**Exercise 2 – Flight service**

**Objective: p**ractice working with process creation, signals and inter process communication using pipes.

**Bonus:** using open-source libraries.

**Overview: modify your first exercise to be an interactive multi process application, divided into two processes, one shall interact with the user and the second shall perform operations on the DB.**

**Motivation**

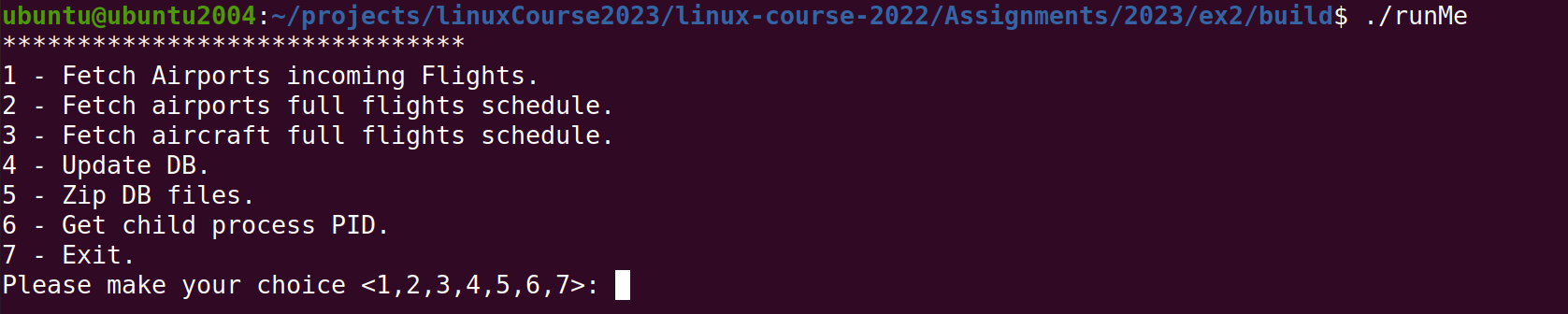
1. Learning how to spawn a new process and write multi process code which is essential for Linux programing as it’s the preferred way to work on Linux. In Linux, processes are relatively light weight, on windows it’s preferred to use threads).
2. Please note that you are using code from your first assignment. It is important to learn to write code in a way that is reusable both for your own productivity and for the ease of collaborating on a shared project.

**Detailed description**

The program shall receive the following options:

Combine the four programs you wrote in your first assignment (reuse your existing code) and implement the following six options (last two are new):

1. Receive airports ICOA code name**s** and output all arrival flights and the flight details.
2. Receive airports name and output the full airport schedule (departures and arrivals) **ordered by time**.
3. Receive list of aircraft names(icao24) and output all flights (departures and arrivals) that it has made.
4. Update the existing airports in the DB with recent data (rerun BASH script).
5. Zip the DB files.
6. Return child process PID and print it.
7. Graceful exit – child shall zip the DB files and terminate (think about collecting the exit status of the child process).



The program shall be divided into 2 processes:

1. Main Process: this process shall create a child-process on initialization, and handle user input and output. When the user chooses options 1-5, the request should be passed from the main process to the child-process and result data shall be received from the child-process**.**
2. Child Process: this process shall receive op-codes(operations) from main-process, perform the required operation and return the data.

**Bonus:** Link your project to libzip assume that libzip will be present on the testing machine.

**Notes:**

* Handle Errors.
* Option 5 – main shall send op code to the child to zip the DB.
* Graceful exit should be implemented by sending SIGUSR1 to child.
* Upon receiving CTRL+C(SIGINT) zip files shall be created like in graceful exit.
* On initialization, existing zipped data should be unzipped and loaded into the child-process’s memory.
* Data will be managed in memory (think of an efficient and easy to manage data structure).

**Submission**

* Add README file inside the zip with the names, IDs, and notes (if needed).
* Use Cmake to build the project.
* Zip all source files together (only .c, .h, .cpp, .hpp, CMakeLists.txt files). Do not include binaries/shared libraries/output files (.o, .out, .so etc.).