Chapter 3 Review

computers process in binary. All data is stored as 0's and 1's.

a cpu has a set of instructions or operation codes (opcodes) that correspond to functions for the cpu to execute. The opcodes are represented as binary numbers. When an instruction is received by the cpu, the cpu performs actions such as moving or storing data or adding numbers together.

Files on the operating system store data. All files consist of binary data. The binary data in some files is structured so that the 0's and 1's can be converted into human readable text. This is called a text file.

8-bit strings of 0's and 1's are converted into human readable characters by means of a look up table, either ASCII or Unicode. ASCII has 256 characters while Unicode has 65,536.

Binary files are files that are not structured to be converted into human readable text using a character encoding/decoding scheme. These files can often be image, audio, executables, or other project files for saving work in a software program.

Some files are executable. The are files that can be run from the command line as computer programs. Executable files can be binary or text files.

Binary executables are composed of a sequence of machine readable instructions that can be directly consumed by the CPU. Text file executables are in a human readable form and require an interpreter (usually from a scripting language such as Python) to be read and converted into instructions that can be sent to the CPU.

Files are made executable in Unix when their executable bit (x) is set.

Any readable file is a text file, even files ending in .html, .xml, .json, .log, .sh, .sql, .cpp, .py, .php., .css, .js, .rs, etc. These are all usually text files.

A text editor is a program that edits text files.

Line editors are text editors that edit text files by altering a file line by line when a user issues commands at the terminal. The contents of the file isn't displayed on screen all at once.

A screen editor is a text editor where all of the text appears on screen and can be edited in more visual ways such as by moving a cursor around the screen.

Both line editors and screen editors are useful for editing text files. Word processing programs should not be used as they introduce special formatting that is not a part of the basic text file. Word processors often save files in binary formats also. Saving as text can prevent this, but often times, a word processor still injects unnecessary formatting into the document. For this reason, system administrators often stick with editors specifically designed for editing text files only.

Some common terminal text editors in use: vi, emacs, nano

Terminal text editors are popular because they can be used easily, even if you are remotely ssh-ing into a machine. They are efficient to operate over a network.

There are GUI versions of the terminal editors and also a growing number of original GUI editors such as: Gvim, Atom, and Sublime text.

It is good to be versatile and capable of working in several text editors in order to handle the various unique circumstances you may find yourself in on the job.

We will focus on Vi in this class.

To open vi, you simply type vi

You can also add a parameter after for the name of the file you would like to open: vi mytext.txt

You can move the cursor around in vi using the keys: h, j, k, and l.

H, M, and L can move you to the top bottom or middle of the screen

G can take you to the last line of a file.

nG where n is a number can take you to a particular line in the file

0 takes you to the beginning of a line

$ takes you to the end of a line

W will move you one word forward

b will move you one word backwards

( ) moves the cursor forward or backward one sentence

{ } moves the cursor forward or backward one paragraph

% move to corresponding grouping symbol ( ), { }, [ ]

Ctrl+f forward one screen

Ctrl+b backwards one screen

`` move back to previous position before last move (back ticks)

typing a number before any of the above commands makes the command happen that many times

d – initiates the delete command, must be combined with a movement command after it in order to determine what/how much to delete

x – delete a single characters

dd – delete a line

r – replace a single character, must be followed by the character to replace the current one with

R – enter a replace mode that continues until you hit escape

i – enter insert mode to type text

A – move to end of line and enter insert mode

I – move to beginning of line and enter insert mode

o – insert line below current line and enter insert mode

O – insert line above current line and enter insert mode

. - repeat last action

u – undo

Ctrl-r - redo

ESC – enter command mode

ESC : - enter ex mode (extended mode, based off the ex line editor)

v – enter visual mode

Ctrl-v – enter visual block mode

ESC : q – exit vi

ESC : w – save file

ESC : q! - exit without saving changes

/ - enter a search mode to look for text

ESC : %s/word/replace/g

y – while in visual mode, copies text

c – cut text while in visual mode

p – paste text