

CE350

Lecture7

PROGRAMMING THE BASH SHELL PART III

by İlker Korkmaz and Kaya Oğuz

Programming the bash

- As pointed in syllabus document, there will be 5 lecture weeks to cover the programming with bash scripts.
- This lecture is the 3rd part on scripting in bash.

References

- The contents of this lecture are prepared with the help of and based on:
 - the textbook, UNIX Shells by Example (Chapter3, and Chapter4)
 - the tutorial, Advanced Bash-Scripting Guide at <http://tldp.org/LDP/abs/html/>

Contents of Lecture7

- Regular expressions
- grep

Just a review example on the previous lecture concepts

- <http://tldp.org/LDP/abs/html/randomvar.html>
 - Example 9.11

Regular Expressions

- A Regular Expression (RE) is a pattern of characters used to be matched in a search.
- In **vi**, RE patterns are enclosed in forward slashes:
 - /patternWord/

RE metacharacters

- RE metacharacters are explained with some pattern examples in Table 3.1. Some RE metacharacters are:
- `^` : `/^love/` #matches all lines beginning with pattern love.
- `$` : `/love$/` #matches all lines ending with love.
- `.` : `/l..e/` #matches lines containing l, followed by 2 characters, followed by an e.
- `*` : `/*love/` #matches zero or more of the preceding character, space, followed by the pattern love.
- `[ ]` : `/[Ll]ove/` # matches lines containing Love or love.
- `[x-y]` : `/[A-Z]ove/` #matches any letter from A through Z followed by ove.
- `[^ ]` : `/[^A-Z]/` #matches any character not in the range between A and Z.
- `\` (escape metacharacter): `/love\./` #matches exactly pattern "love."

RE-match operator in bash

`=~`

- After version 3, RE-match operator, `=~`, is available for bash.
- Some examples:
 - <http://tldp.org/LDP/abs/html/bashver3.html#REGEXMATCHREF>
 - `if [[ "$1" =~ "[a-zA-Z][a-zA-Z]$" ]] # Ends in two alpha chars?`
 - `if [[ $mail =~ "^From " ]] # Match "From" field in mail message.`
 - `if [[ "$line" =~ ^[a-z] ]]`
 - `then # line begins with lowercase character.`

grep

- The grep family consists of the commands grep, egrep, and fgrep.
- grep
 - **g**lobally searches for **r**egular **e**xpressions and **p**rints out the lines.
- egrep
 - extended grep
 - supports more regular expression metacharacters
- fgrep
 - fixed grep OR fast grep
 - treats all characters as literals, which means no metacharacters

How grep works

- NOT to be used with / /
 - as /patternWord/ written in **vi**
- To be used as: **grep patternWord File**
- OR **grep 'patternWord' File**
 - patternWord is to be searched in File
 - If grep is successful, the line from the file will appear on the screen and grep returns an exit status of 0.
 - If the related pattern is not found in file, no output will appear and grep returns an exit status of 1.
 - If the file is not a legitimate one, an error will be sent to screen and grep returns an exit status of 2.
 - If the fileName is not given grep assumes that it is getting the input from either stdin or a pipe.
 - **ps -ef | grep ilker**

grep's RE metacharacters

- Table 4.1 explains grep's RE metachars.
- `^`, `$`, `.`, `*`, `[ ]`, `[^ ]`, `\<`, `\>`, `\(..\)`, `x\{m\}`, `x\{m,\}`
- In textbook,
 - Try to understand Examples 4.11, 4.12, 4.14, 4.16, 4.17, 4.18, 4.21, and 4.28.

grep options

- Check your man pages or help pages for a complete list of grep options available on your UNIX/Linux version.
 - `man grep`
 - OR `grep --help`
- `-n` option precedes the output line with the number of the line the pattern was found.
- `-i` option turns off the case sensitivity.
- `-w` option finds the pattern if it is a word, not part of a word.
- `-v` option prints all lines not containing the given pattern.
 - What is the goal below?
 - `grep -v 'ilker korkmaz' myFile > temp`
 - `mv temp myFile`
- `-c` option prints the total number of lines where the pattern was found.

More grep examples

- Table 4.3 in the textbook shall be examined interactively with the students.

egrep

- Extended grep (egrep) provides additional regular expression metacharacters. However, `\( \)` and `\{ \}` are not allowed.
- `+` : `'[a-z]+ove'` #matches one or more lowercase letters, followed by `ove`.
- `?` : `'lo?ve'` #matches for an `l` followed by either one or zero `o`, followed by `ve`.
- `a|b` : `'love|hate'` #matches for either `love` or `hate`
- `()` : groups characters such as in
 - `'love(able|ly)'`

fgrep

- fixed grep or fast grep
- fgrep does not recognize any metacharacter as being special.
 - fgrep '[A-Z]**[0-9].\$5.00' File
 - The literal string [A-Z]**[0-9].\$5.00 is to be matched.

GNU grep

- Linux uses GNU grep.
- In Linux, rgrep (recursive grep) is also available, which is not a member of the UNIX grep family.
- GNU grep contains two sets of RE metacharacters:
 - basic set: regular version; grep, OR grep -G.
 - extended set: egrep OR grep -E
- Regular grep may also use an extended set metacharacter with a backslash preceded to it:
 - \?, \+, \|, \(\)

GNU grep formats

- Table 4.6 shows GNU grep formats.
- `grep 'pattern' File` #default is basic RE set
- `grep -G 'pattern' File` #same as above
- `grep -E 'pattern' File` #extended RE set
- `grep -F 'pattern' File` #fgrep, no RE
- `grep -P 'pattern' File`
 - # -P option interprets the pattern as a Perl RE

rgrep (grep -R)

- Linux's recursive grep (rgrep) can recursively descend a directory tree.
 - `grep -r 'ilker' ./myDir`
 - OR
 - `rgrep 'ilker' ./myDir`
 - searches recursively for all files under ./myDir directory.

A quick quiz:

- What would be the output of the following commands?
(Example 4.47)
 - `egrep 'Sh|u' dataFile`
 - `grep -E 'Sh|u' dataFile`
 - `grep 'Sh\\u' dataFile`

TO DO:

- The next lecture, Lecture8, will include RE examples with the editor **sed**. So, try to dissect many RE examples before the next lecture.
- Do the LAB exercises given on page 124 of the textbook.