SOME LINUX TOOLS CS 23200

Big Picture

- ✓ Developing programs on *nix computers
- ✓ C Language
 - ▼Familiar aspects of C (variables, operators, basic I/O, control flow, functions)
 - ✓ Pointers
 - ✓ Structures and related constructs
 - ✓ File operations
 - Multi-file programs
 - ✓ Standard library functions
- □ *nix tools

 - Some utilities
 - Shell scripting

What to Expect

A reminder, from the first day of class

- □ The course title: "Introduction to C and UNIX"
 - Can lead to misplaced student expectations
 - Introduction
 - Introduction != easy
 - Introduction == no prior knowledge of C or UNIX is required
 - □C and UNIX
 - Focus is on C and developing C programs on *nix machines
 - Roughly (more than) two-thirds of the course
 - Latter (less than) third of the course talks about some *nix tools, shell scripting
 - Not covering *nix system administration

Outline (Basic Linux Tools)

- Viewing files
- Comparing files
- □ Filtering lines
- Searching multiple files (same directory)
- □ Finding files in different directories
- Searching multiple files (different directories)
- □ Search and replace over multiple files

Viewing Files

Outline (Basic Linux Tools)

- Viewing files
- □ Comparing files
- □ Filtering lines
- □ Searching multiple files (same directory)
- □ Finding files in different directories
- Searching multiple files (different directories)
- □ Search and replace over multiple files

An Example

- □ Suppose we have a bunch of .c files, each with a backup copy
 - Example: program.c and program.c.bak

If no files given, will use standard input instead

- □ Segfault with the current version
- □ Not with the backup version
- Want to see which files are different from their backups

Comparing Files

- □ diff file1 file2
 - Compares files line by line
 - Output
 - By default, prints differences in the files
 - Console-based diff is fine for quick checks
 - Use graphical diff program for any substantial checks (much easier to read)
 - Windows: WinMerge / Beyond Compare
 - Switches:
 - -q: output only if files differ
 - -i: ignore case
 - --ignore-space-change: ignore changes in the amount of whitespace

Try it out on /.../linux_tools/comparing/objPool.c.bak and /.../linux_tools/comparing/objPool.c

An Example

 Find the differences between objPool.c and objPool.c.bak

```
diff --ignore-space-change objPool.c.bak
  objPool.c
```

- □ How to handle too much output?
 - □ Pipe it to less

```
diff --ignore-space-change objPool.c.bak
  objPool.c | less
```

Outline (Basic Linux Tools)

- Viewing files
- Comparing files
- □ Filtering lines
- □ Searching multiple files (same directory)
- □ Finding files in different directories
- □ Searching multiple files (different directories)
- □ Search and replace over multiple files

Filtering Lines

- When you only want to display lines that contain a pattern
- □ Example: lines containing malloc
 - □ Use grep "malloc" [file]
- □ File to search goes at the end of command
- □ No file given: use standard input
 - Useful for filtering output of other commands...

Filtering Other Commands' Output

□ Suppose we want to filter the output of "Is -I" to only list lines with objPool

```
ls -1 | grep "objPool"
```

- □ Output of Is -I is piped to input of grep
- □ grep filters its input (because there is no file name)
- Suppose we want to list only directories in long format

```
ls -altr | grep "^d"
```

■ The ^ is a special character in grep for the beginning of the line

grep Options

- Context: lines before or after the line that matches the grep pattern
 - Can be useful when filtering source code files
 - -A NUM: print NUM lines of context after the matching line
 - B NUM: print NUM lines of context before the matching line
 - -n: print line numbers
- Try grep-ing for malloc in /.../linux_tools/comparing/objPool.c
- □ Use different combinations of the context switches
- □ Which most clearly shows the malloc usage?

Another Example

- □ Suppose you have a bunch of .c and .h files in a directory
- □ You want to find which ones...
 - □ Call malloc
 - Use a struct Node
 - □ Don't call malloc
- □ Can use grep for each of these things

Outline (Basic Linux Tools)

- Viewing files
- Comparing files
- □ Filtering lines
- □ Searching multiple files (same directory)
- □ Finding files in different directories
- □ Searching multiple files (different directories)
- □ Search and replace over multiple files

grep Over Multiple Files

- grep can search multiple files
 - grep expression file1 file2 ...
- Use wildcards
 - grep "malloc" *.c *.h
 - Shell expands *.c to a list of all files in the current directory that end with .c
- □ grep switches:
 - -L: list files that do not match
 - -I: list files that match
 - -c : print a count of matching lines
- □ Example: count the uses of malloc in .c files
 - grep -c "malloc" *.c
- □ Example: count the uses of struct Node
 - grep -c "struct Node" *.c *.h

Other Useful grep Switches

- -I: list files that match
- -i: ignore case
- □ -H: print the file name with each match
- -h: do not print the file name
- -v : invert the search
 - This will select non-matching lines
- □ What does the following do?

grep -i -l -v "error" *.c

- DOES NOT list the .c files that do not contain the word "error" (ignoring case)
- Instead, lists the .c files that have any line that does not contain "error" (ignoring case)

Outline

- Viewing files
- Comparing files
- □ Filtering lines
- □ Searching multiple files (same directory)
- □ Finding files in different directories
- Searching multiple files (different directories)
- □ Search and replace over multiple files

The find Command

find project3/backupFiles -name "*.c"

- Finds all files in project3/backupFiles and its subdirectories that match "*.c"
- Prints one file per line, including relative path from current directory
- □ General form: find dirToSearch criteria

Multiple Criteria

special shell character

- Can search for files that meet multiple conditions
- .c files or .h files modified within the last week

```
find . \( -name "*.c" -o -name "*.h" \) -a -ctime -7

( name matches*.c OR name matches*.h ) AND change time was 7 or fewer days ago

escaped by \ because ( is a
```

Multiple Criteria

□ Any files not ending in ~ accessed more than two weeks ago

□ Files more than 100 MB that were modified within the last 3 hours

```
find . -size +100M -cmin -180
```

What does this do?

```
find . -size -10k \! \( -name "*.txt" -o -name "*~" \)
```

An Aside: Recalling Old Commands

- Suppose you typed a really long command a while ago
- □ You don't want to retype it
- You don't want to hit UP fifty times to try to find it
- history
 - Prints the shell's log of commands you entered

Exercises

- □ Within one of your project directories:
 - Print out all the references to any structure in any .c or .h file
 - Do the same, but for a particular structure (i.e., struct TrieNode)
 - Do the same, but print out some context lines
- □ How many .c or .h files are in your home directory (or subdirectories)?
 - How many have been modified within a week?
 - How many are larger than 1KB?
 - How may have been modified within a week AND are larger than 1KB?

An Aside: Recalling Old Commands

- $\ \square$ history prints a number with each command
- □ Running 173 from the shell will repeat command number 73
- You could look through the history output manually to find the number of your command...
- $\hfill\square$... or you could use grep to filter the output
- □ How might you filter history if you are looking for an old "find" command?

```
history | grep "find"
```

If you know that the find command also referenced ctime, how can you modify the above command?

```
history | grep "find" | grep "ctime"
```

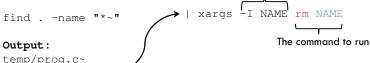
Outline

- Viewing files
- Comparing files
- □ Filtering lines
- □ Searching multiple files (same directory)
- □ Finding files in different directories
- □ Searching multiple files (different directories)
- □ Search and replace over multiple files

xargs

- □ find outputs a list of files
- □ We use xargs to do something with that list
 - Reads a list from standard input
 - Runs a command on each list item
 - Like a loop over the results of find

Essentially defines NAME as a variable that will take on every value in the list



temp/prog.h~

myFile.txt~

taskList.txt~

rm temp/prog.c~

rm temp/prog.h~

rm temp/prog.h~

rm myFile.txt~

rm temp/prog.t~

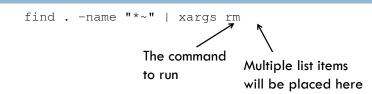
rm temp/prog.t~

rm myFile.txt~

Example

- □ How would we list all .c files in the current directory (or subdirectories) that contain a call to malloc?
 - What are the steps?
 - Get the list of the .c files
 - Loop over the files
 - Use grep to see if they contain malloc

Example



- □ Can eliminate -I VAR if...
 - □ The file name goes at the end of the command
 - And the command can handle multiple files as command-line arguments
 - e.g., grep "malloc" file1.c file2.c file3.c ...

```
find . -name "*.c" | xargs grep -l "malloc"
```

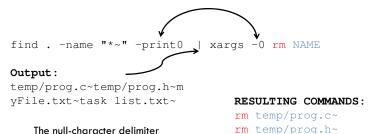
xargs

- Default delimiter for xargs is whitespace
 - Causes problems if file names contain spaces
- □ The -0 flag for xargs assumes null character is delimiter

rm myFile.txt~

rm "task list.txt~"

■ Use the -printO flag for find to get such a list



Example

```
find . -name "*~" -print0 | xargs -0 rm

The command to run

Multiple list items will be placed here
```

- □ Can eliminate -I VAR if...
 - □ The file name goes at the end of the command
 - And the command can handle multiple files as commandline arguments
 - e.g., grep "malloc" file1.c file2.c file3.c ...

```
find . -name "*.c" -print0 | xargs -0 grep -1 "malloc"
```

Outline (Basic Linux Tools)

- Viewing files
- Comparing files

does not show up on the

terminal, but it is there

nonetheless

- □ Filtering lines
- Searching multiple files (same directory)
- □ Finding files in different directories
- Searching multiple files (different directories)
- □ Search and replace over multiple files

Search and Replace

- □ Use a one-line perl script
- Replaces every occurrence of oldstr with newstr perl -p -i -e 's/oldstr/newstr/g' file1 file2 ...
- Combine this with find and xargs to get search and replace over multiple files in several directories
 - e.g., rename the function createNode to createLinkedListNode

Search and Replace Example

- □ Rename the function createNode to createLinkedListNode
- What are the steps and the tools for each step?
 - Get the list of files in which we want to do replacement: use find
 - Loop over the files: use xargs
 - Do the replacement in each file: use the perl script

```
find . \( -name "*.c" -o -name "*.h" \) -print0
| xargs -0 perl -p -i -e
's/createNode/createLinkedListNode/g'
```

Be Careful!

- □ Search and replace for "add" will also catch...
 - addNode
 - gladden
- □ To avoid these, need regular expressions

Summary

- Viewing files
 - cat, less
- Comparing files
 - diff
- Filtering lines
 - grep
- □ Searching multiple files (same directory)
 - grep
- □ Finding files in different directories
 - finc
- □ Searching multiple files (different directories)
 - □ find, xargs, grep
- □ Search and replace over multiple files
 - find, xargs, perl