Lab 04 - Bash: Iterating Over Files

CS 1XA3

Jan. 30, 2018

Recap: Find and Xargs

- Execute commands on find by piping into xargs

```
find dir -name file | xargs command
```

- To avoid issues with special characters, use the -print0 and -0 flags

```
find dir -name file -print0 | xargs -0 command
```

Recap: The Read Command

- The read command parses user input into bash variables

```
read var1 var2 ...
```

- Prompt a user with the -p flag

```
read -p "Enter two words: " var1 var2
```

- The default delimaters a newline, set a new one with the -d flag

```
read -d ';' var1 var2
```

- The default seperator is a space, to set a new one assign IFS (Internal Field Seperator)

```
IFS=':' read var1 var2
```

Iterate Over Files With For-In

To iterate through items inside a directory (but not subdirectories), use the for-in syntax and glob patterns Example

```
shopt -s nullglob
for file in *.tmp
do
        echo "$file"
done
```

Note: the **shopt** handles the situation where there are no .tmp's in the directory, and bash otherwise attempts to iterate over ***.tmp** literally

Iterate Over Files with Find-While

To iterate over items in a directory and all it's subdirectories, pipe find into a while loop

Example

```
find . -name "*.tmp" -print0 |
    while IFS='' read -r -d $'\0' file
    do
        echo "$file"
    done
```

Notes

- *IFS* = stops leading, trailing spaces from being trimmed
- read -d specifies the argument seperator, we give the **NULL** character corresponding to print0

Notes on Find-While

A simpler, far more error prone version of the loop could be written like so

```
find . -name "*.tmp" |
    while read file
    do
        echo "$file"
    done
```

It's best to play safe, and use the extra flags in the previous version

Iterate over Lines of a File

You can use the same trick to read lines from a file Example

```
cat file.txt |
    while read var
    do
        echo $var
    done
```

Chaining Commands

```
# place two commands on one line
command1 ; command2

# command2 only if command1 has no error
command1 && command2

# command2 only if command1 has an error
command1 || command2
```

Useful Miscellaneous Commands

```
wc -l file    # word count lines of file
du -h inp     # human readable size of file or dir
wget url      # download file at url
whoami        # outputs your username
which inp     # shows the full path of inp
sort file     # sorts a file, duh
```