INTERSECT

Data

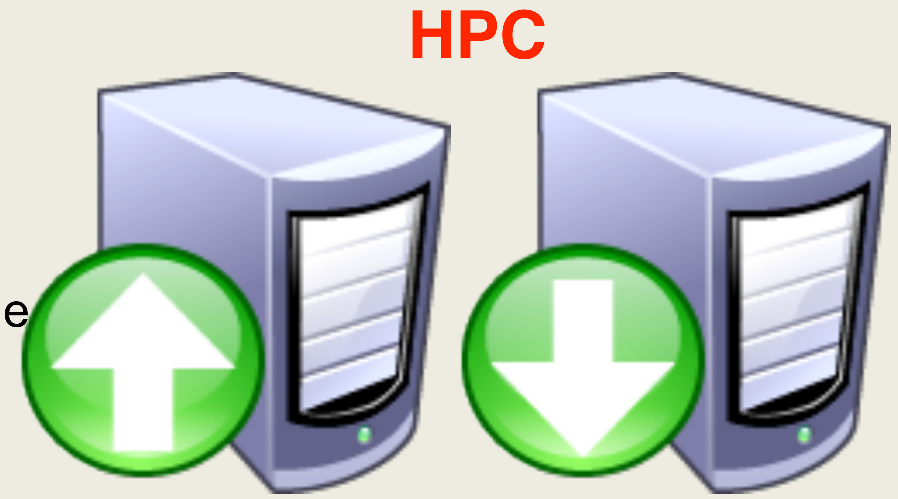
Introduction to Unix for HPC
Raijin version

Unit 1: Using SFTP

Goals:

- Can login via Secure SFTP and see home directory.
- Can transfer a file from local machine via SFTP to home directory.
- Understands the difference between DOS and UNIX formats.
- Can convert files between DOS and UNIX formats.
- Can transfer a file created on the server back to local machine.

The Problem



- We can interact with the command line over SSH.
- We can create directories (folders), move between them.
- We can create files and move them around, delete them, etc.
- We can run a program and get some output.

But how can we

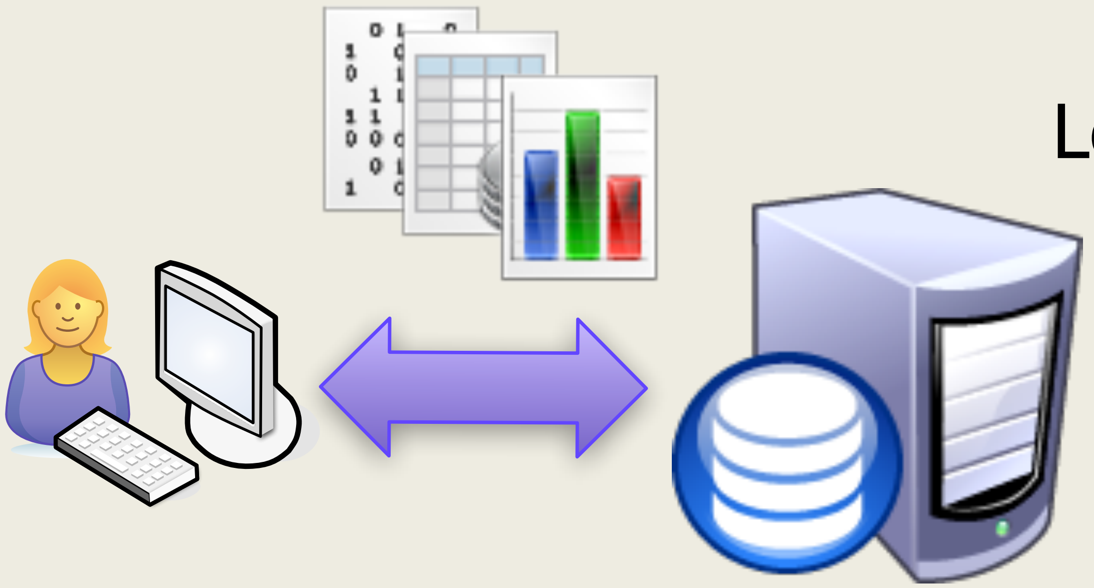
- Upload files, say our datasets, to the HPC machine?
- Download files, say the results of our analysis, from the HPC machine for further analysis locally?

INTERSECT

The Solution

- Well there are two to choose from:
  - SFTP = Secure File Transfer Protocol
  - SCP = Secure Copy (covered later)

Let's start with SFTP



Connecting to an FTP Server

- Raijin runs a SFTP server
- To connect to it you need an FTP client, such as FileZilla.
- You tell the client which server to connect to:

  `raijin.nci.org.au`

- Port: 22
- You authenticate with your username and password.

Exercise 1(a)

Connecting to Raijin and see your local files as well as the files in your home on Raijin.

CR + LF

```
Line 1 <CR><LF>
Line 2 <CR><LF>
Line 3 <CR><LF>
Line 4 <CR><LF>
Line 5 <CR><LF>
```

DOS
Windows

LF

```
Line 1 <LF>
Line 2 <LF>
Line 3 <LF>
Line 4 <LF>
Line 5 <LF>
```

Unix
Linux
Mac OS X

- Because Raijin is a Linux machine and our local machines run Windows, we need to convert our text files.

Exercise 1(b)

Transfer a text file to Raijin and convert.

Command	Description
`file` *`<file>`*	Determines the file type of *`<file>`*
`dos2unix` *`<file>`*	Converts files from DOS/MAC to UNIX text file format
`recode` *`<file>`*	Converts files between various character sets and surfaces, e.g. from UNIX to DOS
`recode latin1..dos` *`<file>`*	Converts file *`<file>`* to DOS format

INTERSECT

Unit 2: Using SCP

Goals:

- Can use **`pscp`** to upload multiple files at once.
- Can use **`pscp`** to download multiple files at once.

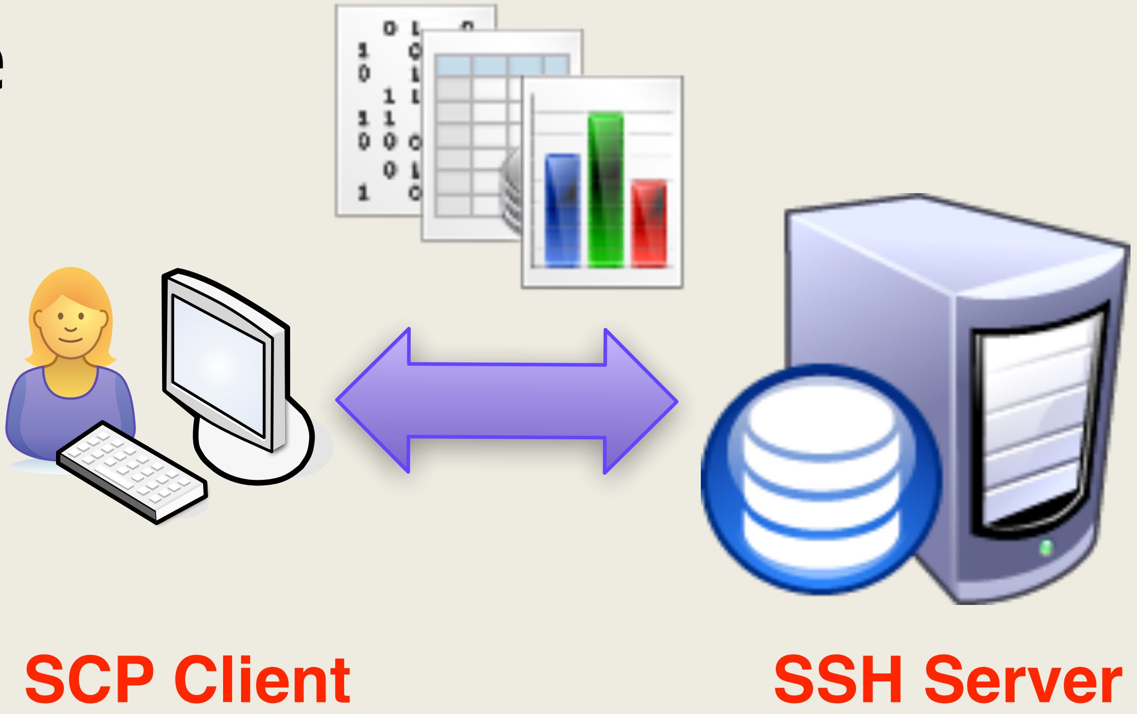
INTERSECT

SCP and PSCP

- Secure Copy (`scp`) is another way to transfer files to and from Raijin.
- Like SFTP, it is secure.
- Unlike SFTP, we'll be invoking it from the Windows command line
- SCP is non-interactive and, therefore, it can be **scripted**.

SCP vs SFTP

- SCP does the same task as SFTP.
- Use which-ever tool you feel more comfortable with.



INTERSECT

PSCP – An SCP Client

- Putty comes with an SCP client `pscp.exe` – Putty Secure Copy.
- We'll be using PSCP in the exercises.
- To use it we need to open a **Windows Command Prompt**.
- An easy way to do this is to select *Run…* from the Start menu and type `cmd`. Then click *OK*.

INTERSECT

PSCP Syntax

- Transfer file from local machine to the HPC machine:

```
pscp <file_name.ext> <user_name>@raijin.nci.org.au:<dest_dir>
```

The local file you want to copy

Your training account user name

Where you want the file to go on the HPC machine

INTERSECT

PSCP Syntax

- Transfer file from the HPC machine to your local machine:

```
pscp <user_name>@raijin.nci.org.au:<path/to/file_name.txt> .
```

Your training account user name

The path on the HPC machine to the file you want to copy

Where to put the file locally. In this case "." for the current working directory

Unit 3: Using Wget

Goals:

- Can use the basic **`wget`** syntax to download a dataset from the web.

INTERSECT

The Problem

- We want to use a dataset that’s available on the web.
- How do we get it on to Raijin?

To do this:

- On Raijin type:

```
wget <URL_to_file>
```

- This will download the dataset to your current working directory

INTERSECT

Unzip it!

- In order to save transfer time many files on the web are bundled in archives and compressed. Often they are zip files.
- Other common archive formats have extensions **`tar`**, **`tgz`**, **`gz`**, **`bz2`**.
- It sounds confusing, but try these simple recipes…

Recipes

File Extension	Command to unzip
`.zip`	`unzip <file_name>.zip`
`.tar`	`tar xvf <file_name>.tar`
`.tgz`	`tar xzvf <file_name>.tgz`
`.gz`	`gunzip <file_name>.gz`
`.bz2`	`bunzip2 <file_name>.bz2`

If the file ends with an extension in the left column, use the corresponding command in the right column.

Unit 4: Editing files in place

Goals:

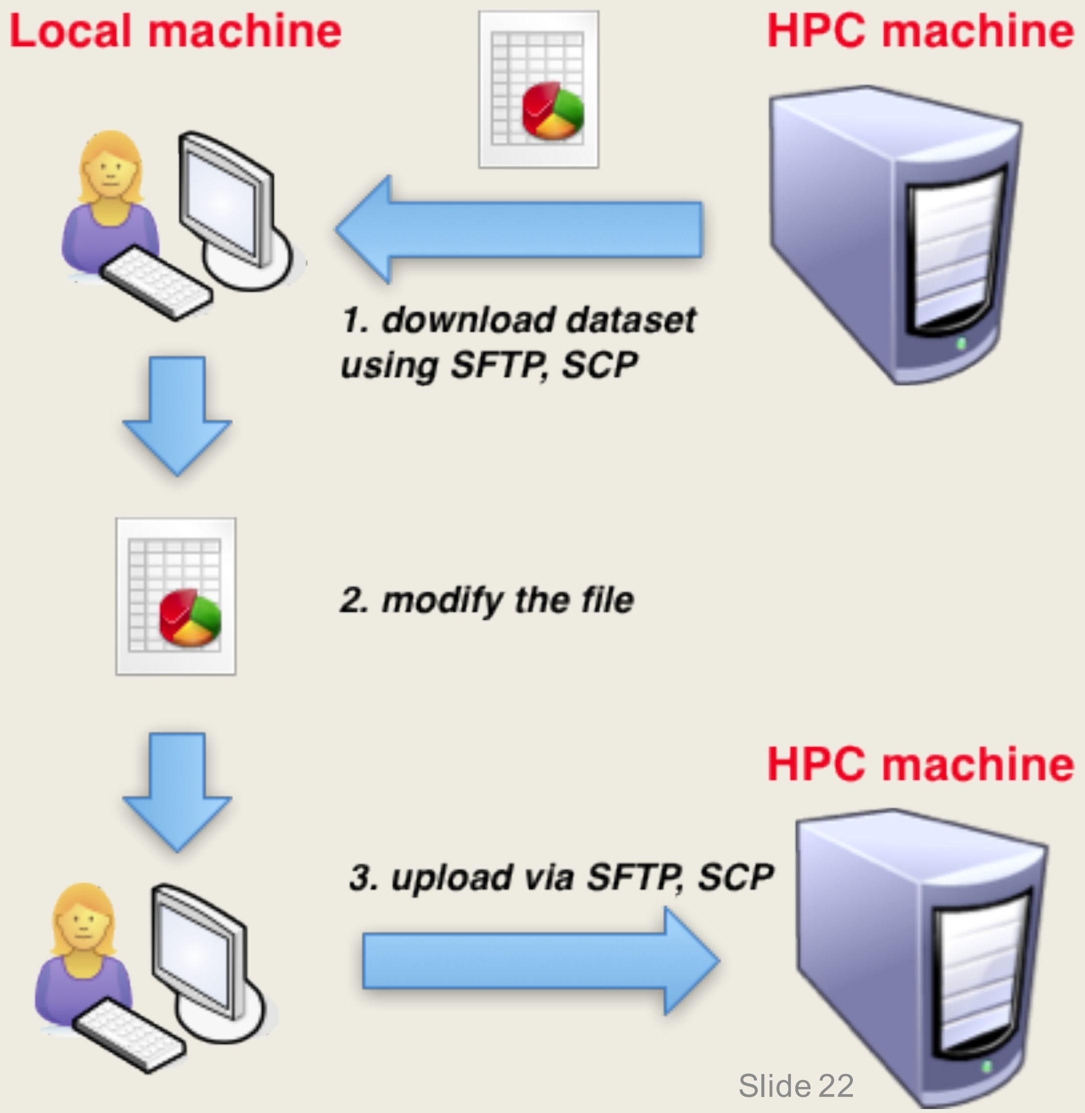
- Can edit a file, save changes, and exit editor.

The Problem

- We want to make a quick change to a
  - Dataset
  - Script

INTERSECT

We could, of course, do this:



But it can get cumbersome…

INTERSECT

Instead we can edit in-situ using Nano

- Invoke with:

```
nano <file_name.txt>
```

- Use the arrow keys to navigate your document
- Update the text by typing, backspace, etc.
- Use Control+O to save the file (^O).
- Use Control+X to quit (^X).

Exercise 4(a)
Editing a file in-situ

Command	Description
`nano` *`<file>`*	Will open file *`<file>`* in a text file editor
`CTRL+O`	Using **`CTRL+O`** within the **`nano`** editor will cause any changes made to the file while editing to be saved to the file
`CTRL+X`	Using **`CTRL+X`** within the **`nano`** editor will cause the editor to close. If you have not already saved your changes you will be asked if you wish to save those changes by answering **`Y`** or **`N`**

INTERSECT

Unit 5: Permissions and Ownership

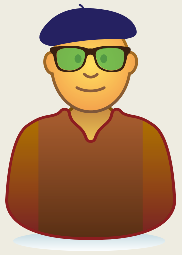
Goals:

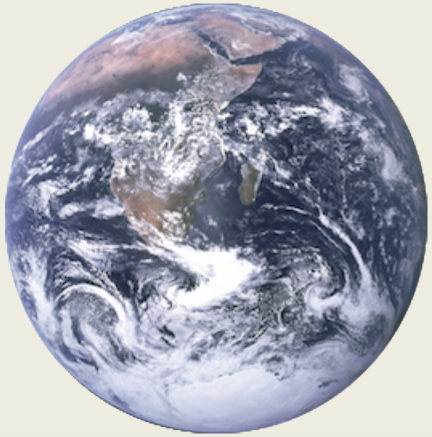
- Can make a file private.
- Can allow another to read and write to a file.
- Can make a script executable.

INTERSECT

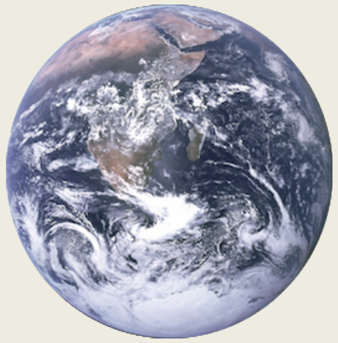
Security

- Security is baked right into Unix
  - All **files** and **directories** have security attributes
  - Access can be limited by **Read**, **Write**, and **Execute** permissions
  - Based on **Owner** – **Group** – **World** model:





For every file and directory:

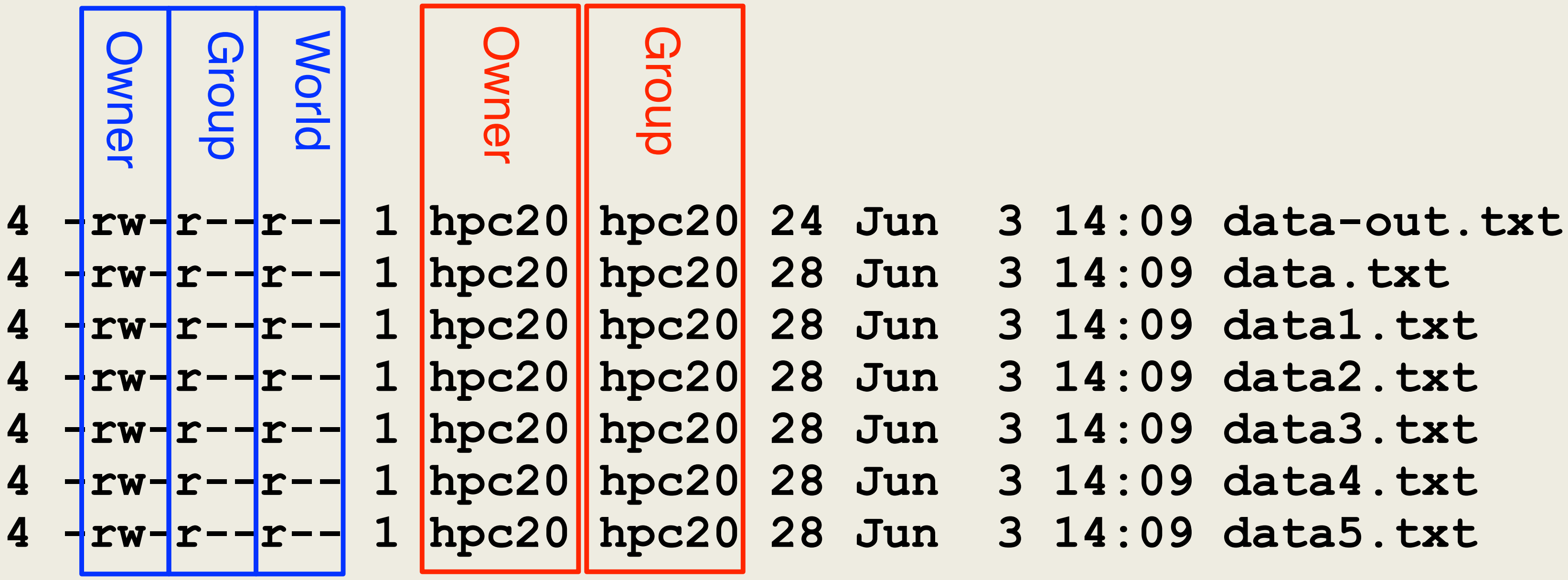


	Owner			Group			World		
Can?	r	w	x	r	w	x	r	w	x
Can't?	-	-	-	-	-	-	-	-	-
	Read	Write	eXecute	Read	Write	eXecute	Read	Write	eXecute

Checking file permissions

`ls -ls` reveals all:

Permissions for file:



```
4 -rw-r--r-- 1 hpc20 hpc20 24 Jun  3 14:09 data-out.txt
4 -rw-r--r-- 1 hpc20 hpc20 28 Jun  3 14:09 data.txt
4 -rw-r--r-- 1 hpc20 hpc20 28 Jun  3 14:09 data1.txt
4 -rw-r--r-- 1 hpc20 hpc20 28 Jun  3 14:09 data2.txt
4 -rw-r--r-- 1 hpc20 hpc20 28 Jun  3 14:09 data3.txt
4 -rw-r--r-- 1 hpc20 hpc20 28 Jun  3 14:09 data4.txt
4 -rw-r--r-- 1 hpc20 hpc20 28 Jun  3 14:09 data5.txt
```

It may come as a surprise, but by default the text files I created are “world readable”!

Handy commands

- **Ch**ange **Own**er (chown)

```
chown <new_owner> <file_name>
```

- **Ch**ange **Gr**ou**p** (chgrp)

```
chgrp <new_group> <file_name>
```

- **Ch**ange **Mod**e (chmod)

```
chmod <new_mode> <file_name>
```

- We'll concentrate on the last one.

Chmod Recipes

Command	Result
`chmod o-rwx` *`<file_name>`*	Forbid others (the world) to read, write and execute.
`chmod u+rwx,g-rwx,o-rwx` *`<file_name>`*	Grant owner full permissions; deny all others access.
`chmod a+w` *`<file_name>`*	Allow everyone (owner, group, and the world) write access.
`chmod u+x` *`<file_name>`*	Allow the owner the execute permission.
`chmod go-w` *`<file_name>`*	Forbid members of group and the world to write to file.
`chmod a=r` *`<file_name>`*	Grant everyone read access (only)