GEOS 436 / 636
Programming and Automation for Geoscientists

– Week 09: Unix – Getting Data –

Ronni Grapenthin
rgrapenthin@alaska.edu
Elvey 413B
x7682

How can you (easily) get data (in an automated fashion)?

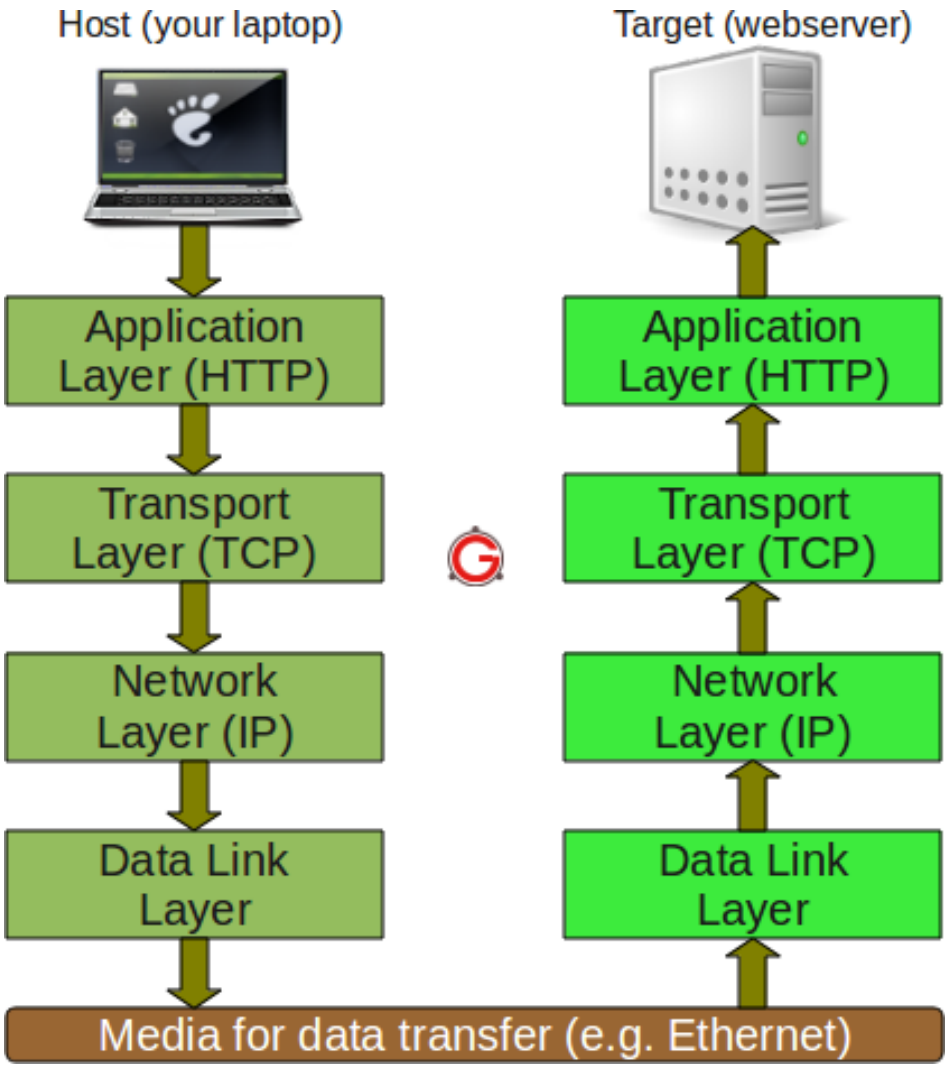
How do computer networks work? (at a high level)

The Internet (pop culture)

Search Google Images for "Internet":

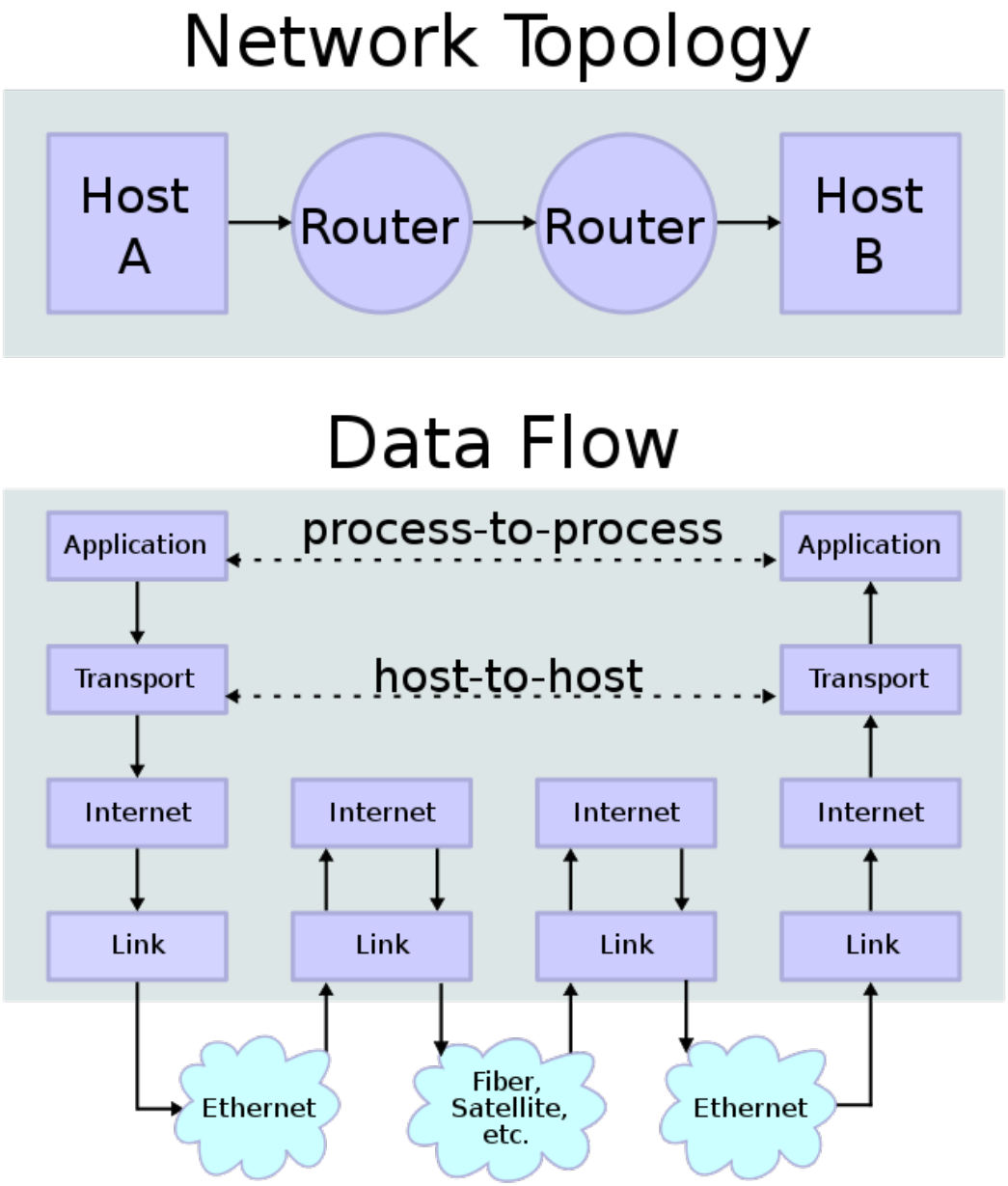
Google Images

Actually (1 client-server pair)



Geek Stuff.com

The Internet (really)



wikipedia

URLs

URLs

The most common form of URI is the Uniform Resource Locator (URL), which is known as the *web address*.

```
https://developer.mozilla.org
https://developer.mozilla.org/en-US/docs/Learn/
https://developer.mozilla.org/en-US/search?q=URL
```

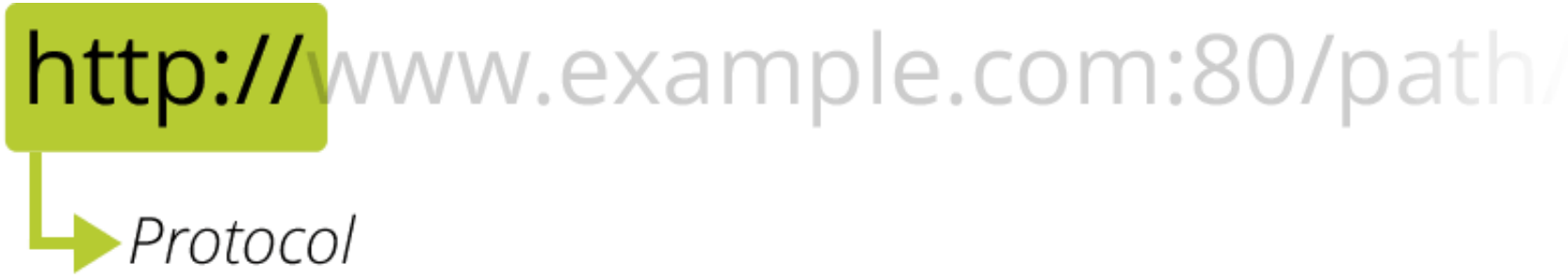
Any of those URLs can be typed into your browser's address bar to tell it to load the associated page (resource).

A URL is composed of different parts, some mandatory and others are optional. A more complex example might look like this:

```
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument
```

https://developer.mozilla.org

URLs: Protocol

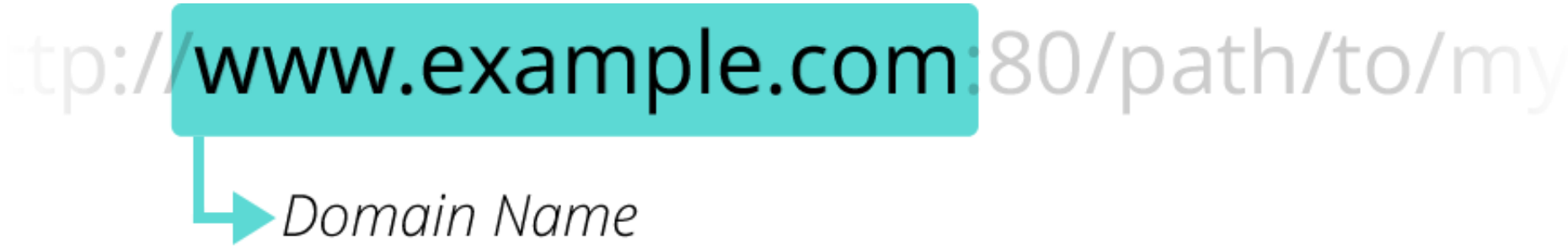


https://developer.mozilla.org

usual protocols:

- http, https – HyperText Transfer Protocol (Secure)
- ftp – FileTransfer Protocol (soon to be obsolete)
- mailto: - sometimes comes up to open email client

URLs: Domain Name



https://developer.mozilla.org

- Can be of varying complexity depending organization
- Generally we have top level domains: `.org`, `.edu`, `.com`, `.de`, ...
- You register your domain at a registrar, then you can have various subdomains
- e.g.: `gi.alaska.edu`
- `www.` traditionally (early web days) to distinguish webserver from ftp server (`ftp.`) from mailserver (`smtp.`, `mail.`) etc. and make it known to the public that you're talking about the Internet

URLs: Port

https://developer.mozilla.org

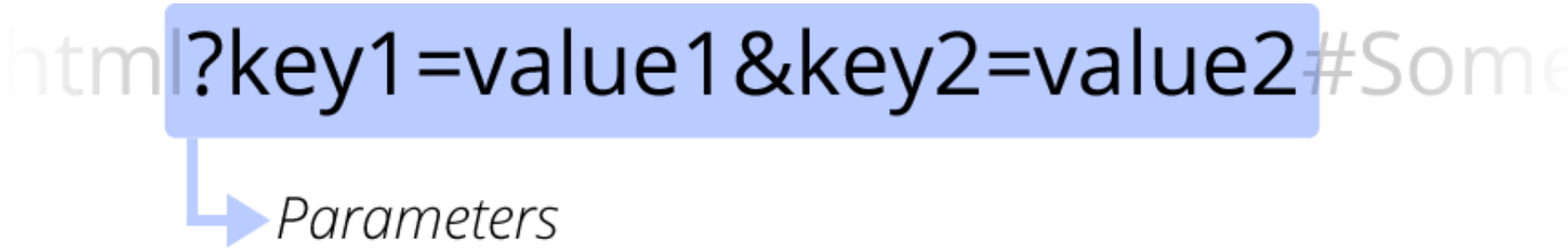
- Indicates which “entrance” to the server is to be used.

URLs: Path

https://developer.mozilla.org

- Used to be a path to a physical file on a webserver.
- These days there's a lot of abstraction happening so it may not reflect actual file structure anymore.

URLs: Query / Parameters

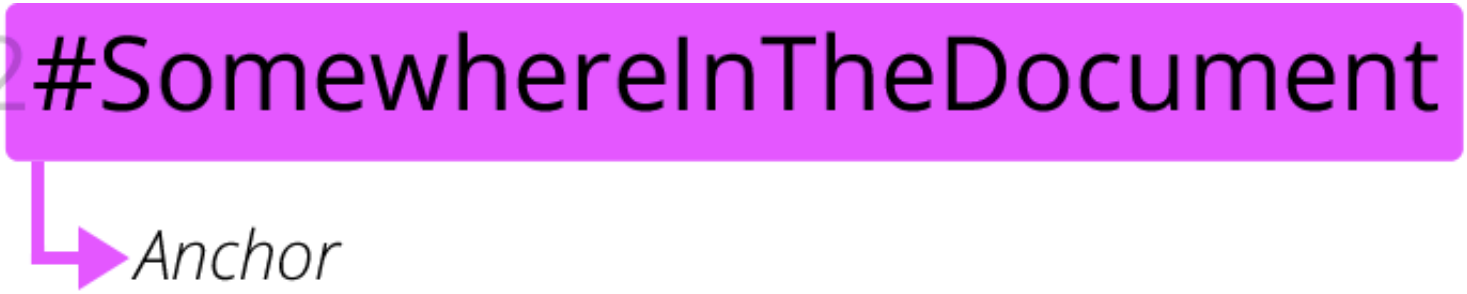


https://developer.mozilla.org

The (for our purposes) very useful part!

- Key-value pairs that are given to the webserver
- Webserver interprets these and responds dynamically to the request
- You may be able to exploit this in download scripts

URLs: Anchor



https://developer.mozilla.org

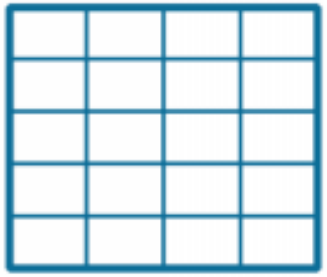
- Sort of a bookmark in a document, only used locally (not sent to server)
- For an HTML website, the browser scrolls to this spot for you (requires that someone put that anchor in the HTML website)
- For audio / video files, browser may go to time it represents

Data Storage

File System

```
C:\folder\music.m4a
```

sysadmin required for integrity and scale



Database / Structured Data

```
SELECT * FROM table;
INSERT INTO table;
```

sysadmin and DBA required for scale, integrity and performance

Object Storage

```
GET /object/KbglBn7qepo
PUT /object/KbglBn7qepo
```

sysadmin not required

ibm

- files: photo, document, ...
- database: subset records of interest
- object storage: the cloud (appears as file but isn't)

A very short list of public data archives

- IRIS: Seismic / infrasound data - `https://ds.iris.edu/ds/nodes/dmc/`
- UNAVCO: geodetic data (GNSS mainly, some SAR) - `ftp://data-out.unavco.org/pub/products/`
- ASF: SAR data - `https://search.asf.alaska.edu/`
- Waterdata USGS - `https://waterdata.usgs.gov/`
- NASA Global Sulfur Monitoring - `https://so2.gsfc.nasa.gov/`

Some allow for automated downloads, others don't.

Data Access

Is it a one-off download? Just do it manually.

If you suspect repeated use, invest some time in writing a script:

1. Explore the archive and strucutre: file-based? webservice?
2. Experiment with various methods of download & unpacking (see later)
3. Generalize for your use case to enable quick downloads in the future (command line arguments for your script).

Working on a remote machine: `ssh`

errorhat.com

```
$> ssh -XY user@server.edu
```

- `X,Y` options enable X11 (graphics) forwarding (view windows rendered on remote machine)
- `X` is more secure, sometimes has issues
- `Y` is a little less secure.
- use `exit` to close on remote machine

Working on a remote machine: `ssh`

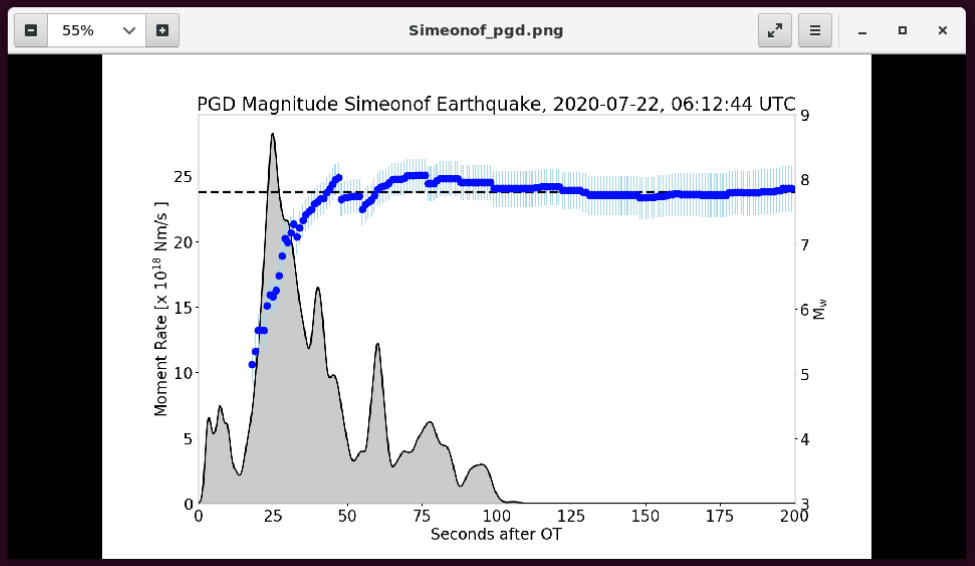
Example:

```
roon@rn-xps:~$ ssh -XY roon@katla.giseis.alaska.edu
roon@katla.giseis.alaska.edu's password:
Last login: Wed Oct 21 22:27:43 2020 from 10.25.249.21
[roon@katla ~]$ hostname
katla.giseis.alaska.edu
[roon@katla ~]$ eog Simeonof_pgd.png

(eog:5344): GLib-GIO-CRITICAL **: 22:28:49.475: g_dbus_proxy_new_sync: assertion 'G_IS_DBUS_CONNECTION (connection)' failed

(eog:5344): dconf-WARNING **: 22:28:49.546: failed to commit changes to dconf: Error spawning command line “dbus-launch --au
tolaunch=0b6aaeaf5fa24319aced552fed269945 --binary-syntax --close-stderr”: Child process exited with code 1

(eog:5344): dconf-WARNING **: 22:28:49.546: failed to commit changes to dconf: Error spawning command line “dbus-launch --au
tolaunch=0b6aaeaf5fa24319aced552fed269945 --binary-syntax --close-stderr”: Child process exited with code 1
```



Getting Data from Strangers: `wget, curl`

Once data access is sorted, you may need to download at the command line:

- `wget`
  - recursive downloads (entire directories)
  - HTTP(s), FTP only
  - default: download to a file
- `curl`
  - single URLs only, more protocols, download and upload
  - preinstalled on MacOS, Windows 10.
  - default: download to stdout, use `-o` to specify output file

For most of your needs they are the same and `wget` will get the job done quickly.

Getting Data from Strangers: `wget`

```
roon@rn-xps:~/wget_curl_test$ wget http://www.grapenthin.org/teaching/geop501/download/lab07/station.txt
--2020-10-21 21:59:04--  http://www.grapenthin.org/teaching/geop501/download/lab07/station.txt
Resolving www.grapenthin.org (www.grapenthin.org)... 74.207.254.32
Connecting to www.grapenthin.org (www.grapenthin.org)|74.207.254.32|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 738 [text/plain]
Saving to: ‘station.txt’

station.txt                   100%[=====================================================>]     738  --.-KB/s    in 0s

2020-10-21 21:59:04 (61.6 MB/s) - ‘station.txt’ saved [738/738]

roon@rn-xps:~/wget_curl_test$ ls
station.txt
```

Getting Data from Strangers: `curl` 1

```
roon@rn-xps:~/wget_curl_test$ curl http://www.grapenthin.org/teaching/geop501/download/lab07/station.txt
Name Lat Lon Elevation Type Number
ANMO 34.9500 -106.4600 1820 1 2
BAR 34.1500 -106.6280 2121 1 3
BMT 34.2750 -107.2600 1987 1 4
CAR 33.9525 -106.7340 1658 1 5
CBET 32.4200 -103.9900 1042 1 6
CL2B 32.2300 -103.8800 2121 1 7
CL7 32.4400 -103.8100 1032 1 8
CPRX 33.0308 -103.8670 1356 1 9
DAG 32.5913 -104.6910 1277 1 10
GDL2 32.2003 -104.3640 1213 1 11
HTMS 32.4700 -103.6000 1192 1 12
LAZ 34.4020 -107.1390 1878 1 13
LEM 34.1660 -106.9720 1698 1 1
LPM 34.3117 -106.6320 1737 1 14
MLM 34.8100 -107.1450 2088 1 15
SBY 33.9752 -107.1810 3230 1 16
SMC 33.7787 -107.0190 1560 1 17
SRH 32.4914 -104.5150 1276 1 18
SSS 32.3500 -103.4100 1072 1 19
Y22A 33.9370 -106.9650 1674 1 20
Y22D 34.0739 -106.9210 1436 1 21
WTX 34.0722 -106.9460 1555 1 22
roon@rn-xps:~/wget_curl_test$ ls
station.txt
```

Getting Data from Strangers: `curl` 2

```
roon@rn-xps:~/wget_curl_test$ curl http://www.grapenthin.org/teaching/geop501/download/lab07/station.txt -o station_curl.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   738  100   738    0     0   6099      0 --:--:-- --:--:-- --:--:--  6099
roon@rn-xps:~/wget_curl_test$ ls
station_curl.txt  station.txt
```

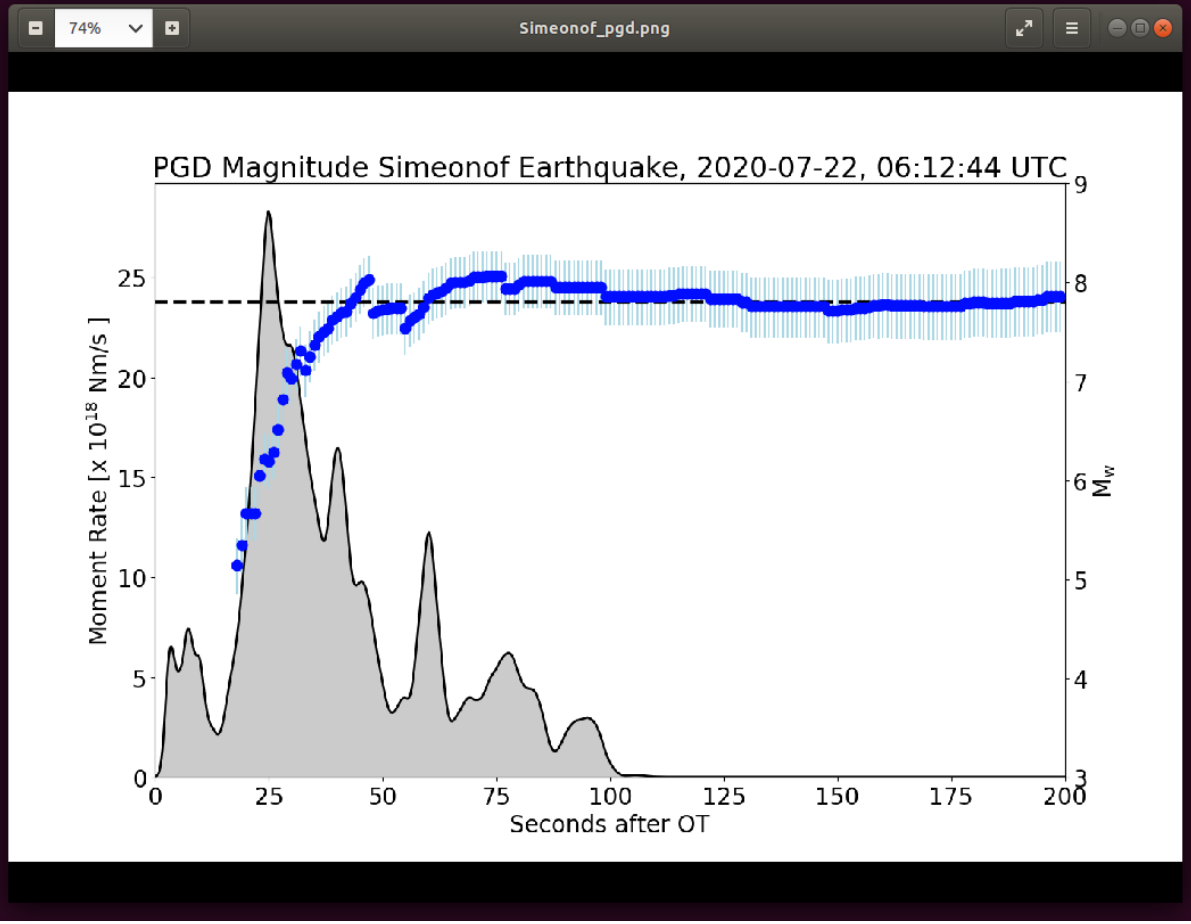
Transferring Data between (your) machines: `scp`

- Some work happens on remote machines
- Need to get some data / configuration files from local machine there
- You may need to retrieve your results from there to local machine
- Or transfer from one remote machine to another remote machine

```
$> scp OPTIONS SOURCE(s) TARGET
$> scp [[u1@]host1:]file1 ...  [[u2@]host2:]file2
```

Transferring Data between (your) machines: scp

```
roon@rn-xps:~/scp_test$ ls
roon@rn-xps:~/scp_test$ scp roon@katla.giseis.alaska.edu:/home/roon/Simeonof_pgd.png .
roon@katla.giseis.alaska.edu's password:
Simeonof_pgd.png                                    100%   98KB   5.4MB/s   00:00
roon@rn-xps:~/scp_test$ ls
Simeonof_pgd.png
roon@rn-xps:~/scp_test$ eog Simeonof_pgd.png
```



Transferring Data between (your) machines: scp

```
roon@rn-xps:~/scp_test$ ls
hello_world.sh  tilt.logs.tar.gz
roon@rn-xps:~/scp_test$ scp hello_world.sh tilt.logs.tar.gz roon@katla.giseis.alaska.edu
:/home/roon/scp_t/
roon@katla.giseis.alaska.edu's password:
hello_world.sh                                100%  178    41.5KB/s   00:00
tilt.logs.tar.gz                              100%   14MB  16.1MB/s   00:00
roon@rn-xps:~/scp_test$ scp * roon@katla.giseis.alaska.edu:/home/roon/scp_t/
roon@katla.giseis.alaska.edu's password:
hello_world.sh                                100%  178    50.5KB/s   00:00
tilt.logs.tar.gz                              100%   14MB  16.7MB/s   00:00
roon@rn-xps:~/scp_test$ ssh roon@katla.giseis.alaska.edu
roon@katla.giseis.alaska.edu's password:
Last login: Wed Oct 21 22:44:01 2020 from 10.25.249.21
[roon@katla ~]$ cd scp_t
[roon@katla scp_t]$ ls
hello_world.sh  tilt.logs.tar.gz
[roon@katla scp_t]$ cat hello_world.sh
#!/bin/tcsh

set week = 10

echo "Hello World!"
echo "This is Programming and Automation for Geoscientists"
echo "We are in week $week"
echo "The labs will be fantastic!"
```

Unpacking Data

- Data (in repositories) are often compressed
- Some form of archival format: gzip, tar-gzip, zip, etc.
- most common tools on command line:
  - `$> gzip file` – creates `file.gz`
  - `$> gunzip file.gz` – unpacks `file`
  - `$> tar cfz archive.tar.gz files/folder` – tars files/folder into one file which can be gzipped
  - `$> tar xfz archive.tar.gz` – ungzips and untars the archive
  - `$> unzip file.zip` – unzips the occassional zip file that comes by
- `tar` is convenient to stuff many files / folders into a single file, which then can be gzip compressed.

(Un)packing Data: `tar, gzip`

```
[roon@katla scp_t]$ ls
hello_world.sh  tilt.logs.tar.gz
[roon@katla scp_t]$ gzip hello_world.sh
[roon@katla scp_t]$ ls
hello_world.sh.gz  tilt.logs.tar.gz
[roon@katla scp_t]$ cat hello_world.sh.gz
N*⍟_hello_world.sh=⍟;⍟0D{⍟b⍟⍟@A⍟⍟
                                    Rj⍟⍟$+⍟⍟]⍟⍟B⍟M⍟⍟ySWMC⍟⍟⍟sJ⍟⍟⍟⍟#{⍟(⍟⍟⍟⍟⍟⍟4T⍟⍟^d*aY
8y⍟⍟e⍟⍟t⍟⍟⍟⍟J⍟⍟)⍟⍟⍟⍟ !⍟⍟⍟⍟⍟⍟⍟⍟⍟%⍟⍟+VN  =a

                                                ⍟⍟⍟;W⍟{⍟⍟⍟⍟u⍟[roon@katla scp_t]$
[roon@katla scp_t]$ gunzip hello_world.sh.gz
[roon@katla scp_t]$ ls
hello_world.sh  tilt.logs.tar.gz
[roon@katla scp_t]$ cat hello_world.sh
#!/bin/tcsh

set week = 10

echo "Hello World!"
echo "This is Programming and Automation for Geoscientists"
echo "We are in week $week"
echo "The labs will be fantastic!"

#EOF
[roon@katla scp_t]$ tar xfz tilt.logs.tar.gz
[roon@katla scp_t]$ ls
hello_world.sh  logs  tilt.logs.tar.gz
[roon@katla scp_t]$ ls logs/* | wc -l
151
[roon@katla scp_t]$ tar czf logs.tar.gz logs
[roon@katla scp_t]$ ls
hello_world.sh  logs  logs.tar.gz  tilt.logs.tar.gz
[roon@katla scp_t]$ ls -lisa
total 45420
97430136843     0 drwxr-xr-x  3 roon users      103 Oct 21 23:13 .
 2273215537     8 drwxr-xr-x 56 roon users     4096 Oct 21 22:57 ..
97430136846     4 -rwxr-xr-x  1 roon users      178 Oct 21 22:44 hello_world.sh
 7522466416    12 drwxr-xr-x  2 roon users     8192 Jun 25 12:20 logs
97430136844 30720 -rw-r--r--  1 roon users 15025505 Oct 21 23:13 logs.tar.gz
97430136847 14676 -rw-r--r--  1 roon users 15026098 Oct 21 22:44 tilt.logs.tar.gz
```

Inspecting Data

- Once you have the data, you need ways to figure out whether they are what you wanted.
- How can you do that quickly without firing up Excel?
- `less`, `head`, `tail`, `cat` - we talked about these tools during the last lecture.

Workflow

1. Manual exploration of archive
2. Tests with wget/curl/etc.
3. Unpacking (tar, gunzip)
4. Data inspection (less, more, cat)

1. Think about generalization
2. Start a shell script with results from manual work
3. Test on small, diverse subsets of data
4. Run script automatically, triggered

Putting it all together

Let's write a script to download some data from UNAVCO . . .