CA170: Week 7

Shell Utilities

grep

```
grep                    search for a string

grep string file

(output of prog) | grep string | grep otherstring

-i    ignore case
-v    return all lines that do NOT match
```

- grep
- man grep
- grep is Substring-oriented (will match inside a word).
  Possible alternative:
  ```
  grep -w
  ```
- grep is Line-oriented (will print whole line that matches).
  Not always good: Hit we want may spread across multiple lines, so grep won't find it. Or conversely, HTML file could be entire file on one line.
  Possible solution:
  - Pre-process the file.
  - If hit spread across multiple lines, use tr or sed to change all (or many) newlines to spaces.
  - If file all on one line, use same tools to introduce new lines at different points.
- Recursive grep:
  - This exists on some (but not all) Unix/Linux.
    ```
    grep -r string .
    grep -r --include="*html" string .
    ```
  - Might be able to leave out the .
- More complex solution to long-line problem:
  - Use egrep to display from 0 to max 40 chars each side of the string:
    ```
    egrep -o ".{0,40}string.{0,40}" file
    ```
  - `.` means any char
  - 0 to max n of any char is `.{0,n}`

string matching / regular expressions

```
^                       start-of-line
$                       end-of-line
.                       any character

where "c" stands for the character:

c*                0 or more instances of c
cc*               1 or more instances of c
grep "  *"        1 or more spaces
.*                   any sequence of characters



where "c" has a special meaning, e.g. is $ or ., etc:

\c                   the character itself
grep "\."            the '.' character itself

recall the two forms of string:

grep '\$'   works (searches for the "$" char instead of end-of-line)
grep "\$"   fails (double quote treatment of $ is different to single quote
treatment of $)
grep "\\$"  works
```

- Exercise: Go to share/testsuite.
  - Find all lines in web pages containing "born".
  - Find all lines containing the "$" symbol.
  - Find all lines containing "born" at start of line.
  - Find all lines containing start of line, then one space, then "born".
  - Find all lines containing start of line, then any number of spaces, then "born".
- Regular expressions - link in notes
  - POSIX syntax - link in notes
  - POSIX character classes - link in notes

cut

```
cut     extract columns or fields of text on command-line

To extract columns 30  to end of line of the ls listing:

  ls -l | cut -c30-

In grep output, extract the 1st field, with delimiter ":"

 grep string *html | cut -f1 -d':'

Extract the 2nd to end fields, with delimiter ":"

 grep string *html | cut -f2- -d':'
```

- Why "-f2-" ?
- Why not "-f2" ?

sed

```
sed     "stream editor" - find and replace text on command-line (and other
things)


sed 's|oldstring|newstring|'    change first match on each line
sed 's|oldstring|newstring|g'   change all matches

'|' is just my choice of a separator.
Other people like '/'
We can actually use any character as a separator (whatever comes first after
"s").


e.g. ls listing that highlights web pages:

 ls -l      | sed "s|\.html| [Web page]|"


e.g. ls listing that changes how my username appears:

 ls -l      | sed "s|$USER|ME|"
```

- Sed - link in notes
- man sed - link in notes
- sed FAQ - link in notes
    - How to do case-insensitive matching (can be tricky)
- sed examples - link in notes

sed examples

- To insert a new line
    - How to insert new line - link in notes
    - On DCU Linux, put a new line in front of every string "www":

      ```
      sed 's|www|\nwww|g'
      ```

    - (recognises "\n")
    - On some other platforms, need to do this:

      ```
      sed 's|www|\
      www|g'
      ```

- To put new lines in front of and after every HTML tag
    - On DCVU Linux

      ```
      sed 's|<|\n<|g'        |

      sed 's|>|>\n|g'
      ```

    - Other platforms

      ```
      sed 's|<|\
      <|g'        |

      sed 's|>|>\
      |g'
      ```

- To substitute back in the pattern we matched

```
# \( ... \) to mark a pattern
# \1 to reference it later

# e.g. change:
# (start of line)file.html: ...
# to:
# <a href=file.html>file.html</a>: ...

# search for:
# ^\(.*\.html\):
# change to:
# <a href=\1>\1</a>:


grep -i $1 *html |

     sed -e "s|^\(.*\.html\):| <a href=\1>\1</a>: |g"
```

tr

- tr - character substitutions

```
# change spaces to new lines:

cat file | tr ' ' '\n'

# convert Windows file format to Unix file format:

tr -d '\015'
```

- man tr
- uses character classes (numeric, alphanumeric, etc.)

awk

- awk - a powerful pattern scanning and processing language

dirname, basename

- useful cutting and pasting with filenames
- dirname - link in notes
- Basename - link in notes

```
$ echo $HOME
/users/group/me

$ dirname $HOME
/users/group

$ basename $HOME
me

$ dirname `dirname $HOME`
/users
```

head and tail

- Head
  - Display the first 100 lines of the output:
    ```
    grep string files | head -100
    ```
  - Note: When head gets the first 100 lines, the pipe closes and grep terminates.
  - As opposed to: Doing the entire grep and then taking the first 100.
  - To see this is true, run the program "yes" (which outputs an infinite number of lines) with head, and you will see it does stop:
    ```
    yes | head -20
    ```
- Tail
  - Display the last 30 lines of the logfile:
    ```
    cat logfile | tail -30
    ```

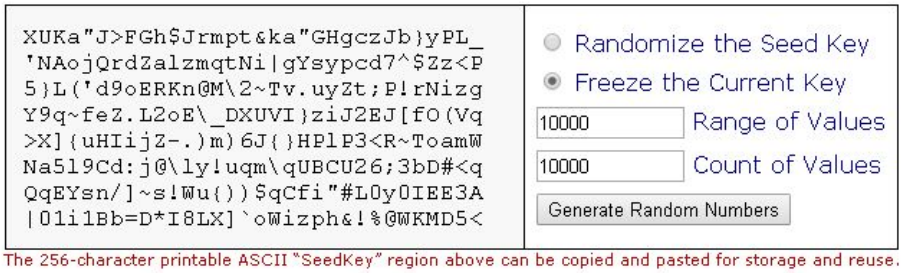
date

```
date                          looks like: "Tue Feb 17 16:28:33 GMT 2009"
CURRENTDATE=`date`            remember backquotes
echo $CURRENTDATE

date "+%b %e"                 looks like: "Jan 21"

date "+%b.%e.log"             can add things to the string

file=`date "+%b.%e.log"`
echo $file
```

- man date - link in notes
- Using date to get unique filename
  - Say web server in response to client needs to make a temporary file.
  - Use date to get a new filename that is unique to the current second:
    ```
    timenow=`date +%H%M%S`
    filename="/tmp/random.$timenow.txt"
    ```
  - Unique to second and nanosecond:
    ```
    date "+%H.%M.%S.%N"
    ```
- Alternative ways of getting unique filename
  - Unique filename based on process ID:
    ```
    filename=/tmp/random.$$.txt
    ```
    - Would be different for each process.
  - Random number generator in Shell:
    ```
    echo $RANDOM
    ```
    - This is a strange environment variable. It does not exist until you try to access it. Then it exists!

    ```
    set | grep -i random
    echo $RANDOM
    set | grep -i random
    echo $RANDOM
    ```

- About random-number generators:
  - A pseudo-random-number generator typically generates a series of random-looking numbers based on a start seed.
  - If the seed is identical, the algorithm will generate the same numbers.
  - So we want a different start seed each time we run.
  - Seed could be based on the current time (second and nanosecond).



  - JavaScript random number generation demo.
  - Usage: Freeze the seed key. Set parameters. Then generate random numbers.
  - Can view JavaScript code to see how it works.