Intro to Linux: Consolidation exercises

Aim to build familiarity with using the terminal, running programs, managing files, building pipelines, and the key concepts of a Unix system

1. Get a file
   1. Open a terminal
   2. Create a directory to work in (use mkdir)
   3. Download <https://raw.githubusercontent.com/itlfiles/exercises/main/UNdata_GDP_2022.txt> and save it into your directory (use wget / Firefox / Chromium)
   4. Check the MD5 hash is 6d3e5b88df5bb938296f4423effce0e6 (use md5sum)
2. Look at the data
   1. Find out how the data is formatted by looking at it (e.g. with less or cat or head or …)
   2. What country/area name is data from Yugoslavia recorded under (use grep)?
   3. How many different countries are represented in the data (use cut to extract fields, uniq to remove duplicate lines, and wc to count things)?
   4. [Harder] Some country names contain special characters (e.g. Côte d'Ivoire, Curaçao) - how are these characters stored in the file (use xxd)? What settings control the interpretation of these characters (the phrase ‘character encoding’ may help for searching)?
3. The data is in a slightly inconvenient format for processing - most programs work by default on space-separated fields, rather than the pipe-delimited double-quoted fields in the file
   1. Transform the data to space-separated, taking care not to accidentally split fields that contain a space. You may want to first replace space characters with underscores. (use tr)
   2. Find the country and year with the highest GDP per capita (use sort)
   3. Find the top 10 countries ranked by GDP per capita in 2015 (for line selection, use awk, or use grep but be careful)
   4. Download a Python release from <https://www.python.org/downloads/> as a source tarball. A tarball is an archive format (i.e. a file that contains multiple files) that is often used with a file compression tool such as gzip or xz
   5. Untar it (probably via tar -xf filename), which will create a directory of Python source code files
   6. Find the 10 largest files in the directory, which could be at any depth - e.g. in Python 3.11.0 the 10th largest file is ./Lib/pydoc\_data/topics.py (use find)
   7. Find duplicated files in the directory, again at any level (use find and e.g. md5sum)
4. Write shell scripts to:
   1. Search the dictionary for words matching a regex
   2. Add a suffix to a list of filenames (e.g. to rename filename to filename.old)
   3. Sort a file by line length
5. Use a grep pipeline to find a word that is an anagram of cceimoooprrrss. Write a shell script to find anagrams of a string argument.  
   Tip: write a shell script to automate writing the grep pipeline, then pipe it into bash to run it.
   1. Investigate how the terminal displays colours - try e.g. ls --color | xxd. The Wikipedia page <https://en.wikipedia.org/wiki/ANSI_escape_code> may help.
   2. The bash prompt is controlled by the variable PS1 (see the bash manual page section PROMPTING for more details). Customise the bash prompt to display in a random colour when printed.
6. Write a shell script to output the date and free disk space every 5 seconds (use date and df). Run the script in the background using nohup to protect it from SIGHUP, and watch the output in nohup.out with tail -f. Find the PID of the process and kill it (use ps and kill).
7. Use command-line programs to decrypt this Vigenere-encrypted text:  
   Fvx qm blv uhua qrkmkgzbmeo gqaqsea bp api wqeg zgwkmf kz blv aigjqec nbnl. Kiibtku nmcml fv vsk zxhlz xf lbur nmcml ca ipc, jnv aw M/F lxxpkij. Jr evvzvvmkvv, wlka uwmgzie hptij zxupli zv t rhzxzknnhz hzzxeawvp, ievowyxp mjpa mj vhv umgvalcyg. Aymg c zxitqtn mqpv ql tlih fz ptpbxvv, mjl limqvg pb vvnxtz bs za teaqzrbxf. Mwv vftowti, rte vom gfufwuqgrbbqua mebxtmigva tvaigymw vv bcgmptpbiia accm wgmvkht jztxu hawfkbcamh nqmj apid. Bawz, xvfdbfll cfc accm tvzfkzamfv, tpfwrv ktp zmru i fgzaexm mq hvskpxt baii abowtc sg ptpbmeo bpmwvdimkvv sebh jpa xpxxyyqxvz'l uwmgzie hpti. Kpxtl ivv aigjqec nbnla, jfz xzhutcm, mq ymjvz mq api giigy begm kghlii igf wcrtp, mq api 201 uimcwpsem, mjl kseahnl bcgmptpbii, igf dpekmogy wxymk fldmtml ohg fv wg vom wpamgt. Ir vnyqyb mj utfl bs didg apijm lrlkmrt yksmw smaccm ioivvsg xym lctm arg mjhb silbphzc uqlm mqpva ugoizv. Bakz uirvl voix gzhiyiqj oxplzectr fv vsk vxgk bs bvhy dpikpxt apip ikg ymeuqgi vz aiqmkuo se ahol limqvg vz se i wkzs jztx.  
   Tip: first recover the length of the key by counting characters in columns - the true length will result in fewer different letters (reflecting the lack of uncommon letters in the plaintext). Then set the shift on each column. Note that only letters have been encrypted - punctuation, spaces and numbers do not affect the column number.
8. Solve <https://projecteuler.net/problem=12> using command-line programs (especially factor and awk). Note that factor accepts multiple arguments either on the command line or on stdin.

Extra (hard) exercises

1. Write a shell script to hash directory contents, in the same way that md5sum hashes file contents - i.e. if dir1 and dir2 contain the same files, the hashes must match, and if not the hashes should differ with high probability. You may decide for yourself whether files with different permissions, owners, groups, modification times, etc. count as the same. NB the ‘size’ of a directory is not (necessarily) a function of its contents
2. Write an interactive [hangman](https://en.wikipedia.org/wiki/Hangman_(game)) program. Write another program to play against the first one.
3. Write a program to render images in the terminal. You may find the characters in <https://en.wikipedia.org/wiki/Block_Elements> helpful
4. Wikipedia includes the fairly astonishing regular expression (0|(1(01\*0)\*1))\*, which matches binary numbers that are multiples of 3. Find a regular expression that matches decimal numbers that are multiples of 3. NB there is some interesting maths relating regular expressions to finite automata that may help.

Exercise solutions

Will appear here after they’re presented

1-3: <https://asciinema.org/a/wCg2Kvd6Tr272b5bfs34jaRpa>

4: <https://asciinema.org/a/ta2BcBzfsKX2xozve9PyxcmpE>

These are by no means the only solutions - there are many possible ways to solve the exercises using different programs, options, pipelines, etc.