Native backup utilities

- Native utilities are those bundled with your operating system
- They're not particularly fancy but have stood the test of time
- available in some form or another on most Unix (&other OS)
  - Portable backup apps, means portable data …
- No worries about a commercial vendor going out of business and abandoning your backup format
- While the old simple backup commands are robust and portable
  - 'managing' the backups may not be as easy or user-friendly, as modern GUI systems in terms of indexing, tracking and locating contents.
  - But most shortcomings can be overcome
    - Either by learning advanced features
    - Or overcoming with scripting

Windows7 has it's own GUI for System Protection

Which handles most protection issues

- Access & security
- Backups

And are best done as a whole, together, but separate from the current *nix tools, many of which have been ported or copied to Windows.

Main commands for backup

- *nix
  - tar – oldest, standard, portable .. But not as powerful or flexible
  - Cpio – better than tar, unless you want minimalist portability
  - dump/restore – popular for commercial Unix, not for Linux
  - rsync – interoperable among all main OS'es here.
  - dd – when all else fails.. Does what it says on the tin.. Disk duplication … bit by bit…DVD's the lot … DR(A)M – huh?
- Win
  - Basically a GUI to configure backups and security and may cover later
- Mac .. .*nix + the following
  - Ditto
  - Time Machine
- GNU …http://www.gnu.org/
- Versions of tar, cpio & rsync available for all of the above OS'es and generally more powerful and flexible

Comparison of main ones…

- *Dump/restore*
  - It is hard to beat *dump* and *restore* for backing up and restoring an entire filesystem
  - But not entirely compatible with some early versions of Linux or OS X
  - *dump* can perform incremental backups
  - *Dump/restore* can leave *atime* and *ctime* (access/create) unchanged after a backup or restore (some modify to these to reflect backup time!)
  - *Dump/restore* can also perform an interactive restore
- *cpio*
  - After *dump*, *cpio* is probably the next best choice
  - It is not as flexible as *dump*
    - No incremental backups without using the likes of find to select the files
    - Modifies either the atime or ctime

Mac filesystems are different…to *nix

- Apples proprietary legacy HFS+ is not compatible with UFS
  - HFS – Hierarchical File system retains legacy Mac filesystem issues such as
    - Forks – multiple sets of data associated with a single file: e.g. thumbnails for images, aliases (soft or symbolic links) although falling out of favour Apple still uses them for aliases.
    - Specialised file attributes – e.g. type, creator, creation date
  - UFS - Unix File system
  - Mac can handle UFS (a format option for Apple disks) but UFS is not commonly used due to backwards compatibility with apps. & data files
  - Indeed standard Unix utilities in earlier versions of OS X (<10.4) could not handle HFS+ properly, so that Mac-specific variants of standard utilities appeared..hfstar, xtar,… together with graphical front ends.. RsyncX, Carbon Copy Cloner, and cross platform apps such as Amanda and BackupPC also used these tools underneath the GUI hood to support HFS+ backups.

Comparison of main ones…

- *tar - tape archive*
  - http://www.gnu.org/software/tar/manual/tar.html
  - Although it can't do all the things *dump* and *cpio* can do, *tar* is well known and almost anyone can read a tar file
  - Even *WinZip* understands *tar* files
- *dd – disk duplicate*
  - *dd* is NOT a backup utility for most people
  - A direct bit-for-bit copy from one location to another, ignores any knowledge of the data structure is copying
    - (it can do type & format conversions, if you can figure it out!)
  - *dd* may be an appropriate solution in some situations e.g. if you need to back up a raw partition for a database

Old tar – tape archive

- The old favourite for many
- Not just for archive & basis of others e.g. Amanda open source
  - But for distributing & sending files
  - Standard Even WinZip can open & read tar files.
  - More portable among Unix systems.
- Two styles..
  - old short option style which is Unix compatible
  - and new long option style … GNU and is more widely available (incl.Win)

$ **sudo tar -ztvf /dev/st0**

$ **sudo tar --gzip --list --verbose --file /dev/st0**

Both commands tell tar to make a (**v**, **verbose**) table of contents (**t**, **list**) ,
use gzip (**z, gzip**) for (de)compression) to/from the **f**, **file** argument**: /dev/st0**,
which could also be the name of a file, or directory, even across the network.

Unlike the original UNIX tar utility,
the GNU version strips the leading / from absolute pathnames.

Tar originally had trouble handling exceptionally long pathnames.

Tar backup – assuming setup in previous slide

Archiving entire filesystem from root
$ **cd /**
$ **sudo tar -cf /dev/st0**

- The tar command then
  - creates an archive(**c**) (x for extraction)
    - actually appends it to existing one if present, creates otherwise
  - on the argument to file **f /dev/st0** (**f**) which could be any file, on the net.
- To (de)compress the archive, use **j** to call bzip2:
  $ **sudo tar -cjf /dev/st0 .**
- To create a backup of a directory mydir
  $ tar cvf /dev/st0 mydir
- To make a verbose archive to dev of everything in the current directory beginning with an a
  tar cvf dev a*

- To **extract** just replace the c (create) with x (extract); otherwise identical.

Table 17-1 tar options

- **--append (-r)** Appends files to an archive
- **--catenate (-A)** Adds one or more archives to the end of an existing archive
- **--create (-c)** Creates a new archive…(**-x** extracts it…)
- **--delete** Deletes files in an archive (not on tapes)
- **--dereference (-h)** Follows symbolic links
- **--diff (-d)** Compares files in an archive with disk files
- **--extract (-x)** Extracts files from an archive …(**-c** creates it…)
- **--help** Displays a help list of tar options
- **--list (-t)** Lists the files in an archive
- **--update (-u)** Like the **-r** option, but the file is not appended if a newer version is already in the archive
- **--verbose (-v)** lists filenames and sizes as they are being archived

The **-c**, **-t**, and **-x** options are used most frequently.
The **-j** option, compresses or decompresses the file by filtering it through bzip2

The -f flag must be followed immediately by the filename (after separator)

Selective restoring with tar …

- Usually, the entire archive is restored to the original state; all files etc.
- But, any selected file can be restored alone, by giving it's original pathname as used in creating the archive.
- A toc (table of contents) can be recreated from the archive to help locate the original pathname
  tar tf /dev/st0 > tocfile
- Which you can browse
- Or grep to find the selected file's pathname.
  grep myfile tocfile
  - Or be really concise… (will only work on archival tape with a rewind option)

  tar xvf /dev/st0 `tar tf /dev/st0 | grep 'myfile'`

  - And give it time to locate the file, before proceeding with restoration

  tar xvf /dev/st0 `tar tf /dev/st0 | grep 'myfile' ; sleep 60`

Example backup setup

- Backup disk/filesystem/directory prior to any change
- Assuming a device
  - E.g fast backup tape **/dev/st0**
  - Or backup hard drive **/dev/hdb**
- Switch to root to backup entire directory if desired
  cd /
- To back up the entire system from root, root privileges are needed.
  - Even for an experienced admin, it's too easy to mess-up
  - User sudo…which requires
    - password to execute – default is user password, but also
    - User must be in sudoers file, permitting use of sudo.

Tarred and ~~feathered~~…

1. tar – moving directories across limited bandwidth?
   1. Move to directory one level above the one to move
   cd old-dir ; cd ..
   2. Use tar within a subshell {designated by the brackets (…) to untar & create the new-dir within a new parent dir: cd.. Note the **-p** option creates it with same permissions as old.
   Tar cf -old-dir | (cd new-dir ; cd .. ; tar xvpf -)

(GNU tar has a –T flag to specify a file with a list of files for backup,
(if – is specified after T, tar uses standard input rather than a filename, which may be handier for pipes … as in print below is piped to standard input)
e.g. **find . -print | tar cvf /dev/rmt/0cbn -T -**
will backup files from current directory (.) to a remote device.
Find can be refined to be more selective, using regexp & appropriate flags.

Ownership, Permissions, attributes

Tar does not restore all file attributes

- By default permissions are determined by the current umask,
  instead of original file permissions before archiving – (overridden by p flag)
- Likewise the *setuid* or *sticky bits* are not restored,
  except for files owned by the user. (overridden by o flag)
  Only root can set these values on other's files.

However some flags can override these defaults:-

- m changes the modification times to that at restore
- **o** (default option) makes you the owner of files you restore from an archive (rather than reassigning them to the original owner – then only root could change the ownership to you)
  - But files restored by root, take the original owners ownership (restoration by root assumes the user couldn't do it himself, consequently assigns the ownership to original owner by default
- **p** restores permissions of original files…

Crond & anacrond … scheduler daemons…

- Daemon –
  - Monitoring process lurking in the background, which activates on some conditions or signals, rather than being called explicitly, although some processes depend on their behaviour and assume their existence.
  - hence it's name, but later 'sanitised / rationalised' to the unimaginative and boring Disk And Execution MONitor
- Crond – the cron daemon – for ALWAYS ON systems
  - executes scheduled tasks on systems that are ALWAYS ON, according to times specified in a timetable in the crontab file
- Anacron – for systems NOT ALWAYS ON

executes tasks scheduled tasks on systems when called, suited to laptops & mobile devices that are NOT ALWAYS ON, and its calling script anacron init, refuses to work when powered only on batteries

Gotcha's : Tarred and ~~feathered~~…**don't double tartar**

1. Tar * omits hidden files such as .profile, .bashrc etc.
   1. Using .* will not fix it, as .* will also match the parent directory ..
   2. Safer & simpler to work from the parent directory
2. The -f flag must be followed immediately by the filename (after separator)

Tar a tarred file and you're really stuck

3. All too easy to retar an already tarred archive…no good way back.

Great if you have tape…a once and for all, get it all!

4. When making 'one last archive' of a directory before final deletion, you may want to have a complete self-contained archive, with no external file references. Therefore you may want to follow all symbolic links, (such as aliases etc.) and make copies of those too, so your archive is complete and self contained. Otherwise, those linked files external to the archive, may be deleted in subsequent changes to the filesystem, and your archive is useless.

crontab … setups

- System crontab files
  - Contain commands which run with root privileges
  - reside at either directory : **/etc/cron.d** or **/etc/crontab**
- User crontab files
  - Contain commands which run as user processes,
  - So there is no user field in the crontab file
  - are set up by the crontab utility at : **/var/spool/cron/crontabs**
  - by default Ubuntu allows any user to have **cron** run commands in their personal crontab files.
  - can be changed by adding users to **cron.allow** or **cron.deny** files
- crontab has 3 meanings and can refer to any of
  - the utility that creates a crontab file
  - A text file in a specific crontab format
  - The name of the actual crontab file e.g. /etc/crontab

Use GNU tar if you can…http://www.gnu.org

Can read all other tar formats (reverse may not apply!)… plus following options:

- -a : resets access time
- -d : reports differences between archive & filesystem, using diff
- -f : target file - supports remote device names.
- -F : runs a script when tar reaches the end of a volume, which can be used to automatically swap volumes with a media changer.
- -p : forces GNU tar to include leading slashes on absolute pathnames;
  default is to suppress leading slash on absolute pathnames
  in creating or reading a tar archive.
- -T can specify a file with a list of files for backup, (- after T uses stdio)
  e.g. find . –print | tar cvf /dev/rmt/0cbn –T -
  will backup files from current directory (.) to a remote device
- -Z and –z : automatically pass the archive through compressor gzip,
- Offers word arguments instead of more cryptic characters
  Instead of tar cvf, you can specify tar –create –verbose –file.

System Crontab format : 7 fields

- Minute hour day-of-month month day-of-week user command
- First five are numeric:
  - Minute : minutes after the hour
  - Hour : 24hr clock
  - Day-of-month : 1-31
  - Month-of-year : 1-12
  - Day-of-week : 0-7, with both extreme values for Sunday in western week.
- * : asterisk – wildcard…for any value

```
20 1   *  *  *   root   /usr/local/bin/checkit
25 9   17 *  *   root   /usr/local/bin/monthly.check
40 23  *  *  7   root   /usr/local/bin/sunday.check
```

Which mean, line by line :

Daily @ 01:20 ; monthly on 17th @ 9:25 ; weekly @ 23:40 on 7th day of week.

Crontab : field allowed values

Crontab : field	allowed values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names, see below)
day of week	0-7 (0 or 7 is Sun, or use names*)

Names are also used for the ``month'' and ``day of week'' fields *

- Use the first three letters of the particular day or month
  - (case does not matter).
- Ranges or lists of names are not allowed.
- But ranges and lists are allowed for numeric fields – next slide.

* locale specific names may apply, western presented here.

User crontab file setup

- User crontab files
- Unlike system crontab files, they do not have a user field,

  and just run as user processes : the user who created the file.
- Are accessed via a crontab command with the following options
  - -l : to list
  - -r : to remove
  - -e : to edit; default editor is nano, but can be reset to editor of your choice by changing the VISUAL or EDITOR environment variable using export etc.

  – (which looks after book-keeping & updating spool files etc)

Crontab : numeric field : range, list and step.

Ranges, step and lists are allowed for numeric crontab fields:-

- Ranges :
  - all integers, including the end ones,
  - separated by a hyphen which specify the range

  e.g. 8-11 for an ``hours'' entry specifies execution at hours 8, 9, 10 and 11.
- Step values can be used in conjunction with ranges.
  - `/<number>'' after range, specifies skips of the number's value.

    E.g., ``0-23/2'' in the hours field, specifies execution every other hour (the alternative in the V7 standard is ``0,2,4,6,8,10,12,14,16,18,20,22'').
- Steps are also permitted after an asterisk,

  - so if you want to say ``every two hours'', just use ``*/2''.
- A list is a set of numbers (or ranges) separated by commas.
  - Examples: ``1,2,5,9'', ``0-4,8-12''.

Default system crontab : /etc/crontab file

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# Format of table entries below :
# minute   hour day_of_month   month      day_of_week    user         command
17 *  * * * root   cd / && run-parts --report /etc/cron.hourly
25 6  * * * root  test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6  * * 7 root  test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6  1 * * root  test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Such a crontab would result in the following schedules:

hourly	: 17 mins past the hour
Daily	: 6:25 am
Weekly	: Sundays at 6:47
Monthly	: 1st of each month @ 6:25

Cron runs only the hourly script if anacron is installed and running; anacron being the scheduler which is activated when user calls.

Which would result in the following behaviour: except for hourly, check if anacron is executable, and if not, change directory to root, and if success then run-parts; which runs all executable files in the current directory,
--report redirects all command output to either std output or std error, whichever it sends its first output.

Alternative to numerics in the first five fields

@reboot	Run once, at startup.	
@yearly	Run once a year,	"0 0 1 1 *".
@annually	(same as @yearly)	
@monthly	Run once a month,	"0 0 1 * *".
@weekly	Run once a week,	"0 0 * * 0".
@daily	Run once a day,	"0 0 * * *".
@midnight	(same as @daily)	
@hourly	Run once an hour,	"0 * * * *".

Standard pre-installed crontab entries

- Cron also checks /etc/cron.d for pre-installed commands
  - mostly in /etc/cron.d
  - But also in /etc/cron.daily and /etc/cron.weekly
  - These provide alternative locations for software packages to install periodic jobs without having to edit the crontab file
- To override default entries
  - Comment them out, (with #) in case you need them later!
- Caution :- crongestion…
  - Subtle, intermittent but recurrent problems can occur on networks if cron runs on lots of networked machines simultaneously,
  - Difficult to diagnose: intermittent, evanescent
  - Fix with random delay in the script or modify configuration file

cpio - use GNU version for compatibility across systems

- GNU tar has improved over original Unix tar, incorporating many of cpio's best extensions
- Otherwise cpio is similar to tar but can read and write archive files in various formats, including the one used by tar.
- Normally cpio reads the names of the files to add to the archive from standard input & writes archive file as standard output.
- When extracting files from an archive, it reads the archive as standard input.
- As with tar, some options can be given in both a short, single-letter form and a more descriptive word form.
- However, unlike with tar, the syntax of the two forms in cpio differs when the option must be followed by additional information.
  - short form: include a SPACE between option and additional information;
  - word form: separate the two with an equal sign and no SPACEs

Times are a changing

Unix, Linux & MAC OS X keep 3 different times for files

- Atime – last access to file time,
  - When last read as in: file read, script run, etc..
- Ctime – last change to attributes time,
  - When attributes, permissions, ownership changed
- Mtime – last modification to contents time
  - When last edit to contents saved

Touch allows changing these times to current or specified time, can be handy to automatically include otherwise unchanged files in an incremental backup based on modification time,
by using find – mtime, which will show files modified since

cpio – assuming backup setup

- To create an output file **(o)** and set the I/O block size to 5120 bytes **(B)** (the default is 512 bytes):

  $ **cd /**

  $ **sudo find . –depth | cpio –oB > /dev/st0**

- To restore the files to the **/home** directory from the preceding backup. The options **–ivmd** extract files **(i)** from an archive verbose mode **(v)**, keep the modification times **(m)** and create directories as needed **(d)** .

  $ **cd /**

  $ **sudo cpio –ivmd /home/* < /dev/st0**

If cpio is so powerful, vs why is tar so popular?

- tar is
  - Simpler and therefore more standard than cpio
    - Universal across all *nixen
    - All tars suport ***tar cf device*** … or tar xf device
    - But see first cpio point
    - GNU tar combines power of cpio with tar's ease of use (GNU tar accepts backup filelist for incremental backups)
- cpio
  - cpio support of basics is variable
    - even for –I (input/restore) and –o (output/backup)
    - Has 40 versions
    - Some with different meanings for the same flags!

cpio

- Works at the file level, unlike ***dump*** (filesystem level)
  - More flexible
- Can take a list of files for backup from standard input
  - for incremental backups if used with find –mtime & touch
  - Touch changes effective atime of file, so can force a file to be included in backup using find –mtime
- Changes access times on backup
  - Has option to reset atime, but changes ctime
- Unlike ***dump***, ***cpio*** cannot
  - Do incremental backups without find (& touch)
  - Leave both atime and ctime unchanged after backup
  - Perform interactive restore (for interactive file selection)

Example dump

dump 3ubdsf 10 1000000 1000000 /home/krf/dump.3 /usr

- Performs a level 3 dump of the filesystem mounted as */usr*
- Updates the *dumpdates* file
- Uses a blocking factor of 10 so blocks are 10240 bytes long
- Uses a density of 1,000,000 bits per inch and a media capacity of 1,000,000 feet

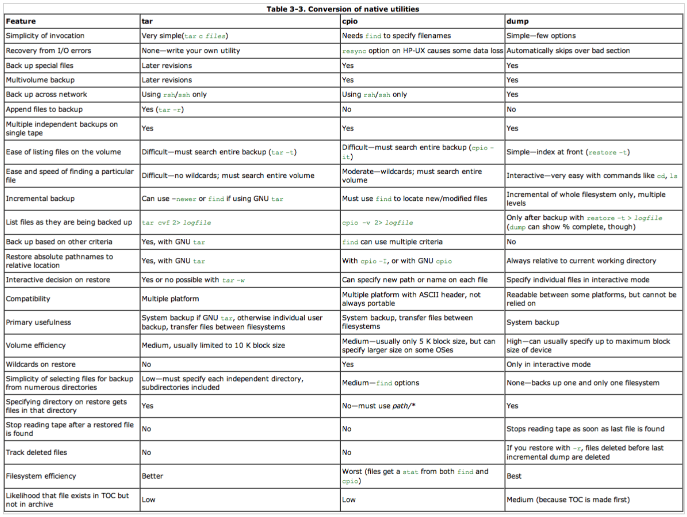

Table 3-3. Conversion of native utilities

Feature	tar	cpio	dump
Simplicity of invocation	Very simple(`tar c` *`files`*)	Needs `find` to specify filenames	Simple—few options
Recovery from I/O errors	None—write your own utility	`resync` option on HP-UX causes some data loss	Automatically skips over bad section
Back up special files	Later revisions	Yes	Yes
Multivolume backup	Later revisions	Yes	Yes
Back up across network	Using `rsh`/`ssh` only	Using `rsh`/`ssh` only	Yes
Append files to backup	Yes (`tar -r`)	No	No
Multiple independent backups on single tape	Yes	Yes	Yes
Ease of listing files on the volume	Difficult—must search entire backup (`tar -t`)	Difficult—must search entire backup (`cpio -it`)	Simple—index at front (`restore -t`)
Ease and speed of finding a particular file	Difficult—no wildcards; must search entire volume	Moderate—wildcards; must search entire volume	Interactive—very easy with commands like `cd`, `ls`
Incremental backup	Can use `-newer` or `find` if using GNU `tar`	Must use `find` to locate new/modified files	Incremental of whole filesystem only, multiple levels
List files as they are being backed up	`tar cvf 2>` *`logfile`*	`cpio -v 2>` *`logfile`*	Only after backup with `restore -t >` *`logfile`* (`dump` can show % complete, though)
Back up based on other criteria	Yes, with GNU `tar`	`find` can use multiple criteria	No
Restore absolute pathnames to relative location	Yes, with GNU `tar`	With `cpio -I`, or with GNU `cpio`	Always relative to current working directory
Interactive decision on restore	Yes or no possible with `tar -w`	Can specify new path or name on each file	Specify individual files in interactive mode
Compatibility	Multiple platform	Multiple platform with ASCII header, not always portable	Readable between some platforms, but cannot be relied on
Primary usefulness	System backup if GNU `tar`, otherwise individual user backup, transfer files between filesystems	System backup, transfer files between filesystems	System backup
Volume efficiency	Medium, usually limited to 10 K block size	Medium—usually only 5 K block size, but can specify larger size on some OSes	High—can usually specify up to maximum block size of device
Wildcards on restore	No	Yes	Only in interactive mode
Simplicity of selecting files for backup from numerous directories	Low—must specify each independent directory, subdirectories included	Medium—`find` options	None—backs up one and only one filesystem
Specifying directory on restore gets files in that directory	Yes	No—must use *path/**	Yes
Stop reading tape after a restored file is found	No	No	Stops reading tape as soon as last file is found
Track deleted files	No	No	If you restore with `-r`, files deleted before last incremental dump are deleted
Filesystem efficiency	Better	Worst (files get a `stat` from both `find` and `cpio`)	Best
Likelihood that file exists in TOC but not in archive	Low	Low	Medium (because TOC is made first)

And less need because of … ZFS ..designed to

- For large systems (by Sun for Solaris – their Unix for Sci.)
  - for large filesystem :- zettabyte = 1000^7, zebibyte = $(2^{10})^7$
    - Current filesystem limit is 256 zebibytes = 2^{78} bytes
  - Not to lose data at filesystem level, not just block level on disk
- Never to lose data
  - uses hierarchical checksums throughout filesystem tree blocks to ensure data validity through
    - Checksums at parent blocks, hierarchically, not with data blocks
      - Avoids checksum corruption if data block corrupted
      - Faster hierarchical (O(log(n)) with tree) error detection
    - copy on writes of all updates
      - Makes most disk writes sequential, instead of random
      - a new write, refreshes media
    - Regular monitoring of hierarchical tree blocks

dump

- Basic dump syntax
  - *dump Lunbdsf blk-factor density size device filesystem*
  - L is the level number, 0 – 9
  - u – update */etc/dumpdates*
  - n – notify members of operator group when *dump* has completed
  - b – blocking factor
  - d – media density
  - s – media size
  - f – which device should be used

State of ZFS

- Even OpenSource Licensing issues
  - (mostly I suspect due to fear of Oracle CEO who took over Sun, and then sued Google for Java JVM infringement)
  - Apple began development to incorporate, dropped in favour of journaled and to avoid licensing problems, but now OpenZFS code seems free from licensing restrictions.
- OpenZFS
  - Eventually released by Ubuntu in version 16.04LTS after being issued by other distributions for a few years.
- May still be issues, advice is to avoid unless needed.
  - Some recommend only for user data, not for root filesystem
  - Probably wiser to wait until any teething issues resolved, but should be by the time you graduate.

But replaced by… rsync

- Rsync – remote synchronise
  - Synchronises filesystems to remote (& local also) disk
  - Only updates changes, to save bandwidth & storage
  - often preferred option for command based system
  - Recommended to use ssh for security
  - Must be installed on both systems ; both executable & in PATH
  - Assumes secure encrypted link, i.e. use SSH
  - Default is to copy files only, need to use ‘**-a**’ flag to copy directories
- General format: rsync source user@dest_host:dest_dir
  - User@ is omitted if same userid on both machines
  - Source is
    - Either a file list separated by spaces
    - Or ***- a dirname*** if a directory dirname is to be copied
      - Followed by / if only the directory contents are copied and the directory itself is not recreated within dest_dir

rsync or swim!?

- There are lots more flags & features…
  - z – compression
  - - - bwlimit
  - - - exclude from backup
- For more information
  - https://rsync.samba.org/
  - https://www.samba.org/ftp/rsync/rsync.html