CECS-343 Adv SWEngr — VCS Project 1

VCS Project 1 — Create Repository
Introduction

This project is 1) to form a development team and 2) to build the first part of our **VCS** (Version Control System) project. This first part only implements an initial use-case: Create Repo (Repository). It also makes a number of simplifying assumptions in order to get to working S/W quickly.

This project will be build in **HTML+Javascript+Node+Express**.

For background material on actual modern VCSs, review on-line user documentation for Fossil, Git, and/or Subversion, etc. Note, in the terminology of a VCS, an "**artifact**" is a particular version of a file; multiple different versions of the file are called artifacts.

The VCS repository holds copies of all artifacts (i.e., all versions) of each file of a project "under configuration control". A file name alone is not sufficient to distinguish between several of its artifacts/versions; hence, within the VCS repository we will use a "code name" for each artifact.

Note, this project will form the basis for the next project, so apply Rule #5 (Clean) as appropriate.

Team

The team is from two to four members. Pick a team name of 3 to 8 letters (e.g., "HelpMe"), with a leading letter and after than you can also include digits, underscores and hyphens (e.g., "H37P-M3"). For the next project, you can change teams and team names.

Use Case #1
Title: Create Repository
Tag-line: Create a repository for the given project source tree (including a "snapshot" of "all" its files) within the project.
Summary: The user needs to keep track of various snapshots of their project, in case they have to "rollback" to a previous good working copy, or in case they want to preserve several experimental feature prototypes of the same project, or in case they want to work on a bug fix without touching the mainline project code until the fix is well-tested.

Each project source tree snapshot includes the current contents of each file in their project tree at that specific snapshot moment during project development. In order to keep track of each snapshot, we want a repository (repo) in the given target folder (where the repo will reside) and we want a copy (the snapshop) of the source folder (where this first project tree currently resides).

The "all files" should exclude "dot-files", their name starts with a period (e.g., ".xtool.rc").

Additionally, on creation of the repository, a snapshot **manifest** (i.e., a summary of files in the snapshot) is created in the repo (as a "dot-file") listing the command particulars (i.e., the "command line" used), the date and time of the command, and **for each project source file** a data item describing that source file's artifact ID and its relative pathname in the source project tree. The artifact ID format is described below. The manifest filename should be ".man-<int>.rc" and a copy should be placed both in the repo and in the source project tree root folder of the create-repo command.

OTOH, each completely different project (e.g., VCS project versus AI-Robot project) should be kept in its own repository.

Artifact ID (ArtID) code names

Weighted checksum: The ArtID (code name) will have 3 parts: "Pa-Lb-Cc" plus the original extension, where "a" is a checksum of the file's relative Path in the source project tree, "b" and the file Length in bytes, and "c" is a checksum of all the Characters (bytes) in the file.

CECS-343 Adv SWEngr — VCS Project 1

The weights by which each character in a string are multiplied are **1, 7, 3, and, 11** in a "loop". Thus, if the file "bot/a/b/fred.txt" (the project root folder is "bot") contains only the string "HELLO WORLD", then the contents checksum C is:

C = **3870** = 1*H + 7*E +3*L +11*L + 1*O + 7*' ' + 3*W + 11*O + 1*R + 7*L + 3*D

(Note, the ASCII numeric value character code of each character is used and we indicated the space character by ' '.) And file size L is:

L = **11**

And the Path "bot/a/b/" checksum P is:

P = **2977** = 1*b + 7*o +3*t +11*'/' + 1*a + 7*'/' + 3*b + 11*'/'

For this version of the source file bot/a/b/fred.txt, the ArtID code name in "PLC" format would be

"`P2977-L11-C3870.txt`"

Modulus: Because the sum can get rather large for a big file, make sure the checksums never get too large by topping each of the three numbers to give the low-order 4 digits.

Project Reports

Standup Status Report, twice weekly: The Standup Status Report is due on or before Friday noon and Sunday midnight, to split up the week's reporting into 2 segments. One report per team – but everybody in your team should be CC'd. It should contain,

o- The **3 Standup Q&A** for each team member
o-The current **Progress Board**, listing WBS tasks and Task phase columns
o- The current **WBS**, tasks & sub-tasks of work on the project

The **3 Standup Q&A**, standard Standup questions: **Q1**. What task(s) have you completed since last status? **Q2**. What task(s) do you plan to complete by next status? **Q3**. What obstacles are blocking your progress (on which task(s))? Consider sub-dividing big tasks into "**Half-Day Rule**" sub-tasks so as to be able to have a task completion (or two) for each status – tasks for which completion is easily **visible** (AKA demonstrable).

WBS: The Work Breakdown Structure is a task hierarchy for the project. You create sub-tasks as you understand what is needed. You don't need to have the details at the start of the project;. (It is always helpful to try to identify a task before you do it.) Example: ID: T24, Name: "Make Tail Follower" Mom: T5 . Here, the kid task T24 is a sub-task of the Mom task T5. For top-level tasks, explicitly indicate T0, the whole project. Build numbers as you go, skipping some if you like. You can change your task hierarchy at any time.

The **Progress Board**, lists each WBS **Task-card** in the **WIP** columns.

WIP: The Work-In-Progress columns are Ready, Working, QA, Done. Ready means the task is clear enough so it can be worked on. Working means a team member has taken ownership and begun work on it. QA means the work has been completed, but now needs a visual check by another team member. Done means the task is finished.

Task-card: A list of the task ID (eg, T24), the team member working it if any (eg, initials "CS" for us), a start date-time if any (eg, "2/7.10am", "2/26.3pm), a QA team member if any (again initials), and an end-date-time if any. A Task-card in the Done WIP column would have all these things. Note that keeping track of tasks, new/old sub-tasks, and completions takes effort, but can be streamlined. Don't forget that merging your completed task code into the team's mainline code requires a bit of integration testing. Tasks that have gone through QA should be demonstrable in class.

CECS-343 Adv SWEngr — VCS Project 1

This Report should be delivered as a **.pdf** file, and each filename should include your course number and section, your team name, the word "Standup", and the date in **YYMMDD format**. E.g., "**343-03-Bozobus-Standup-200221.pdf**".

Testing

Test that the code to implement the Create Repo use-case works

1. On a minimal ptree containing one file:
```
mypt/
  hx.txt  // Contains the string "Hello"; hence, you know the checksum.
```
2. On a tiny ptree containing an extra folder with three files files:
```
mypt2/
  hx.txt  // As above.
  More/  // A sub-folder
    hello.txt // Contains one line: "Hello world".
    goodbye.txt // Contains two lines: "Good" and then "bye".
```
3. Try it by doing a Create Repo on your main source code project tree of files.

Readme File

You should provide a README.txt text file. Be clear in your instruction on how to build and use the project by providing instructions a novice programmer would understand. If there are any external dependencies for building, the README must also list them and how to find and incorporate them. Usage should include an example invocation. A README would cover the following:

- Class number and section
- Project name
- Team name and members
- Intro (use the tag line, above)
- Contents: Files in the .zip submission
- External Requirements (Node, etc.)
- Setup and Installation (if special)
- Sample invocation & results to see
- Features (both included and missing)
- Bugs (if any)

Academic Rules

Correctly and properly attribute all third party material and references, lest points be taken off.

Submission

All Necessary Files: Your submission must, at a minimum, include a plain ASCII text file called **`README.txt`**, all project documentation files (except those already delivered), all necessary source files to allow the submission to be built and run independently by the instructor. [For this project, no unusual files are expected.] Note, the instructor not use use your IDE or O.S.

Headers: All source code files must include a comment header identifying the author, author's contact info (please, no phone numbers), and a brief description of the file.

No Binaries: Do not include any IDE-specific files, object files, binary **executables**, or other superfluous files. We don't use your IDE, your O.S., or your make/model of CPU.

Project Folder: Place your submission files in a **folder named** `X-pY_teamname`. Where X is the class number-section and Y is the project number. E.g., "**343-03-P1-Bozobus**".

JS files: For each JS file, change its name by adding a ".txt" extension to it, producing "foo.js.txt". This is to workaround a few email systems that refuse to send a zip file containing JS files.

Project Zip File: Then zip up this folder. Name the .zip file the **same as the folder name + ".zip"**. Turn in by 11pm on the due date (as specified in the bulletin-board post) by **sending me email** (see the Syllabus for the correct email address) with the zip file attached.
The email subject title should include **the folder name**.

Email Body: Please include your team members' names and campus IDs at the end of the email.

Project Problems: If there is a problem with your project, don't put it in the email body – put it in the README.txt file.

Grading

- 75% for compiling and executing with no errors or warnings
- 10% for clean and well-documented code (Rule #5(Clean))
- 10% for a clean and reasonable documentation files
- 5% for successfully following Submission rules