UNIX Power Tools

Intermediate System Administration DeCal
Spring 2010
Lecture 5

Today

- Learn to use tools like cut, sed, sort, tr, and grep to do amazing text manipulation
- Learn how to use regular expressions
- Learn how to use xargs to get over the limits of command line substitution
- Learn to properly use find

sort(1)

- Does pretty much what it says: Takes input from stdin or a file and sorts it ascending alphanumerically by default
- Can sort by many different criteria or by columns or backwards

```
sort -k2 a.txt b.txt
ls | sort -r
```

- Often used in conjunction with uniq(1):

```
sort classes-taken.txt | uniq
```

- Uniq needs sorted input; use uniq -u for unsorted input

tr(1)

- Used to **TR**anslate characters or classes of characters in an input stream, or delete them. Does not work with strings!

```
tr 'a-z' 'A-Z' names.txt
echo 'Go Bears!' | tr a e
```

cut(1)

- Splits lines into fields with the delimiter of your choice

```
echo "a,b,c" | cut -d, -f1
```

(returns a)

```
echo "Jack eats pie" | cut -d ' ' -f3
```

(returns pie)

```
echo "Jack eats pie" | cut -d, -f1
```

(returns Jack eats pie, since there are no commas)

sed(1)

- **S**tream **ED**itor: takes input and spits it back out with certain modifications

```
sed 's/D/A+/g' < grades.txt
```

(Changes all D's to A+'s in grades.txt on all lines and spits it to stdout)

```
sed 's/John/Jeff' < roster.txt
```

(Changes the first instance of John on each line to Jeff)

```
sed 's/\([^ ]+\) your \([^ ]+\)/\2\1er/g' < insults.txt
```

(Changes eg. “fail your test” to testfailer in file insults.txt)

Regular Expressions

- Regular expressions can be used with grep, sed, pretty much any command line tool
- A superset of the wildcard system you learned before (*/?)
- We will go over a few examples briefly, but you will learn more doing the lab

Regular Expressions

- Match all lines that contain what or What

  `[wW]hat`

- Match all lines that start with x and end with a number or a lowercase letter followed by any character

  `^x.*[0-9a-z].$`

- Find all lines that have no whitespace

  `^\S+$`

Regular Expressions

- You can use these expressions in sed for substitution:

```
sed s/regex1/regex2
```

- You can use these expressions in egrep for matching:

```
egrep "regex1" < file
```

- This has been a **really brief** overview, but regexes are super powerful

xargs

- Trying “rm *” in a huge directory or “rm $(<deleteme.txt)” with a huge file will give “command list too long!”
- Instead:

```
xargs rm < deleteme.txt
find . | xargs rm -f
```

- If your files have names with spaces?

```
find . -print0 | xargs -0 rm -f
```

find power user

- The find command can do way more than just find all the files in a directory. It has predicates!

  ```
  find -iname "TeSt.TxT" -and -type f
  ```

  Finds **files** called test.txt with case insensitivity

  ```
  find -not -name "meh" -or -type -d
  ```

  Finds **directories** or anything not called “meh” (case sensitive)
- Consult the manpage for more predicate goodness.