

Московский Государственный Технический Университет имени Н. Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

УТВЕРЖДАЮ

Зав. кафедрой ИУ6

д.т.н., проф. _____ Сюзев В.В.

" ____ " _____ 2013 г.

Исследование процессов установки и загрузки.
Методические указания по выполнению лабораторной работы
по дисциплине "Операционные системы"

Часть 2.

Исследование процесса загрузки.

МОСКВА 2013

Часть 2. Исследование процесса загрузки.

Загрузка современных ОС является достаточно сложным процессом. По успешному завершению данного процесса можно судить о степени готовности различных аппаратных устройств.

Этапы загрузки как одного семейства ОС, так и различных семейств может отличаться.

Цель 2-й части - исследование основных фаз процесса загрузки ОС семейства Windows и Linux.

Продолжительность работы - 4 часа.

Теоретические сведения

В данной работе рассматриваются:

- Процесс загрузки ОС на примерах Windows и Linux;
- Консольный интерфейс Linux и консольный интерфейс DOS.
- Оконная система X Window System. Оконный менеджер Openbox.

Процесс загрузки Linux.

Процесс загрузки Linux происходит в 6 фаз (см. рис. 1):

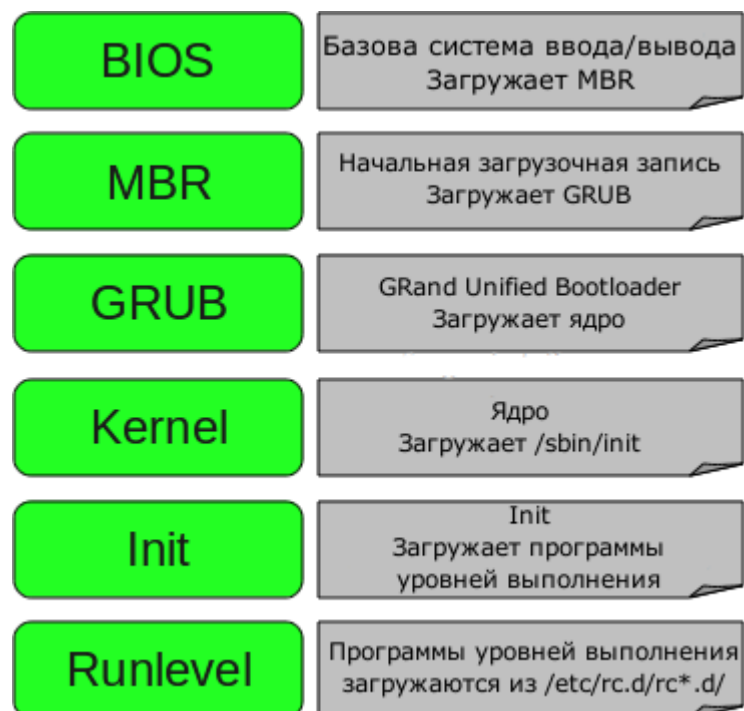


Рисунок 1 – Процесс загрузки Linux.

Не существует строгой последовательности загрузки, поэтому загрузка некоторых дистрибутивов отличается от той, что представлена на рисунке 1.

Однако такая последовательность подходит для большинства популярных дистрибутивов. Отличия могут быть на 2-3 этапе (LILO или другой загрузчик вместо GRUB), 5 этапе (отличия в поведении init).

Первая фаза загрузки совпадает для Windows и Linux.

Загрузчик первой ступени.

Загрузчик первой ступени хранится в главной загрузочной записи на жёстком диске (MBR – Master Boot Record). Строение этой записи показано на рис.2.

Первичный начальный загрузчик представляет собой образ размером 512 байт, который содержит как программный код, так и небольшую таблицу разделов. Первые 446 байт представляют собой собственно первичный загрузчик, который содержит как программный код, так и текст сообщений об ошибках. Следующие 64 байта представляют собой таблицу разделов, которая содержит запись для каждого из четырех разделов диска (по 16 байт каждая). В конце MBR располагаются два байта, которые носят название "магического числа" (0xAA55). Это магическое число служит для целей проверки MBR.

Задача первичного загрузчика - отыскать и загрузить вторичный загрузчик (загрузчик второй ступени). Он делает это, просматривая таблицу разделов в поиске активного раздела. Когда первичный загрузчик обнаруживает активный раздел, он просматривает оставшиеся разделы с целью убедиться, что они не являются активными. После завершения этой проверки с устройства в оперативную память считывается загрузочная запись активного раздела.

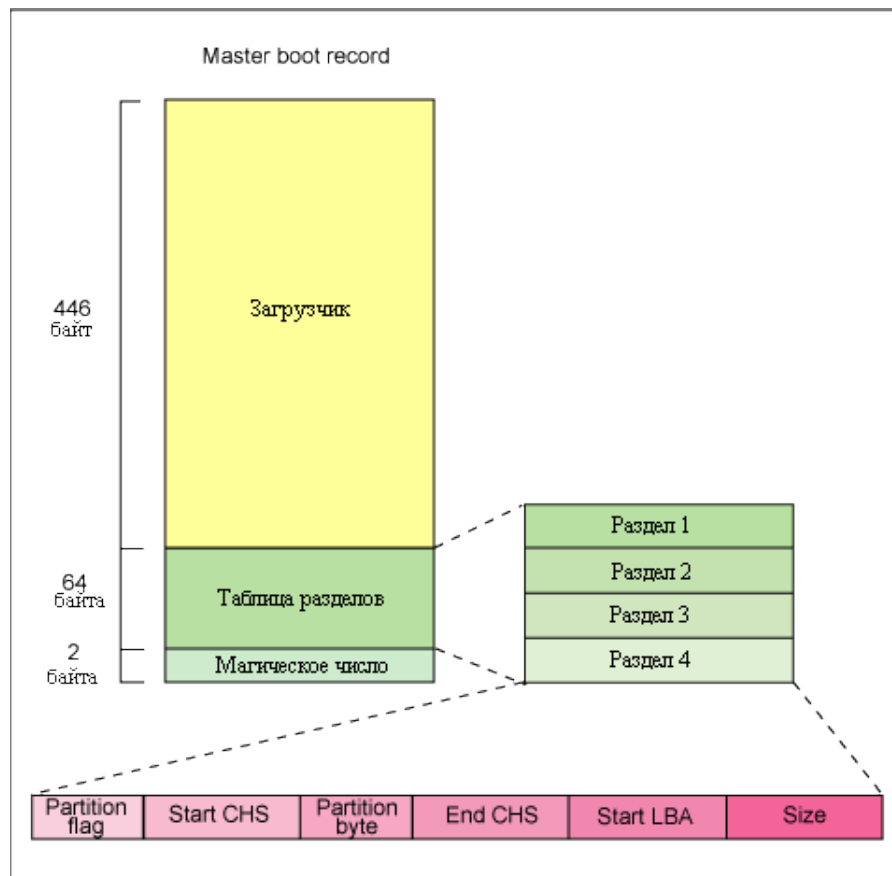


Рисунок 2 – Строение MBR.

Вторичный загрузчик.

Вторичный загрузчик или загрузчик второй ступени также иногда называют загрузчиком ядра. Его задачей на данной стадии является загрузка ядра Linux и, возможно, загрузка начального RAM-диска. Конфигурационный файл Grub обычно лежит в `/boot/grub/grub.conf` (так же `/etc/grub.conf` может быть символьной ссылкой на него). Пример:

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title Xubuntu (2.6.18-194.el5PAE)
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/
    initrd /boot/initrd-2.6.18-194.el5PAE.img
```

После того как загрузчик второй стадии загружен в память, он обращается к файловой системе и выполняет загрузку в память установленного по умолчанию образа ядра и образа `initrd`. Когда эти образы готовы к работе, загрузчик 2-й стадии вызывает образ ядра.

Загрузка ядра

На рисунке 3 представлен процесс загрузки ядра (для Arch Linux).

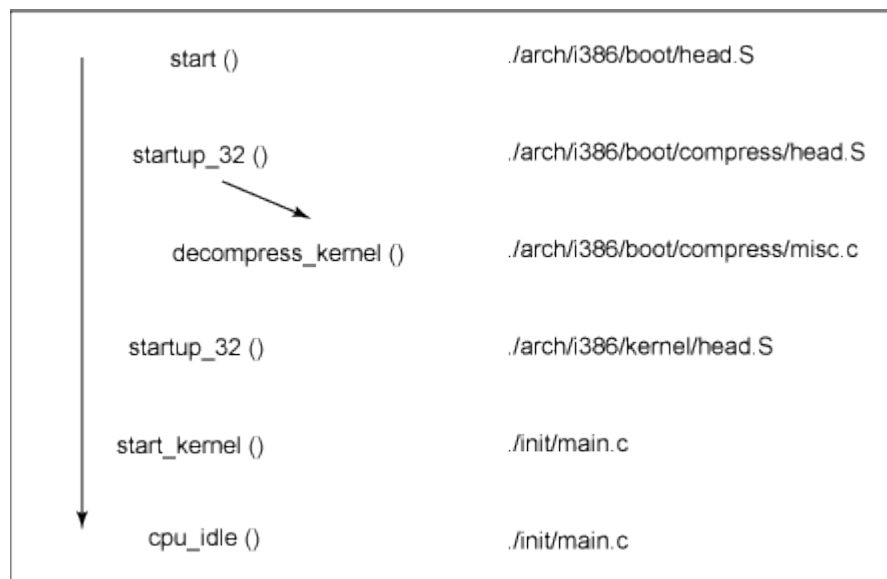


Рисунок 3 – Загрузка ядра Linux (i386).

При обращении к `start_kernel` вызывается длинный список функций инициализации, которые выполняют настройку прерываний, производят дальнейшее конфигурирование памяти и загружают начальный RAM-диск. После этого вызывается функция `kernel_thread` (из `arch/i386/kernel/process.c`), запускающая функцию `init`, которая является первым процессом, выполняющимся в пространстве пользователя. В заключение запускается `idle task`, после чего управление может взять на себя планировщик (`scheduler`) (после вызова `cpu_idle`). Если разрешены прерывания, вытесняющий планировщик (`pre-emptive scheduler`) будет периодически перехватывать контроль для поддержки многозадачности.

В процессе загрузки ядра выполняется загрузка в оперативную память и монтирование начального RAM-диска (`initrd`), который был загружен в память загрузчиком 2-й ступени. Данный `initrd` служит временной корневой файловой

системой в оперативной памяти и позволяет ядру полностью загрузиться, не выполняя монтирование каких-то физических дисков. Так как модули, необходимые для взаимодействия с периферийными устройствами, могут являться частью `initrd`, то ядро получается очень компактным и тем не менее способно поддерживать самые разнообразные аппаратные конфигурации. После загрузки ядра корневая файловая система заменяется (при помощи `pivot_root`); при этом корневая файловая система `initrd` удаляется и монтируется действительная корневая файловая система.

Функция `initrd` позволяет создать компактное ядро Linux, где драйверы скомпилированы как загружаемые модули. Эти загружаемые модули обеспечивают доступ ядра к дискам и файловым системам, которые имеются на этих дисках. Также имеются драйверы для других аппаратных устройств. Так как корневая файловая система представляет собой *файловую систему* на диске, то функция `initrd` обеспечивает для загрузчика возможность обратиться к диску и смонтировать действительную корневую файловую систему. Во встраиваемой системе без жесткого диска `initrd` может представлять собой окончательную файловую систему, или же окончательная файловая система может монтироваться при помощи сетевой файловой системы (Network File System, NFS).

Конфигурация ОС после запуска зависит от уровня выполнения (run level). В Linux 8 уровней выполнения:

- 0 — остановка системы;
- 1 — однопользовательский режим (для специальных случаев администрирования);
- 2 — многопользовательский режим без NFS (то же, что и 3, если компьютер не работает с сетью);
- 3 — полный многопользовательский режим;
- 4 — использование не регламентировано;
- 5 — обычно используется для запуска системы в графическом режиме (X11);

- 6 — перезагрузка системы;
- S (или s) — примерно то же, что и однопользовательский режим, но S и s используются в основном в скриптах.

Конфигурация `init` хранится в файле `/etc/inittab`. Этот файл состоит из записей следующего вида:

`id:runlevels:action:process`

где:

- `id` — идентификатор строки. Это произвольная комбинация, содержащая от 1 до 4 символов. В файле `inittab` не может быть двух строк с одинаковыми идентификаторами;

- `runlevels` — уровни выполнения, на которых эта строка будет задействована.

Уровни задаются цифрами или буквами без разделителей;

- `process` — процесс, который должен запускаться на указанных уровнях. Другими словами, в этом поле указывается имя программы, вызываемой при переходе на указанные уровни выполнения;

- `action` — действие.

В поле `action` стоит ключевое слово, которое определяет дополнительные условия выполнения команды, заданной полем `process`. Допустимые значения поля `action`:

- `respawn` — перезапустить процесс в случае завершения его работы;
- `once` — выполнить процесс только один раз при переходе на указанный уровень;
- `wait` — процесс будет запущен один раз при переходе на указанный уровень и `init` будет ожидать завершения работы этого процесса, прежде, чем продолжать работу;
- `sysinit` — это ключевое слово обозначает действия, выполняемые в процессе загрузки системы независимо от уровня выполнения (поле `runlevels` игнорируется). Процессы, помеченные этим словом, запускаются до процессов, помеченных словами `boot` и `bootwait`;

- `boot` — процесс будет запущен на этапе загрузки системы независимо от уровня выполнения;
- `bootwait` — процесс будет запущен на этапе загрузки системы независимо от уровня выполнения, и `init` будет дожидаться его завершения;
- `initdefault` — строка, в которой это слово стоит в поле `action`, определяет уровень выполнения, на который система переходит по умолчанию. Поле `process` в этой строке игнорируется. Если уровень выполнения, используемый по умолчанию, не задан, то процесс `init` будет ждать, пока пользователь, запускающий систему, не введет его с консоли;
- `off` — игнорировать данный элемент;
- `powerwait` — позволяет процессу `init` остановить систему, когда пропало питание. Использование этого слова предполагает, что имеется источник бесперебойного питания (UPS) и программное обеспечение, которое отслеживает состояние UPS и информирует `init` о том, что питание отключилось;
- `ctrlaltdel` — разрешает `init` перезагрузить систему, когда пользователь нажимает комбинацию клавиш `<Ctrl>+<Alt>+` на клавиатуре. Администратор может определить действия по комбинации клавиш `<Ctrl>+<Alt>+`, например, игнорировать нажатие этой комбинации.

Некоторые другие файлы, использующиеся в процессе загрузки:

- `/etc/modules.conf` (или `/etc/conf.modules`) — файл, определяющий конфигурацию загружаемых модулей;
- `/etc/fstab` — содержит информацию, необходимую для автоматического монтирования файловых систем;
- `/etc/passwd` — различная регистрационная информация, включая пароли;
- `/etc/profile` — глобальный файл профилей — устанавливает переменную `$PATH` и другие важнейшие переменные;
- `/etc/bashrc` — глобальный файл конфигурации `bash`, устанавливает синонимы (алиасы) и функции, и т.п.;
- `/etc/issue` — содержит сообщение, выдаваемое на терминал перед входом в систему (перед запросом имени и пароля); редактировать этот файл с целью

изменения текста сообщения не стоит, он формируется инициализационным скриптом `/etc/rc.d/rc.local`;

- `/etc/motd` — устанавливает сообщение, выдаваемое пользователю после входа в систему (после правильного ввода пароля);

Данная фаза примерно соответствует второй фазе загрузки Windows (для Windows NT загрузчик ядра – NTLDR):

1. `ntoskrnl.exe` (запуск ядра)
2. `hal.dll`
3. `kdcom.dll` (Kernel Debugger HW Extension DLL)
4. `bootvid.dll` (отображает логотип Windows при загрузке и полосу загрузки)

5. `config\system registry`

- a. `HKLM\SYSTEM\CurrentControlSet\Control\Session`

`Manager\BootExecute`

- b. Запуск сервисов в заданном порядке

- c. `*HKLM\SYSTEM\CurrentControlSet\Control\ServiceGroupOrder`

Практическая часть.

Для проведения данной лабораторной работы требуется виртуальная машина VirtualBox с установленным базовым дистрибутивом Debian. В ходе данной ЛР будет рассмотрен процесс загрузки ОС, после чего будет рассмотрен консольный интерфейс Linux на примере ряда программ, затем будут установлены пакеты X Window System и Openbox.

Процесс загрузки.

На рисунке 4 показан экран загрузки Linux.

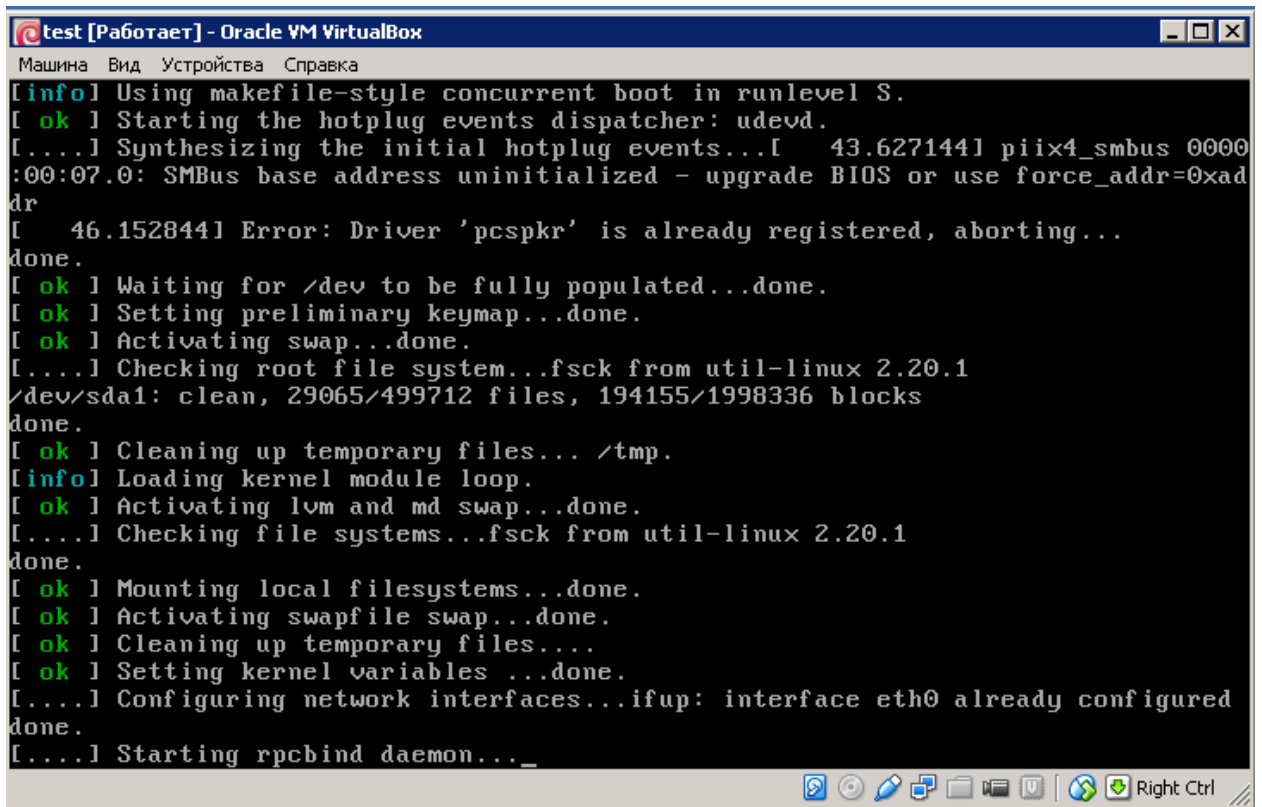


Рисунок 4 – Процесс загрузки Linux.

После загрузки в данной ЛР требуется войти как суперпользователь (root).

Можно выполнить команду `ps tree` и убедиться, что `init` является родителем всех процессов (рисунок 5).

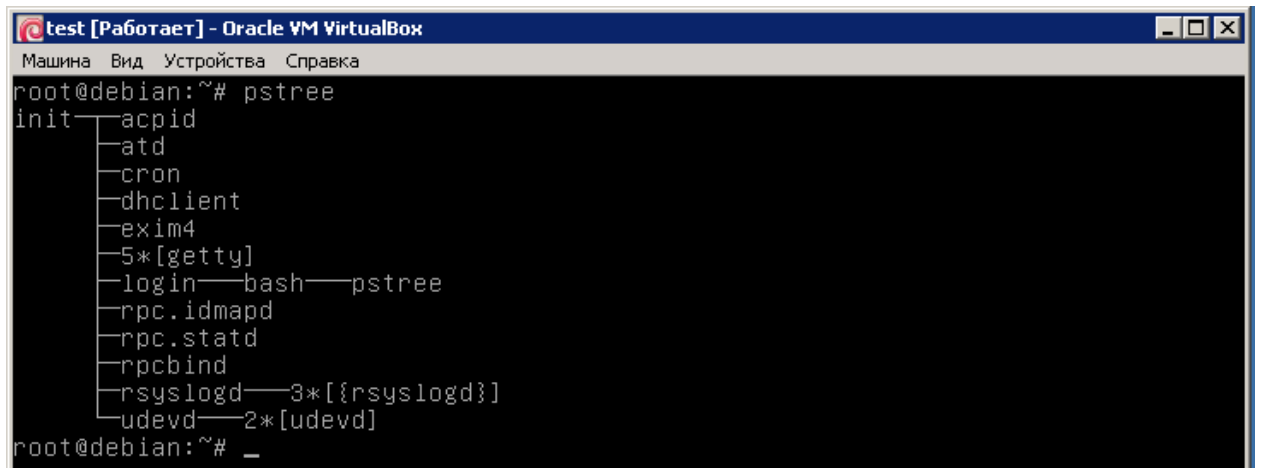


Рисунок 5 – Дерево процессов.

В UNIX-подобных системах существуют журналы событий. В дальнейшем в ходе ЛР будет рассмотрен один из таких системных журналов.

Текстовый редактор vi.

vi – текстовый редактор в UNIX-подобных системах, первая версия была написана в 1976 году. Т.к. в процессе установки Debian (LP1) зеркало архивов сети было отключено, то для использования сетевых репозиторий необходимо исправить файл sources.list. Это один из конфигурационных файлов утилиты apt, в котором хранится информация о подключенных репозиториях. Существует несколько способов добавлений репозиторий (например, с использованием менеджера пакетов Synaptic, графической утилиты, или с помощью консольной утилиты apt). Для ознакомления с возможностями консольных приложений будет использоваться самый «низкоуровневый» метод – правка конфигурационного файла текстовым редактором.

Чтобы открыть конфигурационный файл, необходимо ввести следующую команду в консоль:

```
vi /etc/apt/sources.list
```

Если не используются сетевые репозитории, то открытый файл будет выглядеть как на рисунке 6.

```

#
# deb cdrom:[Debian GNU/Linux 7.2.0 _wheezy_ - Official i386 lxde-CD Binary-1 20
131012-12:56]/ wheezy main
# deb cdrom:[Debian GNU/Linux 7.2.0 _wheezy_ - Official i386 lxde-CD Binary-1 2013
1012-12:56]/ wheezy main
deb http://security.debian.org/ wheezy/updates main
deb-src http://security.debian.org/ wheezy/updates main
# wheezy-updates, previously known as 'volatile'
# A network mirror was not selected during install. The following entries
# are provided as examples, but you should amend them as appropriate
# for your mirror of choice.
#
# deb http://ftp.debian.org/debian/ wheezy-updates main
# deb-src http://ftp.debian.org/debian/ wheezy-updates main
~
~
~
~
~
"/etc/apt/sources.list" 16 lines, 667 characters

```

Рисунок 6 – Изначальная конфигурация репозиторий.

Положение репозитория в файле определяет его приоритет, поиск производится с начала файла, поэтому репозитории обычно сортируют по скорости соединения (первыми идут репозитории на cd-rom и т.д.).

Некоторые команды vi, которые могут быть использованы в данной ЛР:

a/i / A/I вставка за/перед _ / концом/началом строки

R режим замены

r буква на букву

s буква на буквы

o/O вставить строку под/над _

x уничтожить символ

d уничтожить фрагмент

:set nu / nonu нумеровать/не нумеровать строки

: [1,\$] w [>>] [file] записать в file [от 1 до \$ строки]

:q[!] закончить сеанс [форсировано]

Для редактирования файла необходимо выполнить следующую последовательность действий:

1. Ввести :set nu (включение нумерации строк). Курсор окажется на строке 1.
2. Перевести курсор стрелками на начало строки 5.
3. Нажать i (ввод перед текущим символом).
4. Ввести символ #. Это запретит использование репозитория на cd-rom.
5. Стрелками перевести курсор до строки 6.
6. Нажать o (это вставит новую строку снизу и переведёт курсор на неё).
7. Напечатать следующую строку:
deb <http://mirror.mephi.ru/debian/> wheezy main
8. Нажать любую клавишу со стрелкой (это закончит режим ввода набивкой).
9. Нажать o

10. Напечатать следующую строку:

deb-src <http://mirror.mephi.ru/debian/> wheezy main

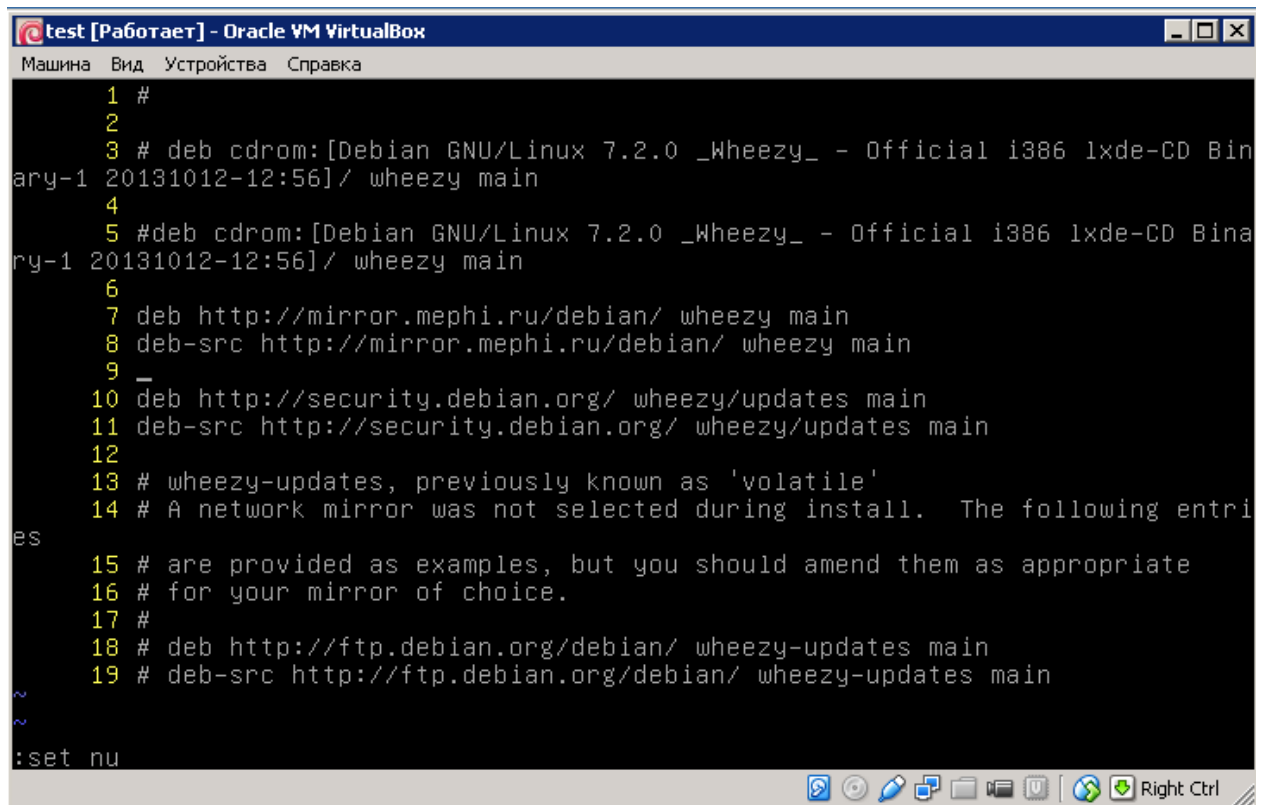
11. Нажать любую клавишу со стрелкой.

12. Нажать o.

13. Нажать любую клавишу со стрелкой. Конфигурационный файл должен выглядеть так, как показано на рисунке 7.

14. Ввести :w (это сохранит изменения).

15. Ввести :q (выход из редактора).



```

1 #
2
3 # deb cdrom:[Debian GNU/Linux 7.2.0 _Wheezy_ - Official i386 1xde-CD Bin
ary-1 20131012-12:56]/ wheezy main
4
5 #deb cdrom:[Debian GNU/Linux 7.2.0 _Wheezy_ - Official i386 1xde-CD Bina
ry-1 20131012-12:56]/ wheezy main
6
7 deb http://mirror.mephi.ru/debian/ wheezy main
8 deb-src http://mirror.mephi.ru/debian/ wheezy main
9 _
10 deb http://security.debian.org/ wheezy/updates main
11 deb-src http://security.debian.org/ wheezy/updates main
12
13 # wheezy-updates, previously known as 'volatile'
14 # A network mirror was not selected during install. The following entri
es
15 # are provided as examples, but you should amend them as appropriate
16 # for your mirror of choice.
17 #
18 # deb http://ftp.debian.org/debian/ wheezy-updates main
19 # deb-src http://ftp.debian.org/debian/ wheezy-updates main
~
:set nu

```

Рисунок 7 – Новая конфигурация репозитория.

После этого необходимо обновить индекс пакетов. Для этого в терминале надо выполнить следующую команду:

```
apt-get update
```

Выполнение этой команды может занять некоторое время.

Текстовый браузер w3m.

Текстовый браузер w3m появился в 1995 году. Несмотря на то, что текстовые браузеры, как правило, предназначены только для просмотра текста

в однооконном режиме, w3m предлагает достаточно широкие возможности: вкладки, загрузку изображений, работу с мышкой, контекстное меню.

Для запуска w3m из консоли необходимо просто ввести в консоли: w3m

Для запуска w3m сразу с конкретного адреса необходимо дописать этот адрес в команде, например:

w3m bmstu.ru

На рисунке 8 показан пример отображения страницы в браузере w3m.

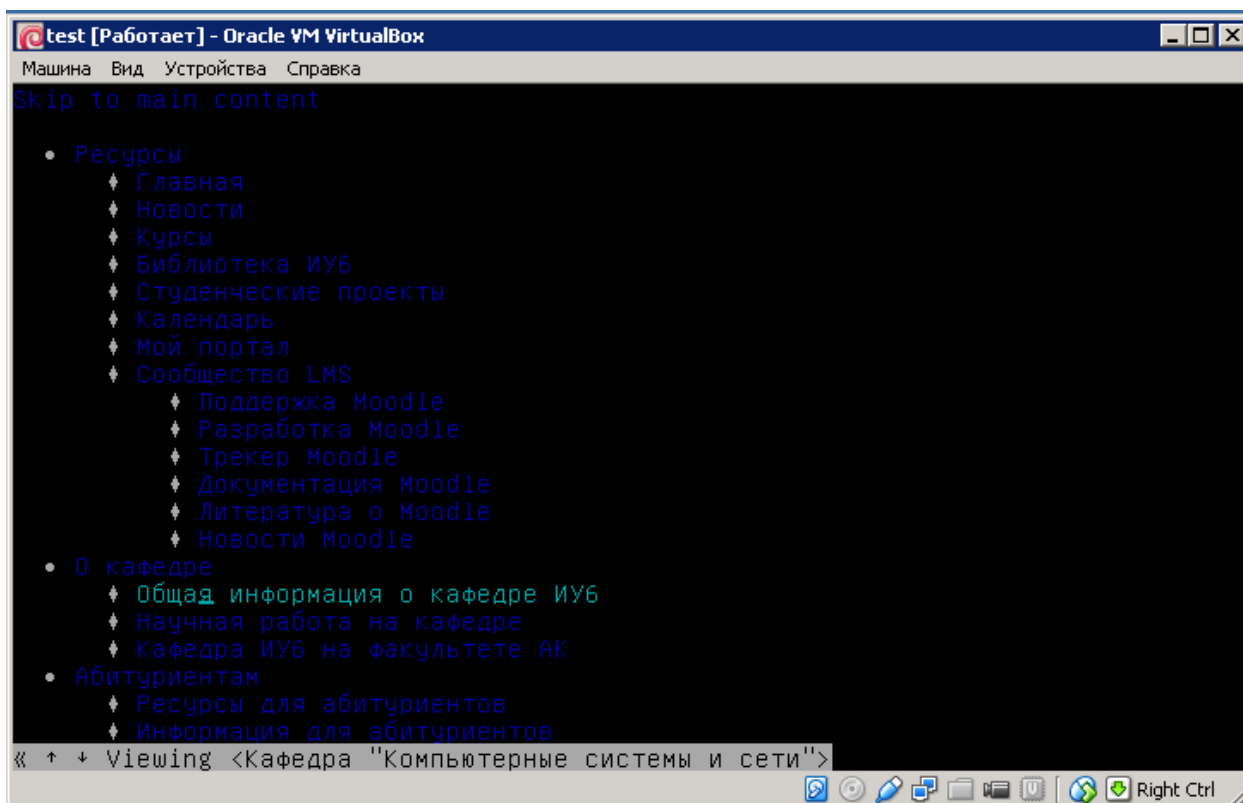


Рисунок 8 – Сайт кафедры ИУ6 в текстовом браузере w3m.

Основные команды w3m:

Shift-U – открывает строку ввода URL

Shift-B – переход на предыдущую страницу

Shift-T – открыть новую вкладку

Shift-[– предыдущая вкладка

Shift-] – следующая вкладка

Shift-H – страница помощи

q – выход

Поле для ввода текста выбирается путём наведения на него курсора и нажатия клавиши ввода, после чего внизу экрана появляется приглашение на ввод текста.

В отличие от DOS, консольный интерфейс UNIX подразумевает многозадачность. Для того чтобы свернуть приложение, необходимо нажать Ctrl-Z, после чего на экран консоли будет выведено сообщение, показанное на рисунке 9.

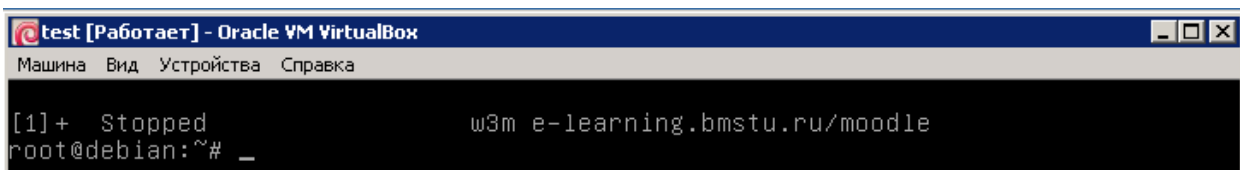


Рисунок 9 – w3m отправлен в фоновый режим.

Цифра в квадратных скобках обозначает номер задания, он необходим для некоторых команд.

Можно удостовериться, что приложение продолжает выполнение, введя, например, `ps -a` (рисунок 10).

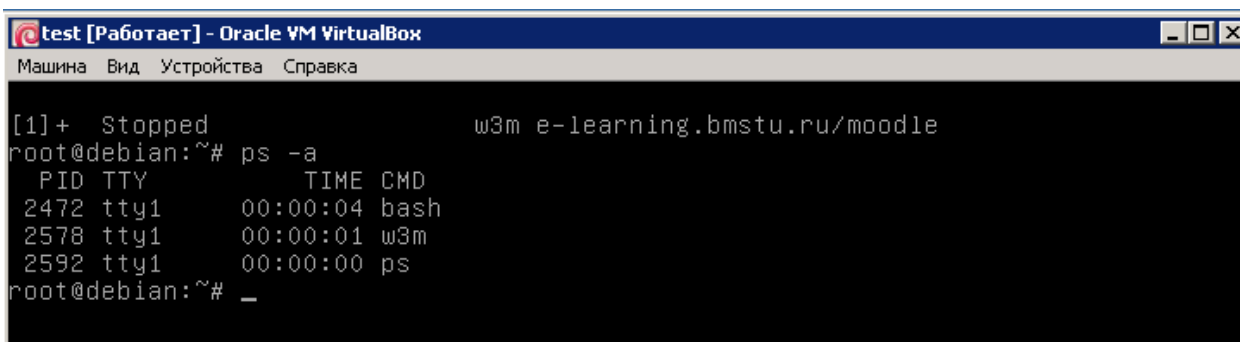


Рисунок 10- Список запущенных пользователем процессов.

Развернуть приложение можно с помощью команды `fg` (foreground), имеющей следующий синтаксис:

`fg %id`

где `id` – номер задания. Чтобы развернуть обратно `w3m`, необходимо ввести следующую команду:

`fg %1`

Установка X Window System и Openbox.

X Window System представляет собой оконную систему, определяющую стандартные интерфейсы и протоколы для работы с графическим интерфейсом пользователя. В данной ЛР будет использована открытая реализация X.Org. Для установки X.Org нужно выполнить следующую команду:

```
apt-get install xorg
```

Данная операция может занять достаточно много времени.

Openbox представляет собой оконный менеджер для X Window System. Он может использоваться как совместно с каким-либо окружением рабочего стола (например, KDE), так и отдельно. В данной ЛР мы будем использовать Openbox отдельно от DE.

После установки оконной системы требуется установить Openbox. Для этого нужно выполнить следующую команду: `apt-get install openbox`

Чтобы активировать графический интерфейс, необходимо запустить x-сервер: `startx`

В результате должен появиться серый рабочий стол с мышкой. По нажатию на правую кнопку мыши должно появиться меню, показанное на рисунке 11.

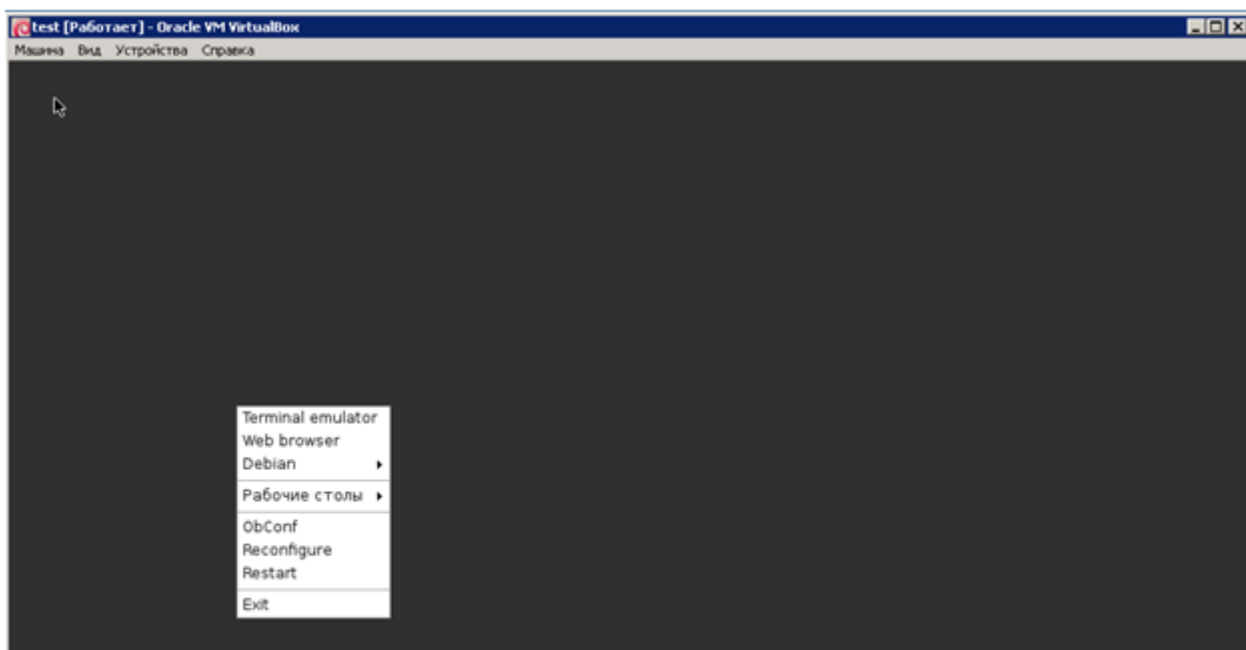


Рисунок 11 – Меню в Openbox.

Если в меню нет пункта «Debian», возможно, отсутствуют некоторые необходимые пакеты:


```
apt-get install menu
```

```
apt-get install menu-xdg
```

После этого в контекстном меню необходимо выбрать «Restart». Если пункт «Debian» не появился и после этих действий, нужно выполнить следующие команды:

```
mkdir ~/.config
```

```
mkdir ~/.config/openbox
```

```
cp /etc/xdg/openbox/rc.xml ~/.config/openbox/
```

```
cp /etc/xdg/openbox/menu.xml ~/.config/openbox/
```

```
cp /etc/xdg/openbox/autostart ~/.config/openbox/
```

Первые две команды создадут для пользователя root папку `./config/openbox`, остальные три скопируют туда настройки из общей папки настроек Openbox в папку настроек конкретного пользователя.

После выполнения этих команд снова выбрать пункт «Restart», после чего необходимый пункт меню должен появиться.

Текстовый браузер w3m в Openbox.

В консоли xterm браузер w3m способен загружать изображения. Для этого необходимо поставить пакет w3m-img:

```
apt-get install w3m-img
```

После успешной установки можно запустить браузер. Сделать это можно либо из консоли (как описывалось выше), либо из контекстного меню:

Debian -> Приложения -> Сеть -> Просмотр веб -> w3m

Пример показан на рисунке 12.

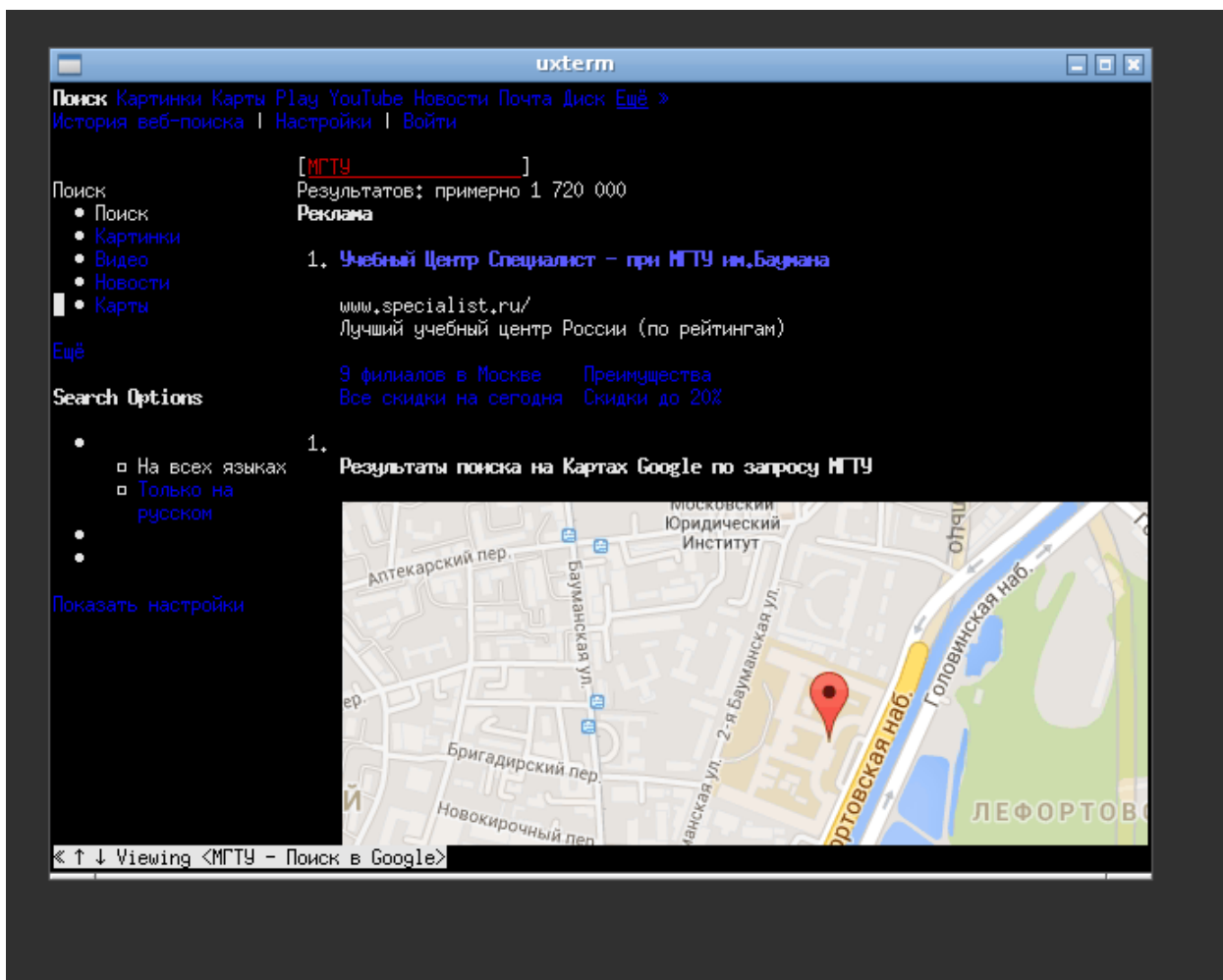


Рисунок 12 – Отображение графики в w3m.

Также в xterm (uxterm) доступно контекстное меню браузера. Для отображения этого меню необходимо нажать правую клавишу мыши.

Просмотр syslog.

syslog представляет собой системный журнал, собирающий в себя все сообщения. Он может быть гибко настроен.

Для просмотра syslog установим блокнот leafpad, т.к. просмотр такого большого файла в vi или других имеющихся утилитах не очень удобен. Установка leafpad:

```
apt-get install leafpad
```

После установки в консоли необходимо выполнить команду, открывающую системный журнал:

```
leafpad /var/log/syslog
```

Пример syslog показан на рисунке 13.

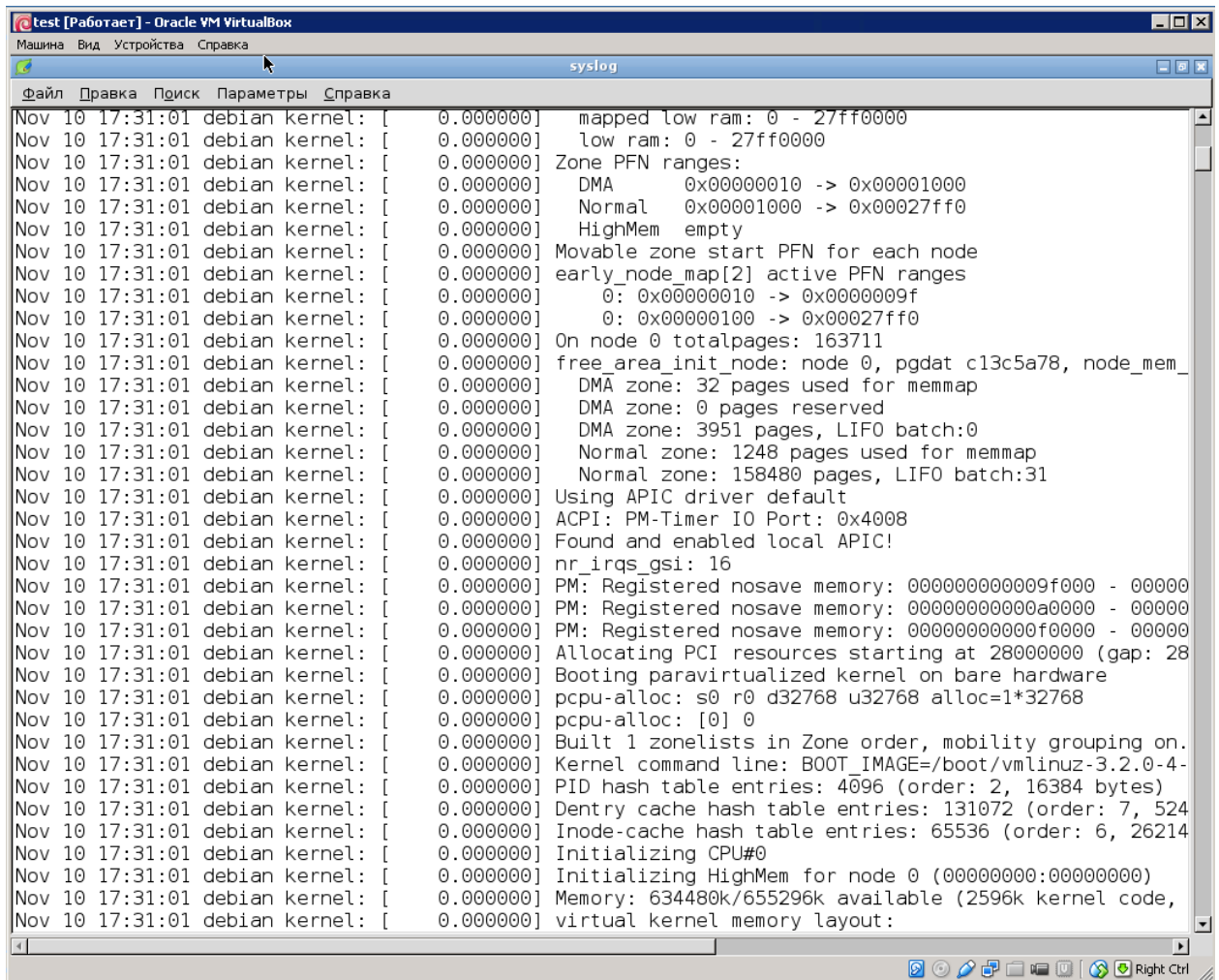


Рисунок 13 – Системный журнал в leafpad.

С помощью поиска можно найти основные события.

Структура системного журнала определяется стандартом RFC. В Debian используется rsyslog (за журналирование отвечает демон rsyslogd), позиционируемый как улучшенная версия syslog, совместимая с оригиналом.

Помимо syslog существуют и другие журналы. Список их можно посмотреть, если ввести в консоли команду:

```
ls /var/log
```

На рисунке 14 представлены папки и файлы журналов, содержащиеся в установленной системе.

```
root@debian:~# ls /var/log
alternatives.log  dmesg      fontconfig.log  mail.info      wtmp
apt               dmesg.0    fsck            mail.log       Xorg.0.log
aptitude         dmesg.1.gz installer      mail.warn     Xorg.0.log.old
auth.log         dmesg.2.gz kern.log       messages
btmtp           dpkg.log   lastlog        news
daemon.log     exim4      lpr.log        syslog
debug          faillog    mail.err       user.log
root@debian:~#
```

Рисунок 14 – Журналы в Debian.

Журнал `auth.log`, помимо информации об авторизации пользователей, содержит информацию о добавлении групп (такая запись появилась после установки X.Org), выходе пользователей из системы и т.д.

В файле `/var/log/apt/history.log` содержится информация о работе утилиты `apt`.

Система журналирования в UNIX-системах является очень гибким инструментом. Например, можно настроить сбор сообщений с удалённых клиентов в централизованный журнал, располагающийся на сервере, используя протоколы UDP (для `syslog`) или TCP (для `rsyslog`).

Загрузка в Windows.

В старых версиях, например, в Windows 98 четыре фазы загрузки:

- * начальной загрузки под управлением BIOS;
- * загрузки драйверов MS-DOS и резидентных программ;
- * инициализации статических VxD в реальном режиме;
- * передачи управления операционной системе защищенного режима и загрузки остальных VxD- драйверов.

Начальная загрузка под управлением BIOS.

Microsoft и ряд производителей оборудования совместно определили спецификацию Plug and Play BIOS, описывающую взаимодействие между базовой системой ввода/вывода Plug and Play, устройствами Plug and Play и дополнительными ПЗУ (иногда называемыми ПЗУ адаптеров). Plug and Play BIOS поддерживает и конфигурирует загрузочные устройства Plug and Play. Кроме этого, она передает специальную информацию Диспетчеру конфигурации в Windows для настройки отдельных адаптеров и устройств.

Загрузка при обычной BIOS на компьютерах, не имеющих Plug and Play BIOS, базовая система ввода/вывода инициализирует все устройства на шине ISA. ISA- платы Plug and Play с собственным ПЗУ должны начинать работу при загрузке компьютера с включением своего ПЗУ.

Загрузка при Plug and Play BIOS использует информацию из энергозависимого ОЗУ, чтобы узнать, какие ISA- платы Plug and Play надо включить, следует ли отображать их ПЗУ и какие нужно выделять им адреса ввода/вывода, каналы DMA и прочие ресурсы. DMA это канал прямого доступа к памяти, позволяющий обойтись без участия микропроцессора. При этом, обмен данными осуществляется непосредственно между памятью и дисковым устройством.

Затем BIOS программирует платы Plug and Play - до того, как начнется самотестирование при включении (power-on self-test, POST). Все платы, конфигурация которых не хранится в BIOS, полностью отключаются для снижения вероятности конфликтов.

Plug and Play BIOS конфигурирует также все устройства на материнской плате. Диспетчер конфигурации может включить некоторые устройства или изменить выделенные им ресурсы.

Загрузка профилей оборудования и драйверов реального режима.

После инициализации BIOS операционная система пытается определить текущую конфигурацию, в том числе - не является ли компьютер стыковочной станцией. Это

осуществляется с помощью профиля оборудования, выбираемого Windows 98 перед обработкой файла CONFIG.SYS. Профиль выстраивается при анализе оборудования, в процессе которого собирается информация об используемых прерываниях, последовательных и параллельных портах BIOS, идентификации BIOS компьютера, данные о Plug and Play BIOS стыковочной станции и, если удастся, OEM - специфические данные о стыковочной станции. Затем аналитический модуль формирует двухбайтовое значение, называемое текущим профилем оборудования или текущей конфигурацией.

У каждого профиля есть свое имя, совпадающее с пунктом меню верхнего уровня в мультikonфигурационном файле CONFIG.SYS. (т.е. с текстом меню, а не с именем раздела в квадратных скобках). Windows 98 автоматически выбирает пункт из этого меню и обрабатывает соответствующий раздел CONFIG.SYS.

Именно в этот момент и начинается обработка файлов CONFIG.SYS и AUTOEXEC.BAT. В Windows 98 они не нужны и используются для совместимости с программами MS-DOS и Windows 3.x. Windows 98 обрабатывает эти файлы практически так же, как и MS-DOS 6.x. Указанные в них драйверы и резидентные программы загружаются в реальном режиме.

Инициализация статических VxD.

VxD - это драйвер виртуального устройства (virtual device driver). Буква x обозначает тип устройства, например VDD - драйвер виртуального устройства для дисплея, а VPD - драйвер виртуального устройства для принтера.

В результате для работы с устройством у нас есть как минимум два уровня. Самый нижний это минидрайвер производителя, который непосредственно работает с оборудованием и знает как оно управляется. Операционная система работает непосредственно с **VXD** и отдает ему команды, а сам **VXD** уже по мере необходимости обращается к минидрайверу.

Сама **Windows** может поддерживать **VXD** двух типов исходя из загрузки: статические и динамические.

Статические загружаются при старте системы, а динамические в любой момент. В момент старта системы (**win.com**) запускается **vmm32.vxd**, который запускает остальные драйвера (в каталоге **windows/system** их много). Згружаемые **VXD** драйвера перечислены по пути `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD`

В состав VMM32.VXD входит загрузчик реального режима, Диспетчер виртуальной машины и большинство статических VxD.

Рассмотрим какие VxD объединяются в файл VMM32.VXD в типичном случае (Точный их список для каждого компьютера будет своим). Эти драйверы обычно указываются в разделе [386enh] файла SYSTEM.INI.

Типичные VxD, объединяемые в файл VMM32.INI.

*vmouse	*configmg	*vwin32	*vfbackup	*vcomm	*ifsmgr
*ios	*vfat	*vcache	*vcond	*int13	*vxdldr
*vdef	*dynapage	*reboot	*vsd	*parity	*biosxlat
*vmcpd	*vkd	*vdd	*ebios	*vtdapi	

VMM32 загружает VxD- драйверы в три этапа:

1. VMM32 загружает базовые драйверы, указанные в реестре, который содержит записи для каждого VxD, не связанного напрямую с конкретным оборудованием. VxD находятся в следующей ветви реестра:

Hkey_Local_Machine\System\CurrentControlSet\Services\VxD

2. Если VMM32 находит в каком-либо разделе реестра параметр **StaticVxD=**, этот драйвер загружается и инициализируется в реальном режиме.

Например, следующая запись загружает *V86MMGR:

SYSTEM\CurrentControlSet\Services\VxD\V86MemoryManger
Description=MS-DOS Virtual 8086 Memory Manager
Manufacturer=Microsoft
StaticVxD=*V86MMGR
EMMEXCLUDE=E000-EFFF

3. VMM32 загружает статические VxD, указанные в строках **device=*VxD** в разделе [386enh] файла SYSTEM.INI. Такие VxD на самом деле загружаются из VMM32 и присутствуют в SYSTEM.INI только для совместимости.

Если какое-то устройство конфликтует с устройством, загруженным по информации из реестра, то устройство, заданное в SYSTEM.INI, имеет преимущество. Однако, если указанное в SYSTEM.INI устройство не будет найдено, то произойдет ошибка.

Многие модели построения драйверов Windows 98 вроде IOS (для драйверов дисков) и сетевые драйверы поддерживают динамическую загрузку. Подобные VxD загружаются не загрузчиком реального режима из VMM32, а загрузчиком устройства, отвечающим за загрузку и инициализацию драйверов в надлежащее время в надлежащем порядке.

Например, для минипорт- драйверов SCSI - адаптера загрузчиком устройства является *IOS. Записи, относящиеся к SCSI- адаптеру, находятся в разделе реестра :

Hkey_Local_Machine\System\CurrentControlSet\Services\Class

Поскольку запись **StaticVxD=xxx** в этом разделе реестра отсутствует, загрузчик реального режима VMM32 ничего не делает, обнаружив данное устройство.

Диспетчер конфигурации пытается найти все узлы устройств, по которым в реестре имеется запись **DevLoader=**. Загрузчик устройства (в предыдущем примере - *IOS) просматривает реестр, находит запись **PortDriver=**, загружает данный и любые связанные с ним драйверы, после чего инициализирует адаптер.

Загрузка операционной системы защищенного режима.

В предыдущей фазе были загружены следующие элементы ОС:

- * WIN.COM - управляет начальной проверкой и загружает компоненты ядра windows 98;
- * VMM32.VXD - создает виртуальные машины и начинает загрузку VxD- драйверов;
- * SYSTEM.INI - в нем отыскиваются записи, отличающиеся от записей в реестре.

Загрузив все статические VxD, VMM32.VXD переключает процессор в защищенный режим, и начинается последняя фаза: загрузка компонентов операционной системы, работающих в защищенном режиме.

Загрузка VxD- драйверов защищенного режима.

Импортируя информацию из Plug and Play BIOS, система инициализирует Диспетчер конфигурации защищенного режима. В противном случае Диспетчер конфигурации формирует дерево устройств Plug and Play путем перечисления устройств и загрузки динамически загружаемых драйверов устройств. Набор этих драйверов идентифицируется за счет загрузки их из особого каталога.

В следующей фазе система приступает к разрешению конфликтов между устройствами в дереве, после чего им сообщается об их конфигурации.

Загрузка других компонентов системы.

Оставшиеся системные компоненты Windows 98 загружаются в следующем порядке:

1. KERNEL32.DLL - содержит основные компоненты Windows, а KRNL386.EXE - загружает драйверы устройств.
2. GDI.EXE и GDI32.EXE - содержат код интерфейса графического устройства.
3. USER.EXE и USER32.EXE - содержат код пользовательского интерфейса.
4. Ресурсы, связанные с пользовательским интерфейсом (шрифты и т.п.).
5. Проверка значений параметров в WIN.INI.
6. Компоненты оболочки и рабочего стола.

Далее на экране появляется приглашение к регистрации, в диалоговом окне которого вы вводите свое имя и пароль. Если Вы не зарегистрировались, то используются значения параметров по умолчанию. Если Windows 98 соответствующим образом сконфигурировать, унифицированную регистрацию на входе в Windows 98 можно использовать и для регистрации в сети.

Системные загрузочные файлы.

В процессе загрузки Windows 98 используются следующие файлы:

* IO.SYS - операционная система реального режима, замещающая MS-DOS. VMM32 и драйверы устройств получают управление от IO.SYS.

* MSDOS.SYS - содержит специальную информацию для Windows /98 и создается для совместимости с приложениями, требующие для своей установки присутствия этого файла.

* CONFIG.SYS и AUTOEXEC.BAT.

* SYSTEM.INI и WIN.INI.

* BOOTLOG.TXT - файл протокола загрузки системы.

Системный файл IO.SYS.

Windows 98 использует новый системный файл IO.SYS, заменяющий системные файлы MS-DOS (IO.SYS, MSDOS.SYS). Этот файл ОС реального режима содержит информацию, необходимую для запуска компьютера.

Для загрузки Windows 98 файлы CONFIG.SYS и AUTOEXEC.BAT больше не нужны, хотя и сохранены для совместимости с некоторыми программами и драйверами.

Ниже приведены драйверы, которые загружаются IO.SYS по умолчанию, если они найдены на жестком диске: HIMEM.SYS, IFSHLP.SYS, SETVER.EXE, DBLSPACE.BIN или DRVSPACE.BIN.

Системный файл MSDOS.SYS.

Windows 98 Setup создает в корневом каталоге загрузочного диска системный файл MSDOS.SYS с атрибутами «скрытый» и «только для чтения». В этом файле прописываются важнейшие пути поиска файлов Windows 98, включая реестр. В MSDOS.SYS поддерживается также раздел [Options], который можно добавить для настройки процесса загрузки.

Пример типичного содержимого этого файла со значениями по умолчанию приведен ниже.

```
[Paths]
WinDir=C:\WINDOWS
WinBootDir=C:\WINDOWS
HostWinBootDrv=C
[Options]
BootMulti=1
BootGUI=1
Network=0
```

Большинство значений в разделе [Options] - булевы, т.е. равны 1 (т.е. параметр активен) или равны 0 (параметр отключен).

Системные файлы CONFIG.SYS и AUTOEXEC.BAT.

В Windows 98 изменились как содержание, так и способ обработки файлов CONFIG.SYS и AUTOEXEC.BAT. Windows 98 автоматически загружает драйверы и устанавливает значения параметров по умолчанию, используя вместо CONFIG.SYS и AUTOEXEC.BAT : файл IO.SYS, реестр и другие механизмы.

Однако на компьютерах, использующих некоторые драйверы и резидентные программы реального режима, по-прежнему требуется их загрузка через CONFIG.SYS и AUTOEXEC.BAT. Кроме этого, эти файлы могут понадобиться для активизации дополнительных функций какого-то программного обеспечения. В тоже время отдельные такие функции можно разрешить иным способом, например, применение длинных командных строк допустимо активизировать за счет изменения свойств COMMAND.COM.

В CONFIG.SYS в дополнение к информации, хранящейся в IO.SYS, могут присутствовать записи, специфичные для каких либо программ. Эти записи обрабатываются в порядке их размещения в файле. Когда обработка CONFIG.SYS заканчивается, в память загружаются все драйверы устройств и исполняется экземпляр командного процессора COMMAND.COM.

Windows 98 загружает диспетчеры памяти сторонних разработчиков, если таковые присутствуют в CONFIG.SYS , однако некоторые из них могут приводить к ошибкам. Аналогично Windows 98 позволяет использовать и командные процессоры сторонних разработчиков, но при этом нельзя будет пользоваться длинными именами файлов (возможны и другие проблемы).

Значения многих часто используемых параметров CONFIG.SYS предопределены в Windows 98, поэтому программа установки удаляет из этого файла (используя ключевое слово REM, чтобы превратить строку в комментарий) целый ряд записей типа files, buffers и stacks, если они эквивалентны значениям по умолчанию.

При наличии AUTOEXEC.BAT каждая его строка обрабатывается в том порядке, в каком порядке она расположена в файле. AUTOEXEC.BAT может содержать и дополнительные записи, специфичные для конкретных программ.

Windows 98 передает командному процессору COMMAND.COM начальное окружение с уже прописанными в PATH каталогами Windows и Windows COMMAND и установленными переменными окружения PROMPT, TMP и TEMP.

Ниже приведены команды AUTOEXEC.BAT имеющие эквиваленты со значениями по умолчанию в IO.SYS ОС Windows 98.

* net start - загружает сетевые компоненты реального режима и проверяет связь с сетевым адаптером. Все сообщения об ошибках записываются в файл NDISLOG.TXT.

* set path - устанавливает указанный путь.

Системные файлы SYSTEM.INI и WIN.INI.

Большинство параметров конфигурации Windows 98 хранится в реестре и задавать их в SYSTEM.INI не нужно.

Например, все параметры из раздела [Сетевые_драйверы] и параметр lanabase= из раздела [nwnblink] перемещены в реестр и более не допускаются в SYSTEM.INI.

Ниже указаны параметры, которые следует устанавливать инструментами Windows :

- * Все параметры, относящиеся к настройке памяти (опция System);
- * Параметры аппаратных средств (Диспетчер устройств в опции System);
- * Все параметры сети (опция Network).

Информация о шрифтах и рабочем столе перемещена из WIN.INI в реестр, в частности:

- * Записи перемещаемые из раздела [Windows];
- * Записи перемещаемые из раздела [WindowsMetrics]

Следующие параметры нужно регулировать с помощью инструментов Windows: параметры мыши, клавиатуры и экрана.

Системный файл BOOTLOG.TXT.

В данном файле регистрируется процесс текущей загрузки Windows 98. Создается при установке, когда Windows 98 запускается впервые. Он содержит информацию о загруженных и инициализированных компонентах и драйверах Windows 98 и о состоянии каждого из них.

Нажав клавишу F8 для запуска системы в интерактивном режиме, можно потребовать ведение протокола загрузки в процессе запуска системы. А запуская WIN.COM из командной строки, можно указать и параметр /b , что поможет выявить проблемы с конфигурацией.

Информация записывается в BOOTLOG.TXT по порядку, и ее можно разбить на пять основных разделов. Ниже приведены разделы, которые помогают выявлять ошибки:

- * Loading real-mode drivers - загрузка драйверов реального режима;
- * Loading VxDs - загрузка VxD - драйверов;
- * System-critical initialization of VxDs - инициализация критических для системы драйверов;
- * Device initialization of VxDs - инициализация устройств VxD;
- * Successful VxD initialization - успешная инициализация VxD - драйверов. (необходимо убедиться, что в данном разделе есть записи: initcomplete=ios и initcopletesuccess).

Ниже показано, какие записи в BOOTLOG.TXT могут содержать информацию о процессе загрузки системы.

Error	При запуске зарегистрированы ошибки
Fail	При запуске зарегистрирован сбой
Dynamic load success	Список динамически загружаемых VxD
INITCOMPLETESUCCESS	Загруженные VxD
LoadStart, LoadSuccess, Loading Device, Loadind VxD	Отображение этапов загрузки
LoadFailed	Загрузка компонента не удалась
SYSCRITINIT, SYSCRITINITSUCCESS	Операции по инициализации системы
DEVICEINIT, DEVICEINITSUCCESS	Операции по инициализации устройств
Dynamic load device, Dynamic init device	Динамическая загрузка и инициализация устройств
Initing, Init Success, Initcomplete, Init, Init done	Инициализирующие операции
Status	Индикатор текущего состояния

Windows XP.

Этапы загрузки Windows XP в основе своей не отличаются от предыдущих версий. Всю загрузку можно представить укрупнено следующими фазами:

- самотестирование при включении (Power-On Self-Test, POST);
- инициализация при запуске (Initial startup process);
- работа загрузчика (Boot loader process) – выбор ОС, если их несколько и распознавание аппаратных средств ;
- загрузка и инициализация ядра;
- регистрация пользователя.

Для успешной загрузки необходима корректная инициализация аппаратных средств и наличие определенных системных файлов.

Самотестирование при включении.

Аналогично Windows 98 при включении или перезагрузке компьютер проходит стадию самотестирования аппаратных средств (POST). В это время компьютер находится под управлением BIOS. При возникновении проблем с аппаратными средствами или с настройками на стадии POST, компьютер сигнализирует об этом серией звуковых сигналов.

Файлы для запуска системы.

После завершения процедуры POST будут нужны следующие системные файлы:

- NTLDR – загрузчик ОС (корневой каталог загрузочного диска);
- Boot.ini – файл, задающий пути к каталогам, в которых установлены копии ОС (корневой каталог загрузочного диска);
- Bootsect.dos – для систем с двойной загрузкой, в которой в качестве альтернативной ОС используется DOS, Windows 3.1x или Windows 9x (корневой каталог загрузочного диска);
- Ntldetect.com – распознаватель аппаратной конфигурации, передает NTLDR информацию об обнаруженных и распознанных средствах (корневой каталог загрузочного диска);
- Ntbootdd.sys – нужен, если на компьютере SCSI стандарт, который, например, позволяет SCSI-контроллерам одновременно управлять 15 дисками (корневой каталог загрузочного диска);
- Ntoskrnl.exe – ядро операционной системы (%SystemRoot%\system32, где %SystemRoot% - путь до каталога с установленной Windows, например, C:\WINNT);
- Hal.dll – уровень аппаратных абстракций, изолирует низкоуровневые подробности функционирования аппаратных устройств от остальной ОС и

предоставляет API для обращения к одноптипным устройствам ((%SystemRoot%\system32);

- Раздел реестра SYSTEM – ключ реестра HKEY_LOCAL_MACHINE\SYSTEM ((%SystemRoot%\system32\config);
- Драйверы устройств – файлы драйверов устройств, установленных в системе ((%SystemRoot%\system32\drivers).

Инициализация при запуске.

После завершения POST начинается процесс инициализации при запуске: на компьютерах x86 BIOS ищет и загружает в память загрузочный сектор, инструкции которого затем загружаются в файл NTLDR.

Для запуска большое значение имеет первый сектор жесткого диска, который содержит главную загрузочную запись (Master Boot Record, MBR) и таблицу разделов (partition table).

Системная BIOS считывает главную загрузочную запись, загружает ее в память и передает ей управление. Код, содержащийся в главной загрузочной записи, сканирует таблицу разделов в поисках системного раздела. В случае успешного поиска в память загружается нулевой сектор системного раздела и выполняется его код. В нулевом секторе находится загрузочный код ОС, который и осуществляет запуск системы.

Загрузочный сектор раздела Windows XP отвечает за выполнение следующих действий:

- распознавание файловой системы и ее использование для поиска загрузчика ОС – NTLDR в корневом каталоге системного раздела (в FAT – загрузочная информация имеет один сектор физической разметки, в FAT32 – два сектора, в NTFS – до 16-ти секторов);
- поиск NTLDR и загрузка его в память;
- старт исполнения кода самозагрузки;

Работа загрузчика (Boot loader).

Загрузчик позволяет выбрать ОС и загрузить файлы из соответствующего раздела. На данном этапе устанавливается 32-х разрядная модель памяти с прямой адресацией (flat addressing), собираются данные об аппаратной конфигурации и создается описание в памяти этих данных, а также передается указатель на это описание в блок загрузчика. Далее NTLDR загружает образ ядра, HAL и драйверы для устройств и FS тома, с которого производится загрузка системы.

Загрузчик NTLDR свою работу начинает с очистки экрана, а затем выполняет следующие действия:

- переключает процессор в режим использования 32-х разрядной модели памяти с прямой адресацией;
- запускает соответствующую минифайловую систему (этот код встроен в NTFS и необходим для доступа к файлам FAT, NTFS);
- читает файл Boot.ini и отображает на экране соответствующее меню для выбора загружаемой ОС;
- если выбрана одна из версий XP, то выполняется Ntdetect.com;
- загружает и запускает ядро ОС Ntoskrnl.exe и передает ему информацию, собранную программой Ntdetect.com.

Командный файл Ntdetect.com обнаруживает и распознает следующие аппаратные компоненты:- шины и адаптеры; - видеоадаптеры ; - клавиатуры; - коммуникационные порты; - флоппи-диски; - устройства ввода и параллельные порты.

Загрузка ядра.

Получив информацию об аппаратных средствах компьютера, NTLDR загружает в память уровень аппаратных абстракций (Hal.dll) и ядро ОС Ntoskrnl.exe. Загрузив и запустив ядро, NTLDR передает ему информацию, собранную программой Ntdetect.com. Далее загрузчик просматривает реестр и загружает раздел HKEY_LOCAL_MACHINE\SYSTEM из %SystemRoot%\system32\config\system.

На данном этапе загрузчик активизирует API для работы с реестром и создает набор управляющих параметров (control set), который будет использоваться для инициализации компьютера. Эти операции являются подготовительными для загрузки драйверов. Значение, заданное в ключе реестра HKEY_LOCAL_MACHINE\SYSTEM\Select, определяет, какой набор управляющих параметров из перечисленных в ключе HKEY_LOCAL_MACHINE\SYSTEM должен использоваться при загрузке. По умолчанию загрузчик использует набор управляющих параметров, определяемых значением Default.

После этого загрузчик устанавливает значение Current ключа Select на номер набора управляющих параметров, которые он будет использовать.

Далее загрузчик сканирует все сервисы, определенные ключом реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services и ищет драйверы устройств, для которых значение Start равно 0x0 (такое значение указывает на то, что драйверы должны быть загружены, но не инициализированы). Как правило, драйверы с такими значениями представляют собой низкоуровневые драйверы устройств (например, драйверы дисков). Значение Group для каждого драйвера устройства определяет порядок, в котором загрузчик должен его загружать. Ключ

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ServiceGroupOrder реестра определяет порядок загрузки.

К моменту завершения этой фазы все базовые драйверы загружены и активны, за исключением, когда один из критически важных драйверов не инициализировался, вследствие чего началась перезагрузка системы.

Инициализация ядра.

В системе Windows XP об инициализации ядра сигнализирует появление графической заставки. Ядро создает ключ HKEY_LOCAL_MACHINE\HARDWARE, используя информацию, полученную от загрузчика. Данный ключ содержит данные об аппаратных средствах, распознавание которых осуществляется каждый раз при запуске системы.

В состав этих данных входит информация об аппаратных компонентах на системной плате и о прерываниях, используемых конкретными аппаратными устройствами.

Ядро создает набор опций управления Clone, копируя в него опции управления из набора CurrentControlSet. Набор опций управления Clone никогда не модифицируется, т.к. он должен представлять собой полностью идентичную копию данных, которые использовались для конфигурирования компьютера и не должны отражать изменений, внесенных в ходе процесса запуска.

На стадии инициализации ядро выполняет следующие операции:

- инициализирует низкоуровневые драйверы устройств, загруженные на предыдущей стадии;
- загружает и инициализирует остальные драйверы устройств, для которых значение ключа Start равно 0x1, причем драйверы загружаются не за счет вызовов BIOS или программ ПЗУ, а с помощью драйверов, загруженных на стадии загрузки ядра;
- запускает программы, например, Chkdsk, которые должны отработать прежде, чем будут загружены какие-нибудь сервисы;
- загружает и инициализирует сервисы (диспетчер сеансов Smss.exe запускает высокоуровневые подсистемы и сервисы Windows XP, а информация, предназначенная для диспетчера сеансов, находится в разделе реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SessionManager);
- создает файл подкачки pagefile.sys;
- запускает подсистемы, необходимые для работы Windows XP (поскольку архитектура подсистем базируется на сообщениях, необходимо запустить подсистему Windows (Win32). Эта подсистема управляет всем вводом/выводом и доступом к дисплею; ее процесс называется Csrss. Подсистема Win32

запускает процесс WinLogon, который в свою очередь запускает несколько других важных подсистем).

Ключ реестра Services.

Помимо наборов управляющих параметров, для загрузки и инициализации сервисов и драйверов Windows XP использует информацию, хранящуюся в реестре под ключами *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\drivename*, где *drivename* - имя конкретного сервиса или драйвера. В ходе фазы инициализации ядра, загрузчик NTLDR и ядро Ntoskrnl.exe просматривают содержимое этих ключей реестра, чтобы определить порядок и способ загрузки драйверов и сервисов.

Загрузка и инициализация драйверов устройств.

Теперь ядро инициализирует низкоуровневые драйверы устройств, которые были загружены на стадии загрузки ядра. В случае ошибки при инициализации одного из драйверов система предпринимает корректирующее действие, основываясь на данных, определенных параметром реестра *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DriverName\ErrorControl*.

Далее Ntoskrnl.exe сканирует реестр, на этот раз - в поисках драйверов устройств, для которых значение раздела *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DriverName\Start* установлено в 0x01. Это всегда так: значение *Group* для каждого драйвера устройства определяет порядок, в котором производится их загрузка. Раздел реестра *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ServiceGroupOrder* определяет порядок загрузки.

В отличие от фазы загрузки ядра, драйверы устройств, для которых значение *Start* установлено на 0x01, загружаются не за счет вызовов BIOS или программ ПЗУ, а с помощью драйверов устройств, загруженных на стадии загрузки ядра и только что инициализированных на этой стадии. Обработка ошибок в процессе инициализации этой группы драйверов устройств также основывается на значении параметра *ErrorControl* для соответствующих драйверов устройств.

Загрузка сервисов.

Диспетчер сеансов (*Smss.exe*) запускает высокоуровневые подсистемы и сервисы (службы) операционной системы. Информация, предназначенная для диспетчера сеансов, находится под ключом реестра *HKEY_LOCAL_MACHINE\SYSTEM\CurrentSet\Control\SessionManager*. Диспетчер сеансов исполняет инструкции, которые содержатся в следующих элементах реестра:

- Параметр *BootExecute*;
- ключ *Memory Management*;
- ключ *Dos Devices*;
- ключ *SubSystems*.

Параметр реестра *BootExecute*.

Параметр реестра *BootExecute* содержит одну или несколько команд, которые диспетчер сеансов (*smss.exe*) выполняет перед загрузкой сервисов. Значением по умолчанию для этого элемента является *Autochk.exe*, т. е. версия *Chkdsk.exe* для Windows XP. Диспетчер сеансов может запустить несколько программ.

После того как диспетчер сеансов выполнит все указанные команды, ядро осуществит загрузку остальных ключей реестра из *%SystemRoot%\System32\Config*.

Ключ *Memory Management*.

В следующий момент диспетчер сеансов иницирует информацию о файле подкачки, необходимую диспетчеру виртуальной памяти. Конфигурационная информация располагается в следующих значимых элементах:

PagedPoolSize:REG_DWORD 0

NonPagedPoolSize:REG_DWORD 0

PagingFiles: REG_MULTI_SZ: c:\pagefile.sys 32

В версиях, предшествующих Windows XP, по мере загрузки драйверов устройств, системных сервисов, а так же пользовательских оболочек, с жесткого диска подгружаются необходимые страницы памяти. В Windows XP происходит упреждающая выборка этих страниц, до загрузки драйверов, которые, будут их использовать.

Механизм упреждающей выборки в Windows XP выполняет следующие функции:

- Осуществляется динамическая трассировка процесса загрузки, для построения списка упреждающей выборки. Во время простоя, или же при использовании программы трассировки загрузки - *Bootvis.exe*, файлы загрузки размещаются в смежных областях диска. Для успешной работы механизма упреждающей выборки, ему необходимо как минимум две загрузки после инсталляции системы, чтобы определить, какие файлы следует разместить на диске. Данный механизм постоянно отслеживает восемь предыдущих загрузок.
- Активизирует выполнение асинхронных операций быстрого ввода/вывода для эффективной загрузки необходимых файлов в ходе выполнения загрузки системы.

Ключ *DOS Devices*.

Затем диспетчер сеансов создает символические ссылки. Эти ссылки направляют определенные классы команд на корректные компоненты файловой системы. Конфигурационная информация для перечисленных ниже устройств DOS содержится в следующих значимых элементах реестра:

PRN:REG_SZ: \DosDevices\ LPT1

AUX:REG_SZ: \DosDevices\ COM1

NUL:REG_SZ: \Device\ Null

UNC:REG_SZ: \Device\ Mup

PIPE: \REG_SZ: \Device\ NamedPipe

MAILSLOT: \REG_SZ: \Device\ MailSlot

Ключ *SubSystem*.

Поскольку архитектура подсистем базируется на сообщениях, необходимо запустить подсистему Windows (Win32), которая управляет всем вводом/выводом и доступом к видеодисплею. Процесс этой подсистемы: носит имя CSRSS. Подсистема Win32 запускает процесс *winlogon*, который, в свою очередь, запускает несколько других важных подсистем.

Конфигурационная информация для необходимых подсистем определяется значением элемента *Required* в ключе реестр, *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\ Control\ SessionManager\ SubSystem*.

Регистрация в системе.

Подсистема Win32 автоматически запускает процесс *Winlogon.exe*, который, в свою очередь, запускает процесс *распорядителя локальной безопасности (Local Security Authority, LSA) - Lsass.exe*. По завершении инициализации ядра необходимо произвести регистрацию пользователя в системе. Процедура регистрации может быть произведена автоматически на основании информации, хранящейся в реестре, или вручную.

На данном этапе Диспетчер управления сервисами (*Service Control Manager*) выполняет загрузку автоматически стартующих сервисов, для которых значение параметра *Start*, расположенного в ключе реестра *HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\ Services\ DriverName*, установлено равным 0x2. На этом этапе сервисы загружаются с учетом установленных для них зависимостей, поскольку их загрузка осуществляется параллельно. Зависимости описываются значимыми элементами *DependOnGroup* и *DependOnService*, расположенными под ключом реестра *HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet*

Services\DriverName.

После регистрации происходят следующие события:

- *Обновляются управляющие наборы (control sets are updated).* Управляющий набор, указанный в элементе *LastKnownGood*, обновляется за счет содержимого элемента *Clone*. *Clone* - копия элемента *CurrentControlSet*, создаваемая каждый раз при запуске компьютера.
- *Установки параметров групповой политики вступают в силу.* Установки групповой политики (*Group Policy*), применимые как к работе конкретного пользователя, так и к компьютеру вступают в силу.
- *Выполняются программы из группы Автозагрузка (Startup).* Windows XP запускает сценарии регистрации, программы запуска, а так же сервисы, которые указаны в реестре и в следующих каталогах:
- *HKEY_LOCAL_MACHINE\Software\Microsoft\ Windows\CurrentVersion \RunOnce;*
- *HKEY_LOCAL_MACHINE\Software\Microsoft\Windows \CurrentVersion\ Policies \Explorer\Run;*
- *HKEY_LOCAL_MACHINE\Software\Microsoft\ Windows \Current Version \Run;*
- *HKEY_ CURRENT_USER \Software \Microsoft\ Windows \Current Version \Run;*
- *%Systemdrive%\ Documents and Settings\ All Users\ Start Menu\Programs \ Startup;*
- *%Systemdrive%\Documents and Settings\%username%\StartMenu\Programs\Startup.*

Параметр Start.

В каждом из ключей, содержащийся под ключом реестра *HKEY_LOCAL_MACHINE\SYSTEM\ <control set>\ Services\ Driver Name* присутствует параметр *Start*, определяющий порядок запуска драйвера или сервиса. Он может иметь следующие значения:

- *0x0 (Boot).* Загрузка драйвера или сервиса осуществляется загрузчиком операционной системы перед инициализацией ядра В качестве примера драйверов с таким режимом загрузки можно привести драйверы дисков.
- *0x1 (System).* Загрузка осуществляется подсистемой ввода/вывода во время инициализации ядра. В качестве примера - драйверы мыши.
- *0x2 (Auto load).* Загружается Диспетчером управления сервисами (*Service Control Manager*). Таким образом, загружаются сервисы, которые должны стартовать автоматически при любых обстоятельствах запуска системы, вне зависимости от типа

сервиса. В качестве примера - драйверы устройств, работающих через параллельный порт. Одним из сервисов, использующих это значение, является сервис *Alert*.

- **0x3 (Load on Demand, Manual).** Загружается Диспетчером управления сервисами (*Service Control Manager*) только в случае получения явной инструкции на загрузку. Сервисы этого типа доступны всегда, но загружаются, только когда пользователь запускает их (например, используя опцию Службы (*Services*) оснастки Управление компьютером (*Computer Management*)).
- **0x4 (Disabled).** Не загружается. Windows XP устанавливает в этот режим драйверы устройств в случае невозможности их загрузки диспетчером сервисов (например, если не установлены соответствующие аппаратные средства). Единственным исключением являются драйверы файловых систем, которые загружаются, даже если для них установлено данное значение в параметре *Start*.

Параметр *ErrorControl*.

Ниже перечислены все возможные значения, которые способен принимать значимый элемент *ErrorControl*, находящийся под ключом реестра *HKEY_LOCAL_MACHINE\SYSTEM\<control/ set> \ Services\ DriverName*.

- ***Ignore* (0x0).** Если при загрузке или инициализации драйвера устройства происходит ошибка, процедура запуска продолжается без вывода сообщения об ошибке.
- ***Normal* (0x1).** Если при загрузке или инициализации драйвера устройства происходит ошибка, процедура запуска продолжается после вывода сообщения об ошибке. Значимые элементы *ErrorControl* для большинства драйверов устройств устанавливаются равными этому значению.
- ***Severe* (0x2).** Если ядро обнаруживает ошибку загрузки или инициализации этого драйвера или сервиса, происходит переключение на управляющий набор *LastKnownGood*. После этого процесс запуска стартует с начала. Если набор *LastKnownGood* уже используется, процедура продолжается, а ошибка игнорируется.
- ***Critical* (0x3).** Используется та же процедура, что и при значении *Severe*, кроме ситуаций, если переключение на набор: управляющих опций уже произошло, но ошибка не ликвидирована, процесс загрузки останавливается и выводится сообщение о сбое.

Особенности загрузки Windows XP.

Как уже было сказано, одним из важных направлений при разработке Windows XP было уменьшение времени загрузки. Можно утверждать, что описанные выше этапы не потеряли своей актуальности.

Наиболее интересным и эффективным для достижения рассматриваемой цели оказался механизм упреждающей выборки необходимой информации с жесткого диска. Суть этого механизма состоит в следующем: при каждой загрузке операционной системы сохраняется информация о всех операциях ввода/вывода логических дисков, а при последующих загрузках эта информация используется для того, чтобы в моменты времени, связанные с задержками инициализации аппаратных средств компьютера, про извести выборку необходимой для дальнейшей загрузки информации в системный кэш, к которому обеспечивается быстрый асинхронный доступ, что уменьшает время поиска требуемых данных на логических дисках. Таким образом, большинство операций обращения и выборки информации с жесткого диска проходит во время задержек инициализации устройств, что значительно ускоряет процесс загрузки. Дополнительно следует сказать, что выше сказано не только об упреждающей выборке, но и о параллельных процессу инициализации операциях ввода/вывода информации с жесткого диска.

Однако параллельно идут не только операции инициализации и обращения к жесткому диску, но и инициализация всех устройств компьютера проводится параллельно друг другу. Кроме этого корпорация Microsoft провела ряд работ с поставщиками аппаратного обеспечения с целью ускорения процесса инициализации оборудования.

Одним из таких мероприятий является спецификация Simple Boot Flag (SBF), которая ускоряет процедуру загрузки за счет того, что в специальном регистре хранится информация в частности о поддержке ОС спецификации Plug-and-Play, об удачности последней загрузки и прочее, что позволяет упростить процедуры проверки и инициализации оборудования.

Далее следует отметить улучшенный загрузчик NTLDR. Эти улучшения таковы, что каждый системный файл считывается за одну операцию обращения к жесткому диску. По оценкам разработчиков скорость работы загрузчика Win XP уменьшилась в 4-5 раз по сравнению с Windows 2000.

В процессе загрузки производится только загрузка и инициализация необходимых драйверов, а все остальные драйвера подгружаются позже по мере необходимости.

В заключении можно сказать, что разработчиками ОС Windows XP действительно удалось найти ряд интересных решений для оптимизации процесса загрузки при сохранении принципов загрузки близкими к принципам или архитектуре семейства NT/2000.

Приложение 2.**Инструкция по переносу виртуальной машины.**

В случае если к ЛР №2 на вашей машине нет установленного в VirtualBox Debian, его можно скопировать. Для этого необходимо выполнить следующие шаги:

1. В «Проводнике» открыть диск C: и найти в нём папку «Debian». Скопировать эту папку в «C:\Пользователи\userXX\VirtualBox VMs», где XX – номер пользователя, под которым выполнен вход.
2. Открыть VirtualBox, выбрать «Машина – Добавить».
3. Открыть скопированную папку и выбрать в ней виртуальную машину.

В случае неполадок с сетью на виртуальной машине:

1. Выключить виртуальную машину.
2. Нажать «Машина – Копировать».
3. Выбрать имя копии виртуальной машины и установить флажок «Сгенерировать новые MAC адреса для всех сетевых адаптеров».
4. После успешного копирования – удалить старую виртуальную машину (со всеми файлами).

Порядок выполнения части 2.

1. Ознакомиться с теоретическими сведениями по процессу загрузки ОС.
2. Исследовать процесс загрузки с различными вариантами основных параметров системных файлов.
3. Создать протокол загрузки (обновить, просмотреть).
4. Дать анализ содержимого системных файлов участвующих в процессе загрузки на конкретном компьютере.
5. Пройти тестирование по теоретическому материалу.

Отчет должен включать:

- название работы и ее цель;
- описание основных шагов процесса загрузки с иллюстрациями из системных файлов, участвующих в данном процессе.