

File systems and transferring files

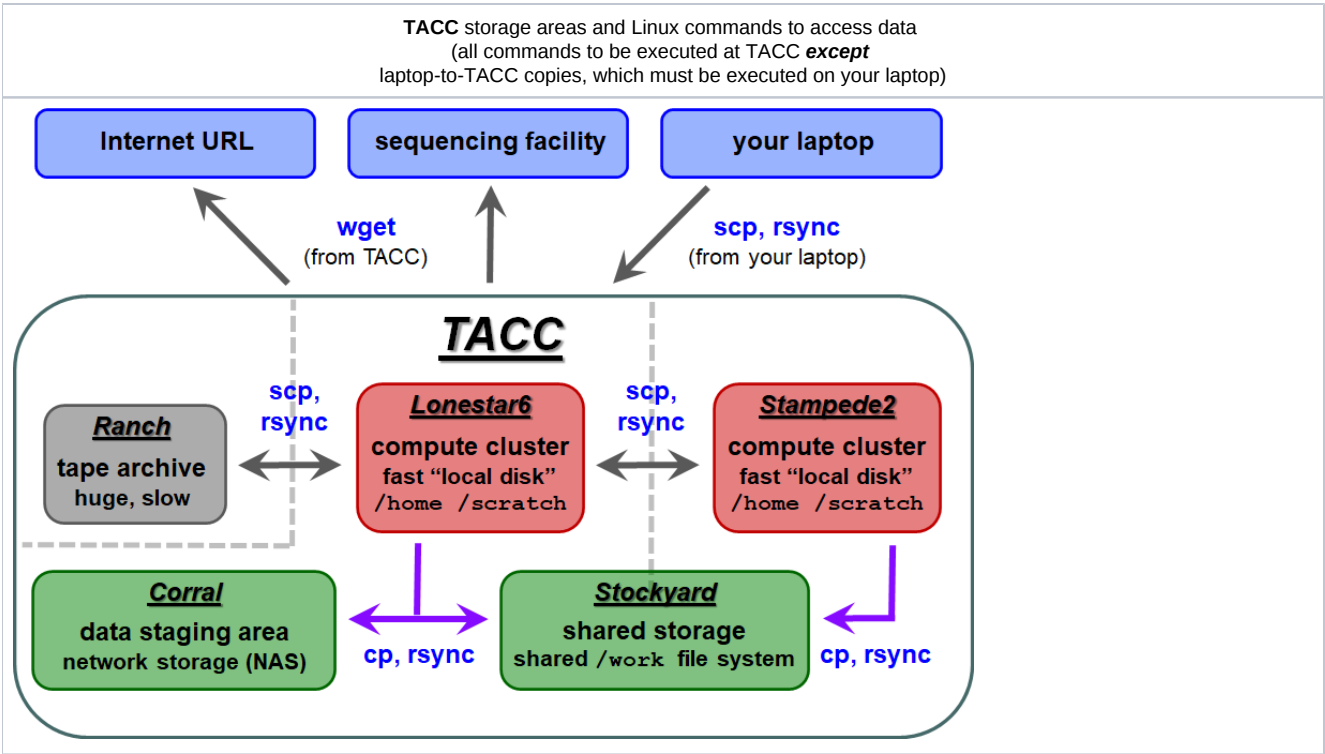
- File systems at TACC
 - Local file systems
 - changing TACC file systems
 - Stockyard (shared Work)
 - Corral
 - Ranch
- About file systems
- Staging your data
 - Download from a link – wget
 - Copy from a corral location - cp or rsync
 - cp
 - local rsync
 - Copy from a remote computer - scp or rsync
 - scp
 - remote rsync
- Scavenger hunt exercise
 - step by step answers

File systems at TACC

The first thing you'll want to do is transfer your sequencing data to TACC so you can process it there. Here is an overview of the different storage areas at TACC, their characteristics, and Linux commands generally used to perform the data transfers:

- **wget** – retrieves the contents of an Internet URL
- **cp** – copies directories or files located on any **local** file system
- **scp** – copies directories or files to/from a **remote** system
- **rsync** – copies directories or files on either local or remote systems

(Read more about [Copying files and directories](#))



Local file systems

There are 3 **local** file systems available on any TACC compute cluster (**stampede2**, **lonestar6**, etc.), and your account has a directory in each of the three.

The 3 file local systems have different characteristics, but all are fast and set up for parallel I/O.

On **lonestar6** these local file systems have the following characteristics:

	Home	Work	Scratch
--	------	------	---------

quota	10 GB	1024 GB = 1 TB	2+ PB (basically infinite)
policy	backed up	not backed up, not purged	not backed up, purged if not accessed recently (~10 days)
access command	cd	cdw	cds
environment variable	\$HOME	\$WORK (different sub-directory for each cluster) \$STOCKYARD (root of the <i>shared</i> Work file system)	\$SCRATCH
root file system	/home	/work	/scratch
use for	Small files such as scripts that you don't want to lose.	Medium-sized files you don't want to copy over all the time. For example, custom programs you install (these can get large), or annotation file used for analysis.	Large files accessed from batch jobs. Your starting files will be copied here from somewhere else, and your final results files will be copied elsewhere (e.g. stockyard , corral , your BRCF POD , or your organization's storage area.

When you first login, the system gives you information about disk quotas and your compute allocation balance in "**SU**" (*system units*).

```

----- Project balances for user abattenh -----
| Name           Avail SUs   Expires | Name           Avail SUs   Expires |
| OTH21095             905   2023-09-30 | MCB21106         1496   2023-09-30 |
| OTH21164             215   2024-05-31 | OTH21180          899   2024-03-31 |
----- Disk quotas for user abattenh -----
| Disk           Usage (GB)   Limit   %Used   File Usage   Limit   %Used |
| /scratch              0.7     0.0     0.00     567           0     0.00 |
| /home1                0.0    11.7     0.01     232           0     0.00 |
| /work               169.0   1024.0   16.50   79361       3000000    2.65 |
-----

```

changing TACC file systems

When you first login, you start in your Home directory. Use the **cd**, **cdw** and **cds** commands to change to your other file systems. Notice how your command prompt helpfully changes to show your location.

Changing file systems at TACC

```

cdw  # cd $WORK
cds  # cd $SCRATCH
cd   # cd $HOME

```



The **cd** (change directory) command with no arguments takes you to your Home directory on *any* Linux/Unix system.

The **cdw** and **cds** commands are specific to the TACC environment.

Stockyard (shared Work)

TACC compute clusters now share a common Work file system called **stockyard**. So files in your Work area do not have to be copied, for example from **stampede2** to **ls6** – they can be accessed directly from either cluster.

Note that there are two environment variables pertaining to the shared Work area:

- **\$STOCKYARD** - This refers to the root of your shared Work area
 - e.g. **/work/01063/abattenh**
- **\$WORK** - Refers to a sub-directory of the shared Work area that is different for different clusters, e.g.:
 - **/work/01063/abattenh/ls6** on **lonestar6**
 - **/work/01063/abattenh/stampede2** on **stampede2**

A mechanism for purchasing larger **stockyard** allocations (above the 1 TB basic quota) from TACC are in development.

The UT Austin Bioinformatics Team, a loose group of bioinformatics researchers, maintains a common directory area on **stockyard**.

The shared BioITeam directory

```
ls /work/projects/BioITeam
```

Files we will use in this course are in a sub-directory there. The **\$SCORENGS** environment variable set in your login profile refers to this path.

Our shared class directory

```
echo $SCORENGS
ls /work/projects/BioITeam/projects/courses/Core_NGS_Tools
```

Corral

corral is a *gigantic* (multiple PB) storage system (spinning disk) where researchers can store data. UT researchers may request up to 5 TB of **corral** storage through the normal TACC allocation request process. Additional space on **corral** can be rented for ~\$80/TB/year.

A couple of things to keep in mind regarding **corral**:

- **corral** is a great place to store data in between analyses.
 - Store your permanent, original sequence data on **corral**
 - Copy the data you want to work with from **corral** to **\$SCRATCH**
 - Run your analyses (batch jobs)
 - Copy your results back to **corral**
- Occasionally **corral** can become unavailable. This can cause any command to hang that tries to access **corral** data!

Ranch

ranch is a *gigantic* (multiple PB) tape archive system where researchers can archive data. All TACC users have an automatic 2 TB **ranch** allocation. UT researchers may request larger (multi-TB) **ranch** storage allocations through the normal TACC allocation request process.

There is currently no charge for **ranch** storage. However, since the data is stored on tape it is not immediately available – robots find and mount appropriate tapes when the data is requested, and it can take minutes to hours for the data to appear on disk. The metadata about your data – the directory structures and file names – is always accessible, but the actual data in the files is not on disk until "staged". See the **ranch** user guide for more information: <https://www.tacc.utexas.edu/user-services/user-guides/ranch-user-guide>.

Once that data is staged to the **ranch** disk it can be copied to other places. However, the **ranch** file system is not mounted as a local file system from the **stampede2** or **ls6** clusters. So remote copy commands are always needed to copy data to and from **ranch** (e.g. **scp**, **rsync**).

About file systems

File systems are storage areas where files and directories are arranged in a *hierarchy*. Any computer can have one or more file systems *mounted* (accessible as local storage). The **df** command can be used on any Unix system to show all the "top-level" mounted file systems. TACC has a lot of temporary file systems, so let's just look at the first 15 and tell **df** to use "human readable" size formatting with the **-h** option:

```
df -h | head -15
```

The rightmost **Mounted on** column gives the top-level access path. Find **/home1**, **/work**, and **/scratch** and note their **Size** numbers!

What do we mean by "hierarchy"? It is like a tree, with the *root file system* (denoted by the leading **/**) as the trunk, sub-directories as branches, sub-sub-directories as branches from branches (and so forth), with files as leaves off any branch.

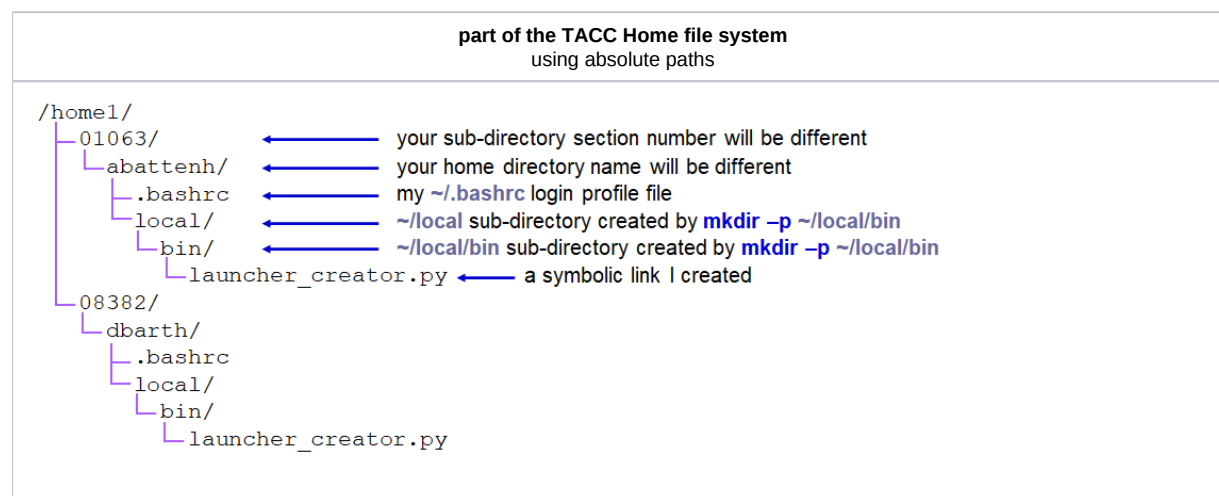
your TACC Home directory using home-directory-relative (~) path syntax

```
~/.bashrc      ← your ~/.bashrc login profile file
~/local/       ← ~/local sub-directory created by mkdir -p ~/local/bin
├── bin/       ← ~/local/bin sub-directory created by mkdir -p ~/local/bin
│   └── launcher_creator.py ← a symbolic link (file) you created
```

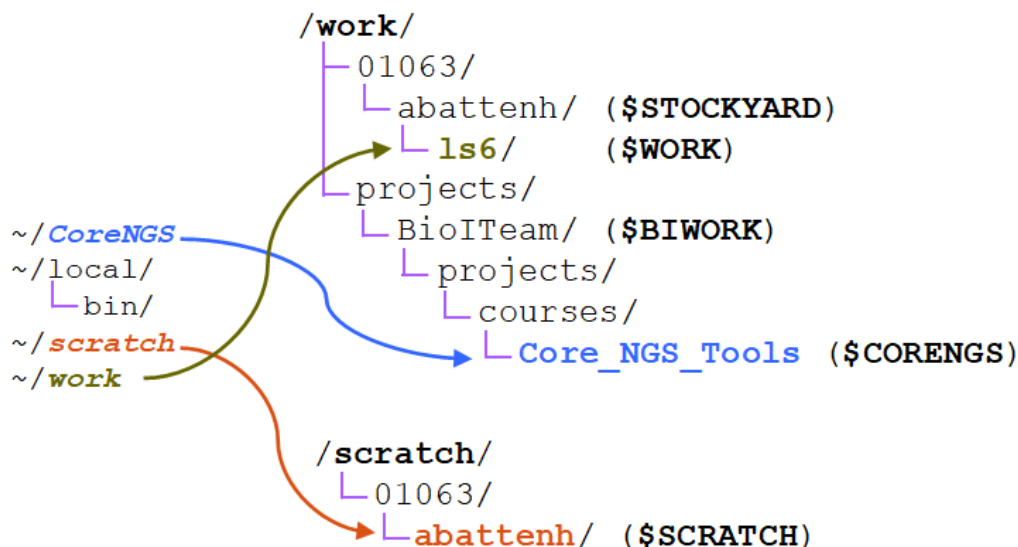
But everyone has a Home directory, so you must only be seeing a part of the Home directory hierarchy. To see the *absolute path* of a directory you're in, use the **pwd -P** command. Note that absolute paths always start with a forward slash (**/**) denoting the root file system.

```
pwd -P
# will show something like this
# /home1/01063/abattenh
```

That shows you that your Home directory (~) is actually 3 levels down in the `/home1` hierarchy:



Here's a depiction of the three file systems as seen from your Home directory (~), showing where the path-valued environment variables represent, and where the three symbolic links (`~/CoreNGS`, `~/scratch`, `~/work`) you created in your Home directory point. Notice that both the Work and Scratch file systems have a top-level hierarchy like we saw in Home above.



On many Linux systems, you can use the `tree` command to view the full file system hierarchy starting from a specified directory:

```
ls6:~$ tree ~
/home1/01063/abattenh
├── CoreNGS -> /work/projects/BioITeam/projects/courses/Core_NGS_Tools
├── local
│   └── bin
│       └── launcher_creator.py -> /work/projects/BioITeam/common/bin/launcher_creator.py
├── scratch -> /scratch/01063/abattenh
├── tmp
│   ├── a
│   └── b
│       └── c
└── work -> /work/01063/abattenh/ls6
```

Staging your data

So, your sequencing center has some data for you. They may send you a list of web or FTP links to use to download the data.

The first task is to get this sequencing data to a **permanent** storage area. This should **NOT** be your laptop! **corral** (or **stockyard**) is a great place for it, a **B RCF pod**, or a server maintained by your lab or company.

Here's an example of a "best practice". Wherever your permanent storage area is, it should have a rational sub-directory structure that reflects its contents. It's easy to process a few NGS datasets, but when they start multiplying like tribbles, good organization and naming conventions will be the only thing standing between you and utter chaos!

For example:

- **original** – for original sequencing data (compressed **FASTQ** files)
 - sub-directories named, for example, by **year_month.<sequencing run/job or project name>**
- **aligned** – for alignment data (**BAM** files, etc)
 - sub-directories named, e.g., by **year_month.<project_name>**
- **analysis** – further downstream analysis
 - reasonably named sub-directories, often by project
- **refs** – reference genomes and other annotation files used in alignment and analysis
 - sub-directories for different reference genomes and aligners
 - e.g. **ucsc/hg38/star**, **ucsc/sacCer3/bwa**, **mirbase/v20/bowtie2**
- **code** – for scripts and programs you and others in your organization write
 - ideally maintained in a version control system such as **git**, **subversion** or **cvs**.
 - can have separate sub-directories for people, or various shared repositories.

Download from a link – wget

Well, you don't have a desktop at TACC to "Save as" to, so what to do with a link? The **wget** program knows how to access web URLs such as **http**, **https** and **ftp**.

Get ready to run **wget** from the directory where you want to put the data.

Don't press Enter after the **wget** command – just put a space after it.

Get ready to wget

```
mkdir -p $SCRATCH/archive/original/2021.core_nginx
cd $SCRATCH/archive/original/2021.core_nginx
wget
```

Here are two web links:

- https://web.corral.tacc.utexas.edu/BioinformaticsResource/CoreNGS/yeast_stuff/Sample_Yeast_L005_R1.cat.fastq.gz
- https://web.corral.tacc.utexas.edu/BioinformaticsResource/CoreNGS/yeast_stuff/Sample_Yeast_L005_R2.cat.fastq.gz

Right-click (Windows) or **Control+click** (Mac) on the 1st link in your browser, then select "**Copy link location**" from the menu. Now go back to your Terminal. Put your cursor after the space following the **wget** command then either **right-click** (Windows), or Paste (**Command-V** on Mac, **Control-V** on Windows). The command line to be executed should now look like this:

wget to retrieve a web URL

```
wget http://web.corral.tacc.utexas.edu/BioinformaticsResource/CoreNGS/yeast_stuff/Sample_Yeast_L005_R1.cat.fastq.gz
```

Now press **Enter** to get the command going. Repeat for the 2nd link. Check that you now see the two files (**ls**), or tree \$SCRATCH to see your Scratch directory hierarchy:

```
ls6:~$ tree $SCRATCH
/scratch/01063/abattenh
|-- archive
|   |-- original
|       |-- 2021.core_nginx
|           |-- Sample_Yeast_L005_R1.cat.fastq.gz
|           |-- Sample_Yeast_L005_R2.cat.fastq.gz
```



By default **wget** creates a file in the current directory matching the last component of the URL (e.g. **Sample_Yeast_L005_R1.cat.fastq.gz** here). You can change the copied file name with **wget's -O** option.

Also note that if you execute the same **wget** more than once, subsequent local files will be named with a **.1**, **.2**, etc. suffix.

Copy from a corral location - cp or rsync

Suppose you have a **corral** allocation or **stockyard** area where your organization keeps its data, and that the sequencing data has been downloaded there. You can use various Linux commands to copy the data locally from there to your **\$SCRATCH** area.

cp

The **cp** command copies one or more files from a **local** source to a **local** destination. It has many options, but the most common form is:

```
cp [options] <source_file_1> <source_file_2> ... <destination_directory> /
```

Make a directory in your Scratch area and copy a single file to it. The trailing slash (**/**) on the destination says the destination is a directory.

Single file copy with cp

```
mkdir -p $SCRATCH/data/test1
cp $CORENGS/misc/small.fq $SCRATCH/data/test1/
ls $SCRATCH/data/test1

# or..
cds
mkdir -p data/test1
cd data/test1
cp $CORENGS/misc/small.fq .

# or..
mkdir -p ~/scratch/data/test1 # use the symbolic link in your Home directory
cd ~/scratch/data/test1
cp $CORENGS/misc/small.fq .
ls
```

(Read more about using [Absolute or Relative pathname syntax](#))

Now copy an entire directory to your Scratch area. The **-r** option says **recursive**.

Directory copy with cp

```
mkdir -p $SCRATCH/data
cds
cd data
cp -r $CORENGS/general/ general/
```

Exercise: What files were copied over?

```
ls general
# or
tree $SCRATCH/data
```

```
BEDTools-User-Manual.v4.pdf SAM1.pdf SAM1.v1.4.pdf
```

local rsync

The **rsync** command is typically used to copy whole directories. What's great about **rsync** is that it **only copies what has changed** in the source directory. So if you regularly **rsync** a large directory to TACC, it may take a long time the 1st time, but the 2nd time (say after downloading more sequencing data to the source), only the new files will be copied.

rsync is a very complicated program, with many options (<http://rsync.samba.org/ftp/rsync/rsync.html>). However, if you use the recipe shown here for directories, it's hard to go wrong:

```
rsync -avW local/path/to/source_directory/ local/path/to/destination_directory/
```

Both the source and target directories are local (in some file system accessible directly from **lonestar6**). Either full or relative path syntax can be used for both. The **-avW** options above stand for:

- **-a** means "archive mode", which implies the following options (and a few others)
 - **-p** – preserve file **p**ermissions
 - **-t** – preserve file **t**imes
 - **-l** – copy symbolic links as **l**inks
 - **-r** – recursively copy sub-directories
- **-v** means **v**erbose
- **-W** means transfer **W**hole file only
 - Normally the **rsync** algorithm compares the contents of files that need to be copied and only transfers the different parts.
 - For large files and binary files, figuring out what has changed (**diff**-ing) can take more time than just copying the whole file.
 - The **-W** option **disables file content comparisons** (skips **diff**-ing).

Since these are all **single-character options**, they can be combined after one option prefix dash (-). You could also use options **-ptlrW**, separately, instead of using **-a** for "archive mode".



Always add a trailing slash (/) after directory names

The trailing slash (/) on the source and destination directories are **very important** for **rsync** – and for other Linux copy commands also!

rsync will create the **last directory level** for you, but earlier levels must already exist.

rsync (local directory)

```
mkdir -p $SCRATCH/data
cds
rsync -avrW $CORENGS/custom_tracks/ data/custom_tracks/
```

Exercise: What files were copied over?

```
ls $SCRATCH/data/custom_tracks
# or
ls ~/scratch/data/custom_tracks
# or
cds; cd data/custom_tracks; ls
# or
tree $SCRATCH/data
```

Now repeat the **rsync** and see the difference.

Use the **Up arrow** to retrieve the previous command from your **bash** command history.

```
rsync -avrW /work/projects/BioITeam/projects/courses/Core_NGS_Tools/custom_tracks/ data/custom_tracks/
```



The **bash** shell has several convenient **line editing** features:

- use the **Up arrow** to scroll back through the command line history; **Down arrow** goes forward
- use **Ctrl-a** to move the cursor to the beginning of a line; **Ctrl-e** to the end
- **Ctrl-k** ("kill") to delete all text on your command line after the cursor
- **Ctrl-y** ("yank") to copy the last killed text to where the cursor is

Once the **cursor** is positioned where you want it:

- Just type in any additional text you want
- To delete text **after** the cursor, use:
 - **Delete** key on Windows
 - **Function-Delete** keys on Macintosh
- To delete text **before** the cursor, use:
 - **Backspace** key on Windows
 - **Delete** key on Macintosh

(Read more about [Command line history and editing](#))

Copy from a remote computer - scp or rsync

Provided that the remote computer is running Linux and you have **ssh** access to it, you can use various Linux commands to copy data over a secure connection.

The good news is that once you have learned **cp** and local **rsync**, remote secure copy (**scp**) and remote **rsync** are very similar!

scp

The **scp** command copies one or more files from a source to a destination, where either source or destination, or both, can be a remote path.

Remote paths are similar to local paths, but have user and host information first:

user_name@full.host.name:/full/path/to/directory/or/file

– Or –

user_name@full.host.name:~/path/relative/to/home/directory

Copy a single file to your **\$SCRATCH/data/test1** directory from the server named **dragonfly.icmb.utexas.edu**, using the user account **corengstools**. When prompted for a password, use the one we have written to the Zoom chat (or copy/paste the password from this file: **\$CORENGS/tacc/dragonfly_access.txt**)

single remote file copy with scp

```
cat $CORENGS/tacc/dragonfly_access.txt
cds
mkdir -p data/test2
scp -p corengstools@dragonfly.icmb.utexas.edu:~/custom_tracks/progeria_ctcf.vcf.gz ./data/test2/
tree ./data/test2
```

Notes:

- The 1st time you access a new host the SSH security prompt will appear
- You will be prompted for your remote host password
 - for security reasons characters will not be echoed
- The **-r** recursive argument works for **scp** also, just like for **cp**
- The **-p** argument says to preserve the file's last modification time
 - otherwise the last modification time of the local copy will be when the copy was done

remote rsync

rsync can be run just like before, but using the remote-host syntax. Here we use two tricks:

- The **tilde (~)** at the start of the path means "**relative to my Home directory**"
- We use the **tilde (~)** in the destination to traverse the **~/scratch** symbolic link created in your home directory.

rsync (remote directory)

```
cat $CORENGS/tacc/dragonfly_access.txt
rsync -avrW corengstools@dragonfly.icmb.utexas.edu:~/custom_tracks/ ~/scratch/data/custom_tracks/
```

Exercise: Was anything copied?

No, because all the source files were already present in the destination directory (you copied the same files earlier) with the same names, file sizes and timestamps. So **rsync** had nothing to do!

Scavenger hunt exercise

Here's a fun scavenger hunt for more practice of the commands you've learned so far.

Hit **Tab Tab** as much as possible to save typing!

To get started:

Play a scavenger hunt for more practice

```
cd
cp -r /work/projects/BioITeam/projects/courses/Core_NGS_Tools/linuxpractice/what what
# or using the $CORENGS environment variable
cp -r $CORENGS/linuxpractice/what what
cd what
cat readme
```

Where are you when you're all done?

```
ls6:~/what/starts/here/changes/the/world
```

step by step answers

From inside your [~/what](#) directory:

Play a scavenger hunt for more practice

```
mkdir starts
cd starts
cp /work/projects/BioITeam/projects/courses/Core_NGS_Tools/linuxpractice/steps/nextInstr .
cat nextInst
```

From inside your [~/what/starts](#) directory:

Play a scavenger hunt for more practice

```
mkdir here
cd here
wget http://web.corral.tacc.utexas.edu/BioinformaticsResource/CoreNGS/step3.txt
cat step3.txt
```

From inside your [~/what/starts/here](#) directory:

Play a scavenger hunt for more practice

```
scp -r /work/projects/BioITeam/projects/courses/Core_NGS_Tools/linuxpractice/changes/ changes/
# or
rsync -ptrvP /work/projects/BioITeam/projects/courses/Core_NGS_Tools/linuxpractice/changes/ changes/
# Note: rsync -avrP ... will also work, but will report an error because the destination file and
# directory ownership cannot be changed to match the source. But the files will be copied, and
# ownership assigned to you.

# Then
cd changes
more largeFile.txt
```

From inside your [~/what/starts/here/changes](#) directory:

Play a scavenger hunt for more practice

```
rsync -avrP corengstools@dragonfly.icmb.utexas.edu:~/the/ the/
# or
scp -r corengstools@dragonfly.icmb.utexas.edu:~/the/ the/

cd the
cat instr5.txt
cd world
cat instr6.txt
```

The path to the directory you're in now should be:

[~/what/starts/here/changes/the/world](#)