МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МУРМАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Методические указания по изучению дисциплины

Б1.О.11.03 Операционные системы
(код и наименование дисциплины)

для направления подготовки 09.03.01 Информатика и вычислительная техника
(код и наименование направления подготовки /специальности)

направленности/профиля Программное обеспечение вычислительной техники и автоматизированных систем
(наименование направленности (профиля) образовательной программы)

Кафедра-разработчик: Цифровых технологий, математики и экономики
(наименование кафедры-разработчика рабочей программы)

Мурманск
2021

Составитель: Возженников А.П., преподаватель каф. ЦТМиЭ

Методические указания по освоению дисциплины рассмотрены и одобрены на заседании кафедры-разработчика

Цифровых технологий, математики и экономики
название кафедры

21.06.2021 протокол № 12
дата

Оглавление

ОБЩИЕ ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Целью дисциплины (модуля) «Операционные системы» является формирование компетенций в соответствии с ФГОС по направлению подготовки 09.03.01 «Информатика и вычислительная техника» и учебным планом в составе ОПОП по направлению подготовки 09.03.01 Информатика и вычислительная техника, направленность (профиль) «Программное обеспечение вычислительной техники и автоматизированных систем».

Задачи:

- систематическое введение в идеи и методы, используемые в современных операционных системах;
- реализация основных механизмов операционных систем на примере ОС Unix.

Процесс изучения дисциплины «Операционные системы» направлен на формирование элементов следующих компетенций:

ОПК-2. Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности

ОПК-5. Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем/

В результате изучения дисциплины обучающийся должен:

Знать:

- современные информационные технологии и программные средства, в том числе отечественного производства при решении задач профессиональной деятельности;
- основы системного администрирования, администрирования СУБД, современные стандарты информационного взаимодействия систем;

Уметь:

- выбирать современные информационные технологии и программные средства, в том числе отечественного производства при решении задач профессиональной деятельности;
- выполнять параметрическую настройку информационных и автоматизированных систем;

Владеть:

- навыками применения современных информационных технологий и программных средств, в том числе отечественного производства, при решении задач профессиональной деятельности;
- навыками инсталляции программного и аппаратного обеспечения информационных и автоматизированных систем;

Данные методические указания по освоению дисциплины «Операционные системы» содержат методические рекомендации к практическим работам, к лабораторным работам и к самостоятельной работе студентов.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Основная литература

1. Пахмурин, Д.О. Операционные системы ЭВМ / Д.О. Пахмурин ; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). – Томск : ТУСУР, 2013. – 255 с. : ил. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=480573

2. Назаров, С.В. Современные операционные системы / С.В. Назаров, А.И. Широков. – Москва : Интернет-Университет Информационных Технологий, 2011. – 280 с. : ил., табл., схем. – (Основы информационных технологий). – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=233197

Дополнительная литература

1. Таненбаум, Э. Современные операционные системы / Э. Таненбаум; пер. А. Леонтьев. - 2-е изд. - Санкт-Петербург [и др.] : Питер, 2004. - 1040 с. : ил. - (Классика computer science). - ISBN 5-318-00299-4. - ISBN 0-13-031358-0 : 375-00.

2. Олифер, В. Г. Сетевые операционные системы / В. Г. Олифер, Н. А. Олифер. - Санкт-Петербург : Питер, 2001. - 544 с. : ил. - ISBN 5-272-00120-6 : 100-00.

3. Гордеев, А. В. Системное программное обеспечение / А. В. Гордеев, А. Ю. Молчанов. - Санкт-Петербург [и др.] : Питер, 2003, 2001. - 736 с. : ил. - ISBN 5-272-00341-1 : 143-80; 126-00; 110-00; 143-80.

4. Кондратьев, В.К. Введение в операционные системы / В.К. Кондратьев. – Москва : Московский государственный университет экономики, статистики и информатики, 2007. – 231 с. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=90922

5. Сафонов, В.О. Основы современных операционных систем / В.О. Сафонов. – Москва : Интернет-Университет Информационных Технологий, 2011. – 584 с. – (Основы информационных технологий). – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=233210

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ РАБОТАМ

Целью практических работ в процессе изучения дисциплины является:

– систематизация и закрепление полученных теоретических знаний и практических умений студентов;

– углубление и расширение теоретических знаний;

– формирование умений использовать справочную и специальную литературу;

– развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;

– формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;

– развитие исследовательских умений.

Объем времени, отведенный на практические занятия, определяется в соответствии с учебным планом специальности и рабочей программой учебной дисциплины.

Основные формы проведения практических работ:

– для овладения знаниями: публичное обсуждение студентами во время практического занятия основных особенностей и свойств изучаемых в курсе задач и методов их решения и др.;

– для закрепления и систематизации знаний: подготовка студентами сообщений к выступлению на практическом занятии, проводимом в форме семинара;

– для формирования умений: практическое решение задач с помощью изучаемых в курсе методов, содержательный сравнительный анализ получаемых результатов.

Практические занятия проводятся с группами студентов.

Контроль работы студентов на практической работе осуществляется в пределах времени, отведенного на практические занятия по дисциплине с предоставлением студентами фактических результатов выполнения практических заданий.

В качестве форм и методов контроля результатов работы студентов на практических занятиях могут быть использованы тестирование, самоотчеты, контрольные работы и др.

Критерием оценки результатов работы студента на практических работах являются:

– умение студента использовать теоретические знания при выполнении практических задач;

– обоснованность и четкость изложения ответа;

– оформление материала в соответствии с требованиями.

Перечень практических работ:

Практическая работа № 1-2. Установка ОС Unix на виртуальную машину

Практическая работа № 3-4. Настройка виртуального аппаратного обеспечения для ОС Unix

Практическая работа № 5-6. Настройка сетевых подключений в ОС Unix

Практическая работа № 7-8. Администрирование пользователей ОС Unix

Практическая работа № 9-10. Установка и настройка программного обеспечения ОС Unix

Практическая работа № 11-12. Установка и настройка Apache, MySQL и PHP серверов для ОС Unix

Практическая работа № 13-14. Установка и настройка почтового сервера Postfix для ОС Unix

Практическая работа № 15-16. Разработка скриптов bash и csh для ОС Unix

Практическая работа № 17-18. Программирование на языке С в ОС Unix

2.4.1. Загрузка

2.4.1.1. Загрузка i386™

1. Компьютер выключен.
2. Включите компьютер. После включения он должен показать способ входа в меню установки BIOS, как правило это клавиши **F2**, **F10**, **Del**, или **Alt**+**S**. Используете те клавиши, которые показаны на экране. В некоторых случаях компьютер может показывать картинку после запуска. Как правило, нажатие **Esc** уберет картинку и позволит вам увидеть необходимую информацию.
3. Найдите установки системы, указывающие ей с какого устройства загружаться. Обычно они обозначаются как "Boot Order", и там как правило отображен список устройств, таких как `Floppy`, `CDROM`, `First Hard Disk`, и так далее.

   Если вы загружаетесь с CDROM, убедитесь, что он выбран. Если вы загружаетесь с USB-носителя или с дискеты, убедитесь, что выбрано соответствующее устройство. Если вы не уверены, посмотрите руководство к компьютеру и/или к его материнской плате.

   Сделайте изменения, затем сохраните их и выйдите. Компьютер должен перезагрузиться.

4. Если вы подготовили "загрузочный" USB-носитель, как описано в Разд. 2.3.7, вставьте его в USB порт перед включением компьютера.

   Если вы загружаетесь с CDROM, потребуется сначала включить компьютер и вставить компакт-диск, как только это станет возможно.

   Замечание: Для FreeBSD версий 7.*x* и более ранних существуют образы загрузочных дискет. Подготовка и использование дискет описаны в Разд. 2.3.7. Для загрузки компьютера первой в дисковод необходимо вставить дискету, содержащую образ `boot.flp`.

   Если компьютер запускается как обычно, и загружает существующую операционную систему, возможны следующие причины:

   1. Диск был вставлен недостаточно рано в процессе загрузки. Оставьте его внутри и перегрузите компьютер.
   2. Установки BIOS, измененные ранее, действуют неправильно. Надо изменять их, пока они не заработают.
   3. BIOS вашего компьютера не поддерживает загрузку с выбранного типа носителя.
5. FreeBSD начнет загрузку. Если загрузка происходит с CDROM, вы увидите что-то вроде этого (информация о версии удалена):

```
Booting from CD-Rom...

645MB medium detected

CD Loader 1.2
```

```
9.
10.    Building the boot loader arguments
11.    Looking up /BOOT/LOADER... Found
12.    Relocating the loader and the BTX
13.    Starting the BTX loader
14.
15.    BTX loader 1.00 BTX version is 1.02
16.    Consoles: internal video/keyboard
17.    BIOS CD is cd0
18.    BIOS drive C: is disk0
19.    BIOS drive D: is disk1
20.    BIOS 636kB/261056kB available memory
21.
22.    FreeBSD/i386 bootstrap loader, Revision 1.1
23.
24.    Loading /boot/defaults/loader.conf
25.    /boot/kernel/kernel text=0x64daa0 data=0xa4e80+0xa9e40
   syms=[0x4+0x6cac0+0x4+0x88e9d]
26.    \
```

Если происходит загрузка с дискеты, вы увидите что-то вроде этого (информация о версии удалена):

```
Booting from Floppy...
Uncompressing ... done


BTX loader 1.00  BTX version is 1.01
Console: internal video/keyboard
BIOS drive A: is disk0
BIOS drive C: is disk1
BIOS 639kB/261120kB available memory


FreeBSD/i386 bootstrap loader, Revision 1.1


Loading /boot/defaults/loader.conf
/kernel text=0x277391 data=0x3268c+0x332a8 |


Insert disk labelled "Kernel floppy 1" and press any key...
```

Следуя инструкциям, уберите дискету с `boot.flp`, вставьте дискету с `kern1.flp` и нажмите **Enter**. Загрузитесь с первой дискеты; последовательно вставляйте остальные диски при появлении соответствующего приглашения.

27. Идет ли загрузка с CDROM, или с USB-носителя, или с дискеты в процессе загрузки появится меню загрузчика FreeBSD:

Рисунок 2-1. FreeBSD Boot Loader Menu

Подождите десять секунд или нажмите **Enter**.

2.4.1.2. Загрузка Sparc64®

Большинство систем Sparc64® настроены на автоматическую загрузку с жесткого диска. Для того, чтобы установить FreeBSD, вам потребуется выполнить загрузку по сети или с компакт-диска, что в свою очередь требует получения доступа к PROM (OpenFirmware).

Чтобы получить доступ к PROM, перезагрузите систему и дождитесь появления сообщений загрузчика. Последние зависят от модели оборудования, но, в общем, выглядят подобно следующим:

```
Sun Blade 100 (UltraSPARC-IIe), Keyboard Present

Copyright 1998-2001 Sun Microsystems, Inc.  All rights reserved.

OpenBoot 4.2, 128 MB memory installed, Serial #51090132.

Ethernet address 0:3:ba:b:92:d4, Host ID: 830b92d4.
```

Если на данном этапе ваша система продолжает загружаться с диска, то для доступа к PROM вам потребуется нажать на клавиатуре сочетания клавиш **L1**+**A** или **Stop**+**A**, или же — послать `BREAK` на последовательной консоли (например, набрав `~#` в tip(1) или cu(1)). Приглашение PROM выглядит подобно следующему:

```
ok          ❶

ok {0}      ❷
```

❶
Однопроцессорные системы выдают такое приглашение.

❷
Приглашение, используемое многопроцессорными системами: цифра обозначает номер активного процессора.

На этой стадии необходимо вставить CDROM в привод и набрать `boot cdrom` в приглашении PROM.

2.4.2. Просмотр результатов тестирования устройств

Последние несколько сотен линий, отображенные на экране, сохраняются и могут быть просмотрены.

Для просмотра буфера нажмите **Scroll Lock**. Это включит прокрутку экрана. Вы можете использовать клавиши навигации или **PageUp** и **PageDown** для просмотра результатов. Нажмите **Scroll Lock** еще раз для отключения прокрутки.

Сделайте это сейчас для просмотра текста, ушедшего за экран, когда ядро закончило тестирование устройств. Вы увидите текст вроде Рис. 2-2, хотя в деталях он будет отличаться в зависимости от устройств, имеющихся в вашем компьютере.

Рисунок 2-2. Типичный вывод Device Probe

```
avail memory = 253050880 (247120K bytes)

Preloaded elf kernel "kernel" at 0xc0817000.

Preloaded mfs_root "/mfsroot" at 0xc0817084.

md0: Preloaded image </mfsroot> 4423680 bytes at 0xc03ddcd4


md1: Malloc disk

Using $PIR table, 4 entries at 0xc00fde60

npx0: <math processor> on motherboard

npx0: INT 16 interface

pcib0: <Host to PCI bridge> on motherboard

pci0: <PCI bus> on pcib0

pcib1:<VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0
on pci0

pci1: <PCI bus> on pcib1
```

```
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11

isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0

isa0: <iSA bus> on isab0

atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1
on pci0

ata0: at 0x1f0 irq 14 on atapci0

ata1: at 0x170 irq 15 on atapci0

uhci0 <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device
7.2 on pci

0

usb0: <VIA 83572 USB controller> on uhci0

usb0: USB revision 1.0

uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr1

uhub0: 2 ports with 2 removable, self powered

pci0: <unknown card> (vendor=0x1106, dev=0x3040) at 7.3

dc0: <ADMtek AN985 10/100BaseTX> port 0xe800-0xe8ff mem 0xdb000000-
0xeb0003ff ir

q 11 at device 8.0 on pci0

dc0: Ethernet address: 00:04:5a:74:6b:b5

miibus0: <MII bus> on dc0

ukphy0: <Generic IEEE 802.3u media interface> on miibus0

ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto

ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xec00-0xec1f irq 9 at
device 10.

0 on pci0

ed0 address 52:54:05:de:73:1b, type NE2000 (16 bit)

isa0: too many dependant configs (8)

isa0: unexpected small tag 14

orm0: <Option ROM> at iomem 0xc0000-0xc7fff on isa0

fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq2 on
isa0

fdc0: FIFO enabled, 8 bytes threshold

fd0: <1440-KB 3.5” drive> on fdc0 drive 0

atkbdc0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0

atkbd0: <AT Keyboard> flags 0x1 irq1 on atkbdc0

kbd0 at atkbd0

psm0: <PS/2 Mouse> irq 12 on atkbdc0

psm0: model Generic PS/@ mouse, device ID 0
```

```
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on
isa0

sc0: <System console> at flags 0x100 on isa0

sc0: VGA <16 virtual consoles, flags=0x300>

sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0

sio0: type 16550A

sio1 at port 0x2f8-0x2ff irq 3 on isa0

sio1: type 16550A

ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0

pppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode

ppc0: FIFO with 16/16/15 bytes threshold

plip0: <PLIP network interface> on ppbus0

ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master UDMA33

acd0: CD-RW <LITE-ON LTR-1210B> at ata1-slave PIO4

Mounting root from ufs:/dev/md0c

/stand/sysinstall running as init on vty0
```

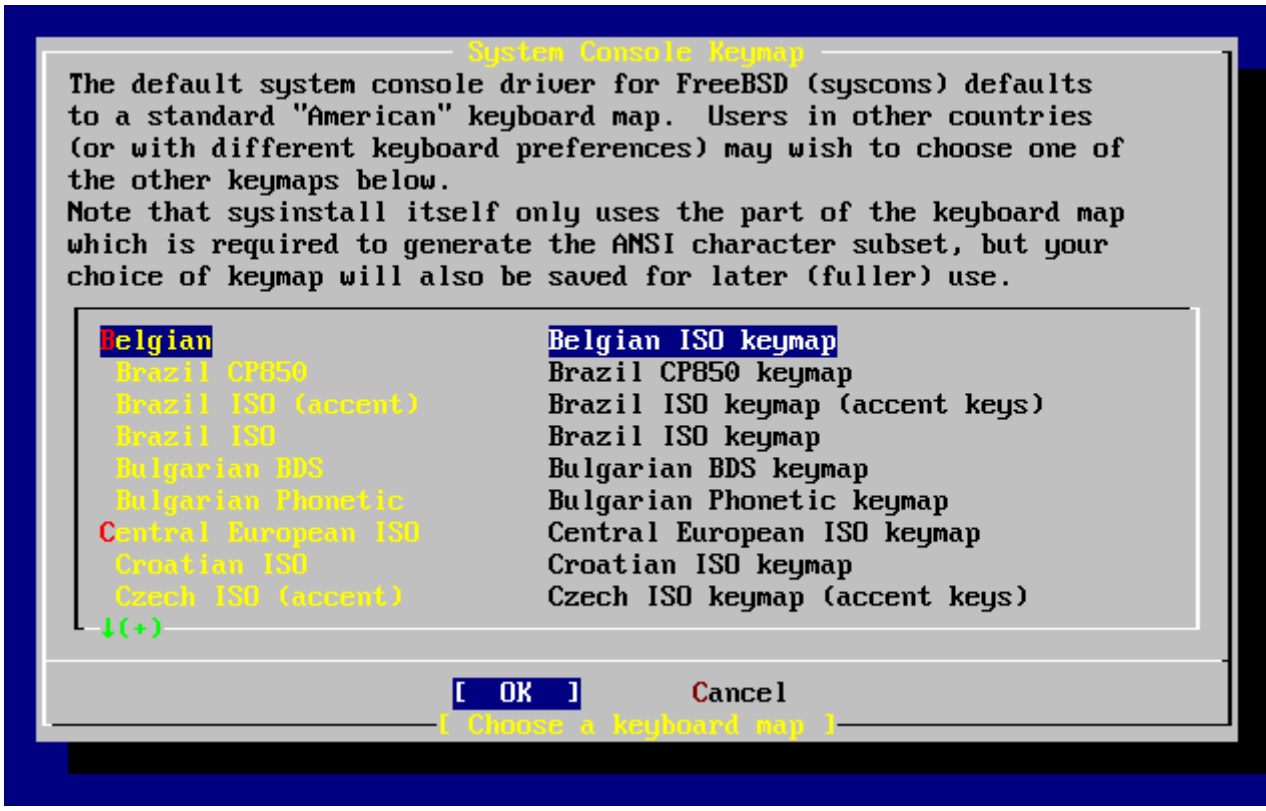
Внимательно проверьте результаты тестирования устройств и убедитесь, что FreeBSD обнаружила все устройства, какие нужно. Если устройство не найдено, его не будет в списке. Собственное ядро позволяет добавлять поддержку устройств, отсутствующих в ядре `GENERIC`, например звуковых карт.

После проверки аппаратных устройств, появится Рис. 2-3. Используйте клавиши навигации для выбора страны, региона или группы. Затем нажмите **Enter**, произойдет выбор страны.

Рисунок 2-3. Меню выбора страны

Если вы выбрали страну **United States**, то будет использована стандартная американская раскладка клавиатуры, если же была выбрана другая страна, то отобразится следующее меню. Используя клавиши навигации, выберите необходимую раскладку и нажмите **Enter**.

Рисунок 2-4. Меню выбора раскладки клавиатуры



После меню выбора страны будет отображено главное меню **sysinstall**.

2.5. Введение в Sysinstall

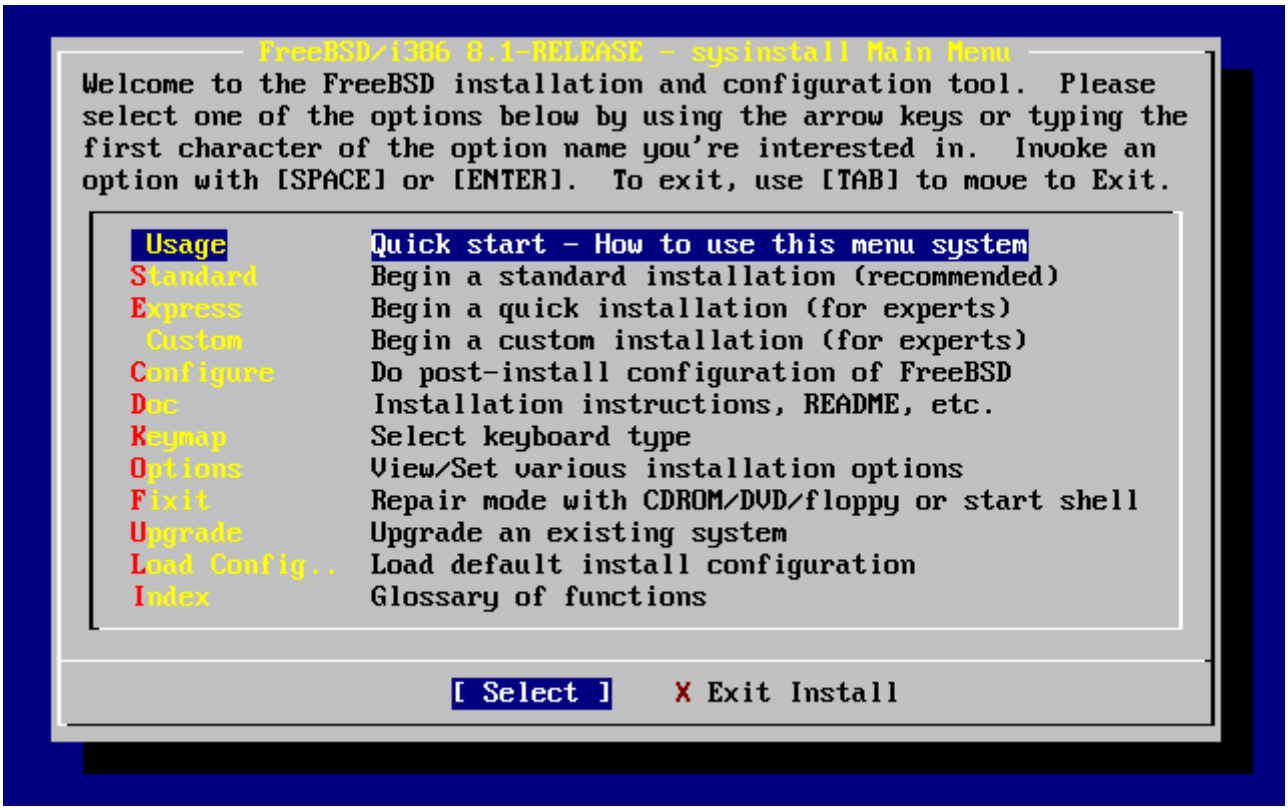
Утилита **sysinstall** это программа установки, предоставляемая проектом FreeBSD. Это консольное приложение, разделенное на несколько меню и экранов, которые вы можете использовать для настройки и управления процессом установки.

Меню **sysinstall** управляется клавишами навигации, **Enter**, **Tab**, пробелом, и другими. Подробное описание клавиш и их функций содержится в информации по использованию **sysinstall**.

Для просмотра этой информации убедитесь, что выбраны пункт **Usage** и кнопка [Select], как показано на Рис. 2-5, затем нажмите **Enter**.

Будут показаны инструкции по использованию меню. После просмотра инструкций, нажмите **Enter** для возврата в главное меню.

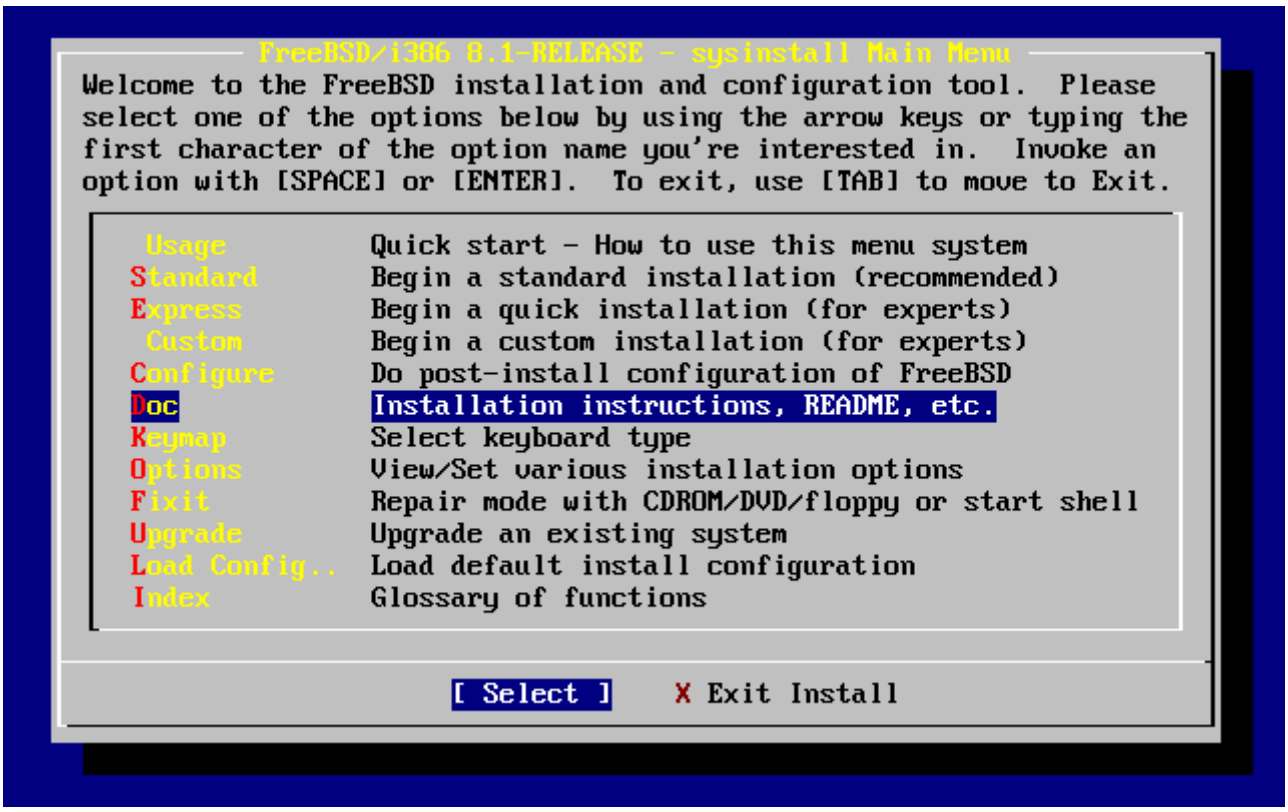
Рисунок 2-5. Выбор Usage в главном меню Sysinstall



2.5.1. Выбор меню документации (Doc)

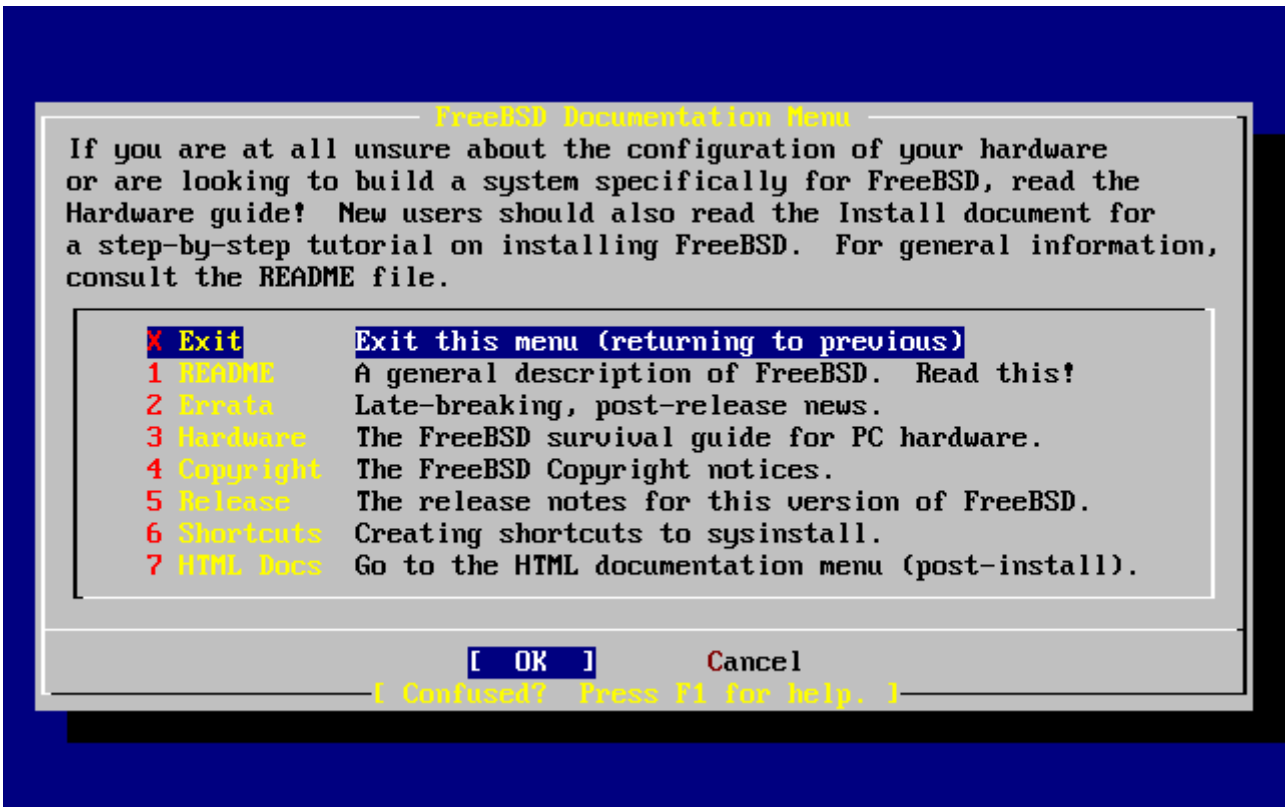
Из главного меню выберите клавишами навигации **Doc** и нажмите **Enter**.

Рисунок 2-6. Выбор меню документации



Будет отображено меню документации.

Рисунок 2-7. Меню документации Sysinstall



Рекомендуется прочитать предоставляемую документацию.

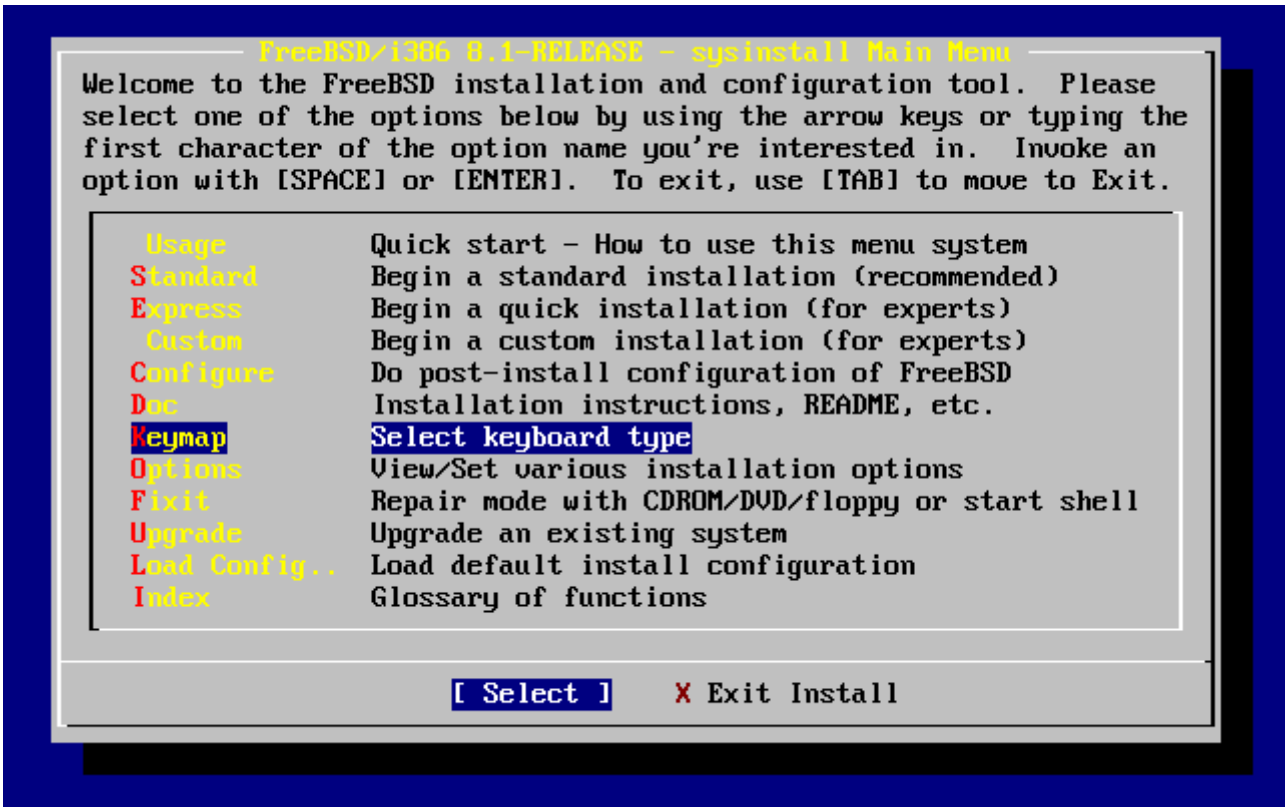
Для просмотра документа выберите его с помощью клавиш навигации и нажмите **Enter**. После прочтения документа нажмите **Enter** для возврата в меню документации.

Для возврата в главное меню выберите **Exit** с помощью клавиш навигации и нажмите **Enter**.

2.5.2. Выбор меню раскладки клавиатуры (Keymap)

Для изменения раскладки клавиатуры выберите из меню с помощью клавиш навигации **Keymap** и нажмите **Enter**. Это потребуется только при использовании нестандартной или не-US клавиатуры.

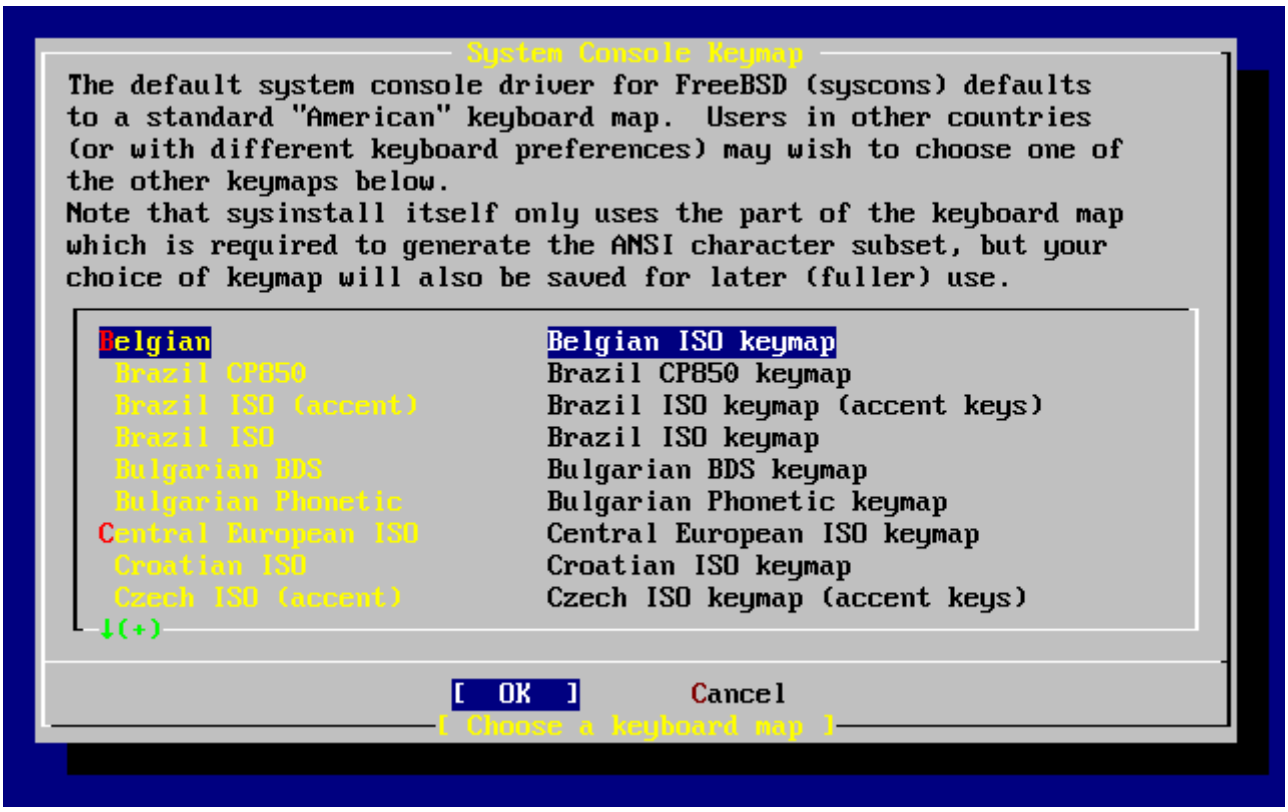
Рисунок 2-8. Выбор меню раскладки клавиатуры



Различные раскладки клавиатуры могут быть выбраны из меню с использованием клавиш навигации, затем следует нажать **Space**. Нажатие **Space** еще раз приведет к отмене выбора. Когда необходимые раскладки будут выбраны, перейдите на [OK] с помощью клавиш навигации и нажмите **Enter**.

На экран выведена только часть списка. Нажав **Tab**, можно выбрать [Cancel], вернуться к раскладке по умолчанию и перейти к главному меню.

Рисунок 2-9. Меню раскладки клавиатуры



2.5.3. Параметры установки (Options)

Выберите пункт **Options** и нажмите **Enter**.

Рисунок 2-10. Выбор параметров установки

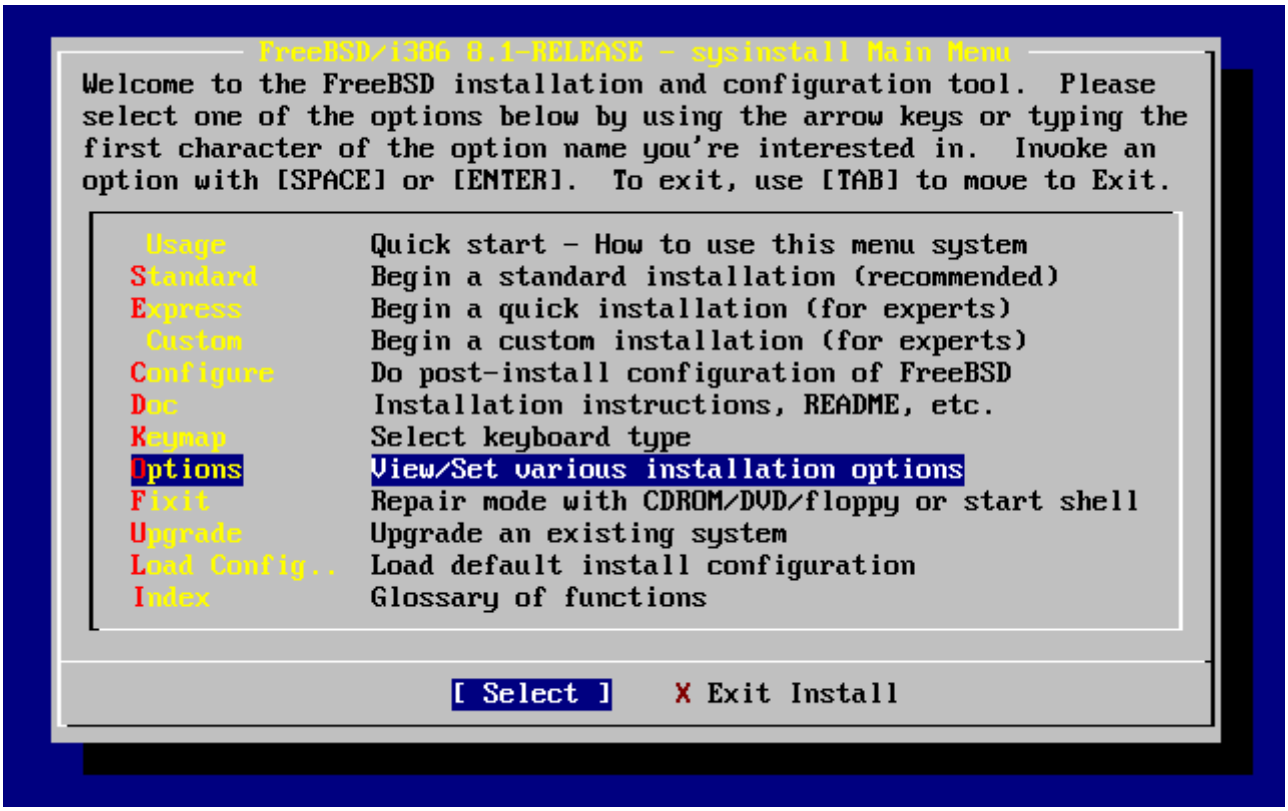
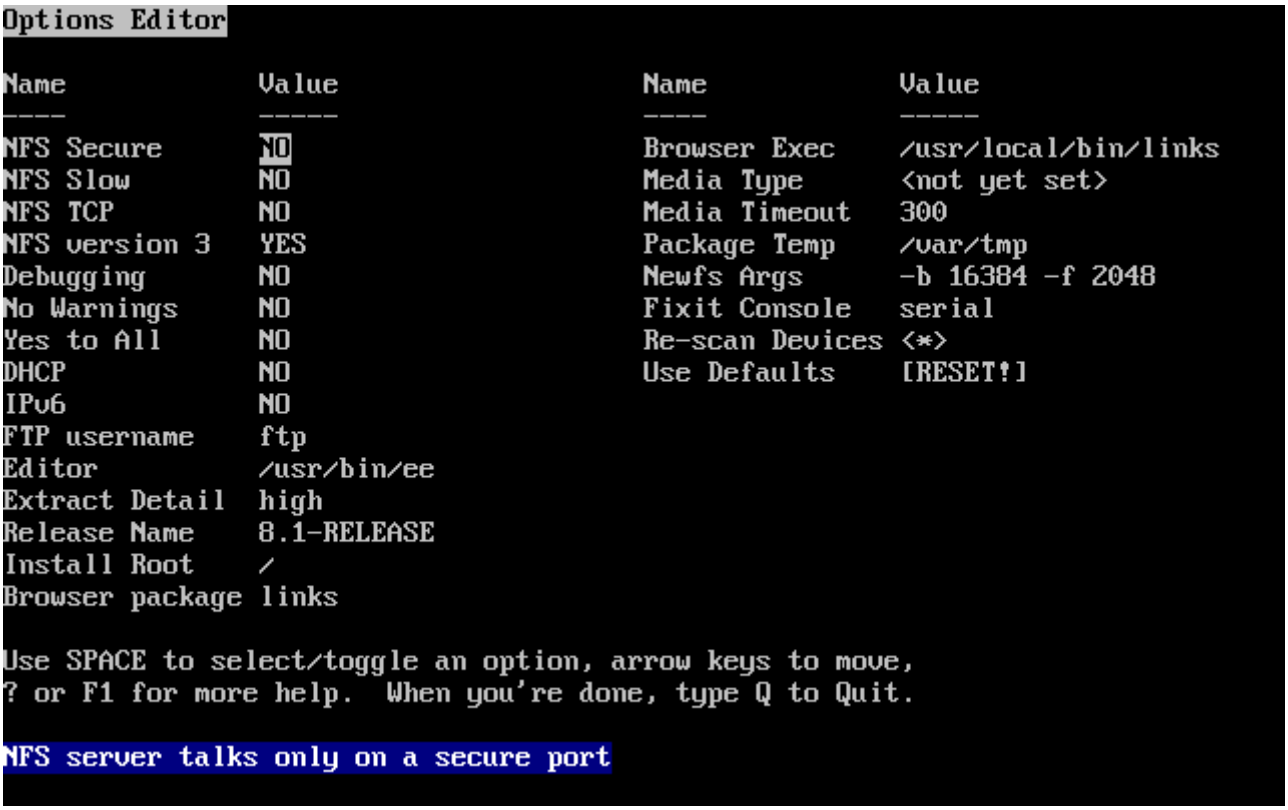


Рисунок 2-11. Параметры Sysinstall



Параметры по умолчанию обычно устраивают большинство пользователей и не нуждаются в изменении. Имя релиза зависит от устанавливаемой версии.

Описание выбранного пункта будет появляться внизу экрана с синей подсветкой. Обратите внимание, что один из параметров — **Use Defaults**, означает сброс всех параметров к значениям по умолчанию.

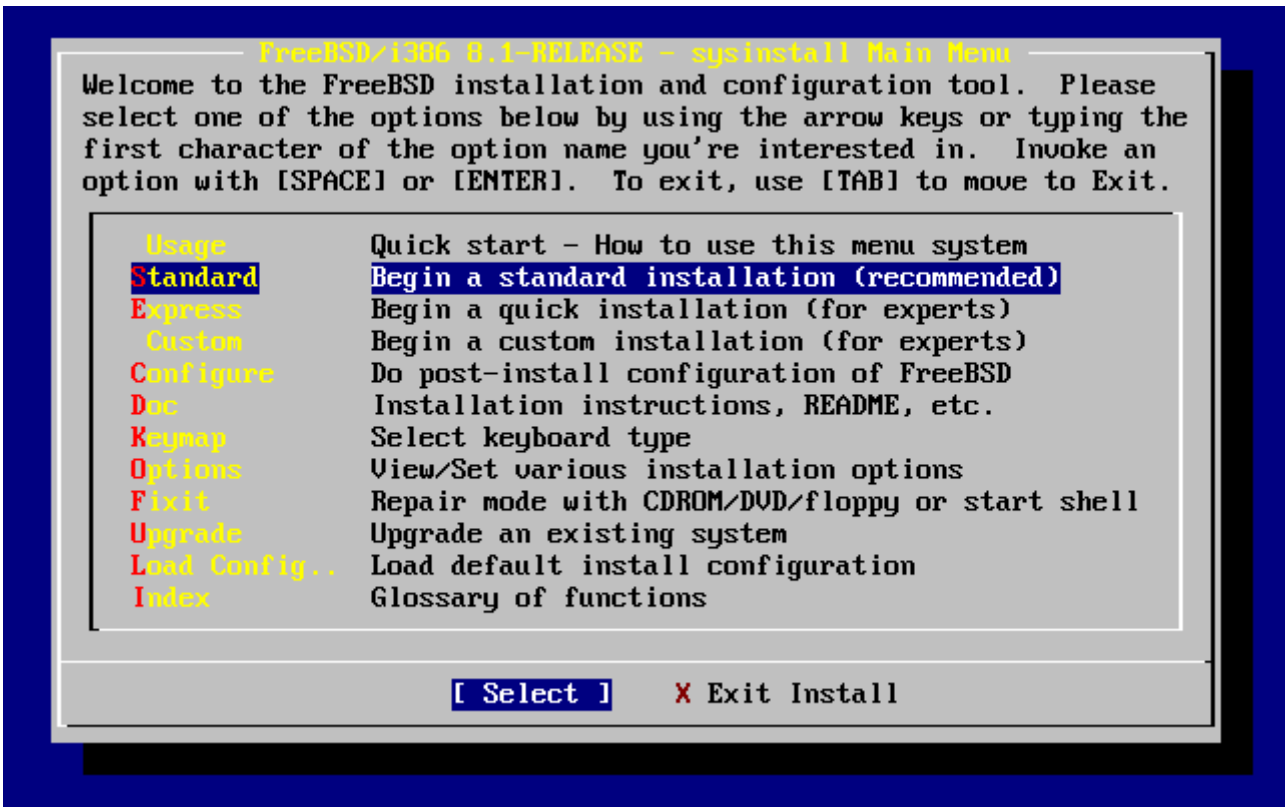
Нажатие **F1** отобразит справку по различным параметрам.

Нажатием **Q** можно перейти к главному меню.

2.5.4. Начало стандартной установки (Standard)

Пункт **Standard** рекомендуется для новых пользователей UNIX® или FreeBSD. Используйте клавиши навигации для выбора пункта **Standard**, а затем нажмите **Enter** для запуска установки.

Рисунок 2-12. Начало стандартной установки



2.6. Выделение дискового пространства

Ваша первая задача — выделить дисковое пространство под FreeBSD и разметить его, чтобы **sysinstall** могла его подготовить. Для этого вам нужно знать, как FreeBSD ищет информацию на диске.

2.6.1. Нумерация дисков в BIOS

Перед установкой и настройкой FreeBSD нужно позаботиться кое о чем, особенно если жестких дисков несколько.

В PC, работающем под BIOS-зависимой операционной системой, такой как MS-DOS® или Microsoft® Windows®, BIOS может отходить от обычного порядка нумерации дисков. Это позволяет пользователю загружаться не только с так называемого "primary master" диска. Это особенно удобно для тех пользователей, кто обнаружил, что простейший и самый дешевый путь делать резервную копию системы — купить второй идентичный первому жесткий диск и регулярно делать копии первого диска на второй, используя **Ghost®** или **XCOPY**. Затем, если первый диск выйдет из строя, будет заражен вирусом или поврежден из-за сбоя операционной системы, он может быть легко восстановлен путем логической перестановки дисков в BIOS. Это все равно что переключить кабели дисков, но без вскрытия корпуса.

Более дорогостоящие системы со SCSI контроллерами зачастую имеют расширения BIOS, позволяющие сходным путем менять порядок до семи SCSI дисков.

Пользователи, привыкшие пользоваться этими полезными функциями, могут быть удивлены, что во FreeBSD результаты не совпадают с ожидаемыми. FreeBSD не использует BIOS, и не знает о "логическом отображении дисков в BIOS". Это может

привести к очень сложным ситуациям, особенно когда диски имеют одинаковую геометрию и содержат точную копию данных друг друга.

При использовании FreeBSD всегда восстанавливайте настройки BIOS к первоначальной нумерации перед установкой системы и оставляйте их в таком виде. Если вам понадобится переключить диски, сделайте это, но путем физического переконфигурирования, вскрыв корпус, переключив перемычки и кабели.

Рассказ о необыкновенных приключениях файлов Билла и Фреда:

Билл разобрал старый Wintel компьютер, чтобы сделать еще один компьютер под FreeBSD для Фреда. Билл установил один SCSI диск как нулевое устройство SCSI и поставил на него FreeBSD.

Фред начал использовать систему, но через несколько дней обнаружил, что старый SCSI диск сообщает о множестве сбоев и сказал об этом Биллу.

Еще через несколько дней Билл решил, что настало время решить проблему, и достал такой же SCSI диск из "заначки" в кладовке. Первая проверка поверхности показала, что диск работает нормально; Билл установил этот диск как четвертое устройство SCSI и скопировал образ диска с нулевого устройства на четвертое. Теперь, когда новый диск был установлен и отлично работал, Билл решил что неплохо бы начать использовать его, и с помощью функции SCSI BIOS поменял порядок дисков, чтобы система могла грузиться с четвертого устройства SCSI. FreeBSD загрузилась и работала без проблем.

Фред поработал еще несколько дней, и скоро они с Биллом решили, что настало время для нового приключения — время обновить версию FreeBSD. Билл удалил нулевое устройство SCSI, потому что оно "подглючивало", и установил на его место такой же диск из "заначки". Затем Билл установил новую версию FreeBSD на новое нулевое устройство SCSI используя дискеты Фреда с интернет сервера FTP. Установка прошла отлично.

Фред использовал новую версию FreeBSD несколько дней и удостоверился, что она вполне подходит для работы в инженерном отделе. Настало время скопировать все архивы со старого диска. Фред смонтировал четвертое устройство SCSI (последнюю копию старой версии FreeBSD) и обнаружил, что ни одного из его драгоценных файлов на четвертом устройстве SCSI нет.

Куда делись данные?

Когда Билл сделал копию с нулевого устройства SCSI на четвертое устройство SCSI, оно стало "клоном". Когда Билл поменял настройки SCSI BIOS, чтобы загрузиться с четвертого устройства SCSI, он всего лишь обманул сам себя. FreeBSD все еще работала с нулевого устройства SCSI. Изменение этих настроек BIOS привело к загрузке части кода Boot и Loader с выбранного в BIOS диска, но после загрузки драйверов FreeBSD настройки BIOS были проигнорированы, и FreeBSD вернулась к нормальной нумерации. Как показано "на пальцах", система продолжила работать с нулевым устройством SCSI, и все данные Фреда остались там, а не на четвертом устройстве SCSI. То, что система грузилась с четвертого устройства SCSI, было всего лишь обманутыми ожиданиями.

Мы рады упомянуть, что данные не были уничтожены или повреждены при нашем исследовании этого феномена. Старое нулевое устройство SCSI было вытащено из груды

железа, и все файлы Фреда вернулись к нему.

Хотя в этом рассказе были использованы SCSI диски, с IDE дисками все точно так же.

2.6.2. Создание слайсов с использованием FDisk

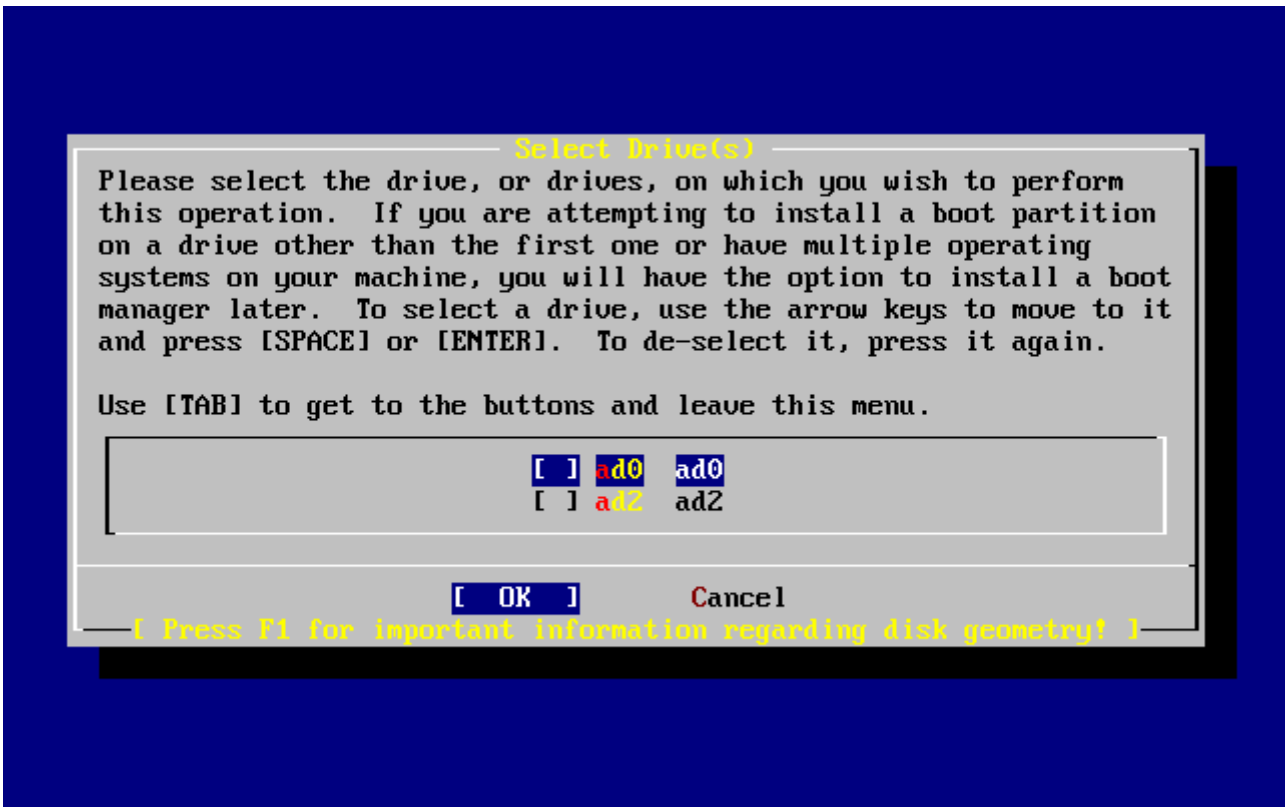
Замечание: Внесенные вами изменения не будут записываться на диск сразу. Если вы думаете, что сделали ошибку, и хотите начать сначала, можете использовать меню для выхода из **sysinstall** и попробовать еще раз или нажатием **U** вызвать опцию **Undo** (отмена). Если вы запутались и не можете выйти, просто выключите компьютер.

После начала стандартной установки в **sysinstall** будет показано это сообщение:

```
                              Message

In the next menu, you will need to set up a DOS-style ("fdisk")
partitioning scheme for your hard disk. If you simply wish to devote
all disk space to FreeBSD (overwriting anything else that might be on
the disk(s) selected) then use the (A)ll command to select the default
partitioning scheme followed by a (Q)uit. If you wish to allocate only
free space to FreeBSD, move to a partition marked "unused" and use the
(C)reate command.

                                [  OK  ]


                      [ Press enter or space ]
```

Нажмите **Enter** как предлагается. Будет показан список всех жестких дисков, обнаруженных ядром во время тестирования устройств. Рис. 2-13 показывает пример системы с двумя IDE дисками. Они были названы `ad0` и `ad2`.

Рисунок 2-13. Выберите диск для FDisk



Вы можете быть удивлены, почему устройства `ad1` здесь нет. Почему оно было пропущено?

Предположим, что у вас есть два жестких диска IDE, один master на первом контроллере IDE, а второй master на втором контроллере IDE. Если FreeBSD пронумерует их в том порядке, в котором нашла, `ad0` и `ad1`, все будет работать.

Но если вы добавите третий диск, как slave устройство на первый контроллер IDE, он станет `ad1`, а предыдущий `ad1` станет `ad2`. Поскольку имена устройств (таких как `ad1s1a`) используются для обращения к файловым системам, вы можете вдруг обнаружить, что некоторые из ваших файловых систем больше не отображаются правильно и вам потребуется изменить конфигурацию FreeBSD.

Для обхода этой проблемы, ядро может быть настроено так, чтобы именовать IDE диски на основе их местоположения, а не порядка, в котором они были найдены. С этой схемой master диск на втором контроллере IDE будет *всегда* устройством `ad2`, если даже нет устройств `ad0` или `ad1`.

Это конфигурация ядра FreeBSD по умолчанию, поэтому на экране показаны `ad0` и `ad2`. У компьютера, с которого был взят этот снимок экрана, есть по одному IDE диску на обеих master каналах IDE контроллеров и ни одного диска на каналах slave.

Вы должны выбрать диск, на который хотите установить FreeBSD, и нажать [OK]. Запустившийся **FDisk** будет выглядеть примерно как Рис. 2-14.

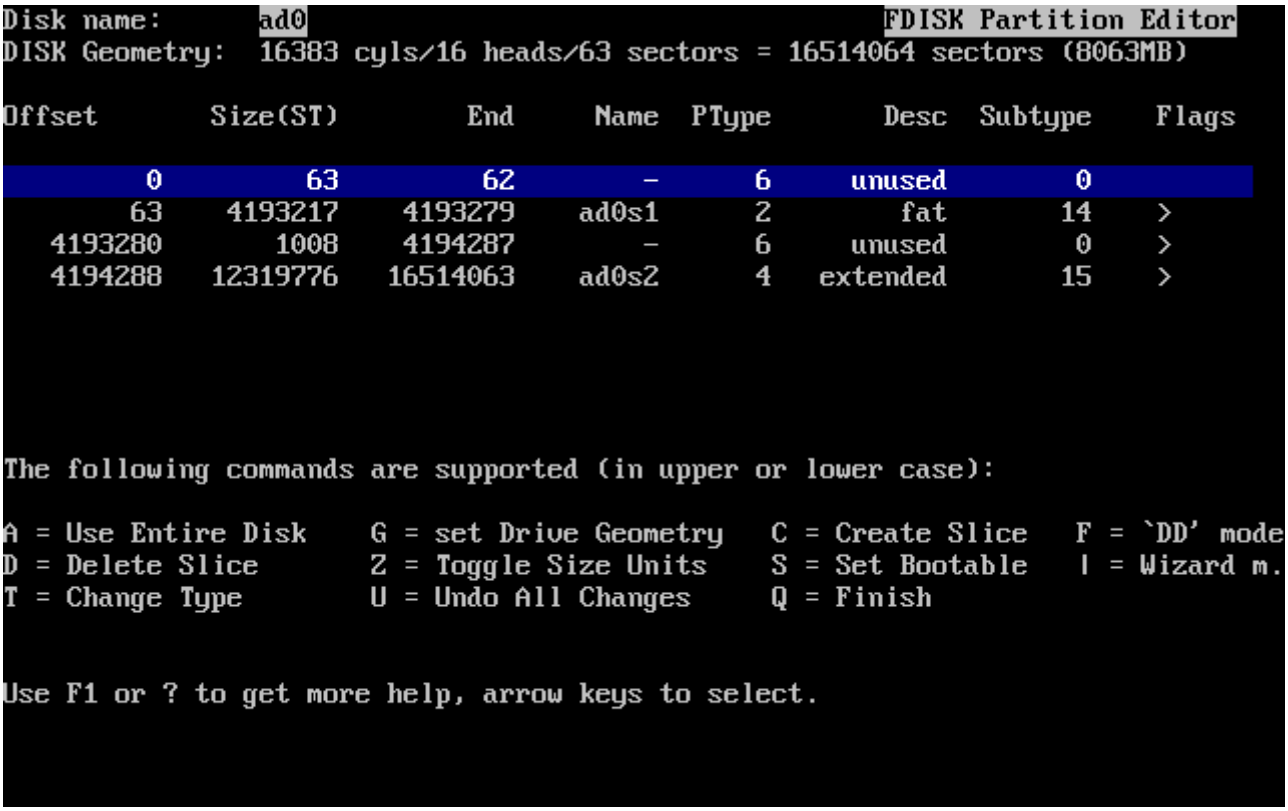
Экран **FDisk** разбит на три раздела.

Первый раздел, занимающая первые две линии экрана, показывает подробную информацию о выбранном в данный момент диске, включая его имя во FreeBSD, геометрию и общий размер диска.

Второй раздел показывает имеющиеся в данный момент на диске слайсы, где они начинаются и заканчиваются, их размер, имя, которое им дала FreeBSD, описание и подтип. На этом примере показаны два маленьких неиспользованных слайса, которые являются артефактами схемы разметки диска на PC. Также показан один большой FAT слайс, который почти всегда является диском `C:` в MS-DOS / Windows, и дополнительный слайс, который может содержать диски с другими буквами для MS-DOS / Windows.

Третий раздел показывает команды, доступные в **FDisk**.

Рисунок 2-14. Типичные разделы `fdisk` перед редактированием



Ваши действия теперь будут зависеть от того, как вы хотите разбить диск на слайсы.

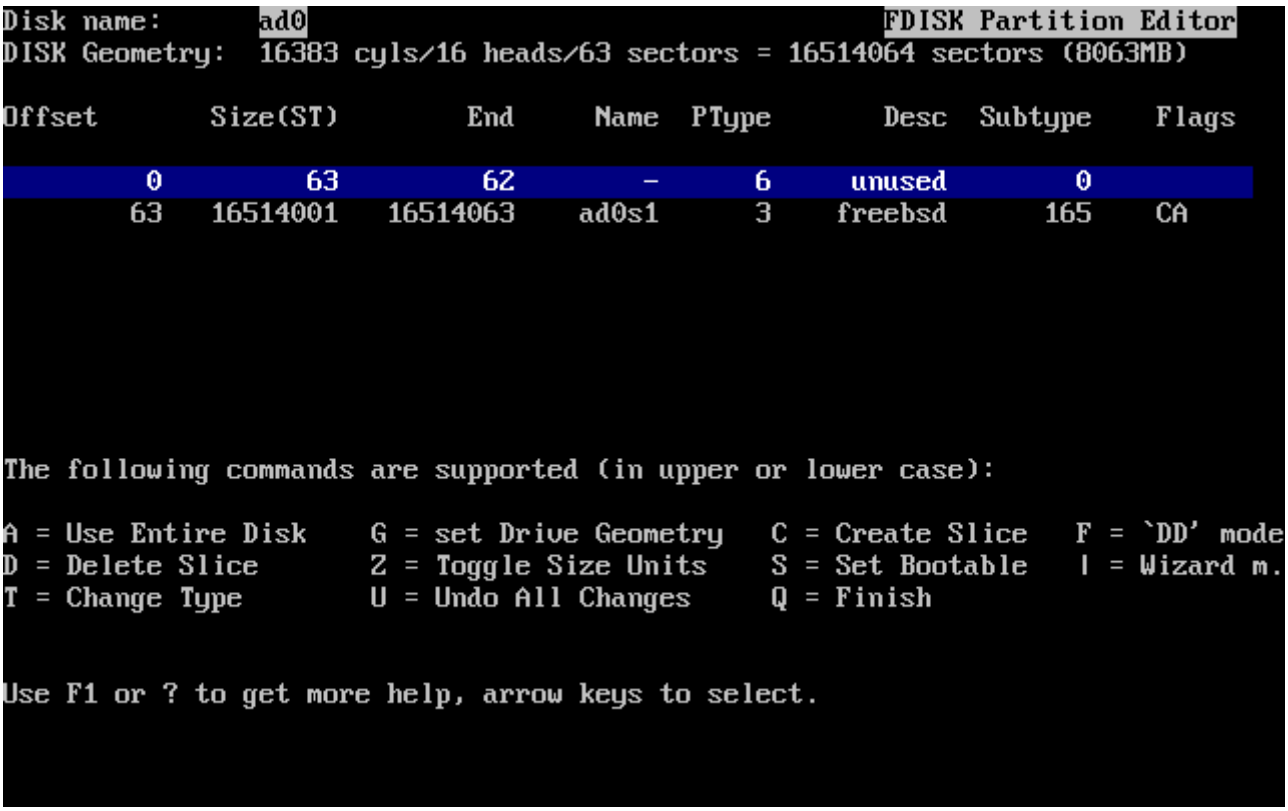
Если вы хотите использовать для FreeBSD весь диск (это приведет к удалению всех других данных на этом диске когда вы подтвердите **sysinstall** продолжение процесса установки), нажмите **A**, что соответствует опции **Использовать весь диск (Use Entire Disk)**. Существующие слайсы будут удалены, и заменены на небольшую область, помеченную как `неиспользуемая (unused)` (это опять же артефакт разметки диска PC), и один большой слайс для FreeBSD. Когда вы сделаете это, нужно выбрать вновь созданный слайс FreeBSD используя клавиши навигации, а затем нажать **S**, чтобы сделать слайс загрузочным. Экран будет похож на Рис. 2-15. Обратите внимание, что `A` в колонке `Flags` означает, что слайс *активен* и с него будет происходить загрузка.

Если вы будете удалять существующий слайс для освобождения места под FreeBSD, выберите слайс, используя клавиши навигации, и нажмите **D**. Затем можете нажать **C**, и

получить приглашение на ввод размера слайса, который вы хотите создать. Введите соответствующее значение и нажмите **Enter**. Значение по умолчанию в этом поле означает наибольший размер слайса, который может быть выбран; это может быть наибольший непрерывный блок неразмеченного пространства или размер всего жесткого диска.

Если вы уже освободили место для FreeBSD (возможно, используя утилиту вроде **PartitionMagic®**), можете нажать **C** для создания нового слайса. Будет также предложено ввести размер слайса, который вы хотите создать.

Рисунок 2-15. Разбиение в Fdisk с использованием всего диска



Когда закончите, нажмите **Q**. Изменения будут сохранены в **sysinstall**, но еще не записаны на диск.

2.6.3. Установка менеджера загрузки (Boot Manager)

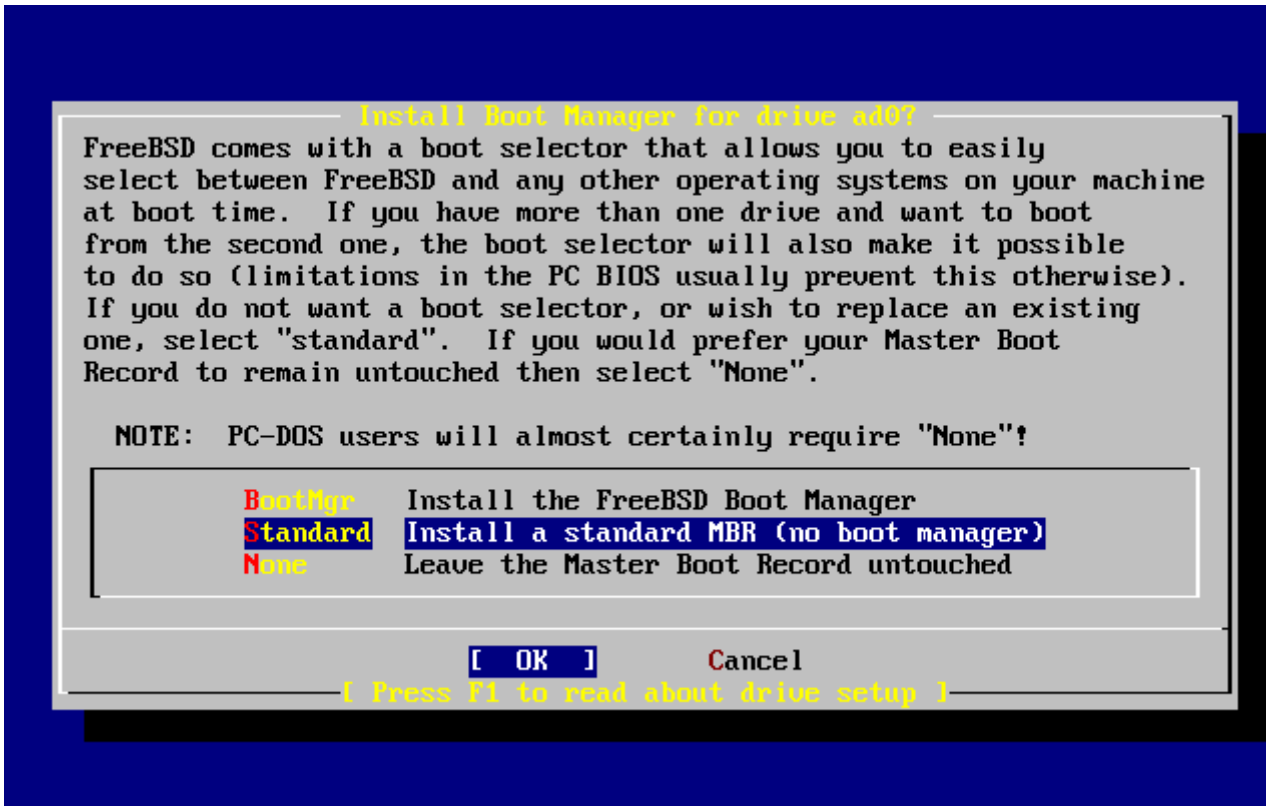
Теперь вам предлагается установить менеджер загрузки. Как правило, нужно выбрать установку менеджера загрузки если:

- У вас больше чем один диск и вы устанавливаете FreeBSD не на первый диск.
- Вы устанавливаете FreeBSD вместе с другой операционной на один и тот же диск, и хотите выбирать при загрузке FreeBSD или другую операционную систему.

Если FreeBSD единственная операционная система, установленная на этом компьютере, и находится на первом жестком диске, подойдет менеджер загрузки **Standard**. Выберите **None** если вы используете менеджер загрузки сторонних разработчиков, способный загрузить FreeBSD.

Сделайте выбор и нажмите **Enter**.

Рисунок 2-16. Меню менеджера загрузки Sysinstall



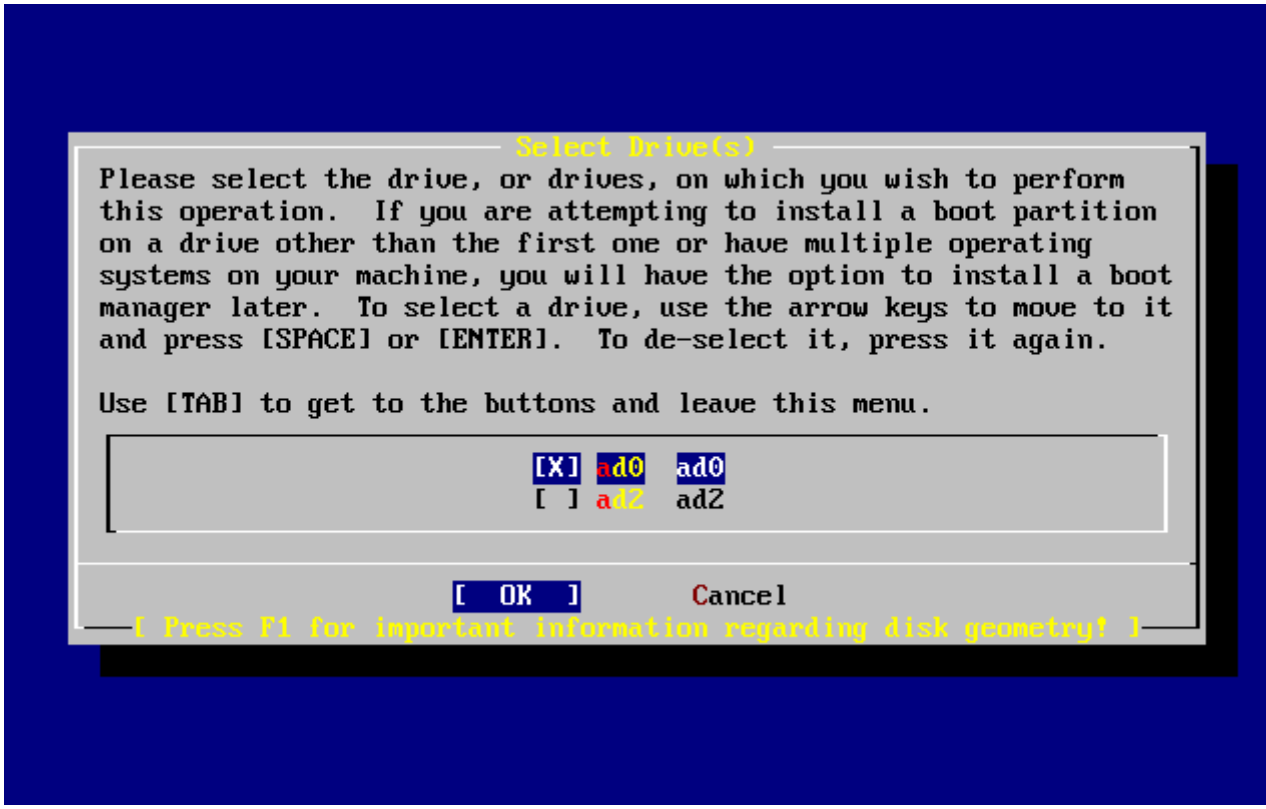
Экран подсказки, вызываемый по нажатию **F1**, описывает проблемы, которые могут быть встречены при попытке совместного использования диска операционными системами.

2.6.4. Создание слайсов на другом диске

Если дисков больше чем один, вернитесь к экрану выбора дисков (Select Drives) после выбора менеджера загрузки. Если вы собираетесь устанавливать FreeBSD более чем на один диск, можете выбрать другой диск и повторить процесс разбиения на слайсы с использованием **FDisk**.

Важно: *Если вы устанавливаете FreeBSD не на первый жесткий диск, потребуется установить менеджер загрузки FreeBSD на оба диска.*

Рисунок 2-17. Выход из выбора диска



Клавиша **Tab** переключает между последним выбранным диском, [OK], и [Cancel].

Нажмите **Tab** один раз для выбора [OK], затем нажмите **Enter** для продолжения установки.

2.6.5. Создание разделов с помощью Disklabel

Теперь вы должны создать несколько разделов внутри каждого только что созданного слайса. Запомните, что у каждого раздела есть буква с `a` до `h`, а разделы `b`, `c`, и `d` имеют соглашения, которых вы должны придерживаться.

Некоторые приложения могут выигрывать от определенных схем разделов, особенно если у вас разделы на более чем одном диске. Тем не менее, для вашей первой установки FreeBSD не нужно слишком углубляться в принципы разбиения диска. Более важно установить FreeBSD и начать ее использовать. Вы всегда можете переустановить FreeBSD для изменения схемы разделов, когда поближе познакомитесь с операционной системой.

Эта схема показывает четыре раздела — один для подкачки и три для файловых систем.

Таблица 2-2. Планирование разделов для первого диска

Раздел	Файловая система	Размер	Описание
a	/	1 GB	Корневая файловая система. Любая другая файловая система будет смонтирована на эту. 1 GB это подходящий размер для этой файловой системы. Вы не будете хранить на ней слишком

Раздел	Файловая система	Размер	Описание
			много данных, а обычная установка FreeBSD разместит здесь около 128 MB данных. Оставшееся пространство используется для временных файлов, а также оставляет возможность расширения для будущих версий FreeBSD, которым может понадобится больше места в `/`.
`b`	N/A	2-3 x RAM	Раздел подкачки находится на разделе `b`. Выбор правильного размера раздела подкачки это немного искусство. Хороший практический способ выбрать размер подкачки это установить его равным двум или трем размерам доступной физической памяти (RAM). Должно быть хотя бы 64 MB подкачки; если в компьютере меньше чем 32 MB памяти — установите размер подкачки равным 64 MB. Если у вас больше одного диска, можно расположить подкачку на каждом диске. FreeBSD будет использовать каждый диск, что серьезно увеличит скорость подкачки. В этом случае, определите общий размер подкачки, который вам нужен (например, 128 MB), и поделите его на число имеющихся дисков (например, два) для определения размера разделов подкачки, которые нужно разместить на каждом вашем диске, в этом примере 64 MB на диск.
`e`	`/var`	512 MB	Каталог `/var` содержит файлы, которые постоянно меняются; логи и другие административные файлы. Многие из этих файлов интенсивно читаются и записываются в процессе ежедневной работы FreeBSD. Размещение их на отдельной файловой системе позволяет FreeBSD оптимизировать доступ к этим файлам без затрагивания других каталогов, не имеющих такой же модели доступа.
`f`	`/usr`	Остальная часть диска (по крайней мере — 8 GB)	Все другие файлы как правило хранятся в каталоге `/usr` и его подкаталогах.

Внимание: Значения, приведённые выше, являются примерными и уместны к использованию лишь опытными пользователями. Остальным — рекомендуется применять опцию автоматического разбиения, называемую `Auto Defaults` в редакторе разделов FreeBSD.

Если вы устанавливаете FreeBSD более чем на один диск, вы должны также создать разделы в других слайсах, которые настроили. Простейший путь сделать это — создать два раздела на каждом диске, один для подкачки, а другой для файловой системы.

Таблица 2-3. Разметка разделов для остальных дисков

Раздел	Файловая система	Размер	Описание
`b`	N/A	Смотрите описание	Как уже обсуждалось, вы можете распространить подкачку на каждый диск. Даже если раздел `a` свободен, соглашение говорит о том, что подкачка находится на разделе `b`.

	Раздел	Файловая система	Размер	Описание
e	/disk*n*		Остальная часть диска	Остальная часть диска занята одним большим разделом. Он легко может быть помещен на раздел `a` вместо раздела `e`. Однако, соглашение говорит, что раздел `a` зарезервирован на слайсе для корневой (`/`) файловой системы. вы можете не следовать этому соглашению, но программа **sysinstall** будет ему следовать, поэтому приняв его, вы сделаете установку несколько проще. Вы можете монтировать эти файловые системы к любой точке; в этом примере предлагается смонтировать их как каталоги `/disk`*`n`*, где *n* это номер, который уникален для каждого диска. Но вы можете использовать другую схему, если захотите.

Теперь, выбрав разметку разделов, можете приступить к их созданию в **sysinstall**. Вы увидите это сообщение:

```
                              Message

Now, you need to create BSD partitions inside of the fdisk
partition(s) just created. If you have a reasonable amount of disk
space (1 GB or more) and don't have any special requirements, simply
use the (A)uto command to allocate space automatically. If you have
more specific needs or just don't care for the layout chosen by
(A)uto, press F1 for more information on manual layout.


                              [  OK  ]

                        [ Press enter or space ]
```

Нажмите **Enter** для запуска редактора разделов FreeBSD, называемого **Disklabel.**

Рис. 2-18 показывает экран только что запущенного **Disklabel**. Экран поделен на три раздела.

Первые несколько линий показывают имя диска, с которым вы сейчас работаете и слайс, содержащий раздел, который вы создаете (здесь **Disklabel** называет это `именем раздела (Partition name)` вместо имени слайса). Этот экран также показывает объем свободного пространства на слайсе, т.е. пространство, выделенное под слайс, но еще не отданное под раздел.

В центре экрана показаны уже созданные разделы, имена файловых систем, содержащихся в разделах, их размер и некоторые опции, применяемые при создания файловых систем.

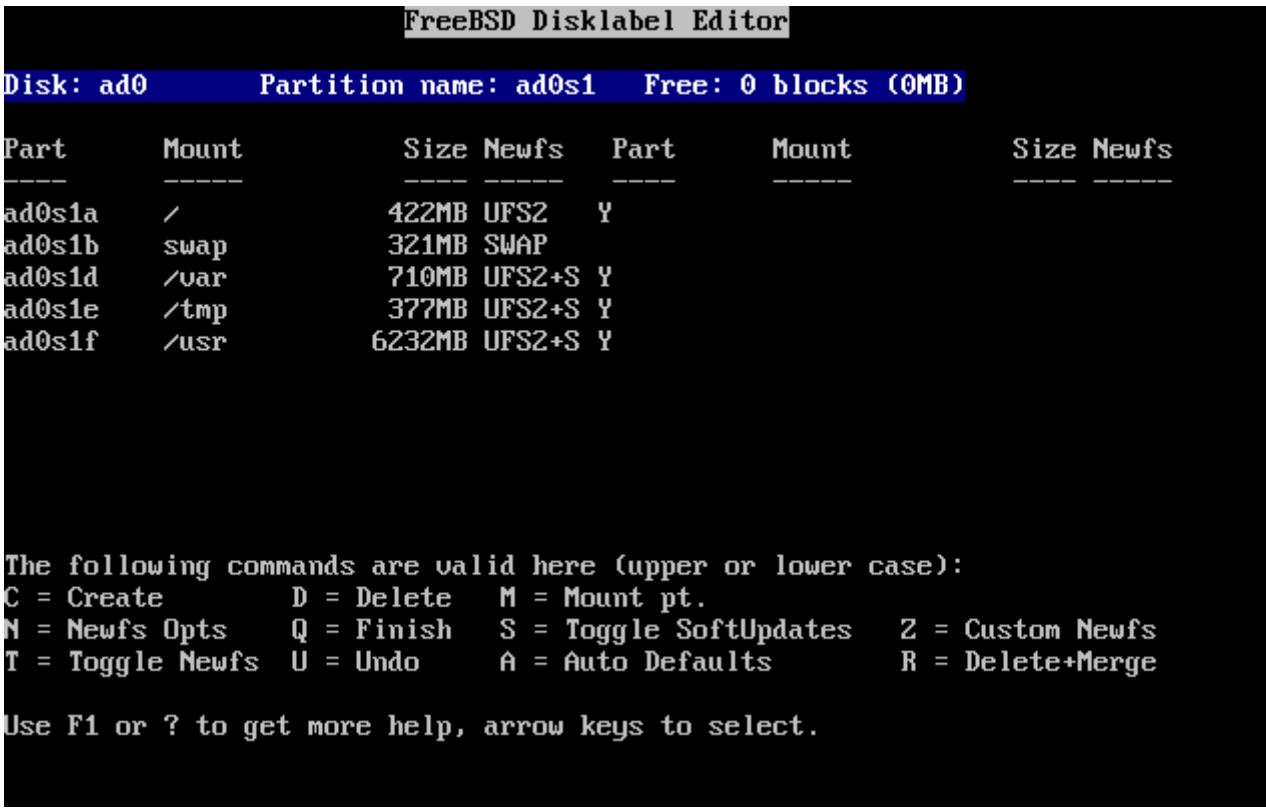
Нижняя треть экрана показывает управляющие клавиши, работающие в **Disklabel.**

Рисунок 2-18. Редактор Sysinstall Disklabel

Disklabel может автоматически создать разделы и присвоить им размеры по умолчанию. Значения размеров по умолчанию вычисляются с помощью внутреннего алгоритма, исходя из емкости диска. Попробуйте это, нажав **A**. Вы увидите экран как на Рис. 2-19. В зависимости от размера диска, значения по умолчанию могут подходить или не подходить вам. Это не имеет значения, если вы не принимаете их.

Замечание: По умолчанию под каталог `/tmp` выделяется собственный раздел вместо использования части раздела `/`. Это помогает избежать заполнения раздела `/` временными файлами.

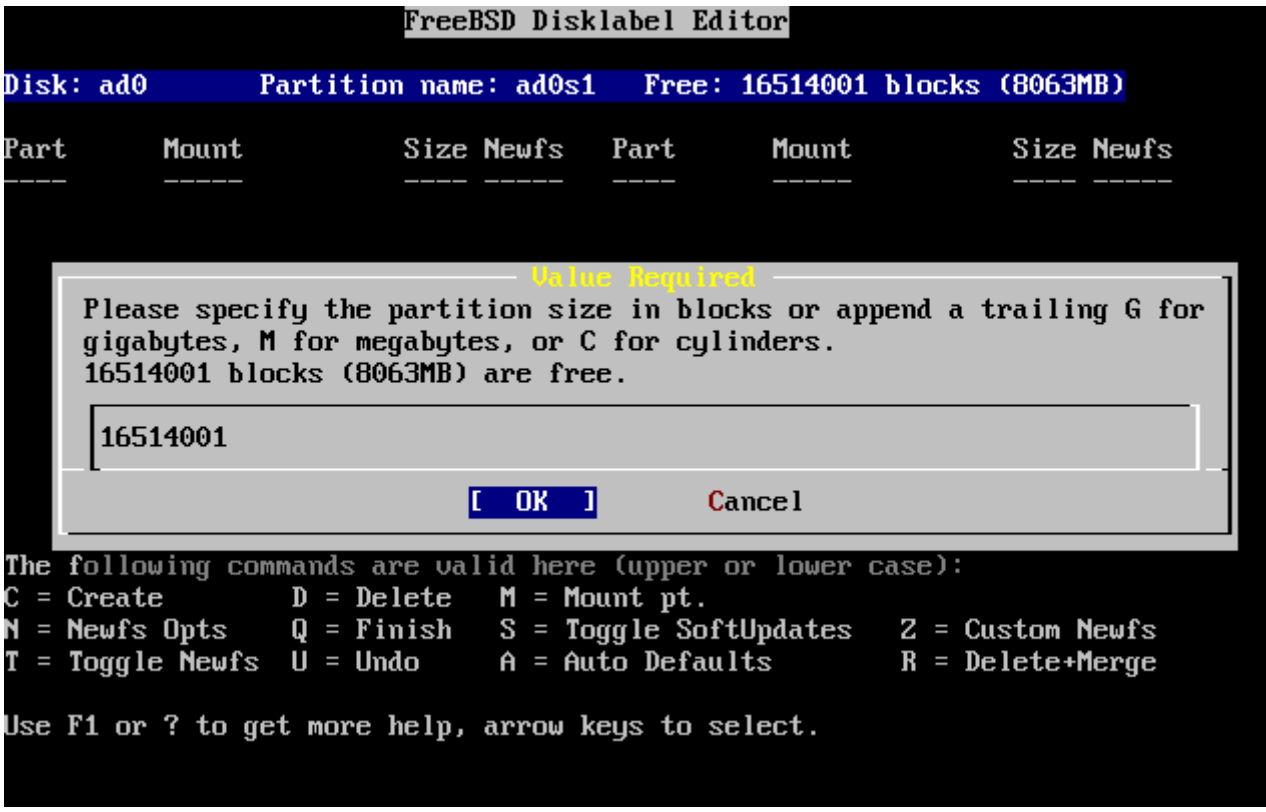
Рисунок 2-19. Редактор Sysinstall Disklabel с установками по умолчанию



Если вы решили не использовать разделы по умолчанию и заменить их на свои, используйте клавиши навигации для выбора первого раздела, затем нажмите **D** для его удаления. Повторите это для удаления всех предложенных разделов.

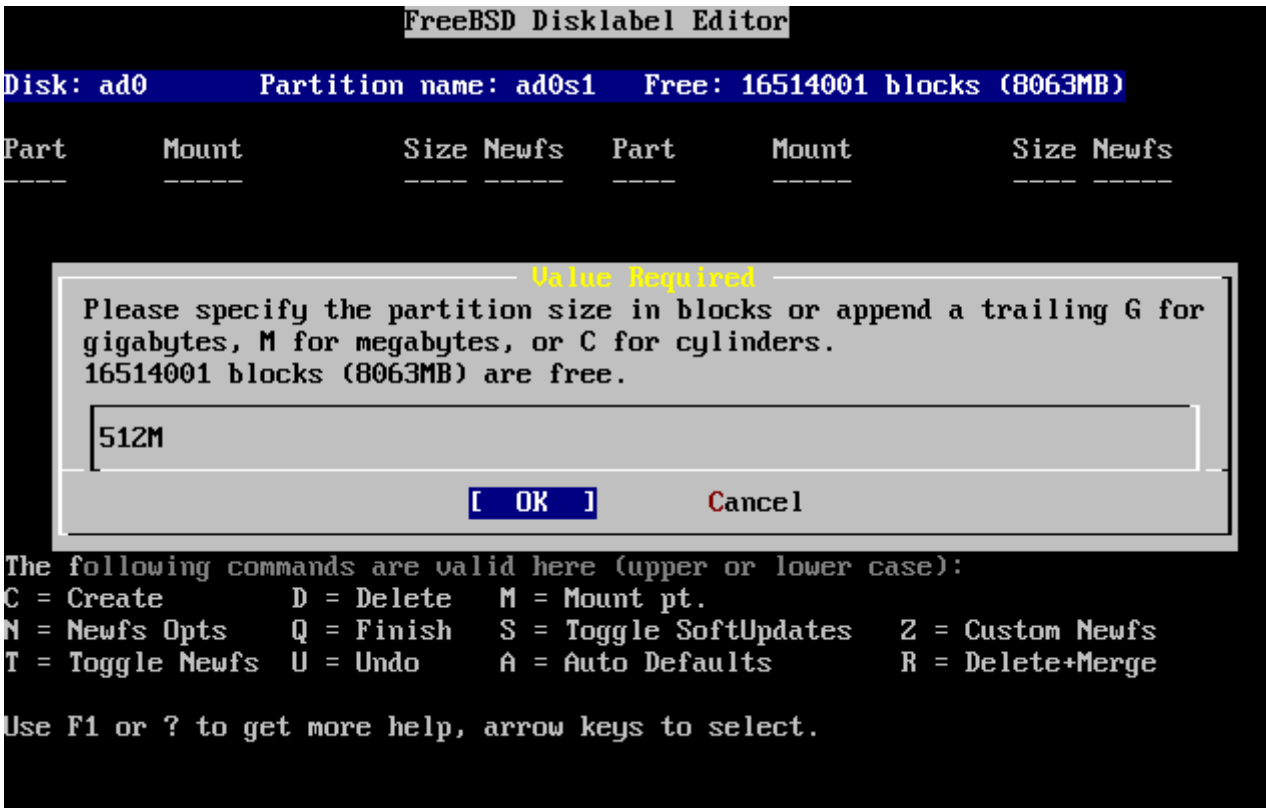
Для создания первого раздела (`a`, монтируемого как `/` — root), убедитесь, что выбран соответствующий слайс вверху экрана и нажмите **C**. Появится диалог, предлагающий выбрать размер нового раздела (как показано на Рис. 2-20). Вы можете ввести количество блоков диска, или количество мегабайт с `M` после номера, или гигабайт с `G`, или цилиндров с `C`.

Рисунок 2-20. Свободное место для корневого раздела



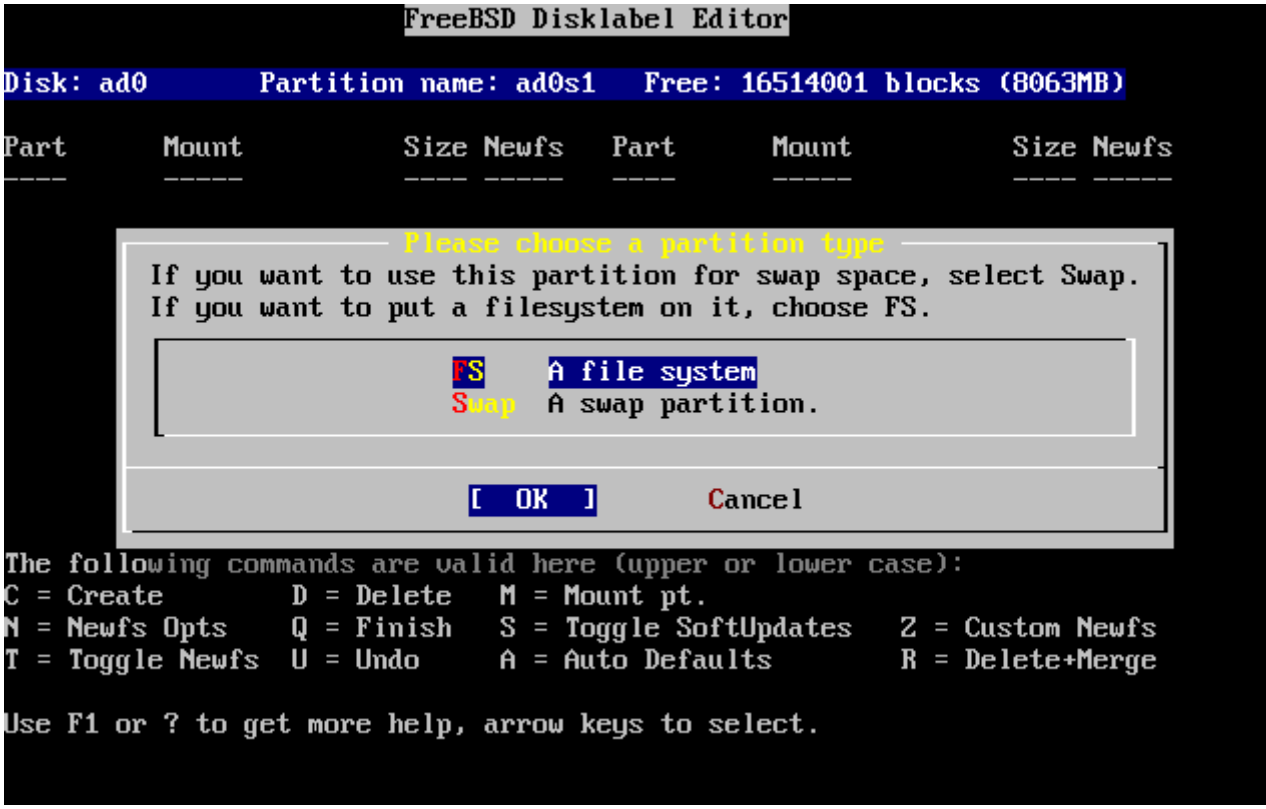
Размер по умолчанию задан для создания корневого раздела на весь слайс. Если вы используете размеры разделов, описанные ранее в примере, удалите это значение используя **Backspace**, а затем введите `512M`, как показано на Рис. 2-21. Затем нажмите [OK].

Рисунок 2-21. Редактирование размера корневого раздела



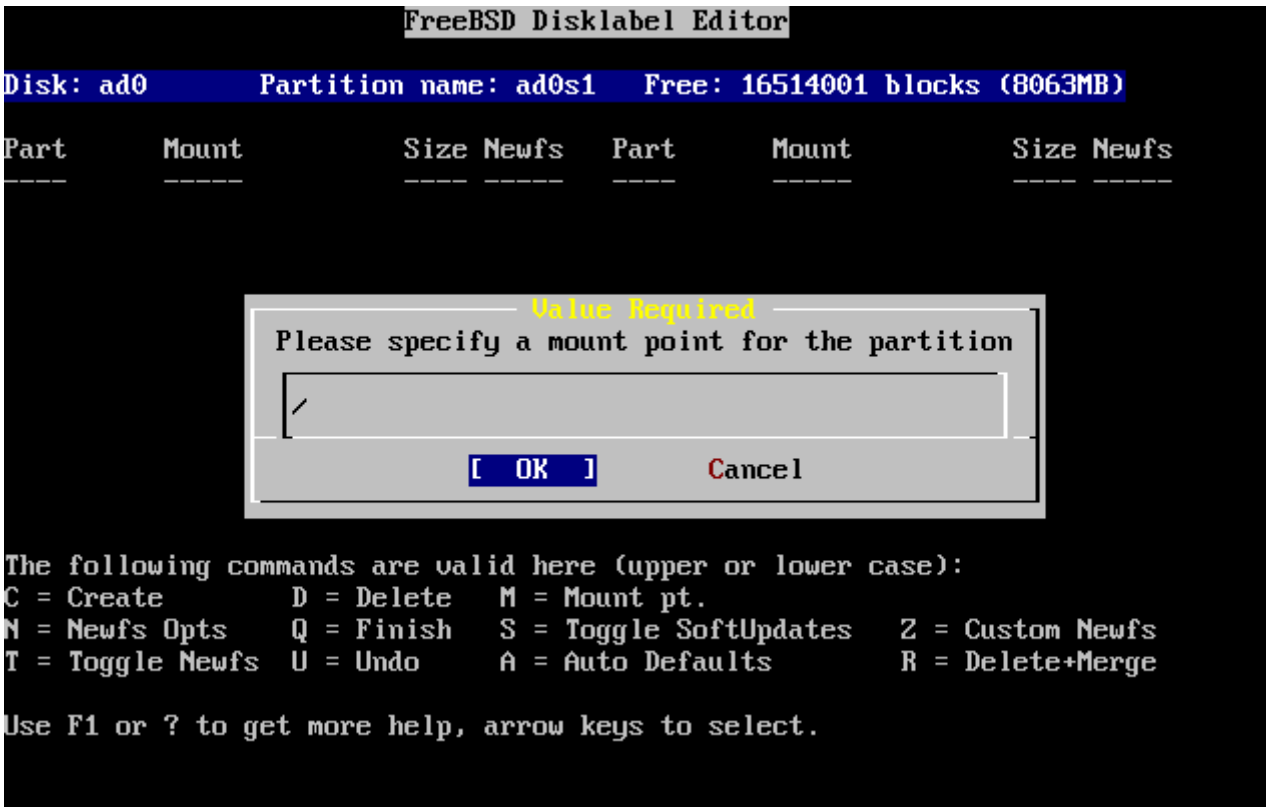
После указания размера раздела вам будет задан вопрос, должен ли этот раздел содержать файловую систему или раздел подкачки. Диалог показан на Рис. 2-22. Первый раздел будет содержать файловую систему, поэтому проверьте, что выбрана **FS** и нажмите **Enter**.

Рисунок 2-22. Выбор типа корневого раздела



Наконец, поскольку вы создаете файловую систему, нужно сказать **Disklabel** где файловая система будет смонтирована. Диалог показан на Рис. 2-23. Точка монтирования корневой файловой системы `/`, поэтому введите `/`, и нажмите **Enter**.

Рисунок 2-23. Выбор точки монтирования корневой файловой системы



На экране будет показан вновь созданный раздел. Вам нужно повторить эту процедуру для других разделов. При создании раздела подкачки вопроса про точку монтирования не будет, поскольку раздел подкачки никогда не монтируется. Когда будете создавать последний раздел, `/usr`, можете оставить предложенный размер как есть, чтобы использовать весь остаток слайса.

Последний экран FreeBSD редактора DiskLabel будет похож на Рис. 2-24, хотя ваш выбор значений может быть другим. Нажмите **Q**, чтобы выйти.

Рисунок 2-24. Редактор Sysinstall Disklabel

2.7. Выбор устанавливаемых компонентов

2.7.1. Выбор дистрибутивного набора (Distribution Set)

Выбор дистрибутивного набора зависит в основном от направления будущего использования системы и от доступного дискового пространства. Предустановленные опции варьируются от наименьшей возможной конфигурации до полной установки. Для новичков в UNIX® и/или FreeBSD лучшим выбором будет одна из этих предустановленных опций. Настройка дистрибутивного набора как правило нужна более опытным пользователям.

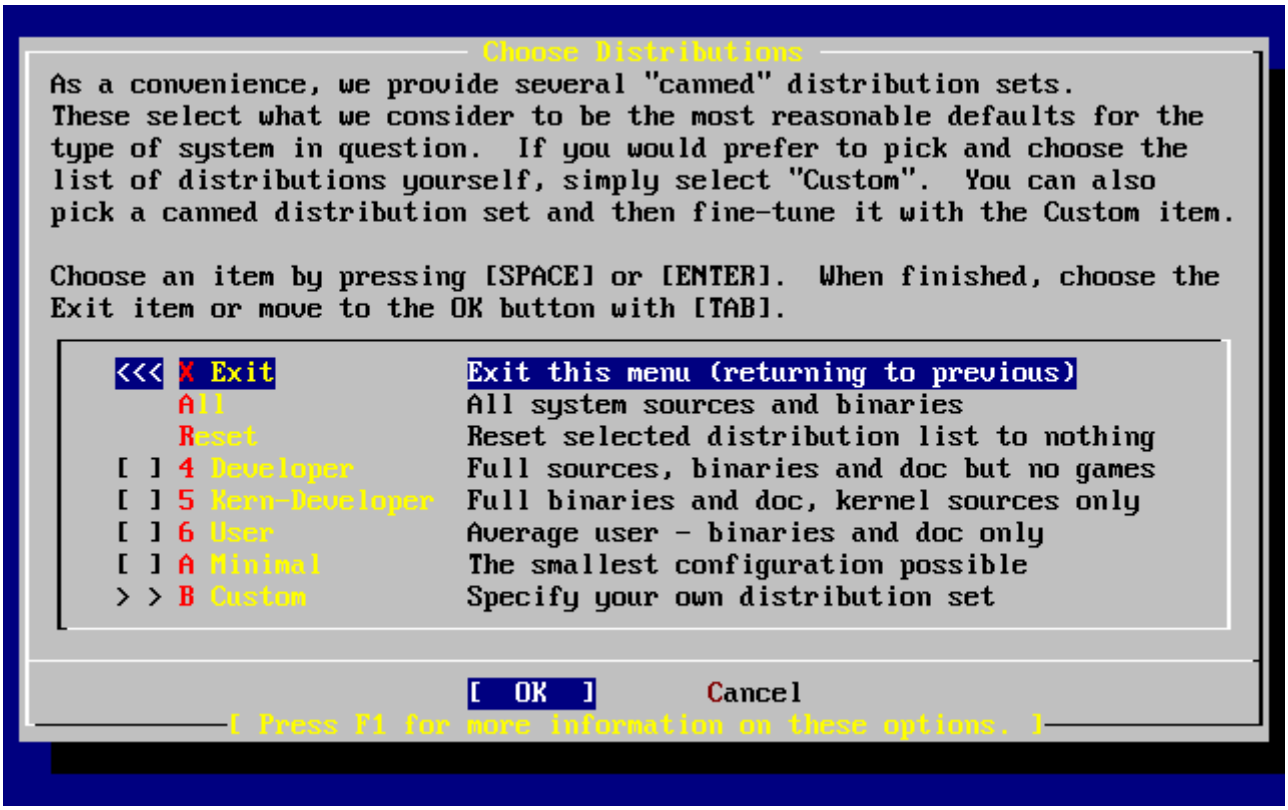
Нажмите **F1** для получения информации о дистрибутивных наборах и их содержимом. После просмотра помощи нажмите **Enter** для возврата к меню выбора дистрибутивного набора.

Если желательно наличие графического интерфейса пользователя, то задачи настройки X сервера и выбора десктопа по умолчанию должны быть выполнены после установки FreeBSD. Более подробная информация по установке и настройке X сервера находится в Гл. 6.

Если планируется пересборка ядра, выберите опцию, включающую исходные тексты. Информация о том, зачем пересобирать ядро и как это сделать, находится на Гл. 9.

Ясно, что наиболее универсальная система включает все. Если места на диске достаточно, выберите **All**, как показано на Рис. 2-25 и нажмите **Enter**. Если есть сомнения относительно того, хватит ли диска, используйте наиболее подходящую опцию. Не беспокойтесь о том, какой выбор будет наилучшим, другие части дистрибутива могут быть добавлены после установки.

Рисунок 2-25. Выбор дистрибутивных наборов



2.7.2. Установка Коллекции Портов

После выбора подходящего дистрибутива можно будет выбрать установку Коллекции Портов FreeBSD. Коллекция Портов — лёгкий и удобный путь установки программ. Коллекция Портов не содержит исходных кодов программ. Это набор файлов, который автоматизирует загрузку, компилирование и установку пакетов программного обеспечения сторонних разработчиков. Гл. 5 показывает, как использовать Коллекцию Портов.

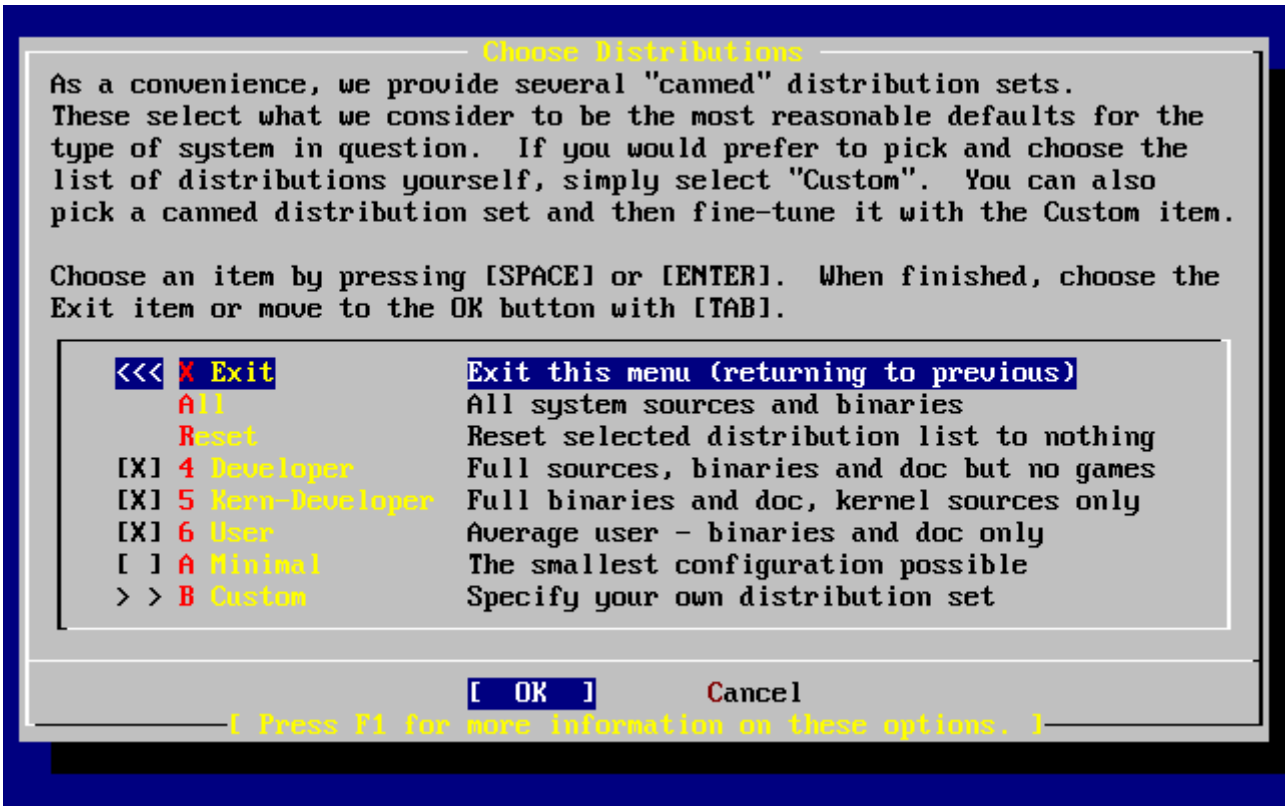
Программа установки не проверяет, есть ли достаточно места. Выберите эту опцию только в том случае, если его достаточно. В FreeBSD 9.0, Коллекция Портов занимает около 500 MB. В более современных релизах это значение всегда больше.

```
                    User Confirmation Requested

Would you like to install the FreeBSD ports collection?


This will give you ready access to over 23,000 ported software packages,
at a cost of around 500 MB of disk space when "clean" and possibly much
more than that if a lot of the distribution tarballs are loaded
(unless you have the extra CDs from a FreeBSD CD/DVD distribution
available and can mount it on /cdrom, in which case this is far less
of a problem).
```

```
The ports collection is a very valuable resource and well worth having
on your /usr partition, so it is advisable to say Yes to this option.

For more information on the ports collection & the latest ports,
visit:
    http://www.FreeBSD.org/ports

                                [ Yes ]     No
```

Выберите [Yes] для установки Коллекции Портов, или [No], чтобы пропустить установку. Нажмите **Enter**, чтобы продолжить. Меню выбора дистрибутивных наборов появится опять.

Рисунок 2-26. Подтверждение выбора дистрибутивного набора



Если вы согласны с выбранными опциями, переместитесь на **Exit**, убедитесь, что выбран [OK] и нажмите **Enter**, чтобы продолжить.

2.8. Выбор источника для установки

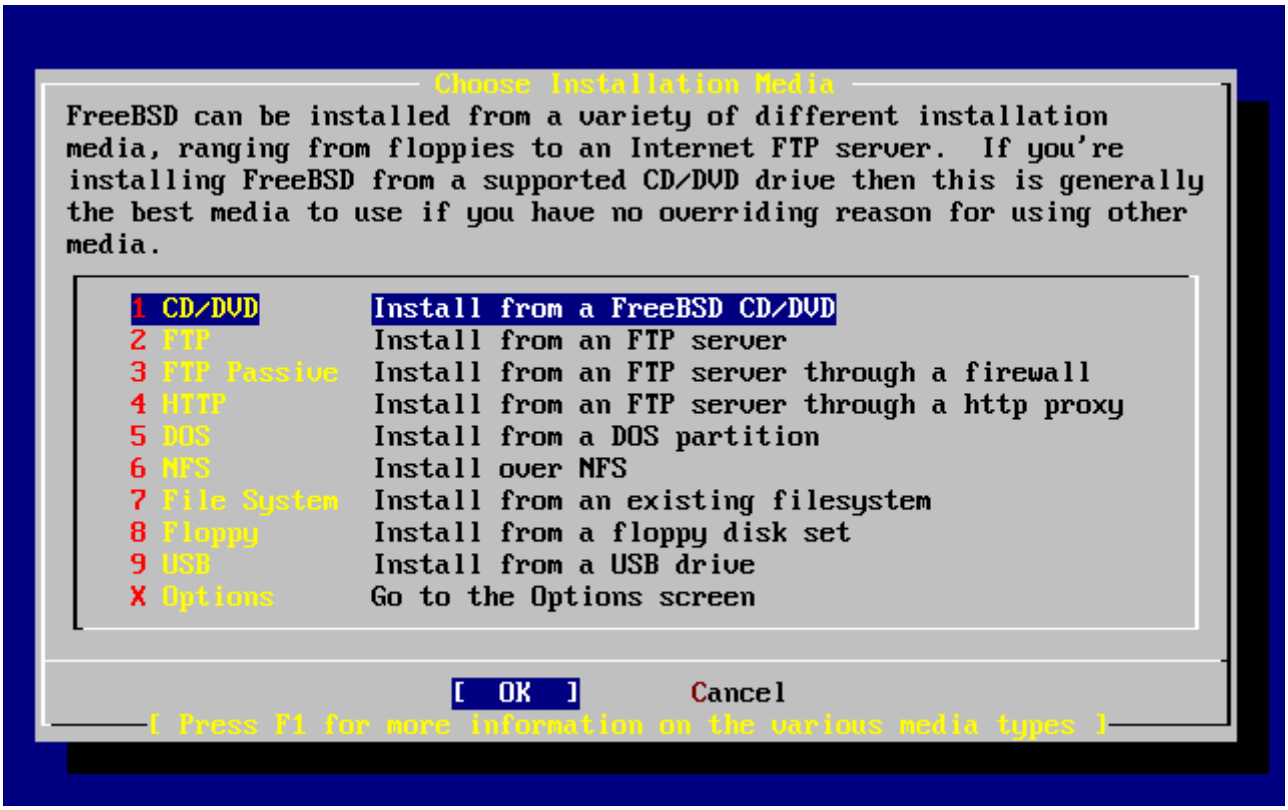
При установке с CDROM или DVD используйте клавиши навигации, для перехода к пункту **Install from a FreeBSD CD/DVD**. Убедитесь, что выбран [OK], и нажмите **Enter** для запуска установки.

При других методах установки выберите соответствующую опцию и следуйте инструкциям.

Нажмите **F1** для просмотра справки по источникам установки. Нажмите **Enter** для возврата к меню выбора

источника установки.

Рисунок 2-27. Выбор источника установки



Режимы установки с FTP: Есть три режима установки через FTP, которые вы можете выбрать: активный FTP, пассивный FTP, или через HTTP прокси.

Активный FTP: **Install from an FTP server**

С этой опцией все закачки по FTP будут выполнены в "активном" режиме. Этот режим не позволяет работать через файрволл, но зачастую позволяет работать со старыми серверами FTP, не поддерживающими пассивный режим. Если соединение прерывается в пассивном режиме (по умолчанию), попробуйте активный!

Пассивный FTP: **Install from an FTP server through a firewall**

Эта опция говорит **sysinstall** использовать "пассивный" режим для работы с FTP. Он позволяет работать через файрволл, не разрешающий входящие соединения на случайных TCP портах.

FTP через HTTP прокси: **Install from an FTP server through a http proxy**

Эта опция говорит **sysinstall** использовать HTTP протокол (как Web-браузер) для работы с FTP через прокси. Прокси будет транслировать все запросы и посылать их на FTP сервер. Это позволяет проходить через файрволл, на котором FTP запрещен, но есть HTTP прокси. В этом случае потребуется указать прокси и FTP сервер.

Для работы с FTP через прокси, необходимо поместить имя сервера как часть имени пользователя после знака "@". Прокси сервер "обманет" настоящий сервер. Например, предположим что вы хотите провести установку с `ftp.FreeBSD.org`, используя FTP через прокси `foo.example.com`, прослушивающем порт 1234.

В этом случае, войдите в меню параметров, установите имя пользователя FTP `ftp@ftp.FreeBSD.org`, а

вместо пароля введите свой адрес email. В качестве источника установки выберите FTP (или пассивный FTP, если прокси его поддерживает), и URL `ftp://foo.example.com:1234/pub/FreeBSD`.

Так как `/pub/FreeBSD` с сервера `ftp.FreeBSD.org` идет через прокси `foo.example.com`, вы сможете провести установку с *этого* компьютера (файлы будут загружены с `ftp.FreeBSD.org` как требуется для установки).

2.9. Подтверждение установки

Теперь можно начинать установку. Это последний шанс отменить установку, и таким образом избежать изменений на жестком диске.

```
                        User Confirmation Requested

 Last Chance! Are you SURE you want to continue the installation?


 If you're running this on a disk with data you wish to save then WE

 STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!


 We can take no responsibility for lost disk contents!


                                  [ Yes ]    No
```

Выберите [Yes] и нажмите **Enter**, чтобы начать.

Время установки сильно зависит от выбранного дистрибутивного набора, источника установки и скорости компьютера. Появится несколько сообщений о статусе процесса установки.

Установка будет завершена, когда отобразится следующее сообщение:

```
                                  Message


Congratulations! You now have FreeBSD installed on your system.


We will now move on to the final configuration questions.

For any option you do not wish to configure, simply select No.


If you wish to re-enter this utility after the system is up, you may

do so by typing: /usr/sbin/sysinstall.


                                  [ OK ]


                          [  Press enter or space  ]
```

Нажмите **Enter** для начала послеустановочной настройки.

Выбор [No] и нажатие **Enter** прервет процесс установки, изменения в систему внесены не будут. Появится следующее сообщение:

```
                                Message

Installation complete with some errors.  You may wish to scroll

through the debugging messages on VTY1 with the scroll-lock feature.

You can also choose "No" at the next prompt and go back into the

installation menus to retry whichever operations have failed.


                                [ OK ]
```

Это сообщение появилось, поскольку ничего не было установлено. Нажатие **Enter** вернет вас в главное меню установки, чтобы выйти из нее.

Лабораторная работа №5. Основы администрирования Unix

2.10. После установки

После успешной установки необходимо настроить множество параметров. Некоторые параметры могут быть заданы из меню параметров после установки, перед загрузкой установленной FreeBSD, или после нее с использованием `sysinstall`, где надо выбрать пункт **Configure**.

2.10.1. Настройка сетевых устройств (Network Device Configuration)

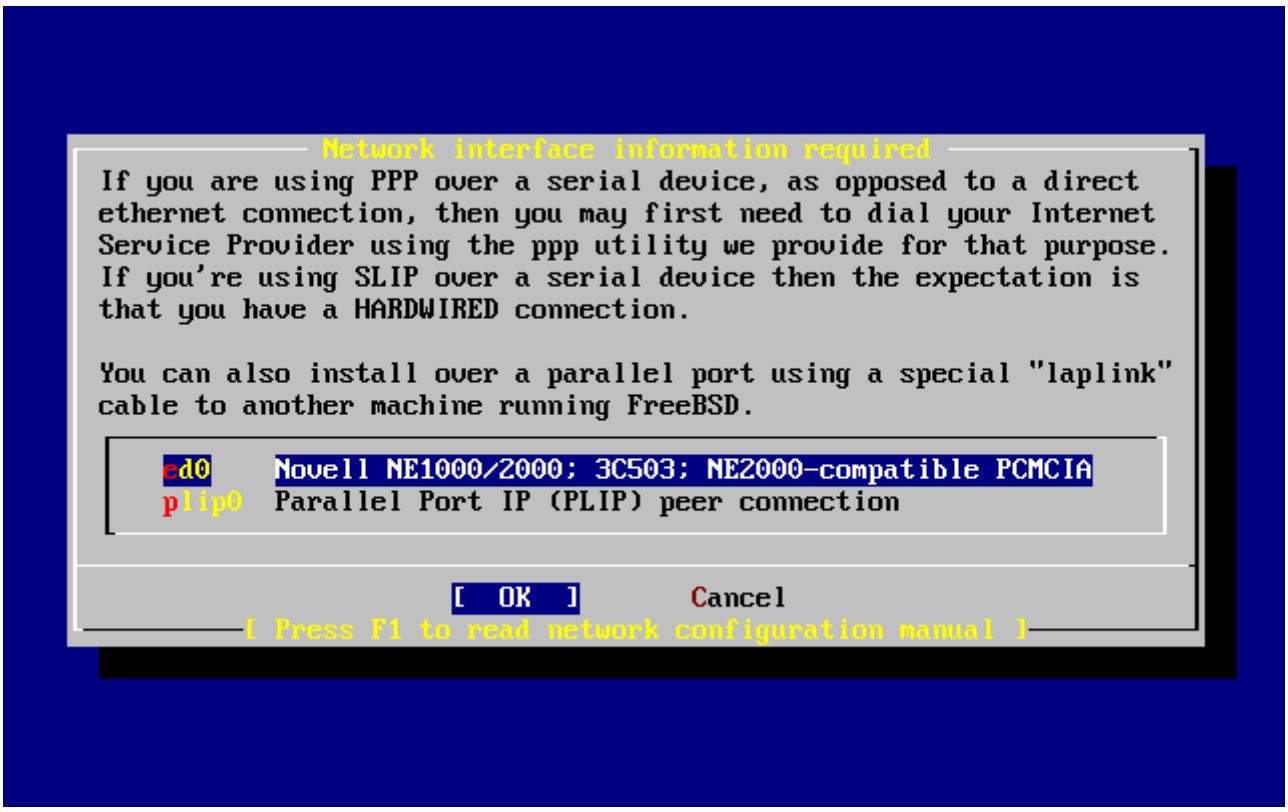
Если вы настраивали PPP для установки через FTP, этот экран не появится, и настройку можно будет произвести позже как описано выше.

Чтобы лучше узнать о локальных сетях и настройке FreeBSD в качестве шлюза/маршрутизатора, обратитесь к главе Сложные вопросы работы в сети.

```
                        User Confirmation Requested

   Would you like to configure any Ethernet or PPP network devices?


                                [ Yes ]    No
```

Для настройки сетевого устройства выберите [Yes] и нажмите **Enter**. Или нажмите [No], чтобы продолжить.

Рисунок 2-28. Выбор Ethernet устройства



Выберите интерфейс для настройки с помощью клавиш навигации и нажмите **Enter**.

```
                    User Confirmation Requested

      Do you want to try IPv6 configuration of the interface?


                              Yes    [ No ]
```

Для частной локальной сети обычный протокол интернет (IPv4) вполне достаточен, поэтому выбрана кнопка [No] и нажат **Enter**.

Если вы хотите подсоединиться к существующей сети IPv6 через сервер RA, выберите [Yes] и нажмите **Enter**. Поиск RA серверов займет несколько секунд.

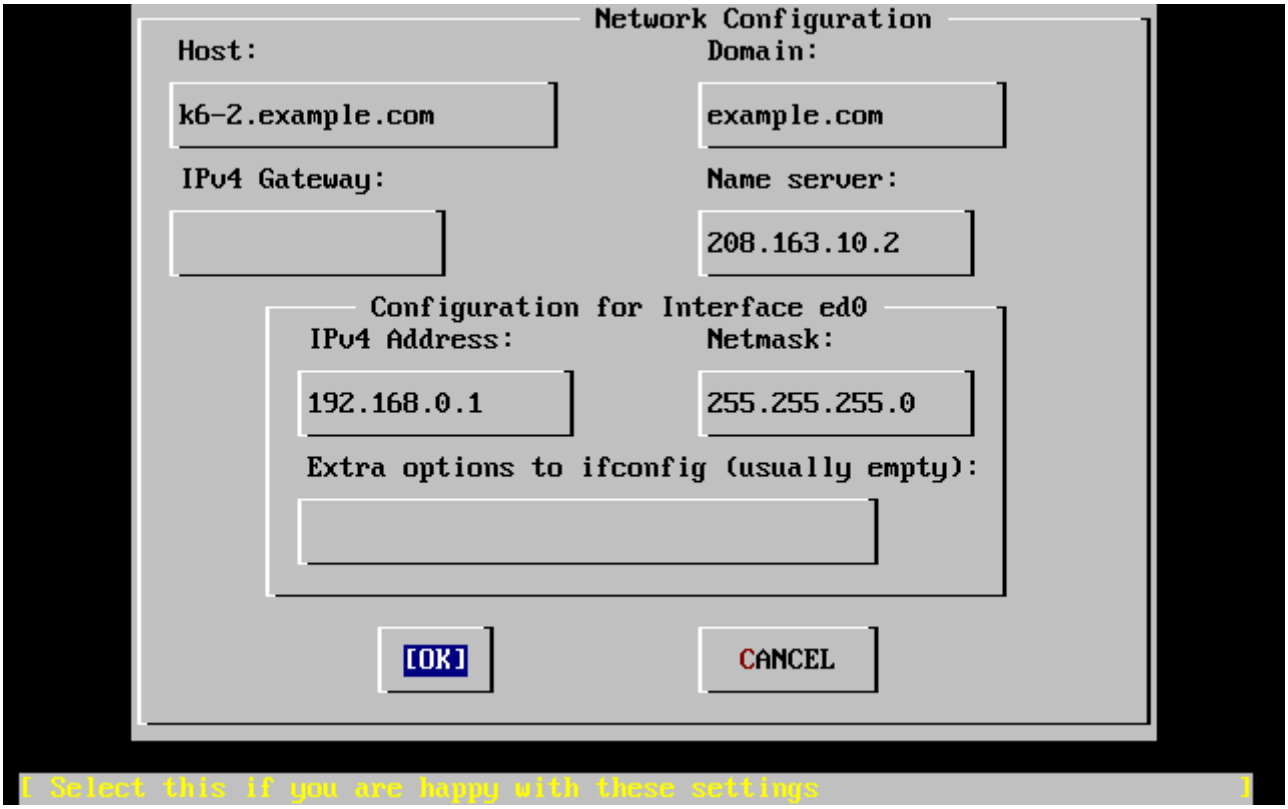
```
                    User Confirmation Requested

      Do you want to try DHCP configuration of the interface?


                              Yes    [ No ]
```

Если DHCP (Dynamic Host Configuration Protocol) не нужен, выберите [No] с помощью клавиш навигации и нажмите **Enter**.

Выбор [Yes] запустит **dhclient**, и, если все пройдет нормально, заполнит информацию о конфигурации сети автоматически. Обратитесь к Разд. 27.5 за более подробными сведениями.

Следующий экран конфигурации сети показывает настройку устройства Ethernet системы, которая будет работать шлюзом для локальной сети.

Рисунок 2-29. Настройка сети для *ed0*



Используйте **Tab** для выбора полей и заполнения их соответствующими данными:

Host

Полное имя хоста, в этом примере `k6-2.example.com`.

Domain

Имя домена, в котором находится ваш компьютер, в этом примере `example.com`.

IPv4 Gateway

IP хоста, пересылающего пакеты наружу локальной сети. Вам потребуется заполнить его, если это компьютер, подключенный к сети. *Оставьте это поле пустым*, если компьютер является шлюзом в интернет для сети. Шлюз IPv4 известен также как шлюз по умолчанию или маршрут по умолчанию.

Name server

IP адрес местного сервера DNS. В этой локальной сети нет DNS сервера, поэтому использован IP адрес DNS сервера провайдера (`208.163.10.2`).

IPv4 address

IP адрес, использованный для этого интерфейса, `192.168.0.1`

Netmask

Адрес блока, использованного для этой локальной сети, это `192.168.0.0` - `192.168.0.255`. с маской сети `255.255.255.0`.

Дополнительные параметры для ifconfig

Любые специфичные для интерфейса опции к `ifconfig`, которые вы хотите добавить. В данном случае ничего.

Используйте **Tab** для выбора [OK] после окончания настройки и нажмите **Enter**.

```
User Confirmation Requested

Would you like to bring the ed0 interface up right now?


[ Yes ]   No
```

Выбор [Yes] и нажатие **Enter** введет компьютер в сеть. Тем не менее, компьютеру все еще требуется перезагрузка.

2.10.2. Настройка шлюза (Configure Gateway)

```
User Confirmation Requested

Do you want this machine to function as a network gateway?


[ Yes ]    No
```

Если компьютер будет шлюзом для локальной сети, пересылая пакеты между другими компьютерами, выберите [Yes] и нажмите **Enter**. Если это обычный компьютер, выберите [No] и нажмите **Enter** для продолжения.

2.10.3. Настройка сервисов интернет (Configure Internet Services)

```
User Confirmation Requested

Do you want to configure inetd and the network services that it provides?


Yes   [ No ]
```

Если выбрана [No], различные сервисы, такие как **telnetd** не будут запущены. Это означает, что удаленные пользователи не смогут зайти по **telnet** на этот компьютер. Локальные пользователи все же смогут заходит на удаленные компьютеры по **telnet.**

Эти сервисы могут быть включены после установки путем редактирования `/etc/inetd.conf` с помощью вашего любимого текстового редактора. Обращайтесь к Разд. 27.2.1 за более подробной информацией.

Выберите [Yes] если хотите настроить эти сервисы во время установки. Появится дополнительный запрос подтверждения:

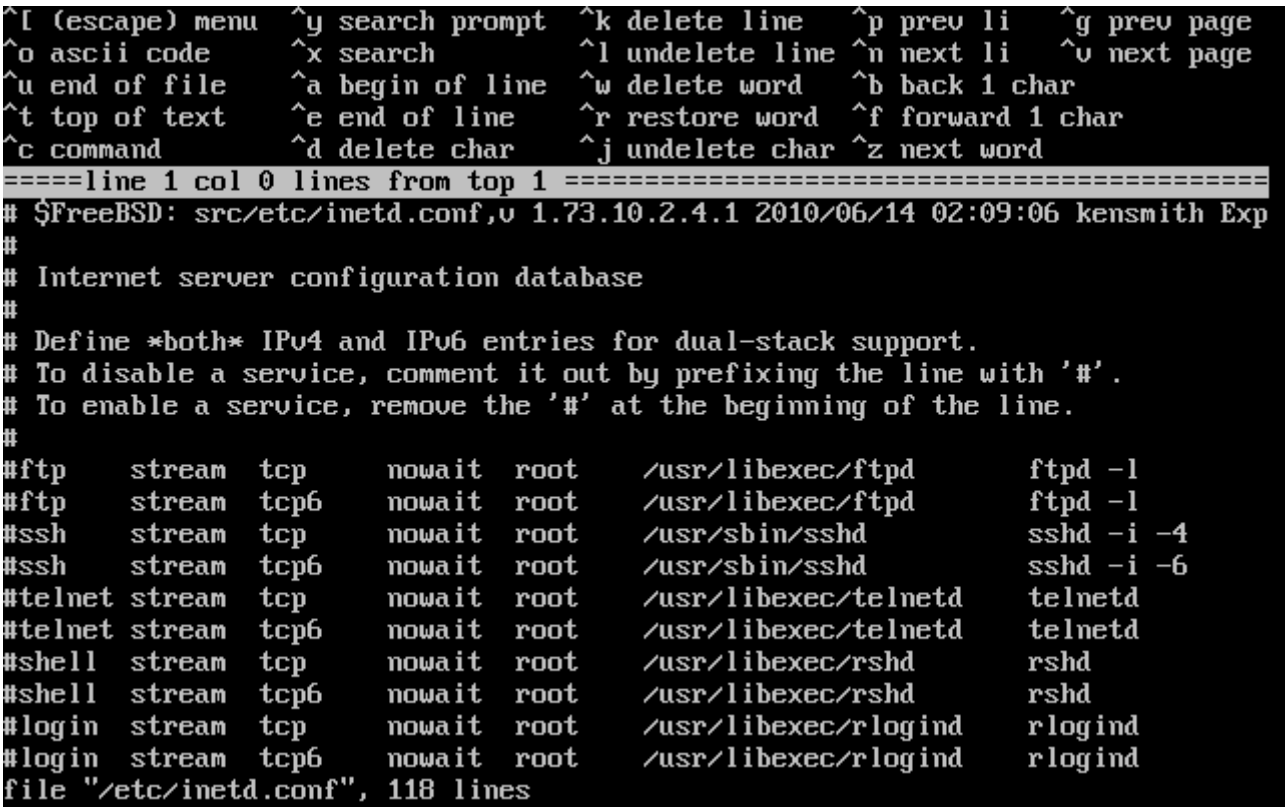
```
                    User Confirmation Requested
The Internet Super Server (inetd) allows a number of simple Internet
services to be enabled, including finger, ftp and telnetd.  Enabling
these services may increase risk of security problems by increasing
the exposure of your system.

With this in mind, do you wish to enable inetd?

                              [ Yes ]   No
```

Нажмите [Yes], чтобы продолжить.

```
                    User Confirmation Requested
inetd(8) relies on its configuration file, /etc/inetd.conf, to determine
which of its Internet services will be available.  The default FreeBSD
inetd.conf(5) leaves all services disabled by default, so they must be
specifically enabled in the configuration file before they will
function, even once inetd(8) is enabled.  Note that services for
IPv6 must be separately enabled from IPv4 services.

Select [Yes] now to invoke an editor on /etc/inetd.conf, or [No] to
use the current settings.

                              [ Yes ]   No
```

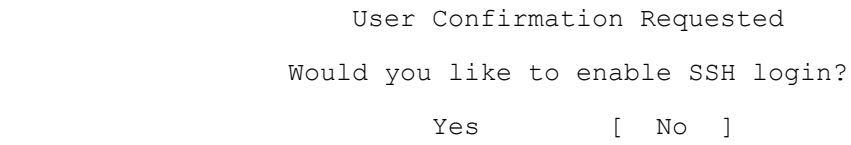
Выбор [Yes] позволит добавить сервисы путем удаления # перед началом строки.

Рисунок 2-30. Редактирование `inetd.conf`



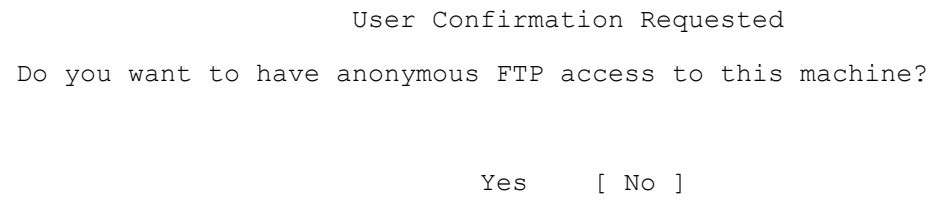
После добавления нужных сервисов нажатие **Esc** отобразит меню, позволяющее выйти с сохранением изменений.

2.10.4. Настройка входа по SSH



Выбор [Yes] активирует запуск sshd(8) — демона для приложения **OpenSSH**, что в свою очередь позволит создавать безопасные удалённые соединения с вашей машиной. За более детальной информацией о **OpenSSH** обратитесь к Разд. 15.11.

2.10.5. Анонимный (Anonymous) FTP



2.10.5.1. Запрещение анонимного FTP

Выбор кнопки по умолчанию [No] и нажатие **Enter** все же позволит пользователям, имеющим учетные записи с паролями, использовать FTP для доступа к компьютеру.

2.10.5.2. Разрешение анонимного FTP

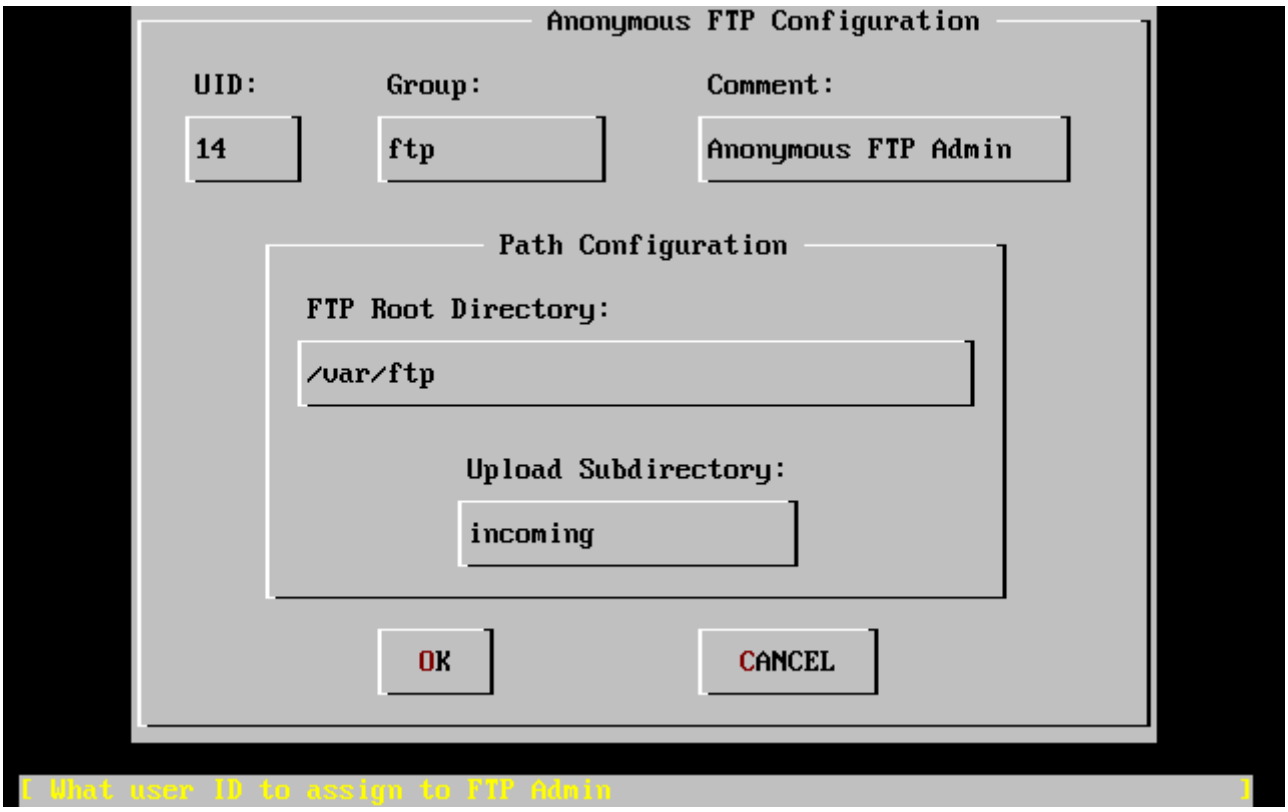
Кто угодно сможет получить доступ к компьютеру, если вы разрешите анонимные соединения FTP. Предварительно должны быть рассмотрены возможные проблемы с безопасностью. Более подробная информация о безопасности находится в Гл. 15.

Чтобы разрешить анонимный FTP, выберите [Yes], используя клавиши навигации, и нажмите **Enter**. От вас потребуется дополнительное подтверждение:

```
                    User Confirmation Requested

Anonymous FTP permits un-authenticated users to connect to the system
FTP server, if FTP service is enabled.  Anonymous users are
restricted to a specific subset of the file system, and the default
configuration provides a drop-box incoming directory to which uploads
are permitted.  You must separately enable both inetd(8), and enable
ftpd(8) in inetd.conf(5) for FTP services to be available.  If you
did not do so earlier, you will have the opportunity to enable inetd(8)
again later.

If you want the server to be read-only you should leave the upload
directory option empty and add the -r command-line option to ftpd(8)
in inetd.conf(5)

Do you wish to continue configuring anonymous FTP?

                        [ Yes ]         No
```

В этом сообщении сказано: если вы хотите разрешить анонимный FTP доступ, то вам также придется активировать (см. Разд. 2.10.3) сам сервис FTP в `/etc/inetd.conf`. Выберите [Yes], нажмите **Enter**, далее отобразится следующий экран:

Рисунок 2-31. Настройка по анонимного FTP по умолчанию



Используя **Tab** для выбора полей ввода, заполните соответствующую информацию:

UID

Идентификатор, который вы намереваетесь присвоить анонимному пользователю FTP. Файлам, загруженным на FTP, будет присвоен этот идентификатор.

Group

В эту группу будет входить анонимный пользователь FTP.

Comment

Строка с описанием анонимного пользователя; она будет внесена в `/etc/passwd`.

FTP Root Directory

Часть иерархии файловой системы, в которой будут храниться файлы, доступные анонимному пользователю FTP.

Upload Subdirectory

Подкаталог, доступный на запись анонимному пользователю FTP.

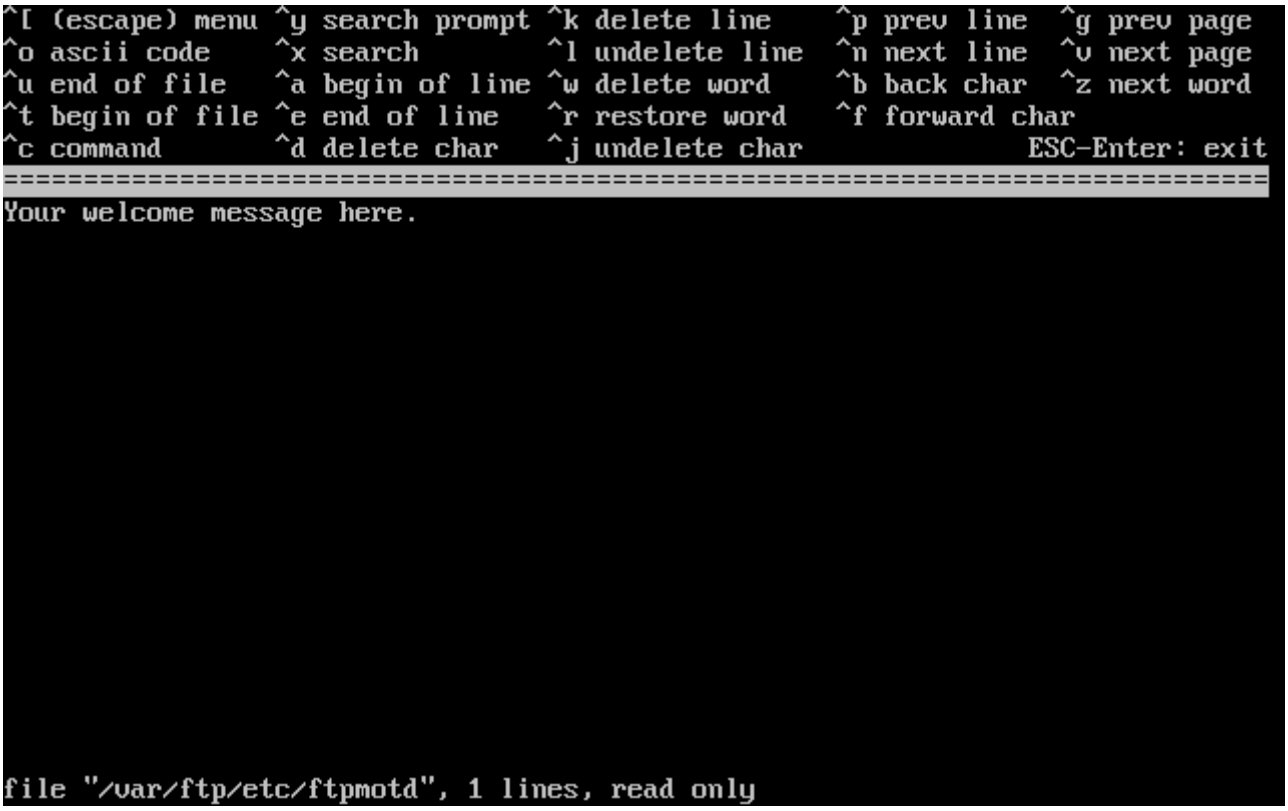
Корневой каталог FTP по умолчанию будет размещен в `/var`. Если в нем предположительно не хватает места для для нужд FTP, можно использовать каталог `/usr`, выбрав корневой каталог FTP (FTP root directory) `/usr/ftp`.

Когда будут выбраны подходящие значения, нажмите **Enter**, чтобы продолжить.

```
                    User Confirmation Requested

Create a welcome message file for anonymous FTP users?


                        [ Yes ]    No
```

Если вы выберете [Yes] и нажмете **Enter**, запустится редактор, позволяющий отредактировать сообщение FTP.

Рисунок 2-32. Редактирование FTP Welcome Message



Этот текстовый редактор называется `ee`. Используйте инструкции, чтобы изменить сообщение, или измените сообщение позже, используя выбранный вами редактор. Обратите внимание, что имя/расположение файла показаны внизу окна редактора.

Нажмите **Esc** и появится меню с пунктом по умолчанию **a) leave editor** (выйти из редактора). Нажмите **Enter**, чтобы выйти и продолжить. Нажмите **Enter** еще раз, чтобы сохранить изменения, если они были сделаны.

2.10.6. Настройка сетевой файловой системы (Configure Network File System)

Сетевая файловая система (Network File System, NFS) позволяет совместно использовать файлы в сети. Компьютер может быть настроен как сервер, клиент, или как то и другое. Обратитесь к Разд. 27.3 за более подробной информацией.

2.10.6.1. Сервер NFS (NFS Server)

```
                    User Confirmation Requested

 Do you want to configure this machine as an NFS server?


                                Yes     [ No ]
```

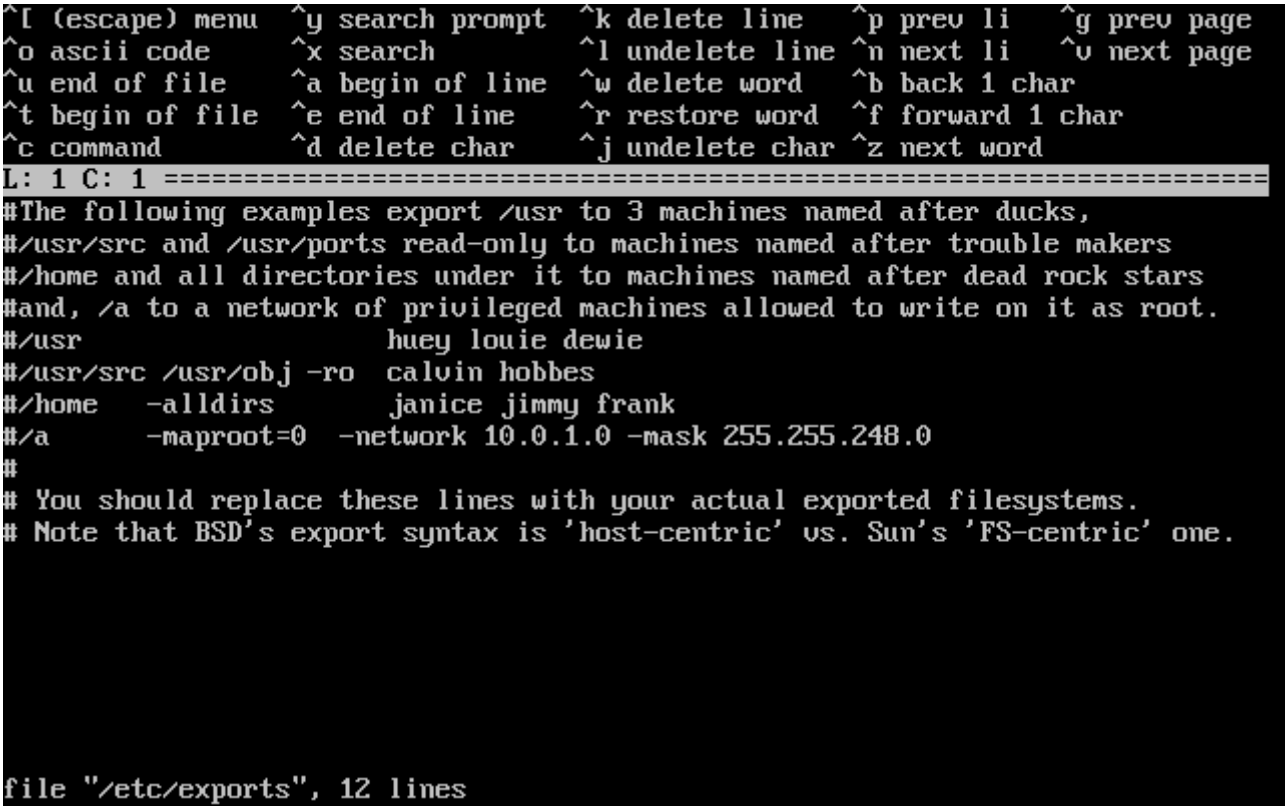
Если вам не нужен NFS сервер, выберите [No] и нажмите **Enter**.

Если выбран пункт [Yes], появится сообщение, говорящее о том, что должен быть создан файл `exports`.

```
                              Message

Operating as an NFS server means that you must first configure an

/etc/exports file to indicate which hosts are allowed certain kinds of

access to your local filesystems.

Press [Enter] now to invoke an editor on /etc/exports

                              [ OK ]
```

Нажмите **Enter**, чтобы продолжить. Запустится текстовый редактор, позволяющий создать и отредактировать файл `exports`.

Рисунок 2-33. Редактирование `exports`

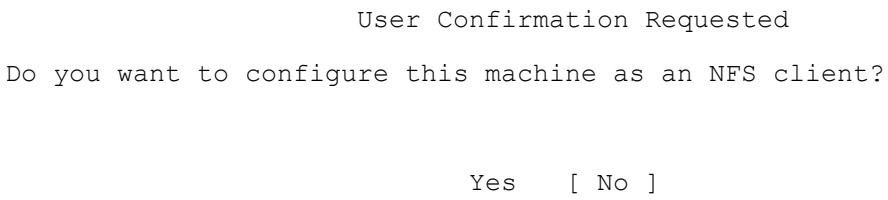


Используйте инструкции для добавления экспортируемых файловых систем сейчас, или позднее с помощью выбранного вами текстового редактора. Обратите внимание, что имя/расположение файла показаны внизу окна редактора.

Нажмите **Esc** и появится меню с пунктом по умолчанию **a) leave editor**. Нажмите **Enter**, чтобы выйти и продолжить.

2.10.6.2. Клиент NFS (NFS Client)

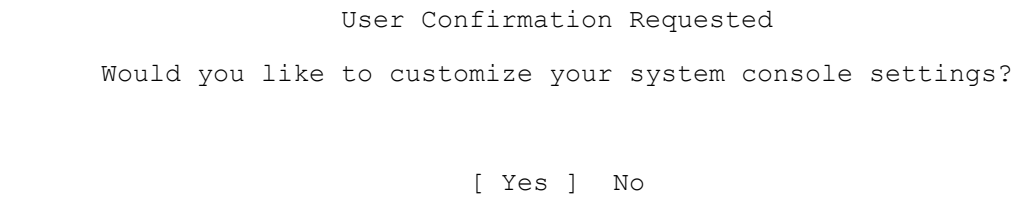
NFS клиент позволяет организовать доступ к серверам NFS.



С помощью клавиш навигации выберите [Yes] или [No], как потребуется, и нажмите **Enter**.

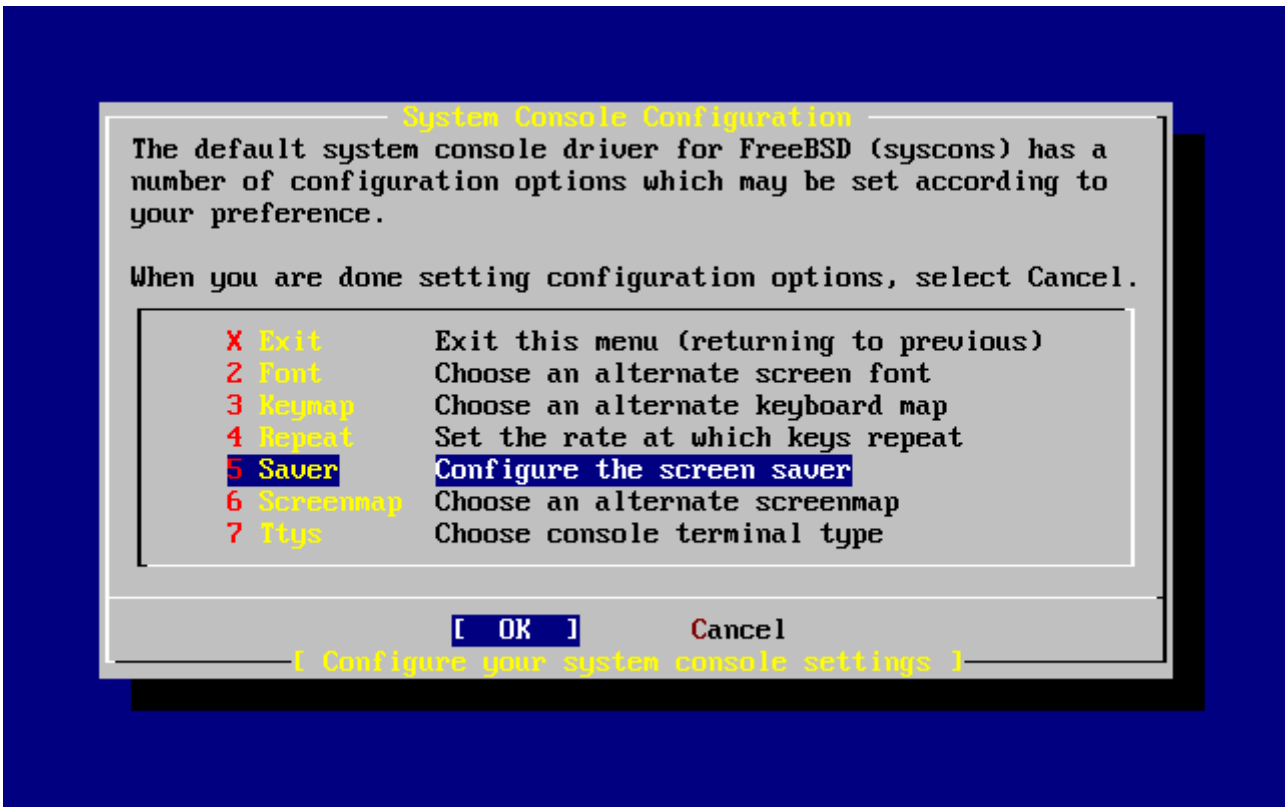
2.10.7. Настройки системной консоли (System Console Settings)

Есть несколько параметров для настройки системной консоли.



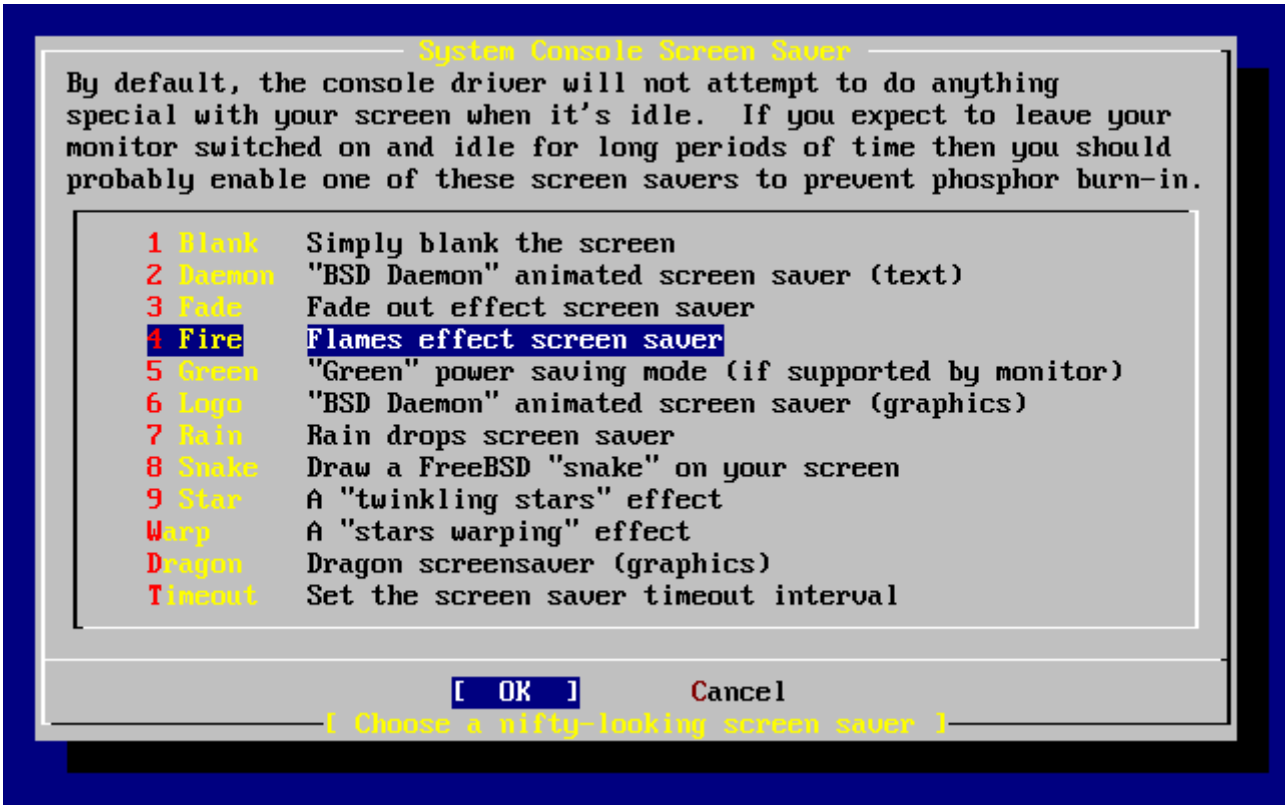
Для просмотра и настройки параметров выберите [Yes] и нажмите **Enter**.

Рисунок 2-34. Параметры настройки системной консоли



Часто используемая опция это хранитель экрана (screen saver). Используйте клавиши навигации для выбора **Saver** и нажмите **Enter**.

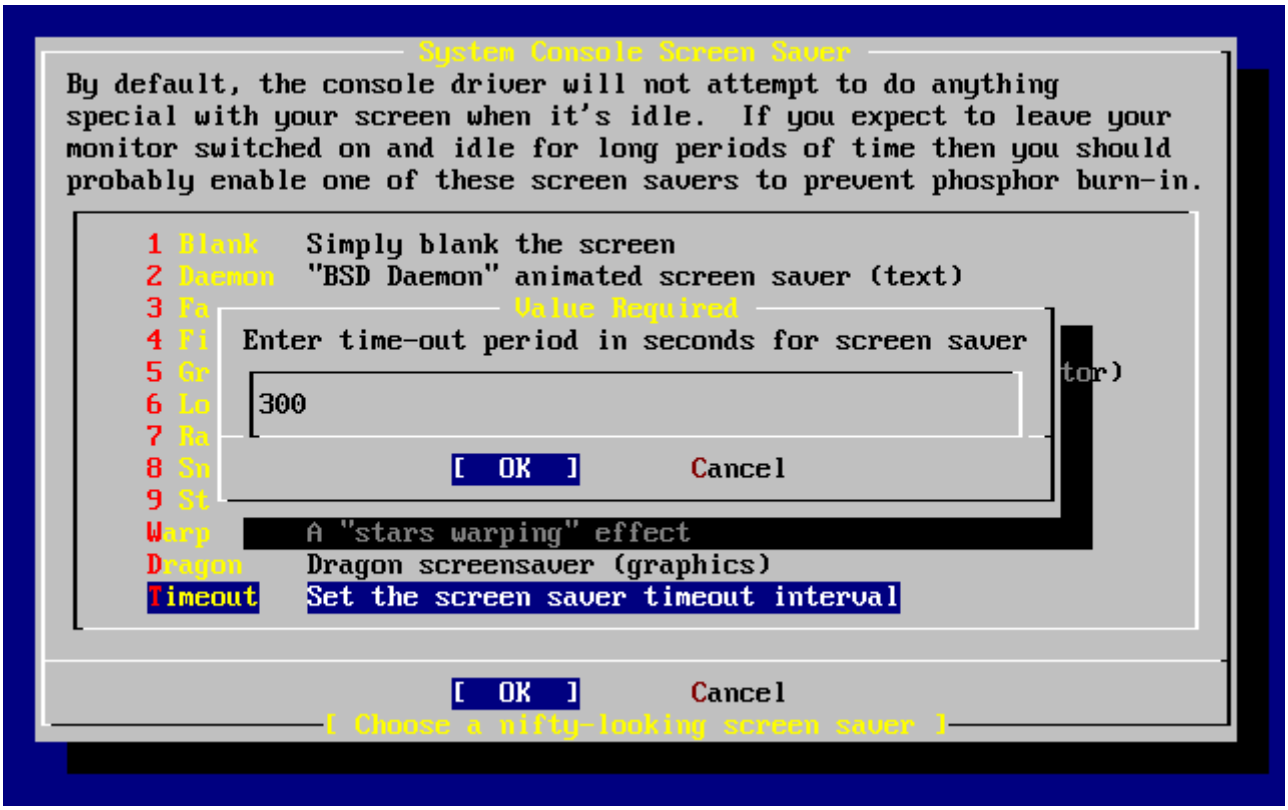
Рисунок 2-35. Параметры хранителя экрана



Выберите подходящий хранитель экрана с помощью клавиш навигации и нажмите **Enter**. Опять появится меню настройки системной консоли.

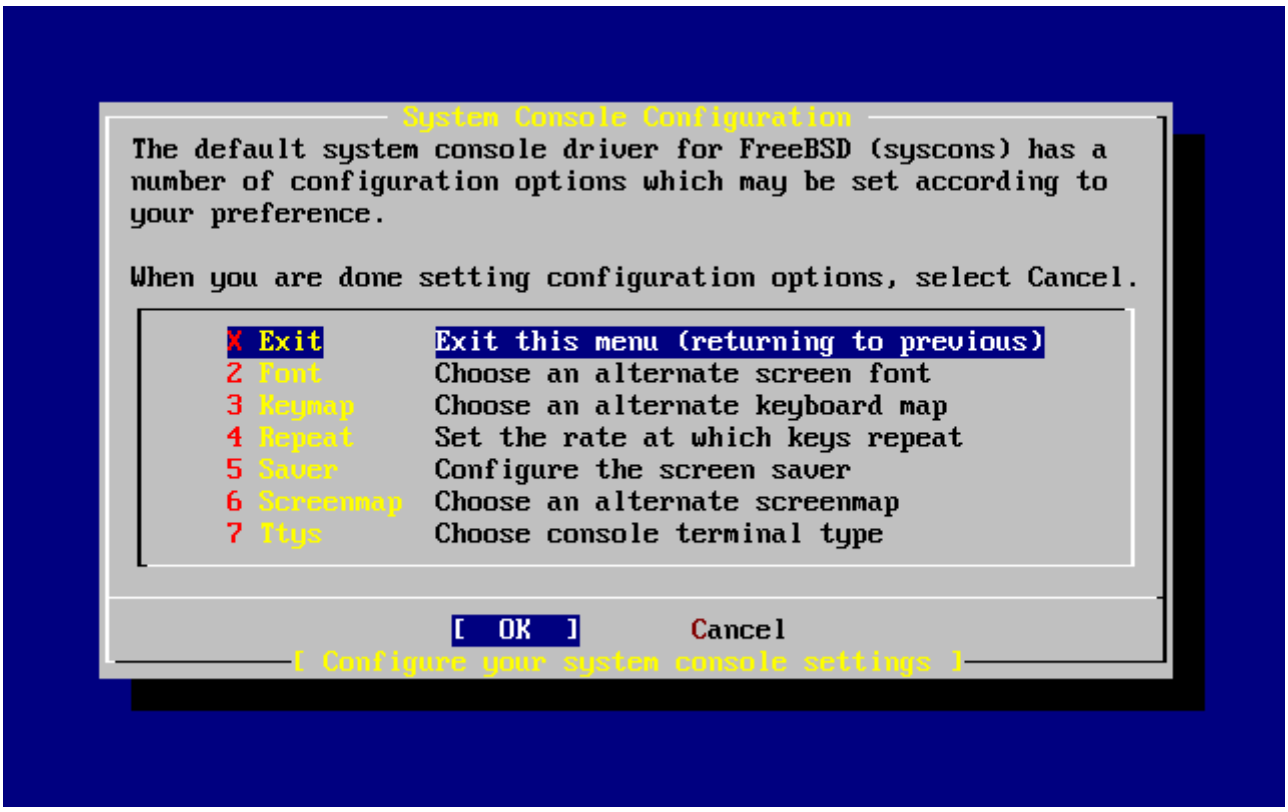
Время по умолчанию 300 секунд. Для изменения временного интервала выберите **Saver** еще раз. В меню настроек хранителя экрана выберите **Timeout** с помощью клавиш навигации и нажмите **Enter**. Появится меню:

Рисунок 2-36. Временной интервал хранителя экрана



Значение может быть изменено, затем выберите [OK] и нажмите **Enter** для возврата в меню настройки системной консоли.

Рисунок 2-37. Выход из меню конфигурации консоли



Выбор **Exit** и нажатие **Enter** вернет вас к послеустановочной настройке.

2.10.8. Установка часового пояса (Setting The Time Zone)

Установка часового пояса на компьютере позволит ему автоматически вносить поправки к местному времени и правильно выполнять другие, связанные с часовым поясом функции.

Пример приведен для компьютера, расположенного в восточном часовом поясе Соединенных Штатов. Ваш выбор будет зависеть от вашего географического положения.

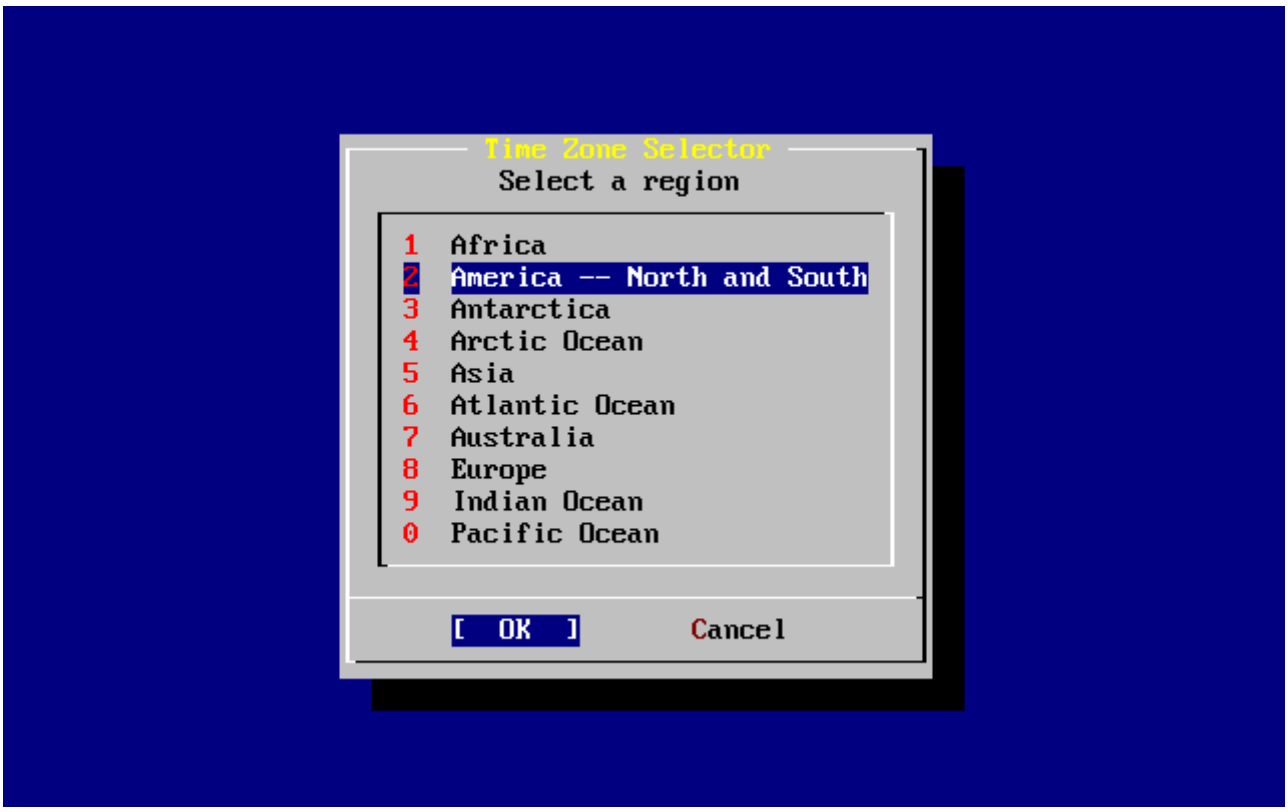
```
User Confirmation Requested

Would you like to set this machine's time zone now?


[ Yes ]   No
```

Выберите [Yes] и нажмите **Enter** для установки часового пояса.

```
User Confirmation Requested

Is this machine's CMOS clock set to UTC? If it is set to local time

or you don't know, please choose NO here!


Yes   [ No ]
```

Выберите [Yes] или [No] в зависимости от настроек часов компьютера и нажмите **Enter**.

Рисунок 2-38. Выбор региона



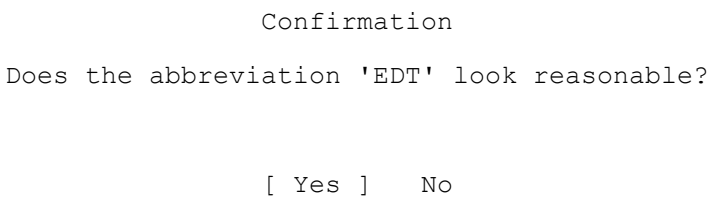
Соответствующий регион выбран с помощью клавиш навигации и нажат **Enter**.

Рисунок 2-39. Выбор страны

Выберите соответствующую страну с помощью клавиш навигации и нажмите **Enter**.

Рисунок 2-40. Выбор часового пояса

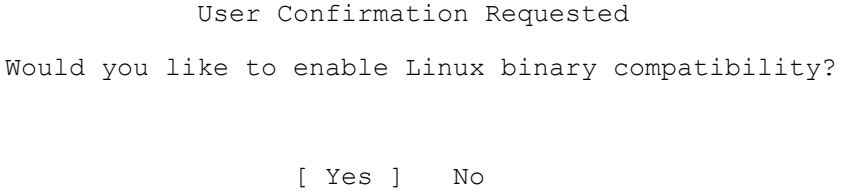
Выбран соответствующий часовой пояс с помощью клавиш навигации и нажат **Enter**.



Правильно будет согласиться с назначением аббревиатуры временного пояса. Если она подходит, нажмите **Enter**, чтобы продолжить послеустановочную настройку.

2.10.9. Совместимость с Linux (Linux Compatibility)

Замечание: Эта информация применима к установке FreeBSD 7.*X*. Если вы устанавливаете FreeBSD 8.*X*, нижеследующее меню предложено не будет.



Выбор [Yes] и нажатие **Enter** позволит запускать программы Linux под FreeBSD. Программа установки добавит соответствующие пакеты для совместимости с Linux.

При установке по FTP, компьютеру потребуется соединиться с интернет. Иногда на сервере ftp нет всех необходимых компонент, например для бинарной совместимости с Linux. Эти компоненты могут быть установлены позже, если потребуется.

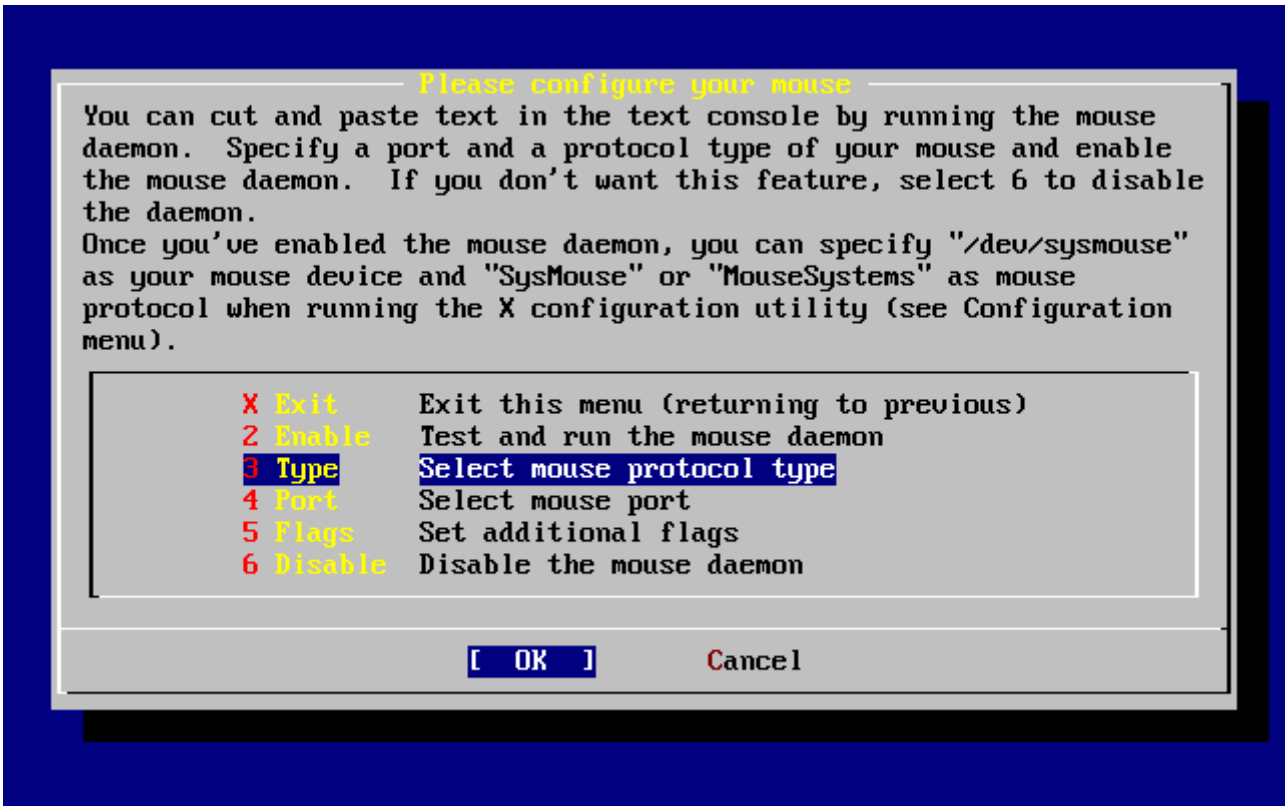
2.10.10. Настройка мыши (Mouse Settings)

Эти настройки позволят вырезать и вставлять текст в консоли и пользовательских программах с помощью трехкнопочной мыши. Если используется двухкнопочная мышь, обратитесь к странице справочника moused(8) после установки, чтобы узнать подробности об эмуляции трехкнопочной мыши. Этот пример приведен для настройки не-USB мыши (например мыши для порта PS/2 или COM):

```
                    User Confirmation Requested

        Does this system have a PS/2, serial, or bus mouse?


                          [ Yes ]     No
```

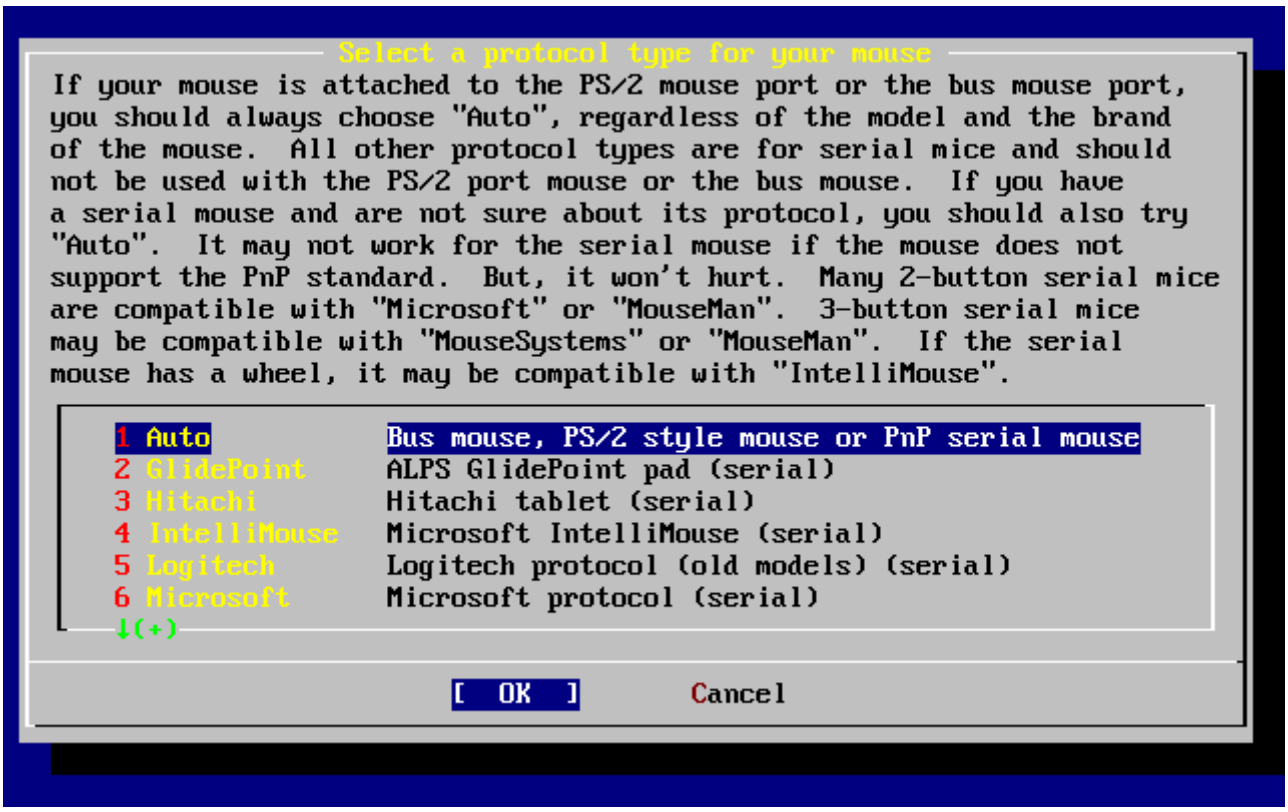
Выберите [Yes] для PS/2 мыши, последовательной мыши или мыши типа bus mouse. Выберите [No] для USB мыши и нажмите **Enter**.

Рисунок 2-41. Выбор протокола мыши



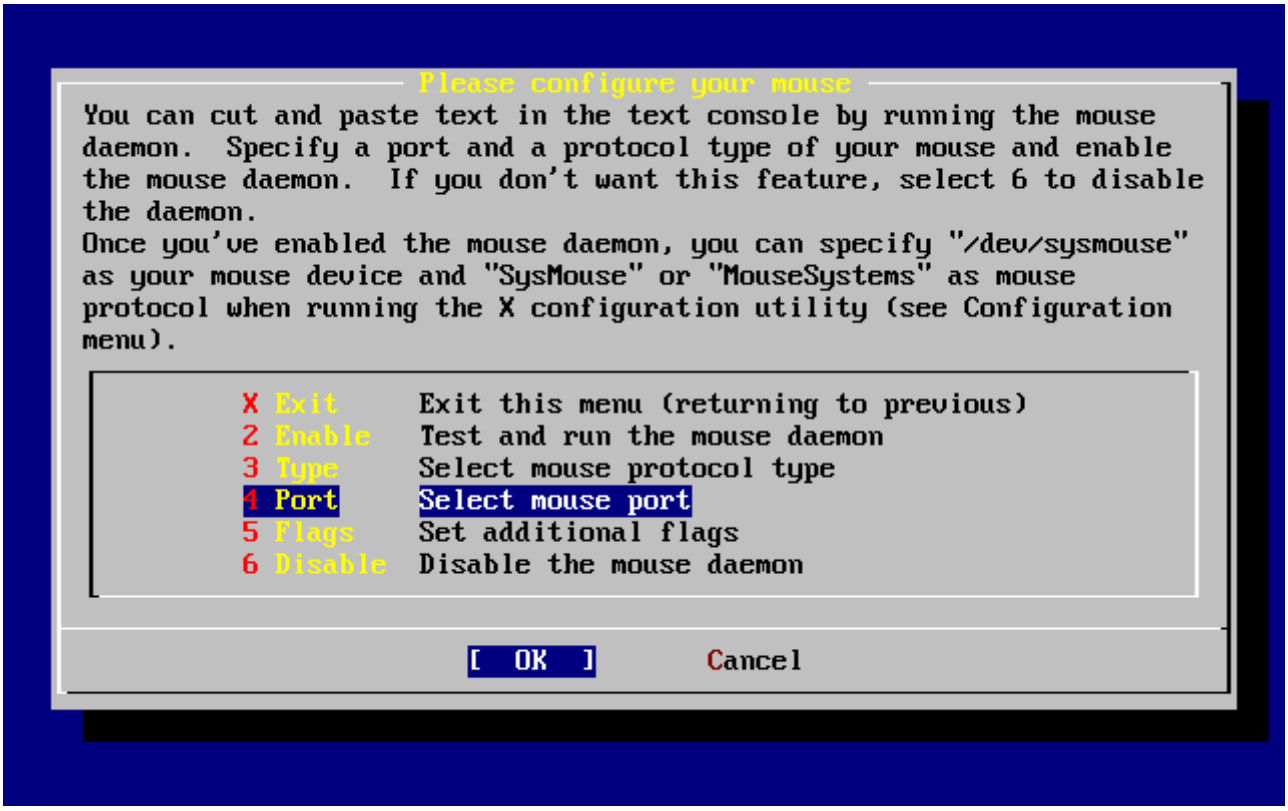
Используйте клавиши навигации для выбора **Type** и нажмите **Enter**.

Рисунок 2-42. Установка протокола мыши



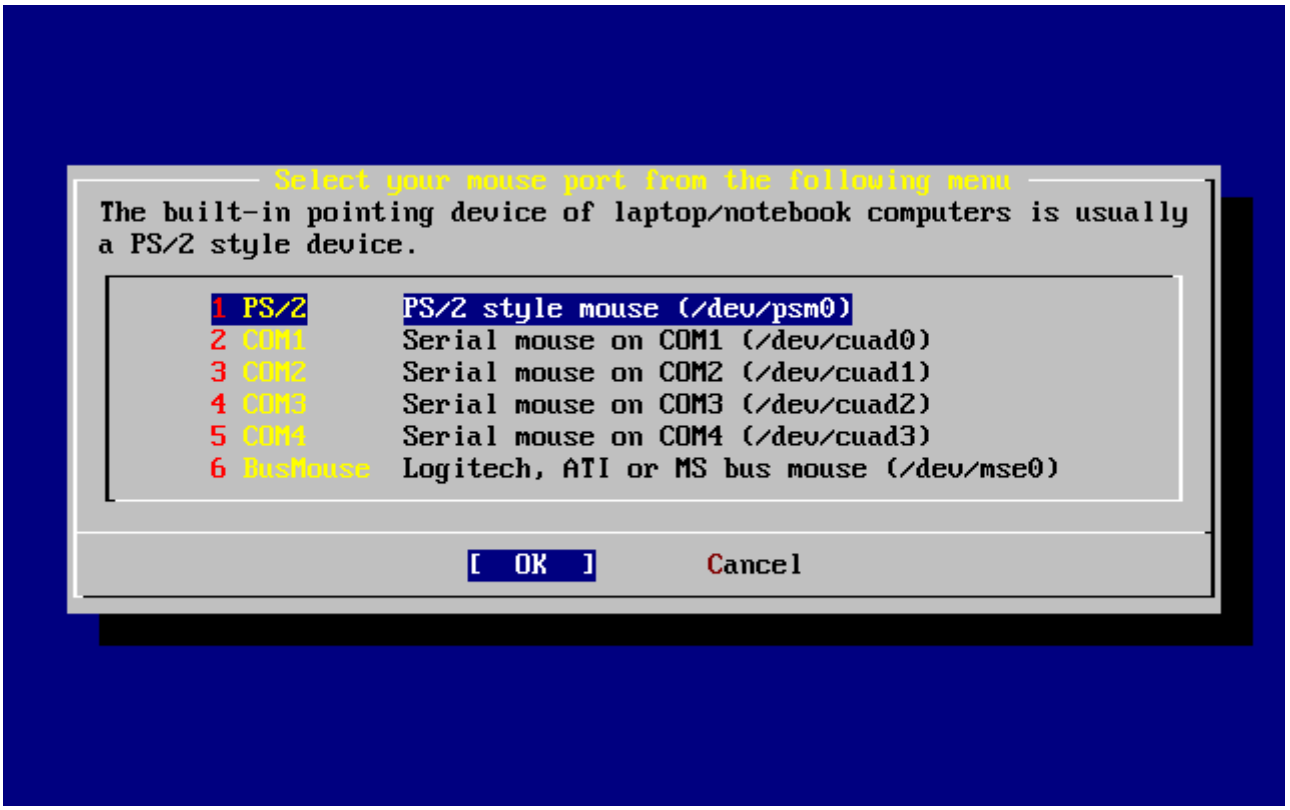
В этом примере использована PS/2 мышь, поэтому подойдет протокол по умолчанию **Auto**. Чтобы изменить протокол, используйте клавиши навигации для выбора другого пункта. Убедитесь, что выбран [OK], и нажмите **Enter** для выхода из меню.

Рисунок 2-43. Настройка порта мыши



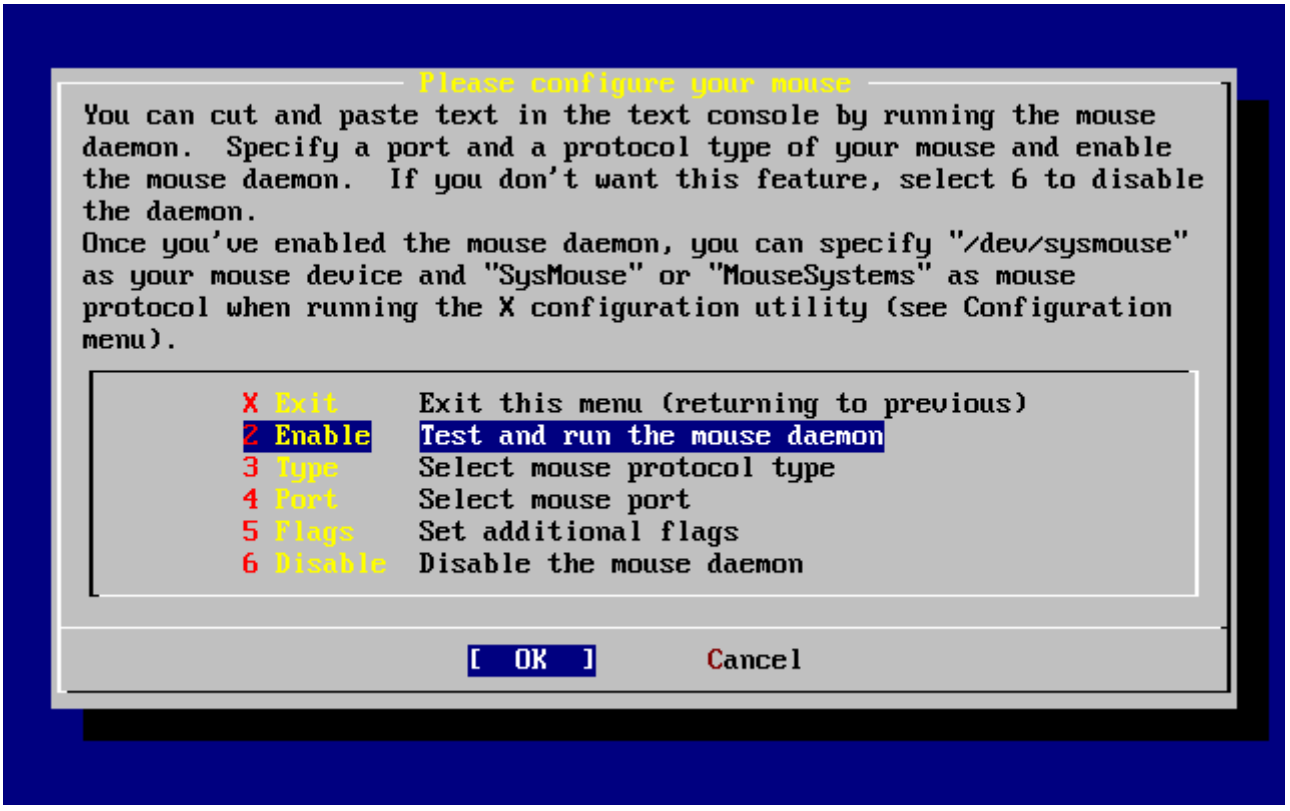
Используйте клавиши навигации для выбора **Port** и нажмите **Enter**.

Рисунок 2-44. Установка порта мыши



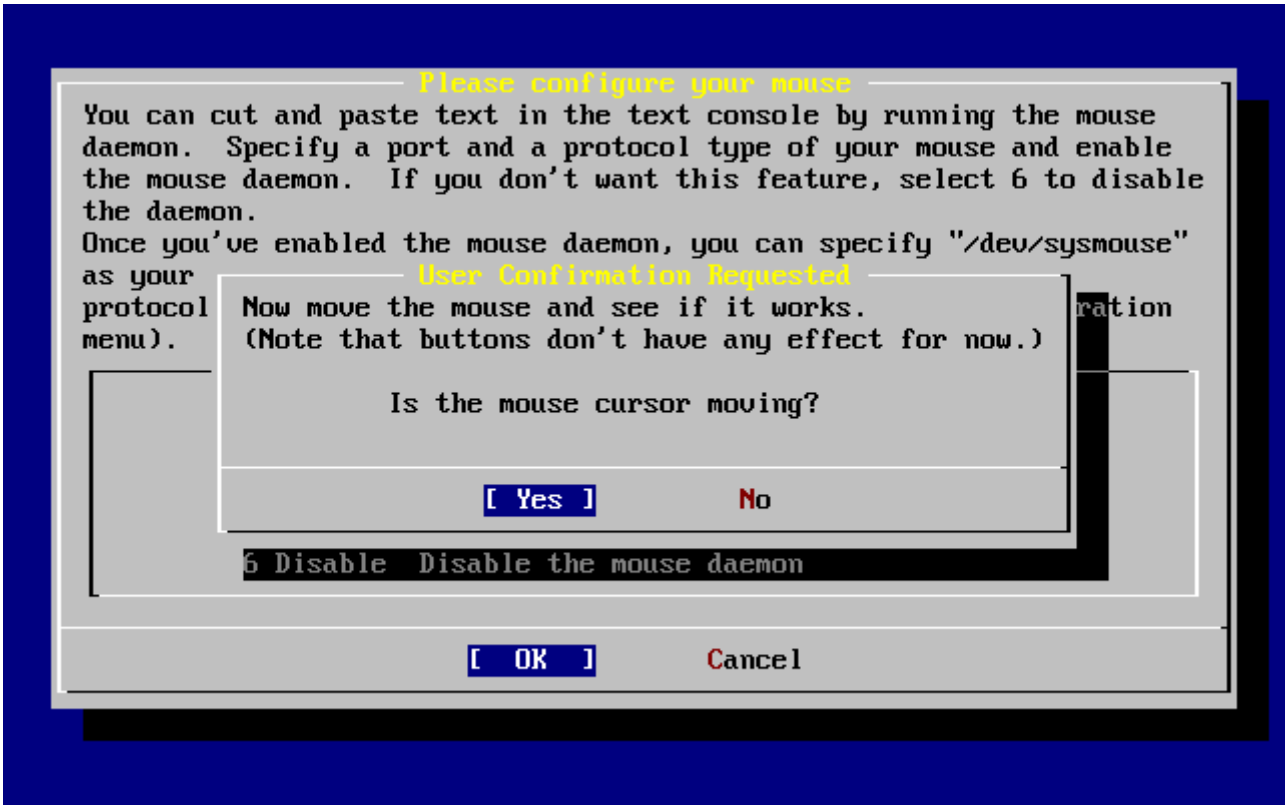
К этой системе подключена мышь PS/2, поэтому подходит значение по умолчанию **PS/2**. Чтобы изменить порт, используйте клавиши навигации и нажмите **Enter**.

Рисунок 2-45. Запуск даемона мыши



Наконец, используйте клавиши навигации для выбора **Enable**, затем нажмите **Enter** для запуска и тестирования даемона мыши.

Рисунок 2-46. Проверка даемона мыши



Подвигайте курсор по экрану и убедитесь, что он движется правильно. Если это так, выберите [Yes] и нажмите **Enter**. Если нет, мышь не была правильно настроена — выберите [No] и попробуйте использовать другие опции настройки.

Выберите **Exit** с помощью клавиш навигации и нажмите **Enter** для возврата к послеустановочной настройке.

2.10.11. Установка пакетов (Install Packages)

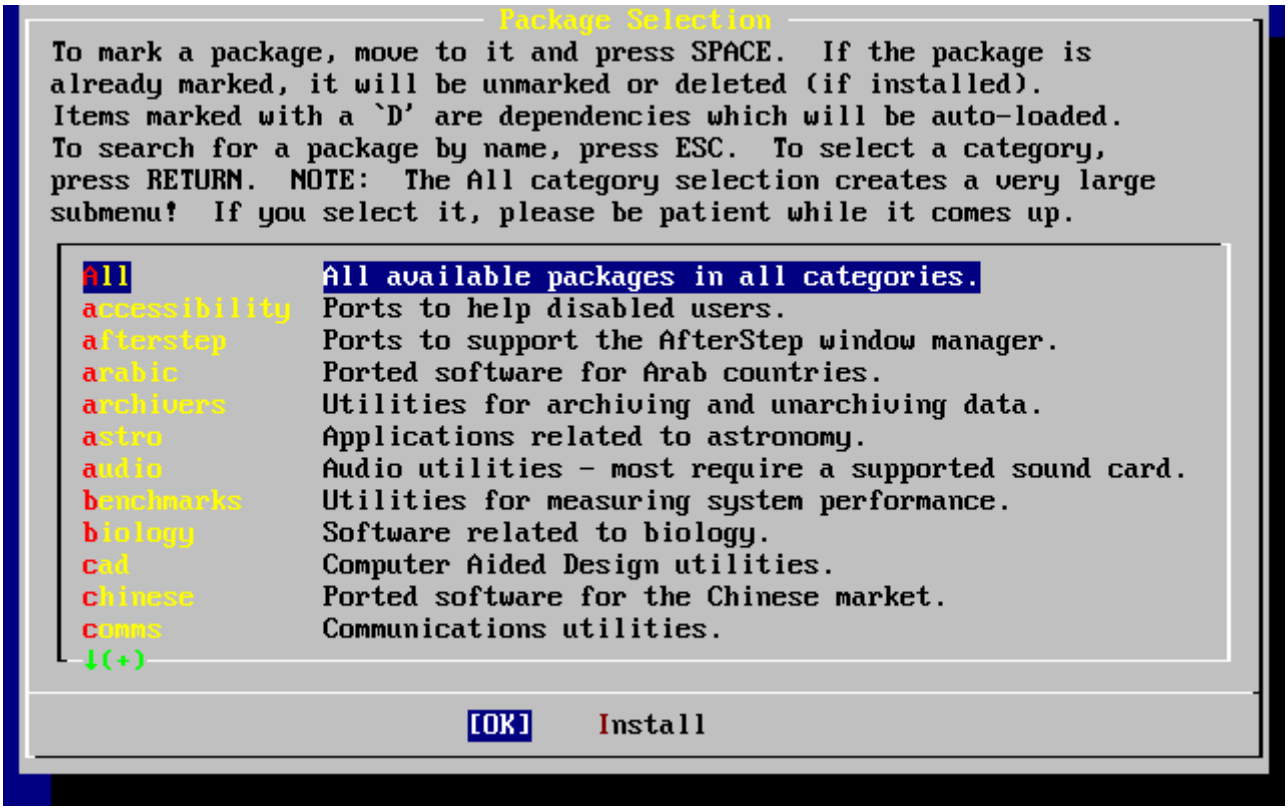
Пакеты — это прекомпилированные бинарные файлы и это удобный способ установки программ.

В качестве примера показана установка одного пакета. Если потребуется, можно установить дополнительные пакеты. После установки для добавления пакетов может быть использована команда `sysinstall`.

```
                    User Confirmation Requested

 The FreeBSD package collection is a collection of hundreds of

 ready-to-run applications, from text editors to games to WEB servers

 and more. Would you like to browse the collection now?


                              [ Yes ]    No
```

Выбор **[Yes]** и нажатие **Enter** приведет к появлению экрана выбора пакетов:

Рисунок 2-47. Выбор категории пакетов

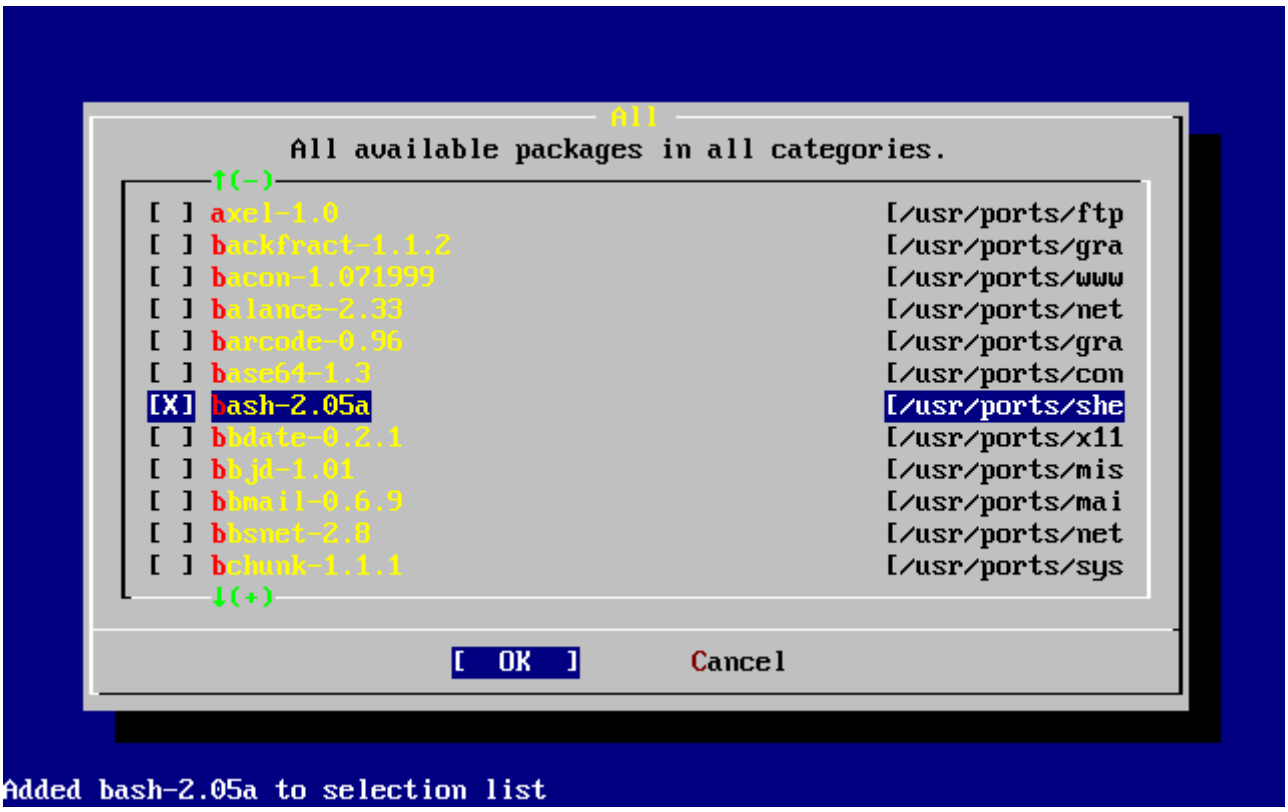


Только пакеты с текущего носителя доступны для установки в любое время.

Все доступные пакеты будут показаны если выбрать категорию **All**, можно также выбирать отдельные категории. Перейдите к выбранной категории с помощью клавиш навигации и нажмите **Enter**.

Появится меню, содержащее доступные в данной категории пакеты.

Рисунок 2-48. Выбор пакетов



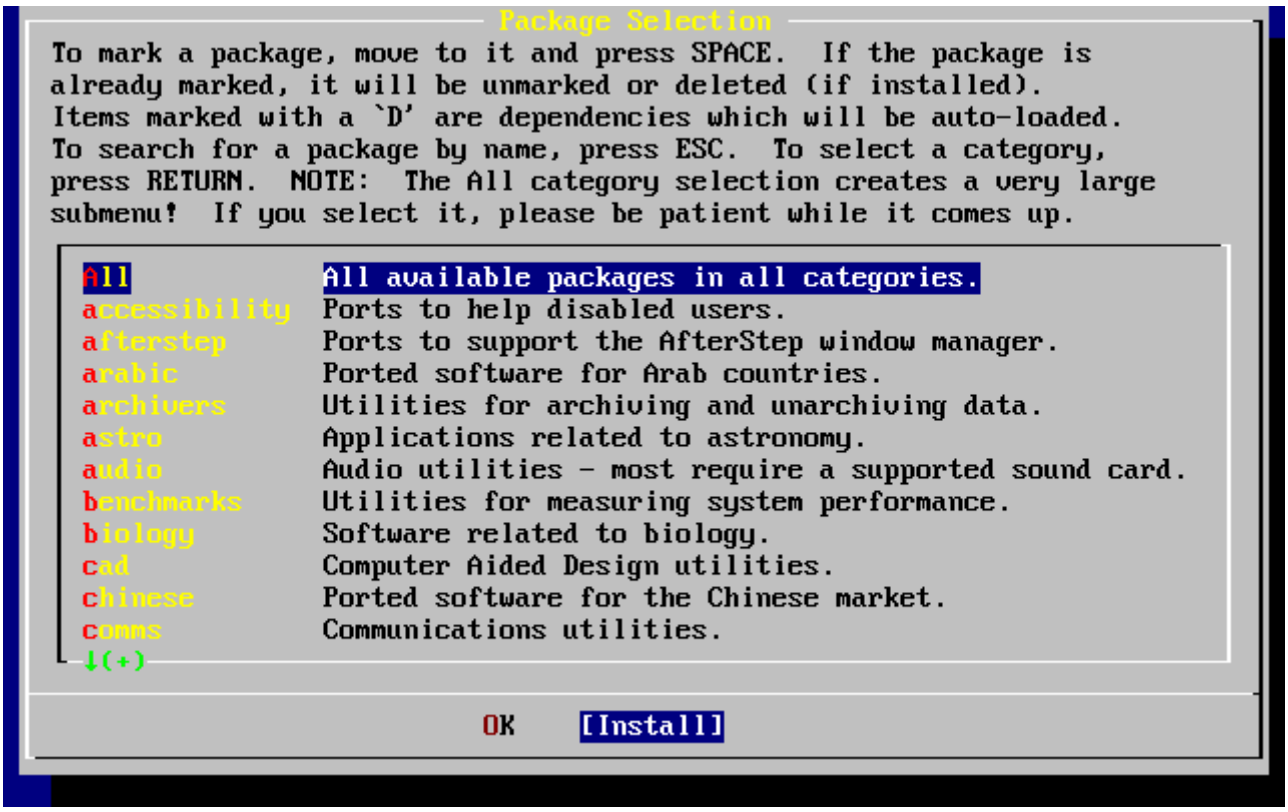
Выбрана оболочка **bash**. Выберите все необходимые пакеты, перемещаясь по меню и нажимая клавишу пробела на выбираемых пакетах. Краткое описание пакета будет появляться в нижней левой части экрана.

Нажатие **Tab** переключает между последним выбранным пакетом, [OK], и [Cancel].

После того, как будет закончена отметка пакетов для установки, нажмите **Tab** один раз для переключения на [OK] и нажмите **Enter** для переключения на меню выбора пакетов.

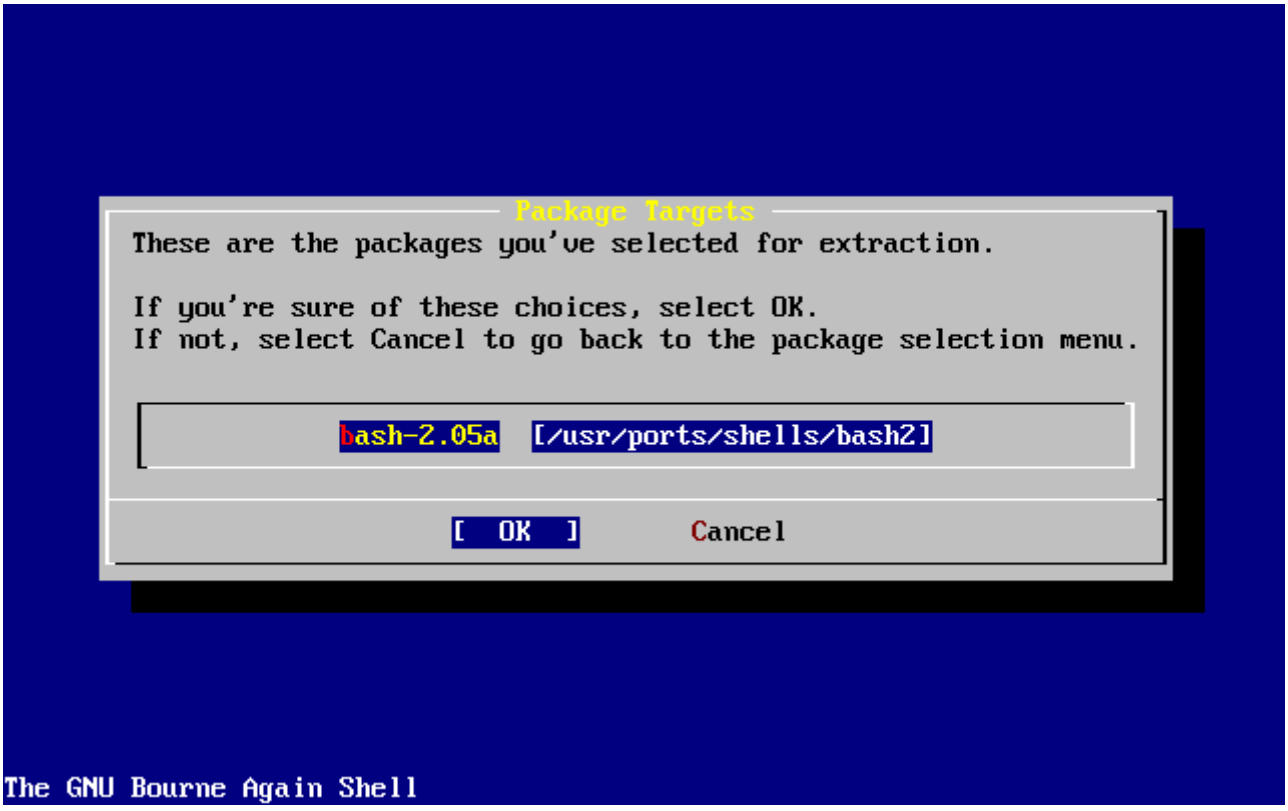
Нажимая клавиши навигации влево или вправо, можно переключаться между [OK] и [Cancel]. Этот метод может быть применен также для выбора [OK] и возврата к меню выбора пакетов нажатием **Enter**.

Рисунок 2-49. Установка пакетов



Используйте **Tab** и клавиши навигации для выбора [Install] и нажмите **Enter**. вам потребуется подтвердить установку пакетов:

Рисунок 2-50. Подтверждение установки пакетов



Выбор [OK] и нажатие **Enter** запустит установку пакетов. Во время установки будут выдаваться сообщения. Обратите внимание на возможные сообщения об ошибках.

После установки пакетов настройка продолжится. Если вы не выбрали ни один из пакетов и хотите вернуться к завершению настройки, выберите Install в любом случае.

2.10.12. Добавление пользователей/групп (Add Users/Groups)

В процессе установки нужно добавить хотя бы одного пользователя, чтобы использовать систему без входа под `root`. Корневой каталог обычно мал и запуск приложений под `root` быстро заполнит его. Ниже показано предупреждение:

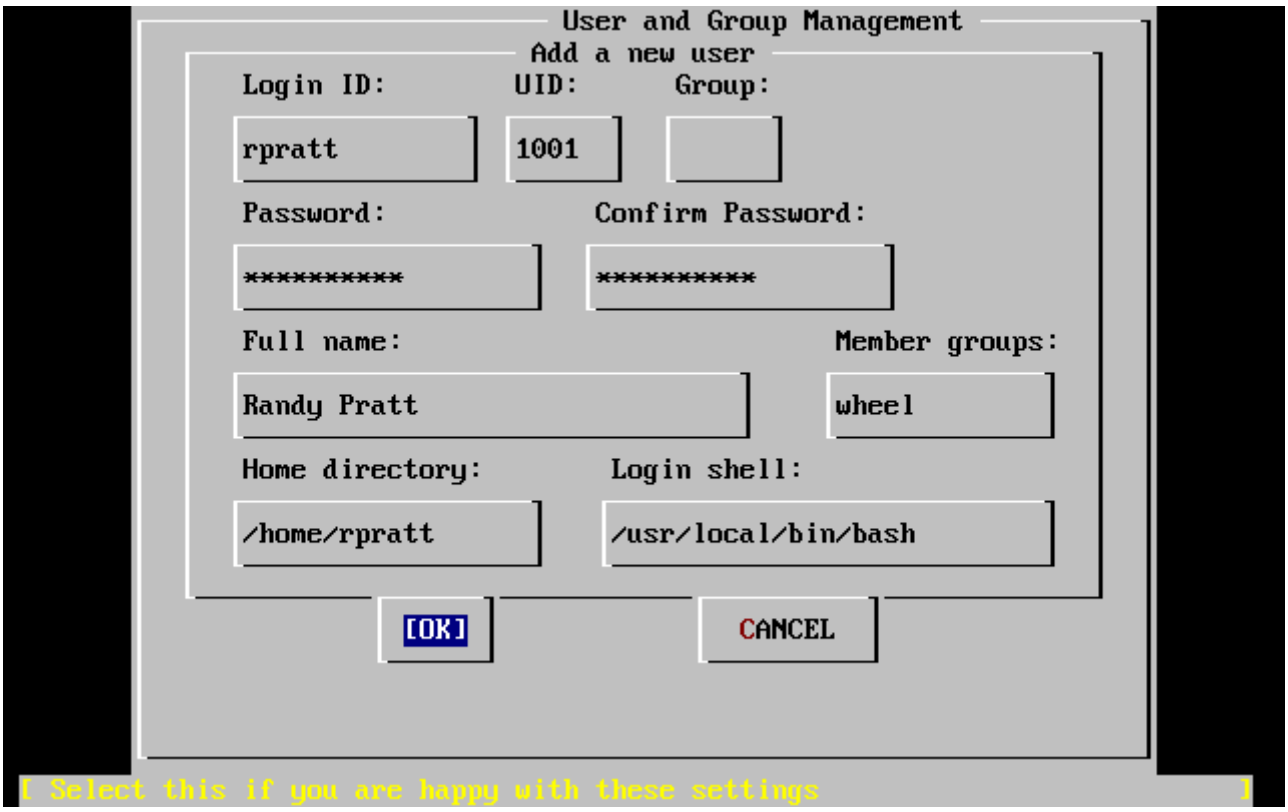
```
                    User Confirmation Requested

Would you like to add any initial user accounts to the system? Adding
at least one account for yourself at this stage is suggested since
working as the "root" user is dangerous (it is easy to do things which
adversely affect the entire system).


                          [ Yes ]    No
```

Выберите [Yes] и нажмите **Enter**, чтобы продолжить добавление пользователя.

Рисунок 2-51. Выбор User (пользователь)

Выберите **User** с помощью клавиш навигации и нажмите **Enter**.

Рисунок 2-52. Вод информации о пользователе



При выборе полей с помощью **Tab** в нижней части экрана будет появляться описание, помогающее ввести необходимую информацию:

Логин (Login ID)

Имя нового пользователя (обязательно).

UID

Числовой ID (идентификатор) для этого пользователя (оставьте пустым для автоматического выбора).

Группа (Group)

Имя группы этого пользователя (оставьте пустым для автоматического выбора).

Пароль (Password)

Пароль этого пользователя (заполняйте это поле с осторожностью!).

Полное имя

Полное имя пользователя (комментарий).

Член групп (Member groups)

Группы, к которым принадлежит пользователь (т.е. имеет права доступа).

Домашний каталог (Home directory)

Домашний каталог пользователя (оставьте пустым для выбора по умолчанию).

Оболочка (Login shell)

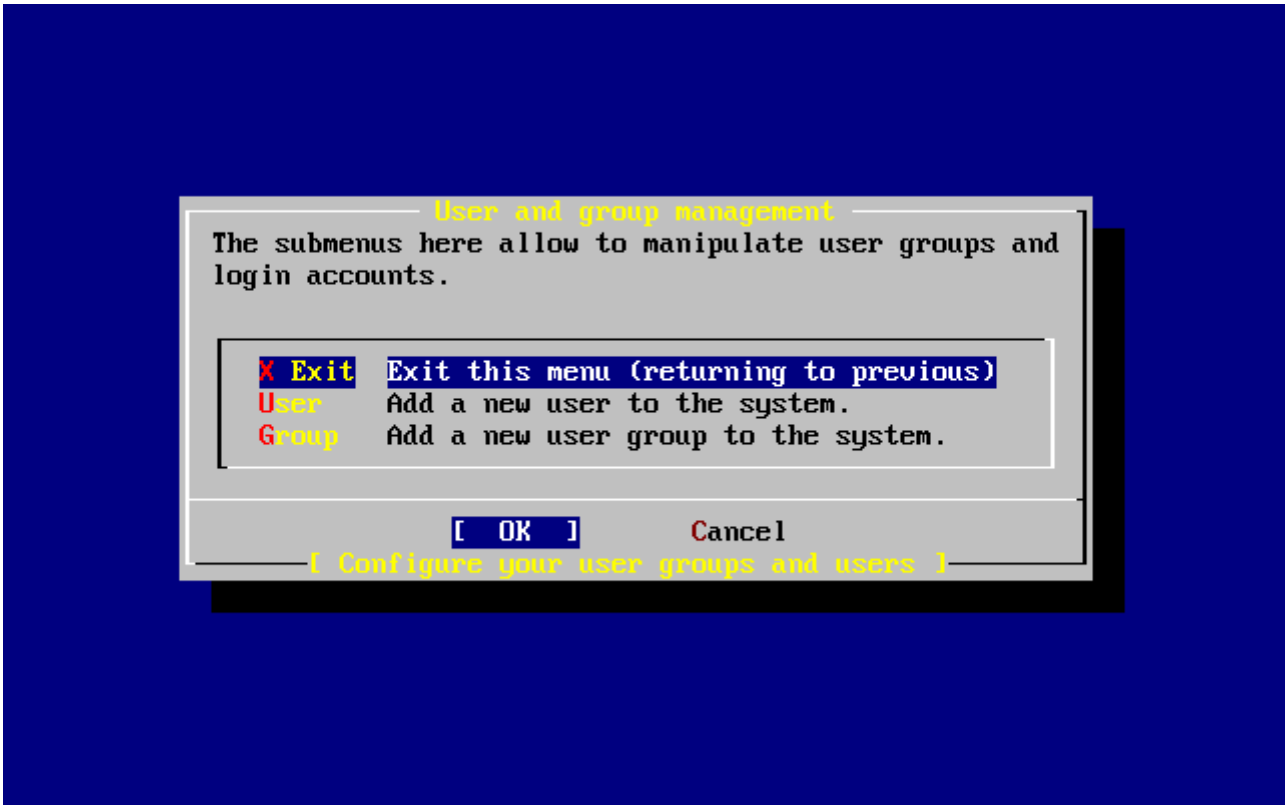
Оболочка пользователя, запускаемая при входе в систему (оставьте пустым для оболочки по умолчанию, например `/bin/sh`).

Оболочка была изменена с `/bin/sh` на `/usr/local/bin/bash` для использования **bash**, которая была перед этим установлена из пакета. Не пытайтесь использовать несуществующую оболочку, вы не сможете войти в систему. Наиболее часто используемая в мире BSD оболочка это C shell, которую можно обозначить как `/bin/tcsh`.

Пользователь был добавлен в группу `wheel`, чтобы иметь возможность стать суперпользователем с привилегиями `root`.

Когда все будет введено, нажмите [OK] и меню управления пользователями и группами (User and Group Management) появится снова:

Рисунок 2-53. Выход из меню управления пользователями и группами



Сейчас также можно добавить группы, если известно, для чего они потребуются. Иначе в это меню можно войти, запустив `sysinstall` после окончания установки.

После завершения добавления пользователей, выберите **Exit** с помощью клавиш навигации и нажмите **Enter** для продолжения установки.

2.10.13. Установка пароля `root`

```
                          Message

 Now you must set the system manager's password.

 This is the password you'll use to log in as "root".


                          [ OK ]


                [ Press enter or space ]
```

Нажмите **Enter** для установки пароля `root`.

Необходимо два раза правильно ввести пароль. Излишне упоминать, что должна быть возможность восстановления пароля, если вы его забудете. Обратите внимание, что ни набираемый пароль, ни звездочки на экран не выдаются.

```
New password:

Retype new password :
```

Установка продолжится после успешного ввода пароля.

Лабораторная работа №6. Основные утилиты Unix

Основные команды
man – вывод справки (# man ls | more)
whatis – поиск по точному совпадению (предварительно сделать # makewhatis)
apropos – поиск по фрагменту (# apropos list | less)

pwd – текущий каталог
ls – список файлов каталога (# ls –lah | less)
cd – сменить каталог (# cd ~)
mkdir – создание каталога
rmdir- удаление каталога

cp – копирование файлов
rm – удаление файлов
mv – переименование файлов
more – вывод файла на экран
find – поиск (#find /home/ -name “test*” -print)
grep – поиск текста в файле (# grep -i substr 1.txt)
ee – текстовый редактор
touch – изменить временные атрибуты доступа и модификации файлов
chmod – изменить права на файл

adduser – добавить пользователя
deluser – удалить пользователя
passwd – задать/сменить пароль
su – выполнение команд от имени другого пользователя
finger – список пользователей

ps – вывести состояние процессов
top – вывести процессы
kill – прекратить исполнение процесса
du – статистика использования диска (#du –shx, du -t)
exit – завершение сеанса
shutdown – завершение работы (-h, -r, -p) (#shutdown –p now #reboot)

jobs – текущие задания

Монтирование файловых систем

mount – монтировать файловую систему (# mount /dev/ hdc /mnt/hdc -t iso9660 -o ro) (#mount –a монтировать все фс из /etc/fstab)
umount – размонтировать файловую систему
mount без аргументов – список смонтированных файловых систем

$ ls -1 /sbin/mount – поддерживаемые файловые системы
# mount_cd9660 /dev/acd0 /cdrom – для любого сд
# mount_msdosfs /dev/ad## /mnt – для любого FAT

/etc/fstab - конфигурация автомонтирования

Локализация

Включение раскладки клавиатуры Windows
/etc/rc.conf

```
keymap="ru.koi8-r.win"
```

Локализация консоли
/etc/csh.login

```
setenv LANG ru_RU.KOI8-R
setenv MM_CHARSET KOI8-R
```

etc/profile

```
LANG=ru_RU.KOI8-R; export LANG
MM_CHARSET=KOI8-R; export MM_CHARSET
```

Работа с ФТП
ftp – программа пересылки файлов
help – справка по команде
open – установить соединение
pwd – текущий каталог
dir – содержимое каталога
cd – сменить каталог
ascii – режим передачи текстовых файлов
binary – режим передачи бинарных файлов
get – получить файл
status – текущий статус
close – прекратить сеанс
exit – выйти из программы

Практическая работа №7. Установка приложений Unix

Стандартная процедура установки программного обеспечения сторонних разработчиков выглядит примерно так:

1. Загрузка программного обеспечения, которое может распространяться в форме исходных текстов или двоичных файлов.
2. Распаковка программного обеспечения из дистрибутивного формата (обычно tar-архива, сжатого при помощи compress(1), gzip(1) или bzip2(1)).
3. Поиск документации (возможно, подойдут файлы `INSTALL`, `README` или несколько файлов из подкаталога `doc/`) и её чтение в поиске описания установки программного обеспечения.

4. Если программное обеспечение распространялось в форме исходных текстов, его компиляция. Сюда может быть включено редактирования файла `Makefile`, запуск скрипта `configure` и другие работы.
5. Тестирование и установка программного обеспечения.

И это только всё проходит нормально. Если вы устанавливаете программный пакет, который был специально перенесён на FreeBSD, то вам может даже потребоваться редактировать код для того, чтобы он нормально заработал.

Если вы хотите, то можете продолжать устанавливать программное обеспечение во FreeBSD "традиционным" способом. Однако FreeBSD предоставляет две технологии, которые могут сохранить вам много усилий: пакеты и порты. На момент написания таким образом были доступны более 23,000 сторонних приложений.

Для любого конкретно взятого приложения пакет FreeBSD является одним файлом, который вы должны загрузить. Пакет содержит уже откомпилированные копии всех команд приложения, а также все конфигурационные файлы и документацию. Загруженным файлом пакета можно управлять такими командами FreeBSD, как pkg_add(1) pkg_delete(1), pkg_info(1) и так далее. Установка нового приложения может выполняться единственной командой.

Порт FreeBSD для приложения является набором файлов, предназначенных для автоматизации процесса компиляции приложения из исходного кода.

Вспомните, что обычно вы должны выполнить некоторое количество шагов, если компилируете программу самостоятельно (загрузка, распаковка, изменение кода, компиляция, установка). Файлы, составляющие порт, содержат всю информацию, необходимую для того, чтобы система сделала это за вас. Вы задаёте пару простых команд, и исходный код приложения автоматически загружается, распаковывается, модифицируется, компилируется и устанавливается.

Действительно, система портов может также использоваться для генерации пакетов, которые позже могут управляться командой `pkg_add` и другими командами управления пакетами, о которых скоро будет рассказано.

Как пакеты, так и порты принимают во внимание *зависимости*. Предположим, что вы хотите установить приложение, которое зависит от некоторой установленной библиотеки. И приложение, и библиотека доступны во FreeBSD в виде портов и пакетов. Если вы используете команду `pkg_add` или систему портов для добавления приложений, то в обоих случаях будет обнаружено, что библиотека не была установлена, и сначала будет автоматически выполнена установка библиотеки.

Видя, что обе технологии весьма похожи, вы можете удивиться, почему во FreeBSD используются обе. И пакеты, и порты имеют свои преимущества, так что выбор используемой вами системы зависит от ваших собственных предпочтений.

Преимущества пакетов

- Сжатый tar-архив обычно меньше, чем сжатый tar-архив, содержащий исходный код приложения.

- Пакеты не требуют никакой дополнительной компиляции. Для таких больших приложений, как **Mozilla**, **KDE** или **GNOME**, это может быть важно, в частности, если вы работаете на медленной системе.
- Пакеты не требуют понимания процесса компиляции программного обеспечения во FreeBSD.

Преимущества портов

- Пакеты обычно компилируются с консервативными параметрами, потому что они должны работать на максимальном количестве систем. При установке из порта вы можете изменять параметры компиляции для того, чтобы (к примеру) генерировался код, специфичный для процессора Pentium IV или Athlon.
- Некоторые приложения имеют опции времени компиляции, связанные с тем, что они могут или не могут делать. К примеру, **Apache** может быть настроен с широким набором различных опций. При построении из порта вы можете не принимать параметры по умолчанию, и задать их самостоятельно.

  В некоторых случаях для одного и того же приложения будут иметься несколько пакетов для указания конкретных настроек. Например, **Ghostscript** имеется как пакет `ghostscript` и как пакет `ghostscript-nox11`, в зависимости от того, установили вы сервер X11 или нет. Такой тип грубой настройки возможен при использовании пакетов, но быстро становится недостижим, если приложение имеет более одного или двух параметров компиляции.

- Условия лицензирования некоторых дистрибутивов программного обеспечения запрещает распространение в двоичном виде. Они должны распространяться в виде исходного кода.
- Некоторые не доверяют дистрибутивам в двоичном виде. При использовании исходного кода вы (по крайней мере теоретически) можете прочесть его и попытаться найти потенциальные проблемы самостоятельно.
- Если у вас есть собственные патчи, вам нужен исходный код для того, чтобы их применять.
- Некоторым нравится иметь исходный код, чтобы его можно было просматривать и править, заимствовать из него (конечно, при разрешающем это лицензионном соглашении) и тому подобное.

Чтобы отслеживать обновления портов, подпишитесь на Список рассылки, посвящённый Портам FreeBSD и Список рассылки, посвящённый ошибкам в портах FreeBSD.

Внимание: Перед установкой любого приложения необходимо зайти на http://vuxml.freebsd.org/, где находится информация по вопросам безопасности приложений.

Вы можете также установить `security/portaudit`, который автоматически проверит все установленные приложения на наличие известных уязвимостей, проверка также будет выполняться перед сборкой какого-либо порта. Вы можете использовать `portaudit -F -a` и после установки пакетов.

В оставшейся части главы будет рассказано, как использовать пакеты и порты для установки и управления программным обеспечением сторонних разработчиков во FreeBSD.

Перед тем, как устанавливать какое-либо приложение, вам нужно знать, что вы хотите и как называется нужное вам приложение.

Список имеющихся для FreeBSD приложений постоянно растёт. К счастью, есть несколько способов найти то, что вам нужно:

- На сайте FreeBSD поддерживается обновляемый список имеющихся приложений для FreeBSD, в котором можно выполнять поиск, по адресу http://www.FreeBSD.org/ports/. Порты разбиты на категории, и вы можете либо выполнить поиск приложения по имени (если его знаете), либо просмотреть список всех приложений, относящихся к определённой категории.
- Dan Langille поддерживает сайт FreshPorts по адресу http://www.FreshPorts.org/. На нём отслеживаются изменения в приложениях из дерева портов, как только они происходят, он позволяет вам "отслеживать" один или несколько портов, и может высылать оповещение по электронной почте при их обновлении.
- Если вы не знаете названия нужного вам приложения, попытайтесь воспользоваться сайтом типа FreshMeat (http://www.freshmeat.net/) для поиска приложения, а затем возвратитесь на сайт FreeBSD, чтобы проверить, есть ли порт для этого приложения.
- Если вы знаете точное имя порта, и хотите определить, в какой категории он находится, используйте команду whereis(1). Просто наберите в приглашении "`whereis file`", где *`file`* - программа, которую вы хотите установить. И если она имеется в системе, об этом будет сообщено, как показано ниже:

- `# whereis lsof`

- `lsof: /usr/ports/sysutils/lsof`

  Это говорит о том, что `lsof` (системная утилита) находится в каталоге `/usr/ports/sysutils/lsof`.

- Ещё одним способом поиска некоторого порта является использование встроенной возможности поиска в Коллекции Портов. Чтобы ею воспользоваться, вы должны находиться в каталоге `/usr/ports`. Очутившись в этом каталоге, выполните команду `make search name=`*`program-name`*, где *`program-name`* – это название программы, которую вы хотите найти. Например, если вы ищете `lsof`:

- `# cd /usr/ports`

- `# make search name=lsof`

- `Port:  lsof-4.56.4`
- `Path:  /usr/ports/sysutils/lsof`
- `Info:  Lists information about open files (similar to fstat(1))`
- `Maint:  obrien@FreeBSD.org`
- `Index:  sysutils`
- `B-deps:`
- `R-deps:`

Вам следует обратить особое внимание на строчку "Path:", так как в ней указывается, где найти порт. Остальная сообщаемая информация для установки порта не нужна, поэтому здесь она описываться не будет.

Для выполнения более глубокого поиска вы можете также использовать `make search key=`*`string`*, где *`string`* представляет собой некоторый текст, относящийся к искомому порту. При этом будет выполнен поиск в именах портов, комментариях, описаниях и зависимостях, и его можно использовать для поиска портов, связанных с некоторой темой, если вы не знаете названия программы, которую вы ищете.

В обоих этих случаях строка поиска нечувствительна к регистру. Поиск "LSOF" приводит к тому же самому результату, что и поиск "lsof".

Для установки пакетов программного обеспечения для FreeBSD из локальных файлов или с сервера в сети вы можете использовать утилиту pkg_add(1).

Пример. Загрузка пакета вручную и его локальная установка

```
# ftp -a ftp2.FreeBSD.org
Connected to ftp2.FreeBSD.org.
220 ftp2.FreeBSD.org FTP server (Version 6.00LS) ready.
331 Guest login ok, send your email address as password.
230-
230-     This machine is in Vienna, VA, USA, hosted by Verio.
230-         Questions? E-mail freebsd@vienna.verio.net.
230-
230-
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /pub/FreeBSD/ports/packages/sysutils/
250 CWD command successful.
ftp> get lsof-4.56.4.tgz
```

```
local: lsof-4.56.4.tgz remote: lsof-4.56.4.tgz

200 PORT command successful.

150 Opening BINARY mode data connection for 'lsof-4.56.4.tgz' (92375 bytes).

100% |***********************************************| 92375       00:00
ETA

226 Transfer complete.

92375 bytes received in 5.60 seconds (16.11 KB/s)

ftp> exit

# pkg_add lsof-4.56.4.tgz
```

Если у вас нет исходных текстов локальных пакетов (например, набор CD-ROM с FreeBSD), то проще всего, наверное, воспользоваться опцией `-r` для pkg_add(1). Это приведёт к тому, что утилита автоматически определит правильный формат объектных файлов и релиз, а затем загрузит и установит пакет с сервера FTP.

```
# pkg_add -r lsof
```

В примере выше нужный пакет будет загружен и установлен без всякого дополнительного взаимодействия с пользователем. Если вместо основного сайта вы хотите указать другое зеркало пакетов FreeBSD, то для переопределения используемых по умолчанию значений вам необходимо задать соответствующим образом значение переменной `PACKAGESITE`. Для загрузки файлов утилита pkg_add(1) использует функцию fetch(3), которая принимает во внимание различные переменные окружения, включая `FTP_PASSIVE_MODE`, `FTP_PROXY` и `FTP_PASSWORD`. Если вы находитесь за сетевым экраном или для работы с FTP/HTTP вам необходимо использовать прокси, то определите соответствующие переменные. Обратитесь к справочной странице по fetch(3) для получения полного списка переменных. Заметьте, что в примере выше вместо `lsof-4.56.4` используется `lsof`. При использовании функций загрузки с сети номер версии в имени пакета должен быть опущен. Утилита pkg_add(1) автоматически загрузит последнюю версию приложения.

Файлы пакетов распространяются в форматах `.tgz` и `.tbz`. Вы можете найти их по адресу ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages или взять с дистрибутива FreeBSD на CD-ROM. Каждый CD из комплекта FreeBSD на 4 дисках (а также PowerPak и тому подобное) содержит пакеты в каталоге `/packages`. Расположение пакетов похоже на то, как организовано дерево `/usr/ports`. Каждая категория имеет собственный каталог, и каждый пакет помещается в каталог `All`.

Структура каталогов системы пакетов соответствует структуре системы портов; они взаимодействуют друг с другом для формирования единой системы пакетов/портов.

pkg_info(1) является утилитой для вывода списка и описаний различных установленных пакетов.

```
# pkg_info

cvsup-16.1          A general network file distribution system optimized for
CV

docbook-1.2         Meta-port for the different versions of the DocBook DTD

...
```

pkg_version(1) является утилитой для вывода отчёта о версиях всех установленных пакетов. Она сравнивает версию имеющегося пакета с текущей версией, находящейся в дереве портов.

```
# pkg_version

cvsup                          =

docbook                        =

...
```

Символы во второй колонке указывают сравнительную разницу в возрасте установленной версии и версии, находящейся в локальном дереве портов.

Символ	Значение
=	Версия установленного пакета соответствует версии, находящейся в локальном дереве портов.
<	Установленная версия старее, чем та, что имеется в дереве портов.
>	Установленная версия новее чем та, что есть в дереве портов. (Скорее всего, локальное дерево портов устарело.)
?	В индексном файле портов установленный пакет не может быть найден. (Это может случиться, например, если установленный порт был удалён из Коллекции Портов или переименован.)
*	Имеется несколько версий пакета.

Для удаления ранее установленных пакетов с программным обеспечением используйте утилиту pkg_delete(1).

```
# pkg_delete xchat-1.7.1
```

Вся информация о пакете хранится в каталоге `/var/db/pkg`. Список установленных файлов и описания всех пакетов могут быть найдены среди файлов этого каталога.

Практическое занятие .Командный интерфейс Unix. Основные команды Bourne Shell

Цель

Научиться использовать основные команды командного интерпретатора Bourne Shell: перемещение по каталогам, просмотр и редактирование файлов, копирование и удаление.

Теоретические сведения

Основные команды

man – вывод справки (# man ls | more)
whatis – поиск по точному совпадению (предварительно сделать # makewhatis)
apropos – поиск по фрагменту (# apropos list | less)

pwd – текущий каталог
ls – список файлов каталога (# ls –lah | less)

cd – сменить каталог (# cd ~)
mkdir – создание каталога
rmdir- удаление каталога

cp – копирование файлов
rm – удаление файлов
mv – переименование файлов
more – вывод файла на экран
find – поиск (#find /home/ -name “test*” -print)
grep – поиск текста в файле (# grep -i substr 1.txt)
ee – текстовый редактор
touch – изменить временные атрибуты доступа и модификации файлов
chmod – изменить права на файл

adduser – добавить пользователя
deluser – удалить пользователя
passwd – задать/сменить пароль
su – выполнение команд от имени другого пользователя
finger – список пользователей

ps – вывести состояние процессов
top – вывести процессы
kill – прекратить исполнение процесса
du – статистика использования диска (#du –shx, du -t)
exit – завершение сеанса
shutdown – завершение работы (-h, -r, -p) (#shutdown –p now #reboot)
jobs – текущие задания

Практическое занятие 5.Разработка пакетного файла Unix

Цель
Научиться разрабатывать пакетные файлы для Unix.

Теоретические сведения

1. Shell

Установка оболочки по умолчанию
chsh
(предварительно установить EDITOR=ee в $HOME/.profile)

/bin/sh – Bourne Shell
/bin/csh – C Shell

Альясы
/bin/sh – Bourne Shell
alias dir=’ls –lha’

/bin/csh – C Shell
alias dir ls -lha

Файлы
* – исполняемые
. – скрытые
/ – директории

Шаблоны
* используется для обозначения любого объекта
? обозначает любой один символ
[ ] обозначает любой символ из указанных в скобках
> поток в существующий файл
>> поток в новый файл
& в конце команды – фоновый режим
; последовательное выполнение
&& || условное последовательное выполнение (cmd1 && echo success; cmd2 || echo fail)

1. Создание файла
touch testscript

2. Изменение прав доступа
chmod +x testscript

3. Выполнение
./testscript

2. Команды Bourne Shell
#!/bin/sh стандартный заголовок
комментарий
: пустая команда

echo “Hello” – вывод в stdout (опция –n без возврата каретки)
read var1, var2, var3, … – чтение из stdin
var=foo – присвоение значения foo переменной var
var=`bar` – присвоение результата команды bar переменной var
$var – извлечение значения переменной
export var – экспорт в переменную среды

```
#!/bin/sh
echo "Hello World"
clear
read x
read y
z=`expr $x + $y`
echo $z
exit 0
```

$1, $2, … – 1, 2… аргументы командной строки
$# – общее количество аргументов
$@ – строка со всеми аргументами
shift [n] - сдвиг параметров на n

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

```
if [ "$1" = "a" ]
then
  echo "param = $1"
else
  echo "param <> a"
fi
echo $#
echo $@
exit 0
```

if *условие* then ... else … fi	if *условие* then ... else … elif … else … fi	while *условие* do … done	until *условие* do … done	for *var* in *list* do … done	case *foo* in bar) … baz) … qux) … esac

Практическое занятие 7.Компилятор cc и средства разработки программ Unix

Этот раздел касается только GNU-компилятора для C и C++, так как он поставляется вместе с системой FreeBSD. Он может быть вызван командами `cc` или `gcc`. Подробности по генерации программ с помощью интерпретатора зависят от конкретного интерпретатора и обычно хорошо описываются в документации и встроенной системе помощи интерпретатора.

Как только вы напишете свое творение, следующим шагом является преобразование его в нечто, что будет (надеемся!) запускать во FreeBSD. Это обычно происходит в несколько шагов, каждый из которых делается отдельной программой.

1. Предварительная обработка исходного кода для удаления комментариев и выполнения других хитростей, наподобие раскрытия макросов в языке C.
2. Синтаксическая проверка вашего кода на предмет выявления несоответствий правилам языка. Если таковые обнаружились, об этом будет сообщено!
3. Преобразование исходного кода в код на языке ассемблера--он очень близок к машинному коду, но все еще понятен человеку.
4. Преобразование ассемблерного кода в машинный код--да, мы имеем в виду биты и байты, здесь только нули и единицы.
5. Проверка на то, что вы использовали такие вещи, как функции и глобальные переменные правильно. Например, если вы обращались к несуществующей функции, на это будет указано.
6. Если вы пытаетесь получить выполнимый файл из нескольких файлов с исходными текстами, работа над тем, как их объединить.
7. Работа над генерацией того, что загрузчик сможет загрузить в память и запустить.
8. Наконец, запись выполнимого файла в файловую систему.

Термин *компиляция* часто используется для обозначения действий на шагах от 1 до 4--остальные шаги называют *компоновкой*. Иногда шаг 1 называют *препроцессорной обработкой*, а шаги 3-4 *ассемблированием*.

К счастью, почти все эти детали скрыты от вас, `cc` как конечная программа управляет за вас вызовом всех этих программ с правильными аргументами; простой вызов

```
% cc foobar.c
```

приведет к компиляции `foobar.c` посредством всех шагов выше. Если у вас имеется для компиляции больше одного файла, просто сделайте нечто вроде следующего

```
% cc foo.c bar.c
```

Заметьте, что синтаксическая проверка--это именно проверка синтаксиса. При этом не выполняется проверка на наличие сделанных вами логических ошибок, типа вхождения в бесконечный цикл или использования пузырьковой сортировки там, где вы хотели использовать двоичную.

Имеется огромное количество параметров для `cc`, все они описаны на справочной странице. Вот несколько из самых важных, с примерами их использования.

`-o filename`

Имя выходного файла. Если вы не используете этот параметр, `cc` сгенерирует выполнимый файл с именем `a.out`.

```
% cc foobar.c                выполнимый файл называется a.out
% cc -o foobar foobar.c      выполнимый файл называется foobar
```

`-c`

Выполнить только компиляцию файла, без компоновки. Полезно для игрушечных программ, когда вы хотите просто проверить синтаксис, или при использовании `Makefile`.

```
% cc -c foobar.c
```

В результате будет сгенерирован *объектный файл* (не выполнимый файл) с именем `foobar.o`. Он может быть скомпонован с другими объектными файлами для получения выполнимого файла.

`-g`

Создать отладочную версию выполнимого файла. Этот параметр указывает компилятору поместить в выполнимый файл информацию о том, какая строка какого файла с исходным текстом какому вызову функции соответствует. Отладчик может использовать эту информацию для вывода исходного кода при пошаговом выполнении программы, что *очень* полезно; минусом является то, что вся эта дополнительная информация делает программу гораздо большей. Обычно вы компилируете с параметром `-g` при работе над программой, а затем, когда убедитесь в работоспособности программы, компилируете "окончательную версию" без параметра `-g`.

```
% cc -g foobar.c
```

При этом будет сгенерирована отладочная версия программы. [4]

`-O`

Создать оптимизированную версию выполнимого файла. Компилятор прибегает к различным ухищрениям для того, чтобы сгенерировать выполнимый файл, выполняющийся быстрее, чем обычно. После опции `-O` вы можете добавить число, указывающее качество оптимизации, но использование этого не защищено от ошибок оптимизации компилятора. Например, известно, что версия компилятора `cc`, поставляемая с FreeBSD версии 2.1.0, при некоторых условиях генерирует неверный код при использовании опции `-O2`.

Обычно оптимизацию используют при компиляции окончательной версии.

```
% cc -O -o foobar foobar.c
```

По этой команде будет создана оптимизированная версия программы `foobar`.

Следующие три флага заставят `cc` проверять ваш код на соответствие известному международному стандарту, часто называемому стандартом ANSI, хотя, строго говоря, это стандарт ISO.

`-Wall`

Включить выдачу всех предупреждений, которые посчитают нужным выдать авторы `cc`. Несмотря на свое название, при этом включается выдача не всех предупреждений, на которые способен компилятор `cc`.

`-ansi`

Выключить большинство, если не все, не-ANSI расширений языка C, имеющиеся в `cc`. Несмотря на название, этот параметр не гарантирует, что ваш код будет строго соответствовать стандарту.

`-pedantic`

Выключить *все* расширения C, имеющиеся в компиляторе `cc`, не соответствующие стандарту ANSI.

Без этих флагов компилятор `cc` позволит вам использовать некоторые нестандартные расширения языка. Некоторые из них весьма полезны, но не будут работать при использовании других компиляторов--фактически одним из назначений стандарта является обеспечение возможности писать код, который будет работать с любым компилятором в любой системе. Такой код называется *переносимым кодом*.

Вообще говоря, вы должны стараться писать как можно более переносимый код, иначе вам позже, может быть, придется полностью переписывать программу, чтобы заставить ее работать где-то еще--и кто знает, что вы будете использовать через несколько лет?

```
% cc -Wall -ansi -pedantic -o foobar foobar.c
```

По этой команде после проверки на соответствие стандарту файла `foobar.c` будет сгенерирован выполнимый файл `foobar`.

`-l`*`library`*

Задает библиотеку функций, которая будет использоваться на этапе компоновки.

В качестве самого распространенного примера можно привести компиляцию программы, использующей некоторые математические функции на языке C. В отличие от других платформ, имеется отделенная от стандартной библиотека, и вам нужно указать компилятору, что ее нужно использовать.

При этом если библиотека называется `lib`*`something`*`.a`, вы передаете программе `cc` аргумент `-l`*`something`*. Например, математическая библиотека называется `libm.a`, так что вы передаете программе `cc` аргумент `-lm`. Известной "хитростью" при работе с математической библиотекой является то, что в командной строке она должна быть указана в качестве последней библиотеки.

```
% cc -o foobar foobar.c -lm
```

По этой команде библиотека математических функций будет скомпонована с `foobar`.

Если вы компилируете код на языке C++, вам нужно добавить параметр `-lg++`, или `-lstdc++` при использовании FreeBSD 2.2 и выше, к аргументам командной строки для компоновки библиотеки функций C++. Либо вы можете запустить вместо `cc` компилятор `c++`, который сделает это за вас. Во FreeBSD `c++` может быть вызван как `g++`.

```
% cc -o foobar foobar.cc -lg++      For FreeBSD 2.1.6 and earlier
% cc -o foobar foobar.cc -lstdc++   For FreeBSD 2.2 and later
% c++ -o foobar foobar.cc
```

В каждом из этих вариантов из файла `foobar.cc` с исходным кодом C++ будет создан выполнимый файл `foobar`. Заметьте, что в Unix-системах файлы с исходным кодом C++ традиционно оканчиваются на `.C`, `.cxx` или `.cc`, в отличие от стиля MS-DOS `.cpp` (который уже использовался для чего-то еще). `gcc` использует это для определения того, какой компилятор использовать; однако это ограничение больше не имеет силы, так что вы можете безнаказанно называть ваши файлы с кодом C++ `.cpp`!

Практическое занятие 8.Разработка программы для Unix

Программа печати "HELLO, WORLD" на языке "C" имеет вид:

```
MAIN ()
{
PRINTF("HELLO, WORLD\N");
}
```

Как пропустить эту программу - зависит от используемой вами системы. В частности, на операционной системе "UNIX" вы должны завести исходную программу в файле, имя которого оканчивается на ".C" , например, HELLO.C , и затем скомпили- ровать ее по команде

```
CC HELLO.C
```

Если вы не допустили какой-либо небрежности , такой как пропуск символа или неправильное написание, компиляция прой- дет без сообщений и будет создан исполняемый файл с именем a.OUT . Прогон его по команде

```
A.OUT
```

приведет к выводу

```
HELLO, WORLD
```

Одним способом обмена данными между функциями является передача посредством аргументов. Круглые скобки, следующие за именем функции, заключают в себе список аргументов; здесь maIN - функция без аргументов, что указывается как (). Опе- раторы, составляющие функцию, заключаются в фигурные скобки { и }, которые аналогичны DO-END в PL/1 или BEGIN-END в ал- голе, паскале и т.д. Обращение к функции осуществляется ука- занием ее имени, за которым следует заключенный в круглые скобки список аргументов. здесь нет никаких операторов CALL, как в фортране или PL/1. Круглые скобки должны присутство- вать и в том случае, когда функция не имеет аргументов. Строка

```
PRINTF("HELLO, WORLD\N");
```

является обращением к функции, которое вызывает функцию с именем PRINTF и аргуметом "HELLO, WORLD\N". Функция PRINTF является библиотечной функцией, которая выдает выходные дан- ные на терминал (если только не указано какое-то другое мес- то назначения). В данном случае печатается строка символов, являющаяся аргументом функции.

Последовательность из любого количества символов, зак- люченных в удвоенные кавычки "...", называется 'символьной строкой' или 'строчной константой'. Пока мы будем использо- вать символьные строки только в качестве аргументов для PRINTF и других функций.

Последовательность \N в приведенной строке является обозначением на языке "C" для 'символа новой строки', кото- рый служит указанием для перехода на терминале к левому краю следующей строки. Если вы не включите \N (полезный экспери- мент), то обнаружите, что ваша выдача не закончится перехо- дом терминала на новую строку. Использование последователь- ности \N - единственный способ введения символа новой строки в аргумент функции PRINTF; если вы попробуете что-нибудь вроде

```
PRINTF("HELLO, WORLD
");
```

то "C"-компилятор будет печатать злорадные диагностические сообщения о недостающих кавычках.

Функция PRINTF не обеспечивает автоматического перехода на новую строку, так что многократное обращение к ней можно использовать для поэтапной сборки выходной строки. Наша пер- вая программа, печатающая идентичную выдачу, с точно таким же успехом могла бы быть написана в виде

```
MAIN()
{
PRINTF("HELLO, ");
PRINTF("WORLD");
PRINTF("\N");
```

}

Подчеркнем, что \N представляет только один символ. Ус- ловные 'последовательности', подобные \N , дают общий и до- пускающий расширение механизм для представления трудных для печати или невидимых символов. Среди прочих символов в языке "C" предусмотрены следующие: \т - для табуляции, \B - для возврата на одну позицию, \" - для двойной кавычки и \\ для самой обратной косой черты.

Переменные и арифметика

Следующая программа печатает приведенную ниже таблицу температур по Фаренгейту и их эквивалентов по стоградусной шкале Цельсия, используя для перевода формулу

C = (5/9)*(F-32).
0 -17.8
20 -6.7
40 4.4
60 15.6
... ...
260 126.7
280 137.8
300 140.9

Теперь сама программа:

```
/* PRINT FAHRENHEIT-CELSIUS TABLE
FOR F = 0, 20, ..., 300 */
MAIN()
{
INT LOWER, UPPER, STEP;
FLOAT FAHR, CELSIUS;
LOWER = 0; /* LOWER LIMIT OF TEMPERATURE
TABLE */
UPPER =300; /* UPPER LIMIT */
STEP = 20; /* STEP SIZE */
FAHR = LOWER;
WHILE (FAHR <= UPPER) {
CELSIUS = (5.0/9.0) * (FAHR -32.0);
PRINTF("%4.0F %6.1F\N", FAHR, CELSIUS);
FAHR = FAHR + STEP;
}
}
```

Первые две строки

```
/* PRINT FAHRENHEIT-CELSIUS TABLE
FOR F = 0, 20, ..., 300 */
```

являются комментарием, который в данном случае кратко пояс- няет, что делает программа. Любые символы между /* и */ иг- норируются компилятором; можно

свободно пользоваться коммен- тариями для облегчения понимания программы. Комментарии мо- гут появляться в любом месте, где возможен пробел или пере- ход на новую строку.

В языке "C" все переменные должны быть описаны до их ис- пользования, обычно это делается в начале функции до первого выполняемого оператора. Если вы забудете вставить описание, то получите диагностическое сообщение от компилятора. Описа- ние состоит из типа и списка переменных, имеющих этот тип, как в

```
INT LOWER, UPPER, STEP;
FLOAT FAHR, CELSIUS;
```

Тип INT означает, что все переменные списка целые; тип FLOAT предназначен для чисел с плавающей точкой, т.е. для чисел, которые могут иметь дробную часть. Точность как INT , TAK и FLOAT зависит от конкретной машины, на которой вы ра- ботаете. На PDP-11, например, тип INT соответствует 16-бито- вому числу со знаком, т.е. числу, лежащему между -32768 и +32767. Число типа FLOAT - это 32-битовое число, имеющее около семи значащих цифр и лежащее в диапазоне от 10e-38 до 10e+38. В главе 2 приводится список размеров для других ма- шин.

В языке "C" предусмотрено несколько других основных ти- пов данных, кроме INT и FLOAT: CHAR символ - один байт SHORT короткое целое LONG длинное целое DOUBLE плавающее с двойной точностью

Размеры этих объектов тоже машинно-независимы; детали приведены в главе 2. Имеются также массивы, структуры и об- ъединения этих основных типов, указатели на них и функ- ции,которые их возвращают; со всеми ними мы встретимся в свое время.

Фактически вычисления в программе перевода температур начинаются с операторов присваивания LOWER = 0; UPPER =300; STEP = 20; FAHR =LOWER; которые придают переменным их начальные значения. каждый от- дельный оператор заканчивается точкой с запятой.

Каждая строка таблицы вычисляется одинаковым образом, так что мы используем цикл, повторяющийся один раз на стро- ку. В этом назначение оператора WHILE:
WHILE (FAHR <= UPPER) { .... }

проверяется условие в круглых скобках. Если оно истинно (FAHR меньше или равно UPPER), то выполняется тело цикла (все операторы, заключенные в фигурные скобки { и }). Затем вновь проверяется это условие и, если оно истинно, опять вы- полняется тело цикла. Если же условие не выполняется (FAHR превосходит UPPER), цикл заканчивается и происходит переход к выполнению оператора, следующего за оператором цикла. Так как в настоящей программе нет никаких последующих операто- ров, то выполнение программы завершается.

Тело оператора WHILE может состоять из одного или более операторов, заключенных в фигурные скобки, как в программе перевода температур, или из одного оператора без скобок, как, например, в

```
WHILE (I < J)
I = 2 * I;
```

В обоих случаях операторы, управляемые оператором WHILE, сдвинуты на одну табуляцию, чтобы вы могли с первого взгляда видеть, какие операторы находятся внутри цикла. Такой сдвиг подчеркивает логическую структуру программы. Хотя в языке "C" допускается совершенно произвольное расположение опера- торов в строке, подходящий сдвиг и использование пробелов значительно облегчают чтение программ. Мы рекомендуем писать только один оператор на строке и (обычно) оставлять пробелы вокруг

операторов. Расположение фигурных скобок менее сущес- твенно; мы выбрали один из нескольких популярных стилей. Вы- берите подходящий для вас стиль и затем используйте его пос- ледовательно.

Основная часть работы выполняется в теле цикла. Темпера- тура по Цельсию вычисляется и присваивается переменной CELAIUS оператором

```
CELSIUS = (5.0/9.0) * (FAHR-32.0);
```

причина использования выражения 5.0/9.0 вместо выглядящего проще 5/9 заключается в том, что в языке "C", как и во мно- гих других языках, при делении целых происходит усечение, состоящее в отбрасывании дробной части результата. Таким об- разом, результат операции 5/9 равен нулю, и, конечно, в этом случае все температуры оказались бы равными нулю. Десятичная точка в константе указывает, что она имеет тип с плавающей точкой, так что, как мы и хотели, 5.0/9.0 равно 0.5555... .

Мы также писали 32.0 вместо 32 , несмотря на то, что так как переменная FAHR имеет тип FLOAT , целое 32 автоматически бы преобразовалось к типу FLOAT (в 32.0) перед вычитанием. С точки зрения стиля разумно писать плавающие константы с явной десятичной точкой даже тогда, когда они имеют целые значения; это подчеркивает их плавающую природу для просмат- ривающего программу и обеспечивает то, что компилятор будет смотреть на вещи так же, как и Вы.

Подробные правила о том, в каком случае целые преобразу- ются к типу с плаваюшей точкой, приведены далее. Сейчас же отметим, что присваивание

```
FAHR = LOWER;
```

проверка

```
WHILE (FAHR <= UPPER)
```

работают, как ожидается, - перед выполнением операций целые преобразуются в плавающую форму.

Этот же пример сообщает чуть больше о том, как работает PRINTF. Функция PRINTF фактически является универсальной функцией форматных преобразований, которая будет полностью описана в главе 7. Ее первым аргументом является строка сим- волов, которая должна быть напечатана, причем каждый знак % указывает, куда должен подставляться каждый из остальных ар- гументов /второй, третий, .../ и в какой форме он должен пе- чататься. Например, в операторе

```
PRINTF("%4.0F %6.1F\N", FAHR, CELSIUS);
```

спецификация преобразования %4.0F говорит, что число с пла- вающей точкой должно быть напечатано в поле шириной по край- ней мере в четыре символа без цифр после десятичной точки. спецификация %6.1F описывает другое число, которое должно занимать по крайней мере шесть позиций с одной цифрой после десятичной точки, аналогично спецификациям F6.1 в фортране или F(6,1) в PL/1. Различные части спецификации могут быть опущены: спецификация %6F говорит, что число будет шириной по крайней мере в шесть символов; спецификация %2 требует двух позиций после десятичной точки, но ширина при этом не ограничивается; спецификация %F говорит только о том, что нужно напечатать число с плавающей точкой. Функция PRINTF также распознает следующие спецификации: %D - для десятично- го целого, %o - для восьмеричного числа, %x - для шестнадца- тиричного, %c - для символа, %S - для символьной строки и %% - для самого символа %.

Каждая конструкция с символом % в первом аргументе функ- ции PRINTF сочетается с соответствующим вторым, третьим, и т.д. Аргументами; они должны согласовываться

по числу и ти- пу; в противном случае вы получите бессмысленные результаты.

Между прочим, функция PRINTF не является частью языка "C"; в самом языке "C" не определены операции ввода-вывода. Нет ничего таинственного и в функции PRINTF ; это - просто полезная функция, являющаяся частью стандартной библиотеки подпрограмм, которая обычно доступна "C"-программам. Чтобы сосредоточиться на самом языке, мы не будем подробно оста- навливаться на операциях ввода-вывода до главы 7. В частнос- ти, мы до тех пор отложим форматный ввод. Если вам надо ввести числа - прочитайте описание функции SCANF в главе 7, раздел 7.4. Функция SCANF во многом сходна с PRINTF , но она считывает входные данные, а не печатает выходные.

Оператор FOR

Как и можно было ожидать, имеется множество различных способов написания каждой программы. Давайте рассмотрим та- кой вариант программы перевода температур:

```
MAIN() /* FAHRENHEIT-CELSIUS TABLE */ {
    INT FAHR;
    FOR (FAHR = 0; FAHR <= 300; FAHR = FAHR + 20)
    PRINTF("%4D %6.1F\N", FAHR, (5.0/9.0)*(FAHR-32.0)); }
```

Эта программа выдает те же самые результаты, но выглядит безусловно по-другому. Главное изменение - исключение боль- шинства переменных; осталась только переменная FAHR , причем типа INT (это сделано для того, чтобы продемонстрировать преобразование %D в функции PRINTF). Нижняя и верхняя грани- цы и размер щага появляются только как константы в операторе FOR , который сам является новой конструкцией, а выражение, вычисляющее температуру по цельсию, входит теперь в виде третьего аргумента функции PRINTF , а не в виде отдельного оператора присваивания.

Последнее изменение является примером вполне общего пра- вила языка "C" - в любом контексте, в котором допускается использование значения переменной некоторого типа, вы можете использовать выражение этого типа. Так как третий аргумент функции PRINTF должен иметь значение с плавающей точкой, чтобы соответствовать спецификации %6.1F, то в этом месте может встретиться любое выражение плавающего типа.

Сам оператор FOR - это оператор цикла, обобщающий опера- тор WHILE. Его функционирование должно стать ясным, если вы сравните его с ранее описанным оператором WHILE . Оператор FOR содержит три части, разделяемые точкой с запятой. Первая часть

```
FAHR = 0
```

выполняется один раз перед входом в сам цикл. Вторая часть - проверка, или условие, которое управляет циклом:

```
FAHR <= 300
```

это условие проверяется и, если оно истинно, то выполняется тело цикла (в данном случае только функция PRINTF). Затем выполняется шаг реинициализации FAHR =FAHR + 20 и условие проверяется снова. цикл завершается, когда это ус- ловие становится ложным. Так же, как и в случае оператора WHILE , тело цикла может состоять из одного оператора или из группы операторов, заключенных в фигурные скобки. Инициали- зирующая и реинициализирующая части могут быть любыми от- дельными выражениями.

Выбор между операторами WHILE и FOR произволен и основы- вается на том , что выглядит яснее. Оператор FOR обычно удо- бен для циклов, в которых инициализация и реинициализация логически связаны и каждая задается одним оператором, так как в этом случае запись более компактна, чем при использо- вании оператора WHILE , а операторы

управления циклом сосре- дотачиваются вместе в одном месте.

Символические константы

Последнее замечание, прежде чем мы навсегда оставим программу перевода температур. Прятать "магические числа", такие как 300 и 20, внутрь программы - это неудачная практи- ка; они дают мало информации тем, кто, возможно, должен бу- дет разбираться в этой программе позднее, и их трудно изме- нять систематическим образом. К счастью в языке "С" предус- мотрен способ, позволяющий избежать таких "магических чи- сел". Используя конструкцию #DEFINE , вы можете в начале программы определить символическое имя или символическую константу, которая будет конкретной строкой символов. Впос- ледствии компилятор заменит все не заключенные в кавычки по- явления этого имени на соответствующую строку. Фактически это имя может быть заменено абсолютно произвольным текстом, не обязательно цифрами

```
#DEFINE LOWER 0/* LOWER LIMIT OF TABLE */
#DEFINE UPPER 300 /* UPPER LIMIT */
#DEFINE STEP 20 /* STEP SIZE */
MAIN () /* FAHRENHEIT-CELSIUS TABLE */
{
INT FAHR; FOR (FAHR =LOWER; FAHR <= UPPER; FAHR =FAHR + STEP)
PRINTF("%4D %6.1F\N", FAHR, (5.0/9.0)*(FAHR-32));
}
```

величины LOWER, UPPER и STEP являются константами и поэ- тому они не указываются в описаниях. Символические имена обычно пишут прописными буквами, чтобы их было легко отли- чить от написанных строчными буквами имен переменных. отме- тим, что в конце определения не ставится точка с запятой. Так как подставляется вся строка, следующая за определенным именем, то это привело бы к слишком большому числу точек с запятой в операторе FOR .

Набор полезных программ

Теперь мы собираемся рассмотреть семейство родственных программ, предназначенных для выполнения простых операций над символьными данными. В дальнейшем вы обнаружите, что многие программы являются просто расширенными версиями тех прототипов, которые мы здесь обсуждаем.

Ввод и вывод символов

Стандартная библиотека включает функции для чтения и за- писи по одному символу за один раз. функция GETCHAR() извле- кает следующий вводимый символ каждый раз, как к ней обраща- ются, и возвращает этот символ в качестве своего значения. Это значит, что после

```
C = GETCHAR()
```

переменная 'С' содержит следующий символ из входных данных. Символы обычно поступают с терминала, но это не должно нас касаться до главы 7.

Функция PUTCHAR(C) является дополнением к GETCHAR : в результате обращения

```
PUTCHAR (C)
```

содержимое переменной 'C' выдается на некоторый выходной но- ситель, обычно опять на терминал. Обращение к функциям PUTCHAR и PRINTF могут перемежаться; выдача будет появляться в том порядке, в котором происходят обращения.

Как и функция PRINTF , функции GETCHAR и PUTCHAR не со- держат ничего экстраординарного. Они не входят в состав язы- ка "C", но к ним всегда можно обратиться.

Копирование файла

Имея в своем распоряжении только функции GETCHAR и PUTCHAR вы можете, не зная ничего более об операциях вво- да-вывода, написать удивительное количество полезных прог- рамм. Простейшим примером может служить программа посимволь- ного копирования вводного файла в выводной. Общая схема име- ет вид: ввести символ
WHILE (символ не является признаком конца файла)
вывести только что прочитанный символ
ввести новый символ
программа, написанная на языке "C", выглядит следующим обра- зом:

```
MAIN() /* COPY INPUT TO OUTPUT; 1ST VERSION */
{
INT C;

C = GETCHAR();
WHILE (C != EOF) {
PUTCHAR (C);
C = GETCHAR();
}
}
```

оператор отношения != означает "не равно".

Основная проблема заключается в том, чтобы зафиксиро- вать конец файла ввода. Обычно, когда функция GETCHAR натал- кивается на конец файла ввода, она возвращает значение , не являющееся действительным символом; таким образом, программа может установить, что файл ввода исчерпан. Единственное ос- ложнение, являющееся значительным неудобством, заключается в существовании двух общеупотребительных соглашений о том, ка- кое значение фактически является признаком конца файла. Мы отсрочим решение этого вопроса, использовав символическое имя EOF для этого значения, каким бы оно ни было. На практи- ке EOF будет либо -1, либо 0, так что для правильной работы перед программой должно стоять собственно либо

```
#DEFINE EOF -1
```

либо

```
#DEFINE EOF 0
```

Использовав символическую константу EOF для представле- ния значения, возвращаемого функцией GETCHAR при выходе на конец файла, мы обеспечили, что только одна величина в прог- рамме зависит от конкретного численного значения.

Мы также описали переменную 'C' как INT , а не CHAR , с тем чтобы она могла хранить значение, возвращаемое GETCHAR . как мы увидим в главе 2, эта величина действительно INT, так как она должна быть в состоянии в дополнение ко всем возмож-

ным символам представлять и EOF.

Программистом, имеющим опыт работы на "C", программа копирования была бы написана более сжато. В языке "C" любое присваивание, такое как

C = GETCHAR()

может быть использовано в выражении; его значение - просто значение, присваиваемое левой части. Если присваивание сим- вола переменной 'C' поместить внутрь проверочной части опе- ратора WHILE , то программа копирования файла запишется в виде:

```
MAIN() /* COPY INPUT TO OUTPUT; 2ND VERSION */ { INT C;
WHILE ((C = GETCHAR()) != EOF) PUTCHAR(C); }
```

Программа извлекает символ , присваивает его переменной 'C' и затем проверяет, не является ли этот символ признаком конца файла. Если нет - выполняется тело оператора WHILE, выводящее этот символ. Затем цикл WHILE повторяется. когда, наконец, будет достигнут конец файла ввода, оператор WHILE завершается, а вместе с ним заканчивается выполнение и функ- ции MAIN .

В этой версии централизуется ввод - в программе только одно обращение к функции GETCHAR - и ужимается программа. Вложение присваивания в проверяемое условие - это одно из тех мест языка "C", которое приводит к значительному сокра- щению программ. Однако, на этом пути можно увлечься и начать писать недоступные для понимания программы. Эту тенденцию мы будем пытаться сдерживать.

Важно понять , что круглые скобки вокруг присваивания в условном выражении действительно необходимы. Старшинство операции != выше, чем операции присваивания =, а это означа- ет, что в отсутствие круглых скобок проверка условия != бу- дет выполнена до присваивания =. Таким образом, оператор

```
C = GETCHAR() != EOF
```

эквивалентен оператору

```
C = (GETCHAR() != EOF)
```

Это, вопреки нашему желанию, приведет к тому, что 'C' будет принимать значение 0 или 1 в зависимости от того, на- толкнется или нет GETCHAR на признак конца файла.

Подсчет символов

Следующая программа подсчитывает число символов; она представляет собой небольшое развитие программы копирования.

```
MAIN() /* COUNT CHARACTERS IN INPUT */
{
LONG NC;

NC = 0;
WHILE (GETCHAR() != EOF )
++NC;
PRINTF("%1D\N", NC);
}
```

Оператор

++NC;

демонстрирует новую операцию, ++, которая означает увеличе- ние на единицу. Вы могли

бы написать NC = NC + 1 , но ++NC более кратко и зачастую более эффективно. Имеется соответст- вующая операция -- уменьшение на единицу. Операции ++ и -- могут быть либо префиксными (++NC), либо постфиксными (NC++); эти две формы, как будет показано в главе 2, имеют в выражениях различные значения, но как ++NC, так и NC++ уве- личивают NC. Пока мы будем придерживаться префиксных опера- ций.

Программа подсчета символов накапливает их количество в переменной типа LONG, а не INT . На PDP-11 максимальное зна- чение равно 32767, и если описать счетчик как INT , то он будет переполняться даже при сравнительно малом файле ввода; на языке "C" для HONEYWELL и IBM типы LONG и INT являются синонимами и имеют значительно больший размер. Спецификация преобразования %1D указывает PRINTF , что соответствующий аргумент является целым типа LONG .

Чтобы справиться с еще большими числами, вы можете ис- пользовать тип DOUBLE / FLOAT двойной длины/. мы также ис- пользуем оператор FOR вместо WHILE с тем, чтобы проиллюстри- ровать другой способ записи цикла.

```
MAIN() /* COUNT CHARACTERS IN INPUT */
{
DOUBLE NC;

FOR (NC = 0; GETCHAR() != EOF; ++NC)
;
PRINTF("%.0F\N", NC);
}
```

Функция PRINTF использует спецификацию %F как для FLOAT , так и для DOUBLE ; спецификация %.0F подавляет печать не- существующей дробной части.

Тело оператора цикла FOR здесь пусто, так как вся рабо- та выполняется в проверочной и реинициализационной частях. Но грамматические правила языка "C" требуют, чтобы оператор FOR имел тело. Изолированная точка с запятой, соответствую- шая пустому оператору, появляется здесь, чтобы удовлетворить этому требованию. Мы выделили ее на отдельную строку, чтобы сделать ее более заметной.

Прежде чем мы распростимся с программой подсчета симво- лов, отметим, что если файл ввода не содержит никаких симво- лов, то условие в WHILE или FOR не выполнится при самом пер- вом обращении к GETCHAR , и, следовательно , программа вы- даст нуль, т.е. Правильный ответ. это важное замечание. од- ним из приятных свойств операторов WHILE и FOR является то, что они проверяют условие в начале цикла, т.е. До выполнения тела. Если делать ничего не надо, то ничего не будет сдела- но, даже если это означает, что тело цикла никогда не будет выполняться. программы должны действовать разумно, когда они обращаются с файлами типа "никаких символов". Операторы WHILE и FOR помогают обеспечить правильное поведение прог- рамм при граничных значениях проверяемых условий.

Подсчет строк

Следующая программа подсчитывает количество строк в файле ввода. Предполагается, что строки ввода заканчиваются символом новой строки \N, скрупулезно добавленным к каждой выписанной строке. MAIN() /* COUNT LINES IN INPUT */ {

```
INT C,NL;

NL = 0;
WHILE ((C = GETCHAR()) != EOF)
```

```
    IF (C =='\N')
    ++NL;
    PRINTF("%D\N", NL); }
```

Тело WHILE теперь содержит оператор IF , который в свою очередь управляет оператором увеличения ++NL. Оператор IF проверяет заключенное в круглые скобки условие и, если оно истинно, выполняет следующий за ним оператор /или группу операторов, заключенных в фигурные скобки/. Мы опять исполь- зовали сдвиг вправо, чтобы показать, что чем управляет.

Удвоенный знак равенства == является обозначением в языке "C" для "равно" /аналогично .EQ. В фортране/. Этот символ введен для того, чтобы отличать проверку на равенство от одиночного =, используемого при присваивании. Поскольку в типичных "C" - программах знак присваивания встречается при- мерно в два раза чаще, чем проверка на равенство, то естест- венно, чтобы знак оператора был вполовину короче.

Любой отдельный символ может быть записан внутри оди- ночных кавычек, и при этом ему соответствует значение, рав- ное численному значению этого символа в машинном наборе сим- волов; это называется символьной константой. Так, например, 'A' - символьная константа; ее значение в наборе символов ASCII /американский стандартный код для обмена информацией/ равно 65, внутреннему представлению символа a. Конечно, 'A' предпочтительнее, чем 65: его смысл очевиден и он не зависит от конкретного машинного набора символов.

Условные последовательности, используемые в символьных строках, также занимают законное место среди символьных кон- стант. Так в проверках и арифметических выражениях '\N' представляет значение символа новой строки. Вы должны твердо уяснить, что '\N' - отдельный символ, который в выражениях эквивалентен одиночному целому; с другой стороны "\N" - это символьная строка, которая содержит только один символ. Воп- рос о сопоставлении строк и символов обсуждается в главе 2.

Подсчет слов

Четвертая программа из нашей серии полезных программ подсчитывает количество строк, слов и символов, используя при этом весьма широкое определение, что словом является лю- бая последовательность символов, не содержащая пробелов, та- буляций или новых строк. /Это - упрощенная версия утилиты 'WC' системы 'UNIX'/

```
#DEFINE YES 1 #DEFINE NO 0
MAIN() /* COUNT LINES, WORDS, CHARS IN INPUT */ {
    INT C, NL, NW, INWORD;

    INWORD = NO;
    NL = NW = NC = 0;
    WHILE((C = GETCHAR()) != EOF) {
    ++NC;
    IF (C == '\N')
    ++NL;
    IF (C==' ' \!\! C=='\N' \!\! C=='\T')
    INWORD = NO;
    ELSE IF (INWORD == NO) {
    INWORD = YES;
    ++NW;
    }
```

```
    }
    PRINTF("%D %D %D\N", NL, NW, NC); }
```

Каждый раз, когда программа встречает первый символ слова, она увеличивает счетчик числа слов на единицу. Пере- менная INWORD следит за тем, находится ли программа в насто- ящий момент внутри слова или нет; сначала этой переменной присваивается " не в слове", чему соответствует значение NO. Мы предпочитаем символические константы YES и NO литерным значениям 1 и 0, потому что они делают программу более удоб- ной для чтения. Конечно, в такой крошечной программе, как эта, это не приводит к заметной разнице, но в больших прог- раммах увеличение ясности вполне стоит тех скромных дополни- тельных усилий, которых требует следование этому принципу с самого начала. Вы также обнаружите, что существенные измене- ния гораздо легче вносить в те программы, где числа фигури- руют только в качестве символьных констант.

Строка

```
    NL = NW = NC = 0;
```

полагает все три переменные равными нулю. Это не особый случай, а следствие того обстоятельства, что операто- ру присваивания соответствует некоторое значение и присваи- вания проводятся последовательно справа налево. Таким обра- зом, дело обстоит так, как если бы мы написали

```
    NC = (NL = (NW = 0));
```

операция \!\! Означает OR , так что строка

```
    IF( C==' ' \!\! C=='\N' \!\! C=='\T')
```

говорит "если с - пробел, или с - символ новой строки, или с -табуляция ..."./условная последовательность \T является изображением символа табуляции/.

Имеется соответствующая операция && для AND. Выражения, связанные операциями && или \!\! , Рассматриваются слева на право, и при этом гарантируется, что оценивание выражений будет прекращено, как только станет ясно, является ли все выражение истинным или ложным. Так, если 'C' оказывается пробелом, то нет никакой необходимости проверять, является ли 'C' символом новой строки или табуляции, и такие проверки действительно не делаются. В данном случае это не имеет принципиального значения, но, как мы скоро увидим, в более сложных ситуациях эта особенность языка весьма существенна. Этот пример также демонстрирует оператор ELSE языка "C", который указывает то действие, которое должно выполняться, если условие, содержащееся в операторе IF, окажется ложным. Общая форма такова:

```
IF (выражение)
оператор-1
ELSE оператор-2
```

Выполняется один и только один из двух операторов, свя- занных с конструкцией IF-ELSE. Если выражение истинно, вы- полняется оператор-1; если нет - выполняется оператор-2. Фактически каждый оператор может быть довольно сложным. В программе подсчета слов оператор, следующий за ELSE , явля- ется опертором IF , который

управляет двумя операторами в фигурных скобках.

Массивы

Давайте напишем программу подсчета числа появлений каж- дой цифры, символов пустых промежутков/пробел, табуляции, новая строка/ и всех остальных символов. Конечно, такая за- дача несколько искусственна, но она позволит нам проиллюст- рировать в одной программе сразу несколько аспектов языка "C".

Мы разбили вводимые символы на двенадцать категорий, и нам удобнее использовать массив для хранения числа появлений каждой цифры, а не десять отдельных переменных. Вот один из вариантов программы: MAIN() /* COUNT DIGITS, WHITE SPACE, OTHERS */ {

```
INT C, I, NWHITE, NOTHER;
INT NDIGIT[10];

NWHITE = NOTHER = 0;
FOR (I = 0; I < 10; ++I)
NDIGIT[I] = 0;

WHILE ((C = GETCHAR()) != EOF)
IF (C >= '0' && C <= '9')
++NDIGIT[C-'0'];
ELSE IF(C== ' ' \!\! C== '\N' \!\! C== '\T')
++NWHITE;
ELSE
++NOTHER;

PRINTF("DIGITS =");
FOR (I = 0; I < 10; ++I)
PRINTF(" %D", NDIGIT[I]);
PRINTF("\NWHITE SPACE = %D, OTHER = %D\N",
NWHITE, NOTHER); }
```

Описание

```
INT NDIGIT[10];
```

объявляет, что NDIGIT является массивом из десяти целых. в языке "C" индексы массива всегда начинаются с нуля /а не с 1, как в фортране или PL/1/, так что элементами массива яв- ляются NDIGIT[0], NDIGIT[1],..., NDIGIT[9]. эта особенность отражена в циклах FOR , которые инициализируют и печатают массив.

Индекс может быть любым целым выражением, которое, ко- нечно, может включать целые переменные, такие как I , и це- лые константы.

Эта конкретная программа сильно опирается на свойства символьного представления цифр. Так, например, в программе проверка

```
IF( C >= '0' && C <= '9')...
```

определяет, является ли символ в 'C' цифрой, и если это так, то численное значение этой цифры определяется по формуле / C - '0'/. Такой способ работает только в том случае, если зна- чения символьных констант '0', '1' и т.д. Положительны, рас- положены в порядке возрастания и нет ничего, кроме цифр, между константами '0' и '9'. К счастью, это верно для всех общепринятых наборов символов.

По определению перед проведением арифметических опера- ций, вовлекающих переменные типа CHAR и INT, все они преоб- разуются к типу INT, TAK что в арифметических выражениях пе- ременные типа CHAR по существу идентичны переменным типа INT. Это вполне естественно и удобно; например, C -'0'- это целое выражение со значением между 0 и 9 в соответствии с тем, какой символ от '0' до '9' хранится в 'C', и, следова- тельно, оно является подходящим индексом для массива NDIGIT.

Выяснение вопроса, является ли данный символ цифрой, символом пустого промежутка или чем-либо еще, осуществляется последовательностью операторов

```
IF (C >= '0' && C <= '9')
++NDIGIT[C-'0'];
ELSE IF(C == ' ' \!\! C == '\N' \!\! C == '\T')
++NWHITE;
ELSE
++NOTHER;
```

Конструкция

```
IF (условие)
оператор
ELSE IF (условие)
оператор
ELSE
оператор
```

часто встречаются в программах как средство выражения ситуа- ций, в которых осуществляется выбор одного из нескольких возможных решений.

Программа просто движется сверху вниз до тех пор, пока не удовлетворится какое-нибудь условие; тогда выполняется соответствующий 'оператор', и вся конструкция завершается. /Конечно, 'оператор' может состоять из нескольких операто- ров, заключенных в фигурные скобки/. Если ни одно из условий не удовлетворяется, то выполняется 'оператор', стоящий после заключительного ELSE, если оно присутствует. Если последнеЕ ELSE и соответствующий 'оператор' опущены (как в программе подсчета слов), то никаких действий не производится. Между начальным IF и конечным ELSE может помещаться произвольное количество групп

```
ELSE IF (условие)
оператор
```

С точки зрения стиля целесообразно записывать эту конст- рукцию так, как мы показали, с тем чтобы длинные выражения не залезали за правый край страницы.

Оператор SWITCH (переключатель), который рассматривается в главе 3, представляет другую возможность для записи раз- ветвления на несколько вариантов. этот оператор особенно удобен, когда проверяемое выражение является либо просто не- которым целым, либо символьным выражением, совпадающим с од- ной из некоторого набора констант. Версия этой программы, использующая оператор SWITCH, будет для сравнения приведена в главе 3.

Функции

В языке "C" функции эквивалентны подпрограммам или функ- циям в фортране или процедурам в PL/1, паскале и т.д. Функ- ции дают удобный способ заключения некоторой части вычисле- ний в черный ящик, который в дальнейшем можно использовать, не

интересуясь его внутренним содержанием. Использование функций является фактически единственным способом справиться с потенциальной сложностью больших программ. Если функции организованы должным образом, то можно игнорировать то, как делается работа; достаточно знание того, что делается. Язык "C" разработан таким образом, чтобы сделать использование функций легким, удобным и эффективным. Вам будут часто встречаться функции длиной всего в несколько строчек, вызывае- мые только один раз, и они используются только потому, что это проясняет некоторую часть программы.

До сих пор мы использовали только предоставленные нам функции типа PRINTF, GETCHAR и PUTCHAR; теперь пора написать несколько наших собственных. так как в "C" нет операции воз- ведения в степень, подобной операции ** в фортране или PL/1, давайте проиллюстрируем механику определения функции на при- мере функции POWER(M,N), возводящей целое м в целую положи- тельную степень N. Так значение POWER(2,5) равно 32. Конеч- но, эта функция не выполняет всей работы операции **, пос- кольку она действует только с положительными степенями не- больших чисел, но лучше не создавать дополнительных затруд- нений, смешивая несколько различных вопросов.

Ниже приводится функция POWER и использующая ее основная программа, так что вы можете видеть целиком всю структуру.

```
MAIN() /* TEST POWER FUNCTION */
{
INT I;

FOR(I = 0; I < 10; ++I)
PRINTF("%D %D %D\N",I,POWER(2,I),POWER(-3,I));
}

POWER(X,N) /* RAISE X N-TH POWER; N > 0 */
INT X,N;
{
INT I, P;
P = 1;
FOR (I =1; I <= N; ++I)
P = P * X;
RETURN (P);
}
```

Все функции имеют одинаковый вид:

```
имя (список аргументов, если они имеются)
описание аргументов, если они имеются
{
описания
операторы
}
```

Эти функции могут быть записаны в любом порядке и нахо- диться в одном или двух исходных файлах. Конечно, если ис- ходная программа размещается в двух файлах, вам придется дать больше указаний при компиляции и загрузке, чем если бы она находилась в одном, но это дело операционной системы, а не атрибут языка. В данный момент, для того чтобы все полу- ченные сведения о прогоне "C"- программ, не изменились в дальнейшем, мы будем предполагать, что обе функции находятся в одном и том же файле.

Функция POWER вызывается дважды в строке

PRINTF("%D %D %D\N",I,POWER(2,I),POWER(-3,I));
при каждом обращении функция POWER, получив два аргумента, вазвращает целое значение, которое печатается в заданном формате. В выражениях POWER(2,I) является точно таким же це- лым, как 2 и I. /Не все функции выдают целое значение; мы займемся этим вопросом в главе 4/.

Аргументы функции POWER должны быть описаны соответству- ющим образом, так как их типы известны. Это сделано в строке
INT X,N;
которая следует за именем функции.

Описания аргументов помещаются между списком аргументов и открывающейся левой фигурной скобкой; каждое описание за- канчивается точкой с запятой. Имена, использованные для ар- гументов функции POWER, являются чисто локальными и недос- тупны никаким другим функциям: другие процедуры могут ис- пользовать те же самые имена без возникновения конфликта. Это верно и для переменных I и P; I в функции POWER никак не связано с I в функции MAIN.

Значение, вычисленное функцией POWER, передаются в MAIN с помощью оператора RETURN, точно такого же, как в PL/1. внутри круглых скобок можно написать любое выражение. Функ- ция не обязана возвращать какое-либо значение; оператор RETURN, не содержащий никакого выражения, приводит к такой же передаче управления, как "сваливание на конец" функции при достижении конечной правой фигурной скобки, но при этом в вызывающую функцию не возвращается никакого полезного зна- чения.

Аргументы - вызов по значению

Один аспект в "C" может оказаться непривычным для прог- раммистов, которые использовали другие языки, в частности, фортран и PL/1. в языке "C" все аргументы функций передаются "по значению". это означает, что вызванная функция получает значения своих аргументов с помощью временных переменных /фактически через стек/, а не их адреса. Это приводит к не- которым особенностям, отличным от тех, с которыми мы сталки- вались в языках типа фортрана и PL/1, использующих "вызов по ссылке ", где вызванная процедура работает с адресом аргу- мента, а не с его значением.

Главное отличие состоит в том, что в "C" вызванная функ- ция не может изменить переменную из вызывающей функции; она может менять только свою собственную временную копию.

Вызов по значению, однако, не помеха, а весьма ценное качество. Оно обычно приводит к более компактным программам, содержащим меньше не относящихся к делу переменных, потому что с аргументами можно обращаться как с удобно инициализи- рованными локальными перемнными вызванной процедуры. Вот, например, вариант функции POWER использующей это обстоятель- ство

```
POWER(X,N) /* RAISE X N-TH POWER; N > 0;
VERSION 2 */
INT X,N;
{
INT P;

FOR (P = 1; N > 0; --N)
P = P * X;
RETURN (P);
}
```

Аргумент N используется как временная переменная; из не- го вычитается единица до тех пор, пока он не станет нулем. Переменная I здесь больше не нужна. чтобы ни происходило с N внутри POWER это никак не влияет на аргумент, с которым первоначально обратились к функции POWER.

При необходимости все же можно добиться, чтобы функция изменила переменную из вызывающей программы. Эта программа должна обеспечить установление адреса переменной /техничес- ки, через указатель на переменную/, а в вызываемой функции надо описать соответствующий аргумент как указатель и ссы- латься к фактической переменной косвенно через него. Мы рас- смотрим это подробно в главе 5.

Когда в качестве аргумента выступает имя массива, то фактическим значением, передаваемым функции, является адрес начала массива. /Здесь нет никакого копирования элементов массива/. С помощью индексации и адреса начала функция может найти и изменить любой элемент массива. Это - тема следующе- го раздела.

Массивы символов

По-видимому самым общим типом массива в "C" является массив символов. Чтобы проиллюстрировать использование мас- сивов символов и обрабатывающих их функций, давайте напишем программу, которая читает набор строк и печатает самую длин- ную из них. Основная схема программы достаточно проста:

```
WHILE (имеется еще строка)
    IF (эта строка длиннее самой длинной из
    предыдущих)
    запомнить эту строку и ее длину напечатать самую длинную строку
```

По этой схеме ясно, что программа естественным образом распадается на несколько частей. Одна часть читает новую строку, другая проверяет ее, третья запоминает, а остальные части программы управляют этим процессом.

Поскольку все так прекрасно делится, было бы хорошо и написать программу соответсвующим образом. Давайте сначала напишем отдельную функцию GETLINE, которая будет извлекать следующую строку из файла ввода; это - обобщение функции GETCHAR. мы попытаемся сделать эту функцию по возможности более гибкой, чтобы она была полезной и в других ситуациях. Как минимум GETLINE должна передавать сигнал о возможном по- явлении конца файла; более общий полезный вариант мог бы передавать длину строки или нуль, если встретится конец файла. нуль не может быть длиной строки, так как каждая строка со- держит по крайней мере один символ; даже строка, содержащая только символ новой строки, имеет длину 1.

Когда мы находим строку, которая длиннее самой длинной из предыдущих, то ее надо где-то запомнить. Это наводит на мысль о другой функции, COPY , которая будет копировать но- вую строку в место хранения.

Наконец, нам нужна основная программа для управления функциями GETLINE и COPY . Вот результат :

```
#DEFINE MAXLINE 1000 /* MAXIMUM INPUT
    LINE SIZE */
    MAIN() /* FIND LONGEST LINE */
    {
    INT LEN; /* CURRENT LINE LENGTH */
    INT MAX; /* MAXIMUM LENGTH SEEN SO FAR */
    CHAR LINE[MAXLINE]; /* CURRENT INPUT LINE */
    CHAR SAVE[MAXLINE]; /* LONGEST LINE, SAVED */

    MAX = 0;
    WHILE ((LEN = GETLINE(LINE, MAXLINE)) > 0)
```

```
    IF (LEN > MAX) {
    MAX = LEN;
    COPY(LINE, SAVE);
    }
    IF (MAX > 0) /* THERE WAS A LINE */
    PRINTF("%S", SAVE);
    }
    GETLINE(S,LIM) /* GET LINE INTO S,RETURN LENGTH */
    CHAR S[];
    INT LIM;
    {
    INT C, I;

    FOR(I=0;I<LIM-1 && (C=GETCHAR())!=EOF && C!='\N';++I)
    S[I] = C;
    IF (C == '\N') {
    S[I] = C;
    ++I;
    }
    S[I] = '\0';
    RETURN(I);
    }

    COPY(S1, S2) /* COPY S1 TO S2;
    ASSUME S2 BIG ENOUGH */
    CHAR S1[], S2[];
    {
    INT I;

    I = 0;
    WHILE ((S2[I] = S1[I] != '\0')
    ++I;
    }
```

Функция MAIN и GETLINE общаются как через пару аргумен- тов, так и через возвращаемое значение. аргументы GETLINE описаны в строках

```
    CHAR S[];
    INT LIM;
```

которые указывают, что первый аргумент является массивом, а второй - целым.

Длина массива S не указана, так как она определена в MAIN . функция GETLINE использует оператор RETURN для пере- дачи значения назад в вызывающую программу точно так же, как это делала функция POWER. Одни функции возвращают некоторое нужное значение; другие, подобно COPY, используются из-за их действия и не возвращают никакого значения.

Чтобы пометить конец строки символов, функция GETLINE помещает в конец создаваемого ей массива символ \0 /нулевой символ, значение которого равно нулю/. Это соглашение ис- пользуется также компилятором с языка "C": когда в "C" - программе встречается строчная константа типа

```
    "HELLO\N"
```

то компилятор создает массив символов, содержащий символы этой строки, и заканчивает его символом \0, с тем чтобы фун- кции, подобные PRINTF, могли зафиксировать конец массива:

! H ! E ! L ! L ! O ! \N ! \0 !

Спецификация формата %S указывает, что PRINTF ожидает стро- ку, представленную в такой форме. Проанализировав функцию COPY, вы обнаружите, что и она опирается на тот факт, что ее входной аргумент оканчивается символом \0, и копирует этот символ в выходной аргумент S2. /Все это подразумевает, что символ \0 не является частью нормального текста/.

Между прочим, стоит отметить, что даже в такой маленькой программе, как эта, возникает несколько неприятных организа- ционных проблем. Например, что должна делать MAIN, если она встретит строку, превышающую ее максимально возможный раз- мер? Функция GETLINE поступает разумно: при заполнении мас- сива она прекращает дальнейшее извлечение символов, даже ес- ли не встречает символа новой строки. Проверив полученную длину и последний символ, функция MAIN может установить, не была ли эта строка слишком длинной, и поступить затем, как она сочтет нужным. Ради краткости мы опустили эту проблему.

Пользователь функции GETLINE никак не может заранее уз- нать, насколько длинной окажется вводимая строка. Поэтому в GETLINE включен контроль переполнения. в то же время пользо- ватель функции COPY уже знает /или может узнать/, каков раз- мер строк, так что мы предпочли не включать в эту функцию дополнительный контроль.

Область действия: внешние переменные

Переменные в MAIN(LINE, SAVE и т.д.) являются внутренни- ми или локальными по отношению к функции MAIN, потому что они описаны внутри MAIN и никакая другая функция не имеет к ним прямого доступа. Это же верно и относительно переменных в других функциях; например, переменная I в функции GETLINE никак не связана с I в COPY. Каждая локальная переменная су- ществует только тогда, когда произошло обращение к соответс- твующей функции, и исчезает, как только закончится выполне- ние этой функции. По этой причине такие переменные, следуя терминологии других языков, обычно называют автоматическими. Мы впредь будем использовать термин автоматические при ссыл- ке на эти динамические локальные переменные. /в главе 4 об- суждается класс статической памяти, когда локальные перемен- ные все же оказываются в состоянии сохранить свои значения между обращениями к функциям/.

Поскольку автоматические переменные появляются и исчеза- ют вместе с обращением к функции, они не сохраняют своих значений в промежутке от одного вызова до другого, в силу чего им при каждом входе нужно явно присваивать значения. Если этого не сделать, то они будут содержать мусор.

В качестве альтернативы к автоматическим переменным мож- но определить переменные, которые будут внешними для всех функций, т.е. Глобальными переменными, к которым может обра- титься по имени любая функция, которая пожелает это сделать. (этот механизм весьма сходен с "COMMON" в фортране и "EXTERNAL" в PL/1). Так как внешние переменные доступны всю- ду, их можно использовать вместо списка аргументов для пере- дачи данных между функциями. Кроме того, поскольку внешние переменные существуют постоянно, а не появляются и исчезают вместе с вызываемыми функциями, они сохраняют свои значения и после того, как функции, присвоившие им эти значения, за- вершат свою работу.

Внешняя переменная должна быть определена вне всех функ- ций; при этом ей выделяется фактическое место в памяти. Та- кая переменная должна быть также описана в каждой функции, которая собирается ее использовать; это можно сделать либо явным описанием EXTERN, либо неявным по контексту. Чтобы сделать обсуждение более конкретным, давайте перепишем прог- рамму поиска самой длинной строки, сделав LINE, SAVE и MAX внешними переменными. Это потребует изменения описаний и тел всех трех функций, а также обращений к ним.

```
#DEFINE MAXLINE 1000 /* MAX. INPUT LINE SIZE*/
CHAR LINE[MAXLINE]; /* INPUT LINE */ CHAR SAVE[MAXLINE];/* LONGEST LINE SAVED HERE*/ INT MAX;/*LENGTH OF LONGEST LINE SEEN SO FAR*/ MAIN()
/*FIND LONGEST LINE; SPECIALIZED VERSION*/ {
    INT LEN;
    EXTERN INT MAX;
    EXTERN CHAR SAVE[];
    MAX = 0;
WHILE ( (LEN = GETLINE()) > 0 )
    IF ( LEN > MAX ) {
    MAX = LEN;
    COPY();
    } IF ( MAX > 0 ) /* THERE WAS A LINE */
    PRINTF( "%S", SAVE ); }
GETLINE() /* SPECIALIZED VERSION */ {
    INT C, I;
    EXTERN CHAR LINE[];

    FOR (I = 0; I < MAXLINE-1


    && (C=GETCHAR()) !=EOF && C!='\N'; ++I)
    LINE[I] = C;
    ++I;
    }
    LINE[I] = '\0'
    RETURN(I)
    }
    COPY() /* SPECIALIZED VERSION */
    {
    INT I;
    EXTERN CHAR LINE[], SAVE[];

    I = 0;
    WHILE ((SAVE[I] = LINE[I]) !='\0')
    ++I;
    }
```

Внешние переменные для функций MAIN, GETLINE и COPY оп- ределены в первых строчках приведенного выше примера, кото- рыми указывается их тип и вызывается отведение для них памя- ти. синтаксически внешние описания точно такие же, как опи- сания, которые мы использовали ранее, но так как они распо- ложены вне функций, соответствующие переменные являются внешними. Чтобы функция могла использовать внешнюю переме- ную, ей надо сообщить ее имя. Один способ сделать это - включить в функцию описание EXTERN; это описание отличается от предыдущих только

добавлением ключевого слова EXTERN.

В определенных ситуациях описание EXTERN может быть опу- щено: если внешнее определение переменной находится в том же исходном файле, раньше ее использования в некоторой конкрет- ной функции, то не обязательно включать описание EXTERN для этой переменной в саму функцию. Описания EXTERN в функциях MAIN, GETLINE и COPY являются, таким образом, излишними. Фактически, обычная практика заключается в помещении опреде- лений всех внешних переменных в начале исходного файла и последующем опускании всех описаний EXTERN.

Если программа находится в нескольких исходных файлах, и некоторая переменная определена, скажем в файле 1, а исполь- зуется в файле 2, то чтобы связать эти два вхождения пере- менной, необходимо в файле 2 использовать описание EXTERN. Этот вопрос подробно обсуждается в главе 4.

Вы должно быть заметили, что мы в этом разделе при ссыл- ке на внешние переменные очень аккуратно используем слова описание и определение. "Определение" относится к тому мес- ту, где переменная фактически заводится и ей выделяется па- мять; "описание" относится к тем местам, где указывается природа переменной, но никакой памяти не отводится.

Между прочим, существует тенденция объявлять все, что ни попадется, внешними переменными, поскольку кажется, что это упрощает связи, - списки аргументов становятся короче и пе- ременные всегда присутствуют, когда бы вам они ни понадоби- лись. Но внешние переменные присутствуют и тогда, когда вы в них не нуждаетесь. Такой стиль программирования чреват опас- ностью, так как он приводит к программам, связи данных внут- ри которых не вполне очевидны. Переменные при этом могут из- меняться неожиданным и даже неумышленным образом, а програм- мы становится трудно модифицировать, когда возникает такая необходимость. Вторая версия программы поиска самой длинной строки уступает первой отчасти по этим причинам, а отчасти потому, что она лишила универсальности две весьма полезные функции, введя в них имена переменных, с которыми они будут манипулировать.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

Целью лабораторных работ в процессе изучения дисциплины является:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать справочную и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений.

Объем времени, отведенный на лабораторные занятия, определяется в соответствии с учебным планом специальности и рабочей программой учебной дисциплины.

Критерием оценки результатов работы студента на лабораторных работах являются:

- умение студента использовать теоретические знания при выполнении практических задач;
- обоснованность и четкость изложения ответа;
- оформление материала в соответствии с требованиями.

Перечень лабораторных работ:

Лабораторная работа № 1. Установка ОС Windows на виртуальную машину.

Лабораторная работа № 2. Настройка виртуального аппаратного обеспечения для ОС Windows.

Лабораторная работа № 3. Настройка сетевых подключений в ОС Windows

Лабораторная работа № 4. Администрирование пользователей ОС Windows.

Лабораторная работа № 5. Установка и настройка программного обеспечения ОС Windows.

Лабораторная работа № 6. Восстановление ОС Windows после сбоя

Лабораторная работа № 7. Командная строка ОС Windows

Лабораторная работа № 8. Командные файлы ОС Windows

Лабораторная работа № 9. ISS сервер на ОС Windows

Лабораторная работа №1. Установка и настройка виртуальной машины

Рассмотрим создание виртуальной машины и начальную настройку VirtualBox для установки операционной системы. В качестве примера для установки была взята ОС Ubuntu, образ которой предварительно записан на DVD диск. Отмечу устанавливать операционные системы на виртуальный компьютер можно и с загрузочного flash накопителя, и виртуального привода, предварительно смонтировав образ системы.

После установки и запуска VirtualBox откроется окно программы (см. рис. 1). Для того что бы создать виртуальную машину необходимо нажать на кнопку **Создать**.

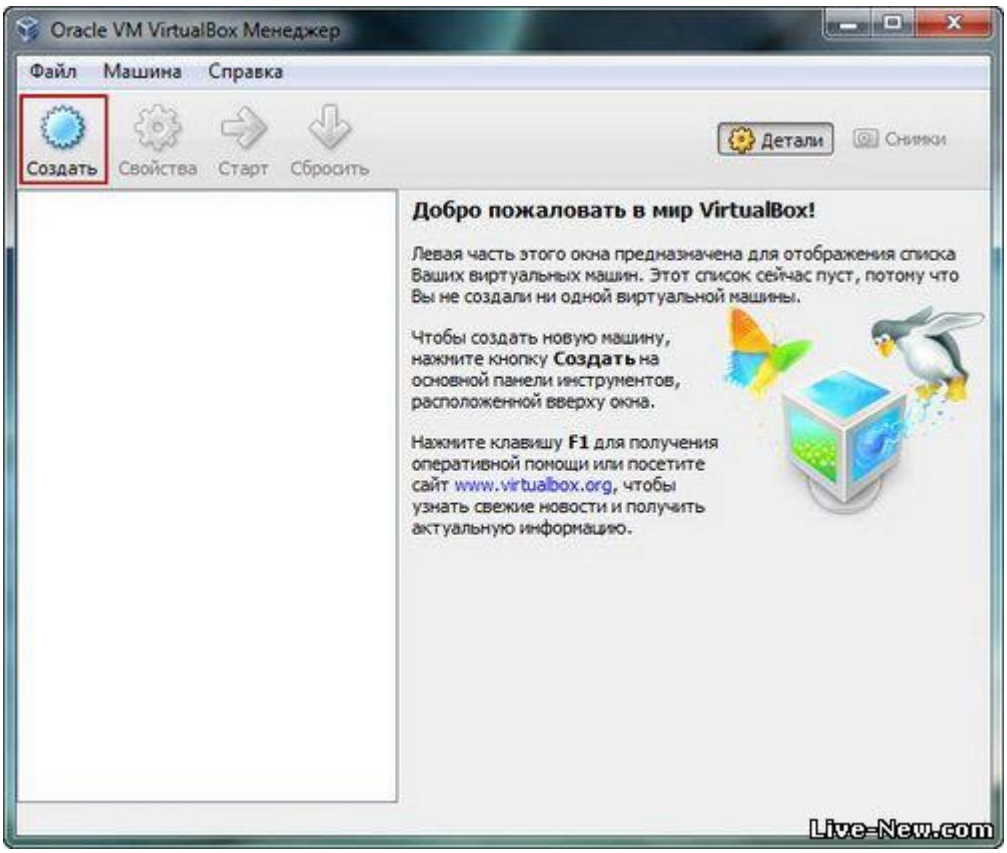


Рисунок 1. Создание виртуальной машины

После чего откроется окно мастера создания новой виртуальной машины (см. рис. 2), с помощью которого вы сможете создать виртуальную машину, нажимаете кнопку **Next**.

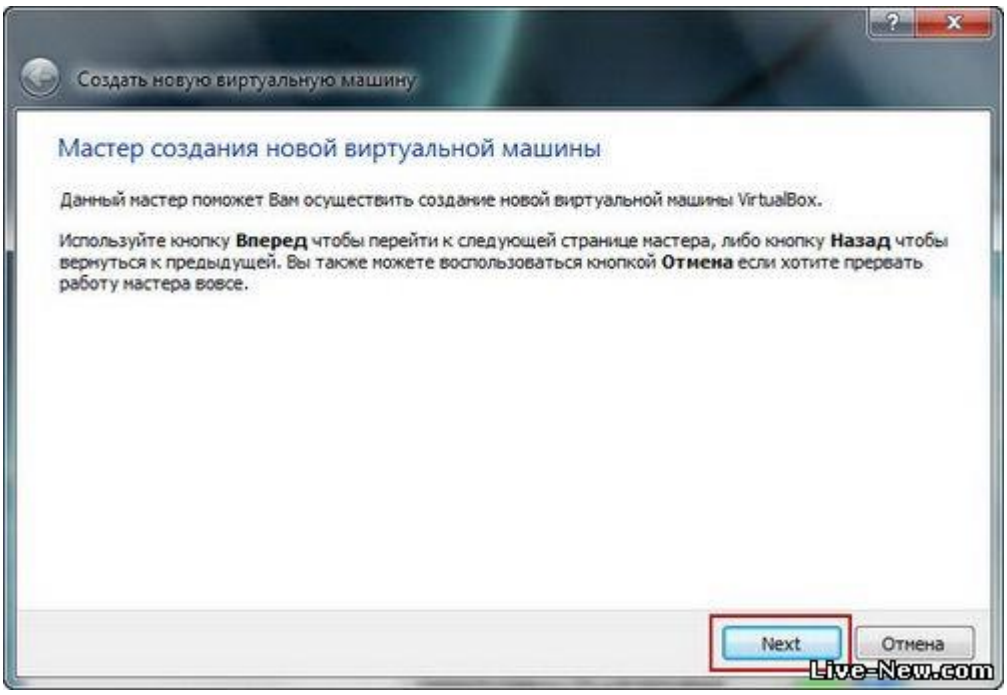


Рисунок 2. Создание виртуальной машины

В следующем окне (см. рис. 3), в поле **Имя** ввести имя виртуальной машины, например: Ubuntu.

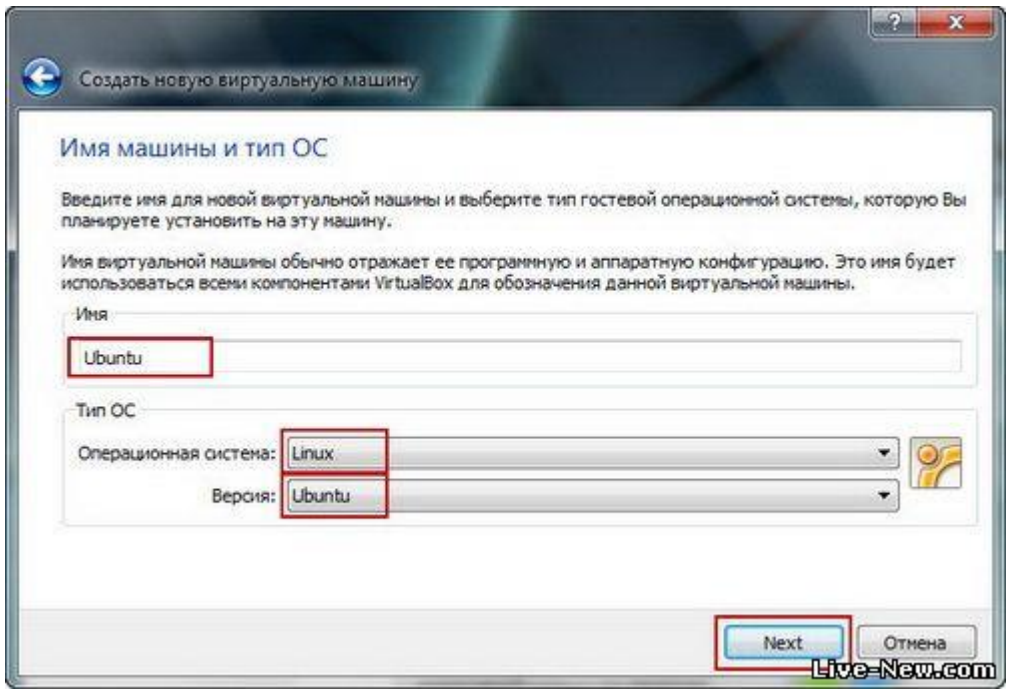


Рисунок 3. Имя машины и тип ОС

В поле **Тип ОС**, выбрать устанавливаемую операционную систему (Linux) и версию (Ubuntu или Ubuntu 64 bit), нажимаете кнопку **Next**.
Отмечу, при вводе имени машины (например: Ubuntu, Linux, Windows и т.д.), поля операционная система и версия заполняются автоматически, если этого не произошло, выполните действия описанные выше. В случае названия машины Linux, Windows и т.д. нужно будет вручную выбрать версию устанавливаемой ОС, например для Windows: Windows XP, Windows 7 и т.д.

В следующем окне (см. рис.4) мастер нам предложит выделить количество оперативной памяти для нашей виртуальной ОС . Отмечу, что системные параметры выделяемые для виртуальной машины будут браться с железа вашего физического ПК и вам следует обратить внимание на количество выделяемых оперативной, видео памяти и других системных ресурсов.

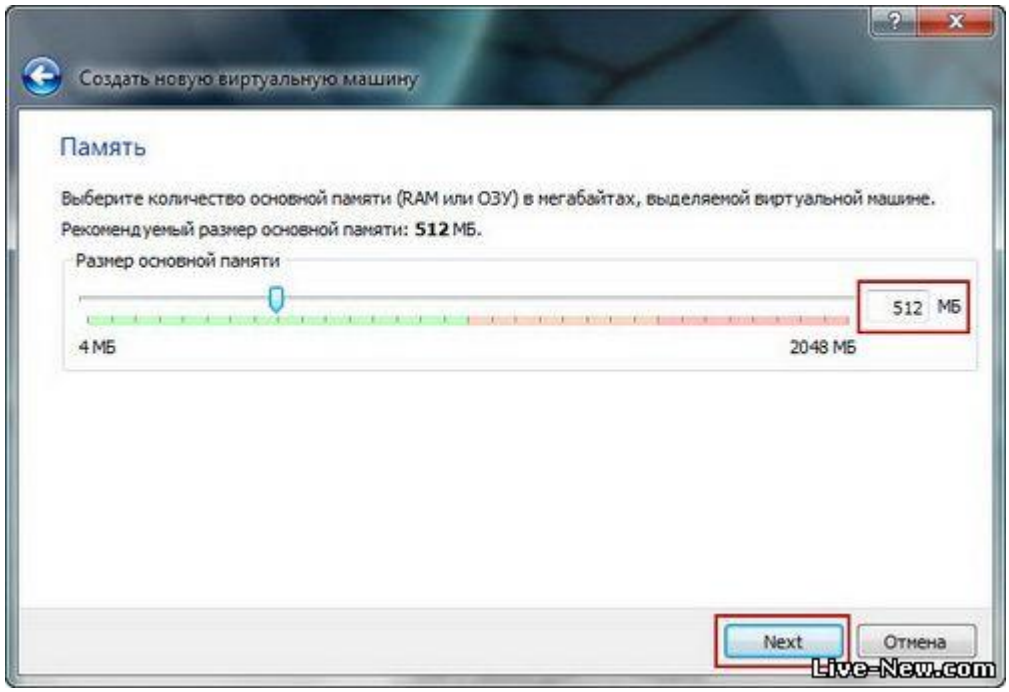


Рисунок 4. Выделение памяти

Исходя из того, что на компьютере физической RAM 2 ГБ и минимальных системных требований для работы Ubuntu, оставим рекомендуемые мастером 512 МБ, которых для знакомства с системой будет достаточно. Вы можете выделить и больше на свое усмотрение, исходя из возможностей вашей основной системы, и для каких целей устанавливается виртуальная ОС, нажимаете кнопку **Next**.

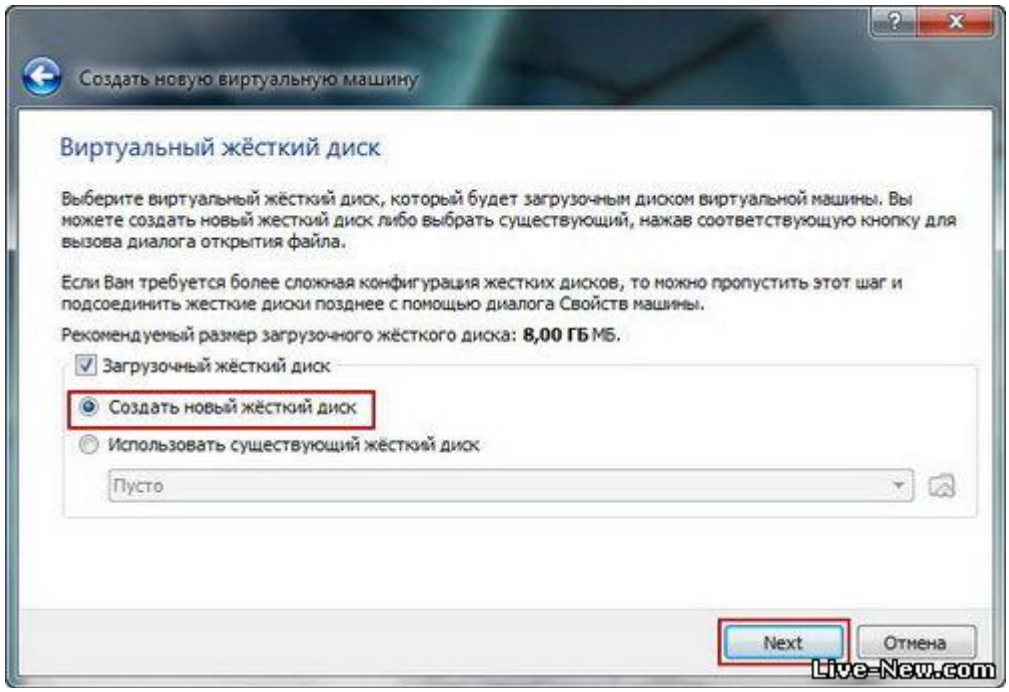


Рисунок 5. Создание виртуального жесткого диска

Далее создаете виртуальный жесткий диск (см. рис.5) оставляете все без изменений и нажимаете кнопку **Next**, в следующем окне вам будет предложено выбрать тип файла для создания виртуального диска (см. рис. 6).

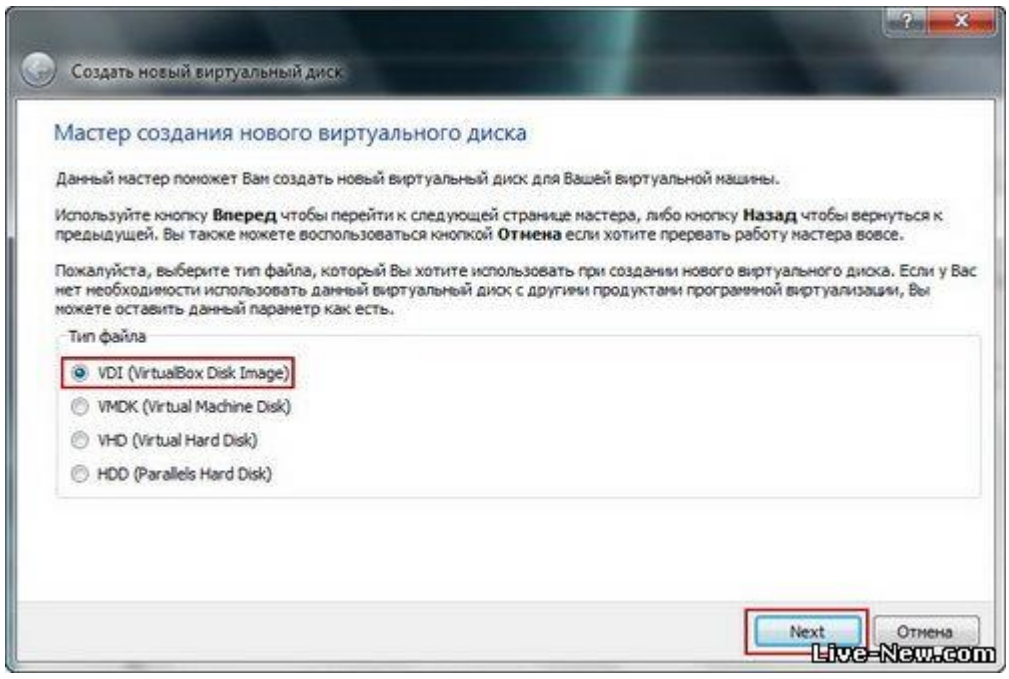


Рисунок 6. Выбор типа файла

Если вы не будете использовать создаваемый вами виртуальный диск с другими программами для создания виртуальных машин, в параметре **Тип файла** оставляете по умолчанию **VDI** (VirtualBox Disk Image), если есть необходимость использовать созданный диск в других приложениях создаете диск с нужным вам типом файла в зависимости от того в какой программе вы его будете использовать, с выбором определились, нажимаете **Next**.

После чего вам будет предложено выбрать **Дополнительные атрибуты** виртуального диска (см. рис. 7), такие как **Динамический** виртуальный диск или **Фиксированный** виртуальный диск, выбираете, как вам удобнее руководствуясь подсказками мастера, нажимаете кнопку **Next**.

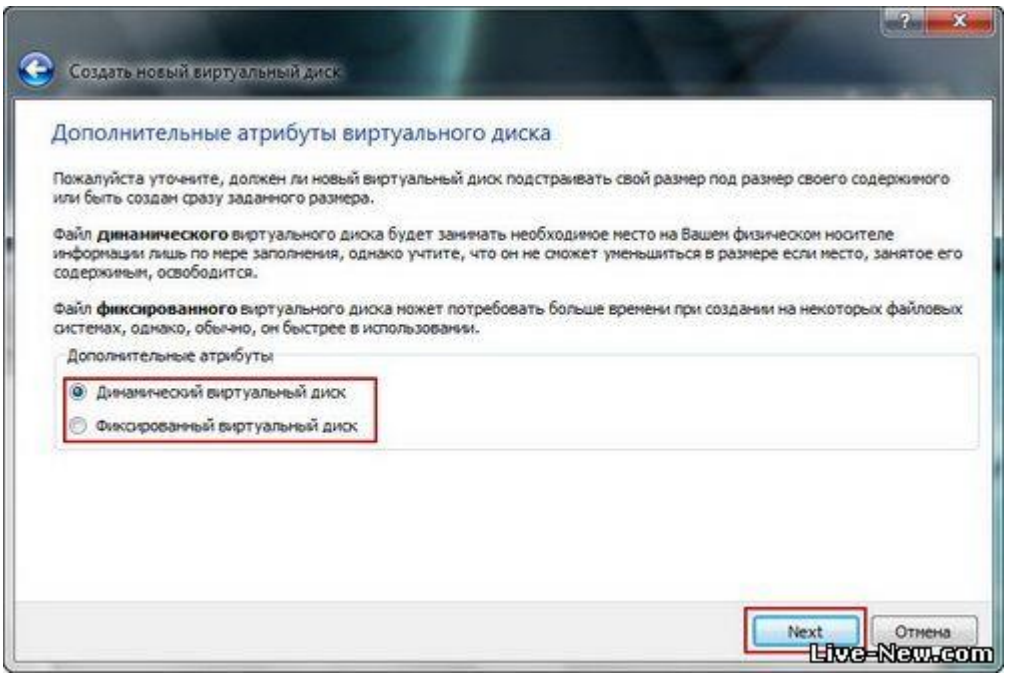


Рисунок 7. Выбор Дополнительных атрибутов

И в следующем открывшемся окне (см. рис. 8) завершаете создание виртуального диска, нажимаете кнопку **Создать** после чего будет запущен процесс создания виртуального диска (см. рис. 9).

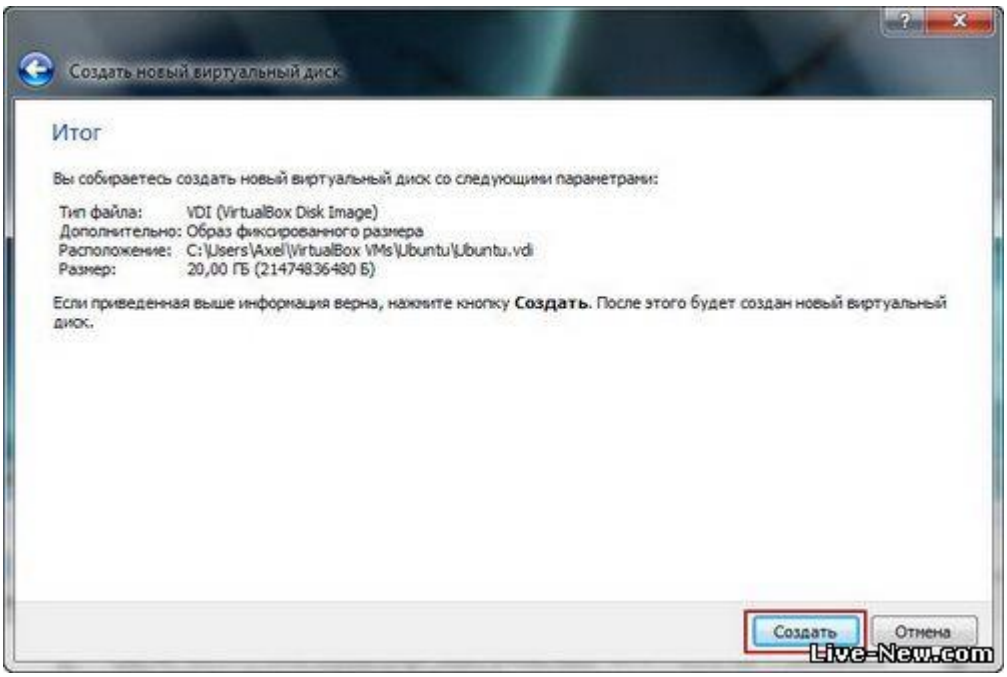


Рисунок 8. Завершение создания виртуального диска

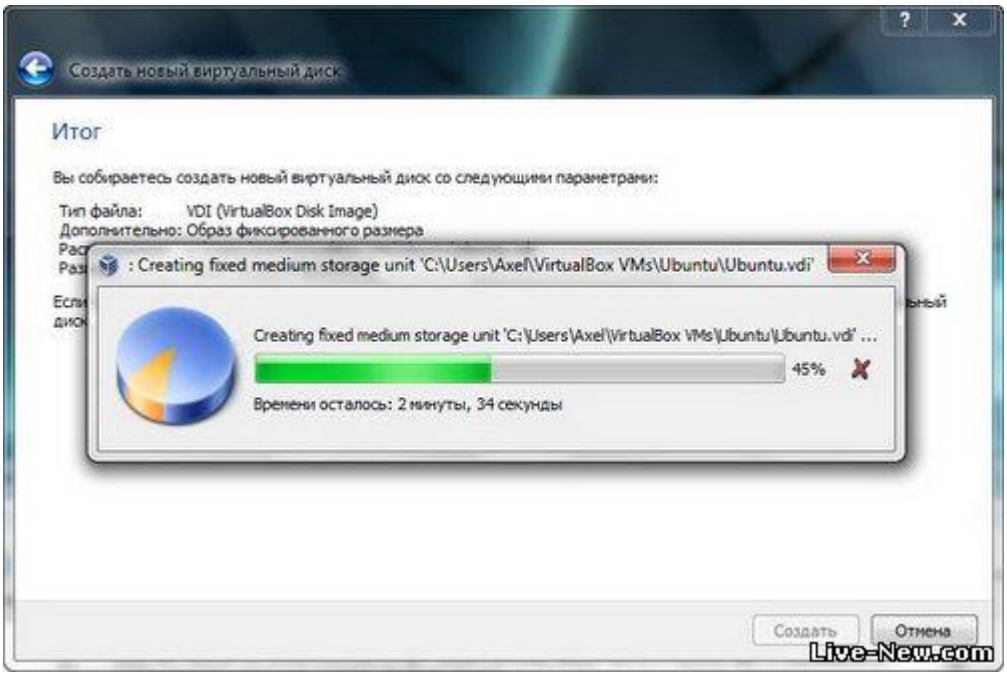


Рисунок 9. Создания виртуального диска

После создания виртуального диска вам будет предложено создать виртуальную машину (см. рис. 10), жмите кнопку **Create**.

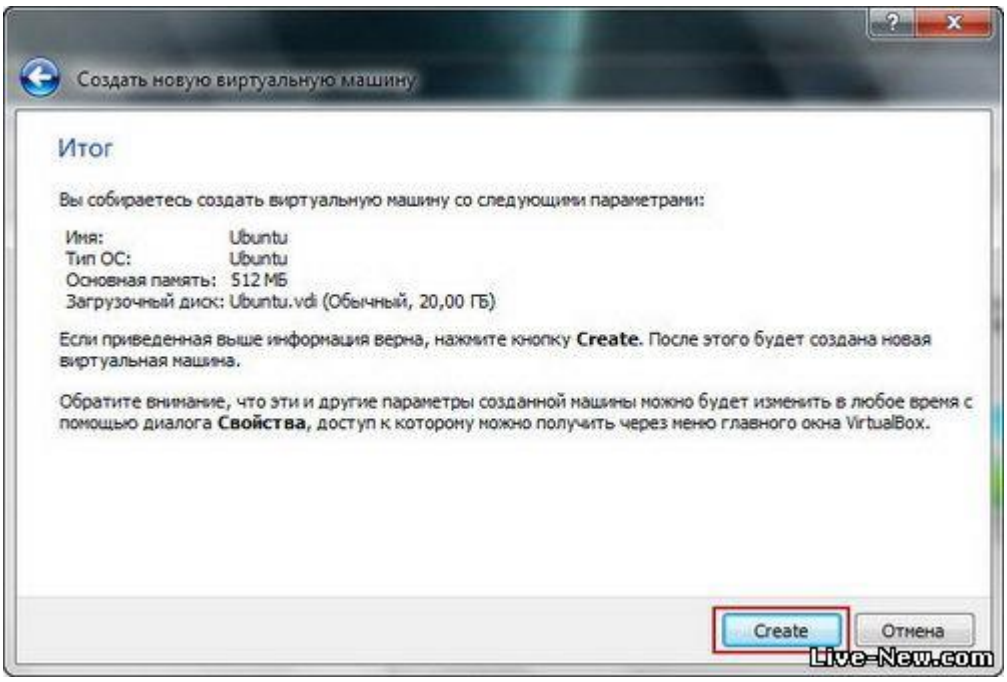


Рисунок 10. Создание виртуальной машины

Перед началом установки операционной системы, в случае необходимости можно изменить системные настройки, такие как: объем оперативной памяти, выделить объем видео памяти, настроить порядок загрузки и другое.

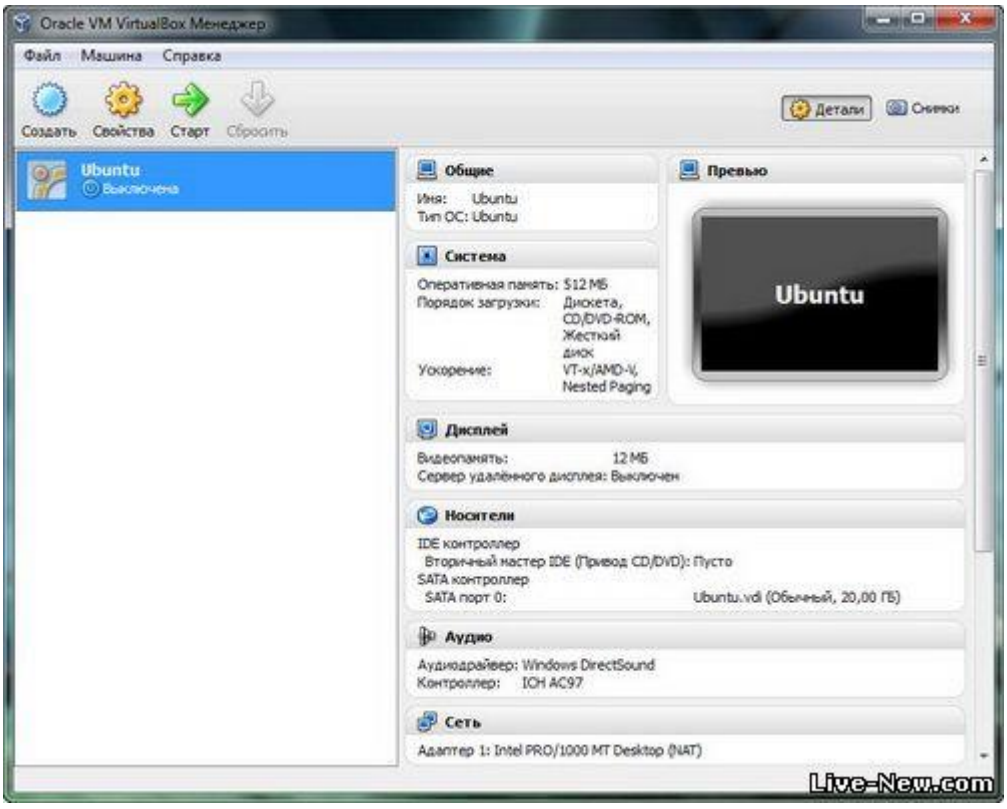


Рисунок 11. Окно менеджера VirtualBox

Для того что бы изменить настройки виртуальной машины, в окне менеджера VirtualBox, нажмите кнопку **Свойства** и в открывшемся окне (см. рис. 12, 13) , вы сможете изменить настройки тех или

иных системных устройств.

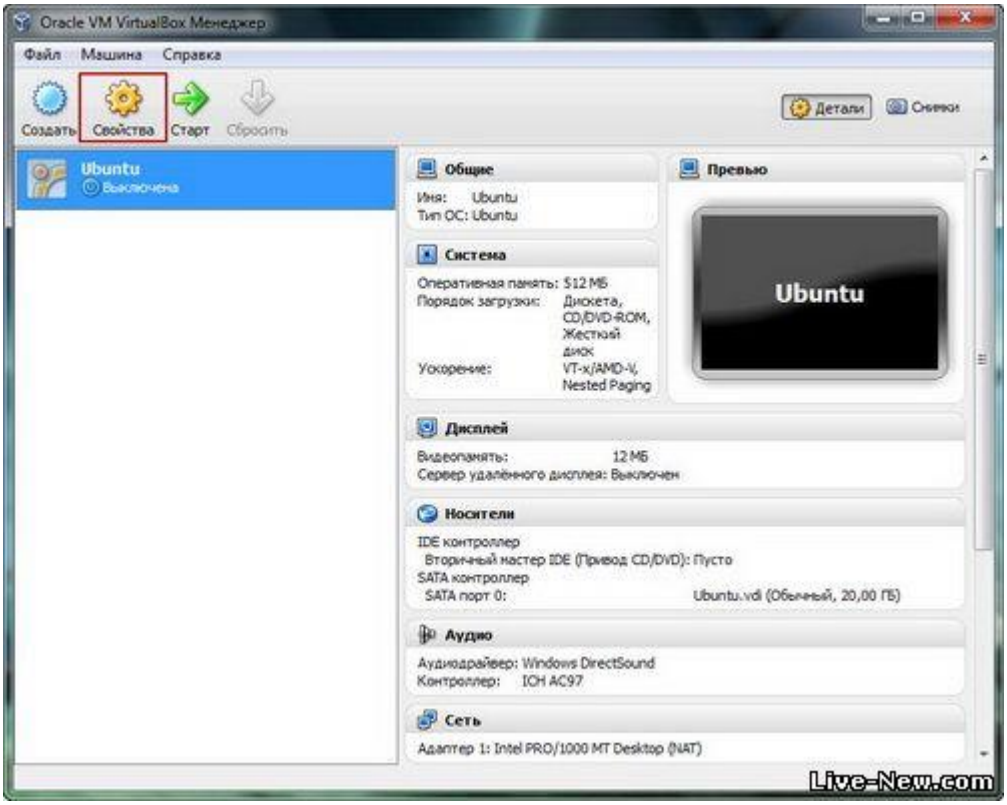


Рисунок 12. Изменение настроек виртуальной машины

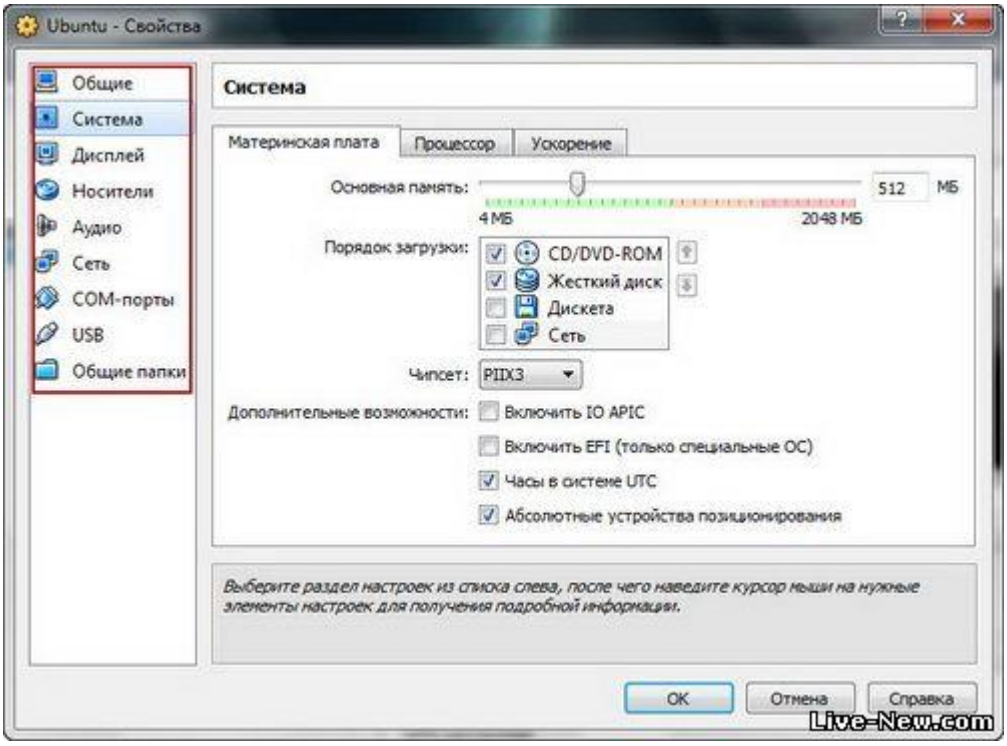


Рисунок 13. Изменение настроек виртуальной машины

Далее приступаем к установке Ubuntu на созданный вами виртуальный компьютер. Для запуска установки операционной системы, вставьте предварительно записанный образ операционной

системы на CD/DVD диск в CD/DVD – ROM нажмите кнопку **Старт** в окне менеджера VirtualBox(см. рис. 14, 15).

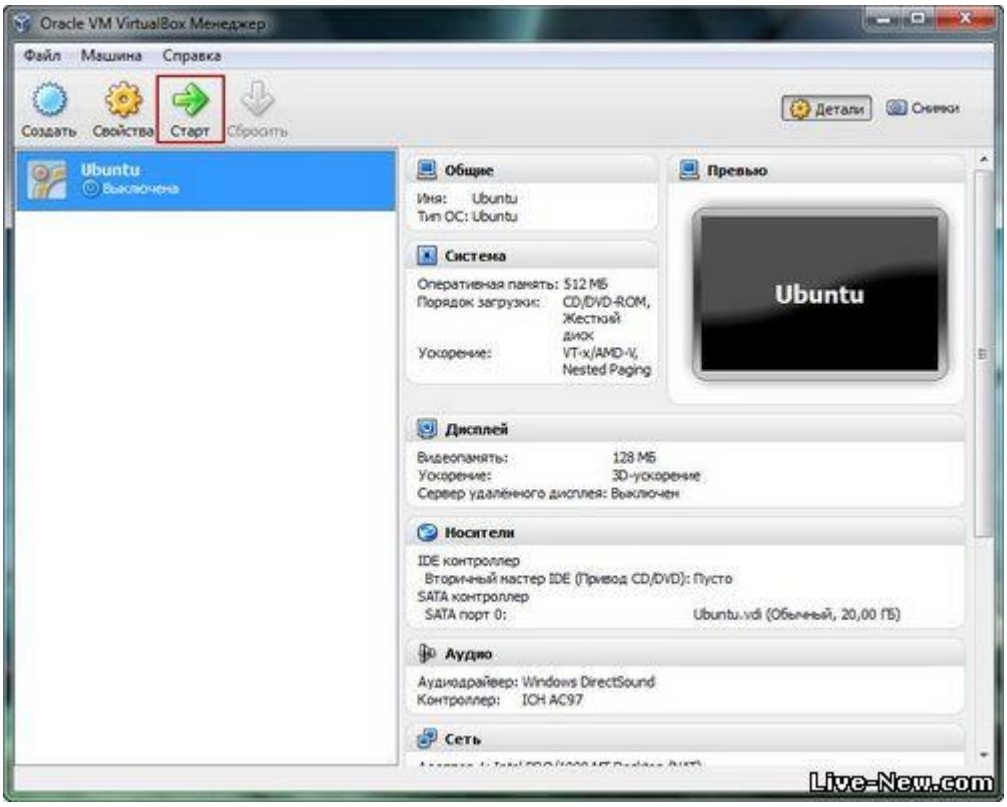


Рисунок 14. Запуск установки ОС

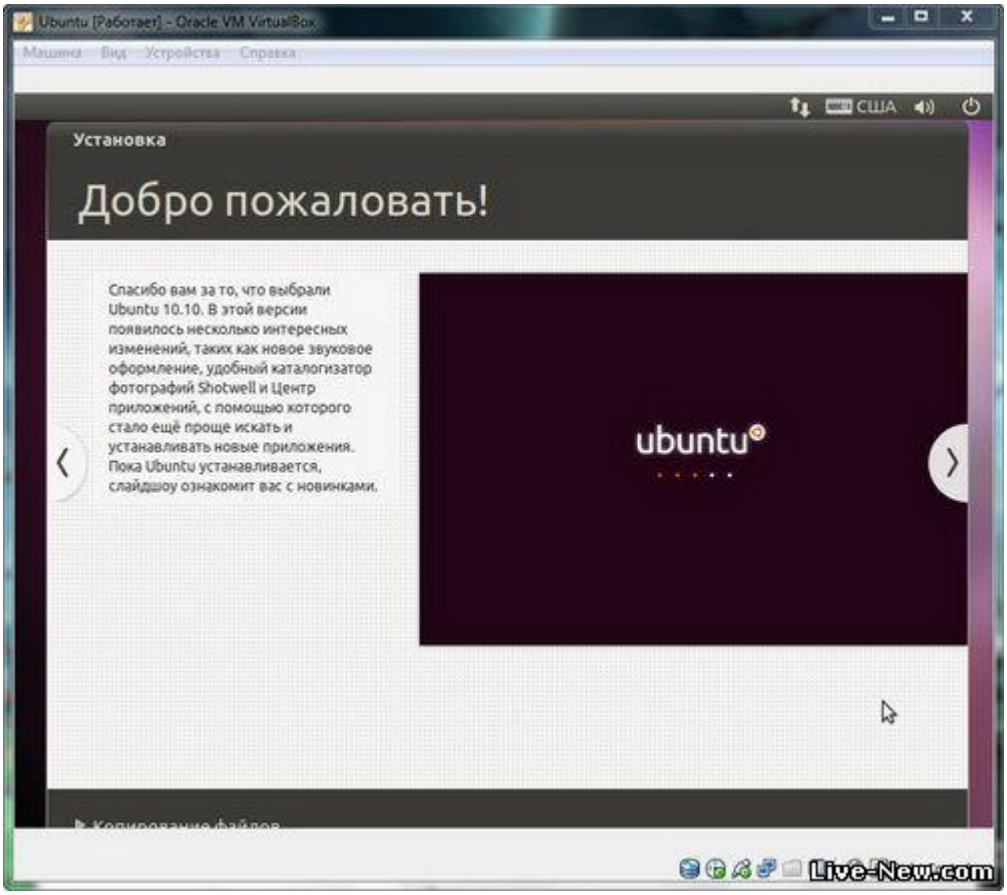


Рисунок 15. Установка ОС

Введение

На сегодня в большинстве организаций в качестве операционных систем для рабочих станций применяются операционные системы семейства Windows. При этом все большее распространение получает ОС Windows XP Professional (в этой статье мы не будем рассматривать Windows XP Home Edition в связи с ее домашним предназначение, хотя большинство рассмотренного будет касаться обеих систем).

Установка ОС

Итак, вы решили установить ОС Windows XP. Для корректной установки рекомендуется создать загрузочный диск с предустановленным Service Pack 1 (тогда вы потратите куда меньше времени на установку). Описывать собственно процесс установки нет особого смысла, так как, я думаю, что у вас есть большой опыт в такого рода работе.

Параметры установки Windows XP

Запустить установку Windows XP можно:

- Из-под MS-DOS с помощью файла winnt.exe (в каталоге I386)
- Из-под Windows с помощью файла winnt32.exe (в каталоге I386)

Набор параметров командной строки у этих двух программ различен.

Параметры winnt.exe таковы:

- **/?** Вызов справки
- **/a** использование специальных средств для людей с ограниченными возможностями
- **/e** задает команду, выполняемую по окончании графической стадии установки ОС
- **/r** включает создание папки в каталоге Windows, которая остается после установки ОС
- **/rx** включает создание временной папки в каталоге Windows, которая будет удалена по окончании инсталляции
- **/s** указывает путь к дистрибутиву Windows. Применяется при установке с сервера сети
- **/t** задает диск для временных файлов. Если этот параметр отсутствует, то используется диск, на котором больше свободного места
- **/u:файл_ответов** задает файл ответов для программы установки Windows
- **/udf:id[,UDF_файл]** указывает идентификатор ID, с помощью которого программа установки Windows определяет значения в UDF_файле (Uniqueness Database File) для модификации файла ответов каждого компьютера при установке системы на множество ПК. Если не указан UDF_файл, то система потребует дискету с файлом при $Unique$.udb

Параметры командной строки winnt32.exe:

- **/?** Справка о программе
- **/checkupgradeonly** производится проверка возможности обновления текущей версии Windows. По окончании проверки будет сгенерирован отчет о возможности установки новой ОС
- **/cmd:command_line** задает команду, которая должна быть выполнена во время завершающей стадии установки ОС
- **/cmdcons** установить консоль восстановления системы и добавить ее вызов в загрузочное меню. Инсталляция ОС не производится
- **/copydir:i386\folder_name** создание дополнительной папки с именем папки в каталоге Windows

- **/copysource:folder_name** создание временной папки в каталоге Windows, по окончании инсталляции папка будет удалена
- **/debug[level]:[filename]** включает протокол отладки (по умолчанию C:\systemroot\Winnt32.log) с заданным уровнем (по умолчанию - 2; "0" -критические ошибки, "1" - обычные ошибки, "2" - предупреждения, "3" - информацию, "4" - детальная информация для отладки)
- **/dudisable** препятствует выполнению динамического обновления (файлы Update Microsoft Windows). Инсталляция выполняется только с первоначальными установками
- **/duprepare:pathname** выполняет модификацию инсталляционного ресурса таким образом, чтобы использовать динамические файлы модификации с сайта Windows Update
- **/dushare:pathname** определяет ресурс, на котором расположены файлы обновлений
- **/m:folder_name** во время установки копируются файлы из дополнительной папки и если они присутствуют, то используются вместо файлов из заданной по умолчанию папки
- **/makelocalsource** указывает программе установки на необходимость скопировать все файлы на локальный жесткий диск. Используется в том случае, если компакт-диск может быть недоступен в процессе установки
- **/noreboot** отключает автоматическую перезагрузку после копирования файлов
- **/s:sourcepath** указывает размещений файлов установки Windows XP (обычно на сервере)
- **/syspart:drive_letter** позволяет скопировать файлы установки на жесткий диск, сделать его активным, перенести на другой компьютер и продолжить установку на этом компьютере. Должен использоваться вместе с ключом /tempdrive
- **/tempdrive:drive_letter** используется вместе / syspart для указания основного раздела, предназначенного для размещения файлов и последующей установки Windows XP
- **/udf:id [,UDB_file]** задает файл базы данных уникальности, модифицирующий файл ответов
- **/unattend** обновляет предыдущую версию Windows в автоматическом режиме. Все пользовательские настройки берутся из предыдущей инсталляции
- **/unattend[num]:[answer_file]** указывается при автоматизированной установке. Имя файла можно опустить, если используется файл Unattend.txt (по умолчанию)

Преобразование файловой системы

Чтобы преобразовать диск из **FAT (FAT32) в NTFS**, воспользуйтесь утилитой Convert. Синтаксис команды

CONVERT том: /FS:NTFS [/V] [/CvtArea:имя_файла] [/NoSecurity] [/X]

- **том** - определяет букву диска (с последующим двоеточием) точку подключения или имя тома.
- **/FS:NTFS** Конечная файловая система: NTFS.
- **/V** Включение режима вывода сообщений.
- **/CVTAREA:имя_файла** Указывает непрерывный файл в корневой папке для резервирования места для системных файлов NTFS.
- **/NoSecurity** Параметры безопасности для преобразуемых файлов и папок будут доступны для изменения всем.
- **/X** Принудительное снятие этого тома (если он был подключен). Все открытые дескрипторы этого тома станут недопустимыми.

Если в вашей организации используется большое количество компьютеров, то необходимо продумать процесс автоматизации установки ОС.

Существует два возможных варианта автоматизации процесса установки:

1. **Автоматизированная установка.** В этом случае используется пакетный файл и сценарий (называемый файлом ответов [1]), благодаря этому отключаются запросы операционной системы, а необходимые данные выбираются из файлов ответов автоматически. Существует пять режимов автоматической установки.
2. **Копирование диска (клонирование).** В этом случае запускается утилита подготовки системы к копированию (Sysprep.exe), которая удаляет идентификатор безопасности (Security Identifier - SID). Затем диск копируется с помощью программы клонирования дисков, например Ghost (Symantec) (http://www.symantec.com/ghost) или Drive Image (Power Quest)

(http://www.powerquest.com/driveimage). После копирования будет выполнена "сжатая" процедура установки (5-10 минут)[2].

Вы установили операционную систему, однако самая тяжелая и продолжительная часть работы еще впереди.

Установка необходимых обновлений

Не взирая на то, что, согласно документации, установка ОС занимает около 1 часа, на самом деле установка, настройка, установка всех критических патчей (обновлений) займет у вас по меньшей мере 4-5 часов (это при условии, что все патчи уже есть у вас на жестком диске или CD-ROM и вам не нужно вытягивать их из Internet).

Итак, вы установили операционную систему. Для дальнейшей установки патчей у вас есть два пути:

1. Воспользоваться службой автоматического обновления Windows Update. Этот путь достаточно хорошо описан в литературе и не требует никаких усилий со стороны программиста. Однако, предположим, что в вашей организации хотя бы 20 компьютеров. Таким образом, вам придется 20 раз воспользоваться этой службой. Если учесть, что объем необходимых патчей составляет на сегодня около 40Мб, то вам придется вытянуть из сети 20*40=800Мб за один раз и в дальнейшем вам необходимо тянуть патчи на каждый компьютер раздельно. Это не самый лучший способ, однако если у вас быстрый канал и ваше руководство не против выбрасывать таким способом деньги, то вам подходит этот путь. Но учтите, что при переустановке ОС вам придется все вытягивать заново[3].
2. Воспользоваться каким-то сканером безопасности для поиска необходимых патчей (обновлений). Для примера рассмотрим бесплатный сканер Microsoft Base Security Analyzer (в данной статье не будет подробно рассматриваться вопрос о методах работы с данным сканером). Данный сканер можно бесплатно загрузить с сайта Microsoft из раздела TechNet.[4] . До начала тестирования необходимо будет извлечь файл Mssecure.xml файл из http://download.microsoft.com/download/xml/security/1.0/nt5/en-us/mssecure.cab Файл Mssecure.xml должен быть помещен в ту же папку, в которой развернут Microsoft Base Security Analyzer

Результатом сканирования будет перечень необходимых патчей, который вы должны будете установить на вашем компьютере.

Рис.1 Результаты сканирования Microsoft Base Security Analyzer

Недостатком данного сканера является то, что он не указывает, какие конкретно обновления вам нужны, а рекомендует обратиться к странице Windows Update, что далеко не всегда удобно. Особенно в том случае, если вы уже вытягивали патчи из Интернета.

В таком случае гораздо удобнее применять коммерческие сканеры безопасности типа LAN Guard Network Scanner или XSpider. Рассмотрим более подробно LAN Guard Network Scanner.

Этот сканер предназначен для поиска уязвимостей в компьютерных сетях не только на базе Windows. Однако, в нашем случае, можно легко воспользоваться ним для поиска уязвимостей на отдельном компьютере. Вам будет рекомендовано посетить конкретные страницы бюллетеня безопасности от Microsoft (рис. 2)

Рис. 2 Результат работы LAN Guard Network Scanner

В таком случае гораздо проще устанавливать обновления и появляется возможность узнать для решения какой уязвимости создано данное обновление.

Анализ процесса установки патчей приведен на рис.3

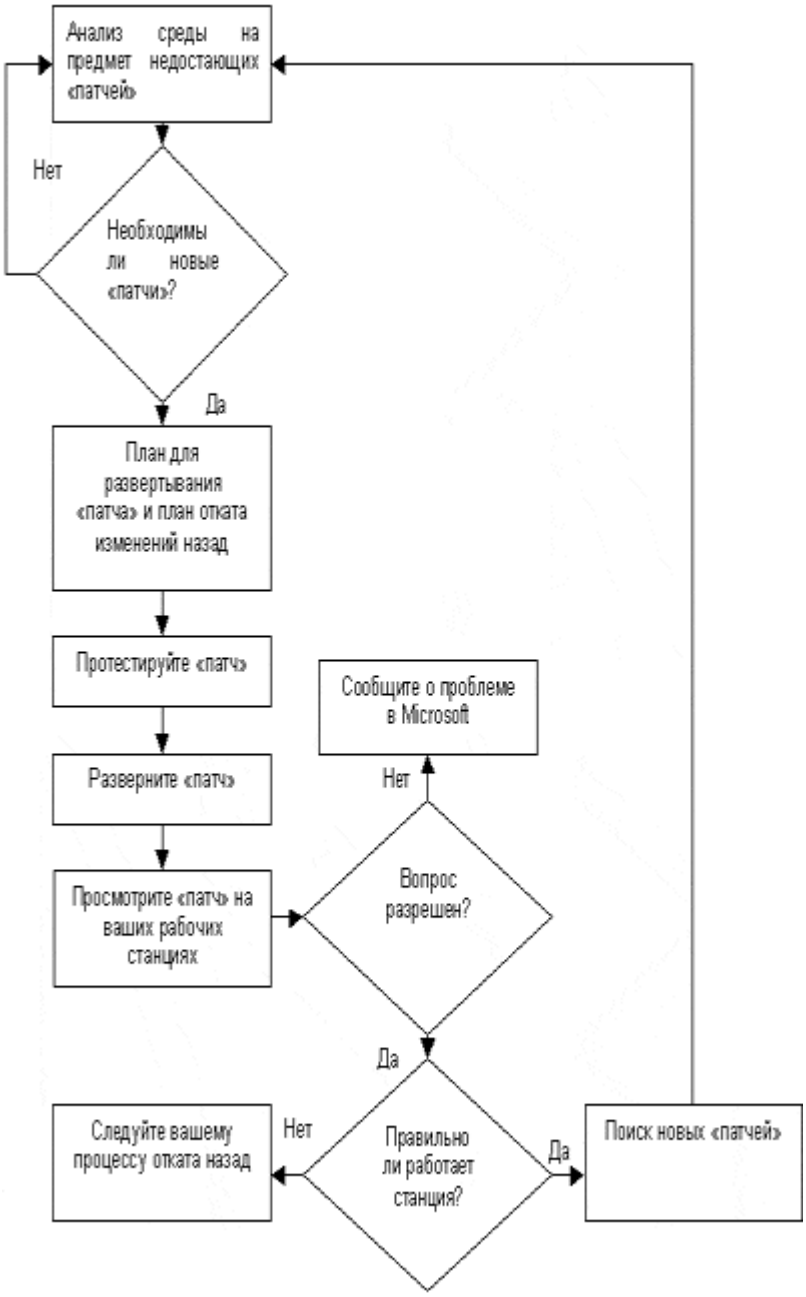


Рис3. Процесс управления установкой обновлений

Стоит исследовать эти шаги подробнее:

- **Анализ.** Посмотрите на текущую среду и потенциальные угрозы. Определите патчи, которые вы должны инсталлировать, чтобы сократить количество угроз вашей среде.
- **План.** Установите, какие патчи надо инсталлировать, чтобы сдерживать потенциальные угрозы и обнаруженные вами уязвимые места. Определитесь, кто будет осуществлять тестирование и инсталляцию, и какие шаги нужно сделать.
- **Тестирование.** Просмотрите доступные патчи и разделите их на категории для вашей среды.
- **Инсталляция.** Инсталлируйте нужные патчи, чтобы защитить вашу среду.
- **Мониторинг.** Проверьте все системы после инсталляции патчей, чтобы удостовериться в отсутствии нежелательных побочных эффектов.

- **Просмотр.** Важной частью всего процесса является тщательный просмотр новых изданных патчей, вашей среды, и выяснение, какие из патчей нужны вашей компании. Если во время просмотра вы обнаружите, что необходимы новые патчи, начните снова с первого шага.

Примечание: Настоятельно рекомендуется сделать резервную копию всей рабочей системы до инсталляции патчей.

Проверка среды на предмет недостающих патчей

Так как это непрерывный процесс, вам нужно убедиться в том, что ваши патчи соответствуют последним установкам. Рекомендуется постоянно следить за тем, чтобы иметь новейшую информацию о патчах. Иногда выпускается новый патч, и вам необходимо установить его на всех станциях. В других случаях в сети появляется новая станция, и на ней нужно установить все необходимые обновления. Вам следует продолжать проверять все ваши станции, чтобы убедиться в том, что на них установлены все необходимые новейшие патчи. Вообще, вопрос установки патчей далеко не так прост, как кажется на первый взгляд и полное рассмотрение этого вопроса выходит за пределы нашей статьи.

Следует учесть, что иногда после установки последующего патча необходимо переустановить предыдущий. В частности в нашей практике такое встречалось неоднократно.

Итак, предположим, что все патчи установлены и ваша система не имеет дырок, связанных с их отсутствием. Учтите, что это состояние только на текущий момент времени, завтра возможно вам придется устанавливать новые патчи. Этот процесс, увы, беспрерывен.

Следующим шагом в вашей работе будет настройка операционной системы.

Настройка Windows XP[5]

Встроенная оптимизация Windows XP.

Самое интересное, что оптимизация в Windows XP производится постоянно. По мере того, как вы запускаете приложения, Windows XP наблюдает за вашим поведением и записывает динамический файл layout.ini. Каждые три дня, после того, как система сочтет компьютер бездействующим, она изменяет физическое местоположение некоторых программ на жестком диске для оптимизации их времени запуска и выполнения.

Windows XP также ускоряет процесс загрузки системы и оптимизирует запуск программ с помощью предсказаний. Windows XP наблюдает за кодом и программами, которые запускаются сразу после загрузки, и создает список, позволяющий предсказать запрашиваемые данные во время загрузки. Точно также при запуске отдельных программ, Windows XP следит за используемыми программой компонентами и файлами. В следующий запуск приложения Windows XP предсказывает список файлов, которые потребуются программе.

Предсказания используются и в ядре Windows XP, и в планировщике задач. Ядро следит за страницами, к которым обращается данный процесс сразу же после его создания. Далее служба создает ряд инструкций предсказания. Когда процесс будет создан в следующий раз, ядро выполнит инструкции предсказания, ускорив выполнение процесса.

Оптимизация диска и ускорение запуска приложений/загрузки тесно работают вместе. Списки, записанные при запуске приложения и при загрузке системы, используются при выполнении оптимизации файловой системы для более быстрого доступа к программам.

Однако можно для оптимизации процесса загрузки использовать бесплатную утилиту BootVis, производства компании Microsoft [6]. Рабочее окно программы приведено на рис. 4

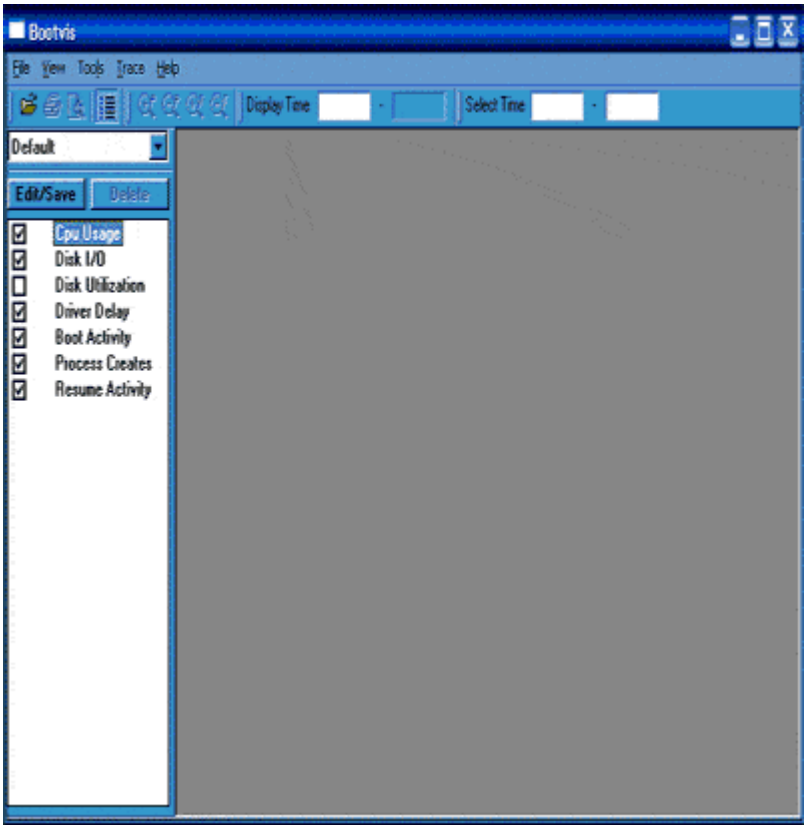


Рис.4 Рабочее окно программы Boot Vis

Настройка выключения компьютера

Причиной того, что Windows XP выгружается (выключается) слишком долго, в большинстве случаев является неправильное завершение некоторых процессов. В этом случае система ожидает в течение заданного интервала времени. Этот интервал задается параметром реестра WaitToKillServiceTimeout который находится в ветке HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control. Значение этого ключа задается в миллисекундах. По умолчанию, это время равно 20000. Исходя из нашего опыта, следует установить его равным 5000, что означает 5 секунд. Не следует устанавливать его меньше, так как в этом случае система будет выгружать программы ранее, чем они смогут сохранить свои данные.

Ускорение графики.

Иконки и обои.

Чистый рабочий стол - это самый лучший рабочий стол. Никогда не ставьте обои! Более странного поступка трудно себе представить. Системной памяти и процессору наверняка найдется лучшее применение, чем играться с красивым фоном и сортировать сотни иконок. Как и в предыдущих версиях Windows XP, чрезмерное количество иконок и обои

требуют большого расхода системной памяти. Особенно тяжелым бременем на систему ложится анимированный рабочий стол.

С другой стороны, удар по производительности не слишком велик, если ваша система оснащена более 256 Мб памяти и нормальным процессором (где-то 1000 MHZ или быстрее). Если же у вас 64 Мб памяти и Pentium 2, то здесь придется серьезно экономить, отключая все, что только возможно.

Память

В опции Memory Usage при установленном размере физической памяти 256 Мб и выше отметьте параметр System Cash. Если памяти на компьютере меньше 256 Мб, то система будет работать быстрее при установленном значении Programs. Реестр Windows содержит несколько ключей, которые позволят настроить оптимальную работу Windows с памятью:

ключ HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management:

ClearPageFileAtShutdown - возможность стирать файл подкачки при выходе из Windows (доступен из локальной политики безопасности). По умолчанию равен 1, что соответствует безопасным настройкам, можно поставить равным 0, что обеспечит максимальное быстродействие при перезагрузке, однако снизит безопасность.

DisablePagingExecutive - запрещает записывать в файл подкачки код (драйвера и т.д.) и требует оставлять их всегда в физической памяти. По умолчанию равен 0. Если у вас объем памяти больше 256 Мб, то рекомендуется присвоить значение 1, что ускорит работу.

Отключаем функцию Prefetch для компьютеров с малым количеством оперативной памяти

На компьютерах с объемом оперативной памяти менее 128 Мб (хотя мы рекомендуем не менее 256 Мб), функция Prefetch может вызвать замедление работы системы, поэтому необходимо ее отключить [7]. Для отключения функции Prefetch необходимо в реестре в ветке HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters выбрать параметр EnablePrefetcher и установить значение, равное 0.

Удаляем файлы prefetch автоматически

Для этого создаем командный файл (*.bat) следующего содержания:

```
del c:\Windows\Prefetch\*.* /Q
```

В случае, если ваша папка находится не на диске C: то смените имя диска.

Очистка этой папки ускорит быстродействие вашей системы.[8]

Уменьшаем время загрузки приложений

Корпорация Microsoft создала параметр, который позволяет ускорить загрузку приложений. Для этого достаточно добавить в свойствах программы ключ /prefetch:1

Правой клавишей мыши нажмите на ярлыке нужной программы и выберите из меню пункт "Свойства"

В строке "Объект" после указания пути к файлу добавьте /prefetch:1 (пробел перед ключом обязателен).

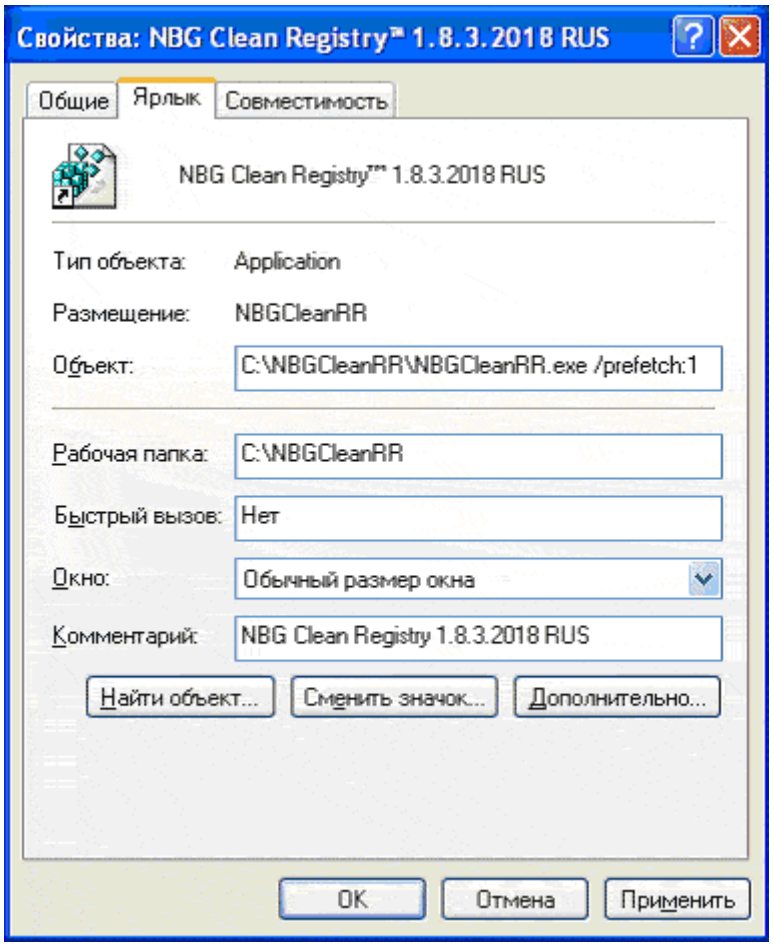


Рис. 5 Добавление параметра

Ядро:

Чтобы процесс закрытия зависшего приложения проходил быстрее необходимо изменить параметр HungAppTimeout в ветке HKEY_CURRENT_USER\Control Panel\Desktop (по умолчанию значение ключа составляет 5000 миллисекунд). Рекомендуемое значение 2000мс. Тут же есть параметры WaitToKillServiceTimeout и WaitToKillAppTimeout, определяющие время ожидания до закрытия зависшей службы или приложения соответственно (значения по умолчанию составляет 20000мс). Рекомендуемое значение 5000мс.

Ключ AutoEndTasks (по умолчанию 0), разрешает системе автоматическое закрытие зависших приложений. При этом, значение его устанавливаем равным 1.

Следует учесть, что при установке значений ниже рекомендуемых, можно столкнуться с проблемой в виде не вовремя снятого приложения или службы.

Нажатие на файл .avi в проводнике вызывает 100% загрузку процессора

Можно столкнуться с проблемой Windows XP при открытии файлов в Проводнике с расширением .avi. При нажатии на такой файл, система пытается прочесть данные из него (размер, ширина, высота и т.д.).

Для решения этой проблемы необходимо в ключе реестра

HKEY_CLASSES_ROOT\SystemFileAssociations\.avi\shellex\PropertyHandler удалить значение по умолчанию {87D62D94-71B3-4b9a-9489-5FE6850DC73E}. Теперь в окне не будут показываться свойства файла.

Снизьте количество эффектов

Благодаря новому виду и GNOME-подобной поддержке скинов, Windows XP выглядит красивее любой предыдущей версии Windows.

Все эти визуальные утехи могут снижать реакцию интерфейса на пользователя. XP запускает несколько тестов для автоконфигурации своего пользовательского интерфейса, чтобы сохранить как удобство, так и красоту, но вы легко можете все исправить. Если исчезающие меню вам больше досаждают, нежели нравятся, а тени под окнами диалогов вам безразличны, то вы можете убрать все лишнее.

Некоторые настройки выполняются через закладку Оформление (Appearance) в свойствах монитора, которые вы можете вызвать, нажав правую клавишу мыши на любой свободной части экрана и выбрав Свойства (Properties).

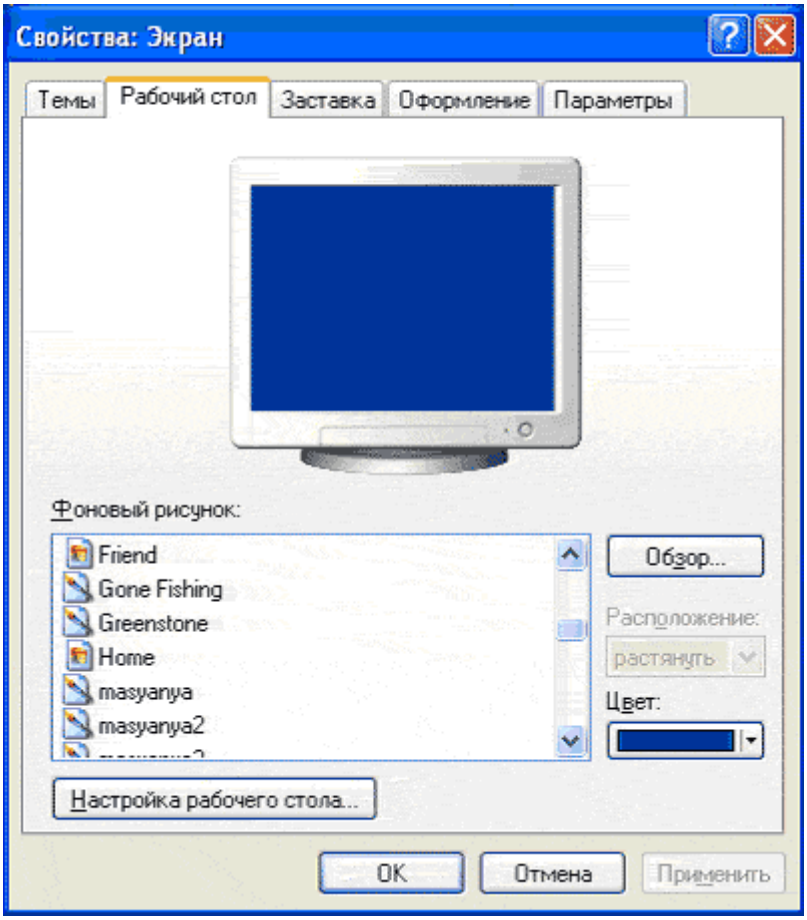


Рис. 4 Свойства экрана

Нажмите клавишу Эффекты (Effects) и вы сможете настроить переходы в меню, тени и шрифт, включая новую технологию улучшения читаемости шрифта Microsoft ClearType. По нашему мнению, ClearType хорош для ноутбуков и ЖК мониторов, но на ЭЛТ текст выглядит слишком жирно и смазано. Учтите, что даже на ЖК мониторе не всем нравится ClearType, так что выбирайте по своему вкусу.

Вы можете и дальше настраивать производительность графического интерфейса через Свойства системы (System Properties). Откройте свойства через Панель управления (Control Panel) или нажмите правой клавишей мыши на значок Мой компьютер (System) и выберите там Свойства (Properties).

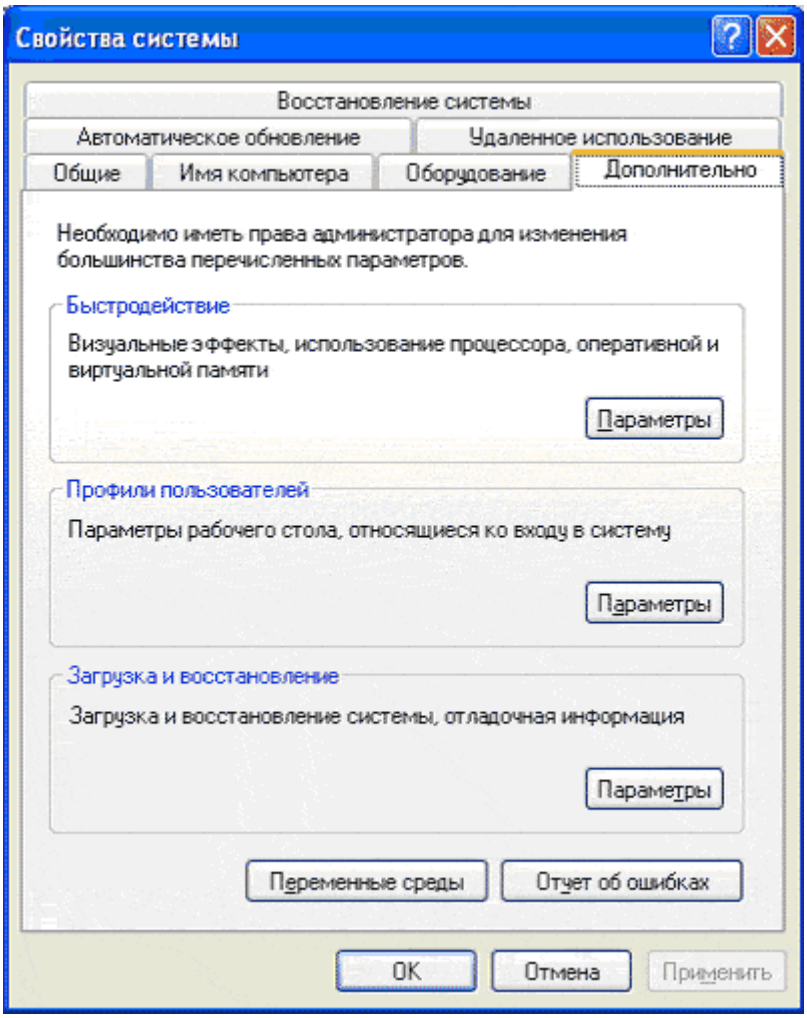


Рис. 5 Свойства системы

Далее перейдите к закладке Дополнительно (Advanced) и нажмите Параметры (Settings) в панели Производительность (Performance). Здесь вы можете указать как максимальную производительность, так и максимальную красивость, равно как выбрать необходимые параметры самому.

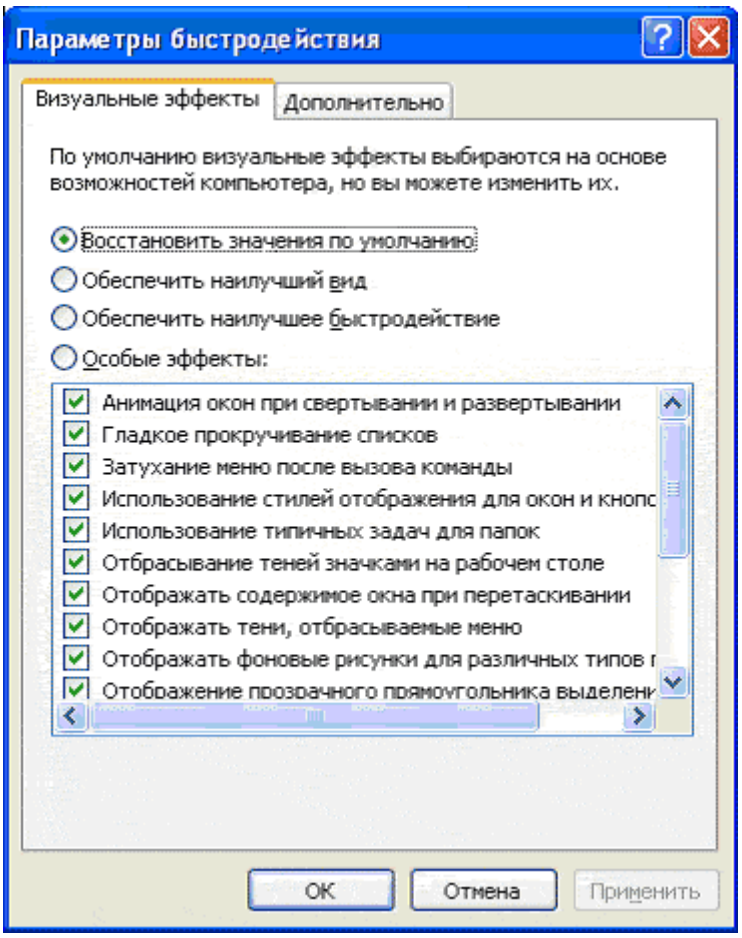


Рис. 6 Параметры быстродействия

Перейдите к закладке Дополнительно (Advanced) в Параметрах быстродействия (Performance Options) и убедитесь, что распределение ресурсов процессора и памяти выставлено на оптимизацию работы программ - вам нужно указывать приоритет фоновых служб и кэша, если только ваш компьютер исполняет роль сервера.

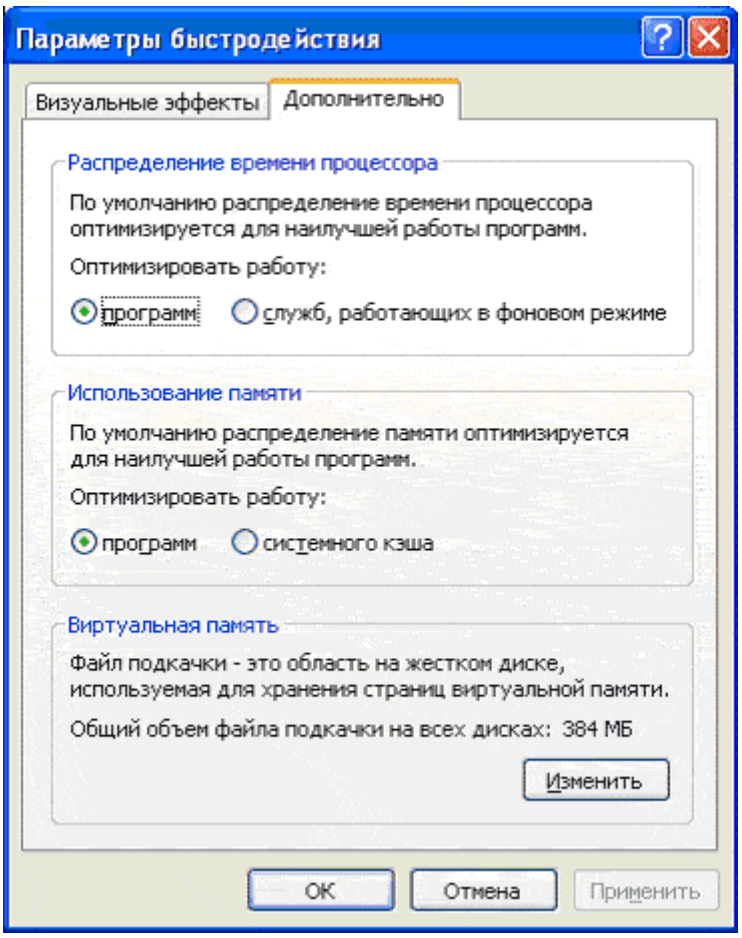


Рис.7 Дополнительные параметры быстродействия

Здесь вы также можете указать размер и местоположение файла подкачки. Но Windows XP обычно сама прекрасно выбирает этот размер.

Быстрое переключение между пользователями

Такая функция доступна в обеих версиях Windows XP, если компьютер не входит в домен. Быстрое переключение позволяет пользователям одного компьютера быстро переключаться между учетными записями без завершения сеанса. Прекрасная функция, если вашим компьютером пользуются мама, папа и всякие сестры-братья, однако такое переключение требует большого расхода оперативной памяти.

Если в систему вошло более одного пользователя, то настройки каждого пользователя, равно как и запущенные программы сохраняются в памяти при переключении к другому пользователю. Скажем, если у вас запущен Word, Excel и какая-нибудь игра, и в это время придет ваш брат, переключит систему на себя и попытается поиграть в Red Faction, он заметит явное падение производительности, до полной остановки игры.

Windows XP автоматически отключает быстрое переключение между пользователями, если компьютер оснащен 64 Мб памяти или меньше. Для максимальной производительности убедитесь, что в одно время в систему заходит только один пользователь. Вы также можете отключить эту функцию, зайдите в Панель управления " Учетные записи пользователей (Control Panel " User Accounts) нажмите кнопку "Переключение пользователей":

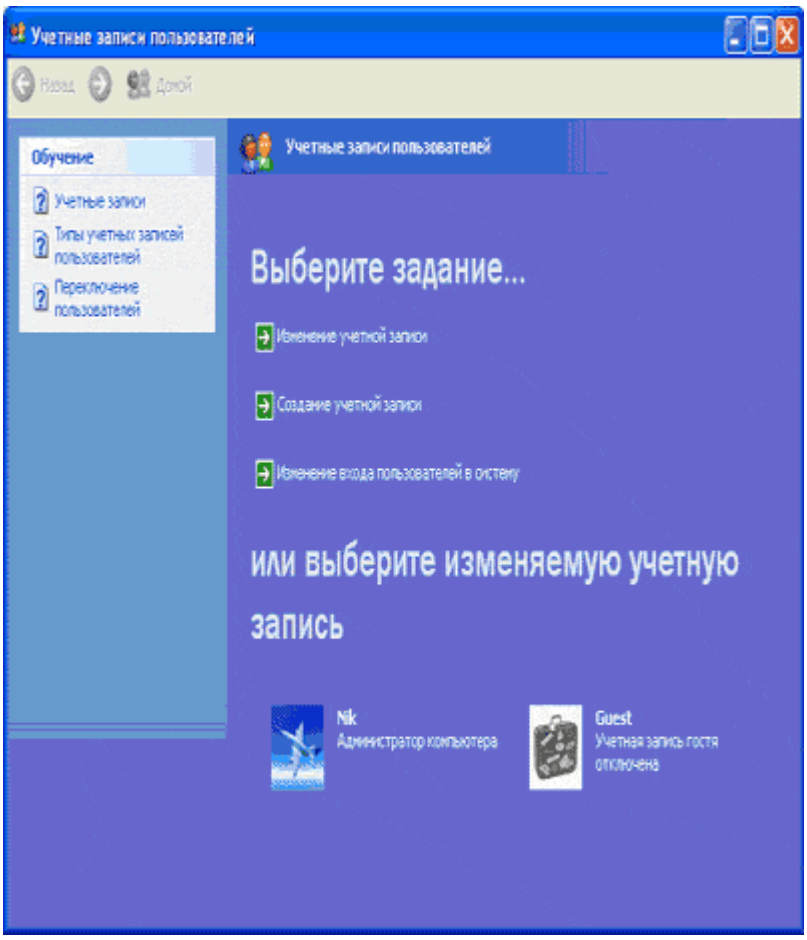


Рис.8 Учетные записи пользователей

и уберите галочку с пункта "Использовать быстрое переключение пользователей".

Рис.9 Выбор параметров входа и выхода из системы

Восстановление системных файлов

Полезная функция, если ваш компьютер не используется исключительно для ресурсоемких задач типа игр. Так что лучше оставить ее включенной. При этом компьютер периодически создает слепки критичных системных файлов (файлы реестра, COM+ база данных, профили пользователей и т.д.) и сохраняет их как "точку отката". Если какое-либо приложение "снесет" вашу систему, или что-то важное будет испорчено, вы можете обратить компьютер в предыдущее состояние - в точку отката.

Точки отката автоматически создаются службой "Восстановления системы" (System Restore) при возникновении некоторых ситуаций типа установки нового приложения, обновления Windows, установки неподписанного драйвера и т.д. Вы можете и вручную создавать точки отката через интерфейс Восстановления системы (System Restore), который можно вызвать, пройдя путь: Пуск " Программы " Стандартные " Служебные " Восстановление системы (Start " Programs " Accessories " System Tools " System Restore).

Рис.10 Восстановление системы

Восстановление системных файлов опирается на фоновую службу, которая минимально сказывается на быстродействии и записывает снимки, отнимающие часть дискового

пространства. Вы можете вручную отвести максимальный объем дискового пространства для данной службы. Вы также можете полностью отключить службу для всех дисков.

Отключить можно, поставив галочку "Отключить службу восстановления". Поскольку служба восстановления системных файлов может влиять на результаты тестовых программ, ее обычно отключают перед тестированием.

Автоматическая очистка диска

Для проведения очистки жесткого диска от ненужных файлов используется программа cleanmgr.exe

Ключи программы:

/d driveletter: - указывает букву диска, которая будет очищаться

/sageset: n - эта команда запускает мастер очистки диска, и создает ключ в реестре для сохранения параметров. Параметр n может принимать значения от 0 до 65535.

/sagerun: n - используется для запуска мастера очистки диска с определенными параметрами, которые были заданы заранее с помощью предыдущего ключа.

Для автоматизации этого процесса можно воспользоваться планировщиком заданий.

Регулярно производите дефрагментацию.

DOS и не-NT версии Windows мало заботились об оптимизации своих файловых систем. Когда вы устанавливаете и удаляете программы, то в различных областях дискового пространства создаются "дыры". В результате свободное место представляет собой не сплошной блок, оно разбросанно по всему диску. При заполнении свободного пространства файлы также оказываются разбросанными по нескольким секторам, что сильно снижает производительность - при обращении к файлу диску приходится читать не один последовательный участок, а несколько произвольно разбросанных.

В NT-версиях Windows, использующих файловую систему NTFS, применяются особые меры для сохранения целостности дискового пространства - но фрагментация все равно происходит. Поэтому вы должны регулярно дефрагментировать ваш жесткий диск, причем регулярность зависит от характера вашей деятельности на компьютере.

В случае использования файловой системы FAT32 дефрагментация еще более необходима!

Если вы часто устанавливаете и удаляете программы, или вы постоянно создаете, перемещаете или удаляете файлы, то вы должны выполнять дефрагментацию раз в неделю. Если же вы долгое время используете одни и те же приложения, при этом вы не слишком часто перемещаете файлы, то вы можете увеличить промежуток между дефрагментациями до одного месяца. Если вы достаточно часто выполняете дефрагментацию, то вы не заметите ощутимого прироста в производительности после дефрагментации. Это совершенно нормально. Если же прирост явно ощутим, то вы слишком долго не выполняли дефрагментацию.

Создаем bat-файл, который, к примеру, назовем defrag.bat следующего содержания:

```
Rem **This batch file is defragmenting your hard drive.**
Rem **To cancel Press Ctrl+C on the keyboard.**
Defrag.exe C: -F
```

Формат команды Defrag:

defrag <том> [-a] [-f] [-v] [-?]

том - Буква диска, или точка подключения (например, c: или d:\vol\mpoint)

-a - Только анализ

-f - Дефрагментация даже при ограниченном месте на диске

-v - Подробные результаты

-? - Вывод справки о команде

Теперь в Планировщике заданий указываем этот файл и ставим его в расписание. Рекомендуется установить запуск каждую неделю (но не меньше 1 раза в месяц). Теперь Дефрагментация диска будет автоматически запускаться в Windows XP.

Вы также можете установить дефрагментацию в расписание и без создания bat-файла, делается это так:

- Заходим в Панель управления -> Назначенные задания -> Добавить задание
- Нажмите Обзор и выберите программу Defrag.exe, находится она в каталоге C:\Windows\System32
- Во время последнего экрана не забудьте поставить галочку около пункта Установить дополнительные параметры после нажатия кнопки Готово
- В строке выполнить после адреса файла добавьте ключ -f

Отключение неиспользуемых служб

Отключите ненужные системные службы (сервисы), ускорив тем самым работу системы. Заодно и памяти немного освободится...

Вот список служб, которые, в принципе, можно безбоязненно отключить:

Автоматическое обновление (Automatic Updates). Учитывая, что обновлять систему можно и вручную, имеет смысл отключить эту службу. Особенно в том случае, если нет постоянного соединения с Интернетом. Следует только не забыть не только отключить службу, но и отменить Автоматическое обновление в одноименной закладке Свойствах системы (System Properties).

Обозреватель сети (Computer Browser). Занимается обновлением списка компьютеров в сети. При отсутствии сети не нужна.

Служба шифрования (Cryptographic Service). Служба безопасного обмена ключами и шифрования передаваемых данных в локальной сети. Если локальной сети нет, то эту службу можно отключить, если сеть есть - думайте сами...

DHCP клиент (DHCP client). Занимается автоматическим распределением IP-адресов. Если сети нет (ни локальной, ни Интернета - даже через модем), то эта служба не нужна.

Служба сообщений (Messenger). Отвечает за прием и отправку сообщений, посланных администратором. При отсутствии сети (и администратора) абсолютно бесполезна.

Сетевые соединения (Network Connections). Управление всеми сетевыми соединениями. Если нет сети (в том числе нет и Интернета), то эта служба не нужна.

Спулер печати (Print Spooler). Если принтера нет, то он не нужен.

Portable media serial number. Отвечает за получение серийного номера переносного музыкального устройства, подключаемого к компьютеру.

Protected Storage. Защита важных данных, в том числе, ключей пользователей; запрещает неавторизированный доступ. Если нет сети (в том числе и Интернета), то эту службу можно отключить (если безопасность не волнует - можно отключить и при наличии сети).

Remote Registry Service. Предназначена для удалённого управления реестром (нужна только администраторам сети).

System Event Notification. Отслеживает системные события. Если все уже настроено и нормально работает, можно отключить.

SSDP Discovery. Обеспечивает работу подключаемых устройств, поддерживающих UPnP (универсальная система Plug & Play, которая, по задумке, должна связывать компьютер с самой различной бытовой техникой, вроде пылесоса или холодильника. Планировщик заданий (Task Scheduler). Запуск приложений в заданное время. Если эта возможность не используется, эту службу можно отключить.

Telephony. Взаимодействие с модемом. Нет модема - отключаем службу.

Telnet. Обеспечивает возможность соединения и удалённой работы по протоколу telnet. Если не знаете (и не хотите знать), что это такое, то эту службу можно отключить.

Uninterruptible power supply. Управляет работой бесперебойных источников питания (UPS). Если UPS с обратной связью нет, то не нужна.

Terminal Service. Служит для подключения к компьютеру по сети и удаленного управления им. Домашнему пользователю она, в общем-то, ни к чему.

Windows time. Синхронизирует время на локальной машине и сервере; если нет time-сервера, то и служба не нужна.

Wireless zero configuration. Служба автоматической настройки беспроводных сетей стандарта 803.11 и 803.11b.

Подчеркну, что этот список - не окончательный, потому что необходимость той или иной системной службы определяется теми задачами, которые выполняются на конкретном компьютере, поэтому каждый должен решать сам, что можно отключить, а что нет. Главное - не переборщить, помня, что последствия необдуманных действий могут быть непредсказуемыми. И еще о последствиях. Для того чтобы уменьшить риск "запороть" систему, имеет смысл перед началом экспериментов со службами сделать резервную копию той ветви реестра, что отвечает за запуск системных служб: открываем regedit, идем в HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services, выбираем в меню File, а там - пункт Export Registry Key.

Еще один метод, позволяющий ускорить работу системы и несколько освободить занимаемую ею оперативную память, заключается в отключении Dr.Watson'a, отладчика, запускаемого по умолчанию при каждом сбое в работе приложений. Чтобы отключить этого "доктора", нужно будет в реестре найти ключ HKEY_LOCAL_MACHINE \SOFTWARE \Microsoft \Windows NT \CurrentVersion \AeDebug и изменить в нем значение параметра Auto на 0.

После такой модификации реестра при возникновении сбоя в работе приложения система будет предлагать либо закрыть его, либо передать отладчику для отладки (если выбрать второе, то запустится Dr.Watson и создаст лог-файл).

Следующий этап - оптимизация интерфейса, призванная ускорить его работу. Заходим в System Properties, открываем закладку Advanced, нажимаем в разделе Performance кнопку Settings и в открывшейся вкладке Visual Effects отмечаем пункт Adjust for best performance, отключая тем самым абсолютно все эффекты. А можно отключить их и по отдельности, оставив те, без которых прожить ну никак нельзя...

Теперь примемся за стартовое меню. Изначально оно открывается с некоторой задержкой (по умолчанию - 400 миллисекунд), регулировать которую можно, изменяя в реестре значение ключа MenuShowDelay, находящегося по адресу HKEY_CURRENT_USER\ControlPanel\Desktop. В случае установки для этого параметра значения 0 меню будет появляться без задержки.

Там же - в реестре - находится еще один параметр, изменение которого приведет к некоторому ускорению работы интерфейса - MinAnimate, включающий анимацию при сворачивании и разворачивании окон, находится по адресу HKEY_CURRENT_USER\ControlPanel\Desktop \WindowsMetrics. Значение 1 - эффект анимации включен, 0 - выключен. Если этого ключа в реестре нет, то создайте его (тип - String). И не забудьте - для вступления подобных изменений в силу необходимо перезагрузить компьютер.

Открытие на NTFS-разделе папок с большим количеством файлов происходит довольно медленно, потому что Windows каждый раз обновляет метку последнего доступа к файлам и на это, естественно, тратиться какое-то время. Для отключения этой функции нужно запустить regedit и по адресу HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet\Control\FileSystem создать параметр типа DWord, назвав его NtfsDisableLastAccessUpdate и присвоив ему значение 1.

Планировщик пакетов QoS (QoS Packet Scheduler). Этот компонент, устанавливаемый только в Windows XP Pro, включает функцию Quality of Service. Данная функция используется для поддержки протокола IPv6, который на сегодня повсеместно не распространен. На данный момент эту службу лучше отключить (удаление QoS Packet

Scheduler из свойств соединения не освобождает канал от резервирования 20% пропускной способности канала). Отключение производится с помощью Групповой политики (gpedit.msc). Выберите Group Policy - Local Computer Policy - Administrative Templates - Network - QoS Packet Scheduler. Включите Limit reservable bandwidth и уменьшите Bandwidth limit с 20% до 0%.

В русской версии Пуск - Выполнить - gpedit.msc - Конфигурация компьютера - Административный шаблоны - Сеть - Диспетчер пакетов - Ограничить резервируемую пропускную способность. В качестве значения параметра указать 0%.

Отключаем Службу индексирования

Служба индексирования создает индексы содержимого и свойств документов на локальном жестком диске и на общих сетевых дисках. Имеется возможность контроля за включением сведений в индексы. Служба индексирования работает непрерывно и почти не нуждается в обслуживании.

Нужно помнить о том, что служба индексирования потребляет большое количество ресурсов процессора. Если вы не пользуетесь активно поиском по контексту файлов, то данную службу можно отключить. Для этого:

- Заходим в Панель управления - Установка и удаление программ - Установка компонентов Windows
- В появившемся списке ищем Службу индексирования и убираем галочку

Отключаем поиск в zip архивах

По умолчанию, поиск в Windows XP производится и в .zip архивах. Скорость поиска возрастет, если отключить эту службу. Для этого необходимо в командной строке набрать:

regsvr32 c:\winnt\\system32\zipfldr.dll /u или же regsvr32 c:\windows\\system32\zipfldr.dll /u

Для включения поиска в .zip архивах:

regsvr32 c:\winnt\\system32\zipfldr.dll или же regsvr32 c:\windows\\system32\zipfldr.dll

Как удалить "скрытые" компоненты Windows XP?

В отличие от Windows 9*/NT, в процессе установки Windows XP нет возможности выбирать необходимые компоненты. На мой взгляд, это правильное решение Microsoft - сначала следует установить операционную систему со всеми ее причудами, а уж затем, поработав, можно решать, что следует оставить, а что нет.

Однако при этом в окне "Add/Remove Windows Components", что присутствует в апплете "Add or Remove Programs" Контрольной панели, удалять-то практически нечего, потому что многие из составляющих Windows скрыты от шаловливых ручек не слишком опытных пользователей. Для решения этой проблемы открываем системную папку Inf (по умолчанию - C:\Windows\Inf), находим в ней файл sysoc.inf, открываем его и удаляем во всех строках слово HIDE. Главное при этом - оставить неизменным формат файла, то есть следует удалять только HIDE, оставляя запятые до и после этого слова.

Для примера - исходная строка и та, что должна получиться:

```
msmsgs=msgrocm.dll,OcEntry,msmsgs.inf,hide,7
msmsgs=msgrocm.dll,OcEntry,msmsgs.inf,,7
```

Сохраняем файл sysoc.inf, открываем "Add/Remove Windows Components" и видим значительно более длинный список, чем тот, что был на этой страничке до проведения описанной выше операции. Правда, и в этом случае много удалить не получится.

Кстати, точно также можно поступить и в случае с Windows 2000...

Настройка жесткого диска.

Проверьте настройки жесткого диска, поскольку файл подкачки находится на диске. Правильная конфигурация его влияет на скорость работы системы. В свойствах системы откройте Device Manager (либо, открыв свойства любого диска в проводнике, закладка Hardware) и просмотрите свойства вашего жесткого диска. Убедитесь, что стоит отметка Enable write caching on the disk в закладке Polices.

Если диск SCSI доступны следующие значения в закладке SCSI Properties: Disable Tagged Queuing и Disable Synchronous Transfers должны быть не отмечены.

Ultra DMA:

Убедитесь что DMA включено для всех IDE устройств системы. Проверить можно в том же Device Manager " IDE ATA/ATAPI controllers " Primary/Secondary IDE Channel " Advanced Settings.

Параметр Device Type позволяет Windows автоматически определять подключенные устройства, если канал свободен установите значение None - это немного ускорит загрузку системы.

Параметр Transfer mode Windows XP ставит, как правило, по умолчанию и позволяет Windows использовать максимальный DMA поддерживаемый устройством либо PIO, убедитесь, что значение установлено DMA if available.

Дополнительные настройки скорости:

Откройте My Computer " Properties " Advanced " Performance Settings " Advanced в параметре Processor scheduling должно быть отмечено значение Programs. В противном случае Windows будут распределять процессорное время равномерно между всеми программами, включая сервисы, что для игр не приемлемо. В опции Memory usage при установленном у вас размере физической памяти 256MB и выше отметьте параметр System cache, если памяти на компьютере меньше 256 MB система будет работать быстрее при установленном значении Programs Аналогичен параметру реестра LargeSystemCache (см. ниже).

Файл BOOT.INI в Windows XP

Специальный текстовый конфигурационный файл boot.ini, который используется в процессе загрузки - один из важнейших системных файлов Windows XP. Этот файл выполняет следующие функции:

- Управление содержимым меню выбора операционной системы

- Управление процессом загрузки
- Задание некоторых параметров системы

Редактировать данный файл можно либо вручную, либо с помощью программы Boot.ini Editor (http://www.dx21.com/SOFTWARE/Dx21/ViewItem.ASP?NTI=2&SI=2&OID=14) . Раздел [boot loader] служит для задания параметров загрузки операционной системы. Параметр timeout = 30 (по умолчанию) определяет количество секунд, в течение которого пользователь может выбирать один из пунктов меню. При timeout = 0 загрузочное меню не отображается. При timeout = -1 меню находится на экране неограниченное время.

Параметр default = определяет путь к загружаемой по умолчанию системе, может указываться в меню "Загрузка системы". В разделе [operation systems] находятся сведения об установленных операционных системах.

Строение файла BOOT.INI в простейшем случае, с одной операционной системой на диске ПК Intel x86, выглядит следующим образом:

```
[boot loader]
timeout=5
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\
WINNT="Windows XP Professional" /fastdetect
```

При использовании двух операционных систем, например, Windows Me и Windows XP, содержимое файла будет примерно таково:

```
[boot loader]
timeout=5
default=C:\
[operating systems]
C:\="Windows Millennium Edition"
multi(0)disk(0)rdisk(0)partition(2)\
WINNT="Windows XP Professional" /fastdetect
```

Здесь:

Multi(0) - порядковый номер адаптера, с которого осуществляется загрузка. Всегда имеет значение 0.

disk(0) - всегда равен 0 (для большинства BIOS)

rdisk(X) - определяет порядковый номер жесткого диска с которого осуществляется загрузка (от 0 до 3)

partition(Y) - порядковый номер раздела жесткого диска, с которого загружается ОС. Нумерация начинается с 1. Не нумеруются расширенные разделы MS DOS (тип 5) и разделы типа 0 - неиспользуемые.

Для восстановления файла boot.ini можно воспользоваться командой bootcfg. Эта команда доступна из режима командной строки, и может быть использована для настройки, извлечения, изменения или удаления параметров командной строки в файле boot.ini

Формат команды:

BOOTCFG /<операция> [<аргументы>]

Параметры:

/COPY - Создает копию имеющегося элемента списка загрузки в секции [operating systems], для которого можно добавить параметры ОС.

/DELETE - Удаляет существующий элемент списка загрузки в секции [operating systems] файла BOOT.INI. Нужно указать номер удаляемого элемента.

/QUERY - Отображает элементы списка загрузки и их параметры.

/RAW - Позволяет указать любой переключаемый параметр, добавляемый для указанного элемента списка загрузки ОС.

/TIMEOUT - Задает значение таймаута.

/DEFAULT - Задает используемый по умолчанию элемент списка загрузки.

/EMS - Позволяет задавать переключатель /redirect без дисплейной работы для указанного элемента списка загрузки.

/DEBUG - Позволяет задавать порт и скорость для удаленной отладки для указанного элемента списка загрузки.

/ADDSW - Позволяет добавлять определенные переключатели для указанного элемента списка загрузки.

/RMSW - Позволяет удалять определенные переключатели для указанного элемента списка загрузки.

/DBG1394 - Позволяет настраивать отладку 1394 порта для указанного элемента списка загрузки.

/? - Вывод справки по использованию.

Примеры:

```
BOOTCFG /COPY /?
BOOTCFG /DELETE /?
BOOTCFG /QUERY /?
BOOTCFG /RAW /?
BOOTCFG /TIMEOUT /?
BOOTCFG /EMS/?
BOOTCFG /DEBUG /?
BOOTCFG /ADDSW /?
BOOTCFG /RMSW /?
BOOTCFG /DBG1394 /?
BOOTCFG /DEFAULT /?
BOOTCFG /?
```

Настройка автоматически выполняемых программ

Одна из типичных проблем, связанных с производительностью, это запуск большого числа программ в процессе загрузки Windows XP. В результате работа операционной системы существенно замедляется.

В процессе установки программа может быть запущена автоматически следующими способами:

1. Добавление в папку Автозагрузка для данного пользователя
2. Добавление в папку Автозагрузка для всех пользователей
3. Ключ Run (компьютера) Ключ реестра HKLM\Software\Microsoft\Windows\CurrentVersion\Run
4. Ключ Run (пользователя) Ключ реестра HKCU\Software\Microsoft\Windows\CurrentVersion\Run
5. Папки Планировщика задач
6. Win.ini. Программы, предназначенные для 16-разрядных версий Windows могут добавить строки типа Load= и Run= этого файла
7. Ключи RunOnce и RunOnceEx. Группа ключей реестра, содержащая список программ, выполняемых однократно в момент запуска компьютера. Эти ключи могут относиться и к конкретной учетной записи данного компьютера [9] HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
8. Групповая политика. Содержит две политики (с именами Запуск программ при входе пользователя в систему). Находятся в папках Конфигурация компьютера >Конфигурация Windows > Административные шаблоны > Система > Вход в систему (Computer configuration > Administrative Templates >System >Logon) и Конфигурация пользователя >Конфигурация Windows > Административные шаблоны > Система > Вход в систему (User configuration > Administrative Templates >System >Logon)
9. Сценарии входа в систему. Настраиваются Групповая политика: Конфигурация компьютера >Конфигурация Windows > Сценарии и Конфигурация пользователя>Конфигурация Windows >Сценарии (входа в систему и выхода из системы)

Для настройки списка автоматически вызываемых программ в состав Windows XP входит утилита Настройка системы (System Configuration Utility) Msconfig.exe, которая позволяет вывести список всех автоматически загружаемых программ. Рабочее окно программы приведено на рис. 11

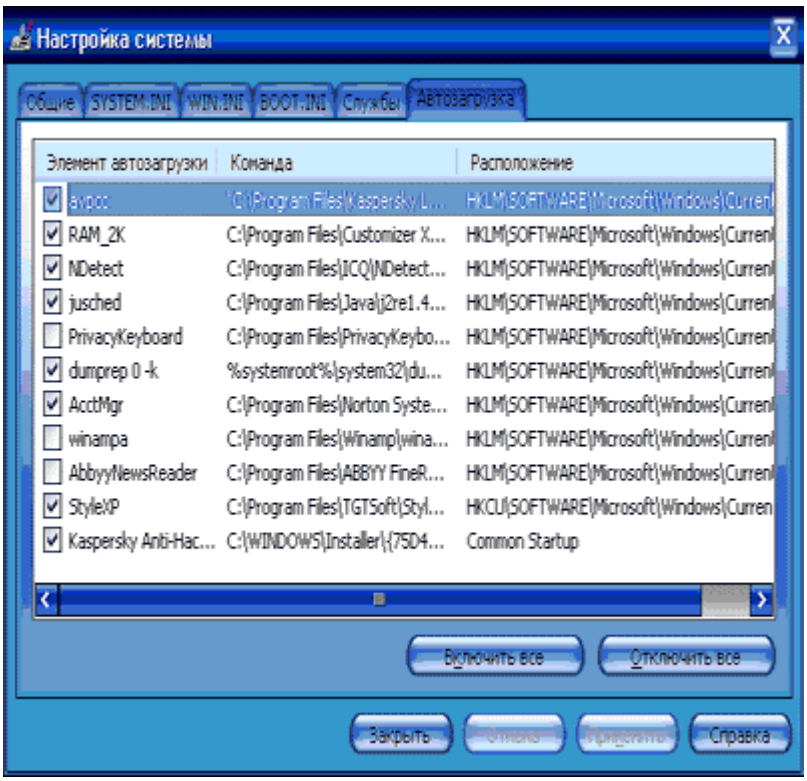


Рис.11 Рабочее окно программы Msconfig

Настройка реестра

Реестр Windows содержит несколько ключей, которые позволят настроить оптимальную работу Windows с памятью. Откройте [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\ ControlSessionManager\MemoryManagement]:ClearPageFileAtShutdown - возможность стирать файл подкачки при выходе из Windows (опция доступна так же в разделе локальной безопасности), при включении приведет к большим задержкам перед перезагрузкой, значение желательно оставить как есть =0.

DisablePagingExecutive - запрещает записывать в файл подкачки код (драйверы, .exe-файлы), и требует оставлять их всегда в физической памяти, если объем памяти больше 256MB установка значения 1 может существенно ускорить работу системы.

LargeSystemCache - этот параметр мы изменяли в Memory usage (см выше).

SecondLevelDataCache - для тех, кто использует старый процессор (до Pentium 2) можно установить размер вашего кэша процессора, значение по умолчанию =0 соответствует 256KB.

Отключение POSIX: Отключение этой не используемой подсистемы может несколько увеличить скорость работы. Чтобы не возиться с удалением файлов и с отключением для этой цели файловой защиты Windows XP откройте [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\ ControlSessionManager\SubSystems] Удалите строки Optional и Posix.

Автоматическая перезагрузка при отказе системы

При отказе системы можно включить автоматическую перезагрузку. Для этого:

- Выбираем Мой компьютер и нажимаем на нем правой клавишей мыши
- Выбираем вкладку Дополнительно
- В разделе Загрузка и восстановление нажимаем кнопку Параметры
- Ставим галочку около пункта Выполнить автоматическую перезагрузку в разделе Отказ системы

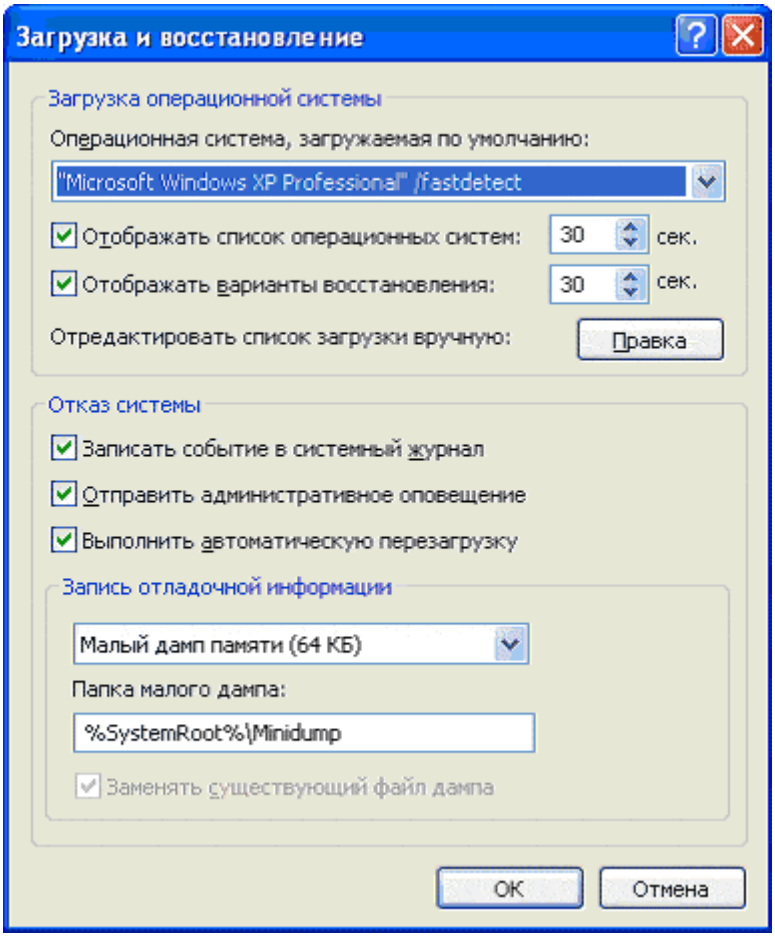


Рис. 12 Панель загрузка и восстановление

Вывод

После того, как вы проведете рекомендуемые операции, вы увидите рост быстродействия вашего компьютера. Однако мы не затронули вопрос настройки системы безопасности вашего компьютера.

Итак, следующий шаг

Настройка системы безопасности Windows XP

Операционная система Windows XP обладает развитой системой безопасности, которая, тем не менее, нуждается в настройке [10]. Мы надеемся, что вы понимаете, что система Windows XP должна устанавливаться на разделах NTFS, что применение файловой системы FAT32 не рекомендуется, исходя из принципов безопасности (встроенные средства безопасности просто не могут быть реализованы при условии применения

FAT32). В случае применения файловой системы FAT 32 почти все утверждения данного раздела теряют для вас всякое значение. Единственный способ включить все разрешения файловой системы - преобразовать диск в формат NTFS.

После чистой установки Windows XP предлагаемые по умолчанию параметры безопасности работают как переключатели типа "включить-выключить". Такой интерфейс носит по умолчанию название Простой общий доступ (Simple File Sharing). Такая конфигурация обладает низким уровнем безопасности, практически совпадающей со стандартной конфигурацией Windows 95/98/Me.

Если вас не устраивает такая конфигурация, вы можете воспользоваться всей мощью разрешений для файлов в стиле Windows 2000. Для этого откройте произвольную папку в Проводнике и выберите Сервис " Свойства папки (Tools " Folder options). Перейдите на вкладку Вид найдите в списке флажок Использовать простой общий доступ к файлам (рекомендуется) (Use File Sharing (recommended) и снимите его. [11]

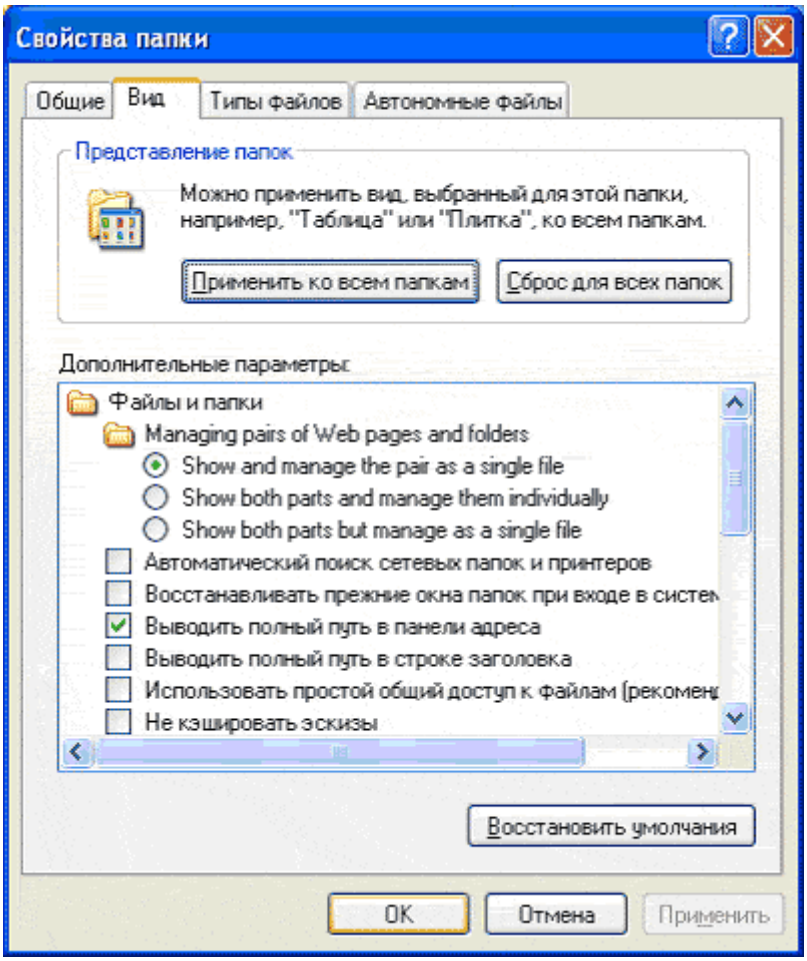


Рис. 11 Свойства папки

Когда вы выключаете простой общий доступ, в диалоговом окне свойств любой папки появляется вкладка Безопасность.

Аналогично осуществляется выдача разрешений на файлы. Все разрешения хранятся в списках управления доступом (Access Control List - ACL).

При установке и удалении разрешений руководствуйтесь следующими основными принципами:

1. Работайте по схеме "сверху-вниз".
2. Храните общие файлы данных вместе.
3. Работайте с группами везде, где это только возможно.
4. Не пользуйтесь особыми разрешениями.
5. Не давайте пользователям большего уровня полномочий, чем это абсолютно необходимо (принцип минимизации полномочий).

Установка разрешения из командной строки

Утилита командной строки cacls.exe доступна в Windows XP Professional позволяет просматривать и изменять разрешения файлов и папок. Cacls - сокращение от Control ACLs - управление списками управления доступом. Ключи командной строки утилиты cacls

Таблица 1

Ключ	Действие
/T	Смена разрешений доступа к указанным файлам в текущей папке и всех подпапках
/E	Изменение списка управления доступом (а не полная его замена)
/C	Продолжить при возникновении ошибки "отказано в доступе"
/G пользователь:разрешение	Выделение пользователю указанного разрешения. Без ключа /E полностью заменяет текущие разрешения
/R пользователь	Отменяет права доступа для текущего пользователя (используется только с ключом /E)
/P пользователь:разрешение	Замена указанных разрешений пользователя
/D пользователь	Запрещает пользователю доступ к объекту

С ключами /G и /P нужно использовать одну их перечисленных ниже букв (вместо слова разрешение):

- F (полный доступ) - эквивалентно установке флажка Разрешить полный доступ (Full Control) на вкладке Безопасность.
- C (изменить) - тождественно установке флажка Разрешить Изменить (Modify)
- R (чтение) - эквивалентно установке флажка Разрешить Чтение и выполнение (Read & Execute)
- W (запись) - равнозначно установке флажка Разрешить запись (Write)

Microsoft Windows XP позволяет предотвратить попадание конфиденциальных данных в чужие руки. Шифрующая файловая система (Encrypting File System - EFS) шифрует файлы на диске. Однако, слудует иметь ввиду, что если вы утеряете ключ для расшифровки, данные можно считать утерянными. Поэтому если вы решите воспользоваться преимуществанми EFS необходимо создать учетную запись агента восстановления, резервную копию собственного сертификата и сертификата агента восстановления.

Если вы предпочитаете работать с командной строкой, то можете воспользоваться программой cipher.exe. Команда cipher без параметров выводит информацию о текущей папке и размещенных в ней файлах (зашифрованы они или нет). В таблице 2 приведен список наиболее часто используемых ключей команды cipher

Таблица 2

Ключ	Описание
/E	Шифрование указанных папок
/D	Расшифровка указанных папок
/S:папка	Операция применяется к папке и всем вложенным подпапкам (но не файлам)
/A	Операция применяется к указанным файлам и файлам в указанных папках
/K	Создание нового ключа шифрования для пользователя, запустившего программу. Если этот ключ задан, все остальные игнорируются
/R	Создание ключа и сертификата агента восстановления файлов. Ключ и сертификат помещаются в файл .CFX, а копия сертификата в файле .CER
/U	Обновление ключа шифрования пользователя или агента восстановления для всех файлов на всех локальных дисках
/U /N	Вывод списка всех зашифрованных файлов на локальных дисках без каких-либо других действий

Агентом восстановления данных (Data Recovery Agent) назначается обычно администратор. Для создания агента восстановления нужно сначала сохдать сертификат восстановления данных, а затем назначить одного из пользователей таким агентом.

Чтобы создать сертификат нужно сделать следующее:

1. Нужно войти в систему под именем Администратор
2. Ввести в командной строке cipher /R: имя файла
3. Введите пароль для вновь создаваемых файлов

Файлы сертификата имеют расширение .PFX и .CER и указанное вами имя.

ВНИМАНИЕ эти файлы позволяют любому пользователю системы стать агентом восстановления. Обязательно скопируйте их на дискету и храните в защищенном месте. После копирования удалите файлы сертификата с жесткого диска.

Для назначения агента восстановления:

1. Войти в систему под учетной записью, которая должна стать агентом восстановления данных
2. В консоли Сертификаты перейдите в раздел Сертификаты - Текущий пользователь " Личные (Current User " Personal)
3. Действие " Все задачи " Импорт (Actions " All Tasks " Import) для запуска мастера импорта сертификатов
4. Проведите импорт сертификата восстановления

При неправильном использования средств шифрования вы можете получить больше вреда, чем пользы.

Краткие рекомендации по шифрованию:

1. Зашифруйте все папки, в которых вы храните документы
2. Зашифруйте папки %Temp% и %Tmp%. Это обеспечит шифрование всех временных файлов
3. Всегда включайте шифрование для папок, а не для файлов. Тогда шифруются и все создаваемые в ней впоследствии файлы, что оказывается важным при работе с программами, создающими свои копии файлов при редактировании, а затем перезаписывающими копии поверх оригинала
4. Экспортируйте и защитите личные ключи учетной записи агента восстановления, после чего удалите их с компьютера
5. Экспортируйте личные сертификаты шифрования всех учетных записей

6. Не удаляйте сертификаты восстановления при смене политик агентов восстановления. Храните их до тех пор, пока не будете уверены, что все файлы, защищенные с учетом этих сертификатов, не будут обновлены.
7. При печати не создавайте временных файлов или зашифруйте папку, в которой они будут создаваться
8. Защитите файл подкачки. Он должен автоматически удаляться при выходе из Windows

Конструктор шаблонов безопасности

Шаблоны безопасности являются обыкновенными ASCII - файлами, поэтому теоретически их можно создавать с помощью обыкновенного текстового редактора. Однако лучше воспользоваться оснасткой Security Templates консоли Microsoft Management Console (MMC). Для этого в командной строке нужно ввести mmc /a в этой консоли выбрать меню File - Add/Remove. В диалоговом окне Add Standalone Snap-in выбрать Security Templates - Add.

Управление оснасткой

Шаблоны безопасности расположены в папке \%systemroot%\security\templates. Количество встроенных шаблонов изменяется в зависимости от версии операционной системы и установленных пакетов обновлений.

Если раскрыть любую папку в Security Templates, то в правой панели будут показаны папки, которые соответствуют контролируемым элементам:

- Account Policies - управление паролями, блокировками и политиками Kerberos
- Local Policies - управление параметрами аудита, пользовательскими правами и настройками безопасности
- Event Log - управление параметрами системного журнала
- Restricted Groups - определение элементов различных локальных групп
- System Services - включение и отключение служб и присвоение права модификации системных служб
- Registry - назначение разрешений на изменение и просмотр разделов реестра
- File System - управление разрешениями NTFS для папок и файлов

Защита подключения к Интернет

Для обеспечения безопасности при подключении к Интернет необходимо:

- Активизировать брандмауэр подключения к Интернет (Internet Connection Firewall) или установить брандмауэр третьих фирм
- Отключить Службу доступа к файлам и принтерам сетей Microsoft

Брандмауэром подключения к Интернет называется программный компонент, блокирующий нежелательный трафик.

Активация Брандмауэра подключения к Интернет.

- Откройте Панель управления - Сетевые подключения
- Щелкните правой кнопкой мыши на соединении, которое вы хотите защитить и выберите из меню пункт Свойства
- Перейдите на вкладку Дополнительно, поставьте галочку Защитить мое подключение к Интернет

Управление задачами и процессами

PRINT

Для каждого приложения, запускаемого под управлением Microsoft Windows, операционная система создает собственную задачу, иными словами, отводит определенный объем оперативной памяти для выполнения этой программы и контролирует работу загруженного в память приложения с ресурсами компьютера. Помимо задач в Windows XP имеются также процессы - виртуальное адресное пространство памяти, отведенное для выполнения программой или самой операционной системой каких-либо процедур. Одна задача может активизировать в Windows несколько различных процессов: например, web-браузер может одновременно обращаться к порту модема для получения и отправки каких-либо данных и отображать на экране результат работы встроенного в web-страницу апплета Java. Каждому процессу автоматически назначается индивидуальный опознавательный номер, так называемый Process ID или PID, предназначенный для однозначной идентификации процесса в системе.

Если запущенная вами программа неожиданно вызвала «зависание» компьютера, в большинстве случаев нет необходимости прибегать к аварийной перезагрузке: вполне достаточно отыскать в памяти вызвавшую сбой задачу и снять ее, то есть принудительно прекратить ее дальнейшее выполнение. Возможна и другая ситуация: закрыв неожиданно «зависшую» программу путем снятия задачи, вы не сможете продолжить прерванную работу с каким-либо файлом или документом. Например, удалив из памяти компьютера Microsoft Word, вы не сможете снова загрузить в него текст, который только что редактировали, поскольку операционная система считает, что этот документ уже используется другой программой. Подобные явления происходят потому, что после снятия задачи в памяти все еще остался инициированный этой задачей процесс - в нашем случае это процесс обработки документа Word. Прекратив выполнение вызывающего сбои процесса, вы сможете продолжить работу с Windows XP в обычном режиме.

Для управления задачами и процессами в Microsoft Windows XP предусмотрена специальная системная утилита, называемая Диспетчер задач (Windows Task Manager), окно которой появляется на экране при нажатии сочетания клавиш Ctrl+Alt+Del (рис. 19.1).

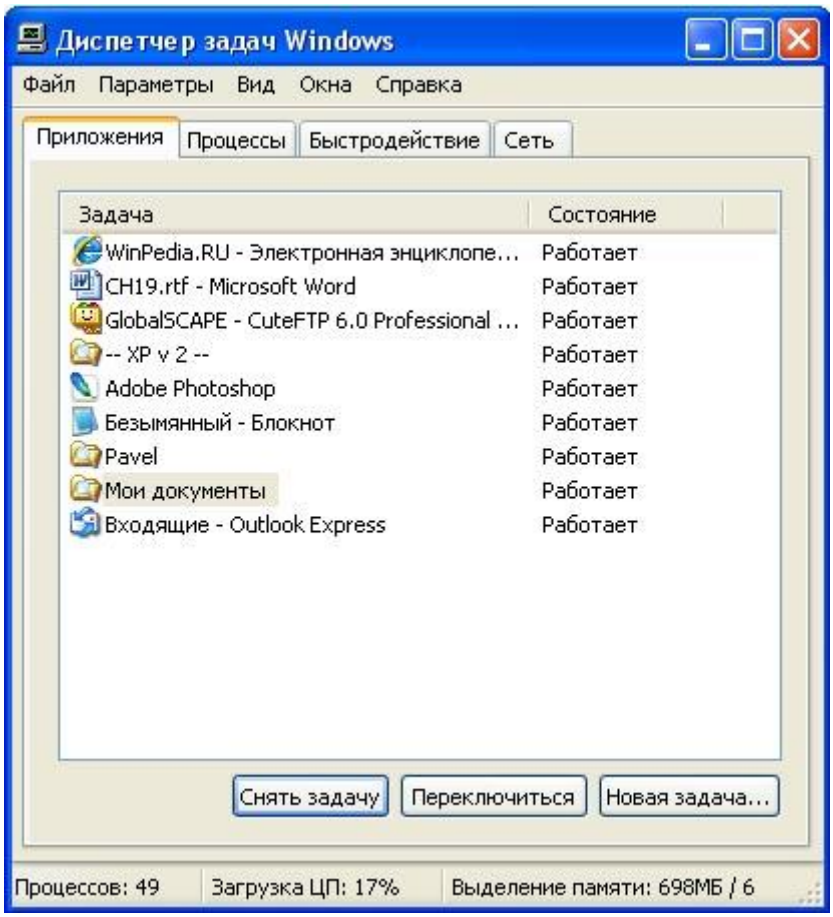


Рис. 19.1. Диспетчер задач Windows

Окно утилиты Диспетчер задач Windows имеет пять функциональных вкладок. Вкладка Приложения (Applications) содержит список всех запущенных в системе задач: в поле Задача (Task) отображается название задачи, в поле Состояние (Status) - ее текущее состояние. Нормальным статусом задачи является состояние Работает (Running). Если задача «зависла» и не отвечает на системные запросы, ее состояние будет определено как Не отвечает (Not responding). Чтобы снять одну из задач, выделите ее щелчком мыши в меню Задача (Task) и нажмите на кнопку Снять задачу (End Task). Чтобы переключиться к какой-либо задаче, то есть открыть на экране ее окно, выделите в списке нужную задачу и щелкните мышью на кнопке Переключиться (Switch To). Вы можете инициировать новую задачу, нажав на кнопку Новая задача (New Task) и указав полное имя и путь к запускаемой программе в соответствующем поле открывшегося окна либо определив эту программу визуально при помощи кнопки Обзор (Browse).

Вкладка Процессы (Processes) дает возможность управлять запущенными в системе процессами.

В меню на данной вкладке отображаются названия процессов - Имя образа (Image Name), определители инициаторов процессов - Имя пользователя (User Name), характеристики аппаратной обработки процессов - ЦП (CPU) и занимаемый процессами объем оперативной памяти - Память (Memory Usage). Вы можете включить отображение других характеристик процессов (например, PID, диапазон ввода-вывода, объем используемого процессом кэша и т. д.), воспользовавшись функцией Выбрать столбцы (Select Columns) командного меню Вид (View). Чтобы прекратить выполнение процесса, выделите его заголовок щелчком мыши и нажмите на кнопку Завершить процесс (End Process). Если вы хотите, чтобы в данном меню отображались процессы, инициированные всеми пользователями вашей системы, установите флажок рядом с пунктом Отображать процессы всех пользователей (Show processes from all users). Вы можете также установить приоритет процесса, если щелкните на его заголовке правой кнопкой мыши, выберите в появившемся меню пункт Приоритет (Set Priority) и укажете назначаемый для данного процесса режим выполнения:

- Реального времени (Realtime)- режим реального времени (все инициируемые процессом запросы выполняются системой по мере поступления);
- Высокий (High)- высочайший приоритет;
- Выше среднего (AboveNormal)- высокий приоритет;
- Средний (Normal)- стандартный приоритет;
- Ниже среднего (BelowNormal)- низкий приоритет;
- Низкий (Low)- наиболее низкий приоритет.

Чем выше приоритет процесса, тем быстрее выполняются инициированные им запросы. Процессам с высоким приоритетом система предоставляет аппаратные и программные ресурсы в первую очередь. При перераспределении приоритетов процессов следует помнить, что если вы установите высокий приоритет какому-либо второстепенному процессу, приоритет одного из жизненно важных для Windows системных процессов может автоматически стать низким, в результате чего скорость работы системы в целом заметно снизится или выполнение этого процесса будет полностью блокировано, что приведет к «зависанию» компьютера. Устанавливать высокий приоритет одному из процессов следует только в том случае, если его выполнение с «нормальным» приоритетом по каким-то причинам затруднено или невозможно.

Вкладка Быстродействие (Performance) окна Диспетчера задач Windows содержит информацию о загрузке процессора, оперативной памяти и об использовании других аппаратных ресурсов компьютера (рис. 19.2).

В частности, индикаторы Загрузка ЦП (CPU Usage) и Хронология загрузки ЦП (CPU Usage History) показывают нагрузку на процессор вашего компьютера, индикаторы Файл подкачки (PF Usage) и Хронология использования файла подкачки (PF Usage History) - использование файла подкачки при системном кэшировании данных, индикатор Физическая память (Physical Memory) - степень загрузки оперативной памяти в килобайтах.

Аналогичным образом вкладка Сеть (Networking) демонстрирует нагрузку на вашу локальную сеть. И наконец, если вы хотите просмотреть список всех работающих в настоящее время с вашей системой пользователей, перейдите ко вкладке Пользователи (Users).

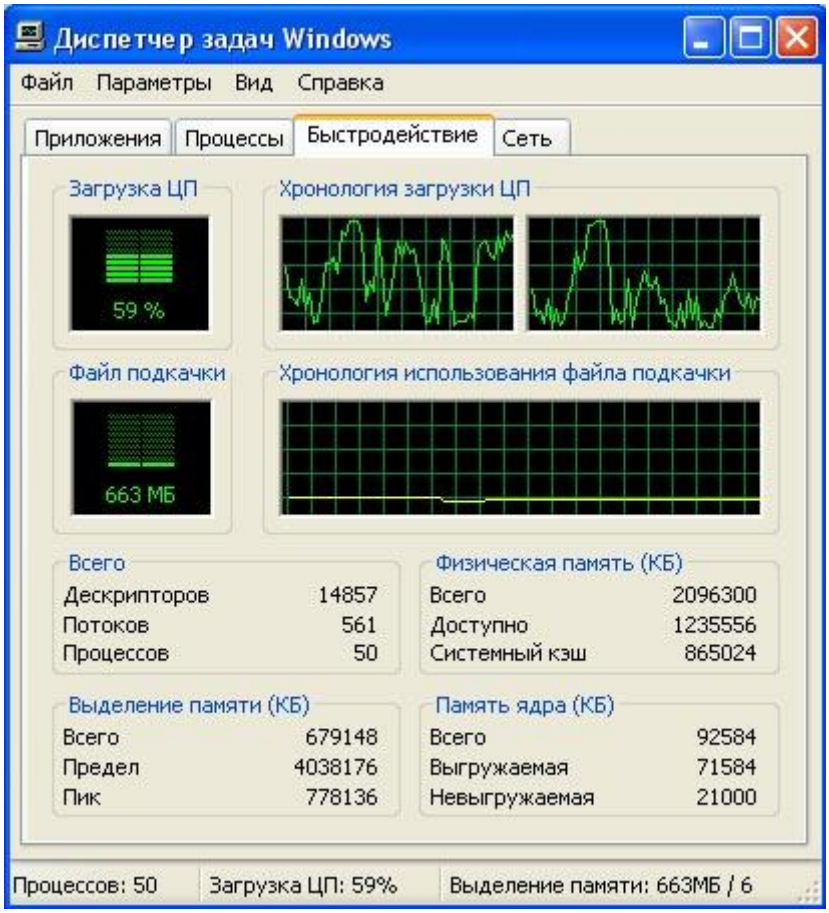


Рис. 19.2. Вкладка Быстродействие окна Диспетчер задач Window

Если вы зарегистрированы в системе как администратор данного компьютера, то можете отключить одного из пользователей, выделив в меню название его сеанса работы с Windows щелчком мыши и нажав на кнопку Отключить (Disconnect). Помимо этого вы можете отправить ему сообщение нажатием на кнопку Отправить сообщение (Send Message). Чтобы завершить текущий сеанс работы с Windows, щелкните мышью на кнопке Выйти из системы (Logoff).

Учетные записи пользователей Windows

Если в разное время с компьютером работает более одного пользователя, у вас может возникнуть необходимость регистрации в системе новой учетной записи для входа в Windows. Поскольку Microsoft Windows XP является многопользовательской операционной системой, различные пользователи, имеющие собственные учетные записи, могут независимо друг от друга настраивать интерфейс Рабочего стола и изменять Темы Windows, работать с собственными файлами и папками (другие пользователи Windows не смогут получить к ним доступ), настраивать собственный набор разрешенных для запуска программ, а также пользоваться независимыми настройками доступа в Интернет и к электронной почте. Регистрация учетной записи с ограниченными возможностями позволит системному администратору допускать к компьютеру неопытных пользователей, запретив им устанавливать новое программное обеспечение, изменять те настройки системы, которые могут повлиять на ее работоспособность, а также запускать некоторые программы.

В Microsoft Windows XP можно зарегистрировать произвольное количество пользователей, причем каждый из них может принадлежать к одной из двух стандартных категорий: Администратор компьютера (Computer Administrator)

или Ограниченная запись (Limited User). Пользователь Windows XP, зарегистрированный в системе как администратор компьютера, обладает следующими правами:

- установка оборудования и программного обеспечения;
- изменение всех системных настроек;
- доступ ко всем файлам, кроме индивидуальных файлов других пользователей;
- создание, удаление и изменение учетных записей пользователей;
- изменение статуса и параметров собственной учетной записи;
- изменение прав доступа других пользователей к ресурсам компьютера.

Пользователь, учетная запись которого относится к категории Ограниченная запись (Limited User), может только изменять собственный пароль для входа в систему и графическое изображение, с помощью которого отображается его учетная запись. Он может также пользоваться теми правами, которые установил для него администратор компьютера.

Создание учетной записи

Войдите в Windows как администратор компьютера и выполните следующие команды: Пуск->Панель управления->Учетные записи пользователей (Start->Control Panel->User Accounts). В появившемся окне Учетные записи пользователей (User Accounts) щелкните мышью на пункте Создание учетной записи (Create a New Account), рис. 19.3.

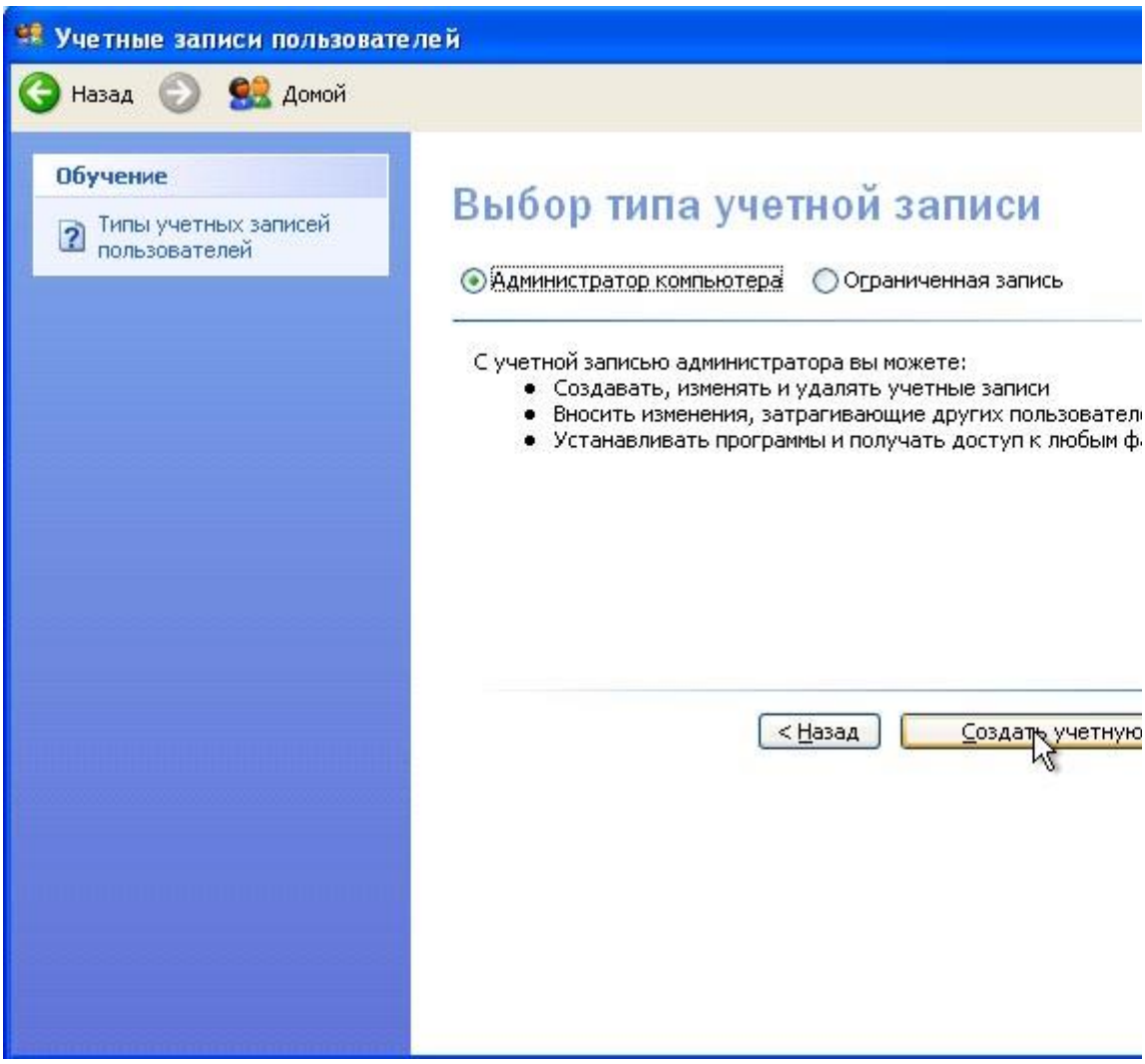


Рис. 19.3. Создание новой учетной записи пользователя Windows XP

В поле Введите имя учетной записи (Type a name for the new account) наберите название новой учетной записи и щелкните мышью на кнопке Далее (Next). Далее укажите тип создаваемой учетной записи - Администратор компьютера (Computer Administrator) или Ограниченная запись (Limited User).

Теперь вам останется только щелкнуть на кнопке Создать учетную запись (Create Account), чтобы создать новую учетную запись пользователя Windows.

Изменение параметров и удаление учетной записи

Войдите в Windows как администратор компьютера и выполните следующие команды: Пуск->Панель управления->Учетные записи пользователей (Start->Control Panel->User Accounts). В появившемся окне Учетные записи пользователей (User Accounts) щелкните на пункте Изменение учетной записи (Change an Account). Вам будут предъявлены значки всех зарегистрированных в системе на данный момент учетных записей пользователей - выберите ту из них, параметры которой вы хотите изменить.

В следующем окне вам предстоит выбрать действие, которое следует произвести

с данной учетной записью.

Изменить название учетной записи

Для этого воспользуйтесь функцией Изменение имени (Change the name) и введите в поле Введите новое имя пользователя (Type a new name for user) новое название учетной записи.

Создать или изменить пароль для входа в систему

Чтобы создать для данного пользователя новый пароль (изменить текущий), воспользуйтесь функцией Создание пароля (Create a password), затем введите в поле Введите новый пароль (Type a new password) новый пароль. Для проверки наберите его еще раз в поле Введите пароль для подтверждения (Type a new password again to confirm), затем укажите в поле Введите слово или фразу, служащую подсказкой о пароле (Type a word or phrase to use as a password hint) любое запоминающееся слово или фразу, которые пользователь сможет сообщить системе в случае, если он забыл пароль.

Изменить тип учетной записи

Воспользуйтесь функцией Изменение типа учетной записи (Change the account type), в предложенном меню выберите тип учетной записи - Администратор компьютера (Computer Administrator) или Ограниченная запись (Limited User) - и щелкните на кнопке Изменить тип учетной записи (Change account type).

Изменить значок учетной записи

Выберите пункт Изменение изображения (Change the Picture). На экране появится меню, содержащее изображения всех имеющихся в Windows XP стандартных значков. Выделите щелчком мыши любой из них и нажмите на кнопку Изменить рисунок (Change Picture), рис. 19.4.

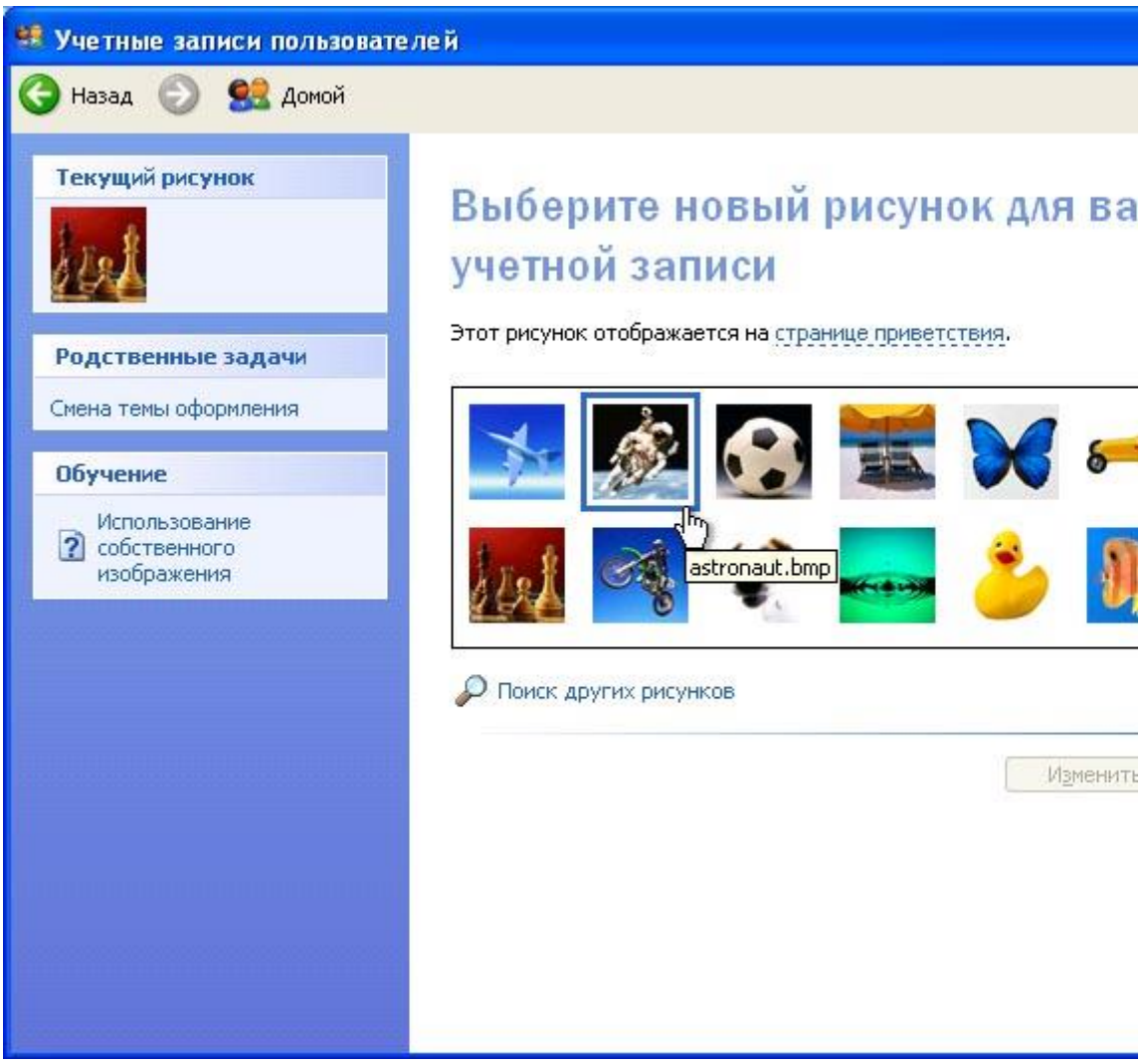


Рис. 19.4. Выбор значка для отображения учетной записи пользователя

Если в качестве значка для своей учетной записи вы хотите использовать нестандартное изображение (например, собственную фотографию), воспользуйтесь функцией Поиск других рисунков (Browse for more pictures) и укажите системе необходимый графический файл.

Удалить пароль

Для этого выберите пункт Удалить пароль (Remove the password) и нажмите в следующем окне кнопку Удалить пароль (Remove Password).

Удалить учетную запись

Для этого выберите пункт Удалить учетную запись (Delete the account) и нажмите в следующем окне кнопку Удалить (Delete Account).

Изменение механизма входа в систему

Вы можете изменить механизм входа в систему для пользователей Windows XP с помощью функции Изменение входа пользователей в систему (Change the way

users log on or off).

Если вы установите флажок рядом с пунктом Использовать страницу приветствия (Use the Welcome screen), при загрузке компьютера или завершении текущего сеанса работы с Windows на экране будет отображаться стартовое окно Windows XP со значками учетных записей всех зарегистрированных в системе пользователей. Если функция отключена, вход в систему будет осуществляться с использованием стандартного механизма Windows NT: пользователь должен будет ввести в специальное окно название собственной учетной записи и пароль вручную.

ПРИМЕЧАНИЕ

Если функция Use the Welcome screen отключена, вы не сможете переключаться между разными сеансами работы в Windows XP с сохранением всех открытых программ и редактируемых файлов. В этом случае при выборе пункта Выход из системы (Log Off) все загруженные приложения будут автоматически закрыты.

Если функция Использовать страницу приветствия (Use the Welcome screen) включена, вы можете воспользоваться возможностью быстрого переключения между разными сеансами работы в Windows XP с сохранением всех запущенных приложений и редактируемых файлов. Например, прервав редактирование документа Microsoft Word, вы можете ненадолго отлучиться от компьютера; в это время другой пользователь может занять ваше место за клавиатурой и начать собственный сеанс работы с Windows XP. Вернувшись на свое рабочее место, вы сможете продолжить редактирование текста с того места, где оно было прервано. Чтобы активизировать функцию быстрого переключения между сеансами работы с системой, установите флажок рядом с пунктом Использовать быстрое переключение пользователей (Use Fast User Switching). Сохраните внесенные в настройки изменения, щелкнув на кнопке Применение параметров (Apply Options).

Резервное копирование данных

Перед возможной переустановкой Microsoft Windows XP, а также в общих целях безопасности рекомендуется периодически производить резервное копирование данных, хранящихся на ваших дисках.

Для того чтобы создать резервную копию данных на каком-либо внешнем носителе информации (например, стример, устройство для записи компакт-дисков, магнитооптический накопитель, накопитель ZIP, DVD, дискета), необходимо выполнить команды Пуск->Все программы->Стандартные->Служебные->Архивация данных (Start->All Programs->Accessories->System Tools->Backup). На экране появится окно мастера Архивация или восстановление (Backup and Restore Wizard).

Щелкните мышью на кнопке Далее (Next). Если вы хотите создать резервную копию своих файлов и папок, в следующем окне установите переключатель в режим Архивация файлов и параметров (Back up files and settings), а если восстановить сохраненные данные - выберите режим Восстановление файлов и параметров (Restore files and settings). Снова нажмите на кнопку Далее (Next).

После этого предстоит выбрать, какие именно данные вы планируете включить в резервную копию. Доступно несколько режимов:

- Мои документы и параметры настройки (My Documents and settings)- архивированию подлежат системные папки Мои документы (My Documents) и Избранное (Favorites) вместе со всем их содержимым, а также настройки Рабочего стола и загруженные из Интернета файлы cookies;
- Документы и параметры настройки всех пользователей данного компьютера

(Everyone's Documents and settings)- архивируются системные папки Мои документы (My Documents) и Избранное (Favorites), настройки Рабочего стола и загруженные из Интернета файлы cookies всех зарегистрированных в системе пользователей Windows XP;

- Всю информацию на данном компьютере (All information on this computer)- создается резервная копия всей хранящейся на компьютере информации, включая системные файлы, которые помещаются на специальный загрузочный диск, позволяющий восстановить Windows в случае полного крушения системы;
- Предоставить возможность выбора объектов для архивации (Let me choose want to backup)- пользователь сам выбирает, какие данные подлежат архивированию.

При выборе режима Предоставить возможность выбора объектов для архивации (Let me choose want to backup) в следующем окне вам предстоит указать программе, какие именно файлы и папки следует включить в состав резервной копии (рис. 19.7).

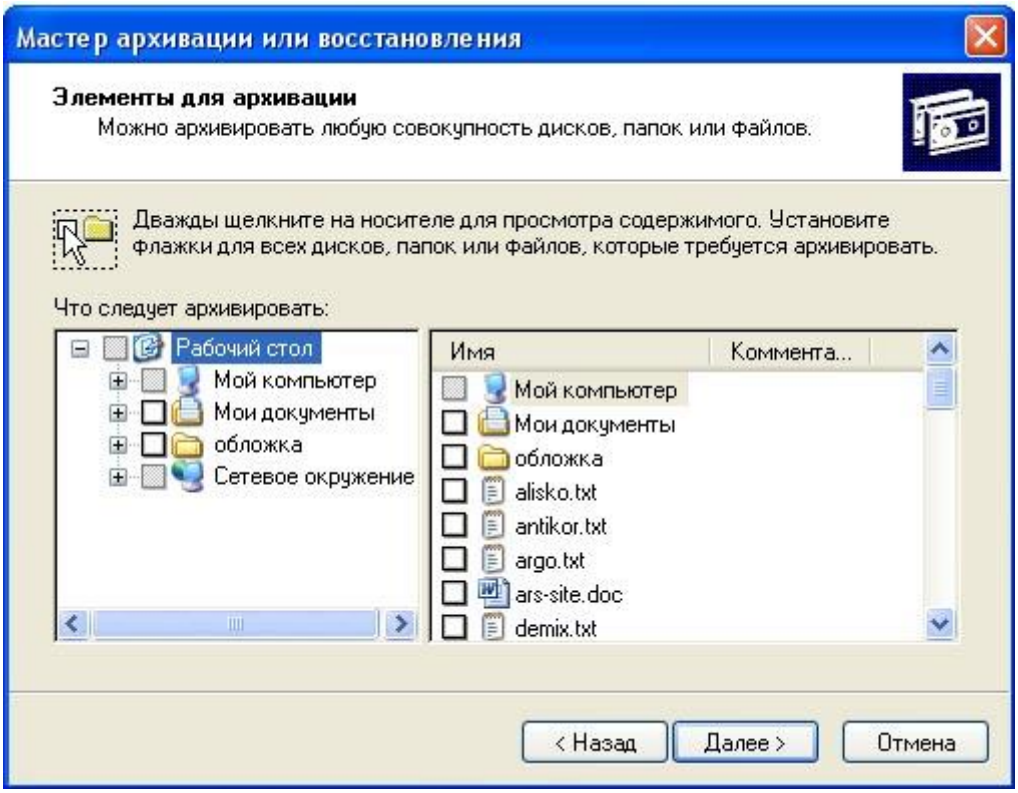


Рис. 19.7. Выбор информации, подлежащей резервному копированию

Щелчком мыши отметьте в левом окне ресурс, на котором расположены подлежащие резервному копированию данные (например, один из жестких дисков компьютера), и в правом окне отобразится содержимое этого ресурса. Установите флажки напротив тех файлов и папок, которые вы хотите включить в резервную копию, и нажмите на кнопку Далее (Next).

Теперь вам предстоит выбрать из предложенного списка носитель, на котором будет создана резервная копия.

Если вы хотите создать резервный архив в другом логическом разделе своего жесткого диска, щелкните мышью на кнопке Обзор (Browse) и укажите программе диск и папку, в которой следует разместить архив. Наберите название создаваемого архива в поле Введите имя для данного архива (Type a name for this backup) и щелкните на кнопке Далее (Next).

По нажатию на кнопку Готово (Finish) вы покинете окно мастера Архивация или восстановление, и начнется резервное копирование.

После того как архивирование информации будет завершено, система откроет специальное

окно, содержащее подробный отчет о проделанной работе.

В случае восстановления созданного ранее резервного архива выберите в окне мастера Архивация или восстановление функцию Восстановление файлов и параметров (Restore files and settings) и нажмите на кнопку Далее (Next). В левом окне мастера появится список всех сделанных вами в разное время резервных копий файлов. Щелкнув мышью на одном из пунктов данного списка, выберите в правом окне диск, на который следует восстанавливать данные.

Щелкните мышью на кнопке Далее (Next), а в следующем окне - на кнопке Готово (Finish). Содержимое вашего архива будет автоматически восстановлено.

Аварийное восстановление системы

Microsoft Windows XP имеет специальный механизм Восстановление системы (System Restore), позволяющий восстановить систему в случае любых непредвиденных сбоев и повреждений. Благодаря этой возможности Windows стала весьма надежной и стабильной платформой, которой не грозит полное разрушение из-за неправильной работы прикладных программ или драйверов.

Установив все необходимые вам приложения, настроив аппаратную конфигурацию компьютера и убедившись в том, что Windows работает корректно и стабильно, вы можете создать так называемую точку восстановления системы (restore point), воспользовавшись утилитой Восстановление системы (System Restore). В этот момент System Restore автоматически создаст резервную копию системного реестра Windows и целого ряда необходимых для работы Windows служебных файлов, сделав своеобразный «моментальный снимок» операционной системы. Если впоследствии какая-либо из установленных вами программ или один из обновленных драйверов оборудования начнет вызывать программные сбои при работе Windows XP, в любую минуту вы сможете вернуться к точке восстановления системы, воссоздав конфигурацию Windows такой, какой она была на момент создания restore point. Подготовив несколько точек восстановления, соответствующих различным событиям в вашей операционной системе (например, установка крупных программных пакетов, игр, появление нового устройства), с помощью программы Восстановление системы вы сможете выбрать любой вариант аварийного «возврата» к предыдущим настройкам Windows.

Чтобы запустить программу Восстановление системы (System Restore), необходимо выполнить следующие команды: Пуск->Все программы->Стандартные->Служебные->Восстановление системы (Start->All Programs->Accessories-> System Tools->System Restore). Интерфейс этой утилиты показан на рис. 19.8.

В первую очередь вам необходимо создать точку восстановления системы. Установите переключатель в верхней части окна в положение Создать точку восстановления (Create a restore point) и щелкните на кнопке Далее (Next). Затем введите в поле Описание контрольной точки восстановления (Restore point description) произвольное имя создаваемой вами точки восстановления и нажмите на кнопку Создать (Create). Все необходимые для восстановления Windows системные файлы будут скопированы.

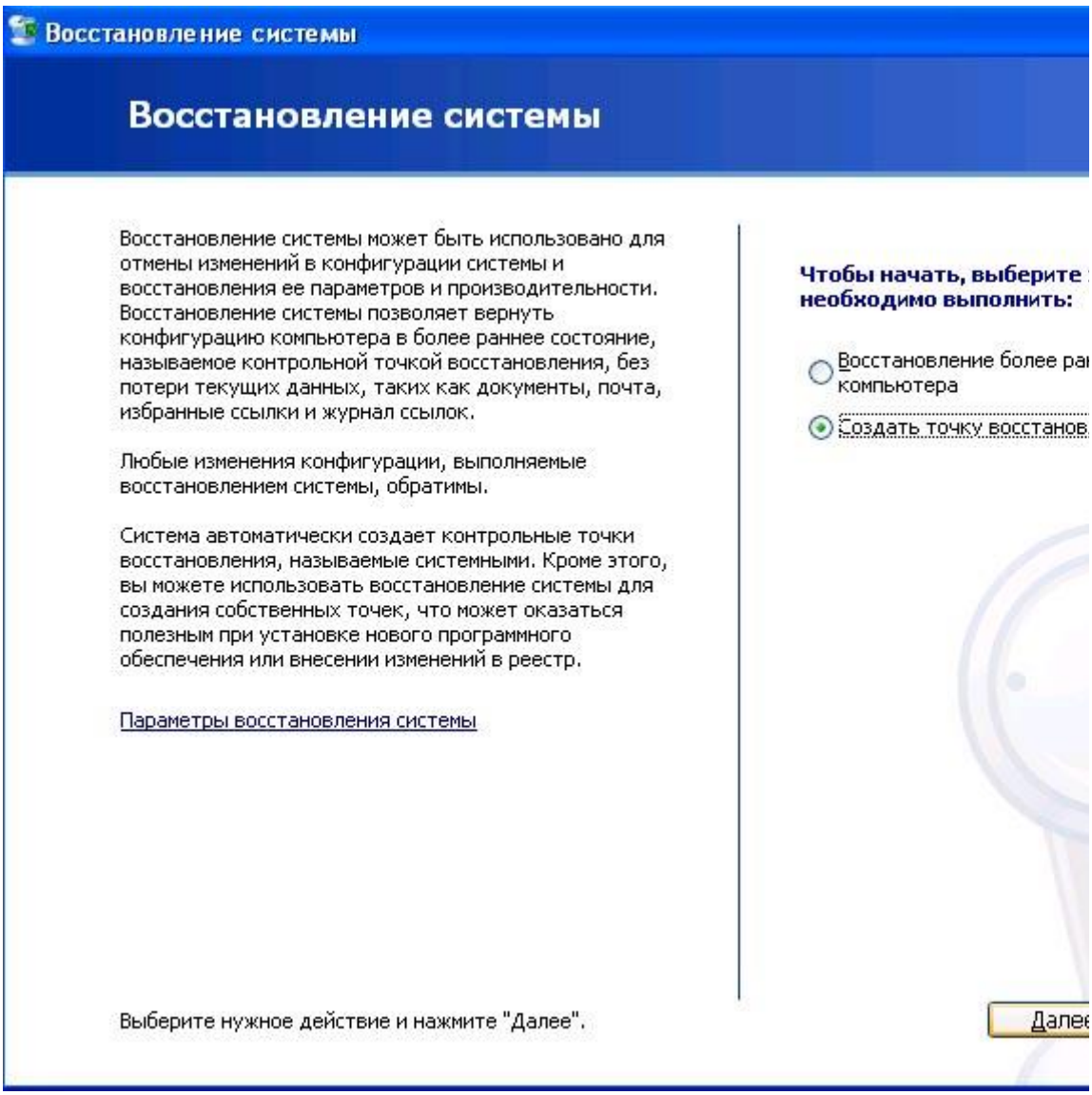


Рис. 19.8. Утилита аварийного восстановления системы

Нажмите на кнопку Закрыть (Close), чтобы покинуть окно Восстановление системы.

Для восстановления поврежденной системы выберите в первом окне программы Восстановление системы режим Восстановление более раннего состояния компьютера (Restore my computer to an earlier time) и нажмите на кнопку Далее (Next). В левой части следующего окна вы увидите изображение календаря, в таблице которого жирным шрифтом выделены даты создания точек восстановления системы (рис. 19.9).

Переключать в календаре месяцы можно щелчком мыши на кнопках с изображением правой и левой стрелки. Выбрав нужный месяц, щелкните на дате, когда была создана нужная вам точка восстановления. В расположенном правее поле вы увидите время ее создания и краткий текстовый комментарий. Можно пролистать список точек восстановления, щелкая на кнопках с правой и левой стрелками над описанием точек восстановления. Выбрав нужную точку восстановления, нажмите на кнопку Далее (Next). Программа Восстановление системы продемонстрирует вам информацию о выбранной точке восстановления системы.

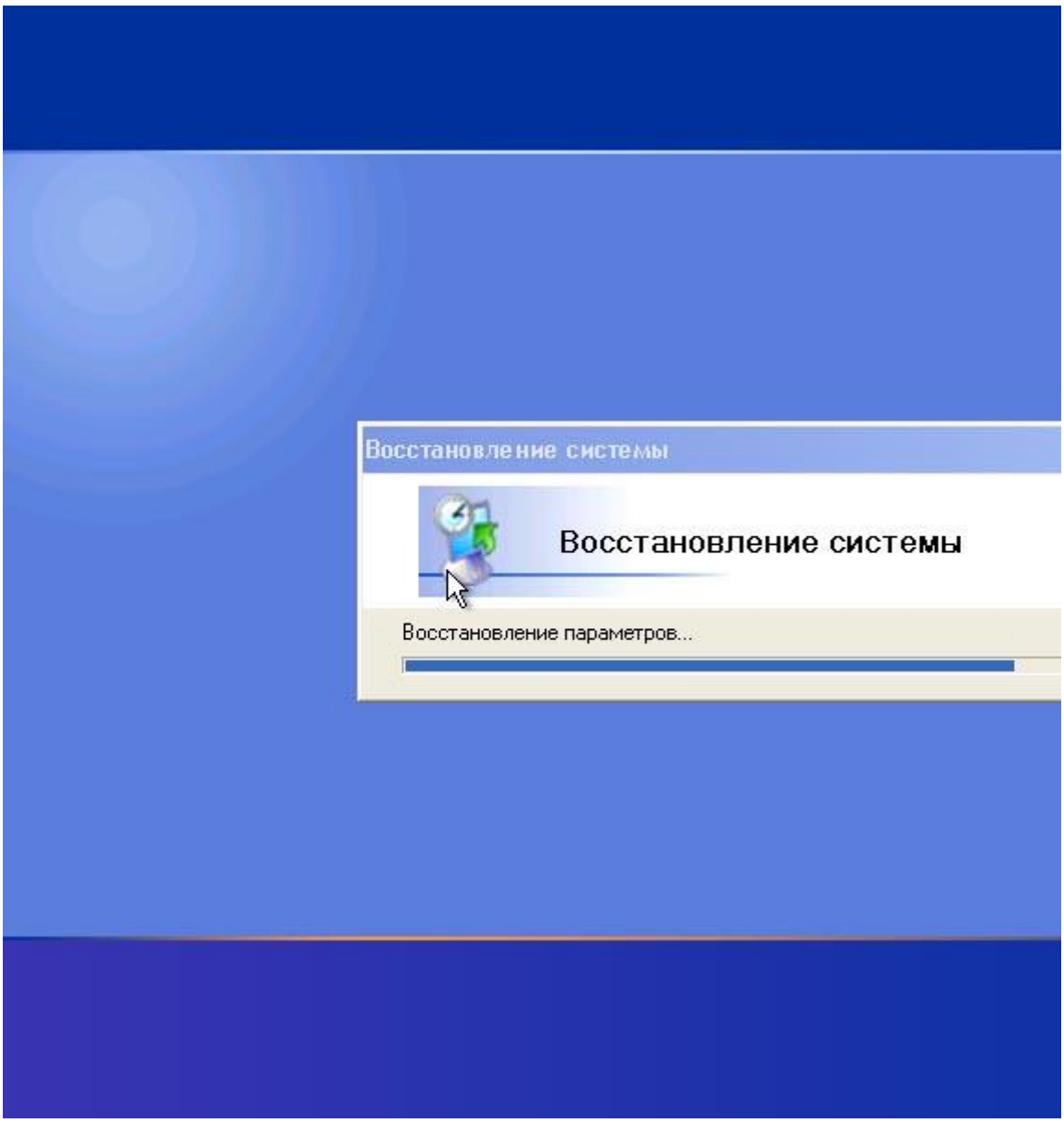


Рис. 19.9. Аварийное восстановление Windows XP

Щелкните на кнопке Далее (Next), чтобы начать восстановление системы. В течение нескольких секунд программа Восстановление системы считает с диска всю необходимую информацию, после чего компьютер будет перезагружен. После завершения процесса реанимации Windows загрузится в штатном режиме с полностью восстановленными настройками.

Восстановление системы не означает, что после перезагрузки компьютера вы потеряете подготовленные вами за последнее время документы или сообщения электронной почты, однако программы, установленные после даты создания точки восстановления, могут не запускаться, так что вам потребуется переустановить их заново.

Если крушение системы произошло раньше, чем вы создали первую точку восстановления, не стоит отчаиваться: утилита Восстановление системы автоматически создает точки восстановления с определенной периодичностью и обязательно - после установки в Windows XP каждой новой программы. Первая точка восстановления генерируется программой Восстановление системы сразу после установки и активации Windows XP.

Несмотря на все удобства и преимущества программы Восстановление системы, мне хотелось бы искренне пожелать вам, чтобы вы пользовались ею как можно реже, поскольку любой сбой, возникающий в работе операционной системы, - это всегда экстраординарное и крайне неприятное событие. Пусть Microsoft Windows XP окажется для вас самой стабильной и надежной операционной системой, а работа с ней всегда доставляет удовольствие.

Управление загрузкой системы при помощи программы Настройка системы

Программа Настройка системы (System Configuration Utility) позволяет гибко управлять параметрами загрузки Windows XP, конфигурировать мультизагрузчик, управлять программами, автоматически запускающимися одновременно с Windows по команде из системного реестра. Для того чтобы запустить программу Настройка системы, откройте окно Запуск программы (Run), выполнив последовательность команд Пуск->Выполнить (Start->Run), введите в поле Открыть (Open) команду msconfig, и щелкните мышью на кнопке OK. Интерфейс программы Настройка системы показан на рис. 19.10.

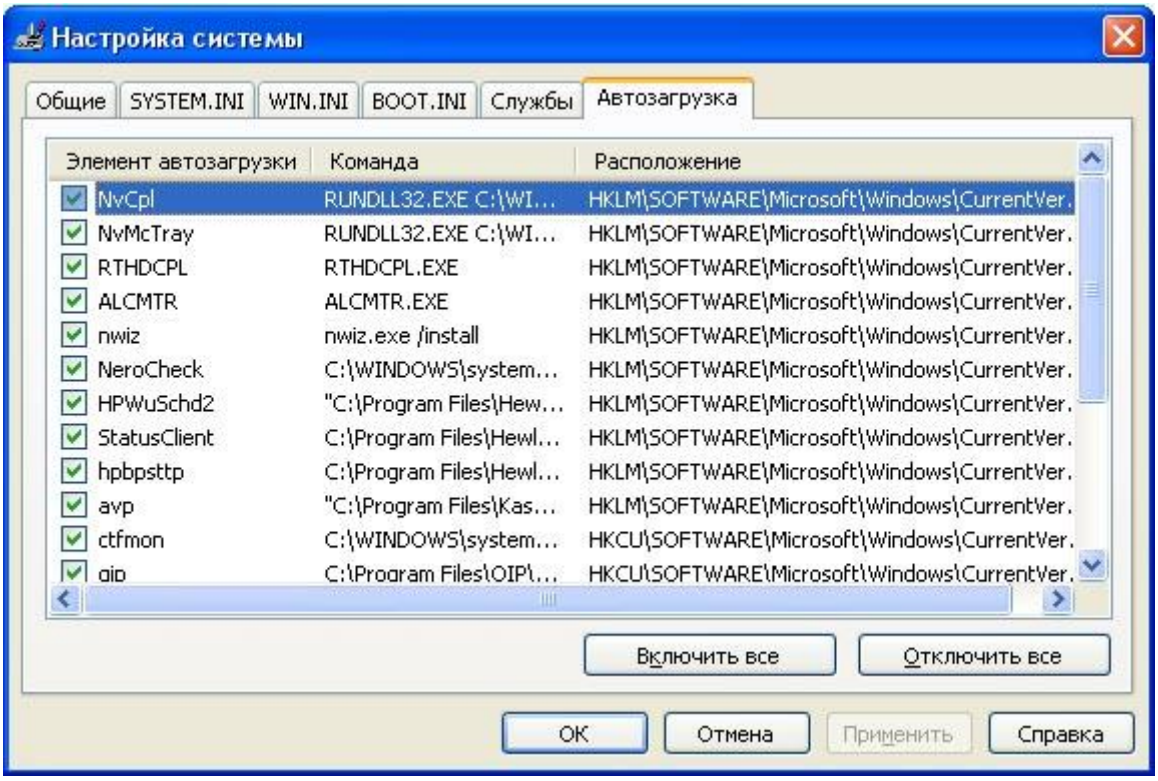


Рис. 19.10. Программа Настройка системы

Окно программы Настройка системы содержит шесть вкладок. Вкладка Общие (General) позволяет управлять режимами запуска операционной системы при помощи переключателя Вариант запуска (Startup Selection), который может быть установлен в одном из возможных положений:

- Обычный запуск - загрузка всех драйверов устройств и запуск всех служб (Normal startup - load all device drivers and services);
- Диагностический запуск - загрузка только основных драйверов и запуск основных служб (Diagnostic startup - load basic devices and services only);
- Выборочный запуск (Selective startup).

В последнем случае вы сможете выбрать те системные компоненты, которые будут обрабатываться и загружаться при запуске системы, установив соответствующие флажки: Обрабатывать файл SYSTEM.INI (Process SYSTEM.INI file) - если флажок установлен, в процессе загрузки Windows будут выполняться инструкции, содержащиеся в файле system.ini; Обрабатывать файл WIN.INI (Process WIN.INI file) - если флажок установлен, в процессе загрузки Windows будут выполняться инструкции, содержащиеся в файле win.ini; Загружать системные службы (Load system services) - если флажок установлен, при запуске ОС будут загружаться все принятые по умолчанию системные службы Windows; Загружать элементы автозагрузки (Load startup items) - если флажок установлен, одновременно с запуском Windows будут загружаться программы, список которых представлен на вкладке Автозагрузка (Startup) программы Настройка системы. Ниже вы сможете выбрать файл boot.ini, управляющей конфигурацией мультизагрузчика: если переключатель установлен в положение Использовать оригинальный BOOT.INI (Use original BOOT.INI), при загрузке будут применяться инструкции, содержащиеся в файле boot.ini, автоматически созданном операционной системой, установив переключатель в положение Использовать измененный BOOT.INI (Use modified BOOT.INI), вы

настроите систему на использование файла boot.ini, скорректированного в программе Настройка системы. Щелчок на кнопке Запустить восстановление системы (Launch System Restore) приведет к запуску программы Восстановление системы. Для того чтобы восстановить ранее измененный файл win.ini или system.ini, щелкните мышью на кнопке Извлечь файл (Expand file), и укажите в открывшемся окне имя файла, источник, из которого он должен быть восстановлен (как правило, в качестве источника выступает заранее подготовленная резервная копия), и папку, в которой файл дожжен быть размещен после извлечения.

Вкладки SYSTEM.INI и WIN.INI позволяют изменять и редактировать инструкции, содержащиеся в файлах system.ini и win.ini соответственно. Сбросив соответствующие флажки, вы сможете исключить любую инструкцию из данных файлов. Чтобы изменить порядок следования инструкций, выделите любую из них щелчком мыши, после чего последовательно нажимайте кнопки Вверх (Move up) или Вниз (Move down). Для того чтобы создать новую инструкцию, щелкните мышью на кнопке Создать (New), чтобы изменить уже существующую инструкцию, выделите ее щелчком мыши и нажмите на кнопку Изменить (Edit). Щелчок на кнопке Поиск (Find) откроет диалоговое окно, в специальном поле которого вы сможете ввести какое-либо ключевое слово - программа автоматически выполнит поиск данного слова по текущему файлу.

Открыв вкладку BOOT.INI, вы сможете отредактировать содержимое одноименного файла, управляющего параметрами мультизагрузчика Windows XP (подробнее о структуре и синтаксисе файла boot.ini читайте в разделе «Настройка мультизагрузчика» гл. 4). Переместить одну из строк файла boot.ini вверх или вниз относительно других строк можно, выделив соответствующую строку щелчком мыши и последовательно нажимая на кнопки Вверх (Move up) или Вниз (Move down). Для того чтобы определить какой-либо из вариантов загрузки принятым по умолчанию (именно эта версия ОС будет загружена автоматически по истечении временного промежутка ожидания, если пользователь не предпринял никаких действий), выделите соответствующую строку щелчком мыши и нажмите на кнопку По умолчанию (Set as default). Нажав на кнопку Проверить все пути загрузки (Check all boot paths), вы сможете выполнить проверку правильности записи инструкций мультизагрузчика. Указав соответствующее число в поле Таймаут (Timeout), вы сможете задать временной интервал в секундах, в течение которого мультизагрузчик будет ожидать, пока пользователь укажет требуемый вариант загрузки системы. Наконец, выделив один из вариантов загрузки щелчком мыши и установив соответствующие флажки в нижней части окна, вы сможете назначить для данного варианта загрузки Windows соответствующие ключи.

Перейдя ко вкладке Службы (Services) вы сможете отключить или включить системные службы, запускаемые одновременно с загрузкой Windows, сбросив или установив соответствующие флажки. Подробнее о системных службах Windows XP будет рассказано далее в гл. 21.

Наконец, вкладка Автозагрузка (Startup) позволяет отменить автоматический запуск некоторых приложений, загружаемых в память одновременно с запуском операционной системы. Ярлыки этих приложений отсутствуют в папке Автозагрузка Главного меню Windows XP, поскольку инструкции, необходимые для их автоматического запуска, хранятся в системном реестре Windows. Сбросив соответствующие флажки, вы сможете отключить запуск требуемых приложений. Нажатие на кнопку Отключить все (Disable all) приведет к отключению автозапуска всех этих программ, щелчок мышью на кнопке Включить все (Enable all) - к включению автозагрузки всех программ из списка.

Изменив все необходимые настройки, последовательно щелкните мышью на кнопках Применить (Apply), и Закрыть (Ok) в окне программы Настройка системы. В случае, если вы изменили какие-либо важные параметры системной конфигурации, может потребоваться перезагрузка компьютера, непосредственно после которой на экран будет выведено системное сообщение, говорящее о том, что загрузка осуществлена с новыми праметрами конфигурации.

Лабораторная работа. Командный интерфейс Windows. Основные команды

Цель

Научиться использовать основные команды командного интерпретатора cmd.exe: перемещение по каталогам, просмотр и редактирование файлов, копирование и удаление.

Теоретические сведения

1. О Командах

Рассмотрение мы начнем с системы команд MS-DOS, как с основы системы команд Windows. В системе Windows можно имитировать сеанс MS-DOS, запустив файл COMMAND.COM. Интерфейс представляет собой текстовый экран и командную строку. Команда вводится с клавиатуры и подтверждается нажатием ENTER.

Типы команд MS-DOS:

- внутренние команды;
- внешние команды.

Внутренние команды - это самые простые, чаще всего используемые команды. Вы не видите эти команды, когда вы просматриваете список файлов, хранящихся на системной дискете МС-ДОС. Они являются составной частью большого файла COMMAND.COM. Когда вы набираете внутреннюю команду, она тут же исполняется.

Имеются следующие внутренние команды МС-ДОС:

Break	**Del**	**Mkdir**	Set
Chdir	**Dir**	Path	Shift
Cls	Echo	Pause	**Time**
Copy	**Exit**	**Prompt**	**Type**
Tree	For	**Rem**	**Ver**
Date	Goto	**Ren**	
If	**Rmdir**	**Vol**	

Внешние команды располагаются на диске как программные файлы. Прежде, чем они смогут выполниться, они должны быть прочитаны с диска.

Любое имя файла с расширением .COM,.EXE или .BAT рассматривается как внешняя команда. Например, такие программы, как FORMAT.COM и DISKCOPY.COM, являются внешними командами. Так как все внешние команды представляют собой файлы, то вы можете создавать свои команды и добавлять их к МС-ДОС. Все созданные вами программы (независимо от используемого языка программирования) будут иметь расширение .EXE.

В WindowsXp существует специальный интерфейс для работы с расширенным набором команд Windows, основанным на системе команд MS-DOS. Для вызова этого интерфейса используется команда CMD.EXE. Этот интерфейс позволяет работать с длинными именами файлов.

2. Опции команд

Для того, что передать МС-ДОС дополнительную информацию, в командах могут быть использованы опции. Если вы включили некоторые опции, МС-ДОС предоставляет подготовленные ранее значения.

Квадратные скобки ([]) указывают на то, что заключенный в них элемент можете отсутствовать при наборе команды. Угловые (<>) скобки указывают на то, что в этом месте должен стоять вводимый вами текст. Имена дисководов не требуются, если вы не хотите точно указать МС-ДОС, на каком дисководе искать заданный файл. Ниже приводится формат всех команд МС-ДОС:

КОМАНДА [ОПЦИИ ...]

Здесь ОПЦИИ могут принимать следующие значения:

ДИСКОВОД : указывает на имя дисковода;
ИМЯ ФАЙЛА : указывает на имя некоторого дискового файла, включая расширение (если оно существует). Опции, относящиеся к имени файла, не указывают ни на устройство, ни на имя дисковода.
ИМЯ ВЕТВИ : указывает на имя ветви или на имя файла в следующем формате:
[<КАТАЛОГ>]\ [<КАТАЛОГ>...]\[<ИМЯ ФАЙЛА>]
ПЕРЕКЛЮЧАТЕЛИ : переключатели - это такие опции, которые контролируют выполнение команды. Им предшествует прямой слэш </> (например /P).
АРГУМЕНТЫ : предоставляют информацию командам. Вам обычно придется выбирать из списка аргументов (например ON/OFF).

3. Ввод и вывод
МС-ДОС всегда предполагает, что ввод идет с клавиатуры, а вывод выдается на дисплей. Однако, поток команд может быть перенаправлен. Ввод может идти из файла, а не с клавиатуры, а вывод идти в файл или на печатающее устройство вместо дисплея. Также, могут быть созданы "каналы", чтобы направить вывод одной команды на вход другой.

4. Как перенаправить вывод
Большинство команд выдают информацию, которая посылается на экран терминала. Используя символ "больше" (>), вы можете перенаправить эту информацию в файл. Например, команда DIR, выдает информацию о содержимом каталога на текущем дисководе на экран. Та же команда запишет информацию в файл NEWFILES, если вы наберете после команды знак "больше" и за ним - имя файла:
```
DIR >NEWFILES
```
Если файл с именем NEWFILES не существует, МС-ДОС создаст его и запишет в него всю информацию о рабочем каталоге текущего дисковода. Если такой файл уже существует, МС-ДОС затрет его содержимое новой информацией.
Если вы хотите добавить каталог или файл к другому файлу (не затирая старую информацию), то используйте два знака "больше" >> для того, чтобы указать МС-ДОС новый режим работы: добавить выходную информацию команды в конец указанного файла. Команда
```
DIR >>NEWFILES
```
добавит информацию о содержимом каталога к существующему уже файлы NEWFILES. Если же он не существует, то МС-ДОС создаст его.
Очень часто удобно использовать для ввода информации в команду не клавиатуру, а подготовленный заранее файл. В МС-ДОС для этого служит знак "меньше" <. Например, команда
```
SORT <NAMES >LIST1
```
сортирует файл NAMES и в отсортированном виде посылает его в файл LIST1

5. Командные каналы
Если вы хотите дать одновременно несколько команд, вы можете связать их через "канал" и послать в МС-ДОС. Например, иногда вам может понадобиться послать выход одной программы на вход к другой. Типичным примером может служить программа, которая выдает информацию по колонкам. Вам может понадобиться их отсортировать.
Объединение в канал осуществляется с помощью специального символа - вертикальная черта (|). Например, команда
DIR | SORT
выдаст на экран отсортированный в алфавитном порядке список файлов вашего каталога. Вертикальная черта означает, что вся выходная информация, сгенерированная командой слева от черты, будет передана для обработки команде, указанной справа.

Каналы могут быть также использованы, когда вам необходимо переадресовать выходную информацию в файл. Если вы хотите отсортировать список файлов вашего каталога и записать его в новый файл (например DIREC.FIL) наберите:
DIR | SORT >DIREC.FIL

МС-ДОС создаст файл с именем DIREC.FIL на текущем дисководе. В файле DIREC.FIL будет храниться отсортированный список файлов каталога текущего диска, так как дисковод не был точно указан в команде. Для того, чтобы использовать дисковод, отличный от текущего, (например дисковод <A>) наберите:
DIR | SORT >A:DIREC.FIL
Эта команда перешлет отсортированные данные в файл DIREC.FIL дисковода <A>.

"Канал" может состоять более, чем из двух команд. Например
DIR | SORT | MORE
отсортирует каталог, покажет полученный список на экран (24 строки за раз) и поместит строку --More-- на последней строчке. Это означает, что осталась еще не просмотренная информация.

Лабораторная работа. Разработка пакетного файла Windows

Цель
Научиться разрабатывать пакетные файлы для Windows.

Теоретические сведения

1. Пакетная обработка

Необходимая последовательность команд может быть помещена в специальный текстовый файл, называемый КОМАНДНЫМ ФАЙЛОМ. Все команды из этого файла будут обрабатываться таким образом, как будто они были последовательно введены с клавиатуры. Каждый командный файл должен иметь расширение .BAT и его можно выполнить, просто набрав его имя без указания расширения.

Расширение BAT получилось от слова Batch(пакет). То есть - это пакетные файлы или файлы с набором команд. Они использовались в MS DOS, используются в Windows 9x, Windows NT и Windows XP. Так как эти файлы обычные текстовые, редактирование их содержимого возможно в любом текстовом редакторе, например в NotePad. Контекстное меню Windows предусматривает специальный пункт меню для редактирования этих файлов.

Две команды МС-ДОС могут быть использованы только в командном файле: REM и PAUSE. Команда REM позволяет включать в командный файл комментарии, которые не будут интерпретироваться как команды во время выполнения данного файла. Команда PAUSE выдает на экран некоторое сообщение и позволяет либо прервать в данной точке выполнение командного файла, либо продолжить его выполнение дальше.

Если во время выполнения командного файла нажать CTRL+C, то на экране появится запрос системы:
Terminate batch job (Y/N)?
(прервать выполнение командного файла (Да/Нет)?)
Если нажать Y, то оставшиеся команды игнорируются, в противном случае выполнявшаяся команда прервется и выполнение продолжится со следующей строки командного файла.

Последней командой в файле пакетной обработки может стоять имя другого командного файла. Это позволяет вызывать один командный файл из другого после того, как первый завершил свою работу.

Перенаправить выходную информацию командного файла возможно, используя символы (< >). Однако, в командном файле нельзя использовать символ канала (|).

2. Командный файл с заменяемыми параметрами

По отношению к командам МС-ДОС, параметр является переменной величиной. В МС-ДОС можно создать командный (.BAT) файл с пустыми (заменяемыми) параметрами. Эти параметры, именуемые %0-%9, могут замещаться значениями, введенными при запуске этого командного файла на исполнение.

Если вы используете параметр %0, он всегда заменяется на имя дисковода (если указано) и имя командного файла.

Может быть использовано не более десяти параметров %0 - %9. Если необходимо определить больше десяти параметров, то используется команда SHIFT.

Если знак процента используется как часть имени файла, то внутри командного файла этот знак набирается дважды. Например, файл ABC%.EXE определяется как ABC%%.EXE

Если используется замещаемый параметр %0, то он всегда будет заменяться на имя дисковода (если определено) и имя этого командного файла. Если нет необходимости ссылаться на командный файл, то используются замещаемые параметры начиная с %1.

3. Исполняемые команды

Эти команды относятся к внутренним командам MS DOS и могут использоваться в командной строке, но их использование обретает смысл только в пакетных файлах.

call Вызов одного пакетного файла из другого.
echo Вывод сообщений и переключение режима отображения команд на экране.
for Запуск указанной команды для каждого из файлов в наборе.
goto Передача управления в отмеченную строку пакетного файла.
if Оператор условного выполнения команд в пакетном файле.
pause Приостановка выполнения пакетного файла и вывод сообщения
rem Помещение комментариев в пакетные файлы и файл CONFIG.SYS.
shift Изменение содержимого (сдвиг) подставляемых параметров для пакетного файла.

@ - Запрещает вывод команды следующей за символом.

@ECHO OFF - Запрещает вывод команд в последующих строках.

%1 - Процент и цифра, начиная с единицы, позволяет добавлять переменные в команды.

Например myname.bat:

```
echo Hello %1
```

Запускаем myname Vasia
Вывод:
Hello Vasia

:LABEL - создает метку, переход к которой можно осуществить командой GOTO.

CALL - используется для вызова другого командного файла. После завершения исполнения вызванного файла будет продолжено выполнение основного файла. Если вызывается несуществующий файл, будет выдано сообщение об ошибке.

CLS - Очищает экран DOS.

ECHO - выводит сообщение на экран. "ECHO Hello World" выведет на экран Hello World при выполнении. Для вывода пустой строки используется "ECHO."

EXIT - закрытие окна MS-DOS.

GOTO LABEL - используется для перехода к конкретной метке.

IF - используется для сравнения значений.
Например:
IF ERRORLEVEL ==3 GOTO MYLABEL

PAUSE - пауза. Предлагает пользователю нажать клавишу

REM - позволяет добавить комментарии в командный файл.

SHIFT - изменяет последовательность параметров командного файла. Дополнительную информацию можно узнать набрав SHIFT /? в DOS

4. Первая программа
Для вывода сообщения в BAT файлах используется команда echo:
Вывод сообщений и переключение режима отображения команд на экране.
ECHO [ON | OFF]
ECHO [сообщение]
Ввод ECHO без параметров позволяет выяснить текущий режим отображения команд.

Пример BAT файла, который выводит на экран сообщение:

```
echo
echo Hello, World
```

Режим Echo включен по умолчанию, поэтому отображается и команда и результат. Чтобы увидеть только результат, необходимо отключить этот режим.

```
echo off
echo hello batch files
```

Первая команда все равно видна. Это можно исправить, сразу вызвав CLS(команда очистки экрана) после отключения режима отображения.

```
echo off
cls
echo hello batch files
```

5. Первая полезная программа.

Пример BAT файла, который создает папку backup и копирует туда содержимое текущей папки:

```
echo off
cls
echo start backup
mkdir backup
copy *.* backup
echo end backup
```

. - это маска файлов. Символ «*» обозначает, что в этом месте может быть любое количество символов. То есть если необходимо скопировать только текстовые файлы, то маска будет выглядеть следующим образом «*.txt»

6. Обработка параметров в BAT файле

Вызывая из командной строки BAT файл, можно обрабатывать параметры вызова. Для получения параметра мы должны использовать символ % и номер параметра. MS DOS заменит эту конструкцию переданным параметром.

```
echo off
cls
echo start backup
mkdir %1
copy *.* %1
echo end backup
```

Вызов такого BAT файла осуществляется с параметром – имя директории, которую нужно создать. Например, если BAT файл называется 1.bat, то вызов его может быть таким:
1.bat backup

В результате, BAT файл создаст папку с именем backup.

7. PAUSE

Pause позволяет остановить выполнение Bat-файла до нажатия клавиши на клавиатуре. Это может быть полезно, если необходимо, чтобы пользователь подтвердил выполнение какого-то действия нажатием на клавишу.

```
echo off
cls
echo  insert disk to A:
pause
copy 1.txt a:
```

8. IF

IF можно использовать для сравнения. Оператор условного выполнения команд в пакетном файле.
IF [NOT] ERRORLEVEL число команда
IF [NOT] строка1==строка2 команда
IF [NOT] EXIST имя_файла команда

NOT Windows выполняет команду лишь в том случае, если условие ложно.
ERRORLEVEL число Условие истинно, если последняя запущенная программа завершилась с кодом возврата, равным либо превышающим

указанное число.
команда Команда, которую следует выполнить в случае истинности условия.
строка1==строка2 Условие истинно при совпадении обеих строк.
EXIST имя_файла Условие истинно, если указанный файл существует.

В случае сравнения строк если сравнение верно, то будет выполнена команда за IF, иначе она будет пропущена.

Создадим general.bat, который будет запускать с параметрами другой bat файл.
call 8.bat A

А в командном файле 8.bat напишем условие.

```
echo off
cls
if "%1"=="A" Attrib.exe
if "%1"=="E" Edit.com
```

Запускаем.
Сработала только строка с параметром А.

9. Goto
Передача управления в отмеченную строку пакетного файла.

GOTO метка
метка Текстовая строка, играющая в пакетном файле роль метки.

Метка должна находиться в отдельной строке программы и начинаться с двоеточия.
Эта команда переводит выполнение Bat-файла на указанную метку. Переделаем прошлый пример на вывод нескольких строк в зависимости от параметров.

```
echo off
cls
if "%1"=="A" GOTO ACOM
if "%1"=="E" GOTO ECOM
:ACOM
echo This is
echo Parameters A
echo Goodbye !
GOTO ENDS
:ECOM
echo This is
echo Parameters E
echo Goodbye !
:ENDS
```

10. Shift
Команда shift позволяет сдвигать параметры.
Изменение содержимого (сдвиг) подставляемых параметров для пакетного файла.
SHIFT
Параметров может быть много. Например, столько:
call 10.bat Hello Params str 123
Первый параметр - это %0 в нем имя вызываемого bat файла.

```
echo off
cls
```

```
echo %0
echo %1
echo %2
echo %3
echo %4
echo ------------------------------
echo shift
echo ------------------------------
shift
echo %0
echo %1
echo %2
echo %3
echo %4
```

После вызова все параметры будут смещены.

11. For

BAT-файл, который пересчитывает все *.jpg и *.bmp файлы:

```
echo "Found Files" > result_file.txt
for %%a in (*.jpg, *.bmp) do echo %%a >> result_file.txt
```

Эти же команды можно набирать непосредственно в командной строке, но тогда вместо %%a пишется %a.

В конструкции for можно через запятую указывать не только маски, но и имена, что можно продемонстрировать следующим примером.

```
for %%a in (1, 2, 3, 4, 5 ) do mkdir %%a
```

эта команда создаст на диске папки с именами 1, 2, 3, 4, 5

Пример можно усложнить, добавив к созданию папки копирование в нее какого-либо файла.

12. Переменные

В BAT-файлах для переменных вы должны используется префикс %%, а в командной строке - %.

DATE /t >%D_Date% дает текущую дату в переменной %D_Date%

13. Несколько примеров bat-файлов

13.1. Как создать файл с произвольным именем из bat файла

Для создания файла в процессе выполнения пакетного файла используется символ перенаправления. Он выглядит так:

>

Т.е. чтобы создать файл нужно перенаправить поток с экрана в файл. Сделать это можно при помощи следующей команды:

```
@echo Start file>C:\1.txt
```

После выполнения этой команды в корне диска С будет создан текстовый файл со строкой Start file.
При создании файла в его имени можно использовать системные переменные или их части. Например, можно создать файл-отчет о работе bat файла с именем, равным дате запуска bat файла. Для этого можно использовать следующие строки в bat файле.

```
set datetemp=%date:~-10%
@echo .>%SYSTEMDRIVE%\%DATETEMP%.txt
```

Эти две строки работают следующим образом. Сначала в памяти создаем переменную datetemp, которой присваиваем 10 символов справа налево от системной переменной DATE. Таким образом, теперь во временной переменной datetemp содержится только текущая дата. Следующией строкой перенаправляем вывод символа точка в файл, имя которого берем из переменной datetemp, а расширение txt указываем явно. Файл будет создан на системном диске компьютера, где выполняется bat файл.
При сборе администратором информации о компьютерах в сети будет удобнее добавить к имени файла имя компьютера. Это легко можно сделать при помощи следующей команды:

```
@echo .>C:\FolderName\%COMPUTERNAME%.txt
```

Эта команда в ходе выполнения пакетного файла создаст на диске С текстовый файл с именем того компьютера, на котором выполняется пакетный файл.
Для создания файла с определенным именем можно использовать любые системные переменные, либо создать свои, на основе системных переменных и/или других данных.

13.2. Как создать папку из bat файла

Для создания папки используется команда MKDIR или ее сокращенный аналог MD. Для создания папки из bat файла нужно использовать следующую команду:

```
MD FolderName
```

После выполнения такой команды будет создана папка FolderName в папке, откуда запущен bat файл. Чтобы создать файл в отличном от запуска bat файла месте, например в корне диска D, используйте явное указание расположения новой папки. Команда будет выглядеть так:

```
MD D:\FolderName
```

При создании папок можно пользоваться системными переменными. Например, можно создать в корне диска D папку с именем текущего пользователя. Для этого понадобится переменная %USERNAME%, а команда будет выглядеть следующим образом:

```
MD D:\%USERNAME%
```

Можно еще более усложнить команду и создать папку с именем текущего пользователя на системном диске его компьютера. Команда для этого будет выглядеть так:

```
MD %SYSTEMDRIVE%\%USERNAME%
```

При создании папок или файлов можно использовать любые системные переменные или их части. Следующий пример демонстрирует создание на системном диске компьютера пользователя папки с именем равным текущей дате.

```
set datetemp=%date:~-10%
MD %SYSTEMDRIVE%\%datetemp%
```

Эта конструкция работает следующим образом.
Первая команда создает в памяти переменную datetemp, которая будет уничтожена по окончании работы bat файла. То тех пор, пока bat файл не закончил свою работу есть возможность оперировать со значением этой переменной. Переменной datetemp присваивается 10 символов справа налево от системной переменной DATE, т.е. от текущей даты. Переменная DATE имеет формат Дн ДД.ММ.ГГГГ. Первые символы слева - имя дня недели и поэтому мы их отбрасываем и присваиваем временной переменной datetemp только текущую дату.
Этим не ограничивается список возможностей при создании папок. Вы можете опрерировать переменными так, как удобно Вам, создавая папки с уникальными, легко читаемыми названиями. Получить список всех переменных можно при помощи команды SET.

13.3. Как перенаправить результат выполнения команд в файл

Часто, при выполнении сложного bat файла в автоматическом режиме проверить результаты его работы бывает затруднительно по многим причинам. Поэтому проще записывать результаты работы команд batch файла в текстовый файл (лог-файл). а потом анализировать правильность работы bat файла по этому логу.
Перенаправить результат работы команд bat файла в лог-файл достаточно просто. Далее будет показано, как это можно сделать.
Создайте bat-файл следующего содержания (скопируйте эти строки в Блокнот и сохраните файл с расширением bat):

```
@echo off
echo Start %time%
echo Create test.txt
echo test>C:\test.txt
echo Copy Test.txt to Old_test.txt
copy C:\test.txt C:\Old_test.txt
echo Stop %time%
```

Первая строка отключает вывод самих команд. Таким образом, в лог-файл будут записаны только результаты их выполнения.
Вторая строка записывает в лог-файл время начала работы пакетного файла.
Третья строка записывает в лог-файл пояснение того, что следующая команда создаст файл test.txt
Команда из четвертой строки создает файл test.txt с корне диска C. Файл создается для примера. Эта команда записывает в файл C:\test.txt слово test

Пятая строка выводит в лог-файл пояснение, что следующая команда выполняет копирование файла из одного места в другое.
Команда в шестой строке копирует созданный файл C:\test.txt в файл C:\Old_test.txt, т.е. создается копия файла под новым именем.
Последняя, седьмая строка содержит команду вывода времени завершения работы пакетного файла. В сумме с записью в лог-файл времени начала работы пакетного файла эти два значения времени дают возможность оценить время работы пакетного файла.

Сохраните этот пакетный файл под именем, например, 1.bat
Предположим, что отчет о работе пакетного файла мы бы хотели хранить в отдельной папке и каждый день записывать отчет с новым именем файла, чтобы была возможность в любой из дней обратиться к логам за предыдущие дни. Причем, имя лог-фала хотелось бы иметь в виде даты работы пакетного файла. Чтобы все это реализовать создадим на диске C (например) папку с именем LOG, т.е. полный путь к ней будет выглядеть C:\LOG. Созданный пакетный файл 1.bat будем запускать следующей командой:

```
1.bat>C:\LOG\%date~-10%.txt
```

Если пакетный файл будет запускаться из Планировщика, то нужно указать полный путь с bat-файлу. Помните, что если в пути есть пробелы, то надо использовать либо кавычки, либо формат 8.3. Т.е., если путь к bat-файлу C:\Program Files\1.bat, например, то в командной строке Планировщика для запуска bat-файла нужно указать одну из следующих строк:

```
"C:\Program Files\1.bat">C:\LOG\%date~-10%.txt
C:\Progra~1\1.bat>C:\LOG\%date~-10%.txt
```

После запуска файла 1.bat в папке C:\LOG будет создан файл с именем, равным дате запуска bat-файла, например, 13.01.2004.txt Это и будет отчет о работе пакетного файла 1.bat
Запуск bat-файла, пример которого показан в первом листинге вверху страницы, указанной выше командой, приведет к созданию лог-файла такого содержания:

```
Start 19:03:27,20
Create test.txt
Copy Test.txt to Old_test.txt
Скопировано файлов: 1.
Stop 19:03:27,21
```

Таким образом, для выполнения перенаправления результатов работы bat-файла в лог-файл нужно использовать символ перенаправления > Синтаксис таков:

```
Путь\ИмяФайла.bat>Путь\ИмяЛогФайла.txt
```

Расширение лог-файла может быть любым. При желании, отчет о выполнении пакетного задания можно оформить даже в виде страницы html (соответствующие теги могут быть выведены в лог-файл так, как выводились комментарии в примере 1.bat) и скопировать его на корпоративный сервер.

13.4. Как автоматически ответить на запрос о подтверждении

Некоторые команды при выполнении требуют подтверждения потенциально опасного действия. Например, такие команды как format или del предварительно запросят подтверждения на дальнейшее выполнение. Если одна из этих команд выполняется в пакетном файле, то запрос на подтверждение остановит выполнение пакетного файла и он будет ожидать от пользователя выбора одного из предложенных вариантов. Причем, если результат выполнение пакетного файла перенаправлен в лог-файл, то пользователь не увидит запроса на подтверждение и batch файл будет выглядеть зависшим.
Для исправления таких неприятностей можно перенаправить нужный ответ в команду. Т.е. выполнить обратное действие для перенаправления вывода результатов работы команды в файл.
Посмотрим на примере как выглядит запрос на подтверждение потенциально опасного действия. Создадим на диске C, например, папку Folder. Создадим в ней или скопируем в нее два любых файла. Далее, откроем командную строку и выполним следующую команду:

```
del C:\Folder
```

Эта команда должна удалить все файлы из указанной папки. Но предварительно будет выдан запрос для подтверждения следующего содержания:

```
C:\Folder\*, Продолжить [Y(да)/N(нет)]?
```

Выполнение команды будет остановлено до тех пор, пока не будет нажата либо клавиша Y, либо клавиша N. При выполнении пакетного файла в автоматическом режиме, его исполнение остановится.
Чтобы избежать этого используем перенаправление. Перенаправление осуществляется при помощи символа

```
|
```

Вертикальная черта говорит о том, что вместо вывода символа на экран его надо «отдать» следующей за символом команде. Проверим работу перенаправления. Выполните в командной строке следующую команду:

```
echo Y|del C:\Folder
```

На экране будет показан запрос на подтверждение удаления всех файлов в папке Folder, но уже с положительным ответом (Y). Все файлы из папки Folder будут удалены.
Будьте осторожны с этой командой.

13.5. Как отключить вывод на экран команд при выполнении пакетного файла

При выполнении пакетного файла на экран, помимо результатов работы команды, выводятся и сами команды. Чтобы отключить вывод команд, можно использовать символ @.
Чтобы не выводить на экран одну команду, можно поставить знак @ в начале этой команды.

```
echo Testing
```

Эта команда выведет на экран команду echo Testing, а на следующую строку - результат ее работы, слово Testing.

```
@echo Testing
```

Эта команда выведет на экран только результат работы команды, т.е. слово Testing. Сама команда выведена не будет.
Если на протяжении выполнения всего файла выводить команды на экран не нужно, то проще первой строкой в пакетном файле написать следующую команду:

```
@echo off
```

Эта команда отключит вывод команд на экран на протяжении выполнения всего пакетного файла. Чтобы сама команда не выводилась, она начинается с символа @.

13.6. Как из одного bat-файла запустить другой

Иногда, при выполнении пакетного файла, возникает необходимость запустить другой пакетный файл. Причем, в некоторых случаях, выполнение основного пакетного файла должно быть приостановлено, пока выполняется вспомогательный файл, а в других вспомогательный файл должен работать параллельно с основным.
Для примера создадим два bat файла. Один с именем 1.bat и содержащий всего одну команду

```
call 2.bat
```

Второй с именем 2.bat и также содержащий одну команду

```
pause
```

Теперь запустим файл 1.bat Откроется окно, в котором будет предложено нажать любую клавишу для продолжения, после нажатия которой окно закроется. Таким образом, вызов из одного пакетного файла другого при помощи команды call останавливает исполнение пакетного файла до тех пор, пока не завершится выполнение пакетного файла, вызванного командой call.
В другом случае, надо запустить из bat файла либо приложение, либо другой пакетный файл, не прерывая выполнения основного пакетного файла. Такое нередко бывает нужно сделать, например, принудительно открыв лог работы пакетного файла, запланированного на ночь, чтобы с утра, пользователь мог проконтролировать правильность его выполнения. Для этого используется команда start Исправим в файле 1.bat строку на

```
start 2.bat
```

и запустим файл 1.bat Теперь открылось окно, в котором для продолжения надо нажать любую кнопку, а окно основного пакетного файла (1.bat) отработав закрылось.
Таким образом, для вызова из одного пакетного файла другого, без остановки работы первого пакетного файла, нужно применять команду start.
Рассмотренные команды start и call могут использоваться не только для запуска других пакетных файлов, но и для запуска любых приложений или открытия файлов.
Например, команда start log.txt, находящаяся в теле пакетного файла, откроет файл log.txt в Notepad без остановки работы пакетного файла.

13.7. Как отправить сообщение из bat-файла

Когда пакетный файл исполняется на одной из машин в сети, то удобно проинформировать администратора об окончании его выполнения при помощи сообщения, отправленного на машину администратора. Сделать это можно, включив в пакетный файл команду

```
net send name Message text
```

Где name имя машины или пользователя, которому адресуется сообщение, а Message text - текст сообщения. После выполнения этой команды пользователю name будет отправлено сообщение.
Обратите внимание на то, что при использовании в тексте сообщения кириллицы текст должен быть набран в кодировке MS-DOS (866 кодовая страница). Иначе сообщение придет в виде нечитаемых символов. Набрать текст в кодировке DOS можно при помощи любого текстового редактора, поддерживающего эту кодировку. Это может быть, например, FAR. Откройте в FAR пакетный файл на редактирование (F4) и нажмите кнопку F8. В верхней строке редактора должна быть указана кодировка DOS, а снизу, у подсказки о быстрых клавишах, у клавиши F8 должна быть надпись Win, говорящая о том, что текущая кодировка DOS и для переключения в кодировку Win надо нажать F8.

13.8. Как автоматизировать удаление файлов по типу

Чтобы очистить диск от временных файлов можно использовать команду

```
del /f /s /q C:\*.tmp
```

Где
/f - удаляет все файлы, даже если у них установлен атрибут только чтение
/s - удаляет файлы из всех подкаталогов
/q - отключает запрос на подтверждение удаления файла
C: - диск, на котором будут найдены и удалены файлы. Можно указать не весь диск, а папку, например, C:\WinNT
*.tmp - тип файлов, которые будут удалены
Будьте аккуратны с ключем /q и типами удаляемых файлов. Команда удаляет, не спрашивая разрешения и при указании неправильного типа файлов может удалить лишнего.

13.9. Как переименовать файлы по маске из пакетного файла

Иногда возникает необходимость переименовать все файлы в папке по шаблону из пакетного файла. Сделать это можно при помощи следующей команды в bat-файле:

```
for /f "tokens=*" %%a in ('dir /b PATH\*.*') do ren PATH\%%a Prefix%%a
```

В этой строке надо заменить PATH\ на путь к файлам, которые будут переименованы, а Prefix на те символы, которые будут добавлены к имени файла при переименовании.
Не помещайте пакетный файл в папку, где происходит переименование, иначе он будет переименован тоже. Если в папке, где происходит переименование файлов есть подпапки, то к имени подпапки также будет добавлен префикс, т.е. подпапки будут переименованы как и файлы.

Если указать определенную маску для типов файлов, которые подлежат переименованию, например, *.txt, а не *.* как в примере, то будут переименованы файлы только указанных типов. Другие файлы и папки переименовываться не будут.

13.10. Как из bat файла обойти проверку даты

Некоторое программное обеспечение при запуске проверяет текущую системную дату. Если дата больше, чем заложено разработчиком, то программа не запускается. Например, разработчик считает, что версия программы может отработать месяц, а потом пользователь должен будет установить обновленную версию программы. С одной стороны это забота о пользователе, который будет иметь в своем распоряжении свежую версию программы с устраненными недочетами, по отношению к прошлым версиям. С другой стороны, производитель вынуждает пользователя скачивать новую версию даже если пользователя полностью устраивает та версия программы, которая у него установлена. Данную проблему можно легко решить при помощи следующего пакетного файла, который будет запускать программу, дожидаться ее завершения и возвращать дату на ту, которая была до запуска программы.

```
set tempdate=%date:~-10%
date 01-01-04
notepad.exe
date %tempdate%
```

В данном примере текущая системная дата сначала сохраняется в переменной, затем (во второй строке) системная дата устанавливается на 1-е января 2004 года, а потом вызывается программа, которая проверяет системную дату. В данном примере это Блокнот. До тех пор, пока открыт Блокнот, пакетный файл находится в ожидании, не завершаясь и не переводя системную дату обратно. Как только Блокнот будет закрыт, пакетный файл продолжит свое выполнение и установит системную дату на сохраненное в переменной tempdate значение, т.е. на то, которое было до запуска пакетного файла.
Не забывайте, что если путь до файла, которым запускается программа, содержит пробелы, то его (путь) необходимо заключить в кавычки. Если путь содержит кириллицу, то при написании пакетного файла необходимо использовать текстовый редактор, поддерживающий кодировку DOS (например, FAR). В противном случае, при запуске пакетного файла будет выведено сообщение о том, что "указанный файл не является внутренней или внешней коммандой...".
Если программа проверяет текущую системную дату только при своем запуске и во время работы больше этого не делает, то пакетный файл можно модифицировать, добавив перед именем исполняемого файла программы оператор start, т.е. наш пример будет выглядеть так:

```
set tempdate=%date:~-10%
date 01-01-04
start notepad.exe
date %tempdate%
```

В этом случае, пакетный файл изменит системную дату, запустит программу и не дожидаясь ее завершения вернет дату на ту, которая была до запуска программы.

13.11. Как в bat файле дождаться появления определенного файла

Иногда необходимо при появлении определенного файла в папке выполнить какое-то действие. Чтобы организовать проверку появления файла в папке можно использовать следующий пакетный файл

```
:test
if exist c:\1.txt goto go
sleep 10
goto test
:go
notepad
```

Такой пакетный файл будет проверять с интервалом 10 секунд наличие в корне диска C файла 1.txt и когда файл 1.txt появится, будет выполнено действие, указанное после метки go, т.е. в этом примере будет запущен Блокнот.
Утилита sleep свободно распространяется в составе Resource Kit. Вы можете её скачать здесь.
Если файл 1.txt большого размера и копируется откуда-то, то может получится так, что пакетный файл проверит его наличие в то время, как файл еще не скопировался или занят другим приложением. В таком случае, попытка выполнить какие-то действия с файлом 1.txt приведет к ошибке. Чтобы этого не произошло пакетный файл можно модифицировать следующим образом

```
:test
if exist c:\1.txt goto go
sleep 10
goto test
:go
rename c:\1.txt 1.txt
if not errorlevel 0 goto go
del c:\1.txt
```

Когда файл 1.txt скопировался на диск C не полностью, либо занят другим приложением, попытка его переименовать вызовет ошибку и цикл будет повторяться до тех пор, пока файл не скопируется полностью либо не будет освобожден. После того, как команда rename c:\1.txt 1.txt будет выполнена без ошибки (т.е. файл свободен), то с ним можно выполнять любые действия. В последнем примере это его удаление.

13.12. Как добавить комментарии в bat-файл

При написании большого пакетного файла очень полезно добавлять комментарии к его основным блокам. Это позволит с легкостью разобраться в том, что делают эти блоки по прошествии времени.
Комментарии можно добавить несколькими способами. Первый больше годится для написания больших комментариев, описывающих либо весь пакетный файл, либо несколько больших его блоков. Код выглядит следующим образом:

```
goto start
-------------------------------------
Этот пакетный файл предназначен
для автоматизации рутинных операций,
выполняемых ночью для синхронизации
содержимого корпоративного ftp-сервера
```

```
с ftp-серверами филиалов
-------------------------------------
Пакетный файл написан 01/01/2004
Последнее исправление внесено 10/02/2004
-------------------------------------
И т.д.
:start
```

Такое написание комментария при запуске пакетного файла передаст управление сразу к команде, следующей за меткой start. За это отвечает оператор перехода goto.
Более простые комментарии можно добавить, начиная строки с команды rem или с двух двоеточий, идущих друг за другом.

```
rem Этот блок устанавливает соединение с удаленным сервером

:: Этот блок проверяет дату изменения файлов
```

Комментирование больших пакетных файлов (как, в принципе, и любого кода) - хороший тон, который значительно облегчает процесс разбора этих файлов другими людьми или самим автором по прошествии значительного времени с момента написания.

СОДЕРЖАНИЕ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К САМОСТОЯТЕЛЬНОЙ РАБОТЕ

Целью самостоятельной работы студентов является:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать справочную и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений.

Самостоятельная работа является одним из видов учебных занятий студентов, проводится внеаудиторно, выполняется студентом по заданию преподавателя, но без его непосредственного участия.

Объем времени, отведенный на самостоятельную работу, определяется в соответствии с учебным планом специальности и рабочей программой учебной дисциплины.

Основные формы самостоятельной работы:

- для овладения знаниями: чтение текста (учебника, первоисточника, дополнительной литературы); конспектирование текста; выписки из текста; работа со справочниками; учебно-исследовательская работа; использование компьютерной техники и Интернета и др.;
- для закрепления и систематизации знаний: работа с конспектом лекций (обработка текста); повторная работа над учебным материалом (учебником, первоисточником, дополнительной литературой); составление плана и тезисов ответа; составление таблиц для систематизации учебного материала; ответы на

контрольные вопросы; аналитическая обработка текста (аннотирование, рецензирование, реферирование и др.); подготовка сообщений к выступлению на семинаре, конференции; подготовка рефератов, докладов; составление библиографии; тестирование и др.;

- для формирования умений: решение задач и упражнений по образцу; выполнение контрольных и расчетно-графических работ.

Самостоятельная работа может осуществляться индивидуально или группами студентов в зависимости от цели, объема, конкретной тематики самостоятельной работы, уровня сложности, уровня умений студентов.

Контроль результатов самостоятельной работы студентов может осуществляться в пределах времени, отведенного на обязательные учебные занятия по дисциплине, может проходить в письменной, устной или смешанной форме, с предоставлением продукта творческой деятельности студента.

В качестве **форм и методов контроля самостоятельной работы** студентов могут быть использованы семинарские занятия, коллоквиумы, зачеты, тестирование, самоотчеты, контрольные работы, защита творческих работ и др.

Критерием оценки результатов самостоятельной работы студента являются:

- уровень освоения студентом учебного материала;
- умение студента использовать теоретические знания при выполнении практических задач;
- обоснованность и четкость изложения ответа;
- оформление материала в соответствии с требованиями.

ТЕМАТИЧЕСКИЙ ПЛАН

-

Наименование тем и содержание самостоятельной работы	Кол-во часов	
	очная	заочная
1. Основные понятия операционных систем	4	4
2. Процессы	4	4
3. Планирование процессов	4	4
4. Кооперация процессов	4	4
5. Алгоритмы синхронизации	4	21
6. Механизмы синхронизации	4	21
7. Тупики	4	21
8. Схемы управления памятью	4	21
9. Виртуальная память	4	21

Вопросы для самоконтроля и темы упражнений.

1 Назначение и функции операционных систем (ОС). Классификация операционных систем.
2 Мультипрограммирование. Режим разделения времени.
3 Многопользовательский режим работы. Режим работы и ОС реального времени.
4 Универсальные операционные системы и ОС специального назначения.
5 Модульная структура построения ОС и их переносимость.
6 Управление процессором. Понятие процесса и ядра.
7 Сегментация виртуального адресного пространства процесса. Структура контекста

процесса. Идентификатор и дескриптор процесса. Иерархия процессов.
8 Диспетчеризация и синхронизация процессов. Понятия приоритета и очереди процессов.
9 Средства обработки сигналов. Понятие событийного программирования. Средства коммуникации процессов.
10 Способы реализации мультипрограммирования. Понятие прерывания. Многопроцессорный режим работы.
11 Управление памятью. Совместное использование памяти. Защита памяти.
12 Механизм реализации виртуальной памяти. Стратегия подкачки страниц.
13 Принципы построения и защита от сбоев и несанкционированного доступа.

СОДЕРЖАНИЕ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ

Контрольная работа (часть 1). Пакетные файлы Windows

Примеры заданий для контрольной работы

1. Рассортировка

Пакетный файл запускается с параметрами – расширениями файлов (например txt, jpg, pas). Автоматически создаются директории с именами, содержащими эти расширения, все файлы с этими расширениями копируются в соответствующие директории.

Например, в некой директории находится несколько файлов pas, несколько фалов doc и несколько файлов txt. Пользователь вызывает пакетный файлс параметрами pas doc, после этого создаются две директории pasfiles и docfiles, все pas-файлы скопированы в первую директорию, все doc-файлы – во вторую.

2. Резервное копирование файлов

Пакетный файл запускается с параметрами – именами файлов (1.txt, photo.jpg, lab01.pas). Автоматически создается директория bakup (если она не существует), файлы-параметры архивируются программой rar.exe в архив, имя которого содержит текущую дату. Архив перемещается в директорию backup.

Например, в некой директории находятся файлы lab01.pas и lab02.pas. Пользователь вызывает пакетный файл с параметрами lab01.pas lab02.pas, после этого создается директория backup, файлы lab01.pas и lab02.pas сжимаются в архив 271008.rar, который перемещается в директорию backup.

3. Слияние файлов

Пакетный файл запускается с параметрами – именами файлов (1.txt, 2.txt). В случае, если имена файлов имеют одинаковые расширения автоматически создается новый файл, имя которого представляет собой сумму имен файлов-параметров, а содержимое – слияние содержимого этих файлов. В случае, если расширения отличаются – выдается сообщение об ошибке и программа завершается.

Например, в некой директории лежат файлы p1.txt и p2.txt. Пользователь вызывает пакетный файл: merge.bat p1.txt p2.txt, после этого создается файл p1p2.txt, в который поочередно записывается содержимое файлов p1.txt и p2.txt.

Контрольная работа (часть 2). Пакетные файлы Unix

Примеры заданий для контрольной работы

1. Рассортировка
Пакетный файл запускается с параметрами – расширениями файлов (например txt, jpg, pas). Автоматически создаются директории с именами, содержащими эти расширения, все файлы с этими расширениями копируются в соответствующие директории.

Например, в некой директории находится несколько файлов pas, несколько фалов doc и несколько файлов txt. Пользователь вызывает пакетный файлс параметрами pas doc, после этого создаются две директории pasfiles и docfiles, все pas-файлы скопированы в первую директорию, все doc-файлы – во вторую.

2. Резервное копирование файлов
Пакетный файл запускается с параметрами – именами файлов (1.txt, photo.jpg, lab01.pas). Автоматически создается директория bakup (если она не существует), файлы-параметры архивируются программой rar.exe в архив, имя которого содержит текущую дату. Архив перемещается в директорию backup.

Например, в некой директории находятся файлы lab01.pas и lab02.pas. Пользователь вызывает пакетный файл с параметрами lab01.pas lab02.pas, после этого создается директория backup, файлы lab01.pas и lab02.pas сжимаются в архив 271008.rar, который перемещается в директорию backup.

3. Слияние файлов
Пакетный файл запускается с параметрами – именами файлов (1.txt, 2.txt). В случае, если имена файлов имеют одинаковые расширения автоматически создается новый файл, имя которого представляет собой сумму имен файлов-параметров, а содержимое – слияние содержимого этих файлов. В случае, если расширения отличаются – выдается сообщение об ошибке и программа завершается.

Например, в некой директории лежат файлы p1.txt и p2.txt. Пользователь вызывает пакетный файл: merge.bat p1.txt p2.txt, после этого создается файл p1p2.txt, в который поочередно записывается содержимое файлов p1.txt и p2.txt.

СОДЕРЖАНИЕ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К РАСЧЕТНО-ГРАФИЧЕСКОЙ РАБОТЕ

Целью расчетно-графической работы является получение опыта в разработке средств автоматизации управления операционными системами.

Задание:

1. Разработать пакетный файл Windows, который выполняет элементарные действия с файлами и директориями, управляемый параметрами запуска из командной строки.
2. Разработать аналогичный пакетный файл Unix.
3. Разработать аналогичную программу на языке C.

Отчет о выполнении РГР должен содержать:

1. Исходные тексты всех программ

2. Тестовые наборы входных и выходных данных