Project 1 CSC 552

Points: 25
Due: TBA. Late submissions are at your own risk.
Purpose: Practice system calls open, exec family, fork; binary encoding of files

Assignment: Write programs to handle separate tasks and another program that simulates the shell by running them sequentially. The programs you will call:

- Convert a file of comma-separated data into a binary encoded file
  - Arguments: Input file name and output file name (via command line)
- Prints the first *n records* in a binary encoded file
  - Arguments: File name, # records (via command line)

Your primary program, named **p1** will perform a simplified simulation of the shell that will itself run two other programs, in order, to accomplish a task. From the command line, start **p1**. It takes as its command-line arguments names of programs, data files, and an integer.

The first program (call this executable **createBin**) takes two file names from its command line. The first is a comma separated file of data that you will prepare from a database noted below and use as the source. The second is a destination file whose data will be stored in binary form. This file should have type *.bin*.

The second program (call this executable **printRecs**) will receive the name of a binary encoded file (written specifically for your chosen data set) from its command line, and an integer *numRecords*. It will print the first *numRecords* records in that file, well-labeled. Then, it will loop, allowing the user to select a record to print by entering the record number. The prompt should provide the allowable range; the program will ignore out-of-range entries. The record to be printed is to be accessed in the file via a seek command. The program may not hold more than one record at any time in RAM. The loop will terminate on entry of -1.

p1 forks a child that **exec**s the program named as its 2nd command line argument (**createBin**) to count the words in the named data file. The 3rd and 4th command-line arguments are the *.csv* input file and the *.bin* output file. **createBin** creates the binary file, and its main() returns the number of records that were stored. The parent that was **wait**ing for its child now obtains this value via the exported argument of the **wait()** and prints it, well-labeled. The parent checks the returned value against the 6th command-line argument (# records to print). It forks another child, and this child **exec**s the program named as the parent's 5th command line argument (**printRecs**), passing it the name of the *.bin* file. It also passes the lesser value between the 6th command-line argument and the value returned by **createBin.** The parent again **wait**s, terminating once its child terminates.

Project 1 CSC 552

Notes:

- One suggested but not required source for the data is in our library. It is a knowledge base named **Statista**. Choose a data set. It must have at least four fields and at least 10 records (but not more than 25; you can select from a larger set). Claim it in the **Claim Data Set** forum on D2L. You may not use one that anyone else has claimed. First come, first served.
    - The data can (and will) be saved in cvs form. Then, clean out the non-data stuff (headers, etc).
    - Link: http://library.kutztown.edu/statista You'll likely need to login.
- The 1st command line argument of any program in C or C++ is the name of the program itself.
- If any file can't be opened, that program should return a value that the parent **p1** process recognizes, causing a contextual error message to be printed, after which **p1** exits.
    - This will be tested.
- You must use all three file handle types, file descriptor, file pointer and stream, between the two programs **p1** will call.
    - Use of *fdopen*() is permitted
- If *numRecords* exceeds the number of records in the *.bin* file, all records are printed.
- All programs are to be executable on their own. For example, the first 5 records in *test.bin* should be output if you issue `printRecs test.bin 5` on the command line.
- Reading and writing of a binary record must occur en masse, i.e. the entire record must be read or written with a single command. This is the only permitted access in a binary encoded file.
- A child process is provided a **copy** of its parent's data space.
- An **exec**'d process replaces its caller's text, data, heap, and stack segment. It retains its caller's pid (how does that affect a wait()?).
- **p1** is to use **execvp** to execute **createBin**, simply passing its argv vector translated one element forward.
- **p1** is to use **execlp** to execute **printRecs**. Don't forget to be sure the last command line argument in the **execlp()** command is NULL.
- Your programs must all be properly modular and documented appropriately.
    - Create a robust Doxygen site. Doxygen is a prominent documenting tool that has several links to documentation and tutorials on the instructor's links page. In particular, basic Doxygen use is described here: http://faculty.kutztown.edu/spiegel/Documentation/Doxygen/WorkingWithDoxygen.pdf
      See: https://faculty.kutztown.edu/spiegel/CSc237/Examples/DoxygenDemo/PolygonList/
        - This project is not object-oriented, so \class tags won't be used, but \file and other tags are to be used so that all items in lists have a brief coarse-grained description.
        - A mainpage.dox file should be created in your directory to provide an overall description on the front page of your Doxygen site.
    - Submit a readme with the Doxygen link. You can also provide any other information you care to include that might be useful to the grading process.

Deliverables:

D2L: Your readme. One file in the Phase 1 dropbox.

Turnin: 3 cpp files, named **p1.cpp, createBin.cpp**, and **printRecs.cpp**. You must also submit a properly written **makefile** that builds all three executables with its default target. Each executable MUST also be able to be made on its own on the command line by entering *make <exec name>* (e.g. *make createBin*). Your program will NOT be graded if you do not submit a proper makefile. 2-5 points penalized for using file names other than those specified (case sensitive).