How to write a shell script

Start Building a Script

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

# your code starts here!
echo "Hello, world!"
```

- Save those codes from the previous page to a file, like `hello.sh`
- Two ways to run it:
- Run it with `$ bash hello.sh`
- `Give it an ‘r’ permission and run it as $ ./hello.ssh`

echo - display a line of text

```
$ echo [OPTION]... [STRING]...
```

options:
`-n`, print without trailing newline
`-e`, enable backslash escapes

examples: echo

(↵ represents the newline here)

```
$ echo "Hello, world!"
Hello, world!↵
$ echo -n "Hello, world!"
Hello, world!
$ echo -e "Hello, world!\n\n\n"
Hello, world!↵↵↵
$ echo -ne "Hello, world!\n"
Hello, world!↵
```

Flow Control

- if
- for
- while
- break
- continue
- case

if

```
if condition; then
    commands
elif condition; then
    commands
else
    commands
fi
```

loops - for, while

```
for VARIABLE in 1 2 3 4 5 .. N;
do
        commands
done
```

```
while condition; do
    commands
done
```

```
for VARIABLE in $(COMMAND);
do
        commands
done
```

break and continue

```
while condition1; do
    if condition2; then
        commands
    elif condition3; then
        continue
    else
        break
    fi
    commands
done
```

`break` - leave the loop
`continue` - go to the next iteration

case

```
case VARIABLE in
    pattern1)
        commands
        ;;
    pattern2)
        commands
        ;;
esac
```

Useful Commands

- cut
- grep
- awk
- sed

You may find some of them useful in HW1!

cut - remove sections from each line of files

```
$ cut OPTION... [FILE]...
```

Options:
`-d`, `-f`
`-c`

examples: cut

```
$ cat /etc/passwd | cut -d : -f 1,6
root:/root
daemon:/usr/sbin
bin:/bin
...
$ ls -l | cut -c 1-10
total 1676
drwxr-xr-x
-rw-r--r--
...
```

grep - print lines matching a pattern

```
$ grep [OPTIONS] PATTERN [FILE...]
```

Options:

```
-c
-n
...
```

`PATTERN` can be **regular expressions**(will cover in *sed*).

examples: grep

```
$ cat /etc/passwd | grep root  # equals to “grep root /etc/passwd”
root:x:0:0:root:/root:/bin/bash

$ cat /etc/passwd | grep root -c
1

$ cat /etc/passwd | grep root -n
1:root:x:0:0:root:/root:/bin/bash
```

awk - pattern scanning and processing language

```
$ awk {AWK language}
```

Basic usage:

```
$ awk ‘{print $1}’
$ awk -F”:” ‘{print $1 $3}’
```

examples: awk

```
$ cat /etc/passwd | awk -F":" '{print $1":"$6}'
root:/root
daemon:/usr/sbin
bin:/bin
...

(same as 'cut -d : -f 1,6'!)
```

sed - stream editor for filtering and transforming text

```
$ sed [OPTION]...
```

simple examples:

```
$ cat a.txt | sed ‘2,5d’ # print a.txt with line 2 to 5 deleted
$ cat a.txt | sed ‘2a apple’ # print a.txt with ‘apple’ inserted
next to line 2 (on line 3)
$ cat a.txt | sed ‘2i apple’ # like the previous but before line 2
```

Sed is a powerful tool to edit texts.
To fully use the power of sed, one needs to learn **regular expressions** first.

Regular Expressions

A regular expression (regex) is an expression describing some form of texts.

A simple word such as “apple” is actually a valid regex.

There are many regexs, and sed uses Basic Regular Expression (BRE).

You can freely combine regex to match almost everything!

BRE syntax

expression	description
a single character, such as a, b, or c	matches itself
*	matches zero or more instances of previous regex
.	matches any character
^	matches the beginning of the pattern space, such as the beginning of a file
$	like ^, but matches the end
[list]	matches any of the character in the list
[^list]	matches any of the character not in the list

BRE syntax

expression	description
\+	like *, but matches one or more
\?	like *, but matches zero or one
\{i\}	matches exactly i times ($0 \leq i \leq 255$)
\{i,j\}	matches i to j times (inclusive)
\{i,\}	matches more than or equal to i times

examples: regex

‘a\{3\}b’ -- matches ‘aaab’

‘[aeiou]’ -- matches vowels

‘.*’ -- matches any string

‘$’ -- matches the end of string

‘\$’ -- matches a dollar sign, ‘\’ is for ‘escaping’

‘\\$’ -- matches strings ending with a backslash

Back to sed

```
$ sed [OPTION] [SCRIPT]...
```

‘2,5d’, ‘2a apple’

Try to read the manual by yourselves!

You can find it here: https://www.gnu.org/software/sed/manual/sed.html#sed-commands-list

examples: sed

examples with regex:

```
$ cat a.txt | sed ‘s/a\{2\}b/bba/g’ # replace all ‘aab’ to ‘bba’
$ cat a.txt | sed ‘s/#.*$//g’ # replace all strings starting with
a ‘#’ to empty lines
$ cat a.txt | sed ‘/^$/d’ # delete all empty lines
```

You can try to use sed by yourselves.
There are tons of examples which you can find on the Internet.
You are also encouraged to read the GNU sed manuals.
Try to learn more about it!

Exercise: where are the bots?

The CSIE workstations lives in the dangerous open Internet. Everyday, we experience countless password brute force attack from hundreds of bots around the world. Let's find out where they are from.

You will be given a list of IPs, with their geo-locations. You task is to transform it into a specific format suitable to be plotted on an online map.

The format of each line is:

```
[latitude];[longitude];[IP];[organization name]
```

If "organization name" is empty, please output "Unknown".

Exercise: where are the bots?

Material: https://www.csie.ntu.edu.tw/~vegetable/exercise_1.tar.gz
Online map: http://www.csie.ntu.edu.tw/~yunchih/map

Prepare environment:

```
$ sudo apt update && sudo apt install xsel -y
$ wget https://www.csie.ntu.edu.tw/~vegetable/exercise_1.tar.gz
$ tar zxf exercise_1.tar.gz
```

Copy output to clipboard + save to a file:

```
$ ./parser.sh | tee output.txt | xsel -ib
```

Test correctness:

```
$ md5sum output.txt
3ab442e3a6c0f8b9700706871545f83c  output.txt
```