Review: Unix Commands

- What are filters?
  - Why are they useful?
- What do we use to send the output from one command to the input of the other command?
- What command do we use to select different columns from a file?
- How can we merge several files in parallel?

Today

- Filters
- Commands in commands

```
echo -n a >&2 | echo b >&2
```


Danger: Deleting a Set of Files

- One solution:

```
find . -name "*~" -exec rm "{}" ";"
```

  - Seems to do forced `rm`, no interaction with user required
  - **LESSON**: Do `find` part first and verify want to do remove
- Alternative (not quite equivalent) solution:

```
yes | rm *~
```

Just in current directory

Example Execution

Answers y to each removal question

```
[sprenkle@hopper personal]$ yes | rm */*~
rm: remove regular file `craw-hobby/index.html~'? rm:
remove regular file `England2008/England2008.html~'? rm:
remove regular file `England2008/Page1.html~'? rm:
remove regular file `England2008/Page2.html~'? rm:
remove regular file `England2008/Page3.html~'? rm:
remove regular file `England2008/Page4.html~'? rm:
remove regular file `England2008/Page5.html~'? rm:
remove regular file `England2008/Page6.html~'? rm:
remove regular file `England2008/Page7.html~'?
[sprenkle@hopper personal]$
```

- Best practice: Do an `ls` using the regular expression to see what files you're going to delete, e.g., `ls */*~`
- Try the `rm` command, when it prompts you, say yes a few times.
- If it seems to be working, kill it and do the command with the `yes |`

Another Useful Shortcut

- On-the-fly modification of a previous command to create a new command
- the Bash shell uses the caret (^) character to perform substitutions:

```
[sprenkle@hopper day3]$ ls -l villains.txt
-rw-r--r-- 1 sprenkle cs397 0 2017-01-19 13:29 villains.txt
[sprenkle@hopper day3]$ ^villains^heroes
ls -l heroes.txt
-rw-r--r-- 1 sprenkle cs397 0 2017-01-19 13:29 heroes.txt
```

New command

Execute Multiple Commands: ;

- Can execute multiple commands on one line
- Example:

```
[sprenkle@python cs397]$ mkdir assigns; cd assigns
[sprenkle@python assigns]$
```


Executing multiple commands

- Example of executing more than one command on the command-line:

```
sleep 5m; mplayer foo.mp3
```

- In my research:

```
start_server
sleep 2m; execute_testcases
```


Using commands in commands: ``

- Syntax: `` `command` ``
  - Backtick: on same key as ~
- Means "execute this command first and use its output in this command"
- Example: I want to check the permissions on all my shell scripts (which end in .sh)
  - Verify that they're executable by me and no one else
  - `` ls -l `find . -name "*.sh"` ``
- Note: these commands will take longer to execute because get answer for "inner" command first

Try These Examples

- `echo "You are in `pwd`"`
- `expr `date +%S` % 10`
  - What does this do?
  - (Break into pieces and figure out how it works)

Using commands in commands

- Examples from my scripts

```
java -cp mail.jar:email.jar grading.Email
  $password $email "$subject" "`cat $filename`"
```

```
jarfiles=`ls $TURNINDIR/$STUDENT/$LAB/*.jar`
for jarfile in $jarfiles
do
    echo "Jar file: $jarfile"
    numJavaFiles=`jar tf $jarfile | grep -c ".java"`
    if [ $numJavaFiles = 0 ]; then
       echo "No Java Files submitted by $STUDENT"
    fi
done
```

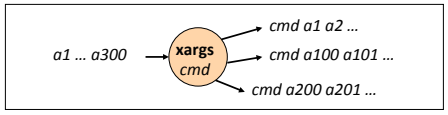

What happens when you run…

- `wc `ls –R ~`

Not the best way to run this command, but just motivating a problem…

xargs

- Unix limits the size of arguments and environment that can be passed down to child
- What happens when we have a list of 10,000 files to send to a command?
- `xargs` solves this problem
  - Reads arguments as standard input
  - Sends them to commands that take file lists
  - May invoke program several times depending on size of arguments



find utility and xargs

- `find . -type f -print | xargs wc -l`
  - `-type f` for files
  - `-print` to print them out
  - **xargs** invokes **wc** 1 or more times
- `wc -l a b c d e f g`
  `wc -l h i j k l m n o`
  …
- Compare to: `find . -type f –exec wc -l {} \;`

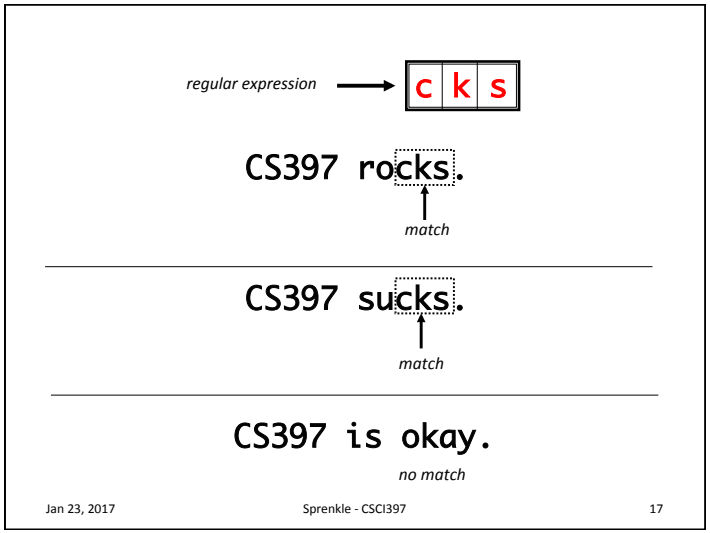
REGULAR EXPRESSIONS

What Is a Regular Expression?

- A regular expression (regex) describes a set of possible input strings
- Regular expressions descend from a fundamental concept in Computer Science called *finite automata theory*
- Regular expressions are endemic to UNIX
  - vi, ed, sed, and emacs
  - awk, tcl, perl and Python
  - grep, egrep, fgrep
  - compilers

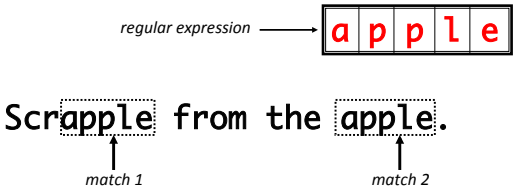
Regular Expressions

- The simplest regular expressions are a string of literal characters to match
- The string ***matches*** the regular expression if it contains the substring



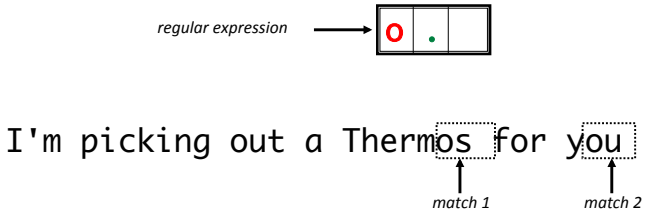
Regular Expressions

- A regular expression can match a string in more than one place



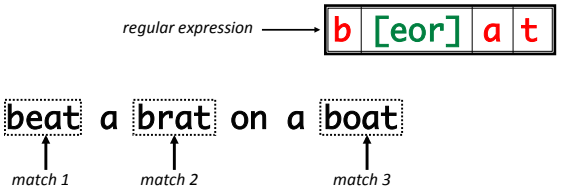
Regular Expressions

- The . regular expression can be used to match any character.



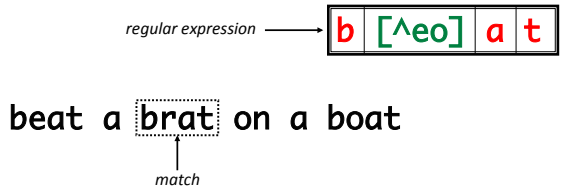
Character Classes

- Character classes [] can be used to match any specific set of characters.



Negated Character Classes

- Character classes can be negated with the `[^]` syntax.



More About Character Classes

- `[aeiou]` will match any of the characters `a`, `e`, `i`, `o`, or `u`
- `[kK]orn` will match `korn` or `Korn`
- Ranges can be specified in character classes
  - `[1-9]` is the same as `[123456789]`
  - `[abcde]` is equivalent to `[a-e]`
  - You can also combine multiple ranges
    - `[abcde123456789]` is equivalent to `[a-e1-9]`
  - Note that the `-` character has a special meaning in a character class ***but only*** if it is used within a range, `[-123]` would match the characters `-`, `1`, `2`, or `3`

Named Character Classes

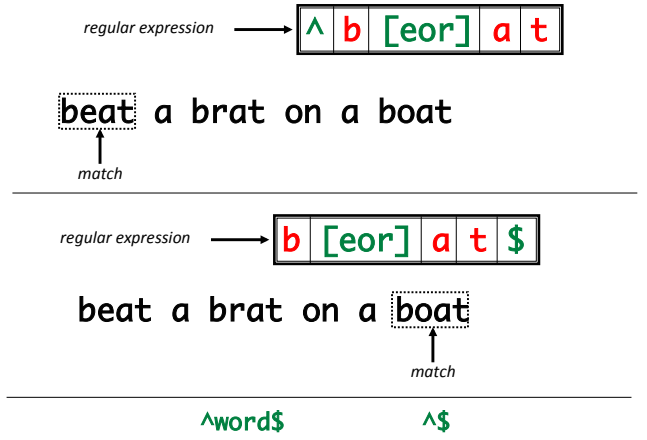
- Commonly used character classes can be referred to by name (*alpha*, *lower*, *upper*, *alnum*, *digit*, *punct*, *cntrl*)
- Syntax `[:name:]`
  - `[a-zA-Z]` → `[[:alpha:]]`
  - `[a-zA-Z0-9]` → `[[:alnum:]]`
  - `[45a-z]` → `[45[:lower:]]`
- Important for portability across languages

Regular Expressions

- Most of what we went through can be used in commands, like `ls`, `cp`, `rm` (be careful!), …
  - I test the `rm` command with `ls` first
- Practice
  - List the files that begin with D
  - List that files that end in .java
  - List the files that begin with D or d
  - List the files that begin with a, b, c, or d and end in .py

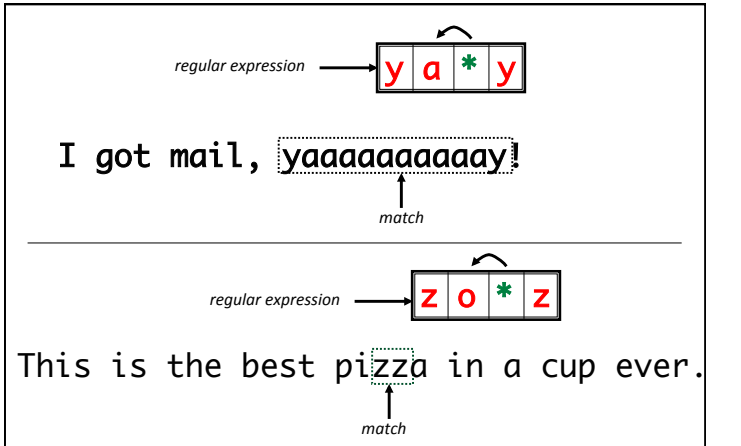
Anchors

- Anchors are used to match at the beginning or end of a line (or both)
- ^ means beginning of the line
- $ means end of the line



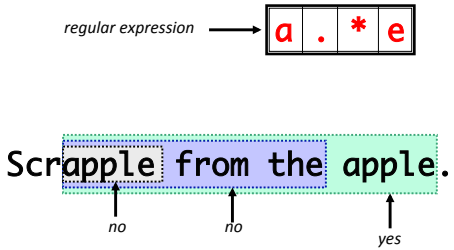
Repetition

- The * is used to define **zero or more** occurrences of the *single* regular expression preceding it.



Match length

- A match will be the longest string that satisfies the regular expression.



Looking Ahead

- More regular expressions