Getting your department account

The instructions are at

Creating a CS account

Getting help

Vijay Adusumalli will be in the CS majors lab in the basement of the Love Building from 11:00am until 1:00pm everyday from Wednesday, January 23, until Monday, January 28, (except for Sunday, January 27) to help you if you cannot get set up.

Getting connected: "logging in"

Fundamental to the idea of Unix is the idea of "logging in". This means

1. Presenting some sort of credentials (maybe in an automated fashion)
2. Having a "shell" process created for you
3. Using the "shell" process

Unconnecting: logging out

When you want to leave the system, ("log out"), you have many choices:

- "exit" should work for any shell, and is usually your best choice
- CTRL-D is very likely to work also
- Most login shells also will accept "logout"
- "bye" is sometimes accepted, but don't count on it

Who else is on the system?

- `who` command
- `w` command
- `users` command

Communicating with other users:

- traditional `Mail`
- `write`
- `wall`
- `talk`
- `irc`

Finding out more with man

- man CMD
- man FILE
- man -k KEYWORD

ls, the root, and your home directory

`ls` is a useful command; like `dir` in the Windows world, it shows you what is in a directory.

The most useful options are

- `-l` : list long information
- `-t` : order via modified time rather than by name
- `-a` : show "dot" (hidden) files
- `-R` : recursive listing
- `-r` : reverse the order
- `-h` : "human" readable

The "root"

On the "root", you expect to find at least these directories:

- /bin
- /etc
- /lib
- /usr
- /var
- /tmp

In addition, you are likely to also see at least these:

- /boot
- /dev
- /home

We do have standards:

- Unix standard
- Linux standard
- Also, try `man hier`.

Your home directory

In addition to any files and directories that you might create in your home directory, there are often a number of other files and directories that might be there.

Generally these are what we call "dot" files: literally, the first character in the file/directory name is a period.

The program `ls` ignores such files unless it has been given the `-a` (all) option.

In addition, what we call "globbing" ignores "dot" files unless you explicitly include such a dot.

So what is "globbing"? The idea behind globbing is to treat the characters "*", "?", "[]", "{}", and "~" as "metacharacters".

- * matches any string of characters
- ? matches any one character
- [] matches a range of characters
- {} matches specific characters
- ~ expands to either your home directory (as unadorned initial ~), or $HOME for another user if ~user

"dot" files/directories and why are they there!?

"Dot" files generally are used by programs such as shells and windowing environments which have a need to have and retain complex state, such as startup commands and current information.

Creating files and directories

the major ways:

- `touch`
- `mkdir`
- `cp / mv`
- redirecting output
- text editors
- programs

changing file permissions

To change permissions, we use the `chmod` command:

```
chmod u+x   # add execute permission for the user
chmod g+x   # add execute permission for group
chmod o+x   # add execute permission for other
chmod a+x   # add execute permission for all users

chmod a-x   # remove execute permission for all users
chmod g-r   # remove read permission for group
```

File permission defaults

Shells generally have an "built-in" command called `umask` which allows the user to specify the default permissions for newly created files.

It actually specifies the complement of the default read-write permissions for newly created files.

```
umask 077    # no one except the user can read or write new files
umask 000    # anyone can read or write all new files (dangerous!)
umask 022    # anyone can read but not write new files
umask 777    # no one (not even the owner) can read or write new files
```

Working with files

```
% cal 1776 > 1776.calendar.txt
% cal 1752 > 1752.calendar.txt
```

We can use many programs to examine this file: `cat`, `less`, and `od` are among the most popular, but also `head`, `tail`, and even `grep` are also quite useful.

Joining files with `cat`

You can use `cat` to join multiple files together:

```
cat 1776.calendar.txt 1752.calendar.txt > both.txt
cat 17{76,52}.calendar.txt > both.txt

# or with append
cat 1776.calendar.txt > both.txt
cat 1752.calendar.txt >> both.txt
```

Copying with cp

```
cp both.txt copy.txt   # copies, but uses new date and
                       # umask default permissions for new file
cp -a both.txt         # creates an exact copy
cp -r .mozilla .mozilla-backup   # creates a new copy of a dir
                                 # but times are new and permissions
                                 # are umask-controlled
cp -a .mozilla .mozilla-exact    # creates an exact copy of a dir
```

Moving files

Moving files with `mv` is usually far faster than copying the same files with `cp`. If a file is in the same filesystem (and if it's in your home directory it very likely is), then `mv` just changes some directory information rather than doing anything with the contents. `cp` of course must work with the contents (unless you are using a very sophisticated filesystem that understands on-the-fly deduplication.)

```
mv both.txt old.txt
mv .mozilla .mozilla-2013-01-22   # notice that no option is necessary
                                  # to specify directory operations
```

Removing files

Removing files is easy — maybe too easy!

```
rm old.txt
rm -rf .mozilla         # remove a directory without complaining
rm -rf /                # remove everything in the system (bad idea!)
rmdir .somedir/         # doesn't work unless .somedir/ is empty
```