Program #6
Due: Tuesday Apr 24th, 2018 at 11:59 pm

Instructor	Dr. Stephen Perkins
Office Location	ECSS 4.702
Office Phone	(972) 883-3891
Email Address	stephen.perkins@utdallas.edu
Office Hours	Tuesday and Thursday 1:30pm – 4:00pm and by appointment

Graders / TAs

CS/SE 3377.002 - Vatsalkumar Patel - Vatsalkumar.Patel@utdallas.edu

CS/SE 3377.501 - Alexander Lee Hayes - alh170730@utdallas.edu

CS/SE 3377.502 – Amrita Nath - axn163530@utdallas.edu

Purpose

Demonstrate the ability to create a multi-file project that is hosted in a GIT repository. Demonstrate the ability to perform Binary File I/O. Demonstrate the ability to visualize binary data using a CDK matrix.

Assignment

Overview

You will be using binary file I/O techniques to open a supplied binary file. The file consists of a header record that is immediately followed by a set of data records. You are to read and display the header record. Using information in the header record, you will read the data records from the file. For each data record, you are to display the results in a CDK window. After display, you are to wait until the user types a character on the keyboard. When they do, you will exit the program.

Git and Github

You will need to set up a free Github account here:

https://github.com/join

Then, follow the "Hello World" tutorial to learn how to create a project and use GitHub.

https://guides.github.com/activities/hello-world/

Once you have created your account in the web interface, you are to:

1. Create a project that represents Homework 6. You choose the name but it would be good idea to include your name or netid in the project name. The project will start out with a ***master*** branch.
2. Create a new branch called ***working***
3. Once you have performed these steps, you are done with the web interface. You can then move to the Linux command line and utilize the ***`git`*** functions to access and use your github repository.
4. You are to make changes to your your code using the working branch. When you get to "milestones" of working code (possibly the phases recommended below), you can merge the ***working*** branch with the ***master*** branch. You may use the "`git merge`" command or the web-interface's "Pull Request" and "Merge" functions. Both provide the same result.
5. After the first merge, and from any merge going forward, the ***master*** branch must contain a version of your code that will compile and run. New changes should continue to be made to the ***working*** branch.

Supplied Binary File

You can copy the supplied binary file to your project (see *Phase 3* of the ***Recommended Project Phases***). The file is located here:

/scratch/perkins/cs3377.bin

Binary File I/O

You will be using binary file I/O techniques to open a supplied binary file. The file consists of a header record that is immediately followed by a set of data records. You are to read and display the header record. Using information in the header record, you will then read the data records from the file. You will display up to 4 of the remaining records.

The header record in the file is described by this class:

```
class BinaryFileHeader
{
 public:

  uint32_t magicNumber;         /* Should be 0xFEEDFACE */
  uint32_t versionNumber;
  uint64_t numRecords;
};
```

Each data record in the file is described by this class (not this is a fixed length string):

```
/*
 * Records in the file have a fixed length buffer
 * that will hold a C-Style string. This is the
 * size of the fixed length buffer.
 */
const int maxRecordStringLength = 25;

class BinaryFileRecord
{
 public:

  uint8_t strLength;
  char   stringBuffer[maxRecordStringLength];
};
```

The records were written to the file as a byte stream of the entire class instance. You can read them the same way.

Visualization

You will be using old-school UNIX tools to provide visualization of the file contents. Specifically, you will be using the CDK third party terminal windowing library based on ncurses. Your graphical display will show the contents of the binary file.

You will create a 3 wide by 5 high display matrix using CDK. The first row of the table will display the three fields found in the file header (the magic number, the version, and the number of records). The rest of the rows will display the two fields of the first 4 records in the file (or the total number of records if less than 4). These two fields contain the string length and the string.

CDK Library

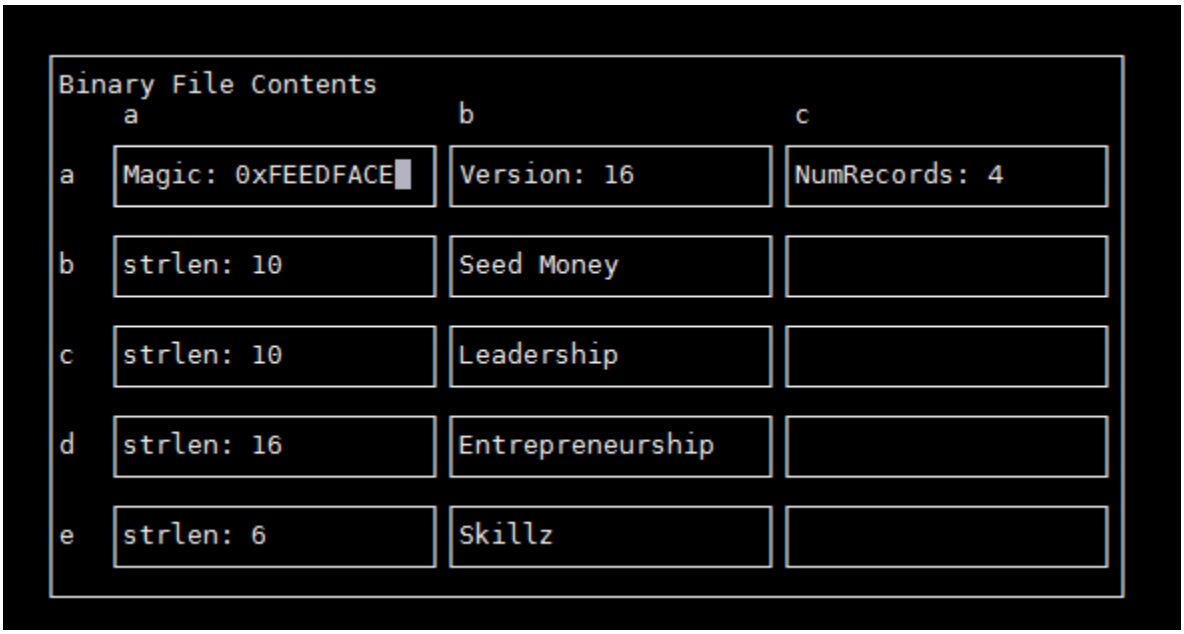
The CDK library is available in source form on the Internet. It is distributed in autoconf format so that you can download, build, and install it.

http://invisible-island.net/datafiles/release/cdk.tar.gz

While you now have the capability to build this from source, DO **NOT BUILD YOUR OWN COPY** for this project. Instead, you will utilize the copy that I have built. See **Hints** below to see how to work with my copy. My copy of the build is installed here:

```
/scratch/perkins/
```

Example Display



Hints

- You can use the Linux utility `hexdump` to view the contents of binary files. Please be aware that they may not look exactly like what you expect.
- The CDK Matrix is based on an offset of 1 (not 0 like arrays). So, 1,1 is the first box.
- Your putty terminal must be large enough to show the matrix of you will get a segfault.
- You may use -Wno-write-strings to suppress const char * warnings.
- Build CDK in a folder such that the full path of the folder has NO spaces in it. Spaces in a folder name will break the build of CDK.
- See the associated file "Fix Putty for curses" if you utilize Putty.
- Your Makefile should pull the project and build
- You should link against MY copy of the CDK library
  `-L /scratch/perkins/lib`
- You will need to link using these libraries:
  `-l cdk -lcurses`
- You should pull CDK headers from my copy of the CDK header files
  `-I /scratch/perkins/include`
- You might consider updating your Shell's MANPATH to include the man pages for CDK
  `MANPATH=$MANPATH:/scratch/perkins/man; export MANPATH`

Deliverables

You must submit your homework through ELearning. Your submission will consist of a single README_<yournetid> file (TXT or PDF only) that includes your name/ netid / and email address. Within this README_<yournetid), you must include the github URL for your project. The TA will clone your project using this URL, cd into your project, and issue make. Your resulting executable will then be tested. All project source files and Makefiles need to have your name, email, and course number commented at the top. As usual, the code must compile and run on cs1.utdallas.edu.

Notes

Your code must be compiled using the **--Wall** compiler flag and must produce no errors/warnings.

No late homework is accepted.

Recommended Project Phases

In order to help with organization, students might consider using these phases as a guideline in how to write the code. Specifically:

Phase 1	• Create a github account • Go through "Hello-World" tutorial • Create new project for Homework 6. • Create new 'working' branch for project • Copy the URL needed to access your github project • Leave the web GUI and move to cs1.utdallas.edu for the next Phase
Phase 2	• On cs1.utdallas.edu, make a new folder for your Homework 6 • Clone your github project using the URL from phase 1 (***git clone <URL>***) • Change directories to the new folder that is created • Use git to move from the master branch to your working branch (***git checkout working***) • Modify the README file • Copy over the CDK example code CC source file. • Create Makefile that compiles the CDK example code. Verify the example runs. o should use -I flags with include paths to MY copy of CDK Header Files o should use -L with library paths to MY copy of CDK Library Files o should use -l flags with libraries cdk and curses • verify backup target working (for backups and program submission) • configure git with your username and email address (***git config ...***) • use git to add the Makefile and the example code CC source file to the project (***git add ...***) • use git to commit the changes to your project with appropriate comments (***git commit -a***) • use git to push your local changes back to github in the cloud (***git push***) • make a backup using your Makefile's backup target
Phase 3	• Modify the CDK example to display a matrix that looks like the one in this file (3x5) • Copy the supplied binary test file into your project and add to your git repository. • Use binary file I/O to read the header record from the binary file. • Display the header record contents into the first row of the matrix • Commit your changes to git • Push your changes to github • Merge your ***working*** branch to your ***master*** branch. • make a backup using your Makefile's backup target
Phase 4	• Use binary file I/O to read the file records • Display up to the first 4 records in the CDK Matrix • Commit your changes to git • Push your changes to github • Merge your ***working*** branch to your ***master*** branch. • make a backup using your Makefile's backup target
Phase 5	• Create a new empty folder in a different area of your file system. • Clone your project from github and make sure your master branch gets all the most recent files • Make sure that you can build and run • If not… go back, fix your repository, and then repeat. • These are the steps the TA will use to grade… make sure the results look like what you want • Upload the README_<yournetid> TXT or PDF file to ELearning with the URL for your Github repository.