CS308: Large Applications Practicum, 2020

Lab 2 (cont'd)

Reference for this task: http://www.vogella.com/tutorials/Git/article.html

Sections for reference from the above article are mentioned in parenthesis in the tasks below.

1 Distributed version control.

1.1 Divide yourselves into teams of five people each. Team members are A (project leader), B, C, D, and E.

1.2 You are going to create a library to perform basic operations in the vector space R^4 (basically a float array having 4 elements.)

1.3 Divide the tasks as follows. **Each function must be a separate .c file.**

1.3.1 Member A **(this must be done in the beginning)**

1.3.1.1 A will create a bare git repository on his/her computer. This has to be named vectorlib.git (11.2)

1.3.1.2 A will push the bare repo into github.

1.3.2 Members A-E create a new local directory (lets call this WRKDIR) where you want to want to do the code development.

1.3.3 Members A-E clone the repository provided by A into their respective WRKDIR.

1.3.4 A should add members B-E as collaborators in Github for this project.

1.3.5 Member A **(A will start this, then other tasks can happen in parallel.)**

1.3.5.1 A will write a main program `(vecmain.c)` which accepts two vectors, and prints their sum, elementwise product, difference of the two vectors, dot product of the two vectors, and the angle between the two vectors. This will be done by calling the respective functions, once they are available. A sample is shown at the end.

1.3.5.2 A will also create `veclib.h` file with the prototype of each function. *(Hope you know the purpose of header files. Header files usually do not contain code, only function prototypes.)*

1.3.5.3 Add and commit these files into the repository.

1.3.5.4 Push changes into the repository. (11.12)

1.3.5.5 It may be good that a skeleton of the `vecmain.c` and `veclib.h` be available first so that other team members know the interface for their respective functions. They will do this when they pull the code. A can coordinate with the others. It is important to have good communication between team members! It is also good to have a predefined protocol for making updates, variable names etc. Having a coding standard is very important. It is even more important when team members are spread out geographically.

1.3.5.6 Create a branch br_add.

1.3.5.7 Write code for addition of two vectors (vecadd.c).

1.3.5.8 After the code is tested, commit and tag it.

1.3.5.9 Merge with master branch.

1.3.5.10 Push changes into the repository.
1.3.5.11 A will keep refining this and keep pushing (and pulling) code from and to the repository. (11.12 and 11.13)

1.3.6 Member B:
1.3.6.1 Pull the latest version of the code. You should get the header file, and some initial version of the main program, which has been created by A.
1.3.6.2 Create a new branch br_prod.
1.3.6.3 Create the function for elementwise product of two vectors `(vecprod.c)`
1.3.6.4 After the code is tested, commit and tag it.
1.3.6.5 Merge with master branch.
1.3.6.6 Push changes into the repository.
1.3.6.7 B will keep refining this and keep pushing (and pulling) code from and to the repository. (11.12 and 11.13)

1.3.7 Member C:
1.3.7.1 Pull the latest version of the code. You should get the header file, and some initial version of the main program, which has been created by A.
1.3.7.2 Create a new branch br_norm.
1.3.7.3 Create the function for norm of two vectors `(vecnorm.c)`
1.3.7.4 After the code is tested, commit and tag it.
1.3.7.5 Merge with master branch.
1.3.7.6 Push changes into the repository.
1.3.7.7 C will keep refining this and keep pushing (and pulling) code from and to the repository. (11.12 and 11.13)

1.3.8 Member D:
1.3.8.1 Pull the latest version of the code. You should get the header file, and some initial version of the main program, which has been created by A.
1.3.8.2 Create a new branch br_dot.
1.3.8.3 Create the function for dot product of two vectors `(vecdot.c)`
1.3.8.4 After the code is tested, commit and tag it.
1.3.8.5 Merge with master branch.
1.3.8.6 Push changes into the repository.
1.3.8.7 C will keep refining this and keep pushing (and pulling) code from and to the repository. (11.12 and 11.13)

1.3.9 Member E:
1.3.9.1 Pull the latest version of the code. You should get the header file, and some initial version of the main program, which has been created by A.
1.3.9.2 Create a new branch br_angle.
1.3.9.3 Create the function for angle between two vectors `(vecangle.c)`
1.3.9.4 After the code is tested, commit and tag it.
1.3.9.5 Merge with master branch.
1.3.9.6 Push changes into the repository.
1.3.9.7 E will keep refining this and keep pushing (and pulling) code from and to the repository. (11.12 and 11.13)

1.3.10 Keep pushing changes into the repository in A's computer. (11.12)
1.3.11 Keep pulling changes to make your local repository up-to-date (11.13) You may need to do this several times during the course of the library development.

1.4 In the end, when development is over, everyone should have the latest version of the code. In case of doubt, just pull the latest.

1.5 Any conflicts that may arise, you will have to resolve them. You can do it by yourself, or by talking to your team to decide how to handle it.

1.6 Play around by modifying code written by other team members. Do not break the master branch. Make branches when you make changes. Merge with master when the code works satisfactorily. Usually it is the team leader who decides what should be merged with the master branch. But you can work out a protocol for deciding what goes into production code.

It must be clear from the commit history how the project evolved, and who has contributed what.