

# Развёртывание компьютерного класса

## Сетевая загрузка

### Создание класса на основе терминального сервера

Мы перебираемся к такой большой теме, как развёртывание компьютерного класса. В качестве некоторого отступления от основного направления коснемся вопроса создания класса на основе терминального сервера.

В состав ПСПО входит 4 дистрибутива: Мастер (большой), Юниор (средний), Лайт (маленький, для старых компьютеров) и Терминал-сервер для ситуации, когда одна машина современная, а остальные "никакие". Терминал-сервер поддерживает концепцию организации класса, при которой никакого администрирования на клиентских машинах не производится. Более того, на этих машинах можно даже не иметь винчестера, главное --- научить их загружаться через сеть. Это можно сделать путём сетевой загрузки, используя протокол PXE, а можно использовать загрузочный CD или USB-накопитель, лишь бы это в конечном счете приводило к загрузке операционной системы по сети.

Начало работы обычного компьютера выглядит примерно следующим образом. После загрузки собственно операционной системы на клиентских машинах запускается X-сервер (та часть графической оболочки, которая умеет рисовать на экране и работает с мышью и клавиатурой), а также средства для работы с локальными к флешкам и подобными устройствами. После запуска X-сервера эти машины подключаются по протоколу взаимодействия графических подсистем, который называется XDMPC, к терминальному серверу, и запуск всех приложений происходит на терминальном сервере. Сервер должен быть достаточно хорошей машиной; скорее всего, любая современная машина подойдёт для класса из 10 компьютеров. На одного пользователя в классе обычно хватает 256 Мб памяти. Если вы совсем ограничены в средствах, можно считать необъёмимый объём памяти так: 256 Мб на сам сервер + 128 Мб на каждый терминальный сеанс. Восемь машин при гигабайте памяти на сервере работают достаточно легко. Если шестнадцать машин работают на 2 Гб памяти, то узким местом системы будет уже с быстроедействие дисковой подсистемы. Возможный вариант в этом случае -- машина с двумя винчестерами, которые работают, как один большой (raid 0). Другой вариант -- увеличить объём памяти и, соответственно, дискового кэша, чтобы больше данных находилось в оперативной памяти.

Основное достоинство терминального сервера --- при его использовании локальные машины не нуждаются в администрировании в принципе, они просто включаются и работают. Когда-нибудь в будущем их можно вообще выбросить и купить вместо них не полноценные компьютеры, а так называемые тонкие клиенты, у которых есть монитор (хороший), клавиатура (тоже хорошая) и очень компактный системный блок без винчестера, с минимальной памятью и usb-портами. Администрирование необходимо только на терминальном сервере.

Очевидными недостатками развёртывания класса на основе терминального сервера является остановка работы всего класса при выходе из строя сервера, несколько меньшая "отзывчивость" интерфейса системы, а так же достаточно высокие требования к серверу.

Более подробно о терминальном сервере можно прочитать здесь;

- [Теория использования терминальных серверов;](#)
- [Терминальный сервер со стороны пользователя;](#)

- [📖 Терминальный сервер со стороны администратора.](#)

## Создание класса на основе самостоятельных компьютеров

По ряду причин вариант использования терминального сервере не рассматривается как основной при использовании ПСПО, поэтому мы возвращаемся к ситуации, когда необходимо, чтобы на локальных машинах была установлена операционная система, причем единообразным образом. Для этого можно использовать автоматизированную сетевую установку системы с сервера, где развернут нужный дистрибутив, например, Линукс Мастер.

Что нужно для того, чтобы происходила сетевая установка? Вначале нужно организовать сетевую загрузку машины. Для этого нужно, прежде всего, установить службу dhcpd (Dynamic Host Configuring Protocol Daemon), пакет называется dhcp-server. Он есть в дистрибутиве Мастер, подробнее процесс описан в [📖 Настройка DHCP-сервера.](#)

Dynamic Host Configuring Protocol в переводе с английского --- "протокол динамической настройки компьютера". При включении компьютера по определённым правилам ему по сети передается набор настроек. Вообще говоря, этот протокол очень сложен, он позволяет передавать сотни настроек разного уровня, в том числе и с точки зрения сетевых протоколов. Например, раздача IP может быть организована динамически (каждому новому компьютеру будет выдаваться очередной IP адрес из набора) --- если вас не очень волнует, что в вашу сеть включится кто-то лишний, либо по MAC-адресам (каждому MAC-адресу соответствует свой жёстко определённый IP). Бывают и другие варианты, всё зависит от того, насколько возможно решить задачу административным путём. В любом случае dhcp не помешает в классе, даже если не планируется сетевая загрузка, в особенности, если в учреждении не один сегмент сети.

Конфигурационный файл dhcpd лежит в /etc/dchp/dhcpd.conf:

```
subnet 10.0.2.0 netmask 255.255.255.0 {}

subnet 172.16.0.0 netmask 255.255.255.0 {
    option routers          172.16.0.1;
    option subnet-mask      255.255.255.0;

    option domain-name-servers 172.16.0.1;

    range dynamic-bootp 172.16.0.101 172.16.0.199;
    default-lease-time 21600;
    max-lease-time 43200;
    filename "pxelinux.0";
    next-server 172.16.0.1;
}
```

Этот пример предполагает, что у сервера есть два сетевых интерфейса: внешний в сети 10.0.2.0/24 и внутренний с адресом 172.16.0.1. Предполагается, что на сервере развернут ретранслятор DNS-запросов 172.16.0.1 (подробнее см. [📖 Служба DNS \(Bind\)](#)), если же его нет, следует указать адрес DNS-сервера провайдера. Обратим внимание на две вещи: все сети, к которым подключена машина, должны быть описаны. Дело в том, что dhcpd - капризный демон, и если он видит сеть, которая в нём не описана, он отказывается работать. И второе --- в одной локальной сети должен быть только один DHCP-сервер.

Какие настройки умеет отдавать DHCP-сервер:

- IP-адрес; как уже было сказано, можно настроить либо динамическую выдачу IP-адреса, либо написать "машина такая-то" (MAC-адрес) и указать, какой IP выдавать ему;
- маршрутизатор по умолчанию (поле `option routers`);

- сервер DNS (поле `option domain-name-servers`), который преобразует IP-адреса в доменные имена и обратно;
- маску сети (поле `option subnet-mask`).
- и другие, например имя домена.

Если выпустить из внимания две другие настройки, `filename` и `next-server`, которые нужны именно для сетевой загрузки и будут рассмотрены позже, то машина будет раздавать всем компьютерам в локальной сети адреса из указанного в поле `range dynamic-bootp` диапазона. Любая машина с установленным ПСПО для автоматической настройки сети запускает клиент `dhcp`, который называется `dhcpcd`.

Остается напомнить, что после любых изменений файла `/etc/dhcp/dhcpd.conf` DHCP-сервер следует перезагружать:

```
# service dhcpd restart
```

## Настройка сетевой загрузки

### Настройка клиента

Итак, служба протокола DHCP настроена. Есть ещё две настройки, связанных с сетевой загрузкой операционной системы: во-первых, для ее работы необходима поддержка сетевой загрузки на клиентах, и во-вторых, для нее необходима передача файлов начальной загрузки по протоколу TFTP.

Как правило, о способности клиента загрузиться по сети свидетельствуют слова в настройках BIOS о том, что загрузочным устройством является сеть, int 18 или 19, либо на сетевой карте есть плата BootROM, то есть прошивка для сетевой загрузки, и при включении компьютера с такой сетевой картой вам говорят: "нажмите Alt+B, Shift+F10 или что-то чтобы настроить параметры сетевой загрузки". В этой настройке сетевой загрузки может быть много вариантов, правильно выбирать вариант под названием "PXE". Протокол PXE включает в себя получение настроек по DHCP и некоторые другие возможности.

Клиенты, поддерживающие протокол PXE, должны быть подключены к локальной сети с работающим DHCP-сервером, и на сервере (указанном в поле `next-server`) должен быть специальный демон, называемый TFTP-сервер. TFTP (Trivial File Transfer Protocol) --- это специальный протокол передачи файлов, крайне упрощённый, чтобы его клиентскую часть (программу, которая умеет скачивать файлы согласно этому протоколу) можно было уместить в памяти сетевой карты. Собственно, BootROM и содержит программу, записанная в ПЗУ сетевой карты, которая загружается в оперативную память и занимается ровно одной вещью: по протоколу PXE получает настройки по DHCP/BOOTP, потом сообразно этим настройкам получает информацию о том, откуда брать специальный файл --- загрузчик, скачивает его, загружает в память и запускает. В отличие от ситуации, когда передаются только настройки сети, в протокол PXE входит стадия скачивания некой программы и запуска её. Всё это делает прошивка сетевой карты.

### Настройка TFTP-сервера

Для установки TFTP-сервера требуется поставить соответствующий пакет, разрешить службу и перезагрузить метасервер `xinetd`, через который и работает сервер `tftp`:

```
# apt-get install tftp-server
# chkconfig tftp on
# service xinetd restart
```

На сервере при установке пакета `tftp-server` появляется каталог `/var/lib/tftpboot`, в который нужно поместить файлы, которые будут скачиваться по протоколу `tftp`, как если бы они лежали в корневой директории сервера. Для того, чтобы получить эти файлы, требуется поставить пакет `syslinux`:

```
# apt-get install syslinux
```

`Syslinux` --- пакет загрузчиков, который позволяет загружать Linux с разных носителей, в частности, в нем есть загрузчик `pxelinux` и некоторые другие программы, предназначенные для сетевой загрузки.

Нас интересует содержимое каталога `/usr/lib/syslinux/`:

```
$ ls /usr/lib/syslinux/
chain.c32      dmitest.c32      mboot.c32  pcitest.c32  vesamenu.c32
com32          ethersel.c32     mbr.bin    pxelinux.0
copybs.com     isolinux-debug.bin memdisk     syslinux.com
cpuidtest.c32 isolinux.bin      menu.c32   syslinux.exe
```

Здесь мы видим файлы нескольких форматов: `.bin`, `.com`, `.c32`, `.0`, `.exe`, которые предназначены для загрузки в разные среды. В частности, те, которые загружаются прямо в сетевую карточку называются `.0` и `.c32`.

Насбудет в первую очередь интересоваться загрузчик `pxelinux.0`. В терминологии последовательности загрузки это вторичный загрузчик, то есть тот, который должен загружаться не из пзу, а непосредственно с устройства. Основная его задача --- подгрузить откуда-то ядро и стартовый виртуальный диск (`initrd`). Загрузка этих двух файлов --- это то, с чего начинается загрузка Linux. Ядро --- это ядро, а в стартовом виртуальном диске находится очень маленький линукс, который содержит все модули, нужные для дальнейшей загрузки и работы системы. Скопируем его в каталог TFTP:

```
# cp /usr/lib/syslinux/pxelinux.0 /var/lib/tftpboot/
```

Кроме того, нам необходим сам загрузочный CD или DVD-диск. Для его раздачи на сервере нужно настроить NFS, как описано в [Установка и настройка FTP-сервера и NFS-сервера](#). Скопируем содержимое диска в каталог `/srv/boot` сервера, а каталог `isolinux` (в нем, в частности, находится ядро и образ виртуального диска начальной загрузки) --- в каталог `tftpboot`:

```
# cp -a /media/cdrom/* /srv/boot
# cp -a /media/cdrom/isolinux /var/lib/tftpboot/
```

У `pxelinux` есть конфигурационный файл, в котором написано, что же ему загружать дальше. В нём можно сделать несколько разных вариантов выбора. Для удобства работы с сетью `pxelinux.0` загружает конфигурационный файл по определённым правилам. Сначала `pxelinux` пытается загрузить конфигурационный файл, который по имени совпадает с шестнадцатеричным представлением полученного по PXE IP-адреса. Это сделано для того, чтобы при жёсткой привязки IP-адреса к MAC-адресу машины для каждой машины можно было бы иметь собственную настройку, и можно машины с одними IP-адресами загружать по одной конфигурации, а другие, из другой сети, по другой конфигурации.

Мы рассмотрим простейшую конфигурацию, общую для всех компьютеров сети. Для этого нужно создать файл `/var/lib/tftpboot/pxelinux.cfg/default` следующего содержания:

```
timeout 0
prompt 1
default install
```

```
label install
kernel ../isolinux/alt0/vmlinuz
append initrd=../isolinux/alt0/full.cz lang=ru_RU
automatic=method:nfs,network:dhcp,server:172.168.0.1,directory:/srv/boot ai live
stagename=altinst
# Все, что после "append " - должно быть записано в одну строчку
```

Рассмотрим синтаксис этого файла. Поле **prompt** означает, ожидать ли подсказки (**boot:**) при загрузке, поле **timeout 0** означает ожидание, пока пользователь не нажмёт enter при загрузке в ответ на приглашение **boot:**. Тут есть один момент: если включён [NumLock](#), **pxelinux** будет ждать, а если выключен --- нет. В поле **default** указано имя варианта загрузки по-умолчанию (в нашем файле конфигурации этот вариант единственный)

Дальше идут настройки для конкретной загрузки. Поле **label** задает имя варианта загрузки. Файл конфигурации может содержать несколько вариантов загрузки, и имя желаемого варианта можно указать в ответ на подсказку ("**boot:**"). Дальше нужно указать, где находится ядро (поле **kernel**) и какие у этого ядра параметры (поле **append**). Параметр ядра **initrd** разбирается загрузчиком **syslinux**, он загружает этот файл (стартовый виртуальный диск), а остальные параметры, которые он не понимает, он передаёт ядру.

В параметре **automatic** указан способ получения файлов установки (здесь NFS, так же возможен FTP), адрес сервера (172.168.0.1) и путь к файлам (/srv/boot).

В результате проделанной работы будет происходить примерно следующее.

- При загрузке клиентской машины она получает IP-адрес по DHCP, и загружает начальный загрузчик **pxelinux.0** (указанный в параметрах DHCP) по протоколу TFTP.
- Загрузчик ищет подходящий файл конфигурации на TFTP-сервере. В результате анализа файла конфигурации загрузчик узнает, где находится ядро Линукс и образ начального виртуального диска (*initrd*), а также получает остальные параметры для ядра.
- Загрузчик загружает по TFTP ядро и образ виртуального диска, и передает управление ядру.
- Ядро после инициализации запускает процесс установки, используя уже NFS для доступа к установочному диску.

# Теория использования терминальных серверов

## Тайное знание

В X.org есть универсальный драйвер для устройств ввода evdev, который можно использовать, в том числе, для клавиатуры и мыши. Единственный недостаток evdev --- ему надо сказать, какой конкретно device-файл использовать, в отличие от mouse, для которого весь ввод сводится в device-файл /dev/mice. Так вот, для указаний девайса есть специальные симлинки в каталогах /dev/input/by-id и /dev/input/by-path.

Посмотрим, как эта проблема решается, пока что вручную, но более или менее однозначно. При этом по ссылке с именем --- текстовым идентификатором в /dev/input/by-id доступно само устройство, поэтому при подключении мыши к любому порту она будет работать.

```
Section "InputDevice"
    Identifier "Mouse using evdev" # Произвольный идентификатор
    Driver      "evdev"
    Option      "Device" "/dev/input/by-id/usb-A4Tech_Ps.2+USB_Mouse-event-
mouse" # Имя устройства, в /dev/input/by-id/ или в /dev/input/by-path/
EndSection
```

## Linux Terminal Server

Мы рассмотрим, как устроен дистрибутив Linux Terminal Server из комплекта ПСПО (в дальнейшем LTSP, Linux Terminal Server Project).

Этот дистрибутив был разработан дополнительно к госзаказу (в рамках которого изготовлены Лёгкий Линукс, Линукс Юниор и Линукс Мастер). Над ним работала принципиально другая команда, из Киева, фактически ресурсы (деньги и, что более важно, людей) пришлось делить на большее количество работы. Дело в том, что команда, которая его разрабатывала, занималась внедрением подобных решений на разных предприятиях, и у них уже были наработки, пригодные для внедрения, правда, только авторами и только в благоприятной среде. Они воспользовались открывшимся ресурсом, чтобы доработать все это до дистрибутива, и, надо сказать, работы было много.

Что он себя представляет?

В документации по ПСПО есть статья Георгия Курячего (["Linux-класс за час, или X-терминалы, тонкие и ленивые"](#)), довольно старая, еще для ALT Linux Compact 3.0.

В ней рассказывается, что для того, чтобы попробовать использовать Linux, совершенно необязательно устанавливать его на компьютер. Самый простой способ сделать это --- загрузить Live CD. Суть Live CD очень проста --- на компакт-диске находится готовая к эксплуатации ОС, которая при загрузке с этого диска размещается частично в оперативной памяти, а основные ресурсы находятся на компакт-диске.

При начале работы с Live CD обычно создаётся очень небольшой виртуальный диск в оперативной памяти, доступный на запись, и второй виртуальный диск, большего размера и доступный только на чтение, состоящий из файлов, находящихся на компакт-диске. Работа с Live CD неудобна тем, что он довольно долго загружается и довольно медленно работает (скорость чтения данных с CD на один---два порядка ниже скорости чтения данных с жесткого диска, а скорость случайного позиционирования у CD ниже на три порядка). Это большой недостаток в случае полноценного использования такой операционной системы, с запуском большого количества программ. Существуют технологии оптимального

размещения файлов на компакт-диске для ускорения работы такой системы, но значительного ускорения всё равно не происходит.

В настоящее время, по сравнению со статьей, которая была написана 2 года назад, существует чуть более правильный вариант --- тот же самый Live CD, только записанный на flash-носитель. По сравнению с Live CD существенно возрастает скорость чтения и исчезают затраты на позиционирование, т.к. flash-носитель --- это устройство прямого доступа (отсутствует понятие позиционирования). Кроме того, его можно так разметить, что часть его будет содержать основные файлы системы, как Live CD, а часть будет доступна для записи пользовательских данных. Такой вариант называется по аналогии --- Live Flash.

Недостаток Live Flash в том, что flash-носители быстро изнашиваются, особенно ограничен их ресурс на перезапись. В ПСПО используется простейший способ построения Live Flash: образ существующего Live CD просто записывается на flash-носитель.

У обеих этих технологий (Live CD и Live Flash) на сегодняшний день есть существенный недостаток: содержимое системной области недоступно на запись, т.е. нельзя изменить состав установленных программ, и т.п. Теоретически, с помощью unionfs можно сконструировать файловую систему, которая будет поддерживать запись в системный раздел, но это приводит к другим проблемам. Одним словом, сбалансированного дистрибутива на базе Live Flash нет.

Ещё одна проблема в том, что существующий парк компьютерной техники может очень различаться по характеристикам и подходить не для всех видов программ и работ. Можно представить себе старый компьютер, на котором Live CD будет работать неприемлемо медленно и, возможно, нестабильно. Часто встречаются именно такие компьютеры, которые, с одной стороны, желательно задействовать в полезной работе, с другой стороны, даже при установке Linux на них, аппаратных ресурсов явно недостаточно для полезной работы. Такие компьютеры мы отнесём к классу машин, на которых нам желательно использовать Linux без установки его на жёсткий диск.

Хотя Live CD и Live Flash и позволяют не устанавливать Linux на жесткий диск, они рассчитаны на работу с достаточно мощными компьютерами, и на старых машинах от них будет немного пользы.

У этой проблемы есть решение, но прежде рассмотрим другую технологию, которая даёт хорошее решение в некоторых других случаях.

## **Толстый клиент**

Идея состоит в следующем. Допустим, у нас есть достаточно новый компьютер, на который Linux можно и поставить, но нам не хочется нам этого делать, например, потому что там стоит купленная вместе с компьютером операционная система, от повреждения которой человек лишается гарантии, или за машиной работают другие пользователи, которые не захотят менять операционную систему. Либо еще одна ситуация: у нас есть операционная система, но компьютеров у нас 20 штук. И каждый раз, когда нам приходит в голову установить что-то или изменить настройку, необходимо будет менять что-то на всех компьютерах по отдельности, а это неудобно.

Было бы неплохо заставить все эти машины каким-то образом работать с одним и тем же системным пространством, условно говоря, диском. Диск им дать один на всех, какие-то мелкие различия, которые существуют, хранить можно и локально. В этом случае администрирование будет намного упрощено, мы просто будем содержимое общего диска модифицировать, в результате работа всех этих компьютеров будет как-то изменяться. Например, мы можем установить пакет на этот диск, и любой из 20 компьютеров загрузится с использованием этого диска как системного и будет использовать этот пакет.

На эту тему тоже есть статья Георгия Курячего, которая была в докладе на тему толстых клиентов на конференции в Переславле (<http://heap.altlinux.org/pereslav12006/kouryachy/abstract.html>).

В случае, когда мы не хотим устанавливать линукс на компьютере, но хотим его использовать, у нас есть следующие решения:

наши компьютеры	локальные решения	linux-сетевые linux-решения
новые	livescd или liveflash	толстые клиенты с загрузкой всего и вся на них по сети
старые	?	тонкие клиенты

Вначале рассмотрим решения для новых компьютеров. Мы могли бы использовать livescd или liveflash, но есть дополнительные ограничения:

- мы не хотим мириться с тем, что livescd медленно работает и трудно организовать записываемую часть
- мы хотим изменять системную часть централизованно для многих компьютеров

Тогда можно поступить так. Жесткий диск, как в случае с livescd или liveflash, не используется вообще, а используется технология загрузки по сети. Есть специальные методы загрузить какую-то небольшую программу через сеть с сервера, которая загрузит большую программу, которая уже делает что-то полезное, например, ядро linux или initrd, который примонтирует сетевой диск.

После чего в случае толстого клиента будет происходить вот что: загруженный по сети специально подготовленный линукс будет рассчитывать, что та часть, которая раньше лежала по livescd, теперь находится на специальном файловом сервере в сети. То есть, отличие live-технологий от толстого клиента минимально, и состоит оно в том, как соединяется компьютер с основным корпусом программного обеспечения. В случае livescd, этот лазерный диск, с которого происходит загрузка компьютера, а в случае толстого терминального клиента это специальный файловый сервер, прописанный в настройках нашего компьютера --- терминального клиента.

Причём здесь возможна следующая ситуация: содержимое livescd перекладывается на сервер, а потом монтируется так же, как монтировалось бы при загрузке с него локально, с той лишь разницей, что во втором случае он будет смонтирован через NFS или другую распределенную файловую систему. Ещё один вариант, если у вас очень много памяти, можно положить образ на http. Клиент его скачает и будет хранить в оперативной памяти. Недостаток (помимо требуемого объема памяти) --- много качать при каждой загрузке. Естественно, при запуске программ с таким образом организованного толстого сервера новые копии их машинного кода в памяти не создаются.

Получается в результате компьютер, жёсткий диск которого вообще не используется, процедура загрузки сетевая, весь корпус программного обеспечения в виде сетевого носителя подключается с специально подготовленного сервера, как и вся процедура загрузки. Машина будет потреблять больше ресурсов, чем если бы Linux был установлен на ее жесткий диск, тем более, если вообще не использовать жесткий диск, то нельзя использовать swarp (впрочем, swarp на сетевом диске иногда используется, но это ненадежно).

Если мы скачали образ с сервера и разместили его целиком в оперативной памяти, то это, несмотря на то, что использует много памяти, дает также и преимущества:

- после загрузки системы сетевая активность минимальна. Раздел загрузки на сервере

- можно отмонтировать, сам сервер выключить, и т.д.
- работает быстрее, чем если бы все было установлено на жестком диске

При такой организации наш компьютер называется толстым терминальным клиентом. Обратите внимание, что это просто продолжение технологий `livedcd`, просто вынесенное в сеть.

Теперь перейдем к ситуации, когда компьютеры старые, и установка Linux на них хоть и возможна, но время установки и загрузки очень большое, да и крупные приложения работать будут очень медленно, засчёт слабого процессора и малого объема оперативной памяти.

В этом случае можно вспомнить, что графическая подсистема X имеет сетевую структуру --- в ней программа, которая отвечает непосредственно за ввод с клавиатуры, мыши, других устройств и вывод графики (X-сервер), отделена от программы, обрабатывающей введенную информацию и генерирующей изображения, которые нужно выводить (X-клиент). Процесс взаимодействия X-клиента с X-сервером может происходить не только локально, но и по сети.

Перенеся работу большинства программ с терминального клиента на терминальный сервер, мы получим вместо толстого тонкий терминальный клиент. Приведем таблицу, в которой указано, где идет работа с устройствами ввода-вывода, где работают программы (задачи), где хранятся файлы.

<b>клиент</b>	<b>ввод-вывод</b>	<b>задачи</b>	<b>файлы</b>
---------------	-------------------	---------------	--------------

<i>толстый</i>	станция	станция	сервер
----------------	---------	---------	--------

<i>тонкий</i>	станция	сервер	сервер
---------------	---------	--------	--------

При этом следует учитывать разницу в терминологии и помнить --- на терминальном сервере у нас работают программы --- X-клиенты, а на тонком терминальном клиенте запущен X-сервер.

Для построения такой системы используется та же программа, что и для локального графического входа в систему --- менеджер дисплеев X (`X display manager`), например `kdm`, входящий в состав окружения рабочего стола KDE. Происходит взаимодействие по следующей схеме:

### **Локальный вход**

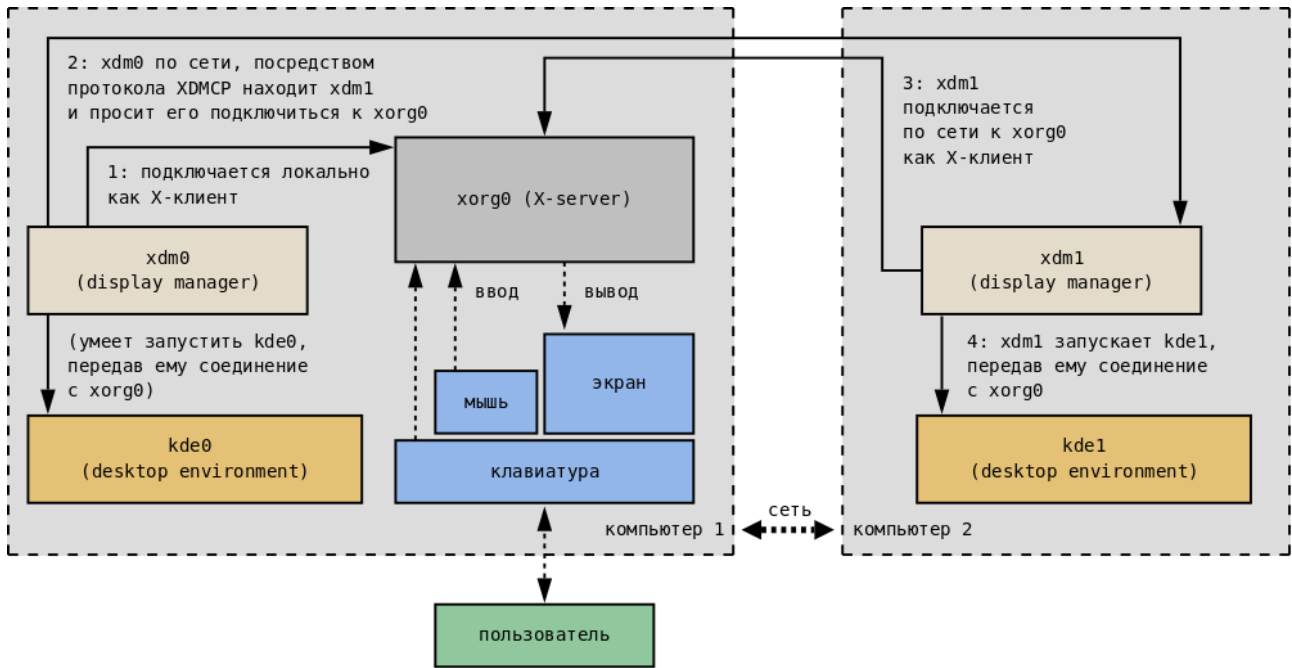
На компьютере 1 запущен X-сервер `xorg0`, работающий с мышкой и экраном, за которым сидит пользователь. К `xorg0` как X-клиент подключен `display manager xdm0`, который может после авторизации пользователя запустить любое приложение, обычно некоторый менеджер окон или целое окружение рабочего стола, например, `kde0`, и передать ему свое соединение с `xorg0`.

### **Удаленный вход**

Но пользователь может попросить `xdm0` запросить сеть и найти там другой, проанонсированный `display manager xdm1`, находящийся на компьютере 2, чтобы `xdm0` передал своё соединение с `xorg0` ему. X-протокол позволяет сделать это по сети. Затем `xdm1` может точно так же запустить `kde1`, уже на компьютере 2, и пользователь сможет с ним удалённо работать.

При такой организации на терминальном клиенте работает две программы: X-сервер `Xorg` и `display manager`. Остальные программы, которые выполняют пользовательские задачи,

выполняются удаленно на терминальном сервере.



## История проекта

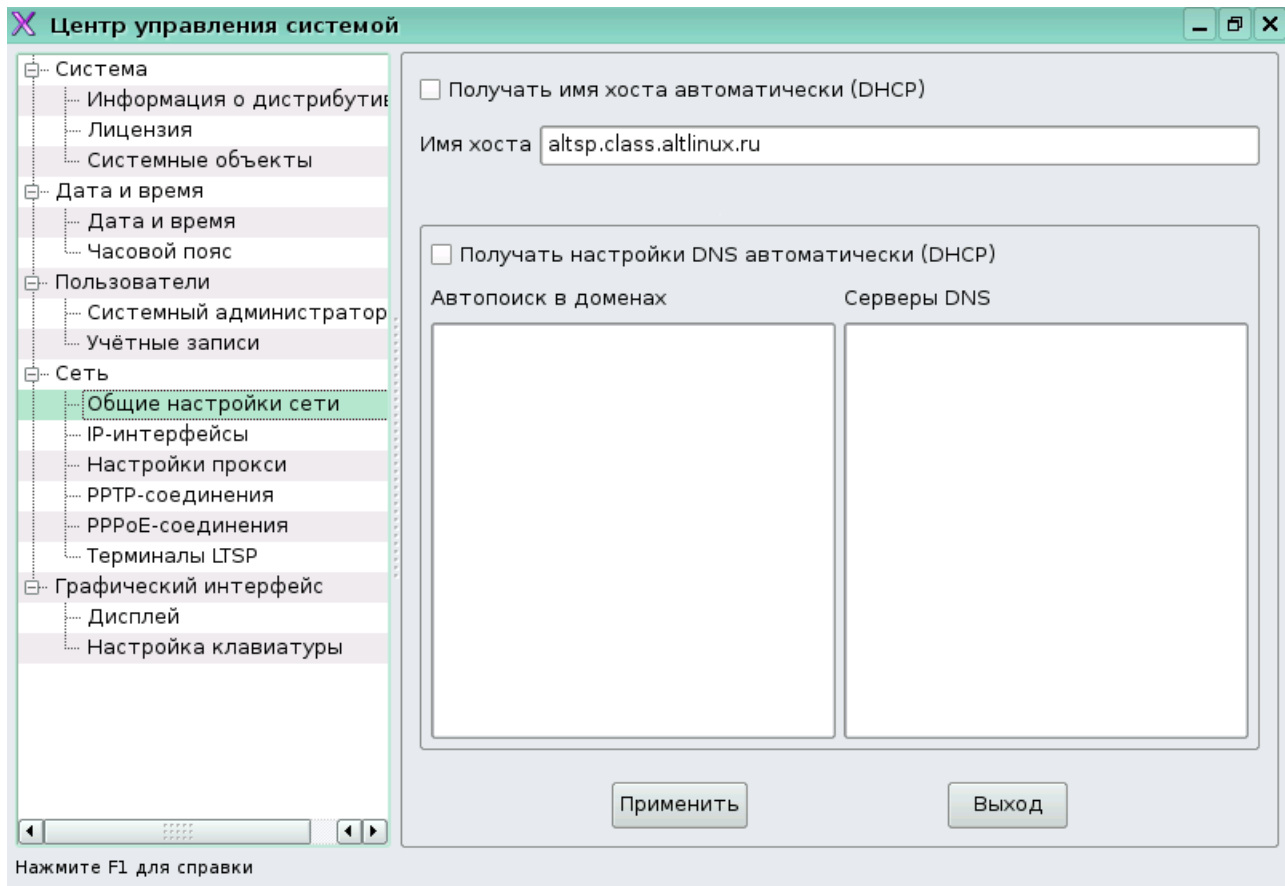
Есть такой довольно давний проект --- LTSP (Linux Terminal Server Project. Без буквы А в начале). У него была довольно активная команда разработчиков, не то в Иркутске, не то в Новосибирске. Суть проекта состояла в том чтобы создать дистрибутив, использующий такую технологию (см. диаграмму), чтобы не нужно было руками все настраивать эти сервисы. Начинался этот LTSP ещё на ядре 2.2, то есть это было давно. Тогда задача могла быть решена только единственным способом: с одной стороны, нужно было собирать два разных ядра, одно для сервера, другое для клиента, причем для клиента оно должно было быть сильно изменено, чтобы происходило автоматическое определение железа и потом эту информацию об этом железе можно было передать на сервер. Еще нужно было организовать достаточно развесистую инфраструктуру по приёму этой информации, адаптации соответствующих настроечных файлов, и т.д.

Со временем выяснилось, что сильное изменение и перекомпиляция ядра не обязательны.

# Терминальный сервер со стороны администратора

Рассмотрим теперь особенности терминального сервера с точки зрения администратора.

Заметим, что на терминальном сервере основной конфигуратор системы --- Alterator --- претерпел некоторые изменения (ср. [Настройка с использованием Центра управления](#)), в частности, расширился раздел настройки сети.



Для правильного функционирования терминальной сети необходимо настроить DHCP для загрузки тонких клиентов (напомним, что тонким клиентом называют компьютер-клиент сети с терминальной архитектурой, в которой все задачи по обработке информации перенесены на сервер). Настройки DHCP-сервера находятся в файле `/etc/DHCP/dhcpd.conf`:

```
ddns-update-style interim;
ignore client-updates;
allow booting;
allow bootp;

option option-128 code 128 = string;
option option-129 code 129 = string;

use-host-decl-names on;

next-server 192.168.0.1;

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.20 192.168.0.250;
    option domain-name "example.com";
    option domain-name-servers 192.168.0.1;
    option broadcast-address 192.168.0.255;
```

```

option routers 192.168.0.1;
option subnet-mask 255.255.255.0;
option root-path "192.168.0.1:/var/lib/ltsp/i586";
if substring( option vendor-class-identifier, 0, 9 ) = "PXEClient" {
    filename "/ltsp/i586/pxelinux.0";
} else if substring( option vendor-class-identifier, 0, 9 ) = "Etherboot" {
    filename "/ltsp/i586/nbi.img";
    #filename "/ltsp/i586/pxelinux.0";
} else {
    option-129 = " initrd=/ltsp/i586/initrd.img";
    filename "/ltsp/i586/vmlinuz";
}
}

subnet 10.0.2.0 netmask 255.255.255.0 {}

```

Здесь объявлена внешняя сеть и присутствует большой блок ее описания. Из всех параметров отметим несколько особенностей:

- Обратите внимание, что здесь присутствует адрес next-server'a (это TFTP-сервер, с которого клиенты получают загрузочные файлы; он может не совпадать с DHCP-сервером)
- Посмотрим на if. Мы знаем, что в процессе загрузки по PXE клиент получает несколько настроек, в числе которых IP, маршрутизация, DNS. Кроме того, предоставляется имя файла с загрузчиком на сервере TFTP. Параметр filename встречается трижды: формат конфигурации DHCP довольно сложный, и в ней могут встречаться условные операторы. В данном случае, условный оператор выбирает разные загрузочные файлы в зависимости от клиента.
- При сетевой загрузке можно реализовать подключение сетевых дисков по протоколу NFS. По сети подключается некоторый каталог на заданном сервере, который передаётся параметром option root\_path. Указанный каталог подключается клиентом как корневая файловая система.
- На самом деле, на этом не заканчиваются те настройки, которые можно передать по DHCP.

Если у вас не работает загрузка по сети (ваша сетевая карта не поддерживает PXE), то необходимо воспользоваться [rom-o-matic](#) (он же Etherboot). Он позволяет изготовить bootrom для сетевой карты. Это значит, что для конкретной карты можно сгенерировать образ, который, будучи загруженным при включении компьютера с какого-нибудь носителя (или даже из BIOS), инициирует сетевую загрузку клиента.

Перейдем к настройке TFTP. Он запускается по умолчанию. Всё, что можно сделать при помощи TFTP --- скачать определённый файл. Если работает PXE, то скачивается `pxelinux.0`. Это загрузчик, он является частью `syslinux`. После того, как он загружается на рабочей машине, `pxelinux.0` совершает некоторые действия. В первую очередь, он скачивает настроечный файл, отобранный следующим образом: сначала он перебирает настроечные файлы, соответствующие IP, если таковых не находится, полному MAC-адресу, далее --- его частям (при необходимости разных сценариев загрузки для разных машин или групп машин). В директории `/var/lib/tftpboot/ltsp/i586/` лежат все файлы, которые могут быть переданы по TFTP:

```

[root@altsp i586]# ls
System.map-2.6.22-led-tc-alt11  nbi.img-2.6.22-led-tc-alt11
config-2.6.22-led-tc-alt11      pxelinux.0
initrd-2.6.22-led-tc-alt11.img  pxelinux.cfg
initrd.img                      vmlinuz
nbi-2.6.22-led-tc-alt11.img     vmlinuz-2.6.22-led-tc-alt11
nbi.img

```

`pxelinux.cfg/default` --- настроечный файл, использующийся по умолчанию. Он выглядит так:

```
DEFAULT          vmlinuz          ro          initrd=initrd.img          root=/dev/nfs
nfsroot=/var/lib/ltsp/i586,udp ip=dhcp
```

Синтаксис здесь такой же, как и везде в `syslinux`. Мы видим, что ядро специфическое, используется специфический `root`. Отметим, что параметр `nfsroot` указывает, где находится корневая ФС.

(В принципе, вовсе не обязательно компилировать NFS-клиент в ядро, возможно сделать аналогичный настроечный файл с обычным ядром.)

Итак, после выполнения загрузчика (`pxelinux.0`) запускается ядро, `initrd` (временная файловая система), подключается корневая файловая система по сети и оттуда происходит дальнейший запуск клиента.

Дальше происходит автоматическое обновление DNS. В качестве DNS-сервера используется `avahi-daemon`. Вообще, `Avahi` --- реализация протокола `Zeroconf`, среди возможностей которого --- работа с широковебательным DNS и обнаружение DNS-служб в сети.

Наконец, рассмотрим NFS, `network file system`, протокол сетевой файловой системы. Сервис, который в терминальном сервере обеспечивает NFS, называется `unfsd`, который представляет из себя урезанную версию обычного `nfsd`. На NFS находятся корневая файловая система для клиентов. В GNU/Linux для сетевых томов NFS используется по умолчанию. Отличительные особенности NFS:

- Во-первых, он реализован поверх UDP с соответствующими ограничениями UDP. То есть, NFS-клиент и NFS-сервер обмениваются друг с другом датаграммами, и тот факт, что операция записи не прошла или что-то случилось, отслеживается на уровне прикладном, а не транспортном. Стоит отметить, что NFS является идиолатентной, то есть несколько одинаковых действий выполняются как одно. Единственная трудно решаемая проблема --- проблема блокировки (файла на запись, например). Раньше блокировок вообще не было, потом появился `nfslockd`.
- Во-вторых, уровень доверия в NFS вынесен на уровень IP. То есть машина с некоторым IP-адресом либо имеет доступ к NFS, либо нет. Так было в NFS версий 2 и 3, так же и в NFS 4. Другое дело, что в случае тонких клиентов мы предоставляем файловую систему любому клиенту, но только на чтение, то есть проблем с блокировкой нет. Настройка NFS-сервера `unfsd` в таком случае довольно проста:

```
/var/lib/ltsp/i586          (ro,no_root_squash,async)
```

Слева указан путь к открываемому на доступ каталогу, справа --- список хостов и параметры в скобках через запятую. Если списка хостов нет, то это означает доступ для всех. Здесь под хостом подразумевается пользователь `root` на хосте. Поясним некоторые из параметров:

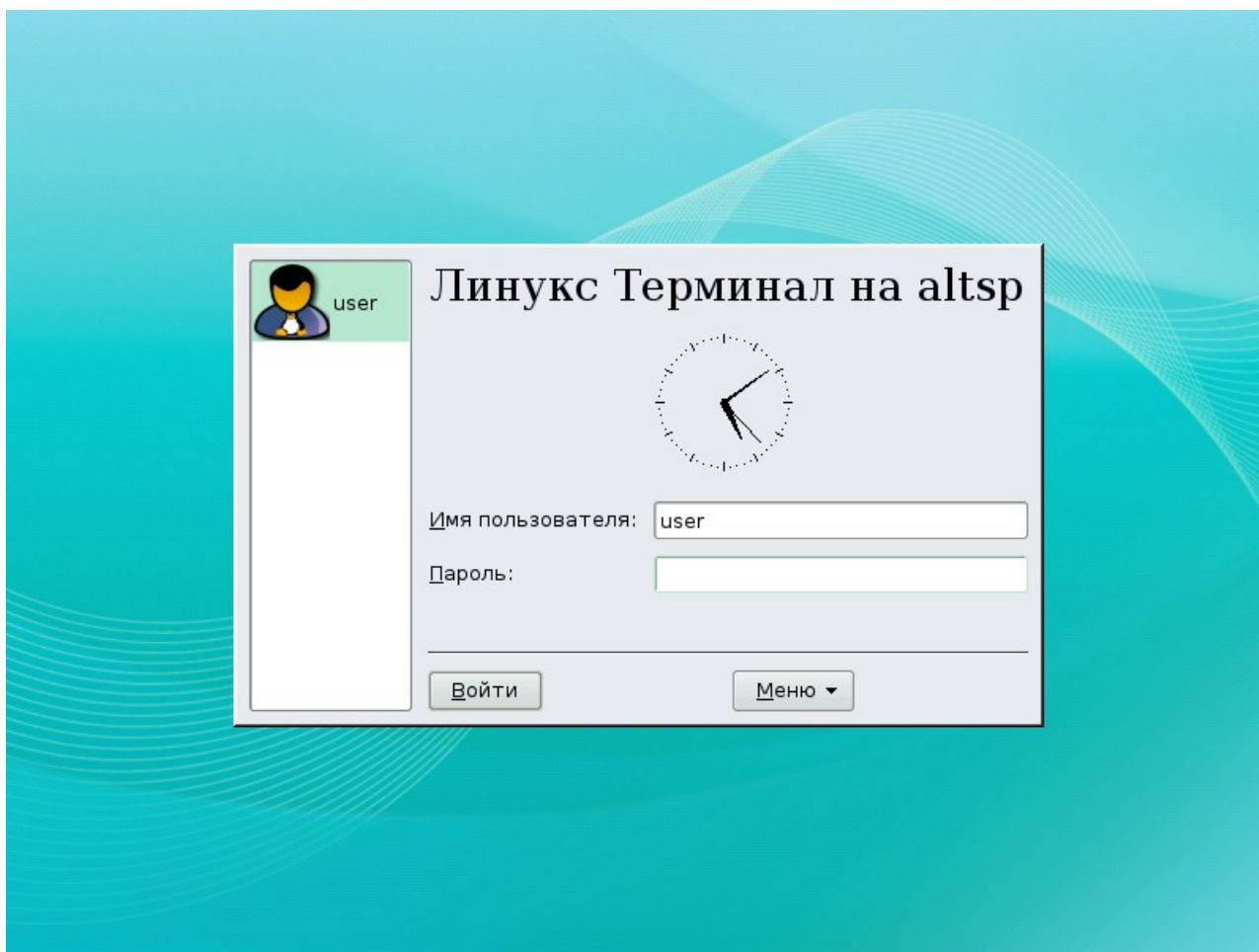
- `ro` --- `readonly`,
- `no_root_squash` --- отключение `squashing'a`, то есть обработки запросов от клиентов с `UID=0` как запросов от пользователя `nobody`. Но, опять же, имеются в виду запросы только на чтение.

## Терминальный сервер со стороны пользователя

Со временем, после появления версии 2.6 ядра Linux и виртуальной файловой системы sysfs, политика по оформлению и регистрации драйверов и аппаратного обеспечения компьютера изменилась и необходимость в сборке отдельного специфического ядра для терминальных клиентов отпала. LiveCD в ПСПО, к примеру, изготавливается из самого обычного ядра с обычными модулями. Проект же LTSP был, к сожалению, сильно привязан к уже разработанной инфраструктуре и системе патчей, а потому непосредственная интеграция его в обычный дистрибутив не представлялась возможной. Сборка LTSP в рамках ПСПО, к примеру, была большим bundle-файлом. Он выкладывался в определенное место, после чего целая коллекция скриптов настраивала нужные сервисы, --- только так это работало. Поэтому был создан LTSP --- Linux Terminal Server Project. Его разработчики скопировали часть инфраструктуры LTSP, избежав при этом необходимости собирать специальное ядро. Результатом работы LTSP стал специализированный дистрибутив "Линукс Терминал", установка которого практически ничем не отличается от установки дистрибутива "Линукс Мастер", за двумя исключениями:

- отсутствует раздел установщика "Разметка диска": дистрибутив требует для работы выделенный сервер, а потому на соответствующей стадии установки требуется лишь подтвердить передачу всего жесткого диска в распоряжение установщика;
- дистрибутив требует наличия в компьютере двух сетевых карт, причем Интернет должен быть доступен через устройство eth0, а локальная сеть (класс, машины в котором будут загружаться по сети) --- через eth1; устройство eth0 при этом должно получать IP-адрес статически (в случае DHCP создается специальная "обертка", позволяющая загрузиться со статическими настройками и затем перенастроиться на использование DHCP), а IP-адрес для eth1 должен быть 192.168.0.1.

Других видимых различий в процедуре установки "Мастера" и "Терминала" нет. Заметим, что пакетный состав дистрибутивов, естественно, различается. Приглашение XDM (в роли которого выступает kdm) на установленном "Линукс Терминале" выглядит так:



Организуем теперь запуск клиентских машин (рабочих станций, или машин-терминалов). В первую очередь необходимо убедиться, что клиенты и сервер находятся в общей СПД. Во вторую --- настроить на клиентах сетевую загрузку (желательно --- по умолчанию). Заметим, что способов загрузить машину по сети есть несколько. Нужный нам вариант называется PXE. Поддерживающие PXE сетевые карты могут получать настройки по DHCP (используется протокол BOOTP) и использовать протокол TFTP для скачивания первичного загрузчика. Во время загрузки по сети экран рабочей станции может выглядеть, например, так:

```

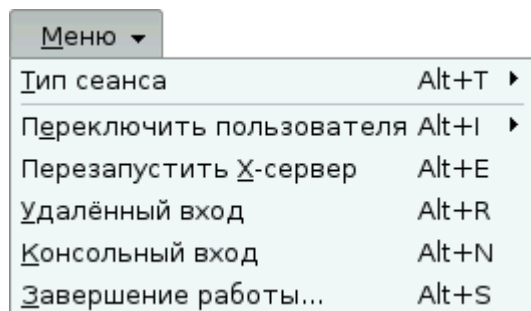
MAC: 08:00:27:6A:75:21 UUID: 56424f58-0000-0000-0000-0800276a7521
Searching for server (DHCP)....
Me: 192.168.0.250, DHCP: 192.168.0.1, Gateway 192.168.0.1
Loading 192.168.0.1:/ltsp/i586/pxelinux.0 ...(PXE).....done

PXELINUX 3.36 0x46c1da28 Copyright (C) 1994-2007 H. Peter Anvin
UNDI data segment at: 0009E000
UNDI data segment size: 1000
UNDI code segment at: 0009F000
UNDI code segment size: 0B1D
PXE entry point found (we hope) at 9F00:0680
My IP address seems to be COA800FA 192.168.0.250
ip=192.168.0.250:192.168.0.1:192.168.0.1:255.255.255.0
TFTP prefix: /ltsp/i586/
Trying to load: pxelinux.cfg/01-08-00-27-6a-75-21
Trying to load: pxelinux.cfg/COA800FA
Trying to load: pxelinux.cfg/COA800F
Trying to load: pxelinux.cfg/COA800
Trying to load: pxelinux.cfg/COA80
Trying to load: pxelinux.cfg/COA8
Trying to load: pxelinux.cfg/COA
Trying to load: pxelinux.cfg/CO
Trying to load: pxelinux.cfg/C
Trying to load: pxelinux.cfg/default
Loading vmlinuz.....

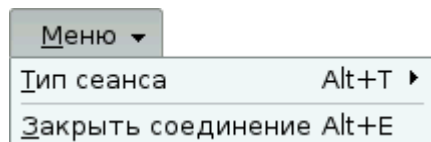
```

Если загрузка прошла успешно, то на экране мы увидим приглашение XDM, ничем с первого взгляда не отличающееся от приглашения XDM на сервере. Однако, как мы понимаем, это результат взаимодействия программы xdm на сервере и программы X (X-сервера) на клиентской машине. Удаленный XDM знает о том, что к нему подключились по сети, и потому предложит другой состав меню.

На машине-сервере:



На машине-клиенте:



Итак, терминальный класс успешно функционирует. При этом, однако, мы должны решить следующие проблемы:

- разделение ресурсов;
- доступ к ресурсам.

Рассмотрим их подробнее.

## Проблема разделения ресурсов

Эта проблема касается разделения ресурсов между пользователями. Если в нашем компьютерном классе используются 4 клиентских и одна серверная машина, то 4

одновременно работающих на одном сервере пользователя запускают, соответственно, 4 комплекта программного обеспечения. Если сервер по-прежнему один, а клиентов 24, то комплектов запущенного ПО будет тоже 24. Когда пользователи запускают N комплектов ПО, то они потребляют почти в N раз больше оперативной памяти, используют почти в N раз больше промежутков процессорного времени и генерируют N потоков обмена данными с дисковой подсистемой (по сравнению с одним комплектом ПО).

Практика эксплуатации терминальных классов показывает, что для нормальной работы пользователей (открыт большой документ в OpenOffice.org, запущен Mozilla Firefox с несколькими вкладками) необходимы следующие объемы оперативной памяти: примерно 256 МВ для нужд сервера, по 256 МВ для каждого из примерно 8 клиентов и по 128 МВ для каждого из клиентов, начиная с девятого (если всего их больше восьми). Экономия при большом количестве клиентов может достигаться, к примеру, за счет эффективного использования механизма разделения памяти. Заметим, что если использовать не KDE (или аналогичную по "тяжести" среду), а, скажем, XFCE, то на 8 клиентов вполне может хватить и 1 GB памяти.

Замеров загрузки CPU и обращений к диску при исследованиях не проводилось. Опыт, тем не менее, говорит, что вместо одного диска разумно использовать RAID-массив. Так или иначе, если в наличии имеется целый класс достаточно старых компьютеров и есть возможность купить хорошую современную машину "под сервер", то использование сетевой загрузки с "тонкими" клиентами является, безусловно, хорошим выбором.

## Проблема доступа к ресурсам

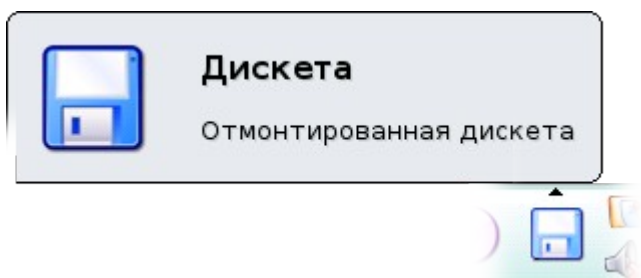
На самом деле мы решили только часть поставленной ранее задачи. У нас корректно разделены запуск и выполнение задач, а также ввод-вывод, связанный с подсистемой X11 (сюда относится в том числе работа с клавиатурой и мышью). Задачи запускаются на машине-сервере, а ввод-вывод X11 осуществляется на рабочей станции. Но таким способом не решаются следующие проблемы:

1. Работа с данными осуществляется на сервере, в то время как для работы со сменными носителями разумно использовать клиентские машины.
2. Если запущенная программа, к примеру, проигрывает звук, то его вывод использует оборудование сервера.

Первую из указанных проблем можно решать по-разному. Первый вариант довольно прозаичен: по окончании работы все пользователи выстраиваются в очередь к администратору класса с флэшками. Это, если вдуматься, весьма разумное решение, особенно в случае учебного класса со строгой дисциплиной работы. Второй вариант использует проделанную создателями "Линукс Терминала" дополнительную работу. Рассмотрим эту возможность подробнее. Выведем список подмонтированных файловых систем:

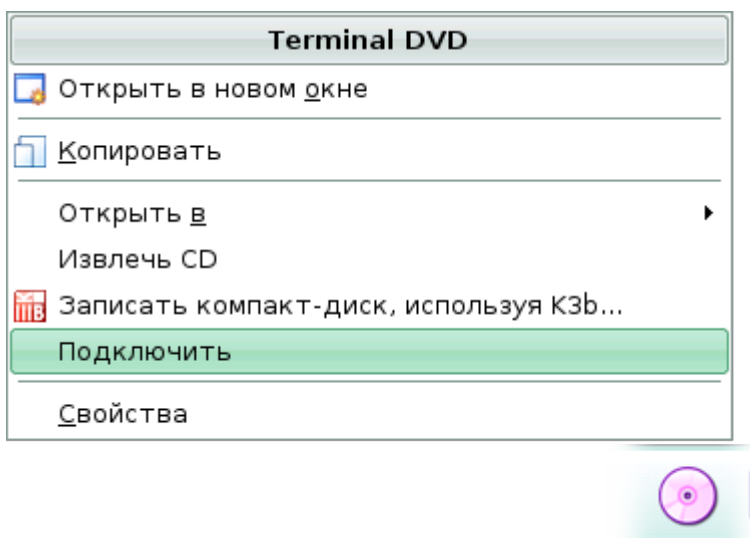
```
$ mount
/dev/hda2 on / type ext3 (rw)
proc on /proc type proc (rw,noexec,nosuid,gid=19)
sysfs on /sys type sysfs (rw)
udevfs on /dev type tmpfs (rw)
devpts on /dev/pts type devpts (rw)
shmfs on /dev/shm type tmpfs (rw)
tmpfs on /dev/tmp type tmpfs (rw,nosuid)
/dev/hda6 on /home type ext3 (rw,nosuid)
/dev/hda5 on /var type ext3 (rw,nosuid)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
ltsufs on /home/user/Drives/floppy0/ type fuse (rw,nosuid,nodev,user=user)
```

Обратим внимание на последнюю строку. Она означает, что на сервере заранее произведено некое "волшебное" действие, которое "возвращает" имеющееся устройство обратно на рабочую станцию. В домашнем каталоге пользователя (который расположен, как и вся файловая система клиента, на сервере) есть подкаталог Drives, а в нем соответствующий подкаталог floppy0, который соответствует дискете, вставленной в машину-клиент.



На скриншоте видна иконка на панели с названием "отмонтированная дискета". Идея проста: устройство, подключенное к локальной машине (рабочей станции) хитрым способом (с помощью системы fuse --- filesystem in userspace) "прокидывается" на сервер. Именно информацию об этом пробросе и сообщила нам программа mount. Тем самым, когда какая-либо программа на сервере обращается к каталогу /home/user/Drives/floppy0, она в действительности заходит на нужную рабочую станцию в соответствующий каталог. Для нелюбознательного пользователя это выглядит точно так же, как работа с обычной машиной.

Таким же образом устроена и работа с CD/DVD-приводами. Функционирование ПО происходит на сервере, при этом диск с клиентской машины доступен с помощью такого же обратного проброса. Кликнем по соответствующей иконке и подмонтируем наш диск, выбрав пункт меню "Подключить":

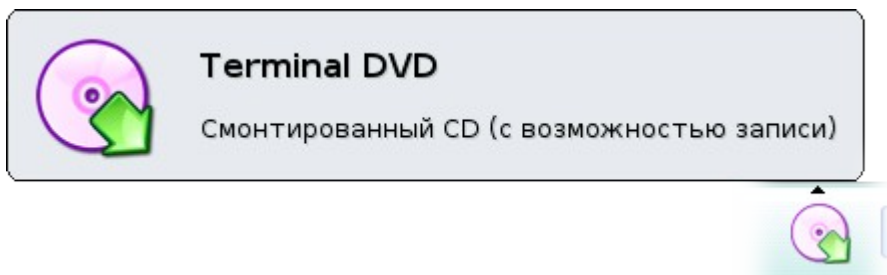


Посмотрим теперь на вывод программы mount:

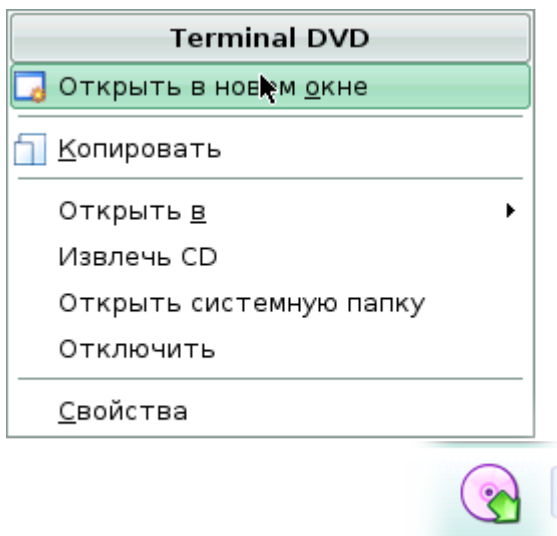
```
$ mount
/dev/hda2 on / type ext3 (rw)
proc on /proc type proc (rw,noexec,nosuid,gid=19)
sysfs on /sys type sysfs (rw)
udevfs on /dev type tmpfs (rw)
devpts on /dev/pts type devpts (rw)
shmfs on /dev/shm type tmpfs (rw)
tmpfs on /dev/tmp type tmpfs (rw,nosuid)
/dev/hda6 on /home type ext3 (rw,nosuid)
/dev/hda5 on /var type ext3 (rw,nosuid)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
ltspfs on /home/user/Drives/floppy0/ type fuse (rw,nosuid,nodev,user=user)
```

```
ltspfs on /home/user/Drives/atacd-hdc/ type fuse (rw,nosuid,nodev,user=user)
/dev/hdc on /media/cdrom type iso9660 (rw,noexec,nosuid,nodev,utf8,user=user)
```

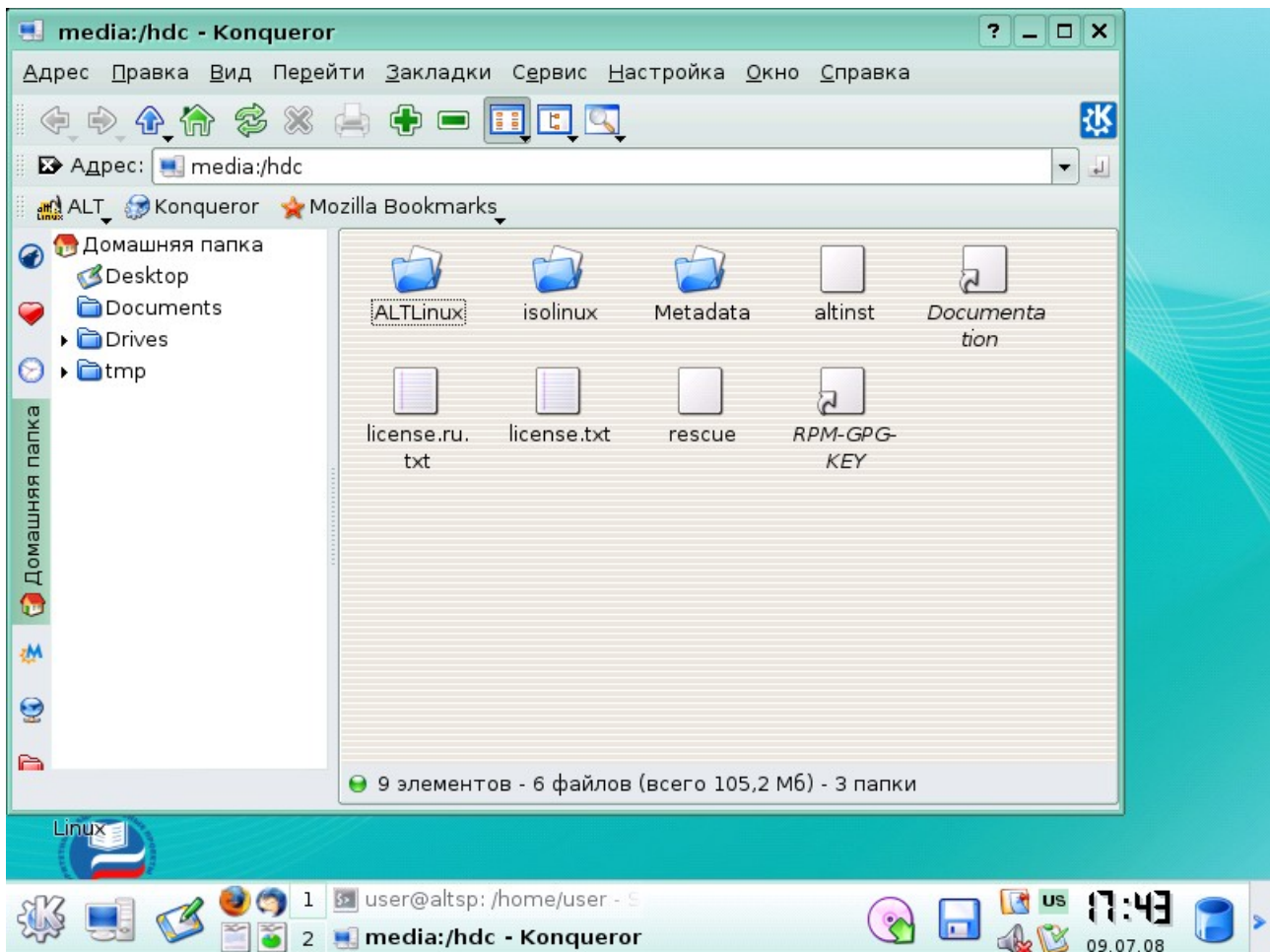
Последние две строки говорят нам о том, что подключение прошло успешно. Изменилась и иконка на панели:



Кликнем по ней и выберем пункт меню "Открыть в новом окне":



Откроется окно с содержимым нашего DVD-диска:



Подобным же образом (проброс сервиса) решается и вторая проблема, касающаяся звукового вывода. Здесь, однако, есть одна неприятная особенность: поскольку работа со звуковой подсистемой в разных видах Unix-систем устроена несколько различающимися способами, то далеко не все программы легко умеют работать с пробросом звука на рабочую станцию. Для тех программ, которые умеют в качестве выходного устройства использовать не конкретные файлы (`/dev/dsp`), а специальные звуковые подсистемы (иными словами, предоставляют пользователю возможность тонкой настройки), есть возможность использовать `esd` --- Enlightened Sound Daemon. Эта программа запускается на клиентской машине, принимает сетевые подключения по специальному протоколу и воспроизводит передаваемые ей аудиоданные. На сервере устанавливается переменная окружения `ESPEAKER`, которая указывает умеющим работать по этому протоколу программам, что для проигрывания звука надо использовать удаленный `esd`-сервер с заданным адресом. Все собранные с поддержкой библиотеки `libesd` программы работают именно так. Аналогичная схема используется и для программ, работающих со звуковой системой `PulseAudio`. Программы, использующие `ALSA` (`Advanced Linux Sound Architecture`), работают с помощью специального `PulseAudio`-плагина (`plug-in`) к `ALSA`. Лишь пишущие по старинке в `/dev/dsp` программы не могут быть настроены для работы по сети. Множество запущенных программ такого типа могут создать какофонию на звуковом устройстве сервера. Для устранения этой проблемы достаточно настроить сервер таким образом, чтобы устройство `/dev/dsp` в системе отсутствовало: вынуть (физически) звуковую карту, отключить (запретить) загрузку ее модуля ядра и т. п. Тогда можно будет эмулировать `/dev/dsp` пробрасываемыми по сети средствами.

Подведем итоги. Терминальный класс дает возможность:

- эффективно эксплуатировать устаревшее оборудование;
- администрировать одну машину вместо целого класса (если рабочий процесс

допускает возможность использования терминальных клиентов; существенно упрощается при этом, например, процедура переустановки ПО).

Недостатки терминального класса следующие:

- Современные X Window System не работают с некоторыми устройствами, к примеру с S3Trio64.
- Один пользователь не имеет возможности работать на двух рабочих станциях одновременно (различные запущенные в одно время сеансы работы должны использовать различные учетные записи).

Заметим, что на сегодняшний день только "Линукс Терминал" обеспечивает "из коробки" концепцию терминального класса и мобильного рабочего места. Достигается это при помощи личных учетных записей, хранящихся на сервере. Использовать чужие учетные записи пользователь не может, так как они защищены неизвестными ему паролями. Каждый пользователь при этом может сесть за любое свободное рабочее место и получить доступ к своей учетной записи, то есть к своему собственному персонализированному окружению. Управление учетными записями централизованно и абсолютно прозрачно (может производиться, к примеру, с помощью Alterator'a).

# Установка и настройка FTP-сервера и NFS-сервера

## Установка FTP-сервера

FTP-сервер позволяет организовать доступ к файлам компьютера по сети, в том числе через интернет. Обычно через FTP предоставляют доступ только на чтение информации. Для развертывания на нашей машине FTP-сервера установим для пакеты vsftpd и anonftp:

```
# apt-get install vsftpd anonftp
```

Поясним сперва, почему одного пакета FTP-сервера vsftpd недостаточно. Первая причина заключается в следующем. Программ, предоставляющих нужную нам функциональность, в хранилище более одной: это proftpd, vsftpd (которую мы и собираемся использовать), pure-ftpd. При этом дерево каталогов, которое соответствует содержимому FTP-сервера, не должно быть жестко привязано к пакету какому-то конкретного FTP-сервера, но в тоже время должно создаваться при развертывании сервера. Именно за это и отвечает пакет anonftp:

```
$ rpmquery -l anonftp
/var/ftp
```

Есть, вторая причина существования пакета anonftp. В дистрибутивах ПСПО, уделяющих большое внимание безопасности, включено множество разных "защит от дурака". В данном случае защита заключается в следующем: установка FTP-сервера и его запуск с настройками по-умолчанию не должны приводить к его автоматическому включению в режиме доступа на запись для всех желающих.

Итак, требуемые пакеты в систему установлены. Займемся их включением. Сервер vsftpd (от *Very Secure FTP Daemon*) использует при своей работе сетевой метасервер xinetd. Для настройки того, какие службы запускаются в системе в процессе инициализации, используется утилита chkconfig. Используемая в ПСПО версия этой утилиты умеет управлять системными службами разного типа: как оформленными в стиле сценариев "start-stop", которые располагаются в `/etc/init.d/`, так и использующими мета-сервер xinetd. При применении chkconfig из командной строки администратор избавлен от необходимости запоминать различный синтаксис для этих двух групп. Посмотрим, какие службы сейчас включены:

```
# chkconfig --list | grep xinetd -A 20
```

```
xinetd based services:
  chargin-tcp:    off
  chargin-udp:   off
  cups-lpd:      off
  daytime-tcp:   off
  daytime-udp:   off
  discard-tcp:   off
  discard-udp:   off
  echo-tcp:      off
  echo-udp:      off
  rsync:         off
  time-tcp:      off
  time-udp:      off
  vsftpd:        off
```

Как видно, все использующие мета-сервер xinetd сетевые службы в настоящее время выключены (в том числе и vsftpd). Если нужно, чтобы FTP-сервер запускался каждый раз при старте машины, нужно проделать следующие операции:

- обеспечить запуск мета-сервера xinetd:

```
# chkconfig xinetd on
```

- включить сам FTP-сервер vsftpd:

```
# chkconfig vsftpd on
```

После чего мы обнаружим, что мы встретились еще с одной отличительной особенностью дистрибутивов ПСПО. Дело в том, что все сетевые сервисы по умолчанию доступны только с локальной машины --- иными словами, подключиться к ним можно только с адреса 127.0.0.1. Это сделано по соображения безопасности, поскольку при таком подходе конфигурация системы по-умолчанию является безопасной. В данном случае следует при помощи текстового редактора закомментировать в файле `/etc/xinetd.conf` строку с параметром `only_from`. Для примера мы воспользуемся не обычным редактором, а утилитой `sed` для внесения этих изменений, добавив в качестве комментария знаки `###`:

- ```
# sed 's/^\s*only_from/###&/' -i /etc/xinetd.conf
# grep only_from /etc/xinetd.conf
###      only_from = 127.0.0.1
```

Отметим, что файл `/etc/xinetd.conf` содержит настройки, общие для всех использующих `xinetd` сервисов. Теперь, поскольку мы изменили содержимое этого конфигурационного файла, нужно перезапустить метасервер `xinetd`, чтобы внесенные нами поправки вступили в силу:

```
# service xinetd restart
```

После этого локальный FTP-сервер начнет принимать входящие соединения на 21-й порт. Для проверки функционирования сервера можно воспользоваться FTP-клиентом `lftp`.

```
$ lftp localhost
lftp localhost:~> ls
lftp localhost:~/> exit
```

Сообщений об ошибках, как видим, нет. Впрочем, файлов на сервере тоже пока нет. Скажем еще несколько слов о дисциплине организации FTP-сервера. В каталоге, который является корнем для службы FTP (в нашем случае это `/var/ftp`) традиционно создается подкаталог `pub`, файлы в котором доступны на чтение. Именно этот каталог и предназначен для работы "публичных" пользователей.

```
# mkdir /var/ftp/pub
```

## Настройка и использование FTP-сервера

Поговорим немного о настройке и использовании FTP-сервера. Во-первых, использовать FTP с системными логином и паролем пользователя совершенно неразумно. Эти данные в протоколе FTP передаются по сети открытым текстом, поэтому "подслушать" их не составляет труда. По этой причине FTP-серверы чаще всего используются лишь для предоставления анонимного доступа на чтение к тем или иным данным. Некоторые FTP-клиенты, заметим, требуют при их использовании обязательного ввода логина и пароля даже при работе с "анонимными" FTP-серверами. В таких случаях используется логин `anonymous` (иногда `ftp`) и какой угодно, но содержащий символ коммерческого `at` (`@`) пароль. Последнее требование, впрочем, предъявляется не всеми FTP-серверами.

Отметим, однако, что в некоторых случаях доступ к данным по протоколу FTP все же защищается паролем. Этим могут заниматься, к примеру, владельцы FTP-серверов, распространяющие сомнительные с точки зрения лицензионной чистоты программы и

данные. Трюк заключается в следующем: в случае возникновения каких-либо претензий всегда можно сослаться на использование пароля и объяснить, что защищенные таким способом файлы не являются общедоступными и, следовательно, свободно по сети не распространяются, а лежат для личного использования.

Поясним теперь, зачем в хранилище помещено более одной программы, реализующей функциональность FTP-сервера. В случае, когда аппаратная конфигурация используемой в качестве сервера машины и скорость ее канала передачи данных оставляют желать лучшего, а компьютеры-клиенты образуют несколько различных категорий пользователей, может возникнуть необходимость использовать специальные возможности таких программ. К таким возможностям относится выставление приоритетов в соответствии с IP-адресами клиентов, ограничения на число одновременных соединений, защита от множественного скачивания программами типа [ReGet](#) и пр. В случае же, когда такие возможности не требуются, разумно использовать более "легковесные" программы.

## Виды FTP-соединения

С использованием протокола FTP может быть связана еще одна проблема. Дело в том, что скачивание файлов по протоколу FTP требует создание специального соединения для передачи данных. IP-адрес и порт для создания этого соединения передаются по сети в пакетах прикладного уровня (в рамках так называемого "управляющего" соединения). В случае "активного" FTP передаются адрес и порт клиента, а случае "пассивного" --- адрес и порт сервера. Понятно, что при использовании трансляции адресов (Network Address Translation, NAT) такой механизм работы может индуцировать довольно неприятные ситуации. К примеру, при работе с "пассивным" сервером строго следующий описанию протокола FTP-клиент (примером может служить Mozilla Firefox) не может проигнорировать полученный таким способом IP-адрес и, если этот адрес оказывается локальным для сети сервера, терпит неудачу при попытке соединения.

Возможным решением данной проблемы может служить тонкая настройка FTP-сервера таким образом, чтобы передаваемый IP-адрес для подключения в "пассивном" режиме различался для клиентов из разных сетей. Иногда оказывается полезным держать несколько FTP-серверов на одной машине и "разруливать" подключения к 21-му порту (ftp) с помощью средств сетевого экрана iptables. При настройке маршрутизатора хорошую службу может сослужить также специальный модуль ip\_nat\_ftp к iptables.

## Установка NFS-сервера

В отличие от FTP, служба NFS является полноценной сетевой файловой системой, поэтому мы сразу будем готовить ее как для сетевой установки, так и для последующего использования для хранения домашних каталогов пользователей в нашем классе. Для работы NFS необходимо следующее.

1. Установить на сервере пакет nfs-server.

```
# apt get install nfs-server
```

2. Проверить конфигурацию службы portmap, она не должна запускаться с ключом -l, который разрешает только локальные подключения. Пример правильного файла конфигурации /etc/sysconfig/portmap (в ПСПО по умолчанию):

```
# Parameters for portmap daemon.  
# See portmap(8) for more details.
```

```
# Specifies additional parameters for portmap.  
#PORTMAP_ARGS="-l"
```

Если строка с ключом `-l` не закомментирована, требуется ее закомментировать и перезапустить службу `portmap` командой `service portmap restart`.

3. В файле `/etc/exports` указать, какие каталоги будут отдаваться по сети. В нашем случае он должен содержать две строчки выглядеть так:

```
$ cat /etc/exports
/srv/boot      *(ro,no_subtree_check,no_root_squash)
/srv/home      172.16.0.0/12(rw,no_subtree_check,no_root_squash)
```

Здесь `172.16.0.0/12` --- адрес локальной сети, в вашем случае он может быть другим.

4. После любых изменений `/etc/exports` необходимо перезапустить службы `nfs`, чтобы она использовала новые настройки:

```
# service nfs restart
```

Содержимое файла `/etc/exports` следует прокомментировать. Во-первых, здесь раздаётся каталог `/srv/boot`, с которого клиенты будут загружаться, причём для них он доступен в режиме "только для чтения". Кроме того, использована опция "no root squash". По умолчанию действует обратная опция "root squash" которая все запросы от лица суперпользователя с клиента на сервере переводит в запросы от лица пользователя "nobody" для большей безопасности. Опция "no root squash" отменяет такое преобразование. В отличие `/srv/boot`, `/srv/home` (будущий домашний каталог сетевых пользователей) необходимо экспортировать на запись.

После запуска `nfs` можно проверить, отдаются ли каталоги, с помощью команды `showmount -e`:

```
# showmount -e
Export list for server:
/srv/boot *
/srv/home 172.16.0.0/12
```