Unix tools

Tuesday, February 16, 2016 4:29 PM

GENERAL

Misc stuff

- "usr" stands for "Unix System Resources", not "user". 😁

Terminals with spammy output

I don't know where else to put this, but ctrl+S will usually freeze the terminal's output so that it doesn't spam so much, then pressing ctrl+C will let you terminate it without having to fight for priority. Afterward, you may need to press ctrl+Q to unfreeze the output. Tenbroya shared this tip with me.

cdpath

Didn't actually look into this yet, so here are some notes from kensodev:

http://www.theunixschool.com/2012/04/what-is-cdpath.html
11:43 kensodev: @Adam13531 look up $CDPATH
11:43 kensodev: when you do cd some-folder
11:43 kensodev: it will always do the right thing
11:43 kensodev: even when you are in that folder already
11:44 kensodev: say you have `~/Code/adam`
11:44 kensodev: when you do `cd adam`
11:44 kensodev: it will look for that in Code
11:44 kensodev: even if you are in a completely different dir
11:44 kensodev: very useful

SCP (secure cp) (reference)

Putting a file on a remote machine:
scp -i D:\Code\JavaScript\learning\aws\Firstkeypair.pem a.txt ec2-user@52.36.170.141:/home/ec2-user

Getting a file from a remote machine is basically the same general way:

scp root@104.236.198.88:/root/BotLand.zip ./

I couldn't figure out how to get directories with spaces in them, so if that happens to you, just move the target file first to a directory without a space.

Note: actually formatted for AWS:

scp -i D:\Code\JavaScript\learning\aws\Firstkeypair.pem admin@35.163.71.233:/home/admin/b.txt ./

Caveats:

- On Windows, you can't seem to specify paths with "C:" in them, otherwise you'll get an error that says "ssh: Could not resolve hostname c: Name or service not known". I fixed this by just always using relative paths. You can further improve this by just using "CD /D <absolute path to folder>" right before using a relative path.
- You cannot use SCP just to create folders on the target machine like you can with "mkdir -p", so you should use SSH just before it, e.g.
  ssh -i D:\Code\JavaScript\learning\aws\Firstkeypair.pem ec2-user@52.36.170.141 "mkdir -p

hello/world"

Note: ideally you would specify an absolute path.

chmod

chmod 1777 /var/tmp

The "1" will set a bit which only allows the owner of a file to delete it. So when you "ls -al /var/tmp", you'll see "drwxrwxrwt" listed.

If you try to use a private key file via "ssh-add" and you see an error that says "WARNING: UNPROTECTED PRIVATE KEY FILE!", then you should probably do "chmod 0600 <file>".

chown

Changes owner of a file. This is useful if you had to copy a file from another user's home folder on your machine.

sudo chown adam <file>

curl

"curl -X POST" will force a post request so that you don't need to provide an empty 'data' argument like this: curl --data "" url.com

ls

"ls -h" will print human-readable file sizes.

Apparently some people will alias "ls" to "ls -al" so that you don't need to type something like "ls -alh" every time you want to see human-readable sizes.

sudo

Example: sudo -iu bldeploy

- "-u" specifies the user.
- "-i" (lowercase "eye") specifies that you want to run in an environment similar to if you'd logged in as that user. This will look at /etc/passwd to find out your login shell, then based on the shell will "source" your various profiles, e.g. ~/.bash_profile.
- This is much easier than doing "sudo su bldeploy", because then you would need to run "bash --login" afterward.

netstat

Use this to figure out what is open on which ports: "netstat -tulnp". Use it with "sudo" to see the program names.

watch

Continually runs/prints the output of a command:

- Run "ls" every 3 seconds:
  - watch -n 3 ls

htpasswd

This is installed via apache2-utils (reference). With this tool, you can generate htpasswd files for use in authentication. These files can technically be shared publicly because they're salted/hashed passwords.

To generate a new file:

htpasswd -n -B -C 20 bldeploy

-n: display results on stdout
-B: use bcrypt (very secure)
-C: set computing time used for the bcrypt algorithm. Higher is slower but more secure.
bldeploy: the username to generate the password for

After doing this, make sure to clear your Bash history!

To verify a password that you've set:

1. Save the htpasswd file as something (e.g. just a file named "p")
2. htpasswd -bv -C 6 ./p bldeploy <password from the generation step>

sed (stream editor) (reference)

Basic usage:

sed -i 's/original/new/g' file.txt

grep

Basic searching

grep -r -n --color --exclude={bundle*} --exclude-dir={node_modules,.git,dist,logs} "lodash" ./

- -r: recursive
- -n: show line number
- --color: show colors in output
- --exclude: exclude **files**
- --exclude-dir: exclude **directories**
- "search term": the thing to search for
- ./: the directory to start the search in
- -i: case insensitive (off by default, meaning searches *are* case-sensitive)

Back references (reference)

I had something like this:

FOO: 'FOO' <-- I want to find that
FOO: 'BAR' <-- I don't want to find that
BAZ: 'BAZ' <-- I want to find that

This involved using a backreference:

grep -E "(\w+): '\1" ./main.js

The "-E" is to use extended regex

Omitting results

"grep -v" inverts all of the matches. For example, if you do a search and it returns 1 result from DirA and 100 results from DirB and you only care about DirA, you can do

<original grep command> | grep -v DirB

Getting colorful results by default

You *used* to be able to set an environment variable that would add "--color=auto" to every grep command, but they deprecated that. Now, you have to make an alias that has the option specified by default:

alias grep="/usr/bin/grep --color=auto"