###### СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ

В. Г. КОБЫЛЯНСКИЙ

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Учебно – методическое пособие

Новосибирск 2018

681.3.066(07)

Кобылянский В. Г. Операционные системы: учебно – методическое пособие. – ­Новосибирск: Изд-во СГУПСа, 2018. – 72 с.

Учебно – методическое пособие содержит материалы для выполнения 5 лабораторных работ по изучению командного режима работы в среде операционных систем Windows и Linux. Описание каждой работы включает теоретический материал, порядок выполнения и контрольные вопросы для самопроверки.

Пособие предназначено для студентов направлений 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика» факультета бизнес-информатики.

Рассмотрено и рекомендовано к изданию на заседании кафедры «Информационные технологии на транспорте»

О т в е т с т в е н н ы й р е д а к т о р

проф.кафедры ИТТ, д-р техн. наук *В. И. Хабаров*

Р е ц е н з е н т

ст. преп. кафедры «Общая информатика» СГУПСа И. Н. Басев

© Кобылянский В. Г., 2018

© Сибирский государственный

университет путей сообщения, 2018 г.

**Содержание**

[Введение 4](#_Toc442910772)

[Лабораторная работа № 1. Командный язык операционной системы Windows 5](#_Toc442910773)

[Лабораторная работа № 2. Командные файлы Windows 22](#_Toc442910774)

[Лабораторная работа № 3. Прохождение программ в среде ОС 31](#_Toc442910775)

[Лабораторная работа № 4. Командный язык операционной системы Linux 41](#_Toc442910776)

[Лабораторная работа № 5. Командные сценарии Linux 55](#_Toc442910777)

[Список источников 65](#_Toc442910778)

[Приложение 1. Порядок создания виртуальной машины TinyCore Plus 66](#_Toc442910779)

# Введение

Пользователи современных операционных систем (ОС) в большинстве случаев используют для управления компьютером графические интерфейсы, в состав основных элементов которых входят окна, иконки, курсоры и т.д., а основным устройством управления является мышь. Достоинством графических интерфейсов является наглядное представление всех ресурсов компьютера (дисков, программ, каталогов и файлов) для непрофессионального пользователя, а недостатком - ограниченные возможности пользователя по управлению ресурсами. Любой графический интерфейс реализует только те функции, которые заложены в него разработчиком этого интерфейса. При этом, например, пользователи не могут выполнить операцию сравнения файлов, определить маршрут передачи сообщений на заданный сайт Интернета и т.д.

Главным интерфейсом для любой ОС является командный интерфейс, представленный набором команд. Количество команд ОС может быть достаточно большим, каждая команда реализует одно действие над заданным ресурсом, а основным устройством управления при этом является клавиатура. Графический интерфейс является надстройкой над командным интерфейсом, т.е. любое действие, заданное мышью, выполняется с помощью соответствующей команды ОС.

Командный интерфейс требует от пользователя более глубоких знаний устройства компьютера и ОС, поэтому он прежде всего предназначен для применения IT-специалистами. Целью данного пособия является оказание помощи студентам в изучении командных интерфейсов ОС Windows и Linux.

# Лабораторная работа № 1. Командный язык операционной системы Windows

1. **Цель работы**

Целью работы является приобретение практических навыков работы в командном режиме операционной системы (ОС) Windows.

1. **Общие сведения о командах**

Диалог пользователя с MS Windows осуществляется в форме команд. Операционная система готова к диалогу, если на экране имеется приглашение A: или C:>. Команда состоит из имени команды и параметров, разделенных пробелами. Команда может набираться как прописными так и строчными латинскими буквами, а завершается нажатием клавиши ENTER. Формат команды следующий:

*команда [ключи] [аргументы ]*

В качестве аргументов команды выступают имена устройств, каталогов или файлов, а ключи уточняют действие команды. Ключи и аргументы не являются обязательными для всех команд, т.е. возможны команды, не имеющие параметров (например, CLS).

Команды бывают двух типов: внутренние и внешние. Внутренние команды выполняются командным процессором (файл Cmd.exe); внешние команды реализуются программами, поставляемыми вместе с операционной системой в виде отдельных файлов.

Для того, чтобы выполнить несколько команд в одной командной строке, надо объединить их с помощью символа условной обработки:

* *команда1 & команда2* — команды выполняются последовательно (сначала первая команда, затем вторая);
* *команда1 && команда2* — вторая команда выполняется только в случае успешного завершения первой команды;
* *команда1 || команда2* — вторая команда выполняется только тогда, когда первая команда не была выполнена;
* *команда1 | команда2* — первая команда выполняется и ее результаты передаются на вход второй команды.

**3. Основные соглашения Windows об именах файлов и устройств**

3.1 Имя устройства

В состав компьютера входит множество устройств, к некоторым из них можно обращаться через команды ОС. К таким устройствам относятся внешние запоминающие устройства (ВЗУ), клавиатура, монитор, принтер, порты.

В качестве ВЗУ используются устройства хранения информации на магнитных, лазерных и магнитооптических дисках, а также различные USB – флэш- накопители, которые обозначаются буквами латинского алфавита. Имена A: и B: зарезервированы за гибкими магнитными дисками. Обратите внимание: после имени диска обязательно ставится двоеточие!

Для смены текущего диска достаточно набрать его имя в командной строке. Например: C: - переключение на работу с жестким диском.

Клавиатура и монитор являются устройствами системного ввода и вывода, для обозначения которых в информатике применяется специальный термин «консоль», поэтому для обращения к этим устройствам используется групповое имя CON. При вводе информации консолью является клавиатура, а при выводе – монитор.

Обмен данными с внешним миром компьютер реализует через специальные устройства, получившие название портов. Для ОС порты и устройства, подключаемые через соответствующий порт к компьютеру, имеют одинаковое имя. В системе известны следующие имена портов: COM1 - COM3 – устройства (порты) с последовательным интерфейсом; LPT1 - LPT3 – устройства (порты) с параллельным интерфейсом. Для принтера в системе зарезервировано имя PRN, а поскольку принтер обычно подключается через параллельный порт, то к нему можно обратиться и по имени порта, например, LPT1

В ОС имеется также зарезервированное имя для фиктивного устройста NUL, которое удобно использовать в командах переназначения вывода.

3.2 Имя файла

Имя файла Windows, включая расширение, может содержать до 255 символов. Допустимы прописные и строчные буквы латиницы и кириллицы, цифры, знаки пунктуации и специальные символы: $, #, &, @, !, %, (, ), {, }, ~, ^, -, \_. Имена файлов не должны совпадать с именами устройств, зарезервированными в системе.

3.3 Имя каталога

Каталог - специальное место на диске, в котором хранятся имена файлов и основные сведения о них. Каждый каталог имеет свое имя и может быть зарегистрирован в другом каталоге. Если каталог А зарегистрирован в каталоге В, то А является подкаталогом В, а В - родительский каталог для А.

Каждый магнитный диск имеет один главный (корневой) каталог и множество подкаталогов. Корневой каталог создается во время форматирования диска, поэтому его размер ограничен. Подкаталоги создаются пользователем и представляют собой файлы специального вида. Каталог, с которым в данное время работает пользователь, называется текущим.

Требования к имени каталога те же, что и к именам файлов, как правило расширение имени для каталога не используется. В ОС имеется несколько зарезервированных имен для каталогов:

. – имя текущего каталога;

.. – имя родительского каталога.

3.4 Имя пути

Путь - это последовательность из имен каталогов, разделенных символом "\". Путь задает маршрут от текущего или корневого каталога к тому каталогу, где находится нужный файл.

Если путь начинается с символа "\", то он называется абсолютным и вычисляется от корневого каталога диска, иначе – путь является относительным и отсчитывается от текущего каталога. Например: CD \DOD\EXE - смена текущего каталога с абсолютным указанием пути; CD ..\LETTER - смена текущего каталога с относительным указанием пути.

3.5 Полное имя файла.

Полное имя файла (или спецификация файла) имеет вид:

*имя диска:[\путь\]имя файла[.расширение]*

Обязательным является указание имени файла, остальные элементы спецификации могут отсутствовать. По умолчанию используются текущий диск и текущий каталог.

Во многих командах в именах файлов могут использоваться метасимволы: «\*» и «?» для выполнения групповых операций. Символ "\*" обозначает любое число любых символов (в том числе ни одного), "?" – один произвольный символ. Например: С:\DOD\EXE\PRINT.BAT - полная спецификация файла; C:\DOD\EXE\\*.BAT - групповая спецификация файлов с любым допустимым именем и расширением BAT.

**4. Cинтаксис основных команд**

Рассмотрим основные команды Windows, сгруппировав их по типам выполнямых операций. Подробную информацию по любой команде можно получить, используя ключ “/?”. При описании команды используются следующие обозначения:

d: - имя диска; path – путь доступа; filename – имя файла.

4.1 Команды для работы с файлами

4.1.1 Команда: ATTRIB.

Назначение: установить атрибуты файла.

Формат: ATTRIB [+A|-A][+H|-H][+R|-R][+S|-S][[d:][path]filename][/S]

Параметры:

+A|-A - добавить/удалить атрибут архивации;

+H|-H - добавить/удалить атрибут скрытого файла;

+R|-R - добавить/удалить атрибут защиты от записи;

+S|-S - добавить/удалить атрибуты системного файла;

/S - работа в заданной директории и поддиректориях.

Комментарий: атрибут архивации используется командами BACKUP, XCOPY; атрибут скрытого файла позволяет исключить файл из списка, выдаваемого командой DIR.

Пример:

ATTRIB +R \*.\* - назначить атрибут «только для чтения» всем файлам из текущего каталога.

4.1.2 Команда: FC

Назначение: сравнение файлов

Формат: FC [/A][/C][/N] [d:][path] filename1[d:][path] filename2

Параметры:

filename1, filename2 - имена сравниваемых файлов;

/A - выдать отличия, как ASCII-cимволы;

/C - считать прописные и строчные буквы идентичными;

/N - выдать номер строки, где обнаружены отличия.

Пример:

FC /A A.TXT B.TXT - сравнение двух текстовых файлов из текущего каталога.

4.1.3 Команда: COPY

Назначение: копирование/слияние файлов.

Формат:

-для копирования файлов

COPY sourse [/A|/B][destination][/A|/B][/V]

-для слияния двух и более файлов

COPY [d:][path]filename1[/A|/B] + [d:][path]filename2[/A|/B] [+...] [destination][/A|/B][/V]

Параметры:

sourse - имя файла , который необходимо копировать; destination - имя файла копии;

/A - файл рассматривается как ASCII-файл;

/B - файл рассматривается как бинарный;

/V - проверка копирования;

filename1,filename2 - имена объединяемых файлов.

Комментарий:

1. если использован ключ /A, то копируется файл до первого символа (CTRL-Z), если стоит ключ - /B, то копируется весь файл;
2. в качестве аргументов команды могут использоваться имена устройств;
3. команда не обрабатывает файлы, хранящиеся в подкаталогах.

Примеры:

COPY A.TXT+\*.TXT /V - добавление в текстовый файл A.TXT содержимого всех текстовых файлов из текущего каталога;

COPY CON FILE.DAT – копирование с клавиатуры в файл FILE.DAT.

4.1.4 Команда: DEL

Назначение: удаление файлов.

Формат: DEL [d:][path]filename [/P]

Параметры:

/P - удаление с подтверждением операции.

Комментарий: файлы с атрибутом защиты от записи этой командой не удаляются; удаленные файлы могут быть восстановлены командой UNDELETE.

Пример:

#### DEL \*.BAK - удаление всех файлов с типа BAK из текущего каталога.

4.1.5 Команда: REN

Назначение: переименование файлов.

Формат: REN [d:][path]filename1 filename2

Параметры:

filename1 - старое имя файла;

filename2 - новое имя файла.

Комментарий: возможно групповое переименование с помощью метасимволов «\*» и «?»; команда не позволяет переименовывать каталоги.

Пример:

REN \*.TXT \*.DOC - смена расширения файлов в текущем каталоге.

4.1.6 Команда: TYPE

Назначение : вывод содержимого текстового файла на экран.

Формат: TYPE [d:][path]filename

Комментарий: в имени файла нельзя использовать глобальные символы.

Пример:

TYPE A.DOC - вывод файла A.DOC на экран.

4.1.7 Команда: XCOPY

Назначение: копирование файлов и каталогов.

Формат: XCOPY sourse [destination][/E][/S][/V][/M]

Параметры:

sourse - имя файла, который надо копировать;

destination - имя файла, в который надо копировать;

/A - копировать те файлы, где установлен атрибут архивации;

/E - копировать и пустые подкаталоги;

/M - копировать файлы с атрибутом архивации, затем этот атрибут убрать;

/S - копировать каталог с подкаталогами;

/V - выполнить проверку операции;

Пример:

XCOPY A:\PROG /M - копирование файлов с включенным атрибутом «архивный».

4.1.8 Команда: MOVE

Назначение: пересылка файлов и переименование каталогов.

Формат: MOVE [/Y] [d:][path]file\_old [d:][path]file\_new

Параметры:

file\_old - старое имя файла;

file\_new - новое имя файла;

/Y – отказ от запроса на подтверждение на замену файлов.

Комментарий: возможна групповая пересылка с использованием метасимволов «\*» и «?».

Пример:

MOVE \*.TXT D:\MYDIR - пересылка всех файлов типа TXT из текущего каталога в каталог D:\MYDIR.

4.1.9 Команда: WHERE

Назначение: поиск файлов по заданному шаблону.

Формат: WHERE [/R каталог] [/Q] [/F] [/T] шаблон

Параметры:

/R – рекурсивный поиск, начиная с указанного каталога;

/Q – возврат только кода завершения без вывода списка найденных файлов;

/F – вывод имен найденных файлов в кавычках;

/T – вывод размера, даты и времени изменения найденных файлов;

Комментарий:

1) поиск проводится в текущем каталоге и в каталогах, указанных в глобальной переменной PATH;

2) шаблон поиска задает имена искомых файлов;

3) в шаблоне можно использовать метасимволы «\*» и «?»;

4) команда применяется в версиях Windows 7 и более поздних.

Примеры:

WHERE DISK\* - поиск файлов, имена которых начинаются строкой ”DISK”;

WHERE /R %WINDIR%\SYSTEM32 \*.LOG – поиск файлов с расширением LOG, начиная с подкаталога SYSTEM32 системного каталога Windows.

4.2 Команды управления ОС

4.2.1 Команда: PATH

Назначение: установить пути поиска исполняемых файлов.

Формат: PATH [[d:][path[;[d:]path][;...]]

Параметры:

path - маршруты, где ОС ищет исполняемые файлы.

Комментарий: указанные в команде маршруты запоминаются в глобальной пере­менной PATH.

Пример:

PATH C:\WINDOWS\SYSTEM32;C:\EXE - при запуске программ, помимо текущего каталога просматривать указанные в команде каталоги.

4.2.2 Команда: PROMPT

Назначение: установить вид системного приглашения.

Формат: PROMPT [параметры]

Параметры:

$$ - символ «$»;

$D - текущая дата;

$G - символ >;

$T - текущее время суток;

$P - текущий каталог на принимаемом по умолчанию устройстве;

$L - символ <;

Комментарий: чаще всего используется системное приглашение в виде указания имени текущего каталога, завершающееся символом «>».

Пример:

PROMPT $T$G - задание системного приглашения в виде указания текущего времени.

4.2.3 Команда: SET

Назначение: установить значение глобальной переменной в области окружения.

Формат: SET [key] [variable=[string]]

Параметры:

variable - имя глобальной переменной,

string – строка символов, задающая значение переменной окружения,

key – ключ ( /P или /A).

Комментарий:

1) если команда введена без параметров, то она вы­водит текущие значения всех глобальных переменных;

2) ключи доступны только при включении расширенной обработки команд;

3) ключ /P позволяет присвоить значение глобальной переменной из входной строки, введенной пользователем, и показывает приглашение promptString перед чтением этой строки; приглашение может быть пустым (SET /P variable=[promptString]);

4) ключ /A указывает, что строка справа от знака равенства является числовым выражением, значение которого вычисляется с использованием арифметических и логических операций (SET /A выражение);

5) для удаления переменной необходимо присвоить ей пустое значение.

Примеры:

SET LIB=D:\TC\LIB - установка значения глобальной переменной LIB;

SET /P MYFILE=Введите имя файла - вывод на экран строки «Введите имя файла » и ввод ответа пользователя в глобальную переменную MYFILE;

SET /A ALFA+GAMMA - вывод на экран суммы числовых значений глобальных переменных ALFA и GAMMA;

SET LIB= - удаление глобальной переменной LIB.

4.2.4 Команда: DATE

Назначение: установить значение системной даты.

Формат: DATE [date]

Параметры:

date – значение даты.

Комментарий: если команда подана без параметров, то она вы­водит текущую дату.

Пример:

##### DATE 25/12/2015

4.2.5 Команда: TIME

Назначение: установить значение системного времени.

Формат: TIME [time]

Параметры:

time – значение времени.

Комментарий: если команда подана без параметров, то она вы­водит текущее время.

Пример:

##### TIME 14:45

4.2.6 Команда: VER

Назначение: вывод версии операционной системы.

Формат: VER

4.2.7 Команда SYSTEMINFO.

Назначение: вывод подробной информации о конфигурации компьютера.

Формат: systeminfo

Комментарий: выводит сведения о версии, типе и изготовителе операционной системы, процессоре, версии BIOS, объеме памяти, часовом поясе и конфигурации сетевого адаптера.

4.3 Команды работы с дисками/каталогами

4.3.1 Команда: CHDIR (CD)

Назначение: изменить текущий каталог.

Формат: CD [d:][path]

#### Комментарий: команда без параметров выводит имя текущего каталога

Пример:

СD .. - сделать текущим родительский каталог.

4.3.2 Команда: CHKDSK

Назначение: проверить диск на наличие системных ошибок.

Формат: CHKDSK [[d:][path]filename][/F][/V]

Параметры:

filename - имена файлов, проверяемых на фрагментацию;

/F - исправлять ошибки, обнаруженные на диске;

/V - печатать имена файлов по мере проверки диска.

Комментарий: команда проверяет структуру каталогов и файлов.

Пример:

CHKDSK A: /F - выполнить проверку гибкого диска.

4.3.3 Команда: DISKCOPY

Назначение: копирование содержимого одного гибкого диска на другой.

Формат: DISKCOPY [d1:[d2:] [/V]

Параметры:

d1 - диск - источник;

d2 - диск - приемник;

/V - выполнить проверку операции.

Комментарий: команда выполняет физическое копирование дисков и работает только с дисками равного объема.

Пример:

DISKCOPY A: A: - физическое копирование дискет при наличии в компьютере одного накопителя.

4.3.4 Команда: FORMAT

Назначение: форматирование логического диска.

Формат: FORMAT d: [/T:tracks /N:sectors] [/F:size] [/Q] [/S] [/U]

Параметры:

/T:tracks - число дорожек на диске;

/N:sectors - число секторов на дорожке;

/F:size - емкость диска;

/Q - быстрое форматирование;

/S - копировать системные файлы после форматирования,

/U - безусловное форматирование;

Комментарий: любой диск, отформатированный без параметра /U, может быть восстановлен командой UNFORMAT.

Пример:

FORMAT A: /F:1440 /S - форматирование системного гибкого диска на 1,44 Мб.

4.3.5 Команда: MKDIR (MD)

Назначение: создать каталог

Формат: MD [d:]path

Параметры:

path - имя создаваемого каталога.

Пример:

MD A:\PROG - cоздать подкаталог первого уровня на гибком диске.

4.3.6 Команда: RMDIR (RD)

Назначение: удалить директорию.

Формат: RD [d:]path

Параметры:

path - имя каталога, который нужно удалить.

Комментарий: удаляемый каталог не должен содержать файлов.

Пример:

RD MYCAT - удалить подкаталог MYCAT в текущем каталоге.

4.3.7 Команда: DIR

Назначение: вывод содержимого каталога на экран.

Формат: DIR [d:] [path] [filename] [/O:order] [/B] [/P] [/S] [/W]

Параметры:

filename - имя файла или файлов;

/O:order - сортировка каталога: (D - сортировка по дате, E - сортировка по расширению; N - сортировка по имени; S - сортировка по размеру);

/B - вывести имена файлов и подкаталогов;

/P - поэкранный вывод;

/S - просмотр подкаталогов;

/W - вывод в широком формате.

Комментарий: ключ /S дает возможность искать файлы в пре­делах всего диска. Пример:

DIR \\*.BAK /S /B - вывод указанных файлов в пределах всего диска.

4.3.8 Команда: VOL

Назначение: вывод метки и серийного номера диска.

Формат: VOL [d:]

Комментарий: команда без параметра выводит метку текущего диска.

Пример:

VOL C: - вывод метки диска C:

4.3.9 Команда: LABEL

Назначение: создание, изменение и удаление метки диска.

Формат: LABEL [d:] [метка]

Комментарий: если при вызове команды не указана метка, то на экран выводится текущая метка диска и выводится запрос на изменение или удаление метки.

Пример:

LABEL A: Петров - запись на диск гибкий диск метки «Петров».

4.4 Переназначение ввода/вывода

Операционная система имеет средства переназначения потоков ввода и вывода данных для любых программ, работающих под ее управлением. Например, если какая-либо программа по умолчанию выводит результаты на экран монитора, то можно этот вывод направить в определенный файл на диске. Для этого имеются специальные команды:

“>>” - переназначить вывод с добавлением новых данных к уже существующим;

“>” - переназначить вывод с замещением данных;

«<» - переназначить ввод данных.

Примеры:

DIR >> CATALOG.DAT - добавление содержимого текущего каталога в существу­ющий файл CATALOG.DAT;

DIR > PRN - вывод содержимого текущего каталога на печать;

4.5 Фильтры и конвейеры

Фильтр - это программа, которая читает заданный поток ввода данных, обра­батывает его, а затем осуществляет вывод. К фильтрам относят команды: FIND, MORE, SORT. С помощью фильтров можно организовать конвейер команд, в котором результаты выполнения одной программы сразу же передаются в качестве входных данных для другой программы. Количество программ, связываемых в один конвейер, может быть произвольным. В конвейере можно использовать команды переназначения ввода или вывода.

##### 4.5.1 Команда: FIND

Назначение: поиск заданной строки символов во входном потоке данных.

Формат: FIND [/I] [/V] string [d:] [path] [filename]

Параметры:

string – строка символов для поиска;

/I – поиск без учета регистра символов;

/V – выводить строки, не содержащие заданную строку string.

Комментарий:

1. выводит на экран все строки из входного потока, содержащие заданную символьную строку;
2. если не указана спецификация файла, в котором требуется проводить поиск, то программа FIND обрабатывает данные, введенные с клавиатуры.

Пример:

FIND “привет” C:\MYLETTER.TXT – поиск слова “привет” в заданном файле.

##### 4.5.2 Команда: SORT

Назначение: сортировка входного потока данных из файла или с клавиатуры.

Формат: SORT [/R] [/+n] [d:] [path] [filename]

Параметры:

/R – сортировка в обратном порядке;

/+n – сортировка строк, начиная с позиции n.

Комментарий:

1) выводит на экран упорядоченные строки из входного потока;

2) если не указана спецификация файла, то программа SORT обрабатывает данные, введенные с клавиатуры.

Пример:

SORT /+10 C:\MYLETTER.TXT – вывести на экран упорядоченные строки из заданного файла; сортировку проводить, начиная с 10 позиции каждой строки. Если файл содержит, например, расписание движения самолетов, и с 10 позиции в каждой строке этого файла хранится название пункта назначения, то сортировка будет проводится по этому названию.

##### 4.5.3 Команда: MORE

Назначение: последовательный вывод на экран входных данных фрагментами по 24 строки (одна экранная страница)

Формат: MORE [d:] [path] [filename]

Комментарий: если не указана спецификация файла, то программа MORE обрабатывает данные, введенные с клавиатуры.

Пример:

MORE C:\ANKETA.DAT – вывод на экран содержимого заданного файла по 24 строки.

##### 4.5.4 Команда: | (вертикальная черта)

Назначение: организация конвейера команд.

Примеры:

DIR | SORT - вывод отсортированного каталога на экран, сортировка проводится по первой позиции каждой строки;

DIR | SORT > CATALOG.DAT - вывод отсортированного ка­талога в файл CATALOG.DAT;

DIR | SORT | MORE - постраничная выдача отсортированного каталога на экран;

DIR | FIND "DIR" - вывод имен подкаталогов из текущего каталога на экран (если Вы работаете в русифицированной версии ОС, то с помощью команды FIND надо искать строку «КАТАЛОГ» вместо «DIR»).

4.6 Сетевые команды

4.6.1 Команда: IPCONFIG

Назначение: вывод информации по сетевым подключениям.

Формат: IPCONFIG [/all]

#### Параметр ALL используется для вывода полной информации.

#### Комментарий: команда без параметров выводит краткую информацию.

Пример:

#### IPCONFIG – вывод краткой информации по сетевым подключениям.

4.6.2 Команда: PING

Назначение: проверка связи с указанным узлом сети.

Формат: PING [адрес] [-n число запросов]

#### Параметры:

#### адрес - IP адрес или доменное имя узла сети;

-n – указывает число посылаемых запросов;

Комментарии: команда посылает указанному узлу сети запросы и в течение заданного времени ожидает ответа. При наличии ответа выводит информацию о времени прохождения запросов. Команда работает на низком уровне стека сетевых протоколов, поэтому работает даже при отказе сетевых служб операционной системы.

Примеры:

#### PING ngs.ru

PING 10.242.46.1

4.6.3 Команда: NET

Назначение: управление сетевыми ресурсами и службами.

Формат: NET [подкоманда]

#### Комментарий: подкоманда указывает требуемое действие, например user – обеспечивает просмотр, добавление и редактирование учетных сведений о пользователях, time – синхронизирует часы всех компьютеров сети, start – выводит список всех запущенных служб сети и т.д.

Примеры:

NET – выводит список всех подкоманд;

NET user – просмотр учетных сведений пользователей.

**5. Порядок выполнения работы**

5.1 Выйти в режим командной строки (Пуск/Программы/Стандартные/Командная строка или Пуск/ Выполнить/cmd.exe)

5.2 Запросить версию ОС, с которой Вы работаете .

5.3 Очистить экран монитора.

5.4 Отформатировать флэш –накопитель, предварительно сохранив его содержимое в отдельном каталоге жесткого диска.

5.5 Задать метку флэш –диска.

5.6 Используя команды работы с каталогами, сделать следующее:

* создать каталог **mydir** на флэш –диске;
* скопировать в него не менее 5 файлов из любого подкаталога жесткого диска;
* вывести содержимое каталога **mydir** на экран, изучить структуру записей каталога и занести их в отчет;
* познакомиться со справкой по команде DIR и отсортировать содержимое каталога **mydir** по имени и по размеру файлов;
* используя команду переназначения вывода, записать содержимое каталога **mydir** в файл **dir.dat** корневого каталога флэш –диска;
* вывести содержимое файла **dir.dat** на экран;
* удалить файл **dir.dat**;
* с помощью конвейера, состоящего из команд DIR и FIND, вывести имена подкаталогов корневого каталога диска С: на экран;

5.7 Скопировать исполяемый файл (свою программу, например, прошлогоднюю лабораторную работу \*.exe) в каталог **mydir.** На примере скопированной программы освоить запуск программ из разных каталогов:

* из текущего каталога;
* из любого каталога с указанием пути;
* из любого каталога с использованием глобальной переменной PATH.

5.8 Посмотреть и занести в отчет значение переменной **path**.

5.9 Удалить каталог **mydir** на флэш –диске.

5.10 Используя команды работы с файлами и дисками, выполнить следующие операции:

* создать каталог **text** на флэш –диске и создать в нем два текстовых файла (**1.txt** и **2. txt**);
* выполнить конкатенацию созданных текстовых файлов, полученному в результате слияния файлу дать имя **a. txt**;
* сравнить построчно файл **a. txt** с файлом **1.txt** и результат сравнения занести в файл **a.prt**, пояснить результат сравнения;
* скопировать исполняемые файлы внешних команд, начинающиеся на "**disk**", из папки C:\WINDOWS\SYSTEM32 в корневой каталог флэш –диска;
* выполнить проверку флэш –диска на наличие системных ошибок;
* удалить файл **a. txt**.

5.11 Вывести на экран значения глобальных переменных области окружения, занести их в отчет. Записать новую переменную TOWN, присвоить ей значение «Новосибирск».

5.12 Определить IP адрес Вашего компьютера;

5.13 Проверить наличие связи с соседним компьютером, предварительно уточнив его IP адрес.

**6. Контрольные вопросы**

1 Спецификация файла, использование мета­символов "\*", "?".

2 Понятие внешних и внутренних команд.

3 Синтаксис команд, основные соглашения об именах устройств.

4 Имя каталога, имя пути. Синтаксис и назначение.

5 В чем отличие в операциях копирования COPY и XCOPY. Пояснить на примере.

6 В чем отличие в копировании файлов командой COPY и BACKUP.

7 Синтаксис и назначение команд работы с каталогами и файлами. Поясните по протоколу.

8 Синтаксис и назначение команд обслуживания дисков. По­ясните по протоколу.

9 Син­таксис и назначение команд переназначения ввода/вывода и фильтрации.

# Лабораторная работа № 2. Командные файлы Windows

**1. Цель работы**

Цель работы: получить навыки написания и отладки простых командных файлов.

**2. Методические указания**

2.1 Общие сведения.

Командный файл (КФ) - специальный текстовый файл, содержащий строки команд, выполняемых за один раз (пакет команд). Строками КФ могут быть команды ОС, вызовы исполняемых файлов (.COM,.EXE,.BAT), метки и специальные команды пакетной обработки. Признаком метки является двоеточие в начале строки.

Командные файлы могут создаваться с помощью любого текстового редактора или командой: COPY CON <имя\_файла>.BAT. Все строки, введенные с клавиатуры, после этой команды будут записаны в КФ, ввод завершается нажатием CTRL+Z.

Основные особенности командных файлов:

* каждый КФ должен иметь расширение .BAT или .CMD;
* запуск КФ на выполнение проводится путем указания его имени в командной строке или из любого файлового процессора;
* для прекращения выполнения КФ необходимо нажать клавиши CTRL+C;
* в КФ можно использовать символы переназначения ввода/вывода и конвейер команд;
* в начале каждой команды в КФ можно вставлять специальный символ @, чтобы предотвратить эхо-воспроизведение этой строки на экране;
* cмена устройства и каталога в команде оказывает воздействие на выполнение последующих команд этого КФ;
* установка переменных окружения также влияет на выполнение последующих команд.

2.2 Синтаксис специальных команд пакетной обработки

Все команды пакетной обработки являются внутренними командами.

2.2.1. Команда: CALL

Назначение: вызывает командный файл из другого командного файла.

Формат: CALL [d:][path] filename [parameters]

Параметры:

filenames - имя вызываемого КФ;

parameters - параметры,передаваемые вызываемому КФ;

Комментарий: управление возвращается в вызывающий КФ.

Пример:

CALL SAM - вызов КФ SAM из текущего каталога.

2.2.2. Команда: ECHO

Назначение: включение/отключение вывода команд во время выполнения КФ на экран монитора или вывод сообщения.

Формат: ECHO [ON|OFF] [message]

Параметры:

ON - включить вывод команд на экран;

OFF - выключить вывод команд на экран;

message - текст сообщения.

Комментарий: если команда подана без параметров, то выводится текущее состояние (ON или OFF).

Пример:

@ECHO OFF - отключить вывод команд на экран, команда ECHO тоже не выводится на экран .

2.2.3. Команда: FOR

Назначение: циклическое выполнение команды.

Формат: FOR %%var IN (set) DO command

Параметры:

var - имя переменной, возможные значения от A до Z;

set - список имен или текстовых строк, которые должны быть подставлены в цикле на место %%var;

command - выполняемая команда.

Комментарий: формальная переменная %%var последовательно заменяется фактическими значениями из списка set и при каждой замене выполняется команда command. Если FOR запускается не из пакетного файла, а из командной строки, то вместо двух символов “%” нужно включить только один.

Пример:

FOR %%E IN (A,B,C) DO DEL %%E.OBJ - удаление из текущего каталога файлов A.OBJ, B.OBJ и C.OBJ.

2.2.4. Команда: GOTO

Назначение: передача управления на строку КФ, следующую за заданной меткой.

Формат: GOTO label

Параметры: label - метка, на которую передается управление.

Пример:

GOTO END - переход на указанную метку.

2.2.5. Команда: IF

Назначение: условное выполнение команд.

Формат:

а) для проверки значения глобальной переменной ERRORLEVEL

IF [NOT] ERRORLEVEL value command

б) для сравнения двух символьных строк

#### IF [NOT] string1 = = string2 command

в) для проверки существования заданного файла

#### IF [NOT] EXIST [d:][path] filename command

Параметры:

value – целое число в диапазоне от 0 до 255;

command - выполняемая команда;

string1,string2 - текстовые строки.

Комментарий: если условие истинно, то выполняется указанная команда. Условие в первом формате истинно, если значение ERRORLEVEL больше или равно заданному значению value.

Примеры:

а) IF NOT EXIST FILE.DAT ECHO Файл не существует;

б) IF ERRORLEVEL 1 GOTO END - переход на метку END, если значение ERRORLEVEL больше или равно 1.

2.2.6. Команда: PAUSE

Назначение: реализация паузы при выполнении КФ.

Формат: PAUSE [comment]

Параметры: comment - текстовая строка.

Комментарий: по этой команде выполнение КФ приостанавливается и на экран выводится сообщение comment. Длина сообщения не может превышать 121 символ. Выполнение КФ возобновляется при нажатии на любую клавишу, кроме CTRL+C или CTRL+BREAK, которые завершают выполнение КФ.

Пример:

PAUSE Установите новый флэш-диск: - пауза с выдачей сообщения на экран.

2.2.7. Команда: REM

Назначение: ввод комментария.

Формат: REM [comment]

Параметры: comment - строка коммантария.

Комментарий: команда REM может использоваться для временного отказа от выполнения отдельных команд КФ. Если эхо-сопровождение включено, то строка comment выводится на экран, в противном случае – не выводится.

2.2.8. Команда: SHIFT

Назначение: сдвиг значений параметров командной строки запуска КФ на одну позицию влево.

##### Формат: SHIFT

Комментарий: используется при передаче данных в КФ с помощью замещаемых параметров. Например, после команды SHIFT в %0 будет находится первый параметр из командной строки, а в %9 - десятый параметр.

2.2.9. Команда CHOICE

Назначение: опрос пользователя и запись порядкового номера ответа из списка в переменную окружения ERRORLEVEL.

Формат: CHOICE [/C string] [/T n /D key] [/M text]

Параметры:

string – символьная строка, содержащая список возможных ответов пользователя;

key – вариант ответа, который будет выбран по умолчанию, если пользователь в течение n секунд не вводит свой вариант ответа;

text – строка символов, которая будет выведена на экран перед возможными вариантами ответа.

Комментарий: в Windows XP команда отсутствует, т.к. появилась возможность ввода значений в переменные окружения с помощью команды SET /P (см. лаборараторную работу № 1). В последующих версиях, включая Windows 10, команда присутствует.

Пример:

CHOICE /C ABCD /T 5 /D A /M “Сделайте Ваш выбор”

2.3 Передача данных в командный файл

Часто возникает необходимость создать один КФ, но выполнять его каждый раз с различными данными. Возможны два способа передачи данных в КФ: а) с помощью списка параметров, б) с помощью глобальных переменных.

При использовании первого способа в тексте КФ можно применить до 10 формальных параметров, имеющих имена %0…%9. Реальные значения этим параметрам присваиваются с помощью фактических параметров, задаваемых при запуске КФ через пробелы:

*имя\_ КФ параметр1 параметр2 ... параметр9*

При выполнении КФ формальный параметр %1 получает значение *параметр1*, формальный параметр %2 получает значение *параметр2* и т.д. В формальном параметре %0 хранится имя КФ.

Операционные системы Windows позволяют создавать КФ, у которых число фактических параметров превышает 10. Для этого необходимо воспользоваться командой SHIFT.

Символ “%” можно использовать в командах внутри КФ при указании имен файлов и каталогов. Для этого необходимо в тексте КФ повторить его дважды. Например, имя файла f%.exe должно выглядеть как f%%.exe.

При использовании второго способа (глобальных переменных) перед запуском КФ или во время его выполнения с помощью команды SET необходимо создать глобальные переменные и присвоить им заданные значения. Имена этих переменных при использовании в командах КФ заключаются в символы “%” (например, %FILE% ). При выполнении КФ операционная система извлекает соответствующие значения переменных из области окружения. Например, результатом исполнения следующей последовательности команд будет вывод на экран монитора содержимого файла, имя которого будет введено пользователем с клавиатуры.

SET /P MYFILE=Введите имя файла

TYPE %MYFILE%

**3. Порядок выполнения работы**

3.1 Создать файл с исходными данными согласно варианту задания из таблицы 1. Номер варианта выбирается по номеру студента из группового журнала, значения данных передаются с помощью параметров. Добавление записей в файл данных проводить с помощью разработанного командного файла WRITE\_FD.BAT со следующей программной функцией:

* отключить эхо-сопровождение;
* очистить экран;
* если не заданы параметры запуска КФ, то идти на метку HELP, иначе записать значения параметров запуска в файл данных и идти на конец;
* метка HELP;
* информировать пользователя о структуре и содержании полей записи в файле данных и синтаксисе команды, например:

WRITE\_FD ФИО ГОД\_РОЖД ПОЛ СРЕДНИЙ\_БАЛЛ

Таблица 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Предметная область** | **Наименование и номер начальной позиции полей записи** | **Поле для сортировки** | **Поиск** | **Способ опроса пользователя** |
| 1 | Студенческая группа. | фамилия -1, год рождения – 20, пол -25, средний балл - 30 | фамилия, по возрастанию | по году рождения | set |
| 2 | Магазин | код товара -1, наименование товара -5, группа товара – 20, цена- 30, количество - 37 | код товара | по группе товара | set |
| 3 | Результаты сессии | фамилия студента– 1, дисциплина – 15, оценка – 35, фамилия преподавателя - 40 | фамилия студента, по убыванию | по фамилии преподавателя | choice |
| 4 | Студенческая группа. | фамилия -1, год рождения – 20, пол -25, средний балл - 30 | год рождения, по возрастанию | по полу | choice |
| 5 | Магазин | код товара -1, наименование товара -5, группа товара – 20, цена- 30, количество - 37 | наименование товара, по возрастанию | по наименованию товара | choice |
| 6 | Результаты сессии | фамилия студента– 1, дисциплина – 15, оценка – 35, фамилия преподавателя - 40 | дисциплина, по возрастанию | по фамилии студента | choice |
| 7 | Студенческая группа. | фамилия -1, год рождения – 20, пол -25, средний балл - 30 | средний балл, по убыванию | по среднему баллу | set |
| 8 | Магазин | код товара -1, наименование товара -5, группа товара – 20, цена- 30, количество - 37 | наименование товара, по убыванию | по группе товара | set |
| 9 | Результаты сессии | фамилия студента– 1, дисциплина – 15, оценка – 35, фамилия преподавателя - 40 | фамилия преподавателя, по возрастанию | по фамилии студента | choice |
| 10 | Договорной отдел | Шифр договора (1), заказчик (10), источник финансирования (30), сумма договора (50), дата начала (60), дата окончания (70) | сумма договора, по возрастанию | по заказчику | choice |
| 11 | Студенческая группа. | фамилия -1, год рождения – 20, пол -25, средний балл - 30 | средний балл, по убыванию | по полу | choice |
| 12 | Туристическая фирма | Номер тура (1), страна (5), город (20), продолжительность тура (40), дата отправления (45). | дата отправления, по возрастанию | по стране | choice |
| 13 | Магазин | код товара -1, наименование товара -5, группа товара – 20, цена- 30, количество - 37 | цена, по возрастанию | по наименованию товара | set |
| 14 | Результаты сессии | фамилия студента– 1, дисциплина – 15, оценка – 35, фамилия преподавателя - 40 | оценка, по возрастанию | по фамилии преподавателя | choice |
| 15 | Магазин | код товара -1, наименование товара -5, группа товара – 20, цена- 30, количество - 37 | цена, по убыванию | по группе товара | choice |
| 16 | Результаты сессии | фамилия студента– 1, дисциплина – 15, оценка – 35, фамилия преподавателя - 40 | оценка, по убыванию | по фамилии преподавателя | set |
| 17 | Магазин | код товара -1, наименование товара -5, группа товара – 20, цена- 30, количество - 37 | количество, по возрастанию | по наименованию товара | set |
| 18 | Отдел кадров | Фамилия (1), год рождения (20), место рождения (25), должность (45), адрес, телефон (55) | год рождения, по возрастанию | по должности | choice |
| 19 | Отдел кадров | Фамилия (1), год рождения (20), место рождения (25), должность (45), адрес, телефон (55) | год рождения, по убыванию | по месту рождения | set |
| 20 | Расписание поездов | Номер поезда (1), пункт назначения (5), время отправления (30), время в пути (40). | время отправления, по возрастанию | по пункту назначения | Set |
| 21 | Магазин | код товара -1, наименование товара -5, группа товара – 20, цена- 30, количество - 37 | количество, по убыванию | по группе товара | choice |
| 22 | Бухгалтерия | Табельный номер (1), фамилия (7), оклад (30), надбавка (40), наличие налогового вычета (45), | оклад, по возрастанию | gо наличию налогового вычет | Set |
| 23 | Магазин | код товара -1, наименование товара -5, группа товара – 20, цена- 30, количество - 37 | код товара, по возрастанию | по наименованию товара | set |
| 24 | ГИБДД | Госномер автомобиля(1), модель (10), год выпуска (25), тип двигателя (30), адрес регистрации (40), дата регистрации (65) | год выпуска, по убыванию | gо модели | set |
| 25 | Магазин | код товара -1, наименование товара -5, группа товара – 20, цена- 30, количество - 37 | код товара, по убыванию | по группе товара | set |

3.2 Отладить КФ WRITE\_FD.BAT и ввести в файл данных 10 записей. Отредактировать файл данных с помощью текстового редактора для выравнивания полей у всех записей в соответствии с заданием, результат записать их в протокол.

3.3 Написать КФ FIND\_FD.BAT для поиска информации в файле данных по заданному полю:

* отключить эхо-сопровождение;
* очистить экран;
* если параметр запуска не задан, то идти на метку HELP, иначе искать записи в файле данных, содержащие заданную параметром запуска строку символов,и вывести их на экран,идти на конец;
* метка HELP
* информировать пользователя о синтаксисе запроса

Результат поиска должен представлять набор из нескольких записей.

3.4 Отладить КФ FIND\_FD.BAT. Реализовать несколько запросов на поиск информации в файле данных по различным полям записей, занести результаты в протокол.

3.5 Написать КФ SORT\_FD.BAT для сортировки файла данных в соответствии с вариантом задания:

* отключить эхо-сопровождение;
* очистить экран;
* провести информирование пользователя;
* сортировать записи файла данных, результат записать во временный файл;
* вывести на экран временный файл;
* удалить временный файл.

3.6 Отладить КФ и записать отсортированный файл данных в протокол.

3.7 Написать КФ OBRAB.BAT, со следующей программной функцией:

* отключить эхо-сопровождение;
* очистить экран;
* организовать с помощью команд CHOICE или SET /P (см. табл. 1) опрос пользователя через меню, через которое можно будет вызывать ранее разработанные файлы WRITE\_FD.BAT, SORT\_FD.DAT FIND\_FD.BAT, а также просматривать файл данных и завершать работу КФ;
* в зависимости от выбора пользователя вызвать требуемый КФ.

**4. Контрольные вопросы**

1 Отличие КФ от других исполняемых файлов.

2 Способы передачи данных в КФ.

3 Специальные команды КФ.

4 Циклическое исполнение команд в КФ.

5 Организация ветвлений в КФ.

# Лабораторная работа № 3. Прохождение программ в среде ОС

**1. Цель работы**

Целью работы является изучение двух вариантов технологического процесса обработки программы пользователя в среде операционной системы, а также изучение программы-библиотекаря.

**2. Методические указания**

2.1 Общие положения

Операционные системы всех современных ЭВМ предполагают наличие трех основных этапов обработки программ, написанных на любом алгоритмическом языке:

* трансляция, обеспечивающая перевод программы с алгоритмического языка на язык машинных команд (преобразова­ние исходного модуля в объектный);
* организация межпрограммных связей, позволяющая объединять независимо транслированные объектные модули в единый перемещаемый загрузочный модуль;
* загрузка в оперативную память (ОП) и выполнение.

Трансляция исходных модулей проводится с помощью специальных программ-компиляторов, которые обычно создают два файла: двоичный файл типа .obj, содержащий объектный модуль, и текстовый файл типа .lst , содержащий текст исходной программы и диагностические сообщения компилятора. Для каждого алгоритмического языка имеются собственные компиляторы.

На вход компилятора могут также подаваться так называемые файлы включения, имена которых указываются в исходном модуле с помощью ключевого слова INCLUDE. Эти файлы обычно имеют расширение .h и с точки зрения операционной системы представляют собой дополнительные исходные модули, без которых работа основной программы будет невозможной.

Компилятор включает в состав объектного модуля служебную информацию, необходимую для организации межпрограммных связей между независимо транслированными объектными модулями. Организация межпрограммных связей проводится с помощью программы, которая в различных операционных системах имеет различные названия: компоновщик, построитель задачи или редактор связей.

Компоновщик последовательно разрешает все внешние ссылки, получая информацию о них из словарей внешних символов (ESD) каждого объектного модуля, включаемого в формируемый загрузочный модуль. Процесс разрешения внешней ссылки заключается в поиске модуля с именем, указанным в ESD, и включении его в состав загрузочного модуля, причем поиск может проводиться только в файлах, указанных пользователем.

Объектные модули могут храниться в последовательных или в библиотечных файлах. Последовательный файл обычно содержит один объектный модуль и имеет тип .obj, библиотечный файл имеет тип .lib и представляет собой совокупность нескольких объектных модулей и оглавления. Доступ к отдельным модулям библиотеки проводится через оглавление, которое содержит имена всех модулей и их относительные адреса внутри библиотеки. Достоинством библиотечного способа хранения модулей является удобство выполнения групповых операций (копирование, пересылка), а также возможность обеспечения доступа ко всем модулям библиотеки после проведения однократной операции открытия файла. Для создания и ведения библиотечных файлов в состав операционной системы обычно включается специальная программа библиотекарь.

Загрузка модуля, сформированного компоновщиком, проводится в любой свободный участок ОП достаточного размера, поэтому перед загрузкой необходимо выполнить настройку всех его адресных констант по месту конкретного расположения в ОП. По способу настройки загрузочные модули Windows делятся на два типа: COM и EXE. Расширение имени файла, в котором хранится загрузочный модуль, совпадает с типом модуля. Модули типа COM являются самонастраивающимися, но имеют ограничения на предельный размер - 64Kбайта. Модули типа EXE не имеют ограничений по размеру, но для их настройки требуется выполнение определенных действий со стороны операционной системы.

Современные компиляторы после успешного формирования объектного модуля могут динамически вызывать компоновщик без дополнительных команд со стороны пользователя.

В настоящее время для разработки программного обеспечения часто используются интегрированные системы программирования, включающие транслятор, компоновщик, текстовый редактор, отладчик и другие программы. К ним относятся системы Паскаль, Си, Delphi, MS Visual Studio и др. Применение интегрированных систем позволяет скрыть от программиста многие детали, связанные с подготовкой программ к выполнению, и повысить производительность его труда.

2.2 Выполнение программ в среде операционной системы

В этом режиме управление процессом прохождения программы проводит непосредственно пользователь, используя соответствующие команды и ключи для вызова системных обрабатывающих программ. Командный режим используется в большинстве ОС (Windows, Linix и др.).

2.2.1 Управление компилятором

При выполнении лабораторной работы студентам предлагается использовать алгоритмический язык Си или С++, поэтому рассмотрим кратко основные возможности компилятора из интегрированной среды Borland C++Builder. Компилятор хранится в файле с именем bcc32.exe и вызывается с помощью команды:

*bcc32 [ключ1 ключ2...] имя\_файла1 имя\_файла2...*

где *ключ* - последовательность символов, начинающаяся с символа "-" и указывающая режим работы компилятора;

*имя\_файла*  - спецификация обрабатываемого файла в соответствии с соглашениями Windows.

Обработка файлов компилятором проводится в зависимости от их типов по следующим правилам:

* .asm - вызывается транслятор ассемблера MASM для перевода файла в объектный код;
* .obj и .lib - подключаются к формируемому загрузочному модулю на этапе компоновки как отдельный объектный модуль или как библиотека объектных модулей;
* .c и .cpp - компилируется в объектный или загрузочный модуль в зависимости от указанного пользователем режима.

Компилятор Си имеет большое количество управляющих ключей, которые можно разделить на следующие группы: ключи управления компиляцией, ключи обработки исходного текста программы, ключи сообщений об ошибках, ключи программы, ключи моделей памяти, ключи процессора и ключи сегментов памяти.

Ключи записываются в командной строке в произвольном порядке. Размер командной строки ограничен, поэтому при необходимости задания большого числа ключей можно использовать файл конфигурации bcc32.cfg, куда компилятор обращается по умолчанию. Приоритет ключей, указанных в командной строке, выше, чем у ключей файла, поэтому пользователь всегда может отменить режимы, указанные в файле. Отключение режима проводится путем указания ключа с дополнительным символом "-" справа.

В большинстве случаев пользователь может не задавать ключи, используя значения, установленные по умолчанию. Для выполнения лабораторной работы могут потребоваться следующие ключи управления компиляцией:

-с - обработать исходную программу без компоновки загрузочного модуля;

-Ipath – указывает путь path к include-файлам (типа .h).

-Lpath – указывает путь path к библиотечным файлам (типа .lib).

2.2.2 Управление компоновщиком

При выполнении лабораторной работы предполагается использование компоновщика из среды Borland C++Builder, вызываемого в виде отдельной самостоятельной программы ilink32.

Компоновщик поддерживает два метода компоновки - статический и динамический. Статическая компоновка проводится с использованием библиотечных файлов типа .lib путем включения в состав загрузочного модуля всех необходимых модулей из библиотечных файлов. Динамическая компоновка основана на том, что в состав загрузочного модуля включаются только ссылки на библиотечные модули, а непосредственное их извлечение из библиотек и загрузка в оперативную память проводятся на этапе выполнения программы. Для этого метода можно использовать только специальные библиотеки динамической компоновки (Dinamic Link Library), находящиеся в файлах типа .dll.

На вход компоновщика, кроме объектных и библиотечных файлов, необходимо подключить файл определения загрузочного модуля (.def) и файл ресурсов (.res). Файл определения модуля задает некоторые характеристики формируемого загрузочного модуля (например, можно ли перемещать программу в памяти или запускать несколько копий программы одновременно), файл ресурсов описывает ресурсы, включаемые в загрузочный модуль (к ресурсам относятся курсоры, диалоги и иконки). В подавляющем большинстве случаев достаточно использовать стандартные значения этих параметров, задаваемые по умолчанию, и не указывать имена этих файлов.

Компоновщик хранится в файле ilink32.exe и вызывается с помощью команды, имеющей следующий вид:

*ilink32 [ключи] startup objfiles, [exefile], [mapfile], [libraries], [deffile], [resfile]*

#### Здесь обозначено:

*ключи* – список ключей, управляющих режимом компоновки; перед каждым ключом должен быть символы “/” или “-”;

*startup* – модуль инициализации, спецификация которого определяется типом формируемого исполняемого модуля; например

c0x32.obj - для формирования консольного Windows приложения;

c0x32w.obj - для формирования консольного Windows приложения, использующего кодировку символов Unicode;

c0w32.obj - для формирования графического Windows приложения;

c0d32.obj - для формирования DLL библиотеки;

c0pkg32.obj - для формирования модуля компонента, включаемого в среду программирования.

*objfiles* – список спецификаций объектных модулей, разделенный пробелами;

*exefile* – спецификация исполняемого файла (.exe или .dll);

*mapfile* – спецификация текстового файла, в который будет записана карта памяти загрузочного модуля, содержащая информацию об относительных адресах всех объектных модулей, включенных в состав загрузочного модуля, и сообщения об ошибках, возникших на этапе компоновки (тип файла .map);

*libraries* - список спецификаций библиотечных файлов, разделенный пробелами; спецификации библиотек должны быть расположены в определенном порядке:

1. системные библиотеки (например, для выполнения простых однопоточных приложений необходимо подключить библиотеку cw32.lib );
2. список личных библиотек;
3. IMPORT32.LIB - библиотека импорта функций Windows API;
4. математические библиотеки (если они необходимы).

*deffile* - спецификация файла определения модуля (.DEF), в случае отсутствия этой спецификации компоновщик задает свойства по умолчанию;

*resfile* - спецификация файла ресурсов (.RES), в случае отсутствия этой спецификации компоновщик включает в исполняемый файл стандартные ресурсы Windows.

Для управления формированием карты памяти можно использовать следующие ключи:

/m - создать карту памяти;

/x - не создавать карту памяти;

/s - создать подробную карту памяти;

/t - создать файл типа .com;

/Lpath – указывает путь path к библиотечным файлам (типа .LIB).

Обратите внимание: ключи компилятора и компоновщика являются регистрозависимыми, т.е. значения –s и -S являются различными ключами !

Команда задает обязательный порядок записи специфика­ций файлов, если какая-либо из спецификаций опускается, то порядок не должен изменяться. Например, если не требуется создание карты памяти, то обращение к компоновщику будет следующим:

*ilink32 [ключи] startup objfiles, [exefile],, [libraries], [deffile], [resfile]*

Если не указана спецификация exefile,то сформированный загрузочный модуль будет записан в файл, имя которого совпадает с именем первого из перечисленных в командной строке объектных модулей.

2.2.3 Программа-библиотекарь

Программа tlib предназначена для создания и ведения библиотек объектных модулей, наличие которых позволяет значительно облегчить работу пользователя при создании больших программных систем. Командная строка для обращения к библиотекарю имеет следующий вид:

*tlib имя\_библиотеки операция,[имя\_файла]*

где *имя\_библиотеки* - спецификация библиотечного файла в соответствии с соглашениями Windows;

*операция* - вид выполняемой операции;

*имя\_файла* - спецификация файла, в который будет записано оглавление библиотеки.

При работе с библиотекой возможно выполнение следующих операций:

+mn (добавление модуля);

-mn (удаление модуля);

\*mn (извлечение модуля из библиотеки без его удаления);

-+mn (замена модуля);

\*-mn (извлечение модуля из библиотеки с его удалением).

Здесь "mn" - спецификация файла, содержащего модуль, с которым проводится работа. Допускается выполнение операций с несколькими модулями, при этом спецификации файлов разделяются символом "пробел".

Программа tlib является 16-разрядной и на современных 64-разрядных версиях Windows может не запускаться даже при установленном режиме принудительного запуска в более ранних версиях Windows.

2.2.4 Примеры работы в среде Windows

Приведем пример командного файла, в котором проводится раздельная компиляция, компоновка и выполнение программы, состоящей из трех модулей (main.cpp, mod1.cpp и mod2.cpp) при компоновке необходимо сформировать подробную карту памяти:

*bcc32 -c –I.\include main.cpp*

*bcc32 -c –I.\include mod1. cpp*

*bcc32 -c –I.\include mod2. cpp*

*ilink32 -s –L.\lib c0x32 main mod1 mod2, main, main,cw32 import32*

*main*

Здесь с помощью ключей –I и –L указываются относительные пути доступа к файлам включения (типа .h) и файлам библиотек (типа .lib). Эти пути необходимо указать, если на Вашем компьютере не установлена интегрированная система программирования С++ Builder. Второй способ указания путей – с помощью внесения изменений в файлы конфигурации компилятора (bcc32.cfg) и компоновщика (ilink32.cfg).

Для формирования личной библиотеки объектных модулей можно использовать библиотекарь tlib.

Рассмотрим порядок работы пользователя при подготовке к выполнению консольной программы сложной структуры, состоящей из головной программы, записанной в файле main.c, и двух вызываемых подпрограмм-функций, записанных в файлах mod1.c и mod2.c соответственно. Все файлы содержат программы, написанные на Си или С++, являются исходными модулями.

Возможны два варианта подготовки программы - без использования личной библиотеки объектных модулей и с ее использованием. Первый вариант предполагает выполнение следующих этапов:

* независимая трансляция каждого исходного модуля в отдельный объектный модуль;
* компоновка загрузочного модуля;
* выполнение загрузочного модуля.

В соответствии с этим пользователь должен набрать следующую последовательность команд:

*bcc32 -c –I.\include main.cpp*

*bcc32 -c –I.\include mod1. cpp*

*bcc32 -c –I.\include mod2. cpp*

*ilink32 -s –L.\lib c0x32 main mod1 mod2, main,main,cw32 import32*

*main*

Здесь при компоновке загрузочный модуль будет записан в файл main.exe, а карта памяти - в файл main.map.

Второй вариант требует выполнения следующих этапов:

* независимая трансляция исходных модулей;
* создание библиотеки и запись в нее объектных модулей;
* компоновка загрузочного модуля с использованием библиотеки;
* выполнение загрузочного модуля.

В этом случае команды пользователя должны иметь следующий вид:

*bcc32 -c –I.\include main.cpp*

*bcc32 -c –I.\include mod1. cpp*

*bcc32 -c –I.\include mod2. Cpp*

*tlib mylib +mod1.obj +mod2.obj*

*ilink32 -s –L.\lib c0x32 main, main,main,cw32 mylib import32*

*main*

Приведенные примеры позволяют детально ознакомиться с процессом подготовки программы к выполнению. Пользователь, конечно, может использовать возможности совмещения этапов трансляции и компоновки, заложенные в компилятор. В этом случае последовательность команд будет следующей:

*bcc32 main mod1 mod2*

*main*

После выполнения первой команды будет сформирован загрузочный модуль, записанный в файл main.exe. Вторая команда обеспечивает загрузку программы в ОП и ее выполнение.

**3. Порядок выполнения работы**

3.1 Изучить исходные тексты программы, состоящей из трех функций, каждая из которых хранится в отдельном файле (a.cpp, b.cpp, c.cpp).

**a.cpp:**

#include <stdio.h>

void b(void),c(void);

void main(void){

printf("Моя фамилия - Петров\n");

b();

c();

}

**b.cpp:**

#include <stdio.h>

void b(void){

printf("Имя - Петр\n");

}

**c.cpp:**

#include <stdio.h>

void c(void){

printf("Отчество - Петрович !\n");

}

3.2 Отредактировать функции таким образом, чтобы на экран выводились фамилия, имя и отчетство студента, выполняющего лабораторную работу, а также текущая дата.

3.3 Разработать командный файл, реализующий сборку и выполнение программы без использования отдельной команды вызова компоновщика.

3.4 Разработать командный файл, реализующий сборку и выполнение программы с вызовом компоновщика отдельной командой. При сборке программы задать режим формирования подробной карты памяти.

3.5 Разработать командный файл, реализующий сборку и выполнение программы с применением библиотекаря, предварительно разместив откомпилированные модули b.cpp и c.cpp в личной библиотеке объектных модулей.

3.6 На основе анализа карты памяти определить размеры машинного кода функций a.cpp, b.cpp, c.cpp, заполнить следующую таблицу и пояснить результаты сравнения.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя функции** | **Размер исходного модуля, байт** | **Размер объектного модуля, байт** | **Размер машинного кода, байт** |
| a.cpp |  |  |  |
| b.cpp |  |  |  |
| c.cpp |  |  |  |

**4. Контрольные вопросы.**

1. Основные этапы прохождения программ в среде ОС.

2. Библиотечный файл: назначение, организация, достоинства и недостатки.

3. Типы загрузочных модулей.

4. Компилятор: назначение, способы управления.

5. Компоновщик: назначение, способы управления.

6. Библиотекарь: назначение, способы управление.

7. Карта памяти: назначение и структура.

# Лабораторная работа № 4. Командный язык операционной системы Linux

**1. Цель работы**

Целью работы является изучение командного интерфейса и особенностей файловой системы ОС Linux.

**2. Общие сведения о файловой системе Linux**

Основным отличием ОС Linux от Windows с точки зрения пользователя является организация файловой системы. В Linux файловая система представляет собой единую иерархическую структуру, отображаемую в виде дерева, корню которого соответствует основной (корневой) каталог, а листьям – файлы. Корневой каталог имеет имя «/» и содержит системные файлы и подкаталоги.

Под файлом в Linux понимается не только поименованная совокупность информации на ВЗУ, но и любое устройство, которое может хранить, поставлять или потреблять информацию. При этом устройство подключается (монтируется) к существующему дереву файловой системы в указанной пользователем точке с помощью команды **mount,** после чего пользователь может обращаться к любым доступным файлам, при этом в имени никак не отражается имя устройства, на котором файл находится или создается.

Linux поддерживает следующие типы файлов: *обычные, каталоги, каналы, блочные специальные, символьные специальные, ссылки.* Понятие обычного файла в Linux и Windows близки, но в Linux отсутствует деление на текстовые и бинарные файлы, т.е. все обычные файлы представляются в виде последовательность байтов без какой-либо дополнительной структуры.

Любой файл в файловой системе Linux имеет так называемый индексный дескриптор, который и хранит всю необходимую информацию о файле. Для каждого файла номер индексного дескриптора уникальный, а имя файла является всего лишь ссылкой на данный дескриптор. Каталоги в Linux устанавливают соответствие между именем и номером дескриптора файла. Возможна ситуация, когда несколько элементов каталога ссылаются на один номер дескриптора, в этом случае мы будем иметь несколько ссылок на один и тот же файл, т.е. в Linux имеется возможность присвоить различные имена одному файлу.

*Ссылки* могут быть жесткими и мягкими (символьными). Жесткая ссылка может создаваться только для файлов и ссылается на номер дескриптора оригинального файла. Такая ссылка используется в пределах одной файловой системы, т.к. уникальность номеров дескрипторов обеспечивается только в пределах отдельной файловой системы. При удалении жесткой ссылки или оригинального файла счетчик ссылок в дескрипторе уменьшается на единицу, а реальное удаление файла с диска проводится только тогда, когда удаляется последняя ссылка, и счетчик ссылок становится равным нулю. Восстановление удаленного файла в Linux невозможно.

Мягкие ссылки могут указывать на объекты разных файловых систем (файлы и каталоги), т.к. ссылаются не на дескрипторы, а на имена этих объектов. Удаление мягкой ссылки не приводит к удалению объекта, на которую указывает эта ссылка. Удаление оригинального объекта не приводит к автоматическому удалению мягких ссылок, но делает такую ссылку неработоспособной.

*Канал* - это временный файл ограниченного размера, информация из которого после чтения исчезает. Каналы используются для организации обмена данными между различными процессами.

*Специальные файлы* - соответствуют устройствам (блочные - дискам, символьные - всем прочим). Введение этого понятия позволяет единообразно организовать обмен информацией с любым источником или приемником информации. Например, для чтения флэш-накопителя достаточно обычным образом открыть файл /dev/usb, позиционироваться в нем и прочесть нужное число блоков. Т.е. прикладные программы одинаково обмениваются информацией с обычным файлом, каналом или устройством.

Полное имя показывает местоположение файла или каталога в файловой системе. Существует два типа имени: *абсолютное* и *относительное*. Абсолютное всегда начинается с символа «/», обозначающего корневой каталог. Кроме того, этот символ используется и в абсолютном, и в относительном имени в качестве разделителя имен каталогов и файлов (например, /u/home/bpi). Абсолютное имя показывает положение файла по отношению ко всей иерархической структуре. Относительное имя показывает положение файла по отношению к текущему или домашнему каталогу.

Имена каталогов и файлов в Linux являются регистрозависимыми, т.е. имена newfile и NEWFILE обозначают разные файлы. Если имя файла начинается с символа «.» (точка), такой файл называется скрытым и его характеристики не будут выводиться при просмотре содержимого каталогов.

Каждый файл имеет права доступа, которые определяют,кто и что может делать с содержимым файла. Возможны три категории пользователей файла - владелец (u), группа пользователей, в которую входит владелец (g), остальные пользователи (o), и три типа права доступа - чтение (r) , запись (w), выполнение (x).

Права доступа хранятся в дескрипторе файла в виде строки из 10 символов. Первый символ строки обозначает тип файла (дефис – это обычный файл, d – каталог, l –мягкая ссылка, b – блочный специальный файл, c – символьный специальный файл). Далее в строке идут три группы, состоящие из трех символов каждая: первая группа определяет права владельца файла, вторая – права группы пользователей, третья - права остальных пользователей. В каждой из этих групп первый символ определяет право доступа для чтения, второй – для записи, третий – для выполнения файла. Если в какой – либо позиции в этих группах выводится символ «-» (дефис), то соответствующее право доступа отключено. В таблице 3 представлена система определения прав доступа.

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Право** | **Обозначение** | **Файл** | **Каталог** |
| Чтение | r | Файл можно посмотреть и скопировать | Можно посмотреть список входящих файлов |
| Запись | w | Файл можно изменить и переименовать | Можно создавать и удалять файлы |
| Выполнение | x | Файл можно запустить на выполнение (скрипты и программы) | Можно входить, делать текущим |
| Выполнение от имени владельца | s | Файл можно запустить на выполнение с правами его владельца | Все объекты, создаваемые внутри каталога, наследуют его группу прав доступа |

Например,

а) **-rw-r--r--**

В приведенном примере владелец имеет право читать и изменять файл, члены группы и остальные пользователи могут только читать файл,

б) **-rwx- - - - - -** только владелец файла имеет право читать, изменять и выполнять файл.

в) **drwxr-x--x** владелец может просматривать, изменять и входить в каталог, члены группы могут входить и просматривать его, все остальные - только входить.

**3. Общие сведения о командах Linux**

Обратите внимание:

* в командах при указании имен файлов можно использовать метасимволы (\* - заменяет любое количество символов (в том числе ни одного), ? - заменяет любой одиночный символ, [список символов] - заменяет одиночный символ из указанного списка или диапазона символов);
* в одной командной строке можно записывать несколько команд, разделяя их символом «;», при этом они будут выполняться последовательно;
* все ключи в командной строке Linux являются регистрозависимыми;
* некоторые команды могут не поддерживаться в Вашей версии командного процессора;

3.2 Команды для работы с каталогами

3.2.1 Команда: **mkdir**

Назначение: создать каталог

Формат: mkdir имя\_каталога1 [имя\_каталога2...]

Пример: mkdir abc mykat

Комментарий: для того, чтобы создать каталог, должны быть правильно установлены права доступа.

3.2.2 Команда **ls.**

Назначение: просмотр каталога.

Формат: ls [-опции] [путь]

Пример**:** ls abc

Комментарий: Чтобы увидеть имена скрытых файлов, используйте опцию **а**. Для получения информации о типах файлов (каталог, исполняемый файл, ссылка), используйте опцию **F**. При использовании этой опции в поле имени выводится символ, который определяет тип файла (табл. 4).

Для получения подробной информации о файлах и каталогах используйте опцию ***l*.** При этом о каждом файле и каталоге вы получите следующую информацию: тип файла, права доступа, число ссылок, имя владельца, имя группы, размер, дата последнего изменения, имя файла или каталога.

Таблица 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Тип файла** | Каталог | Исполняемый файл | Мягкая ссылка | Обычный файл |
| **Символ** | / | \* | @ | отсутствует |

3.2.3 Команда **cd.**

Назначение: переход в указанный каталог.

Формат: cd [имя\_каталога]

Пример: cd /u/home/bpi

Комментарий: для перехода в домашний каталог используйте команду cd без параметров. Для перемещения по файловой системе можно использовать сокращенные имена каталогов, приведенные в табл. 5.

Таблица 5

|  |  |
| --- | --- |
| **Символ** | **Значение** |
| **~** | Домашний каталог |
| **.** | Текущий каталог |
| **..** | Родительский каталог |

3.2.4 Команда  **pwd**.

Назначение: вывод абсолютного имени текущего каталога.

Пример**:** pwd

3.2.5 Команда  **rmdir**.

Назначение: удаление каталогов.

Формат: rmdir [-опции] имя\_каталога

Пример: rmdir kat1 kat 2

Комментарий: для удаления каталога, содержащего файлы, используйте опцию r, без указания этой опции команда не будет выполняться. Для этой цели можно также применить команду rm –rf имя\_каталога.

3.3 Команды для работы с файлами

3.3.1 Команда  **cat**.

Назначение: просмотр текстовых файлов.

Формат: cat имя\_файла

Пример: cat file1

Комментарий: для просмотра больших файлов используйте команду **more**, так как она позволяет осуществлять постраничный просмотр файлов (страница соответствует размеру экрана).

3.3.2 Команда  **more**.

Назначение: постраничный просмотр текстовых файлов.

Формат: more имя\_файла

Пример: more /etc/passwd

3.3.3 Команда  **head**.

Назначение: просмотр указанного числа начальных строк файла.

Формат: head [-n] имя\_файла.

Параметр: n - количество выводимых строк (по умолчанию выводится 10 строк)

Пример: head /etc/passwd

3.3.4 Команда  **tail**.

Назначение: просмотр указанного числа конечных строк файла .

Формат: tail [-n] имя\_файла

Параметр: n - количество выводимых строк (по умолчанию выводится 10 строк)

Пример: tail /etc/passwd

3.3.5 Команда  **cp**.

Назначение: копирование файлов и каталогов.

Формат: cp [-опции] исходный\_файл целевой\_файл

Примеры:

1. cp abc1 ./tmp/November – копирование файла;

2. cp –a kat\_1 kat\_2 – копирование каталога kat\_1 в каталог kat\_2.

Комментарий: Команда **cp** с опцией **r**  позволяет копировать каталоги вместе с входящими в них файлами и каталогами.

3.3.6 Команда  **mv**.

Назначение: перемещение и переименование файлов и каталогов.

Формат: mv [-опции] старое\_имя новое\_имя

Пример: mv file1 file2

3.2.7 Команда  **rm**.

Назначение: удаление файлов.

Формат: rm [-опции] имя\_файла

Пример: rm file1 file2

Комментарий: если Вы хотите, чтобы команда запрашивала подтверждение на удаление файла, то используйте опцию i

3.3.8 Команда  **touch**.

Назначение: создание пустого файла.

Формат: touch [-опции] имя\_файла

Пример: touch myfile

3.3.9 Команда  **chmod**.

Назначение: изменение прав доступа к файлу или каталогу. Права доступа к файлу может изменить только владелец или суперпользователь (администратор).

Формат: chmod режим имя\_файла

Здесь «режим» имеет следующую структуру и способ записи:

**КТО ДЕЙСТВИЯ ПРАВА**

**=** установить право

**-**  лишить права

**+** дать право

**u (user)** - владелец файла

**g (group)** - группа, к которой

принадлежит

владелец файла

**o (others)** - все остальные

**r** - чтение

**w** - запись

**x** - выполнение

Права доступа могут быть заданы в команде не только в символьном виде, но и в цифровой форме (восьмеричное значение). Связь между цифровой и символьной формами приведена в табл. 6.

Таблица 6

|  |  |  |
| --- | --- | --- |
| **ЦИФРОВАЯ ФОРМА**  двоичная восьмеричная | | **СИМВОЛЬНАЯ ФОРМА** |
| 111  110  101  100  011  010  001  000 | 7  6  5  4  3  2  1  0 | rwx  rw-  r-x  r--  -wx  -w-  - -x  - - - |

Примеры:

1. chmod g-x june - лишить членов группы права на выполнение файла ./june;

2. chmod 774 october – дать все права доступа к файлу ./october владельцу и членам группы и право чтения для остальных пользователей системы;

3. chmod g+rw,o+r april или chmod 764 april ***–***добавить права чтения и записи в файл ./april членам группы пользователей и права чтения для всех остальных пользователей.

3.3.10 Команда  **find** .

Назначение: используется для поиска и вывода имен файлов, соответствующих заданной строке символов.

Формат: find начальная\_точка\_поиска [-опции],

где «начальная\_точка\_поиска» определяет каталог, начиная с которого по всем подкаталогам будет вестись поиск.

Примеры:

1. Вывести на экран имена файлов из домашнего каталога и его подкаталогов, начинающихся на f:

find ~ -name "f\*" –print, где

-name - после этой опции указывается имя файла, который нужно найти,

"f\*" - строка символов, определяющая имя файла,

-print - опция, задающая вывод результатов поиска на экран.

2. Вывести на экран имена файлов в каталоге /etc, начинающихся с символа p:

find /etc -name "p\*" –print

Комментарий: команда пытается обратиться к каждому из найденных файлов, поэтому в случае отсутствия соответствующих прав доступа на экране рядом с именем найденного файла будет выводиться сообщение об отсутствии прав доступа. Для того, чтобы отфильтровать сообщения об таких ошибках, необходимо направить выходной поток ошибок на фиктивное устройство **null**:

find /etc -name "p\*" –print 2>null

3.3.11 Команда **wc**.

Назначение: вывод числа символов или строк в файле.

Формат: wc [-опции] имя\_файла

Примеры:

1. wc –c file1 - выводит число символов в файле file1,

2. wc –l file2 - выводит число строк в файле file1.

3.3.12 Команда **ln**.

Назначение: создание ссылки на файл.

Формат: ln [-опции] имя\_файла имя\_ссылки

Пример: ln folder1/file1 intro- создает в текущем каталоге жесткую ссылку с именем intro на файл folder1/file1. Для создания мягкой ссылки необходимо использовать опцию –s.

3.3.13 Команда **mount**.

Назначение: подключение (монтирование) нового устройства к файловой системе.

Формат: mount [-опции] имя\_файла имя\_ссылки

Пример: mount –t vfat /dev/fd0 /tc – монтирует файловую систему из раздела fd0 в каталог /tc

3.4 Команды для управления сеансом работы пользователя

3.4.1 Команда **who**.

Назначение: - вывод списка активных пользователей.

Формат: who

Комментарий: для вывода имени текущего пользователя существует команда **whoami.**

3.4.2 Команда **uname**.

Назначение: вывод информации о версии операционной системы.

Формат: uname

3.4.3 Команда **type**.

Назначение: определение типа команды (внутренняя или внешняя).

Формат: type команда

Пример: type find

Комментарий: для внутренних команд на экран выводится сообщение о том, что команда является встроенной в командный процессор, для внешних - имя каталога, в котором находится соответствующая программа**.**

3.4.4 Команда **passwd**.

Назначение: изменение пароля.

Формат: passwd

Комментарий: после ввода команды необходимо ввести старый пароль (old pasword) и дважды ввести новый пароль (new password).

3.4.5 Команда **exit**.

Назначение: завершение сеанса работы с командным интерпретатором.

Формат: exit

**4. Программное обеспечение для выполнения работы**

В подавляющем большинстве случаев на компьютерах студентов установлена ОС Windows, поэтому для выполнения лабораторной работы возможны два варианта:

* установить Linux в отдельный раздел жестого диска и мультизагрузчик, позволяющий загружать Windows или Linux по выбору пользователя;
* установить в Windows среду виртуализации (например, Oracle Virtual Box, VMWare Workstation и т.д.), на которую в качестве гостевой ОС установить Linux.

Рекомендуется использовать второй вариант, причем в качестве среды виртуализации удобнее применить Oracle Virtual Box, получив ее дистрибутив бесплатно на сайте разработчика по адресу

<https://virtualbox.download-windows.org/> .

Для выполнения лабораторной работы можно использовать любую версию Linux: Ubuntu, ArchLinux, Fedora, TinyCore Linux и др. Рекомендуется установить TinyCore Linux по следующим причинам:

* система имеет небольшой размер (в минимальной конфигурации 10 Мбайт, в максимальной - 86 Мбайт);
* может работать на системах с 48 Мб ОЗУ;
* может обмениваться файлами с файловой системой Windows через общие папки;
* использует в качестве интерпретатора команд компактное приложение BusyBox, представляющее собой набор UNIX-утилит командной строки и поставляемое в виде одного исполняемого файла;
* имеет кросс – платформенную библиотеку инструментов для разработки графического интерфейса пользователя;
* распространяется бесплатно.

Дистрибутивы TinyCore различных конфигураций можно скачать с сайта разработчиков по адресу <http://tinycorelinux.net/downloads.html>. Перечень команд, включенных в BusyBox, можно посмотреть с помощью команды **busybox**, справку по любой команде можно получить, указав для нее опцию -?. Порядок установки максимальной конфигурации TinyCore Plus приведен в приложении 1.

Создание виртуальной машины разбивается на два этапа:

* распаковка дистрибутива TinyCore Plus из файла образа диска *Core Plus-5.3.iso* и загрузка его в ОЗУ;
* установка виртуальной машины с возможностью загрузки с жесткого диска.

После завершения первого этапа возможно выполнение всех команд BusyBox, но при этом все создаваемые файлы размещаются в ОЗУ и после завершения работы не сохраняются. На втором этапе с помощью инсталлятора TC\_Install на виртуальном жестком диске создается файловая система и возможно полноценное сохранение всех созданных файлов.

**5. Порядок выполнения работы**

1. Ознакомиться с инструкцией по созданию машины с операционной системой TinyCore Plus в среде виртуализации Oracle VM Virtual Box.

2. Создать виртуальную машину следующей конфигурации (см. приложение 1): имя машины – TinyCore Linux; объем оперативной памяти – 256 Мбайт; объем виртуального жесткого диска – 1 Гбайт; тип виртуального жесткого диска – динамический, тип файловой системы – ext4.

3. Войти в систему, определить имена текущего пользователя и текущего каталога.

4. Создать в домашнем каталоге структуру каталогов, показанную на рис.1. В именах каталогов вместо символа «Х» указать свой номер из группового журнала.

домашний каталог

katX\_1

katX\_2

katX\_3

ab1

ab2

mmm

fon1

fon2

fon3

Рис.1

5. Перейдите в каталог katX\_3.

5.1. Посмотрите содержимое этого каталога, используя в команде **ls** различные опции.

5.2. Проверьте, существует ли в корневом каталоге файл с именем *unix*.

5.3. Существует ли в каталоге */var/spool* подкаталог с именем *cron*?

5.4. Посмотрите содержимое вашего домашнего каталога. Кто является владельцем его файлов и подкаталогов?

6. Введите команды для выполнения следующих действий:

6.1. Создайте новый каталог с именем *newdir* в вашем домашнем каталоге.

6.2. Создайте новый каталог с именем *fun* в каталоге *~/newdir*.

6.3. Переместите каталог *newdir* в каталог *ab1*.

6.4. Создайте три новых каталога с именами *one*, *two*, *three* в вашем домашнем каталоге одной командой.

6.4. Удалите эти каталоги одной командой.

6.5. Попробуйте удалить каталог *~/katX\_2* командой **rmdir**. Что получилось? Определите, какой ключ необходимо задать, чтобы этот каталог был удален.

7. С помощью справки определите, какая опция команды **ls** позволяет просматривать не только содержимое указанного каталога, но и подкаталогов, входящих в него.

8. Определите при помощи справки, какой набор опций команды **ls** позволяет отсортировать выводимый список с развернутым описанием файлов, по времени последнего изменения.

9. Посмотрите справку по следующим командам: cd, pwd, mkdir, rmdir, rm.

10. Создайте в домашнем каталоге указанные ниже подкаталоги и файлы и с помощью команды **chmod** задайтеуказанные права доступа, считая, что по умолчанию таких прав нет:

*drwxr--r-- ... Africa*

*drwx--x--x ... Asia*

*-r-xr--r-- ... Russia*

*-rw-rw-r-- ... France*

11. Лишите владельца файла *./France* права на чтение.

12. Что произойдет, если вы попытаетесь просмотреть файл .*/France* командой **cat**?

13. Что произойдет, если вы попытаетесь скопировать файл *./France*?

14. Дайте владельцу файла *./France* право на чтение и выполните п. 11 и 12

15. Лишите владельца каталога *./Asia* права на выполнение.

16. Перейдите в каталог *./Asia*. Что произошло?

17. Дайте владельцу каталога *./Asia* право на выполнение и выполните п. 15.

18. Создайте в домашнем каталоге ссылку с именем *mirror* на файл *Russia* и с помощью команды **ls –l** убедитесь, что ссылка создана.

19. Выведите на экран постранично имена файлов из каталога */etc*, начинающиеся с символа «h».

20. Используя команду **history** посмотрите содержимое буфера команд.

21. Выведите на экран и проанализируйте содержимое каталога */bin*, определите типы файлов.

# Лабораторная работа № 5. Командные сценарии Linux

**1. Цель работы**

Целью работы является изучение языка командных сценариев ОС Linux.

**2. Методические указания**

2.1 Общие сведения

Командный сценарий (shell script, командный файл) – это исполняемый текстовый файл, состоящий из команд ОС, исполняемых командным интерпретатором. Сценарий часто называют shell–программой и используют для администрирования и управления ОС. Командный язык Linux, в отличие от языка команд Windows, является языком высокого уровня, поддерживающим ввод-вывод переменных, ветвления (if-then-else, case), циклы (for, while, until), работу с функциями, пошаговый режим отладки и т.д.

В дистрибутивы Linux обычно включаются несколько командных интерпретаторов, список которых можно посмотреть в файле /etc/shells. В ОС TinyCore входят три командных интерпретатора: BusyBox, sh, ash. Физически два последних интерпретатора являются встроенными модулями библиотеки BusyBox.

Запуск сценария может проводиться несколькими способами:

* запустить интерпретатор, указав ему в качестве аргумента имя файла сценария, например: ***sh myscript.sh***
* запустить интерпретатор, связав стандартный поток ввода с именем файла сценария, например ***sh –s < myscript.sh***
* выполнить сценарий в текущем экземпляре shell командой «точка», например: ***. myscript.sh***
* установить для сценария файловый атрибут «исполняемый» и запустить как обычную программу: ***chmod 755 myscript.sh; ./myscript.sh***

При написании сценария необходимо соблюдать следующие требования:

* + - в первой строке обычно указывается имя интерпретатора, который будет исполнять сценарий, например *#!/bin/sh*
* в конце сценария обязательно вводится символ перевода строки (нажатие ENTER).

2.2 Переменные сценария

Каждый запущенный экземпляр командного интерпретатора shell имеет определенный набор стандартных глобальных переменных, значения которых можно просмотреть командами **env** или **printenv**. В таблице 7 приведены назначения некоторых переменных.

Пользователь может использовать в сценариях как глобальные переменные интерпретатора, так и собственные локальные переменные. Имена переменных, используемых в сценариях, представляют собой последовательность букв, цифр и символов подчеркивания. Имя должно начинаться с буквы или символа подчеркивания.

Таблица 7

|  |  |
| --- | --- |
| **Переменная** | **Значение** |
| ? | Код завершения последней команды (0 – нормальное завершение) |
| $ | PID текущего процесса shell |
| ! | PID последнего фонового процесса |
| # | Число параметров, переданных в shell |
| \* | Список параметров shell в виде одной строки |
| @ | Список параметров shell в виде набора слов |
| - | Флаги, передаваемые в shell |
| HOME | Имя домашнего каталога |
| PATH | Пути поиска исполняемых файлов |
| PS1 | Вид системной подсказки |
| SHELL | Имя каталога, в котором зарегистрирован командный интерпретатор |
| USER | Имя учетной записи текущего пользователя |

Переменные задаются сразу со значением (возможно с пустым) и всегда имеют строковый тип. При объявлении переменной ее имя и значение должны быть записаны без пробелов относительно символа равенства. Например, создадим две переменных, присвоив переменной VAR\_1 значение “245”, а переменной VAR\_2 пустое значение:

VAR\_1=245

VAR\_2=

Если для переменной задается значение, содержащее пробелы, то нужно заключать его в кавычки:

VAR\_3=”program files”

В качестве значения переменной можно присвоить результат выполнения команды Linux, указав имя команды в обратных кавычках (обычно этот символ находится на клавише для ввода символов «~» и «ё»):

VAR\_2=`pwd`

Значение переменным можно задать в диалоговом режиме, например команда read A B C присваивает значения переменным A, B и C.

Вывод значений переменных на экран проводится командой ***echo*** с указанием символа “$” перед именем переменной:

echo $VAR\_3

При необходимости выполнения конкатенации значения переменной и произвольной символьной строки используются фигурные скобки:

echo ${VAR\_3}binary

Командный интерпретатор имеет возможность обработки переменных как целых чисел, выполняя операции сложения (+), вычитания (-), умножения (\*), целочисленного деления (/) и получения остатка от деления (%):

A=`expr $x + $y`; B=`expr $x - $y`

C=`expr $x / $y`; D=`expr $x % $y`; E=`expr $x “\*” $y`

Обратите внимание: символ умножения должен быть взят в кавычки, т.к. это служебный символ интерпретатора, а имена переменных и знаки операций всегда разделяются пробелами.

Удаление переменных выполняется командой **unset**, например unset VAR\_3.

Все переменные доступны только в том процессе shell , в котором они были объявлены. Для того чтобы к переменным можно было обращаться из дочерних процессов, их следует экспортировать. Возможны два способа экспорта:

* при создании переменной (например, export VAR\_4=myfolder);
* после создания переменной (например, export VAR\_4 )

Команда ***export*** без аргументов выводит список всех экспортированных переменных интерпретатора.

Для выполнения командного сценария Linux создает отдельный экземпляр командного интерпретатора, поэтому переменные ‘\*’, ‘#’ и ‘@’ имеют собственные значения для каждого запущенного сценария. Все переменные, которые известны сценарию во время выполнения, образуют его окружение, включающее переменные, которые сценарий получает путем наследования от родительского процесса (от интерпретатора shell) и собственные локальные переменные.

2.3 Параметры сценария

Сценарий может принимать аргументы, используя механизм формальных и фактических параметров. Для описания формальных параметров в программе используются специальные переменные, имена которых состоят из одной цифры в диапазоне от 0 до 9, перед которой стоит символ ‘$’.

При запуске сценария в командной строке указываются значения фактических параметров, которые подставляются на место соответствующих формальных параметров. Формальный параметр $0 содержит имя сценария, параметр $1 – значение первого фактического параметра, параметр $2 – второго фактического параметра и т.д. Общее число параметров содержится в переменной # .

Если число фактических параметров превышает число формальных параметров, то в сценарии можно использовать команду **shift** **n**, которая сдвигает список формальных параметров относительно списка фактических параметров на **n** позиций (по умолчанию n=1). Обратите внимание: значение параметра $0 не изменяется при использовании команды **shift** !

При загрузке системы команд­ный интерпретатор Shell выполняет команды, содержащие­ся в файле **/etc/profile**, затем команды, содержащиеся в файле **$HOME /.profile**. Для того, чтобы при входе в систему автома­тически устанавливались необходимые переменные, следует откорректировать файл **.profile**, находя­щийся в Вашем домашнем каталоге.

2.4 Операторы языка Shell

2.4.1 Оператор test

Оператор test проверяет выполнение заданного условия и возвращает логическое значение «истина» (0) или «ложь» (1).

Общий формат оператора:

*test условие или [ условие ]*

Обратите внимание: вторая форма записи оператора предусматривает наличие обязательных пробелов между квадратными скобками и выражением условия. Если строка условия в команде test пустая, то возвращается значение 1.

Возможны условия нескольких видов:

1) проверка файлов или переменных

формат команды: *test ключ имя\_файла | имя\_переменной*

*ключ* задает режим проверки и может принимать следующие значения

-f - указанный файл является обычным файлом;

-d - указанный файл является обычным каталогом;

-с - указанный файл является символьным устройством;

-r - указанный файл доступен для чтения;

-w - указанный файл доступен для записи;

-s - указанный файл не является пустым;

-n - указанная переменная имеет значение (не пустая);

-z - указанная переменная не имеет значения.

Примеры команды:

test –f /usr/user1/file\_1 возвращает 0, если файл /usr/user1/file\_1 является обычным файлом;

test –w /usr/user1/file\_1 возвращает 0, если файл /usr/user1/file\_1 доступен для записи;

test –n $ALFA возвращает 0, если переменная ALFA имеет значение.

2) сравнение строк

формат команды: *test строка1 отношение строка2*

операции отношения строк: = (равно) или != (не равно) .

примеры команд:

test $A = $B возвращает 0, если значение переменной A равно значению переменной B

test $A != $B возвращает 0, если значение переменной A не равно значению переменной B

Обратите внимание: операция отношения обязательно с двух сторон окружается пробелами.

3) сравнение целых чисел.

формат команды: *test число1 отношение число2*

операции отношения чисел:

-eq - равно; -ne - не равно;

-gt - больше; -ge - больше или равно;

-lt - меньше; -le - меньше или равно.

2.4.2 Оператор if

Условный оператор if имеет следующий формат:

if *условие* then

*список\_операторов*

[elif *условие* then

*список\_операторов* ]

[else *список\_операторов*]

fi

Здесь *условие* – любой оператор, возвращающий значение true или false (обычно используется оператор *test*);

*список\_операторов -* операторы, разделенные точкой с запятой (;).

Пример оператора:

if [ -f /tmp/myfile ] then

cat /tmp/myfile

else

ls /tmp > /tmp/myfile; cat /tmp/myfile

fi

2.4.3 Оператор case

Оператор множественного выбора case имеет следующий формат:

Case *строка* in

шаблон\_1) *список\_операторов;;*

шаблон\_2) *список\_операторов;;*

шаблон\_n) *список\_операторов;;*

esac

Пример оператора:

echo –n ‘please enter symbol from A to C: ‘

read ENTRY

case $ENTRY in

A|a) echo ‘Apple’;;

B|b) echo ‘Baby’;;

C|c) echo ‘Coke’;;

\*) echo ‘Please type A, B or C ! ’;;

esac

2.4.4 Оператор for

Оператор предопределенного цикла for имеет следующий формат:

for *имя* [in *список*] do

*список\_операторов*

done

Если список значений для переменной не задан, то в качестве списка используется значение внутренней переменной @, содержащей список аргументов сценария (см. табл. 7). Пример оператора:

LIST=”word 1 word2 word3 word4”

for VAL in $LIST do

echo $VAL

done

2.4.5 Операторы while и until

Операторы while и until предназначены для создания циклов с неопределенным числом исполнения списка операторов. *Список\_операторов* оператора while исполняется только тогда, когда *условие* истинно, а для оператора UNTIL – когда условие ложно. Выход из циклов проводится путем контроля значения *условия* или принудительно по команде BREAK.

Форматы операторов:

|  |  |
| --- | --- |
| while *условие* do  *список\_операторов*  done | until *условие* do  *список\_операторов*  done |

2.4.6 Использование функций в сценариях

Функции Shell описываются следующим образом:

имя\_функции ()

(

*список\_операторов*

)

Функции поддерживают механизм формальных и фактических параметров. Формальные параметры являются локальными и имеют имена от 0 до 9. Если функция должна вернуть код завершения, то используется оператор *return.*

2.5 Создание сценария

Файл сценария можно создать несколькими способами. Первый способ основан на команде **echo** и используется для создания только очень простых сценариев. Если файл должен содержать несколько строк, то в команде **echo** необходимос помощью ключа **-e** разрешить обработку специальных символов и для перевода строки применять символы ‘\n’, например:

echo -e "ls -l \nls -F" > echo\_primer.sh

Здесь в файл echo\_primer.sh будут записаны две строки:

ls –l

ls –F

Второй способ использует команду cat:

cat >cat primer.sh

< команды вводятся с клавиатуры>

нажатие ENTER

Ctrl+D

Третий способ основан на применении текстового редактора **vi** или его клонов (**vim**, **elvis**)**.** Редактор запускается следующим образом: **vi имя\_файла**. Если файл с именем **имя\_файла** не существует, то он будет создан. Редактор **vi** может находиться в одном из трех режимов: в командном режиме, в режиме ввода и в режиме последней строки. После запуска редактор находится в командном режиме. Переход из командного режима в режим ввода осуществляется нажатием клавиши ‘i’ (input) или ‘a’ (add). После нажатия клавиши ‘i’ текст будет вводиться с текущей позиции курсора, а после нажатия клавиши ‘a’ текст будет вводиться со следующей позиции после курсора. Возврат в ко­мандный режим осуществляется нажатием клавиши ‘Esc’. Переход из командного режима в режим последней строки проводится вводом символа ‘:’ (двоеточие)

Сводка по основным командам редактора приведена в таблице 8

Таблица 8

|  |  |
| --- | --- |
| Команда | Назначение |
| *Позиционирование курсора* | |
| H | перейти в начало файла |
| G | перейти в конец файла |
| {n}G | перейти на строку с номером n |
| *Работа с буфером* | |
| yy | копировать текущую строку в буфер |
| {n}yy | копировать n строк, начиная с текущей, в буфер |
| dd | удалить текущую строку в буфер |
| {n}dd | удалить n строк, начиная с текущей, в буфер |
| p | вставить содержимое буфера после текущей строки |
| P | вставить содержимое буфера перед текущей строкой |
| *Редактирование текста* | |
| o | вставить пустую строку после текущей строки |
| O | вставить пустую строку перед текущей строкой |
| i | вставить произвольный текст перед курсором |
| a | вставить произвольный текст после курсора |
| cw | заменить слово |
| dw | удалить слово |
| {n}dw | удаление n слов |
| u | отменить действие последней команды |
| /строка | искать указанную строку вперед по тексту |
| ? строка | искать указанную строку назад по тексту |
| *Команды режима последней строки* | |
| :w | сохранить изменения |
| :w имя\_файла | сохранить изменения в указанном файле |
| :q | выход |
| :q! | выход без сохранения |
| :wq | выход с сохранением |
| :s/строка\_1/строка\_2 | Поиск и замена строки «строка\_1» на «строка\_2» |
| :help | вызов справки |
| :set nu | вывод номеров строк |

**3. Порядок выполнения работы**

3.1 Освоить способы создания сценария, описанные в разделе 2.5.

3.2 Разработать и отладить сценарий для создания файла данных с информацией согласно таблицы 1 (см. лабораторную работу № 2), количество записей в файле – не менее 10. Данные в сценарий передавать через параметры, форматирование полей записи проводить путем циклического добавления необходимого числа пробелов, например:

read fam

echo $fam >> file\_dat

len\_fam=`expr length $fam`

while [ $len\_fam –lt 20 ]

do

#записываем пробел без перевода строки

echo –n “ “ >>$file\_dat;

len\_fam=`expr $len\_fam + 1`

done

Здесь переменная fam будет хранить фамилию, введенную пользователем, а len\_fam –текущую длину введенного значения.

3.3 Изучить и реализовать различные способы запуска сценария (см. п.2.1).

3.4 Разработать и отладить сценарий для поиска в файле данных заданной пользователем строки символов.

3.5 Разработать и отладить сценарий для сортировки файла данных по заданному пользователем полю записи.

3.6 Разработать и отладить сценарий, формирующий меню для опроса пользователя (пункты меню - добавление данных, поиск данных, сортировка данных, выход) и выполнение этого меню с помощью соответствующих сценариев, разработанных в п.3.2 – 3.5 задания. В сценарий необходимо передать в виде параметров номер Вашей группы и номер бригады; при работе сценарий должен в первой строке выводить текущую дату и значения принятых параметров, а затем – строки меню

4. Контрольные вопросы

1. Командный сценарий: определение, назначение.

2. Способы запуска командного сценария.

3. Способы передачи данных в командный сценарий.

4. Переменные в командном сценарии: типы, область видимости, допустимые операции.

5. Организация циклов в сценарии.

6. Организация ветвлений в сценарии.

7. Поясните алгоритм форматирования записей при выполнении п. 3.1 лабораторной работы.

8. Какие командные интерпретаторы входят в TinyCore Linux ?

# Список источников

1. Д. Дж. Баррет. Linux: основные команды. Карманный справочник . –М.: КУДИЦ-ОБРАЗ, 2005. – 288 с.

2. Список команд Windows [Электронный ресурс]. – URL <http://ab57.ru/cmdlist.html> (дата обращения 25.10.2015).

3. Командные файлы Windows [Электронный ресурс]. – URL <http://ab57.ru/cmd.html> (дата обращения 25.10.2015).

4. Бережная установка Tiny Core Linux [Электронный ресурс]. – URL <http://tinycorelinux.ru/topic12-berezhnaya-ustanovka-tiny-core-linux.html> (дата обращения 25.10.2015).

5. Русскоязычный форум TinyCore Linux [Электронный ресурс]. – URL <http://tinycorelinux.ru> (дата обращения 25.10.2015).

# Приложение 1. Порядок создания виртуальной машины TinyCore Plus

Процесс установки разберем на примере среды виртуализации VirtualBox-5.0.4-102546-Win и дистрибутива Core Plus-5.3.

Этап 1. Установка виртуальной машины без файловой системы.

1. Скачайте с сайтов производителей листрибутивы среды виртуализации (<http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html>).

и операционный системы (<http://tinycorelinux.net/downloads.html>).

2. Запустите файл *VirtualBox-5.0.4-102546-Win.exe* и установите VirtualBox, следуя указаниям установщика.

3. Откройте программу Virtual Box, в появившемся окне нажмите кнопку «Создать».

4. Введите имя виртуальной машины (ВМ), например, TinyCore Plus; выберите операционную систему Linux, версию Other Linux и нажмите кнопку Вперед (рис. П.1).

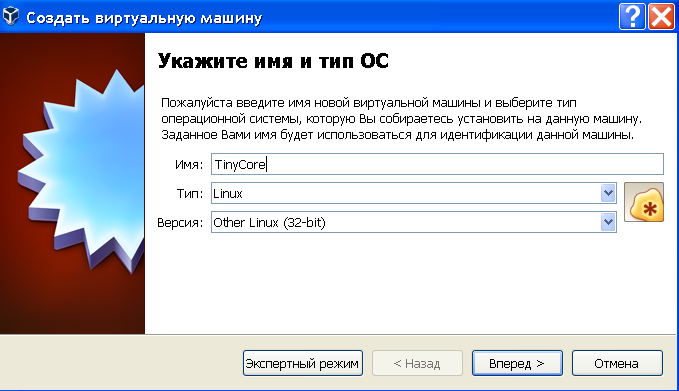


Рис. П.1

5. Укажите размер оперативной памяти ВМ (не менее 256 Мбайт), нажмите кнопку *Вперед* (рис. П.2).

6. Создайте динамический виртуальный жесткий диск в виде файла типа .VDI размером 512 Мбайт, имя файла может быть любым (например, TinyCore). Нажмите кнопки *Вперед* и *Создать* (рис. П.3, рис. П.4).

7. В окне VirtualBox нажмите кнопку *Настроить* и в появившемся окне настроек (рис. П.5) выберите раздел *Носители*, в списке носителей - пункт *Пусто*, в атрибутах оптического привода установите значение *Первичный слэйв IDE* и, сделав щелчок на значке диска, укажите имя файла с образом CD диска, в котором находится дистрибутив TinyCore. После этого в списке носителей появится имя файла образа, например CorePlus-5.3.iso и первый этап создания ВМ завершается.

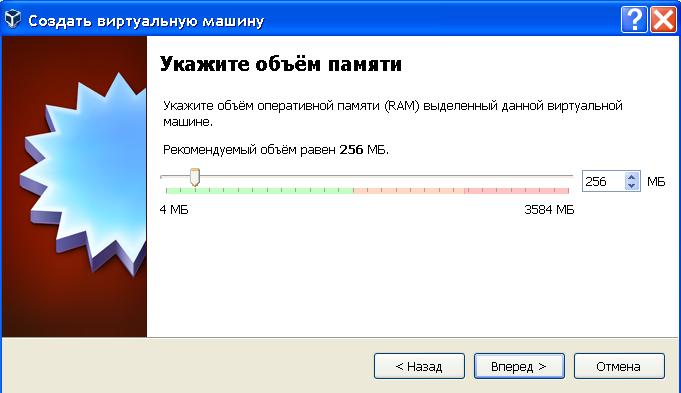


Рис. П.2

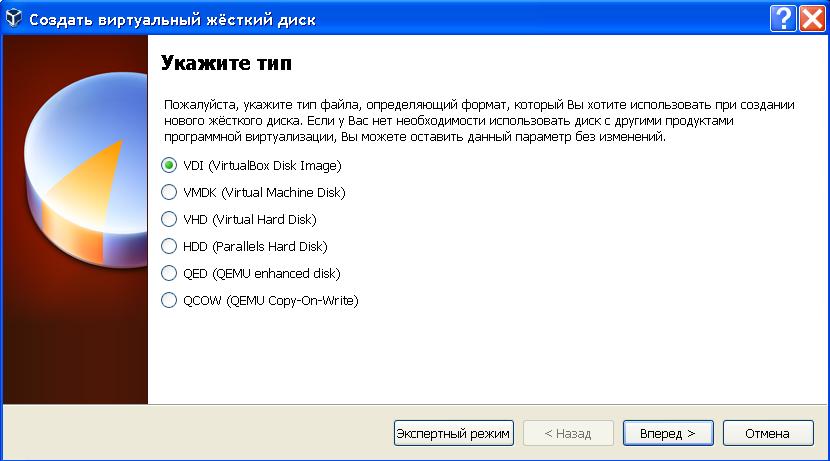


Рис. П.3

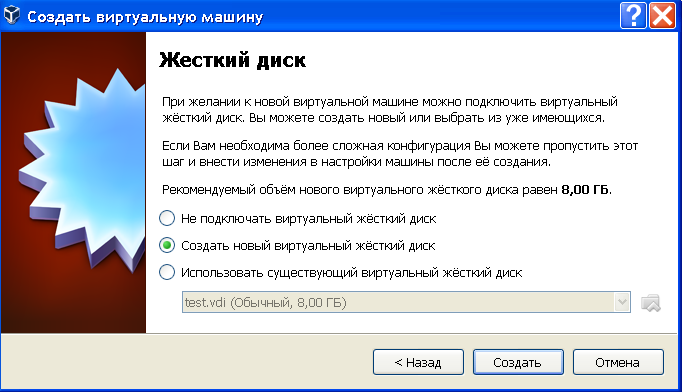


Рис. П.4

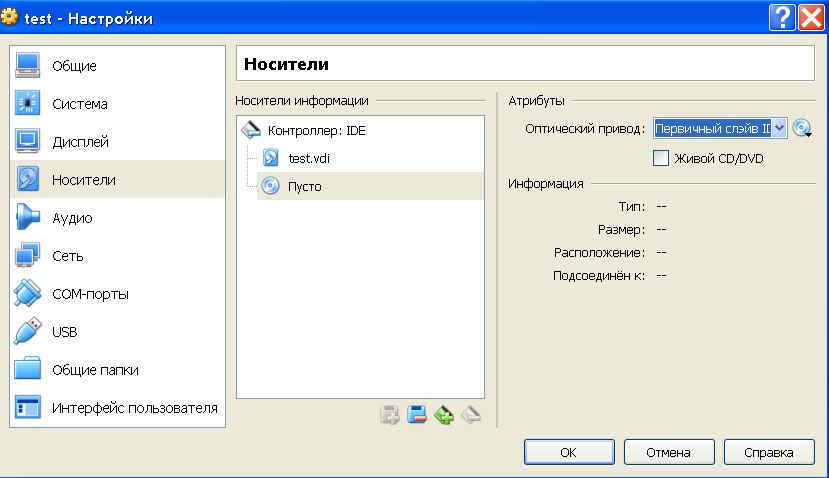


Рис. П.5

8. Запустите созданную виртуальную машину, выбрав для загрузки вариант с установками по умолчанию (рис. П.6).

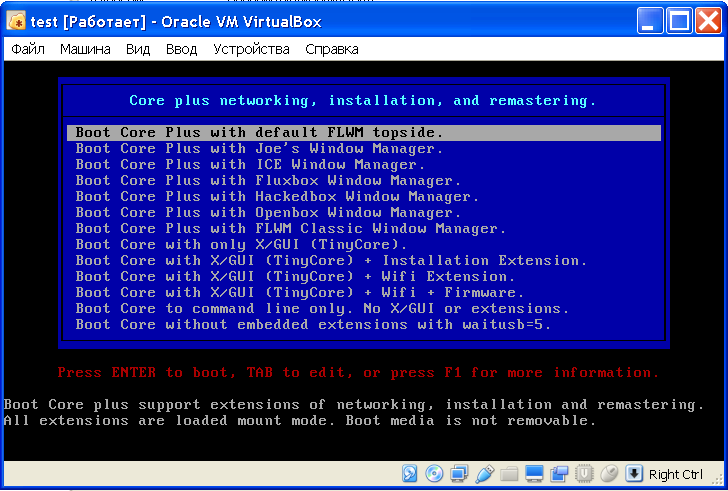


Рис. П.6

Этап 2. Установка полного варианта виртуальной машины

9. Запустите созданную виртуальную машину, выбрав для загрузки вариант “Boot Core with X/GUI (TinyCore) + Installation Extension” (рис. П.6).

10. Через иконку TinyCore или через контекстное меню Application/TC\_Install запустите установщик *TC\_Install*, в окне которого установите следующие параметры (рис. П.7).

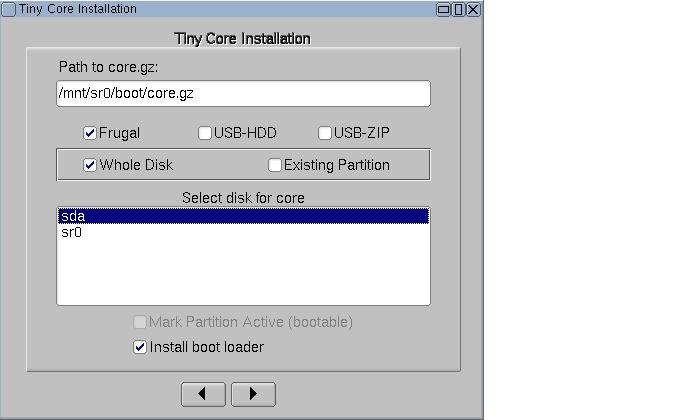


Рис. П.7

11. В следующем окне (рис. П.8) выберите тип файловой системы (*ext4*)

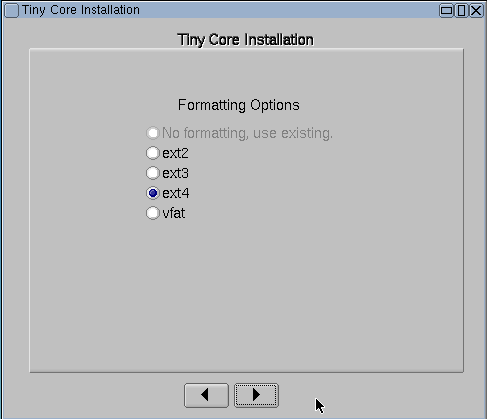


Рис. П.8

12. Выберите тип установки и запустите процесс установки нажатием на кнопку Proceed (рис. П.9, рис. П.10).

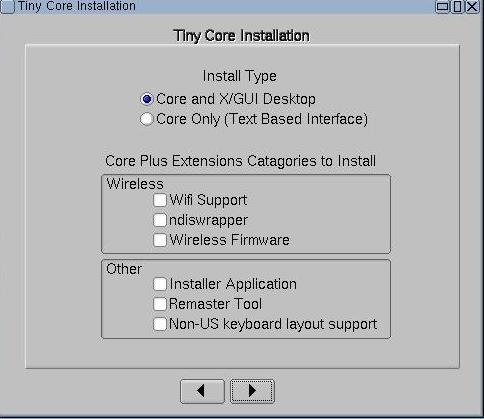


Рис. П.9

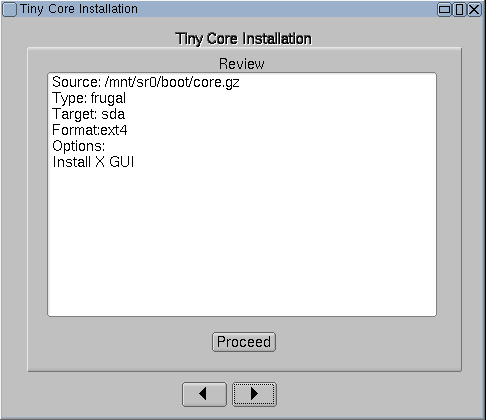


Рис. П.10

13. По окончании установки удалите носитель, с которого осуществлялась загрузка, и перезагрузите компьютер.

Учебное издание

*Кобылянский Валерий Григорьевич*

**Операционные системы**

Учебно – методическое пособие