CSE 4/589: PA2 Template

CSE 489/589
Programming Assignment 2

Instructions for installing/using the Template

What is this document about?
This page provides instructions for the template of Programming Assignment 2.

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 to complete PA2.

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 PA2 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 PA2 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_assignment2*. You should create this folder directly inside the directory mentioned in section 3.2 at https://goo.gl/KzTh0J on the machine assigned to you, and NOT at any other path or sub-directory.

```
$ mkdir cse489589_assignment2
```

➔ Change into the assignment directory.

```
$ cd cse489589_assignment2
```

➔ Download the starter script.

```
$ wget https://ubwins.cse.buffalo.edu/cse-489_589/pa2/assignment2_init_script.sh
```

➔ Give the starter script execute permission.

```
$ chmod +x assignment2_init_script.sh
```

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

```
$ ./assignment2_init_script.sh
```

➔ The script will prompt you for some answers.
- ◆ Enter your UBIT username (without the @buffalo.edu part).
- ◆ 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]
  - ● simulator.c / simulator.cpp
  - ● abt.c / abt.cpp
  - ● gbn.c / gbn.cpp
  - ● sr.c / sr.cpp
- ◆ include [folder]
  - ● simulator.h
- ◆ object [folder]
- ◆ Makefile [file]

➔ The *src* folder contains three .c/.cpp files, one for each of the three protocols you need to implement. You should NOT add any new source (*.c/*.cpp) files. For a given protocol, all code will go inside its respective file.

➔ You should NOT modify the simulator.c / simulator.cpp, simulator.h, or the Makefile, in any way. You will be assigned a zero (0) grade if any of these files are 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.

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

```
$ cd <your-UBIT-name>
```

- ◆ Build all three protocols

```
$ make
```

- This will create three executables named *abt, gbn* and *sr* in the same directory.
- To clean up the executable and object files

```
$ make clean
```

➔ Running your program.
- Each of the three protocols' binary can be run independently, by supplying the command-line parameters listed in section 3.6 at https://goo.gl/KzTh0J
- When the simulator terminates, it will print some statistics (apart from the debug messages).
- *Note that running the simulator this way is fine only when you are debugging and/or trying to improve the performance of your implementation. For obtaining results for analysis/graphs, you should use the experiment script described below.*

Scripts

➔ The final grading will be a combination of automated tests on your implementation and manual grading of your analysis report.

➔ We have included the following scripts:
- run_experiments
- sanity_tests
- basic tests
- advanced_tests

which you will need to use to generate your reports and test/grade your implementation

➔ To run the scripts
- Change to the cse489589_assignment2/grader directory, then execute

```
$ ./run_experiments -h
OR
$ ./sanity_tests -h
OR
$ ./basic_tests -h
OR
$ ./advanced_tests -h
```

to get the list of parameters you need to pass to run them. The parameters are very similar to what you would pass if you were running a given protocol's binary directly.

➔ **run_experiments**
- You SHOULD use ONLY this script for running any experiments. If you run experiments (and report the results) without the use of this script, we will not be able to verify your results and will result in loss of points.
- The script runs your binary (abt/gbn/sr) 10 times for a given set of input parameters. The 10 different runs use different seed values and hence will provide you with different output values. You will probably want to take average of these 10 runs for the results you will include in your report.
- The output is a single csv file containing the result of each of the 10 runs.
- The script provides some optional parameters, to change the output filename and format.

➔ **Sanity_tests, basic_tests, advanced_tests**
- These scripts run the test suites described under SANITY Tests, BASIC Tests, and ADVANCED Tests in section 5.1 in the PA2 description document.
- They invoke the run_experiments script, read the output generated in the csv file, and verify certain properties we expect to see in your results.
- The scripts will fail if your implementation violates any of those properties for any of the seed values used by run_experiments.

➔ To update the scripts
- ◆ Change to cse489589_assignment2 directory, then execute

```
$ ./assignment2_update_grader.sh
```

Packaging and Submission

➔ Packaging your code for submission
- ◆ Change to cse489589_assignment2 directory, then execute

```
$ ./assignment2_package.sh
```

- ◆ This will create a tarball, named as <your-UBIT-name>_pa2.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.