ГОСУДАРСТВЕННЫЙ КОМИТЕТ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПО ВЫСШЕМУ ОБРАЗОВАНИЮ
УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
КАФЕДРА АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ

РАБОТА С ОПЕРАЦИОННОЙ СИСТЕМОЙ UNIX ТИПА SunOS

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторному практикуму по курсу "Операционные системы" для студентов направления Т28 "Информатика и вычислительная техника"

УФА 1995

Составители: М.Я.Парфенова, В.Ю.Арьков

УДК 681.3

Работа с операционной системой UNIX типа SunOS: Методические указания к лабораторному практикуму по курсу "Операционные системы" для студентов направления Т28 "Информатика и вычислительная техника"/Уфимск. гос. авиац. техн. ун-т; Сост. М.Я.Парфенова, В.Ю.Арьков. Уфа, 1995.- 41 с.

Содержатся основные сведения, необходимые для работы с операционной системой UNIX в текстовом режиме и ее сетевыми приложениями. Рассматривается структура файловой системы, многозадачный режим выполнения процессов и возможности обработки данных в сети с использованием приложения PC - NFS (Network File System). Практическое изучение функциональных особенностей системы иллюстрируется примерами формирования простых и сложных команд по обмену данными между удаленными терминалами. Обсуждается методика и порядок выполнения лабораторных работ, в приложении приводятся вспомогательные материалы.

Библиогр.: 6 назв.

Рецензенты: Л.Р.Черняховская, Р.В.Насыров

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Операционная система UNIX - это многозадачная система (ОС),под управлением которой одновременно могут выполняться несколько задач. Она предназначена для работы на станциях, имеющих множество подключенных терминалов. Для обслуживания терминалов в мультипрограммном режиме требуется большой объем оперативной и внешней памяти (8 Мбайт ОЗУ, 1 - 100 Гбайт ПЗУ), достаточное быстродействие (50 МГц и более). UNIX- серверы предназначены для размещения мощных программных продуктов, хранения и обработки больших объемов информации [1]. В качестве терминалов могут быть различные периферийные устройства: персональные компьютеры (ПК),модемы,факсы. Программы для выполнения могут запускаться с терминала и использовать все доступные в данный момент ресурсы рабочей станции. Поэтому нет необходимости доводить, например,каждый ПК до большой мощности. Особенно эффективно использование UNIX - серверов при распределенной обработке данных.Для этого разработаны системы управления базами данных (СУБД) типа Oracle, Informix,работающие под управлением ОС UNIX. Т.е. информация может находиться на различных рабочих станциях, различных дисках, но система работает таким образом,что данные составляют одно целое - единую БД. При обработке больших объемов информации особым преимуществом является наличие интерфейса клиент-сервер, когда пользователь работает только с той информацией, которая его интересует, а не со всем объемом.

ОС UNIX является альтернативой вычислительной сети. Скорость передачи данных на порядок выше, чем в ВС, но эффективность достигается при подключении не менее 8 терминалов.

ОС UNIX реализована на языке СИ. Имеются различные версии и типы UNIX в зависимости от фирмы разработчика: Hewlett Packard, IBM, SUN, INTEL и др.

Подключение персональных компьютеров (ПК) в локальную сеть с UNIX - серверами может осуществляться по протоколу TCP/IP с помощью пакета PC-NFS. При этом пользователи получают следующие

возможности:

1) использования Sun как файл - сервера;
2) эмуляции на ПК терминала Sun (режим TELNET);
3) организации системы клиент - сервер (ПК формирует SQL - запросы, сервер Sun их обрабатывает);
4) непосредственного обмена файлами между ПК по протоколу FTP.

ЛАБОРАТОРНАЯ РАБОТА N 1
ИЗУЧЕНИЕ ПРИНЦИПОВ ФУНКЦИОНИРОВАНИЯ ОПЕРАЦИОННОЙ СИСТЕМЫ UNIX ТИПА SunOS

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение принципов функционирования многозадачной операционной системы Unix типа SunOS в режиме эмуляции терминала Sun на ПК и использования Sun как файл-сервера.

2. ОБЩИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Процесс

Все действия в ОС UNIX оформлены как процессы. Процесс представляет собой совокупность выполняемых программ или одну выполняемую программу, которые вызываются при исполнении системной команды. Процесс может породить один или несколько других процессов, которые могут выполняться параллельно [2]. Кроме того, программу можно построить из одновременно (параллельно) выполняющихся процессов, что значительно сокращает время решения задачи. Для параллельного выполнения различных частей программы можно использовать многопроцессорное оборудование. Положенные в основу ОС UNIX механизмы поддерживают многопроцессорность. При наличии одного процессора процессы выполняются в режиме разделения времени.

Система включает следующие основные компоненты:

ядро - управляющую программу, которая выполняет функции уп-

равления памятью, процессором, исполнением всех программ, а также обслуживает внешние устройства;

интерпретатор команд shell, он анализирует команды, вводимые с терминала либо из командного файла, и передает для выполнения их в ядро системы. shell является также языком программирования, на котором можно написать командные файлы (shell-файлы). При входе в ОС пользователь получает копию интерпретатора shell в качестве родительского процесса. Далее, после ввода команды пользователем создается порожденный процесс, называемый процессом-потомком. Т.е. после запуска ОС каждый новый процесс функционирует только как процесс-потомок уже существующего процесса. В ОС Unix имеется возможность динамического порождения и управления процессами.

2.2. Загрузка системы

После включения компьютера на экране появляется сообщение о начальной загрузке:

boot:

В ответ необходимо ввести имя устройства (жесткий диск, флоппи-диск, стримерная лента и др.), с которого загружается ядро операционной системы в оперативную память, и имя файла. Сразу после загрузки ядра порождаются процессы с идентификаторами 0 и 1. Процесс 1 - это вызов системной программы init, главного диспетчера процессов. init, в свою очередь, инициализирует интерпретатор команд shell и передает ему на выполнение файл настройки rc.

При входе в систему для каждого терминала, с которого возможен вход в ОС, порождается процесс getty, при этом задаются характеристики и режимы работы терминала. getty запускает процесс login, который выдает на экран сообщение "login:" и система переходит в состояние ожидания. Схема вызовов системных программ и последовательность порождения процессов при загрузке ОС UNIX представлена на рис. 2.1.

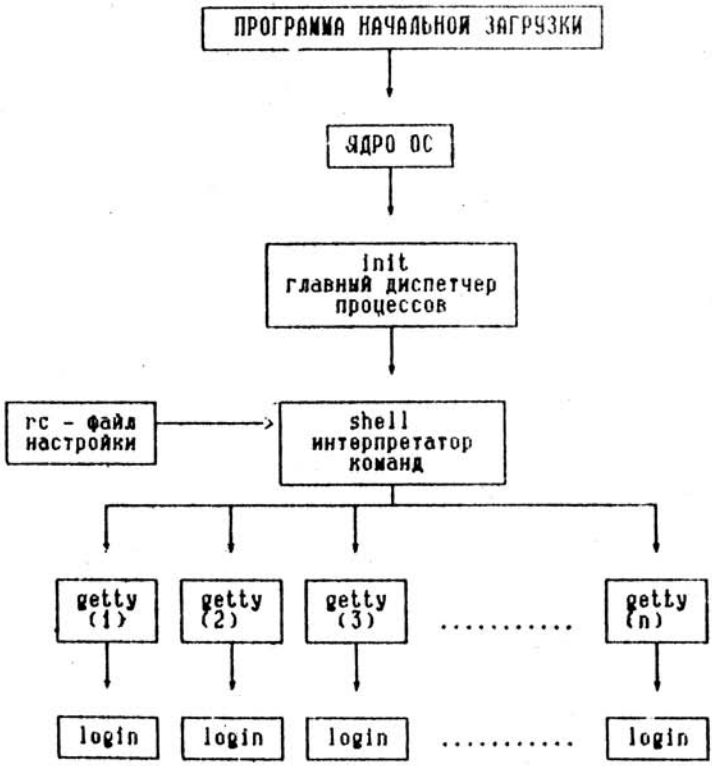
ПРОГРАММА НАЧАЛЬНОЙ ЗАГРУЗКИ
ЯДРО ОС
init
главный диспетчер
процессов
rc - файл
настройки
shell
интерпретатор
команд
getty
(1)
getty
(2)
getty
(3)
.....
getty
(n)
login
login
login
.....
login

Рис. 2.1

2.3. Вход пользователя в систему

Процедуры входа пользователя могут отличаться в разных версиях ОС UNIX. Если есть запрос на ввод идентификатора пользователя

login:

нужно ввести свой идентификатор и нажать <Enter>. В противном случае набрать команду эмуляции терминала для работы в текстовом режиме

telnet <имя Unix - сервера>

и затем ответить на запрос login. После этого появляется запрос на ввод пароля:

password:

Значение пароля проверяется в системном файле password, где приводятся и другие сведения о пользователях. После правильного ответа (он не отображается на экране) пользователь имеет доступ к ресурсам системы. В этот момент выполняются системные файлы /etc/profile и .profile и пользователь начинает работать со своим собственным shell.

2.4. Выход из системы

Окончание сеанса пользователя

exit

Возвращение в состояние login

Ctrl - D

Выход (в зависимости от версии ОС)

logout или login

Alt - X

Выключение системы только из корневого каталога

/etc/shutdown

2.5. Работа с процессами

2.5.1. Команда для вывода информации о процессах

ps [-aklx] [number]

a - вывод информации обо всех активных процессах;

k - анализ файла причины аварийного окончания процесса;

l - полная информация о процессах;

x - печать информации о процессах;

number - номер процесса.

Пример:

ps -al

Команда ps без параметров выводит информацию только об активных процессах,запущенных с данного терминала,в том числе и фоновых. На экран выводится подробная информация о всех активных процессах в следующей форме:

```
F S UID PID PPID C   PRI NI   ADDR SZ WCHAN TTY TIME COMD
1 S 200 210 7    0   28  20   80   30 703a  03  0:07 cc
1 R 12  419 7    117 51  20   56   20       03  0:12 p
```

F - флаг процесса (1 - в оперативной памяти, 2 - сис.емный процесс,4 - заблокирован в ОЗУ,20 - находится под управлением другого процесса, 10 - подвергнут свопингу);

S - состояние процесса (S - задержан, R - выполняется; W - ожидает завершения другого процесса);

UID - идентификатор пользователя;

PID - идентификатор процесса;

PPID - номер родительского процесса;

C - степень загруженности процессора;

PRI - приоритет процесса, вычисляется по значению переменной NICE,и чем больше число, тем меньше его приоритет;

NI - переменная NICE для вычисления приоритета, принимает значения от 1 до 19;

ADDR - адрес процесса в памяти;

SZ - объем ОЗУ, занимаемый процессом;

WCHAN - имя события, до которого процесс задержан, для активного процесса - пробел;

TTY - номер терминала (CO - консоль);
TIME - время выполнения процесса;
COMD - команда, которая породила процесс.

2.5.2. Команда изменения приоритета

nice [аргумент] command

Пример. Увеличение приоритета процесса на 2

nice - 2

При запуске процессов с одного терминала в фоновом режиме команда должна содержать в конце метасимвол &. Например, вызов компилятора языка СИ для обработки программы primer в фоновом режиме:

cc primer.c &

При активизации фонового процесса ядро системы сообщает назначенный ему номер.

2.5.3. Прекращение процесса до завершения

kill [- Sig] N_процесса

Sig - номер сигнала.

Sig = -15 означает программное (нормальное) завершение процесса, номер сигнала = -9 - уничтожение процесса. По умолчанию Sig= -9.

Вывести себя из системы можно командой kill -9 0 . Пользователь с низким приоритетом может прервать процессы, связанные только с его терминалом.

2.6. Простые команды

Получение информации о работающих пользователях

who

Ответ представляется в виде таблицы, которая содержит следующую информацию:

идентификатор пользователя;
идентификатор терминала;
дата подключения;

время подключения.

Сведения о команде (мануал по системе)

man <название_команды> [more] [q]

more - постраничное листание клавишей ENTER;

q - выход.

Исправление символов в конце командной строки

Ctrl - H

Изменение пароля

passwd

На запрос системы необходимо сообщить старый пароль, затем ввести новый.

Сообщение имени специального файла стандартного вывода, соответствующего терминалу пользователя

tty -S

Вывод содержимого файла на экран

cat <имя файла>

Вывод содержимого каталога на экран

ls [маршрут и имя каталога]

Если аргумент не указан, выдается содержимое текущего каталога.

Удаление файла

rm <имя файла>

Печать файла

lpr <имя файла>

Подсчет количества работающих пользователей командой wc (word count - счет слов)

who | wc

Содержимое файла text_1 пересылается в файл text_2 командой

cat text_1>text_2

Электронная почта передает файл всем пользователям, перечисленным в командной строке

mail asu < file.txt

Удаление файлов text.1, text.2, text.3

rm text.?

или

```
  rm text.[123]
  rm text.[1-3]
```
Просматриваются файлы text.1 и text.2
```
  cat  text.1,text.2
```
Добавление файла text.1 в конец файла text.2
```
  cat text.1 >> text.2
```
Просмотр всех команд, введенных с консоли пользователем
```
  history
```
Чистка экрана
```
  cls
```

2.7. Формирование сложных команд

Сложная команда имеет следующий формат:

имя команды [флаги] [параметры] [метасимволы]

Имя команды может содержать от 2 до 9 букв и цифр;

флаги - это одна или несколько букв со знаком минус(-);

параметры - передаваемые значения для обработки;

метасимволы интерпретируются как специальные операции:

& - процесс выполняется в фоновом режиме, не дожидаясь окончания предыдущих процессов;

* - шаблон, распространяется на все оставшиеся символы;

| - программный канал : стандартный вывод одного процесса является стандартным вводом другого;

> - переадресация вывода в файл;

< - переадресация ввода из файла;

? - шаблон, распространяется только на один символ ;

; - если в списке команд команды отделяются друг от друга точкой с запятой, то они выполняются друг за другом;

&& - эта конструкция между командами означает, что последующая команда выполняется только при нормальном завершении предыдущей команды (код возврата 0);

|| - последующая команда выполняется, только если не выполнилась предыдущая команда (код возврата 1);

( ) - группирование команд в скобки;

( ) - группирование команд с объединенным выводом;

[ ] - указание диапазона или явное перечисление (без запятых);

>> - добавление содержимого файла в конец другого файла.

Примеры:

трансляция СИ - программы в фоновом режиме

```
cc primer.c &
```

удаление всех файлов с именем text

```
rm text.*
```

просмотр файлов text.1 и text.2 и вывод их на печать

```
(cat text.1; cat text.2) | lpr
```

Резидентная программа для приема сообщений на ПК в реальном времени (запускается из системы DOS)

```
listener  -b
```

listener принимает сообщение, при этом выдается звуковой сигнал и на экране появляется окно с текстом сообщения. Кр ме посылаемого сообщения, указывается дополнительная информация:

```
FROM: @ 129.200.09.01        IP - адрес отправителя
FROM_TERM: /dev/ pts/1       порядковый номер пользователя
TO: ALL                      кому предназначено сообщение
MESSAGE:  "Текст сообщения ".
```

Для выхода из окна сообщения нужно нажать клавишу ALT. Таблица IP - адресов с именами машин хранится в файле C:\NFS\hosts. На сервере такой же файл находится по маршруту /etc/hosts . Его можно просмотреть командой

```
cat  /etc/hosts
```

Отправка сообщения на другой терминал (выполняется команда из DOS)

```
send < кому > < что >
```

Примеры:

посылка сообщения всем пользователям с именем user на все доступные машины

```
send   user    " текст сообщения "
```

это сообщение всем, кто работает в сети

```
send  @  "текст сообщения"
```

посылка сообщения на машину с именем host

```
send @host  " текст сообщения "
```

сообщение на сервер mercury

```
send @mercury  " HELLO "
```

Посылка сообщений в ОС UNIX (TELNET)

```
msgclnt < кому > < что >
```

это сокращенное название " message client ".

Пример сообщения пользователям asu

```
msgclnt  asu  'hey !'
```

В результате выдается звуковой сигнал, текст сообщения, от кого оно получено, время и дата посылки.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с теоретической частью к лабораторной работе.
2. Войти в систему UNIX.
3. Проверить наличие связи с UNIX - сервером.
4. Получить информацию о работающих пользователях, подсчитать их количество и запомнить в файле. Установить резидентную программу подачи звукового сигнала при поступлении сообщения с другого терминала.
5. Послать приветствие на соседний терминал.
6. Используя редактор операционной системы UNIX VI (см. приложение), написать программу на языке СИ и запустить ее на трансляцию в фоновом режиме.
7. Получить подробную информацию о всех активных процессах.
8. Просмотреть приоритет своего процесса и ускорить его выполнение за счет повышения приоритета.
9. Сравнить время выполнения процесса с низким и высоким приоритетом.
10. Используя редактор ОС UNIX VI, создать текстовый файл (с

расширением TXT) и командой CAT просмотреть его на экране.

11. Просмотреть содержимое текущей директории.
12. С помощью электронной почты послать файл на соседний терминал.
13. Объединить текстовые файлы в единый файл и просмотреть его на экране.
14. Показать преподавателю исходный текст программы на языке СИ, текстовый файл, файл с сохранением количества пользователей.
15. Продемонстрировать выполнение программы.
16. Удалить свои файлы.
17. Выйти из системы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите основные функции и назначение многозадачной операционной системы Unix и ее отличительные особенности от однопрограммной системы DOS.
2. Какое назначение имеет ядро системы и интерпретатор команд ?
3. В чем заключается понятие "процесс" и какие операции можно выполнить над процессами?
4. Как задаются и выполняются простые и сложные команды?
5. Каким образом можно послать сообщение на другой удаленный терминал и на файл-сервер?
6. Что такое эмуляция терминала, файл-сервер, клиент_сервер?

ЛАБОРАТОРНАЯ РАБОТА N 2
ИЗУЧЕНИЕ ФАЙЛОВОЙ СИСТЕМЫ И ФУНКЦИЙ ПО ОБРАБОТКЕ И УПРАВЛЕНИЮ ДАННЫМИ

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение структуры файловой системы ОС UNIX, изучение команд создания, удаления, модификации файлов и

каталогов, функции манипулирования данными.

2. ОБЩИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Файловая структура системы UNIX

В операционной системе UNIX файлами считаются обычные файлы, каталоги, а также специальные файлы, соответствующие периферийным устройствам (каждое устройство представляется в виде файла). Доступ ко всем файлам однотипен, в том числе и к файлам периферийных устройств. Такой подход обеспечивает независимость программы пользователя от особенностей ввода-вывода на конкретное внешнее устройство.

Файловая структура Unix имеет иерархическую древовидную структуру [3]. В корневом каталоге размещаются другие каталоги и файлы, включая 5 основных каталогов:

bin - большинство выполняемых командных программ и shell-процедур;

tmp - временные файлы;

usr - каталоги пользователей (условное обозначение);

etc - преимущественно административные утилиты и файлы;

dev - специальные файлы, представляющие периферийные устройства; при добавлении периферийного устройства в каталог /dev должен быть добавлен соответствующий файл (черта / означает принадлежность корневому каталогу).

Текущий каталог - это каталог, в котором в данный момент находится пользователь.При наличии прав доступа пользователь может перейти после входа в систему в другой каталог.Текущий каталог обозначается точкой (.), родительский каталог,которому принадлежит текущий, обозначается двумя точками (..). Структура файловой системы представлена на рис. 2.1.

2.2. Создание связи с файлом

Полное имя файла может включать имена каталогов, включая

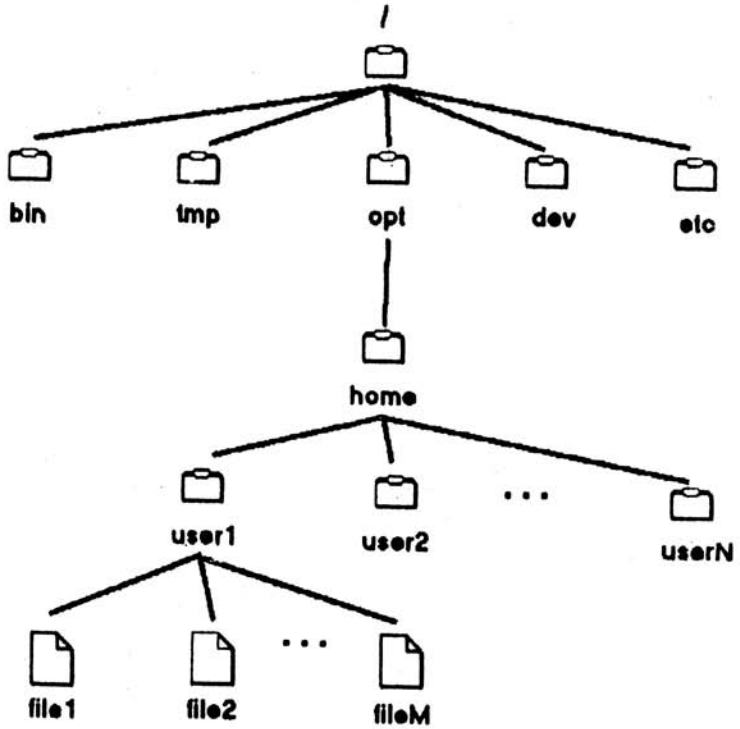

Рис.2.1

корневой, разделенных косой чертой, например:

/user/file_1

Первая косая черта обозначает корневой каталог, и поиск файла будет начинаться с него, а затем в каталоге user. Один файл можно сделать принадлежащим нескольким каталогам. Для этого используется команда ln (link):

ln <имя файла 1> < имя файла 2>

Имя 1-го файла - это полное составное имя файла, с которым устанавливается связь; имя 2-го файла - это полное имя файла в новом каталоге, где будет использоваться эта связь. Новое имя может не отличаться от старого. Каждый файл может иметь несколько связей, т.е. он может использоваться в разных каталогах под разными именами.

2.3. Определение доступа к файлам

В Unix различаются 3 уровня доступа:

1) доступ владельца файла;

2) доступ группы пользователей, к которой принадлежит владелец файла;

3) остальные пользователи.

Для каждого уровня существуют свои байты атрибутов, значения которых расшифровываются следующим образом:

r - разрешение на чтение;

w - разрешение на запись;

x - разрешение на выполнение;

- - отсутствие разрешения.

Первый символ байта атрибутов определяет тип файла и может интерпретироваться со следующими значениями;

- - обычный файл;

d - каталог;

b - блок-ориентированный специальный файл, который соответствует таким периферийным устройствам, как накопители на магнитных дисках;

c - байт-ориентированный специальный файл, который мо-

мет соответствовать таким периферийным устройствам, как принтер, терминал.

В домашнем каталоге пользователь имеет полный доступ к файлам (read, write, execute; r,w,x).

Атрибуты файла можно просмотреть командой

```
ls  -l
```

и они представляются в следующем формате:

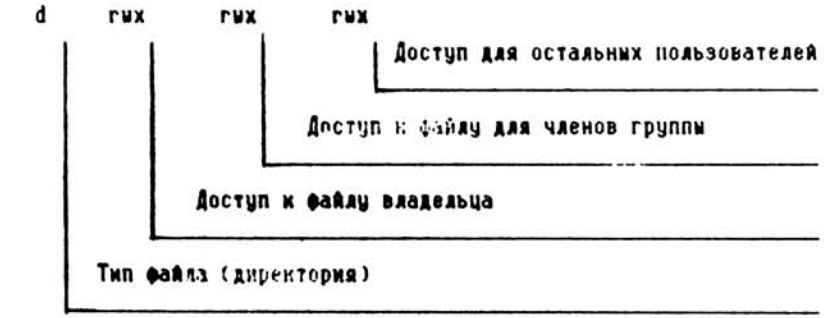

Рис.2.2

Пример. В соответствии с рис. 2.2 получим листинг содержимого текущей директории user_1 :

```
ls -l
- rwx  --- --- 1 student 100 Mar 10  10:30  file_1
- rwx  --- r-- 1 adm     200 May 20  11:15  file_2
- rwx  --- r-- 1 student 100 May 20  12:50  file_3
```

После байтов атрибутов на экран выводится следующая информация о файле:

число связей файла;
имя владельца файла;
размер файла в байтах;
дата создания файла (или модификации);
время;
имя файла.

Атрибуты файла, а соответственно и доступ к нему, можно изменить командой

chmod <Коды защиты> <Имя файла>

Коды защиты могут быть заданы в числовом или символьном виде. Для символьного кода используются:

знак плюс (+) - добавить права доступа;

знак минус (-) - отменить права доступа;

r,w,x - доступ на чтение, запись, выполнение;

u,g,o - владельца, группы, остальных.

Примеры.

1. Изменение атрибутов

```
chmod d+w, o+rw file_1
```

2. Чтение атрибутов

```
ls-l  file_1
```

Ответ:

```
rwx-w-rw-1 student 100 Mar 1010:30 file_1
```

3. Отмена атрибута записи у остальных пользователей

```
chmod  O-W  file_1
```

Коды защиты в числовом виде могут быть заданы в восьмеричной форме. Для контроля установленного доступа к своему файлу после каждого изменения кода защиты нужно проверять свои действия с помощью команды ls -l .

2.4. Работа с файлами

Создание файла. Символ > используется как для переадресации, так и для создания файла. Пример:

```
>letter
```

Вывод содержимого файла:

```
cat  <имя файла>
```

Конкатенация файлов (объединение):

```
cat file_1  file_2 > file_12
```

Переименование файла file_1 в file_2:

```
mv file_1  file_2
```

Перемещение файлов file_1, file_2, file_3 в указанную дирек-

торию:

```
mv  file_1 file_2 file_3   directory
```

Удаление файлов командой rm:

```
rm  file_1  file_2   file_3
```

Удаление каталогов командой rmdir:

```
rmdir  dir_1  dir_2
```

Создание каталога командой mkdir:

```
mkdir  dirname
```

Вывод содержимого каталога:

```
ls [-acdfgilqrstvCFR] namedir
```

Если в качестве namedir указано имя файла, то выдается вся информация об этом файле.

Значения ключей:

- l - листинг включает всю информацию о файлах;
- t - сортировка по времени модификации файлов;
- a - в список включаются все файлы, в том числе и те, которые начинаются с точки;
- s - размеры файлов указываются в блоках;
- d - вывести имя самого каталога, но не содержимое;
- r - сортировка строк вывода;
- i - указать идентификационный номер каждого файла;
- v - сортировка файлов по времени последнего доступа;
- q - непечатаемые символы заменить на знак ?;
- c - использовать время создания файла при сортировке;
- g - то же, что -l, но с указанием имени группы пользователей;
- f - вывод содержимого всех указанных каталогов, отменяет флаги - l, -t, -s, -r и активизирует флаг -a;
- C - вывод элементов каталога в несколько столбцов;
- F - добавление к имени каталога символа / и символа * к имени файла, для которых разрешено выполнение;
- R - рекурсивный вывод содержимого подкаталогов заданного каталога.

Переход в другой каталог:

cd namedir

Вывод имени текущего каталога:

pwd

Поиск в файле по шаблону:

grep [-vcilns] <шаблон поиска> <имя файла>

Значение ключей:

v - выводятся строки, не содержащие шаблон поиска;

c - выводится только число строк, содержащих или не содержащих шаблон;

i - при поиске не различаются прописные и строчные буквы;

l - выводятся только имена файлов, содержащие указанный шаблон;

n - перенумеровать выводимые строки;

s - формируется только код завершения.

Примеры.

1. Напечатать имена всех файлов текущего каталога, содержащих последовательность "student":

grep - l student *

2. Определить имя пользователя, входящего в ОС UNIX, с терминала tty23:

who | grep tty23

Шифрование информации:

crypt [password]

Программа crypt создает зашифрованный текст с помощью символов ключа (password). Для восстановления текста необходимо сообщить этот же ключ.

Пример. Зашифровать текстовый файл с именем letter с ключом W-PO-t3:

crypt < letter > letter.crypt

на сообщение Enter key: необходимо ввести ключ W-PO-t3. Знаки <,> в командной строке означают переадресацию ввода и вывода.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с файловой структурой ОС UNIX. Изучить команды работы с файлами.
2. Используя команды ОС UNIX, создать два текстовых файла. Войти в редактор VI,набрать текст и сохранить оба файла.
3. Полученные файлы объединить в один файл и просмотреть на экране.
4. Создать новую директорию и переместить в нее полученные файлы.
5. Вывести полную информацию о всех файлах и проанализировать уровень доступа.
6. Добавить для всех трех файлов право чтения членам группы и остальным пользователям.
7. Воспроизвести коды защиты файлов.
8. Создать еще один каталог.
9. Установить дополнительную связь объединенного файла с новым каталогом, но под другим именем.
10. Сделать текущим новый каталог и вывести на экран расширенный листинг о его файлах.
11. Произвести поиск заданной последовательности символов в файлах текущей директории и получить перечень соответствующих файлов.
12. Зашифровать свои файлы с использованием ключа, а затем просмотреть их в редакторе в расшифрованном виде.
13. Получить информацию об активных процессах и именах других пользователей.
14. После сдачи отчета о проделанной работе необходимо удалить свои файлы и каталоги.
15. Выйти из системы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что считается файлами в ОС UNIX?
2. Объясните назначение связей с файлами и способы их создания.

3. Что определяют атрибуты файлов и каким образом их можно просмотреть и изменить?
4. Какие методы создания и удаления файлов, каталогов вы знаете?
5. В чем заключается поиск по шаблону?
6. Какие возможности имеются в системе ОС UNIX для шифрования данных?
7. Какой командой можно получить список работающих пользователей и сохранить его в файле?

ЛАБОРАТОРНАЯ РАБОТА N 3
СЕТЕВАЯ СРЕДА СИСТЕМЫ UNIX

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение возможностей сетевых приложений системы UNIX, передача и обработка данных между удаленными терминалами и файл - сервером.

2. ОБЩИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Сетевая среда

В состав операционной системы UNIX типа SunOS входит программный комплекс PC-NFS (Network File System) для выполнения сетевых функций. PC-NFS ориентирован для конкретной ОС персонального компьютера (PC) и включает драйверы для работы в сети и дополнительные утилиты. При нормальной загрузке PC-NFS выдается следующее сообщение:

PC-NFS is installed

и указывается имя PC и IP - адрес, например:

uran 129.200.9.6.

2.2. Утилиты PC-NFS

ping <имя сервера> - проверка связи с сервером.

ftp (File transfer protocol) - протокол передачи файлов.

Загрузка fpt (в среде PC-NFS):

ftp [Enter]

После этого можно выполнить нижеперечисленные функции:

установка связи с файл-сервером

open <имя сервера>

При этом выдается запрос имени и пароля пользователя, который хочет установить связь:

name: <идентификатор пользователя>

password: <пароль>

получение списка доступных команд для FTP

знак вопроса (?)

подсказка о конкретной команде

help <имя команды FTP>

настройка на передачу двоичных файлов

bin

настройка на передачу текстовых файлов в коде ASCII

ascii

Разница между текстовыми файлами DOS и UNIX в том, что в DOS-файле в конце строки ставится возврат каретки и перевод строки, а в UNIX-файле только перевод строки [4].

получение имени текущей директории

pwd

листинг содержимого текущей директории

ls

копирование файла с сервера на локальную машину

get

пересылка файла с локальной машины на сервер

put

окончание сеанса с сервером

close

выход из ftp

bye

ftpd - программа, переводящая локальную машину в режим

ftp-сервера.

netstat - получение статистических данных о приеме и передаче информации.

unix2dos - преобразование текстовых файлов из формата UNIX в формат DOS.

nfsping <имя сервера> - проверка связи с сервером по протоколу высокого уровня (ping - это проверка физической связи, nfsping - проверка логических связей)

arp- получение адреса Internet и Ethernet карты.

rsh (remote shell) - запуск команд на сервере с локальной машины.

В табл 2.1 приводится семиуровневая модель OSI с указанием протоколов, реализующих взаимодействие различных платформ.

Таблица 2.1

N п/п	Название уровня	Протоколы
1	Физический (Physical)	Ethernet, Token Ring,etc
2	Канальный (Link)	Ethernet, Token Ring,etc
3	Сетевой (Network)	IP
4	Транспортный (Transport)	TCP
5	Сеансовый (Session)	TLI, Berkeley, Strems
6	Предст.данных (Presentation)	Null
7	Прикладной (Application)	NFS,TELNET,FTP,SNMP

Sun Microsystems вместе с другими фирмами предлагают разработки, позволяющие взаимодействовать в рамках одной среды таким системам, как DOS, Windows, Macintosh, OS/2, UNIX.

Любой каталог UNIX-сервера можно использовать в качестве сетевого диска на локальной машине. Такие каталоги называются разделяемыми. Для просмотра списка разделяемых каталогов используют

shell-команду dfshares. Чтобы подключить сетевой диск, используется команда DOS

net use f: host:/dir/subdir

f - имя сетевого диска;

host - имя сервера;

/dir/subdir - имя каталога.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Проверьте наличие связи с файл - сервером на низком и высоком уровне.
2. Установите связь с файл - сервером по FTP.
3. Скопируйте текст и исполняемые файлы с сервера на локальную машину и обратно в режиме BIN и ASCII.
4. Преобразуйте текстовый файл из UNIX, полученный в режиме BIN, в DOS - формат.
5. Выведите информацию о статистике передачи.
6. Получите адрес Ethernet - карты.
7. Установите FTP-соединение между локальными машинами.
8. Скопируйте файл между локальными машинами в режиме BIN и ASCII.
9. Проверьте, какие каталоги UNIX-сервера являются разделяемыми.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Функции и назначение стандарта сетевой файловой системы фирмы SUN.
2. Какие значения имеют IP - адреса у локальной машины и сервера и как их можно узнать?
3. Какие сетевые утилиты вы знаете?
4. Какие утилиты служат для проверки связей?
5. Как реализован протокол передачи файлов, какие используются команды?
6. Назовите команды для получения статистики работы сети.

7. Как получить IP и Ethernet-адрес машины?
8. Сколько уровней имеет модель OSI , какими средствами реализован каждый из уровней?
9. Можно ли копировать файлы между локальными машинами без использования UNIX - сервера?
10. Каким образом можно подключить диск SUN на ПК?

ЛАБОРАТОРНАЯ РАБОТА N 4
СОЗДАНИЕ КОМАНДНЫХ ФАЙЛОВ НА ЯЗЫКЕ SHELL-ИНТЕРПРЕТАТОРА

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение методов создания и выполнения командных файлов на языке shell-интерпретатора.

2. ОБЩИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Назначение языка shell-интерпретатора

Мы изучили взаимодействие с интерпретатором shell, использовав командную строку. shell является также и языком программирования, который применяется для написания командных файлов (shell-файлов) [5]. shell-файл содержит одну или несколько подряд выполняемых команд (процедур), а имя файла используется как имя создаваемой сложной команды.

2.2. Переменные shell

Для обозначения переменных shell используется последовательность букв, цифр и символов подчеркивания; переменные не могут начинаться с цифры. Присваивание значений переменным проводится с использованием знака = , например, PS2 = '<' . Для обращения к значению переменной перед ее именем ставится знак $. Их можно разделить на следующие группы:

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

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

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

Простые переменные. shell присваивает значения переменным справа налево:

```
x= $z  z=1000
echo $x
1000
```

Здесь переменной x присвоено значение z, заданное справа.

Позиционные переменные. Переменные вида $n, где n - целое число, используются для идентификации позиций элементов в командной строке с помощью номеров, начиная с нуля. Например, в командной строке

```
cat text_1 text_2...text_9
```

интерпретатор расчленяет строку,и аргументы идентифицируются параметрами $1...$9. Для имени команды всегда используется $0. Итак, $0 - это cat, $1 - это text_1, $2 - это text_2 и т.д. Для присваивания значений позиционным переменным используется команда set, например:

```
set  arg_1  arg_2... arg_9
```

Здесь $1 присваивается значение первого аргумента arg_1, $2- arg_2 и т.д. Для доступа к аргументам используется команда echo, например

```
echo  $1  $2  $9
arg_1  arg_2  arg_9
```

Для получения информации о всех аргументах (включая последний) используют метасимвол *. Пример:

```
echo  $*
arg_2  arg_3...  arg_10  arg_11  arg_12
```

С помощью позиционных переменных shell удается сохранить имя команды с помощью $0 и ее аргументы (с помощью $1 $2..). Для выполнения команды интерпретатор shell должен передать ей аргументы; их порядок может регулироваться с помощью позиционных переменных.

Специальные переменные. Переменные - ? # $! устанавливаются только shell. Они позволяют с помощью команды echo получить следующую информацию:

- - текущие флаги интерпретатора (установка флагов может быть изменена командой set);
- # - число аргументов, которое было сохранено интерпретатором при выполнении какой-либо команды;
- ? - код возврата последней выполняемой команды;
- $ - числовой идентификатор текущего процесса PID;
- ! - PID последнего фонового процесса.

2.3. Арифметические действия, анализ цепочек символов

Команда expr (express -- выражать) вычисляет выражение expression и записывает результат в стандартный вывод. Элементы выражения разделяются пробелами; символы, имеющие специальный смысл в командном языке, нужно экранировать. Строки, содержащие специальные символы, заключают в апострофы. Используя команду expr, можно выполнять сложение, вычитание, умножение, деление, взятие остатка, сопоставление символов с указанием числа совпадающих символов и т.д.

Пример. Сложение, вычитание:

```
b=196
a=' expr 1781 - $b'
```

умножение *, деление /, взятие остатка % :

```
d=' expr $a + 125 '*' 209'
c=' expr $d % 13'
```

Здесь знак умножения заключается в апострофы (можно использовать такие кавычки или символ \), для того чтобы интерпретатор не воспринимал его как метасимвол. Во второй строке переменной

'c' присваивается значение остатка от деления переменной d на 13 (d % 13).

Сопоставление символов с указанием числа совпадений:

```
concur=' expr 'abcdefgh' : 'abcde' '
echo   $concur
```

ответ 5.

Операция сопоставления обозначается двоеточием (:) . Результат - переменная concur.

Подсчет числа символов в цепочках символов. Операция выполняется путем сопоставления цепочки с цепочкой, генерируемой интерпретатором. Создадим цепочку chain:

```
chain='The program is written in Assembler'.
Set=' expr $chain: '*''
Echo $Set
```

ответ 41. Здесь результат подсчета обозначен переменной Set.

2.4. Встроенные команды

Встроенные команды являются частью интерпретатора и не требуют для своего выполнения проведения последовательности поиска файла команды и создания новых процессов. Встроенные команды:

cd [dir] -- назначение текущего каталога;

exec [cmd [arg...]] <имя файла> - выполнение команды, заданной аргументами cmd и arg, путем вызова соответствующего выполняемого файла.

umask [-o | -s] [nnn] - устанавливает маску создания файла (маску режимов доступа создаваемого файла, равную восьмеричному числу nnn: 3 восьмеричных цифры для пользователя,группы и других). Если аргумент nnn отсутствует,то команда сообщает текущее значение маски. При наличии флага -o маска выводится в восьмеричном виде, при наличии флага -s - в символьном представлении:

set, unset - режим работы интерпретатора, присваивание значе-

ний параметрам;

eval [* arg] - вычисление и выполнение команды;

exit [n] - приводит к прекращению выполнения программы, передает код возврата, равный нулю, в вызывающую программу;

trap [cmd] [cond] - перехват сигналов прерывания, где
cmd - выполняемая команда;
cond=0 или EXIT - в этом случае команда cmd выполняется при завершении интерпретатора;
cond=ERR - команда cmd выполняется при обнаружении ошибки;
cond - символьное или числовое обозначение сигнала, в этом случае команда cmd выполняется при приходе этого сигнала;

export [name [=word]...] -- включение в среду. Команда export объявляет,что переменные name будут включаться в среду всех вызываемых впоследствии команд;

wait [n] - ожидание завершения процесса.Команда без аргументов ожидает завершения процессов, запущенных синхронно. Если указан числовой аргумент n, то wait ожидает фоновый процесс с номером n;

read name... - команда вводит строку со стандартного ввода и присваивает прочитанные слова переменным, заданным аргументами name;

Пример. Пусть имеется shell-файл data,содержащий две команды:

```
echo -n  'Please write down your name:"
read name
```

Если вызвать файл на выполнение, введя его имя, то на экране появится сообщение

```
Please write down your name:_
```

Программа ожидает ввода с клавиатуры (в данном случае -- фамилии пользователя). После ввода фамилии и нажатия клавиши <ввод> команда выполнится и на следующей строке появится знак-приглаше-

ние.

2.5. Управление программами

Предварительно нужно рассмотреть команды true и false. Команды служат для установления требуемого кода завершения процесса: true - успешное завершение, код завершения 0, false- неуспешное завершение, код может иметь несколько значений, с помощью которых определяется причина неуспешного завершения. Коды завершения команд используются для принятия решения о дальнейших действиях в операторах цикла while и until и в условном операторе if. Многие команды UNIX вырабатывают код завершения только для поддержки этих операторов [6].

2.6. Условный оператор if

Оператор if проверяет значение выражения.Если оно равно true, shell выполняет следующий за if оператор, если false, то следующий оператор пропускается. Формат оператора if :

```
if <условие>
then
    list1
else
    list2
fi
```

2.7. Команда test

Команда test (проверить) используется с условным оператором if и операторами циклов. Действия при этом зависят от кода возврата test. Test проводит анализ файлов, числовых значений, цепочек символов. Нулевой код выдается,если при проверке результат положителен, ненулевой код-при отрицательном результате проверки. В случае анализа файлов синтаксис команды следующий:

```
test  [ -rwfds]  file
```

где

r - файл существует и его можно прочитать(код завершения 0);

w - файл существует и в него можно записывать;

f - файл существует и не является каталогом;

d - файл существует и является каталогом;

s - размер файла отличен от нуля.

При анализе числовых значений команда test проверяет, истинно ли данное отношение, например, равны ли A и B . Сравнение выполняется в формате:

test A	-eq	B равносильно	A = B?
	-ne		A<> B?
	-ge		A>= B?
	-le		A<= B?
	-gt		A > B?
	-lt		A < B?

Кроме команды test, имеются еще некоторые средства для проверки:

! - операция отрицания инвертирует значение выражения, например, выражение if test true эквивалентно выражению if test ! false;

o - двухместная операция "ИЛИ" (or) дает значение true, если один из операндов имеет значение true;

a - двухместная операция "И" (and) дает значение true, если оба операнда имеют значение true.

2.8. Циклы

2.8.1. Операторы цикла с условием

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

```
while list1
do
    list2
done
```

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

Оператор цикла с условием while false имеет формат:

```
until list1
do
    list2
done
```

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

2.8.2. Оператор цикла с перечислением

```
for name in [wordlist]
do
  list
done
```

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

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

2.9. Ветвление по многим направлениям case

Команда case обеспечивает ветвление по многим направлениям в зависимости от значений аргументов команды. Формат:

```
case <string> in
  s1) <list1>;;
```

```
    s2) <list2>;;
        .
        .
        .
    sn) <listn>;;
  esac
```

Здесь list1,list2...listn - список команд.Производится сравнение шаблона string с шаблонами s1,s2...sk...sn. При совпадении выполняется список команд, стоящий между текущим шаблоном sk и соответствующими знаками ;;.

Пример.

```
  echo -n  'Please, write down your age'
  read age
  case $age in
     test  $age  -le 20)  echo  'you are so young' ;;
     test  $age  -le 40) echo  'you are still young' ;;
     test  $age  -le 70) echo   'you are too young' ;;
     *)echo  'Please,write down once more'
  esac
```

В конце текста помещена звездочка * на случай неправильного ввода числа.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Составьте и выполните shell-программы, включающие следующие действия:

1. Вывод на экран списка параметров командной строки с указанием номера каждого параметра.
2. Присвоение переменных A,B и C со значениями 10, 100, 200, вычисления и вывод результатов по формуле

   $$D=(A*2 + B/3)*C$$

3. Формирование файла со списком работающих пользователей, вывод на экран этого списка в алфавитном порядке и общее количество пользователей.

4. Переход в другой каталог, формирование файла с листингом каталога и возвращение в исходный каталог.
5. Запрос и ввод имени пользователя, сравнение с текущим логическим именем пользователя и вывод сообщения: верно/неверно.
6. Запрос и ввод имени файла в текущем каталоге и вывод сообщения о типе файла.
7. Циклическое чтение системного времени в фоновом режиме и очистка экрана в заданный момент.
8. Циклический просмотр списка работающих пользователей и выдача сообщения при появлении заданного имени в списке.
9. Циклическое отправление разных сообщений на другую машину.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какое назначение имеют shell-файлы?
2. Как создать shell-файл и сделать его выполняемым?
3. Какие типы переменных используются в shell-файлах?
4. В чем заключается анализ цепочки символов?
5. Назовите встроенные команды, используемые в shell-файлах.
6. Какие встроенные команды используются в shell-файлах?
7. Как производится управление программами?
8. Назовите операторы создания циклов.
9. Назовите несколько способов создания shell-файлов.

ПРИЛОЖЕНИЕ

ТЕКСТОВЫЙ РЕДАКТОР VI

1. ЗАПУСК ПРОГРАММЫ

В составе ОС Unix обычно поставляются текстовые редакторы. Для вызова редактора vi используется команда vi:

vi [+line] [-R] [-x] [-r] [-t] file...

где +line -- номер строки, с которой Вы хотите начать редактирование;
R -- читать;
x -- просмотр файла, зашифрованного командой crypt, или редактирование обычного текста с шифрованием при записи на диск;
r -- восстановление файла после системного краха.

Команду вызова редактора можно использовать в форме vi+/word/file - начало редактирования файла file с первой строки,которая содержит слово word, или в форме vi+file - начало редактирования файла с последней строки.

2. СТРУКТУРА РЕДАКТОРА

2.1. Режимы работы редактора

Работая с редактором, Вы находитесь или в одном из его командных режимов, или в режиме ввода текста. В простейшем случае для вызова редактора введите vi text и <Enter>.

В данный момент Вы находитесь в командном режиме vi. Перейти в режим ввода текста можно с помощью команд добавления текста, которые не отображаются на экране после их ввода:

a/A -- ввод текста после курсора/после конца строки;
i/I -- вставка текста перед курсором/с начала данной строки;
o/O -- образовать пустую строку ниже/выше имеющейся.

Для выполнения команд после ввода текста нужно перейти в командный режим vi, нажав клавишу ESC. В режиме ввода текста набирайте текст, нажимая <Enter> в конце строки. Курсор в режиме ввода текста можно перемещать вправо клавишей пробела и влево клавишей BACKSPACE.

Для перехода в командный режим vi нажмите клавишу ESC. Теперь Вы находитесь в командном режиме vi. Рассмотрим две команды редактора vi: повторение последней команды (.); откат последней команды (u).

2.2. Переход в режим ex

Команды режима ex начинаются с символа : и отображаются в

нижней части экрана. После нажатия клавиши ESC или <Enter> происходит возврат в командный режим. Команды режима ex:
:w -- запись текста в файл;
:r -- чтение файла;
:e -- редактирование нового файла;
:e! -- выход из старого файла без его сохранения и начало редактирования нового;
:n -- авторедактирование;
:wq -- запись текста и выход из редактора;
:x -- запись текста только при наличии в нем изменений;
:q! -- оставить текст в рабочей области и закончить редактирование;
:! -- запуск команды Shell (например :!ls);
:!sh -- временный выход в Shell (возврат в vi Ctrl-D).

3. ОСНОВНЫЕ КОМАНДЫ vi

3.1. Перемещение курсора

Кроме клавиш со стрелками, для перемещения курсора используют клавиши: CTRL-H -- влево; CTRL-N -- вниз; CTRL-P -- вверх; SPACE -- вправо. Команды перемещения курсора: h -- на одну позицию влево; l -- на одну позицию вправо; j -- на одну позицию вниз; k -- на одну позицию вверх; 0 -- к началу текущей строки; $ -- к концу текущей строки; H -- к началу экрана; M -- на середину экрана; L -- к концу экрана; nG -- к строке с номером n (на последнюю строку, если номера n нет); % -- к символу парной скобки, если курсор находится под одной из них.

3.2. Удаление и замена букв

Команды удаления:
dw -- до конца текущего слова;
d0 -- удаление начала строки;
d$ -- удаление конца строки;
dd -- удаление всей строки;
x -- удаление символа;
~ -- замена строчных букв на прописные и наоборот.

3.3. Определение текущей позиции

После ввода пользователем в командном режиме CTRL-G в нижней части экрана появится статусная информация в соответствии с положением курсора в тексте, включающая имя файла; сведения о проведенной ранее модификации; номер текущей строки; общее число строк; расстояние курсора от начала файла до курсора (в процентах).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Керниган Б.В., Пайк Р. UNIX - Универсальная среда программирования / Пер. с англ. - М.: Финансы и статистика, 1992. - 304 с.

2. Пупков К.А. и др. Освоение операционной системы UNIX. - М.: Радио и связь, 1994.- 113 с.

3. Баурн С. Операционная система UNIX.- М.:Мир,1986 - 463 с.

4. Дегтяров Е.К. Введение в UNIX. - М.: МП "Память", 1991 - 128 с.

5. Браун П.Д. Введение в операционную систему UNIX. - М.: Мир; 1987 - 287 с.

6. Sun ОС для пользователей - Open Windows (EU - 113): Учебное пособие / Пер. с англ. - М.: МГУ REDLab, 1993 - 215 с.

Составители: ПАРФЕНОВА Мария Яковлевна
АРЬКОВ Валентин Юльевич

РАБОТА С ОПЕРАЦИОННОЙ СИСТЕМОЙ UNIX ТИПА SunOS

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторному практикуму по курсу
«Операционные системы» для студентов направления Т28
«Информатика и вычислительная техника»

Редактор З. Г. Кашаева
ЛР № 020258 от 30.10.91
Подписано к печати 06.09.95. Формат 60x84 1/16 Бумага оберточная.
Печать плоская. Усл. печ. л. 2,5. Усл, кр.-отт. 2,5. Уч.-изд, 2,4,
Тираж 150 экз. Заказ № 614. Бесплатно.

Уфимский государственный авиационный технический университет.
Уфимская типография № 2 Министерства печати и массовой информации
Республики Башкортостан
450000, Уфа-центр, ул. К. Маркса, 12