

---

# CSE 374

# Programming Concepts & Tools

Hal Perkins

Fall 2015

Lecture 5 – Regular Expressions, grep, Other Utilities

---

# Where we are

---

- Done learning about the shell and it's bizarre "programming language" (but pick up more on hw3)
  - Today: Specifying string patterns for many utilities, particularly grep and sed (also needed for hw3)
  - Next: sed
- 
- And then: a real programming language – C

# Globbing vs Regular Expressions

---

- “Globbing” refers to shell filename expansion
- “Regular expressions” are a different but overlapping set of rules for specifying patterns to programs like grep. (Sometimes called “pattern matching”)
- More distinctions:
  - Regular expressions as in CS/mathematics
  - “Regular expressions” in grep
  - “Extended regular expressions” in egrep
    - Same as grep –E
  - Other variations in other programs...

# Real Regular Expressions

---

- Some of the crispest, elegant, most useful CS theory out there. What computer scientists know and ill-educated hackers don't (to their detriment).
- A regular expression  $p$  may “match” a string  $s$ .
- If  $p =$ 
  - $a, b, \dots$  matches the single character (basic reg. exp.)
  - $p_1 p_2, \dots$ , if we can write  $s$  as  $s_1 s_2$ , where  $p_1$  matches  $s_1$ ,  $p_2$  matches  $s_2$ .
  - $p_1 | p_2, \dots$  if  $p_1$  matches  $s$  or  $p_2$  matches  $s$ 
    - (in egrep, for grep use `\|`)
  - $p^*$ , if there is an  $i \geq 0$  such that  $p \dots p$  ( $i$  times) matches  $s$ .
    - (for  $i = 0$ , matches the zero-character string  $\epsilon$ )

# Conveniences

---

- Most regular expressions allow various abbreviations for convenience, but these do not make the language any more powerful
  - $p^+$  is  $pp^*$
  - $p?$  is  $(\epsilon \mid p)$
  - $[zd-h]$  is  $z \mid d \mid e \mid f \mid g \mid h$
  - $[\wedge a-z]$  and  $.$  are more complex, but just technical conveniences (entire character set except for those listed, or a single character  $.$ )
  - $p\{n\}$  is  $p \dots p$  ( $p$  repeated  $n$  times)
  - $p\{n,\}$  is  $p \dots pp^*$  ( $p$  repeated  $n$  or more times)
  - $p\{n,m\}$  is  $p$  repeated  $n$  through  $m$  times

# grep – beginning and end of lines

---

- By default, grep matches each line against  $.^p.^*$
- You can anchor the pattern with  $^$  (beginning) and/or  $\$$  (end) or both (match whole line exactly)
- These are still “real” regular expressions

## \* is greedy

---

- For example, find sections in an xml file:  
`egrep '<foo>.*</foo>' stuff.xml`
  - The .\* matches as much as possible, even over an intermediate '</foo>'
  - Use [^chars] or other regular expressions to anchor the search so it matches less
- But that does not mean that .\*p.\* will match any string
  - still need to match p.

# Gotchas

---

- Modern (i.e., gnu) versions of grep and egrep use the same regular expression engine for matching, but the input syntax is different for historical reasons
  - For instance, \{ for grep vs { for egrep
  - See grep manual sec. 3.6
- Must quote patterns so the shell does not muck with them – and use single quotes if they contain \$ (why?)
- Must escape special characters with \ if you need them literally: \. and . are very different
  - But inside [ ] many more characters are treated literally, needing less quoting (\ becomes a literal!)

# Previous matches – back references

---

- Up to 9 times in a pattern, you can group with  $(p)$  and refer to the matched text later!
  - (Need backslashes in sed.)
- You can refer to the text (most recently) matched by the  $n^{\text{th}}$  group with  $\backslash n$ .
- Simple example: double-words `^\(\([a-zA-Z]*\)\)\1$`
- You cannot do this with actual regular expressions; the program must keep the previous strings.
  - Especially useful with sed because of substitutions.

# Other utilities

---

- Some very useful programs you can learn on your own:
  - find (search for files, e.g., `find /usr -name words`)
  - diff (compare two files' contents; output is easy for humans and programs to read (see patch))
- Also:
  - For many programs the `-r` flag makes them recursive (apply to all files, subdirectories, subsubdirectories, ...).
  - So “delete everything on the computer” is `cd /; rm -rf *` (be careful!)

WHENEVER I LEARN A  
NEW SKILL I CONCOCT  
ELABORATE FANTASY  
SCENARIOS WHERE IT  
LETS ME SAVE THE DAY.

OH NO! THE KILLER  
MUST HAVE FOLLOWED  
HER ON VACATION!



BUT TO FIND THEM WE'D HAVE TO SEARCH  
THROUGH 200 MB OF EMAILS LOOKING FOR  
SOMETHING FORMATTED LIKE AN ADDRESS!



IT'S HOPELESS!

EVERYBODY STAND BACK.



I KNOW REGULAR  
EXPRESSIONS.

