МИНОБРНАУКИ РОССИИ

ФГБОУ ВО «Ижевский государственный технический университет имени М.Т.Калашникова»

В.Г. Тарасов

Методические указания

к выполнению лабораторной работы № 1

на тему «Знакомство с FreeBSD»

по дисциплине «Операционные системы»

|  |
| --- |
| Ижевск 2018  Рег. номер |

Методические рекомендации по выполнению лабораторной работы составлены в соответствии с рабочей программой учебной дисциплины, разработанной на основе Федерального государственного образовательного стандарта по направлению 09.03.04 «Программная инженерия» профиль «Разработка программно-информационных систем» программа подготовки «Академический бакалавриат» при изучении дисциплины «Операционные системы»

Рецензент: *П.В. Лекомцев, канд. техн. наук, доцент*

Составитель: *В.Г.Тарасов, канд. техн. наук, профессор*

Рекомендовано Советом института ИВТ для использования в учебном процессе в качестве учебно-методических материалов для студентов, обучающихся по направлению09.03.04 «Программная инженерия» профиль «Разработка программно-информационных систем» программа подготовки «Академический бакалавриат» при изучении дисциплины «Операционные системы»(протокол № \_\_ от «\_\_\_» марта 2018 г.)

СОДЕРЖАНИЕ

1. Общие положения4

2. Порядок выполнения работы4

3. Установка и настройка виртуальной машины Virtual BOX5

4. Установка ОС на виртуальную машину Virtual BOX7

5. Информация о системе. Работа с командами bash16

6. Добавление пользователей с привилегией root18

7. Написание программ в среде FreeBSD20

8. Лабораторное задание22

9. Содержание отчета22

10. Контрольные вопросы23

Список рекомендуемой литературы23

Приложение 1. Список терминов24

Приложение 2. Схемы разбиения жесткого диска27

Приложение 3. Работа с EE(Easy Editor)32

Приложение 4. Информация о пользователе33

Приложение 5. Процесс загрузки FreeBSD34

Приложение 6. Структура каталогов37

Приложение 7. Команды FreeBSD38

Приложение 8. Информация об оболочках39

ЛАБОРАТОРНОЕ ЗАНЯТИЕ №1. ЗНАКОМСТВО С FreeBSD

ЦЕЛЬ РАБОТЫ: изучение техники и технологии установки ядра и дополнительных пакетов операционной системы семейства Linux и проверки ее работоспособности с использованием сред виртуального компьютера.

1. ОБЩИЕ ПОЛОЖЕНИЯ

Операционные системы семейства Linux являются бесплатными и подпадают под так называемую лицензию GNU (General Public License - универсальную общественную лицензию), или GPL. ОС Linux рассматривается многими как наиболее жизнеспособный конкурент, противостоящий Microsoft на рынке ОС. Все большее число пользователей становятся приверженцами Linux, либо используют эту ОС совместно с ОС Windows. В то же время установка операционной системы на компьютер является нетривиальной задачей, которая требует внесения изменений в жизненно важные области жесткого диска и может привести к нарушению работоспособности компьютера. Поэтому лучшим способом изучения процессов установки и настройки новой ОС является использование виртуальных машин.

Виртуальная машина - это программа, которую вы запускаете из своей операционной системы. Программа эмулирует реальную машину. Как и в случае с реальной машиной, вы можете установить на виртуальную машину операционную систему, причем неважно Windows, MacOS или Unix. Таким образом, вы можете тестировать различные операционные системы, не покидая своей. У виртуальной машины есть BIOS, жесткий диск (отведенное место на вашем жестком диске), CD-ROM (ваш CD-ROM или подключенный ISO-образ), сетевые адаптеры для соединения с вашей реальной машиной, сетевыми ресурсами или другими виртуальными машинам и т.д. Вы можете без проблем обмениваться файлами между основной операционной системой (host) и гостевой операционной системой (guest). Это осуществляется простым перетаскиванием файлов из файлового менеджера клиента в окно гостевой системы или в обратном направлении. Удобство виртуальной машины для тестирования автоматической установки просто неоценимо. Достаточно просто подключить загрузочный ISO-образ вместо CD-ROM в настройках виртуальной машины, и установка системы пойдет точно так же, как и на реальной машине.

В курсе лабораторных работ предполагается изучение ОС FreeBSD-12-RELEASE-i386/amd64 (и выше) с применением виртуальной машины Oracle VM VirtualBOX.

2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Лабораторная работа выполняется фронтально в учебно-исследовательской лаборатории кафедры программного обеспечения. После установки ОС на виртуальную машину производится проверка ее работоспособности. При этом происходит знакомство с наиболее часто используемыми командами ОС, создание нового пользователя, изучение тестового редактора **ee** (краткая справка о командах редактора содержится в приложении 3) и написание программ на языке программирования C/C++.

Предлагаемые ниже задания 1-7 определяют содержание лабораторной работы и задают порядок ее выполнения.

В составлении и проверке заданий участвовали студенты кафедра ПО Муллахметов Данил и Чигвинцева Влада.

3. УСТАНОВКА И НАСТРОЙКА ВИРТУАЛЬНОЙ МАШИНЫ VIRTUAL BOX (**Задание 1**)

Краткое описание задачи: подготовим среду для установки ОС. Установим и настроим виртуальную машину.

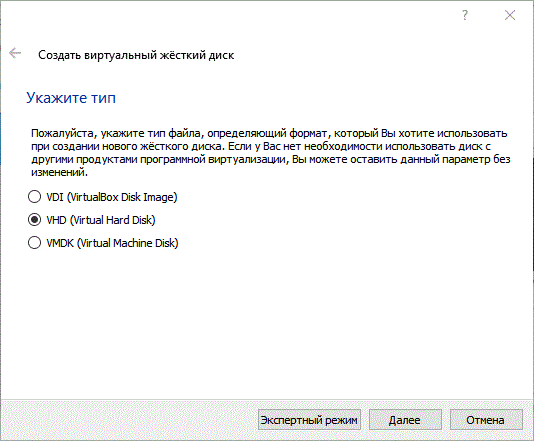
Скачать VirtualBOX можно из официального источника по ссылке <https://www.virtualbox.org/wiki/Downloads>.

При создании новой ВМ выбрать тип BSD (рис.3.1) и версию FreeBSD и назвать машину. Разрядность версии выбрать в зависимости от разрядности вашей ОС.Выделить объем оперативной памяти для ВМ не меньше рекомендуемого (рис.3.2).

### C:\Users\1\Desktop\ОС\Изображения\Лаб1\Лаб 1 рис. 3.1.GIF C:\Users\1\Desktop\ОС\Изображения\Лаб1\рис. 3.2.GIF

Рис.3.1 Рис.3.2

Создать новый виртуальный жесткий диск типа VHD (см. Приложение 1) (рис.3.3, рис.3.4).



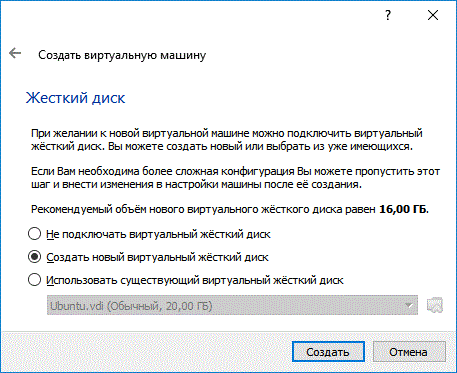


Рис.3.3 Рис.3.4

Выбрать 1 из 2 форматов хранения: разница в том, что фиксированный диск работает быстрее, а динамический занимает весь предоставленный объем лишь по мере заполнения (рис.3.5). Также если потребуется увеличить размерность дискового пространства, то в случае фиксированного виртуального жесткого диска это сделать невозможно. Иконка справа от имени виртуального диска отвечает за его расположение. Не рекомендуется располагать его на одном диске с основной ОС. Размер диска установить не меньше рекомендуемого (рис.3.6).

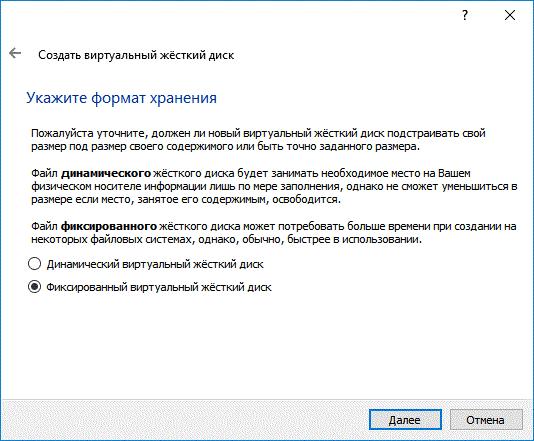
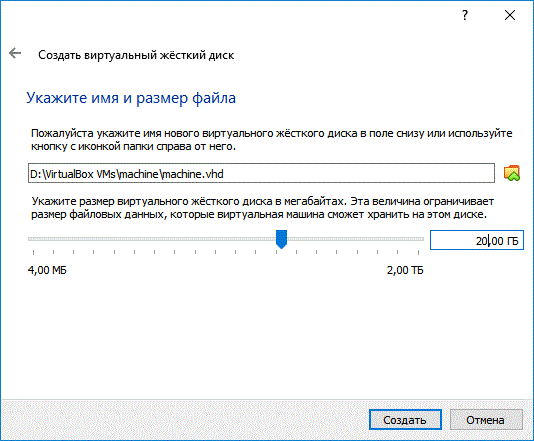


Рис.3.5 Рис.3.6

После установки ВМ при желании можно зайти в ее настройки в VirtualBOX и увеличить число процессоров для ускорения работы. Здесь же можно изменить имя, тип и версию ОС вместе с другими параметрами (рис.3.7, рис.3.8).

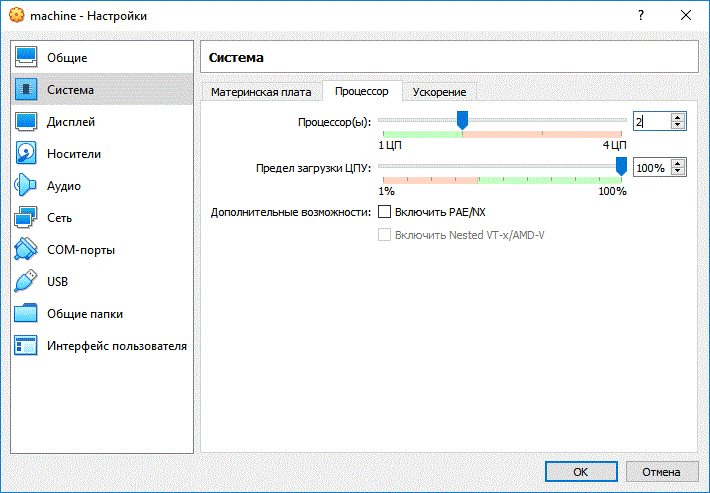
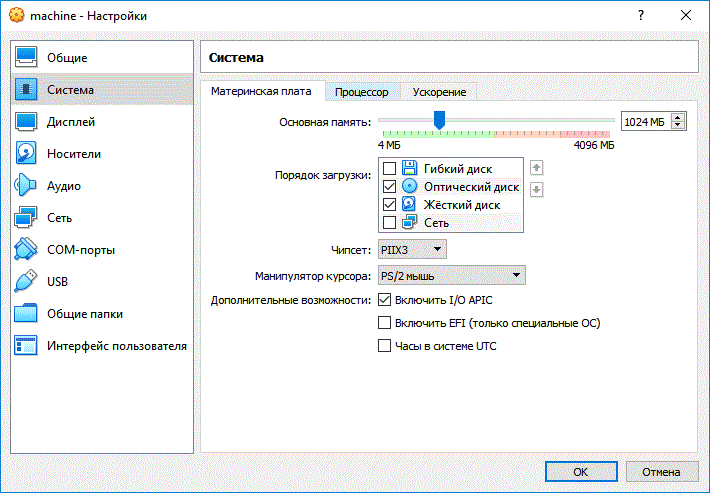


Рис.3.7 Рис.3.8

Скачать ISO образ последней версии FreeBSD на официальном сайте www.freebsd.org/ru/. В разделе «аппаратная совместимость» можно узнать, какая именно платформа совместима с процессором ПК, на который устанавливается ОС (в большинстве случаев подходят amd64 и i386) (рис.3.9).

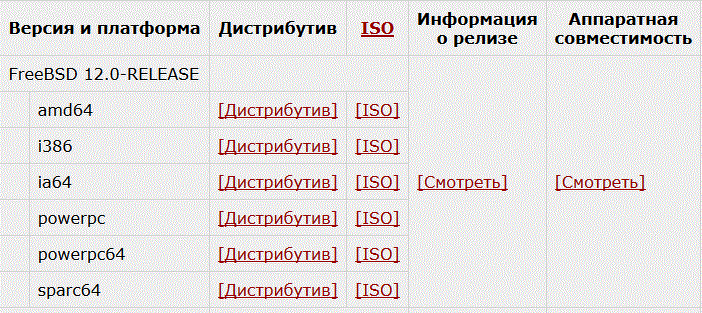
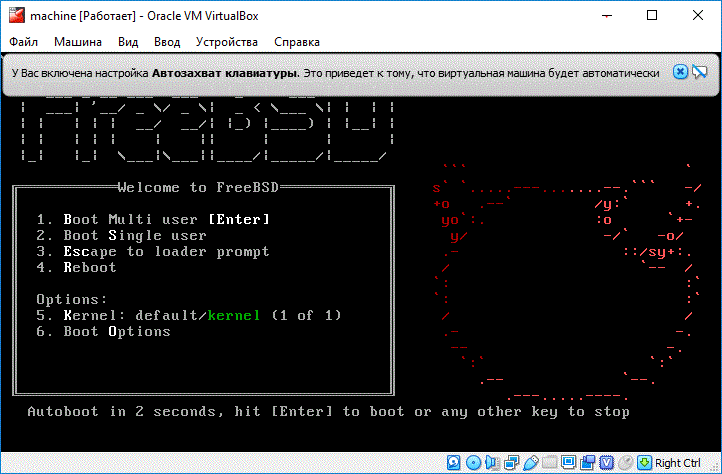


Рис.3.9

При первом запуске ВМ подключить скачанный образ (рис.3.10). Отобразится меню загрузчика FreeBSD (рис.3.11). Установка начнется сама, с течением времени (после 10 секунд), либо ее можно воспроизвести самому, нажав Enter. После чего на экране появится экран приветствия, после полноценного определения устройств в системе, описанный в следующем разделе.



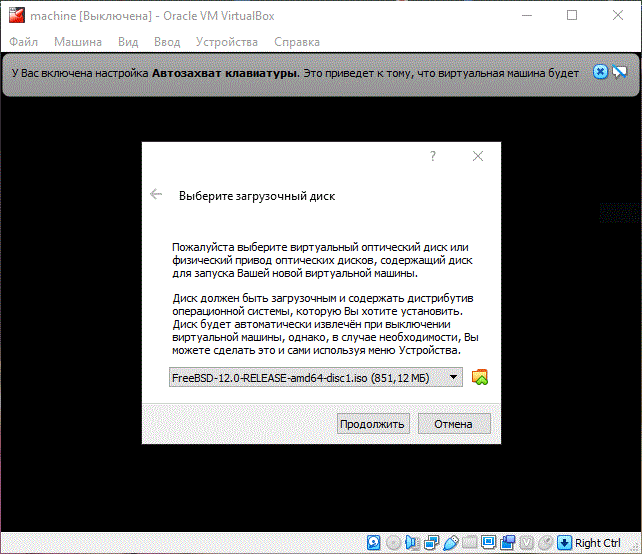


Рис.3.10 Рис.3.11

4. УСТАНОВКА ОС НА ВИРТУАЛЬНУЮ МАШИНУ VIRTUAL BOX

(**Задание 2**)

Краткое описание задачи: установим и настроим основные компоненты ОС, необходимые для ее дальнейшего запуска и полноценной работы.

В окне приветствия вы увидите варианты работы установочного носителя (рис.4.1). Установочный носитель может использоваться одним из трёх способов: для установки FreeBSD, как Live CD (см. Приложение 1), или просто для доступа к оболочке FreeBSD. Используйте клавиши навигации для выбора опции, а Enter - для подтверждения выбора. Выбор опции «Install» вызовет программу-установщик.



Рис.4.1

После этого запустится «мастер установки» - программа bsdinstall. В зависимости от используемой системной консоли bsdinstall может предложить выбрать отличную от настроенной по умолчанию раскладку клавиатуры. При выборе раскладки клавиатуры достаточно оставить стандартную (рис.4.2).

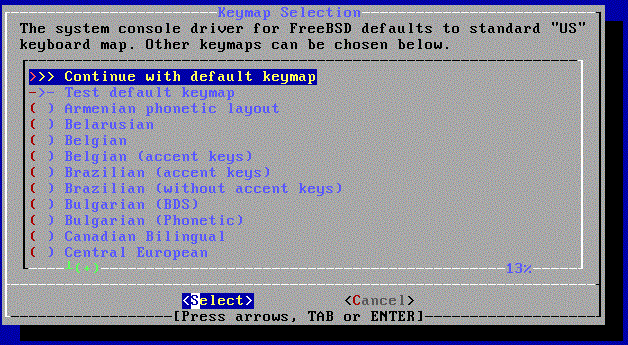


Рис.4.2

Далее bsdinstall предложит указать имя хоста для устанавливаемой системы (рис.4.3). Указать имя новой системы и выбрать компоненты для установки (рис.4.4). Определение перечня компонентов для установки в наибольшей мере зависит от планируемого использования системы и от количества доступного дискового пространства. Ядро (см. Приложение 1) и набор утилит FreeBSD (вместе называемые «базовой системой») устанавливаются всегда.

В зависимости от типа установки, некоторые из следующих компонентов могут не появляться. Позже их можно будет скачивать из интернета, а для начала лучше установить только lib32.

Дополнительные компоненты:

* doc - дополнительная документация, преимущественно исторического характера. Документация, предоставляемая Проектом Документирования FreeBSD может быть установлена позже;
* games - несколько традиционных игр BSD, в том числе fortune, rot13, и другие;
* lib32 - библиотеки совместимости для запуска 32-битных приложений на 64-битных версиях FreeBSD;
* ports - коллекция Портов FreeBSD (см. Приложение 1);
* src - исходный код системы. FreeBSD распространяется с полным исходным кодом как для ядра, так и для программ базовой системы. Для большинства приложений исходный код системы не нужен, однако он может потребоваться при построении некоторых программ, распространяемых в виде исходных кодов (например, драйверов или модулей ядра), или для разработки FreeBSD. Полное дерево исходных кодов требует 1 ГБ дискового пространства, пересборка всей системы FreeBSD требует дополнительно 5 ГБ пространства.

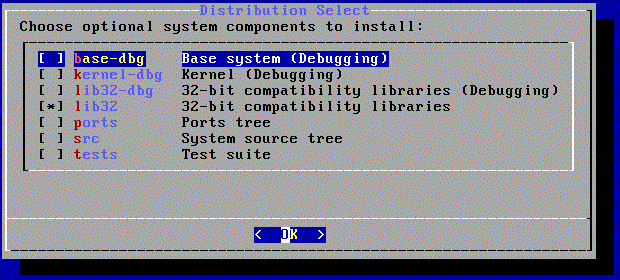
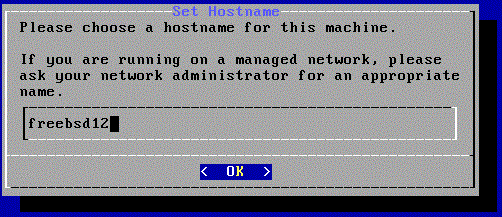


Рис.4.3 Рис.4.4

Есть четыре способа осуществить разбиение дискового пространства для FreeBSD. Шаблонное (guided) разбиение автоматически настраивает разделы диска, ручное (manual) разбиение позволяет опытным пользователям создавать разделы согласно своим требованиям. Также, есть возможность вызвать командный интерпретатор, в котором можно будет непосредственно запускать утилиты наподобие gpart, fdisk и bsdlabel, использовать файловую систему ZFS (см. Приложение 1).

Выбрать разбивку жесткого диска Auto(UFC) (рис.4.5). Под разбивку выделить весь диск (рис.4.6). Для FreeBSD может быть выделен весь диск или только его часть. Если выбирается «Entire Disk», то создается стандартное разбиение, занимающее весь диск. Выбрав «Partition», вы получите создание разделов в неиспользуемой области диска. Это значит, что данные, которые уже есть на диске, останутся не тронутыми, а та часть диска, которая является неиспользуемой, будет местом для установки ОС.

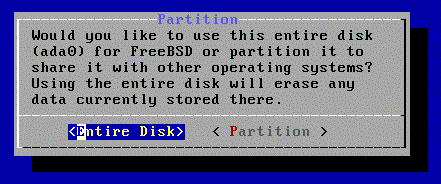


Рис.4.5 Рис.4.6

Как правило, схема GPT (рис.4.7) является наиболее подходящей для PC-совместимых компьютеров.

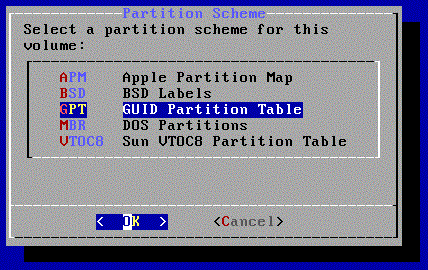


Рис.4.7

Для более старых операционных систем, которые несовместимы с GPT, может потребоваться разбиение MBR. Остальные схемы разбиения используются для нераспространенных или старых компьютерных систем (см. Приложение 2). Стандартная установка FreeBSD со схемой GPT создаст как минимум три раздела:

* freebsd-boot - загрузочный код FreeBSD;
* freebsd-ufs - файловая система UFS FreeBSD;
* freebsd-swap - FreeBSD область подкачки.

Выбрать таблицу разбивки GPT. По завершении разбиения дискового пространства внимательно просмотрите результат (рис.4.8). Если была допущена ошибка, то вам предоставляется возможность либо вернуть конфигурацию к исходному состоянию нажав «Revert», либо выполнить автоматическое переразбиение выбрав «Auto». Также разделы могут быть созданы, изменены или удалены вручную. Если результат разбиения корректен, выберите «Finish» для продолжения установки.

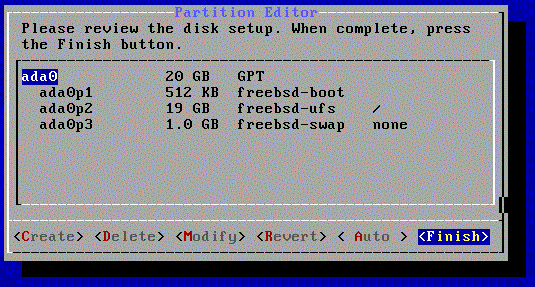


Рис.4.8

На данном этапе установки у вас есть возможность прервать установку и предотвратить изменение данных на жестком диске. Для продолжения выберите «Commit» (осуществить, совершить) (рис.4.9).

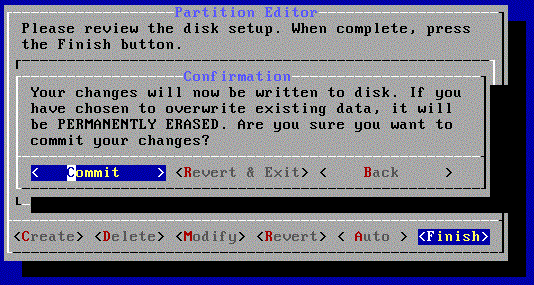


Рис.4.9

Если необходимо внести изменения, для возвращения к редактору разделов нажмите «Back». Выбор «Revert & Exit» дает возможность выйти из установщика без внесения изменений на жесткий диск.

Продолжительность установки варьируется в зависимости от выбранного дистрибутива, способа установки и быстродействия компьютера. Далее последует очередь сообщений, информирующих о ходе установки (рис.4.10).

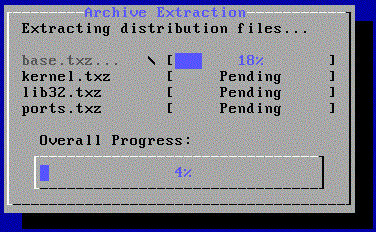


Рис.4.10

Первым делом установщик запишет информацию о разделах на диск и отформатирует разделы посредством newfs (см. Приложение 1). Далее последует проверка целостности файлов дистрибутива, чтобы удостовериться, что они не были повреждены во время загрузки или чтения с установочного носителя. И в заключение, проверенные файлы распаковываются на диск. Как только запрошенные файлы дистрибутива распакуются, bsdinstall приступит к выполнению послеустановочных конфигурационных задач.

После успешной установки FreeBSD появится меню настройки различных опций. Настройки опций могут быть изменены путем повторного входа в соответствующие разделы финального конфигурационного меню перед загрузкой в свежеустановленную систему FreeBSD.

Установка пароля пользователя root - обязательна. Задать и подтвердить пароль для пользователя root (рис.4.11). При вводе пароль не отображается на экране. Далее будет отображен перечень всех сетевых интерфейсов (см. Приложение 1), найденных на компьютере. Выбрать сетевой интерфейс для настройки (он будет совпадать с тем, что указан на скриншоте) (рис.4.12).

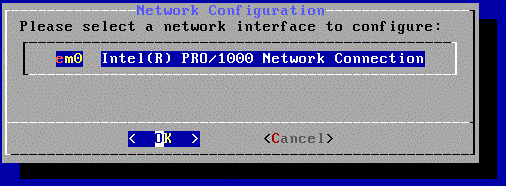
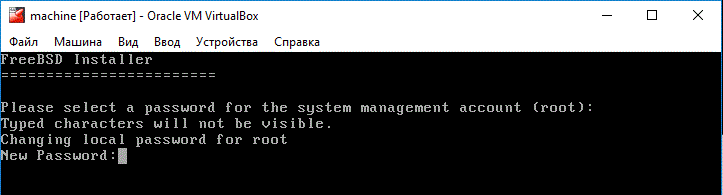


Рис.4.11 Рис.4.12

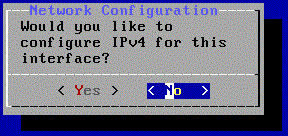
Далее последует настройка сетевых протоколов. На данный момент мы не будем их устанавливать, так как сначала нам стоит ознакомиться с работой самой системы без доступа к сети. Сначала программа установщик предложит подключиться к сети IPv4 (см. Приложение 1) (рис.4.13). Это наиболее распространённый сетевой протокол. Сейчас нам это не нужно, поэтому следует выбрать «No».

Рис.4.13

IPv6 (см. Приложение 1) это более новый сетевой протокол. Программа установки предложит осуществить подключение к сети IPv6, выберите в этом меню «No» (рис.4.14).

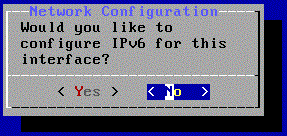


Рис.4.14

Установка часового пояса для вашей машины позволит ей автоматически корректировать время согласно местным законам и правильно выполнять остальные зависимые от часового пояса функции. Выбрать регион (рис.4.15), страну (рис.4.16) и область проживания (рис.4.17). Согласиться с сокращением названия области, если оно слишком длинное.



Рис.4.15 Рис.4.16

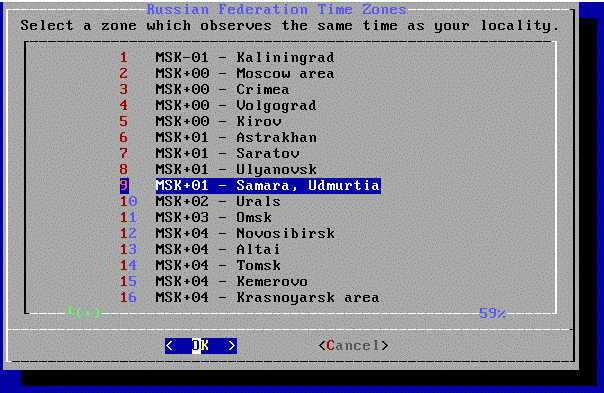


Рис.4.17

Подтвердите, что аббревиатура для часового пояса является приемлемой. Если данная опция настроена верно, то нажмите клавишу Enter для продолжения послеустановочного конфигурирования (рис.4.18).

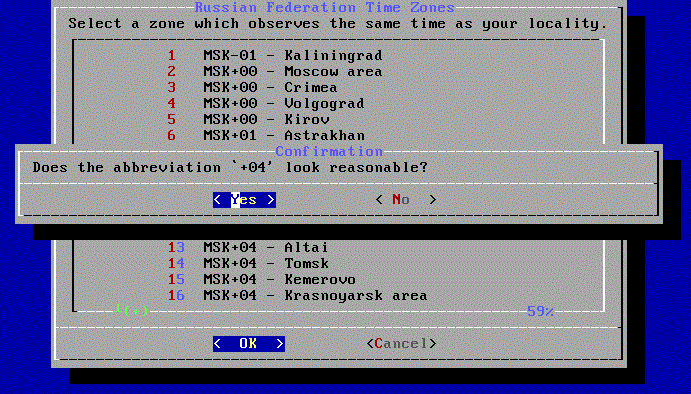


Рис.4.18

Пропустить настройку даты (рис.4.19) и времени (рис.4.20).

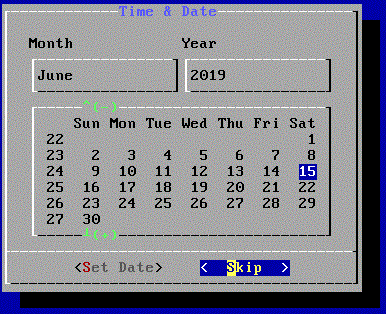
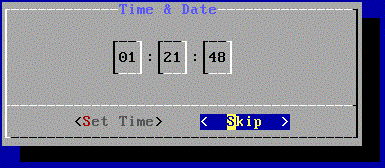


Рис.4.19 Рис.4.20

Далее установщик предложит выбрать службы, автоматически запускающиеся вместе с системой:

* sshd - Secure Shell (SSH) демон для безопасного удаленного доступа;
* moused - обеспечивает использование мыши в системной консоли;
* ntpd - Network Time Protocol (NTP) демон для автоматической синхронизации времени;
* powerd - Системная утилита для контроля потребляемой мощности и профилей энергосбережения.

Нам нужна «чистая» система, поэтому все отметки нужно снять (рис.4.21).

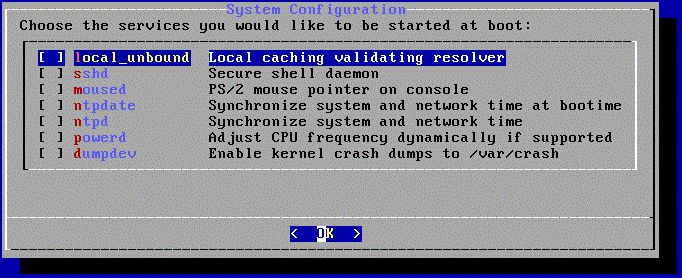


Рис.4.21

При желании настроить параметры безопасности (рис.4.22). В дальнейшем все это можно изменить. Здесь есть различные параметры:

* скрывать запущенные процессы других пользователей;
* скрывать запущенные процессы других групп пользователей;
* запретить чтение сообщений ядра непривилегированным пользователям;
* запрещать отладку непривилегированным пользователям;
* случайный PID новым процессам (номер процесса случайным образом);
* вставлять защищенную страницу памяти в расширяемых сегментах;
* удалять временные файлы при старте системы;
* запретить Syslog;
* запретить запуск почтового сервиса.

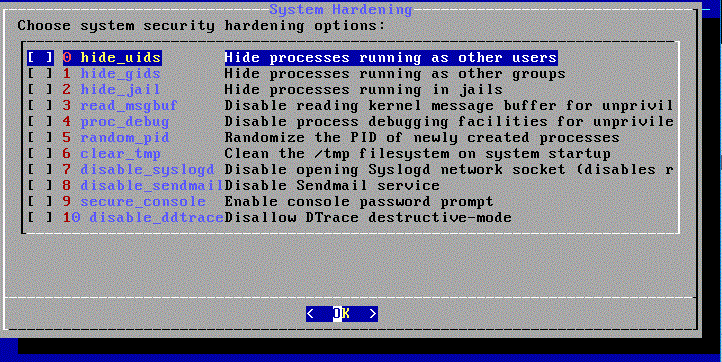


Рис.4.22

Еще на этапе установки в систему можно добавить дополнительных пользователей кроме пользователя root (рис.4.23). Мы сделаем это позже при помощи специальной команды, а пока стоит выбрать «No».

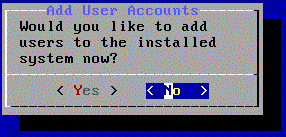


Рис.4.23

После того, как установка и конфигурирование завершены, вам предоставляется заключительная возможность подкорректировать настройки (рис.4.24).

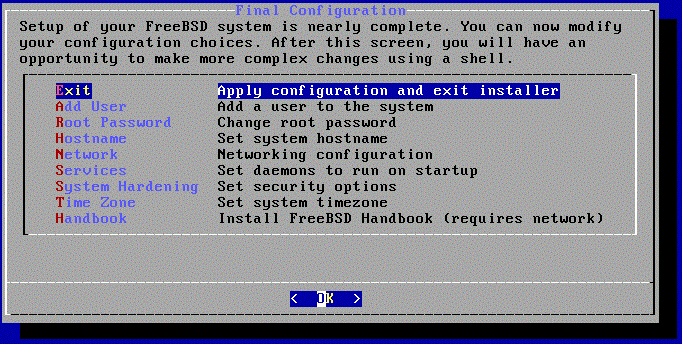


Рис.4.24

По завершении настройки для выхода из финального конфигурационного меню выберите Exit. bsdinstall уточнит, есть ли какие настройки, которые необходимо выполнить до перезагрузки в свежеустановленную систему. Для входа в командный интерпретатор новой системы выберите «Yes», для перехода к последнему шагу установки нажмите «No» (рис.4.25).

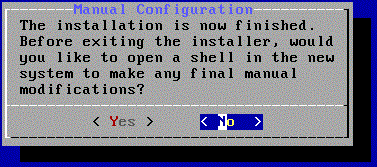


Рис.4.25

При необходимости дальнейшего конфигурирования или особой установки выбор «Live CD» переведет установочный носитель в режим Live CD.

Для нас же, установка завершена, и для перезагрузки компьютера и запуска новой системы FreeBSD следует выбрать «Reboot» (рис.4.26).

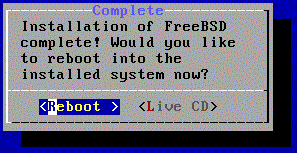


Рис.4.26

Не забудьте извлечь установочный CD, DVD или USB-накопитель, иначе компьютер может снова с него загрузиться. Для этого найдите на верхней панели меню вашей виртуальной машины пункт «Устройства» и нажмите на него. В появившемся подменю выберите пункт «Оптические диски», теперь мы можем увидеть, какие диски уже подключены к нашей машине. Выберите «Изъять диск из привода». Сделать это все нужно максимально быстро, или установка системы начнется с начала. Данная последовательность действий показана на рис. 4.27 для наглядности.

Осторожно! Пока не выполнены все предыдущие пункты, включая отправку на перезагрузку, извлекать диск с установщиком нельзя.

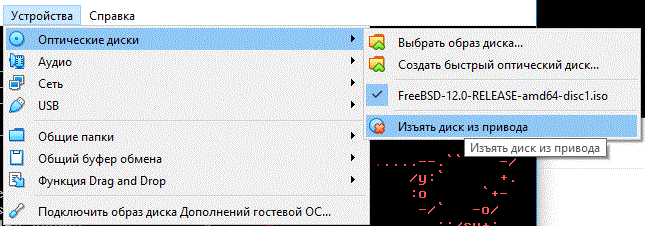


Рис.4.27

5. ИНФОРМАЦИЯ О СИСТЕМЕ. РАБОТА С КОМАНДАМИ BASH

(**Задание 3**).

Сейчас перед нами только что установленная система FreeBSD и что именно с ней нужно делать – непонятно. Для начала давайте зайдем в нее. При включении система сама предложит нам ввести логин и пароль пользователя. На данный момент у нас имеется только один пользователь – **root**, пароль для которого мы создали во время установки. Введем их. Перед нами появится приветствие системы, содержащее краткую информацию о ней (рис.5.1). Она нас пока не интересует. А вот что нам нужно – это приглашение.

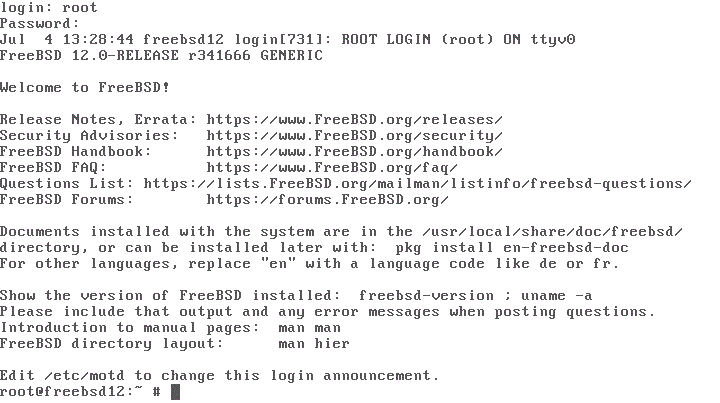


Рис.5.1

Приглашение – это самая последняя строка на рис.5.1. Давайте рассмотрим его поближе:

**root@freebsd12:~ #**

Все то, что мы видим до двоеточия, позволяет нам определить, в какой системе и в каком аккаунте мы находимся:

**root** – имя пользователя;

**@** - разделительный знак;

**freebsd12** – имя системы.

После двоеточия следует знак **~**, который указывает, что мы сейчас находимся в домашнем каталоге пользователя. Мы можем узнать его имя и расположение при помощи команды **pwd**, а содержимое – командой **ls**.

Команда **pwd** покажет вам, где вы находитесь, т.е. текущий рабочий каталог. А **ls** – выдаст список файлов, находящихся в данном каталоге.

Давайте попробуем. В результате мы должны получить ту же ситуацию, что и на рис.5.2.

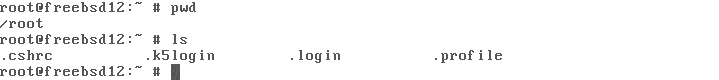


Рис.5.2

Так как мы сейчас используем аккаунт пользователя root, то нашим домашним каталогом стала папка «root», содержащая некоторые файлы. Домашний каталог в системе FreeBSD – это личная папка пользователя, в которой он может хранить свои файлы и программы. Но для пользователя root использование этой папки в нашей работе может быть опасно для системы, так как она хранит важные файлы, которые можно случайно повредить.

Поэтому, для дальнейшего выполнения лабораторной работы мы создадим нового пользователя, чьим домашним каталогом будет являться изначально пустая папка /usr/home/user\_name.

Помимо домашнего каталога на данный момент уже существует еще несколько папок. Мы пока ничего не можем сделать с ними и содержащимися в них файлами, но можем на них посмотреть. Помочь перейти в другую папку может команда **cd**. Например, давайте посмотрим на папку /usr. Она хранит большинство пользовательских утилит и приложений. Воспользуемся командой **cd /usr**, а затем осмотримся при помощи команды **ls**. Должно получится так же, как и на рис.5.3.

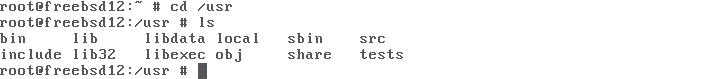


Рис.5.3

Обратите внимание, символ ~ сменился на /usr. Это значит, что мы успешно сменили наше расположение. Если бы у нас были другие пользователи, то мы бы увидели здесь еще и папку /home, которая содержала бы их домашние каталоги. Подробнее о содержимом папки можно узнать из приложения 6, а сейчас, давайте вернемся обратно в домашний каталог. Для этого воспользуемся командой **cd ~**.

И, наконец, мы переходим к самому последнему знаку в приглашении и это - **#**. Он указывает на то, обладает ли текущий пользователь правами root или нет. В нашем случае мы зашли как супер-пользователь root, так что разумеется, мы ими обладаем. Но если бы мы зашли как обычный пользователь, то увидели бы вместо **#** знак **%**. Убедится в этом можно будет при выполнении следующего пункта лабораторной работы.

Подробнее о других командах системы FreeBSD мы поговорим позже, либо вы можете узнать о них из приложения 7. А о структуре каталогов – из приложения 8.

6. ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЕЙ С ПРИВИЛЕГИЕЙ ROOT (**Задание 4**)

Теперь, когда мы успешно установили практически «голую» систему FreeBSD и разобрались, как с ней работать, можно приступить к добавлению нового пользователя. Для этого изначально стоит войти в систему как пользователь **root**. Если вы уже сделали это в предыдущем пункте, то просто продолжим.  
 Добавление хотя бы одного пользователя позволит эксплуатировать систему, исключая необходимость входа под учетной записью **root**. Работа в системе с правами пользователя **root** особенна тем, что по существу нет ограничений или защиты от действий пользователя. Вход под обычным пользователем является более благоразумным и безопасным.

Чтобы начать создание нового пользователя, введем в командной строке команду **adduser**. Сразу после этого начнется его создание, заполните поля согласно примеру на рис.6.1 (полное описание этих полей вы можете найти в приложении 5).

Поля uid, login group, login class, home directory, home directory permissions можно оставить пустыми. Обязательно добавить пользователя в группу **wheel**, чтобы позже присвоить ему права пользователя root. В поле shell выбрать текстовый интерфейс общения с ОС (см. Приложение 8).

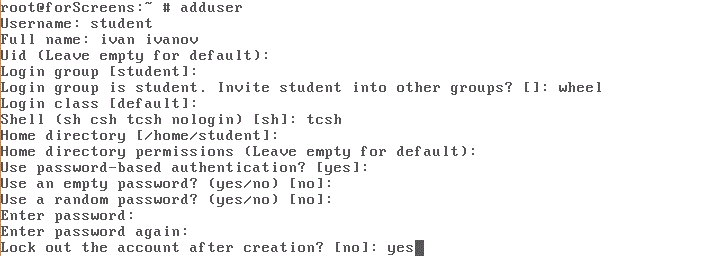


Рис. 6.1

После ввода и подтверждения пароля вы сможете просмотреть полную информацию о созданном пользователе (рис. 6.2).

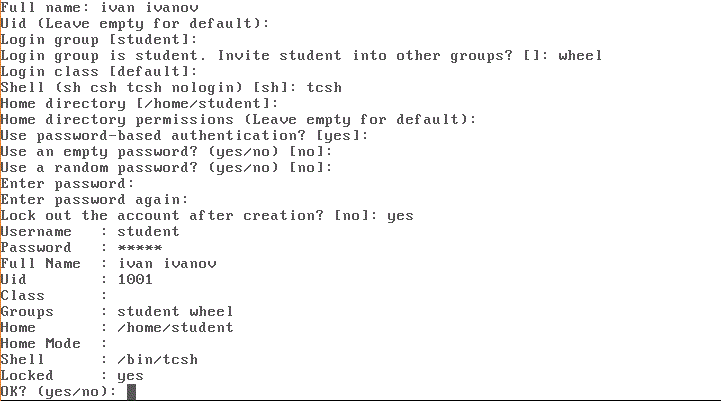


Рис.6.2

Проверьте правильность введенных данных и ответьте «yes» на вопрос «OK?». Следующим пунктом вам будет предложено создать еще одного пользователя. В данной работе он нам не понадобится, поэтому ответьте «no» на вопрос "Add another user?".

Теперь у нас есть пользователь, имеющий свою домашнюю папку. В предыдущем пункте мы рассматривали папку /usr. Теперь, когда у нас есть созданный пользователь, мы можем убедиться, что папка /usr/home была создана. Откроем папку /usr и осмотрим содержимое. Папка /home есть. Давайте перейдем в нее. Так как мы уже внутри /usr, то можно не прописывать полный путь, а просто использовать **cd home**. Осмотримся в ней. Так как пользователь student был создан только что, ожидаемо, что его домашний каталог окажется пустым. Произведенные нами действия в самой системе будут выглядеть так, как показано на рис.6.3.

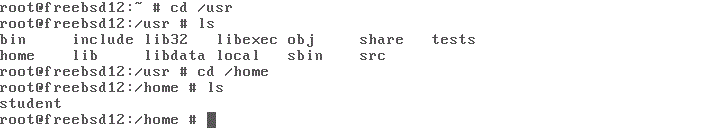


Рис.6.3

Теперь же, давайте выйдем из аккаунта пользователя **root** и зайдем уже как **student**. Для этого воспользуемся командой **exit** и введем в появившиеся поля нужные логин и пароль. После того как зашли, мы находимся в домашней папке текущего пользователя /usr/home/student, которую только что осматривали. Убедимся в этом командой **pwd**. При повторном осмотре командой **ls** папка /home все так же пуста (рис.6.4).

C:\Users\1\Desktop\ОС\Изображения\Лаб1\Скрины для новых методичек\Для нового 5 пункта\рис 6.3.GIF

Рис.6.4

Теперь, на рис. 6.4 мы хорошо видим, что символ **#** сменился на **%**. Значит, у пользователя student на данный момент нет root-прав. Чтобы включить root-права для данного пользователя, нужно воспользоваться командой **su**. Если вы снова хотите вернуться к обычному пользователю, то просто воспользуйтесь командой **exit**. В данной лабораторной работе root-права нам пока не понадобятся, но, не умея их подключать, мы не сможем выполнить последующие задания.

7. НАПИСАНИЕ ПРОГРАММ В СРЕДЕ FREEBSD

(**Задание 5)**

Краткое описание задачи: напишем простую программу с использованием редактора **ee** (подробнее см. приложение 3).

Интерфейс работы с редактором ee можно увидеть при его использовании или на рис.7.1. Основные команды прописаны прямо над рабочим полем. Для открытия файла редактором нужно использовать команду ee filemame. Если же файл с таким именем не существует, то он будет создан. Создадим файл test.c. Напомним, что все наши файлы располагаются в домашнем каталоге пользователя. Проверить его наличие там можно уже знакомой нам командой **ls**.

Вызовем команду **ee test.c** и посмотрим на результат (рис.7.1).

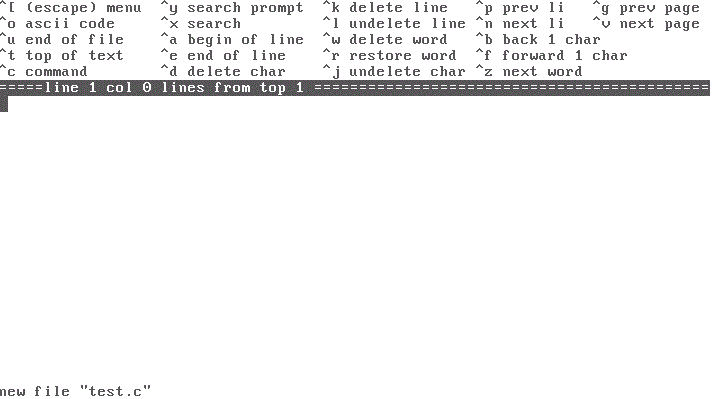


Рис.7.1

Файл test.c был успешно создан. Теперь запишем в файл небольшую программу и сохраним его.

#include <stdio.h>

int main()

{ printf(“Hello, world!\n”);

return 0;

}

Чтобы сохранить файл, вызовем меню при помощи клавиши ESС (рис.7.2.).

Теперь у нас есть два способа сохранить файл.

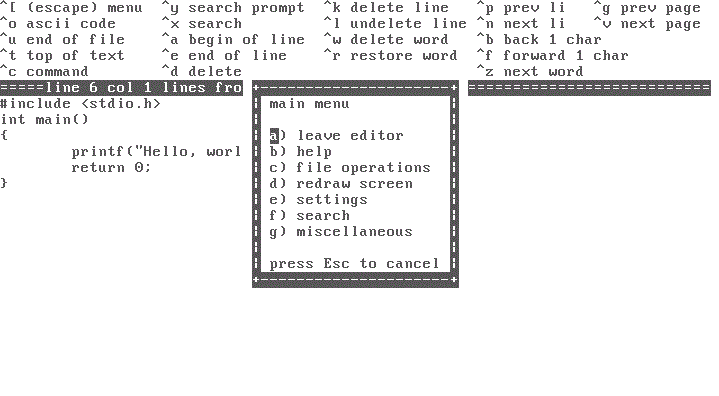


Рис.7.2

Первый способ: выбрать пункт file operations и открыть подменю работы с файлом (рис.7.3).

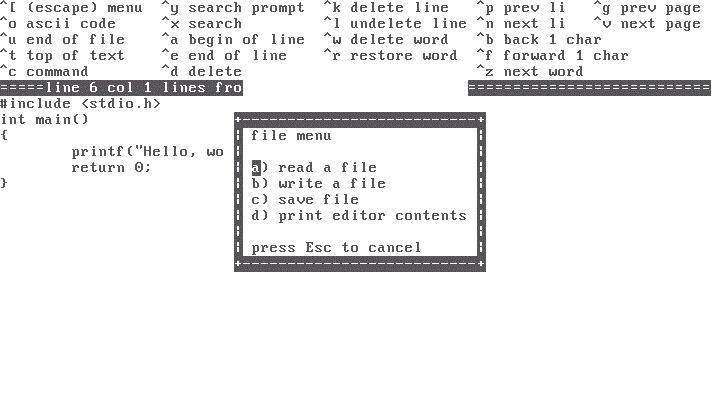


Рис.7.3

В этом подменю можно:

**read a file** (прочесть файл) – считывание уже существующего файла и помещение его содержимого в позиции курсора текущего файла;

**write a file** (записать в файл) – запись текущего файла. Если файл уже был сохранен, а введенное имя совпадает с существующим, то файл будет перезаписан. Если нет, то создастся копия этого файла под другим именем, но в текущей папке;

**save file** (сохранить текущий файл) – сохраняет файл в той папке, из которой был вызван редактор ee;

**print editor contents** (распечатать файл) – распечатывает файл.

Для сохранения файла просто выберем пункт **save file**.

Второй способ еще проще: после вызова общего меню, выбираем пункт **leave editor** (покинуть редактор) и если файл не был сохранен, то редактор сам предложит нам сделать это (рис. 7.4).

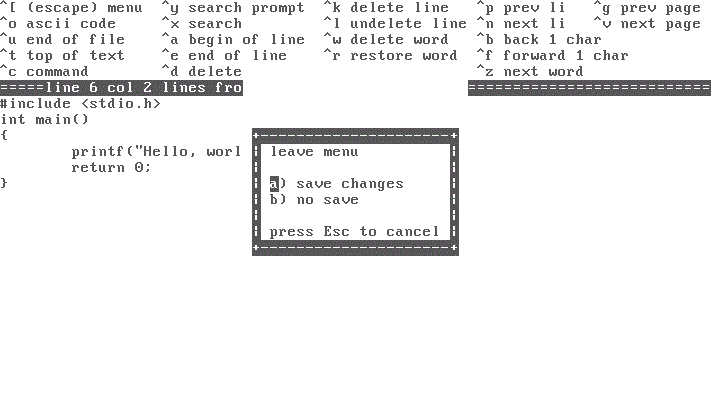


Рис.7.4

Просто выбираем **save changes** (сохранить изменения) и выходим из редактора.

Теперь, когда файл создан, мы можем его просмотреть командой **cat filename**, чтобы удостовериться, что наша программа не содержит ошибок. Команда   
**cat filename** позволяет просматривать содержимое файла, не внося изменений. Если вам нужно будет по какой-либо причине удалить файл, то воспользуйтесь командой **rm filename**.

А сейчас перейдем к компиляции кода программы и для этого воспользуемся компилятором clang++.

Сначала получим объектный файл командой clang++ –c test.c .

Если теперь посмотреть содержимое домашнего каталога, то увидим новый файл *test.o*, что обеспечено опцией *-с*. Получение загрузочного файла из объектного производится программой компоновщиком. Программа *clang++* может выполнить и работу компоновщика командой clang++ –o test test.o .

Созданный загрузочный файл будет называться *test* (без расширения). Опция *–o* позволяет в командной строке задать имя загрузочного файла. Теперь в домашнем каталоге пользователя student лежат три файла (рис.7.5).

C:\Users\1\Desktop\ОС\Изображения\Лаб1\Скрины для новых методичек\Для нового 5 пункта\рис 7.5.GIF

Рис.7.5

Запуск программы из рабочей папки производится командой ./test (рис.7.6).

C:\Users\1\Desktop\ОС\Изображения\Лаб1\Скрины для новых методичек\Для нового 5 пункта\рис 7.6.GIF

Рис.7.6

Мы изучили базовые команды, научились работать с редактором ee и даже написали и запустили небольшую программу. Обучающая часть окончена, и вы можете перейти к выполнению лабораторной работы.

8. ЛАБОРАТОРНОЕ ЗАДАНИЕ

Получите от преподавателя задание на разработку алгоритма и программы на языке программирования С.

Подготовьте наборы тестовых начальных значений и соответствующих результатов, необходимых для проверки правильности разработанной программы.

Используя представленные в описании примеры работы с текстовыми редакторами и справочные материалы, произведите необходимые вычисления на виртуальной машине под управлением установленной ОС, используя команды процессора bash. С некоторыми из них вы уже знакомы, подробнее же можно узнать из приложения 7.

9. СОДЕРЖАНИЕ ОТЧЕТА

* Цель работы.
* Текст варианта задания.
* Расчет тестовых начальных значений и соответствующих результатов, необходимых для проверки правильности разработанной программы.
* Текст программы с подробными комментариями.
* Результаты экспериментирования на ЭВМ.
* Выводы.

10. КОНТРОЛЬНЫЕ ВОПРОСЫ

Что такое виртуальная машина и как она реализована?

Какие этапы включает в себя процесс установки ядра ОС?

Как распределяется дисковое пространство под устанавливаемую ОС и создаются логические разделы внутри дискового пространства?

Какую информацию в себе несет приглашение?

Какие вы знаете наиболее часто используемые команды bash? Что они делают?

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Уэлш М., Далхаймер М.К., Кауфман Л. Запускаем Linux. – Пер. с англ. – СПб: Символ-Плюс, 2001.
2. Таймэн Б. FreeBSD 6. Полное руководство. – М.: Вильямс, 2007.
3. ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/.

ПРИЛОЖЕНИЕ 1

СПИСОК ТЕРМИНОВ

* ядро - это основная часть операционной системы FreeBSD. Оно ответственно за управление памятью, параметры безопасности, работу с сетью, доступ к дискам и многое другое;
* виртуальный жесткий диск типа VHD - обычно, VirtualBOX использует собственный формат виртуальных дисков - файлы Virtual Disk Image (VDI). В частности, этот формат используется, когда вы создаете мастером новую виртуальную машину. Так же VirtualBOX полностью поддерживает работу с популярным открытым форматом VMDK, который используется во многих других программах виртуализации, в частности VMware. Наконец, VirtualBOX также полностью поддерживает формат VHD, применяемый Microsoft;
* программа bsdinstall - программа для установки FreeBSD. Система меню bsdinstall контролируется клавишами навигации, а также Enter, Tab, Space и другими;
* коллекция Портов FreeBSD - это простой и удобный способ установки программ. Она не содержит исходных кодов, необходимых для компиляции приложений. Коллекция Портов - это множество файлов, при помощи которого автоматизируется загрузка, компиляция и установка программных пакетов сторонних разработчиков. Предупреждение: Программа установки не проверяет наличие свободного места. Поэтому выбирайте эту опцию лишь тогда, когда имеется достаточно свободного места на жестком диске. Что касается FreeBSD 9.0, Коллекция Портов занимает около 500 MB дискового пространства. Учтите, что для более новых версий FreeBSD занимаемое Коллекцией Портов дисковое пространство будет расти;
* файловая система ZFS - разработана компанией SunTM, основана на использовании метода пулов устройств хранения данных. Это значит, что емкость носителя занимается только тогда, когда она становится необходимой для сохранения данных. ZFS также была разработана с упором на максимальную целостность данных, поддерживая снимки (snapshot), множество копий и контрольные суммы данных. Новая модель репликации данных, известная как RAID-Z, подобна RAID-5, но специально разработана для предотвращения повреждений данных при записи;
* newfs - утилита используется для инициализации и очистки файловых систем перед первым использованием. Строит файловую систему на указанном специальном файле;
* сетевой интерфейс - физическое или виртуальное устройство, предназначенное для передачи данных между программами через компьютерную сеть. Сетевое взаимодействие Linux-компьютера происходит через сетевые интерфейсы. Любые данные, которые компьютер отправляет в сеть или получает из сети проходят через сетевой интерфейс. Интерфейс определён реализацией модели TCP/IP для того чтобы скрыть различия в сетевом обеспечении и свести сетевое взаимодействие к обмену данными с абстрактной сущностью. Для каждого устройства, поддерживаемого ядром, существует сетевой интерфейс. Существует соглашение о наименовании интерфейсов, в соответствии с которым имя интерфейса состоит из префикса, характеризующего его тип, и числа, соответствующего номеру интерфейса данного типа в системе. Так, например, ppp0 соответствует первому интерфейсу PPP, а eth1 соответствует интерфейсу второго сетевого адаптера Ethernet. Обратите внимание на то, что нумерация интерфейсов начинается с 0;
* сетевой протокол - набор правил и действий (очерёдности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами. Разные протоколы зачастую описывают лишь разные стороны одного типа связи. Названия «протокол» и «стек протоколов» также указывают на программное обеспечение, которым реализуется протокол. Новые протоколы для Интернета определяются IETF, а прочие протоколы - IEEE или ISO. ITU-T занимается телекоммуникационными протоколами и форматами. Наиболее распространённой системой классификации сетевых протоколов является так называемая модель OSI, в соответствии с которой протоколы делятся на 7 уровней по своему назначению - от физического (формирование и распознавание электрических или других сигналов) до прикладного (интерфейс программирования приложений для передачи информации приложениями). Сетевые протоколы предписывают правила работы компьютерам, которые подключены к сети. Они строятся по многоуровневому принципу. Протокол некоторого уровня определяет одно из технических правил связи. В настоящее время для сетевых протоколов используется модель OSI (Open System Interconnection - взаимодействие открытых систем, ВОС);
* IPv4 (англ. Internet Protocol version 4) - четвёртая версия интернет протокола (IP). Первая широко используемая версия. Протокол описан в RFC 791 (сентябрь 1981 года), заменившем RFC 760 (январь 1980 года). IPv4 использует 32-битные (четырёхбайтные) адреса, ограничивающие адресное пространство 4 294 967 296 () возможными уникальными адресами. Традиционной формой записи IPv4 адреса является запись в виде четырёх десятичных чисел (от 0 до 255), разделённых точками. Через дробь указывается длина маски подсети;
* IPv6 (англ. Internet Protocol version 6) - новая версия протокола IP, призванная решить проблемы, с которыми столкнулась предыдущая версия (IPv4) при её использовании в Интернете, за счёт использования длины адреса 128 бит вместо 32. Протокол был разработан IETF. После того, как адресное пространство в IPv4 закончится, два стека протоколов - IPv6 и IPv4 - будут использоваться параллельно (англ. dual stack), с постепенным увеличением доли трафика IPv6, по сравнению с IPv4. Такая ситуация станет возможной из-за наличия огромного количества устройств, в том числе устаревших, не поддерживающих IPv6 и требующих специального преобразования для работы с устройствами, использующими только IPv6;
* DHCP (англ. Dynamic Host Configuration Protocol - протокол динамической настройки узла) - сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер». Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP, и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок. Протокол DHCP используется в большинстве сетей TCP/IP;
* Live CD - FreeBSD Live CD находится на том же CD диске, что и установочная программа. Это удобно для тех пользователей, которые всё еще размышляют о пригодности для них ОС FreeBSD и желают проверить некоторые функциональные возможности до начала установки. При работе с Live CD следует учесть следующее: для получения доступа к системе необходимо осуществить аутентификацию. Допустимое имя пользователя - root, пароль - пустой. Так как система работает непосредственно с CD, производительность будет заметно ниже чем у системы, установленной на жесткий диск. Live CD предоставляет в распоряжение командную строку, а не графический интерфейс;
* точки монтирования - объяснение системе, как добраться до ваших данных и сделать их доступными для использования. Системе нужно объяснить три простые вещи: какая файловая система на разделе, какой файл-устройство вам нужно, куда его подключить для просмотра - точка монтирования. Каталог, в котором вы будете просматривать содержимое ваших разделов, называется точкой монтирования (mount point). Поэтому нужно объяснить системе (командой или через графический интерфейс) - что вы хотите смонтировать, куда и что за файловая система на этом разделе;
* терминал - окружение, где можно вводить команды и получать на них ответ, это может быть физический терминал или терминал на компьютере;
* патч - это дополнительная программа, которая предназначена для изменения основной программы;
* утилита - вспомогательная компьютерная программа в составе общего программного обеспечения для выполнения специализированных типовых задач, связанных с работой оборудования и ОС;
* форматирование диска - процесс разметки магнитного диска (жёсткого диска, дискеты). Операция, в процессе которой магнитная поверхность диска разбивается на сектора и дорожки, которые заполняются пустыми данными.

ПРИЛОЖЕНИЕ 2

СХЕМЫ РАЗБИЕНИЯ ЖЕСТКОГО ДИСКА

П. 2. 1. APM

APM (Apple Partition Map), используемая на PowerPC(R) Macintosh(R). Схема разделов Apple Partition Map (APM) была представлена на Macintosh II в 1987 году. Её дизайн был хорош, и она продержалась с небольшими изменениями на протяжении двадцати лет. Но за последние годы начали проявляться её ограничения. Различные переменные, адресующие блоки, описаны как 32-битные unsigned long, в итоге со стандартным размером блока 512 байт максимальный размер диска может быть 2 TB. Диски размером 2 TB уже доступны даже в сегменте desktop-компьютеров и 32-битное ограничение уже достигнуто. Поэтому Apple заранее внедрил новую систему GUID partition table (GPT).

В отличие от DOS APM не содержит загрузочного кода (вспомним MBR в первом секторе диска под DOS). Загрузчики Open Firmware и EFI сами содержат код для загрузки с дисков APM. Также, в отличие от DOS, APM может описывать столько разделов, сколько требуется (в DOS же есть 4 Primary partition, один из которых может быть описан как Extended partition, а уже в нём можно создавать достаточно большое количество разделов).

Вот пример типичной раскладки "UNIVERSAL HD" с поддержкой Mac OS 9 (рис.П.2.1).

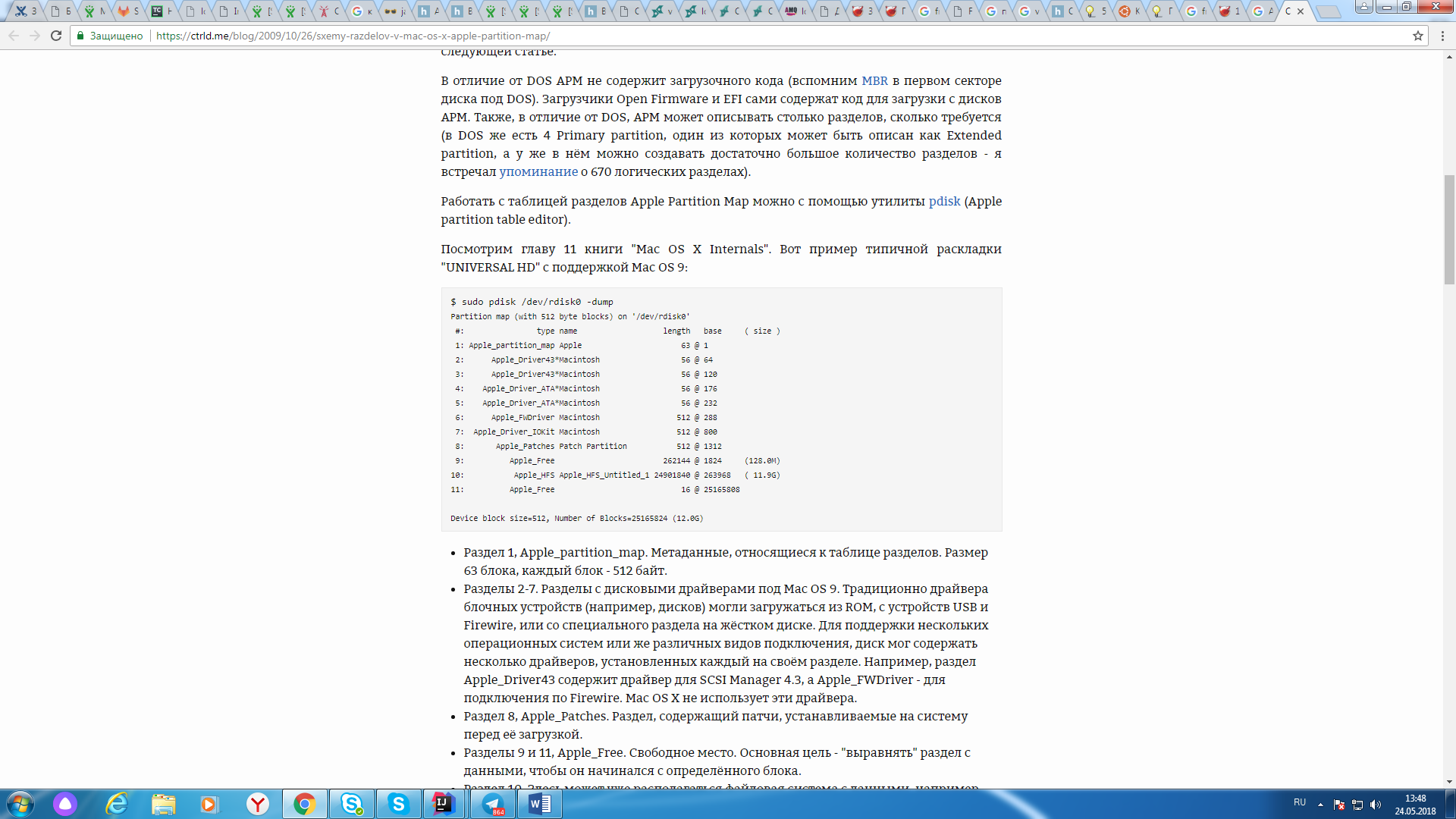


Рис.П.2.1

* раздел 1, Apple\_partition\_map - метаданные, относящиеся к таблице разделов. Размер 63 блока, каждый блок - 512 байт;
* разделы 2-7 - разделы с дисковыми драйверами под Mac OS 9. Традиционно драйвера блочных устройств (например, дисков) могли загружаться из ROM, с устройств USB и Firewire, или со специального раздела на жёстком диске. Для поддержки нескольких операционных систем или же различных видов подключения, диск мог содержать несколько драйверов, установленных каждый на своём разделе. Например, раздел Apple\_Driver43 содержит драйвер для SCSI Manager 4.3, а Apple\_FWDriver - для подключения по Firewire. Mac OS X не использует эти драйвера;
* раздел 8, Apple\_Patches - раздел, содержащий патчи, устанавливаемые на систему перед её загрузкой;
* разделы 9 и 11, Apple\_Free - свободное место. Основная цель - "выравнять" раздел с данными, чтобы он начинался с определённого блока;
* раздел 10 - здесь может уже располагаться файловая система с данными, например, HFS.

П. 2. 2. BSD

Метки BSD без MBR, иногда называемые <<dangerously dedicated mode>>. Утилита bsdlabel устанавливает, проверяет или изменяет метку BSD раздела на диске или в файле, содержащем образ раздела. Кроме того, bsdlabel может установить на диск код загрузчика.

П. 2. 3. GPT

GUID (Globally Unique Identifier) - статистически уникальный 128-битный идентификатор.

GUID Partition Table (GPT) - стандарт формата размещения таблиц разделов на физическом жестком диске. Он является частью Расширяемого микропрограммного интерфейса (англ. Extensible Firmware Interface, EFI) - стандарта, предложенного Intel на смену BIOS. EFI использует GPT там, где BIOS использует Главную загрузочную запись (англ. Master Boot Record, MBR).

В отличие от MBR, которая начинается с исполняемой двоичной программы, призванной идентифицировать и загрузить активный раздел, GPT опирается на расширенные возможности EFI для осуществления этих процессов. Однако MBR присутствует в самом начале диска (блок LBA 0) как для защиты, так и в целях совместимости. Собственно, GPT начинается с Оглавления таблицы разделов (англ. Partition Table Header).

GPT использует современную систему адресации логических блоков (LBA) вместо применявшейся в MBR адресации «Цилиндр - Головка - Сектор» (CHS). MBR, доставшаяся по наследству со всей своей информацией, содержится в блоке LBA 0, оглавление GPT - в блоке LBA 1. В оглавлении содержится адрес блока, где начинается сама таблица разделов, обычно это следующий блок - LBA 2. Количество разделов не ограничено стандартом и зависит от операционной системы (технически ограничено порядка 264 разделами из-за разрядности полей). Так в Microsoft Windows в таблице разделов резервируется место для 128 записей по 128 байт каждая (в GNU/Linux ядро поддерживает до 256 разделов). Таким образом для таблицы разделов в Windows резервируется 16 384 байт (при использовании сектора размером 512 байт - это будет 32 сектора), так что первым используемым сектором каждого жёсткого диска в ней будет блок LBA 34.

Кроме того, GPT обеспечивает дублирование - оглавление и таблица разделов записаны как в начале, так и в конце диска.

Теоретически, GPT позволяет создавать разделы диска размером до 9,4 ЗБ (9,4 × 1021 байт) (при размере сектора 512 байт, иначе - больше), в то время как MBR может работать только до 2,2 ТБ (2,2 × 1012 байт).

GPT позволяет назначать разделам идентификаторы GUID, имена и атрибуты, независимо от внутренних UUID файловых систем, их меток и прочего, и позволяет обращаться по таким именам вместо менее удобных меток и номеров разделов. Благодаря поддержке Юникода в именах и щадящих ограничений на них, разделы могут быть названы на любом языке и сгруппированы по папкам.

Ниже приведена диаграмма, схематично поясняющая формат GUID Partition Table (рис.П.2.2). Каждый логический блок (LBA) имеет размер ровно 512 байт. Отрицательные адреса логических блоков говорят о том, что их нумерация начинается с конца тома (диска), причем последний адресуемый блок имеет адрес −1.

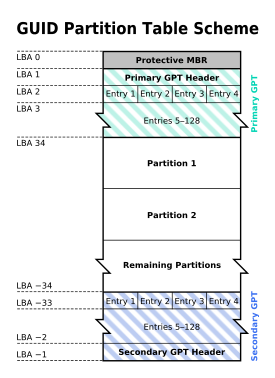


Рис.П.2.2

П. 2. 5. MBR

Главная загрузочная запись (англ. master boot record, MBR) - код и данные, необходимые для последующей загрузки операционной системы и расположенные в первых физических секторах (чаще всего в самом первом) на жёстком диске или другом устройстве хранения информации. Применялась с 1983 года (начиная с PC DOS 2.0) до широкого внедрения UEFI и схемы GPT в 2010-х.

MBR содержит небольшой фрагмент исполняемого кода, таблицу разделов диска (англ. partition table) и специальную сигнатуру.

Функция MBR - «переход» в тот раздел жёсткого диска, с которого следует исполнять «дальнейший код» (обычно - загружать ОС). На «стадии MBR» происходит выбор раздела диска, загрузка кода ОС (происходит на более поздних этапах алгоритма).

В процессе запуска компьютера, после окончания начального теста (Power-on self-test - POST), Базовая система ввода-вывода (BIOS) загружает «код MBR» в оперативную память (в IBM PC обычно с адреса 0000:7c00) и передаёт управление находящемуся в MBR загрузочному коду.

В процессе загрузки компьютера x86 вначале всегда отрабатывается BIOS. На этой стадии, кроме тестирования и инициализации оборудования компьютера, происходит также и выбор устройства, с которого будет происходить дальнейшая загрузка. Это может быть дискета, жёсткий диск, сетевой ресурс, встроенное ПЗУ или любое иное устройство (алгоритм выбора загрузочного устройства может быть различным и зависит от реализации BIOS). После выбора загрузочного устройства BIOS полностью передаёт этому устройству управление всей дальнейшей загрузкой.

В случае, если устройство имеет только один раздел (как, например, дискета или сетевая загрузка), то выбор однозначен, и загрузка продолжается сразу с этого устройства. Однако, если устройство содержит несколько разделов, каждый из которых потенциально может быть загрузочным (как, например, в случае жёстких дисков), то возникает неопределённость: с какого именно раздела производить загрузку. Для разрешения неоднозначности по выбору раздела было предложено вынести этот вопрос из ведения BIOS и передать этот выбор самому устройству. Возникла идея использовать для этого небольшую программу, записанную на самом носителе, которая и осуществляла бы данный выбор. Так появилась концепция MBR.

Таким образом, потенциальное наличие нескольких загрузочных разделов, среди которых необходимо осуществить выбор - это ключевой момент в необходимости появления и отработки MBR. Для устройств с единственным (или однозначно заданным) загрузочным разделом концепция MBR лишена смысла и не используется.

Ниже описан алгоритм загрузки компьютера с использованием MBR.

BIOS (до MBR):

* BIOS проводит начальную инициализацию оборудования (POST);
* BIOS определяет, с какого устройства производить дальнейшую загрузку: дискета, флеш-накопитель, жёсткий диск и т. д. (выбор устройства зависит от версии и от настроек BIOS). В данном описании рассматривается только случай загрузки с жёсткого диска);
* BIOS считывает один сектор (512 байт), т.е. непосредственно MBR, который находится по адресу: «цилиндр 0, головка 0, сектор 1», и помещает его в область памяти по физическому адресу 0x7C00;
* BIOS проверяет, что этот сектор оканчивается сигнатурой 55ААh (если это не так, то управление возвращается обратно в BIOS);
* BIOS передаёт управление по физическому адресу 0x7C00 (то есть сектору MBR), предварительно записав в регистр DL номер диска, с которого этот сектор считан. Для первого жёсткого диска это значение будет равно 80h (128 в десятичной системе), для дисковода A: равно 0. Кроме того, Plug-n-Play BIOS может записать в регистры ES:DI указатель на структуру «$PnP».

Выбор загрузочного раздела и проверка целостности MBR:

* MBR копирует себя на другой адрес, чтобы освободить место для загрузчика ОС (к примеру MBR фирмы Microsoft копирует себя на адрес 0000:0600);
* MBR просматривает по очереди все записи о разделах и ищет первую запись об «активном» («загрузочном») разделе (то есть ищет раздел, отмеченный как 80h);
* в случае успеха (раздел, помеченный как 80h - найден) MBR запоминает номер этого раздела (если просмотрены все 4 записи и не найден раздел, помеченный как 80h, то вызывается INT 18h (отображается сообщение об ошибке посредством вызова BIOS программного прерывания 18h). Это возвращает управление обратно в BIOS, что может приводить либо к загрузке BASIC, либо к повторной попытке загрузить систему с диска, либо к перезагрузке компьютера - в зависимости от версии и реализации BIOS);
* MBR просматривает все оставшиеся записи и проверяет, что это единственный активный раздел (что больше разделов, помеченных 80h, на данном физическом диске не существует). Если находятся другие разделы, помеченные 80h (и/или хотя бы один раздел содержит неправильную метку), то MBR выводит сообщение об ошибке (обычно это что-то типа «Invalid partition table»), после чего система зависает в бесконечном цикле, из которого можно выйти только перезагрузкой компьютера.

На этом заканчивается проверка MBR и начинается подготовка к загрузке ОС:

* MBR считывает первый сектор логического диска (VBR - Volume Boot Record, или Volume Boot Sector), помеченного как «загрузочный» (80h), и помещает этот сектор по физическому адресу 0x7C00;
* MBR проверяет, что данный сектор заканчивается сигнатурой 55AAh (если этой сигнатуры в этом месте нет, то выводится сообщение «Missing operating system» и компьютер подвисает, требуется перезагрузка);
* MBR передаёт управление загрузочному сектору выбранного раздела диска.

П. 2. 6. PC98

Разновидность MBR, используемая компьютерами NEC PC-98.

П. 2. 7. VTOC8

Volume Table Of Contents, используемая компьютерами Sun SPARC64 и UltraSPARC.

В архитектуре хранилища мэйнфреймов IBM Volume Table Of Contents или VTOC - это структура данных, которая обеспечивает размещение наборов данных, находящихся на определенном томе диска. Это функциональный эквивалент MBR или GUID (GPT) на персональном компьютере и корневого каталога запоминающего устройства (дискета, usb-флеш-накопитель, жесткий диск) на ПК или миникомпьютере, например на Unix или Linux, на DOS или Windows. В ней перечислены имена каждого блока данных, а также размер, расположение и разрешения. Кроме того, она содержит вхождение каждой области смежного свободного пространства на томе. Третья запись на первой дорожке первого цилиндра любого тома DASD (т. е. пакета дисков) известна как метка тома и должна содержать указатель на расположение VTOC. VTOC добавляется на диск во время инициализации, благодаря программе поддержки устройства инженерных сетей -ICKDSF. Изначально VTOC был разработан для пакетов съемных дисков.

Чтобы найти набор данных, программа обычно опрашивает каталог z/OS, чтобы найти нужный том. Найдя правильный том, используется VTOC, чтобы выяснить, где на диске хранится набор данных.

VTOC состоит из последовательности 140-байтовых записей, известных как блоки управления набором данных (Data Set Control Blocks) или DSCBs. Существует десять типов DSCB.

ПРИЛОЖЕНИЕ 3

РАБОТА С EE(Easy Editor)

Редактор ee с точки зрения интерфейса является гибридным, и все основные действия могут быть выполнены в нём тремя способами: с помощью управляющих клавиш, о которых только что говорилось (и более подробные сведения о которых можно почерпнуть из man ee), через меню или прямыми командными директивам.

Для редактирования (или создания) файла наберите # ee filename.

Главное меню редактора вызывается нажатием клавиши **Escape** и содержит следующие пункты:

a) выйти из редактора;

b) подсказка;

c) операции с файлами;

d) обновить экран;

e) параметры;

f) поиск;

g) разное.

Каждый из указанных пунктов вызывает панель соответствующих подменю. Выбор требуемого пункта осуществляется клавишами управления курсором (Down и Up) или нажатием соответствующей литерной клавиши (например,**a** — для выхода из редактора). Выход из меню осуществляется повторным нажатием клавиши **Escape**. Та же клавиша в любом из подменю возвращает в окно редактирования текста.

Команды для работы с ee:

* help — вывод подробной подсказки;
* read — вставка указанного файла в позицию курсора файла текущего;
* write — запись текущего файла под произвольным именем;
* exit — выход из редактора с записью выполненных изменений;
* quit — выход из редактора без записи изменений;
* line — вывод номера текущей строки и количества символов в ней;
* expand — преобразование табуляции в пробелы;
* noexpand — не преобразование :) табуляции в пробелы;
* file — распечатать файл;
* char — вывод числового значения ASCII-кода текущего символа;
* case — поиск, чувствительный к регистру символов;
* nocase — поиск, не чувствительный к регистру;
* !cmd — выполнение команды шелла внутри редактора; вывод команды можно через "мышиный" буфер перетащить в окно редактирования текста;
* 0-9 и далее — переход на строку с указанным номером.

Дополнительные подсказки по использованию находятся в верхней части экрана, прямо над рабочим полем редактора.

ПРИЛОЖЕНИЕ 4

ИНФОРМАЦИЯ О ПОЛЬЗОВАТЕЛЕ

* Username - имя, которое будет набирать пользователь для входа в систему. Часто оно формируется из объединенных вместе первой буквы имени и фамилии;
* Full name - полное имя пользователя;
* Uid - идентификатор пользователя. Обычно это поле не заполняется, система сама присвоит ему значение;
* Login group - имя группы для этого пользователя. Обычно это поле также не заполняется, система поставит значение по умолчанию;
* Invite user into other groups? - перечень групп, в которые будет внесен пользователь;
* Login class - обычно оставляется пустым для принятия значения по умолчанию;
* Shell - интерактивная оболочка для этого пользователя. В примере была выбрана оболочка tcsh;
* Home directory - домашний каталог пользователя. Как правило, значение по умолчанию является корректным;
* Home directory permissions - права на домашний каталог пользователя. Значение по умолчанию является корректным в большинстве случаев;
* Use password-based authentication? - обычно "yes";
* Use an empty password? - обычно "no";
* Use a random password? - обычно "no";
* Enter password - пароль для этого пользователя. Набираемые символы не отображаются на экране;
* Enter password again - пароль необходимо ввести еще раз (для сверки);
* Lock out the account after creation? - обычно "no".

ПРИЛОЖЕНИЕ 5

ПРОЦЕСС ЗАГРУЗКИ FreeBSD

Процесс включения компьютера и загрузки операционной системы называется процессом первоначальной загрузки или просто загрузкой. Процесс загрузки FreeBSD предоставляет большие возможности по гибкой настройке того, что происходит при запуске системы, позволяя вам выбирать из различных операционных систем, установленных на одном и том же компьютере или даже из различных версий той же самой операционной системы или установленного ядра.

Включение компьютера и запуск операционной системы приводят к интересной дилемме. По определению до запуска операционной системы компьютер не умеет ничего. В том числе и не знает, как запускать программы с диска. Так что компьютер не может запустить программу с диска без операционной системы, но программы операционной системы находятся на диске, но как запустить операционную систему?

В эпоху начала компьютеризации термин начальная загрузка применялся к механизму, используемому для загрузки операционной системы, и затем был сокращен до просто «загрузки».

На оборудовании архитектуры x86 за загрузку операционной системы отвечает BIOS (Basic Input/Output System). Для этого BIOS ищет на жестком диске MBR (Master Boot Record), которая должна располагаться в определенном месте на диске. BIOS может загрузить и запустить MBR, и предполагается, что MBR может взять на себя остальную работу, связанную с загрузкой операционной системы.

Выполняемую часть MBR обычно называют менеджером загрузки (boot manager), в особенности, если она взаимодействует с пользователем. В этом случае менеджер загрузки, как правило, занимает большее пространство на первом треке диска или внутри файловой системы ОС. (Менеджер загрузки иногда называют загрузчиком (boot loader), но во FreeBSD этот термин используется для описания более поздней фазы загрузки). Среди популярных менеджеров загрузки стоит отметить boot0 (он же Boot Easy, стандартный менеджер загрузки FreeBSD), Grub, GAG и LILO. Из перечисленных менеджеров загрузки в MBR помещается только boot0.

Если на вашем диске установлена только одна операционная система, то стандартной MBR будет достаточно. Такая MBR выполняет поиск на диске первого загрузочного (активного) слайса, после чего запускает с этого слайса код загрузки оставшейся части операционной системы. Утилита fdisk по умолчанию устанавливает именно такую MBR, на основе файла /boot/mbr.

Если на ваших дисках установлено несколько операционных систем, то вы можете установить другой менеджер загрузки, который может выдать список различных операционных систем и позволит вам выбрать одну из них для загрузки.

Оставшаяся часть системы начальной загрузки FreeBSD разделяется на три этапа. Первый этап запускается из MBR, и он знает достаточно для перевода компьютера в особое состояние и загрузки второго этапа. Второй этап может делать несколько больше до запуска третьего этапа. Третий этап заканчивает работу по загрузке операционной системы. Работа разделена на эти три этапа, потому что стандарты ПК ограничивают размеры программ, которые могут быть запущены на первом и втором этапах. Последовательное выполнение работ позволяет FreeBSD получить более гибкий загрузчик.

Затем стартует ядро, которое начинает опознавать устройства и выполняет их инициализацию. После завершения процесса своей загрузки, ядро передает управление пользовательскому процессу с именем init, который выполняет проверку дисков на возможность использования. Затем init запускает пользовательский процесс настройки ресурсов, который монтирует файловые системы, выполняет настройку сетевых адаптеров для работы в сети и вообще осуществляет запуск всех процессов, обычно выполняемых в системе FreeBSD при загрузке.

Код MBR или менеджера загрузки время от времени называют нулевой стадией процесса загрузки. MBR для FreeBSD находится в /boot/boot0. Это копия MBR, так как настоящая MBR должна располагаться в специальном месте диска, вне области FreeBSD. boot0 очень прост, так как программа в MBR может иметь размер, не превышающий 512 байт.

Концептуально первый и второй этапы загрузки являются частями одной и той же программы, в одной области диска. Из-за ограничений на объем дискового пространства они были разделены на две, но вы всегда должны устанавливать их вместе. Они копируются инсталлятором или утилитой bsdlabel из общего файла /boot/boot.

Они располагаются вне файловых систем, на первом треке загрузочного слайса, то есть там, где boot0 или любой другой менеджер загрузки ожидает найти программу, которую следует запустить для продолжения процесса загрузки. Количество используемых секторов легко может быть вычислено из размера файла /boot/boot.

boot1 очень прост, так как он не может иметь размер, превышающий 512 байт, и знает лишь о метке диска FreeBSD, хранящей информацию о слайсе, для того, чтобы найти и запустить boot2.

boot2 устроен несколько более сложно, и умеет работать с файловой системой FreeBSD в объёме, достаточном для нахождения в ней файлов, и может предоставлять простой интерфейс для выбора и передачи управления ядру или загрузчику.

Так как загрузчик устроен гораздо более сложно, и дает удобный и простой способ настройки процесса загрузки, boot2 обычно запускает его, однако раньше его задачей был запуск непосредственно самого ядра.

Передача управления загрузчику является последним, третьим этапом в процессе начальной загрузки, а сам загрузчик находится в файловой системе, обычно как /boot/loader. Загрузчик являет собой удобный в использовании инструмент для настройки при помощи простого набора команд, управляемого более мощным интерпретатором с более сложным набором команд.

Во время инициализации загрузчик пытается произвести поиск консоли, дисков и определить, с какого диска он был запущен. Соответствующим образом он задаёт значения переменных и запускает интерпретатор, которому могут передаваться пользовательские команды как из скрипта, так и в интерактивном режиме.

Затем загрузчик читает файл /boot/loader.rc, который по умолчанию использует файл /boot/defaults/loader.conf, устанавливающий подходящие значения по умолчанию для переменных и читает файл /boot/loader.conf для изменения в этих переменных. Затем с этими переменными работает loader.rc, загружающий выбранные модули и ядро.

И наконец, по умолчанию загрузчик выдерживает 10-секундную паузу, ожидая нажатия клавиши, и загружает ядро, если этого не произошло. Если ожидание было прервано, пользователю выдается приглашение, которое воспринимает простой набор команд, с помощью которых пользователь может изменить значения переменных, выгрузить все модули, загрузить модули и окончательно продолжить процесс загрузки или перезагрузить машину.

ПРИЛОЖЕНИЕ 6

СТРУКТУРА КАТАЛОГОВ

Файловая система FreeBSD является ключевым моментом в понимании устройства всей системы. Самым важным понятием является, несомненно, корневой каталог, обозначаемый символом «/». Корневой каталог монтируется самым первым на этапе загрузки и содержит все необходимое, чтобы подготовить систему к загрузке в многопользовательский режим. Корневой каталог также содержит точки монтирования (см. Приложение 1) для остальных файловых систем, которые монтируются во время перехода в многопользовательский режим.

Для того чтобы посмотреть содержимое текущей директории существует команда ls, а чтобы перемещаться по каталогам cd.

Полное описание иерархии файловой системы есть в официальной документации. Здесь же перечислены лишь наиболее важные каталоги:

* / - корневой каталог файловой системы;
* /bin/ - основные утилиты, необходимые для работы как в однопользовательском, так и в многопользовательском режимах;
* /boot/ - программы и конфигурационные файлы, необходимые для нормальной загрузки операционной системы;
* /boot/defaults/ - конфигурационные файлы с настройками по умолчанию, используемые в процессе загрузки операционной системы;
* /dev/ - файлы устройств;
* /etc/ - основные конфигурационные файлы системы и скрипты;
* /etc/defaults/ - основные конфигурационные файлы системы с настройками по умолчанию;
* /etc/mail/ - конфигурационные файлы для систем обработки почты;
* /etc/periodic/ - файлы сценариев, выполняемые ежедневно, еженедельно и ежемесячно;
* /rescue/ - статически собранные программы для восстановления после сбоев;
* /root/ - домашний каталог пользователя root;
* /sbin/ - системные утилиты и утилиты администрирования, необходимые для работы как в однопользовательском, так и в многопользовательском режимах;
* /tmp/ - временные файлы. Содержимое /tmp обычно теряется во время перезагрузки системы;
* /usr/ - большинство пользовательских утилит и приложений;
* /usr/bin/ - пользовательские утилиты и приложения общего назначения;
* /usr/include/ - стандартные заголовочные файлы для языка C;
* /usr/lib/ - файлы стандартных библиотек;
* /usr/local/ - локальные пользовательские приложения, библиотеки, и т.д. Также используется по умолчанию коллекцией портов;
* /usr/ports/ - коллекция портов FreeBSD (опционально);
* /usr/sbin/ - системные утилиты и утилиты администрирования (исполняемые пользователем);
* /usr/src/ - исходные тексты BSD и/или программ.

ПРИЛОЖЕНИЕ 7

КОМАНДЫ FreeBSD

Для первоначальной работы с ОС необходимо знать основные команды терминала (см. Приложение 1), чтобы выполнять поставленные задачи. Команды могут быть как у терминала (exit), так и у системной утилиты (pkg install, install является командой pkg). Также в команды можно передавать параметры, которые обозначаются знаком "-" (ls –l, где –l - параметр). Ниже приведен список основных команд, которые могут понадобиться в первое время использования FreeBSD:

* прервать выполняющуюся команду: ^C (Ctrl-C);
* выйти из shell: exit;
* показать документацию к команде ls. Вместо ls можно подставить любую другую команду: man ls;
* показать текущую дату и время: date;
* показать директорию, в которой сейчас находимся: pwd;
* перейти в директорию /usr/local/etc/: cd /usr/local/etc/. Вместо /usr/local/etc/ можно указать любую другую директорию, в которую есть доступ;
* перейти в каталог на один уровень вверх: cd ..;
* перейти в свою домашнюю директорию: cd ~;
* перейти в корневую директорию: cd /;
* показать содержимое текущей директории: ls;
* показать подробно содержимое текущей директории (в выводе команды указаны даты, права на файлы/директории): ls –l;
* показать список всех файлов, включая скрытые файлы: ls –a;
* создать директорию newdir в папке tmp: mkdir /tmp/newdir/;
* удалить директорию /tmp/newdir/ . Вместо /tmp/newdir/ можете указать нужную вам папку: rmdir /tmp/newdir/;
* прочитать несколько байт файла и определить тип файла: file filename;
* вывести на экран содержимое файла filename: less filename;
* создать файл filename, если он не существует или обновить время создания, если такой файлик имеется: touch filename;
* удалить файл filename: rm filename;
* показать текущее имя пользователя: whoami;
* стать root’ом (пользователь должен быть в группе «wheel»): su;
* добавить пользователя username в группу «wheel» (или отредактируйте /etc/group): pw usermod username -G wheel;
* показать используемое и свободное место на всех примонтированных файловых системах (-h = для большей читабельности для людей, -m = если надо узнать размер в Мб): df, df –h, df –m;
* перезагрузить ОС: reboot;
* выключить ПК, немедленно: shutdown -p now.

ПРИЛОЖЕНИЕ 8

ИНФОРМАЦИЯ ОБ ОБОЛОЧКАХ

Интерпретатор командной строки, или shell (shell - оболочка) - это та программа, которая принимает команды от пользователя и исполняет их.

К функциям оболочки относятся:

* взаимодействие с пользователем (редактирование командной строки, история команд и т.д.);
* обработка (расширение) шаблонов имен («\*», «?» и т.д.);
* перенаправление ввода/вывода команд;
* управление заданиями.

Кроме того, shell - это специализированный язык программирования, в котором есть переменные, конструкции while, if, for и т.д., функции и много чего еще. Он позволяет писать как несложные сценарии для автоматизации повседневных задач, так и довольно сложные программы.

Shell - это не одна конкретная программа. Исторически существует несколько подвидов оболочек; «генеалогическое древо» представлено ниже.

|  |
| --- |
| http://www.inp.nsk.su/%7Ebolkhov/teach/inpunix/03shells.gif |

По умолчанию во **FreeBSD** используется sh для обычных пользователей и csh для root.

Основные оболочки для freebsd (кратко):

1. **Bourne shell**, исполняемый файл: **sh**. До сих пор подавляющее большинство Unix-подобных систем имеют /bin/sh;
2. **Bourne again shell**, исполняемый файл: **bash**. Одна из самых популярных оболочек на сегодняшний день. Стандарт для Linux;
3. **Z shell**, исполняемый файл: **zsh**. Имеет ряд преимуществ перед bash, касающихся в основном работы в интерактивном режиме;
4. **TENEX C Shell**, исполняемый файл: **tcsh**. Является оболочкой по умолчанию в FreeBSD. Самая простая оболочка в настройке.

О выборе какой-то конкретной оболочки очень сложно что-то сказать. Любителям языка С понравится tcsh. Любителям Linux или командной строки UNIX понравится bash. Вообще список функций, поддерживаемых оболочками bash, tcsh, zsh, очень похож, а потому выбор оболочки является скорее делом вкуса, тем более что большинство этих функций даже не будет использовано. Различия существенны лишь для менее распространенных оболочек, которые здесь не рассматриваются.

Если вы все же решили работать не в уже существующей оболочке, то помните, что менять оболочку нужно для конкретного пользователя, а не системную.

Команда # echo $SHELL используется для просмотра текущей оболочки. Для изменения оболочки можно выбрать опцию –s команды chsh. Например, если вы хотите изменить интерпретатор на bash, выполните: chsh –s /usr/local/bin/bash. Выбранная оболочка должна быть в файле /etc/shells. При установке интерпретаторов из коллекции портов это обычно делается автоматически. Если же это не так, вам нужно будет самостоятельно добавить соответствующую строчку в этот файл. Например, если вы установили bash вручную и поместили его в каталог /usr/local/bin, нужно набрать: # echo «/usr/local/bin/bash» >> /etc/shells и лишь после этого использовать команду chsh.

Ниже представлено более подробное описание оболочек.

П. 9. 1. Bash

Как и все Unix-оболочки, он поддерживает автодополнение имён файлов и директорий, подстановку вывода результата команд, контроль за порядком выполнения, операторы ветвления и цикла. Ключевые слова, синтаксис и другие основные особенности языка были заимствованы из sh. Другие функции вроде истории команд были скопированы из csh и ksh. Bash в основном удовлетворяет стандарту POSIX, но с рядом расширений. Кроме того, большинство сценариев SH можно запустить в Bash без изменений. Bash требует дополнительных пользовательских конфигураций, без которых работает хуже, однако эта настройка занимает время.

Особенности:

* оболочка довольно большая и медленная;
* доступны локальные переменные = > можно делать рекурсивные функции;
* можно использовать «!» для отрицания значения;
* могут быть функция и переменная с одним именем;
* поддержка одномерных массивов;
* можно избирательно приостанавливать выполнение процессов и возобновлять их выполнение в дальнейшем;
* можно искать и подставлять команды в списке истории команд;
* и многое другое.

Пример синтаксиса:

*#!/usr/bin/env bash*

T1=’foo’

T2=’bar’

**if** [[ $T1 == “$T2” ]]

**then**

echo 'условие выполняется'

**else**

echo 'условие не выполняется'

**fi**

П. 9. 2. Zsh

Существует мнение, что в zsh нашли свое воплощение все прогрессивные тенденции таких развитых оболочек, как bash и tcsh. В zsh поддерживается весь набор встроенных команд, стандартизированный для POSIX shell, большинство команд из развитых оболочек bash и tcsh, ну и, разумеется, специфичные для zsh команды. При этом zsh считается менее интуитивной оболочкой. Zsh без собственных настроек лучше не использовать, также эта оболочка является модульной: для расширения возможностей необходимо подключить или установить дополнительные модули. Если не знать всех особенностей оболочки (большинство из которых все равно не будут использоваться), то zsh будет неотличим от bash. Поэтому использование данной оболочки не рекомендуется.

Особенности:

* продвинутый автокомплит, например: cd /u/l/sb + tab превратится в cd /usr/local/sbin/;
* есть функция «меню», благодаря которой можно перемещаться по директориям еще и стрелками;
* есть горячие клавиши, которые **можно** настроить под себя;
* присутствует история команд;
* расширенная поддержка переменных и массивов;
* полностью настраиваемый;
* исправление опечаток;
* и многое другое.

Пример синтаксиса:

function title {

if [[ $TERM == “screen” ]]; then

# Use these two for GNU Screen:

print –Nr $’\033k’$1$’\033’\\

print –Nr $’\033]0;’$2$’\a’

elif [[ $TERM == “xterm” || $TERM == “rxvt” ]]; then

# Use this one instead for Xterms:

print –Nr $’\033]0;’$\*$’\a’

fi

}

П. 9. 3. Tcsh

Tcsh объединяет преимущества синтаксиса csh и удобные средства настройки и интерактивной работы bash. Данная оболочка позволяет восстанавливать прошлые команды клавишами со стрелками и редактировать их. Присутствует автозавершение имён файлов по нажатию клавиши табуляции, можно переключаться в каталог, в котором вы были ранее. Данная оболочка настраивается проще других. При прочих равных условиях она идентична оболочке csh.

Особенности:

* доступна навигация по командной строке и редактирование её компонентов с помощью клавиш управления курсором и не только;
* возможность автодополнения не полностью введённых элементов командных конструкций;
* просмотр буфера истории команд, поиск в нём и извлечение команд для повторного исполнения;
* фоновое исполнение команд и средства управления запущенными заданиями;
* средства тонкой настройки поведения оболочки в интерактивном режиме, в том числе, вида приглашения командной строки с целью придания ему наибольшей информативности и (или) компактности;
* и многое другое.

Примеры синтаксиса:

foreach i ( d\* )

switch ( $i )

**case** d?:

echo $i is short

breaksw

default:

echo $i is long

endsw

end

**if** ( $days > 365 ) **then**

echo This is over a year.

Endif

П. 9. 4. Sh

Данная оболочка не требует никаких настроек, в ней можно работать сразу после запуска. Это неплохой вариант для тех, кто после выполнения лабораторных работ не собирается работать с FreeBSD.

Примеры синтаксиса:

**if** [ $days –gt 365 ] **then**

echo This is over a year.

**Fi**

**for** i **in** d\*

**do**

**case** $i **in**

d?) echo $i is short ;;

\*) echo $i is long ;;

**esac**

**done**