Help Topic: The Duload Toolbox Scheme and Utilities

Spring 2020

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

The Duload Toolbox Scheme

The Duload Toolbox is a set of scripts designed to facilitate the transmission and management of files between a local computer and a remote computer. The Duload Toolbox is built around the usage scenario depicted below, designed to support a user working on a local computer with far less storage capacity than required to have a local copy of a group's entire project files:

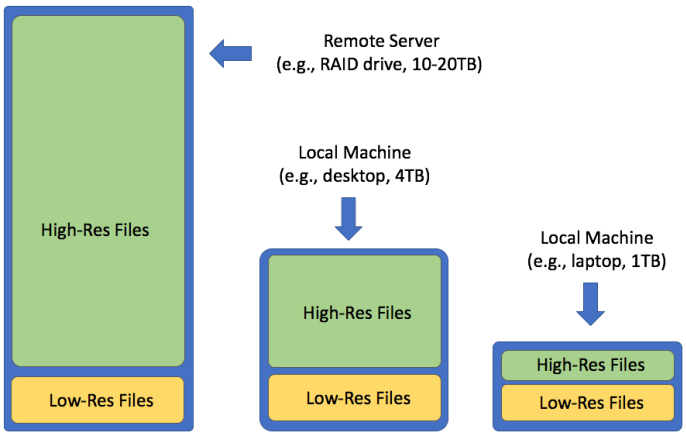

Figure 1: **The Server and Local Machine**: Typically a user has far less memory on a local machine, and typically may want all or most low resolution versions of files, but only a small subset of the high resolution versions.

A remote server, with access to a much larger hard drive, acts as the master copy, having a complete archive of the group's full project data. The server may itself have its own redundancy in the form of a RAID setup or daily offsite backups. (In the LAMSS/PavLab case, both)

The Duload Toolbox Scripts

The primary scripts for moving files to and from the remote server are `dload.pl` and `uload.pl` for downloading and uploading respectively. Each script is a Perl wrapper for the GNU/Linux command `rsync`, for remotely syncing file systems. This command is very powerful but prone to unintended results with even slightly mis-typed commands. The wrappers reduce chances for mistakes, but also simplify the interface for the user.

The third primary script is `rrf.sh`. This is used on a local machine to remove *raw* files, i.e., the high-resolution versions of their smaller counterparts. Of course you can just remove files with the

`rm` command, but `rrf.sh` makes certain tasks much easier.

The Duload Scheme is Not Version Control

The Duload tools do not provide version control. They simply make it easier to synchronize portions of your local project folder with a remote project folder. The upside of not being a version control system is that the local footprint is much smaller. In SVN for example, a *copy* of everything is stored locally, doubling the project footprint on the local computer.

Some conveniences of version control systems are indeed provided by the Duload tools (courtesy of `rsync` of course). For example:

- You can delete a file or folder locally, and simply type `dload.pl` to get it back.
- If the master project folder on the remote machine has had additions or modifications, you can just type `dload.pl` to pull down only the files that have changed.
- Similarly, if you are updating your local project folder and get interupted, `dload.pl` will pick up where you left off upon reconnection.
- If you'd like to check the differences between the master remote project folder and your local folder, you can just type `dload.pl --dry-run` or simply `dload.pl -d`, to see what files *would* be updated before actually pulling down these updates.

Again, this functionality is courtesy of the underlying `rsync` tool. But with `rsync`, you would have to remember the exact URL/path name of the folder you are currently in. If you get it wrong, you could start downloading unintended content requiring time consuming steps to counter the mistake.

Duload is distinctly *not* like version control in the following senses:

- If you upload a "wrong" version of a file, overwriting a previously uploaded correct version on the remote server, there is no way to undo this.
- You cannot remotely delete a file in the master folder on the server. You must log on to the remote machine and delete it. (Removing files is meant to be a bit hard)

The Special Designation of Raw Files

In the Duload scheme, the string `"raw_"` is used to begin either a file name, or directory name, to indicate a high-resolution version. As an example, a folder may contain drone video files shot at 4K `(3840x2160)` resolution, and the same files at a lower resolution `(1137x640)`. For example:

```
$ ls -hl
-rw-rw-rw-  1 joe  staff   1.2G Aug 14 13:59 RAW_DJI_0003.MOV   // ~30x as big!
-rw-rw-rw-@ 1 joe  staff   3.9G Aug 14 14:50 RAW_DJI_0004.MOV
-rw-r--r--@ 1 joe  staff    35M Aug 14 16:07 DJI_0003.mp4
-rw-r--r--@ 1 joe  staff   135M Aug 14 16:16 DJI_0004.mp4
```

The user may need the high resolution videos if composing an important project video. But perhaps only the low-resolution video is wanted after the project video is made, and the higher resolution version can "archived" by not including it on the local drive.

Similar but different file organization styles may be used. For example, a user may choose not to rename any files, but simply put the lower resolution files into different folders:

```
$ ls project
raw_vids/
   DJI_0003.MOV
   DJI_0004.MOV
vids/
   DJI_0003.MOV
   DJI_0004.MOV
```

In the above example, the "raw" nature of the files are indicated in the directory portion in the full path name `raw_vids/DJI_0003.MOV`. A group's project folder may be constructed with several directory layers and different file naming conventions. In the Duload scheme, any file or directory beginning with the term `"raw_"` (case insensitive) in its full path name is regarded as a raw *version* of a lower resolution file elsewhere in the project folder.

How are raw files used by the Duload utilities? The `dload.pl` utility can be invoked with the `--raw` flag to ignore raw files upon download. And since these raw files may be dispersed in several locations and layers in a folder, the `rrf.sh` utility helps hunt these files down for removal when space is needed locally.

No Such Thing as a Duload Folder or File

A folder structure used with Duload utilities is the same as any other folder or directory in a GNU/Linux or MacOS system. The *convention* indicating a raw file is merely a convenience for pulling down files from the server using `dload.pl` and removing files locally with `rrf.sh`.

The user has flexibility in how raw files are demarcated: as part of the file name, part of the directory name, or any part of the full path name. The tools treat the `"raw_"` string as case insensitive, i.e., the following files are all considered raw files:

- `raw_video.mp4`
- `RAW_IMG_0123.mov`
- `raw_videos/IMG_0123.mov`
- `files/raw_logs/LOG_GUS_18_8_2017____20_24_14.alog`

The following files are not considered raw files:

- `draw_video.mp4`
- `IMG_0123_RAW.mov`
- `videos_raw/IMG_0123.mov`

How Un-Raw Files are Made

By default a file in a project may be considered "un-raw". The notion of a raw file only comes into play when a lower-resolution or smaller version of an original file is created. At that point the

original file is simply regarded as raw, and usually re-named as such or put into a folder named as such, to allow the Duload tools to work their magic.

Some ways to make lower-resolution files:

- Saving clip of a video, for example a critical 20 second clip of an otherwise 30 minute video. Quicktime on MacOS supports this.
- Creating a lower resolution version of a video. The TotalVideoConverter app and `ffmpeg` utility can do this.
- Creating a lower resolution version of a photo. The `convert` utility in the ImageMagick package can do this. The MacOS Preview utility can also do this and many other tools.
- Creating a clip of a MOOS alog file. The `alogclip` tool.
- Creating a thinner version of a MOOS alog file. The `aloggrep`, `alogrm` and `alogpare` tools.

If You Really Need to Go Back In Time

As mentioned before, Duload utilities do not comprise a version control system. It is meant to be a light-weight toolbox for maintaining a master archive and easily having just the portions you want on a local machine.

That being said, one of the super nice features we lose by not having a version contol system, is the ability to go back to a previous version if someone truly does hose an important file on the master copy. In my experience, this is very rare but does happen. There are two ways to recover from this in the Duload scheme.

First, you may be lucky that a colleague has a local copy of the critical file and has not yet downloaded the newer broken file. That person's local copy can be used to restore the original file. It may even be that *you* are that colleague if you have files downloaded on two or more local machines. Second, if you arrange to have your server backed up nightly, if you have this service implemented in a journaling manner (think Time Machine for MacOS users), then you can use the back up service to help you recover the critical file.

In short - the Duload scheme comes from a conscious tradeoff to have a lightweight and fast mechanism to access a remote archive, at the expense of a full-on version control system. For source code development, version control is a must. For data archiving and access, the cost of version control is viewed here as being not worth the loss in space efficiency.

Other GNU/Linux Tools At Play

To effectively manage files on your local machine, especially where local space is limited, several other GNU/Linux tools are really useful.

- The `df` command ("display free disk space") shows you the remaining space on your local hard drive(s). Favorite options are `df -hl`. This tool tells you when you may need to free up some space by removing raw files with `rrf.sh`
- The `du` command ("display disk usage") shows you how much space is used in your current directory and recursively below. When you're running low on local disk space, this tool helps

you find the pigs. Favorite options are `du -d 1 -h` on the Mac, and `du --max-depth=1 -h` in GNU/Linux.

- The `rsync` command is perhaps the most powerful and complex of the bunch, which is why we chose to write the duload wrappers. But certain common command line options are simple go-to options for many things. And `rsync`, although originally conceived for remote file copying, can certainly be used for local copies. For copying or moving a large directory between two locally mounted drives, don't use the `mv` or `cp` commands. They are cumbersome if the operation is interupted. Use `rsync -aP` instead.
- The `find` command has evolved since the earliest days of Unix and is on all GNU/Linux and MacOS systems. Like many GNU/Linux tools, the output of find can be piped as input into another command line tool. This makes sequences of actions, like find-then-delete, possible. For example, if you want to remove all powerpoint files on your computer, you can:

```
find / -name '*.pptx' -exec rm -rf {} \;
```

  The `rrf.sh` utility is essentially a specialized wrapper for the `find` command.