

UTA Tech Orientation

What's with the tech stuff?

Fall, 2011

MTA-Tech

Rutledge Chin Feman (rutledge@cs.brown.edu)

(taken from Chris Siden, MTA Tech 10-11, who just copied it from Robert Mustacchi, MTA Tech 09-10, who mostly lifted from Andrew Brindamour, MTA Tech 08-09, parts stolen from Colin Gordon, MTA Tech 07-08, whose presentation shamelessly built upon the work of Dave Pacheco, MTA Tech 06-07, who no doubt stole from Michael Shim, MTA Tech 05-06, and so on...)

Summary

- **Why is tech stuff important?**
- Filesystem
- Using shell commands
- File permissions
- Miscellaneous

Why is tech stuff important?

- TAs spend more time than most undergrads working on departmental systems
- TAs use more parts of the system than most undergrads
- TAs have access to files which are not public
- TAs should be able to help students with basic technical problems

As a result...

- Consequences of mistakes are larger in scope
 - releasing solutions, grades, exams publicly
 - messing up grades
- You need to be familiar with parts of the system some people never know about
- You should be aware of the common technical issues students face

Summary

- Why is tech stuff important?
- **Filesystem**
- Using shell commands
- File permissions
- Miscellaneous

Filesystem

- Filesystem is a tree
 - Unlike Windows, there are no “drives” - just one FS
- *Paths* are a sequence of directories to find a particular file, separated by '/'
- One directory is your *working directory* – if your path doesn't start with '/', the path is relative to this directory
 - Absolute path - /course/cs015
 - Relative path – mywork/project

Filesystem

- Change working directory with 'cd' command
- See current working directory with 'pwd'
 - Also usually displayed in your prompt
- Special paths
 - '.' = current directory
 - '..' = one level up from current directory
- Example: current directory is /course/cs017/bin
 - Where will you be if you cd ../../handin../../../../.. ?
 - Answer: /course

Filesystem: important places

- Home directory - `/home/user/` == `~user/` (`~` for you)
 - Files/folders starting with `'.'` are “hidden” (not just in home)
 - Most of your preferences are stored in these “dotfiles” in your home directory
 - These files include `.vimrc`, `.emacs`, `.alias`, `.environment`, `.bashrc`
- Course directory `/course/csXXX/`
 - different for each course, but usually includes certain common folders such as `asgn`, `handin`, `web`, etc.
 - **IMPORTANT:** Don't confuse `~/course/csXXX` with `/course/csXXX`! The first is your work for that course, the other is material for the course.

Filesystem: snapshots

- 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 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)

Summary

- Why is tech stuff important?
- Filesystem
- **Using shell commands**
- File permissions
- Miscellaneous

Using shell commands

- Usual syntax: *program options arguments*
 - e.g., `grep -i -r "lambda" /course/cs017`
- To find out about syntax and various options, use *man commandname*
 - e.g., `man grep`
 - This is your friend! Use *frequently!*

Some useful shell commands

- `man(1)` – manual pages
- `cd` – change directory
- `ls(1)` – list files in a directory
- `pwd(1)` – print working directory
- `mv(1)` – move files
- `cp(1)` – copy files

Shell magic

- *, ? are wildcards which refer to any number of any character or 1 of any character, respectively
 - e.g., *.c means all files ending in “.c”
- Type “&” after a command name to run it in the background
 - allows you to run other commands, which is pretty darn useful (otherwise you'll have one terminal per command...)
 - e.g., xpdf foo.pdf &

Job control

- Press CTRL-C to kill a running process.
- Press CTRL-Z to suspend a running process.
- “jobs” shows you running background processes (jobs) (%X refers to job #X)
- Use “fg” and “bg” to move jobs to foreground and background, respectively
 - “bg” resumes a job you suspended, but in the background.
- Start a program in the foreground by accident?
Press CTRL-Z and type “bg” - very common!

Manipulating input and output

- Use | (pipe) to connect output of one command to input of another
 - *e.g.*, `find ~ -name "*.o" | grep kern`
(prints files ending in .o whose names contain kern)
- Use > or < to redirect input/output to/from files
 - *e.g.*, `ls -R / > newfile`
- Common uses of pipes:
 - *grep* for filter output – `acommand | grep "filtertext"`
 - *less* to scroll through output – `bigoutputcmd | less`

Tab completion

- 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.
- This makes things MUCH faster!
 - Note: cannot autocomplete .snapshots (always)

Summary

- Why is tech stuff important?
- Filesystem
- Using shell commands
- **File permissions**
- Miscellaneous

File permissions: intro

- Each file and directory has *permissions*
- 3 classes of people, 3 permissions each
 - owner, owning group, everybody else
 - read, write, execute
 - always described in that order
- “ls -l” shows permissions as:
 - `rw-rw-rwx` (everybody can r/w/x)
 - `rw-rw-----` (owner and group can r/w)
 - `rw-----` (owner can r/w)

File permissions: meaning

- 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

File permissions: changing

- `chmod(1)` changes file permissions
- `chmod [args] newmode files...`
 - most useful arg is “-R” = recursive
- `newmode` can be numeric or *symbolic*

File permissions: numeric

- Sometimes, permissions represented as 3 octal (base-8) digits
 - 1st digit = user, 2nd digit = group
 - 3rd digit = everyone else
- Each digit is sum of
 - 4 = read, 2 = write, 1 = execute
- *e.g.*, 775 (rwxrwxr-x)
- Don't worry too much about this (unless you're an HTA)

File permissions: examples

- `chmod ug+rw file1 file2 file3`
 - Allow user and group to read and write (doesn't affect execute bit)
- `chmod g=u file1 file2`
 - Make group perms same as user perms
- `chmod o= file1 file2`
 - Set permissions to "" (removes all privileges) for "other people"
- `chmod a+X file1` – gives execute if others have execute

File permissions: symbolic perms

- symbolic is 1 or more of ugoa, plus 1 of +-=, plus 1 or more of rwx (and a few others)
 - ugoa = user, group, other, all
 - + adds, - removes, = sets

File permissions: usage

- In course directories, files usually have one of two permissions:
 - Public files (stencil code, documentation, etc) should have permissions 775 (rwxrwxr-x)
 - Private files (solutions, exams, grades, etc) should have permissions 770 (rwxrwx---)
 - Most files should have group csXXXta
 - Exceptions: head TA only stuff
- A student's work should not be world-readable
 - Make sure ~/course does not have world-execute

File permissions: defaults

- Files you create are owned by you and belong to the group 'ugrad' by default or, if sgid bit is set on the parent directory, the group of the parent directory
- Course directories should have this bit set
- `chmod +s <directory>`
- `man chmod` for more details on the sgid bit

File permissions: defaults (2)

- Permissions on files you create are everything MINUS your 'umask'
- Default is 022 = `rw-r--r--`
 - This is a problem in course directories:
 - Groups can't write your files – fellow TAs won't like that!
 - Everyone can read your files – not so good for solutions!
 - To fix, set your umask to 007 = `rw-rw-r--`
- Use *umask* command to see/change umask
 - `man umask(1)` – that is, run 'man 1 umask'

File permissions: Windows

- Very simple: don't ever use them.
- Changing the Windows permissions of a file breaks the Linux ones (which most people use most of the time).
- Always log into a Linux system (or use a Cygwin shell from a Windows system) and use `chmod(1)` to change permissions.

Summary

- Why is tech stuff important?
- Filesystem
- Using shell commands
- File permissions
- **Miscellaneous**

Email

- Should check your email daily
- Forward email using
mailconfig -a *you@whatever.com*
- To read mail, use pine from command line, or thunderbird, etc.
- Never send passwords via email

Groups

- Affect important environment variables, handin scripts, etc.
- Control permissions.
- So make sure you have csXXXta group!
- Students for your course should have csXXXstudent group.

Remote access

- ssh: allows you to get a shell from a computer outside the department
- Can also access email and shells remotely, using VPN, ssh, or web, etc.
- See CS Department Documentation
<http://www.cs.brown.edu/facilities/system>
- Intro course TAs – try to familiarize yourself with setting up ssh – you tend to get lots of questions about this.

Department Utilities

- *finger* – see where someone is logged in
- *floor* – display graphical picture of where people are logged in.
- *sunlist* – shows a histogram of who's in the sunlab by course. Useful to see how crowded your hours will be
- *zwrite* – use to send another user a message. Widely used.

Miscellaneous

- Test accounts: csXXX000. Password is in /course/csXXX/admin/csXXX.kerbpas
– Should use to test permissions
- /pro/tasupport has useful programs
– Sort of...
- Learn LaTeX, vim/emacs, advanced commands with Sunlab minicourses

Resources

- Sunlab consultants (sit at 9a)
- mta@cs.brown.edu (Meta-TAs): email this first with random technical questions (related to TA'ing)
- problem@cs.brown.edu: if you have problems with the departmental systems
- <http://www.cs.brown.edu/facilities/system>
Great documentation on everything
- <http://wiki.cs.brown.edu/>
Internal Wiki (lots of useful info)