# Version control using BASH

## Approach

We use a parent script to call all other scripts, providing a single-entry point, and making it more user friendly, because the parent script automatically prints out a menu and tells the user what they can do. The parent script is also useful because it continually prints out the information which repository is open and which file checked out, making it easier to understand what is happening.

### Command vs. menu-based

We originally planned to go with a command-based system inspired by GIT, however we changed to a menu-based system. A big reason why was the fact that not all commands can be always used, e.g., a file cannot be checked-out if another one is already checked-out. In a command-based approach, the user may get confused about what commands they can use, whereas displaying a menu makes it clear.

### Folder structure

When a repository is created, existing files are moved inside a “.vc” folder. That way, we ensure files are hidden when not checked out. When a file is checked-in, a new subfolder in “.vc” is created and only the edited file is stored, while all other files are soft links, to avoid duplication of files.   
Each repository has a “.vc” folder, so it we also use it for storing logs and keeping track of checked-out files.

### Text files as storage

Since BASH has limited support for global variables, we used various text files to save information. For example, “*currently-open-repo.txt*” is used to track if a user has opened a repository. Use of the UID enables multiple users to open repositories simultaneously.

### Managing permissions

Inside of repository-index.txt we store a list of repositories, and UIDs which have READ/WRITE access. We use this instead of changing permissions on repository files since it allows us to check the permission of an entire repository before running a certain script, whereas using “chmod” would only result in errors after running a script, potentially leaving the script half-done. Furthermore, it makes it easier to change permissions since they are all stored inside of the single repository-index.txt file.

## Problems encountered

### ln --relative

Since our scripts are located outside of the working directory, creating soft links inside a repository resulted in invalid absolute links.  
We first used the “-r” option to create a relative link, but found it is not supported on some devices, mainly Mac computers. We replaced it with the “cd” command to first change the directory and then run the ln command to create soft links.

### Calling scripts from other scripts

The “Edit a file” option needs to check out the file automatically. It asks for the filename and calls the checkout.sh script.  
However, checkout.sh again prompts the user for a filename because checkout can be used independently.

To avoid double prompting, we pass the filename as an argument when calling checkout.sh from editfile.sh.   
If the checkout option is used independently, the argument is **not** passed.  
Inside of checkout.sh, the user is asked for additional input only if the argument is empty.

### Bundled compiler

Creating “makefiles” for compilation was quite complicated due to having to correctly list dependencies of files. We opted for compilation using the gcc or javac command, where the order in which files are listed is not important.

## Impressions

As a group we are satisfied with the project, we have implemented additional features and we worked well as a team. We would comfortably bounce ideas off each other and if we had a problem, we would work together to solve it.

Word count: 589 words