Introduction to Computing

Bash Scripts & Interesting Programs

Cain Susko

2023-06-28

What is a `Script`

`Scripting`, on computers, is when you – the user – writes down a list of things for your computer to do. `Scripts` are called *scripts* and not *programs* because they only automate the use of other programs (like the ones we talk about later) – however if a script becomes large enough, it could be called a program.

For example, consider the following script:

```
echo 'Hello, my name is scripty !'
```

Where `echo` is a *program* used by this script to help automate printing some text to the screen.

Regular Expressions

Of course, there are much more interesting programs & tools to use in our scripts other than `echo`. One extremely useful program that nearly every aspiring computer-nerd should know is `grep`.

`grep` stands for *Global Regular Expression search and Print*. But – what in the world is a "Regular Expression"?

Regular expressions are similar to variables in math, like 'x' or 'y', where you can write that 'x>10' (which means 'x must be greater than 10'). Therefore 'x' could be 11 (11 IS greater than 10), but 'x' could NOT be 9 (9 is IS NOT greater than 10)

Regular Expressions do the same thing, but for WORDS (or strings, as programmers like to call them). So, like in math how we have symbols like '>' (greater than) and '<' (less than), regular expressions also have special symbols

```
^ = start of a line
$ = end of a line
. = any letter
```

But how is this useful ? Well, lets suppose we have a `directory` with hundreds of different files inside of it, but we want to know if a specific one is in there. We could manually list the files using `ls` and search with our eyes, however this would take a while. So instead, we should use `grep` to search through our list of files.

Lets suppose we want to find a file called `greg.sh`. To search for it in our current directory using `grep`, we could use the command:

```
ls | grep ^greg
```

Where `|` is called *pipe*, and "pipes" the output of `ls` to `grep` instead of printing it to the screen. This allows `grep` to search through the list of files and then print the ones that match our expression:

```
^greg

which means, lines that must start with "greg"
```

This command will output any lines that start with `greg` – so, if we also had a file called `greg.txt`, it would also be printed by the `grep` command

However, if we wanted to make sure that *only* `greg.sh` is output by the `grep` command, then we would use the command:

```
ls | grep ^greg\.sh$
```

Where the `\.` is used because in regular expressions, `.` means any letter, but we just want a normal '.'.

Examples

```
^greg           = lines that must start with 'greg'
x$              = lines that must end with 'x'
^greg\.sh$      = lines that only contain 'greg.sh'
^\.\.\.$        = lines that only contains '...'
^...$           = lines that only contain 3 letters
^a.c$           = lines that must start with 'a', have any letter
                  in the middle, and end with 'c'

abc             equals ^a.c$
abbc    does not equal ^a.c$
```

Exercises

0. Make a Regular Expression that matches your name
1. Search for a file a directory using `grep`
2. make a file using `touch` and `nano` that contains

   ```
   ostechnix
   Ostechnix
   o&technix
   linux
   linus
   unix
   technology
   hello world
   HELLO world
   ```

   and then make a script that will find all lines with 'nix' in it Hint: use the command `grep nix my new file`.
3. Make it so that your new script can take an input and search for it in a file