Hamilton College Submission System
Documentation for Professors

Paul Magnus ’18, Ines Ayara ’20, and Matthew R. Jenkins ’20, advised by Alistair Campbell

1 Scripts

1.1 Student Scripts

All student scripts are in the `submit/bin` folder.

- `submit` - This is the submission script that students will use.
- `timeleft` - This script allows the student to check how much longer they have until the project is due.
- `submit_terminal` - This script is mainly intended for use with editors such as the CSPy editor. This opens up a new xterm window and runs submit in the new terminal.
- `.submit` - This is a hidden script run by submit_terminal. It is not intended to be called directly.
- `.csTemplate` - This is a template for setting the student’s course environment to work with this submit system. This script should not be called directly but instead be edited for each course and renamed as such.

1.2 Professor Scripts

All professor scripts are in the `submit/pbin` folder.

- `makeTemplate` - This script creates a simple submission directory. It can also be used for minor edits to the submission directory. Most edits not supported by this script must be done manually.
- `extension` - This script gives a single student an extension on the given project.
- `verify` - This verifies that the course directory was built correctly. It only checks directories and files that could have been built with the makeTemplate script.
- `submitAs` - This allows a professor to submit a project as any student.

2 submit

This is the main script of the submission system. Students will use this script to submit a project. The usage and optional arguments are explained below as they are shown when `submit --help` is run.

```
Usage: submit [OPTION]... PROJECT
This script submits files for evaluation

Mandatory arguments to long operations are mandatory for short options too
  -c, --course <number>        sets the number of the student's course
  --folder <path>              sets the folder to be submitted
  -f, --force                  do not ask for confirmation
  -h, --help                   print this help documentation
  -i, --interactive            use interactive mode to input unkown data
  -p, --professor <name>       sets the name of the student's professor
  -s, --silent                 hide non-error messages
  -v, --verbose                explain what is being done

If course and professor are not set then they will be determined based on
the current course environment of the user. If folder is not set it will
default to the current working directory.

The -s, --silent and -v, --verbose options cannot be set at the same time
```

Most users will want to use either `submit PROJECT` or `submit -f PROJECT`. If the student is not currently in a course environment they will need to set the course and professor arguments or use interactive mode.

3 timeleft

This script allows students to check on the remaining time for a project. This script will only work correctly for projects in which a `duedate` file exists in the project directory. Otherwise, this script will tell the student that the project does not have a current duedate. The usage and optional arguments are explained below as they are shown when `timeleft --help` is run.

```
Usage: timeleft [OPTION]... PROJECT
Prints the time remaining on the current PROJECT for the user

OPTIONS:
  -c, --course <number>        sets the current course number, if not set
                               the default comes from the course environment
  -h, --help                   print this help documentation
  -p, --professor <name>       sets the name of the student's professor
```

If the student is currently in a course environment, they will only need to call `timeleft PROJECT` as the other arguments allow for control when the course environment is not set.

4 submit_terminal

This script is primarily intended for use by the CSPy editor and any similar editors that are to be built in the future. This script opens up a new xterm window and runs the submit script in that window. The same arguments can be passed to `submit_terminal` as can be sent to `submit`.

5 .submit

This script should not be called directly. It is used by `submit_terminal` to create a better interface when the new terminal is used.

6 .csTemplate

This file is a template of a course environment script for use with this submission system. This script is not intended to be used directly but should be changed for each course. Most sections of code in this template are optional and are marked as such in the script itself. Only 3 pieces of the code are marked as VITAL and must be present for the submission script to work correctly as explained below. This script was built by Paul Magnus '18 based off of the course environment scripts written by Professor Alistair Campbell and Professor Mark Bailey.

COURSE

This is necessary for any course environmnet so that the $COURSE variable is set properly. The code is as follows

```
export COURSE=110
printf "Setting CS$COURSE programming environment...\n"
```

The important part of this script is the first line. The number after `COURSE=` must be the Hamilton Computer Science course number.

PROFESSOR

This is necessary for any course environment so that the $PROFESSOR variable is set properly. The code is as follows

```
export PROFESSOR=acampbel
```

Similarly to the COURSE assignment, this must be changed to reflect the professor of the course. The name after `PROFESSOR=` is the username of the professor collecting the project submissions. This may or may not be the actual professor for the student's section of the course depending on how project submission should work for the given course.

PATH

This section of code gives the user access to the current directory so that the `submit` and `timeleft` scripts can be run at any time by the student. As long as the course environment script is in the `submit/bin` folder with the other student scripts, this code should not need to be changed. The code is as follows

```
# Get directory of script, resolving links
SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do
    DIR="$( cd -P "$( dirname "$SOURCE" )" && pwd )"
    SOURCE="$(readlink "$SOURCE")"
    [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE"
done
DIR="$( cd -P "$( dirname "$SOURCE" )" && pwd )"

printf "    Adding submission programs to environment\n"
export PATH="$DIR:$PATH"
```

The first 8 lines of this code is a very helpful script that gets the exact location of this script even if there are links and references used. This directory is then added to the user's $PATH variable that is used by the system when searching for executable files in the bash terminal.

7 makeTemplate

This script creates an empty submission directory. By default the system will take all files in the current directory when submit is called and no tests will be run. The project can be made into a currently submittable project by using the `-c, --current` or by setting the duedate using `-d, --due`. Students can submit to a project either if it is current or if the duedate is in the future. Please do not use both current and duedate on the same project as it will be submittable as long as either one allows. The `-p, --prof` argument is in case the current user is not the same as the professor creating this submit directory. The help documentation for the script is as follows

```
Usage: makeTemplate [OPTION]... CLASS PROJECT
Build a basic submit directory for the PROFESSOR, CLASS, and PROJECT

OPTIONS:
  -c, --current <bool>  sets whether this project is a current project for
                        the class. <bool> can be 't', 'true', 'T', 'True',
                        'f', 'false', F', or 'False'
  -d, --due <date>      sets the date and time that the project is due
                        if this is not set then there is no automatic
                        handling of the due date by any of the bash scripts
                        <date> must be in a form recognized by the gnu date
                        command
  -h, --help            print this help documentation
  -p --prof <name>      sets the name of the professor, if this is not set
                        then the default is the current user
  -v, --verbose         explain what is being done

makeTemplate can be used to build a new project submit directory or update
a project that already exists through use of the options.
```

8 extension

This script gives one student an extension on the given project. This will allow the student to submit the project until the extension date. The usage is shown in the help documentation below.

```
Usage: extension [OPTION]... STUDENT CLASS PROJECT DATE
Gives the STUDENT an extension on PROJECT for CLASS until DATE, default
professor is the current user

OPTIONS:
  -h, --help            print this help documentation
  -p, --prof <name>     sets the professor to <name>
```

9 verify

This script checks that the submit directory is built correctly. This can only check for the existance of files and their permissions and this only checks for the base required files for submit to work properly. The usage is shown in the help documentation below.

```
This program is intended to be run by the professor to verify that their
directory is set up in a way that the submit program will be able to use
without errors.

Usage: verify [OPTION]... PROFESSOR CLASS PROJECT
Verify all files for PROFESSOR, CLASS, and PROJECT exist

OPTIONS:
  -f, --fix             fix any errors that are detected, if possible
  -h, --help            print this help documentation
  -v, --verbose         explain what is being done
```

10 submitAs

This script allows a professor to submit a project as any user. This script will still submit even if the project is not current and/or the duedate has passed. The usage is shown in the help documentation below.

```
Usage: submitAs [OPTION]... USER PROJECT
This script submits files for evaluation as the given user

Mandatory arguments to long operations are mandatory for short options too
  -c, --course <number>         sets the number of the student's course
  --folder <path>               sets the folder to be submitted
  -f, --force                   do not ask for confirmation
  -h, --help                    print this help documentation
  -i, --interactive             use interactive mode to input unkown data
  -p, --professor <name>        sets the name of the student's professor
  -s, --silent                  hide non-error messages
  -v, --verbose                 explain what is being done

If course and professor are not set then they will be determined based on
the current course environment of the user. If folder is not set it will
default to the current working directory.

The -s, --silent and -v, --verbose options cannot be set at the same time
```

11 Manual Edits

For the following explanation we will use an example project created using `makeTemplate` for CS110 with Professor Campbell for the bots project.

11.1 Required and Optional Files

By default, submit will copy all files from the student's current working directory when submitting. This is controlled by `required\_files` and `optional\_files`. For our example, both of these files are in `submit/acampbel/110/bots`. Submit will require the user to have all files listed in `required\_files` in order to submit. Furthermore, it will copy over all files listed in `optional\_files` if they exist in the user's current working directory. If at any point in `required\_files` or `optional\_files` a line contains only a , then all remaining files from the user's current working directory will be copied and no further checks will be made for required files. An example of this is shown below.

required_files:

```
bots.cspy
bots_supplement.cspy
```

optional_files:

```
bots_optional.cspy
```

When submitting, this will ensure that the user has `bots.cspy` and `bots_supplement.cspy` in order to submit. Also, if the user has `bots_optional.cspy`, that file will be submitted as well.

11.2 Test Script

If the created project should have any tests evaluated on it these should be handled in the `run_all_tests` file. This file is automatically generated by makeTemplate but does not run any tests on the submitted code.

To edit this file for our example project we go to the file `submit/acampbel/110/bots/tests/run_all_tests`. At the end of the submit script, this file recieves one argument: the folder containing the recently submitted project. The code in this file should be changed to a bash script that will run tests on the folder given as an argument.

12 Student Submit Folders

Inside of the project folder there is a directory named `students`. When a student submits, their submission is placed in here. The most recent submission is stored in a folder named `students/username/last-submit`. This is the folder that is sent to `run_all_tests`. Also, a new submit folder is made. If it is the first submission this folder is `students/username/submit` and if it is not then it is named `students/username/submit-1` with the number incrementing with each new submission. In this way, all submissions are retained and the most recent submission is also easily accessable. Furthemore, the file `students/username/submit-time` keeps track of the submission time of each submission folder. This allows for an alternative way to find when a student submitted the project besides checking the time when the folder was modified/created in case that changes for some reason.