

Администрирование UNIX-подобных ОС

- Установка ОС
- Установка дополнительного ПО
- Загрузка ОС
- Управление пользователями
- Управление системой, системное администрирование
- Сетевое администрирование

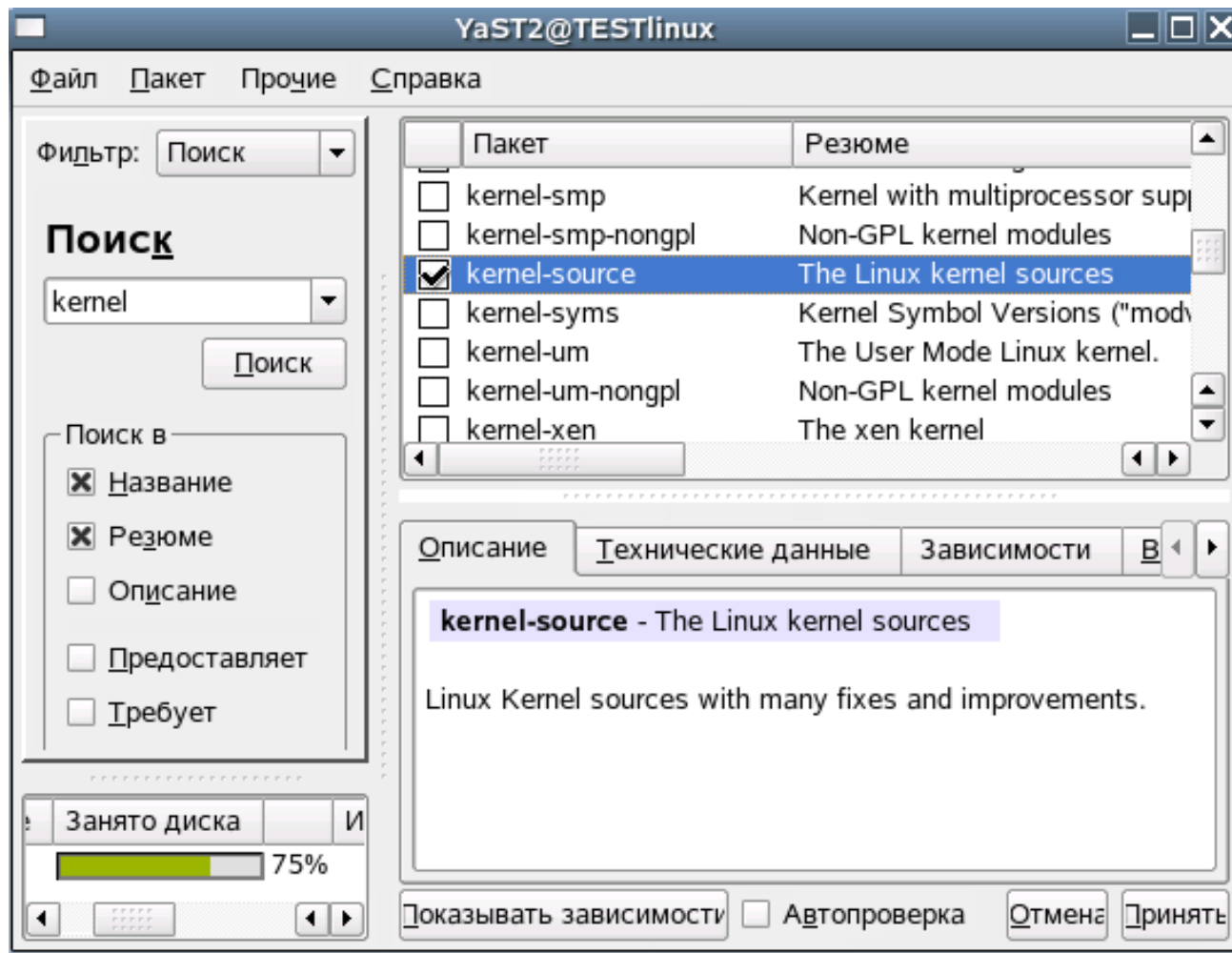
Установка ОС и дополнительного ПО (пример Linux)

- разница в подходах к защищенности для закрытых продуктов и программ с открытыми исходными текстами
- выбор поставщика дистрибутива
 - Red Hat, Debian, Mandrake
 - Slackware
 - Openwall Linux, Trustix Security Linux, ALT Linux, ASPLinux
- механизм тестирования и устранения неисправностей – BTS (Bug Tracking System)
- Сертифицированные дистрибутивы
 - см. <http://fstec.ru> , «Государственный реестр сертифицированных средств защиты информации»
 - поддерживаемые компаниями РФ: Альт Линукс, МСВСфера
 - прочие: Red Hat, SUSE, Oracle Enterprise, CentOS

Установка ОС и дополнительного ПО (пример Linux)

- В современных ОС осуществляется под управлением программ-менеджеров пакетов ПО (PM – package manager), например, RPM – Redhat Package Manager.
- В задачи менеджера пакетов входит
 - хранение в компактном виде ПО (обычно используются архиваторы tar, cpio и компрессоры gzip, bzip2)
 - обеспечение гарантий подлинности ПО, например, с помощью цифровой подписи
 - определение зависимостей устанавливаемого ПО от другого ПО, библиотек, ядра и т.п.
 - ведение базы установленного ПО, например, для последующего удаления, модернизации, выдачи информации об установленном ПО и т.п.
 - Хотя менеджер пакетов предназначен в основном для установки уже собранного (готового к использованию) ПО, исходные тексты программ также могут быть оформлены в виде пакетов и установлены с помощью менеджера.

GUI менеджера пакетов в SuSe



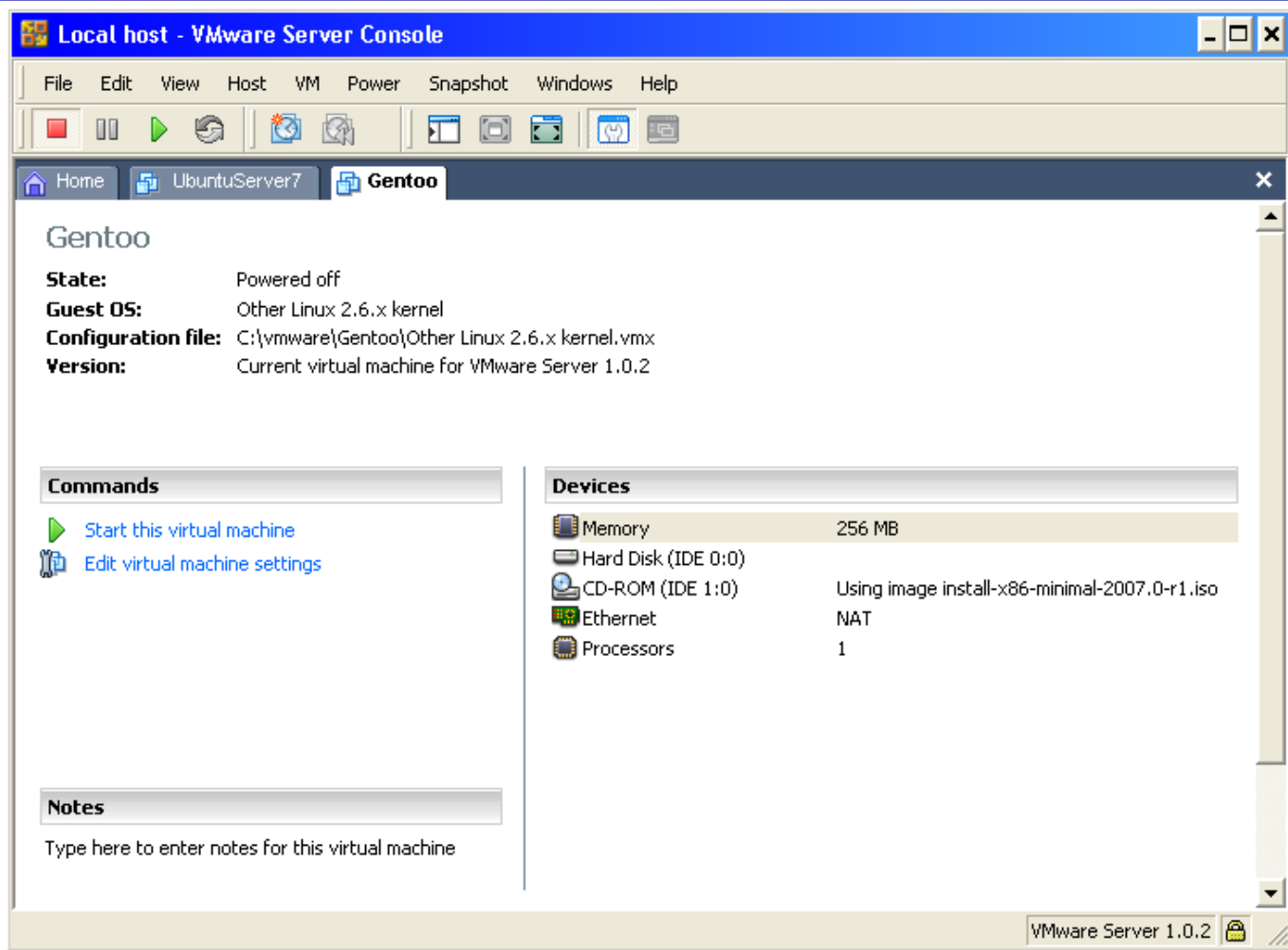
Загрузка установочного CD



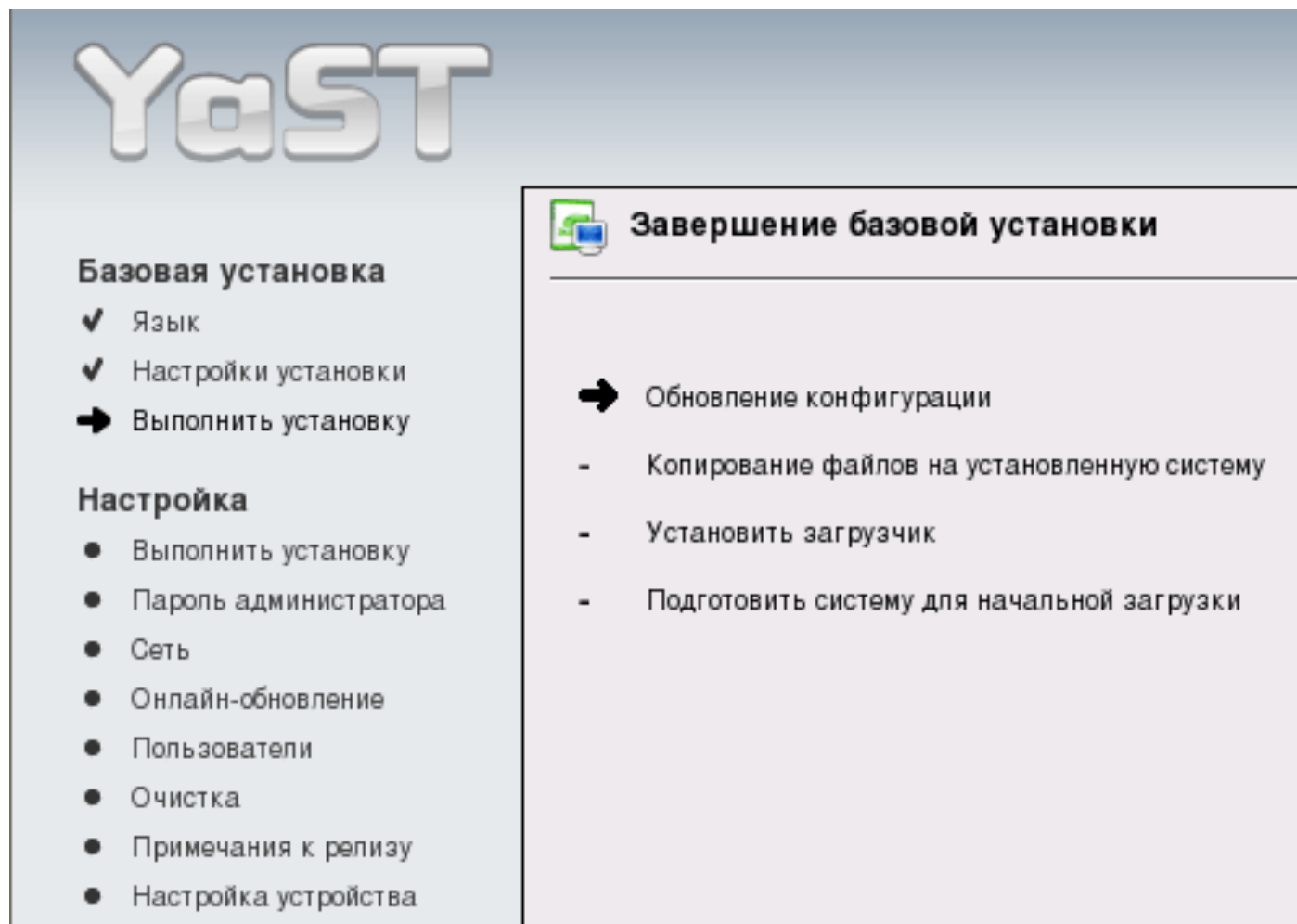
Start or install Ubuntu
Start Ubuntu in safe graphics mode
Install with driver update CD
Install in text mode
Install a server
Text mode install for manufacturers
Install a command-line system
Check CD for defects

F1 Help F2 Language F3 Keymap F4 VGA F5 Accessibility F6 Other Options

Установка GNU/Linux (Gentoo) в VMware Server



Установка SuSe на этапе после выбора пакетов дополн. ПО



The screenshot shows the YaST (Yast) installation wizard interface. At the top left, the YaST logo is displayed in a large, stylized font. Below the logo, the interface is divided into two main sections. The left section contains a list of installation steps, with 'Выполнить установку' (Perform installation) highlighted by a right-pointing arrow. The right section, titled 'Завершение базовой установки' (Completion of basic installation), contains a list of tasks to be performed, with 'Обновление конфигурации' (Update configuration) highlighted by a right-pointing arrow.

YaST

Базовая установка

- ✓ Язык
- ✓ Настройки установки
- ➔ Выполнить установку

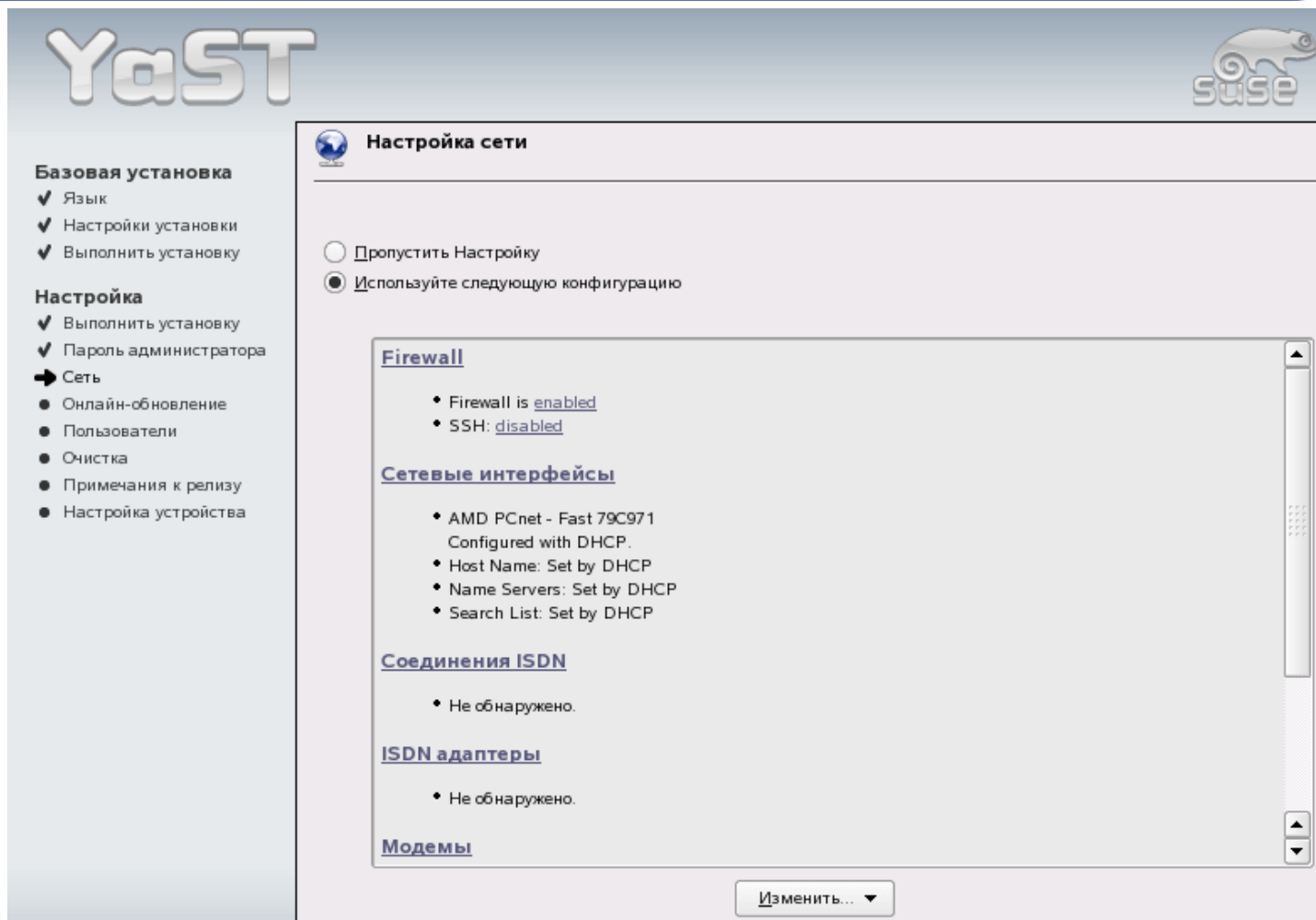
Настройка

- Выполнить установку
- Пароль администратора
- Сеть
- Онлайн-обновление
- Пользователи
- Очистка
- Примечания к релизу
- Настройка устройства

Завершение базовой установки

- ➔ Обновление конфигурации
- Копирование файлов на установленную систему
- Установить загрузчик
- Подготовить систему для начальной загрузки

Установка SuSe 9.3 на этапе задания параметров сети



Репозитории

yum repolist

```
Loaded plugins: fastestmirror, refresh-packagekit, replace, security
```

```
Loading mirror speeds from cached hostfile
```

- * base: centos-mirror.rbc.ru
- * extras: centos-mirror.rbc.ru
- * passenger: mirror.hmhc.harvard.edu
- * rpmforge: apt.sw.be
- * updates: centos-mirror.rbc.ru
- * webtatic: uk.repo.webtatic.com

repo id	repo name	status
base	CentOS-6 - Base	6,575
extras	CentOS-6 - Extras	43
google-chrome	google-chrome - 64-bit	2
passenger	Red Hat Enterprise 6 - Phusion Passenger	9
rpmforge	RHEL 6 - RPMforge.net - dag	4,718
updates	CentOS-6 - Updates	439
webtatic	Webtatic Repository EL6 - x86_64	185

Пример добавления репозитория

Установка открытого PGP-ключа репозитория:

```
# rpm --import https://fedoraproject.org/static/0608B895.txt
```

Копирование и установка пакета описания репозитория:

```
# wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
rpm -ivh epel-release-6-8.noarch.rpm
```

Изменение приоритета репозитория (priority=10)

```
# yum install yum-priorities
```

```
# cat epel.repo
```

```
[epel]
```

```
name=Extra Packages for Enterprise Linux 6 - $basearch
```

```
#baseurl=http://download.fedoraproject.org/pub/epel/6/$basearch
```

```
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-6&arch=$basearch
```

```
failovermethod=priority
```

```
enabled=1
```

```
priority=10
```

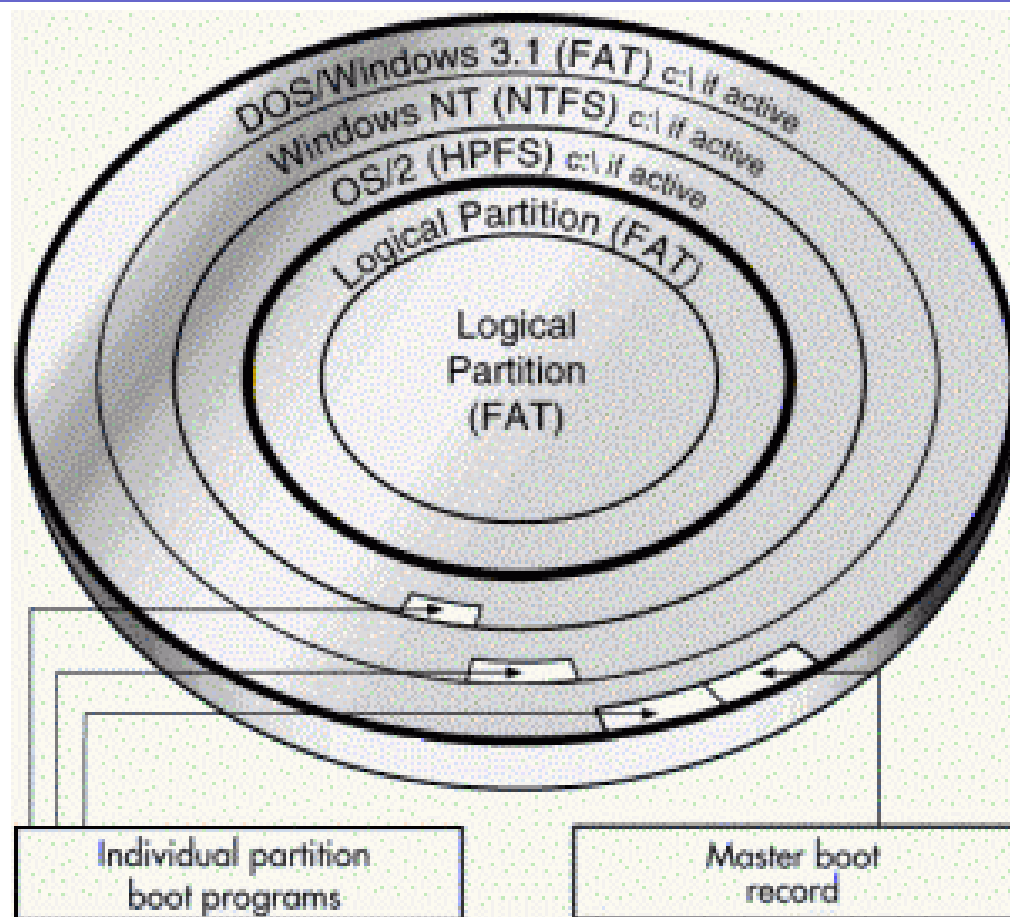
```
gpgcheck=1
```

```
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
```

Системы, собираемые из исходных текстов (Source Based distributions)

- Предыдущий дистрибутив (SuSe) – представитель наиболее распространенных систем, основанных на скомпилированных бинарных пакетах
- Ряд UNIX-подобных систем (FreeBSD, дистрибутив Gentoo-Linux) базируются на т.н. портах (FreeBSD) или портежах (portages, Gentoo)
- Часть дистрибутива (базовые компоненты, Distribution) устанавливаются из скомпилированных пакетов, большая часть пользовательских приложений собирается из исходного кода непосредственно на компьютере пользователя
- Нормальное использование и обновление FreeBSD, Gentoo требует доступа в Интернет

Загрузка ОС на ПК (IBM-совм.)



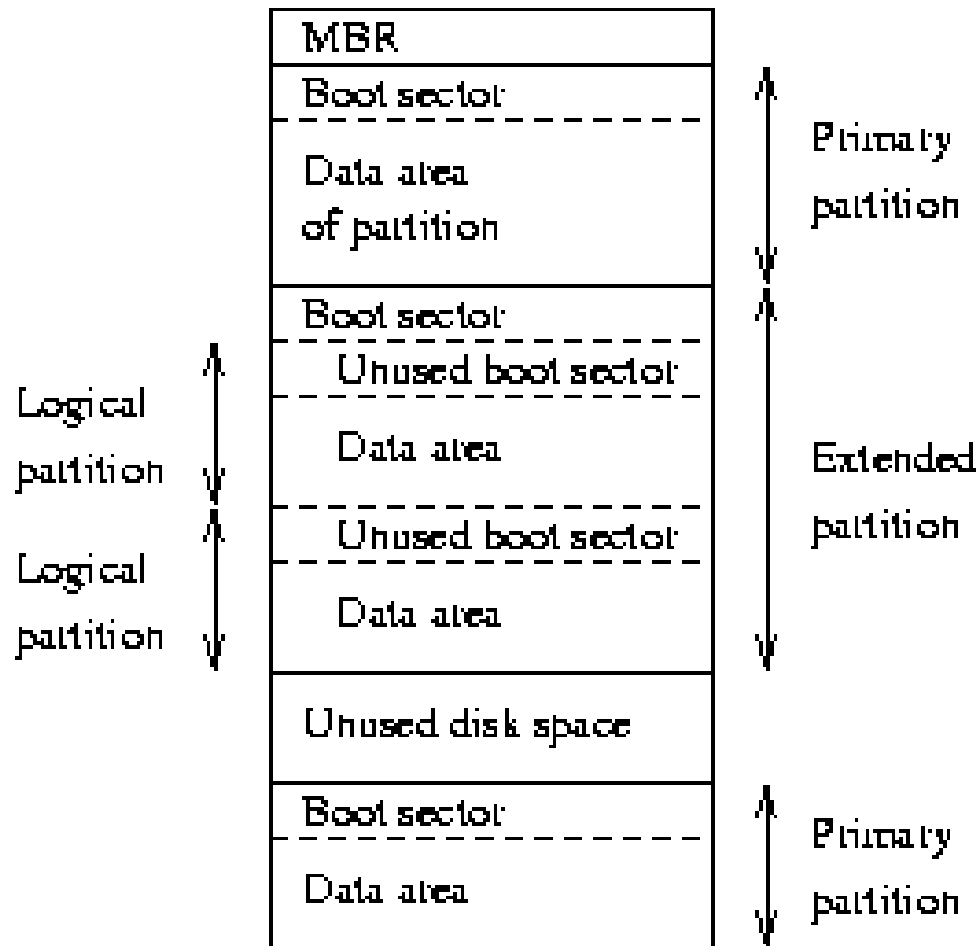
Boot-секторы разделов
LILO, GRUB и т.п.

Стандартный MBR, либо
LILO, GRUB и т.п.

В MBR могут быть размещены:

- т.н. стандартный код, который, определяя флаг активности раздела, передает управление на загрузчик этого раздела
- специальная загрузочная программа «boot manager», например, LILO (Linux LOader), GRUB и т.п.

Типы разделов, использование «расширенного» раздела

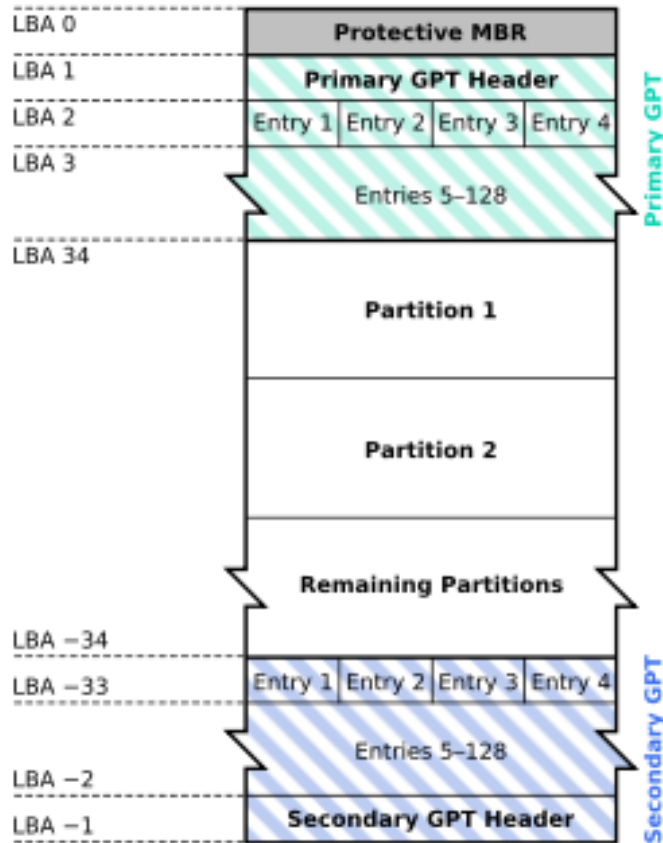


UEFI- Unified Extensible Firmware Interface

- Ограничения BIOS IBM
 - 16-bit режим (процессор 8088)
 - 1 MB адресуемого пространства (8088)
- UEFI
 - 32 или 64-битный режим
 - требует, чтобы разрядность ОС и микропрограммы соответствовали друг-другу (64-битный UEFI => 64-битная ОС)
 - поддерживает кроме стандартной схемы разделов, GUID Partition Table (GPT)
 - поддерживает сервисы: boot (текстовая и графическая консоль управления устройствами, шинами, загрузкой, содержит шелл) и runtime (часы, доступ к микропрограмме)
 - ОС, способные грузиться с использованием UEFI называют (U)EFI-совместимыми (W2K8, Linux/GRUB)

GPT

GUID Partition Table Scheme



- MBR => LBA0
- Partition Table => LBA1
- Каждый логический блок (LBA) имеет размер ровно 512 байт.
- В конце диска находится вторая таблица разделов

GPT формат описания раздела

Смещение	Размер	Содержание
0	16 bytes	Partition type GUID
16	16 bytes	Unique partition GUID
32	8 bytes	First LBA
40	8 bytes	Last LBA
48	8 bytes	Attribute flags (read-only и т.п.)
56	72 bytes	Partition name (36 символов в UTF-16)

Загрузка UNIX-подобной ОС

- загрузчик (bootstrap loader)
 - реализуется по-разному на разном оборудовании, например, на ПК выполняется загрузка 1 сектора диска.
- загрузка ядра
- инициализация оборудования, конфигурирование драйверов и загрузка модулей ядра (если поддерживаются)
- монтирование корневой ФС в режиме read-only
 - положение корневой ФС может быть определено ядром программой rdev или программами boot-manager
- процесс init (/sbin/init) с PID=1, выполнение стартовых скриптов
- запуск демонов, в т.ч. терминального getty
- Для выключения системы необходимо завершить все процессы системы, синхронизировать разделы диска, содержащие ФС и соответствующие буферы, демонтировать ФС, например, с помощью программы shutdown. Для перезагрузки применяют shutdown -r now.

Начало работы ядра

```
Loading linux.  
Console: colour EGA+ 80x25, 8 virtual consoles  
Serial driver version 3.94 with no serial options enabled  
tty00 at 0x03f8 (irq = 4) is a 16450  
tty01 at 0x02f8 (irq = 3) is a 16450  
lp_init: lp1 exists (0), using polling driver  
Memory: 7332k/8192k available (300k kernel code, 384k reserved, 176k  
data)  
Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M  
Loopback device init  
Warning WD8013 board not found at i/o = 280.  
Math coprocessor using irq13 error reporting.  
Partition check:  
  hda: hda1 hda2 hda3  
VFS: Mounted root (ext filesystem).  
Linux version 0.99.pl9-1 (root@haven) 05/01/93 14:12:20
```

Процесс загрузки с мини-корневой файловой системой initrd

- При сборке и установке ядра нужно учитывать особенности процесса загрузки ядра во многих современных дистрибутивах (многоэтапный процесс)
 - Слева – обычная загрузка, справа – загрузка с мини корневой файловой системой (ФС). Т.о., помимо сборки ядра, нужно еще и сделать образ мини-корневой ФС initrd (initial RAM disk).
- | | |
|--|--|
| •Запуск менеджера загрузки | •Запуск менеджера загрузки |
| •Загрузка и запуск ядра | •Загрузка и запуск ядра с минимум встроенных компонентов |
| •Монтирование корневой файловой системы (ФС) | •Загрузка образа содержимого мини-корневой ФС (initrd) в память |
| •Запуск процесса init из /sbin/init | •Монтирование RAM-диска с мини-корневой ФС |
| •Выполнение стартовых скриптов из /etc/rc.d/ в соответствии с целевым уровнем загрузки | •Запуск исполняемого файла /linuxrc, подгрузка необходимых модулей, возможно, интерактивно |
| | •Монтирование обычной корневой ФС |
| | •Запуск процесса init из /sbin/init и т.д. |

Уровни работы (исполнения) UNIX-подобной ОС (sysvinit)

- Большинство дистрибутивов Linux используют вариант init sysvinit, основанный на System V init. BSD варианты традиционно не поддерживают «уровни».
- Уровни (run levels) – состояния ОС (можно менять командой telinit):

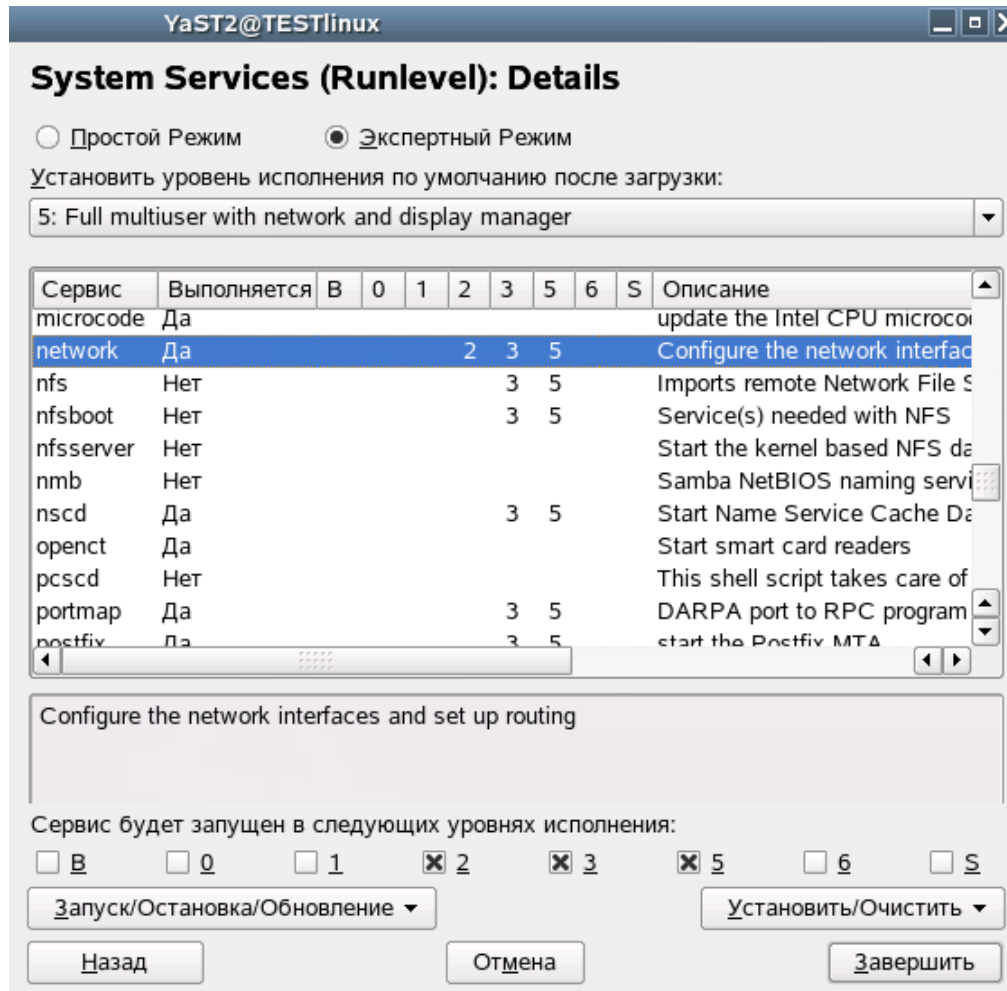
0	Halt the system. Остановка ОС
1	Single-user mode. Однопользовательский режим
2-5	Normal operation (user defined). Часто 3 – а/ц терминал, 5 – X Window
6	Reboot. Перезагрузка

Уровни конфигурируются в /etc/inittab (формат id:runlevels:action:process) :

```
id:3:initdefault:          1:2345:respawn:/sbin/mingetty tty1
l0:0:wait:/etc/rc.d/rc.halt 2:2345:respawn:/sbin/mingetty tty2
l1:1:wait:/etc/rc.d/rc.single 3:2345:respawn:/sbin/mingetty tty3
l2:2345:wait:/etc/rc.d/rc.multi
l6:6:wait:/etc/rc.d/rc.reboot x:5:respawn:/etc/X11/prefdm –nodaemon

m1:23:respawn:/usr/local/sbin/mgetty -n 5 tty00
```

GUI управления уровнями в SuSe

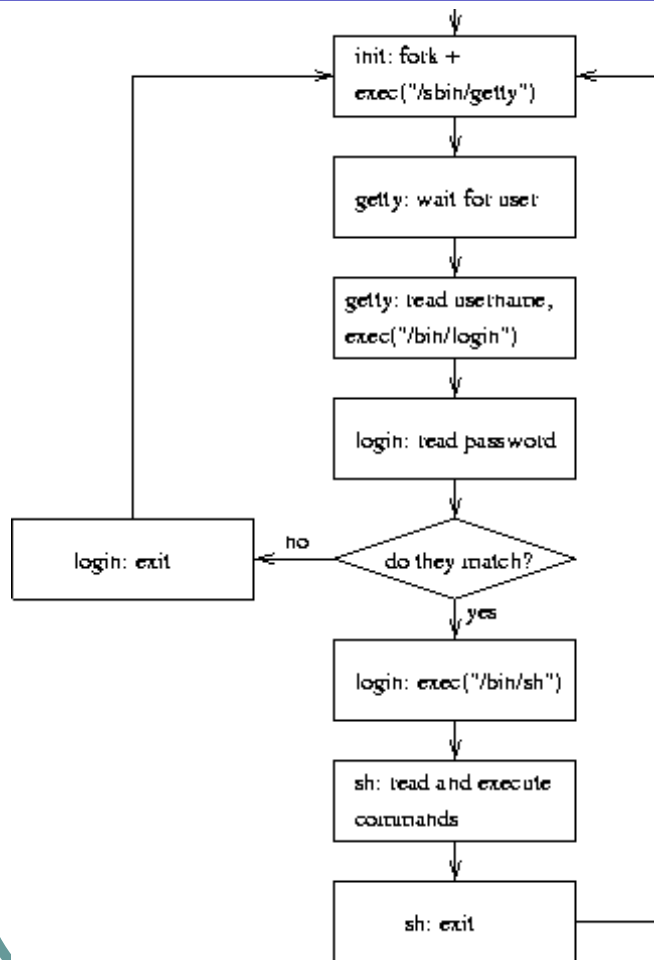


Результат – создание символической связи в каталоге, соответствующем уровню. Например, для запуска и остановки сети (скрипт network) на 3 уровне, нужно сделать следующие СВЯЗИ:

```
ln -s /etc/init.d/network \
/etc/init.d/rc3.d/S05network
```

```
ln -s /etc/init.d/network \
/etc/init.d/rc3.d/K17network
```

Регистрация пользователей в системе, «ВХОД В СИСТЕМУ» (login)



Программа login выполняет:

- аутентификацию пользователя
- устанавливает атрибуты на линию
- стартует шелл

Важные файлы для программы login, кроме баз passwd, shadow, group:

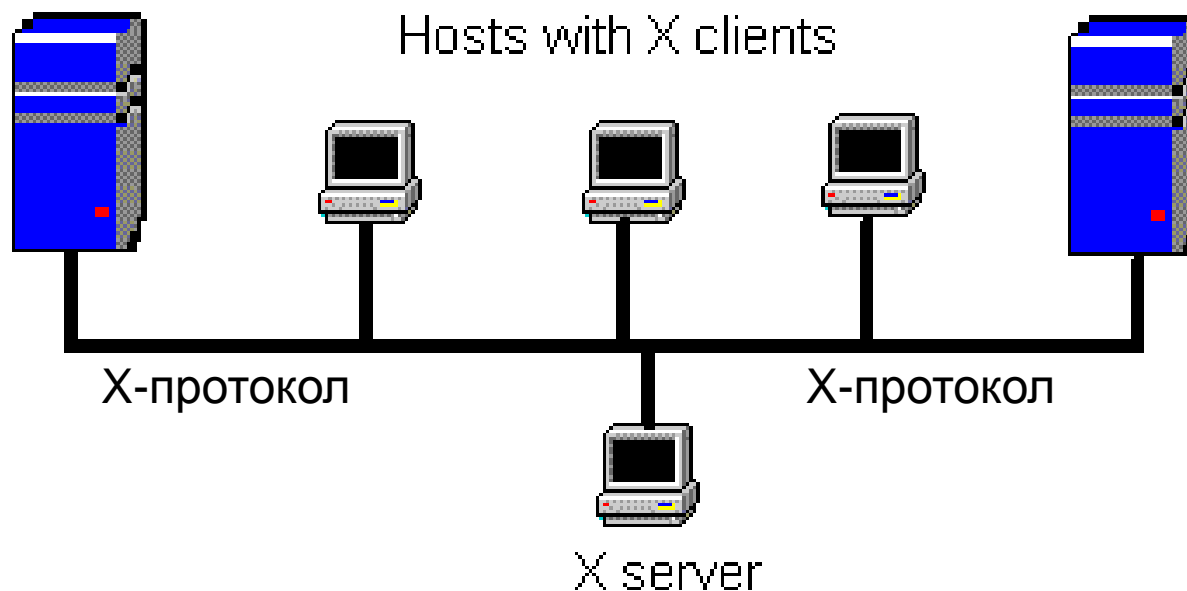
- /etc/motd
- /etc/nologin
- /var/run/utmp (w, who)
- /var/log/wtmp (last)

Система X использует xdm (менеджер дисплея) вместо login

Начало выполнения шелла

- При запуске любой шелл выполняет predetermined стартовые скрипты. Положение и названия этих скриптов зависит от шелла, но обычно (более точно см. `man sh`, `man bash`) для `sh` это:
 - `/etc/profile` (редактируется системным администратором, в некоторых дистрибутивах – набор скриптов в `/etc/profile.d/`)
 - `.profile` (редактируется пользователем)
- При `login`-запуске шелла `bash` запускаются после `/etc/profile`: `~/.bash_profile`, `~/.bash_login`, `~/.profile`. При завершении работы шелла, запускается `~/.bash_logout`
- При интерактивном `non-login` запуске `bash` (или удаленном запуске через сетевой демон) исполняет: `/etc/bash.bashrc`, `~/.bashrc`

X Window System, архитектура



X-сервер имеет унифицированный доступ к приложениям, расположенным на разных компьютерах (X-клиентах).

Т.о. окна на рабочем столе пользователя соответствуют приложениям, запущенным на разных компьютерах

Компоненты технологии

- X-сервер - процесс, выполняющийся на компьютере, к которому присоединен дисплей и обрабатывающий данные, приходящие от клавиатуры и манипулятора мышь.
- X-клиент взаимодействует с сервером по X-протоколу, посылая и получая команды с сервера.
- `xeyes -display c1r38?n??.cs.vsu.ru:0.0 &`
- Особенностью системы X-Window является то, что она не имеет встроенной возможности управлять самими окнами с помощью мышки или клавиатуры. Чтобы это можно было осуществить, нужен специальный клиент - менеджер окон (window manager, WM). Существует множество реализаций WM.

Менеджер окон и менеджер дисплея

- Обычно X-сервер и менеджер окон стартуют из-под менеджера дисплея (X Display Manager, DM), который представляет собой аналог `getty+login` текстового режима.
- Дистрибутивы ОС Linux содержат `xdm`, `kdm`, `gdm`, в зависимости от выбранной среды (Motif, KDE или Gnome)
- DM может управлять как локально запущенным X-сервером, так и удаленным X-терминалом по протоколу XDMCP.
- Выбор WM и другие настройки DM обычно содержатся в соответствующих конфигурационных файлах (например, `/etc/X11/gdm/gdm.conf` для `gdm`) и скриптах, например, `/etc/X11/xdm/Xsession`.
- `kdm`, `gdm` позволят интерактивно выбирать WM.

Управление пользователями в UNIX

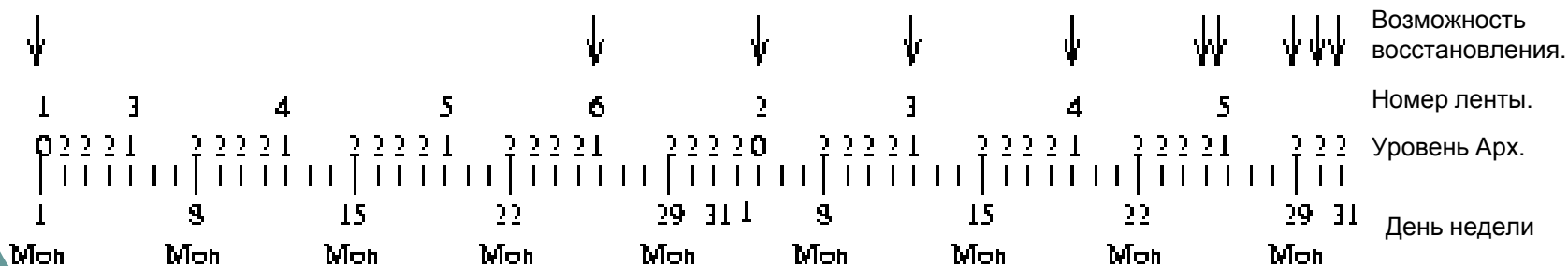
- редактирование файла `/etc/passwd` (и, возможно, `/etc/shadow`) с помощью редактора `vi`
- редактирование файла `/etc/group` с помощью `vi`, если нужна новая группа
- Создание домашнего каталога `mkdir`
- Копирование "скелета" `/etc/skel` в домашний каталог
- Установка атрибутов для домашнего каталога и его содержимого с помощью `chown` и `chmod` с ключом `-R`:
 - `cd /home/ivanov`
 - `chown -R ivanov.users .`
 - `chmod -R go=u,go-w .`
 - `chmod go= .`
- Установка пароля с помощью `passwd`
- Задание персональной информации и шелла: **`chfn`**, **`chsh`**
- Удаление – обратная задача. Найти файлы: `find / -user ivanov`
- Временное блокирование: `chsh -s /usr/local/lib/no-login/security ivanov`

Начало выполнения шелла

- При запуске любой шелл выполняет predetermined стартовые скрипты. Положение и названия этих скриптов зависит от шелла, но обычно для sh это:
 - /etc/profile (редактируется системным администратором)
 - /etc/profile.d/ (в некоторых дистрибутивах, редактируется системным администратором)
 - .profile (редактируется пользователем)
- Для шелла bash иногда используются
 - .bash_login, .bash_profile, .bash_logout

Архивирование: tar, cpio, dump

- Простейшее архивирование (утилиты BSD tar и AT&T cpio) :
 - `tar -cvzf /dev/ftape /home` `tar -czf 123.tgz ./` `ls | cpio -o >archive.cpio`
- Восстановление
 - `tar -xvzf /dev/ftape /`
- Многоуровневое архивирование
 - Два типа: полное и изменений (инкрементное)
 - Для 10 лент можно определить, например, 3-уровневое A:
 - 2 ленты 0 уровня (первая пятница месяца, полное архивирование)
 - 4 ленты 1 уровня (1 раз в неделю, пятница, архивирование изменений. В месяце может быть 5 пятниц.)
 - 4 ленты 2 уровня (понедельник, вторник, среда и четверг, архивирование изменений)



Конфигурирование сети

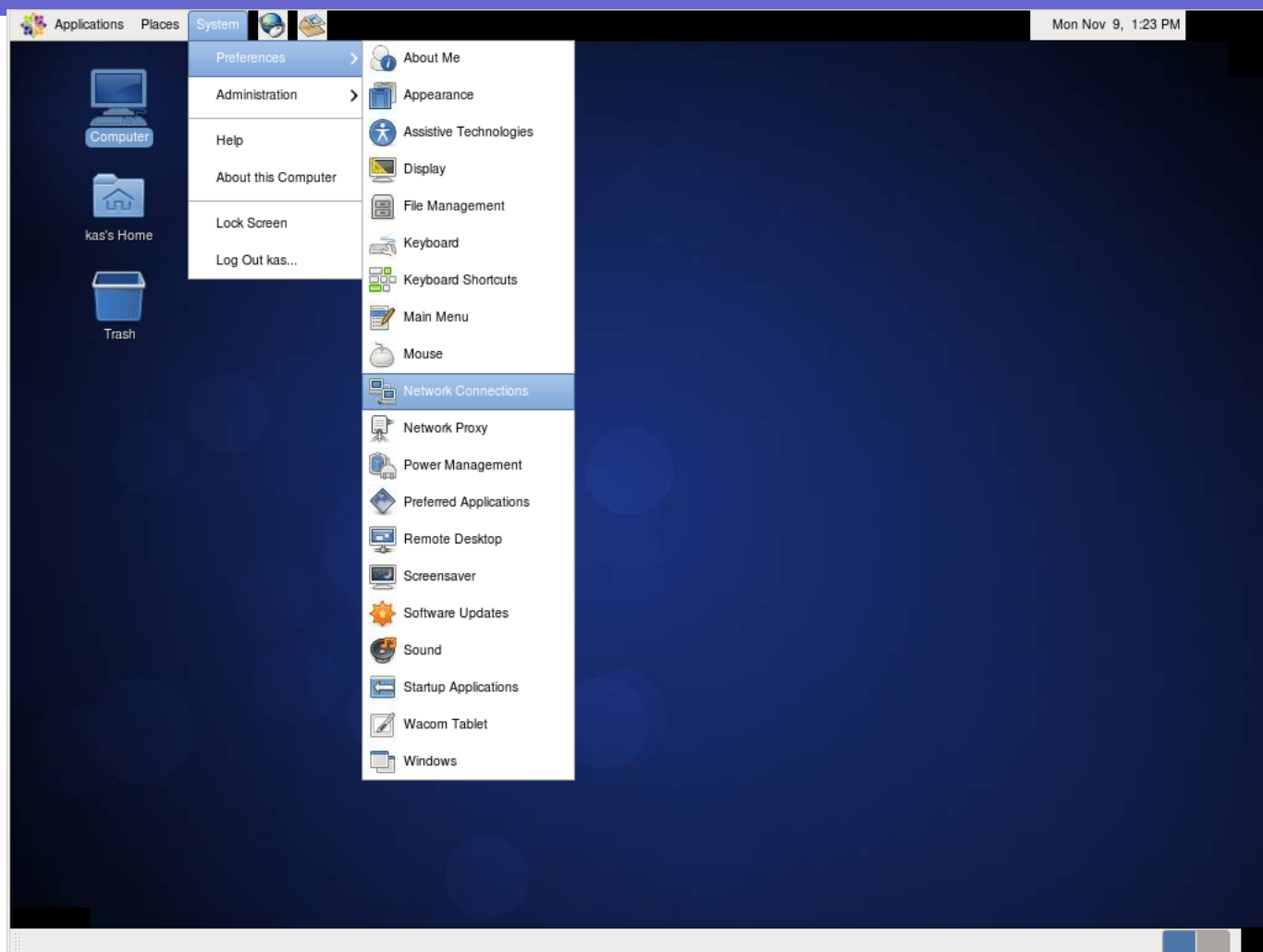
- с помощью утилит командной строки: `ifconfig` и более новой `ip` (`man ip`)
- «традиционное» конфигурирование: `/etc/init.d/network => /etc/sysconfig/network-scripts: ifcfg-eth?, ifcfg-System_eth?, ifup, ifdown`

```
$ cat /etc/sysconfig/network-scripts/ifcfg-System_eth0
```

```
TYPE=Ethernet
DEVICE=eth0
HWADDR=00:16:3E:25:A4:4E
BOOTPROTO=none
IPADDR=62.76.220.14
PREFIX=24
GATEWAY=62.76.220.2
DNS1=10.16.1.205
IPV6INIT=no
ONBOOT=yes
DOMAIN=cs.vsu.ru
NM_CONTROLLED=no
```

- конфигурирование с использованием утилиты `system-config-network-tui`:
`/etc/sysconfig/networking/devices`, `/etc/sysconfig/networking/profiles`
- конфигурирование с использованием системы `NetworkManager`, состоящей из демона, `nmcli`, `nmtui`, поддерживающей `GNOME control-center`.

Конфигурирование сети, GUI



Службы наблюдения, протоколирования. Система регистрации событий BSD SYSLOG

- В ОС UNIX нет встроенной системы аудита, а есть регистратор событий SYSLOG (XPG4-UNIX, CISCO, BSD, RFC-3164,5424)
 - facility – система, источник сообщения
 - kern, user, mail, daemon, auth, local0 .. local7
 - соответствующие константы LOG_KERN, ...
 - priority/level – важность сообщения
 - emerg, alert, crit, err, warning, notice, info, debug
 - соответствующие константы: LOG_EMERG, ...
 - Использует дейтаграммный транспорт – UDP/IP, порт 514 (при запуске демона syslogd в сетевом режиме: `syslogd -r`)

Конфигурация SYSLOG

```
#  
# Конфигурационный файл демона syslogd  
#  
kern.debug          /var/adm/syslog/kern.log  
kern.debug          /dev/console  
daemon.debug       /var/adm/syslog/daemon.log  
auth.debug         /var/adm/syslog/auth.log  
syslog.debug       /var/adm/syslog/syslog.log  
*.notice;mail.info /var/adm/syslog/mail  
*.crit             /var/adm/syslog/critical  
kern.err           @nix.cs.vsu.ru  
*.emerg           *  
*.alert           andrey, sergey  
auth.warning       ivan
```

IOS и SYSLOG

- На стороне маршрутизатора с IOS:

- `router(config)# logging 62.76.220.14`
- `router(config)# logging source-interface ethernet 0`
- `router(config)# logging facility local7`

- На стороне машины-логгера [62.76.220.14] в конфигурации демона `syslog.conf`:

- `local7.* /var/log/cisco.log`

Недостатки и достоинства SYSLOG

- **Некоторые недостатки:**
 - события поступают от приложений и проверить правильность их нельзя;
 - нельзя проверить источник сообщения;
 - тип сообщения определяет приложение;
 - нет защиты от привилегированного приложения/пользователя, что частично компенсируется немодифицируемостью вывода (например, печать, почта)
- **Некоторые достоинства:**
 - поддерживает прием сообщений по сети;
 - возможно создание изолированного сервера-регистратора;
 - поддерживается сетевым оборудованием: маршрутизаторы, управляемые коммутаторы, точки доступа беспроводных сетей и т.п.
- **Развитие: SYSLOG-NG, RSYSLOG (более распространен) : TLS, ms временные метки, возможность работать через TCP, внешние СУБД)**

MINIX, Linux, GNU

- MINIX, микроядро, Andrew Tanenbaum, 1987



- Linux v. 0.01, монолитное ядро, i386, Linus Torvalds, 1991



- Linux v 1.0, 1994

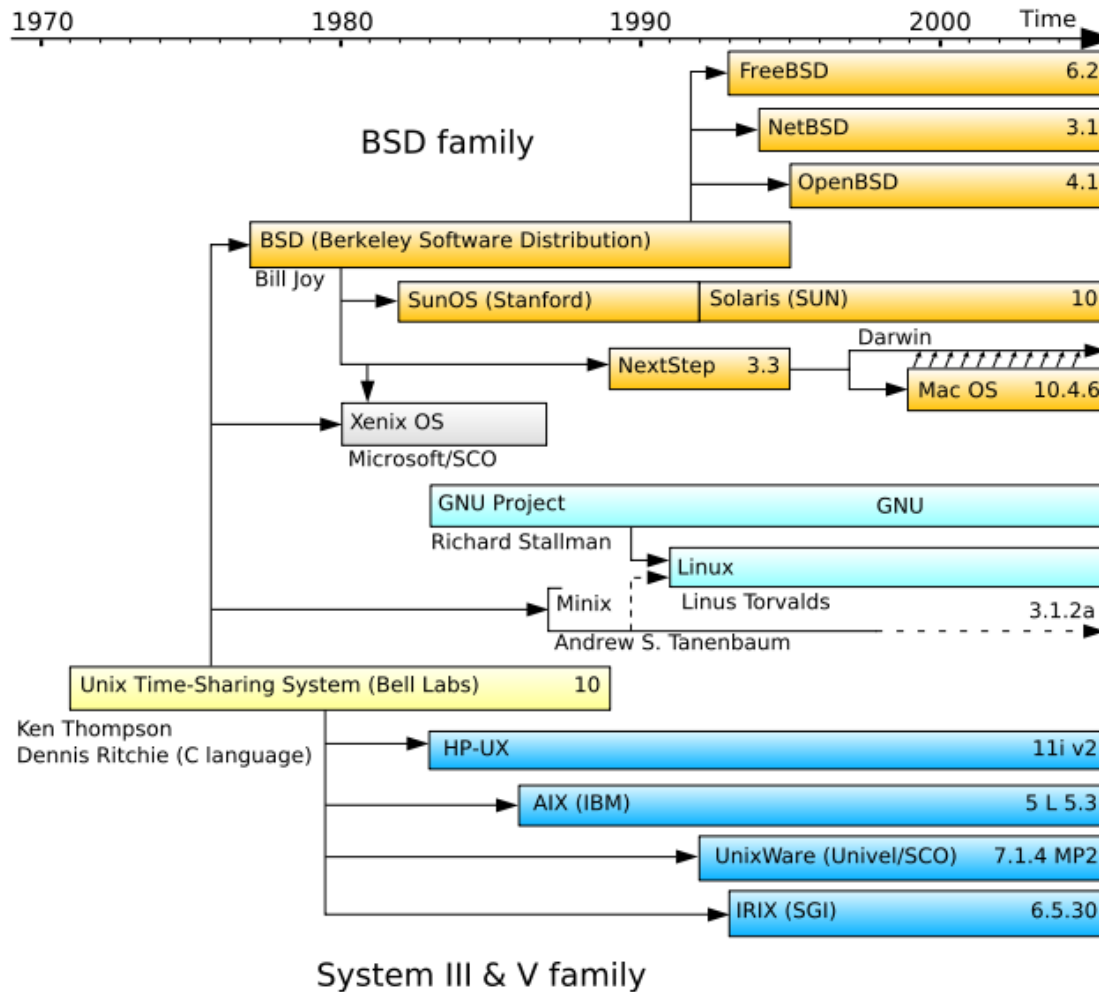
- Linux v 2.0, 1996

- GPL, FSF, Richard Stallman, см.:

<http://www.cs.vsu.ru/~kas/doc/unix/GNU.htm>

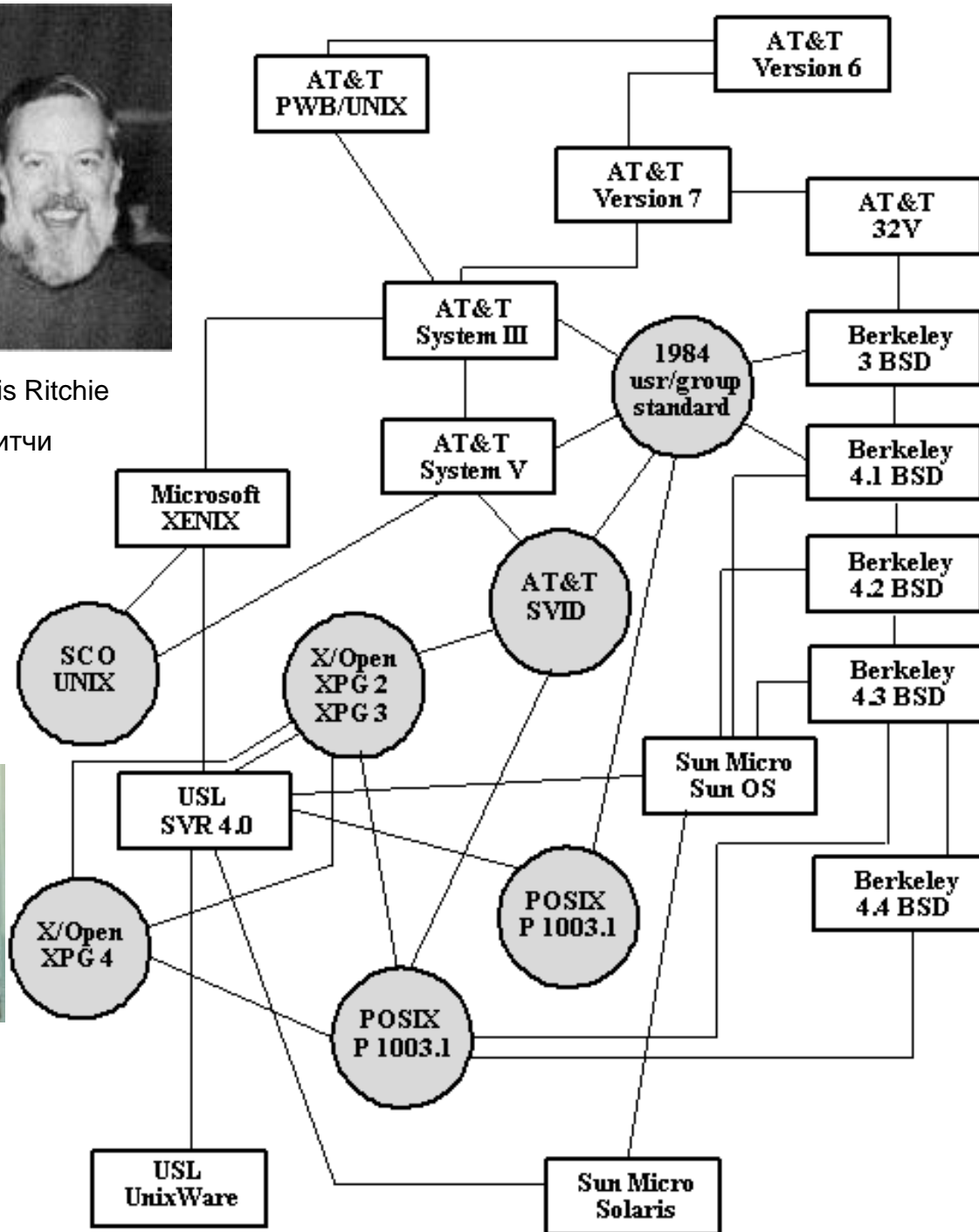


BSD, GNU/Linux, SystemV





Ken Thompson & Dennis Ritchie
 Кен Томпсон и Денис Ритчи



Bill Joy
 Билл Джой



Направления SystemV и BSD

- SystemV UNIX (AT&T)
 - Предоставление лицензий (в т.ч. образовательным учреждениям), поставка системы в исходных текстах фирмой AT&T, 1975
 - Объединение различных версий в System III, 1983, затем System V, 1984
 - Стандартизация SVID (System V Interface Definition)
 - Выпуск последующих версий SVR2-SVR4, 1984-1989
 - Продажа UNIX-бизнеса AT&T - Novell, 1993
 - Передача Novell права на ТМ и на сертификацию ПО на ТМ консорциуму X/Open (после объединения с OSF, The Open Group)
 - Продажа UNIX-бизнеса Novell - Santa Cruz Operation (SCO), 1995
 - Продажа UNIX-бизнеса SCO - Caldera Systems (позже переименована в SCO Group), суды против Linux-поставщиков (лицензионные отчисления), суды SCO Group ⇔ Novel (права на исходный код System V), 2000 по н.в.
- Berkeley Software Distribution
 - UNIX 6 ред., университет в Беркли, 1975 (Билл Джой)
 - 1BSD, 2BSD – PDP11, гранты DARPA, 1978
 - 3BSD – 1979, 4.3BSD (TCP/IP, виртуальная память) 1981
 - 4.4BSD (в последствии основа FreeBSD) 1993, суды...

Взаимодействие процессов

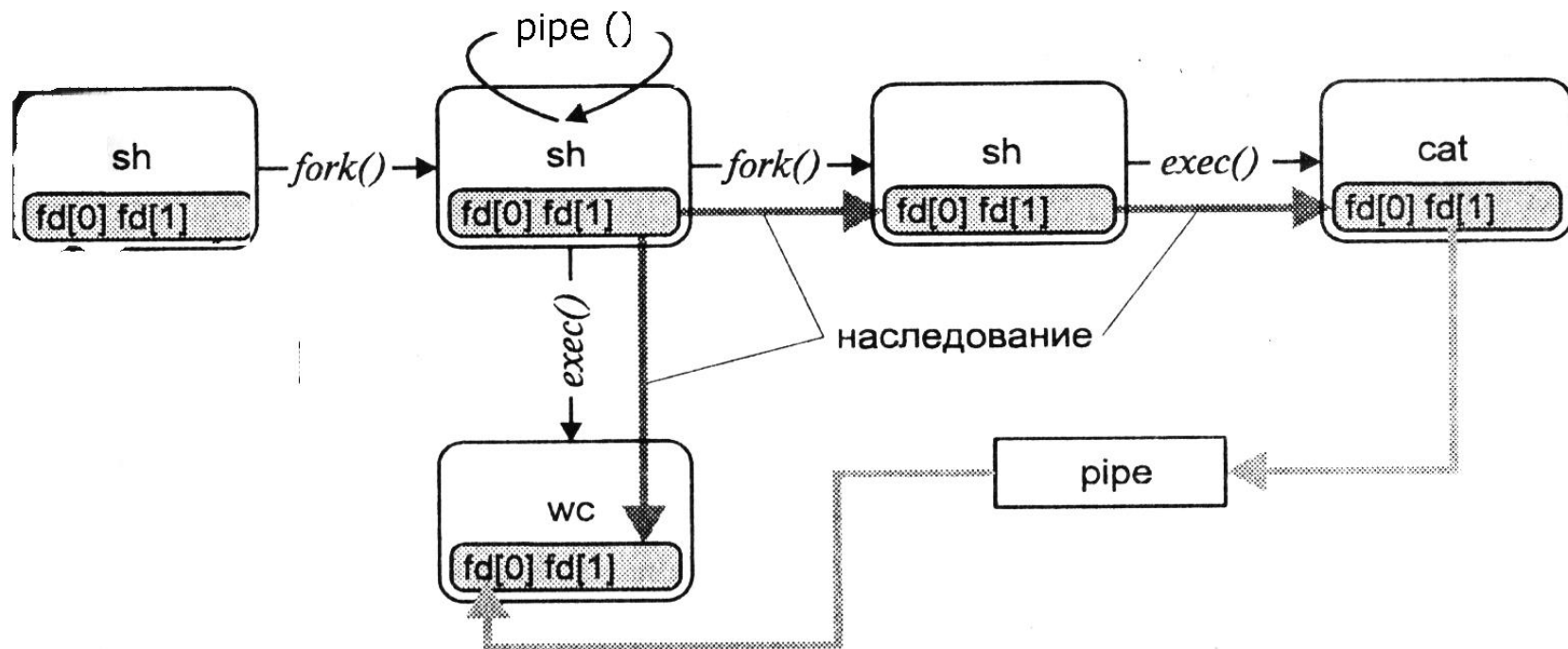
- каналы (pipes)
- именованные каналы (FIFO, named pipes)
- сигналы
- SystemV IPC: семафоры, очереди сообщений, разделяемая память
- сокеты (BSD sockets)

Каналы. Пример: cat filename | wc

Каналы (неименованные каналы) основаны на pipe (int *filedes)

filedes[0] – запись в канал; filedes[1] - чтение из канала

Область применения – родственные процессы



Именованные каналы (fifo, клиент)

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#define FIFO "fifo.1"

main()
{
    int writefd;

    writefd=open(FIFO,O_WRONLY);
    write(writefd, "Hi!\n", 4);
    close(writefd);
    unlink(FIFO);
    exit(0);
}
```

Сигналы

```
#include <signal.h>

static void sig_hndlr(int signo)
{
    signal(SIGINT, sig_hndlr);
    printf ("Got Signal\n");
}

main() {

    signal(SIGINT, sig_hndlr);
    signal(SIGUSR1, SIG_DFL);
    signal (SIGUSR2, SIG_IGN);

    while(1)
    pause() ;
}
```

Утилиты:
man kill

Надежные (reliable) сигналы

```
static void sig_hndlr(int signo)
{
    printf("Got signal!\n");
}

void(*mysignal(int signo,void(*hndlr)(int)))(int)
{
    struct sigaction act,oact;
    act.sa_handler=hndlr;
    sigemptyset(&act.sa_mask);
    act.sa_flags=0;
    if (signo!=SIGALRM)
        act.sa_flags|=SA_RESTART;
    sigaction(signo,&act,&oact);
}

main()
{
    mysignal(SIGINT,sig_hndlr);
    mysignal(SIGUSR1,SIG_DFL);
    mysignal(SIGUSR2,SIG_IGN);
    while(1) pause();
}
```

Для управления набором сигналов `sigset_t` применяют: `sigemptyset()`, `sigfillset()`, `sigaddset()`, `sigdelset()`.
Для управления диспозицией: `sigaction()`

Структура `sigaction`:

```
struct sigaction {
    void (*sa_handler)(int);
    void (*sa_sigaction)(int, siginfo_t *, void *);
    sigset_t sa_mask;
    int sa_flags;
    void (*sa_restorer)(void);
}
```

Интерфейс сигналов POSIX.1,
основанный на 4.2BSD

IPC SystemV, операции семафоров. Файл - shmем.h.

```
#define MAXBUFF 80
#define PERM 0600

typedef struct mem_msg
{
    int segment;
    char buff[MAXBUFF];
} Message;
```

```
static struct sembuf proc_wait[1]={1,-1,0};
static struct sembuf proc_start[1]={1,1,0};
static struct sembuf mem_lock[2]={0,0,0,0,1,0};
static struct sembuf mem_unlock[1]={0,-1,0};
```

См. подробнее (IPC SystemV):

http://www.citforum.ru/operating_systems/bach/glava_98.shtml

Элементы списка операций:

- номер семафора;
 - код операции;
 - флаги.
- В данной примере:
• 0 – разблокировано
• 1 - заблокировано

Коды операций и значения семафоров

- Ядро меняет значение семафора в зависимости от кода операции, который может быть равен 0, больше 0 или меньше 0
- Если код операции имеет положительное значение, ядро увеличивает значение семафора и выводит из состояния приостанова все процессы, ожидающие наступления этого события.
- Если код операции равен 0, процесс ожидает обнуления семафора
 - ядро проверяет значение семафора: если оно равно 0, ядро переходит к выполнению других операций;
 - в противном случае ядро увеличивает число приостановленных процессов, ожидающих, когда значение семафора станет нулевым, и процесс "засыпает".
- Если код операции имеет отрицательное значение и
 - если его абсолютное значение не превышает значение семафора, ядро прибавляет код операции (отрицательное число) к значению семафора. Если результат равен 0, ядро выводит из состояния приостанова все процессы, ожидающие обнуления значения семафора.
 - Если результат меньше абсолютного значения кода операции, ядро приостанавливает процесс до тех пор, пока значение семафора не увеличится. Если процесс приостанавливается посреди операции, он имеет приоритет, допускающий прерывания; следовательно, получив сигнал, он выходит из этого состояния.
- semop (дескриптор, структура с операциями, число операций)

IPC SystemV, сервер

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
#include<sys/shm.h>
#include"shmem.h"

main()
{
    Message *msgptr;
    int semid,shmid;
    key_t key;

    key=ftok("srv",'A');
    shmid=shmget(key,sizeof(Message),PERM|IPC_CREAT);
    msgptr=(Message *)shmat(shmid,0,0);
    semid=semget(key,2,PERM|IPC_CREAT);
    semop(semid,&proc_wait[0],1);
    semop(semid,&mem_lock[0],2);
    printf("%s",msgptr->buff);
    semop(semid,&mem_unlock[0],1);
    shmdt(msgptr);
    exit(0);
}
```

Утилиты:

ipcs

lpcrm

semtool

в код добавлены небольшие преднамеренные ошибки ... может быть

IPC SystemV, клиент

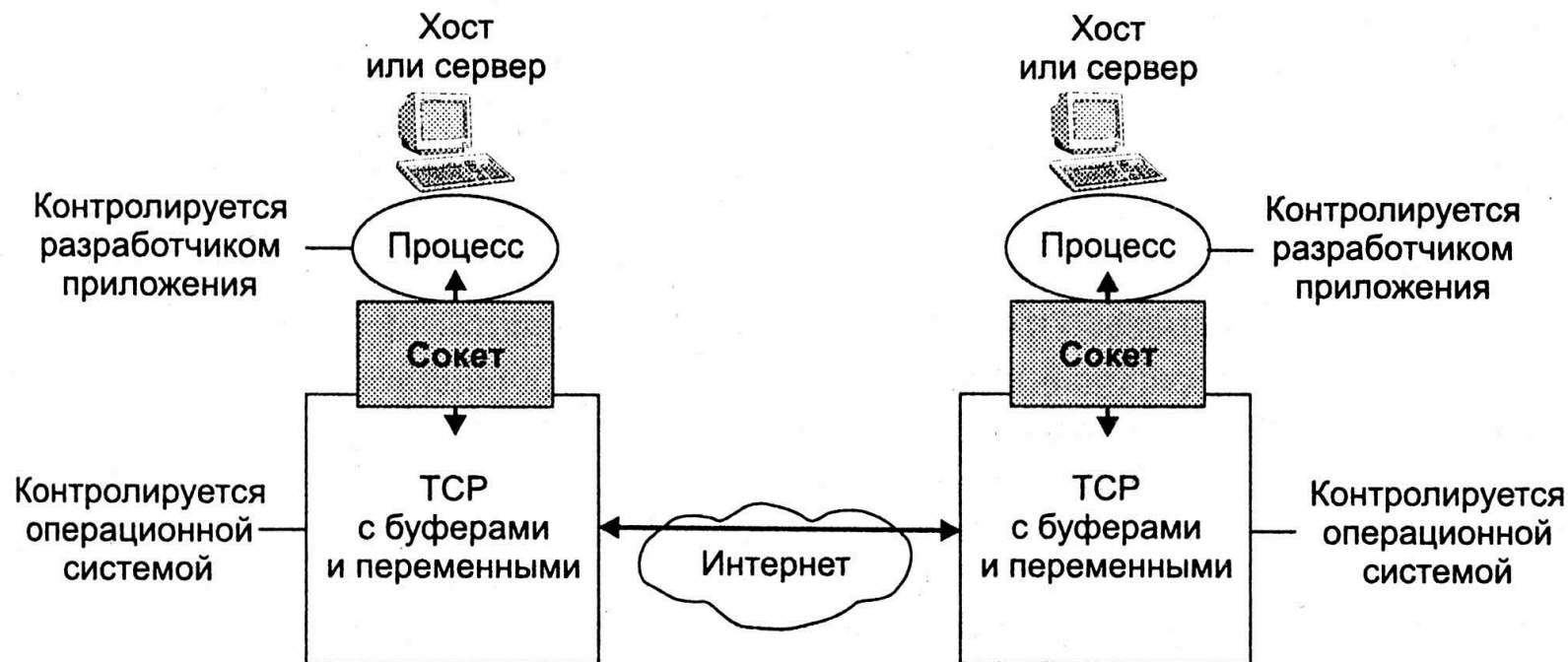
```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
#include<sys/shm.h>
#include"shmem.h"

main()
{
    Message *msgptr;
    key_t key;
    int semid,shmid;

    key=ftok("server",'A');
    shmid=shmget(key,sizeof(Message),0);
    msgptr=(Message *)shmat(shmid,0,0);
    semid=semget(key,2,PERM);
    semop(semid,&mem_lock[0],2);
    semop(semid,&proc_start[0],1);
    printf(msgptr->buff,"Hi!\n");
    semop(semid,&mem_unlock[0],1);
    semop(semid,&mem_lock[0],2);
    shmdt(msgptr);
    shmctl(shmid,IPC_RMID,0);
    semctl(semid,0,IPC_RMID);
}
```

в код добавлены небольшие преднамеренные ошибки ... может быть.

Процессы приложения, сокеты и протокол транспортного уровня

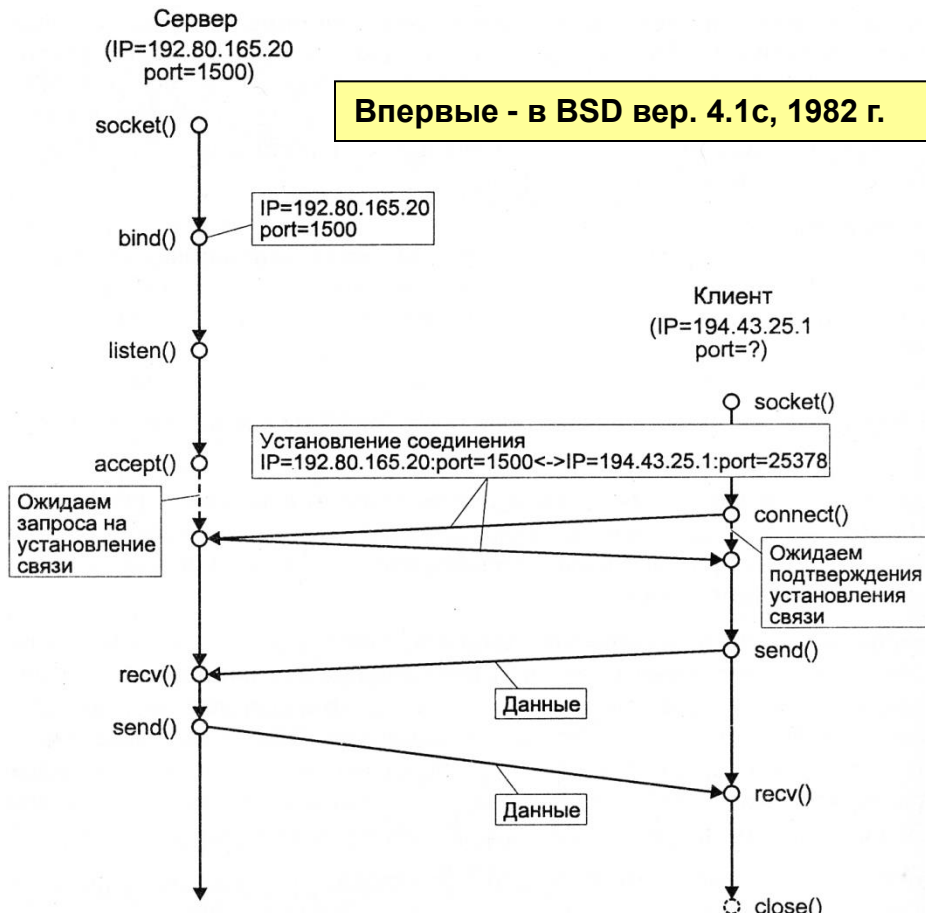


Сокет - конечная точка сетевых коммуникаций

- имеет тип (raw, dg, stream, packet) и ассоциированный с ним процесс.

- существуют внутри коммуникационных доменов: *UNIX, Internet*

Программирование с использованием BSD сокетов



```
main()
{
    int s,ns,pid,nport;
    struct sockaddr_in serv_addr,clnt_addr;
    struct hostent *hp;
    char buff[80],hname[80];

    nport=1500;
    nport=htons((u_short)nport);
    s=socket(AF_INET,SOCK_STREAM,0);
    bzero(&serv_addr,sizeof(serv_addr));
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=nport;
    bind(s,(struct sockaddr *)&serv_addr,sizeof(serv_addr));
    printf("Server is OK!\n");
    listen(s,5);
    while(1)
    {
        int addrlen;
        addrlen=sizeof(clnt_addr);
        bzero(&clnt_addr,sizeof(clnt_addr));
        ns=accept(s,(struct sockaddr *)&clnt_addr,&addrlen);
        pid=fork();
        if (pid==0)
        {
            int nbytes;font;
            while (nbytes=(recv(ns,buff,sizeof(buff),0))!=0)
            {
                send(ns,buff,nbytes,0);
            }
            close(ns);
            exit(0); }
        close(ns);
    }
}
```

Сервер BSD Sockets

```
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<stdio.h>
#include<fcntl.h>
#include<netdb.h>

#define PORT 1500

main()
{
    int s,ns,pid,nport;
    struct sockaddr_in serv_addr,clnt_addr;
    struct hostent *hp;
    char buff[80],hname[80];

    nport=PORT;
    nport=htons((u_short)nport);
    s=socket(AF_INET,SOCK_STREAM,0);
    bzero(&serv_addr,sizeof(serv_addr));
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=nport;
    bind(s,(struct sockaddr *)&serv_addr,sizeof(serv_addr));
    printf("Server is OK!\n");
    listen(s,5);
```

```
while(1)
{
    int addrlen;
    addrlen=sizeof(clnt_addr);
    bzero(&clnt_addr,sizeof(clnt_addr));
    ns=accept(s,(struct sockaddr *)&clnt_addr,&addrlen);
    pid=fork();
    if (pid==0)
    {
        int nbytes,font;
        while (nbytes=(recv(ns,buff,sizeof(buff),0))!=0)
        {
            send(ns,buff,nbytes,0);
        }
        close(ns);
        exit(0);
    }
    close(ns);
}
```

Утилиты:

netstat

telnet hostname port

в код добавлены небольшие преднамеренные ошибки ... может быть

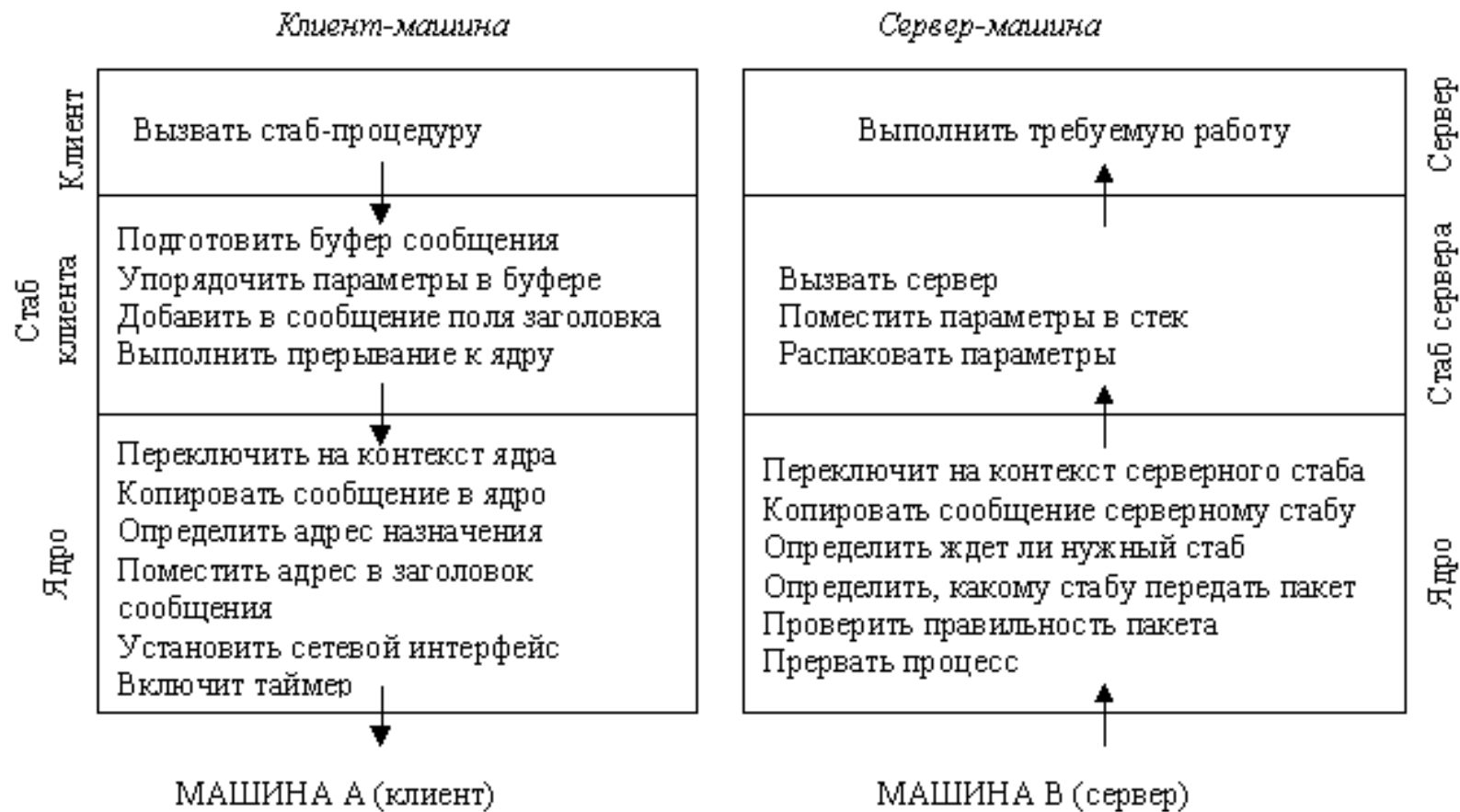
Удаленный вызов процедур

- RPC (Remote Procedure Call) — удаленный вызов процедур.
- Если сокеты – своеобразное продолжение файлового интерфейса для сети, то RPC – продолжение локальных процедур для сети.
- Основная идея RPC - скрыть распределенный характер вызова процедур.
- Клиент и сервер являются независимыми от сетевой реализации, а вызовы процедур имеют стандартный интерфейс.
- Впервые RPC был реализован компанией Sun Microsystems в 1984 году в рамках ее продукта NFS (Network File System - сетевая файловая система).

Этапы RPC

1. Программа-клиент производит локальный вызов процедуры, называемой заглушкой (stub). Задача заглушки - принять аргументы, предназначенные удаленной процедуре, преобразовать их в формат XDR и сформировать сетевой запрос. Упаковка аргументов и создание сетевого запроса называется **сборкой (marshalling)**.
2. Сетевой запрос **пересылается** по сети на удаленную систему. При этом могут быть использованы различные транспортные протоколы, например, UDP или TCP, в IP-сетях.
3. На удаленном хосте заглушка сервера ожидает запрос и при получении извлекает параметры - аргументы вызова процедуры. **Извлечение (unmarshalling)** проводит необходимые преобразования (например, изменения порядка расположения байтов).
4. Заглушка сервера выполняет вызов процедуры-сервера, которой адресован запрос клиента, передавая ей полученные по сети аргументы.
5. После выполнения процедуры управление возвращается в заглушку сервера, передавая ей требуемые параметры. Как и заглушка клиента; заглушка сервера преобразует возвращенные процедурой значения, формируя сетевое сообщение-отклик, который передается по сети системе, от которой пришел запрос.
6. Операционная система передает полученное сообщение заглушке клиента, которая передает значения отклика клиенту, воспринимающему это как обычный возврат из процедуры.

Порядок вызова процедур в RPC



Реализация RPC

- Реализация:
 - Демон portmapper – сервисный брокер
 - Сами серверы RPC
- Службы, основанные на RPC: NFS, NIS

```
#  
# /etc/rpc - miscellaneous RPC-based services  
#  
portmapper      100000  portmap sunrpc  
rstatd          100001  rstat rstat_svc rup perfmeter  
rusersd         100002  rusers  
nfs             100003  nfsprog  
ypserv          100004  ypprog  
mountd          100005  mount showmount  
ypbind          100007  
walld           100008  rwall shutdown  
yppasswdd       100009  yppasswd  
ypupdated       100028  ypupdate
```

Network Information System (YP, NIS, NIS+, NYS)

- Разработка Sun Microsystems, начальное название Yellow Pages, YP
- Обеспечивает сетевой доступ к базе данных административной информации о пользователе, узле и т.п.
- Состав: сервер, библиотека клиента, утилиты администрирования.

Данные NIS

- Данные хранятся в таблицах, т.н. «maps»:

Master File	Map(s)
/etc/hosts	hosts.byname, hosts.byaddr
/etc/networks	networks.byname, networks.byaddr
/etc/passwd	passwd.byname, passwd.byuid
/etc/group	group.byname, group.bygid
/etc/services	services.byname, services.bynumber
/etc/rpc	rpc.byname, rpc.bynumber
/etc/protocols	protocols.byname, protocols.bynumber
/usr/lib/aliases	mail.aliases

Network File System (NFS)

- Наиболее известная служба, основанная на RPC
- Реализация
 - на стороне сервера – демон(ы) `rpc.nfsd` (NFS Daemon)
 - таблица разделяемых каталогов - `/etc/exports`
 - на стороне клиента – демон `biod` (Block I/O Daemon)

```
# exports file for nix.cs.vsu.ru
/home          nix1(rw) nix2(rw)
/usr/docs      nix3(ro)
/var/spool/mail nix*.cs.vsu.ru (rw)
```

→ `mount -t nfs nix.cs.vsu.ru:/home /home/users`

Программа `mount` соединится с демоном `rpc.mountd` на узле `nix.cs.vsu.ru`
Передача данных будет происходить с помощью системных вызовов,
посылаемых демону `rpc.nfsd`

iptables

- ipfwadm (ipfw, BSD, Win, Linux) => ipchains (stateless, Russell) => netfilter/iptables (statefull) ...=> nftables (правила транслируются в псевдокод)
- набор правил для пакетов:
 - критерий => действие (ACCEPT, DROP, REJECT, MASQUERADE ...)
 - цепочки (продолжение идеи ipchains):
 - PREROUTING
 - INPUT
 - FORWARD
 - OUTPUT
 - POSTROUTING
 - таблицы (новое в iptables – объединение цепочек по функциям):
 - filter (по-умолчанию INPUT, FORWARD, OUTPUT),
 - nat (PREROUTING, OUTPUT, POSTROUTING), mangle (PREROUTING, OUPUT,)
новая nat (добавлены INPUT, FORWARD, POSTROUTING)
 - raw (PREROUTING, OUTPUT)
 - вывод содержимого таблицы: iptables -t nat -L ; iptables-save.; iptables-restore;
 - Ip6tables; ip6tables-save; ip6tables-restore
- conntrack
- netstat-nat

/etc/init.d/iptables

Исполняемые файлы:

iptables-save - вывод правил IP Tables на STDOUT

iptables-restore – восстановление IP Tables с STDIN

Переменные:

`IPTABLES=iptables`

`IPTABLES_DATA=/etc/sysconfig/${IPTABLES}` (правила)

`IPTABLES_FALLBACK_DATA=${IPTABLES_DATA}.fallback`

`IPTABLES_CONFIG=/etc/sysconfig/${IPTABLES}-config` (конфиг)

`$ sudo /etc/init.d/iptables save`

```
-rw-----. 1 root root 2453 Nov  9 12:49 /etc/sysconfig/iptables
```

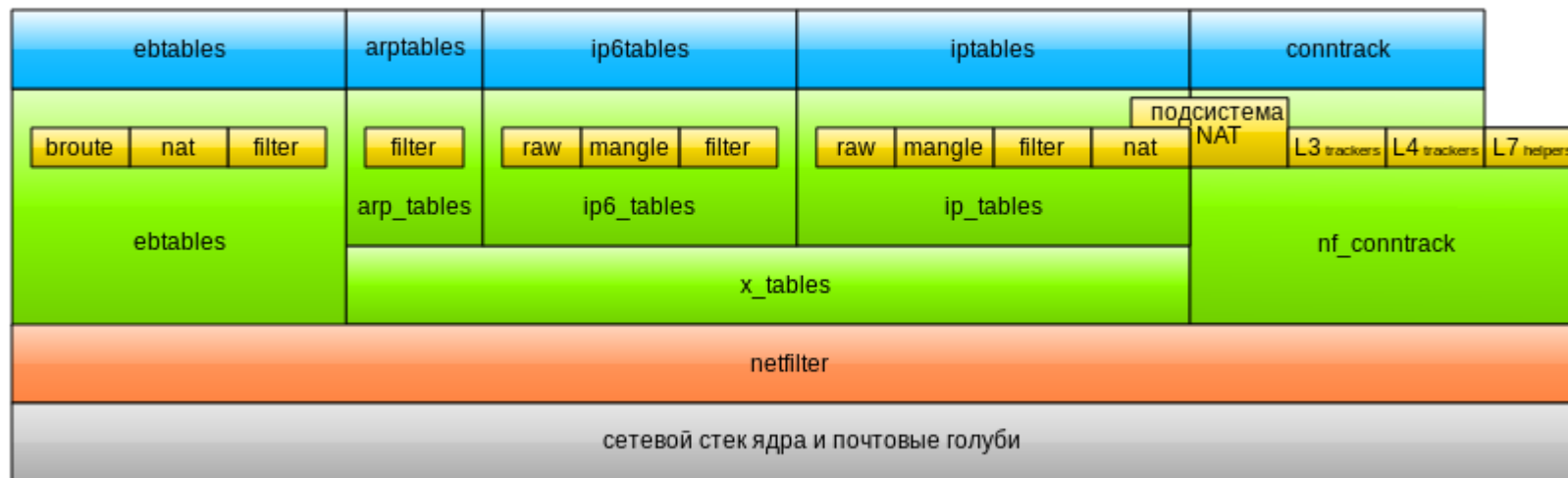
```
-rw-----. 1 root root 1974 Jul 24 05:10 /etc/sysconfig/iptables-config
```

```
-rw-----. 1 root root 2556 Nov  9 12:49 /etc/sysconfig/iptables.save
```

Netfilter

Компоненты Netfilter

Jan Engelhardt, 2008-06-17, updated 2008-12-13
Sergey Ptashnick, 2010-02-24 (перевод, уточнения)



Userspace-утилиты

Компоненты ядра

<http://ru.wikibooks.org/wiki/Iptables>

Политики безопасности (в части управления доступом)

- Политика безопасности – интегральная характеристика защищаемой системы
- Политика безопасности включает:
 - множество возможных операций над объектами
 - для каждой пары субъект-объект множество разрешенных операций, являющееся подмножеством всего множества возможных операций
- Типы политик безопасности (в части управления доступом):
 - Дискреционная (дискретная, Discretionary Access Control - DAC)
 - Мандатная (полномочная, Mandatory Access Control MAC)

Типы политик безопасности (в части управления доступом)

- Типы политик безопасности
 - Дискреционная (дискретная, Discretionary Access Control - DAC)
 - все объекты и субъекты идентифицированы
 - права доступа субъекта к объекту определяются внешним правилом
 - Мандатная (полномочная, Mandatory Access Control MAC)
 - все объекты и субъекты идентифицированы
 - задан упорядоченный набор меток секретности
 - каждому объекту присвоена метка секретности – уровень секретности
 - каждому субъекту присвоена метка секретности – уровень доступа
 - Помимо «вертикальных» (levels), могут задаваться горизонтальные (categories) уровни контроля
 - Основная цель данной системы – противодействие возникновению информационных каналов сверху вниз

Разграничение доступа в «классическом» UNIX (DAC)

- Классическая схема:
 - UID=0 (GID=0) – суперпользователь
 - все остальные – «простые» пользователи
- SetUID, SetGID
- SUDO (/etc/sudoers)

AppArmor, Application Armor

- Модуль ядра Linux AppArmor (броня) позволяет связывать программы с профилями безопасности, ограничивающими возможности программ, т.о. обеспечивается подход MAC.
- GNU лицензия
- Включен в ядро 2.6.36 и старше
- Неясные перспективы в связи с SELinux. Одно из отличий: AA использует абсолютные пути ФС, а SELinux – индексные дескрипторы inode (отсюда нюансы с жесткими линками и обновлением/замещением файлов)

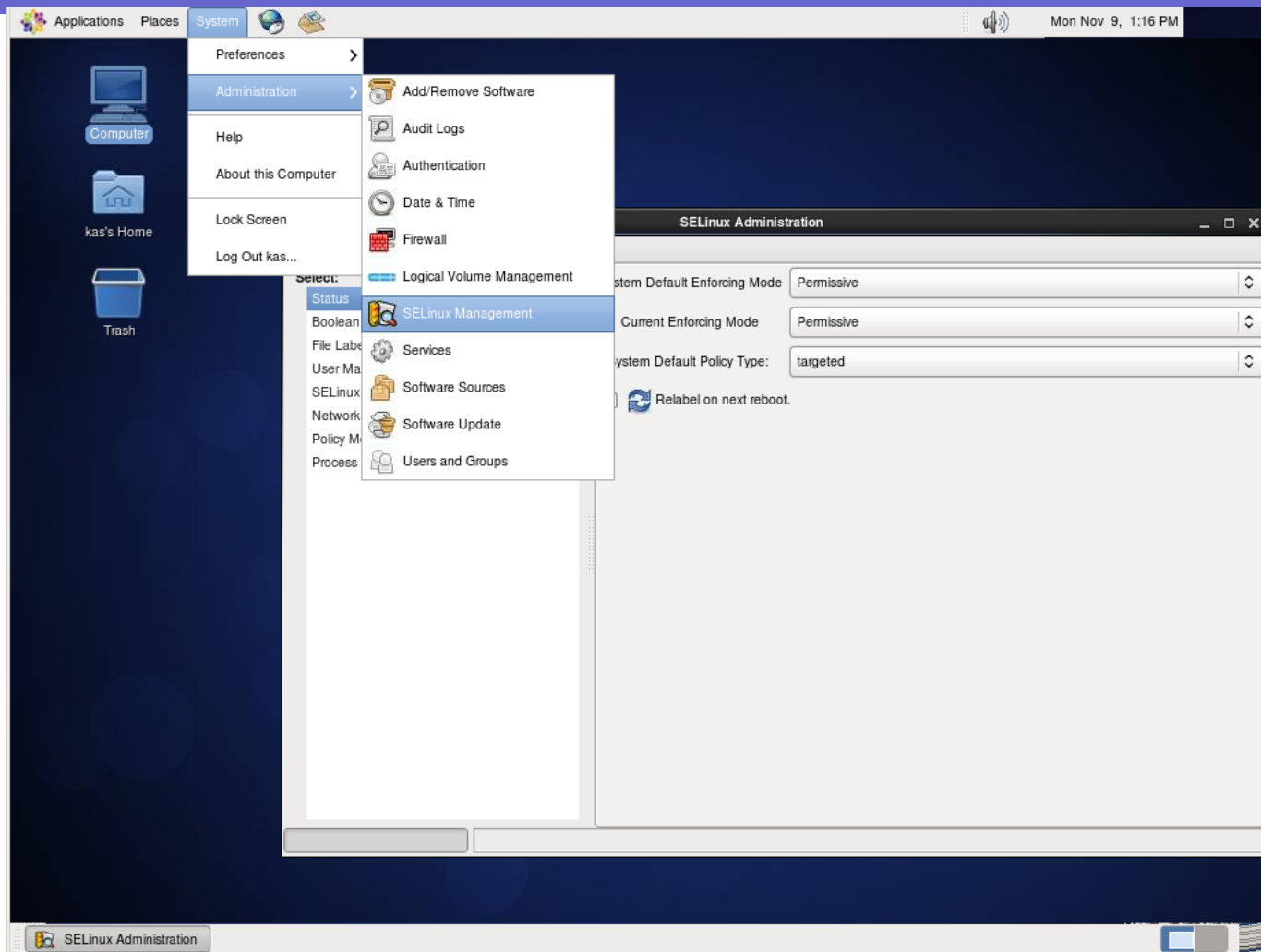
SELinux (Security-Enhanced Linux)

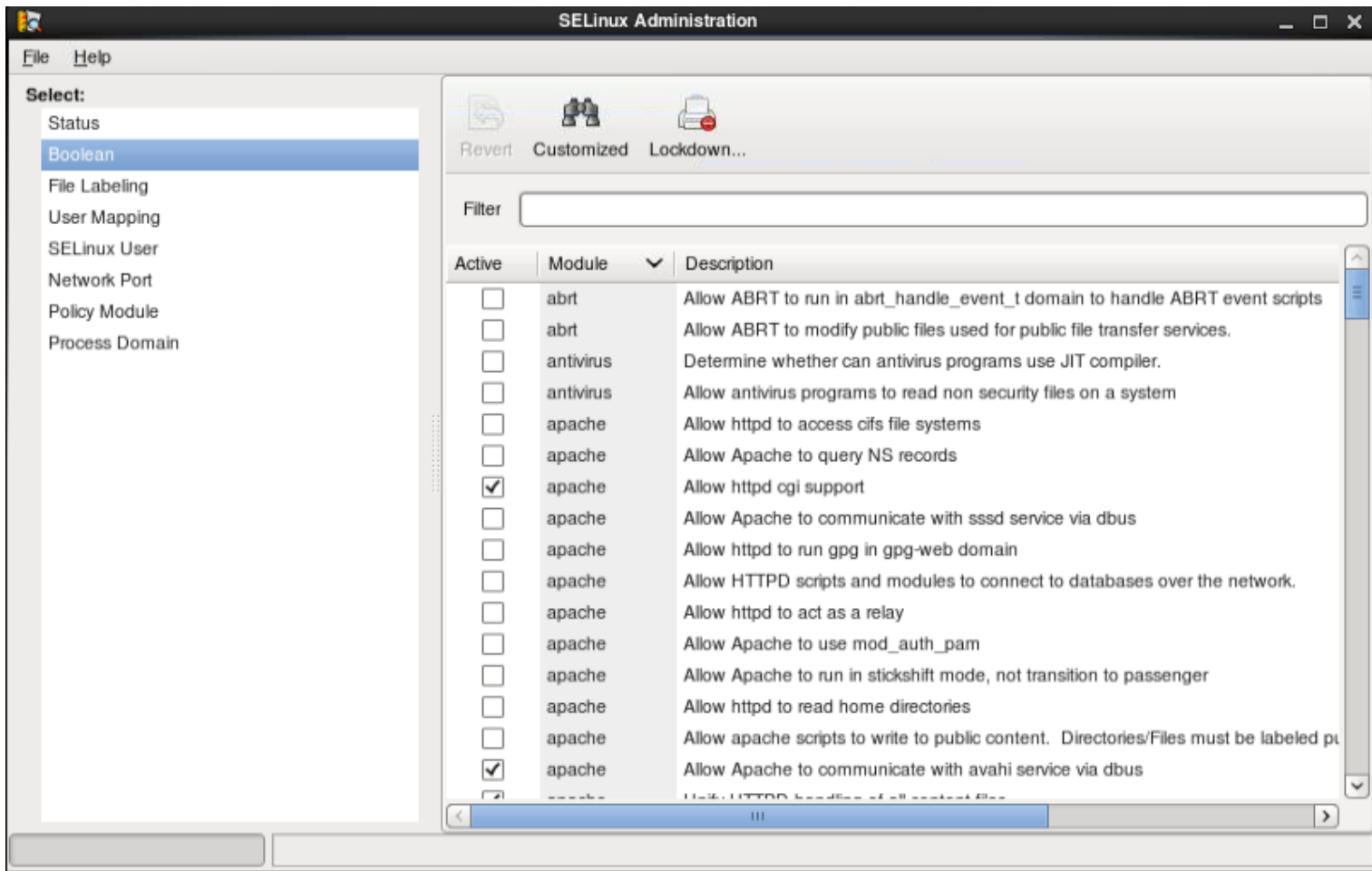
- SELinux
 - разработан NSA с GNU лицензией
 - реализация мандатного метода доступа, действующего **после** дискреционного, т.о. это гибридный DAC/MAC подход к ИБ.
 - реализовано в ядре Linux
 - по-умолчанию включен в RedHat/CentOS
- RedHat (/etc/sysconfig/selinux)
 - SELINUX = disabled, permissive, enforcing
 - SELINUXTYPE = targeted, strict, test...
 - Перезагрузка!

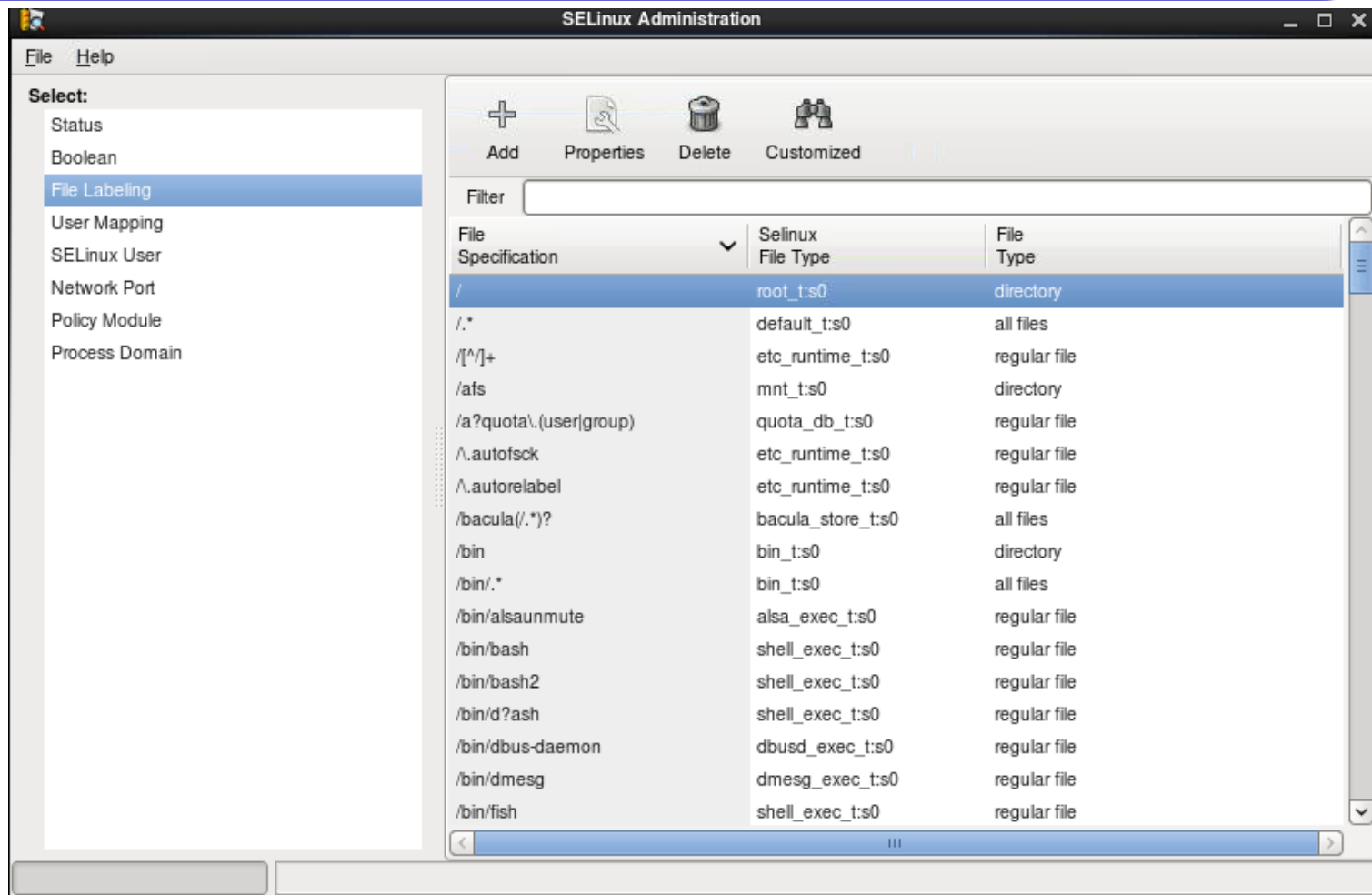
SELinux

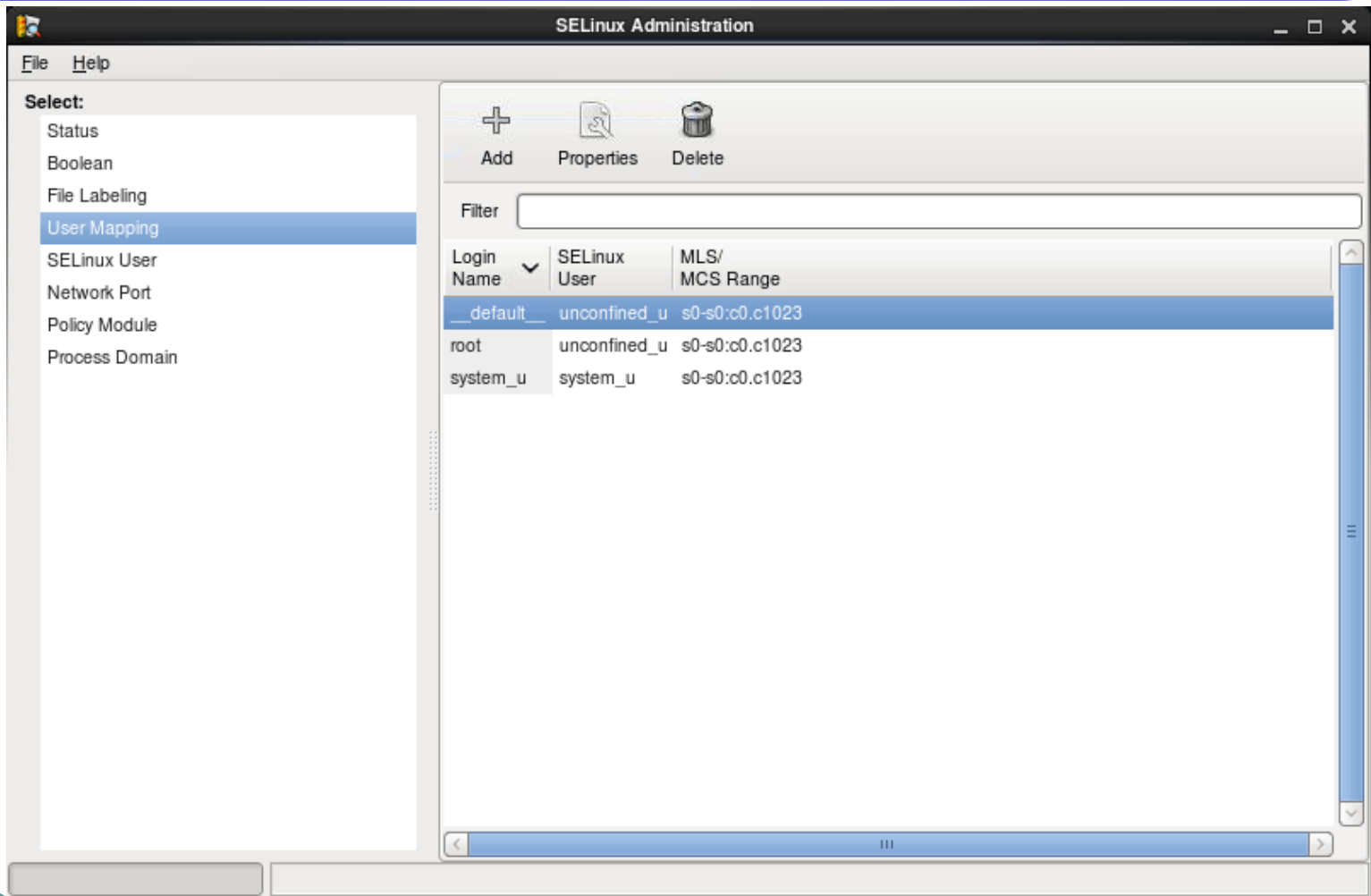
- Наряду с UID/GID добавляет к файлам/процессам:
 - пользователя SELinux (SELinux user)
 - роль (role)
 - домен/тип (domain/type)
 - метки секретности (sensitivity label)
- Требования
 - модули ядра SELinux, AUDIT
 - ФС с "extended attributes" (модуль ядра XATTRs)
 - готовая (default) политика
 - утилиты

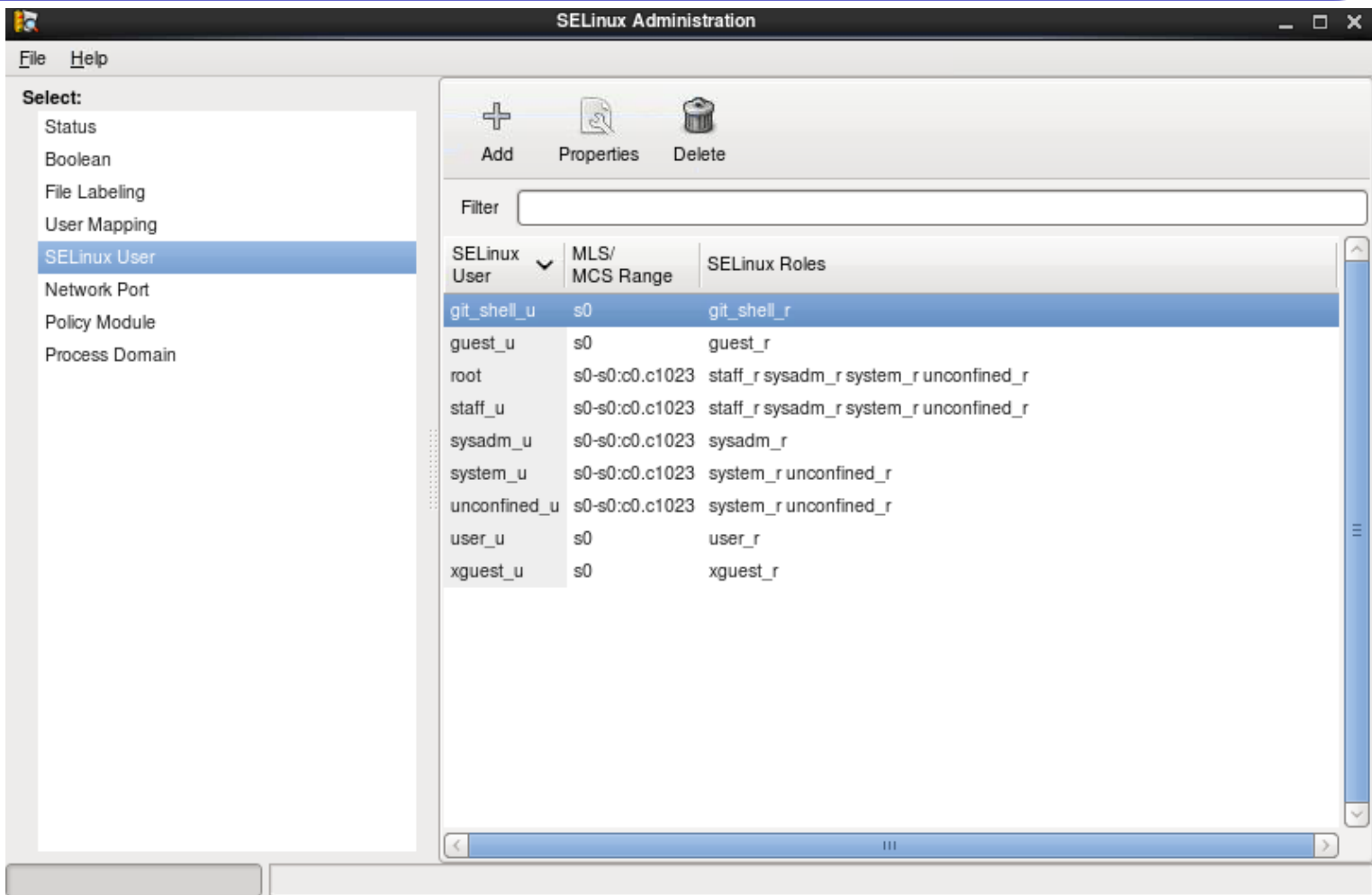
Управление SELinux в CentOS с помощью GUI

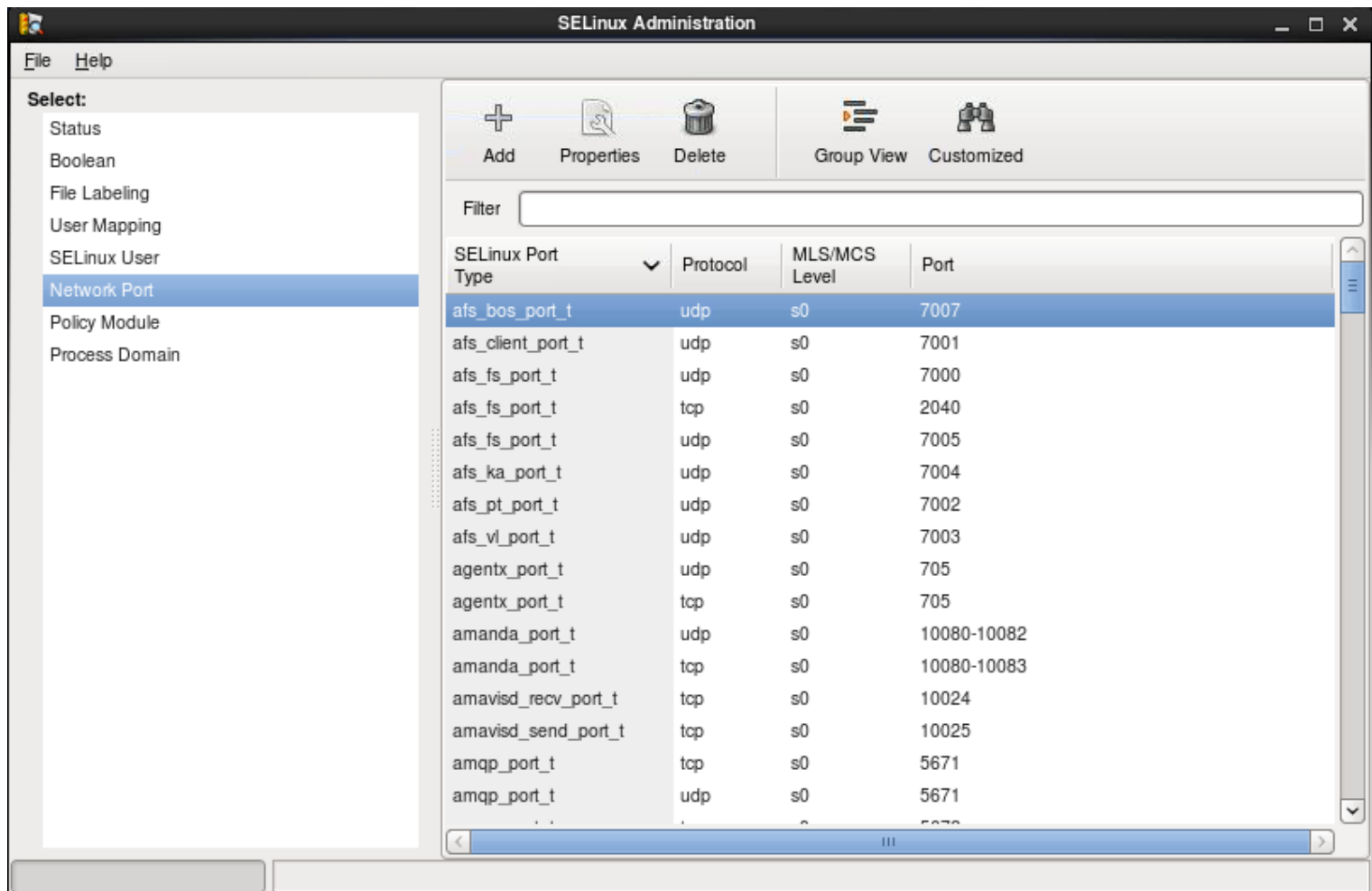


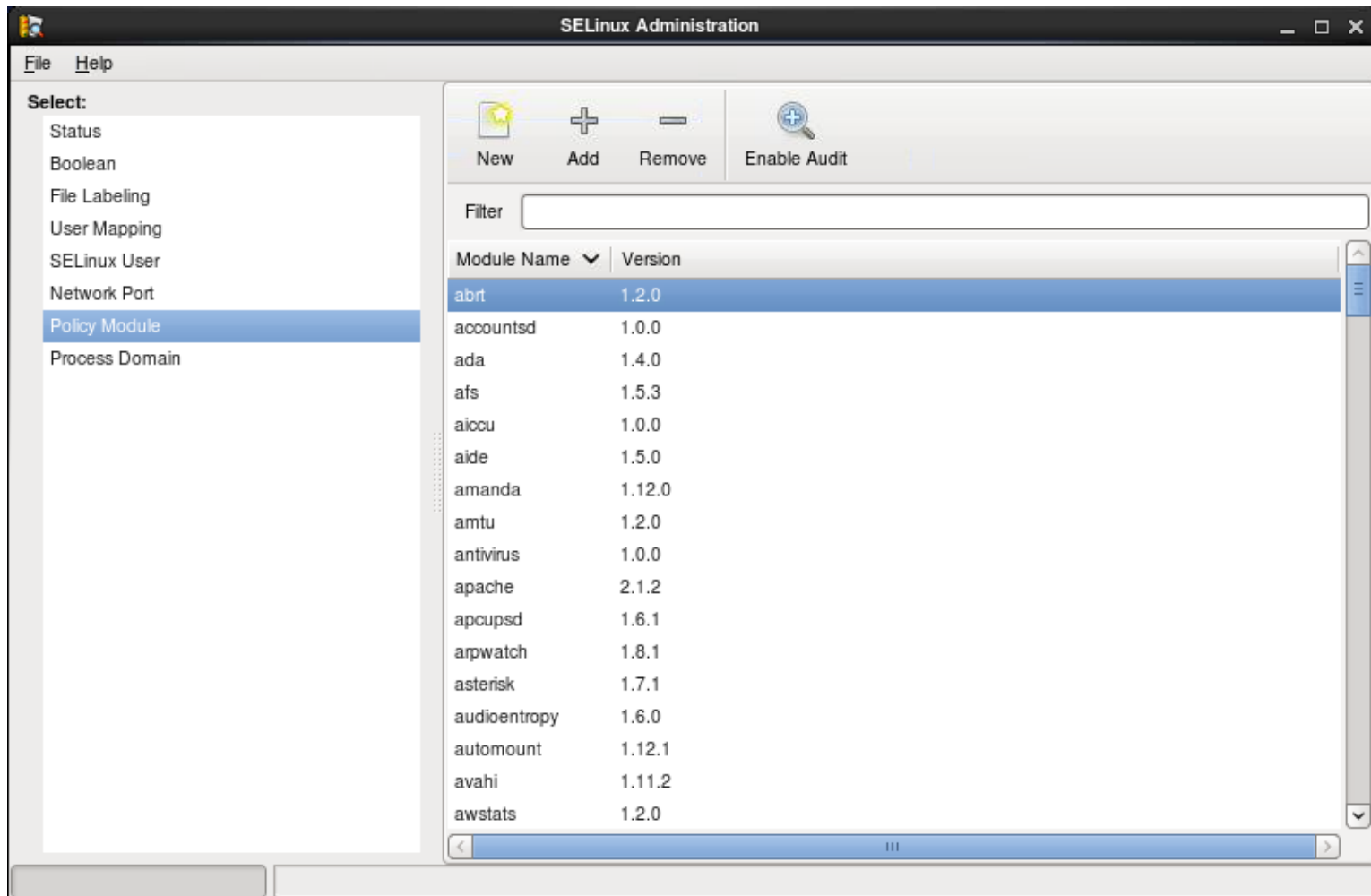


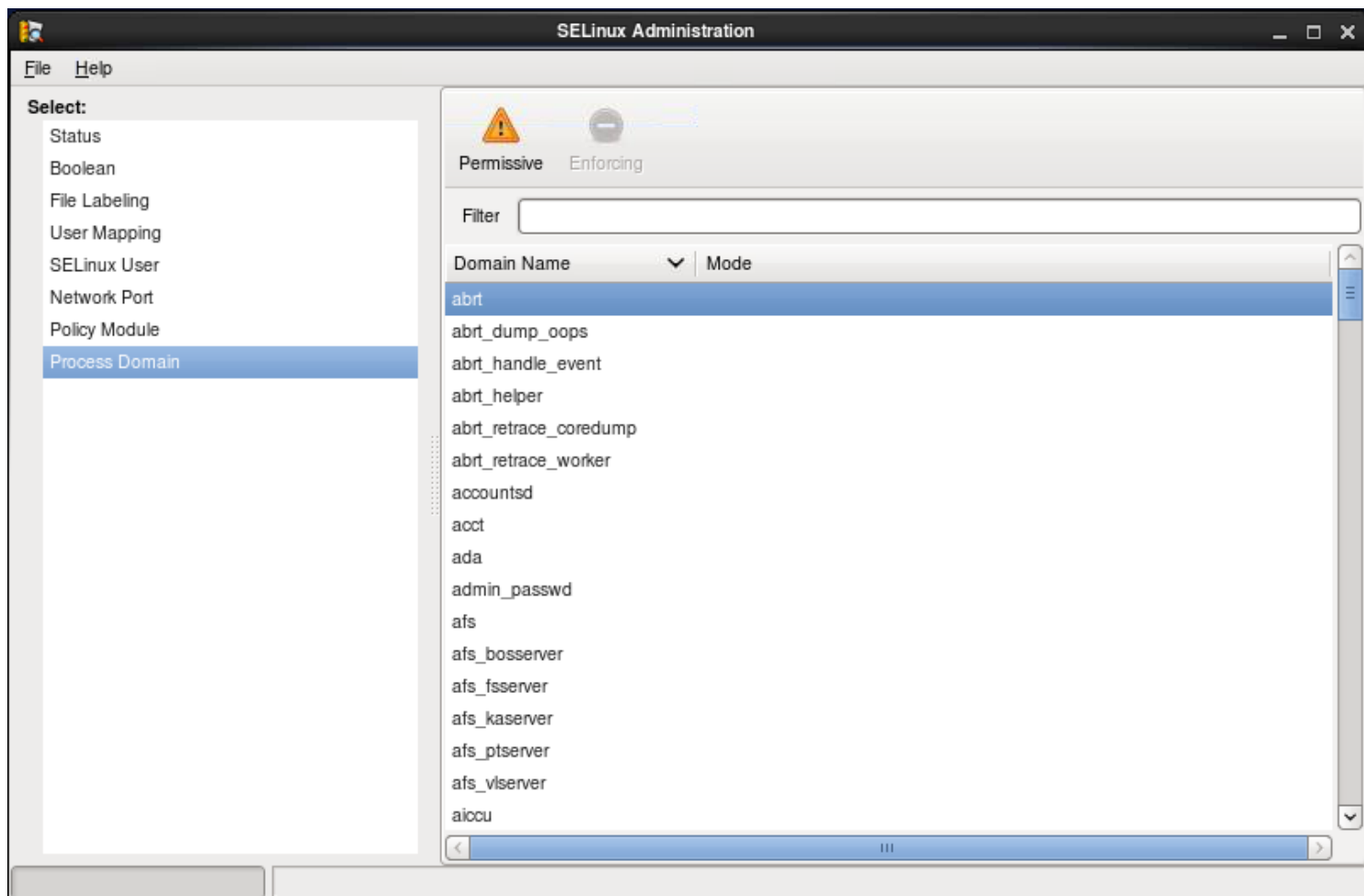












SELinux

- `~]# ls -Z file1`
- `-rwxrw-r-- user1 group1 unconfined_u:object_r:user_home_t:s0 file1`
- SELinux context (контекст) включает:
 - user (`unconfined_u`), SELinux user
 - role (`object_r`), атрибут RBAC, SELinux users авторизованы для ролей, а роли – для доменов
 - type (`user_home_t`), доступ разрешен только, если существует правило в дейст. политике
 - level (`s0` или `s0-s1:c0.c1023`) атрибут MLS и MCS (мин_ур_чув-макс_ур_чув:мин_кат-макс_кат) В targeted политике RedHat используется одна чувствительность `s0`, и конструкция `s0-s0:c0.c1023` авторизует для всех категорий
- Режимы, SELINUX= enforcing, permissive, disabled :
 - Enforcing: отказ в доступе, основанный на правилах политик SELinux
 - Permissive: журналирование `/var/log/audit/audit.log`
 - Disabled: выключен
- Управление режимами
 - `sudo setenforce 1 (0) ; getenforce ; sestatus`
- Политики SELINUXTYPE= targeted, strict
 - именам политик соответствуют директории ниже `/etc/selinux/`
 - Targeted – защищаются только выделенные сетевые демоны
 - Strict – полная защита

Утилиты SELinux

- Задание контекста для всей конкретной файловой системы на этапе монтирования. Например, чтобы обеспечить чтение файлов Apache на NFS-монтированных директориях:

```
mount -t nfs -o context=system_u:object_r:httpd_sys_content_t  
server1.example.com:/shared/scripts/var/www/cgi
```

- Запуск исполняемых файлов в заданном контексте:

```
runcon user_u:system_r:httpd_t ~/bin/contexttest
```

SELinux пользователь

```
~]# semanage login -l
```

LoginName	SELinux_User	MLS/MCS Range
default	unconfined_u	s0-s0:c0.c1023
root	unconfined_u	s0-s0:c0.c1023
system_u	system_u	s0-s0:c0.c1023

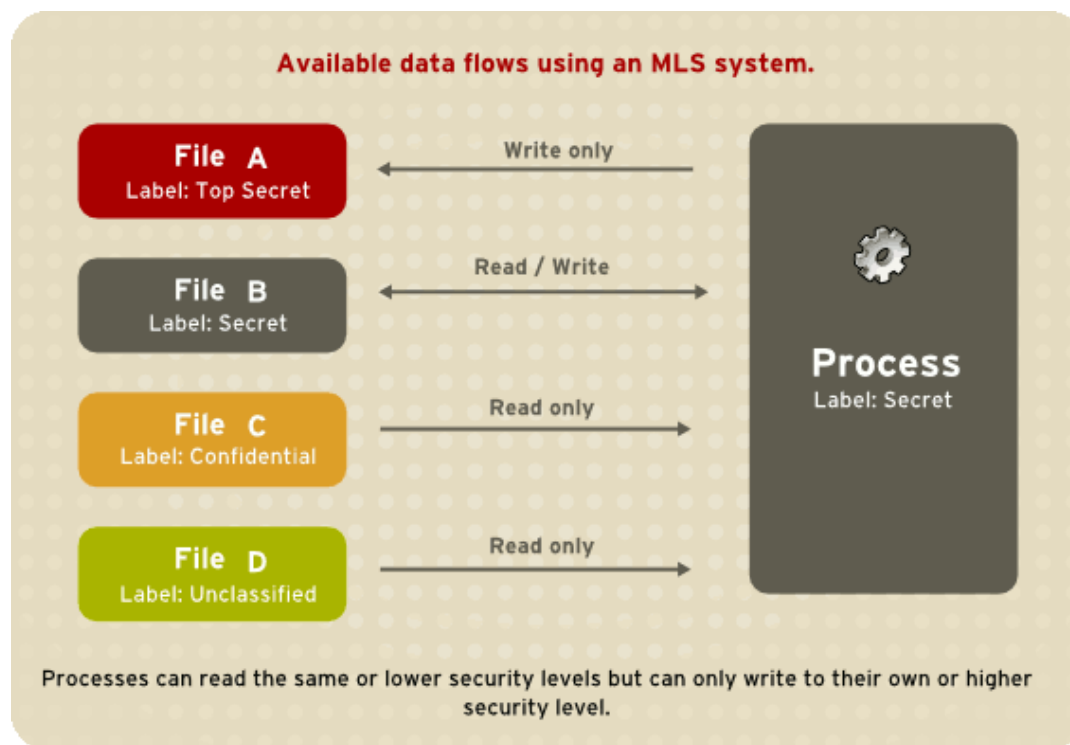
Multi-Level Security (MLS) – использует модель Bell-La Padula MAC

Multi-Category Security (MCS)

Multi-Level Security (MLS)

Multi-Level Security (MLS) – использует модель Bell-La Padula MAC

Требуется установка пакета `selinux-policy-mls` и выбор политики `SELINUXTYPE=mls`



Базовые домены и пользователи в RedHat 6

User	Domain	X Window System	su or sudo	Execute in home directory and /tmp/ (default)	Networking
sysadm_u	sysadm_t	yes	su and sudo	yes	yes
staff_u	staff_t	yes	only sudo	yes	yes
user_u	user_t	yes	no	yes	yes
guest_u	guest_t	no	no	no	yes
xguest_u	xguest_t	yes	no	no	Firefox only

В Red Hat по-умолчанию действует политика Targeted

Согласно ей, помеченные (targeted) процессы выполняются в ограниченном (confined) домене, а непомеченные – в неограниченном (unconfined). Утилиты `ls -Z`, `is -Z`, `ps -Z` дают информацию об этом.

```
~]$ ps -eZ | grep httpd
```

```
unconfined_u:system_r:httpd_t:s0 8883 ? 00:00:00 httpd
```

Утилиты `chcon`, `restorecon`, `semanage` используют для работы с контекстам. Временно :

```
chcon -t unconfined_exec_t /usr/sbin/httpd (-R -рекурсивно)
```

Постоянно: `semanage fcontext -a -t unconfined_exec_t /usr/sbin/httpd`

Директория `/etc/selinux/targeted/contexts/files/` содержит постоянные контексты.

Восстановление: `restorecon -v /usr/sbin/httpd`

Новые файлы и директории наследуют типы от вышестоящих директорий

Пользователи SELinux

- Пользователи:
 - `semanage login -l`
 - `id -Z`
 - `semanage login -a -s user_u olduser`
 - `useradd -Z user_u newuser`

Пример работы SELinux

```
~]$ ls -Z /usr/bin/passwd (файл помечен passwd_exec_t типом)
-rwsr-xr-x root root system_u:object_r:passwd_exec_t:s0
/usr/bin/passwd
```

```
~]$ ls -Z /etc/shadow (файл помечен shadow_t типом)
-r----- root root system_u:object_r:shadow_t:s0 /etc/shadow
```

- Правило политики RedHat разрешает процессам домена `passwd_t` писать/читать файлы типа `shadow_t`.
- Правило политики RedHat задает разрешение `entrypoint` домену `passwd_t` типу `passwd_exec_t`.
- Пользователь запуская приложение `passwd` переводит свой шелл в домен `passwd_t` домен. Требуется прямое правило (иначе по умолчанию отказ) политики (которое есть в RedHat) позволяющее приложениям выполняющимся в домене `passwd_t` доступ к файлам помеченным типом `shadow_t`.

Автовосстановление контекстов файлов

- SELinux поддерживается стартовым скриптом «restorecond», который используется для восстановления контекстов файлов, указанных в `/etc/selinux/restorecond.conf`
- Использует подсистему ядра Inotify для получения уведомлений о событиях файловой системы, например, создание файла.

Barbora Ancincova, Red Hat Enterprise Linux 6 Security-Enhanced Linux, Red Hat Inc. , 2012-2015

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux