ТЕРМИНАЛЬНЫЕ РЕШЕНИЯ ПОД LINUX

Поддубный Виталий
ЦКП «ЛинКом»
сайт: www.lincom.su
почта: linuxtula@yandex.ru
г.Тула
18.03.2014г.
Обновлено: 25.06.2014г.

ВНИМАНИЕ! Данная статья распространяется в виде «как есть», поэтому автор не несёт ответственности за возможный ущерб, прямой или косвенный, понесённый в результате действий пользователей, выполняющих настройку согласно инструкций в данной статье.

Вы имеете право использовать материалы из данной статьи частично или полностью с обязательной ссылкой на первоисточник и автора.

Все работы по тестированию предложенных решений проводились в системах Ubuntu и Debian и дистрибутивах на их основе, но приведённые в статье команды и настройки могут работать и в других системах Linux.

Если вы считаете, что данная статья оказалась для вас полезной, вы можете материально помочь автору в дальнейшей работе над данной статьёй или другими, посвящёнными операционной системе Linux. Речь идёт о любой сумме, которую вы сочтёте разумной в качестве оплаты за труды автора. Деньги можно перевести несколькими способами:

Яндекс-кошелёк: 410011014464436

Webmoney-кошелёк: R163250656294

Также вы можете присылать свои замечания, советы, дополнения, делиться своими наработками и т.д.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной статье описываются различные варианты настройки терминальных серверов под управлением операционной системы Linux.

Согласно Википедии терминальный сервер (или сервер терминалов) — это сервер, предоставляющий клиентским компьютерам свои вычислительные ресурсы (процессорное время, память, дисковое пространство) для решения различных задач, и как правило применяется для удалённого обслуживания пользователя с предоставлением рабочего стола.

Терминальные решения актуальны как в крупных корпорациях, так и в небольших фирмах, например, для ускорения работы программ на базе 1С. Как правило, в небольших организациях работа с программами 1С устроена следующим образом: на каком-либо компьютере установлена база данных 1С и открыта в общий доступ, а клиенты подключаются к базе по сети. В результате из-за особенностей баз 1С работа на клиентских компьютерах довольно медленная, отчёты могут формироваться несколько минут, а если к одной базе одновременно подключается несколько пользователей, то работа становится просто невыносимой. В таких случаях терминальный сервер решает проблему со скоростью, поскольку все операции выполняются на нём, а на клиентские компьютеры транслируется только изображение рабочего стола сервера. Каких-либо особых требований к клиентским компьютерам не предъявляется.

Наибольшее распространение терминальные серверы получили на базе операционной системы Windows, но стоимость лицензий на такое решение не всем организациям по карману: необходимо приобретать не только операционную систему Windows Server, но и терминальные лицензии на каждое рабочее место. Да и с лицензионной политикой корпорации Microsoft тоже не всё гладко: лицензионные соглашения написаны довольно расплывчато, некоторые пункты можно трактовать двояко, в лицензиях довольно легко запутаться, и даже дипломированные консультанты из техподдержки Microsoft'а могут ответить далеко не на все вопросы.

Некоторые разработчики предлагают терминальные решения для настольных операционных систем Windows (XP, 7, 8 и т.д.), что несколько снижает общую стоимость сервера, однако такие системы в зависимости от версии позволяют подключаться максимум 10-ти пользователям, и согласно лицензионной политике компании Microsoft вы в любом случае обязаны купить дополнительные лицензии для каждого подключаемого клиента, так что экономия в итоге оказывается невелика.

И тут как всегда на помощь приходит операционная система Linux. Не будем описывать всех достоинств этой системы, они и так уже всем давно известны, скажем лишь, что под неё существуют как платные, так и бесплатные варианты различных терминальных серверов. Как и полагается, у бесплатных терминальных решений есть некоторые недостатки и ограничения, но для небольших организаций они не столь существенны, и данные решения можно смело применять на практике.

Существует несколько вариантов терминальных серверов, работающих по различным протоколам:

• по протоколу NX: различные сборки FreeNX, RX@Etersoft, X2Go, NeatX, Xpra, 2X Terminal

Server (бесплатные), NX NoMachine (коммерческий);

• по протоколу RDP: Xrdp+X11rdp, Xrdp+VNC (бесплатные), LX-Server (коммерческий), Ulteo Open Virtual Desktop (бесплатные и коммерческие версии);

• по протоколу XDMCP: LTSP, XDMCP+VNC (бесплатные).

Также операционная система Linux позволяет запускать графические утилиты по протоколу SSH, для этого на клиентских системах Linux используется специальная команда, а в системах Windows — связка Putty+Xming.

У каждого варианта есть свои достоинства и недостатки, о которых будет сказано ниже. Некоторые терминальные решения будут рассмотрены более подробно.

БЛАГОДАРНОСТИ

Хочу выразить благодарность Dimbor'у, участнику форума unixforum.org и автору многих патчей и доработок для сервера FreeNX, за консультации и внесённые исправления.

РЕКОМЕНДАЦИИ

Перед началом установки любого терминального сервера хочу дать несколько рекомендаций:

1) нужно обязательно создать и настроить профиль каждого пользователя; если пользователей очень много, то можно создать и полностью настроить профиль одного пользователя, например, *admin*, затем скопировать его в каталог **/etc/skel** и дальше для применения этих настроек для других пользователей воспользоваться скриптом:

```
#!/bin/bash
admin=ADMINNAME
user=USERNAME
echo "Обработка профиля пользователя" "${user}"
echo "Ввод нового пароля пользователя"
passwd "${user}"
echo "Создание профиля пользователя по шаблону"
rm -Rf /home/"${user}"/* /home/"${user}"/.[a-zA-Z0-9]*
cp -r /etc/skel/* /etc/skel/.[a-zA-Z0-9]* /home/"${user}"
chown -R "${user}":"${user}" /home/"${user}"
chmod -R 0770 /home/"${user}"
echo "Выполнение настроек профиля WINE"
sed -i "s/${admin}/${user}/g" /home/"${user}"/.wine/system.reg
sed -i "s/${admin}/${user}/g" /home/"${user}"/.wine/user.reg
sed -i "s/${admin}/${user}/g" /home/"${user}"/.wine/userdef.reg
find /home/"${user}" -name '*.desktop' -exec sed -i "s/${admin}/${user}/g" {} \;
mv /home/"${user}"/.wine/drive_c/users/"${admin}" /home/"${user}"/.wine/drive_c/users/"${user}"
ln -s /home/"${user}" /home/"${user}"/.wine/dosdevices/d:
ln -s /home/"${user}" /home/"${user}"/.wine/drive_c/users/"${user}"/"Мои документы"
ln -s /home/"${user}"/"Изображения" /home/"${user}"/.wine/drive_c/users/"${user}"/"Мои рисунки"
ln -s /home/"${user}"/"Видео" /home/"${user}"/.wine/drive_c/users/"${user}"/"Мои фильмы"
ln -s /home/"${user}"/"Музыка" /home/"${user}"/.wine/drive_c/users/"${user}"/"Моя музыка"
ln -s /home/"${user}"/"Рабочий стол" /home/"${user}"/.wine/drive_c/users/"${user}"/"Рабочий стол"
#rm /home/"${user}"/"${user}"
```

Для копирования настроенного профиля сразу для всех пользователей, кроме основного с именем *admin*, используйте скрипт:

```
#!/bin/bash
admin=ADMINNAME
#Изменяем системные пароли пользователей
for user in $( sed -n "s/^\([^:]*\):.*:\/bin\/bash$/\1/p" /etc/passwd ); do
    if [ -d "/home/${user}" ]; then
        if [ ! "/home/${user}" = "/home/${admin}" ]; then
<КОПИРУЕМ СЮДА СОДЕРЖИМОЕ ПРЕДЫДУЩЕГО СКРИПТА БЕЗ ПЕРЕМЕННЫХ>
      fi
    fi
done
```

2) если вы планируете использовать терминальный сервер в режиме Desktop, т.е. с загрузкой рабочего стола на клиентской машине, то рекомендуется устанавливать «легковесные» оболочки *XFCE4* или *LXDE*; если в настройках сервера требуется прописать команду запуска оболочки, то прописываем **`startxfce4`** или **`startlxde`** соответственно, это решение повысит быстродействие терминальной сессии, все окна будут прорисовываться гораздо быстрее, особенно с LXDE, а нагрузка на сервер снизится.

1. ПРОТОКОЛ NX

Протокол NX был разработан итальянской компанией NoMachine для эффективной работы с классической системой X Window. Этой системе свойственны большие задержки при передаче информации и неоптимальный расход полосы пропускания, но протокол NX отлично решает эти проблемы. На текущий момент в большинстве терминальных решений используется протокол NXv3, но в конце сентября 2013 года компания анонсировала сервер NoMachine 4.0, работающий по протоколу NXv4. К сожалению компания-разработчик прекратила поддержку всех протоколов предыдущих версий, а исходный код нового протокола полностью закрыт, открыт исходный код только компонентов, распространяемых под лицензией GPL и включенных в состав продуктов NoMachine.

Под операционную систему Linux существует несколько решений терминальных серверов, работающих по протоколу NX.

Оригинальный сервер NX Server от компании-разработчика NoMachine — коммерческий, существует несколько версий, отличающихся друг от друга возможностями, стоимость лицензий — от $124,50 до $22494,50 в год. Также существуют бесплатные версии для некоммерческого использования, которые можно использовать в качестве удалённого рабочего стола (наподобие R-Admin или VNC).

Компания NoMachine частично открыла исходный код своего проекта, и в 2004 году появился бесплатный и открытый сервер под названием FreeNX. У него конечно же возможностей меньше, чем у коммерческого варианта, но для настройки полноценного терминального сервера в небольших организациях его вполне хватает. Исходный код компонентов оригинального NX-сервера версии 3.5 был доступен на сайте NoMachine, теперь его можно найти только на сторонних сайтах.

На официальном сайте FreeNX доступна версия 0.7.3 от 18.08.2008 года, с тех пор проект признаков жизни не подаёт, хотя о его закрытии объявлено не было.

На базе FreeNX было сделано множество сборок. Например, на сайте launchpad.net можно скачать сборки под Ubuntu, также собственные сборки есть в репозиториях ALT Linux. Помимо этого некоторые компании делали собственные сборки сервера и клиента.

В России довольно популярной стала сборка RX@Etersoft от питерской компании «Этерсофт». В данной сборке устранены проблемы, связанные с кириллицей, а также выполнены некоторые доработки, в частности обеспечена нормальная работа другого продукта компании — Wine@Etersoft, который позволяет запускать популярные в России бухгалтерские программы, такие как 1С, Консультант+, Гарант, некоторые банк-клиенты и т. д. Последние версии сервера и клиента RX@Etersoft для разных дистрибутивов Linux доступны на FTP-сервере компании. Данная сборка может использоваться бесплатно, в случае необходимости приобретается лицензия с коммерческой техподдержкой.

Также свою сборку сервера и клиента попыталась сделать компания 2X, правда менее удачно. Последняя версия — 1.5.0 от 23.08.2006 года, с тех пор проект не развивается. Ознакомиться с кратким описанием, а также скачать последнюю версию терминального сервера и клиента в виде RPM-пакетов можно с официального сайта 2X.

Одним из существенных недостатков проекта FreeNX — использование целой связки языков программирования: Bash, C, Expect, в результате исправление ошибок и написание новых версий модулей превращается в целое искусство. Поэтому проект очень мало поддерживается сообществом, а большинство библиотек «кочует» из одной сборки в другую практически в неизменном виде. Компания Google решила исправить проблему, и в 2009 был анонсирован проект NeatX. Терминальный сервер полностью переписан на языке Python с добавлением скриптов на Bash и единственным модулем на C для повышения производительности. Пока что проект находится в стадии разработки, но уже стали появляться тестовые сборки, например, они доступны на сайте того же launchpad.net и «Этерсофта».

Под Linux существует ещё один терминальный сервер — X2Go, который тоже работает по протоколу NX, однако это уже совсем другой продукт: у него своя серверная и клиентская часть, он не совместим с другими NX-клиентами, но зато легко интегрируется с любым сервером, который для хранения профилей пользователей использует PostgreSQL или OpenLDAP, например, MS Active Directory. Подобный функционал есть только в коммерческих версиях NX-сервера от NoMachine.

1.1. СЕРВЕР FREENX

1.1.1. Установка сервера FreeNX

В интернете можно найти множество описаний установки сервера FreeNX на различные системы, но везде принцип один и тот же.

Сначала необходимо установить из репозиториев и сконфигурировать сервер SSH:

```
sudo apt-get install openssh-server openssh-client
```

или

```
sudo apt-get install ssh
```

Далее отредактируйте файл настроек SSH-сервера **/etc/ssh/sshd_config**:

1) для увеличения безопасности желательно изменить порт по умолчанию 22 на какой-нибудь другой, например, 2122:

```
Port 2122
```

Этот же порт затем нужно прописать в настройках сервера.

2) найдите и отредактируйте строки, как показано ниже, если они отсутствуют, пропишите вручную:

```
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile %h/.ssh/authorized_keys
```

3) если для авторизации будут использоваться данные системных пользователей (***ENABLE_USERMODE_AUTHENTICATION="1"***) или сервер SSH (***ENABLE_SSH_AUTHENTICATION="1"***), то в строке ***AllowUsers*** необходимо прописать всех пользователей сервера FreeNX, включая пользователя с именем **nx**:

```
AllowUsers nx user1 user2 user3
```

или

```
AllowUsers nx user1@127.0.0.1 user2@127.0.0.1 user3@127.0.0.1
```

С этими настройками доступ к SSH-серверу с других компьютеров будет невозможен.

Если настроена авторизация с использованием собственной базы пользователей NX-сервера (***ENABLE_PASSDB_AUTHENTICATION="1"***) или режима суперпользователя (***ENABLE_PASSDB_AUTHENTICATION="1"***), то эти настройки выполнять не нужно.

4) выполните другие настройки безопасности SSH на своё усмотрение.

После проделанных настроек перезапустите SSH-сервер:

```
sudo service ssh restart
```

Устанавливаем дополнительные пакеты:

```
sudo apt-get install bc cifs-utils coreutils cron cups cups-bsd cups-client curl dbus-x11 expect findutils foomatic-db foomatic-db-engine gawk grep libc-bin login logrotate netcat openssl psmisc python sed sudo util-linux x11-utils x11-xkb-utils x11-xserver-utils xauth zenity libasound2 libfreetype6 libjpeg8 libpng12-0 libx11-6 libxcomposite1 libxdamage1 libxfixes3 libxmu6 libxmuu1 libxpm4 libxrandr2 libxtst6 xfonts-base xkb-data zlib1g
```

Возможно, что большинство из этих пакетов у Вас в системе уже будут установлены, а остальные автоматически скачаются при установке сервера из репозиториев. Но если Вы устанавливаете сервер из исходников или сторонних сборок, то не поленитесь лишний раз выполнить эту команду.

Теперь устанавливаем сам сервер FreeNX, для чего необходимо получить нужные пакеты. Сделать это можно несколькими способами:

1) установить пакеты из репозиториев своей системы, если они отсутствуют, то подключить сторонний репозиторий и выполнить установку из него.

Пример установки для Ubuntu 12.04 или ниже:

```
sudo apt-add-repository ppa:freenx-team
sudo apt-get update
sudo apt-get install freenx freenx-server
```

ВНИМАНИЕ! Последняя версия FreeNX в этом репозитории скомпилирована для Ubuntu 12.04

Precise Pangolin, в Ubuntu 12.10 и выше она работать не будет из-за несовместимости этой версии с библиотекой **libcairo2** версии 1.12 и выше. Решение проблемы описано в п.5 раздела 1.1.4 «Возможные проблемы и способы их решения». С более новой версией библиотеки (1.16 и выше) проблем не возникает.

2) скачать исходники и скомпилировать их под свою систему; в состав сервера FreeNX входит довольно много модулей и компилировать придётся каждый в отдельности; этот вариант наиболее сложный, но и наиболее правильный, поскольку модули будут скомпилированы с заголовками библиотек, используемых конкретно в вашей системе, и вероятность конфликтов из-за несовместимости версий библиотек снижается;

3) скачать и установить готовые бинарные пакеты.

Российская компания «Этерсофт» предлагает свои сборки сервера под различные системы, проект называется RX@Etersoft. Это коммерческий продукт, в который включены патчи, улучшающие работу сервера и устраняющие некоторые проблемы. Сборки можно использовать бесплатно и без ограничений, но перед стартом терминальной сессии будет каждый раз появляться предупреждение об отсутствующем файле лицензии с предложением приобрести её. В данном разделе будет рассмотрен процесс установки и настройки именно этого варианта сервера с версиями пакетов *rx-etersoft_1.1.3-eter5* и *nx_3.5.1-eter12*, которые были последними на момент написания статьи и выполненные на основе исходных кодов оригинального сервера NX-Server 3.5.0.

Скачиваем с FTP-сервера «Этерсофта» последнюю стабильную версию RX под свою систему. Нам потребуются пакеты *rx-etersoft*, *nx* и утилита администрирования *nxsadmin*. Устанавливаем их при помощи утилиты DPKG или любой другой.

Во время установки создастся пользователь и группа **nx**.

В старых версиях RX@Etersoft 1.1.2 этому пользователю необходимо задать пароль:

```
sudo passwd -u nx
```

В версии 1.1.3 и выше этого делать не нужно!

Выполняем предварительную настройку сервера, сделать это можно при помощи специального скрипта:

```
sudo /usr/bin/rxsetup
```

Если скрипт во время работы выдал сообщение об ошибке, выполняем настройки вручную. Часть настроек мы уже сделали ранее, поэтому выполняем остальные:

1) присваиваем нужные права файлам сервера печати CUPS:

```
sudo chmod 755 /usr/sbin/cupsd
sudo chmod 711 /usr/lib/cups/backend/ipp
```

2) генерируем ключ NX-сервера:

```
sudo nxsetup --install --setup-nomachine-key
```

Примечание: данная команда генерирует ключ сервера по умолчанию, но в целях повышения безопасности рекомендуется сгенерировать уникальный ключ, запустив эту команду без параметра `--setup-nomachine-key`

3) удаляем файл **known_hosts**:

```
sudo rm -fr /var/lib/nxserver/home/.ssh/known_hosts
```

4) запускаем сервер:

```
sudo service rx-etersoft start
```

5) добавляем скрипт запуска сервера в автозагрузку:

```
sudo update-rc.d rx-etersoft defaults
```

6) если в системе используется SELinux, применяем политики по умолчанию:

```
sudo restorecon -Rv /var/lib/nxserver
```

Если SELinux в системе не используется, команду выполнять не нужно

7) проверяем, что сервер запустился нормально:

```
sudo nxsetup --test
```

Желательно, но необязательно, все ошибки с пометкой WARNING исправить и запустить тест ещё раз.

На этом предварительное конфигурирование сервера закончено.

После проделанных действий в каталоге **/var/lib/nxserver/home/.ssh/** должны появиться ключи. Если вы сгенерировали свой уникальный ключ, то из этого каталога необходимо скопировать файл *client.id_dsa.key* на флэшку и импортировать его в NX-клиент на каждой машине. Если этого ключа нет или по каким-либо причинам вы решили создать новый, воспользуйтесь командой:

sudo nxkeygen

Если сгенерирован ключ по умолчанию с опцией ***--setup-nomachine-key***, то этот шаг пропускаем.

Затем немного настроим сервер. Конфигурирование оригинального FreeNX осуществляется путем редактирования файла **/etc/nxserver/node.conf**. В сборках RX@Etersoft в целях упрощения настроек этот файл разбит на несколько, расположенных в каталоге **/etc/nxserver/node.conf.d/**.

NX-сервер для авторизации изначально использует пароли системных пользователей, но можно создать свою базу данных пользователей. Раскомментируем и изменим параметр ***ENABLE_PASSDB_AUTHENTIFICATION*** с ***"0"*** на ***"1"***. Этим мы разрешаем авторизацию при помощи базы паролей NX-Server, и необходимость прописывания всех пользователей в файл настройки SSH-сервера отпадает, но пользователям придётся запоминать сразу два пароля: на вход в систему и доступ к серверу. Поскольку для авторизации пользователей можно использовать только один из четырёх режимов, для исключения конфликтов раскомментируем строки с другими способами авторизации (***ENABLE_USERMODE_AUTHENTICATION***, ***ENABLE_SSH_AUTHENTICATION*** и ***ENABLE_SU_AUTHENTICATION***) и везде прописываем значение ***"0"***.

Далее добавляем пользователей NX-сервера и задаём пароли для них:

sudo nxserver --adduser <USERNAME>

sudo nxserver --passwd <USERNAME>

Рекомендую имена пользователей и пароли для них делать такими же, как в системе, чтобы не забивать голову лишней информацией ни себе, ни другим.

При попытке повторного подключения в клиенте всегда будет появляться окно со списком запущенных сессий. Чтобы не учить бухгалтеров нажимать на кнопку *New*, изменяем параметр ***ENABLE_SHOW_RUNNING_SESSIONS*** на ***0***. При этом будет автоматом запускаться новый сеанс, а при наличии приостановленных сессий будет выводиться окно с их списком.

На время настроек рекомендуется выставить максимальный уровень ведения логов ***NX_LOG_LEVEL=7***. После того, как настройка полностью завершена и мы убедились, что всё работает нормально, можно отключить ведение логов (ставим обратно ***0***), чтобы не создавать лишнюю нагрузку на систему.

Если вы для протокола SSH используете не 22-й порт, а какой-либо другой (например, 2122), то его обязательно надо указать в строке ***SSHD_PORT=2122***

Если вы хотите отключить шифрование трафика, то в строке ***EXTERNAL_PROXY_IP*** нужно прописать IP-адрес сетевой карты, которая работает с локальной сетью, а также в файерволле открыть диапазон TCP-портов на входящий трафик, начиная от **[$DISPLAY_BASE+4000]** (если значение по умолчанию указанной переменной в настройках не меняли, то открываем диапазон портов от 1000 до 5000).

ВНИМАНИЕ! Отключение шифрования трафика немного повышает скорость работы терминальной сессии, но влияет на безопасность системы, поэтому данную настройку рекомендуется использовать только при подключении клиентов внутри локальной сети!

Более подробное описание настроек приведено в следующем разделе.

Проверяем правильность конфиг-файла командой:

sudo nxloadconfig --check

Программа конечно же укажет на ошибку в параметре ***COMMAND_START_CDE=cdwm***, не обращайте внимание, этот параметр для системы Solaris.

Примечание: если на сервере установлена оболочка XFCE или LXDE, то можно задействовать этот параметр для запуска одной из указанных оболочек.

После всех изменений сохраняем файл и перезапустим сервер:

sudo service rx-etersoft restart

или

sudo /etc/init.d/freenx-server restart

Администрировать сервер можно при помощи сторонней утилиты *nxsadmin*, которая обязательно должна быть запущена от имени root'а:

```
sudo /usr/sbin/nxsadmin
```

На этом базовые настройки серверной части завершены, можно устанавливать на клиентские компьютеры NX-Client или OpenNX и пробовать подключиться.

1.1.2. Файлы конфигурации сервера FreeNX

В данном разделе описан основной файл конфигурации сервера FreeNX — **node.conf**, располагающийся в каталоге **/etc/nxserver/**. За основу взята конфигурация сборки RX@Etersoft. В данной сборке файл **node.conf** заменён каталогом **/node.conf.d/**, в котором расположено несколько конфигурационных файлов, названных по группе настроек. Но общая суть и процесс конфигурирования от этого нововведения практически не меняются.

В описании приведено содержимое каждого файла и значение по умолчанию для каждого параметра. Чтобы изменить настройку, раскомментируйте нужную строку и пропишите в неё значение, после чего перезапустите NX-сервер.

Обозначения настроек некоторых параметров таковы:: ***0 = False (запретить)***, ***1 = True (разрешить)***, если не требуется указать другое значение (номер порта, команду запуска и т. д.)

• Файл 00-general.conf

В файле содержатся основные настройки сервера RX@Etersoft.

SERVER_NAME="$(hostname)" — имя хоста, используемое NX-сервером. Рекомендуется использовать в случаях, когда имя NX-сервера должно отличаться от имени сервера по умолчанию.

EXTERNAL_PROXY_IP="" — IP-адрес сервера, используемый для незашифрованных сеансов. Пропишите его, если требуется использовать определённый внешний IP или автоопределение не работает.

SSHD_PORT=22 — порт, используемый для подключения к SSH-серверу (можно посмотреть в файле **/etc/sshd_config**)

• Файл 01-auth.conf

Файл содержит настройки авторизации и защиты.

ENABLE_USERMODE_AUTHENTICATION="0" — включение режима пользователя в качестве одного из методов авторизации; этот режим позволяет обычному пользователю запустить NX-сервер и его оболочку от своего имени и подключаться к серверу, используя определённые ручные настройки в клиенте.

ENABLE_PASSDB_AUTHENTICATION="0" — включение авторизации при помощи базы паролей NX-сервера.

ENABLE_SSH_AUTHENTICATION="1" — включение авторизации через SSH-сервер; в этом режиме сервер SSH должен быть настроен на приём паролей через localhost.

ENABLE_SU_AUTHENTICATION="0" — включение авторизации через суперпользователя; для работы этого режима необходимо, чтобы пользователь **nx** был включен в группу **users**, иметь пароль для входа в систему и рабочую графическую оболочку, которая принимает параметр **-c**.

ВНИМАНИЕ! Обязательно должен быть включен только один из четырёх методов авторизации!

ENABLE_USER_DB="0" — включение режима проверки, чтобы все пользователи находились в базе паролей NX-сервера независимо от метода авторизации.

ENABLE_FORCE_ENCRYPTION="0" — включение принудительного режима шифрования трафика; может быть использован, если пользователи в настройках клиента не используют шифрования.

SSHD_CHECK_IP="0" — включение режима проверки IP-адреса клиента сервером SSH; в этом режиме в подключении NX-клиента к серверу будет отказано, если сервер SSH не экспортирует переменные SSH_CONNECTION и SSH_CLIENT в окружение NX-сервера; возможные значения: **0** — все соединения будут приниматься независимо от IP-адреса, **1** — проверяется удалённый

IP-адрес, и если он не определён, в подключении будет отказано.

ENABLE_SLAVE_MODE="1" — включение ведомого режима; в этом режиме пользователю достаточно авторизоваться один раз, а все дальнейшие подключения будут осуществляться с помощью NX-Node в ведомом режиме; это может быть полезно при использовании одноразовых паролей или для уменьшения записей в логах utmp и wtmp; также время запуска сеанса в этом режиме гораздо быстрее, что особенно важно, когда на сервер должны быть проброшены принтеры или каталоги; для работы этого режима бинарный файл **nxserver-helper** должен быть установлен в каталог **PATH_BIN** (как правило это каталог **/usr/bin/**).

ENABLE_LOG_FAILED_LOGINS="1" — включение режима протоколирования неудачных попыток авторизации; при включении этого режима все неудачные попытки входа будут записаны в **auth.log**; режим может быть использовать против перебора паролей в сочетании с другими утилитами, например, **fail2ban**; по умолчанию протоколирование ведётся в системный журнал.

• Файл 02-restriction.conf

В файле содержатся настройки различных ограничений и доступа.

DISPLAY_BASE=1000 — начальный номер базового дисплея, с которого стартуют сессии.

SESSION_LIMIT=200 — максимальное количество одновременно запущенных сессий.

SESSION_USER_LIMIT=200 — максимальное количество сессий, запущенных одним пользователем. По умолчанию значение берётся из **SESSION_LIMIT**.

DISPLAY_LIMIT=200 — количество дисплеев, зарезервированных для сессий; значение должно быть больше или равно значению максимального количества сессий.

ENABLE_PERSISTENT_SESSION="all" — разрешить NX-агенту запуск сессий для пользователей или групп с ключом "persistent" — возможность подключения к уже запущенной сессии (используется для режимов Shadow и VNC); значение по умолчанию — **"all"** (разрешено для всех пользователей), также можно указать имена конкретных пользователей через запятую или группы в формате @GROUPNAME.

DISABLE_PERSISTENT_SESSION="" — имена пользователей и групп, для которых запуск сессий с ключом "persistent" должен быть отключен; настройка особенно полезна, если для предыдущего параметра **ENABLE_PERSISTENT_SESSION** указано значение **"all"**.

ENABLE_MIRROR_VIA_VNC=1 — включение зеркального отображения запущенных сессий через VNC; при запуске сессия отмечается как «возобновляемая» и «VNC-зеркальная»; каким образом работает данный параметр, я не разобрался.

ENABLE_DESKTOP_SHARING=1 — включение общего доступа к локальному дисплею :0 через VNC; при запуске сессия помечается как «возобновляемая» и «локальная через VNC»; для работы данной настройки вы должны иметь права доступа к дисплею, иначе она работать не будет; каким образом работает данный параметр, я также не разобрался.

ENABLE_SESSION_SHADOWING_AUTHORIZATION=1 — разрешение «теневого» подключения к сессии; данная функция позволяет незаметно подключиться к уже запущенной сессии какого-либо пользователя, при этом не мешая ему работать и позволяя следить за его действиями без стороннего вмешательства; по умолчанию слежка допускается только для одного пользователя; если NX-сервер обнаруживает исполняемый файл **nxshadowacl**, он выдаёт запрос, для какого пользователя разрешена слежка; пользователь добавляется командой:

sudo nxshadowacl <USERNAME>

Код вывода: **0** — сохранять куки в файл сессии для других пользователей; **1** — не сохранять куки. Также проверьте, чтобы пользователю было разрешено следить с правами администратора:

sudo nxshadowacl <USERNAME> <ADMINNAME>

Код вывода: **0** — разрешить слежку и добавить в список, **1** — не разрешать слежку.

В NX-сервере 3.0 существует возможность для пользователей выдавать разрешение на «теневое» подключение к своим сессиям других пользователей: **0** — подключаться без запроса разрешения, **1** — (по умолчанию) выводится запрос на разрешение подключения.

ENABLE_INTERACTIVE_SESSION_SHADOWING=1 — разрешить «теневое» подключение в интерактивном режиме; возможные значения: **1** — разрешить взаимодействие «теневой» сессии с

основной (следящий пользователь может выполнять любые действия вместе с основным пользователем, получается функция удалённой помощи наподобие R-Admin), **0** — подключение в режиме «только просмотр», никаких действий следящий пользователь выполнять не сможет.

ENABLE_CLIPBOARD="both" — включение или отключение буфера обмена; возможные значения: **"client"** — содержимое буфера обмена может быть скопировано на клиенте и вставлено внутрь NX-сессии (обмен от клиента к серверу), **"server"** — содержимое буфера обмена может быть скопировано внутри NX-сессии и вставлено на клиентском компьютере (обмен от сервера к клиенту), **"both"** — разрешён двусторонний обмен между сервером и клиентом (обмен от клиента к серверу и наоборот), **"none"** — буфер обмена отключен.

ENABLE_PULLDOWN_MENU="1" — включение или отключение выпадающего графического меню для остановки или прекращения сессий, запущенных в бескорневом режиме (rootless); возможные значения: **1** — включено, в этом случае меню будет появляться, когда курсор мыши располагается ближе к середине верхней части экрана, сессию можно приостановить или прекратить, нажав мышкой на иконку; **0** — отключено, в этом случае для вызова меню используется комбинация клавиш Ctrl+Alt+T.

USE_PROCESSOR_TASKSET="" — данная опция служит для балансировки нагрузки на ядра процессора; например, чтобы запустить оболочку KDE на другом ядре, пропишите в строке **COMMAND_STARTKDE** значение **"taskset -c 2 -- startkde"**; NX-сервер запускается с параметром **$COMMAND_TASKSET -cp "$USE_PROCESSOR_TASKSET" $$**, и если значение параметра **$USE_PROCESSOR_TASKSET** указать как **3,4**, то сервер будет проводить балансировку нагрузки между 3-м и 4-м ядрами процессора; при пустом значении никаких балансировок проводиться не будет.

ENABLE_ADVANCED_SESSION_CONTROL="0" — разрешить расширенный контроль сессий; если вы установите значение **"1"**, то вы сможете запускать новые приложения в уже запущенной бескорневой сессии (режим rootless), используя команду "add <rest of name>" в качестве имени сессии, при этом клиент выдаст сообщение о выполнении.

ENABLE_SHOW_RUNNING_SESSIONS="1" — разрешить просмотр запущенных сессий; если вы установите значение **"0"**, то NX-сервер будет показывать только завершённые сессии, и вы не сможете возобновлять или завершать запущенные сессии, в этом случае для автоматического возобновления сессии рекомендуется установить значение параметра **ENABLE_AUTORECONNECT="1"** в файле **07-misc.conf**.

NX_ACL_DIR="/etc/nxserver/acls" — если значение этого параметра не пустое и указан существующий каталог, то система ACL запускается следующим образом: при старте сессии (node_find_application) вызывается скрипт **nxacl.apps**, который анализирует ACL-файлы в каталоге **NX_ACL_DIR** и ищет разрешения для пользователей или групп в данной командной строке; имена ACL-файлов заданы как имена пользователей, групп или «все»; порядок поиска разрешений таков: пользователь — группа — все; с более подробной информацией по параметру **NX_ACL_DIR** можно ознакомиться в файле **$NX_ETC_DIR/acls/README**.

Примечание: ACL или Access Control List — список контроля доступа, который определяет, кто или что может получать доступ к конкретному объекту, и какие именно операции разрешено или запрещено этому субъекту проводить над объектом (материал взят из Википедии).

• Файл 03-logging.conf

В файле содержатся настройки ведения логов.

NX_LOG_LEVEL=0 — уровень ведения логов: 0 — без записи; 1 — только ошибки; 2 — предупреждения; 3 — важная информация; 4 — подключения клиентов к серверу; 5 — вся информация; 6 — отладочная информация; 7 — ошибки приложений, выводимые в стандартный поток.

В процессе настроек рекомендуется использовать максимальный уровень 7, когда все настройки выполнены и система работает в штатном режиме, логи можно отключить (уровень 0).

NX_LOG_SECURE=1 — включение защиты логов. Если защиту отключить, прописав 0, то скорость работы NX-сервера немного повысится, но пароли пользователей будут доступны в файле

журнала в открытом виде.

NX_LOGFILE=/var/log/nxserver.log — путь до файла логов. Перед включением логов убедитесь, что этот файл существует и доступен для чтения и записи пользователю *nx*.

SESSION_LOG_CLEAN=1 — очистка каталога *$HOME/.nx/C-<hostname>-<display>-<session_id>* от лог-файлов сессий после завершения сеанса. При успешном завершении сессии лог-файлы сохраняются с именем *T-C-<hostname>-<display>-<session_id>*, при завершении с ошибками — *F-C-<hostname>-<display>-<session_id>*. По умолчанию очистка каталога включена.

SESSION_HISTORY=2592000 — время хранения файлов истории сессий (в секундах). Значение по умолчанию 2592000 секунд соответствует примерно 30 дням; значение 0 отключает сохранение истории сессий, а отрицательное значение включает постоянное хранение.

• Файл 04-forwarding.conf

В файле содержатся настройки перенаправления между серверами и балансировке нагрузки.

Чтобы использовать данные настройки, необходимо сначала сгенерировать ключи, при помощи которых будет происходить связь между главным и второстепенными серверами.

Генерируем ключи на главном сервере:

```
sudo nxsetup --generate-admin-keys
```

В текущем каталоге появится ключ **nxadmin.id_dsa.key.pub**, который необходимо скопировать на второстепенные сервера, а затем на каждом из них выполнить команду:

```
sudo nxsetup --setup-slave-key=/путь/до/nxadmin.id_dsa.key.pub
```

Также на второстепенных серверах необходимо переименовать скрипт расчёта нагрузки:

```
sudo mv /usr/bin/nxcheckload.sample /usr/bin/nxcheckload
```

Материал взят с сайта etersoft.ru.

После проделанных действий можно проводить настройки перенаправления и балансировки.

Настройки перенаправления соединений.

ENABLE_SERVER_FORWARD="0" — включение режима автоматического перенаправления соединений; данное решение позволит иметь цепочку из нескольких NX-серверов, но при этом для всех соединений необходимо использовать SSL-шифрование; по умолчанию отключено.

SERVER_FORWARD_HOST="" — вторичные сервера, на которые будут перенаправляться все соединения от основного сервера.

SERVER_FORWARD_PORT=22 — порт, на который будут перенаправляться все соединения от основного сервера.

SERVER_FORWARD_KEY="/usr/NX/share/client.id_dsa.key" — секретный ключ шифрования, используемый серверами.

Настройки балансировки нагрузки между серверами.

Перед тем, как использовать балансировку нагрузки, необходимо выполнить одну из настроек:

а) настроить перенаправление всех соединений, поступающих на главный сервер, на вторичные сервера;

б) открыть в общий доступ базу данный сессий посредством NFS (не рекомендуется на данный момент из-за возможности «состояния гонки дисплеев»; что это за состояние, я не разобрался).

LOAD_BALANCE_SERVERS="" — сюда необходимо прописать все сервера, между которыми будет производиться балансировка нагрузки.

LOAD_BALANCE_ALGORITHM="random" — алгоритм, используемый для балансировки нагрузки между серверами; существуют три алгоритма:

"load" — каждый сервер вызывает скрипт **nxcheckload**, который возвращает текущее значение нагрузки; по умолчанию, скрипт считает нагрузку в виде суммы средней нагрузки на процессор, свободной памяти, произведении мощности ядер на их количество и, наконец, числу запущенных X-сессий, в итоге будет выбран тот сервер, значение **nxcheckload** которого будет наименьшим;

"round-robin" — этот алгоритм циклически перебирает сервера из списка

LOAD_BALANCE_SERVERS; если алгоритм взял последний элемент списка, то следующим станет первый, и так по кругу;

"random" — логика поведения аналогична алгоритму **load**, только в данном случае каждый сервер вернет случайное значение нагрузки.

ENABLE_LOAD_BALANCE_PREFERENCE="0" — данный параметр позволяет пользователям выбирать предпочитаемый сервер при подключении, но при использовании автоматической балансировки нагрузки он игнорируется алгоритмами балансировки.

• Файл 05-sound.conf

Файл содержит настройки воспроизведения звуков в терминальной сессии

ENABLE_ESD_PRELOAD="0" — включение воспроизведения звука при помощи EsounD; если установить значение **"1"**, NX-сервер будет пытаться автоматически установить звук, используя ESD Media Helper; в настоящее время данная установка работает только в NX-клиенте под Windows.

ESD_BIN_PRELOAD="esddsp" — исполняемый файл запуска звукового сервера EsounD; убедитесь, что этот файл существует, находится в нужном каталоге и выполняется, при включении директивы **ENABLE_ESD_PRELOAD**.

ENABLE_ARTSD_PRELOAD="0" — включение воспроизведения звука при помощи звукового синтезатора aRts (входил когда-то в состав KDE); если установить значение **"1"**, NX-сервер будет пытаться автоматически установить звук, используя ARTSD Media Helper; в настоящее время данная установка работает только в NX-клиенте под Linux.

ARTSD_BIN_PRELOAD="artsdsp" — исполняемый файл запуска звукового синтезатора aRts; убедитесь, что этот файл существует, находится в нужном каталоге и выполняется, при включении директивы **ENABLE_ARTSD_PRELOAD**.

• Файл 06-path.conf

Файл содержит настройки путей к различным каталогам и исполняемым файлам на сервере.

USER_FAKE_HOME=$HOME — базовый каталог в профиле каждого пользователя, в котором располагается каталог **.nx**; данный параметр рекомендуется использовать в случаях, когда домашние каталоги пользователей расположены в общем доступе на сервере NFS; обратите внимание, что каталог для каждого пользователя должен быть уникален, поэтому рекомендуется использовать переменную $USER при написании полного пути.

DEFAULT_X_WM="" — команда запуска оконного менеджера по умолчанию; выполняется в том случае, когда в настройках клиента установлено значение UNIX-CUSTOM; значение по умолчанию пустое (без запуска менеджера окон);

KILL_DEFAULT_X_WM="1" — при включении этого параметра работа оконного менеджера будет прекращена сразу после завершения запущенной программы, в противном случае NX-сервер будет ожидать завершения работы оконного менеджера.

USER_X_STARTUP_SCRIPT=.Xclients — файл сценария запуска X-сервера в терминальной сессии пользователя; в зависимости от дистрибутива это может быть **.Xclients**, **.xinitrc** или **.Xsession**.

DEFAULT_X_SESSION=/etc/X11/xdm/Xsession — скрипт запуска X-сеанса; в зависимости от дистрибутива это может быть **/etc/X11/xdm/Xsession**, **/etc/X11/Sessions/Xsession** или **/etc/X11/xinit/xinitrc**.

BOOTSTRAP_X_SESSION="0" — в некоторых дистрибутивах перед запуском сессии необходимо выполнить скрипты, чтобы получить готовую среду; установите значение **"1"**, чтобы команда из ключа **DEFAULT_X_SESSION** выполнялась перед запуском сессии.

COMMAND_START_KDE=startkde4 — скрипт запуска оболочки KDE4; выполняется в том случае, когда в настройках клиента установлено значение UNIX-KDE.

COMMAND_START_GNOME=gnome-session — скрипт запуска оболочки Gnome; выполняется в случае, когда в настройках клиента установлено значение UNIX-GNOME.

COMMAND_START_CDE=cdwm — скрипт запуска оболочки CDE (используется на системах Solaris); выполняется в случае, когда в настройках клиента установлено значение UNIX-CDE.

ПРИМЕЧАНИЕ: поскольку этот ключ в системах Linux бесполезен, его можно использовать для запуска в терминальных сессиях других оболочек, например, XFCE или LXDE.

COMMAND_XTERM=xterm — команда запуска эмулятора терминала Xterm; выполняется в случаях, когда в настройках клиента установлено значение XTERM.

COMMAND_XAUTH=/usr/bin/xauth — команда запуска утилиты авторизации Xauth.

COMMAND_CUPSD=/usr/sbin/cupsd — команда запуска демона сервера печати Cupsd.

COMMAND_MD5SUM="md5sum" — команда запуска утилиты генерации контрольных сумм MD5.

COMMAND_RDESKTOP=rdesktop — команда запуска RDP-клиента Rdesktop.

COMMAND_VNCVIEWER=vncviewer — команда запуска VNC-клиента VNC-Viewer.

COMMAND_VNCPASSWD="$PATH_BIN/nxpasswd" — команда запуска утилиты изменения паролей на VNC-сервере VNC-Passwd; по умолчанию используется встроенный NX-Passwd.

COMMAND_X11VNC=x11vnc — команда запуска VNC-сервера X11VNC.

COMMAND_TASKSET=taskset — команда запуска утилиты Taskset, позволяющей ограничивать программам количество используемых ядер процессора.

• Файл 07-misc.conf

В файле содержатся различные настройки, не подпадающие под другие группы.

ENABLE_AUTORECONNECT="0" — разрешить автоматические переподключение; при установке значения **"1"** сервер будет автоматически возобновлять случайно оборванные сессии.

EXPORT_USERIP="0" — включение экспорта IP-адреса пользователя в NX-node.

EXPORT_SESSIONID="1" — включение экспорта идентификатора сессии в NX-node.

NODE_AUTOSTART="" — этот параметр может быть присвоен любому исполняемому файлу, запускающемуся после старта сессии, как *$NODE_AUTOSTART {start | restore}*.

ENABLE_ROOTLESS_MODE="1" — разрешить использование бескорневого режима (rootless); по умолчанию включено.

ENABLE_USESSION="1" — разрешить записи в базы данных журналов *utmp/wtmp/lastlog* при помощи команды **COMMAND_SESSREG**; для корректной работы этой функции убедитесь, что пользователь **nx** входит в группу **utmp** или **tty**, либо команда вызывается перед началом работы данной функции.

COMMAND_SESSREG="sessreg" — команда записи данных в базы данных журналов *utmp/wtmp/lastlog*.

ENABLE_EXTERNAL_NXDESKTOP="0" — разрешить использование внешней команды **'rdesktop'**; для включения функции необходимо установить значение **"1"**; если исполняемый файл **nxdesktop** не будет найден, то значение **"1"** будет установлено автоматически.

ENABLE_EXTERNAL_NXDESKTOP_KEYBOARD="1" — эта опция определяет, с каким параметром для определения раскладок клавиатуры должна запускаться команда **'rdesktop'**: либо с ключом **-k** (раскладка клавиатуры берётся из настроек сервера), либо раскладка будет определяться автоматически.

ENABLE_EXTERNAL_NXVIEWER="0" — разрешить использование внешней команды **'nxviewer'**; для включения функции необходимо установить значение **"1"**; если исполняемый файл **nxviewer** не будет найден, то значение **"1"** будет установлено автоматически.

NODE_APP_WAIT_TIMEOUT="5" — время задержки в секундах перед вызовом прекращения работы NX-агента.

SESSION_TTL=0 — время жизни приостановленной сессии (в секундах); для использования этой функции необходимо установить значение больше 0 и раскомментировать строку в файле **/etc/cron.d/rx-etersoft**.

CRITICAL_TIME=7200 — время в секундах, после которого сессия будет завершена принуди-

тельно; используется только с включенной опцией **SESSION_TTL**.

• Файл 08-bash.conf

В файле находятся настройки выполнения скриптов BASH в профиле пользователя.

ENABLE_SOURCE_BASH_PROFILE="1" — включение данного параметра означает, что сервер будет выполнять скрипт **~/.bash_profile** перед запуском приложений после входа в систему; по умолчанию включено.

ENABLE_SOURCE_PROFILE="1" — включение данного параметра означает, что сервер будет выполнять скрипт **/etc/profile** перед запуском приложений после входа в систему; по умолчанию включено.

• Файл 09-cups.conf

В файле содержатся настройки печати.

Следующие настройки выполняются в том случае, когда на NX-сервере в качестве рабочего окружения применена оболочка KDE и отключен параметр **ENABLE_CUPS_SERVER_MODE** (значение **"0"**).

ENABLE_KDE_CUPS="0" — при установке значения **"1"** NX-сервер будет автоматически записывать файл $KDE_PRINTRC и помещать его в текущий используемый сокет.

ENABLE_KDE_CUPS_DYNAMIC="0" — дополнительный параметр, устанавливающий запись хоста в скрипт nxcups-gethost, который динамически пробует все возможные записи для нахождения текущего хоста печати; запрос выполняется в следующем порядке: *CUPS_SERVER (env var)*, *~/.cups/client.conf*, *$KDE_PRINTRC*, *$CUPS_DEFAULT_SOCK*, *localhost*; этот параметр будет полезен с включенной опцией *ENABLE_CUPS_SERVER_EXPORT="1"*.

KDE_PRINTRC="$KDEHOME/share/config/kdeprintrc" — путь к файлу $KDE_PRINTRC; если значение не установлено, то файл $KDE_PRINTRC рассчитывается автоматически.

ENABLE_CUPS_SERVER_EXPORT="1" — автоматический экспорт переменной среды *CUPS_SERVER*.

ENABLE_CUPS_SEAMLESS="0" — автоматическая загрузка необходимых PPD-файлов с клиентского компьютера на сервер; для корректного проброса, который активируется при запуске NX-агента, необходима небольшая задержка **CUPS_SEAMLESS_DELAY**; на стороне клиента необходимо использовать пропатченный **cupsd**.

CUPS_SEAMLESS_DELAY="10" — время задержки в секундах перед стартом NX-агента, необходимое для корректного проброса PPD-файлов с клиентского компьютера на сервер.

ENABLE_FOOMATIC="1" — использование базы Foomatic; с включенным параметром (значение **"1"**) NX-сервер будет интегрировать базу драйверов Foomatic в список доступных драйверов PPD при помощи команды *$COMMAND_FOOMATIC*.

COMMAND_FOOMATIC="/usr/bin/foomatic-ppdfile" — полный путь к исполняемому файлу **foomatic-ppdfile** (данный файл содержится в пакете **foomatic-db-engine**).

В следующих параметрах необходимо прописать пути к файлам и каталогам, соответствующие установкам сервера печати CUPS в вашем дистрибутиве.

CUPS_BACKEND="/usr/lib/cups/backend" — путь к каталогу **backend**.

CUPS_IPP_BACKEND="$CUPS_BACKEND/ipp" — путь к исполняемому файлу **ipp**.

CUPS_DEFAULT_SOCK="/var/run/cups/cups.sock" — путь к файлу **cups.sock**.

CUPS_ETC="/etc/cups" — путь к каталогу настроек сервера печати CUPS.

CUPS_PidFile="1" — использование PID-файла для сервера печати CUPS; по умолчанию **"1"** (использовать).

CUPS_LogLevel="info" — уровень ведения логов сервера печати CUPS в режиме пользователя;

лог-файл располагается в каталоге **~/.nx/<sess_id/cups/log/**; возможные значения: **"debug"** (режим отладки) и **"info"** (режим информации); по умолчанию включен **"info"**.

ENABLE_CUPS_SERVER_MODE="0" — разрешить использование сервера печати CUPS совместно с **sudo**, чтобы пользователь мог печатать через системный CUPS; в отключенном состоянии (значение **"0"**) для каждой сессии запускается собственный процесс CUPS от имени пользователя, в который после диалогов добавляется драйвер принтера в каталог **/etc/nxserver/ppd**; при включении параметра (значение **"1"**) используется системный **cupsd** для проброса клиентских принтеров на сервер и автоматическая установка драйверов для них.

ENABLE_CUPS_DIALOG="0" — включение диалога настроек для старых принтеров.

• Файл 10-samba.conf

В файле содержатся параметры проброса каталогов посредством Samba.

ENABLE_SAMBA_PRELOAD="0" — установка параметров Samba перед запуском сессии; если установить значение **"1"**, то NX-сервер перед запуском сессии автоматически настроит порты 445 и 139 и будет перенаправлять запросы с них на нужный порт сервера Samba; это позволит просматривать пробрасываемые ресурсы в локальной сети, например, при помощи Konqueror.

SMB_MOUNT_OPTIONS="iocharset=utf8,file_mode=0660,dir_mode=0770" — дополнительные параметры монтирования для команды **mount.cifs**; значение по умолчанию пустое, но для корректного отображения кириллических символов можно прописать, например, *SMB_MOUNT_OPTIONS="iocharset=utf8,codepage=cp866,file_mode=0660,dir_mode=0770"*

SHARE_FAST_MOUNT="0" — включение или отключение монтирования в фоновом режиме; в последних версиях не поддерживается.

ENABLE_SHARE_MULTIMOUNT="0" — патч от **Dimbor**'а, позволяющий переподключать проброшенные каталоги в случае запуска нескольких сеансов с одного компьютера; при установке значения **"0"** в случае нескольких терминальных сессий, запущенных с одного компьютера, многократный проброс ресурсов может не работать или работать некорректно, а при однократном пробросе после завершения сессии ресурсы могут потеряться для других сессий; при установке значения **"1"** NX-сервер перед закрытием одной сессии проводит поиск другой активной сессии и переподключение проброшенных каталогов на контролируемый ею порт.

WIN_CP_CONVERT_CHAIN=">cp1252 cp1251>" — конвертирование кодировок для корректного отображения международных символов в именах ресурсов NX-клиента под Windows; параметр может быть пустым или содержать одну или несколько пар кодировок — аргументов типа *"[from]>[to]"* для команды **iconv**; если параметр пустой, то это эквивалентно системной кодировке; пример использования: *">cp1252 cp1251>" == "UTF-8>cp1252 cp1251>UTF-8"*.

• Файл 11-nxagent.conf

Дополнительные параметры настройки NX-Agent; возможные значения описаны в документации NoMachine.

AGENT_EXTRA_OPTIONS_RFB=""
AGENT_EXTRA_OPTIONS_RDP=""
AGENT_EXTRA_OPTIONS_X="-nolisten tcp -dpi 96"

AGENT_STARTUP_TIMEOUT="60" — время ожидания (в секундах) NX-агентом от начала сессии, после которого будет выдана ошибка подключения.

AGENT_FONT_SERVER="" — сервер шрифтов, используемый NX-агентом; по умолчанию пустое значение, означающее, что сервер не используется; для корректной работы необходимо, чтобы клиенты имели тот же набор шрифтов, что и в файле сервера шрифтов **/etc/X11/XF86Config**.

• Файл 12-nxproxy.conf

Файл содержит настройки параметров NX-Proxy.

PROXY_TCP_NODELAY="" — включение или отключение использования TCP_NODELAY на

прокси-сервере. В старых ядрах Linux наблюдаются проблемы с этой опцией в сокетах, что может привести к обрывам соединений. Этот параметр не задан по умолчанию, чтобы клиенты сами решили, нужно ли его использовать. Отключение этого параметра (значение **"0"**) позволяет не использовать TCP_NODELAY, но может привести к потере взаимодействия в сессиях.

PROXY_EXTRA_OPTIONS="" — дополнительный опции NX-Proxy. Возможные значения описаны в документации NoMachine.

• Файл 50-numlockx.conf

В файле содержатся настройки поведения клавиши NumLock при подключении к терминальной сессии.

NUMLOCKX=numlockx — команда для управления включением клавиши NumLock (должен быть установлен соответствующий пакет).

NUMLOCK_METHOD="system" — параметры управления клавишей NumLock в терминальной сессии: **"on"** — клавиша автоматически включается при старте сессии; **"off"** — клавиша автоматически отключается при старте сессии; **"system"** — используются текущие системные параметры без запуска программы *numlockx*.

1.1.3. Установка и настройка клиента FreeNX.

На сегодняшний день существует несколько версий клиентов, работающих по протоколу NXv3. Оригинальный клиент версии 3.5.0 от компании NoMachine для загрузок больше не доступен, но для операционной системы Windows есть сборка FreeNX Client Community Edition, созданная силами сообщества Unixforum'a, в которой устранено очень много проблем и повышено быстродействие. Компания «Этерсофт» сделала установочный комплект этой сборки, скачать который можно с FTP-сервера.

Сборки для системы Linux доступны в репозиториях некоторых дистрибутивов и на FTP-сервере «Этерсофта» наряду с серверной частью.

Сейчас активно развивается проект OpenNX с открытым исходным кодом, компания «Этерсофт» также выпустила сборку на его основе, и именно её рекомендуется использовать в качестве клиента для терминального сервера RX@Etersoft. Другие сборки OpenNX также доступны в репозиториях некоторых систем.

Существует ещё несколько клиентов с открытым исходным кодом: kNX, QtNX, но они не получили широкого распространения. И наконец для мультипротокольного клиента Remmina в репозиториях доступен пакет **remmina-plugin-nx**, позволяющий этому клиенту подключаться к серверу FreeNX.

Основной принцип работы, внешний вид и назначение меню у всех клиентов примерно одинаковое, но если OpenNX и другие клиенты с открытым исходным кодом хоть как-то русифицированы, то оригинальный NX-Client от компании NoMachine и сборки на его основе идут только на английском языке, поэтому рассмотрим установку и настройку именно его.

Итак, идём на FTP-сервер компании «Этерсофт», открываем папку **/stable**, далее выбираем наш дистрибутив и его версию и скачиваем два пакета: ***nx*** или ***opennx*** и ***nxclient***, и устанавливаем их.

После установки запускаем мастер настройки ***NX Connection Wizard***. Если при первом запуске клиент выдаст сообщение, что не может найти некоторые модули, и попросит указать папку, где они расположены, прописываем путь **/usr** либо **/usr/lib/nxclient**.

Настраиваем сеанс: в поле *Session* вводим имя сессии (например, *Terminal*), в строке *Host* указываем адрес сервера, в строке *Port* — порт SSH (по умолчанию 22), и выбираем скорость соединения (рекомендую пока что оставить по умолчанию *ADSL*). В следующем окне указываем систему *Unix* и графическую среду, используемую на сервере, значение разрешения экрана можно оставить по умолчанию *«Available area»*, галочку *«Disable encryption of all traffic»* тоже пока не ставим. А в следующем окне устанавливаем обе галочки *«Create shortcut on desktop»* (создать ярлык сессии на рабочем столе) и *«Show the Advanced Configuration dialog»* и переходим в режим расширенных настроек.

В первой вкладке — *General* — проверяем все наши настройки, в секции *Server* ставим галочку

"*Remember my password*", чтобы не набирать свой пароль постоянно при входе. Если при настройке сервера вы сгенерировали собственный ключ, то его нужно импортировать, нажав *«Key»*, затем *«Import»,* выбрать файл ключа *client.id_dsa.key*, который был скопирован с сервера, и далее *«Save»*. Для сброса ключа на значение по умолчанию нажимаем *«Default»* и *«Save»*. Если же на сервере был сгенерирован ключ по умолчанию, то ничего импортировать не нужно.

В секции *Desktop* устанавливаем скорость соединения. Данный параметр выбирайте из расчёта: чем ниже скорость соединения, тем ниже качество изображения, меньше нагрузка на сеть и выше скорость отрисовки окон. Для низкоскоростного подключения через интернет желательно выбрать *MODEM* или *ISDN*; для «слабых» компьютеров рекомендую установить скорость ADSL, поскольку в этом случае клиент будет меньше потреблять ресурсов. Как показала практика, настройки ADSL вполне хватает для комфортной работы и в высокоскоростных локальных сетях. При выборе скорости WAN или LAN нагрузка на локальную сеть будет выше, поскольку уровень сжатия трафика будет меньше. Хотя при нынешних скоростях локальных сетей минимум в 100 МБит/с это уже не играет большой роли, но при большом количестве компьютеров, подключаемых одновременно к серверу, этими параметрами лучше не злоупотреблять.

В этой же секции в типе сеанса по умолчанию используется *Unix*, но клиент также может подключаться к Windows-серверу по протоколу RDP, VNC-серверу по протоколу RFB, а также к уже запущенной терминальной сессии другого пользователя аналогично VNC-серверу (параметр *Shadow*). В настройках сеанса VNC нужно указать только адрес сервера, порт (по умолчанию 5900) и пароль пользователя. Для подключения к терминальному серверу Windows в настройках указываем адрес сервера и домен, в секции *Authentication* указываем параметры авторизации: использовать логин и пароль пользователей NX, задать логин и пароль вручную либо показывать при старте сессии экран приветствия, чтобы пользователь мог сам ввести свой логин и пароль. В секции *Session Type* указываем либо загрузку рабочего стола в терминальной сессии, либо команду запуска нужного приложения без загрузки рабочего стола.

На настройках сеанса *Unix* остановимся подробнее. Для него доступны оболочки *KDE*, *GNOME*, *CDE* (для которых какие-либо настройки отсутствуют), *XDM* и ручная настройка *Custom*.

Графическая оболочка CDE используется в системах Solaris и под Linux'ом она бесполезна, но вместо неё можно использовать другие оболочки, например, XFCE, LXDE, MATE или Unity, если в настройках сервера в строке **COMMAND_START_CDE** прописать команду запуска соответствующей оболочки, а в клиенте — выбрать оболочку CDE.

Параметр XDM позволяет подключаться к серверу по протоколу XDMCP, описание которого приведено в разделе 3.

Параметр *Custom* позволяет задать настройки запуска терминальной сессии в «бесшовном» режиме. В секции *Application* можно выбрать один из трёх параметров: запуск консоли, запуск клиентского скрипта **XSession** на сервере или запуск определённой команды, а в секции *Options* — дополнительные параметры: *New Virtual Desktop* (новый виртуальный рабочий стол) — запуск приложения без загрузки рабочего стола, в этом режиме программа запускается внутри чёрного родительского окна терминальной сессии; *Floating Window* — запуск приложения в «бескорневом» режиме (rootless), когда программа запускается без загрузки графической оболочки и родительского окна, создаётся ощущение, как будто программа запущена на локальном компьютере. Для этого параметра доступны ещё две дополнительные настройки: *Disable X agent encoding* и *Disable taint of X replies*. Эти настройки позволяют отключить кодирование трафика NX-агентом и передавать его в «сыром» режиме, что улучшает сжатие, но при этом страдает внешний вид программ.

Пример настройки NX-клиента в режиме rootless приведён в конце этого раздела.

Теперь вернёмся к настройкам клиента во вкладке *Genera*l, а именно — к последней секции *Display*. Здесь рекомендую сделать следующие настройки:

1) чтобы при разрешении на полный экран панель меню в терминальной сессии не выходила за аналогичную панель на локальном компьютере, в окне разрешения экрана выбираем параметр *Custom* и выставляем разрешение: по горизонтали — полное разрешение монитора, по вертикали — на 50 пикселей меньше от полного разрешения. Например, если текущее разрешение монитора составляет 1366x768, то в терминальной сессии размер окна выставляем 1366x718.

2) при подключении через низкоскоростной интернет рекомендую выполнить дополнительные на-

стройки передачи данных: ставим галочку «*Use custom settings*», затем в меню *Settings* в секции *Images* устанавливаем параметры компрессии рисунков и цвета; рекомендую сделать все по минимуму, в частности установить галочки «*Use both JPEG and RGB compression*» (использование JPEG и цветового сжатия) и «*Use custom JPEG Quality*» (настройка уровня сжатия JPEG вручную), а ползунок — на 0. При этом немного ухудшится качество рисунков, но скорость отрисовки окон возрастёт. Параметр *Only use RGB compression* рекомендуется использовать на высокоскоростных соединениях, поскольку высокое качество изображений увеличивает объём передаваемых данных.

Параметр *Use Plain X bitmaps* полностью отключает сжатие картинок и передаёт изображения и цвета как есть, поэтому его рекомендуется использовать только на высокоскоростных соединениях или в локальной сети.

В секции *Perfomance* по желанию можно включить дополнительные параметры:

• *Disable the render extension* — отключение расширения Render, бывает необходимым в случаях, когда нужно перенести сессию на дисплей X-сервера с отсутствующими расширением Render.

• *Disable the backing-store* — отключение использования внешней памяти, снижает производительность системы в целом, но бывает полезно в случаях, когда нужно снизить потребление памяти на стороне сервера или клиента;

• *Disable the composite extension* — отключение использования расширения Composite X-сервера NX-агентом;

• *Disable the shared memory extension* (отключение расширения Shared Memory) и *Disable emulation of shared pixmap* (отключение эмуляции Shared Pixmap) — отключение этих параметров увеличивает производительность системы, поскольку пиксельные изображения, создаваемые X-сервером не могут быть общими, но должны постоянно обновляться; но если какое-либо приложение для своей работы использует пиксельные изображения, то оно может не запуститься.

Во вкладке *Advanced* в секции *Network* можно указать различные параметры соединения и передачи данных: *«Disable Encryption of all traffic»* — отключение шифрования траффика, немного повышает скорость работы, но в целях безопасности требует определённых настроек на сервере, разрешающих подключение только внутри локальной сети; *«Disable ZLIB stream compression»* — отключение сжатия траффика, позволяет снизить нагрузку на процессор на сервере и повысить быстродействие, но нагрузка на сеть увеличится; *«Connect through a HTTP proxy»* — подключение через прокси-сервер (если таковой имеется).

В секции *System* устанавливаем количество оперативной памяти и кэша на диске, которые будут отводиться на сеанс (это очень полезно для "слабых" машин). Параметр *«Disable deferred screen updates»* (отключение отложенных обновлений экрана) позволяет изменить уровень шифрования; по умолчанию для различных соединений используется разный уровень шифрования: для WAN — уровень 1, для MODEM, ISDN или ADSL — уровень 2; активация этого режима позволит использовать уровень 1 вместо уровня 2; для соединений LAN этот параметр не работает.

В NX-клиенте для систем Windows есть ещё два параметра: *«Grab the keyboard when the client has focus»* — позволяет отключить захват клавиш Alt+Tab и PrintScreen для использования их в терминальной сессии, и «*Disable DirectDraw for screen rendering»* — отключение DirectDraw и использование вместо него GDI; эта опция бывает полезна для решения возможных конфликтов, например, из-за одновременного использования DirectDraw NX-клиентом и другими приложениями, из-за чего процессор загружается на 100%.

В третьей вкладке — *Services* — в секции *Devices* для проброса принтеров и локальных папок на сервер ставим галочки «*Enable SMB printing and file sharing»* и «*Enable CUPS printing».* Для работы первого режима папки и принтеры на клиентском компьютере должны быть открыты в общий доступ по протоколу SMB/CIFS, для второго — только принтеры по протоколу IPP, а также нужно установить права у файла *ipp*:

```
sudo chmod 755 /usr/lib/cups/backend/ipp
```

На стороне сервера должны быть выполнены соответствующие настройки.

Далее нажимаем *Add* и выбираем ресурсы для проброса в терминальную сессию из списка. В секции *Multimedia* можно включить трансляцию звука из терминальной сессии на локальный компьютер, но лучше этого не делать, потому что скорость работы терминальной сессии заметно снизится.

Хочется заметить, что проброс ресурсов — это один из «больных» вопросов сервера FreeNX.

Например, проброс папок может не работать, для проброшенных принтеров может потребоваться установка драйверов при каждом подключении. Со звуком тоже возникают проблемы, если в качестве звуковой системы используется PulseAudio. Решение проблемы со звуком описано на форуме unixforum.org, там же доступны необходимые скрипты.

В следующей вкладке — *Environment* — всё оставляем как есть, но в в случае отклонений от настроек по умолчанию прописываем остальные параметры для NX-клиента: в секции *User NX Directory* указываем каталог, куда будут сохраняться временные файлы от сессий, галочку *«Remove old session files»* оставляем включенной; в секции *System NX Directory* указываем путь к директории, в которой хранятся файлы и библиотеки клиента, в *System CUPS Daemon* — расположение исполняемого файла демона системы печати CUPS. Здесь же меняем шрифты заголовков и сообщений в терминальных сессиях.

После всех проделанных действий на рабочем столе вы увидите ярлык на файл *<имя_сессии>.nxs*. Запустить сеанс можно, нажав два раза мышкой на этот ярлык. При первом старте клиент выдаст запрос на импорт публичного ключа сервера SSH (если вы конечно не отключали *PubkeyAuthentication*), а затем сервера NX, в обоих случаях нажимаем *Yes*.

В случае разрыва соединения при повторном подключении NX-клиент выдаст сообщение, что ваш сеанс уже запущен. В этом случае у вас есть два варианта: продолжить сеанс с места разрыва, выбрав сеанс и нажав *Resume,* либо завершить запущенный ранее сеанс и начать новый, выбрав сеанс и нажав *Terminate*, затем, когда он исчезнет из списка, нажать *New*.

Как уже было сказано выше, NX-клиент может работать в режиме *Rootless*, когда при подключении к серверу программа стартует сразу без загрузки графической оболочки, при этом создается ощущение, что она запущена локально на компьютере пользователя. К сожалению этот режим довольно «капризный», может работать нестабильно, при этом потребляет больше ресурсов клиентского компьютера. Для полноценной работы этого режима рекомендуется компьютер с 64-битным процессором и минимум 1ГБ оперативной памяти.

Рассмотрим пример запуска программы 1С в этом режиме. Для начала на сервере создаём скрипт со следующим содержимым:

```
#!/bin/bash
# запускаем все в одной оболочке
/bin/bash << EOF
# если возникают проблемы с клавиатурой, то раскомментируем строку ниже:
# /etc/nxserver/fixkeyboard
# настройка переключения с русской клавиатуры на английскую через Ctrl+Shift
setxkbmap -rules xorg -model pc105 -layout "ru,us" -variant "winkeys," -option
"grp:ctrl_shift_toggle,grp_led:scroll"
# установка точки вместо запятой на дополнительной клавиатуре
xmodmap -e "keycode 91 = KP_Delete KP_Decimal KP_Delete KP_Decimal"
# Команда запуска 1С:
env WINEPREFIX=$HOME"/.wine" wine "C:\Program Files\1Cv77\BIN\1cv7.exe" enterprise &
#
EOF
```

Теперь сохраняем скрипт под любым именем, например **term1s**, и присваиваем права **755**. Затем копируем скрипт в домашний каталог каждого пользователя либо в отдельный общий каталог, доступный всем пользователям.

На клиентском компьютере запускаем NX-Client, нажимаем "*Configure*" и изменяем настройку «*Desktop»* на *Custom*; далее выбираем «*Settings*», ставим галочку «*Run the following command*» и в строке ниже прописываем путь к скрипту: **/home/<username>/term1s**. Следим, была включена настройка «*Floating window*».

Сохраняем настройку, вводим логин/пароль и любуемся на 1С в обрамлении системных окошек.

Оригинальные варианты описанного выше скрипта приведены на форумах forum.ru-board.com и unixforum.org.

Более подробное описание NX-клиента версии 3.5 на английском языке можно найти в архивных копиях сайта NoMachine.

1.1.4. Возможные проблемы и способы их решения.

Теперь рассмотрим проблемы, которые могут возникнуть при подключении клиента к серверу.

1) Если при первом подключении выдаётся сообщение "*The NX service is not available or the NX access was disabled on host ...*", жмём *Detail* и смотрим:

```
...
NX> 200 Connected to address: 192.168.1.2 on port: 22
NX> 202 Authenticating user: nx
NX> 208 Using auth method: publickey
NX> 204 Authentication failed.
```

Причина: либо вы не импортировали в NX-клиент ключ *client.id_dsa.key* с сервера, как это было описано выше, либо на сервере был сгенерирован новый ключ. Возвращаемся к первому разделу и читаем про ключи.

2) В процессе подключения может выдаваться сообщение "*Authentification failed for user <имя>*", но при локальном входе под именем этого же пользователя авторизация проходит нормально. Значит нас не пускает SSH-сервер. Возвращаемся к самому началу шага 1, где описана необходимая настройка SSH-сервера, и выполняем требования.

3) При подключении клиент "зависает" на стадии "*Negotiating link parameter*", после чего пишет, что не может подключиться и предлагает завершить сессию. Причина такого поведения: в клиенте во вкладке *Advanced* стоит галочка "*Disable encryption of all traffic*" (о чем было сказано выше). Выход: либо настройте NX-сервер на передачу данных в незашифрованном виде, либо снимите галочку.

4) В системах Linux при включенном менеджере окон Compiz NX-клиент "зависает": окно сеанса с запущенной программой темнеет, однако все меню открываются нормально, и можно даже выйти из запущенного сеанса. Поэтому не рекомендуется использовать менеджер окон Compiz и NX-Client на одном компьютере.

5) При запуске NX-клиента всё проходит нормально, появляется окошко с логотипом NoMachine, загружается рабочий стол, и тут клиент вылетает с сообщением *"The connection with the remote server was shut down. Please check the state of your network connection."*

Такое поведение проявляется в случаях, когда на сервере установлена более новая версия операционных систем Linux (Ubuntu 12.10 и выше, Debian 7.0 и выше и т. д.) и старая сборка FreeNX. Как выяснилось, все версии сервера FreeNX и RX@Etersoft вплоть до версии 1.1.2 с пакетами *nx-3.5.1-eter6* и *rx-etersoft-1.1.2-eter2* не совместимы с пакетом *libcairo2* версии 1.12, для них требуется пакет версии 1.16 и выше или 1.10 и ниже.

Выход из данной ситуации: либо пересоберите пакеты с применением специального патча, либо в файле настроек сервера **/etc/nxserver/node.conf** пропишите опцию, которая решит проблему, но ухудшит качество изображения:

```
AGENT_EXTRA_OPTIONS_X="-norender -nolisten tcp"
```

Третий вариант — попробуйте поискать готовые сборки для своей версии системы, в которых данная проблема уже решена.

6) При запуске Windows-приложений под Wine в терминальной сессии клиент вылетает без каких-либо ошибок; при запуске NX-клиента через консоль и попытке выполнить те же действия в консоли появляется ошибка:

```
X Error of failed request:  BadValue (integer parameter out of range for operation)
  Major opcode of failed request:  149 (RENDER)
  Minor opcode of failed request:  34 (RenderCreateLinearGradient)
  Value in failed request:  0x0
  Serial number of failed request:  83569
  Current serial number in output stream:  83613
```

Здесь причина кроется в несовместимости Wine версий с 1.3.0 по 1.5.4 включительно с сервером FreeNX. В данном случае рекомендуется использовать либо новые версии Wine 1.5.13 и выше, либо более старые 1.2.3 или ниже. Версии с 1.5.5 по 1.5.12 не тестировались, поэтому их также не рекомендуется к использованию.

7) в старых версиях сервера существует проблема автоматического подключения локальных принтеров клиента к серверу FreeNX; в случае, если для клиентских принтеров есть «родные» PPD-

файлы для CUPS, делаем следующее:

вариант №1: скопировать PPD-файл в домашний каталог каждого пользователя в папку с сессией: **/home/<user>/.nx/<имя_сессии>/cups/ppd/<имя_принтера>.ppd**

вариант №2: в каталоге **/etc/nxserver** создаем каталог **ppd** и копируем в него файл **<printername>.ppd**; имя файла должно совпадать с именем клиентского принтера.

При подключении принтеры будут отображаться в общем списке с именем вид ***<CLIENT>_<PRINTERNAME>***.

Если «родных» PPD-файлов нет, остаётся только добавлять принтер на сервер вручную.

8) Очень много жалоб поступало, что не работает печать из терминальной сессии на клиентский компьютер. В качестве решения можно использовать следующий алгоритм: настроить проброс локальной папки на сервер, далее на сервере создать PDF-принтер и настроить его так, чтобы PDF-файлы сохранялись в проброшенной папке, а затем на клиентском компьютере полученный файл открывать и распечатывать.

9) В Windows возможна проблема при копировании текста из терминальной сессии на локальную машину, когда вместо текста копируются одни знаки вопроса; если эта проблема присутствует, то ее можно устранить, отредактировав реестр на клиентской машине: ищем ключ **[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage]**, создаём строковый параметр *(string)* с именем *1252* и присваиваем значение ***c_1251.nls***

10) Существует проблема несовместимости систем Linux между собой: если на сервере установлен Debian или Ubuntu, а на клиентских компьютерах, — например Mandriva или Fedora, то возникают проблемы с курсором, а некоторые кнопки клавиатуры не работают; проблема обсуждается на форуме unixforum.org.

В качестве выхода из такой ситуации предлагается создать скрипт **nxkeys.sh** со следующим содержимым:

```
#!/bin/bash
xmodmap -e "keycode 91 = KP_Delete KP_Decimal KP_Delete KP_Decimal"
xmodmap -e "keycode 111 = KP_Up"
xmodmap -e "keycode 114 = KP_Right"
xmodmap -e "keycode 113 = KP_Left"
xmodmap -e "keycode 116 = KP_Down"
xmodmap -e "keycode 119 = KP_Delete"
xmodmap -e "keycode 115 = KP_End"
xmodmap -e "keycode 117 = KP_Next"
xmodmap -e "keycode 112 = KP_Prior"
xmodmap -e "keycode 110 = KP_Home"
xmodmap -e "keycode 104 = KP_Enter" &
```

В данном скрипте прописаны неработающие кнопки. Затем на клиентском компьютере копируем этот скрипт в домашний каталог пользователя и прописываем в автозагрузку:

```
xmodmap /home/USERNAME/nxkeys.sh
```

11) В некоторых системах после обновления может не работать *setxkbmap* под NX-сессией, выдавая ошибку *«Error loading new keyboard description»*. Данная проблема и способ её устранения описаны на сайте bugzilla.novell.com.

Для решения проблемы создаем симлинк в нужном каталоге:

```
sudo ln -snf /var/lib/xkb/compiled /usr/share/X11/xkb/compiled
```

Решения других возможных проблем можно найти на форуме unixforum.org в разделе *Бизнес и бухгалтерия под Линукс → Терминальные решения* или на сайте ArchWiki.

Выводы по использованию FreeNX неоднозначны: с одной стороны это самый популярный терминальный сервер под Linux с самой высокой скоростью работы, с другой — некоторые заявленные функции работают нестабильно или не работают вовсе, а проблемы возникают на ровном месте, поэтому для его настройки требуются квалифицированные специалисты, способные их решить. Плюс компания NoMachine больше не поддерживает протокол NXv3, многие ветки разработки и репозитории заброшены и не обновляются, так что на текущий день по сути проект FreeNX для российских пользователей поддерживается только благодаря сообществу Unixforum'a и компании «Этерсофт», которая выпускает сборки под новые системы со всеми включенными патчами. Но в общем итоге сервер можно рекомендовать к использованию.

1.2. СЕРВЕР X2GO

Сервер X2Go — ещё один вариант реализации терминального сервера по протоколу NX от немецких разработчиков. Он имеет свою собственную серверную и клиентскую части, которые не совместимы с другими вариантами серверов и клиентов NX. Работает он чуть медленнее, чем оригинальный NX-сервер, но зато в последней версии 4.0 есть дополнительные возможности, которые пока что отсутствуют в бесплатных версиях FreeNX (например, интеграция с MS Active Directory), он проще в настройках, выше стабильность, всё начинает работать практически сразу (проброс папок, принтеров, звука, режим rootless и т. д.). Также он лучше поддерживается разработчиками, оперативно выходят сборки под новейшие версии дистрибутивов Linux, довольно быстро устраняются обнаруженные серьёзные ошибки. Хотя стоит признать, что многие патчи «перекочевали» в эту разработку из FreeNX, включая все патчи с Unixforum'a.

Возможности X2Go согласно описания:

• возобновление сессии в случае обрыва;

• поддержка каналов с узкой полосой пропускания (иными словами — поддержка работы с низкоскоростным интернетом);

• поддержка LDAP;

• поддержка монтирования накопителей с клиентского компьютера на сервер;

• поддержка проброса принтеров с клиентского компьютера на сервер;

• поддержка звука в терминальной сессии;

• поддержка авторизации по смарт-картам и USB-ключам.

На официальном сайте X2Go инструкция по установке и администрированию довольно скудная, много материала ещё в процессе написания. В интернете можно найти инструкции по установке прошлых версий X2Go, но к текущей версии 4.0 их довольно сложно применить, уж очень много всего поменялось. Поэтому описание ниже основано на собственных наработках, а также опыте других пользователей.

1.2.1. Установка и стандартная настройка сервера X2Go

Установить сервер X2Go можно двумя способами: подключить репозиторий для своей системы и установить готовые пакеты, либо скомпилировать из исходников.

В этом пункте будет рассмотрена установка и настройка стандартного набора компонентов сервера X2Go без настройки авторизации через LDAP и сервера для «тонких» клиентов.

Для установки готовых пакетов из репозиториев выполняем следующие шаги:

1) Предварительно устанавливаем SSH-сервер и другие необходимые пакеты:

```
sudo apt-get install python-software-properties openssh-server
sudo service ssh restart
```

2) Добавляем ключ авторизации (только для Debian!):

```
apt-key adv --recv-keys --keyserver keys.gnupg.net E1F958385BFE2B6E
```

3) Подключаем репозиторий:

• в Debian'e добавляем его в **/etc/apt/sources.list** вручную или через графическую оболочку:

```
# X2Go Repository
deb http://packages.x2go.org/debian stable main
# X2Go Repository (sources)
deb-src http://packages.x2go.org/debian stable main
```

• в Ubuntu выполняем в консоли команду:

```
sudo add-apt-repository ppa:x2go/stable
```

Примечание: в репозиториях X2Go доступна версия Baikal с долгосрочной поддержкой до 2015 года.

4) обновляем список пакетов:

```
sudo apt-get update
```

В Debian'e после обновлений индексов выполняем дополнительную команду:

sudo apt-get install x2go-keyring && apt-get update

5) устанавливаем серверную часть:

sudo apt-get install x2goserver x2goserver-xsession

После установки в систему автоматически добавляется пользователь и группа **x2gouser**.

6) устанавливаем дополнительные пакеты для подключения в терминальной сессии проброшенных каталогов:

sudo apt-get install x2goserver-fmbindings

7) для печати из терминальной сессии на клиентские компьютеры, подключенные через интернет, устанавливаем дополнительные пакеты:

sudo apt-get install cups x2goserver-printing cups-x2go

После установки создаём на сервере принтер Cups-X2Go: в списках доступных устройств выбираем *Generic CUPS-X2Go*, из базы данных выбираем принтер *Generic* и далее указываем драйвер *CUPS-X2Go*.

Принцип действия таков: принтер Cups-X2Go — это обычный PDF-принтер; когда в терминальной сессии на него отправляют задание на печать, формируется PDF-файл, который через SSH-порт автоматически отправляется на клиентский компьютер и далее, в зависимости от настроек клиента X2Go, распечатывается на принтер, указанный по умолчанию в настройках клиента, либо открывается для просмотра. Такое решение позволяет использовать принтеры любых моделей без необходимости установки драйверов на сервере.

Данное решение рекомендуется применять для клиентов, подключающихся к терминальному серверу через интернет. Если таких клиентов нет, то лучше использовать стандартный сервер CUPS с принтерами, открытыми в общий доступ, а указанные выше пакеты можно не устанавливать.

Вариант настройки печати, когда терминальный сервер X2Go и сервер печати CUPS расположены на разных компьютерах, описан на официальном сайте wiki.x2go.org.

8) если планируется работа клиентов через веб-интерфейс без установки клиентской части X2Go, устанавливаем пакеты дополнительные пакеты:

sudo apt-get install apache2 x2goplugin-provider

После выполненных настроек клиентский веб-интерфейс будет доступен по адресу:

http://<SERVER_IP_OR_HOSTNAME>/x2goplugin.html

Для доступа к серверу через браузер на стороне клиента должен быть установлен специальный плагин для браузеров **x2goplugin**; как его установить, читайте в разделе по установке и настройке клиента X2Go. Если с других компьютеров интерфейс окажется недоступным, то попробуйте создать символическую ссылку на файл **x2goplugin.html** в каталог **/var/www** и затем из каталога **/etc/apache2/conf.d** удалить файл **x2goplugin.conf**, который является симлинком на файл **/etc/x2goplugin-apache.conf**.

И ещё один нюанс: по умолчанию терминальные сессии через веб-интерфейс будут работать только с оболочкой XFCE4. Если у вас на сервере установлена другая оболочка, отредактируйте указанный выше файл **x2goplugin.html**, прописав в строке ***command*** команду запуска для установленной оболочки.

Инструкция по установке веб-оболочки взята с официального сайта wiki.x2go.org.

Ниже приведено краткое описание дополнительных модулей для сервера, а вы решайте сами, какие вам нужны:

• **cups-x2go** — виртуальный принтер X2Go для сервера печати CUPS; рекомендуется использовать вместе с сервисом **x2goserver-printing**;

• **x2goserver-printing** — сервер печати для X2Go, предназначен в первую очередь для работы через низкоскоростные соединения или с принтерами, подключенными к «тонким клиентам»; если сервер находится в высокоскоростной локальной сети, то этот сервис не нужен;

• **x2goserver-fmbindings** — модуль, добавляющий в терминальной сессии на рабочий стол и в файловый менеджер ярлык проброшенной через X2Go-клиент папки; данный модуль работает с любыми оболочками, но при использовании определённых графических окружений он может быть заменён другими: **plasma-widget-x2go** — для KDE4, **x2golxdebindings** — для LXDE5,

x2gomatebindings — для MATE; также есть модули для других оболочек, которые больше не поддерживаются и в репозиториях не доступны, но существуют в виде исходников: **x2gotrinitybindings** — для Trinity3.5 (форк KDE 3.5), **x2gognomebindings** — для Gnome2, **x2gokdebindings** — для KDE3.5;

• **x2goserver-compat** — пакет для совместимости старых версий клиентов X2Go с текущей версией сервера;

• **x2goserver-pyhoca** — пакет, добавляющий функциональность при использовании клиента PyHoca: переименование названий сессий в окне и более детальная настройка параметров клавиатуры; этот пакет необязателен, клиент PyHoca сможет работать и без него;

• **x2goserver-xsession** — дополнение, позволяющее серверу X2Go обрабатывать скрипт ***Xsession*** при запуске сессии;

• **x2godesktopsharing** — дополнение, позволяющее одному пользователю предоставлять доступ к текущей сессии другим пользователям (аналогично VNC Server и RAdmin);

• **x2goplugin-provider** — дополнение, позволяющее клиентам подключаться к серверу через браузер без установки клиентской части; в этом случае на стороне клиента должно быть установлено специальное дополнение **x2goplugin**.

Хочу уделить небольшое внимание ещё одному модулю — **X2Go Broker**, позволяющему централизованно управлять настройками сессий клиентов X2Go. В большинстве случаев клиенты X2Go запускаются в автономном режиме, и пользователь сам настраивает одну или несколько сессий для подключения к серверам X2Go, а все настройки сохраняются на локальных компьютерах; данный модуль позволяет хранить настроенные профили подключения непосредственно на сервере, а для получения настроек с сервера клиент X2Go должен быть запущен в режиме Broker Mode, при этом локально настроенные профили игнорируются. Это решение полезно в масштабах крупных предприятий для балансировки нагрузки. Модуль состоит из нескольких пакетов, устанавливается он командой:

```
sudo apt-get install x2gobroker x2gobroker-agent x2gobroker-authservice x2gobroker-daemon x2gobroker-wsgi python-x2gobroker
```

Поскольку в данной статье рассматривается настройка терминальных решений в масштабах небольшого предприятия, то настройка модуля **X2Go Broker** описана не будет, инструкция по его установке доступна на официальном сайте в разделе Installation X2Go, по использованию клиента X2Go в режиме Broker Mode — в разделе Advanced.

Сейчас в процессе разработки находится ещё один модуль — **X2Go Admin Center**, который позволит администрировать пользователей, группы, сессии и многое другое через графический интерфейс на Qt4. Текущую нестабильную версию можно скачать из Git-репозиториев и собрать готовый DEB-пакет:

```
git clone git://code.x2go.org/x2goadmincenter.git
cd ~/x2goadmincenter
dpkg-buildpackage -b -tc
```

После компиляции получим два пакета — **x2goadminserver** (устанавливается на сервере) и **x2goadmincenter**, (устанавливается на компьютере администратора).

На этом установка и настройка серверной части закончена. По умолчанию для подключения к серверу используются логины и пароли системных пользователей, также возможно настроить авторизацию через LDAP-сервер (Microsoft Active Directory на Windows-сервере или контроллер домена на Samba в Linux-сервере).

Установка из исходного кода описана на официальном сайте: можно скачать либо исходники стабильной текущей версии, либо нестабильные из git-репозитория. Правда придётся скачивать, компилировать и устанавливать каждый модуль отдельно, в итоге процесс установки сервера может затянуться на несколько часов.

1.2.2. Установка и настройка клиента X2Go

Существует два клиента X2Go: стандартный на базе Qt4 и PyHoca на базе Python, причём для последнего есть модули для работы в командной строке и через графическую оболочку.

Стандартный клиент работает в системах Linux, Windows и MacOS, PyHoca — только для Linux и Windows.

В операционных системах Linux клиентская часть устанавливается аналогично серверной: либо из репозиториев, либо из исходников. Для установки стандартного клиента выполняем команду:

```
sudo apt-get install x2goclient
```

Для установки клиента PyHoca с графической оболочкой:

```
sudo apt-get install pyhoca-gui
```

с командной строкой:

```
sudo apt-get install pyhoca-cli
```

На компьютере системного администратора устанавливаем плагин, позволяющий настраивать сервер через веб-интерфейс:

```
sudo apt-get install x2goplugin
```

После установки административный веб-интерфейс доступен по адресу:

```
http://<SERVER_IP_OR_HOSTNAME>/x2goplugin.html
```

Плагин работает в браузерах Google Chrome, Mozilla Firefox, Konqueror и Epiphany.

Примечание: в операционных системах Windows для печати с виртуального принтера Cups-X2Go на сервере на локальный принтер пользователя необходимо установить дополнительно Ghostscript и GSview, либо Foxit Reader.

Помимо указанных выше модулей существует ещё несколько, например, в репозиториях доступен модуль **pinentry-x2go**, используемый для авторизации по SMART-картам, ввода пин-кода или паролей в клиенте X2Go (бывает полезно для дополнительной защиты доступа к серверу X2Go), **python-x2go** — модуль для интеграции поддержки клиента X2Go с приложениями, написанными на языке Python, с использованием основанного на Python'е клиентского API; более подробную документацию об этом можно почитать, установив пакет **python-x2go-doc**. А если покопаться в исходниках, то вполне возможно, что среди них найдётся ещё несколько полезных модулей, которые потом можно скомпилировать и использовать, но этот вопрос более подробно мною не изучался.

После установки клиента запускаем его, в меню *Опции → Установки* выполняем необходимые настройки, коих не так уж и много. Во вкладке «Общие» настраиваем поведение программы в системном трее в процессе работы, авторизацию через сервер LDAP, если у Вас имеется контроллер домена, а также указываем порт, на котором работает SSH-сервер. Во вкладке «Печать» указываем локальный принтер, на который будет производиться печать из терминальной сессии, и настраиваем его свойства, либо прописываем вручную команду для печати и настраиваем дополнительные параметры, либо указываем программу, при помощи которой будут открываться документы для просмотра перед печатью. Если вы не желаете, чтобы каждый раз перед печатью выводился диалог настроек, снимите галочку «Открывать этот диалог перед печатью».

Теперь создаём новую сессию для подключения к нашему серверу, открываем меню *Сессия → Новая сессия* и настраиваем.

В первой вкладке «Сессия» вводим имя сессии, прописываем адрес сервера, имя пользователя, порт SSH-сервера (если в настройках сервера ничего не меняли, то оставляем по умолчанию 22), указываем путь к ключу RSA/DSA, если таковой имеется; если отдельный ключ не создавали, ставим галочку «Автоматическое соединение с SSH-ключом по умолчанию или программой ssh-agent». Сейчас в связи с распространением безлимитного интернета редко где можно встретить прокси-сервер, так что настройка «Использовать прокси-сервер для SSH-соединения» вряд ли понадобится. И наконец в секции «Тип сессии» выбираем графическую оболочку, используемую на сервере. На этой настройке хочется остановится поподробнее. Дело в том, что помимо стандартных оболочек KDE, Gnome, LXDE, XFCE и Unity можно выбрать подключение к RDP-серверу, XDMCP-серверу, настройка «Соединение с локальным десктопом» позволит подключиться к уже существующей терминальной сессии, как в VNC-сервере (для использования этой возможности на сервере обязательно должен быть установлен модуль **x2godesktopsharing**); настройка «Приложение» активирует режим Rootless, позволяющий запускать программы без загрузки рабочего стола

(создаётся ощущение, будто программа запущена локально). Более подробно использование этого режима описано в конце пункта 1.1.3 «Установка и настройка клиента FreeNX».

В вкладке «Соединение» устанавливаем скорость и качество изображения. Чем выше скорость соединения, тем выше нагрузка на локальную сеть, при выборе низкой скорости нагрузка будет меньше, а качество передаваемой картинки — ниже, а в некоторых приложениях вместо иконок могут появиться чёрные квадраты. Здесь же выбираем метод сжатия картинки и качество изображения, от этих настроек также будет зависеть скорость работы.

В третьей вкладке — «Установки» — настраиваем дополнительные параметры. Выбираем разрешение окна терминальной сессии: «Полноэкранный режим», «Другой» с возможностью выбора разрешения вручную, либо «Использовать весь дисплей» в случае, когда у вас к компьютеру подключено несколько мониторов, можно поставить галочку Xinerama для расширения окна терминальной сессии на другие мониторы. Установку DPI (количество точек на дюйм) рекомендую оставить стандартно 96, при бóльших значениях увеличатся размеры шрифта и значков. Галочку «Активировать звук» лучше снять, музыку на сервере мало кто станет слушать, а лишние нагрузки на сеть нам ни к чему; в настройках клавиатуры поставьте галочку «Использовать текущие установки», чтобы не возникало проблем с раскладками; самую нижнюю галочку «Печать на стороне клиента» оставляем включенной.

Ну и наконец в последней вкладке — «Экспорт каталогов» — выбираем локальные каталоги, которые будут проброшены на сервер. Если имеются проблемы с отображением кириллицы, то ставим галочку «Кодировка имён файлов» и экспериментируем с настройками. Галочку «Использовать SSH-туннель для экспорта файловой системы» рекомендую оставить.

Сохраняем все настройки. В дальнейшем параметры созданных сессий редактируются через меню *Сессия → Управление сессиями*. Чтобы каждый раз не запускать клиента, рекомендую в этом же меню для сессии создать ярлык на рабочем столе, указав нужную сессию и нажав соответствующую кнопку.

Когда все настройки выполнены, запускаем сессию, вводим логин и пароль и пробуем подключиться к нашему серверу.

На этом стандартная установка и настройка клиента X2Go закончена. Возможные проблемы, которые могут возникнуть с клиентом X2Go на этапе настройки и ввода в эксплуатацию, рассмотрены в п.1.2.4 «Возможные проблемы и способы их решения».

И напоследок рассмотрим вариант авторизации по RSA/DSA ключам без ввода пароля. Настройка авторизации производится на стороне клиента. В Linux-системах генерируем ключи командой:

ssh-keygen -t rsa или ***ssh-keygen -t dsa***

В каталоге **/.ssh**, расположенном в домашнем каталоге пользователя, должно появится два файла: закрытая часть **id_{rsa,dsa}** и открытая часть **id_{rsa,dsa}.pub**.

Теперь копируем открытую часть на сервер командой

ssh-copy-id -i ~/.ssh/id_rsa.pub USERNAME@SERVER

или

ssh-copy-id -i ~/.ssh/id_dsa.pub USERNAME@SERVER

В Windows-системах данные операции выполняются следующим образом: открываем командную строку **cmd**, переходим в папку с установленным клиентом X2Go:

cd C:\Program Files\x2goclient

Создаём пару ключей:

C:\Program Files\x2goclient>ssh-keygen.exe -t dsa

Копируем открытую часть на сервер:

C:\Program Files\x2goclient>scp.exe
/cygdrive/C/DOCUME~1/<username>/.x2go/.ssh/id_dsa.pub
username@remoteserver:

Далее входим на сервер, используя SSH:

C:\Program Files\x2goclient>ssh.exe username@remoteserver

Копируем содержимое ключа в файл **authorized_keys**, последовательно выполнив команды:

```
cd ~/.ssh
cat ~/id_dsa.pub >> authorized_keys
rm ~/id_dsa.pub
```

После проделанных операций в настройках сессии клиента X2Go указываем созданный RSA/DSA-ключ авторизации и можем запускать терминальные сессии без запроса логина и пароля.

Материал взят с официального сайта из раздела Advanced.

Существует ещё один вариант авторизации — вход по SMART-картам. Для его использования необходимо установить дополнительные пакеты:

```
sudo apt-get install pcscd pcsc-tools gnupg-agent pinentry-x2go
```

Поскольку он применяется крайне редко, останавливаться на нём не будем, более подробно с процессом настройки можно ознакомиться на официальном сайте также в разделе Advanced.

В данном описании рассмотрен автономный режим клиента X2Go, который используется в подавляющем большинстве случаев. В этом режиме пользователь сам настраивает сессии для подключения к серверу X2Go, а все профили сохраняются локально. Однако в масштабах больших предприятий управлять настройками клиентов становится крайне сложно. Поэтому в клиенте предусмотрен ещё один режим — *Broker Mode*, когда все настройки сессий клиент получает от сервера, а локальные профили при этом игнорируются. Инструкция по использованию клиента X2Go в этом режиме доступна на официальном сайте в разделе Advanced.

1.2.3. Настройка сервера «тонких клиентов» X2Go

В компьютерных технологиях «тонким клиентом» принято назвать компьютер-клиент с минимальным набором комплектующих и программного обеспечения, используемый в сетях клиент-серверной или терминальной архитектурой и переносящий основную часть задач по обработке информации на сервер. Преимущества такого решения очевидны: достаточно приобрести один мощный компьютер, который будет выступать в роли сервера, а в роли «тонкого клиента» может быть любой компьютер, даже совсем старый и непригодный для использования с современными программами, в котором имеется минимальный набор: материнская плата со встроенной видеокартой, память ОЗУ и сетевая карта, способная работать по протоколу PXE; наличие жёсткого диска необязательно.

Теперь перейдём к настройке сервера X2Go и системы «тонких клиентов».

В данном случае на стороне сервера придётся настраивать две службы: собственно сам терминальный сервер X2Go и сервер X2Go PXE, предоставляющий ресурсы для загрузки «тонких клиентов» по протоколу PXE. Для повышения общей производительности разработчики рекомендуют настраивать эти серверные службы на двух разных компьютерах или виртуальных машинах. В качестве операционных систем для сервера желательно использовать в первую очередь Debian, но можно и Ubuntu, поскольку именно под ними ведётся разработка и тестирование.

Также для полноценной работы «тонких клиентов» должна быть высокоскоростная локальная сеть со скоростью передачи данных не менее 100мб/с, но возможно их использование и через интернет, но с потерей качества.

Итак, сначала устанавливаем и настраиваем обычный терминальный сервер X2Go, как это описано выше.

Вторым шагом устанавливаем и настраиваем сервер X2Go PXE:

```
sudo apt-get install x2gothinclientmanagement
```

Далее описывается процесс сборки системы для «тонкого» клиента.

После установки открываем файл конфигурации и редактируем его под наши потребности в случае необходимости:

```
sudo editor /etc/x2go/x2gothinclient_settings
```

Запускаем процесс сборки системы для «тонких клиентов» в изолированной среде с предоставлением прав ***chroot***, так что будьте внимательны:

```
sudo x2gothinclient_create
```

Начнётся долгий процесс установки системы. Во время установки будет предложено настроить язык системы, кодировку локали и раскладку клавиатуры по умолчанию. В качестве кодировки используем стандартную UTF-8, например, en_US.UTF-8 или ru_RU.UTF-8. После установки система будет расположена в каталоге **/opt/x2gothinclient/chroot** (если в настройках не указан другой каталог).

Теперь редактируем файлы в каталоге **/opt/x2gothinclient/etc/**:

• **x2gothinclient_init** — скрипт инициализации, его редактировать нельзя!

• **x2gothinclient_start** — скрипт запуска клиента X2Go в качестве менеджера входа для «тонких» клиентов; по умолчанию настроен запуск в стандартном режиме без авторизации на LDAP-сервере (non-LDAP Mode), но для корректной работы его нужно немного подредактировать, например, изменить параметры клавиатуры ***--kbd-layout=us*** и ***--kbd-type=pc105***; для запуска клиента в режиме LDAP Mode необходимо в скрипте раскомментировать все строки для него и закомментировать для режима non-LDAP;

• **x2gothinclient_sessions** — файл содержит примерную конфигурацию профиля для запуска клиента X2Go в режиме non-LDAP; вы можете либо отредактировать уже существующий файл, либо создать сессию в клиенте X2Go на каком-нибудь локальном компьютере и далее скопировать полученный файл на сервер:

scp ~/.x2goclient/sessions
root@<pxeserver.hostname.domain>:/opt/x2gothinclient/etc/x2gothinclient_sessions

Когда установка и настройка системы «тонкого» клиента закончена, запускаем скрипт для сохранения изменений:

sudo x2gothinclient_update

ВНИМАНИЕ! Этот скрипт необходимо запускать после каждого изменения настроек профиля «тонких» клиентов.

Система готова! Но для для подключения «тонких» клиентов к серверу необходимо ещё настроить сервисы PXE-Boot и TFTP. Сначала ещё раз посмотрим содержимое файла конфигурации и в случае необходимости отредактируем его:

sudo editor /etc/x2go/x2gothinclient_settings

И если нас всё устраивает, запускаем скрипт:

sudo x2gothinclient_preptftpboot

Если среда PXE-Boot была настроена ранее, скрипт завершится с предупреждением. Для совмещения сервера X2Go PXE с другими аналогичными сервисами (например, LTSP), необходимо выполнить настройки вручную, шаблоны для настроек расположены в каталоге **/usr/share/x2go/tce/tftpboot**.

Для проверки правильности настроек сервера установите в BIOS'e компьютера загрузки по сети (PXE или Etherboot).

Для полного удаления созданной системы для «тонких» клиентов запускаем команду:

sudo x2gothinclient_cleanup

И наконец для полноты счастья выполняем последние настройки: сервера DHCP для автоматического получения IP-адреса и NFS для общего доступа каталогов.

Для настройки сервера DHCP редактируем файл конфигурации:

sudo editor /etc/dhcp3/dhcpd.conf

В примере показана настройка для сети 192.168.0.0/24:

```
option domain-name "<your-localdomain.org>";
option domain-name-servers <ip-of-dns-server>;

# x2go thin client range/group
subnet 192.168.0.0 netmask 255.255.255.0 {
range 192.168.0.100 192.168.0.199;
filename "/pxelinux.0";
next-server 192.168.0.250;
}
```

Сохраняем настройки и перезапускаем сервер DHCP:

sudo service dhcp3-server restart

Для настройки NFS-сервера редактируем файл:

sudo editor /etc/exports

Прописываем в него следующую строку:

```
/opt/x2gothinclient/chroot 192.168.0.0/24(ro,async,no_root_squash)
```

Сохраняем изменения и перезапускаем сервер:

sudo /etc/init.d/nfs-kernel-server restart

Корневая файловая система монтируется по протоколу NFSv3.

На этом настройки полностью завершены, запускаем «тонкий клиент» и пробуем систему в действии.

Описание настроек для «тонких клиентов» взято с официального сайта wiki.x2go.org.

И напоследок в установленной системе «тонких клиентов» можно установить дополнительные пакеты для загрузки клиента без рабочего стола и панели быстрого запуска, поддержки монтирования CD- и USB-носителей и SMART-картридеров.

Открываем консоль и делаем установленную систему «тонких клиентов» корневой:

```
chroot /opt/x2gothinclient/chroot
```

Устанавливаем дополнительные пакеты:

```
sudo apt-get install x2gothinclient x2gousbmount x2gocdmanager x2gosmartcardrules
```

ВНИМАНИЕ! Эти пакеты можно устанавливать только в системе «тонких клиентов» CHROOT! Ни в коем случае не устанавливайте их на основную операционную систему на сервере!

1.2.4. Возможные проблемы и способы их решения.

Все основные проблемы сервера X2Go и способы их решения рассматриваются на форуме unixforum.org. Все рекомендации, описанные ниже, взяты с него же.

1) В операционной системе Windows при использовании клиента в режиме *rootless* в заголовках окон отображается только надпись *VcXrv X* вместо нормальных надписей.

В качестве одного из вариантов решения предлагается заменить встроенный *Xming* на *Cygwin*. Сначала скачиваем последнюю версию Cygwin с официального сайта, устанавливаем, далее открываем настройки клиента X2Go, переходим во вкладку *«Установки X-сервера»*, ставим галочку *«Использовать другой X-сервер»*, в строке *«Исполняемый файл»* прописываем полный путь до файла запуска (например, *c:/cygwin/bin/XWin.exe*), далее чуть ниже ставим галочку *«Запускать X-сервер при запуске X2Go Client»* и в строке *«аргументы командной строки:»* прописываем:

```
-nodecoration -rootless -notrayicon -clipboard
```

Сохраняем настройки и перезапускаем клиент.

2) Если на сервере или на клиентском компьютеры имена и пароли пользователей заданы с использованием русских букв, то клиент их не воспринимает, запустить сессию невозможно. Особенно это проявляется в системах Windows.

Вообще использование имён и паролей с русскими буквами в любой системе считается моветоном, поэтому настоятельно рекомендуется пересоздать профили пользователей, присвоив им логины и пароли на латинице. Но если это представляет большую проблему, тогда попробуйте запускать клиента с ключём **`--portable`**:

```
x2goclient --portable --home=(x2godrive):/x2goclient/
```

3) В операционных системах Windows клиент не печатает на локальный клиентский принтер, а если в настройках печати у клиента стоит поставить галочку «Открывать этот диалог перед печатью», то при печати клиент вылетает с ошибкой 50.

Одна из причин — соединение может блокировать сетевой экран в антивирусе Касперского или встроенный брандмауэр Windows. Попробуйте отключить их и запустить печать ещё раз.

Также проверьте, установлены ли у вас программы Ghostscript и GSview, если нет, то установите.

Если и это не помогает, попробуйте выполнить следующее: откройте настройки клиента X2Go, перейдите во вкладку *«Печать»*, в секции *«Настройки печати»* поставьте галочку *«Команда печати»* и в строке ниже пропишите команду:

```
"c:\Program Files\Ghostgum\gsview\gsprint.exe" -query -color
```

Также вместо *gsprint* можно использовать сторонние программы, например, *Foxit Reader* с ключом **`/p`**, в этом случае команда печати выглядит так:

```
"C:\Program Files\FoxitReader\Foxit Reader.exe" /p
```

4) На Windows-клиентах иногда наблюдается проблема с переключением раскладок клавиатуры и включением клавиши NumLock. В этом случае поможет использование **setxkbmap** на сервере; создаём скрипт *keyfix.sh*:

sudo editor /bin/keyfix.sh

Прописываем в него следующее содержимое:

```
#!/bin/sh
setxkbmap -layout 'us,ru' -option 'grp:alt_shift_toggle,grp_led:scroll'
```

Делаем его исполняемым:

sudo chmod 0755 /bin/keyfix.sh

А затем под профилем каждого пользователя на сервере прописываем его в автозагрузку. Например, в оболочке XFCE открываем *Настройки → Сеансы и запуск → Автозапуск приложений* и добавляем его туда.

Также в этой оболочке для решения проблем с раскладками можно предложить такое решение: в апплете «Раскладки клавиатуры» выставить «Модель клавиатуры» *Стандартсная PC 101 клавиша.*

5) Если клиент X2Go в Windows7 видит только локальные принтеры и не видит сетевые, то последние можно подключить с использованием *Local Port*: открываем *Пуск → Панель управления → Устройства и принтеры → Добавить принтер →* выбираем *«Добавить локальный принтер»*, далее — *«Создать новый порт»* и из списка → *«Local Port»*. В строке *«Имя порта»* вводим сетевое имя принтера в формате **\\<SERVERNAME>\<PRINTERNAME>**, и наконец выбираем драйвер принтера из списка.

6) У Linux-клиентов возникают проблемы, когда интерфейс клиента переключался только на английский, в этом случае открываем на редактирование файл **~/.pam_environment** и оставляем только строки:

```
LANGUAGE=ru:en
LANG=ru_RU.UTF-8
```

7) Если клиент не может соединиться с сервером, выдавая сообщение типа «Ключ для сервера не найден», попробуйте на клиентском компьютере очистить все записи из файла **/home/USERNAME/.ssh/known_hosts**

8) На клиентских компьютерах с операционной системой Linux в терминальной сессии в режиме «Приложение» возможно «обрезание» окон снизу. В качестве решения проблемы предлагается на клиентском компьютере установить пакет **devilspie2**, затем создать файл конфигурации:

editor ~/.config/devilspie2/1cv8.lua

Копируем в этот файл следующее содержимое:

```
if (get_class_instance_name()=="1cv8") then
if ((get_window_type()=="WINDOW_TYPE_DIALOG") or
(get_window_type()=="WINDOW_TYPE_UTILITY")) then
if (not (string.find(get_window_property("_NET_WM_STATE"), "MODAL"))) then
undecorate_window();
end
end
end
```

Выводы по терминальному серверу X2Go пока что только положительные: хотя по сравнению с FreeNX у него ниже скорость работы, но зато минимум настроек и головной боли, максимум функционала и удобства. Команда немецких разработчиков очень хорошо постаралась, чтобы сделать из «сырой» когда-то разработки продукт, достойный внимания и уважения.

1.3. СЕРВЕР XPRA

Ещё одна утилита — Xpra, также работает по протоколу NX, но её сложно назвать «терминальным сервером», поскольку по принципу действию она отличается от классических терминальных решений.

Xpra является частью проекта PartiWM и позволяет выполнять графические приложения на клиентском компьютере в бескорневом (rootless) или бесшовном (seamless) режиме без привязки к текущему X-сеансу. Она является аналогом консольных оконных менеджеров *Tmux* и *Screen*, но в отличии от них нацелена на работу с X-сервером.

Принцип действия утилиты таков: на сервере утилита *Xpra* через консоль запускает в режиме демона нужную программу с заданным идентификатором сеанса, а на клиенте также при помощи команды в консоли происходит присоединение к сеансу с этим идентификатором. В отличие от пере-

направления X-сервера на клиентский компьютер по протоколу SSH, описанного в разделе 4, у данного решения есть несколько преимуществ:

1) более высокая скорость работы;

2) в случае разрыва соединения приложение остаётся запущенным на сервере, и можно продолжить работу с другого компьютера.

Но имеются и недостатки:

1) нельзя подключиться к одному сеансу сразу нескольким пользователям, поэтому для каждого пользователя нужно создавать отдельный сеанс; в итоге получится, что на сервере будет постоянно запущено несколько сессий, в том числе и неиспользуемые;

2) неудобство в работе: для работы сеанс сначала надо запустить на сервере вручную от имени пользователя, и только потом подключаться с клиентских компьютеров; эту проблему можно решить при помощи скриптов автозапуска;

3) завершить сессии можно также только вручную на сервере.

У этой утилиты нет разделения на «серверную» и «клиентскую» части, но у исполняемого файла есть разделение ключей для выполнения на сервере и на клиенте.

Утилита является кроссплатформенной, работает в системах Linux, Windows и MacOS, но в последних двух она может работать только в режиме клиента. Установочный комплект доступен на официальном сайте в разделе *Downloads*, а под Linux он имеется практически во всех официальных репозиториях. Для установки программы в системах Debian или Ubuntu достаточно выполнить команду:

```
sudo apt-get install xpra
```

Также на официальном сайте в разделе Wiki & Bugs → Source and Binaries доступны исходники последних версий с инструкциями по установке в различных системах.

Теперь приведу несколько примеров по использованию. Итак, сначала на сервере запускаем нужное приложение через *Xpra*, например, текстовый редактор *gedit*, и присваиваем произвольный номер дисплея, например, 100, команда запуска будет выглядеть так:

```
xpra start :100 --start-child=gedit
```

Протестируем соединение, подключившись к запущенному сеансу локально на сервере:

```
xpra attach :100
```

Теперь соединяется с клиентского места:

```
xpra attach ssh:SERVERNAME_OR_IP:100
```

Вместо ***SERVERNAME_OR_IP*** в команду подставляем имя сервера или его IP-адрес.

По умолчанию *Xpra* для обмена данными использует протокол SSH, но можно обойтись и без него, воспользовавшись протоколом TCP. В этом случае команда запуска приложения через *Xpra* на сервере будет выглядеть так:

```
xpra start :100 --start-child=xterm --bind-tcp=0.0.0.0:10000
```

Соответственно на клиенте для присоединения к запущенному сеансу выполняем команду:

```
xpra attach tcp:SERVERNAME_OR_IP:10000
```

Более полный список серверных и клиентских команд можно получить, набрав в консоли:

xpra --help или ***man xpra***

Изначально *Xpra* — это консольное приложение, но в последних версиях установочных комплектов идёт графическая утилита для клиентских компьютеров *Xpra Launcher* (команда запуска: ***xpra_launcher***), позволяющая упростить подключение к запущенным сеансам на сервере, сохранить текущую конфигурацию подключения с последующей загрузкой.

Также к серверу *Xpra* можно подключаться при помощи сторонней графической утилиты Winswitch, которая может работать по протоколу VNC, NX или RDP. Она также присутствует в репозиториях систем Linux, существуют версии и под другие системы.

Выводы по данной утилите таковы: работает всё довольно быстро, но из-за определённых неудобств использовать её можно только в масштабах небольшого предприятия с числом компьютеров не более пяти.

2. ПРОТОКОЛ RDP

Протокол RDP (Remote Desktop Protocol) был разработан компанией Citrix и затем приобретён корпорацией Microsoft. Для подключения к терминальному серверу RDP в операционной системе Windows можно использовать встроенный клиент «Подключение к удалённому рабочему столу» (mstsc.exe), в системе Linux лучшим является мультипротокольный клиент Remmina, но можно использовать и FreeRDP, GnomeRDP, TSClient, RDesktop.

RDP-сервер в операционной системе Linux можно организовать тремя способами: связкой Xrdp+X11rdp, связкой Xrdp+VNC, либо при помощи коммерческого LX-Server. Наиболее предпочтительным является первый вариант, поскольку он работает достаточно быстро и несложен в настройке. Второй вариант имеет довольно низкую скорость работы даже в локальной сети, а проект LX-Server закрыт и больше не поддерживается, к тому же его можно установить далеко не на каждый дистрибутив Linux.

2.1. СЕРВЕР XRDP+X11RDP

Эту связку можно смело рекомендовать в качестве RDP-сервера в небольших организациях. Всё устанавливается и настраивается достаточно просто, скорость отрисовки окон вполне приемлема для комфортной работы, для доступа к терминальным сессиям используются логины и пароли системных пользователей, сервер корректно работает со всеми программами, кириллические буквы отображаются корректно, буфер обмена работает в обе стороны. Под Windows можно использовать встроенный клиент «Доступ к удалённому рабочему столу», под Linux подойдёт любой клиент, в частности Remmina версии не ниже 0.9, Gnome-RDP, TS-Client. Вопрос интеграции с контроллером домена и Active Directory не прорабатывался, но думаю, это тоже возможно.

К недостаткам связки Xrdp+X11rdp можно отнести невозможность проброса локальных ресурсов на сервер (принтеры, папки, порты, звук), поддержку цветовой гаммы максимум 16bit (при более высоких значениях работа становится крайне медленной и нестабильной), но это скорее проблемы пакета Xrdp, и может быть в новых версиях они будут устранены. В старых версиях Xrdp версии 0.5.2 и ниже наблюдались проблемы с русской раскладкой клавиатуры, но в версии 0.6.0 они были решены. Также отсутствует программа администрирования сервера, все события приходится просматривать в логах, а пользователей отключать через консоль.

Также в последних нестабильных версиях XRDP, скачанных из GIT-репозиториев, в некоторых системах наблюдаются проблемы с кириллицей (не работает ввод русских букв, папки имеют английские имена и т. д.), но я думаю, что к моменту официального выхода они будут устранены. Если у вас такое проявляется, используйте предыдущую стабильную версию XRDP.

Если для Вас эти недостатки очень критичны, рекомендую обратить внимание на другие виды терминальных серверов, например, X2Go, описание которого приведено выше. Но если рассматривать их в масштабах небольшой фирмы с десятью компьютерами, то они на самом деле они не так страшны: к локальным принтерам и папкам можно открыть доступ по сети, звук и цветовая гамма более 16 бит очень редко используются даже в операционных системах Windows, а порты COM и USB можно пробросить различными способами. С вариантами проброса COM-порта можно ознакомиться на сайтах Etersoft и habrahabr.ru, с пробросом порта USB — на сайте babinov.ucoz.ru.

У коммерческих RDP-серверов на базе Windows достоинств гораздо больше, а все перечисленные недостатки отсутствуют, но когда дело доходит до подсчёта стоимости лицензий, которые почти всегда переваливают за отметку в сто тысяч рублей, тут уж решать клиенту: либо заплатить и не думать о проблемах, либо всё-таки смириться со всеми недостатками, но сэкономить приличную сумму денег, которую потом можно пустить на другие цели, например, на премии сотрудникам. Крупные фирмы как правило предпочитают первый вариант.

Установка связки Xrdp+X11rdp сводится к двум основным этапам: установка и настройка пакета Xrdp и компиляция X11rdp.

Чтобы значительно облегчить себе задачу, можно воспользоваться скриптами с сайта scarygliders.net, позволяющими скачать и скомпилировать обе компоненты из исходников, и затем настроить всё и сразу. Скачиваем необходимые скрипты из Git-репозитория:

```
git clone https://github.com/scarygliders/X11RDP-o-Matic.git
cd ./X11RDP-o-Matic
```

Запускаем скрипт *X11rdp-o-matic.sh* с минимальным количеством модулей:

```
sudo bash ./X11rdp-o-matic.sh --justdoit
```

Команда для полной компиляции со всеми возможными модулями выглядит так:

```
sudo bash ./X11rdp-o-matic.sh --justdoit --withjpeg --withsound --withkerberos --withfreerdp --withpamuserpass --withneutrino --withxrdpvr
```

Скрипт автоматически создаст все нужные каталоги, установит пакеты, необходимые для компиляции, скачает исходники Xrdp и X11rdp и всё скомпилирует под вашу систему, а также оптимизирует будущий сервер под многоядерные процессоры. Время компиляции зависит от мощности компьютера и интернета и может составлять от 15 минут до 2 часов. После компиляции скрипт автоматически создаст и установит два пакета: **xrdp** и **x11rdp**.

После того, как первый скрипт закончит работу, запускаем второй — *RDPsesconfig.sh*:

```
sudo bash ./RDPsesconfig.sh
```

Отвечаем на несколько вопросов (какую графическую оболочку использовать, каким пользователям разрешить доступ к терминальному серверу), и настройка на этом закончена. Теперь пробуем подключиться к нашему серверу при помощи любого RDP-клиента.

Но как показала практика, в более новых системах этот скрипт может работать некорректно, например, компиляция и установка проходят без ошибок, а при попытке подключения может выдаваться сообщение о неправильном логине и пароле, либо клиент вообще «вылетает» без каких-либо ошибок. В таких случаях рекомендуется скомпилировать и установить эту связку вручную описанным далее способом.

2.1.1. Установка X11rdp

В данном разделе описан процесс компиляции X11rdp из исходных текстов.

1) Устанавливаем пакеты для компиляции X11rdp:

```
sudo apt-get install automake automake1.9 build-essential subversion gcc libice-dev pkg-config zlib1g-dev cvs autoconf libtool libbison-dev libssl-dev libpam0g-dev libx11-dev libxfixes-dev xfonts-base
```

В более старых версиях Debian и Ubuntu устанавливаем дополнительные пакеты для компиляции X-сервера:

```
sudo apt-get build-dep xserver-xorg-core
```

2) Скачиваем исходники X11rdp из SVN под профилем пользователя (не root'a!):

```
svn co svn://server1.xrdp.org/srv/svn/repos/main/x11rdp_xorg71
```

Последняя версия ревизии на момент написания статьи — 299

Для более старых версий Debian и Ubuntu скачиваем архив с исходниками и распаковываем его:

```
wget http://server1.xrdp.org/xrdp/x11rdp_xorg71.tar.gz
tar xvf x11rdp_xorg71.tar.gz
```

3) Создаём каталог, где будут находиться все скомпилированные компоненты (владельцем каталога обязательно должен быть root!):

```
sudo mkdir /opt/X11rdp
```

4) Переходим в каталог с исходниками и компилируем:

```
cd ~/x11rdp_xorg71
time sudo sh buildx.sh /opt/X11rdp
```

Время компиляции в зависимости от мощности компьютера может занять от 15 минут до 2 часов.

5) создаём симлинк в каталоге **/usr/bin**:

```
sudo ln -s /opt/X11rdp/bin/X11rdp /usr/bin/X11rdp
```

Чтобы не компилировать систему каждый раз, желательно после компиляции создать готовый DEB-пакет. Как это сделать, можно почитать на сайте useunix.ru.

При сборке DEB-пакета рекомендую поставить в зависимости пакеты *libice6*, *libpam0g*, *libssl1.0.0*, *libx11-6*, *libxfixes3*, *xfonts-base*, *xrdp* и *zlib1g*.

Проверяем работу X11rdp:

```
/usr/bin/X11rdp :1
```

Если всё нормально, то в консоли должен быть вывод:

```
X11rdp, an X server for xrdp
Version 0.5.0
Copyright (C) 2005-2008 Jay Sorg
See http://xrdp.sf.net for information on xrdp.
Underlying X server release 70100000, The X.Org Foundation
Xorg Release 7.1
Screen width 800 height 600 depth 24 bpp 32
dpix 100 dpiy 100
buffer size 1920000
error opening security policy file /etc/X11/xserver/SecurityPolicy
Could not init font path element /opt/X11rdp/lib/X11/fonts/TTF/, removing from list!
Could not init font path element /opt/X11rdp/lib/X11/fonts/OTF, removing from list!
Could not init font path element /opt/X11rdp/lib/X11/fonts/CID/, removing from list!
Could not init font path element /opt/X11rdp/lib/X11/fonts/100dpi/, removing from
list!
Could not init font path element /opt/X11rdp/lib/X11/fonts/75dpi/, removing from
list!
```

Прерываем работу сервера X11rdp клавишами *Ctrl+C*.

На этом первая часть закончена, каких-либо особых настроек X11rdp не требует.

2.1.2. Установка Xrdp

Пакет Xrdp можно рассматривать как «прослойку» между клиентским компьютером и сервером, которая обеспечивает доступ к удалённому рабочему столу по протоколу RDP.

В большинстве источников рекомендуется скачать исходники Xrdp из git-репозитория и скомпилировать вручную.

Я же рекомендую скачать исходники Xrdp версии не ниже 0.6.0 из репозиториев своей системы (Debian или Ubuntu) и собрать готовый deb-пакет. Во-первых, это более правильный метод, а во-вторых, меньше придётся возиться с настройками.

Если в репозиториях вашей системы уже есть пакет Xrdp версии 0.6.0 и выше, то установите его и переходите сразу к настройкам.

В более старых версиях Xrdp 0.5.0 и 0.5.2 возможна проблема с русской клавиатурой.

Итак, приступим:

1) Устанавливаем пакеты для правильной компиляции XRDP:

```
sudo apt-get install debhelper libssl-dev libpam0g-dev autoconf
automake libtool libx11-dev libxfixes-dev libfreerdp-dev pkg-config
```

2) Скачиваем файлы xrdp_0.6.0.orig.tar.gz, xrdp_0.6.0-1.dsc и xrdp_0.6.0-1.debian.tar.gz в домашний каталог, распаковываем архив и накладываем патчи под профилем пользователя (не root'а!)

```
dpkg-source -x ~/xrdp_0.6.0-1.dsc
```

3) Переходим в каталог с исходниками Xrdp и собираем пакет:

```
cd ./xrdp-0.6.0
dpkg-buildpackage -b -tc
```

4) После компиляции устанавливаем пакет:

```
sudo dpkg -i ~/xrdp_0.6.0-1_i386.deb
```

Теперь пример компиляции Xrdp из git-репозитория.

1) Устанавливаем дополнительные пакеты для правильной компиляции:

```
sudo apt-get install git libssl-dev libpam0g-dev
```

2) Устанавливаем Xrdp из репозитория и тут же удаляем его:

```
sudo apt-get install xrdp -y ; apt-get remove xrdp -y
```

Этот шаг позволит установить все зависимости, создать группу xrdp и внести в неё пользователей. Вместе с нужными пакетами установятся дополнительно сервер и клиент VNC, но их потом можно будет удалить.

3) Скачиваем исходники Xrdp:

```
git clone https://github.com/FreeRDP/xrdp.git xrdp.git
```

4) компилируем Xrdp, выполняем последовательно команды под профилем пользователя (не

root'a!):

```
cd xrdp.git
git checkout 4cd0c118c273730043cc77b749537dedc7051571
./bootstrap
./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var
make
```

5) устанавливаем Xrdp:

```
sudo make install
```

6) создаём нужные каталоги и копируем в них RSA-ключи для терминальных сессий:

```
sudo su -
mkdir /usr/share/doc/xrdp
mv /etc/xrdp/rsakeys.ini /usr/share/doc/xrdp/
chmod 600 /usr/share/doc/xrdp/rsakeys.ini
chown xrdp:xrdp /usr/share/doc/xrdp/rsakeys.ini
```

7) редактируем скрипт **/etc/init.d/xrdp**, исправляем строки или прописываем их вручную, если они отсутствуют:

```
PIDDIR=/var/run
SESMAN_START=yes
```

8) добавляем сервис **xrdp** в автозагрузку:

```
sudo update-rc.d xrdp defaults >/dev/null
```

9) добавляем пользователя **xrdp**:

```
sudo adduser --system --disabled-password --disabled-login --home /var/run/xrdp --no-create-home --quiet --group xrdp
```

10) изменяем права на каталог с конфигурационными файлами:

```
sudo chown -R xrdp:xrdp /etc/xrdp
```

11) запускаем сервер:

```
sudo service xrdp start
```

На этом установка и первоначальная настройка Xrdp закончена, переходим теперь к окончательным настройкам Xrdp и запуску сервера.

2.1.3. Настройка Xrdp

1) Для начала нужно отредактировать файл **/etc/xrdp/xrdp.ini**, чтобы использовался только X11rdp:

```
sudo editor /etc/xrdp/xrdp.ini
```

Полностью всё удаляем и прописываем следующее содержимое:

```
[globals]
bitmap_cache=yes
bitmap_compression=yes
port=3389
crypt_level=low
channel_code=1
max_bpp=16

[xrdp1]
name=sesman-X11rdp
lib=libxup.so
username=ask
password=ask
ip=127.0.0.1
port=-1
xserverbpp=1
```

Параметры ***max_bpp*** и ***xserverbpp*** были специально ограничены цветовой палитрой 16 бит, поскольку сервер некорректно работает с палитрой 24 бита и выше.

2) в некоторых источниках рекомендуется использовать скрипт *Xsession* вместо оригинального *startwm.sh*, для этого нужно создать резервную копию оригинального скрипта *startwm.sh* и сделать

символическую ссылку на *Xsession*. В Xrdp версии 0.6.0 и выше это делать не имеет смысла, поскольку в ней скрипт *startwm.sh* выполняет некоторые условия, а затем запускает *Xsession*.

2) Как было сказано выше, при старте терминальной сессии в окне ввода логина и пароля не работает переключение раскладок; проблему можно решить следующим способом: редактируем обозначенный выше файл **/etc/xrdp/startwm.sh**:

```
sudo gedit /etc/xrdp/startwm.sh
```

В самом конце перед строкой **. /etc/X11/Xsession** прописываем:

```
setxkbmap -layout "us,ru(winkeys)" -model "pc105" -option
"grp:ctrl_shift_toggle,grp_led:scroll"
```

Это решение позволит правильно определить локаль при запуске сессии и активирует переключение раскладок клавиатуры.

4) если на сервере установлено несколько графических оболочек либо был заменён скрипт *startwm.sh* на *Xsession*, как это было описано в предыдущем шаге, то в корне домашнего каталога каждого пользователя создаём файл с именем **.xsession** (впереди точка обязательна!), присваиваем ему права 644, делаем владельцем соответствующего пользователя и группу и прописываем в него строку с нужной оболочкой:

- для Gnome2 в Debian: `gnome-session`
- для Gnome2 в Ubuntu 10.04 и выше: `gnome-session -f`
- для Gnome-Shell Classic: `gnome-session --session=gnome-fallback`
- для Unity-2D: `gnome-session --session=ubuntu-2d`
- для KDE: `startkde`
- для XFCE: `startxfce4`
- для LXDE: `startlxde`
- для MATE: `mate-session`

Для других оболочек, отсутствующих в списке, нужно прописать соответствующую команду запуска.

Для оболочки Gnome можно сделать дополнительную настройку — убрать задний фон в терминальной сессии, для этого прописываем в файл следующее содержимое:

```
background=`gsettings get org.gnome.desktop.background picture-uri`
gsettings set org.gnome.desktop.background picture-uri ''
gnome-session --session=gnome-fallback
gsettings set org.gnome.desktop.background picture-uri $background
```

5) по умолчанию в Xrdp количество разрешённых подключений — 10; чтобы его увеличить, редактируем файл **/etc/xrdp/sesman.ini** и в секции **[Sessions]** строке **MaxSessions** прописываем нужное количество сессий.

6) запускаем XRDP:

```
sudo service xrdp start
```

Если всё настроено нормально, то при запуске ошибок быть не должно. На всякий случай можно проверить, запущены ли нужные службы:

```
ps ax | grep rdp
```

Команда должна вывести запущенные службы **xrdp** и **xrdp-sesman**.

Теперь пробуем подключиться к нашему RDP-серверу при помощи любого клиента.

ВНИМАНИЕ! Перед подключением в настройках клиента обязательно использовать цветовую палитру не более 16bit (с более высокими значениями скорость работы будет очень низкой), а также нужно отключить воспроизведение звука, если такая настройка имеется. Клиенты Remmina с версией ниже 0.9 к серверу не подключаются, поскольку они не поддерживают протокол RDPv5.

Как уже было сказано выше, какая-либо утилита администрирования у сервера отсутствует, точнее она присутствует в виде исполняемого файла **xrdp-sesadmin**, но пока не работает, и если возникает необходимость отключить клиента, например, по причине «зависшей» сессии, делаем следующее: открываем файл **/var/log/xrdp-sesman.log** и смотрим PID процесса X11rdp, принадлежащего конкретному пользователю, подключенному с конкретного IP-адреса, далее при помощи команды `ps ax | grep PID` проверяем, что процесс до сих пор работает, и завершаем его ко-

мандой ***kill -9 PID***. Вместо **PID** соответственно подставляем идентификатор PID нужного процесса.

Также наблюдаются проблемы, когда в случае разрыва соединения клиент уже не может подключиться к текущей сессии. Если у вас такое происходит, попробуйте пересобрать пакет Xrdp, предварительно поправив исходники, в файле **/sessman/session.c** найдите строки и отредактируйте так, как показано ниже:

```
if (g_strncmp(name, tmp->item->name, 255) == 0
        //&&
        //tmp->item->width == width &&
        //tmp->item->height == height &&
        //tmp->item->bpp == bpp &&
        //tmp->item->type == type
            )
```

При написании данного раздела использованы материалы с сайтов scarygliders.net, ubuntovod.ru, debet.kiev.ua и habrahabr.ru.

2.2. СЕРВЕР XRDP+VNC

VNC (Virtual Network Computing) — это система удалённого доступа к рабочему столу по протоколу RFB (Remote Frame Buffer). Изначально VNC-серверы не предназначены для терминального доступа, поскольку они дают доступ только к рабочему столу того пользователя, который уже работает в системе, а если пользователь не залогинился на сервере, то соответственно VNC-сервер под его профилем не запущен и подключение к нему будет невозможно.

Как уже говорилось выше, пакет Xrdp является своеобразной «прослойкой» между сервером и клиентом, и он позволяет превратить VNC-сервер в в терминальный сервер RDP. В отличие от связки Xrdp+X11rdp данный вариант ещё проще в настройке, достаточно всего лишь установить из репозиториев 2 пакета и начать использовать, но он обладает одним существенным недостатком: из-за особенностей VNC-сервера скорость отрисовки окон будет ниже, особенно это заметно в программах, запущенных под Wine, поэтому рекомендовать этот вариант можно с трудом, и то только в тех случаях, когда другие варианты использовать невозможно (например, не компилируется X11rdp). Также некоторые программы могут не запуститься, например, офисный пакет FreeOffice от компании SoftMaker при запуске выдаёт ошибку *«Xrender initialization failed»*.

Описание установки и настройки Xrdp было приведено выше, поэтому сразу перейдём к VNC-серверу.

При установке пакета Xrdp из репозиториев по умолчанию предлагается установить пакет *vnc4server*, но можно также использовать *tightvncserver*, а в более старых системах — *vnc-server*. С другими VNC-серверами у меня «подружить» Xrdp не получилось.

Итак, устанавливаем любой обозначенный выше VNC-сервер из репозиториев (рекомендую TightVNC), затем запускаем его. Для каждого из серверов команда запуска будет своя, например, для TightVNC команда будет

sudo tightvncserver

для vnc4server соответственно

sudo vnc4server

При первом запуске программа попросит создать пароль на соединение с нашим сервером. Все данные сохраняются в каталоге **/root/.vnc/**. Когда заменяем один VNC-сервер на другой, каталог **.vnc** во всех профилях пользователей желательно полностью удалить.

Этот пароль затем можно будет изменить соответствующей командой:

sudo tightvncpasswd *или* ***sudo vnc4passwd***

Файл **/etc/xrdp/xrdp.ini** редактировать не нужно, поскольку в нём по умолчанию в секции ***[xrdp1]*** настроен ***sesman-Xvnc***.

Запускаем любой RDP-клиент и пробуем подключаться к нашему серверу. В качестве логина и пароля используем данные любого системного пользователя. Не забываем перед подключением в настройках клиента выставить цветовую гамму не более 16bit и отключить передачу звука.

2.3. СЕРВЕР ULTEO OVD

Проект Ulteo был создан Гаэлем Дювалем — одним из основателей некогда популярной системы Mandriva Linux.

Продукт Open Virtual Desktop (далее OVD) не является «терминальным сервером» в его классическом понимании, как остальные, его корректнее назвать «сервером приложений», предоставляющим клиентским компьютерам доступ к приложениям или рабочим столам, запущенным на других серверах под управлением операционных систем Windows или Linux. Но в некоторых случаях его можно использовать и в качестве классического терминального сервера с доступом клиентов через любой браузер.

Принцип действия таков: предположим, в сети имеется терминальный сервер на базе системы Linux или Windows, расположенный на одном компьютере (назовём его Сервер1), на котором находятся приложения для общего пользования, и сервер OVD, расположенный на другом компьютере (далее OVD-сервер). В настройках OVD-сервера регистрируем Сервер1, а также приложения, к которым необходимо открыть доступ клиентам. Далее на клиентском компьютере открываем браузер, в адресной строке набираем адрес OVD-сервера, вводим логин и пароль, и затем запускаем программы, которые физически расположены на Сервере1. Обмен данными между Сервером1 и OVD-сервером производятся по протоколу RDP.

Такой подход очень удобен в случаях, когда в организации находятся несколько терминальных серверов, к которым необходимо открыть доступ клиентам. В этом случае OVD-сервер объединяет общедоступные ресурсы воедино, и сотрудникам не надо задумываться, к какому серверу нужно подключиться, чтобы получить доступ к тому или иному приложению, достаточно открыть любой браузер и начать пользоваться нужными программами.

Обмен данными между сервером OVD и терминальными серверами в сети происходит по протоколу RDP. В операционных системах Windows Server используется встроенный терминальный сервер, а в системах Linux — связка собственного доработанного *Xrdp* и *Uxda-Server*, которая несовместима со стандартными RDP-клиентами.

У сервера OVD есть две редакции: Community Edition — с открытым исходным кодом, распространяемая бесплатно, и Premium Edition — распространяется на коммерческой основе по подписке, в неё включены дополнительные модули с закрытым исходным кодом.

Основные возможности сервера Ulteo OVD:

• доступ клиентов к приложениям в режиме Портала или Рабочего стола через любой браузер с установленной Java;

• работа в операционных системах Linux и Windows Server;

• интеграция с сервером каталогов LDAP, Microsoft Active Directory или Novell eDirectory;

• проброс ресурсов (принтеры, звук);

• проброс файлов и папок клиента на сервер в терминальной сессии;

• автоматическое восстановление сессии в случае обрыва;

• балансировка нагрузки между серверами;

• удобная оболочка администрирования с интуитивно понятным интерфейсом и кучей настроек и отчётов; благодаря ей можно провести тонкую настройку сервера под свои требования, контролировать доступ пользователей, следить за нагрузкой на сервера и т. д.

Дополнительные возможности версии Premium Edition:

• собственный терминальный клиент с дополнительными настройками;

• поддержка авторизация по PC/SC-совместимым смарт-картам;

• интеграция с серверами Alfresco, Microsoft SharePoint и т. д.;

• доступ к приложениям с мобильных устройств на базе Android и iOS с использованием собственного клиента;

• запуск приложений в «бесшовном» режиме (seamless), создаётся ощущение, как будто приложение установлено локально на компьютере; значки для запуска приложений можно поместить на рабочий стол или в пусковое меню;

• дополнительный модуль Ulteo OVD Gateway, позволяющий организовать собственный защищённый канал SSL VPN для доступа к серверу OVD из интернета.

С более подробным перечнем характеристик обоих редакций можно ознакомиться на официальном сайте.

2.3.1. Установка сервера Ulteo OVD

Сервер Ulteo OVD состоит из трёх основных компонент: менеджера сессий, сервера приложений и клиентского веб-портала.

Менеджер сессий — это центральная часть сервера OVD, предназначенная для настройки и управления всей системой в целом. Через менеджер сессий регистрируются терминальные сервера, настраивается доступ к приложениям, создаются пользователи и группы и т. д. Он построен на базе LAMP-сервера (Linux Apache MySQL PHP), необходимые для его работы пакеты устанавливаются автоматически в качестве зависимостей. Менеджер сессий можно установить из существующих пакетов только в операционной системе Linux, для установки в системе Windows необходимо скомпилировать его из исходников.

Во время установки менеджера сессий по-умолчанию скачивается архив с подсистемой, в которую входит сервер приложений, файловый сервер и предустановленный рабочий стол Ulteo на базе оболочки XFCE. Это позволяет быстро развернуть сервер Ulteo OVD.

Сервер приложений и файловый сервер устанавливается на терминальные хосты, к которым нужно открыть доступ пользователям. Во время установки и настройки для регистрации необходимо указать IP-адрес сервера, на котором расположен менеджер сессий. Сервер приложений может работать как в системах Linux, так и в Windows.

Веб-портал предназначен для доступа клиентов к приложениям через любой браузер. Его установка необязательно, если используется клиент Ulteo OVD Native Client.

Все эти компоненты разработчики рекомендуют устанавливать на разных компьютерах, но можно всё сделать и на одном, в этом случае сервер OVD превращается в классический терминальный сервер.

Рассмотрим установку и настройку сервера Ulteo OVD 4.0 Community Edition на примере Ubuntu 12.04 LTS Precise Pangolin и Debian 7.0 Wheezy. Для облегчения процесса все компоненты будем устанавливать на один компьютер.

Примечание: в более новых версиях Ubuntu сервер OVD может не установиться из существующих пакетов, в этом случае необходимо пересобрать все компоненты из исходников под вашу систему.

Описание взято с официального сайта и с русскоязычного сайта Ulteo. Также были использованы материалы с сайта habrahabr.ru.

ВНИМАНИЕ! Для сервера приложений в текущей версии Ulteo OVD 4.0 на хост-системе рекомендуется использовать ядро версии 3.2 или ниже! С более новыми версиями ядра могут возникнуть проблемы. Я в своих экспериментах использовал ядро 3.8 и особых проблем не обнаружил, но они могут появиться в процессе эксплуатации.

Все действия во время установки выполняем под профилем root'a, поэтому открываем терминал и авторизуемся:

```
sudo su
```

Далее последовательно выполняем шаги:

1) Добавляем репозиторий в общий список источников:

```
editor /etc/apt/sources.list.d/ulteo_ovd.list
```

Прописываем в файл следующее содержимое и сохраняем:

для Ubuntu 12.04:

```
deb http://archive.ulteo.com/ovd/4.0/ubuntu precise main
```

для Debian 7.0:

```
deb http://archive.ulteo.com/ovd/4.0/debian wheezy main
```

Примечание: если вы используете коммерческую версию Premium Edition, то к основному репозиторию добавляем дополнительный; для Ubuntu 12.04 прописываем в этот же файл ещё одну строку:

```
deb http://archive.ulteo.com/ovd/4.0/premium/ubuntu precise main
```

Обновляем список пакетов:

```
apt-get update
```

2) Устанавливаем ключ проверки подлинности:

```
apt-get install ulteo-keyring
apt-get update
```

Если вы планируете использовать все компоненты сервера OVD на одном компьютере, то достаточно установить пакет **easy-install**, который в свою очередь автоматически устанавливает все необходимые компоненты сервера и дополнительные программы в качестве зависимостей:

```
apt-get install ulteo-ovd-easy-install
```

После установки создаём базу OVD в сервере MySQL, входим в оболочку администрирования, выполняем необходимые настройки, создаём пользователей, группы, публикуем приложения, и сервер полностью готов к работе. Все эти настройки описаны ниже.

Ну а для тех, кто не ищет лёгких путей, рассмотрим установку каждой компоненты сервера отдельно.

2.3.2. Установка и настройка менеджера сессий

Сначала устанавливаем менеджер сессий — самую главную часть сервера. Выполняем пошагово команды:

1) устанавливаем сервер MySQL и создаём базу OVD (во время установки сервера вводим пароль root'a):

```
apt-get install mysql-server
mysql -u root -p -e 'create database ovd'
```

Примечание: сервер баз данных MySQL можно установить на другом хосте или использовать существующий, в этом случае нужно указать его IP-адрес и имя базы во время настройки менеджера сессий.

2) Устанавливаем менеджер сессий и консоль администрирования:

```
apt-get install ulteo-ovd-session-manager ulteo-ovd-administration-console ulteo-ovd-l10n
```

Во время установки необходимо ввести IP-адрес сервера, на котором будет располагаться менеджер сессий, оставляем по умолчанию **127.0.0.1**, далее задаём логин и пароль администратора: вводим логин **admin**, пароль придумываем сами.

На последнем шаге будет предложено скачать архив с подсистемой для упрощения развёртывания сервера приложений. Архив имеет размер более 650 МБ, и если интернет медленный, то этот шаг можно пропустить, а архив с базовой подсистемой скачать отдельно и скопировать вручную в нужный каталог:

```
wget -c http://archive.ulteo.com/ovd/4.0/subsystem/base.tar.gz
mv base.tar.gz /var/cache/ulteo/sessionmanager/
```

Итак, базовая установка менеджера сессий завершена, но он ещё не сконфигурирован. Открываем любой браузер и вводим адрес http://127.0.0.1/ovd/admin, далее в административном окне — логин и пароль администратора, который мы задали во время установки, и приступаем к конфигурированию.

Поскольку сразу после первой установки менеджер сессий ещё не настроен, то при первом входе автоматически откроется меню *Configuration → Database Settings*, в котором необходимо настроить доступ к базе данных. Если сервер баз данных MySQL был установлен на тот же хост, что и менеджер сессий (как описано выше), то всё оставляем по умолчанию, а в полях *Database username* и *Database password* вводим логин **root** и соответственно его пароль, после этого нажимаем *Save*. В противном случае необходимо указать нужные параметры (IP-адрес сервера MySQL, логин и пароль администратора, имя базы и префикс таблиц).

Если при входе в систему весь интерфейс на английском языке, то открываем меню *Configuration → System Settings* и в строке *Administration Console language* вместо *Autodetect* выбираем родной русский язык и нажимаем *Save*. Теперь переходим в меню *Настройки → Настройки*

сессии и в строке *Язык по-умолчанию для сессии* также выбираем русский и нажимаем *Сохранить*.

На этом предварительное конфигурирование менеджера сессий закончено.

2.3.3. Установка и настройка сервера приложений

Следующим этапом устанавливаем сервер приложений и файловый сервер. Как уже говорилось в самом начале, сервер приложений OVD работает только с ядром версии 3.2 или ниже, поэтому перед продолжением настроек необходимо установить ядро нужной версии, а все остальные ядра — удалить.

1) Устанавливаем ядро версии 3.2 для виртуальных машин:

```
sudo apt-get update
sudo apt-get install linux-virtual
```

После установки перезагружаем сервер с установленным ядром.

2) Ищем и удаляем все остальные версии ядра. Проверяем, какие версии ядра установлены в системе:

```
dpkg -l | grep linux-image
```

Удаляем лишние версии:

```
sudo apt-get remove --purge linux-image-XXX
```

Теперь ещё раз перезагружаем сервер.

3) Далее возможно два варианта установки сервера приложений:

вариант №1: установка пакета *SlaveServer*, в этом случае на существующую систему устанавливаются все необходимые пакеты сервера приложений, файлового сервера и веб-сервера, а все программы, установленные на компьютере, становятся сразу доступными для публикаций; однако если вы планируете использовать сервер приложений в режиме рабочего стола, то необходимо дополнительно устанавливать оболочку XFCE4, с другими оболочками сервер Ulteo OVD не работает.

Сначала устанавливаем пакет SlaveServer и дополнительные функциональные пакеты:

```
apt-get install ulteo-ovd-slaveserver ulteo-ovd-slaveserver-role-aps
ulteo-ovd-slaveserver-role-fs ulteo-ovd-slaveserver-role-web
```

Если планируется использование режима рабочего стола, то устанавливаем дополнительный пакет Desktop, который в свою очередь установит пакеты оболочки XFCE4 в качестве зависимостей:

```
apt-get install ulteo-ovd-desktop
```

Имейте ввиду, что после перезагрузки компьютера с установленным сервером приложений оболочка XFCE4 на нём станет используемой по-умолчанию.

вариант №2: установка пакета с базовой подсистемой, в этом случае будет развёрнут архив с подсистемой на базе Ubuntu 10.04 Lucid Lynx с минимальным набором программ, в которою уже включен сервер приложений, файловый сервер и рабочий стол на базе оболочки XFCE4; этот вариант можно использовать на серверах без графической оболочки, но все программы, установленные на основном компьютере, в базовой подсистеме доступны не будут, их придётся добавлять отдельно, используя *chroot*, либо использовать заранее подготовленный архив со своей системой и набором программ.

Устанавливаем пакет с базовой подсистемой:

```
apt-get install ulteo-ovd-subsystem
```

В конце установки с сервера менеджера сессий будет скачан архив с базовой подсистемой **base.tar.gz** и распакован в каталог **/opt/ulteo**.

В обоих вариантах во время установки необходимо указать адрес или имя сервера с установленным менеджером сессий. Поскольку в нашем случае всё устанавливается на один компьютер, то вводим адрес **127.0.0.1**.

На этом конфигурирование завершено, перезапускаем подсистему:

```
service ulteo-ovd-subsystem restart
```

Если устанавливали SlaveServer, то команда перезапуска будет такой:

```
service ulteo-ovd-slaveserver restart
```

Теперь, если войти в административную оболочку менеджера сессий и открыть меню *Серверы*

→ *Незарегистрированные сервера*, в нём появится сервер приложений. Нажимаем кнопку *Зарегистрировать*, и он появится во вкладке *Серверы → Серверы*.

Поскольку у нас все компоненты установлены на одном компьютере, то необходимо задать имя и перенаправление сервера приложений. В столбце *Имя* нажимаем ссылку **localhost** и в открывшемся ниже меню *Настройки* задаём параметры: в строке *Имя либо адрес для обращения к серверу* прописываем реальный IP-адрес в сети или имя компьютера, на который установлен сервер приложений (адреса типа **127.0.0.1** или **localhost** система не принимает!), в конце строки нажимаем *Определить*; в строке *Отображаемое имя* — любое имя, например, **APS Demo**, в конце строки — *Определить*. После этого появится сообщение «Сервер APS Test успешно изменён», других сообщений об ошибках быть не должно! Нажимаем кнопку *В рабочий режим*.

Теперь создаём пользователей и настраиваем доступ к приложениям. Переходим во вкладку *Пользователи → Пользователи*, создаём пользователей, присваиваем им логин и пароль.

Примечание: если у вас используется контроллер домена с MS Active Directory, Novell eDirectory или LDAP, то пользователей можно не добавлять, а для авторизации уже имеющихся пользователей достаточно выполнить интеграцию OVD с сервисом, для этого открываем меню *Настройки → Настройка службы каталогов*, выбираем сервис и прописываем все необходимые данные, после чего нажимаем *Сохранить*.

Далее создаём группу пользователей и включаем в неё созданных пользователей. Переходим во вкладку *Группы пользователей*, создаём статическую группу, например, **allusers**, а чтобы все пользователи автоматически в неё добавлялись, нажимаем в настройках *Сделать по-умолчанию*.

Все приложения хранятся на сервере, но пока они не опубликованы, они пользователям недоступны. Для публикации приложений открываем *Мастер публикаций*, в нём выбираем *Использовать существующие группы* и ставим галочку на созданной нами группе, на следующем шаге выбираем из списка приложения для публикации, далее задаём имя публикации и на последнем шаге нажимаем *Подтвердить*. Теперь, если перейти во вкладку *Группы пользователей* и открыть настройки группы, публикация должна в них появиться. Управлять публикацией можно из меню *Приложения → Группы приложений*.

Примечание: для публикации приложений, которых нет в списке, переходим во вкладку *Приложения → Статические приложения*, и в секции *Добавить приложение* прописываем нужные данные.

ВНИМАНИЕ! Не пытайтесь подключиться к серверу приложений при помощи стандартного RDP-клиента! Во-первых, у вас ничего не получится, а во-вторых, такая попытка приведёт к сбою сервера приложений, в этом случае восстановить его работоспособность поможет только перезагрузка компьютера.

На этом предварительная настройка сервера приложений закончена.

2.3.4. Установка и настройка веб-портала

И наконец последний шаг — устанавливаем Web-портал. На этом шаге во время установки у меня появилась ошибка — попытка перезаписать уже существующие файлы, идущие в пакете **libfreerdp1**. Поэтому сначала удалим этот пакет:

```
apt-get remove --purge libfreerdp1 libfreerdp-plugins-standard
```

Теперь со спокойной душой и чистой совестью устанавливаем Web-портал:

```
apt-get install ulteo-ovd-web-client ulteo-ovd-guacamole
```

Во время установки будет выведен вопрос о том, нужно ли подключать портал к существующему менеджеру сессий, или же использовать его как общий веб-портал для нескольких менеджеров сессий. Здесь выбираем *Yes*, и на следующем шаге необходимо указать адрес менеджера сессий.

Последний пакет предназначен для доступа клиентов через HTML5 без использования Java. Это бывает необходимо в тех случаях, когда среду Java установить невозможно, например, в мобильных устройствах. Но во время установки скрипты могут отрабатываться некорректно, и после этого в окне веб-клиента в строке *Type* будет доступна только Java; чтобы задействовать HTML5, нужно отредактировать файл **/etc/ulteo/webclient/config.inc.php**, раскомментировав в нём строку `define('RDP_PROVIDER_HTML5_INSTALLED', true);`, для этого достаточно выполнить команду:

```
sed -i "s@// define('RDP_PROVIDER_HTML5_INSTALLED', true);@define('RDP_PROVIDER_HTML5_INSTALLED', true);@g" /etc/ulteo/webclient/config.inc.php
```

На этом установка и базовая настройка сервера Ulteo OVD завершена.

2.3.5. Настройка веб-клиента

Для доступа к веб-порталу можно использовать любой браузер с установленной средой Java.

Теперь после проделанных настроек открываем браузер на любом компьютере и вводим адрес Web-портала http://SERVER-IP/ovd/.

Поскольку веб-портал сделан на базе Java, то при первом входе появится запрос на запуск Java-приложений. Необходимо разрешить постоянное выполнение и сохранить выбор, чтобы запрос не появлялся при каждом входе. Если использование Java невозможно, то в этом случае доступ к приложениям будет осуществляться при помощи HTML5.

После успешной проверки совместимости системы откроется окно с логотипом Ulteo и вводом логина и пароля. По умолчанию логин вводится вручную, но также есть возможность выбрать его из списка, если в менеджере сессий открыть меню *Настройки → Настройки Веб-интерфейса* и в строке *Показать список пользователей* выбрать вариант **Да** и далее *Сохранить*. Если сервер OVD будет работать только в локальной сети без доступа в интернет, то удобнее настроить второй вариант, если же доступ будет из интернета, то лучше оставить всё как есть.

Чуть ниже строк ввода логина и пароля нажимаем на строку *Расширенные настройки* (*Advanced settings*), и развернётся меню с дополнительными настройками.

В опции *Режим* (*Mode*) доступны настройки *Рабочий стол* (*Desktop*) и *Портал* (*Portal*). В режиме *Рабочего стола* при подключении загружается обычный рабочий стол с оболочкой XFCE4 и ярлыками опубликованных приложений. В режиме *Портала* в окне браузера загружается только список опубликованных приложений и файловый менеджер для управления файлами, а каждая программа выполняется в отдельном окне.

В следующей опции — *Type* — выбираем тип сессии: *Java* или *HTML5*. Хочу заметить, что сессия *HTML5* работает только в режиме *Портала*, поэтому её рекомендуется выбирать только в крайних случаях, когда использование среды Java невозможно по каким-либо причинам, например, на мобильных устройствах.

При выборе режима Рабочего стола в настройках доступна дополнительная опция — *Полный экран* (*Fullscreen*), и варианты — *Да* или *Нет*. В варианте по умолчанию *Нет* загрузка рабочего стола происходит в окне браузера, при выборе варианта *Да* сессия запускается в полноэкранном режиме. Как переключаться между полноэкранным режимом и рабочим столом компьютера, я пока не нашёл.

Соответственно в опции *Язык* (*Language*) выбираем наш родной *Русский*, а в *Раскладка клавиатуры* (*Keyboard layout*) рекомендую выбрать раскладку, используемую на клиентском компьютере по-умолчанию. Стоит отметить, что при выборе типа сессии *HTML5* опция *Раскладка клавиатуры* недоступна.

После проделанных настроек нажимаем заветную кнопку *Подключить* (*Connect*) и приступаем к работе.

Вывод можно сделать только один: это реально работает! Проброс ресурсов работают, со звуком проблем нет, административная оболочка интуитивно понятна и позволяет провести тонкие настройки сервера, ничего не зависает.

Недостатка пока было обнаружено два:

1) если вы будете использовать сервер OVD на одном компьютере, то для него нужна конфигурация помощнее;

2) система довольно долго завершает сессии, а закрывать браузер до полного завершения сессии нежелательно.

И даже невзирая на ограничения, наложенные в плане использования операционных систем, данное решение можно смело рекомендовать к использованию в масштабах любой организации.

2.4. LX-SERVER

LX-Server — это коммерческий терминальный сервер от словацкой компании Thinstuff, позволяющий подключаться к рабочим столам на Linux-сервере по протоколу RDP. Стоимость лицензий составляет от $46 на одно подключение до $359 на безлимитное количество подключений. Более подробно с прайсом можно ознакомиться на официальном сайте.

В состав продукта входит собственный веб-сервер, через который запускается административная оболочка, сервер баз данных PostgreSQL, и даже собственный X-сервер с оболочкой TWM, который может быть использован в случаях, когда на сервере не установлена графическая оболочка.

К сожалению продукт больше не развивается и не поддерживается уже несколько лет, в демо-режиме позволяет подключиться одному пользователю без ограничений по времени при условии некоммерческого использования. Последняя версию — 1.2.1 build 5366 от 17 ноября 2008 года — можно скачать на странице загрузки, либо воспользоваться прямой ссылкой на установочный файл (~60МБ) и инструкцию.

Скачиваем установочный файл в домашний каталог и запускаем установку в терминале:

```
wget -c https://www.thinstuff.com/releases/lxserver-1.2.1-5366.sh
sudo bash ~/lxserver-1.2.1-5366.sh
```

Появится приветствие с дополнительной информацией, здесь нажимаем *Enter*, в следующем окне для установки вводим *i* (от английского слова *install*) и нажимаем два раза *Enter*, далее знакомимся с лицензионным соглашением, пролистывая его клавишей *Enter*, и в самом конце страницы соглашаемся с ним: вводим *yes* и нажимаем *Enter*. Программа выдаст полную информацию о нашей системе и запросит разрешения на её анонимную отправку производителю, здесь опять пролистываем список при помощи *Enter*, а в самом конце, если ничего не хотим отправлять, вводим *no* и нажимаем *Enter*. Начнётся процесс установки.

Программа со всеми компонентами устанавливается в каталог **/opt/thinstuff/rdpserver**, другой каталог выбрать невозможно. Такое решение позволяет установить и запустить только одну копию сервера.

И именно на этом этапе в некоторых дистрибутивах Linux появляется ошибка. Дело в том, что в установочном комплекте идёт собственный веб-сервер Lighttpd и сервер баз данных PostrgreSQL, программа установки проверяет, запущены ли эти компоненты. Но они на деле оказываются не загружены, поскольку на этапе запуска веб-сервер не может найти файл конфигурации, а сервер PostgreSQL — создать базу данных. Попытки правки скриптов и запуска этих сервисов вручную с последующим общим рестартом LX-Server'a тоже никаких результатов не дают. В частности такое у меня происходило при попытке установить сервер на Debian. На последнюю версию Ubuntu всё установилось и заработало нормально.

В конце установки программа просканирует систему на наличие графических оболочек и выдаст запрос, какой оконный менеджер будет запускаться при входе в систему. Можно указать 4 менеджера: KDE, Gnome, IceWM или встроенный TWM. Если на вашем сервере установлен старый добрый Gnome2, то для его использования вводим *g*, нажимаем *Enter,* далее — пароль root'a и нажимаем *Enter*. Затем на запрос об автоматическом старте системы при запуске компьютера вводим yes, нажимаем Enter.

ВНИМАНИЕ! С новым Gnome-Shell сервер работать не будет! Хотя он и распознает его как оболочку Gnome, но при попытке подключения клиент будет вылетать сразу после ввода логина и пароля. В случае необходимости можно указать потом другую оболочку в настройках через веб-интерфейс.

После полной установки и запуска сервера открываем любой браузер и вводим в адресную строку https://localhost или IP-адрес сервера. Обратите внимание: в адресе обязательно указываем протокол **https**. Для входа в оболочку администрирования используем логин и пароль root'a. Для подключения к серверу используем логин и пароль любого системного пользователя.

С подключениями к серверу тоже возникли проблемы: клиент Remmina версии 0.9.99.1 подключиться к серверу не смог, с более старой версией 0.8.3 особых проблем не возникло. Также нормально смогли подключиться Gnome-RDP и TSClient. Но даже несмотря на это, во время сеанса возникали проблемы, например: все программы и меню были только на английском, хотя при старте был указан русский язык; раскладка клавиатуры в окне ввода логина и пароля не переключалась;

русские буквы в именах файлах и папок отображались «кракозябрами»; некоторые приложения не запустились вообще, при попытке установить сетевой принтер через утилиту *system-config-printer* клиент вылетел. При использовании Windows-клиента локальные ресурсы на сервер не пробросились (звук, принтеры, порты, папки). Может быть некоторые из этих проблем решить и можно, но я не стал разбираться с ними.

Более подробно об установке и настройке этого сервера можно почитать на сайтах hrafn.me и citkit.ru.

Выводы печальны: сервер больше не поддерживается, слишком много недоработок, которые уже никогда устранены не будут. В общем установить его можно только чисто из любопытства, но на практике на современных системах применять не рекомендуется.

3. ПРОТОКОЛ XDMCP

Протокол XDMCP появился в 1989 году и предназначается для предоставления стандартного механизма для запроса сервиса входа в систему автономным дисплеем. Протокол передаёт данные в незашифрованном виде, поэтому не рекомендуется для применения в сетях общего доступа без дополнительных модулей шифрования.

Для подключения к удалённому рабочему столу по протоколу XDMCP в операционной системе Linux можно использовать любой менеджер загрузки: GDM, KDM, MDM и т. д., клиент Remmina с дополнительным плагином *remmina-plugin-xdmcp*; в операционной системе Windows можно воспользоваться программой Xming, о которой более подробно будет рассказано в разделе 4.

3.1. СЕРВЕР LTSP

Сервер LTSP (Linux Terminal Server Project) предназначен для подключения к нему «тонких» клиентов, не имеющих собственной операционной системы. В качестве «тонких» клиентов могут выступать любые, даже очень старые компьютеры без жёсткого диска, главное условие — сетевая карта должна поддерживать загрузку по протоколу PXE. Как сказано в описании к нему, начиная с 5-й версии, сервер LTSP поддерживает работу и с «толстыми» клиентами.

Принцип действия таков: в прошивку сетевой карты встроен загрузчик PXE-Linux, который запускается при старте компьютера и делает запрос к DHCP-серверу. Получив от него IP-адрес и адрес TFTP-сервера, он делает запрос к этому серверу на загрузку ядра Linux. После того, как ядро Linux загружено в оперативную память, всё управление уже передаётся ему. Ядро Linux снова делает запрос к DHCP-серверу, в результате которого получает свой IP-адрес, адрес NFS-сервера, на котором находится корневая файловая система, а также путь к этой файловой системе на диске. После монтирования корневой файловой системы NFS производится вызов файла **/sbin/init**, который находится на ней, и дальнейшее управление передаётся ему.

Теперь перейдём к практике. Устанавливаем стандартный LTSP-сервер и все необходимые пакеты для минимальной базовой настройки:

```
sudo apt-get install ltsp-server ltsp-server-standalone ltsp-manager tftpd-hpa nfs-kernel-server nfs-common isc-dhcp-server openssh-server
```

Многие зависимости у вас установятся вместе с LTSP-сервером, но на всякий случай мы их укажем. В более старых версиях систем Ubuntu и Debian некоторые пакеты могут быть установлены с другими именами.

Теперь настраиваем всё по порядку. Настройку будем проводить по самой простой схеме, когда все сервисы располагаются на одном сервере. Это конечно не есть хорошо, но в пределах одной небольшой организации вполне допустимо.

Если необходимо, чтобы DHCP-сервер работал на определённой сетевой карте, открываем файл:

```
sudo editor /etc/default/isc-dhcp-server
```

Раскомментируем строку и прописываем в неё нужный интерфейс:

```
INTERFACES="eth1"
```

Теперь для этой сетевой карты необходимо прописать статический IP в файле **interfaces**:

```
sudo editor /etc/network/interfaces
```

В примере ниже статический IP-адрес присвоен сетевой карте **eth1**:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
#iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 192.168.0.1
netmask 255.255.255.0
network 192.168.0.0
```

Настраиваем DHCP-сервер, от которого PXE-загрузчик сетевой карты будет получать IP-адрес. Открываем на редактирование файл:

sudo editor /etc/dhcp/dhcpd.conf

Прописываем следующие настройки:

```
authoritative;

subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.20 192.168.0.250;
  option domain-name-servers 192.168.0.1;
  option domain-name "unix.nt";
  option server-name "ltsp0.unix.nt";
  option routers 192.168.0.1;
  option broadcast-address 192.168.0.255;
  option subnet-mask 255.255.255.0;
  default-lease-time 600;
  max-lease-time 7200;
  option root-path "localhost:/opt/ltsp/i386/";
# если TFTP располагается на другом сервере, раскомментируем строку
#ниже и прописываем его адрес:
# next-server 192.168.0.2;
# get-lease-hostnames true;
  filename "ltsp/i386/pxelinux.0";
}
```

В данном примере клиентам будут выделяться IP-адреса в диапазоне 192.168.0.20 — 192.168.0.250, в качестве DNS-сервера и шлюза используется этот же сервер, опции ***root-path*** и ***filename*** имеют непосредственное отношение к бездисковой загрузке.

После проделанных настроек перезапускаем DHCP-сервер:

sudo service isc-dhcp-server restart

Следующий шаг — настройка TFTP-сервера. Как правило он начинает работать сразу после установки и каких-либо особых настроек не требуется, но с ним вскрылся один довольно неприятный момент: после перезагрузки сервер автоматически не стартует, а при выполнении команды ***service tftpd-hpa start*** выдаётся сообщение, что сервер уже запущен, хотя на самом деле его в процессах нигде нет. Поэтому я поступил следующим образом: открываем файл **rc.local**:

sudo editor /etc/rc.local

и перед строкой ***exit 0*** прописываем команду:

```
service tftpd-hpa restart
```

На всякий случай можно убедиться, что TFTP-сервер прослушивает 69-й порт:

sudo netstat -lnp | grep :69

Третий шаг — создание клиентского образа LTSP и настройка NFS-сервера.

В большинстве систем на базе Debian и Ubuntu (например, LinuxMint) при запуске сборки будет выдаваться ошибка, поскольку LTSP использует информацию о системе из файла **/etc/lsb-release**, поэтому в данном файле нужно указать родительский дистрибутив. Например, для LinuxMint13, сделанного на базе Ubuntu 12.04, исправления будут выглядеть так:

sudo cp /etc/lsb-release /etc/lsb-release.old
sudo sed -e 's/ID=.*/ID=Ubuntu/g' /etc/lsb-release -i
sudo sed -e 's/CODENAME=.*/CODENAME=lucid/g' /etc/lsb-release -i

Совет взят с форума rosinka.rosix.ru.

Обновляем список пакетов в репозитории:

```
sudo apt-get update
```

Затем запускаем сборку клиентского образа для 32-хбитной архитектуры:

```
sudo ltsp-build-client --arch i386
```

Образ создаётся в каталоге **/opt/ltsp/i386**. В ходе выполнения команды также создаётся образ с конфигурационными файлами в каталоге **/var/lib/tftpboot/ltsp/i386/**. Конфигурационный файл **default** сетевого загрузчика **pxelinux** находится в **/var/lib/tftpboot/ltsp/i386/pxelinux.cfg/**.

После успешной установки согласно многих источников нужно открыть доступ к корневой файловой системе образа через NFS-сервер. Поскольку у меня все сервисы располагаются на одном компьютере, то всё заработало и без него. Если же сервисы будут работать на разных серверах, то это всё-таки необходимо будет сделать. Итак, открываем на редактирование файл:

```
sudo editor /etc/exports
```

В него прописываем строку:

```
/opt/ltsp/i386    *(ro,no_root_squash,async,subtree_check)
```

В данном случае доступ открыт в режиме read-only (только чтение). Не рекомендуется открывать доступ к образу в режиме read-write (чтение и запись)!

Теперь даём команду, чтобы демон NFS-сервера перечитал файл конфигурации:

```
sudo invoke-rc.d nfs-kernel-server reload
```

На этом настройку можно считать законченной.

В документации сказано, что использование **initrd.img** необязательно при использовании NFS-сервера. Однако не всегда получается примонтировать NFS-систему из **initrd.img**. Поэтому в таких случаях желательно предоставить пользователю возможность выбора. Для этого редактируем файл:

```
sudo editor /var/lib/tftpboot/ltsp/i386/pxelinux.cfg/default
```

Прописываем в него следующее содержимое:

```
DEFAULT LINUX
DISPLAY boot.msg
TIMEOUT 30
PROMPT 1

LABEL LINUX
KERNEL vmlinuz-2.6.18-6-486
APPEND root=/dev/nfs nfsroot=192.168.0.1:/opt/ltsp/i386/

LABEL LINUX-OLD
KERNEL vmlinuz-2.4.34.lan
APPEND nfsdir=192.168.0.1:/opt/ltsp/i386/ lang=ru ramdisk_size=100000 root=/dev/nfs
rw nfsroot=192.168.0.1:/opt/ltsp/i386/ ip=dhcp

LABEL LINUX-i
KERNEL vmlinuz-2.6.18-6-486
APPEND root=/dev/nfs nfsroot=192.168.0.1:/opt/ltsp/i386/ initrd=initrd.img-2.6.18-6-
486

LABEL LINUX-OLD-i
KERNEL vmlinuz-2.6.18-6-486
APPEND nfsdir=192.168.0.1:/opt/ltsp/i386/ lang=ru ramdisk_size=100000 root=/dev/nfs
rw nfsroot=192.168.0.1:/opt/ltsp/i386/ ip=dhcp initrd=initrd.img-2.6.18-6-486
```

Все IP-адреса и ядра **vmlinuz** и **initrd** исправляем на свои.

Создаём файл **boot.msg**:

```
sudo touch /var/lib/tftpboot/ltsp/i386/boot.msg
sudo editor /var/lib/tftpboot/ltsp/i386/boot.msg
```

Прописываем в него произвольное содержимое, которое будет отображаться при загрузке, например:

```
network boot
choose kernel
-------------
LINUX
LINUX-i
```

Напоследок на сервере тонких клиентов создаём учётные записи пользователей.

Теперь пробуем загрузиться с любого клиентского места, выставив в БИОС'е загрузку с сетевой карты по протоколу PXE. В итоге мы должны увидеть на экране окно приглашения с вводом логина и пароля. Я думаю, можно извернуться и прописать для каждого пользователя логин и пароль и привязать их к MAC-адресу соответствующей сетевой карты, в итоге пользователю будет достаточно включить компьютер и дождаться загрузки рабочего стола. Но в такие тонкости я не вдавался.

В условиях локальной сети всё работает относительно быстро (хотя X2Go и FreeNX на мой взгляд работают быстрее), ресурсы пробрасываются без проблем, флэшки и звук работают сразу без лишних «плясок с бубном», но видео работает довольно медленно. С другой стороны, сервер в первую очередь предназначен для работы, а не для просмотра видеороликов. Поэтому данный вариант сервера можно рекомендовать для работы в локальной сети со скоростью передачи данных не ниже 100 Мб/с.

И ещё несколько советов:

1) при смене IP-адреса сервера необходимо выполнить команду:

```
sudo ltsp-update-sshkeys
```

2) при изменении ядра на сервере или устранении неполадок выполняем команду:

```
sudo ltsp-update-image
```

3) для администрирования сервера можно использовать графическую утилиту ***ltsp-manager***.

Конечно же возможности LTSP-сервера на этом не ограничиваются, для более крупных предприятий можно настроить балансировку нагрузки, кластеры и т. д., для этого установив соответствующие пакеты.

ВНИМАНИЕ! В репозиториях многих систем доступны пакеты **ltsp-client** и **ltsp-client-core**, которые содержат необходимые компоненты для монтирование ресурсов, работы звука и т. д. Их ни в коем случае нельзя устанавливать на компьютер в обычном режиме! Они устанавливаются автоматически в клиентском образе создаваемой системы.

При написании данного раздела использованы материалы с сайтов xgu.ru и pro-spo.ru.

3.2. СЕРВЕР XDMCP+VNC

Как было сказано в п.1.2, VNC-серверы не предназначены для терминального доступа, поскольку запускаются под профилем пользователя при входе в систему. Более того: VNC-сервер по умолчанию занимает порт 5900, и если к одному и тому же порту попытаются подключиться 2 клиента одновременно, то они либо будут работать одновременно в одном и том же сеансе, мешая друг другу, либо одного из клиентов «выбросит» из сессии.

Выхода из этой ситуации два:

1) автоматом запустить несколько сессий, каждой из которых присвоить свой порт, при этом будут загружены неиспользуемые сессии, а проблема с одним портом не решается;

2) настроить супердемон inetd, чтобы при подключении на порт 5900 автоматически запускался VNC-сервер.

Первый пункт нас конечно же не устраивает, поэтому выбираем второй. Рассмотрим на примере пакета *vnc4server*. Итак, устанавливаем сначала нужные пакеты:

```
sudo apt-get install vnc4server openbsd-inetd
```

Запускаем VNC-сервер:

```
sudo vnc4server
```

При запуске программа попросит задать пароль на соединение с нашим сервером.

Далее редактируем файл */etc/inetd.conf*:

```
sudo gedit /etc/inetd.conf
```

Прописываем в конец файла строку:

```
5900 stream tcp nowait USERNAME /usr/bin/Xvnc Xvnc -inetd -query localhost -once
-depth 24 securitytypes=none
```

В данном случае вместо **USERNAME** надо прописать пользователя, от чьего имени будет запускаться VNC-сервер. Параметр **-query localhost** указывает VNC-серверу, что авторизация будет

происходить локально, используя менеджер загрузки (GDM, KDM, MDM и т.д.). Помимо этого можно «поиграть» с настройками **geometry**, которые зададут разрешение сервера VNC. Все параметры можно увидеть, набрав в консоли **Xvnc -h**

И наконец последний шаг: настроить менеджер загрузки, чтобы он принимал входящие подключения по протоколу XDMCP. Для настройки менеджеров GDM, KDM и MDM открываем на редактирование файл **/etc/gdm/gdm.conf**, **/etc/kde4/kdmrc** или **/etc/mdm/mdm.conf** соответственно и в секцию ***[xdmcp]*** прописываем или раскомментируем строку ***Enable=true***. В итоге должно получиться следующее (на примере менеджера MDM):

```
[xdmcp]
Enable=true
```

После проделанных настроек перезагружаем компьютер и пробуем подключиться при помощи любого XDMCP-клиента, например, Remmina.

И как всегда у данного сервера довольно много недостатков: невысокая скорость отрисовки окон, невозможность восстановить сессию в случае обрыва связи, высокие требования к сети. Про проброс ресурсов и буфер обмена можно забыть. Поэтому данный способ рекомендовать к широкому применению на предприятиях нельзя, а в статью он включен только для ознакомления и общего развития.

При написании данного раздела использованы материалы с сайта xlabz.org.

4. ПРОТОКОЛ SSH

Как уже было сказано выше, операционная система Linux позволяет запускать программы на клиентском компьютере по протоколу SSH без загрузки рабочего стола. Получается как бы «бесшовный» терминал, аналогичный режиму «rootless» в сервере FreeNX. Это решение работает немного быстрее, чем VNC, но всё-таки ощутимо медленнее, чем другие терминальные серверы (NX и RDP), поскольку в данном случае идёт шифрование и компрессия трафика. Поэтому такое решение можно использовать только в локальных сетях со скоростью 100мб/с. Даже через WiFi со скоростью 54мб/с работа будет не очень комфортной, не говоря уже о работе через интернет.

Все настройки довольно простые, для этого на сервере достаточно установить пакет **ssh** или **openssh-server**, а на клиентском компьютере под управлением Linux достаточно пакета **openssh-client**. После установки на сервере откройте файл конфигурации **/etc/ssh/sshd_config** и проверьте, чтобы следующие строки были раскомментированы, если их нет, пропишите вручную:

```
X11Forwarding yes
X11DisplayOffset 10
```

Далее нужную программу можно запустить командой:

```
ssh -XC USERNAME@IP_SERVER program
```

В данной команде ключ **-X** — передавать изображение, **-C** — компрессия данных (может снижать скорость передачи), **USERNAME** — имя пользователя на сервере, под чьим профилем будет запущена программа, **IP_SERVER** — IP-адрес нашего сервера, **program** — команда запуска программы на сервере. После нажатия клавиши *Enter* вводим пароль пользователя и работаем. Только учтите, что все ресурсы, с которыми работает программа (папки, принтеры и т. д.) расположены на сервере. Дабы пользователь не вводил каждый раз длинную и непонятную ему команду, можно создать запускаемый скрипт, прописать в него эту команду и поместить на рабочий стол.

Чтобы сервер не запрашивал каждый раз пароль, можно добавить клиентский компьютер в доверенные, для этого сначала на клиентском компьютере создаём ключи под профилем пользователя:

```
ssh-keygen
```

Ключи сохранятся в каталоге **/home/USERNAME/.ssh/**. Теперь открытую часть ключа *id_rsa.pub* нужно скопировать на сервер в домашний каталог нужного пользователя:

```
ssh-copy-id USERNAME@IP_SERVER
```

Всё делаем под профилем пользователя (не root'а!). На сервере должен появиться ключ **authorized_keys**, расположенный в каталоге **/home/USERNAME/.ssh/**. Также на сервере проверьте настройки **/etc/ssh/sshd_config**, чтобы строка **AuthorizedKeysFile** была раскомментирована и в ней прописано указанное выше имя ключа, и если оно другое (например, **authorized_keys2**), исправить на нужное и перезагрузить сервер.

С операционной системой Windows всё гораздо сложнее: для работы по протоколу SSH нужно установить дополнительно две программы: PuTTY и Xming. Скорость работы через эти программы будет ниже, чем в системе Linux.

ВНИМАНИЕ! Материал, описанный ниже, взят с сайтов bozza.ru и debback.blogpost.ru, но настроить работающую систему согласно этих описаний у меня не получилось! В причинах пока что разбираться не стал, поэтому привожу материал по принципу «как есть».

Сначала скачиваем и настраиваем клиента SSH — PuTTY. Найти его можно на официальном сайте, а русскую сборку — на putty.org.ru. Далее запускаем **putty.exe**, открываем меню **Connection → SSH → X11**, ставим галочку ***Enable X11 Forwarding***, а в строке ***X Display Location*** вводим IP-адрес клиентского Windows-компьютера, с которого мы подключаемся к Linux-серверу. Чтобы русские буквы в сессии отображались корректно, необходимо установить правильную кодировку, для этого открываем меню **Window → Translation** и вверху в секции *Character set translation* из выпадающего списка выбираем ***UTF8***. И наконец переходим во вкладку **Session** и выполняем настройки: в строке ***Host Name (or IP address)*** прописываем IP-адрес нашего сервера, в строке ***Port*** по умолчанию оставляем **22** (если только вы его не меняли на сервере), чуть ниже в секции ***Connection Type*** должно быть выбрано **SSH**, в строке ***Saved Session*** вводим имя нашей сессии, например, **Xming**, и нажимаем справа ***Save***. Теперь в списке выбираем только что созданную сессию, нажимаем ***Open*** и вводим логин и пароль пользователя на сервере. На этом настройки PuTTY закончены, оставляем сессию в открытом виде (можно окошко просто свернуть), переходим к настройкам Xming.

Скачиваем установочный файл программы с официального сайта. Последняя версия, доступная для публичного скачивания на момент написания статьи, — 6.9.0.31 от 4 мая 2007 года. Более новые версии пока не доступны. После установки запускаем программу настройки ***Xlaunch.exe***.

На первом шаге выбираем способ интеграции с рабочим столом Windows: каждое приложение в своём окне, полноэкранный режим, все программы в одном окне либо всё в одном окне без заголовка. Рекомендую первый вариант. Также в нижней строке ***Display number*** указываем номер дисплея, например, 1, поскольку на сервере дисплей 0 как правило бывает занят запущенным X-сервером.

На втором шаге предлагается запускать Xming без клиента, автоматически запускать какую-либо программу при старте Xming, либо подключиться к серверу по протоколу XDMCP. Поскольку третий вариант в данном случае нам не нужен, второй — тоже (клиент PuTTY уже настроен и запущен), остаётся первый вариант.

На третьем шаге предлагается настроить дополнительные параметры соединения. Галочку ***Clipboard*** оставляем, она активирует буфер обмена, в строке *Additionar parameters for Xming* прописываем строку:

```
-dpi 96 -xkblayout us,ru -xkbvariant ,winkeys
```

Здесь ***-dpi 96*** — размер шрифтов, ***-xkblayout us,ru*** — раскладки клавиатуры, используемые в сессии, ***-xkbvariant basic,winkeys*** — параметры для раскладок.

И наконец на последнем шаге нажимаем ***Save Configuration*** и нажимаем «Готово». В системном трее появится значок Xming, означающий ожидание подключения. Теперь переходим в окошко с PuTTY и пробуем запустить из командной строки любую программу, например, текстовый редактор:

```
gedit &
```

В конце команды обязательно ставим амперсанд &, означающий, что программа должна запускаться в фоновом режиме. Если всё прошло успешно, мы увидим на рабочем столе окошко текстового редактора Gedit.

ЗАКЛЮЧЕНИЕ

Из всего описанного выше я бы мог дать следующие рекомендации по использованию терминальных решений в системе Linux.

Конечно же самым быстрым и самым лучшим на мой взгляд является коммерческий NX Server от итальянской компании NoMachine, работающий по протоколу NXv4, но за него надо платить.

Из бесплатных вариантов в первую очередь я бы рекомендовал X2Go. Особых недостатков у него не обнаружено, скорость работы приемлема, хотя и ниже, чем у FreeNX, но всё начинает рабо-

тать сразу «из коробки» без дополнительных настроек. Также у проекта очень хорошая поддержка, оперативно выходят версии под новые системы.

Также смело можно рекомендовать проект Ulteo OVD. Устанавливается без проблем, очень удобная оболочка администрирования, скорость работы нормальная, проброс ресурсов работает. Конечно же собственный клиент повышает удобство работы и добавляет функционал, но он доступен только за отдельную ежегодную плату.

Хотя с натяжкой, но всё-таки можно рекомендовать сервер FreeNX, а в частности — сборку RX@Etersoft от компании «Этерсофт». К сожалению у проекта довольно много недостатков, не все заявленные функции работают как надо, да и настройка сервера потребует определённых усилий и квалификации специалиста, чтобы решить проблемы, возникающие в процессе настройки и эксплуатации. Плюс неоперативно выходят сборки под новые системы, а некоторые ветки проекта заброшены и не обновляются, например, оригинальный проект FreeNX и репозиторий FreeNX и NeatX для Ubuntu на launchpad.net. Но за высокую скорость работы эти недостатки можно и простить.

На четвёртое место я бы поставил связку Xrdp + X11Rdp; хотя в ней есть некоторые недостатки: проблемы с пробросом ресурсов от клиента на сервер, отсутствие утилиты администрирования, работа с ограниченной цветовой гаммой, но она работает более-менее стабильно и не требует особых усилий в настройке, поэтому в рамках небольшого предприятия её использовать можно.

Также не возникает проблем с сервером «тонких клиентов» LTSP: в терминальной сессии работает проброс ресурсов, звука, принтеров, однако довольно сильно нагружается сеть, требуется высокоскоростное соединение и по умолчанию не шифруется трафик, поэтому его можно рекомендовать только для использования во внутренних локальных сетях, если что-то не получается с X2Go.

Такие варианты, как Xpra, Xrdp+VNC и SSH я бы рекомендовал в качестве резервных, когда ни один из рекомендованных выше невозможно использовать по каким-либо причинам.

И наконец устаревшие и заброшенные проекты 2X Terminal Server, NeatX, XDMCP+VNC и LX-Server я вообще не рекомендую применять ни в каком виде!