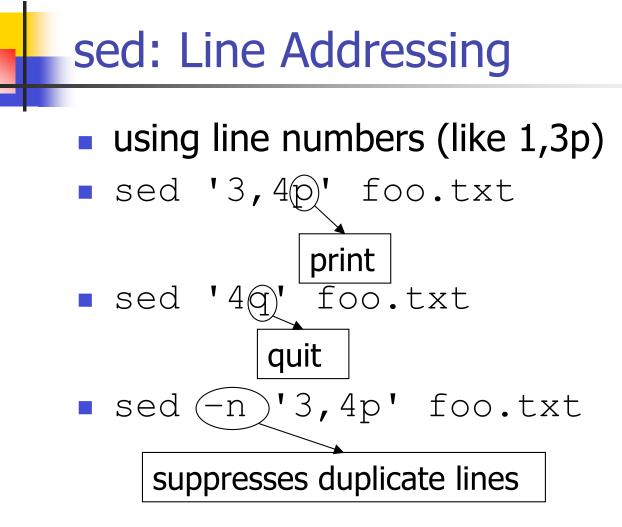


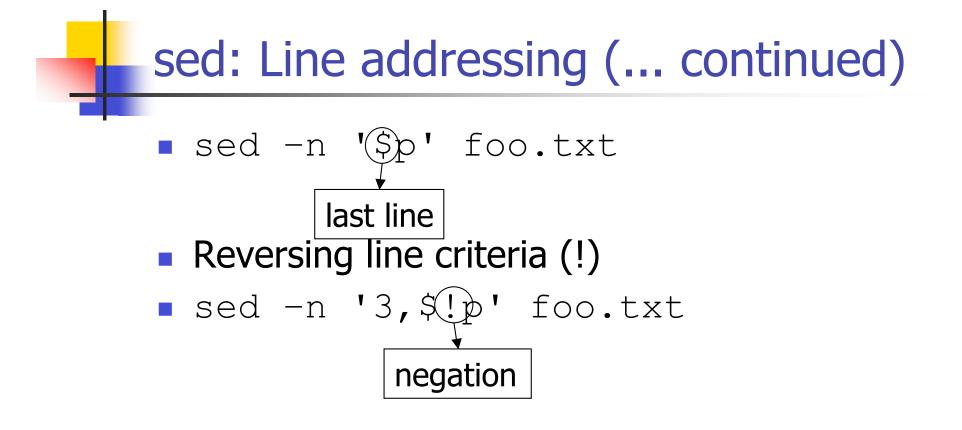
CS 2204

*Notes adapted by Doug Bowman from notes by Mir Farooq Ali and other members of the VT CS faculty

sed

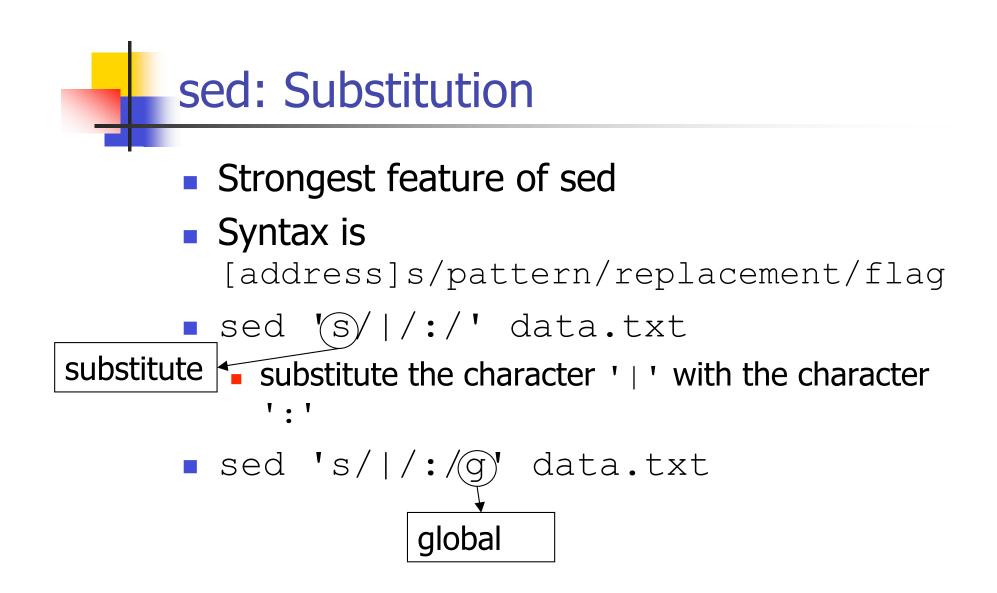
- Stream editor
- Originally derived from ed line editor
- Used primarily for non interactive operations
 - operates on data streams, hence its name
- Usage:
 - sed options 'address action' file(s)
 - Example: sed '1\$s/^bold/BOLD/g' foo





sed: Context Addressing

- Use patterns/regular expressions rather than explicitly specifying line numbers
- sed -n '/^From: /p' \$HOME/mbox
 - retrieve all the sender lines from the mailbox file
- ls -l | sed -n '/^....w/p'



(C) 2005 Doug Bowman, Virginia Tech CS Dept.

sed: Using files

- Tedious to type in commands at the prompt, especially if commands are repetitive
- Can put commands in a file and sed can use them

sed example script

/^[A-Z]/i\

NEW SENTENCE

/[tT]rash/d

s/GWB/George W. Bush/

(C) 2005 Doug Bowman, Virginia Tech CS Dept.

awk

- Powerful pattern scanning and processing language
- Names after its creators Aho, Weinberger and Kernighan
- Most commands operate on entire line
- awk operates on fields within each line
- Usage:
 - awk options [scriptfile] file(s)
 - Example: awk -f awk.script foo.txt

awk: Processing model

```
BEGIN {commands executed before any
input is read}
{
Main input loop for each line of
input
}
END {commands executed after all
input is read}
```

awk: First example

Begin Processing
BEGIN {print "Print Totals"}
Body Processing
{total = \$1 + \$2 + \$3}
{print \$1 " + " \$2 " + " \$3 " =
 "total}
End Processing
END {print "End Totals"}

(C) 2005 Doug Bowman, Virginia Tech CS Dept.

Input and output files

Input

- 22 78 44
- 66 31 70
- 52 30 44
- 88 31 66

Output

Print Totals

- 22 + 78 + 44 = 144
- 66 + 31 + 70 = 167
- 52 + 30 + 44 = 126
- 88 + 31 + 66 = 185

End Totals

awk: command line processing

awk '\$2 == "computers" {print}' sales.dat

- Input
- 1 clothing 3141
- 1 computers 9161
- 1 textbooks 21312
- 2 clothing 3252
- 2 computers 12321
- 2 supplies 2242
- 2 textbooks 15462

- Output
- 1 computers 9161
- 2 computers 12321

awk: Other features

- Formatted printing using printf
- Conditional statements (if-else)
- Loops
 - for
 - while
 - do-while

awk: Associative arrays

- Normal arrays use integers for their indices
- Associative arrays with strings as their indices
- Example: Age["Robert"] = 56

awk: Example

```
# salesDeptLoop.awk script
BEGIN {OFS = "\t"}
{deptSales [$2] += $3}
END {for (item in deptSales)
{
    print item, ":", deptSales[item]
    totalSales += deptSales[item]
    # for
    print "Total Sales", ":", totalSales
    # END
```

Input and output

Input

- 1 clothing 3141
- 1 computers 9161
- 1 textbooks 21312
- 2 clothing 3252
- 2 computers 12321
- 2 supplies 2242
- 2 textbooks 15462

Output

computers : 21482
supplies : 2242
textbooks : 36774
clothing : 6393
Total Sales : 66891

A final awk example

```
BEGIN {count=0; sum=0}
\{if(\$1>=10 \&\& \$1 <= 100)\}
  count++
  sum+=$1
  }
else{
  print $1,"is not a double digit number"
  }
}
END{print "There are", count, "double
  digit numbers, which sum to", sum}
```

```
(C) 2005 Doug Bowman, Virginia Tech CS Dept.
```