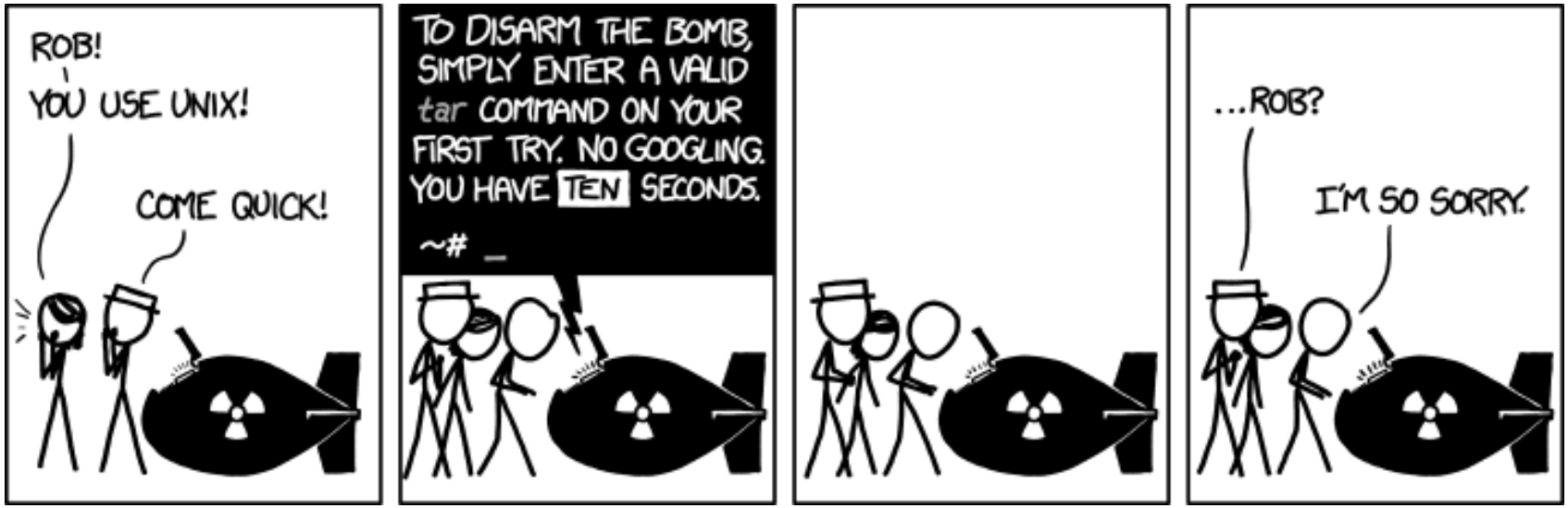
Lecture 5

Additional useful commands

COP 3353 Introduction to UNIX



diff

- diff compares two text files (can also be used on directories) and prints the **lines** for which the files differ. The format is as follows:

  diff [options] <original file> <newfile>

- Some options
  - [-b] Treats groups of spaces as one
  - [-i] Ignores case
  - [-r] Includes directories in comparison
  - [-w] Ignores all spaces and tabs
  - <*original file*> - Specifies one file to compare
  - <*newfile*> - Specifies other file to compare

  Example:

```
diff -w testprog1.c testprog2.c
```

Results of diff

- Above the results, you'll see some letters and numbers.
- In this traditional output format, **a** stands for *added*, **d** for *deleted* and **c** for *changed*.
- Line numbers of the original file appear before a/d/c and those of the modified file appear after.
- Angle brackets appear at the beginning of lines that are added, deleted or changed. Addition lines are added to the original file to appear in the new file.
- Deletion lines are deleted from the original file to be missing in the new file.
- By default, lines common to both files are not shown. Lines that have moved will show up as added on their new location and as deleted on their

*

grep

- *grep* is a very useful utility that searches files for a particular *pattern*. The pattern can be a word, a string enclosed in single quotes, or a regular expression.
- usage
  - `grep int *.c` (find all occurences of the pattern 'int' in all files with a .c extenstion)
  - `grep 'main()' testprog1.c` (enclosing the pattern in quotes is useful when using special characters)
  - `grep 'm.*n' myfile` (the . matches a single character, the .* matches any number of characters; this finds anything starting with an *m* and ending with an *n*)
- The way that regular expressions can be described is somewhat complex in grep: see the following tutorial for more help:

grep has many options; a few are noted below

-i ignore case -n display the line numbers
-l display only names of files and not actual lines

Some grep patterns

- Bracketed expressions

  [1357] matches 1 or 3 or 5 or 7

  [^1357] with a beginning ^ matches *not* (1,3,5,7)

- Range expressions

  [b-g] matches b, c, d, e, f, g

- Named classes of expressions

  [:digit:], [:alnum:]

- Special symbols

  ? The preceding item is optional and matched at most once.

  * The preceding item will be matched zero or more times.

  + The preceding item will be matched one or more times.

  . This matches any single character

*

Some grep patterns continued

- Matching at the beginning and end

  ^ matches the beginning of the line, thus ^#include would math any lines with a #include at the beginning of the line.

  $ matches the end of line

  \< matches the beginning of a word

  \> matches the end of a word

- | the *or* operator

  grep dog | cat

- *egrep and fgrep are extended versions*

The tar command - IMPORTANT!

- Tar is a utility for creating and extracting archives. It is very useful for archiving files on disk, sending a set of files over the network, and for compactly making backups
- General form:

  tar *options filenames*

- Commonly used Options

  -c insert files into a tar file

  -f use the name of the tar file that is specified

  -v output the name of each file as it is inserted into or extracted from a tar file

  -x extract the files from a tar file

  -t list contents of an archive

  -z will gzip / gunzip if necessary

Creating an Archive with Tar

- The typical command used to create an archive from a set of files is illustrated below. Not that each specified filename can also be a directory. Tar will insert all files in that directory and any subdirectories. The *f* flag is used to create an archive file with a specific name (usually named with an extension of .tar).
- Examples

```
tar -cvf proj.tar proj
```

  - If proj is a directory this will insert the directory and all files and subdirectories (recursively) with the name of the archive being proj.tar Note that if proj.tar already existed it will simply be overwritten; previous information will be lost.

```
tar -cvf prog.tar *.c *.h
```

  - All files ending in .c or .h will be archived in prog.tar

Extracting files from a Tar Archive

- The typical tar command used to extract the files from a tar archive is illustrated below. The extracted files have the same name, permissions, and directory structure as the original files. If they are opened by another user (archive sent by email) the user id becomes that of the user opening the tar archive.
- Examples

```
tar -xvf proj.tar
```

  - Extract all files from proj.tar into the current directory. Note that proj.tar remains in the current directory.

```
tar -xvzf proj.tar.gz
```

  - Extract files but also unzip files in the process

cmp and gzip

- cmp
  - Compares two files byte by byte and tells you where they differ. Generally used for binary and executable files.
  - usage: `cmp myfile1.o myfile2.o`
- gzip / gunzip
  - this is one of the compression utilities that reduces the size of a file to take up less space on your drive. It should be used with some care. There are a lot of options available.
  - Examples to compress and restore a file called bigfile:
    - `gzip bigfile` (compresses file, flag -v can be used for verbose mode to give information; note: original file is gone! The compressed file has an extension of .gz
    - `gzip -d bigfile.gz` (this restores a .gz file)
    - `gunzip bigfle.gz` (same as line above)
  - When you experiment with this use copies of files rather than the originals until you are comfortable with how gzip works!