./missing-semester - Shell Tools and Scripting - Exercises

06 Sep 2020

Course located at: missing.csail.mit.edu

Exercises

- Read man ls and write an ls command that lists files in the following manner
  - Includes all files, including hidden files
  - Sizes are listed in human readable format (e.g. 454M instead of 454279954)
  - Files are ordered by recency
  - Output is colorized

每个OPTION字符参数指代特定的含义

```
$ ls -ahltc
-rw-r--r--   1 user group 1.1M Jan 14 09:53 baz
drwxr-xr-x   5 user group  160 Jan 14 09:53 .
-rw-r--r--   1 user group  514 Jan 14 06:42 bar
-rw-r--r--   1 user group 106M Jan 13 12:12 foo
drwx------+ 47 user group 1.5K Jan 12 18:08 ..
```

- Write bash functions marco and polo that do the following. Whenever you execute marco the current working directory should be saved in some manner, then when you execute polo, no matter what directory you are in, polo should cd you back to the directory where you executed marco. For ease of debugging you can write the code in a file marco.sh and (re)load the definitions to your shell by executing source marco.sh.

```
#!/usr/bin/env sh
marco () {
  pwd > /tmp/marco
}
polo () {
  cd $(cat /tmp/marco)
}
```

```
marco() {
   export MARCO=$(pwd)
}

polo() {
   cd "$MARCO"
}
```

Say you have a command that fails rarely. In order to debug it you need to capture its output but it can be time consuming to get a failure run. Write a bash script that runs the following script until it fails and captures its standard output and error streams to files and prints everything at the end. Bonus points if you can also report how many runs it took for the script to fail.

```
#!/usr/bin/env bash

n=$(( RANDOM % 100 ))

if [[ n -eq 42 ]]; then
   echo "Something went wrong"
   >&2 echo "The error was using magic numbers"
   exit 1
fi

echo "Everything went according to plan"
```

solve.sh:

将样例程序存为random-fail.sh
再运行 sh ./solve.sh
>&2 可以看作是将stdout 重定位到stderr (&2 获取stderr的地址)

```
#!/usr/bin/env bash

runs=0
error=0
while [ true ]; do
    bash random-fail.sh 1>>/tmp/solve-out.txt 2>>/tmp/solve-error.txt
    ((++runs))
    if [ -s /tmp/solve-error.txt ]; then
        break
    fi
done
cat /tmp/solve-out.txt /tmp/solve-error.txt
rm /tmp/solve-{out,error}.txt
echo "The command took $runs runs to throw an error."
```

output:

```
...
Everything went according to plan
Everything went according to plan
Something went wrong
The error was using magic numbers
The command took 25 runs to throw an error.
```

As we covered in the lecture `find`'s `-exec` can be very powerful for performing operations over the files we are searching for. However, what if we want to do something with all the files, like creating a zip file? As you have seen so far commands will take input from both arguments and STDIN. When piping commands, we are connecting STDOUT to STDIN, but some commands like `tar` take inputs from arguments. To bridge this disconnect there's the `xargs` command which will execute a command using STDIN as arguments. For example `ls | xargs rm` will delete the files in the current directory. Your task is to write a command that recursively finds all HTML files in the folder and makes a zip with them. Note that your command should work even if the files have spaces (hint: check `-d` flag for `xargs`)

```
$ touch test{0..5}.html test{0..5}.jpg filename\ spaces\ {a..f}.html
$ mkdir subdir && cd $_
$ touch test{0..5}.html test{0..5}.jpg filename\ spaces\ {a..f}.html
$ cd ..
$ find . -name '*.html' | xargs -d '\n' tar czf archive.gzip
$ tar tvf archive.gzip
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./filename spaces a.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./filename spaces b.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./filename spaces c.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./filename spaces d.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./filename spaces e.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./filename spaces f.html
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/filename spaces a.ht
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/filename spaces b.ht
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/filename spaces c.ht
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/filename spaces d.ht
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/filename spaces e.ht
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/filename spaces f.ht
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/test0.html
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/test1.html
```

先用wildcard 生成test文件，包括次一级目录
查找相关文件，生成list后 将参数传递给xargs(将
STDIN转换为命令行参数) -d 是指定参数分隔符

```
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/test2.html
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/test3.html
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/test4.html
-rw-r--r-- noble/noble       0 2020-09-09 14:45 ./subdir/test5.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./test0.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./test1.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./test2.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./test3.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./test4.html
-rw-r--r-- noble/noble       0 2020-09-09 14:47 ./test5.html
```

(Advanced) Write a command or script to recursively find the most recently modified file in a directory. More generally, can you list all files by recency?

```
# Most recently modified file
## find files of type file, printing seconds-since-epoch and filename\n with
## sort that numerically (low-to-high), grab the last line, and cut just file
find . -type f -printf '%T@%p\n' | sort -n | tail -n 1 | cut -c 22-

# List all files by recency
## find files, print 'TimeFilename\n', sort high-to-low, remove time info
find . -type f -printf '%T@%p\n' | sort -rn | cut -c 22-
```

Zach Heller - 2021

通过man xx 去查看各个命令的具体用法
细节内容进行相关搜索
学习知识，进行知识拓展，逻辑构建，系统认知，全局战略
https://man7.org/linux/man-pages/man1/find.1.html 关于find的参数信息
https://man7.org/linux/man-pages/man1/sort.1.html
https://man7.org/linux/man-pages/man1/tail.1.html
https://www.man7.org/linux/man-pages/man1/cut.1.html