Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра автоматизации обработки информации (АОИ)

Ю. Б. Гриценко

ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ, СЕТИ И ТЕЛЕКОММУНИКАЦИИ

Методические указания к выполнению лабораторной работы

2016

Корректор: Осипова Е. А.

Гриценко Ю. Б.
Вычислительные системы, сети и телекоммуникации: методические указания к выполнению лабораторной работы. — Томск: Факультет дистанционного обучения, ТУСУР, 2016. — 61 с.

СОДЕРЖАНИЕ

Введение 4
1 Интерфейсы операционных систем 5
1.1 Интерфейс командной строки ОС Windows 5
1.2 Интерфейс командной строки ОС Unix 25
2 Задание к лабораторной работе 51
3 Варианты заданий на выполнение 52
Список литературы 61

ВВЕДЕНИЕ

Целью дисциплины «Вычислительные системы, сети и телекоммуникации» является формирование у студента профессиональных знаний по теоретическим основам построения и функционирования компьютеров вычислительных систем, телекоммуникационных вычислительных сетей и коммуникаций, их структурной и функциональной организации, программному обеспечению, эффективности и перспективам развития.

В рамках изучения дисциплины необходимо выполнить одну лабораторную работу.

Выбор варианта лабораторной работы осуществляется по общим правилам с использованием следующей формулы:

$$\mathbf{V = (N \times K) \ div \ 100,}$$

где V — искомый номер варианта,

N — общее количество вариантов,

div — целочисленное деление,

при V= 0 выбирается максимальный вариант,

K — код варианта.

Варианты заданий представлены в разделе 3 методических указаний.

1 ИНТЕРФЕЙСЫ ОПЕРАЦИОННЫХ СИСТЕМ

1.1 Интерфейс командной строки ОС Windows

Интерфейсы операционных систем можно разделить на два класса: графические интерфейсы пользователя (GUI) и **интерфейсы командной строки (CPI — Command Prompt Interface)**.

Графический интерфейс пользователя в Windows обеспечивается процессом Explorer и, как правило, хорошо знаком большинству пользователей персональных компьютеров. Графические интерфейсы систем на платформе Unix бывают различными, что связано с большим количеством версий, но работа в них мало чем отличается от работы в операционной системе Windows. Графический интерфейс в Linux чаще всего реализуется с помощью графических оболочек KDE и Gnome. Загрузка операционной системы Linux Red Hat заканчивается тем, что на экране появляется окно графической оболочки Gnome.

Эффективная профессиональная работа опытного пользователя с операционной системой компьютера немыслима без овладения интерфейсом, обеспечиваемым командной строкой [1]. Преимуществом данного интерфейса служит возможность более гибко управлять ресурсами системы, чем с помощью графического интерфейса.

Интерфейс командной строки в ОС Windows присутствует, но играет для пользователей вспомогательную роль. В свое время он формировался как некое подмножество команд интерфейса Unix-подобных систем и особого развития не получил. Однако следует сказать, что интерфейс командной строки во многих нештатных ситуациях остается единственным средством определения рассогласований и «тонкой настройки» аппаратно-программных средств. В последних версиях операционных систем Microsoft (Windows Server 2008) интерфейс командной строки получил даль-

нейшее развитие и превратился в мощный инструмент администрирования системы — оболочку Power Shell.

В новых версиях операционных систем Windows с учетом роста сложности аппаратной и программной частей компьютерных систем добавлен ряд команд, позволяющих решать задачи администрирования системы. Часть команд, заимствованных из MS DOS, получили дополнительные возможности. Например, такие команды, как dir, copy, xcopy; rename и др., в новых редакциях Windows могут работать с длинными именами файлов.

Включение режима командной строки может быть выполнено двумя способами (рис. 1.1):

1. Нажать на экране кнопку «Пуск» —> «Выполнить», затем в появившемся окне набрать cmd.

2. Выбрать из главного меню: «Пуск» —> «Программы» —> «Стандартные» —> «Командная строка».

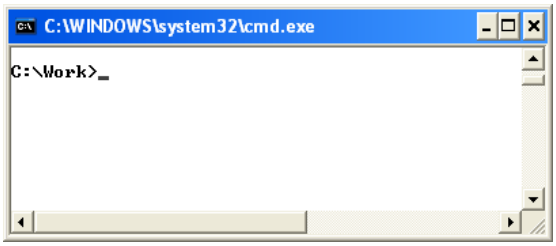

Рис. 1.1 — Окно «Командная строка»

Здесь указываются версия работающей операционной системы и строка приглашения с активным диском и рабочим каталогом (папкой). Выключение режима обеспечивается набором в строке приглашения команды exit и ее выполнением при нажатии клавиши <Enter>.

Смена текущего диска указывается путем указания его имени и двоеточия на конце. Например, чтобы перейти на диск D, необходимо указать D: и нажать клавишу <Enter>.

Перечень команд. В состав внутренних команд Windows входит около 70 команд. Перечень команд можно посмотреть с помощью команды HELP.

Поскольку весь перечень команд перекрывает размер экрана дисплея, то для ознакомления с каждым элементом перечня следует использовать полосу прокрутки окна или вызов на экран частей перечня постранично. Для этого следует набрать более сложную команду, состоящую из конвейера двух команд ***HELP*** | ***MORE*** (рис. 1.2).

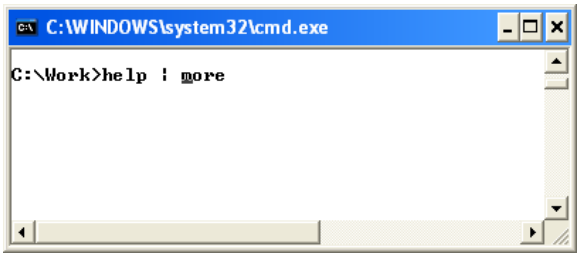

Рис. 1.2 — Ввод команд в командной строке

Список команд лучше рассмотреть по группам:

– команды справочной системы (табл. 1.1);
– команды файловой системы (табл. 1.2);
– команды управления работой ОС (табл. 1.3);
– команды пакетных (командных) файлов (табл. 1.4).

Таблица 1.1 — Команды справочной системы

HELP	Выводит справочную информацию о системе команд с версии Windows 2000
HELP имя_команды	Выводит справочную информацию о набранной команде
имя_команды /?	Выводит справочную информацию о набранной команде

Таблица 1.2 — Команды файловой системы

ATTRIB	Отображение и изменение атрибутов файлов
CD	Вывод имени либо смена текущей папки
CHDIR	
CHKDSK	Проверка диска и вывод статистики

Окончание табл. 1.2

COMP	Сравнение содержимого двух файлов или двух наборов файлов
COPY	Копирование одного или нескольких файлов в другое место
DEL	Удаление одного или нескольких файлов
DIR	Вывод списка файлов и подпапок из указанной папки
DISKCOMP	Сравнение содержимого двух гибких дисков
DISKCOPY	Копирование содержимого одного гибкого диска на другой
ERASE	Удаление одного или нескольких файлов
FC	Сравнение двух файлов или двух наборов файлов и вывод различий между ними
FIND	Поиск текстовой строки в одном или нескольких файлах
FINDSTR	Поиск строк в файлах
FORMAT	Форматирование диска
LABEL	Создание, изменение и удаление меток тома для дисков
MD	Создание папки
MKDIR	
MOVE	Перемещение одного или нескольких файлов из одной папки в другую
PUSHD	Сохранение значения текущей активной папки и переход к другой папке
POPD	Восстановление предыдущего значения текущей активной папки, сохраненного с помощью команды PUSHD
PRINT	Вывод на печать содержимого текстовых файлов
RD	Удаление папки
REN	Переименование файлов и папок
RENAME	
REPLACE	Замещение файлов
RMDIR	Удаление папки
SORT	Сортировка ввода
TREE	Графическое отображение структуры папок заданного диска или заданной папки
TYPE	Вывод на экран содержимого текстовых файлов
VERIFY	Установка режима проверки записи файлов на диск
VOL	Вывод метки и серийного номера тома для диска
XCOPY	Копирование файлов и дерева папок

Таблица 1.3 — Команды управления работой ОС

ASSOC	Вывод или изменение связи между расширениями имени и типами файлов
AT	Выполнение команд и запуск программ по расписанию
BAEAK	Включение/выключение режима обработки комбинации клавиш CTRL+C
CACLS	Отображение/редактирование списков управления доступом к файлам различных пользователей
CHCP	Просмотр номера текущей кодовой страницы или изменение текущей кодовой страницы консоли
CHKNTFS	Отображение или изменение выполнения проверки диска во время загрузки

Окончание табл. 1.3

CLS	Очистка экрана на консоли
CMD	Запуск еще одного интерпретатора командных строк
COLOR	Установка цвета текста и фона, используемых по умолчанию
COMPACT	Отображение/изменение сжатия файлов в разделах NTFS
CONVERT	Преобразование дисковых томов FAT в NTFS
DATE	Вывод либо установка текущей даты
DOSKEY	Редактирование и повторный вызов командных строк. Создание макросов
FTYPE	Вывод либо изменение типов файлов, используемых при сопоставлении по расширениям имен файлов
GRAFTABL	Позволяет отображать расширенный набор символов в графическом режиме
MODE	Конфигурирование системных устройств
MORE	Последовательный вывод данных по частям размером в один экран
PATH	Вывод либо установка пути поиска исполняемых файлов
PROMPT	Изменение приглашения в командной строке
RECOVER	Восстановление читаемой информации с плохого или поврежденного диска
SET	Вывод, установка и удаление переменных среды
START	Запуск программы или команды в отдельном окне
SUBST	Сопоставляет заданному пути имя диска
VER	Вывод сведений о версии операционной системы

Таблица 1.4 — Команды пакетных (командных) файлов

CALL	Вызов одного пакетного файла из другого
ECHO	Вывод сообщений и переключение режима отображения команд на экране
ENDLOCAL	Конец локальных изменений среды для пакетного файла
EXIT	Завершение работы программы
FOR	Организация циклов для обработки наборов файлов или строк в файле
GOTO	Передача управления в отмеченную строку пакетного файла
IF	Оператор условного выполнения команд в пакетном файле
PAUSE	Приостановка выполнения пакетного файла и вывод сообщения
REM	Помещение комментариев в пакетные файлы
SETLOCAL	Начало локальных изменений среды для пакетного файла
SHIFT	Изменение содержимого (сдвиг) замещаемых параметров для пакетного файла

Кроме команд, перечисленных в таблицах, имеется еще одна группа для работы в компьютерных сетях. Перечень этих команд может быть получен командой ***NET /?*** (рис. 1.3) (табл. 1.5).

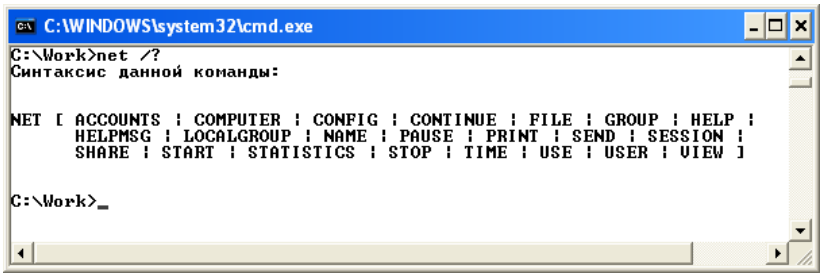

Рис. 1.3 — Вызов помощи по команде /?

Таблица 1.5 — Сетевые команды

NET ACCOUNTS	Обновление учетной базы пользователей, паролей и параметров подключения
NET COMPUTER	Добавление и удаление имени компьютера в базе данных домена
NET CONFIG	Сведения о настраиваемых службах и их изменение
NET CONTINUE	Активизация приостановленной службы, имя которой указано в качестве параметра
NET FILE	Вывод имен открытых файлов на сервере и количества их блокировок
NET GROUP	Вывод, добавление и изменение глобальных групп на сервере домена
NET HELPMSG	Выдача справок об ошибках и предупреждающих сообщениях
NET LOCALGROUP	Отображение и изменение локальных групп
NET NAME	Добавление и удаление имени, называемого псевдонимом. Псевдоним — имя компьютера, принимающего сообщения
NET PAUSE	Приостановка работы службы, указанной параметром в команде
NET PRINT	Отражение состояния, управление заданиями и очередями принтеров
NET SEND	Пересылка сообщения адресату: пользователю, компьютеру, псевдониму
NET SESSION	Вывод списка подключенных к компьютеру пользователей и его изменение
NET SHARE	Создание и удаление совместно используемых ресурсов сети
NET START	Вывод списка запущенных служб и его изменение
NET STATISTICS	Вывод содержимого журнала статистики для служб компьютера или сервера
NET STOP	Остановка работы службы, указанной параметром в команде
NET TIME	Синхронизация часов компьютеров, включенных в сеть
NET USE	Подключение компьютеров сети к сетевым ресурсам
NET USER	Добавление, редактирование и просмотр учетных сведений пользователей
NET VIEW	Просмотр списков компьютеров, доменов и общих ресурсов на указанном компьютере

Для вызова помощи для конкретных сетевых команд следует набирать net имя_команды /?.

Справочная информация по различным командам свидетельствует, что командой, набираемой в командной строке, является собственно имя команды, за которым могут следовать ключи (опции) — указания, модифицирующие поведение команды. Квадратные скобки в пояснениях обозначают, что эта информация не является обязательной при наборе команды. Ключи начинаются со знака / (слэша) и состоят из одного или нескольких символов. Кроме ключей, после команды могут следовать аргументы (параметры) — названия объектов, над которыми должна быть выполнена команда. Очень часто аргументами служат имена файлов и каталогов.

Ввод команды заканчивается нажатием клавиши <Enter>, после чего команда передается на исполнение командному процессору. В результате выполнения команды на экране дисплея могут появиться сообщения о ходе выполнения команды или об ошибках, а появление очередного приглашения (мигающего курсора) свидетельствует об успешном выполнении введенной команды и ожидании ввода следующей [1].

Замещаемые символы (метасимволы). Параметр в командной строке команды может включать замещаемые символы «?» и «*». Символ вопросительный знак заменяет один любой символ. Символ звездочка может заменять любую последовательность символов.

Пусть в текущем каталоге содержится произвольный набор файлов. Команда DIR без параметров по умолчанию покажет нам весь перечень файлов в директории (рис. 1.4).

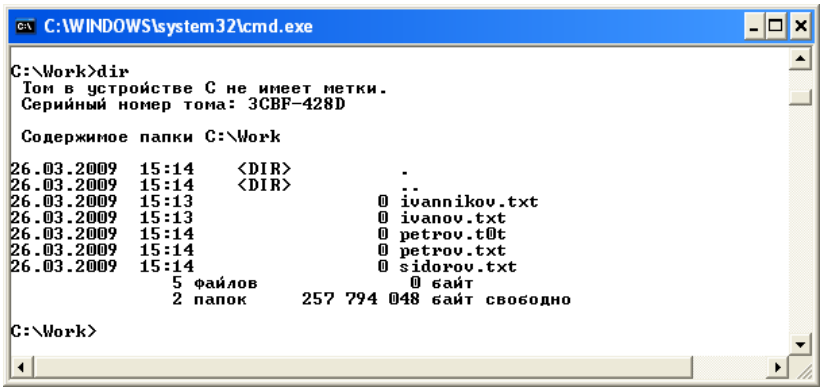

Рис. 1.4 — Результат выполнения команды DIR

Применим для вывода команды замещаемые параметры. Результат выполнения команды DIR *.TXT будет следующим (рис. 1.5). Строка с файлом PETROV.T0T не будет отображена, так как мы указали команде DIR показать все файлы с любым именем (символ *), но имеющим только расширение .TXT.

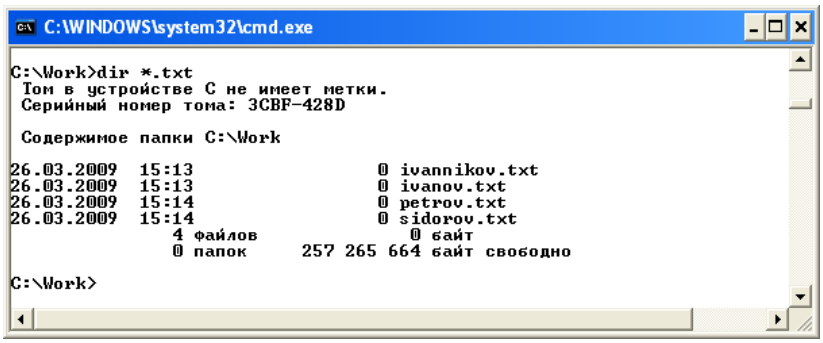

Рис. 1.5 — Результат выполнения команды DIR *.TXT

Выполним последовательно команды:

1. DIR PETROV.T* (рис. 1.6).
2. DIR PETROV.T? (рис. 1.7).
3. DIR PETROV.T?? (рис. 1.8).

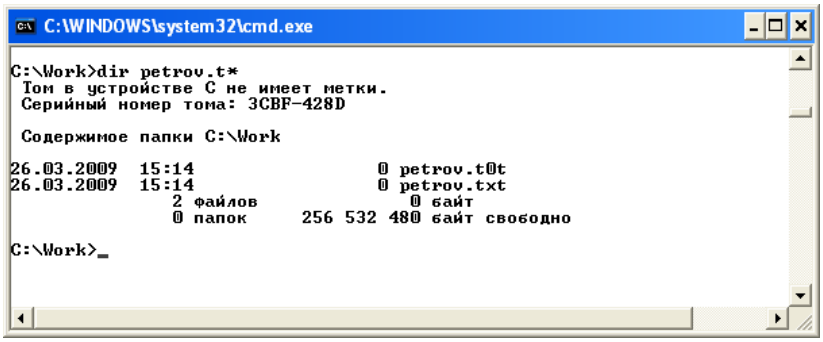

Рис. 1.6 — Результат выполнения команды DIR PETROV.T*

Первая команда говорит, что необходимо показать файлы, у которых имя «petrov», расширение начинается с символа «t», далее могут идти любые символы.

Вторая команда говорит, что необходимо показать файлы, у которых имя «petrov», расширение начинается с символа «t», далее может быть только один любой символ.

Третья команда говорит, что необходимо показать файлы, у которых имя «petrov», расширение начинается с символа «t», далее может быть только два любых символа.

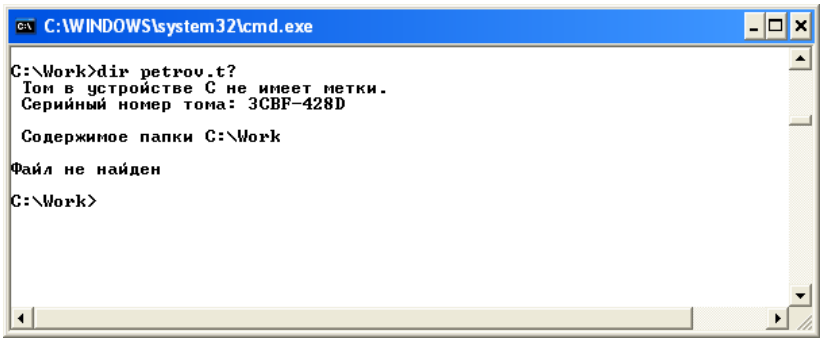

Рис. 1.7 — Результат выполнения команды DIR PETROV.T?

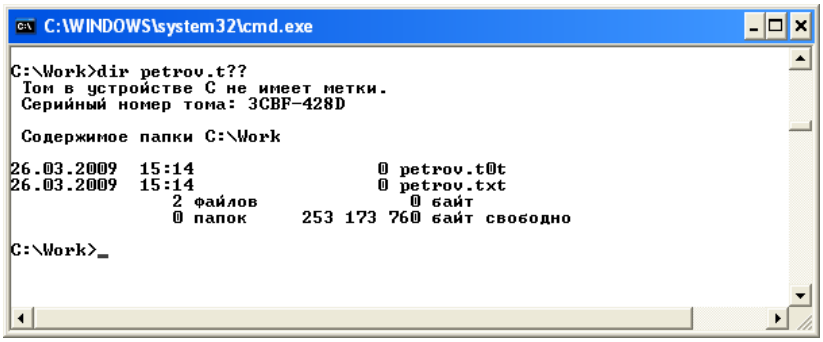

Рис. 1.8 — Результат выполнения команды DIR PETROV.T??

Как видно из рисунков, результаты выполнения первой и третьей команд совпали, так как расширение состоит из трех символов.

А результат выполнения второй команды не вывел ни одного файла, удовлетворяющего заданному условию.

Стандартные потоки ввода-вывода и перенаправление потоков. Термин CONsole используется для обозначения стандартных потоков ввода-вывода. Когда говорят о вводе с консоли, подразумевается ввод с клавиатуры. Когда говорят о выводе на консоль, подразумевают вывод на экран монитора. Существуют специальные символы для перенаправления стандартных потоков ввода-вывода:

> *приемник* — перенаправить стандартный вывод в приемник (если файл-приемник существует, то он будет создан заново);

>> *приемник* — перенаправить стандартный вывод в приемник (если файл-приемник существует, то он будет сохранен, а информация будет записана в конец файла);

< *источник* — перенаправить стандартный ввод из источника;

передатчик* | *приемник — передает вывод одной команды на вход другой.

Приведем несколько примеров с использованием перенаправления потоков:

DIR > FILES.TXT — содержимое текущего каталога записать в текстовый файл;

TYPE FILE.TXT >> ARXIV.TXT — добавить в конец файла ARXIV.TXT содержимое файла FILE.TXT;

DATE < DATE.TXT — установить новую системную дату, значение взять из файла DATE.TXT;

TYPE PETROV.TXT | SORT — распечатывает на экране файл petrov.txt, отсортировав его строки.

Возможна комбинация символов перенаправления потоков:

TYPE PETROV.TXT | SORT > PETROV_SORT.TXT — записывает в файл PETROV_SORT.TXT содержимое файла PETROV.TXT, отсортировав его строки.

Создание текстовых файлов можно выполнить следующей командой:

COPY CON ДИСК:ПУТЬ\ИМЯ_ФАЙЛА.РАСШ

Например, для создания файла F1.TXT в текущем каталоге необходимо выполнить команду COPY CON F1.TXT.

Этой командой ввод текста с клавиатуры (консоли con) осуществляется в новый, создаваемый этой операцией файл F1.TXT. После набора каждой строки следует нажимать клавишу <Enter>. Окончание набора файла должно заканчиваться нажатием клавиш <Ctrl>+Z или F6 (признаком окончания файла), а затем <Enter>. Недостатком применения этой команды является то, что редактировать можно только текущую строку файла. После нажатия клавиши <Enter> ранее введенные строки уже недоступны. Таким образом, команда copy использует простейший однострочный редактор.

Атрибуты файлов. Каждый файл и каталог, находящиеся в компьютере, могут иметь атрибуты — характеристики, отражающие свойства объ-

екта, которые используются операционной системой для корректной работы с ними. Атрибутами файла (файлов) могут быть следующие значения:

R — «только для чтения», то есть нельзя модифицировать файл и уничтожить его;

A — «архивный», т. е. +A обозначает, что снимались копии данного файла; соответственно, –A — файл является вновь созданным;

H — «скрытый», скрытые файлы не показываются командой DIR и некоторыми программными оболочками;

S — «системный», этот атрибут показывает, что файл является принадлежностью операционной системы.

Изменение атрибутов файлов осуществляется командой ATTRIB. Формат команды

ATTRIB +|–АТРИБУТ ДИСК:\МАРШРУТ\ИМЯ ФАЙЛА /S

Установка любого атрибута производится знаком + (плюс), отмена — знаком – (минус). Можно задавать изменение сразу нескольких атрибутов в любой последовательности. Ключ /S, стоящий в конце формата, указывает, что процесс изменения атрибутов файла (файлов) распространяется не только на текущий каталог, но и на все каталоги, подчиненные текущему.

Примеры:

ATTRIB +A +H +R PRIMER.TXT — присваивает файлу сразу три атрибута: только для чтения, архивный и скрытый файлу PRIMER.TXT.

ATTRIB –R A:*.* /S — с использованием шаблона снимает атрибут «только для чтению» у всех файлов диска A, т. е. находящихся на дискете.

Разработка командных файлов. Командный файл — это группа последовательных команд настройки компьютера на определенный режим или выполнение определенных операций. В простейшем случае командный файл может быть представлен в виде определенной последовательности отдельных команд операционной системы. Разработка командных

файлов является мощным средством автоматизации подготовительных работ пользователей по настройке среды их работы [1].

При разработке командных файлов следует руководствоваться следующими правилами [1]:

1. Вызов на исполнение командного файла осуществляется командой следующего формата:

ДИСК:\ПОЛНЫЙ_ПУТЬ\ИМЯ_КОМ.ФАЙЛА [P1 P2 ... P10]

Содержимое в квадратных скобках указывает, что командный файл может иметь до 10 фактических параметров, замещающих формальные параметры, присутствующие в тексте файла. Команда SHIFT позволяет снять это ограничение.

2. Имя командного файла образуется по обычным правилам. Расширением должно быть только сочетание BAT или CMD.

3. Если текущим является каталог (папка), содержащий командный файл, то полный путь к командному файлу можно не указывать.

4. Командный файл выполняется командным процессором строка за строкой.

5. Выполнение командного файла может быть прекращено командами <Ctrl>+<Break> или <Ctrl>+C.

6. Из командного файла можно вызывать другой командный файл командой CALL (с возвратом) или обычной командой вызова (без возврата).

7. Командный файл может содержать любые внешние и внутренние команды операционной системы, а также специальные внутренние команды.

8. Формальные параметры, включаемые в строки командного файла, имеют вид %0, %1 и т. д. до %9. Фактические значения параметров вводятся в строке вызова командного файла; вводимые параметры подставляются на место формальных параметров %1, %2 и т. д. по порядку. На место

формального параметра %0, если он встречается в тексте командного файла, подставляется имя самого командного файла.

9. Для обращения к переменным окружения их имена следует заключать в знаки %, например %TEXT%.

10. Перед выполнением очередной строки командного файла ее значение выводится на экран. Вывод любой строки командного файла на экран подавляется, если строка начинается с символа @.

Рассмотрим ***особенности применения специальных команд***:

Команда ECHO предназначается для отключения «эха» на экране дисплея, то есть она не позволяет выводить лишнюю информацию на монитор (блокирует выдачу на экран последовательностей команд, включенных в командный файл, и текстовых сообщений при выполнении этих команд). Форматы команды:

ECHO OFF — запрет вывода на экран;

ECHO ON — разрешение вывода на экран;

ECHO (без параметров) — запрос состояния эха (ON или OFF);

ECHO + текстовое сообщение — вывод текстового сообщения на экран;

ECHO %имя переменной окружения% — вывод текущего значения переменной окружения;

ECHO. (с точкой) — вывод пустой строки.

При использовании команды ECHO следует помнить:

– при запуске системы по умолчанию устанавливается режим «ECHO ON»;

– режим «ECHO OFF» действует только до конца командного файла или до очередного переключения режима командой «ECHO ON»;

– ECHO влияет только на вывод сообщений командного файла, но не влияет на вывод сообщений из программ пользователей, даже если они используют команды операционной системы;

– для подавления самой команды «ECHO OFF» надо поставить впереди знак @.

Для лучшего понимания содержимого командного файла используются комментарии, вводимые с помощью ***команды REM*** (remark — примечание). Командный процессор полностью игнорирует всю информацию, которая размещается за словом REM. Команда очень полезна, когда в командный файл включаются пояснения, описания работы файла или отдельных его команд, тестирования и отладки.

Для приостановки выполнения командного файла используется ***команда PAUSE***. Команда имеет формат:

PAUSE сообщение

При остановке работы командного файла на экране появляется текст строки сообщения в режиме ECHO ON, а под ним фраза «Press any key to continue» — Нажмите любую клавишу для продолжения (для продолжения работы файла).

Команду полезно использовать в тех случаях, когда, например, на экран дисплея выводится большое количество информации порциями по страницам, чтобы пользователь мог ее прочитать, осмыслить и перейти к следующему фрагменту. Команда полезна и в случаях, когда необходимо выполнить какие-то вспомогательные действия, например:

```
@ECHO ON
PAUSE Установите дискету с .... на дисковод A:
@ECHO OFF
```

Кроме того, команду PAUSE можно использовать и для управления работой командного файла. Если в ответ на команду PAUSE нажать <Ctrl>+C, то появляется вопрос «Terminate batch job (Y/N)?» — Завершить выполнение задания (командного файла)? Выбор Y — останавливает выполнение командного файла, а N — обеспечивает продолжение его выполнения. В случаях когда командный файл выполняется с частыми остановками и появление множества фраз «Press any key to continue» нежелательно, строка с командой может выглядеть как PAUSE> NUL, то есть вывод переадресуется в несуществующее устройство nul.

Команда GOTO позволяет изменить привычную последовательность выполнения операторов (команд) командного файла. Когда командный процессор встречает строку с оператором GOTO, то он просматривает все строки файла и отыскивает соответствующую метку — строку с двоеточием. Двоеточие может быть и не в первой позиции строки. Идентификатор метки должен иметь до восьми символов. Больше, чем восемь символов, в идентификаторе не воспринимается. Команда GOTO может использоваться самостоятельно или совместно с операцией IF.

Команда IF — условное выполнение команд, организует разветвление при выполнении командного файла. Формат оператора IF

IF условие команда.

В качестве условия обычно используются:

– проверка наличия файла. В этом случае в качестве условия записывается фраза exist диск:путь\имя_файла.расш;

– проверка кода завершения отдельных программ по значению внутренней переменной системы с именем ERRORLEVEL. В этом случае в качестве условия записывается фраза «ERRORLEVEL значение». Условие

считается истинным, если код завершения равен или больше параметра значение. Значение переменной errorlevel может формироваться многими утилитами и прикладными программами;

– проверка идентичности двух символьных строк. Строка условие при этом записывается в виде

строка_1==строка_2 (двойной знак =)

Предваряя любому из перечисленных условий слово NOT, можно проверять противоположное условие.

Для многократного выполнения отдельных команд применяется команда FOR. Она позволяет обрабатывать целые группы файлов. Команда имеет следующие форматы:

FOR %%переменная IN (набор) DO команда — для строк командных файлов;

FOR %переменная IN (набор) DO команда — для режима командной строки (автономного выполнения команды).

В качестве параметров команды используются:

– переменная — однобуквенная переменная, последовательно принимающая значения слов или имен файлов, перечисленных в параметре (набор);

– (набор) — одно или несколько символьных слов или спецификаций файлов. Спецификация файла имеет вид диск:путь\имя_файла.расш. Допускаются шаблоны групповых операций. Слова и спецификации файлов разделяются пробелами или запятыми. Максимальная длина строки набора — не более 127 символов;

– команда — команда DOS, выполняемая для каждого слова или файла из параметра набор.

Обычно в командный файл можно передавать до 9 параметров, иногда этого недостаточно. ***Команда SHIFT*** (сдвиг) позволяет сдвигать строку параметров влево на один параметр. Применение данной команды позволяет снять ограничение на число параметров.

В учебном пособии приведены лишь фрагменты описания команды, более подробную информацию по командам лучше смотреть непосредственно, вызывая помощь в командной строке операционной системы.

Команда CHOICE — ожидает ответа пользователя. Данная команда является внешней, то есть необходимо наличие файла choice.exe. Данный файл должен располагаться в текущей папке или в какой-либо системной, чтобы операционная система могла найти его. Формат команды выглядит следующим образом:

CHOICE [/C[:]варианты] [/N] [/S] [/T[:]c,nn] [текст]

/C[:]варианты — варианты ответа пользователя.

По умолчанию строка включает два варианта: YN

/N Ни сами варианты, ни знак вопроса в строке приглашения не отображаются.

/S Учитывать регистр символов.

/T[:]c,nn Ответ «c» выбирается автоматически после nn секунд ожидания текста Строка приглашения

После выполнения команды переменная ERRORLEVEL приобретает значение, равное номеру выбранного варианта ответа.

Приведем ***несколько примеров командных файлов***.

Пример 1 [1]. Пусть требуется создать командный файл test1.bat, который будет копировать из текущего каталога на дискету ряд текстовых файлов с проверкой правильности записи и удалением исходных файлов.

Перед каждым удалением файла должно выдаваться предупреждающее сообщение. В момент приостановки можно прервать дальнейшее выполнение командного файла, нажав клавиши <Ctrl>+C.

```
:LOOP
COPY %1.TXT A:/V
PAUSE УДАЛЯЮ СКОПИРОВАННЫЙ ФАЙЛ
DEL %1.TXT
SHIFT
IF NOT %1.==. GOTO LOOP
```

Запуск этого файла следует выполнять командой

Test1.bat 01 02 03 04 05 06 07 08 09 10 11 12 и т. д., если текстовые файлы имеют имена 01.txt, 02.txt, 03.txt и т. д. Обратите внимание, что:

– по умолчанию здесь используется режим ECHO ON. В противном случае сообщения команды PAUSE были бы не видны;

– расширения текстовых файлов присоединяются к имени непосредственно в командах выполняемого файла.

Пример 2 [1]. Создадим файл test2.bat таким образом, чтобы можно было отыскивать и просматривать нужный файл в любом каталоге. Учитывая, что файлы могут иметь большие размеры, превышающие емкость одного экрана, обеспечим поэкранный просмотр файлов. Имя нужного файла будем задавать в качестве параметра в строке вызова файла test2.bat. Например: test2.bat proba.txt

Здесь имя искомого файла proba.txt служит фактическим параметром, значение которого должно заменить формальные параметры %1 внут-

ри командного файла. Таких параметров строка вызова может иметь от %1 до %9. Командный файл test2.bat будет иметь следующее содержание.

```
ECHO OFF
CLS
IF /%1==/ GOTO ERROR1
IF NOT EXIST %1 GOTO ERROR2
TYPE %1 | MORE
GOTO END
:ERROR1
ECHO ВЫ забыли указать имя искомого файла!
:GOTO END
:ERROR2
ECHO ФАЙЛА %1 на этом диске нет!
:END
```

В этом фрагменте два слэша / в операторе IF играют роль скобок.

Пример 3. Командный файл просит ввести цифру, соответствующую имени пользователя, и в зависимости от того, какая цифра была введена, устанавливает текущим тот или иной рабочий каталог и открывает окно CMD.

```
:BEGIN
    ECHO Введите номер пользователя
    ECHO 1 – Алексей 2 – Петр 3 – Иван 4 – остальные
    CHOICE /C:1234
    IF ERRORLEVEL 4 GOTO WORK
    IF ERRORLEVEL 3 GOTO IVAN
```

```
        IF ERRORLEVEL 2 GOTO PETER
        IF ERRORLEVEL 1 GOTO ALEX
        GOTO BEGIN
:IVAN
        CD IVAN
        START
:PETER
        CD PETER
        START
:ALEX
        CD ALEX
        START
:WORK
        CD WORK
```

1.2 Интерфейс командной строки ОС Unix

Большинство версий операционной системы Unix имеют графический интерфейс, подобный интерфейсу, используемому на компьютерах Macintosh и впоследствии в IBM-совместимых компьютерах с операционной системой Windows. Однако интерфейс командной строки до сих пор остается популярным среди программистов.

Изучение команд Unix-подобных операционных систем можно вести, установив систему на жесткий диск своего компьютера или загрузив Unix (Linux) с компакт-диска (CD) без инсталляции на жесткий диск. Удобно воспользоваться эмулятором Unix для операционной системы Windows. Эмулятор можно легко найти в Интернете (любая поисковая система находит различные эмуляторы по поисковому запросу: «эмулятор Unix»).

Приглашение к вводу команды в Unix может выглядеть по-разному: # — это приглашение для суперпользователя (root), вошедшего в систему; $ или [имя@localhost имя]$ — для обычных пользователей. Помощь по командам Unix можно получить, набрав man и через пробел — имя команды, например: man gnome.

Вход в систему производится в диалоге, когда система запрашивает имя пользователя и его пароль.

Выход из системы может производиться по-разному. Для выхода из системы служит команда logout, по которой прекращается сеанс работы с данным пользователем, но система не завершает свою работу. Прекратить текущий сеанс работы можно также, нажав одновременно три клавиши Ctrl+Alt+Backspace.

Для полного завершения работы нажимается Ctrl+Alt+Del.

Перечень команд. В состав Unix входит более трехсот команд. Данная операционная система обладает более богатыми методами управления ресурсами, чем ОС Windows.

При изучении системы команд Unix необходимо иметь в виду, что многие команды операционных систем MS DOS и Windows совпадают по имени и, частично, по функциям с командами Unix (табл. 1.6).

Таблица 1.6 — Список команд Unix

at	Выполнение команд и запуск программ по расписанию
cat	Вывод на экран содержимого файлов
cd	Выбор имени либо смена текущей папки
chmod	Изменение атрибутов прав доступа к файлам
cmp	Поиск различий между файлами (до первого различия)
cp	Копирование одного или нескольких файлов в другое место
date	Вывод текущей даты
df	Определение свободного пространства диска
diff	Поиск всех различий в файлах
du	Определение занятого пространства диска
ed	Вызов текстового редактора
exit	Выход из системы

Окончание табл. 1.6

kill	Послать сигнал процессу. Завершить процесс.
ls	Вывод списка файлов в каталоге
mail	Вызов почтового клиента
man	Вызов помощи по командам
mesg	Установка режима запрета или разрешения сообщений
mkdir	Создание каталога
mv	Перенос файлов
news	Запуск клиента новостей
nice	Запуск программы с пониженным приоритетом
nohup	Выполнение программы после отключения терминала
pr	Распечатка файла по 66 строк
ps	Получение списка процессов
pwd	Определение своего рабочего каталога
rm	Удаление файла
rmdir	Удаление каталога
sh	Переход в порожденный командный процессор (shell)
tail	Вывод на экран содержимого файлов с конца
time	Информация о времени выполнения команды
touch	Заменяет время модификации файла на настоящее
uname	Информация о системе
wc	Подсчет числа строк, слов и символов в файле
who	Вывод активных пользователей
who am i	Вывод собственного имени
write	Установка связи с другим пользователем

Файлы и процессы, являются центральными понятиями операционной системы UNIX. Файловая подсистема управляет файлами, размещает записи файлов в отведенные для них места, управляет свободным пространством, доступом к файлам и поиском данных для пользователей.

Работа с файлами ведется с помощью команд. Команда представляет собой имя исполняемого файла (двоичного или текстового, так называемого скрипта, написанного на одном из специальных командных языков) или имя внутренней команды самого процессора. При активизации каждой такой команды операционная система создает процесс. Процессы взаимодействуют с подсистемой управления файлами и с аппаратными средствами, используя для этого совокупность специальных команд, таких как ***open***

(для того, чтобы открыть файл на чтение или запись), ***close***, ***read***, ***write***, ***stat*** (запросить атрибуты файла).

Подсистема управления процессами ядра ОС отвечает за синхронизацию процессов, их взаимодействие, распределение памяти и планирование выполнения процессов, По характеру выполнения процессы могут быть фоновыми и привилегированными. Любой запускаемый процесс по умолчанию будет выполняться как привилегированный (***foreground***). Это значит, что такой процесс постоянно связан с терминалом ЭВМ и делает невозможным выполнение еще каких-либо действий с системой, пока не завершится.

Фоновый процесс (***background***) после запуска освобождает терминал и позволяет перейти к другой задаче, не дожидаясь его завершения. Фоновая обработка наиболее пригодна для процессов, которые долго выполняются. Программы, выполняющиеся в виде фоновых процессов, называются демонами (***daemon***). В любой момент времени в системе существуют десятки процессов, которые были запущены при старте операционной системы, вызваны ядром для обслуживания каких-либо событий, добавлены пользователем при запуске какой-либо задачи.

Обычно большинство процессов находится в состоянии ожидания — сна (sleep), не мешая остальным и дожидаясь сигнала для активизации. Кроме того, в системе можно найти процессы, закончившие работу, но еще не получившие разрешения на выгрузку из основной памяти, — эти процессы называются зомби. В ядре операционной системы находится таблица процессов, каждая запись которой описывает состояние одного из процессов.

Основное отличие **файловой системы Unix** от файловой системы Windows заключается в том, что в Unix отсутствует такое понятие, как логическое устройство. При указании пути к файлу в Unix имя устройства не

упоминается. Дерево каталогов Unix «растет» из одного корня. Корневой каталог имеет предопределенное имя / (слэш). Этот же символ применяется и для разделения подкаталогов. Полный путь к файлу в Unix выглядит следующим образом:

/каталог1/каталог2/каталог3/ … /файл.

Физически разные компоненты дерева каталогов Unix могут размещаться на разных дисках, но логически они принадлежат одной древовидной структуре с одним корневым узлом. Для объединения файловых систем различных устройств в одну структуру используется операция монтирования.

Сущность этой операции заключается в том, что каждое физическое устройство можно рассматривать, как свою собственную файловую систему (файловую систему устройства) с корневым каталогом /. Если этот раздел диска объявлен в операционной системе как корневой раздел (root), его каталог становится корневым каталогом всей файловой системы (файловой системы ЭВМ). Файловые системы остальных устройств должны быть смонтированы в каталогах файловой системы ЭВМ.

Операция монтирования связывает корневой каталог монтируемого раздела (устройства) с выбранным каталогом файловой системы ЭВМ — точкой монтирования. В результате монтирования корневой каталог файловой системы устройства получает имя каталога, являющегося точкой монтирования, благодаря чему файловая система устройства «привязывается» к файловой системе ЭВМ в точке монтирования.

Таким образом, для монтирования файловой системы устройства в файловой системе ЭВМ необходимо сначала в файловой системе ЭВМ создать каталог, который будет точкой монтирования, а затем соединить две файловые системы командой ***mount.***

После монтирования сменных носителей их нельзя извлекать из устройства без демонтирования файловой системы. Для этой цели служит команда ***umount.***

При просмотре содержимого каталога можно увидеть, что информация о файле начинается с кода, содержащего 10 символов:

-rwxrwxrwx, или **drwxrwxrwx**, и **lrwxrwxrwx.**

Первый символ кода указывает тип файла:

– ***символ —*** означает, что это обычный файл, текстовый или двоичный, содержащий данные или программу;

– ***символ d*** указыет, что данный файл является каталогом;

– ***символ l*** указывает на то, что данный файл является символьной ссылкой.

Символы ***rwx*** определяют права доступа к файлу. Доступ к файлу могут иметь три категории пользователей:

– владелец файла;

– выделенная группа пользователей;

– остальные пользователи (не являющиеся владельцами файла и не входящие в выделенную группу).

Всем им могут быть установлены следующие права:

– ***символ r*** — разрешено чтение файла;

– ***символ w*** — разрешена запись в файл;

– ***символ x*** — разрешен запуск файла на исполнение.

Эти права всегда перечисляются подряд в порядке: ***rwx***. Если какое-либо право не предоставлено, вместо соответствующего символа ставится -. Например, ***r - -*** означает, что разрешено только чтение; ***-wx*** означает, что разрешены запись в файл и его исполнение.

Поскольку права определяются для трех видов пользователей, указанная триада повторяется трижды и образует запись из 9 символов, первые три из которых относятся к владельцу, вторые — к группе и третьи — к остальным пользователям. Например, запись rwx--xr-- означает, что владельцу файла разрешено все, выделенной группе — только запуск на исполнение, остальным пользователям — только чтение.

Иногда указанные девять символов кодируются числом. Ключ к расшифровке цифрового кода приведен в виде таблицы 1.7.

Таблица 1.7 — Ключ расшифровки цифрового кода прав доступа

4	2	1	4	2	1	4	2	1
r	w	x	r	w	x	r	w	x
1 цифра			2 цифра			3 цифра		
4+2+1=7			4+2+1=7			4+2+1=7		

Установка и изменение режима доступа к файлу производятся с помощью команды ***chmod***.

Формат команды chmod (change mode) для установки режима: ***chmod <режим> <файлы>***

Пример использования команды: $ chmod 644 f1 f2 f3,

где 644 соответствует rw-r--r--

Формат команды chmod для изменения режима:

chmod <изменения> <файлы>

В изменениях используются обозначения:

– r — read — права на чтение;

– w — write — права на запись;

– x — execute — права на выполнение;

– u — user — права для владельца файла;

– g — group — права для группы пользователей;

– o — other — права для остальных пользователей;

– a — all — прва для всех пользователей;

– = — назначить права;

– + — добавить права;

– - — отнять права.

Пример использования команды:

```
$ ls -l
-r-------- ... f1
-r-------- ... f2
-r-------- ... f3
$ chmod  a = r, u + w    f1 f2 f3
```

или (эквивалентный вариант изменения прав доступа)

```
$ chmod  u = rw, go = r  f1 f2 f3
$ ls -l
-rw-r--r-- ... f1
-rw-r--r-- ... f2
-rw-r--r-- ... f3
$ chmod o-r f1 f2 f3
$ ls -l
-rw-r----- ... f1
-rw-r----- ... f2
-rw-r----- ... f3
```

Другие пользователи, не входящие в группу, потеряли право читать файлы.

В Unix операционных системах есть также возможность использовать **метасимволы**. Метасимволы служат для подстановки любых строк и символов. В именах файлов в командах языка заданий Shell:

– * — представляет произвольную строку (возможно, пустую);

– ? — любой одиночный знак;

– [C1 - C2] — любая литера из диапазона C1–C2 (в стандарте ASCII).

Примеры:

1) $ ls c?

c1 c2 c3 cs cz

2) $ ls c*

c1 c12 c2 c23 c3 cs cs1 cxy cz

3) $ ls ?1*

c1 c12

4) $ ls *1*

c1 c12 cs1

5) $ ls c [12 x y z]

c1 c2 cz

6) ls c [12 x y z *]

c1 c2 c12 c25 cz cxy

Стандартные файлы. Многие команды работают по умолчанию со стандартными файлами:

– Standard Input (S.I.) — стандартный ввод;

– Standard Output (S.O.) — стандартный вывод;

– Diagnostic Output (D.O.) — диагностический вывод.

Однако есть средства изменения умолчания, т. е. возможность указать другие файлы вместо стандартных. Можно также в качестве диагностического вывода использовать стандартный вывод. Эти средства называются перенаправлением (редирекцией) ввода и вывода.

Примеры:

1. Перенаправления стандартного ввода:

$ cat < this_file

2. Одновременные перенаправления ввода и вывода:

$ cat < left > right

3. Перенаправления стандартного вывода:

$cat > newfile

4. Соединение команд каналами (***pipeline***)

$ who | wc –l — создание списка активных пользователей и подсчет их числа (count); 19 — ответ, то есть 19 пользователей.

$ ls -l /tmp | grep vladimir | sort +3nr | lpr

листинг каталога /tmp	поиск записей, содержащих строку vladimir	сортировка (по 4-му полю) найденных записей	печать упорядоченного списка

5. Одновременный стандартный вывод и перенаправление вывода

$ ls -l | ***tee*** dirconts — команда одновременно выводит содержимое текущего каталога на экран и в файл dirconts.

Сообщения об ошибках, возникающих при выполнении команд, выводятся на диагностический вывод, по умолчанию это (как и стандартный вывод) — на экран.

Диагностический вывод тоже может быть перенаправлен в любой файл. Для этого используется дескриптор файла (целое), который для стандартных файлов равен:

0 — Standard input;

1 — Standard output;

2 — Diagnostic output.

Если вы хотите, чтобы сообщения об ошибках нигде не проявлялись, направьте их на /dev/null.

Пример:

$ cat somefile > outfile 2> errfile,

где знак > эквивалентен 1>.

Разработка командных файлов. Для того чтобы текстовый файл можно было использовать как командный, существует несколько возможностей. Можно вызвать оболочку ***shell*** (интерпретатор команд, подаваемых с терминала или из командного файла, — это обычная программа, которая не входит в ядро операционной системы UNIX) как команду, обозначаемую ***sh***, и передать ей файл ***f1*** как аргумент или как перенаправленный вход: $ sh fl или $ sh < f1.

Файл можно выполнить и в текущем экземпляре shell. Для этого существует специфическая команда. (точка). Пример: ***.f1***

Еще один способ, это сделать текстовый файл исполняемым с помощью команды chmod. Пример: chmod 711 f1.

Shell имеет в своем составе функциональные возможности, благодаря которым его можно смело назвать языком программирования. К этим функциональным возможностям относятся:

– переменные;

– управляющие структуры (типа if);

– подпрограммы (в том числе командные файлы);

– передача параметров;

– обработка прерываний.

Переменные Shell

В языке Shell версии 7 определение переменной содержит имя и значение: var = value.

Доступ к переменной — по имени со знаком $ спереди:

fruit = apple (определение);

echo $fruit (доступ);

apple (результат echo).

Таким образом, переменная — это строка. Возможна конкатенация строк:

```
$ fruit = apple
$ fruit = pine$fruit
$ echo $fruit
pineapple
$ fruite = apple
$ wine = ${fruite}jack
$ echo $wine
applejack
$
```

Другие способы установки значения переменной — ввод из файла или вывод из команды, а также присваивание значений переменной — параметру цикла for из списка значений, заданного явно или по умолчанию.

Переменная может быть:

– частью полного имени файла: $d/filename, где $d — переменная (например, d = /usr/bin);

– частью команды:

$ S = "sort + 2n + 1 - 2" (наличие пробелов требует кавычек "")
$ $S tennis/lpr
$ $S basketball/lpr
$ $S pingpong/lpr
$

Однако внутри значения для команды не могут быть символы |, >, <, & (обозначающие канал, перенаправления и фоновый режим).

Предопределенные переменные Shell. Некоторые из них можно только читать. Наиболее употребительные:

HOME — «домашний» каталог пользователя; служит аргументом по умолчанию для cd;

PATH — множество каталогов, в которых UNIX ищет команды;

Изменение PATH:

$ echo $PATH	- посмотреть;
:/bin:/usr/bin	- значение PATH;
$ cd	- «домой»;
$ mkdir bin	- новый каталог;
$ echo $HOME	- посмотреть;
/users/maryann	- текущий каталог;
$ PATH = :$HOME/bin:$PATH	- изменение PATH;
$ echo $PATH	- посмотреть;
:/users/maryann/bin:/bin:/usr/bin	- новое значение PATH.

Пример 1 (установка переменной Shell выводом из команды):

$ now = `date` (где `` - обратные кавычки)

$ echo $now

Sun Feb 14 12:00:01 PST 1985

$

Пример 2 (получение значения переменной из файла):

$ menu = `cat food`

$ echo $menu

apples cheddar chardonnay (символы возврата каретки заменяются на пробелы).

Переменные Shell — аргументы процедур

Это особый тип переменных, именуемых цифрами.

Пример:

$ dothis grapes apples pears (процедура).

Тогда позиционные параметры (аргументы) этой команды доступны по именам:

$1 = `grapes`

$2 = `apples`

$3 = `pears`

и т. д. до $9. Однако здесь также есть команда ***shift***, которая сдвигает имена на остальные аргументы, если их больше 9 (окно шириной 9).

Другой способ получить все аргументы (даже если их больше 9): $*, что эквивалентно $1$2... Количество аргументов присваивается другой переменной: $# (диез). Наконец, имя процедуры — это $0; переменная $0 не учитывается при подсчете $#.

Структурные операторы Shell

Оператор цикла **for**

Пусть имеется командный файл makelist (процедура)

```
$ cat makelist
sort +1 -2 people | tr -d -9 | pr -h Distribution | lpr.
```

Если вместо одного файла people имеется несколько, например: adminpeople, hardpeople, softpeople,..., то необходимо повторить выполнение процедуры с различными файлами. Это возможно с помощью for-оператора. Синтаксис:

```
for <переменная> in <список значений>
do <список команд>
done
```

Ключевые слова for, do, done пишутся с начала строки.

Пример (изменим процедуру makelist):

```
for file in adminpeople, hardpeople, softpeople
do
    Sort +1 -2 $file | tr ... | lpr
done.
```

Можно использовать метасимволы Shell в списке значений.

Пример:

```
for file in *people (для всех имен, кончающихся на people)
do
  ...
done.
```

Если in опущено, то по умолчанию в качестве списка значений берется список аргументов процедуры, в которой содержится цикл, а если цикл не в процедуре, то список параметров командной строки (то есть в качестве процедуры выступает команда).

Пример:

```
for file
do
    ...
done
```

Для вызова makelist adminpeople hardpeople softpeople будет сделано то же самое.

Условный оператор ***if***

Используем имена переменных, представляющие значения параметров процедуры:

```
sort +1 -2 $1 | tr ... | lpr
```

Пример неверного вызова:

makelist (без параметров), где $1 неопределен. Исправить ошибку можно, проверяя количество аргументов — значение переменной $# посредством if-оператора.

Пример: (измененной процедуры makelist):

```
if test $# -eq 0
then
    echo "Вы должны указать имя файла"
    exit 1
else
    sort +1 -2 $1 | tr ... | lpr
fi
```

Здесь ***test*** и ***exit*** — команды проверки и выхода. Таким образом, синтаксис оператора if:

```
 if <если эта команда выполняется успешно, то>;
 then <выполнить все следующие команды до else или, если его нет, до fi>;
```

Ключевые слова ***if***, ***then***, ***else*** и ***fi*** пишутся с начала строки.

Успешное выполнение процедуры означает, что она возвращает значение true = 0 (zero) (неуспех — возвращаемое значение не равно 0).

Оператор exit 1 задает возвращаемое значение 1 для неудачного выполнения makelist и завершает процедуру.

Возможны вложенные if. Для else if есть сокращение ***elif***, которое одновременно сокращает fi.

Команда ***test***

Не является частью Shell, но применяется внутри Shell-процедур.

Имеется три типа проверок:

– оценка числовых значений;

– оценка типа файла;

– оценка строк.

Для каждого типа свои примитивы (операции op).

1. Для чисел синтаксис такой: N op M, где N, M — числа или числовые переменные;

op принимает значения: -eq, -ne, gt, -lt, -ge, -le.

2. Для файла синтаксис такой: op filename, где op принимает значения:

– -s (файл существует и не пуст);

– -f (файл, а не каталог);

– -d (файл-директория (каталог);

– -w (файл для записи);

– -r (файл для чтения).

3. Для строк синтаксис такой: S op R, где S, R — строки или строковые переменные или op1 S, где op принимает значения:

– = (эквивалентность);

– != (не эквивалентность);

– op1 принимает значения:

– -z (строка нулевой длины);

– -n (ненулевая длина строки).

Несколько проверок разных типов могут быть объединены логическими операциями -a (AND) и -o (OR).

Примеры:

```
$ if test -w $2 -a -r S1
> then cat $1 >> $2
> else echo "невозможно добавить"
> fi
$
```

В некоторых вариантах ОС UNIX вместо команды ***test*** используются квадратные скобки, т. е. if [...] вместо if test ... .

Оператор цикла ***while***

Синтаксис:

```
while <команда>
do
<команды>
done
```

Если «команда» выполняется успешно, то выполнить «команды», завершаемые ключевым словом done.

Пример:

```
if test $# -eq 0
then
        echo "Usage: $0 file ..." > &2
        exit
fi
```

```
while test $# -gt 0
        do
if test -s $1
        then
        echo "no file $1" > &2
        else
        sort + 1 - 2 $1 | tr -d ... (процедуры)
fi
        shift (* перенумеровать аргументы *)
done
```

Процедуры выполняются над всеми аргументами.

Оператор цикла ***until***

Инвертирует условие повторения по сравнению с while

Синтаксис:

```
until <команда>
do
<команды>
done
```

Пока «команда» не выполнится успешно, выполнять команды, завершаемые словом done.

Пример:

```
if test S# -eq 0
then
        echo "Usage $0 file..." > &2
        exit
fi
```

```
        until test S# -eq 0
        do
        if test -s $1
        then
        echo "no file $1" > &2
else
        sort +1 -2 $1 | tr -d ... (процедура)
fi
        shift (сдвиг аргументов)
 done
```

Исполняется аналогично предыдущему.

Оператор выбора ***case***

Синтаксис:

```
case <string> in
string1) <если string = string1, то выполнить все следующие команды до ;; >
;;
string2) <если string = string2, то выполнить все следующие команды до ;; >
;;
string3) ... и т. д. ...
esac
```

Пример:

Пусть процедура имеет опцию -t, которая может быть подана как первый параметр:

```
.................
together = no
case $1 in
```

```
-t)     together = yes
        shift ;;
-?)     echo "$0: no option $1"
        exit ;;
esac
        if test $together = yes
        then
        sort ...
fi
```

где ? — метасимвол (если -?, т. е. «другая» опция, отличная от -t, то ошибка). Можно употреблять все метасимволы языка Shell, включая ?, *, [-].

Использование временных файлов в каталоге /tmp

Это специальный каталог, в котором все файлы доступны на запись всем пользователям.

Если некоторая процедура, создающая временный файл, используется несколькими пользователями, то необходимо обеспечить уникальность имен создаваемых файлов. Стандартный прием — имя временного файла $0$$, где $0 — имя процедуры, а $$ — стандартная переменная, равная уникальному идентификационному номеру процесса, выполняющего текущую команду.

Хотя администратор периодически удаляет временные файлы в /tmp, хорошей практикой является их явное удаление после использования.

Комментарии в процедурах

Они начинаются с двоеточия :, которое считается нуль-командой, а текст комментария — ее аргументом. Чтобы Shell не интерпретировал метасимволы ($, * и т. д.), рекомендуется заключать текст комментария в одиночные кавычки.

В некоторых вариантах ОС UNIX примечание начинается со знака #.

Пример процедуры

```
:'Эта процедура работает с файлами, содержащими имена'
: 'и номера телефонов,'
:'сортирует их вместе или порознь и печатает результат на'
:'экране или на принтере'
:'Ключи процедуры:'
:'-t (together) - слить и сортировать все файлы вместе'
:'-p (printer) - печатать файлы на принтере'
if test $# - eq 0
then
        echo "Usage: $ 0 file ... " > & 2
        exit
fi
together = no
print = no
while test $# -gt 0
do case $1 in
-t)     together = yes
        shift ;;
-p)     print = yes
        shift ;;
-?)     echo "$0: no option $1"
```

```
        exit ;;
*) if test $together = yes
then
        sort -u +1 -2 $1 | tr ... > /tmp/$0$$
        if $print = no
then
        cat /tmp/$0$$
        else
        lpr -c /tmp/$0$$
fi
        rm /tmp/$0$$
        exit
else if test -s $1
then   echo "no file $1" > &2
        else    sort +1 -2 $1 | tr...> /tmp/$0$$
if $print = no
then   cat /tmp/$0$$
else   lpr -c /tmp/$0$$
fi
                rm /tmp/$0$$
        fi
shift
fi;;
        esac
done.
```

Процедура проверяет число параметров $# и, если оно равно нулю, завершается. В противном случае она обрабатывает параметры (оператор case). В качестве параметра может выступать либо ключ (символ, предва-

ряемый минусом), либо имя файла (строка, представленная метасимволом *). Если ключ отличен от допустимого (метасимвол ? отличен от t и p), процедура завершается. Иначе в зависимости от наличия ключей t и p выполняются действия, заявленные в комментарии в начале процедуры.

Обработка прерываний в процедурах

Если при выполнении процедуры получен сигнал прерывания (от клавиши BREAK или DEL, например), то все созданные временные файлы останутся неудаленными (пока это не сделает администратор) ввиду немедленного прекращения процесса.

Лучшим решением является обработка прерываний внутри процедуры оператором ***trap***. Синтаксис:

trap 'command arguments' signals...

Кавычки формируют первый аргумент из нескольких команд, разделенных точкой с запятой. Они будут выполнены, если возникнет прерывание, указанное аргументами signals (целые):

– 2 — когда вы прерываете процесс;

– 1 — если вы "зависли" (отключены от системы) и др.

Пример (развитие предыдущего):

```
case $1 in
.....
*) trap 'rm /tmp/*; exit' 2 1 (удаление временных файлов)
if test -s $1
.............
rm /tmp/*
```

Лучше было бы:

```
trap 'rm /tmp/* > /dev/null; exit' 2 1
```

так как прерывание может случиться до того, как файл /tmp/$0$$ создан и аварийное сообщение об этом случае перенаправляется на null-устройство.

Выполнение арифметических операций: expr

Команда expr вычисляет значение выражения, поданного в качестве аргумента, и посылает результат на стандартный вывод. Наиболее интересным применением является выполнение операций над переменными языка Shell.

Пример суммирования 3 чисел:

```
$ cat sum3
expr $1 + $2 + $3
$ chmod 755 sum3
$ sum3 13 49 2
64
$
```

Пример непосредственного использования команды:

```
$ expr 13 + 49 + 2 + 64 + 1
129
$
```

В expr можно применять следующие арифметические операторы: +, -, *, /, % (остаток). Все операнды и операции должны быть разделены пробелами.

Заметим, что знак умножения следует заключать в кавычки (одинарные или двойные), например: '*', так как символ * имеет в Shell специальный смысл.

Более сложный пример expr в процедуре (фрагмент):

```
num = 'wc -l < $1'
tot = 100
count = $num
avint = 'expr $tot / $num'
avdec = 'expr $tot % $num'
```

```
while test $count -gt 0
do ...
```

Здесь wc -l осуществляет подсчет числа строк в файле, а далее это число используется в выражениях.

Отладка процедур Shell

Имеются три средства, позволяющие вести отладку процедур.

1. Размещение в теле процедуры команд echo для выдачи сообщений, являющихся трассой выполнения процедуры.

2. Опция -v (verbose = многословный) в команде Shell приводит к печати команды на экране перед ее выполнением.

3. Опция -x (execute) в команде Shell приводит к печати команды на экране по мере ее выполнения с заменой всех переменных их значениями; это наиболее мощное средство.

2 ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа по курсу «Вычислительные системы, сети телекоммуникации» состоит из двух частей: практической и теоретической.

В ***практической части*** студенты должны разработать командные файлы.

При разработке необходимо учесть возможность обработки различных ошибок ввода данных, например неправильного запуска программ (с недостаточным количеством параметров или с неправильными параметрами), и предусмотреть вывод сообщения об ошибке и подсказки.

Программа может быть реализована как в виде командных файлов ОС Windows, так и в виде скриптов Shell ОС Unix по выбору студента; представляется на проверку преподавателю в виде отдельного файла наряду с отчетом по лабораторной работе в формате Microsoft Word. Каждая строчка командного файла должна сопровождаться подробными комментариями.

Теоретическая часть содержит ряд вопросов по учебному пособию, на которые студент должен дать развернутый и аргументированный ответ.

3 ВАРИАНТЫ ЗАДАНИЙ НА ВЫПОЛНЕНИЕ

Вариант 1

Практическая часть

Разработать командный файл, создающий, копирующий или удаляющий файл, указанный в параметре строки при запуске командного файла, в зависимости от выбранного ключа /n , /c , /d.

Теоретическая часть

1. Дайте определение термину «система». Какие характеристики используют для описания систем?

2. Опишите основные характеристики накопителей на жестком магнитном диске.

Вариант 2

Практическая часть

Разработать командный файл, который бы проверял событие: «Запускали сегодня его уже или нет». Если файл уже запускали, то выйти из программы, если нет, то файл должен запустить какой-либо текстовый редактор. Для определения события выполните сравнение дат (последнего запуска и текущей) через переменные, а не через файлы. Вам поможет системная переменная %DATE% и команда SET (под ОС Windows).

Теоретическая часть

1. Дайте определение понятия «вычислительная система».

2. Чем отличается физическая организация магнитного диска от логической?

Вариант 3

Практическая часть

Разработать командный файл, добавляющий вводом с клавиатуры содержимое текстового файла (в начало или в конец в зависимости от ключей /b /e). Используйте команды перенаправления потоков ввода-вывода.

Теоретическая часть

1. История развития вычислительной техники.

2. Перечислите достоинства и недостатки различных файловых систем.

Вариант 4

Практическая часть

Разработать командный файл, который бы проверял событие: «Запускали сегодня его уже или нет». Если файл уже запускали, то выйти из программы, если нет, то файл должен запустить какой-либо текстовый редактор. Для определения события сохраните текущую дату и дату последнего запуска в файлы и выполните сравнение файлов командой FC. Результат сравнения можно определить, используя команду IF ERRORLEVEL (под ОС Windows).

Теоретическая часть

1. Опишите электронные вычислительные машины и приведите их классификацию.

2. Как в ОС Windows на платформе NT можно управлять дисками и файловыми системами?

Вариант 5

Практическая часть

Разработать командный файл, регистрирующий время своего запуска в файле протокола run.log и автоматически запускающий некоторую программу (например, антивирусную и т. п.) по пятницам или 13 числам. Определение даты запуска можно выполнить одним из двух возможных способов: использовать сравнение переменных (вырезать подстроку из системной переменной %DATE% и сравнить с числом. См. команду SET); сохранить текущую дату в файл и выполнить в нем поиск.

Теоретическая часть

1. Сравните определения архитектуры вычислительной системы и архитектуры электронной вычислительной машины.

2. Какие преимущества в ОС Windows на платформе NT дает использование файловой системы NTF пятой версии?

Вариант 6

Практическая часть

Разработать командный файл, который дописывал бы имя файла, полученного входным параметром в сам файл N количество раз. N — также задается параметром.

Теоретическая часть

1. В чем отличие принстонской архитектуры ЭВМ от гарвардской архитектуры ЭВМ?

2. Назовите последовательно уровни сетевой модели OSI.

Вариант 7

Практическая часть

Разработать командный файл, который в интерактивном режиме (командный файл «задает вопросы», а пользователь на них отвечает) мог бы дописывать в файл текст, удалять строки из файла и распечатывать на экране содержимое файла. Удаление строк можно реализовать либо через команду поиска строк в файле, либо через команду организации циклов FOR.

Теоретическая часть

1. Какие свойства ЭВМ относятся к общим, а какие к индивидуальным свойствам?

2. Что входит в состав физической инфраструктуры сети?

Вариант 8

Практическая часть

Разработать командный файл, который бы получал в качестве параметра какое-либо имя и проверял, определена ли такая переменная среды или нет, и выводил соответствующее сообщение.

Теоретическая часть

1. Какие свойства имеет CISC-архитектура процессора?

2. В чет состоит отличие локальной вычислительной сети от глобальной компьютерной сети?

Вариант 9

Практическая часть

В некотором файле хранится список пользователей ПК и имена их домашних каталогов. Каждый пользователь и имя его каталога в отдельной строке. Необходимо разработать программу, которая просматривает данный файл и в интерактивном режиме (командный файл «задает вопросы», а пользователь на них отвечает) задает вопрос — копировать текущему пользователю (в его домашний каталог) какой-либо заданный файл в качестве параметра или нет. Если «Да» то программа копирует файл.

Теоретическая часть

1. Какие свойства имеет RISC-архитектура процессора?

2. Что такое сеть периметра?

Вариант 10

Практическая часть

Разработать командный файл, который помещает список файлов текущего каталога в текстовый файл и в зависимости от ключа сортирует по какому-либо полю. Реализовать два варианта: с использованием только команды DIR, с использованием команд DIR и SORT.

Теоретическая часть

1. Опишите основные «вехи» развития микропроцессоров семейства x86-64.

2. В чем состоит отличие технологии VPN от Dial-up сервиса?

Вариант 11

Практическая часть

Разработать командный файл, создающий, копирующий или удаляющий каталог, указанный в командной строке, в зависимости от выбранного ключа (замещаемого параметра) /n , /c , /d.

Теоретическая часть

1. Какие режимы работы имеют микропроцессоры семейства x86-64 и что они собой представляют?

2. Что Вы знаете о Microsoft Active Directory?

Вариант 12

Практическая часть

Разработать командный файл, который бы выводил в зависимости от ключа на экран имя файла с самой последней или с самой ранней датой последнего использования в текущем каталоге.

Теоретическая часть

1. Опишите существующие единицы информации и их представление.

2. В чем состоит преимущество использования доменов?

Вариант 13

Практическая часть

Разработать командный файл, который бы получал в качестве аргумента имя текстового файла и выводил на экран информацию о том, сколько символов, слов и строк в текстовом файле. Количество символов равно размеру файла.

Теоретическая часть

1. Дайте определения видам памяти и их месту в иерархии.

2. Приведите обзор семейства протоколов TCP/IP.

Вариант 14

Практическая часть

Разработать командный файл, копирующий произвольное число файлов, заданных аргументами из текущего каталога в указываемый каталог. Используйте проверку на пустые параметры и команду SHIFT.

Теоретическая часть

1. Что собой представляет адресация и распределение памяти в реальном режиме работы микропроцессора Intel x86?

2. Приведите примеры протоколов транспортного и прикладного уровней TCP/IP.

Вариант 15

Практическая часть

Разработать командный файл, который получал бы в качестве параметра какой-либо символ и в зависимости от второго параметра вырезал или сохранял в заданном файле все строки, начинающиеся на этот символ. Можно выполнить с помощью команды FOR (под ОС Windows).

Теоретическая часть

1. Что собой представляет адресация и распределение памяти в защищенном режиме работы микропроцессора Intel x86?

2. Как строится адресация TCP/IP четвертой версии?

Вариант 16

Практическая часть

Разработать командный файл (аналог команды tail в Unix). Командный файл печатает конец файла. По умолчанию — 10 последних строк.

Явно можно задать номер строки, от которой печатать до конца. Если задание будет выполняться под ОС Unix, команду tail использовать нельзя.

Теоретическая часть

1. Что собой представляет адресация и распределение памяти в архитектуре AMD64?

2. Назовите номер сети и номер хоста, если у компьютера IP адрес 192.32.09.220 и маска подсети 255.255.255.0.

Вариант 17

Практическая часть

Разработать командный файл, который бы склеивал текстовые файлы, заданные в качестве аргументов, и сортировал бы строки результирующего файла в зависимости от ключа по убыванию или по возрастанию.

Теоретическая часть

1. Как в ОС Windows на платформе NT можно узнать такие параметры, как общий размер физической памяти и размер памяти ядра ОС?

2. Как строится процесс разрешения DNS?

Вариант 18

Практическая часть

Разработать командный файл, который формировал бы ежемесячный отчет об изменениях в рабочем каталоге (файлы измененные). Под ОС Windows можно воспользоваться анализом атрибутов файлов.

Теоретическая часть

1. Что такое файл подкачки и как им можно управлять в ОС Windows на платформе NT?

2. Какие утилиты диагностики сети Вы знаете?

Вариант 19

Практическая часть

Разработать командный файл, который формировал бы ежемесячный отчет об изменениях в рабочем каталоге (файлы созданные, удаленные). Необходимо хранить список файлов в файле истории.

Теоретическая часть

1. Приведите классификацию устройств ввода-вывода.

2. Опишите протоколы HTTP и HTTPS.

Вариант 20

Практическая часть

Выполняющий в зависимости от ключа один из 3–х вариантов работы:

– с ключом /n дописывает в начало указанных текстовых файлов строку с именем текущего файла;

– с ключом /b создает резервные копии указанных файлов;

– с ключом /d удаляет указанные файлы после предупреждения.

Количество обрабатываемых файлов может быть переменным и задаваться в качестве параметров.

Теоретическая часть

1. Опишите основные характеристики устройств внешней памяти.

2. Опишите протоколы SMTP и POP3.

СПИСОК ЛИТЕРАТУРЫ

1 Гриценко Ю. Б. Вычислительные системы, сети и телекоммуникации : учебное пособие / Ю. Б. Гриценко. — Томск : ФДО, ТУСУР, 2016. — 134 с.

2 Назаров С. В. Операционные системы : практикум / С. В. Назаров, Л. П. Гудыно, А. А. Кириченко; под ред. С. В. Назарова — М.: КУДИЦ-ПРЕСС, 2008. — 464 с. : ил.