UTA Tech Orientation

Spring 2019

Overview

- Why is Tech Stuff Important?
- The Filesystem
- Shell Commands
- File Permissions
- Miscellaneous

Why is Tech Stuff Important?

Why is TA Tech Stuff Important?

- Different way of interacting with the department file system than as a student of a course
- Most courses will use the department file system directory (`/course/cs####`) communally for both their staff and the entire course simultaneously
  - Need to make sure that the correct groups of people can access only the correct files
    - e.g. students can see the website, but not the answers to their current project
  - You will have access to files which are not public and not just yours
- This presentation will help you solve basic technical problems that occur frequently during a TAship. **You can refer back to this during the semester.** Even the most experienced TAs forget this stuff sometimes...

The Filesystem

The Filesystem (1/2)

- A *filesystem* is a single tree, stemming from one root
- The “root directory” is literally just “`/`”
- To access files further up the tree, we begin at the root, “`/`”, and append a sequence of directories to find a particular file.
  - Directories in the path are separated by “`/`”s
  - Example: `/course/cs0150` or `/gpfs/main/home/eavril`

The Filesystem (2/2)

- The current directory you are in is your *working directory*
- A path can be abbreviated by being referred to relative to your *working directory*
- If a path does not lead with a "`/`" then it is relative.
  - Working directory: `/gfps/main/home/zkirsche`
  - Absolute path: `/gfps/main/home/zkirsche/cs0150/AndyBot`
  - Relative path: `cs0150/AndyBot`

Navigating the Filesystem (1/3)

- Change working directory with '`cd`' command
- Print working directory with '`pwd`'
  - Also usually displayed in your prompt
- Special paths
  - '.' = current directory
  - '. .' = one level up from current directory
  - '~' = your home directory (`/gfps/main/home/<your login>`)
- Example: current directory is `/course/cs0170/bin`
  - Where will you be if you `cd ./../handin/.././.../..`?

Navigating the Filesystem (2/3)

If your current directory is `/course/cs0170/bin`, where will you be after `cd ./../handin/../../../..`?

A. `/`
B. `/course`
C. `/course/cs0170`
D. `/course/cs0170/bin`

Navigating the Filesystem (3/3)

If your current directory is `/course/cs0170/bin`, where will you be after `cd ./../handin/.././../..`?

A. `/`
B. **`/course`**
C. `/course/cs0170`
D. `/course/cs0170/bin`

Navigating the Filesystem: Important Places

- Your home directory, `/gpfs/main/home/<yourlogin>` or `~`
- Your course's directory `/course/cs####/`
  - Oftentimes, the course directory has directories like:
    - `handin`: where students can submit their projects to
    - `www`: the website!
    - `bin`: course-specific commands, such as, cs0150_handin
    - `admin`: a subdirectory just for (H)TAs
- There is an important difference between `~/course/cs####` and `/course/cs####`!
  - The first is your personal directory for that course, the other is the shared course directory

Navigating the Filesystem: Hidden Files

- Files/folders starting with '`.`' are “hidden”
  - When you list, ‘`ls`’, the contents of a directory, you will not see the hidden files
  - If you would like to see all files in a directory including hidden files, use ‘`ls -a`’
- Because of the leading ‘`.`’, hidden files are often referred to as ‘dotfiles’
- Files are usually hidden because they serve some system-level internal function, like preferences, and are not needed to be seen by the user on an everyday basis
  - There are a lot of hidden preference files in your home directory
  - For example, `~/.bashrc` is used to set up your terminal settings every time you open a new terminal! You could customize your terminal from here if you so choose!

Snapshots (1/4)

- An automatic backup system. Sweet!
- Every directory has a hidden "`.snapshots`" directory.
  - You can always "`cd .snapshots`"
- Each snapshot directory has hourly, nightly, and weekly snapshot subdirectories that are labeled with timestamps
- Each of these contains the contents of the original folder at the time the timestamp lists (note the timestamps are in GMT not EST)
  - As time goes on, to preserve memory/space, snapshots are condensed from hourly, to nightly, to weekly and then eventually deleted

Snapshots (2/4)

What happens if you accidentally make a solution file in your course directory world-readable, and it gets saved on a snapshot?

A. Ask your HTA to remove it from the snapshot
B. Ask the MTAs to remove it from the snapshot
C. E-mail problem@cs.brown.edu to remove it from the snapshot
D. You need to wait until the snapshot expires, since you can't delete from a snapshot

Snapshots (3/4)

What happens if you accidentally make a solution file in your course directory world-readable, and it gets saved on a snapshot?

A. Ask your HTA to remove it from the snapshot
B. Ask the MTAs to remove it from the snapshot
C. E-mail problem@cs.brown.edu to remove it from the snapshot
D. You need to wait until the snapshot expires, since you can't delete from a snapshot

Snapshots (4/4)

- **WARNING**: No one (not even system administrators) can modify snapshots
- The snapshots will be an exact replica, preserving the file states and the file permissions
- This makes it possible for a mistake in file permissions to persist in snapshots for the duration of a snapshot's life
    - Worst case scenario (will be covered more later): Accidental allowed access to private files, such as grades or handins or solutions, to the public for an undetermined amount of time
- Foreshadow: it is very important to make sure that permissions are set correctly on sensitive materials such as grades and solutions.

Shell Commands

Shell Commands

- Usual syntax: `<program> [options...] [arguments...]`
  - Example: `grep -i -r "lambda" /course/cs0170`
- To find out about syntax and various options, use the `man` command
  - `man` is short for "manual"
  - This will explain just about everything you need for a particular command. It is incredibly helpful, use this command frequently!
  - Example usage: `man grep`

Helpful Shell Commands

- `man` – manual pages
- `cd` – change directory
- `ls` – list files in a directory
- `pwd` – print working directory
- `mv` – move files (and rename files)
- `cp` – copy files
- `grep` – search for a string with a file

Helpful Shell Tricks (1/2)

- `*` is a wildcard which refers to any number of any character
  - e.g., `*.java` means all files ending in ".java" or `cs0150_*` is any file beginning with "cs0150_"
- Type "`&`" after a command name to run it in the background
  - By default, a terminal command takes the entire focus of the terminal until the command finishes–that is called the "foreground". For things like ls, cd, pwd, etc., that can be almost instantaneous, but for running a program, like xpdf, eclipse, or gedit, you might run those for hours!
  - Background means that the terminal will run the command but then allow you to keep using the terminal as normal
  - Example usage: `evince demoDocument.pdf &`

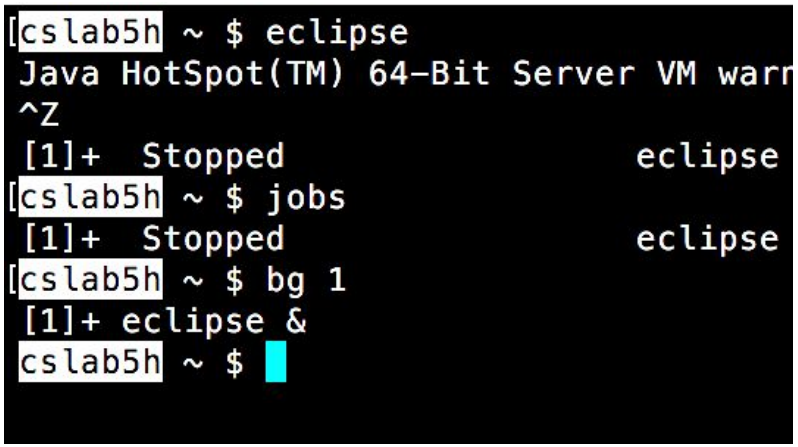
Helpful Shell Tricks (2/2)

- Press `CTRL-C` to kill (end and close immediately) a running process.
- Press `CTRL-Z` to suspend (pause, regaining terminal control) a running process.
- `jobs` shows you running processes
- Use `fg` and `bg` to move jobs to foreground and background, respectively
  - `bg` resumes a job you suspended, but in the background.
  - `fg` brings a job in the background to the foreground.

Helpful Shell Tricks: An Example

You go to open Eclipse and accidentally forget to add the &, now it's running in the foreground and you can't use your terminal. Eclipse is working fine, so you don't want to "kill" it, but you want to move the program to the background.

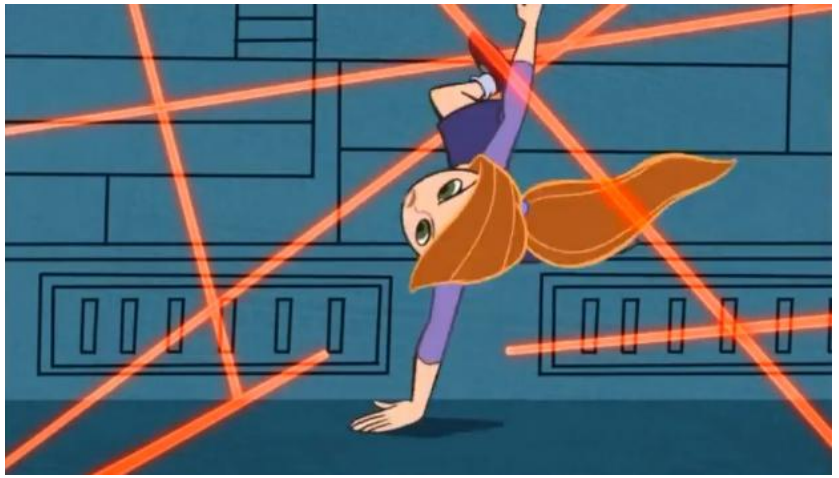
- Step 1: Suspend Eclipse to regain control of your terminal.
  - Send a Stop Signal with `CTRL-Z`
- Step 2: Identify the "job number" that corresponds to Eclipse.
  - Print what the terminal is running using `jobs`
- Step 3: Move Eclipse to the background
  - Background the program by job number, `bg 1`



Helpful Shell Tricks - Tab Autocomplete

- You can press “tab” in the shell to autocomplete paths.
- Type part of a name, then hit tab.
  - If there's only one possibility, the rest of the name will be filled in.
  - If there are several possibilities, the common part will be filled in, and a list of possibilities will be shown if you hit tab again.

File Permissions



Overview of Permissions

- Every file and directory in the UNIX filesystem is protected and shared through permissioning
- Permissions dictate who can interact with a file or directory and to what extent they can interact with the file or directory

Overview of Permissions: *How*

These are the way that files or directories can be interacted with:

- Files
  - `read` – can view contents of file
  - `write` – can change contents of file
  - `execute` – can run file as a program
- Directories
  - `read` – can list files in the directory
  - `write` – can create, rename, delete files from the directory
  - `execute` – can follow paths through directory

Overview of Permissions: *Who*

Each of these three categories may have unique permissions (read/write/execute) to a particular file or directory:

- Owner
  - The user who owns the file/directory
  - Can’t be a group of users (e.g. can’t be cs-1230ta)
- Group
  - The group or user who owns the file/directory
- World
  - Any user who has access to the filesystem

Permissions (1/4)

Permissions can be viewed using ls -l and look like this:

```
-rwxr-xr-- 1 twd cs-0330ta 0 Jan 16 17:18 test.txt*
```

- Owner
  - The owning user is `twd`, which is denoted as the first of the two names
  - The owner permissions are "`rwx`", denoting read, write, and execute are all allowed
- Group
  - The group who owns this file is `cs-0330ta`, as denoted by the second name
  - The group permissions are "`r-x`", so only read and execute are allowed
- World
  - Anyone in the world has "`r--`", only read, access to this file (assuming they have access to the directory in which it lives)

Permissions (2/4)

- Sometimes, permissions are represented as 3 octal (base-8) digits
  - 1st digit → user
  - 2nd digit → group
  - 3rd digit → world
- Each digit is determined as the sum of read, write, and execute permissions where:
  - `4 = read, 2 = write, 1 = execute`
  - For example:
    - `775` represents `rwxrwxr-x`
    - `770` represents `rwxrwx---`
    - `641` represents `rw-r----x`

Permissions (3/4)

You want to make a file so that it is

- Read, write, and execute access from the owner
- Read and execute access from the group
- No access from the world

Which of the following permissions is correct?

A. 012
B. 770
C. 750
D. 777

Permissions (4/4)

You want to make a file so that it is

- Read, write, execute access from the owner `rwx` ⇒ `4 2 1` ⇒ `7`
- Read and execute access from the group `r-x` ⇒ `4 0 1` ⇒ `5`
- No access from the world `---` ⇒ `000` ⇒ `0`

Which of the following permissions is correct?

A. 012
B. 770
C. 750
D. 777

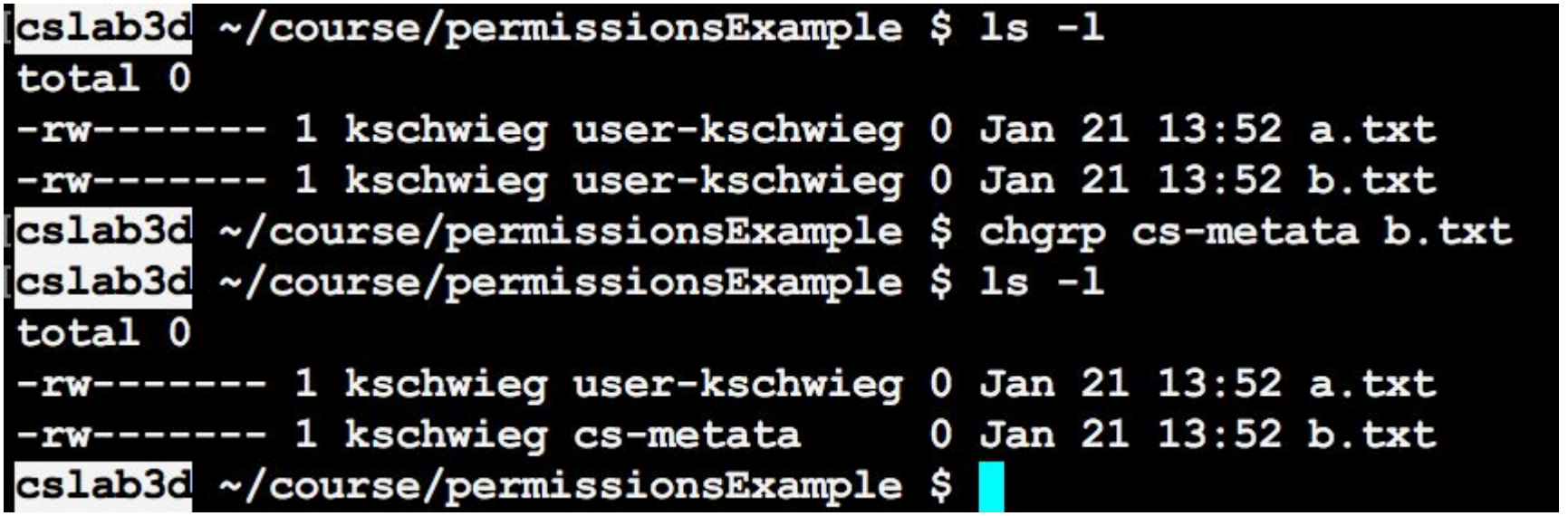
Changing Permissions (1/2)

- `chmod` changes file permissions
- Usage: `chmod [args] <newmode> <files…>`
  - `newmode` are the permissions you are switching to
  - The -R flag is a useful argument. It means Recursive and will go into all directories in the selected files.
- `<newmode>` can be numeric or *symbolic (won't be covered)*

```
cslab3d ~/course/permissionsExample $ ls -l
total 0
---------- 1 kschwieg user-kschwieg 0 Jan 21 13:52 a.txt
-rw-rw---- 1 kschwieg cs-metata     0 Jan 21 13:52 b.txt
cslab3d ~/course/permissionsExample $ chmod 777 a.txt
cslab3d ~/course/permissionsExample $ ls -l
total 0
-rwxrwxrwx 1 kschwieg user-kschwieg 0 Jan 21 13:52 a.txt*
-rw-rw---- 1 kschwieg cs-metata     0 Jan 21 13:52 b.txt
cslab3d ~/course/permissionsExample $
```

Changing Permissions (2/2)

- `chgrp` changes the owning group
- Usage: `chgrp [args] <newgroup> <files…>`
  - `newgroup` is the group you are switching to
  - The -R flag is a useful argument. It means Recursive and will go into all directories in the selected files.



Using File Permissions (1/2)

- In course directories, files usually have one of two permissions:
  - Public files (stencil code, docs, etc) usually have permissions `775` (`rwxrwxr-x`)
  - Private files (solutions, grades, etc) usually have permissions `770` (`rwxrwx---`)
- Most files should have group `cs-####ta`
  - HTA only information should have `cs-####hta` as the group
- Student's work should not be world-readable

Using File Permissions (2/2)

- When you make files, the default permissions will often be restricted to just you
  - There are exceptions to this rule, always check the permissions of sensitive files
- Be sure to `chmod` and `chgrp` as appropriate
  - If you don't, your fellow TAs might not be able to access the files which will be annoying for them!
- Always ask if you're in doubt about what the permissions of a file should be!

Miscellaneous

Remote Access

- `ssh` allows you to get a shell from a computer outside the department
- `sftp` command-line interface for moving files over a remote connection
- Fast-x is a graphical interface that allows you to access the department systems from another computer
- Cyberduck - ***don't use it!***
- More resources covered here: http://www.cs.brown.edu/facilities/system
- Note: differences in personal computers can make remote access and working at home different for different students. Be mindful!

Helpful Commands

- `elbow` – see information about a user, can search by name or login
  - May or may not be working for you
- `su / sux` – log in as another user.
- `zwrite` – use to send another user a message.***
- `ncdu` - if you run out of space, can delete from cache

Getting Paid

- In order to be paid for your work, you must log your hours and submit weekly timesheets on Workday
- Workday can be found at http://www.brown.edu/go/wd
- Any questions should be directed to your HTA or the MTAs

Credit vs Pay

- You can choose to TA for credit or pay
- If you're TAing for credit, register for cs0081 or cs0082 and don't log hours after TA camp!
- If you're TAing for pay, log all of the hours you work
- If you're TAing 15, 17, 33, or 123, you're TAing for credit **and** pay — this means that you will be paid for hours spent grading and the credit will cover everything else

Resources

- Sunlab consultants (sit at 9a)
- mta@cs.brown.edu can help answer questions or redirect you to resources as necessary
- problem@cs.brown.edu manages the departmental systems and handle many tech-related issues
- http://www.cs.brown.edu/facilities/system Great documentation on everything
- http://cs.brown.edu/courses/ta/pubs/uta_missive.pdf

Questions?