Автор:
22.12.10 18:19 - Последнее обновление 07.01.12 14:15

Загрузка Linux (Kubuntu 7.10) по сети.

Возникла необходимость загрузки линуксовых тонких клиентов по сети, используя PXE в обучающих классах. Выбор линукса в силу некоторых причин пал на Kubuntu 7.10, а загружаться они у нас должны с сервера FreeBSD. Линуксы должны грузиться в read only, для того чтобы пользователи не могли испортить систему, что несколько усложняет ситуацию.

Итак, с чего начнем... Начнём с установки Kubuntu на жеский диск обычного компьютера. Эта процедура крайне несложная и расписывать её смысла нет никакого. По умолчанию, система ставится целиком в /, то есть диск на партиции не разбивается и вся структура каталогов находится в корне, что в принципе, нас устраивает. Единственное, что можно сделать при такой установке отключить swap раздел, так как использовать мы его все равно не будем. И еще для установки системы не стоит выделять более 3-4 Гб места на жестком диске, если мы хотим скопировать образ диска целиком. Если же планируется копировать файлы установленной системы, тогда размер раздела не принципиален.

Теперь пора разобраться как у нас будет загружаться система по сети.
1. Компьютер (клиент) стартует и, если сетевая карта поддерживает PXE, отправляется DHCPREQUEST на DHCP-сервер.
2. DHCP-сервер определяет клиента по MAC-адресу и выдает ему ip-адрес и указание (filename) откуда брать загрузчик линукса (pxelinux.0).
3. Клиент получает указание filename и пытается скачать его используя протокол TFTP
4. Скачав загрузчик, клиент может скачать ядро (vmlinuz) и структуру системы (initrd) из той же директории, что и загрузчик. используя тот же TFTP.
5. После того как vmlinuz и initrd скачаны и запущены, остальная часть системы монтируется по NFS.
В результате мы имеем рабочую систему, работающую с NFS разделом как с собственным жестким диском.

Итак система у нас установлена.
Теперь необходимо её настроить. Собственно здесь мы можем убрать ненужные рюшечки, для ускорения загрузки и работы системы, удалить ненужные пакеты и добавить нужные (например Firefox + flash), сделать автологин в KDE, записать MAC-адрес. Самое главное - это установить пакеты portmap и nfs-common для работы с шарами nfs. В нашем случае это делается командой из консоли **sudo apt-get install portmap nfs-common**
После того как мы настроили систему под себя пора скопировать её образ.

Смотрим вывод команды mount:

```
user@diskless:~$ mount
```

```
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nodev)
/sys on /sys type sysfs (rw,noexec,nodev)
-----------------------------------------------
***********************************************
-----------------------------------------------
```

в нашем конкретном случае Kubuntu встала целиком на раздел /dev/sda1 его и будем копировать.

Загружаемся с LiveCD можно того же Kubuntu и, не монтируя /dev/sda1, начинаем копировать весь раздел на... собственно куда нам надо.
Можно скопировать на другой раздел:

```
ubuntu@livecd# sudo cp /dev/sda1 /mnt/newhdd/obraz_kubuntu.image
```

а можно и сразу на наш сервер через ssh

```
ubuntu@livecd# sudo scp /dev/sda1 admin@server:obraz_kubuntu.image
```

Если наш раздел составляет 3-4 Гб, то по локальной сети образ перепишется минут за 5-10, что очень даже недолго.

Так... переписали. Пробуем смонтировать:

```
root@server# mdconfig -a -t vnode -f /home/admin/obraz_kubuntu.image -u 9
md9
root@server# mount_ext2fs /dev/md9 /share/kubuntu/
root@server# ls -la /share/kubuntu/
total 102
```

```
drwxr-xr-x  22 root  wheel   4096 Apr 15 12:44 .
drwxrwxr-x   5 root  wheel    512 Apr  9 19:32 ..
drwxr-xr-x   2 root  wheel   4096 Apr 10 13:11 bin
drwxr-xr-x   3 root  wheel   4096 Apr  8 13:01 boot
lrwxrwxrwx   1  root  wheel     11 Apr  8 12:27  cdrom -> media/cdrom
drwxr-xr-x   4 root  wheel   4096 Oct 16 22:11 dev
drwxr-xr-x 107 root  wheel   4096 Apr 14 20:33 etc
drwxr-xr-x   3 root  wheel   4096 Apr 14 20:28 home
drwxr-xr-x   2 root  wheel   4096 Oct 16 22:04 initrd
lrwxrwxrwx   1  root  wheel     33 Apr  8 13:01  initrd.img -> boot/initrd.img-2.6.22-14-generic
drwxr-xr-x  17 root  wheel   8192 Apr  9 12:21 lib
drwx------   2 root  wheel  16384 Apr  8 12:27 lost+found
drwxr-xr-x   3 root  wheel   4096 Apr  8 14:06 media
drwxr-xr-x   2 root  wheel   4096 Oct  8  2007 mnt
drwxr-xr-x   2 root  wheel   4096 Oct 16 22:04 opt
drwxr-xr-x   2 root  wheel   4096 Oct  8  2007 proc
drwxr-xr-x   7 root  wheel   4096 Apr 14 14:42 root
drwxr-xr-x   2 root  wheel   4096 Apr  9 14:05 sbin
drwxr-xr-x   2 root  wheel   4096 Oct 16 22:04 srv
drwxr-xr-x   2 root  wheel   4096 Oct  4  2007 sys
drwxrwxrwt   2 root  wheel   4096 Apr  9 05:53 tmp
drwxr-xr-x  12 root  wheel   4096 Apr 14 15:00 usr
drwxr-xr-x   4 root  wheel   4096 Apr 10 17:45 var
drwxr-xr-x  14 root  wheel   4096 Apr 10 15:38 var2
lrwxrwxrwx   1  root  wheel     30 Apr  8 13:01  vmlinuz -> boot/vmlinuz-2.6.22-14-generic
```

Что ж прекрасно, имеем рабочую систему. Надо слегка её подредактировать.
Во первых, местный fstab

```
root@localhost# cat /share/kubuntu/etc/fstab
# /etc/fstab: static file system information.
#
#
proc            /proc           proc    defaults        0       0
/dev/nfs        /               nfs     defaults        1       1
#none           /tmp            tmpfs   defaults        0       0
#none           /var/run        tmpfs   defaults        0       0
#none           /var/lock       tmpfs   defaults        0       0
#none           /var/tmp        tmpfs   defaults        0       0
```

Самое главное для нас здесь - указание /dev/nfs.

Во вторых, необходимо внести изменение в /share/kubuntu/etc/initramfs-tools/initramfs.conf
BOOT=nfs

В третьих, для отключения swap правим /share/kubuntu/etc/sysctl.conf вставляем туда строчку
vm.swappiness = 0

В четвертых, можно подредактировать /share/kubuntu/etc/X11/xorg.conf и установить драйвер vesa, для корректной работы с разными видеокартами.

Для загрузки в read only этих настроек недостаточно, зато этого хватит для загрузки в read/write и тестирования имеющейся системы, а так же подшлифовки напильником, если что не так.

Теперь настраиваем наш FreeBSD сервер
1. Редактируем настройки DHCP. Файл /usr/local/etc/dhcpd.conf, надо добавить следующее

```
#diskless configuration
filename "pxelinux.0";
option root-path "192.168.0.1:/share/kubuntu";
```

Перезапускаем демон командой **/usr/local/etc/rc.d/isc-dhcpd.sh restart**

2. Настраиваем первичную загрузку по TFTP.
Проверяем /etc/rc.conf опция inetd_enable="YES" должна присутсвовать.
Редактируем /etc/inetd.conf раскоментируем строчку:

```
tftp    dgram   udp     wait    root    /usr/libexec/tftpd      tftpd  -l -s /tftpboot
```

Которая, будет запускать демон tftpd и расшаривать директорию /tftpboot на сервере. Перезапускаем inetd командой **/etc/rc.d/inetd restart**

3. Кладём в эту директорию
а) загрузчик линукса по PXE файл **pxelinux.0** из комплекта syslinux,
домашняя страница здесь ,
скачать можно с kernel.org здесь .
б) копируем в эту директорию ядро линукса

```
root@server# cp /share/kubuntu/boot/vmlinuz-2.6.22-14-generic /tftpboot/vmlinux-kubuntu
root@server# cp /share/kubuntu/boot/initrd.img-2.6.22-14-generic /tftpboot/initrd-kubuntu
```

в) Создаём директорию(!) /tftpboot/pxelinux.cfg и кладём туда файл default со следующим содержимым:

```
LABEL linux
KERNEL  vmlinuz-kubuntu
APPEND root=/dev/nfs initrd=initrd-kubuntu nfsroot=SERVER:/share/kubuntu ip=dhcp ro
```

где nfsroot=SERVER:/share/kubuntu - это NFS шара на машине SERVER (можно просто ip-адрес указать вместо имени)

4. создаём NFS шару для продолжения загрузки
а) проверяем /etc/rc.conf, строка **nfs_server_enable="YES"** должна присутствовать
б) Редактируем файл /etc/exports

```
/share/kubuntu    -maproot=0     -network 192.168.0.0 -mask 255.255.255.0
```

в) Перезапускаем mountd командой **killall -HUP mountd**

Пробуем запустить тонкого клиента, если все правильно было сделано, он загрузится. Но загрузится он с правами read/write. Теперь переходим к самому интересному, а именно построению клиента, который может работать полностью в read only режиме.

Не смотря на специфичность нашего дистрибутива, наверняка найдутся общие моменты и для других Linux систем. Конкретно с Kubuntu невозможно полностью загрузить систему в read only. Необходимы права на запись в разделы /var (самые важные /var/run,

/var/log, /var/lock), /tmp и в домашнюю директорию пользователя (предположим /home/user) от которого стартует KDE.
Вероятным решением этой проблемы является создание в оперативной памяти разделов и монтирование их в указанные директории. По умолчанию, ramdisk-и создаются при запуске initrd. В нашем случае по умолчанию все рамдиски равны 64 Mb, что нас очень даже устраивает. Поменять размер рамдиска можно добавив в файл /tftpboot/pxelinux.cfg/default в строку APPEND параметр ramdisk_size=16000, получатся рамдиски по 16Mb. Но, повторюсь, нас устроят как раз рамдиски на 64 Mb.

Использовать мы будем два рамдиска, первый под раздел /tmp, второй для /var и /home/user
Если с /tmp всё более-менее понятно, так как он и должен стартовать пустым, то что делать с /var и /home не своесм ясно. Не долго думая, было принято решение держать эти директории в другом месте, а при старте системы копировать их содержимое в рамдиск. Вообще не совсем понятно что в /var делает директория lib причем, весьма немаленькая. Убираем её на /usr а в /var делаем на неё симлинк.

Итак, предварительная настройка, можно сделать и на сервере:

```
root@server# cd /share/kubuntu
root@server# mv var/lib usr/local/varlib
root@server# cp -rp var usr/local
root@server# ln -s /usr/local/varlib var/lib
root@server# cp -rp home/user usr/local/var
root@server# rm -rf home/user
root@server# ln -s /var/user home/user
```

Таким образом, мы скопировали интересующие нас эталонные директории в usr/local линукс системы. Домашняя диретория пользователя будет храниться в /var на линуксовом рамдиске, а в директории /home будет присутсвовать симлинк на /var/user.
Теперь создадим скрипт в /share/kubuntu/etc/rcS.d/S00diskless, который будет подготовливать систему к загрузке в read only:

```
root@server# cat /share/kubuntu/etc/rcS.d/S00diskless
#!/bin/sh
#script for mounting diskless workstations
/sbin/mke2fs -q -m 0 /dev/ram0
/sbin/mke2fs -q -m 0 /dev/ram1
```

```
/bin/mount /dev/ram0 /var
/bin/mount /dev/ram1 /tmp

/bin/chown root:root /var
/bin/chmod 0755 /var
# Необходимо для корерктной загрузки системы
/bin/mkdir /var/tmp && chmod 4777 /var/tmp
/bin/mkdir /var/lock
/bin/mkdir /var/run
/bin/mkdir /var/log

/bin/chown root:root /tmp
/bin/chmod 0777 /tmp

/bin/cp -a /usr/local/var/* /var/
```

Решение несколько грубоватое, но ничего лучше в голову не пришло. Вроде работает и не жужжит, так что оставил как есть

Последнее изменение: Tue Sep 30 10:46:38 2008

Автор: Dark

источник http://www.ounix.ru/index.php?page=article&id=28