его недоступности.

Все маршрутизаторы регулярно рассылают свою маршрутную таблицу путѐм широковещания, обычно таблица рассылается с учѐтом правила расщепленного горизонта (split horizon). По умолчанию протокол рассылает сообщения с интервалом 30 с.

Для каждой строки таблицы существует таймер (по умолчанию 180 с.), если за данное время маршрутизатор не получает информацию о действующем маршруте, он считаются недоступными.

2.5. Пакет для создания виртуальных вычислительных сетей

Пакет WMware6.0 использует технологию виртуализации UML Каждая виртуальная машина представляет собой систему на основе Linux, в которой уже установлен весь набор необходимых для выполнения работы программ. Машины могут обмениваться данными по виртальным широковещательным сетям или подключаться к реальной сети (последнее в работе не используется). Для каждой машины в момент еѐ первого запуска создаѐтся собственный образ диска, который при необходимости может быть смонтирован в основной системе. В машине так же уже включена IP-маршутризация (как таковая), но ещѐ не произведены никакие настройки сетевых интерфейсов.

3. Выполнение задания

Каждый студент выполняет индивидуальное задание, в котором описывается структура создаваемой сети.

Для работы студенту следует использовать следующие прикладные программы:

– эмулятор терминала (например, Konsole);

– текстовый редактор (vim, mcedit, nano);

– набор служебных программ для работы с сетью (ip, arp, ping, traceroute, telnet) в составе WMware6.0.

– ПО для динамической настройки маршрутных таблиц на основе протокола RIP (Quagga) в составе WMware6.0.

– ПО для динамической настройки параметров IP на основе протокола DHCP в составе NetKit.

Рекомендуется открывать каждую виртуальную машину в отдельной закладке Konsole.

3.1. Создание и запуск виртуальной машины WMware6.0

Для запуска виртальной машины используется команда vstart, основным параметром которой выступает идентификатор машины. При отсутствии виртулаьной машины с указанным идентификатором она будет создана. Кроме того, следует указать идентификаторы виртуальных сетей, подключѐных к сетевым интерфейсам виртальной машины. Типичное использование команды vstart приведено ниже.

```
$ vstart routerl --conO=this  --ethO=netl  --ethl=net2 --eth2=net3
```

Данная команда создаѐт и запускает виртуальную машину с именем *rourterl,* три интерфейса которых подключены к трѐм виртуальным широковещательным сетям. Задание одних и тех же имѐн виртуальных сетей для разных машин позволяет организовать обмен данных между ними.

Параметр -conO=this указывает на то, что консоль машины будет вывыден на тот же тарминал, откуда она запущена. Для завершения работы машины можно, например, выполнить в ней команду shutdown.

```
# shutdown -hP now
```

Команду запуска машины рекомендуется записать в файл сценария[2]. Виртуальную машину так же можно выключить командой vcrash имя_машины.

При настройке виртуальный машины используется стандартный подход: редактирование файла конфигурации, затем перезапуск соответствующей службы[3].

3.2. Настройка сетевых интерфейсов

Обычно существует два способа задания адреса и маски сети интерфейса: «на лету», например с помощью команды ip, и перманентно, с помошью файлов настроек[4]. В работе необходим только второй способ.

В Debian для настройки IP-адресов (а также MAC-адресов, масок, и маршрутизаторов) используется файл /etc/network/interfaces. Типичный пример этого файла для компьютера с одним интерфейсом приведен ниже.

```
# это локальная петля,  еѐ адрес  127.*.*.*
auto 1o
iface lo inet loopback
# Адрес  127.0.0.1  можно не указывать
auto ethO
# статическое назначение адреса
```

[2]Если забыли про атрибуты, то можно вызывать его как sh routerl. sh

[3]В больших службах обычно работает операция reload, перечитывающая конфигурацию без остановки службы, но здесь мы еѐ применять не будем.

[4]Какие именно это файлы - увы, зависят от конкретного linux/unix.

```
# iface ethO inet static
# адрес
address  192.168.20.100
# сеть -- необязательна, достаточно маски
network 192.168.20.0
# маска
netmask 255.255.255.0
# маршрутизатор по умолчанию
gateway 192.168.20.1
```

Вот пример этого файла для маршрутизатора локальной сети. Маршрут по умолчанию связан со следующим маршрутизатором провайдера сетевых услуг.

```
auto 1o
iface lo inet loopback auto ethO
# локальная сеть iface ethO
inet static address
192.168.25.1 netmask
255.255.255.0 auto ethl #
интернет iface ethl inet static
address 89.235.148.66
network 89.235.148.64
netmask 255.255.255.248
gateway 89.235.148.65
```

После изменения этого файла следует перегрузить либо все сетевые настройки, либо только изменѐнный интерфейс.

```
# invoke-rc.d networking restart # или /etc/init.d/networking restart
# ifdown eth2;   ifup eth2 # только интерфейс eth2
```

В ряде случаев статическая настройка IP-адреса весьма неудобна. В этом случае используется протокол DHCP, благодаря которому DHCP-клиет может получить всю информацию о настройке интерфейса у DHCP-сервера.

Для включения DHCP-сервера в виртуальной машние NetKit нужно:

- включить его в список запускаемых при старте служб: update-rc. d dhcp3-server defaults;
- отредактировать файл конфигурации /etc/dhcp3/dhcpd. conf (см. ниже);
- запустить службу: invoke-rc.d dhcp3-server restart.

Пример файла конфигурации для указанного выше маршрутизатора:

```
subnet 89.235.148.64 netmask 255.255.255.248 {}; # Тут ничего не делаем.
subnet 192.168.45.0 netmask 255.255.255.0
```

```
{
   range 192.168.45.100 192.168.45.160; # диапазон адресов option routers
   192.168.45.1; # маршрутизатор (по-умолчанию)
}

# Особый      случай:
host special {
hardware ethernet 00:03:B0:64:60:A1; # Этому MAC-адресу... fixed-address
192.168.45.160; # ... всегда выдавать этот IP. }
```

DHCP-сервер быть запущен до клиента. У DHCP-клиентов при этом в файле сетевой конфигурации достаточно указать для данного интейрейса следующее.

```
auto ethO
iface ethO inet dhcp
# MAC-адрес   (для станций,   получающих определлѐный
IP)
hwaddress ether 00:0b:6a:3a:30:a6
```

К сожалению, при каждой перезагрузке виртуальная машина может получить новый MAC-адрес, поэтому при привязке IP-адреса к MAC необходимо прописывание MAC-адреса в конфигурацию клиента.

При запуске if up ethO можно наблюдать обмен DHCP-пакетами, которые передаются по канальному уровню.

3.3. Анализ сетевого трафика, тестирование сети и конфигурации компьютера

Служебная программа tcpdump позволяет выводить информацию об IP-пакетах, проходящих через сетевой стек того или иного интерфейса. Пример использования tcpdump.

```
# tcpdump -tn -i ethO # все IP-пакеты на ethO,  анализ 96-ти байт
#  tcpdump -tne -i ethO # то же,  но с MAC-адресами
# tcpdump -tnv -i eth2 -s  1518 # подробный анализ пакета,  для RIP
```

Для анализа доступности отдельных узлов может использоваться служебная программа ping, для анализа пути пакета между узлами - traceroute, работающая на основе изменения поля TTL IP-пакета.

Для показа текущих адресов и интерфейсов может быть использована команда ip a, для таблицы маршрутизации - ip r. Для работы с кешем протокола ARP используется команда arp.

3.4. Динамическая маршрутизация с помощью протокола RIP

Для поддержки протокола RIP служит программа Quagga. Для еѐ использования с протоколом RIP необходимо:

- включить его в список запускаемых при старте служб: update-rc.d quagga defaults;
- отредактировать файлы конфигурации: /etc/quagga/daemons (включить в нѐм службу ripd) и /etc/quagga/ripd.conf;
- запустить службу: invoke-red quagga restart.

Пример файла конфигурации /etc/quagga/ripd. conf.

```
hostname routerl
! пароль на сервер
password zebra
! RIP работает на двух указаных интерфейсах
router rip
network ethl
network eth2
! Уменьшаем значения таймеров для удобства опытов.
timers basic  10 30 30
! Следует распространять все известные ядру ОС маршруты
redistribute kernel
!  И ещѐ все присоединенные непосредственно   (на всякий случай)
redistribute connected
!  Файл журнала
log file /var/log/quagga/ripd.log
```

Для просмотра текущего состояния таблиц службы следует подкючится к еѐ интерфейсу, например так: telnet localhost ripd. Основная команда в интерфейсе - show ip rip.

В текущем NetKit обнаружены две проблемы, связанные с Quagga:

- для остановки службы приходится сказать killall zebra ripd, операция invoke . . . restart - не помогает[5];
- при ручном выключении (ifdown) и последующем включении интерфейса (ifup) служба RIP игнорирует пришедшие от данного интерфейса RIP-пакеты.

4. Проверка работы

В ходе выполнения работы студент должен составить отчѐт, содержащий описание конфигурации сети и еѐ иллюстрацию.

Для защиты лабораторной работы студент должен ответить на контрольный вопросы и, возможно, внести некоторые изменения в работу. Примерные варианты контрольных вопросов приведены ниже.

1.Какие задачи решает протокол ARP/IP/DHCP/RIP?

2.Почему после старта системы при работе только протокола RIP arp-таблица - пуста?

3.Продемонстрируйте изменение маршрутных таблиц и таблиц протокола RIP при отключении сетевого интерфейса маршрутизатора.

4.Какой IP-адрес используется в поле назначения при широковещательной передачи? Почему?

5.Что такое правило расщеплѐнного горизонта в RIP?

6.За сколько времени в вашем примере гарантированно происходит полный обмен маршрутной информацией (если данные в сети не теряются)?