CS 35L

Week 2

TA: Tomer Weiss
Jan-14-2016

goo.gl/4Dz78c

Slides

PTE

- Will pass sign-in sheet
- Attend each class and sign your name
- PTE given during second week of classes

Assignment 2 Instructions (email\piazza)

https://piazza.com/class/ij094jwyhvp56n?cid=150

Please make sure you have followed the instructions below, before you submit on ccle.

1) The buildwords script must accept input from STDIN and output to STDOUT. So,

you should be able to run the script as shown on the webpage

cat foo.html | ./buildwords | less

or as

cat hwnwdseng.htm | ./buildwords | less

or as

./buildwords < hwnwdseng.htm > hwords

So, please make sure you do NOT have any input/output filenames (hwnwdseng.htm or hwords) in your buildwords script.

2) Your buildwords script should just read from stdin and write only the hawaiian words to stdout. That is all it should do. All the commands related to the english and hawaiian spell checkers must go into the lab2. log, and NOT the buildwords script.

3) The sameln script should work with an absolute OR a relative path as argument. For example,

Absolute

./sameln /usr/home/directory/

or as

Relative

./sameln directory/

./sameln directory

(In these cases, the directory is at the same level as the sameln script.).

4) The ordering priority for the sameln script should be as follows

a) The starting period '.' takes precedence over everything

b) Lexicographic ordering (this means that both 'A' and 'a' come before 'Z' and 'z')

c) You can do ascii based ordering if there is a clash in non-alphabet characters such as '*', '-' etc.

5) You don't have to conform to the 80 column limit for buildwords and sameln scripts. The limit is only for the lab2.log file.

6) Your scripts MUST run on the seas net machines! They will be graded on the seasnet machines(lnxsrv09, lnxsrv07).

Assignment 2

- Deadline - 01-15-2016
- Sample shell that can help:
  - https://ccle.ucla.edu/mod/forum/view.php?id=976217

Finding duplicate files

- Reminder: man cmp
- What is the output of cmp in case of duplicate files?

Hard Links

- Reminder: man ln
- How to create link?

Exit: Return value

Check exit status of last command that ran:

```
echo $?
```

Value	Meaning
0	Command exited successfully
>0	Failure during redirection
1-125	Command exited unsuccessfully. The meanings
126	Command found, but file was not executable
127	Command not found
>128	Command died due to receiving a signal

Check file type

- Reminder: man test
- For example, checking for regular files

Sample Script

- Hints and skeleton for how to “glue” together all commands in previous slides
-

for Loops

```
for i in atlbrochure*.xml
do
    echo $i
    mv $i $i.old
    sed 's/Atlanta/&, the capital of the South/' < $i.old > $i
done
```

while and until loops

while *condition*
do
    statements
done

until *condition*
do
    statements
done

break and continue

- Pretty much the same as in C/C++

Regular Expressions

- Notation that lets you search for text that fits a particular criterion, such as “starts with the letter a”
- http://regexpal.com to test your regex expressions
- Simple regex tutorial: https://www.icewarp.com/support/online_help/203030104.htm

Functions

- Must be defined before they can be used
- Can be done either at the top of a script or by having them in a separate file and source them with the “dot” (.) command.

Example

```
# wait_for_user --- wait for a user to log in
#
# usage: wait_for_user user [ sleeptime ]
wait_for_user ( ) {
    until who | grep "$1" > /dev/null
    do
        sleep ${2:-30}
    done
}
```

Functions are invoked the same way a command is

```
wait_for_user tolstoy        Wait for tolstoy, check every 30 seconds
wait_for_user tolstoy 60          Wait for tolstoy, check every 60 seconds
```

The position parameters ($1, $2, etc) refer to the function's arguments.
The return command serves the same function as exit and works the same way

```
answer_the_question ( ) {
    ...
    return 42
}
```

Simple Execution Tracing

- To get shell to print out each command as it's execute, precede it with "+"
- You can turn execution tracing within a script by using:

  `set -x`: to turn it on

  `set +x`: to turn it off

Lab work

Regular Expressions reference

Backreferences

- Match whatever an earlier part of the regular expression matched
  - Enclose a subexpression with \(and \).
  - There may be up to 9 enclosed subexpressions and may be nested
  - Use \digit, where digit is a number between 1 and 9, in a later part of the same pattern.

Pattern	**Matches**
\(ab\)\(cd\)[def]*\2\1	abcdcdab, abcdeeecdab, abcdddeeffcdab, ...
\(why\).*\1	A line with two occurrences of why
\([[:alpha:]_][[:alnum:]_]*\) = \1;	Simple C/C++ assignment statement

Operator Precedence (High to Low)

Operator	Meaning
[. .] [= =] [: :]	Bracket symbols for character collation
metacharacter	Escaped metacharacters
[ ]	Bracket expressions
\(\) *digit*	Subexpressions and backreferences
* \{ \}	Repetition of the preceding single-character regular expression
no symbol	Concatenation
^ $	Anchors

sed

- Now you can extract, but what if you want to replace parts of text?
- Use sed!

```
sed 's/regExpr/replText/'
```

- Example

```
sed 's/:.*//' /etc/passwd  Remove everything
      after the first colon
```

Text Processing Tools

- sort: sorts text
- wc: outputs a one-line report of lines, words, and bytes
- lpr: sends files to print queue
- head: extract top of files
- tail: extracts bottom of files

More on Variables

- Read only command

```
hours_per_day=24 seconds_per_hour=3600 days_per_week=7      Assign values
readonly hours_per_day seconds_per_hour days_per_week   Make read-only
```

- Export: puts variables into the environment, which is a list of name-value pairs that is available to every running program

```
PATH=$PATH:/usr/local/bin     Update PATH
export PATH                   Export it
```

- env: used to remove variables from a program's environment or temporarily change environment variable values
- unset: remove variable and functions from the current shell

Parameter Expansion

- Process by which the shell provides the value of a variable for use in the program

```
reminder="Time to go to the dentist!" Save value in
          reminder
sleep 120                  Wait two minutes
echo $reminder             Print message
```

Pattern–matching operators

path=/home/tolstoy/mem/long.file.name

Operator	Substitution
${*variable#pattern*}	If the pattern matches the beginning of the variable's value, delete the shortest part that matches and return the rest.
Example: ${path#/*/}	Result: tolstoy/mem/long.file.name
${*variable##pattern*}	If the pattern matches the beginning of the variable's value, delete the longest part that matches and return the rest.
Example: ${path##/*/}	Result: long.file.name
${*variable%pattern*}	If the pattern matches the end of the variable's value, delete the shortest part that matches and return the rest.
Example: ${path%.*}	Result: /home/tolstoy/mem/long.file
${*variable%%pattern*}	If the pattern matches the end of the variable's value, delete the longest part that matches and return the rest.
Example: ${path%%.*}	Result: /home/tolstoy/mem/long

String Manipulation

- ${string:position}: Extracts substring from $string at $position
- ${string:position:length} Extracts $length characters of substring $string at $position
- ${#string}: Returns the length of $string

$IFS (Internal Field Seperator)

- This variable determines how Bash recognizes fields, or word boundaries, when it interprets character strings.
- $IFS defaults to whitespace (space, tab, and newline), but may be changed
- **echo "$IFS"** (With $IFS set to default, a blank line displays.)
- More details:
  - http://tldp.org/LDP/abs/html/internalvariables.html

Accessing Shell Script Arguments

- Positional parameters represent a shell script's command line arguments
- For historical reasons, enclose the number in braces if greater than 9

```
#! /bin/sh
#test script
echo first arg is $1
echo tenth arg is ${10}

> ./argtest 1 2 3 4 5 6 7 8 9 10
```

Quotes

- Three kinds of quotes
  - Backticks ``

```
$ echo $PATH /bin:/usr/bin:/usr/X11R6/bin:
  /usr/local/bin
```

Basic Command Searching

- $PATH variable is a list of directories in which commands are found

```
$ echo $PATH /bin:/usr/bin:/usr/X11R6/bin:
  /usr/local/bin
```