

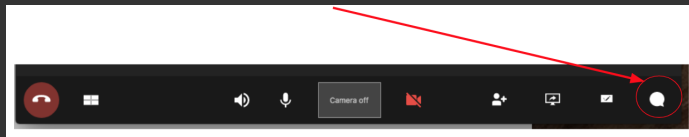
Managing many files with Disk ARchiver (DAR)

ALEX RAZOUMOV
alex.razoumov@westgrid.ca



To ask questions

- Websteam: email **info@westgrid.ca**
- Vidyo: use the GROUP CHAT to ask questions



- Please mute your microphone unless you have a question
- Feel free to ask questions via audio at any time

Parallel file system limitations

- Lustre object storage servers (OSS) can get overloaded with lots of small requests
 - ▶ `/home`, `/scratch`, `/project`
 - ▶ Lustre is very different from your laptop's HDD/SSD
 - ▶ a bad workflow will affect all other users on the system
 - Avoid too many files, lots of small reads/writes
 - ▶ use `quota` command to see your current usage
 - ▶ maximum 5×10^5 files in `/home`, 10^6 in `/scratch`, 5×10^6 in `/project`
- (1) organize your code's output
 - (2) use **TAR** or another tool to pack files into archives and then delete the originals

TAR limitations

- TAR is the most widely used archiving format on UNIX-like systems
 - ▶ first released in 1979
- Each TAR archive is a sequence of files
 - ▶ each file inside contains: a header + some padding to reach a multiple of 512 bytes + file content
 - ▶ EOF padding (some zeroed blocks) at the end of the TAR file
- Designed for sequential write/read on tape drives \Rightarrow there is **no index for random access** to TAR contents \Rightarrow extracting could be very inefficient
- Third-party tools can add indexing to TAR :
 - ▶ <https://github.com/devsnd/tarindexer> is a Python tool for indexing TAR files for fast access (writes a separate file)
 - ▶ RAT (<https://github.com/mcuadros/go-rat>) is an extension to embed the index at the end of the TAR file itself; any RAT-produced TAR file is compatible with standard TAR

Disk ARchiver (DAR)

<http://dar.linux.free.fr>

- Written from the ground up as a modern replacement to TAR
- Open-source, first released in 2002, actively maintained (latest release Mar-30)
 - ▶ full / differential / incremental backup
 - ▶ each archive includes an index \Rightarrow fast file list/restore
 - ▶ build-in compression on a file-by-file basis
 - more resilient against data corruption
 - can avoid compressing already compressed files, e.g. `-Z "*.mp4" -Z "*.gz"`
 - ▶ strong encryption
 - ▶ can split archives at 1-byte resolution
 - ▶ supports extended file attributes, sparse files, hard and symbolic/soft links
 - ▶ can detect corruption in both headers and saved data, recover with minimal data loss
 - ▶ can merge two archives into a new one, e.g. can convert differential/incremental backup to full
- Full DAR /TAR comparison <http://dar.linux.free.fr/doc/FAQ.html#tar>

DAR on Compute Canada clusters

- System version of DAR is somewhat old, will throw (as far as I can see) harmless warnings due to a bug in 2.5.3

```
[login1:~]$ which dar
/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09/bin/dar
[login1:~]$ dar --version
dar version 2.5.3, Copyright (C) 2002-2052 Denis Corbin
...
```

- Latest version on Cedar / Graham / Béluga

```
/home/razoumov/dar/bin/dar --version
dar version 2.6.3, Copyright (C) 2002-2019 Denis Corbin
...
alias dar=/home/razoumov/dar/bin/dar
```

- Compiling your own

```
wget https://sourceforge.net/projects/dar/files/dar/2.6.3/dar-2.6.3.tar.gz
unpack and cd there
./configure --prefix=$HOME/dar --disable-shared
make
make install-strip
$HOME/dar/bin/dar --version
```

Seeking time

- On a laptop let's quickly generate 17GB of random data (don't do this on a cluster!)
- \$RANDOM is between 0 and $32767 = 2^{15} - 1$

```
cd /Volumes/boa/tmp    # on an external HDD (slower for a purpose)
mkdir test && cd test
for num in $(seq -w 000 999); do
    echo $num
    # generate a binary file (1-33)MB in size
    dd if=/dev/urandom of=test"$num" bs=1024 count=$(( RANDOM + 1024 ))
done
```

- Create TAR and DAR archives

```
cd ..
time tar cvf all.tar test          # 5m57.084s; created all.tar
time dar -w -c all -g test         # 6m27.853s; created all.1.dar
```

- Extract a single file: to clear cache, before each command delete test/test596, unmount the drive, mount it again, cd into parent directory

```
time tar xvf all.tar test/test596      # 0m7.876s    0m7.884s    0m7.785s
time dar -O -w -x all -v -g test/test596  # 0m0.525s    0m0.662s    0m0.594s
# dar -O -w -x all -v -g test           # will restore everything
```

Basic archiving and extracting

- Let's quickly generate 134MB of random data

```
cd ~/tmp
mkdir test && cd test
for num in $(seq -w 000 999); do
    echo $num
    # generate a binary file (8-264)kB in size
    dd if=/dev/urandom of=test"$num" bs=8 count=$(( RANDOM + 1024 ))
done
```

- Create a basic DAR archive

```
dar -w -c all -g test          # create DAR archive all.1.dar
dar -l all                     # list its contents
mkdir restore
dar -R restore/ -O -w -x all -v -g test/test596    # extract one file
dar -R restore/ -O -w -x all -v -g test            # extract entire directory
```

all .1.dar
base name
slice name
file/backup name

Incremental backups

- Create full backup `monday.1.dar`

```
/bin/rm all.1.dar
dar -w -c monday -g test
```

- Create first incremental backup `tuesday.1.dar`

```
dd if=/dev/urandom of=test/tue1 bs=8 count=$(( RANDOM + 1024 )) # add another random file
dar -w -A monday -c tuesday -g test
```

- Create second incremental backup `wednesday.1.dar`

```
dd if=/dev/urandom of=test/wed1 bs=8 count=$(( RANDOM + 1024 )) # add another random file
dd if=/dev/urandom of=test/wed2 bs=8 count=$(( RANDOM + 1024 )) # add another random file
/bin/rm test/test999
dar -w -A tuesday -c wednesday -g test
```

Incremental backups (cont.)

- Check each backup for test/test999

```
dar -l monday | grep test999      # it is there
dar -l tuesday | grep test999     # it is there
dar -l wednesday | grep test999   # shows REMOVED ENTRY
```

- Restore latest backup

```
dar -R restore -O -x wednesday    # will restore only last incremental backup (wed1, wed2)
```

- To restore everything ending with the latest backup, start with the full backup and go sequentially through all incremental backups

```
dar -R restore -O -w -x monday     # restore the full backup
dar -R restore -O -w -x tuesday
dar -R restore -O -w -x wednesday
```

Limiting the size of each slice

... to 10MB

```
dar -s 10M -w -c monday -g test      # from 134MB we get all.{1..14}.dar slices
dar -O -x monday                     # will extract all slices with 'monday' basename
```

Limiting the number of files in each slice with multidar

- `multidar()` function is included into `darFunctions.sh`

```
source /path/to/darFunctions.sh
multidar
multidar test 300
```

```
rm restore/*
dar -R restore -O -w -x test-aaa
dar -R restore -O -w -x test-aab
dar -R restore -O -w -x test-aac
dar -R restore -O -w -x test-aad
```

```
rm restore/*
for f in test-aa{a..d}
do
    dar -R restore/ -O -w -x $f
done
```

Backup

- `backup()` function is included into `darFunctions.sh` 📖 **let's study it**

```
backup
backup show
backup 0      # create full backup
dd if=/dev/urandom of=test/item001 bs=8 count=$(( RANDOM + 1024 )) # add random file
backup 1      # create first incremental backup
dd if=/dev/urandom of=test/item001 bs=8 count=$(( RANDOM + 1024 )) # add random file
backup 2      # create second incremental backup
...
```

- You can always go back (likely not `backup 0`)

```
backup 1      # this will overwrite all1.1.dar and remove all{2..}.1.dar
backup 2
...
backup show
```

- **Note 1:** With incremental backup, you do not need the full backup in `$BDEST` !!! ⇒ you can backup a large HDD/SSD to a much smaller USB drive!
- **Note 2:** another wonderful Linux/Mac/FreeBSD open-source backup utility is BORG
<https://borgbackup.readthedocs.io>

Restore

- `restore()` function is included into `darFunctions.sh` 📖 let's study it

```
restore
restore -l test999           # pay attention to [Saved] tag
restore -n 2 test/test999    # will not restore as this file is not saved in that backup
restore -n 0 test/test999    # will restore but this is not safe

restore -x test/test999      # the safest option
restore -x test               # restore the entire directory
```

- DAR does not understand wildmasks \Rightarrow need to specify full relative (to `$BREF` in `backup()`) directory or file path

Encryption

- Uncomment the encryption flag in `backup()` and source it

```
source /path/to/darFunctions.sh
/bin/rm -rf backups/all*
backup 0      # provide password (same password twice)
backup 1      # provide password (password for 0, then password for 1 twice)
...

restore -x test      # will ask for a password
```

- **Note:** each backup has its own password, which may not be convenient ...

Summary

- DAR is a modern-day command-line replacement to TAR with many nice features
- With all today's scripts and workflows, please exercise common sense
 - ▶ make sure you understand what you are doing: don't use these scripts as black boxes!
 - ▶ don't delete any file that you want to keep until you are completely sure it is in the archive
 - ▶ when running `backup()`, monitoring DAR file sizes really helps
 - ▶ these scripts assume that you are below your quota (so you can write files), have read and write permissions, etc
 - when above quota, if you are conscious about security, you can modify `multidar()` to write `.fullList` and `.partial*` to `/tmp` with `mktemp` and then set a trap at the beginning of `multidar()` to remove these files on exit
 - details in <https://www.putorius.net/working-with-temporary-files.html>

Questions?