CSE 4/589: PA3 Template

CSE 489/589
Programming Assignment 3

Instructions for installing/using the Template

What is this document about?
This page provides instructions for the template of Programming Assignment 3.

Do I need to use this template?
YES. It is mandatory to use this template.

What is the template?
The template is a collection of a basic directory structure along with some source code and scripts/programs that are required for you complete PA3.

Why a template?
In the past offerings of this course, students lost points because they failed to follow certain naming/submission conventions that we specify in the assignment description and expect while grading. The template exists to avoid such cases.

Can I just use parts of the template (and do the rest manually)?
NO. If you use scripts/programs not included in the template (or do things manually), we cannot guarantee that your submission will adhere to the PA3 specification and hence may result in the loss of points.

What Operating Systems (OS) will this work on?
We have tested the scripts on Linux based OSes only. They will NOT run natively on Windows. Although there is no reason for them to not work on environments that try to emulate Linux on Windows (e.g. cygwin), we will not be able to provide you support for them. Same goes for Mac OS X. However, feel free to post questions about this on the course blog. Other students facing the same issue might be able to help you.

What if I have questions about this template?
Piazza is the central place to ask questions for everything related to the course, including this template. Make sure to use the correct category for PA3 when asking questions about this template to ensure a quick response.

Will this work on my personal linux machine?
Most likely, yes. However, given there are so many different flavors of Linux and configurations available, we cannot test on each one.

Installation

➔ Create a new directory/folder to house the assignment. For the purpose of this assignment, we will use the name *cse489589_assignment3*. You should create this folder directly inside the directory mentioned in section 3.4 at https://goo.gl/HYHcyQ on one of the dedicated hosts, and NOT at any other path or sub-directory.

```
$ mkdir cse489589_assignment3
```

➔ Change into the assignment directory.

```
$ cd cse489589_assignment3
```

➔ Download the starter script.

```
$ wget https://ubwins.cse.buffalo.edu/cse-489_589/pa3/assignment3_init_script.sh
```

➔ Give the starter script execute permission.

$ chmod +x assignment3_init_script.sh

➔ Run the starter script

- This needs to be run only once. Running the script again can potentially overwrite your existing work.

$ ./assignment3_init_script.sh

➔ The script will prompt you for some answers.

- Enter your UBIT username (without the @buffalo.edu part).
- Enter your Full name.
- Choose the programming language: C or C++.
- Confirm the details.
- Wait for script to finish; On success, it will display a message.

➔ When the script exits (successfully)

- You should see a directory structure, with some scripts and a folder created in the current directory.
- If the script fails for some reason, it will try to give you a hint about the possible reason. Fix the problem, delete everything inside the current directory (except the startup script itself) and try again.

The instructions above are meant to be executed once, before you start working on the assignment. Once the template is in place, you should not have to execute any of the steps mentioned above, again.

About

➔ The script will create a directory named after the UBIT username supplied in the earlier step. This directory is where all source code and Makefile will reside.

➔ The directory already contains the following folders/files:

- src [folder]
  - <your-UBIT-name>_assignment3.c /<your-UBIT-name>_assignment3.cpp
- include [folder]
- object [folder]
- Makefile [file]

➔ You should add any new source (*.c/*.cpp) files that you need inside the src folder only. If you keep any new *.c/*.cpp files outside this, they will not be picked up by the Makefile.

➔ You should add any new header (*.h) files that you need inside the include folder only. If you keep any new *.h files outside this, they will not be picked up by the Makefile.

➔ You should NOT modify the Makefile, in any way. You will be assigned a zero (0) grade if it is modified.

Usage

➔ For the template to work and to avoid any naming convention mistakes, you should NOT modify the *Name and/or Paths* of any of the directories/folders/files that are included in the template. You are free to add new files, in accordance with the instructions above.

➔ You can of course add your code inside the <your-UBIT-name>_assignment3.c /<your-UBIT-name>_assignment3.cpp file included below the /*Start Here*/ comment line. Do NOT change the code lines above that. Further, the grader expects the main() function to reside in the file named <your-UBIT-name>_assignment3.c /<your-UBIT-name>_assignment3.cpp. You should not modify the the first line of the main() function in any way.

➔ Building your program.
- Change into the directory named after you UBIT name.

  $ cd <your-UBIT-name>

- Build the program

  $ make

- This will create an executable named *assignment3* in the same directory.
- To clean up the executable and object files

  $ make clean

➔ Adding a new source (.c/.cpp) or a header (.h) file
- In case of a source (.c/.cpp) file, add it inside the *src* folder
- In case of a header (.h) file, add it inside the *include* folder.

Controller

➔ The final grading will make use of automated tests and will use a controller very similar to the one included with this template.

➔ We have included two files/scripts:
- controller
- topology_example

which you will need to generate the control messages and test your implementation.

➔ To run the controller
- Change to the cse489589_assignment3/controller directory, then execute

  $ ./controller -h

  to get the list of parameters/options you need to pass to run them. The parameters/options are generally one of control message types and the additional arguments needed to generate the control packet.

➔ **topology_example**
- This is an example topology_file that we provide so that you can test your controller and implementation with it.
- You can modify this file to create your own topologies
- The final grading will use many different topologies to test your implementation

➔ *We will occasionally update the scripts to add new features/fix bugs. You should lookout for announcements on piazza and check the version number of the scripts you have against version number on piazza and update if necessary.*

➔ To update the controller
- Change to cse489589_assignment3 directory, then execute

  $ ./assignment3_update_scripts.sh

Sample Code

https://ubwins.cse.buffalo.edu/cse-489_589/pa3/swetankk_pa3.tar

Grader

➔ The automated grader expects the output to adhere to the format/syntax described in the assignment description document.

➔ We are providing a grader to you so that you can verify the correct execution of your code before submission.

➔ The grader reads configuration from a configuration file located in **grader/grader.cfg**. The configuration details will be provided to you by the course staff.
If you are a UB student taking CSE 4/589, you need to add/edit the following fields under the [SSH] section:

- user: username that has ssh access to each of the five dedicated hosts
- id: absolute path (on the local machine) to the private key of the key-pair that allows you to ssh to each of the five dedicated hosts

Do not modify any other sections of the configuration file.

➔ The grader can run on any machine (e.g., your personal laptop) as long as it can ssh to the five dedicated hosts.

➔ To run the grader

- Change to the cse489589_assignment3/grader directory, then execute

```
$ ./grader_controller -h
```

- The **-c** option requires the path to the grader.cfg file.
- The **-d** option requires the path to an empty directory (on the five dedicated hosts) where assignment files will be stored/built, e.g., /local/Spring_2017/<your-UBIT-name>/grader_staging/
- The **-s** option requires the path to your submission tarball.
- The **-ctrl** option requires the path to the controller binary on the local machine
- The **-t** option supports the following tests:
  - author
  - init
  - rupdates (Distance Vector Updates in the Grading Rubric)
  - rtable (Routing Table in the Grading Rubric)
  - update
  - crash
  - data (Data Plane in the Grading Rubric)
  - bonus
- Requires: Python 2.7.x

➔ To update the grader

- Change to cse489589_assignment1 directory, then execute

```
$ ./assignment3_update_scripts.sh
```

Packaging and Submission

➔ Packaging your code for submission

- Change to cse489589_assignment3 directory, then execute

```
$ ./assignment3_package.sh
```

- This will create a tarball, named as <your-UBIT-name>_pa3.tar, in the same directory.
  Note that this step does NOT submit your assignment, just packages it.
  If you are a UB student taking CSE 4/589, you need to use the submit_cse489/submit_cse589 (available on CSE servers) script to submit this tarball.

➔ Submission (if you are a UB student taking CSE 4/589)

- Make a submission on a CSE server using the submit_cse489/submit_cse589.
- Take a snapshot as a proof, along with the *$ date* command immediately following the submission.