C++ Internship - STL Workshop @ PCA (DCTI, ACE)

Coding Test - May 2018

Adrian Cuţitoiu - Software Developer, Advanced Installer
Victor Ciura - Technical Lead, Advanced Installer

Intro

Git is a source code control system, a crucial tool used in software development to track the changes of a set of files over time.This information is stored in a special data structure called a repository.

The **Git repository** is stored in a subdirectory, called **.git**, of the project directory (also called the root directory). The repository consists mainly of **commits** (a commit is a set of file changes).

Let us consider a ***case insensitive*** file system (meaning that **"**`C:\file.txt`**"** and **"**`C:\FilE.tXt`**"** are the same path). The file changes in a commit we will focus on are:

- *Additions* - files which are absent in the previous commit, but present in the current commit
- *Modifications* - files which are present in both the previous and the current commit, but have different content / modified date (for simplicity a **date** will be an ***unsigned integer*** **/ a timestamp**).
- *Deletions* - files which are present in the previous commit, but absent in the current commit

To compute the differences, Git needs to compare the current project revision with the previous project revision. However, we need to also test our code before making a git commit. This means that we need to compile the code. As a result, many object files are created. By default, when scanning the filesystem git adds these files to the changes for the next commit we make.

Do we need to discard the addition of these files? No, we can do something better. We can specify a **`.gitignore`** configuration file. When Git detects a changed file, it tries to match the path of the changed file with one of the rules specified in the **`.gitignore`** file. If one rule is applied successfully, the changed file is ignored.

For simplicity, our only rule for excluding a file is **case insensitive substring matching** between the path of the file and any of the strings in the **`.gitignore`** file.

Example:

Previous revision	
File path	Date modified
C:\repo\ConTaiNer\deque.h	4888
C:\repo\ConTaiNer\map.h	4863
C:\repo\ConTaiNer\queue.h	4863
C:\repo\ConTaiNer\set.h	4863
C:\repo\ConTaiNer\sTaCK.h	4863
C:\repo\ConTaiNer\vector.H	4863
C:\repo\ptr\auto_PTR.h	4800
C:\repo\strings\StrinG.cPp	4000
C:\repo\strings\StrinG.h	4000

Current revision	
File path	Date modified
C:\repo\ConTaiNer\deque.h	4888
C:\repo\ConTaiNer\map.h	5000
C:\repo\ConTaiNer\queue.h	5113
C:\repo\ConTaiNer\set.h	5200
C:\repo\ConTaiNer\sTaCK.h	4863
C:\repo\ConTaiNer\vector.H	5211
C:\repo\ptr\shared_ptr.h	5233
C:\repo\ptr\unique_ptr.h	5189
C:\repo\strings\StrinG.cPp	5000
C:\repo\strings\StrinG.h	5000

Commit changes	
File path	Status
C:\repo\ConTaiNer\map.h	Modified
C:\repo\ConTaiNer\queue.h	Modified
C:\repo\ConTaiNer\set.h	Modified
C:\repo\ConTaiNer\vector.H	Modified
C:\repo\ptr\auto_PTR.h	Deleted
C:\repo\ptr\shared_ptr.h	Added
C:\repo\ptr\unique_ptr.h	Added
C:\repo\strings\StrinG.cPp	Modified
C:\repo\strings\StrinG.h	Modified

By applying the rules in the following **`.gitignore` file** the changes are filtered:

.gitignore
STRINGS
vector.h
.CpP
\uN
\q

Filtered changes	
File path	Status
C:\repo\ConTaiNer\map.h	Modified
C:\repo\ConTaiNer\set.h	Modified
C:\repo\ptr\auto_PTR.h	Deleted
C:\repo\ptr\shared_ptr.h	Added

Task 1:

You are required to compute the changes (which files are added, modified and deleted) between two consecutive revisions of the project (commits).

Input:

You have two input files which represent the previous project state and the current project state: **`previous.in`** and **`current.in`**.

Each line of these files will have a pair consisting of the ***path of the file*** and the ***date modified*** (timestamp) of the file separated by a *space*.

For simplicity, it is *guaranteed* that the file paths:

- **are sorted in case insensitive lexicographical ascending order**
- **do not contain any spaces**
- **contain only characters from the latin alphabet, backward slashes, underscores and dots**

Output:

You will print 3 files:

1. **`added.out`** - containing the file paths of the added files *in case insensitive lexicographical ascending order*
2. **`modified.out`** - containing the file paths of the modified files *in case insensitive lexicographical ascending order*
3. **`deleted.out`** - containing the file paths of the removed files *in case insensitive lexicographical ascending order*

Each file path will be printed on a new line.

Input/Output example:

previous.in	current.in
C:\repo\ConTaiNer\deque.h 4888 C:\repo\ConTaiNer\map.h 4863 C:\repo\ConTaiNer\queue.h 4863 C:\repo\ConTaiNer\set.h 4863 C:\repo\ConTaiNer\sTaCK.h 4863 C:\repo\ConTaiNer\vector.H 4863 C:\repo\ptr\auto_PTR.h 4800 C:\repo\strings\StrinG.cPp 4000 C:\repo\strings\StrinG.h 4000	C:\repo\ConTaiNer\deque.h 4888 C:\repo\ConTaiNer\map.h 5000 C:\repo\ConTaiNer\queue.h 5113 C:\repo\ConTaiNer\set.h 5200 C:\repo\ConTaiNer\sTaCK.h 4863 C:\repo\ConTaiNer\vector.H 5211 C:\repo\ptr\shared_ptr.h 5233 C:\repo\ptr\unique_ptr.h 5189 C:\repo\strings\StrinG.cPp 5000 C:\repo\strings\StrinG.h 5000

added.out	modified.out	deleted.out
C:\repo\ptr\shared_ptr.h C:\repo\ptr\unique_ptr.h	C:\repo\ConTaiNer\map.h C:\repo\ConTaiNer\queue.h C:\repo\ConTaiNer\set.h C:\repo\ConTaiNer\vector.H C:\repo\strings\StrinG.cPp C:\repo\strings\StrinG.h	C:\repo\ptr\auto_PTR.h

Task 2:

You are required to update the algorithm to filter the results you output in the 3 files above (added.out, modified.out, deleted.out) using information from a **`.gitignore`** file.
Our rule for ignoring a changed file is **case insensitive substring matching** between the **path of the changed file** and **any** of the strings defined in the **`.gitignore`** file.
In other words, if a string in the **`.gitignore`** file is a substring of the changed file's path, that file is ignored.

Input:
The file **`gitignore.in`** contains the subpaths. Each subpath is on one line.
It is **guaranteed** that:

- **do not contain any spaces**
- **contain only characters from the latin alphabet, backward slashes, underscores and dots**

Output:
The output structure is essentially the same. You have to print the 3 files(added.out, modified.out, deleted.out), this time filtered using the **`gitignore.in`** file.
The restrictions for these files from the first task are kept.

Input/Output example:

previous.in	current.in	gitignore.in
C:\repo\ConTaiNer\deque.h 4888 C:\repo\ConTaiNer\map.h 4863 C:\repo\ConTaiNer\queue.h 4863 C:\repo\ConTaiNer\set.h 4863 C:\repo\ConTaiNer\sTaCK.h 4863 C:\repo\ConTaiNer\vector.H 4863 C:\repo\ptr\auto_PTR.h 4800 C:\repo\strings\StrinG.cPp 4000 C:\repo\strings\StrinG.h 4000	C:\repo\ConTaiNer\deque.h 4888 C:\repo\ConTaiNer\map.h 5000 C:\repo\ConTaiNer\queue.h 5113 C:\repo\ConTaiNer\set.h 5200 C:\repo\ConTaiNer\sTaCK.h 4863 C:\repo\ConTaiNer\vector.H 5211 C:\repo\ptr\shared_ptr.h 5233 C:\repo\ptr\unique_ptr.h 5189 C:\repo\strings\StrinG.cPp 5000 C:\repo\strings\StrinG.h 5000	sTRings vEcTor.h .cpP \un \q

added.out	modified.out	deleted.out
C:\repo\ptr\shared_ptr.h	C:\repo\ConTaiNer\map.h C:\repo\ConTaiNer\set.h	C:\repo\ptr\auto_PTR.h

Bonus Task:

There are other ways of specifying rules for the file. Consider a **.gitignore** file consisting of **regular expressions.** A changed file is ignored if its path is matched by any of the regular expressions. Your task is to update the algorithm to perform **case insensitive regular expression** matching instead of case insensitive substring matching.
Note that C++ has a regular expressions library implementation:
http://en.cppreference.com/w/cpp/regex

Input:
The file **gitignore_regex.in** contains a list of regular expressions. Each expression is on one line.

Output:
Output the 3 files(added.out, modified.out, deleted.out), filtered using the **gitignore.in** file.
The restrictions for these files from the first task are kept.

Input/Output example:

previous.in	current.in	gitignore_regex.in
C:\repo\ConTaiNer\deque.h 4888 C:\repo\ConTaiNer\map.h 4863 C:\repo\ConTaiNer\queue.h 4863 C:\repo\ConTaiNer\set.h 4863 C:\repo\ConTaiNer\sTaCK.h 4863 C:\repo\ConTaiNer\vector.H 4863 C:\repo\ptr\auto_PTR.h 4800 C:\repo\strings\StrinG.cPp 4000 C:\repo\strings\StrinG.h 4000	C:\repo\ConTaiNer\deque.h 4888 C:\repo\ConTaiNer\map.h 5000 C:\repo\ConTaiNer\queue.h 5113 C:\repo\ConTaiNer\set.h 5200 C:\repo\ConTaiNer\sTaCK.h 4863 C:\repo\ConTaiNer\vector.H 5211 C:\repo\ptr\shared_ptr.h 5233 C:\repo\ptr\unique_ptr.h 5189 C:\repo\strings\StrinG.cPp 5000 C:\repo\strings\StrinG.h 5000	.*\.cPp$ ^C:\\repo.*e_pTr .*sTRiNg.* .*que.*

added.out	modified.out	deleted.out
C:\repo\ptr\shared_ptr.h	C:\repo\ConTaiNer\map.h C:\repo\ConTaiNer\set.h C:\repo\ConTaiNer\vector.H	C:\repo\ptr\auto_PTR.h

Send your solutions at:

- ➢ **adrian.cutitoiu@caphyon.com**
  OR
- ➢ **nicolae.telechi@caphyon.com**