Copyright ©2019-2020 Authors

pyCOVID: Worldwide COVID-19 Pandemic Data Presentation with Updates

9022

Abstract ***With ongoing COVID-19 pandemic, a data presentation software is needed for people to get updates about daily infections or deaths in their country daily, which is made possible with modern programming technique. pyCOVID, a Python program, is developed to solve this problem. It can present data with user-defined limitations with updates, displaying data in a user-friendly Graphical User Interface and can be used off-line.***

CONTENTS

INTRODUCTION

A disastrous pandemic of Coronavirus Disease 2019 (COVID-19/2019-nCOV) had struct the world. With more than 79,673,754 infection and 1,761,381 deaths before 4:56pm CET, 28 December 2020, it had become one of the most severe epidemic disease in the world nowadays [5]. So, to get information about current COVID-19 status in your country, an application which can present daily and cumulative deaths and infections is needed. With the development of Python and pyQt, one can create a fancy Graphical User Interface in simple steps. So, with the help of modern programming technique, the program pyCOVID is produced. It has a friendly Graphical User Interface with ultrafast data presentation speed, which is not affected by Internet speed. It can also work off-line, making it portable to places with even no Internet.

1. DESIGN AND IMPLEMENTATION

1.1 File Organization

All source code, except global `configure` and major `Makefile`, is organized in two directories: `src` and `doc`. `src` mainly contains codes that are used to retrieve data, convert data into Structure Query Language (SQL), load the data into database with the user interface while `doc` includes documentation in LaTeXformat. There are also two wrappers named `PyCOVID` and `PyCOVID.py`, which is used as a wrapper under GNU/Linux and Microsoft Windows.

1.2 The Installer

The installer is designed to model the installation systems consisting of GNU BinUtils, GNU Compiler Collection and GNU Make, or, `configure - make` pathways. Firstly, a `configure` script, which is a Bash Script and will check for all executables, Python modules an LaTeXmacro packages, will be executed. Then, `make` should be executed to build this project. Some codes are borrowed from LinuxMiniPrograms Project [6].

The `Makefile` located under the root directory will either build (by target `all`) or clean up (by target `clean` and `distclean`) the project by initializing `Makefiles` in `src` and `doc` folder. Two targets, `update` and `upgrade`, is used to update the database systems. There is also an `OFFLINE` target, which can build an offline-distributable packages named `OFFLINE.txz`. It contains the main executables, wrappers, modules, databases and this documentation in PDF but without building and update systems.

Except `clean` and `distclean`, the `Makefile` located under `src` consists of two targets: `data.db` and pseudo target `$(OBJECT)`. The former is a SQLite Database which acts as the main datasource of this program while the latter converts pyQt user interfaces (Files with `.ui`

suffix) into Python modules. The SQLite database is generated by following steps: Firstly, data file is retrieved from upstream sources from `https://datahub.io/core/covid-19/r/countries-aggregated.csv` in target `countries-aggregated.csv` by `curl`, which is later converted to SQL `INSERT`s by a `gawk` script `ParceCountry.gawk` in target `data.db.sql`. A Python script `sqlite_insert.py` is then used to pump all these SQL data into a single database named `data.db` in the target with the same name. SQL import is optimized by using `PRAGMA synchronous = OFF` to reduce synchronise frequency. The last image `COVID-19.jpg` which is not mentioned in `Makefile` is the icon of this project.

The `Makefile` under `doc` directory will generate a PDF documentation (this one) based on its LaTeXinput. The documentation class `ylarticle.cls` are borrowed from `YLBook.cls` Project [7], which is a further extension of `StylishArticle.cls` [2] and `IEEEtran.cls` [3]. Some LaTeXdistribution like TeXLive may not support `-verbose` parameter, but as no error will occur, this parameter is kept. The `Reference.bib` records all references used in this article.

1.3 Dependencies

The `configure` script needs Bash with its version higher than 4.3. It will not be installed by default in BSD and that installed macOS by default will not work. It also requires a `readline` which supports `-f` option, which can be found at GNU CoreUtils. That installed in BSD or macOS by default will not work.

As is specified in `configure`, GNU Make is used as the skeleton of the installer, cURL is used to download CSV files while GNU AWK is used to transform CSV into SQL `INSERT`s. For the main program which use Python as its interpreter, pyQt tools like `pyuic5` is used to convert pyQt user interfaces into Python modules. Documentation is built by `pdflatex`.

To build the documentation, we also need a functional pdfLaTeXdistribution like MikTeXor TeXLive with `amsfonts`, `amsmath`, `amssymb`, `booktabs`, `calc`, `caption`, `enumitem`, `float`, `graphicx`, `inputenc`, `fancyhdr`, `geometry`, `hyperref`, `hyphenat`, `ifpdf`, `ifthen`, `lastpage`, `mathptmx`, `microtype`, `times`, `titletoc`, `titletoc`, `xcolor` and `url`.

1.4 Compatibility

This software is coded under Windows with MSYS2. It should also run smoothly in CygWin, MSYS2, GNU/Linux, BSD and other UNIX-like operating systems. It is not tested under macOS.

For users under MSYS2 or CygWin, please be aware that firstly, the Python interpreter should be correctly configured. You may only use those provided by CygWin Installer or `pacman` under MSYS2 because Python distributions like official Python for Windows, WinPython or Anaconda may failed to understand the file path inside GNU/Linux. Secondly, if you use Git as your version control system, remember to change the default line endings by `git config --global core.autocrlf input`. Lastly, if you met permission problems, please firstly grant executing permission by `chmod +x configure`.

1.5 The Main Program

The `pyCOVID` you can see at the root directory of this program is a wrapper written in Bash Script. Some of its codes are borrowed from LinuxMiniPrograms Project [6]. It will trigger the real `pyCOVID.py` inside `src` directory, which have a Graphical User Interface by pyQt version 5 and SQLite driver. There is also a `pyCOVID.cmd`, which is a wrapper for Microsoft Windows users.

Files with `ui` suffix is generated by Qt Designer. It is further converted by `pyuic5` and is imported as Python module. The user interfaces mainly consists of two forms: `MainWindow`, `UpdateDialogue` and `AboutDialogue`. From `MainWindow`, you may connect to databases and perform queries while from the latter you may visit sites related to COVID-19 and version information.

The full procedures are consisted with four steps: After connecting to the database, one may start a query by either setting some limitations, or to input the SQL themselves. The software will the perform the query and then report the result to the user in a table view listed beneath the input box.

1.6 The Database

The upstream source of this database is provided by DataHub.io [1], licensed under *Public Domain and Dedication License* which is freely available for non-commercial use.

The database is implemented in a local database file driven by SQLite version 3. To ensure fast retrieval of data, the ERD of this database is very simple. There is only one large table, consists of fields that might get used.

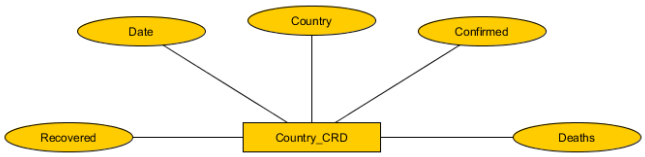

Figure 1. ERD of the database

2. USAGE

2.1 Installation

When installing this software, you may firstly run `./configure` to make sure all dependencies are satisfied with `make` afterwards to build the software and database. To build a clean binary distributable package, please run `make OFFLINE`

2.2 Execution

To execute pyCOVID, whet you only need to do is to execute `./pyCOVID.py`. If there's no error in the installation process, a window will be seen. You may firstly click menu item "Connect" in "Files", which establishes the connection between this software and the database. Then, you may set limitations in "Search by LIMITATIONS" tab, or by typing SQL on your own into "SQL Interpreter" tab. There is also a "History" tab which shows all previous queries.

Ordering of the result is supported by the table widget. You may click on the headline of the view to order a Specific column. You may also export the query by using "Export Result as…" menu inside "File" menu bar. You may export the current view in Comma Separated Value, Tab Separated Value, Microsoft Excel 97-2003 or JavaScript Object Notation format.

In "About" window, you may get version information and license of this program. It also provides a handful of websites, making the user easy to find professional information.

2.3 Update

To update data, please execute `make upgrade`. It will automatically delete previous data and get one new from the upstream datasource.

3. DISCUSSION

With the aim of building the most user-friendly system with available technique, this application have several highlights. The first highlight is AWK is used to parse the CSV file. Using high-level interpreted languages like Python, Perl or R will be too slow while using compiled languages like C or C++ will be too time-consuming to maintain. GNU CoreUtils like `cut` or `paste` cannot parse CSVs with quoted values correctly, due to the reason that they read and cut the files based on commas without noticing whether they are wrapped in pairs of quotes. Written in C, AWK is the fastest text parser and is available in all UNIX-like systems. That makes the software extremely fast and portable.

Secondly, the database management system used is SQLite3, a local single-user single-file database. Comparing to those databases with a Client-Server model, SQLite3 is lightweight and powerful. There is no need to worry about Internet connection, firewall status or database migration, as SQLite database can be read everywhere locally. Client-Server databases have to be operated with slow and comprehensive database management software like pgAdmin, DBeaver or JetBrains DataGrip, while SQLite databases can be quick and easily manipulated by tools like DB Browser for SQLite or SQLite Studio with a friendly Graphical User Interface. The database driver is also supported internally in Python, meaning that external modules or drivers like `psycopg2` is not needed.

Thirdly, although pyQt is much slower and memory-consuming comparing to those implemented by pygame, pyQt is much more user-friendly. Its layout is very close to those everyday applications used by the majority, making it easier to understand and operate.

Comparing to other data providers like Bing Pandemic map, the greatest advantage of this software is that all data is kept off-line, making it possible to query without an internet connection, which is of critical importance in Third-World countries where access to Internet is costly and slow. The performance of this software my not be affected with Internet connections except when performing an update, and once the database generated, it can be transported by USB stick or DVDs offline, which is 100% portable.

However, like all the systems in the word, there are also some limitations. Firstly, to ensure coding conveniences, the skeleton of the installer is built by GNU Make and Bash, making it not that portable in Microsoft Windows without the help of tools like Windows Subsystem of Linux (WSL), CygWin or MSYS2. A graphical installer with complete environment for Microsoft Windows is still needed. It will be implemented by embedding MSY2, just like how GNU Octave is distributed.

Secondly, for those who work remotely without the access of an X Windows System, the UI cannot be initiated without the help of remote desktop systems like XRDP or XVNC, making it unusable if a GUI cannot be provided. To solve this problem, a command-line version of this software is needed

to be implemented. It will be implemented by using Terminal User Interface C libraries like `ncurses`. There are also problems existed in the implementation of table view which displays the data. It will lose response if large volume of data is provided. Multi-threading is tried but no improvement can be seen.

Thirdly, there are existing problems with `PRAGMA synchronous = OFF` in SQLite, which is used to accelerate writing speed. Accidents like power loss will stop syncing, which may cause inconsistency thus database corruption. Thee are also known problems in filesystems that do not or have limited implementation of locks like NFS [4]. Another problem is that the SQL user inserted is not carefully checked. Although the software connects to a local database only, it would also cause potential problems.

Lastly, in database design, there are only one large table inside the database, which is not going to work out the advantage of using relational database systems. It is due to the lack of complexity of the upstream data sources and although space can be saved by splitting text fields like `Country` into country ID and make a new table, it will be slower to query and update. So, as the primary goal of this database is to ensure fast speed, this optimization is not performed.

REFERENCES

[1] Datahub.io. Novel coronavirus 2019 - dataset - datahub - frictionless data. `https://datahub.io/core/covid-19`, 2020.

[2] Mathias Legrand. Stylish article version 2.1. `http://www.LaTeXTemplates.com`, 2015.

[3] Michael Shell. Ieeetran – document class for ieee transactions journals and conferences version 1.8b. `https://ctan.org/pkg/ieeetran`, 2015.

[4] UNKNOWN. How to corrupt an sqlite database file. `https://www.sqlite.org/howtocorrupt.html`, UNKNOWN.

[5] WHO. Who coronavirus disease (covid-19) dashboard. `https://covid19.who.int/`, 2020/12/28, 4:56pm CET.

[6] YU Zhejian. Linuxminiprograms. `http://github.com/YuZJLab/LinuxMiniPrograms`, 2019.

[7] YU Zhejian. Ylbook.cls. `https://github.com/YuZJLab/YLBook.cls`, 2019.