Buffalo Linkstation Backups via SSH

Andrew Richards, 18th January 2012 (http://www.acrconsulting.co.uk/contact.html)

Introduction

I have a Buffalo 'Linkstation Live with BitTorrent' (LS-CH1.0TL) device with firmware version 1.56 (I'd upgraded it to the latest firmware version). I particularly like the fact that Buffalo use XFS for the data area filesystem. These notes may work for some other varieties of LinkStation.

I'd like to use NFS to access this so I can make backups of my Linux systems whilst retaining file ownership details. Unfortunately neither NFS nor SSH is offered by Buffalo on this device. My efforts to set this up myself have got me to an alternative solution: Enabling ssh/sftp, with which I can mount the device. This with lots of help from articles and posts on nas-central.org and other sites.

I've subsequently managed to get Debian Squeeze running on the device so I can roll my own NFS server, but many of the steps in this guide were still useful to give me ssh access in the first place, and to enable me to make decent backups of the device before replacing its installed software with Debian. If you do subsequently wish to continue by installing Debian, you can find my guide to that here,

> http://buffalo.nas-central.org/wiki/Debian_Squeeze_on_LS-CHLv2

Alternative approaches to this problem

Alternatives to the method below are to downgrade the firmware (downgrade = telnet, ssh available) and use a customised Buffalo kernel & NFS module: http://buffalo.nas-central.org/wiki/Building_a_kernel_from_Buffalos_source & http://buffalo.nas-central.org/wiki/Ready-to-use_NFS_kernel ...or to avoid the firmware downgrade see here for how to hack the firmware to enable telnet, ssh: http://buffalo.nas-central.org/wiki/Open_Stock_Firmware_LS-XHL

Alternative methods for enabling ssh with earlier versions of the firmware include,

> http://forum.buffalo.nas-central.org/viewtopic.php?f=71&t=22588
>
> http://buffalo.nas-central.org/index.php/Open_Stock_Firmware

and this thread is also worth a look,

> http://forum.buffalo.nas-central.org/viewtopic.php?f=71&t=24655

Warning

Standard disclaimer: Proceed at your own risk etc; these steps are likely to void your warranty, and it's possible any mistakes in this document could lead to loss of your data.

Summary of Steps & Hurdles

A few things to overcome,

- Identify the device [family] – NAS-Central website very helpful here (maybe it's not necessary to identify the device after all for the steps below, but I find it does help when looking for info on how to solve other problems with the device).
- Access the hard disk directly - achieved by opening up the device and taking out the hard

disk so it can be plugged into a PC.

- Non-standard disk layout: The unusual GPT method of partitioning is used, so MBR style partitioning & mount tools can't be used.
- Backup via ssh: the sshfs tool enables remote systems to be mounted via ssh/sftp, but sshfs has a couple of unexpected quirks that require workarounds.

Although the device is Linux-based (see the text `Linux version 2.6.31.8` in the device's `/var/log/messages`, although note that the text `root@ubuntu` doesn't in fact indicate it's Ubuntu based, merely that the kernel was built on a machine named ubuntu), it's heavily customised.

Note that I use '`#`' below to indicate commands that I run as `root`, either on a workstation (`PC#`) or on the Linkstation itself (`LS#`).

Opening the case & taking out the hard disk

Peel off both labels on the underside of the device to reveal plastic catches, press these gently with a screwdriver to release; you should now be able to open the case; I think it came apart from the rear of the device first. Unscrew the 3 screws in the PCB and the fan retaining screw to remove the hard disk (the screw is in the top of the case near the back). You should now be able to unplug the PCB from the hard drive. Pictures to give you a better idea here,

http://www.yamasita.jp/linkstation/2008/11/081112__disassemble.html

Hardware notes

The device is ARM9-based, specifically 'ARM9/Kirkwood'. This is categorised as a LS-CHLv2; details here,

http://buffalo.nas-central.org/wiki/Category:LS-CHLv2

note that the CPU is marked 88F6281-A1, confirming this identification. This is important to end up in the right part of the nas-central.org forums for this device. It's got 64MByte RAM. It is architecturally identical to the Linkstation Pro LS-XHL,

http://buffalo.nas-central.org/wiki/Open_Stock_Firmware_LS-XHL

The ARM CPU has different generations for its machine code. Confusingly although the hardware is 'ARM9' as above, I think the instruction set is ARMv5, at least that what it looks like here, http://www.riscos.info/index.php/ARM_System-on-Chips#Marvell_Kirkwood_88F6281

The hard disk inside is a Seagate Barracuda 7200:12. This had firmware CC44 – find the firmware version marked on the plastic label on the front of the hard disk, which I've upgraded to CC49 (see Seagate website).

Backup stock setup

With http://forum.buffalo.nas-central.org/viewtopic.php?f=39&t=13551 as inspiration I attached the disk from the Linkstation into my desktop PC. I backed up the first 10 GByte of the hard disk, which should contain everything for booting and then some (I'm assuming the Linkstation disk is connected as `/dev/sdb`):

```
PC# dd if=/dev/sdb of=some.filename bs=1M count=10000
PC# bzip2 -9 some.filename
```

(just reverse this process to restore the disk to a 'fresh' state – might then need to trigger a rebuild for the device to regenerate an [empty] data partition). If you successfully mount the partitions

(below) you may prefer to use `rsync`-based backups of each partition instead, these are likely to be more compact and you'll be more in control when restoring them if necessary. Also remember to backup the boot data (part of the first 17408 bytes on the disk according to `parted` map below, although some of that includes the partition table),

```
PC# dd if=/dev/sdb of=filename bs=17408 count=1
```

(I'm not 100% sure it's 17408 bytes, or 17407 bytes – depends whether `parted` counts from 0 or 1). Perhaps more useful than large files like the above is to copy the filesystems in a form you can use later – this is particularly handy if you replace your Buffalo software with e.g. Debian, since you will have a backup of the original software so you can look at the Buffalo scripts etc., which may be helpful in getting stuff to work. Here's how: You should backup the boot and root partitions (`/dev/sdb1`, `/dev/sdb2`) containing the Buffalo setup; you may also wish to backup the data partition `/dev/sdb6`. `rsync` is a good tool to achieve this and keep almost all the [meta] data attached to the files too – but something like Clonezilla is likely to be better still if you end up needing a backup you can restore rather than just one you may like to use for reference. Here's my suggestion if you choose to use `rsync` for this (NB '`/mnt/`' not '`/mnt`' in the `rsync` commands),

```
cd directory-where-you-wish-to-store-the-backups
umount /mnt
mount /dev/sdb1 /mnt
rsync -aH /mnt/ linkstation.boot.partition
umount /mnt
mount /dev/sdb2 /mnt
rsync -aH /mnt/ linkstation.root.partition
umount /mnt
```

Disk partitioning with GPT

If you've had problems with mounting individual partitions it may be that you're using a non-GPT aware system.

GPT partitioning is described here http://en.wikipedia.org/wiki/GUID_Partition_Table . `parted` is one option to explore this, also see tools here http://www.rodsbooks.com/gdisk/download.html

On a GPT-aware system you can mount the partitions like this,

```
PC# mount /dev/sdb1 /mnt
```

To a non GPT-aware system it's non-trivial to mount these partitions. However using http://viaforensics.com/computer-forensics/howto-mount-hfs-image-partition-linux.html as inspiration here's how,

- Examine current partition layout with `parted` to get location of each partition – assuming Linkstation disk is connected as `/dev/sdb`,

```
PC# parted /dev/sdb unit B print
Model: ATA ST31000528AS (scsi)
Disk /dev/sdb: 1000204886016B
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number Start       End           Size          File system    Name  Flags
 1     17408B      1024000511B   1023983104B   ext3           primary
 2     1024000512B 6144000511B   5120000000B   ext3           primary
 3     6144000512B 6144001023B   512B                         primary
 4     6144001024B 6144001535B   512B                         primary
 5     6144001536B 7168000511B   1023998976B   linux-swap(v1) primary
 6     7168000512B 992000000511B 984832000000B xfs            primary
```

- The 1st partition will be the `/boot` partition; the 2nd will be the Linux root (`/`) partition; the 3rd and 4th look to be dummies; the 5th partition is swap, and the 6th partition is the data area.
- Since the 1st, 2nd and 6th partitions should contain filesystems you can mount them using the 'Start' value revealed by parted as an offset, so for example,

  ```
  PC# mount -o ro,loop,offset=17408 /dev/sdb /mnt
  ```

- Notice the `ro` above to mount read-only – but you can mount read-write (`rw`) if you wish. I'd recommend having a backup first though. That said, sometimes mounting ro fails with an error message suggesting you look at `dmesg | tail`, which helpfully (for me) said `EXT3-fs: INFO: recovery required on readonly filesystem.` / `EXT3-fs: write access unavailable, cannot proceed.` , so that mounting `rw` may be needed anyway to mount a partition. **Update:** It looks like Buffalo's default approach to mounting partitions (at least for boot & root partitions) is to leave them 'dirty' on shutdown. I've now converted my system to Debian so can no longer check Buffalo's mount options – this means that it may not be possible to mount these partitions read-only unless an fsck is carried out; mounting read-write means that the fsck can occur, if I remember my filesystem lore correctly.
- Gotcha: mkfs may be troublesome on a non-GPT system, not sure how to do that without having a device to point it at...

Enable ssh

All the plumbing for `sshd` is there, so there's not too much to do to turn it on – the trick is bypassing Buffalo's checks that can disable it again. I like keypair authentication so that's what I'll enable for `root` (avoids having to know/discover `root` password to subsequently login). I'll ignore the ramdisk image and concentrate on the root partition since this is mounted by the time the system starts `sshd`.

- Mount the Linkstation root partition read/write,
  ```
  PC# umount /mnt
  PC# mount /dev/sdb2 /mnt
  ```
- or if your system isn't GPT-aware (the above commands will give an error if so) (use `parted` to determine offset on your system as above,
  ```
  PC# umount /mnt
  PC# mount -o rw,loop,offset=1024000512 /dev/sdb /mnt
  ```
- Put your ssh public key into `/mnt/root/.ssh/authorized_keys` (append to `authorized_keys` just in case it already exists; I've assumed your key is in `.ssh/id_rsa.pub` in `root`'s home directory).

  ```
  PC# cd /mnt/root
  PC# mkdir .ssh
  PC# chmod go-rwx .ssh
  PC# cat ~/.ssh/id_rsa.pub >> .ssh/authorized_keys
  PC# cd ..
  ```

- Enable `sshd`: I *think* the startup script for the Linkstation is `etc/init.d/rcS`. This script starts the extensions in `etc/rc.d/extensions.d`, including `etc/init.d/sshd.sh`. This `sshd.sh` script checks `ENABLE_SFTP` in `etc/nas_feature` and exits if this is zero (which it is). Unfortunately the device regenerates `etc/nas_feature` [if it has changed?], so changing `ENABLE_SFTP` doesn't work. Instead comment out the `exit 0` near the top of `etc/init.d/sshd.sh`.
- The above appears to be insufficient. Inspiration comes from http://forum.buffalo.nas-central.org/viewtopic.php?f=71&t=22588 to comment out the `nas_configgen -c sftp`

line in `etc/init.d/sshd.sh`.

- Change `PermitRootLogin` to `yes` and `UsePAM` to `no` in `etc/sshd_config`. (and if you want keypair authentication only, change `PasswordAuthentication` to `no` in the same file).
- (Optional) To fix log entries on the Linkstation like this:

  ```
  sshd[2896]: lastlog_perform_login: Couldn't stat /var/log/lastlog: No such
  file or directory
  sshd[2896]: lastlog_openseek: /var/log/lastlog is not a file or directory!
  ```

  create an empty file,

  ```
  PC# touch var/log/lastlog
  ```

  or if you've already replaced the hard disk in the Linkstation, connect to it via ssh then,

  ```
  LS# touch /var/log/lastlog
  ```

- Note that password-based SSH login and SSH login for non-`root` users needs additional steps (not tried), probably including users having their own home directories containing an `.ssh` subdirectory (each user needs to own their own `.ssh` directory not readable to other system users); adding users via the Buffalo web GUI will create users with the home directory `/home` – not `/home/`*username*, although the device does include the `vipw` utility so changing the home directory is likely to be straightforward. There are several threads on forum.buffalo.nas-central.org about this including,

  http://forum.buffalo.nas-central.org/viewtopic.php?f=39&t=21013

Mounting the Linkstation over ssh/sftp

Having got ssh working I realised that this was sufficient for my immediate requirement of doing backups and keeping ownership details intact (UIDs), by mounting using sshfs. Based on,

http://blog.damontimm.com/how-to-mount-a-sftp-folder-ssh-ftp-on-ubuntu-linux-using-sshfs-fuse/

Install sshfs on your client machine (desktop etc) – e.g. on Ubuntu,

```
PC# apt-get install sshfs
```

or Fedora,

```
PC# yum install sshfs
```

To mount the Linkstation as `root` (assuming you've named it `nas` in `/etc/hosts`),

```
PC# sshfs -o hard_remove root@nas:/mnt/disk1/share /mnt
```

or you may wish the Linkstation to be available to non-`root` users. In that case,

```
PC# sshfs -o allow-other,hard_remove root@nas:/mnt/disk1/share /mnt
```

A problem with this is that if non-`root` users copy their own files to the Linkstation they are owned by `root`. The solution is probably to separately `mount` for each user, yuck.

Unmount with,

```
PC# fusermount -u /mnt
```

Snapshot backups

Command-line access (via `ssh`) enables snapshot backups: Snapshots share links (inodes) to unchanged files; only if a file is changed does it need another copy – this saves masses of space.

This idea comes from Linux Server Hacks (Hack 42) by Rob Flickenger (publ. O'Reilly). Due to a

weird bug with sshfs `rsync` can't update a file without using `--inplace`; but this would overwrite the snapshots too. Therefore the use of the `-b` and `--backup-dir` options and `rm` below to work around this. The bug is described here,

https://bugs.launchpad.net/ubuntu/+source/rsync/+bug/501532

Example: Backup `/home` with `rsync`,

```
PC# rsync -aHuz  --progress -b --backup-dir=/mnt/homebackup.workarea
                                                    /home /mnt/homebackup.latest
PC# ssh nas rm -rf /mnt/disk1/share/homebackup.workarea
```

Create a snapshot of this with today's date e.g.,

```
PC# ssh nas cp -al /mnt/disk1/share/homebackup.latest
                          /mnt/disk1/share/homebackup.latest.2012-01-19
```

In the commands above you'll notice the same path is written differently, according to whether it's from the PC's perspective (`/mnt/xyz` – assuming you mounted it with the `sshfs` command above), or the Linkstation's (`/mnt/disk1/share/xyz`).

Repeat this procedure for subsequent backups: The `.latest` structure gets updated with any new files; this adapts nicely to a script.

Loose Ends

Although it's not possible connect via `ssh` as root with the `root` password, other services are probably still usable with the `root` password. Therefore you may wish to disable the `root` password. Do so by using `vipw` on the Linkstation and inserting a '*' at the start of the [encrypted] `root` password field. Just remove the '*' if you ever wish to re-enable the `root` password.

Feedback

Feel free to contact me directly about this document, or to discuss this document here,

http://forum.buffalo.nas-central.org/viewtopic.php?f=71&t=24789

I hope this guide helps some Linkstation owners out there. I expect to update this document with any corrections, and also any success reports on other models/firmware versions of Linkstation (let me know your Linkstation model & firmware version if you'd like to submit a success report).