sed and awk

Henning Schulzrinne
Dept. of Computer Science
Columbia University

sed: string replacement

- Line-oriented tool for pattern matching and replacement (stream editor)
- Not really a programming language (cf. awk)
- E.g., apply same change to lots of source files
- Filter, i.e., does not modify input file

sed description

- pattern a text → add to output
- address s /regex/replacement/
- address d → delete line
- delete lines 1-10: `sed -e '1,10d‘`
- delete comments: `sed -e '/^#/d‘`
- print only matching: `sed -n -e '/regexp/p`
- convert Unix to DOS: `sed -e 's/$/\r/' myunix.txt > mydos.txt`

awk

- Special-purpose language for line-oriented pattern processing
- pattern {action}
- action =
  - if (conditional) *statement* else *statement*
  - while (conditional) statement
  - break
  - continue
  - variable=expression
  - print expression-list

awk

- Patterns = boolean combinations of regular expressions and relational expressions
- awk –f *program* < *input* > *output*
- Also delimiter-separated fields:
  `BEGIN {FS=`*c*`}`
- Examples:
  - Print lines longer than 72 characters:
    length > 72
  - print first two fields in opposite order
    { print $2,$1 }

awk examples

- Add up first column, print sum and average
  {s += $1 }
  END {print “sum is”, s, “average is”, s/NR}
- Print all lines between start/stop words:
  /start/,/stop/
- Print all lines whose first field differs from previous one:
  $1 != prev {print; prev = $1}

awk applications

- Unix file processing, e.g., as part of a pipe
- avoid read loop in shells, Perl
- more intuitive syntax than shell scripts
- best limited to line-at-a-time processing