# Linux Cheat Sheet

Login Username: ~~root~~ training

Login Password: password

## Notes

1. You can pipe the output of one command into another using "|" (without quotes).

   Example: $ cmd1 | cmd2

2. You can redirect output to a file using the ">" or ">>" operator. ">" clobbers the file's content (will replace an existing file content with the new content) and ">>" appends the output to the file.

   Example 1: $ history > file.txt

   Example 2: $ cat *.dmp >> ../bigfile.dmp

3. Everything is a file on a Linux system.
4. File extensions are meaningless.
5. Directories are separated by forward slashes "/"
6. **Absolute Path** is the path of a file starting from the root of the filesystem (/).
   Example: /home/user/file.txt
7. Relative Path is the path of a file from the current directory.
   Example 1: file.txt
   Example 2: ../../file.txt
8. Files with spaces either need to be escaped with a backslash or put in quotes when referencing them:
   "/this is a/file with spaces"
   /this\ is\ a/file\ with\ spaces

~ Represents the user's home directory

. Represents the current directory

.. Represents the parent directory (one level higher than the current directory)

# Common Commands

· *cat* – concatenate and print files. Prints files to standard out.

The following prints out a file and pipes it to the less command for examination:
$ cat file | less

· *cd* – change directory. Navigate to a specified directory

The following changes to the user's (user) desktop directory:
$ cd /home/user/Desktop

The following changes to the user's home directory:
$ cd ~

The following changes to the parent directory:
$ cd ..

The following changes to two directories higher than the parent directory:
$ cd ../../..

· *clear* – clear the screen

$ clear

· *cp* – copy a file or files

Copy file.txt to a folder called newlocation:
$ cp file.txt /path/to/newlocation

Copy file.txt back two directories:
$ cp file.txt ../..

Recursively copy a folder to a newlocation:
$ cp –R folder /path/to/newlocation

· *find* – find a file or files General usage:
$ find [location] [options]

Find a file on the system named (case sensitive) "file.txt":
$ find / -name file.txt

Find a file on the sytem named (case insensitive) "file.txt":
$ find / -iname file.txt

Find a file on the system that has an extension ".exe":
$ find / -iname *.exe

Find all text files in a folder and delete them:
$ find . -iname *.txt –exec rm `{}` \;

· *grep/ egrep* – print matching lines in a file for some pattern

By default grep/egrep will print out the lines that match a particular pattern. There are several options that you can use with these commands and you can also combine them as needed. The options shown here can be used with either command. The following is a simple grep statement for a pattern (replace "pattern" with the pattern you want to search for) within a file called file.txt:
$ grep pattern file.txt

The following is a grep statement to find a case insensitive pattern within file.txt:
$ grep –i pattern file.txt

The "-v" option of grep allows one to find all lines that do NOT contain the pattern specified:
$ grep –v pattern file.txt

The recursive option ("-R") allows one to search through all files in the current directory for a particular, non-case sensitive pattern.
$ grep –iR pattern file.txt

The list option ("-l") allows one to only list files that match a particular pattern:
$ grep –l pattern file.txt

The list option ("-c") allows one to count the number of lines that match a particular pattern:
$ grep –c pattern file.txt

The egrep command allows one to match more than one pattern as shown below:
$ egrep "(pattern1|pattern2|pattern3)" file.txt

· *history* – Show past typed commands Basic usage:
$ history | grep "some command"

Clear history:
$ history -c

· *less* – allows one to page through output.

This command is often used with output piped from another command:
$ grep –Ri pattern folder | less

This command can also be used directly against a file:
$ less file.txt

- *ls* – list out a directory

  The following lists out the current directory:
  $ ls

  The following lists out a particular directory:
  $ ls /path/to/particular/directory

  The following lists out all files/subdirectories of a particular directory:
  $ ls –R /path/to/particular/directory

- *man* – obtain information about a command

  $ man [cmd]

- *mkdir* – create a new directory

  $ mkdir [newdirectory name]

- *mv* – move a file/folder

  $ mv file.txt /path/to/new/location

- *pwd* – show the present working directory

  $ pwd

- *rm* – remove a file/or files

  Removing one file:

  $ rm file.txt

  The "-f" option suppresses messages when deleting files:

  $ rm –f file.txt

  Removing an entire directory (BE CAREFUL WHAT YOU REMOVE):

  $ rm –Rf junkdirectory

- *rmdir* – remove a directory

$ rmdir directory

- *strings* – retrieve the printable strings from a file

  The following retrieves all printable ASCII strings:
  $ strings –a file > strings-output.txt

  The following retrieves all printable Unicode strings:
  $ strings –a –e l file > strings-output.txt

  Note: You can use the –t flag as "-t d" to recover the offset of each string along with the string

- *tree* – show the entire tree of a folder its files/subfolders

- *xxd* – show a hexdump of the file

  $ xxd file.txt

  $ [some command] file.txt | xxd

- *wc* – count the number of words/lines/characters/bytes in a file

  The following counts the number of lines in a file:
  $ wc –l file.txt

# Windows vs Linux Commands

| Windows | Linux |
|---|---|
| help [cmd] | man [cmd] |
| dir | ls |
| cd | pwd |
| cd [path] | cd [path] |
| cls | clear |
| copy [old] [new] | cp [old] [new] |
| del [file] | rm [file] |
| move [old] [new] | mv [old] [new] |
| tree | tree ls -R |
| mkdir [dir] | mkdir [dir] |
| rmdir [dir] | rmdir [dir] |
| type [file] | cat [file] |
| netstat –an | netstat –an |
| ipconfig /all | ifconfig -a |