rsync --link-dest

Local, rotated, quick and useful backups!

arse-ink --link-dest

Local, rotated, quick and useful BACKups!

(Baby got Back ups?)

Scope

- No complete scripts will be presented
- Just enough so that a competent scripter will be able to build what they need
- Unixes used: OpenBSD, FreeBSD, Solaris, AIX, Linux. Should work on anything rsync runs on.

Simple RSYNC backups

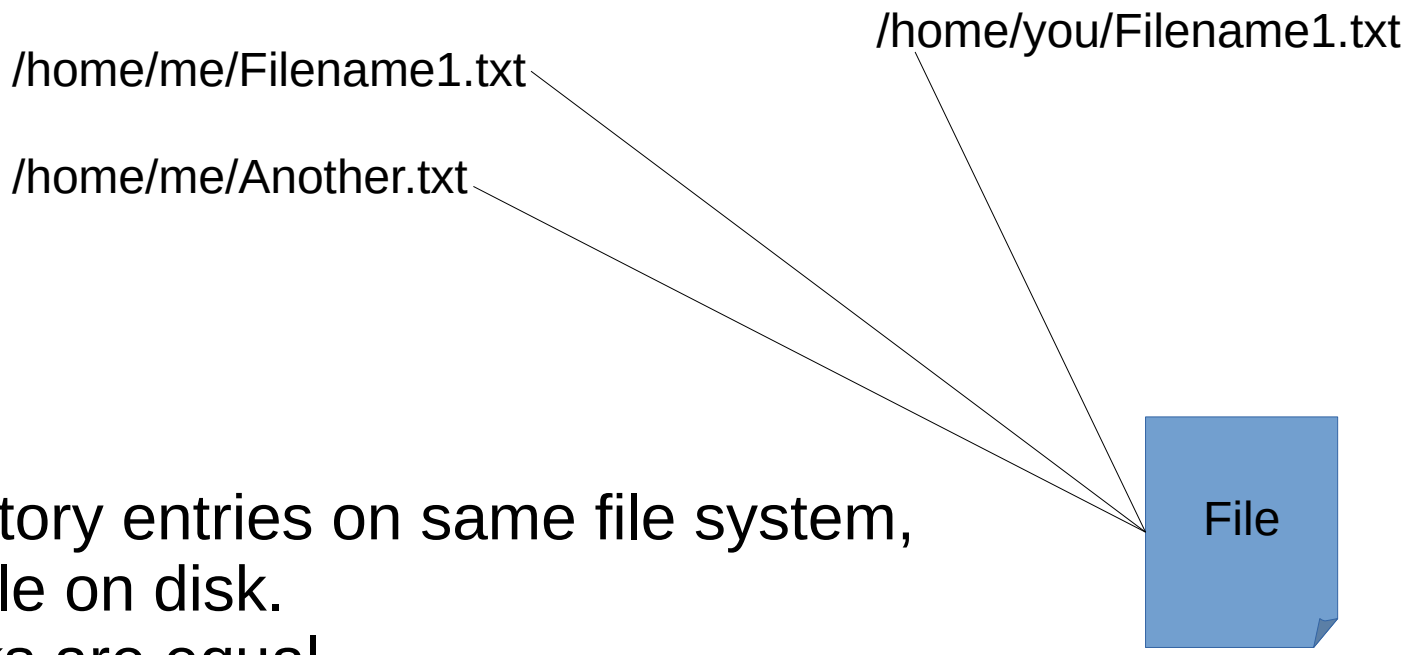
- From backup sources to central disk storage
- Copies over what changed.
- Copies over just the PARTS that change
- (after first backup) VERY fast and efficient.

BUT…

- Not rotated. No archive. BARELY counts as a backup.

NOT OUR TOPIC TODAY. :)

Crash review: hard linked files



- Three directory entries on same file system, only ONE file on disk.
- All hard links are equal.
- File exists until LAST link is removed.
- Again...all links are equal.

Better rsync backup – with hard links

- Create new directory for new backup.
- Hard link everything in old directory to new directory, duplicating directory tree structure.
- Rsync from target system to backup system’s new directory
- Unchanged files stay a link, changed files get overwritten, but previous copies remain.

Even better yet --link-dest !!

- Three way rsync – Source, PREVIOUS copy, NEW copy.
- New files: copied over.
- Unchanged files: hard link from --link-dest directory (!!)
- Changed files – Copied over (but with rsync bandwidth usage)

rsync w/links

Backup 1 (full!)

File 1

File 2

File 3

Backup 2

File 1

File 2’ (changed)

(deleted)

File 4 (added)

Backup 3

File 1

File 2’ (no chg)

File 4 (no chg)

File 5 (added)

Backup 4

File 1

File 2” (changed)

File 4 (no chg)

File 5 (no chg)

Benefits

- Rotated! History!
- Minimal disk use, minimal network traffic
- EVERY backup is “full”, but as fast as an incremental!
- WONDERFULLY USEFUL as it sits on the backup server.
- Communications over SSH, automatic key logon.
- Backup client? rsync! ANY version
- Restore client? rsync, scp, vi, whatever.
- Backup systems that your “primary” backup solution doesn’t recoginize
- MORE THAN JUST A BACKUP.

Down-sides

(you knew there had to be some)

- No geographic diversity
- File ownership, permissions CAN be munged.
- Root access needed to systems being backed up.
- Not a “bare metal” restore.
- Best for restoring data and config files
- End up with some complicated file systems.
- MS Windows.
- A failed backup can really balloon your disk needs.
- du … not fast. Not at all fast.

Not just backups. This is Unix!

Assuming `/bu/<machname>/<date>/`

- Which systems is nholland on?
  ```
  # grep nholland /bu/*/2016-08-23/etc/passwd
  ```
- When did nholland's account get created?
  ```
  # grep nholland /bu/fs3/*/etc/passwd
  ```
- How's the database dump growing?
  ```
  # ls -l /bu/fs3/*/db/dumpfile.txt
  ```
- Any time you have a question about all machines...
- File change detection (IDS)?
- Change ownership/permissions (non-root analysis).
- Systems doing self-backups

Doing It.

- “Projects” exist. So what.
- Rsync *{-options}* --link-dest *{prevbu} {source} {newbu}*
- Date your backups (yyyy-mm-dd)
  - Maybe most recent as “curr”?
  - Sortable naming
  - M-yyyy-mm-dd for monthly?
- Pre-create backup directories (2000-00-00, 2000-00-01, 2000-00-02, etc).

Doing It (part 2)

- Create new backup dir.
- Make backup between source, previous, and new
- Delete oldest afterwards – keeps a constant number of backups.
- Want to “pull” a backup out of rotation? Rename it! Create new replacement. (note: “cost” will increase with time)
- Save output of rsync to a file – backup log.
- Create backup reports from the rsync log files.

Doing it...part 3

- Chunk your data. Even though you don’t want to.
- Symlink from /bu to actual storage spaces
- Watch your free space carefully. Don’t run out.
- Know what your ‘du’ command does with hard links.
- Use otherwise “wasted” space – local disk on VM hosts.
- Test your restores.
- Beware of reversing trust.

Doing it … part 4

- One script to run the job – “bu”
- Second script to grab the output from automatic runs – “bucron”
  - Run just the specified job? – or –
  - Run all jobs in specified directory?
  - pgrep | wc -l your rsyncs, hold off until there are fewer than X running (20 to 30?)
- head, tail, basename, dirname, df, du, grep are your friends.

Rsync options
(beyond --link-dest)

- -a (you want this. Covers a lot of things)
- -H (Preserver hard links. Probably)
- --stats (Summary statistics. For report)
- --progress (eh. Maybe not.)
- --force
- -z (Compress – varies depending on use)

rsync option --exclude-file

- Some things, you don’t want backed up. Ever.
- Syntax is somewhere between tricky, black magic and just broken.
- Start with a default, then add to it as needed.

```
+ /
- /mnt
- /proc
- /tmp
- /ramtmp
- /dev
- /sys
```

Hardware

- Very modest, unless you have a lot of local, high-speed systems.
- Lots of cheap but redundant disk storage.
- Slow CPU on backup system may reduce load on machines being backed up.
- Compression may or may not improve overall performance.
- Memory – usually determined by file system, not rsync tasks
- 1 core, 1G RAM is often more than sufficient.

Lessons

- DO NOT run AV on the system.
- --link-dest need only be on the BU system; not the host being backed up.
- If backing up the backups to tape, beware massive numbers of hard links. And be ready for issues on restore.
- Disk redundancy on your backup system

FreeBSD/ZFS variant

- Each system gets its own ZFS partition.
- df shows all!
- No --link-dest, use ZFS snapshots
- ZFS SEND snapshots to another machine
  - Destination MUST be “Read Only”
  - atime is not your friend.
- Good luck. You may need it. Found to be about as stable as a pig on stilts (granted...vmware, insufficient RAM, insufficient “tuning”.)

Questions? Comments?
Sarcastic Remarks?