Linux 101 Command Line Cheat Sheet

Abstract

Fundamental Linux/Unix commands for the Linux/Unix command line learner. If you are experienced with Linux/Unix: you have probably mastered these commands. If not: you are in the right place.

Note: Some of the examples below presume files and paths that might not match your particular system and tool installation.

Where to Acquire

These tools are installed natively in most Unix/Linux distributions, as well as OS X.

Examples/Use Case

- bash basics
- cat
- cd
- echo
- ls
- networking
- passwd
- ping
- pwd
- sudo

---

bash basics

Tab-completion:

Folks who are new to the Unix/Linux command line often attempt to type everything by hand. This may work well if you type quickly and accurately. Most of us are **much** better off using tab completion.

Note that Windows PowerShell also supports tab completion, but it handles ambiguity differently. See the PowerShell cheat sheet for more information.

Type the following, and then press the <TAB> key:

```
$ cat /etc/pas
```

Then press <TAB>.

Note that it autocompletes to /etc/passwd.

Now try tabbing with ambiguity:

```
$ cd ~/Do
```

Then press <TAB><TAB>.

Note that it offers two choices: Documents/ Downloads/.

Now add a "w" and press <TAB>:

```
$ cd ~/Dow
```

Press <TAB>. It autocompletes to ~/Downloads/.

cat

Display a file:

```
$ cat example.txt
```

Concatenate (cat) FileA.txt and FileB.txt, create FileC.txt:

```
$ cat FileA.txt FileB.txt > FileC.txt
```

---

cd

Change Directory (cd) to the /tmp directory:

```
$ cd /tmp
```

Change to the home directory. The following commands are equivalent for the "student" user: "~" means home directory (for example: /home/student):

```
$ cd
$ cd ~
$ cd /home/student
```

Change to the parent directory. For example: if you are in /tmp/subdirectory/, this will change your working directory to /tmp/:

```
$ cd ..
```

---

echo

Print (echo) the string "Cylon":

```
$ echo Cylon
```

Create or overwrite the file example.txt, containing the string "Cylon":

```
$ echo Cylon > example.txt
```

Append the string "Cylon" to the file example.txt:

```
$ echo Cylon >> example.txt
```

---

ls

List the files in the current directory (equivalent to the cmd.exe "dir" command):

```
$ ls
```

List the files in the current directory, long output (-l), all files including "hidden" files that begin with a "." (-a):

```
$ ls -la
```

List the files in the current directory, long output (-l), all files (-a), sort by time (-t):

```
$ ls -lat
```

List the files in the current directory, long output (-l), all files (-a), reverse (-r) sort by time (-t):

```
$ ls -lart
```

---

networking

Show network interface configuration:

```
$ ifconfig
```

Show network interface configuration using "ip":

```
$ ip a
```

Restart networking:

```
$ sudo /etc/init.d/networking restart
```

---

passwd

Change your password:

```
$ passwd
```

---

ping

ping a host forever (until CTRL-C is pressed), see if it is up (and unfiltered):

```
$ ping 10.5.11.25
```

ping a host 3 times, see if it is up (and unfiltered):

```
$ ping -c3 10.5.11.25
```

---

pwd

Print Working Directory (pwd), show the current directory:

```
$ pwd
```

---

sudo

Run a command as root:

```
$ sudo command
```

Open a root bash shell:

```
$ sudo bash
```

Additional Info

A printable PDF version of this cheatsheet is available here:
LinuxCLI101

Cheat Sheet Version

Version 1.0

Linux Command Line Cheat Sheet

Abstract

The following examples may be typed in the terminal, but copy/paste will work fine (be sure to omit the prompt).

- To copy in Firefox: press CTRL-C
- To paste into a terminal: press SHIFT-CTRL-V (or Edit->Paste)

Many of these examples will use the "cat example.txt | command" syntax. This is safer than the equivalent syntax of "command < example.txt".

Why? Most everyone learning the Unix/Linux commandline has accidentally reversed the " <" sign (read) with the ">" sign (write), accidentally overwriting a file. The syntax of "cat example.txt | command" is therefore safer. Please feel free to use whatever syntax you are most comfortable with.

On a related note, "There is more than one way to do it," as Larry Wall once said. You may come up with different ways to perform the following, and perhaps better ways as well. Feel free to share your CLI Kung Fu with us for possible inclusion!

Where to Acquire

These tools are installed natively in most Unix/Linux distributions, as well as OS X.

Examples/Use Case

- awk
- checksums
- cut
- file
- grep
- head
- sed
- sort
- wc
- xxd

---

awk

Print the length of each line of a file (/etc/passwd in this case), followed by the line itself:

```
$ cat /etc/passwd | awk '{print length, $0;}'
```

Print the 2nd field from a file using the string 'Mozilla/' as a delimiter:

```
$ cat /var/log/apache2/access.log | awk -F "Mozilla/" '{print
$2}'
```

Print the last period delimited field

```
$ cat domains.txt | awk -F "." '{print $(NF)}'
```

---

checksums

Generate the MD5 checksum of a file:

```
$ md5sum /etc/passwd
```

Generate the SHA1 checksum of a file. The three following commands are equivalent:

```
$ sha1sum /etc/passwd
$ shasum /etc/passwd
$ shasum -a1 /etc/passwd
```

Generate the SHA-256 checksum of a file:

```
$ shasum -a256 /etc/passwd
```

Generate the SHA-512 checksum of a file:

```
$ shasum -a512 /etc/passwd
```

---

cut

Cut the 2nd field from a file, using the space as a delimiter:

```
$ cat /var/log/dpkg.log | cut -d' ' -f2
```

Cut the 6th field from a file, using the colon as a delimiter:

```
$ cat /etc/passwd | cut -d: -f6
```

Cut the 2nd and 3rd field from a file, use the comma as a delimiter:

```
$ cat /labs/honeytokens/pilots.csv | cut -d, -f2-3
```

Cut beginning at the 7th field, to end of line, using the space as a delimiter:

```
$ cat /var/log/dpkg.log | cut -d' ' -f7-
```

Cut the 6th field, using the double-quote (") as a delimiter, and escaping it to treat it as a literal character:

```
$  cat  /var/log/apache2/access.log  | cut -d\" -f6
```

Cut the beginning at the 11th character, to end of line:

```
$ ifconfig | cut -c11-
```

---

file

Determine the file type, using the file's magic bytes:

```
$ file /usr/local/bin/*
```

---

grep

Search for lines containing the string "bash", case sensitive:

```
$ grep bash /etc/passwd
```

Search for lines containing the string "bash", case insensitive:

```
$ grep -i bash /etc/passwd
```

Search for lines that do not contain the string "bash", case insensitive:

```
$ grep -vi bash /etc/passwd
```

Search for lines containing the string "root", case sensitive, plus print the next 5 lines:

```
$ grep -A5 root /etc/passwd
```

---

head

Print the first 10 lines of a file:

```
$ head -n 10 /etc/passwd
```

---

sed

grep for lines containing "Mozilla", then change "Mozilla" to "MosaicKilla":

```
$ grep Mozilla /var/log/apache2/access.log | sed
"s/Mozilla/MosaicKilla/g"
```

grep for lines containing "Mozilla", then delete all characters up to and including "Mozilla":

```
$ grep Mozilla /var/log/apache2/access.log | sed
"s/^.*Mozilla//g"
```

grep for lines containing "Mozilla", then delete all characters that precede "Mozilla":

```
$ grep Mozilla /var/log/apache2/access.log | sed
"s/^.*Mozilla/Mozilla/g"
```

---

sort

The following examples will run strings on a file, search for user-agent (ignore case), and use various sort options

Simple alphabetic sort (may include duplicates)

```
$ strings /pcaps/fraudpack.pcap | grep -i user-agent | sort
```

Sort and unique lines. The two following sets of commands are equivalent:

```
$ strings /pcaps/fraudpack.pcap | grep -i user-agent | sort -u
$ strings /pcaps/fraudpack.pcap | grep -i user-agent | sort |
uniq
```

Get a numeric count of each unique entry:

```
$ strings /pcaps/fraudpack.pcap | grep -i user-agent | sort |
uniq -c
```

Get a numeric count of each unique entry, perform a numeric sort of that count:

```
$ strings /pcaps/fraudpack.pcap | grep -i user-agent | sort |
uniq -c | sort -n
```

Sort and unique lines, print the length of each unique line followed by the line itself, perform a reverse numeric sort of that count:

```
$ strings /pcaps/fraudpack.pcap | grep -i user-agent | sort -u |
awk '{print length, $0}' | sort -rn
```

Sort on the the second comma separated field

```
$ cat /bonus/alexa/top-1m.csv sort -t, -k2
```

```
```

---

wc

Determine number of lines in a file (the flag is the letter "ell", not the number one):

```
$ wc -l /etc/passwd
```

xxd

xxd creates a hexdump, or converts a hexdump into binary. A lot of malware hex-encodes web traffic or malicious payloads (such as DOS executables) in order to avoid signature matching. Useful hex patterns to look for are 4d5a90 (the magic bytes for a DOS executable: "MZ<90>"), and "DOS mode" (444f53206d6f6465, see commands below).

xxd cannot natively handle percent-encoded hex, such as "%63%67%69%2D%62%69%6E", but can if the percent signs are removed (see below).

Convert the string "DOS mode" to hex, grouped in sets of 4 hex characters (default):

```
$ echo -n "DOS mode" | xxd
0000000: 444f 5320 6d6f 6465                      DOS mode
```

Convert the string "DOS mode" to hex, ungrouped:

```
$ echo -n "DOS mode" | xxd -g0
0000000: 444f53206d6f6465                  DOS mode
```

Convert the hex string "444f53206d6f6465" to binary:

```
$ echo 444f53206d6f6465 | xxd -r -p
DOS mode
```

Use sed to remove the percent signs from the percent-encoded hex string "%63%67%69%2D%62%69%6E", then translate to binary:

```
echo "%63%67%69%2D%62%69%6E" | sed "s/\%//g" | xxd -r -p
cgi-bin
```

Additional Info

A printable PDF version of this cheatsheet is available here:
LinuxCLI

Cheat Sheet Version

Version 1.0

Windows and Linux Terminals & Command Lines

Tools and Tips for SEC301 and SEC401

Getting started:

- **c:\>** denotes a command to be run from Windows' cmd.exe
- **user$** is for a Linux command
- **root#** means the Linux command needs to be run as a privileged, root user
- Linux commands are generally case-sensitive; Windows commands are generally not
- Mac terminals will, in most ways, act like Linux/Bash terminals

---

What directory am I in?

```
c:\>  cd
user$ pwd
```

What files are in this directory?

```
c:\>  dir
user$ ls -l
```

Copy a file

```
c:\>  copy file.txt copy.txt
user$ cp file.txt copy.txt
```

Erase a file

```
c:\>  erase file.txt
user$ rm file.txt
```

Print the contents of a file to the screen

```
c:\>  type file.txt
user$ cat file.txt  (Just dump the raw file)
user$ strings file.txt
(Dump only the readable characters)
```

See one screen at a time

```
c:\>  type file.txt | more
user$ cat file.txt | more
user$ more file.txt  (Same, just shorter)
user$ less file.txt  (Same, but you can go up
and down; q to quit)
```

Put text into a file

```
c:\>  echo “Four score” > 1.txt
user$ echo “Four score” > 1.txt
```

Add text to a file

```
c:\>  echo “and seven years” >> 1.txt
user$ echo “and seven years” >> 1.txt
```

Combine two files

```
c:\>  type 1.txt 2.txt > 3.txt
user$ cat 1.txt 2.txt > 3.txt
```

Check who you're logged in as

```
c:\>  whoami
user$ whoami
```

Hide command error messages

```
c:\> YOUR COMMAND 2>nul
user$ YOUR COMMAND 2>/dev/null
```

Find files in a filesystem

```
c:\> dir c:\ /b/s | find “password”
user$ find / -name *password*
user$ locate password (same, but faster)
root# updatedb (update the database for
locate by indexing everything in the drive)
```

View all environment variables

```
c:\>  set
user$ env
```

View one environment variable

```
c:\>  set Path
c:\>  echo %Path%
user$ env | grep PATH
user$ echo $PATH
```

What are environment variables?

They give your terminal context for running certain commands. For example, the PATH variable, in most operation systems, tells your terminal which directories to look in for programs when you type one in.

Note: the current directory, . (period), is in the Windows path by default - but not in Linux. So in Linux, we must be explicit when running something in our current working directory:

Run john when it's in your directory

```
c:\>  john.exe
user$ ./john
```

SANS The most trusted source for cybersecurity training, certifications, degrees, and research

See which ports the computer is listening for connections on

```
c:\>  netstat -nao
c:\>  netstat -naob (Same, but lists process
name; requires Administrator)
user$ netstat -ant
root# netstat -pant  (Same, but lists pid and
name; requires root)
```

Look for lines containing specific text, e.g. 9999

```
c:\>  netstat -naob | find 9999
root# user$ netstat -pant | grep 9999
```

See what tasks are running

```
c:\>  tasklist
c:\>  wmic process list full (Same, more info)
user$ ps -aux
```

Get more info about a specific process id, e.g. 45

```
c:\>  wmic process where ProcessID=45
user$ ps -Flww -p 45
```

Check the system's hostname

```
c:\>  hostname
user$ hostname
```

List processes that run at startup

```
c:\>  wmic startup full list
user$ ls -l /etc/init.d
user$ crontab -l
user$ systemctl list-unit-files | grep
enabled
user$ less /home/user/.bashrc
```

(There are other places where startup tasks can be stored in Linux, but these are the most common)

Scan a host to look for open ports, e.g. 192.168.1.100

```
c:\>  nmap 192.168.1.100
user$ nmap 192.168.1.100
```

Scan a subnet of hosts and see what is really running on open ports

```
c:\>  nmap 192.168.1.1-254 -sV
user$ nmap 192.168.1.1-254 -sV
```

Scan all 65,536 ports on a given host

```
c:\>  nmap 192.168.1.100 -p0-65535
user$ nmap 192.168.1.100 -p0-65535
```

Ping another host four times

```
c:\>  ping 192.168.1.200
user$ ping -c4 192.168.1.200
```

Connect to port 25 to see what banner it sends back, e.g. SMTP or 25/TCP

```
c:\>  nc.exe 192.168.1.100 25
(Not installed by default)
user$ nc 192.168.1.100 25
(Usually available)
```

See your IP address(es)

```
c:\>  ipconfig
user$ ip addr
```

Get help for a command (these work for most commands)

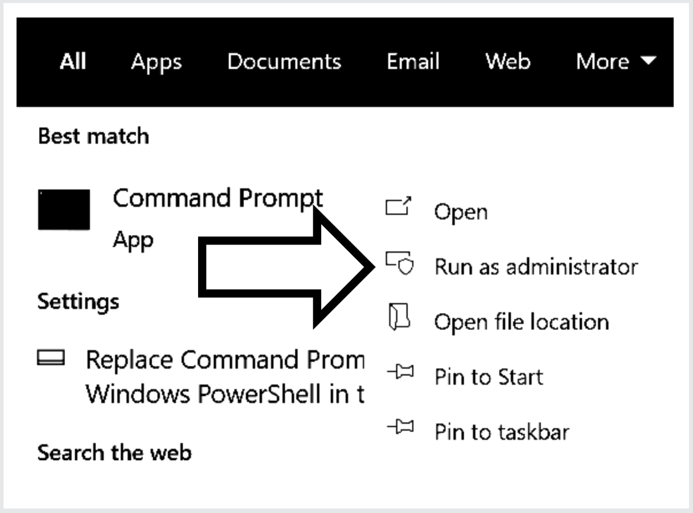
```
c:\>  cd /?
user$ man cd (quit with q)
user$ cd --help (shorter output)
```

See what path executables will run from

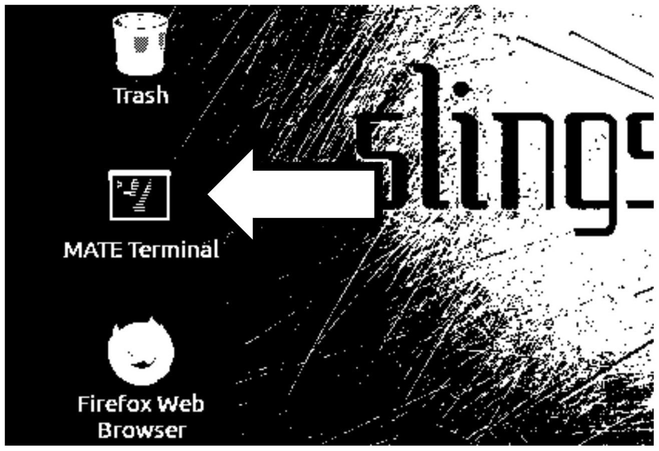
```
c:\>  echo %PATH%
user$ echo $PATH
```

Opening a Command Prompt/Terminal

In Windows, reach the cmd.exe command prompt by clicking the Windows button and typing **cmd**. If you need to run as a privileged user, right-click **Command Prompt** and choose **Run as administrator**.



In Slingshot Linux, open a Bash terminal by double-clicking **MATE Terminal** on the desktop. In some Linux systems, you can use **<Ctrl><Alt>t** as a keyboard shortcut. Use **su** to switch to a privileged/root user or **sudo YOUR COMMAND** to run a single command as root.



The most trusted source for cybersecurity training, certifications, degrees, and research