Versionable

Track the version of the software and the software components.

The following functionalities provided:

- Obtain the `current` version of the project (the cloned code repository)
- Obtain the `installed` version of the project (installed on the system)
- Compare the current and installed repository versions
- Provide the mechanism for keeping the version at one central place and share with 3rd parties (CMake).

How to use

- Clone the 'Versionable' in the root of the project as the git submodule under the 'Versionable' directory
- In the root of the project create the directory called 'Version'
- Inside the 'Version' directory define the mandatory shell scripts (see the next section)

Mandatory shell scripts

The following scripts are mandatory:

- `version.sh` which provides the information about your project version:

```
#!/bin/bash
export VERSIONABLE_NAME="PROJECT_NAME"                # Mandatory
export VERSIONABLE_NAME_NO_SPACE="PROJECT_NAME"       # Optional, but recommended
export VERSIONABLE_VERSION_PRIMARY="1"                # Mandatory
export VERSIONABLE_VERSION_SECONDARY="0"              # Mandatory
export VERSIONABLE_VERSION_PATCH="0"                  # Mandatory
```

CMake integration

To integrate the version information for your software follow the steps:

Pass the values of the environment variables from the `version.sh` to the CMake script:

```
source ../../Version/version.sh && \
  cmake -DVERSIONABLE_VERSION_PRIMARY=$VERSIONABLE_VERSION_PRIMARY \
  -DVERSIONABLE_VERSION_SECONDARY=$VERSIONABLE_VERSION_SECONDARY \
   -DVERSIONABLE_NAME=$VERSIONABLE_NAME ..
```

Note: Paths to files and scripts of the example command depend on your project's directories setup.

In the `CMakeList.txt` use the provided values:

```
set(VERSIONABLE_NAME ${VERSIONABLE_NAME})
set(VERSION_MAJOR ${VERSIONABLE_VERSION_PRIMARY})
set(VERSION_MINOR ${VERSIONABLE_VERSION_SECONDARY})

configure_file(
        "${CMAKE_CURRENT_SOURCE_DIR}/YourLibrary.h.in"
        "${CMAKE_CURRENT_SOURCE_DIR}/YourLibrary.h"
)
```

Note: To fully support 'Versionable' you should include the CMake snippet. Here is the example of the code:

```
cmake_minimum_required(VERSION 3.22)

set(VERSIONABLE_VERSION_EXECUTABLE ON) # <-- To support the Version binary
which prints the version of the project
include(${CMAKE_CURRENT_SOURCE_DIR}/../Versionable/CMake/CMakeLists.txt)
```

Use the version information in the header files:

- `YourLibrary.h`:

```
#define VERSION_MAJOR 1
#define VERSION_MINOR 0
#define VERSIONABLE_NAME YourLibrary
```

- `YourLibrary.h.in`:

```
#define VERSION_MAJOR @VERSION_MAJOR@
#define VERSION_MINOR @VERSION_MINOR@
#define VERSIONABLE_NAME @VERSIONABLE_NAME@
```

The version information is available in your code (example):

```
#include "YourLibrary.h"

namespace YourLibrary::Info {

    static const std::string getVersion() {
        std::string majorVersion = std::to_string(VERSION_MAJOR);
        std::string minorVersion = std::to_string(VERSION_MINOR);
        return majorVersion + "." + minorVersion;
    }
}
```

Additional variables

The following environment variables can be defined for the `version.sh` script:

```
export VERSIONABLE_SNAPSHOT=true
```

If set to true, the 'Dependable' `install_dependencies` script will rebuild and reinstall the dependency on each run.

```
export VERSIONABLE_HOMEPAGE="YOUR_PROJECTS_S_WEBSITE"
export VERSIONABLE_DESCRIPTION="The description of the project."
```

Overriding scripts

To override the `installed.sh` script create your version of the script under the `Version` directory. Executing the `installed.sh` script from the `Versionable` directory will redirect to your version of the script and execute it.

The same rule apply to the `current.sh` script.