DAT470/DIT065
Computational techniques for large-scale data

Assignment 1
Deadline: 2024-04-15 23:59

Problem 1 (12 pts)

In this problem, we practice some fundamental UNIX commands and solve simple tasks of analyzing a bunch of files. Each subproblem is worth 1 point. In each subproblem, write in your report the command(s) you used to solve this task, in addition to presenting answers to the questions presented in that subproblem (if any).

(a) Log in to `bayes`.

(b) Create a directory under your home directory called `assignment1_problem1`. Make that directory your working directory (change into that directory).

(c) Download the Linux kernel version 6.6.17 sources from `https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.7.5.tar.xz`.

(d) Extract the contents of the tarball in that directory.

(e) Determine the size of the tarball before extraction and the size of the extracted contents in *human-readable* units.

(f) The Linux Kernel consists of C code. C code is organized into *header files* (file extension `.h`) and *source files* (file extension `.c`). Determine the total number of header and source files in the sources, respectively.

(g) Write a one-liner that determines the 10 *longest* C code files (either a header or a source file) by the number of lines in the file, and stores their names in a file called `longest.txt`. List the names of the files in your answer.

(h) Write a one-liner that reads the names of the files from `longest.txt`, computes the SHA-1 checksum of each file in the list, and outputs the checksums into a file called `longest.sha1sum`, such that it can be used to verify the correctness of these files. Include the content of `longest.sha1sum` in your answer.

(i) Write a one-liner that determines how many *unique* files there are that contain the string `Linus Torvalds`. Note that the string can occur multiple times in a file; only count each file once. Include the number of such files in your answer.

(j) Certain components of the kernel come under different licenses. The C source files (`.c` files) contain a standard comment that identifies the license as described in `Documentation/process/license-rules.rst`: the `.c` files contain a line that looks like
`// SPDX-License-Identifier: <SPDX License Expression>`

For example, the file `linux-6.7.5/lib/random32.c` begins with the following line:
`// SPDX-License-Identifier: GPL-2.0`
Write a one-liner that selects all `.c` files that contain the `SPDX-License-Identifier` and outputs, for each such file, the string *filename license*, and output the content to a file called `licenses.txt`. For example, the entry corresponding to the above-mentioned file would be
`linux-6.7.5/lib/random32.c GPL-2.0`.

(k) Write a one-liner that determines the most common license in `licenses.txt`. Include the license and the count of its occurrences in your answer.

(l) Download the files `longest.txt`, `longest.sha1sum`, and `licenses.txt` from `bayes` to your own computer.

Problem 2: Information of computers (12 pts)

Construct a single shell script (`.sh` file) or a Python script that you can run using Slurm on `bayes`, `markov`, and `shannon`, and use the script to collect the following information about the systems (each subproblem is worth 1 point):

- The model of and the clock frequency of the CPU
- The number of physical CPUs (sockets in use), the number of cores, and the number of hardware threads
- The instruction set architecture of the CPU
- The cache line length
- The amount of L1, L2, and L3 cache
- The amount of system RAM
- The number of GPUs and model of the GPU(s)
- The amount of RAM on the GPU(s)
- The filesystem of `/data` and `/datainbackup`
- The total amount of diskspace and the amount of free space on `/data` and `/datainbackup`
- The version of the Linux kernel running on the system and the GNU/Linux distribution and its version running on the system
- The filename and the version of the default Python 3 interpreter available on the system

Do note that there are differences between the way the disks have been setup on `bayes`, and `markov` and `shannon`, see the DSAI compute infrastructure canvas page for details.

Include the information you gathered in your report.

Hints

- The following commands are probably useful (not an exhaustive list and valid alternatives exist):
  - `cd`
  - `cpuinfo`
  - `cat`
  - `cut`
  - `df`
  - `du`
  - `find`
  - `getconf`
  - `grep`
  - `lsb_release`
  - `lscpu`
  - `lshw`
  - `mkdir`
  - `nvidia-smi`
  - `scp`
  - `sftp`
  - `sed`
  - `sha1sum`
  - `sort`
  - `ssh`
  - `tail`
  - `tar`
  - `tee`
  - `uname`
  - `uniq`
  - `wc`
  - `wget`
  - `xargs`
- You cannot log in to `markov` and `shannon`; you must execute the code using slurm
- Read the documentation on the DSAI compute infrastructure canvas carefully

Returning your assignment

Return your assignment on Canvas. Your submission should consist of a report that answers all questions as PDF file (preferably typeset in LaTeX) called `assignment1.pdf`. In addition, you should provide the code you used in Problem 2 as either `assignment1_problem2.sh` or `assignment1_problem2.py`. Do *not* deviate from the requested filenames.