# ACIT2420 Midterm – Feb 19, 2019

The ACIT2420 midterm is a hands-on test, having you build a shell script with multiple actions. It will address common tasks similar to those you have encountered in weeks 1-6 in of our course.

**Grading**: Each of the script actions will be graded for correctness, on a scale of 0-3 (like the labs). The overall script will be evaluated too, for completeness, conciseness, and readability. There is a test script provided, that we expect to run without hiccups.

- *completeness* refers to the number of feature points that you address;
- *testability* refers to successfully running the test script and producing an appropriate log file
- *conciseness* refers to avoiding duplicate code where possible; and
- *readability* refers to how clear and easy-to-read your scripts are; comments & indenting help a lot

The sum of the midterm grade components will be scaled appropriately in the gradebook, and will account for 15% of your final grade.

**Permitted**: this is an open-book exam. Feel free to search online, but be cautioned that you will not have time to look up the answer to every question you have.

**Not permitted**: no “ask a friend” is permitted. If we see any evidence of messaging, incoming or outgoing, explicit or accidental, using an app or website, you will be **disqualified** on the spot. Close your messaging, email and social media apps on all your devices before starting the test. Mute your devices or turn them off during the exam, please.

**Submission**: a “midterm” dropbox is provided on D2L. Submit your shell script there, not zipped. The dropbox has a due date/time of 12:50 on midterm day, giving you two hours to complete your script, plus five minutes grace in case the system gets overloaded at the end of the midterm.

**Solution**: a solution will be posted on D2L a few hours after the test period has concluded.

**Requirements** are shown following.

# Teamwork Script Background

Your pointy-haired-boss neither understands nor is willing to pay for using github.com to manage the source code for your company's next killer app project. He has tasked you with building a shell script, `teamwork.sh`, to help your software team collaborate through a development folder on your CentOS 7 system.

You plan to setup a `/home/killerapp` folder on your system to hold the killer app sources, and to create accounts for each member of the software team so that they can remotely login to your system to work on the project.

The intended use of your script is `./teamwork.sh action filename`, with the required actions described in the requirements section. The `setup` and `cleanup` actions are meant to be run as root, while all others are meant to be run by a software team user. Files and folders inside the development folder should grant all permissions to everyone.

There will be three subfolders inside the development folder: `source`, `wip` and `test`. The `source` folder contains the stable version of the app, with copies of all of the source code files. The `wip` folder contains those source code files that are being worked on. The `test` folder contains those source code files with completed changes, ready to test and review by other team members. The owner of any source code file should be the team member currently working on that file, or who last worked on it.

The project folder itself should contain a `worklog` file, recording team activity carried out using your script. Entries in it should show the date & time, the user responsible, and a suitable log message. To facilitate testing, this log file should be world-writable.

Sample worklog resulting from running the test script:

```
Mon Feb 18 09:19:08 PST 2019 root: Killer app project setup!
Mon Feb 18 09:19:08 PST 2019 dilbert: server.js checked out
Mon Feb 18 09:19:08 PST 2019 asok: utils.js checked out
Mon Feb 18 09:19:08 PST 2019 wally: bogus.js cannot be checked out
Mon Feb 18 09:19:08 PST 2019 asok: config.js checked out
Mon Feb 18 09:19:08 PST 2019 wally: config.js has already been checked out
Mon Feb 18 09:19:08 PST 2019 dilbert: server.js has been proposed
Mon Feb 18 09:19:08 PST 2019 wally: server.js has been rejected
Mon Feb 18 09:19:08 PST 2019 asok: config.js has been proposed
Mon Feb 18 09:19:08 PST 2019 dilbert: config.js has been approved
Mon Feb 18 09:19:09 PST 2019 wally: config.js cannot be rejected
Mon Feb 18 09:19:09 PST 2019 asok: utils.js has been discarded
```

At this point:
- "source" should contain config.js (asok), server.js (root) and utils.js (root)
- "test" should contain server.js (dilbert)

# Teamwork Script Requirements

The “teamwork” script can have up to two parameters. The first, if specified, is the action name, and the second, if specified, is the filename to act on.

Each request should be logged, and the log entry should show success or failure.

Note that the “current user” is whichever team member is running the script from their shell session.

Each action is described below:

**help**

- display a list of valid actions
- also show this list if no action is specified
- no log entry is needed for this

**setup**

*Gets everything ready for team collaboration.*

- Create accounts for dilbert, wally and asok. Their passwords can all be “KillerApp27”, as they can change that themselves later.
- Create a “team” group, and add the team members to it
- Create a development folder, “/home/killerapp”, suitable for use with the teamwork
- Create subfolders “source”, “wip” and “test” inside it
- Retrieve the current version of the app source code files, server.js, utils.js and config.js; save these in the “source” folder
- Make sure all file & folder permissions inside your “killerapp” project folder grant all rights to the team
- add a log entry

*At this point, “wip” and “test” are empty, and “source” contains all of the source code files with root as owner.*

**cleanup**

- remove the killerapp folder
- delete the team group
- delete the team member accounts

*No trace of the killer app is left on your system now.*

**checkout filename**

- if the requested file is in “source”, but not in either “wip” or “test”, then copy it to “wip” and change the owner there to the user who ran the script
- add a log entry

**discard filename**

- if the requested file is in “wip”, and it belongs to the current user, then delete it from “wip”
- add a log entry

**propose filename**

- if the requested file is in “wip”, and it belongs to the current user, then move it to “test”
- add a log entry

**reject filename**

- if the requested file is in “test”, and it **doesn’t** belong to the current user, then delete it
- add a log entry

**approve filename**

- if the requested file is in “test”, and it **doesn’t** belong to the current user, then move it to “source”
- add a log entry