COOL STUFF WITH VIM AND BASH

Made by Georgie Lee for

SoC UNIX Workshop 07 August 2020

But first, some QoL enhancements

Make sure you have CTRL+SHIFT+V (paste) and CTRL+SHIFT+C (copy) enabled.

Create `~/.inputrc` and add these in

```
"\e[A": history-search-backward
"\e[B": history-search-forward
set show-all-if-ambiguous on
set completion-ignore-case on
```

Now, you can do case-insensitive file and directory name completion!

(Warm-up)

Make a list of numbers with Vim using

```
:put =range(1,10)
```

What happens? Try it!

(Warm-up)

Now increment every number by 5!

```
%s/\d+/\=submatch(0)+5/g
```

Cool!

Automated code testing with Bash

```
.
├── ProblemSet.pdf
├── input
│   ├── 1.in
│   ├── 2.in
│   ├── 3.in
│   └── 4.in
├── output
│   ├── 1.ans
│   ├── 2.ans
│   ├── 3.ans
│   └── 4.ans
├── solve
└── solve.c
```

Given these inputs, check if the program `solve` produces correct outputs. i.e. `./solve 1.in` matches `1.ans`.

We can run the program by hand e.g. `./solve 1.in` for each input file and verify correctness against each output file.

We can run the program by hand e.g. `./solve 1.in` for each input file and verify correctness against each output file.

But we can do better!

We can make a file `run.sh` with content:

```
./solve 1.in 1.out && diff 1.out 1.ans
./solve 2.in 2.out && diff 2.out 2.ans
./solve 3.in 3.out && diff 3.out 3.ans
./solve 4.in 4.out && diff 4.out 4.ans
```

With the power of Vim, we can type this quickly!

`./run.sh` will give no output if all is well :)

We could write a simple script that settles everything for us, regardless the of number of test cases.

```
echo "Running test cases..."
dir="."
i=0
for f in `ls ${dir}/input | sort -V`
  do
    let i++
    echo Case $i:
    echo ${dir}/input/$f
    ./solve < $dir/input/$f > $dir/output/${f%%.*}.out
    diff $dir/output/${f%%.*}.ans $dir/output/${f%%.*}.out
  done
```

We could write a simple script that settles everything for us, regardless the of number of test cases.

```
echo "Running test cases..."
dir="."
i=0
for f in `ls ${dir}/input | sort -V`
  do
    let i++
    echo Case $i:
    echo ${dir}/input/$f
    ./solve < $dir/input/$f > $dir/output/${f%%.*}.out
    diff $dir/output/${f%%.*}.ans $dir/output/${f%%.*}.out
  done
```

You have just automated your homework.

Mass rename files with Vim and Bash

from this

```
1-sql_practice.sql
10-sql_practice.sql
11-sql_practice.sql
12-sql_practice.sql
13-sql_practice.sql
2-sql_practice.sql
3-sql_practice.sql
4-sql_practice.sql
5-sql_practice.sql
6-sql_practice.sql
7-sql_practice.sql
8-sql_practice.sql
9-sql_practice.sql
```

→

to this

```
01-sql_practice.sql
02-sql_practice.sql
03-sql_practice.sql
04-sql_practice.sql
05-sql_practice.sql
06-sql_practice.sql
07-sql_practice.sql
08-sql_practice.sql
09-sql_practice.sql
10-sql_practice.sql
11-sql_practice.sql
12-sql_practice.sql
13-sql_practice.sql
```

Recently did this when I realized there were more exercises than I had anticipated!
Not 0-padding gives annoying sorting issues ><

Process

Process

1. Read the filenames into vim

Process

1. Read the filenames into vim
2. Mass substitution

Process

1. Read the filenames into vim
2. Mass substitution
3. Run each line with bash

1. Read the filenames into vim

```
ls *.sql | vim -
```

2. Mass substitution part 1

```
:%s/.*/\="mv ".submatch(0)." ".submatch(0)/g
```

2. Mass substitution part 1

```
:%s/.*/\="mv ".submatch(0)." ".submatch(0)/g
```

```
1-sql_practice.sql
10-sql_practice.sql
11-sql_practice.sql
12-sql_practice.sql
13-sql_practice.sql
2-sql_practice.sql
3-sql_practice.sql
4-sql_practice.sql
5-sql_practice.sql
6-sql_practice.sql
7-sql_practice.sql
8-sql_practice.sql
9-sql_practice.sql
```

→

```
mv 1-sql_practice.sql 1-sql_practice.s
mv 10-sql_practice.sql 10-sql_practice
mv 11-sql_practice.sql 11-sql_practice
mv 12-sql_practice.sql 12-sql_practice
mv 13-sql_practice.sql 13-sql_practice
mv 2-sql_practice.sql 2-sql_practice.s
mv 3-sql_practice.sql 3-sql_practice.s
mv 4-sql_practice.sql 4-sql_practice.s
mv 5-sql_practice.sql 5-sql_practice.s
mv 6-sql_practice.sql 6-sql_practice.s
mv 7-sql_practice.sql 7-sql_practice.s
mv 8-sql_practice.sql 8-sql_practice.s
mv 9-sql_practice.sql 9-sql_practice.s
```

2. Mass substitution part 2

```
:%s/sql \([1-9]\)-/\="sql 0".submatch(1)."-"/g
```

2. Mass substitution part 2

```
:%s/sql \([1-9]\)-/\="sql 0".submatch(1)."-"/g
```

```
sql_practice.sql 1-sql_practice.s
-sql_practice.sql 10-sql_practice
-sql_practice.sql 11-sql_practice
-sql_practice.sql 12-sql_practice
-sql_practice.sql 13-sql_practice
sql_practice.sql 2-sql_practice.s
sql_practice.sql 3-sql_practice.s
sql_practice.sql 4-sql_practice.s
sql_practice.sql 5-sql_practice.s
sql_practice.sql 6-sql_practice.s
sql_practice.sql 7-sql_practice.s
sql_practice.sql 8-sql_practice.s
sql_practice.sql 9-sql_practice.s
```

→

```
mv 1-sql_practice.sql 01-sql_prac
mv 10-sql_practice.sql 10-sql_pra
mv 11-sql_practice.sql 11-sql_pra
mv 12-sql_practice.sql 12-sql_pra
mv 13-sql_practice.sql 13-sql_pra
mv 2-sql_practice.sql 02-sql_prac
mv 3-sql_practice.sql 03-sql_prac
mv 4-sql_practice.sql 04-sql_prac
mv 5-sql_practice.sql 05-sql_prac
mv 6-sql_practice.sql 06-sql_prac
mv 7-sql_practice.sql 07-sql_prac
mv 8-sql_practice.sql 08-sql_prac
mv 9-sql_practice.sql 09-sql_prac
```

3. Run each line with bash

```
:w !sh
```

3. Run each line with bash

```
:w !sh
```

This basically renames each file to be 0 padded.
Seems easy to do by hand? **Try 2048 files.**

My helpful bash aliases and functions for working seamlessly between windows and wsl file systems.

```bash
# opens (dora the) explorer in cur dir
alias dora='explorer.exe .'

# copies pwd output
alias cpwd='pwd | clip.exe'

# turns pwd output from UNIX → WINDOWS path format and cop
alias ccpwd='wslpath -w "$(pwd)" | clip.exe'

# usage: cdd "<CTRL+SHIFT+V>" to go to a windows directory
cdd() {
  cd "$(wslpath -a "$1")"
}
```

Thanks for coming to my TED talk.

- My Instagram
- My LinkedIn
- georgie@u.nus.edu