

## СОДЕРЖАНИЕ

1. Основные требования и порядок выполнения лабораторных работ . . . . .	4
2. ЛАБОРАТОРНАЯ РАБОТА № 1	
ОС Linux: основы работы, исследование системы. . . . .	6
2.1. Цель лабораторной работы . . . . .	6
2.2. Задание на выполнение лабораторной работы. . . . .	6
2.3. Основные приемы работы . . . . .	7
2.3.1. Назначение ОС Linux . . . . .	7
2.3.2. Вход в систему и идентификация пользователя . . . . .	7
2.3.3. Добавление регистрационной записи обычного пользователя. . . . .	8
2.3.4. Удаление регистрационной записи обычного пользователя . . . . .	8
2.3.5. Текстовые и графические режимы работы. . . . .	9
2.3.6. Интерпретатор shell. Файлы и права к ним . . . . .	9
2.3.7. Обзор основных команд. . . . .	10
2.3.8. Менеджер файлов Midnight Commander . . . . .	18
2.3.9. Простейшие текстовые редакторы . . . . .	18
2.4. Вопросы к защите лабораторной работы . . . . .	19
2.5. Список рекомендуемой литературы . . . . .	19
3. ЛАБОРАТОРНАЯ РАБОТА № 2	
ОС Linux: программирование на языке командного интерпретатора. . . . .	19
3.1. Цель лабораторной работы . . . . .	19
3.2. Задание на выполнение лабораторной работы . . . . .	20
3.3. Основные приемы работы . . . . .	20
3.3.1. Программирование на языке командного интерпретатора . . . . .	20
3.3.2. Встроенные команды интерпретатора shell . . . . .	20
3.3.3. Примеры программ . . . . .	21
3.4. Вопросы к защите лабораторной работы . . . . .	24
3.5. Список рекомендуемой литературы . . . . .	24

## **1. ОСНОВНЫЕ ТРЕБОВАНИЯ И ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ**

Настоящее пособие предназначено для студентов направления 230100 (бакалавр), выполняющих лабораторные работы по дисциплине "Информатика". В пособие включены материалы по лабораторным работам № 1, 2.

Продолжительность каждой лабораторной работы - 4 часа.

Целью проведения лабораторных работ является формирование компетенций:

- ОК-1 - Владеет культурой мышления, способность к обобщению, анализу, восприятию информации, постановке цели и выбору путей ее достижения;

- ОК-11 - Осознает сущность и значение информации в развитии современного общества, владеет основными методами, способами и средствами получения, хранения и переработки информации;

- ОК-12 - Имеет навыки работы с компьютером как средством управления информацией;

- ПК-2- Осваивать методики использования программных средств для решения практических задач;

- ПК-5 - Разрабатывать компоненты программных комплексов и баз данных, использовать современные инструментальные средства и технологии программирования.

В процессе выполнения лабораторных работ осуществляется:

- закрепление основных теоретических положений, изложенных в лекциях на примере широко используемых в различных областях ОС UNIX/Linux (Red Hat и др.);

- освоение приемов и методов работы с текстовыми и графической консолями;

- освоение приемов работы с встроенным редактором программ Midnight Commander;

- изучение технологии перенаправления команд и поиска файлов;

- получение навыков настройки прав доступа к файлам;

- использование технологии программирования на языке Bourne shell.

Лабораторная работа состоит из следующих этапов:

- 1) домашняя подготовка;

- 2) выполнение работы на компьютере в соответствии с заданием;

- 3) сдача выполненной работы преподавателю на персональном компьютере;

- 4) оформление отчета;

- 5) защита лабораторной работы.

В процессе домашней подготовки студент:

- изучает лекционный материал, материалы по темам данного пособия и дополнительной литературы;
- знакомится с заданием на выполнение лабораторной работы;
- готовит отчет по выполнению лабораторной работы (пункты, отмеченные знаком \*).

Выполнение лабораторной работы производится во время занятий в классе ЛВС кафедры ВМКСС МГТУ ГА в присутствии преподавателя. В процессе выполнения лабораторной работы студент последовательно выполняет задание, фиксируя результаты в рабочей тетради или на листах формата А4. По завершению работы - демонстрирует преподавателю результаты.

Сдача работы преподавателю на персональном компьютере заключается в демонстрации выполненной работы и выполнении непосредственно при преподавателе дополнительных индивидуальных заданий.

После приема преподавателем лабораторной работы на ПК студент:

- сохраняет результаты лабораторной работы в домашнем каталоге /home/user/ в файле со своей фамилией, например, ivanov.

Отчет по каждой лабораторной работе должен содержать:

- название работы\*;
- цель лабораторной работы\*;
- задание на выполнение лабораторной работы\*;
- краткие комментарии по выполнению лабораторной работы\*;
- распечатки файлов результатов, подписанные преподавателем.

Защита лабораторной работы преподавателю проводится по контрольным вопросам и при наличии оформленного отчета. После защиты лабораторной работы делается соответствующая запись на отчете студента.

## **2. ЛАБОРАТОРНАЯ РАБОТА № 1**

### **ОС LINUX: ОСНОВЫ РАБОТЫ, ИССЛЕДОВАНИЕ СИСТЕМЫ**

#### **2.1. Цель лабораторной работы**

Целью данной работы является формирование компетенций: ОК1, ОК11, ОК12, ПК2, ПК5, в том числе получение практических навыков и умений работы в ОС Linux (Red Hat, ASPLinux), а также исследование системы.

#### **2.2. Задание на выполнение лабораторной работы**

- 1) Загрузить операционную систему. Основные этапы монтирования системы внести в отчет по лабораторной работе.
- 2) Идентифицировать себя с помощью одной из регистрационных записей обычного пользователя (*login – user, pass- user123*).
- 3) Определить количество текстовых и графических консолей. Выполнить переключение между консолями (используя горячие клавиши).
- 4) Просмотреть файлы */etc/passwd* , */etc/shadow* (под root).
- 5) Просмотреть файл */etc/group*.
- 6) Определить, находится ли пользователь *user2* в системе, если нет, то добавить, используя команду *adduser*.
- 7) Изменить владельца текущего сеанса и просмотреть информацию о процессе (использовать команды *su* и *ps* ).
- 8) Создать текстовый файл, используя команду *cat* или *touch*, установить права на исполнение (*chmod*).
- 9) Получить список файлов текущего каталога с указанием размера, времени создания и изменения, имени владельца, таблицы прав.
- 10) Вывести постранично список скрытых файлов каталога */etc* (*ls –a|less*).
- 11) Используя текстовые и графическую консоль, выполнить:
  - ввод и вывод основных команд;
  - перенаправление команд;
  - использование каналов;
  - настройку прав доступа.
- 12) Выполнить поочередно поиск файлов: */etc/passwd*, *1.txt*, а также группу файлов по шаблонам: *\*.txt*, *\*.exe*, *L\**, *L??\**, *[ab]\*a*;
- 13) Найти в текущем каталоге все файлы, в именах которых встречается хотя бы один символ в верхнем регистре.
- 14) Изучить интерфейс и выполнить основные файловые операции, используя встроенный редактор программ *Midnight Commander (mc)*.
- 15) Выполнить монтирование сменных устройств (*CD-R*, *floppy*, *flash*) средствами графической оболочки.
- 16) Скопировать на сменное устройство созданный файл (п.7 задания).
- 17) Размонтировать сменное устройство.

- 18) Проверить командой `df`, что сменное устройство размонтировано.
- 19) Провести исследование системы:
  - а) получить информацию о пользователях, находящихся в системе;
  - б) получить информацию о пользователях и процессах, недавно завершивших работу;
  - в) определить, кто и когда зарегистрировался в системе;
  - г) получить информацию о зарегистрированных в настоящий момент пользователях;
  - д) определить, сколько свободного дискового пространства и индексных дескрипторов доступно в разделе смонтированного диска;
  - е) оценить стабильность работы и загрузку системы;
  - ж) получить информацию о таблице взаимодействия процессов.
- 20) Просмотреть файл `fstab`.
- 21) Результаты исследований занести в отчет.
- 22) По окончании занятий перезагрузить систему (`reboot`).

## **2.3. Основные приемы работы**

### **2.3.1. Назначение ОС Linux**

ОС Linux - это название ядра операционной системы, базового программного обеспечения низкого уровня, которое управляет аппаратной частью компьютера. В то же время эта полноценная операционная система включает набор утилит и прикладных программ, который может варьироваться. Все компоненты системы, включая исходные тексты, распространяются с лицензией на свободное копирование и установку для неограниченного числа пользователей. Она поддерживает стандарты открытых систем и протоколы сети Internet.

В зависимости от состава программных компонент выделяют следующие дистрибутивы Linux: Red Hat, Caldera, SuSe, Debian, Slackware.

При выполнении лабораторной работы используется Red Hat Linux - многозадачная, многопользовательская сетевая операционная система. Многозадачность - очень важное достоинство Linux. Система устроена так, что под каждую задачу, выполняемую пользователем, выделяется определенное количество ресурсов. Ресурсы компьютера, такие как, например, оперативная память, не передаются приоритетной задаче (как это делается в Windows), а используются параллельно несколькими приложениями. Это повышает производительность системы и снижает риск ее «зависания».

### **2.3.2. Вход в систему и идентификация пользователя**

После полной загрузки операционной системы появится подсказка, имеющая примерно следующий вид:

```
Red Hat Linux release 6.0 (Hedwig)
Kernel 2.2.5-22 on an i686
login:
```

Далее следует ввести имя пользователя и пароль. По имени пользователя система опознает (идентифицирует) Вас как одного из пользователей, которые могут работать в системе. Для идентификации пользователя:

- введите в поле ввода *Login*: регистрационное имя обычного пользователя.

После ввода имени пользователя система выполняет аутентификацию, для чего требуется ввести пароль, соответствующий данному пользователю.

- введите в поле *Password*: пароль пользователя.

Если пароль или имя пользователя введены неправильно, система потребует снова ввести имя пользователя и пароль. При корректном имени и пароле запускается одна из консолей или графическая оболочка.

Если при установке системы не было создано ни одной регистрационной записи обычного пользователя, введите имя суперпользователя *root* и добавьте соответствующую запись.

### **2.3.3. Добавление регистрационной записи обычного пользователя**

Для того чтобы добавить регистрационную запись обычного пользователя, необходимо:

- перейти в текстовую консоль или режим эмуляции терминала и войти в систему как суперпользователь;

- в поле ввода *Имя пользователя* ввести имя пользователя, состоящее из маленьких латинских букв и цифр и начинающееся с буквы;

- в поле ввода *Полное имя* ввести обычное имя пользователя, записанное латинскими буквами (например, Ivan Ivanov);

- в поле ввода *Пароль* ввести пароль этого пользователя;

- в поле ввода *Подтвердите пароль* ввести этот же пароль повторно, далее нажать на кнопку *Добавить*.

Команда добавления пользователя *adduser* создает регистрационную запись для нового пользователя (для чего вносит изменения в ряд конфигурационных файлов системы) и создает так называемый домашний каталог */home/имя\_пользователя*, содержащий некоторые необходимые файлы и подкаталоги. Этот каталог и находящиеся в нем файлы принадлежат пользователю, и он имеет полный набор прав для работы как с существующими, так и вновь создаваемыми объектами в этом каталоге.

Рекомендуется создать хотя бы одного обычного пользователя. Не рекомендуется пользоваться правами суперпользователя без необходимости.

### **2.3.4. Удаление регистрационной записи обычного пользователя**

Чтобы удалить ошибочно созданную регистрационную запись:

- необходимо работать с правами суперпользователя;

- ввести в командной строке команду *userdel имя\_пользователя*;

Для удаления регистрационной записи, а также принадлежащих пользователю файлов необходимо ввести в командной строке команду

*userdel - r имя\_пользователя*

Эта команда удаляет каталог */home/имя\_пользователя* и все его содержимое.

Удаление регистрационной записи пользователя невозможно, если он в данный момент зарегистрирован в системе или работает какой-либо процесс, запущенный от его имени.

### **2.3.5. Текстовые и графические режимы работы**

В ходе загрузки системы создаются несколько консолей - виртуальных устройств ввода-вывода, которыми могут пользоваться различные компоненты и пользовательские программы системы. В стандартной настройке Red Hat работает с 7 виртуальными консолями (6 текстовых и 1 графическая), из которых в каждый момент времени только одна может быть связана с реальной (физической) консолью, т.е. является активной. Консоли, представляющие информацию только в текстовом виде с использованием экранных шрифтов в форматах видеосистемы компьютера, называются *текстовыми*. В таких консолях используется интерфейс командной строки. Другие консоли (*графические*) представляют информацию в графическом виде, используя Графический пользовательский интерфейс (GUI). Как правило, в одной из консолей автоматически запускается графическая среда X Windows System и графическая оболочка GNOME. Для перехода между консолями используется сочетание клавиш [CTRL]+ [ALT]+ [Fn], n - номер консоли, находится в интервале от 1 до 12. Например, чтобы сделать активной консоль с номером 4, следует нажать клавиши [CTRL]+ [ALT]+ [F4].

### **2.3.6. Интерпретатор shell. Файлы и права к ним**

Файл принадлежит создавшему его пользователю, а также группе, членом которой данный пользователь является. Пользователи, имеющие доступ к файлу, делятся на три категории:

- Владелец файла, создавший его;
- Члены группы, являющейся владельцем файла;
- Остальные пользователи.

Владелец файла может самостоятельно определять, кому позволено производить запись в файл, читать его содержимое, а также запускать файл на выполнение, если он является исполняемым. Пользователь с правами root может отменить практически все ограничения, заданные пользователем.

Доступ к созданному файлу может осуществляться тремя способами:

- путем чтения, при этом содержимое файла отображается на экране;
- путем записи, при этом содержимое файла редактируется или удаляется;
- путем выполнения, если файл содержит сценарий интерпретатора shell либо является программой.

После создания файла система сохраняет о нем всю информацию, в частности:

- раздел диска, где физически находится файл;
- размер файла;
- идентификатор владельца файла, а также тех, кому разрешен доступ к файлу;
- индексный дескриптор;

- дата и время последнего изменения файла;
- режим доступа к файлу.

Изменять режим доступа к файлам можно с помощью команды `chmod`. Аргументы этой команды могут быть заданы либо в числовом виде (абсолютный режим), либо в символьном (символьный режим).

Общий формат команды `chmod` для символьного режима:

`chmod [кто] оператор [разрешения] файл (список файлов)`

Значения параметра *кто*:

- u Владелец файла,
- g Группа, являющаяся владельцем файла,
- o остальные пользователи,
- a Все (владелец, группа и остальные пользователи)

Значения параметра *оператор*:

- + Добавление разрешения,
- Удаление разрешения,
- = Установка заданного разрешения

Значения параметра *разрешения*:

- r Право чтения,
- w Право записи,
- x Право выполнения,
- X Установка права выполнения только в том случае, если для какой-либо категории пользователей уже задано право выполнения,
- s Установка бита SUID или SGID для владельца или группы,
- t Установка sticky-бита,
- u Установка тех же прав, что и у владельца файла,
- g Установка тех же прав, что и у группы, являющейся владельцем файла,
- o Установка тех же прав, что и у других пользователей.

### 2.3.7. Обзор основных команд

Синтаксис многих команд предусматривает использование параметров довольно сложного формата, указываемых в командной строке после имени команды. Полное описание команд и их параметров можно получить, набрав

`man имя_команды`

Дополнительную информацию по команде :

`info имя_команды`

После ввода вышеперечисленных команд ОС Linux открывает на нескольких, сменяющих друг друга экранах описание нужной команды. Если не помните правильный синтаксис имени нужной команды, введите команду `man` с параметром `-k`, затем ключевое слово для поиска нужной команды. Система выполнит поиск в своих файлах справки, содержащей это ключевое слово. Для этой команды имеется также псевдоним `argporos`. Например, если ввести команду:

`man ls`



ОС Linux выведет на экран справку о команде `ls`, в том числе обо всех ее параметрах. По команде:

```
man -k ls
```

выводится из справки список всех команд, в которых есть слово `ls`.

Команда `apropos ls` аналогична команде `man -k ls`.

Ниже приведены наиболее употребительные команды и наиболее частые форматы.

#### а) команды управления файлами

**ls [opt] [file1 file2 ...]** - вывод имен файлов текущего каталога. В качестве параметров можно задать имена каталогов, содержимое которых нужно вывести, или имена файлов, информацию о которых нужно получить. Опции команды позволяют получить список дополнительной информации:

**ls** - список файлов текущего каталога (краткий формат).

**ls -al** - получить список файлов текущего каталога с указанием размера, времени создания и изменения, имени владельца, таблицы прав и других данных, например:

```
-rwxr--r-- 2 nata group 34 Nov 10 10:34 a.out
```

где `-rwxr--r--` - права доступа (за исключением первого символа, обозначающего тип файла: `-` — обычный файл, `d` — каталог, `p` — именованный канал, `b` — специальное блочное устройство, `c` — специальное символьное устройство) на чтение (read -символ `r`), запись (write -символ `w`), выполнение (execute - символ `x`). Наличие прав обозначается соответствующим символом, а отсутствие - символом `-`;

`2` — число жестких связей (hard link) данного файла;

`nata` - имя владельца -пользователя (user owner) файла. Владелец-пользователем вновь созданного файла является пользователь, запустивший процесс, который и создал файл;

`group` — имя владельца - группы (group owner). Порядок назначения владельца группы зависит от конкретной версии UNIX;

`34` - размер файла;

`Nov 10` - дата последнего изменения;

`10:34` - время последнего изменения;

`a.out` - имя файла.

**ls -aC** — просмотр скрытых файлов;

**ls -C nata** - вывод списка файлов каталога `nata` в несколько колонок в алфавитном порядке;

**ls -RC /home/nata/bin** - рекурсивный просмотр каталогов, например, `/home/nata/bin`;

**ls -tC** - сортировка по времени модификации, все вновь созданные файлы размещаются в начале списка;

**ls -ctC** - сортировка по изменению статуса (изменение владельца или прав доступа). Если ключ `t` не задан, то ключ `c` игнорируется.

**cd [dir]** - сменить текущий каталог. При задании без параметра – происходит переход в домашний каталог пользователя;

**cp файл1 файл2** - копировать файл. Если вместо имени второго файла указать каталог, то **файл1** копируется в каталог **файл2** с тем же именем, при этом в имени первого файла допускается использование подстановочного символа “звездочка”;

**rm файл1** – удалить файлы с указанными именами. Допускается использование подстановочного символа “звездочка” и другие специальные возможности. Например, команда *rm \*m\** позволит удалить все файлы, в именах которых встречается буква *m*;

**mkdir [имя\_каталога1]...** – создать новый каталог;

**rmdir [имя\_каталога]...** – удалить пустой каталог;

**ln [-опция] source target** - создает жесткую связь имени *source* с файлом, адресуемым именем *target*. При использовании опции *-s* будет создана символическая ссылка;

**pwd** - вывести имя текущего каталога;

**cmp [-опция] файл1 файл2** - сравнить два файла, указанных в качестве аргумента. Если файлы одинаковы, то никакого сообщения не выводится, в противном случае выводятся данные о первом несоответствии между этими файлами, например:

*file1 file2 differ: char 15, line 6* ,

найдено различие в 15 символе 6-й строки.

#### б) управление выводом на экран

**cat [-опция] файл** - выводит содержимое *файла* на экран терминала. Использование ключа *-v* целесообразно при просмотре нетекстового файла. В этом случае вывод “непечатных ” символов, которые могут нарушить настройки терминала, будет подавлен;

**more [-опция] файл** – выводит стандартный входной поток на экран порциями по 24 строки, ожидая нажатия клавиши *Пробел* для вывода очередной порции. Досрочно завершить вывод можно, нажав клавишу *Q*;

**less** - выводит стандартный входной поток на экран порциями по 24 строки, ожидая нажатия клавиши *Пробел* для вывода очередной порции. В отличие от команды *more* поддерживает возможность прокрутки вверх и поиска;

**head [-n] файл** – просмотреть только начало (первые *n* строк) файла;

**tail [-опция] файл** – просмотреть конец (последние *n* строк) файла;

#### в) поиск файлов

**find имя\_каталога [-ключ]** - выполнить поиск файла в файловой системе, начиная с каталога *имя\_каталога*, используя различные критерии:

- **name** – поиск по искомому имени файла, например:

*find / -name sh* ,

по этой команде будет осуществляться поиск в каталоге */* файла с именем *sh*;

**-print** – обеспечивает вывод информации. Например, для вывода полного имени исполняемого файла командного интерпретатора Bourne shell необходимо ввести команду:

```
find / -name sh -print 2 >/dev/null ;
```

Для фрагментарного поиска по имени файла (только в последней части спецификации файла), например, `*core*`, следует ввести команду:

```
find ~ -name '*core*' -print
```

**- size [размер]** – поиск по заданному размеру. Например, для поиска файлов размером больше 10 Мбайт по всей файловой системе необходимо ввести команду:

```
find . -size +20480 -print ;
```

**- atime** - поиск по последнему времени модификации. Например, поиск файлов с именем `file1`, обращение к которым было более 15 дней назад:

```
find / -name file1 -atime +15 -print ;
```

Для автоматического удаления всех найденных файлов с именем `core` (образ процесса, создаваемый при неудачном его завершении и используемый в целях отладки), последнее обращение к которым было более месяца (+30) назад, следует ввести команду:

```
find / -name core -atime +30 -exec rm {} \ ;
```

Следует отметить, что каждый раз при запуске команды, указанной после ключа `exec`, создается новый процесс. Это приводит к увеличению нагрузки на систему и излишнему потреблению ресурсов процессора и оперативной памяти. Однако при необходимости выполнить операцию, например такую как `rm`, над большим количеством файлов, эффективнее сначала построить список файлов, а затем запустить команду `rm` лишь один раз, передав ей этот список в качестве параметра.

Из приведенного выше видно, что при работе с командой `find` чаще всего используется опция `-name`. После нее в кавычках должен быть указан шаблон имени файла. Если необходимо найти все файлы с расширением `txt` в Вашем начальном каталоге, укажите символ `'.'` в качестве путевого имени. Имя начального каталога будет извлечено из переменной `$HOME`.

```
find ~ -name "*.txt" -print
```

Чтобы найти все файлы с расширением `txt`, находящиеся в текущем каталоге, следует воспользоваться такой командой:

```
find . -name "*.txt" -print
```

Для нахождения в текущем каталоге всех файлов, в именах которых встречается хотя бы один символ в верхнем регистре, введите следующую команду:

```
find . -name "[A-Z]*" -print
```

Команда `find . -print` аналогична команде `ls -Rfl`, но в последнем случае выводимый список будет длиннее, т.к. в процессе обхода команда `ls` отмечает каждый новый каталог, а команда `find` не обращает внимание на каталог `..` ;

**which [-ключ]** - поиск выполняемых файлов. Данная команда встроена в оболочку, позволяет определить точное местонахождение файла и передает результаты своего выполнения в стандартный выходной поток. В оболочке C команда *which* позволяет определить, какие из команд являются встроенными, а какие псевдонимами.

#### г) исследование и мониторинг системы

Для управления дисковым пространством используются команды *df*, *du* и *ulimit*:

**df [-ключ]** – команда определяет, сколько свободного дискового пространства и индексных дескрипторов доступно в разделе смонтированного диска.

По умолчанию команда используется без параметров и выводит объем свободного пространства, например:

```
/          (/dev/hdb1  ): 260836 blocks  12034 files
/home      (/dev/sda1  ): 260836 blocks  2104 files
```

В первом столбце содержится точка монтирования данной файловой системы. Затем в круглых скобках следует имя смонтированного физического устройства (в UNIX все устройства являются файлами, даже сама файловая система). Следующий столбец отображает число свободных блоков размером по 512 байт. В последнем столбце выводится количество файлов, содержащихся на данном устройстве.

При использовании ключей:

**-k** – вывод данных осуществляется в блоках по 1024 байт, или в килобайтах. При этом данные выводятся в формате, принятом в системе BSD:

Filesystem	1024-	blocks	Used	Available	Capacity
Mounted on					
/dev/hdb1	1112646	972611	140035	88%	/
/dev/sda1	961374	720104	241270	75%	/home

В первом столбце указано имя устройства, на котором расположена файловая система. Во втором столбце отображается размер файловой системы в блоках по 1 Кбайт. В третьем столбце выводится число используемых блоков, а в четвертом – число свободных блоков. В пятом столбце выводится процент использования диска. В последнем столбце указывается точка монтирования системы;

**-P** – информация отображается в формате, определенном в стандарте POSIX, который аналогичен формату, принятому в BSD;

**-t** - информация отображается в формате, который близок к стилю, используемому в SYSTEM V. Данные выводятся в блоках размером по 512 байт, кроме того, приводится информация как о количестве блоков, так и о количестве индексных дескрипторов;

**-i** - предназначен для подсчета количества индексных дескрипторов (не поддерживается стандартом POSIX). Выводимая информация имеет следующий вид:

Filesystem	Inodes	IUsed	IFree	%IUsed	Mounted on
/dev/hdb1	301056	93059	207997	31%	/
/dev/sda1	260096	17280	242816	7%	/home

В качестве параметров команде *df* можно передать имя файла или список имен файлов. В этом случае отображается информация только о тех файловых системах, которые содержат указанные файлы.

**du [- ключ]** - команда определяет, какой объем диска занимает конкретный каталог. Вызов команды без параметров позволяет получить данные о текущем каталоге. Если в качестве параметра указать имя каталога, то будет отображена информация обо всех каталогах, расположенных в иерархии ниже текущего. Если в качестве параметра указано имя файла, не являющееся каталогом, то не выводится никакой информации.

Команда *du* имеет четыре ключа:

**-k** – имеет то же значение, что и для команды *df*, при этом данные об использовании дискового пространства представляются в килобайтах;

**-a** – задает вывод данных всех перечисленных файлов. При этом полученный результат аналогичен результатам выполнения команды *ls -ls*;

**-s** – задает ограниченный вывод, только данные об указанном каталоге, например: *13500 /home/nata/bin*, где 13500 – размер каталога, выраженный в блоках по 512 байт;

**-x** – не выводятся данные о файлах, находящихся в других файловых системах. Таким образом проверяются данные, хранящиеся в указанном каталоге локального диска;

**ulimit** – выводит или устанавливает значение пределов, ограничивающих использование задач системных ресурсов (времени процессора, памяти, дискового пространства);

**top** – команда выдает непрерывно обновляемую таблицу всех задач, выполняющихся на компьютере, включая системные, с указанием объема используемых ресурсов. Для завершения работы команды необходимо нажать клавишу *Q*;

**ps** – выводит информацию о существующих процессах. При использовании различных опций можно получить следующую информацию:

**-al** - выдает в форме таблицы список пользовательских процессов, запущенных в системе;

**-F** – статус процесса (системный, блокировки памяти и т.д.);

**-A** – состояние всех процессов;

**-S** – состояние процесса (О – выполняется процессором, S – находится в состоянии сна, R - готов к выполнению, I - создается, Z - зомби);

**-ef** – распечатывает имя программы, породившей процесс, вместе со всеми параметрами;

- **n name** – состояние всех процессов, порожденных командами, имена которых указаны в списке *name*;

- **g list** – показать все процессы, запущенные пользователями групп, номера которых указаны в списке. Например, *ps -g 0* -показать все процессы группы 0, т.е. root. Номера групп указываются в списке через запятую или пробел;

- **l** – длинный формат вывода состояния процессов;

- **p** - состояние процессов, идентификаторы которых указаны в списке, например: *ps -p "12499, 17772"* – определить состояние процессов с идентификаторами (PID) 12499 и 17772;

**w [- ключ]** – команда информирует о том, что делают в системе зарегистрированные пользователи, например:

```
(9:12 am up 30 min, 3 users, Load average, 0.00, 0.52, 1.22)
user      TTY      FROM    LOGIN@   IDLE   JCPU   PCPU       what
user      tty1    -       8.44 am  27:50   0.24s   0.03s   /bin/sh/usr
userpts   /0      -       8.52 am  29:48   0.00s   ?       -
```

Первая строка содержит текущее время, сколько времени компьютер работает без перезагрузки, число пользователей и загрузка машины. Затем следует строка, содержащая заголовки столбцов: *user* – имя пользователя, связанного с данным устройством *tty*; *TTY* – имя терминала (консоли); *LOGIN@* -первоначальное время регистрации; *IDLE* - количество времени, на протяжении которого пользователь ничего не вводил с клавиатуры; *ICPU* – общее время центрального процессора, использованного всеми процессами на этом терминале; *PCPU* – общее время центрального процессора для всех активных процессов на этом терминале; *what* – название и параметры текущей выполняемой команды. Далее следует список пользователей, и чем они заняты. Знак ? означает, что процесс ожидает связи с терминалом, однако в текущий момент связь отсутствует. Команда имеет три ключа:

- **h** – подавляет заголовки;

- **l** – отображает информацию в расширенном виде (используется по умолчанию);

- **s** – отображает информацию в краткой форме (выводятся столбцы *user*, *tty*, *idle*, *what*);

Конкретного пользователя можно проверить, введя команду

*w имя\_пользователя*

**who [-ключ]**– выдается список пользователей, зарегистрированных в данный момент в системе. Например:

```
nata      tty1      Nov 2      14:30
alex      tty4      Nov 2      14:15
```

где - *nata* – имя пользователя,

*tty1* - номера его терминала,

*Nov 2* - дата,

14:30 - время подключения.

Согласно стандарту POSIX, команда должна иметь несколько ключей, влияющих на внешний вид выводимой информации:

- b – выводит время последней перезагрузки;
- d – выводит список “умерших” процессов (dead processes), которые не были повторно порождены;
- H – выводит заголовки столбцов;
- l – перечисляет номера tty, ожидающих регистрации пользователей;
- T – выводит состояние канала связи с каждым из терминалов (+ означает, что данный терминал доступен для записи, а – означает, что терминал для записи не доступен);
- t – выводит момент последнего изменения системного времени;
- s – выводит имя пользователя, tty и время регистрации в системе (используется по умолчанию);
- u – выводит время простоя для каждого терминала;
- m – выводит информацию только о текущем терминале;
- r – выводит текущее состояние системы;
- p – перечисляет все активные процессы, порожденные процессом *init*;
- g – перечисляет только пользовательские имена и количество пользователей;

Пример результата выполнения команды *who -THu* :

<i>USER</i>	<i>MESG</i>	<i>LINE</i>	<i>LOGIN-TIME</i>	<i>IDLE</i>
<i>nata</i>	+	<i>tty1</i>	<i>nov 10 18:44</i>	.
<i>oleg</i>	-	<i>tty3</i>	<i>nov 10 19:53</i>	<i>old</i>
<i>alex</i>	+	<i>tty4</i>	<i>nov 10 18:53</i>	<i>old</i>

Из примера видно, что только пользователь *nata* находится в активном состоянии. Пользователи *oleg* и *alex* не обращались к своим терминалам на протяжении дня. Кроме того, пользователю *oleg* доступ к терминалу запрещен; **last [-ключ]** – позволяет определить, кто и когда зарегистрировался в системе. Для выдачи результатов она пользуется файлом */etc/utmp*, в котором зафиксированы моменты входа-выхода пользователей и перезагрузки системы. При использовании команды без параметров будет выведен список в обратном порядке всех, кто работал в системе.

Для ограничения размера списка в качестве параметра следует указать некоторое число, например,

*last -25*

выводит список последних 25 пользователей. Введя команду *last reboot*, можно просмотреть список последних перезагрузок;

**finger** – команда позволяет определить, находится ли в системе некоторый пользователь. Введя команду

*finger – имя\_пользователя*

можно получить разнообразную информацию, включающую и время последней регистрации данного пользователя в системе;

**at [-ключ] время\_запуска** - считывает команды стандартного потока ввода и группирует их в задания *at*, которые будут выполнены в указанное пользователем время. . Например:

*at now + 2minutes*

Для выполнения задания будет запущен командный интерпретатор, в среде которого и будут исполнены команды.

**uptime** – позволяет оценить стабильность и загрузку системы. Данная команда выводит только первую строку информации команды *w*, например,

*9:12 pm up 10 days, 10:51, 4 users, load average: 0.01, 0.03, 0.22)*

### 2.3.8. Менеджер файлов **Midnight Commander (mc)**

Программа **Midnight Commander** полифункциональный менеджер файлов, работающий в текстовом режиме (т.е. в текстовой консоли или терминале). Интерфейс программы похож на двухпанельные менеджеры файлов **Norton Commander** для **MS-DOS**, **FAR** и **Windows Commander** для **Windows**, а по набору функций не уступает лучшим из них. Файловые операции *mc* выполняются аналогично.

### 2.3.9. Простейшие текстовые редакторы

Для работы в текстовой консоли **RedHat** можно воспользоваться несколькими простейшими текстовыми редакторами, которые позволяют изменить конфигурационный файл системы или набрать текст сценария. В текстовом режиме, как и в оболочке **KDE**, можно использовать профессиональную систему подготовки текста *emacs* (включающую в качестве макроязыка язык программирования высокого уровня), однако ее рассмотрение выходит за пределы данного пособия.

В простых случаях можно воспользоваться встроенным редактором программы *Midnight Commander (mc)*. Для того чтобы отредактировать текстовый файл во встроенном редакторе *mc*, выберите нужный файл в активной панели и нажмите клавишу **F4**. В открывшемся окне редактора можно вводить или редактировать текст. При необходимости следует использовать кнопки операций с блоками текста или поиска по образцу или, нажав клавишу **F9**, открыть меню, позволяющее устанавливать пользовательские настройки редактора, или осуществлять такие операции, как форматирование текста и обработка при помощи макросов.

Существующий несколько десятков лет текстовый редактор *vi* имеет очень специфическую систему команд и сохраняется в современных системах **UNIX (Linux)** во многом лишь по традиции. Однако некоторые старые командные файлы (скрипты) могут по умолчанию вызывать данный редактор для редактирования файлов пользователя. В этом случае понадобится выйти из текстового редактора *vi* без сохранения изменений: поместить курсор с помощью клавиши *Backspace* в ту часть окна, где расположен текст; далее



набрать символ `:` (нажав клавиши *Shift* - `:`), курсор вместе с набранным символом переместится в нижнюю строку экрана (поле команд); ввести в этом поле последовательность символов *q!* и нажать клавишу *Enter*.

#### **2.4. Вопросы к защите лабораторной работы**

- 1) Перечислите этапы монтирования системы при загрузке ОС.
- 2) Перечислите особенности работы в текстовой и графической консолях.
- 3) Что понимается под монтированием файловой системы?
- 4) Как осуществляется монтирование устройств?
- 5) Приведите формат команды монтирования/размонтирования устройств.
- 6) Перечислите пользователей системы.
- 7) Как добавить пользователя, группу пользователей в систему?
- 8) Какая команда используется для изменения владельца текущего сеанса?
- 9) Какую информацию содержат файлы `/etc/shell`, `/etc/passwd`?
- 10) Где и в каком поле записи указывается командный интерпретатор пользователя?
- 11) Как можно изменить режим доступа к файлам?
- 12) Поясните поля записи из файла `/etc/passwd`.
- 13) Перечислите файлы, относящиеся к служебным учетным записям.
- 14) Укажите формат команд, используемых для исследования системы.
- 15) Перечислите команды для идентификации файлов.
- 16) Укажите формат команды для получения информации о процессах, связанных с терминалом.
- 17) Поясните синтаксис команды поиска файлов с использованием различных комбинаций ключей.

#### **2.5. Список рекомендуемой литературы**

- 1) Робачевский А. Операционная система UNIX. - СПб.: BHV-Санкт-Петербург, 2007. - 528 с.
- 2) Команды Linux. Справочник. - К.: ТИД ДС, 2002. - 688 с.
- 3) Романчева Н.И. Системное программное обеспечение: учебное пособие. – М.: МГТУ ГА, 2005. – Ч. 1.

### **3. ЛАБОРАТОРНАЯ РАБОТА №2**

#### **ОС LINUX: ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ КОМАНДНОГО ИНТЕРПРЕТАТОРА**

##### **3.1. Цель лабораторной работы**

Целью данной работы является формирование компетенций: ОК1, ОК11, ОК12, ПК2, ПК5, в том числе получение практических навыков и умений программирования на языке командного интерпретатора, получение навыков отладки сценариев.

### 3.2. Задание на выполнение лабораторной работы

- 1) Написать и отладить программу на языке интерпретатора Borne:
  - используя команду `echo`;
  - используя команду управления форматированием вывода (`printf`).
- 2) Написать и отладить программу, обрабатывающую ввод из потока `STDIN`.
- 3) Написать и отладить программу, считывающую данные из потока и использующую несколько переменных в аргументе команды `read`. При запуске сценария задать число переменных в списке меньше (больше) числа аргументов, считанных из потока.
- 4) Написать сценарий для выполнения проверки значений введенных переменных.
- 5) Написать и отладить программу вычисления длины окружности и площади круга.
- 6) Создать файл функций *functions.main*, включающий одну функцию, которая будет загружена интерпретатором команд, протестирована, изменена, а затем повторно загружена.

### 3.3. Основные приемы работы

#### 3.3.1. Программирование на языке командного интерпретатора

Интерпретатор (shell) представляет собой интерфейс командной строки. По своей функциональности он похож на интерпретатор `COMMAND.COM` системы `MS DOS`. Одной из задач интерпретатора является обеспечение безопасного и структурированного доступа к ядру UNIX-подобных операционных систем, т.е. фактически это программный уровень, который предоставляет среду для ввода команд, обеспечивая тем самым взаимодействие между пользователем и ядром операционной системы. Кроме того, встроенный в интерпретатор мощный язык программирования используется для решения различных задач: от автоматизации повторяющихся команд до написания сложных интерактивных программ обработки данных, получения информации из небольших баз данных.

В среде UNIX существует много командных интерпретаторов, среди которых можно выделить такие как `sh`, `tcsh`, `ksh`, `csch`, `bash`, в Linux по умолчанию используется `bash`.

#### 3.3.2. Встроенные команды интерпретатора shell

<code>:</code>	Нуль, всегда возвращает истинное значение.
<code>.</code>	Считывание файлов из текущего интерпретатора shell
<code>break</code>	Применяется в конструкциях <code>for</code> , <code>while</code> , <code>until</code> , <code>case</code> .
<code>cd</code>	Изменяет текущий каталог.
<code>continue</code>	Продолжает цикл, начиная следующую итерацию.
<code>echo</code>	Записывает вывод в стандартный поток вывода.
<code>eval</code>	Считывает аргумент и выполняет результирующую команду.
<code>exit</code>	Выход из интерпретатора shell.
<code>export</code>	Экспортирует переменные, вследствие чего они доступны для

	текущего интерпретатора shell.
<i>pwd</i>	Отображает текущий каталог.
<i>read</i>	Просматривает строку текста из стандартного потока.
<i>readonly</i>	Превращает данную переменную в переменную "только для чтения".
<i>return</i>	Выход из функции с отображением кода возврата.
<i>set</i>	Управляет отображением различных параметров для стандартного потока вводных данных.
<i>shift</i>	Смещает влево командную строку аргументов.
<i>test</i>	Оценивает условное выражение.
<i>times</i>	Отображает имя пользователя и системные промежутки времени для процессов, которые выполняются с помощью интерпретатора shell.
<i>trap</i>	При получении сигнала выполняет определенную команду.
<i>type</i>	Интерпретирует, каким образом интерпретатор shell применяет имя в качестве команды.
<i>ulimit</i>	Отображает или устанавливает ресурсы интерпретатора shell.
<i>umask</i>	Отображает или устанавливает режимы создания файлов, заданные по умолчанию.
<i>unset</i>	Удаляет из памяти интерпретатора shell переменную или функцию.
<i>wait</i>	Ожидает окончания дочернего процесса и сообщает о его завершении.

### 3.3.3. Примеры программ

Ниже приводятся примеры нескольких способов написания простой программы вывода фразы "Hello, Linux!" на языке интерпретатора Bourne.

Пример 1.

```
#!/bin/sh
# Первая программа "Hello, Linux!",
# реализованная для интерпретатора Bourne
echo
echo " Hello, Linux! "
echo
exit 0
```

Пример 2.

```
#!/bin/sh
# Программа, "с командой printf",
# реализованная для интерпретатора Bourne
printf "\n" с командой printf\n\n!"
exit 0
```

Пример 3.

```
#!/bin/sh
# Программа, "воспринимающая ввод с клавиатуры",
```

```

#реализованная для интерпретатора Bourne
echo
echo -n "Введите name:  "
read name
echo
echo "Hello,  ${name}!"
echo
exit 0

```

В примерах 1-3 первые строки содержат последовательность символов `#!`, сообщающую ОС, что за ней следует имя командного интерпретатора, в котором сценарий должен выполняться. В данном случае это интерпретатор Bourne, `/bin/sh`. Строки, начинающиеся с символа `#`, являются комментариями, и интерпретатор их игнорирует. Команда `echo` выводит свои аргументы в стандартный вывод (обычно это экран). Команда `exit` завершает работу программы и возвращает код выхода родительской программе (обычно это командный интерпретатор). Код завершения, равный 0, указывает, что программа нормально завершила работу. Код, отличный от 0, сообщает об ошибке. Если статус завершения не указан явно, то программа возвращает код завершения последней выполненной команды.

Простого вывода сообщений на экран недостаточно для решения задач, поэтому язык сценариев использует переменные. Все переменные в программах командного интерпретатора хранятся как строки. Кроме присвоения значений переменным и их использования в сценарии, командный интерпретатор предоставляет возможность обрабатывать ввод из потока STDIN. Для чтения пользовательского ввода используется команда `read`. Команда `read` в качестве аргументов может иметь несколько переменных (пример 4). В этом случае всякий пустой символ трактуется как символ разделитель между значениями, присваиваемыми разным переменным. Если число аргументов, прочитанных из потока ввода, меньше, чем число переменных в списке, оставшимся переменным не присваивается никаких значений. Если аргументов больше, чем переменных, то все оставшиеся значения присваиваются последней переменной.

Пример 4.

```

#!/bin/sh
echo
echo -n "Введите три числа, разделенные пробелами или символом
табуляции: "
read var1 var2 var3
echo
echo "The of var1 is:  ${var1}"
echo "The of var2 is:  ${var2}"
echo "The of var3 is:  ${var3}"

```

```
echo
exit 0
```

Для операций с числами с плавающей запятой, а также обработки сложных выражений, где порядок операций изменен, применяется команда *bc* (пример 5).

Пример 5.

```
#!/bin/sh
# Данная программа вычисляет длину окружности
# и площади круга
pi="3.14159265"      # Переменной присваивается число pi
# пользователю выводится сообщение и запрашивается радиус
# окружности
echo
echo -n "Введите радиус: "
read radius
# Вычисляемые значения сохраняются в переменных
cir=$(echo $radius*2*$pi | bc )
area=$(echo $radius^2*$pi | bc )
# Программа выводит результаты и завершает работу
printf "\n\nThe circumference is:\t$cir\n"
printf "The area is:\t\t$area\n\n"
exit 0
```

Сценарий не относится к компилируемым программам, поскольку он интерпретируется построчно. После ввода текста программы и сохранения необходимо сделать файл выполняемым. Для этого используется команда:

```
chmod u+x text (сценария).
```

Эта команда устанавливает права владельца на запуск файла *text*. Для запуска файла необходимо ввести в командной строке: *./text*

Пример 6.

```
#!/bin/sh
echo -n "What is your first name : "
read F_NAME
echo -n "What is your surname : "
read S_NAME
printf "\n\n What is your first name :\t$ F_NAME \n"
exit 0
```

Данный сценарий предназначен для выполнения проверки значений переменных (переменным присваиваются исключительно одни символы).

Пример 7.

```
#!/bin/sh
```

```

# functions.main
# findit: интерфейс для базовой команды find
findit ()
# findit
if [ $# -lt 1 ]; then
echo "usage :findit file"
return 1
fi
find / -name $1 -print
exit 0

```

В данном примере создается файл функций *functions.main*, включающий одну функцию. Эта функция будет загружена интерпретатором команд, протестирована, изменена, а затем повторно загружена. Данный код лежит в основе интерфейса для базовой команды *find*. Если команде не передаются аргументы, то возвращается значение 1 (что свидетельствует о возникновении ошибки). Обратите внимание, что ошибочная конструкция фактически является отображенным именем функции (если же была использована команда *\$0*, интерпретатор команд просто возвращает сообщение *sh*). Причина отображения подобного сообщения заключается в том, что файл не является файлом сценария.

### 3.4. Вопросы к защите лабораторной работы

- 1) Что понимается под термином “пользовательская среда UNIX”?
- 2) Поясните общую структуру скрипта.
- 3) В каком виде хранятся переменные в программах командного интерпретатора?
- 4) Приведите примеры сложных синтаксических конструкций получения значений переменной.
- 5) Перечислите внутренние переменные shell, используемые в скриптах.
- 6) В каких случаях используется команда *bc* ?
- 7) Поясните процедуру создания файла функций.

### 3.5. Список рекомендуемой литературы

- 1) Робачевский А. Операционная система UNIX. - СПб.: BHV-Санкт-Петербург, 2007. - 528 с.
- 2) Романчева Н.И. Системное программное обеспечение: учебное пособие. – М.: МГТУ ГА, 2005. – Ч. 1.