CpSc 207 Exam 3

Fall 2011

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (5 pts) sed is more powerful than grep. What can sed do that grep cannot?
2. (20 pts) For each of the following, either give the UNIX sed command or explain the sed command. Use the file /usr/dict/words.

|  |  |
| --- | --- |
| Action | Command |
| Display only the lines in *words* that start with a “vr” and end with an “y” |  |
| Replace all occurrences of “io” with “ae” in the *words* file; display all lines |  |
| Substitute the 3rd occurrence of the letter “i” with an “e” ; show only modified lines |  |
| For all words that contain “tion”, replace each “e” in the word with an “x”; show only modified lines. |  |
| In lines 8 through 10 of the file, substitute the first “c” with a “p”; show all lines. |  |
| Delete lines 8 through 17 of the *words* file |  |
| Delete all lines in the *words* file that begin with “x” |  |
|  | sed y/abcdefg/zabcdef/ /usr/dict/words |
|  | sed s/ing/\&\&/g /usr/dict/words |
|  | sed -n ‘1,10N;s/\n//p’ /usr/dict/words |

1. (5 pts) Explain what sed, grep and awk have in common.
2. (5 pts)Explain the difference between the following two “for” commands

for i in \* for i in $\*

1. (5 pts) The following awk commands produced the output below. Notice that when the user entered “a4” the response was that it found an integer AND a string. However, when 44t was entered, the response was only that a string was found. Why doesn’t the 44t also result in “Found an integer”?

[whit@linux awkegs]$ cat patmat

/[0-9]+$/ { print "Found an integer" }

/[A-z]+/ { print "Found a string"}

[whit@linux awkegs]$ awk -f patmat

1234

Found an integer

abcd

Found a string

a4

Found an integer

Found a string

44t

Found a string

1. (30 pts) For each of the following, either give the UNIX awk command or explain the awk command. For the purpose of these questions, assume that the file *recs* contains:

John Goose 25 single

Jane Doe 22 eligible

Steve Moose 44 married

James Elk 12 not looking

Susie Horse 31 single

|  |  |
| --- | --- |
| Action | Command |
| Print column 1 followed by column 2 for all records that contain a “g” |  |
| Print all records where column 3 is greater than 20. |  |
| Print all records from Steve through Susie. |  |
| “Nicely” print column 3 followed by column 4 (space between the columns) |  |
| Print all records where column 3 is greater than 20 and less than 40. |  |
|  | awk ‘$2 ~ /^D/ ’ recs |
|  | awk ‘$1 ~ /^J/ {print $2, $4}’ recs |
|  | awk ‘$3 ~ /2$/ ’ recs |
|  | awk ‘ {print $1, $2  print $4  print $3  }’ recs |
|  | awk ‘{ t = t + $3 }  END { print t }’ < recs |

1. (5 pts) What is the output of this awk script? (use the recs file from the previous problem)

awk ‘BEGIN { FS = “e” }

{ print $2 }

END { print NR }’ < recs

1. (10 pts) What is the output of the following script called **tryit**, if the command

tryit –yz 4 passwd

is issued from the command line?

zflg=0

yflg=0

set -- `getopt z:y $\*`

echo $1 $2 $3 $4 $5 $6

while [ $1 != -- ]

do

case $1 in

-z) zflg=1

zarg=$2

shift

;;

-y) yflg=1

;;

esac

shift

done

shift

if [ $zflg = 1 -a $yflg = 1 ]

then

echo $1 $zarg $yflg $zflg

elif [ $yflg = 1 ]

then

echo $1 $yflg

elif [ $zflg = 1 ]

then

echo $1 $zarg $zflg

fi

1. (15 pts) Write a complete shell script named “backup” that accepts a list of file names as arguments and creates backups of any argument that is in fact a file. For each name that is not a file, echo to the screen “name” will not be backed up. Backed up files are to be placed in the user’s HOME directory and the extension .bak is to be added to the original file name. For example, if “a” is a file and “d” is a directory in the working directory, then the command **backup a d** would create a file named “a.bak” in the home directory and print to the screen “d is not a file and not backed up”.

**Sed help**

**Delete**

[address]d

address

* /RE/ only lines containing string
* line number
* line addressing symbol used in RE

^ start of line

$ end of line

* ! can be used after address so only lines not matching
* mixture of above

1,$d -- deletes 1st to last line

1,/^$/d -- deletes 1st through blank line

**Substitute**

[address]s/pattern/replacement/flags

replacement

& Replaced by the string matched by the RE

\n n is a single digit, Match the nth substring previously specified

\ Escape the & and \

flags

n -- any number from 1 to 512 -- replace only nth occurrence

g -- global

p – prints modified lines

w file -- write to the file

**Transform**

[address]y/abc/xyz transforms each a to an x, b to a y, and c to z

**awk help**

awk actions are enclosed in { }.

awk commands are placed in single quotes

BEGIN { action } perform the actions before processing the file

END {action} perform the actions after processing

{ action } perform actions to each input record

pattern [ {action} ] perform action on only those records that match the pattern

pat1,pat2[{action}] perform action in the range of records from pat1 to pat2

$1 is the first field, $2 is the second

$0 is the entire record

NF is number of fields

NR is number of records