COMP30023: Introduction to projects

COMP30023 Head Tutors

March 24, 2023

1 Introduction

In this document, we will introduce how COMP30023 projects are structured, and outline how you can make the most from the infrastructure which we provide.

Firstly, some attributes and expectations:

- As per the handbook, there will be 2 projects weighted 15% each
- Projects are to be completed individually
- Submissions must be in written in C
- Submissions must compile and run on COMP30023-provided VMs, and should produce deterministic output
- Submissions will be checked for plagiarism

2 Submission

The submission process may be slightly different to what you were used to. We expect:

- Code to be submitted to your assigned Gitlab repository named `comp30023-2023-project-x` in the subgroup with your username of the group `comp30023-2023-projects` on gitlab.eng.unimelb.edu.au.
- **AND** The full-40 digit SHA1 hash of your chosen commit to the relevant Project Assignment on the LMS.

Failure to complete both successfully will result in mark deductions.

Upon submitting the hash, you should receive an automated comment indicating whether your submission is valid/reachable on Gitlab. An example success message is: `Taking commit 2e24d35658d1b552952d226f5362fa5863d09a3c from 2021-05-19T10:21:49Z, day 0`.

Note that it acknowledges the hash which you submitted and the time of submission (given to us by Canvas). From this, we calculate the day number: day 0 means that the submission is on-time, day 1 means that it's 1 day after the deadline, and so on.

We do not perform calculations on fractional days, and thus, it is up to you to consider whether it's worthwhile to make further submissions (if submitting late).

If you have an extension, please note that we will subtract your extension from this day number **when calculating your project mark**.

3 Compilation

Unlike COMP20003 and COMP20007, we do not impose the usage of skeleton files or a particular project structure (in terms of files/directories). You're welcome to use either `gcc` or `clang`, and anything in the C standard library (libc), and POSIX, provided it works on the VM, but excluding calls to other libraries/services.

To ensure that your projects can be successfully tested, please make sure that there is a `Makefile` at the root of your repository, which compiles the executable(s) defined in the specification to the root of the repository also.

You're welcome to write test cases and scripts (possibly in other languages) and use other build systems during implementation. Feel free to keep these files for the final submission.

You can, but do not have to include copies of input files in your repository, as they will be provided by the CI/marking environment.

Do not hard code file paths for test cases or files, or make assumptions about where your executable will be launched from.

4 Testing

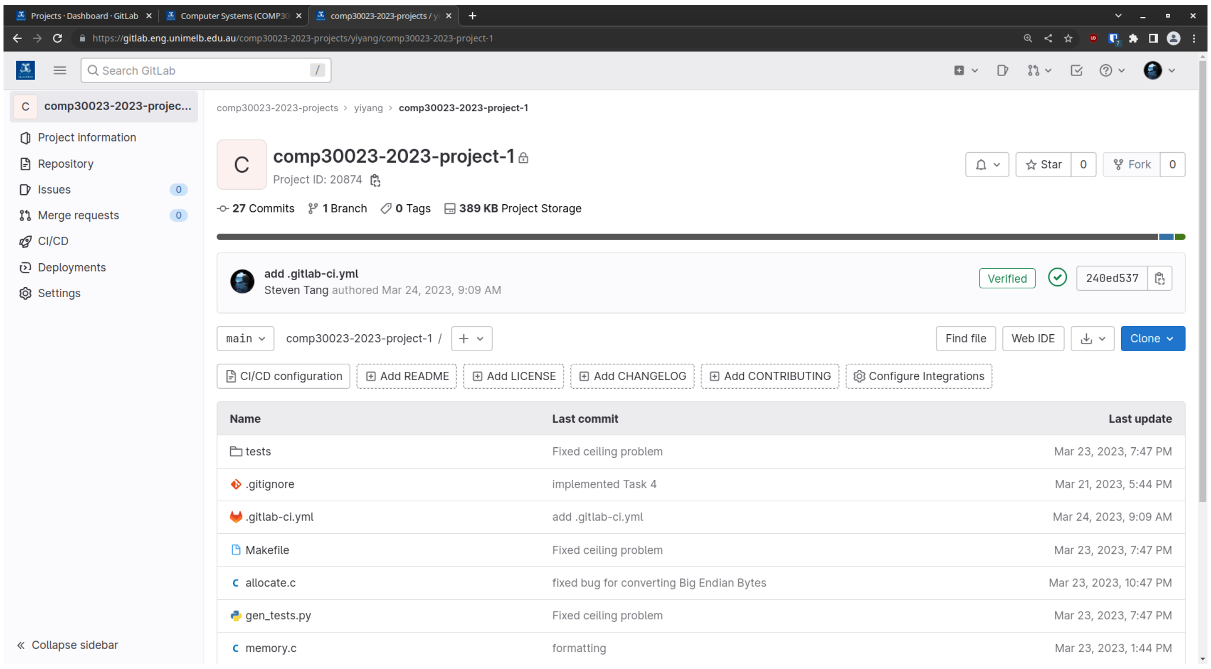
A breakdown of visible and hidden marks will be provided in project specifications.

We will endeavour to set up Gitlab Pipelines before the release of each project. The CI (Continuous Integration) will allow you to test your code against visible cases and receive some limited feedback before the submission deadline.

You can access the results of automated tests by following these steps:

1. Ensure that `.gitlab-ci.yml` is at the root of your repository.

   After placing `.gitlab-ci.yml` at the root of your repository, every pushed commit should trigger automated tests against your code.



comp30023-2023-project-1

Project ID: 20874

27 Commits · 1 Branch · 0 Tags · 389 KB Project Storage

add .gitlab-ci.yml
Steven Tang authored Mar 24, 2023, 9:09 AM — Verified — 240ed537

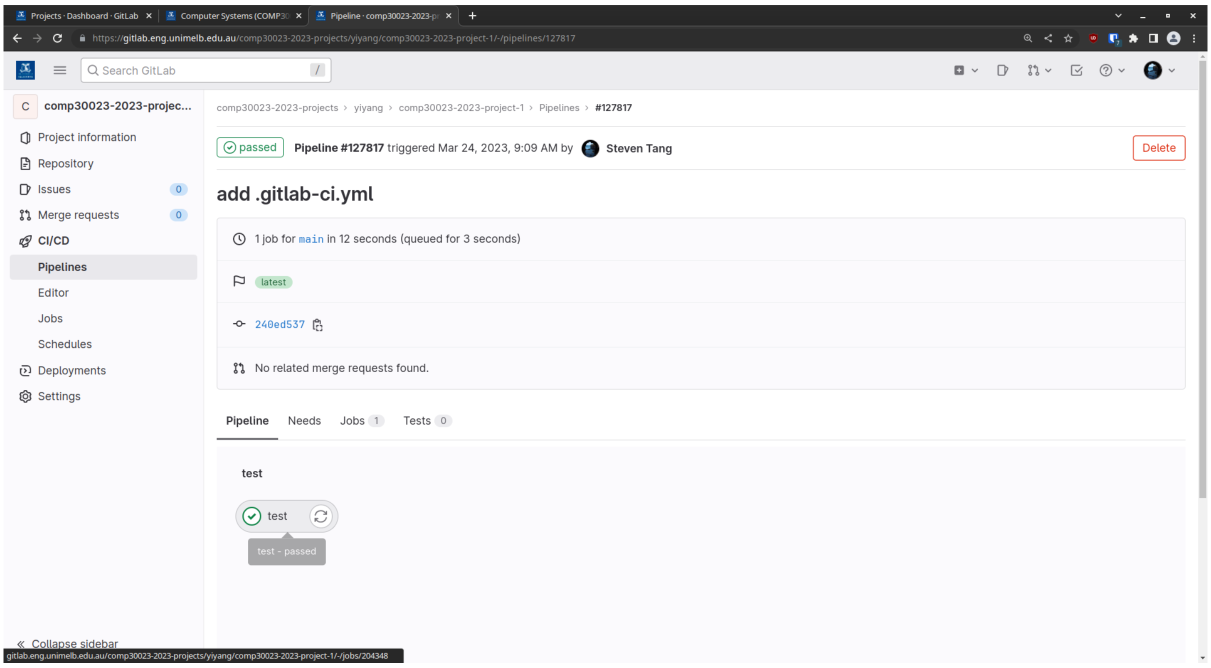
main / comp30023-2023-project-1 /

Name	Last commit	Last update
tests	Fixed ceiling problem	Mar 23, 2023, 7:47 PM
.gitignore	implemented Task 4	Mar 21, 2023, 5:44 PM
.gitlab-ci.yml	add .gitlab-ci.yml	Mar 24, 2023, 9:09 AM
Makefile	Fixed ceiling problem	Mar 23, 2023, 7:47 PM
allocate.c	fixed bug for converting Big Endian Bytes	Mar 23, 2023, 10:47 PM
gen_tests.py	Fixed ceiling problem	Mar 23, 2023, 7:47 PM
memory.c	formatting	Mar 23, 2023, 1:44 PM

2. After pushing a commit, you should see either a green tick mark, a blue pending progress

indicator, or a red cross to the right of the commit information. Click this icon.

3. On the next screen, click the icon again, and access the `test` pipeline stage.



comp30023-2023-projects › yiyang › comp30023-2023-project-1 › Pipelines › #127817

passed Pipeline #127817 triggered Mar 24, 2023, 9:09 AM by Steven Tang

add .gitlab-ci.yml

1 job for main in 12 seconds (queued for 3 seconds)

latest

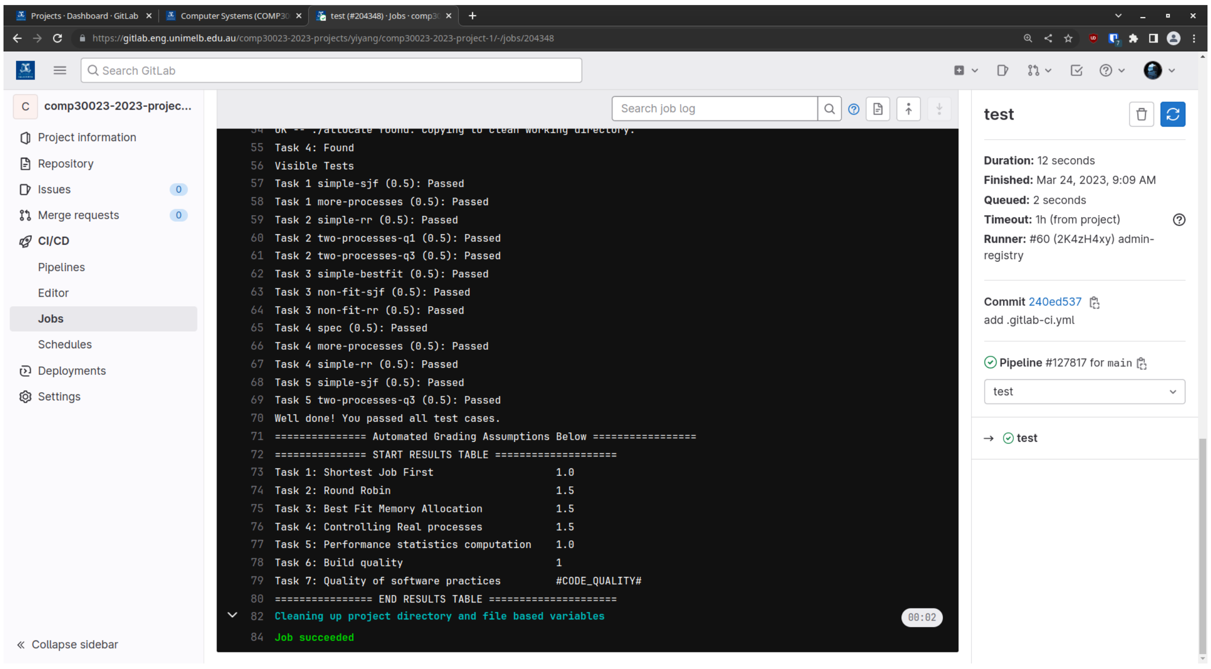
240ed537

No related merge requests found.

test

test - passed

4. This will bring you to the very bottom of the test transcript for the latest commit.



```
OK -- ./allocate found. Copying to clean working directory.
Task 4: Found
Visible Tests
Task 1 simple-sjf (0.5): Passed
Task 1 more-processes (0.5): Passed
Task 2 simple-rr (0.5): Passed
Task 2 two-processes-q1 (0.5): Passed
Task 2 two-processes-q3 (0.5): Passed
Task 3 simple-bestfit (0.5): Passed
Task 3 non-fit-sjf (0.5): Passed
Task 3 non-fit-rr (0.5): Passed
Task 4 spec (0.5): Passed
Task 4 more-processes (0.5): Passed
Task 4 simple-rr (0.5): Passed
Task 5 simple-sjf (0.5): Passed
Task 5 two-processes-q3 (0.5): Passed
Well done! You passed all test cases.
=============== Automated Grading Assumptions Below =================
=============== START RESULTS TABLE ====================
Task 1: Shortest Job First                    1.0
Task 2: Round Robin                           1.5
Task 3: Best Fit Memory Allocation            1.5
Task 4: Controlling Real processes            1.5
Task 5: Performance statistics computation    1.0
Task 6: Build quality                         1
Task 7: Quality of software practices         #CODE_QUALITY#
================ END RESULTS TABLE =====================
Cleaning up project directory and file based variables
Job succeeded
```

test

Duration: 12 seconds
Finished: Mar 24, 2023, 9:09 AM
Queued: 2 seconds
Timeout: 1h (from project)
Runner: #60 (2K4zH4xy) admin-registry

Commit 240ed537
add .gitlab-ci.yml

Pipeline #127817 for main

Note that the marks shown in the results table of the transcript is **indicative of marks** you will receive for **visible test cases (and tasks)**. Please compare it with the number of possible marks indicated in the specification.

The CI will fail (with a red cross) when *any* visible test case fails, and you may receive an email about a failing pipeline from Gitlab.

We do not care about whether you use the CI when implementing your projects. However, CI usage may be taken into account when considering extensions, plagiarism, and submission-related issues.

Unless an error or alternative solution has been identified, in which case marks will be retroactively awarded by re-testing all submissions, marks for automated components are final. There will be no partial marks given for uncompilable or incorrect programs, nor 'implementation effort'.

Please note that when your submission fails in CI, it will likely fail the same way in the marking environment. Debug your program **on your VM** in this case. Make sure that you're happy with what the CI reports.

5 Reading the Transcript

Here is an example transcript.
As it's pretty straightforward, you may want to skip this section.

5.1 Header

The header should show that the test was executed on a COMP30023 runner, and the version of the test script.

```
Running with gitlab-runner 15.9.1 (d540b510)
on admin-registry 2K4zH4xy, system ID: s_18ccc91b5924
...
$ /test.sh
COMP30023 2023 Project 1 Before Deadline Tests
v1, last modified 23/03/23
```

5.2 Commit Log

Next is the commit log. The top commit is the one that's being tested.

```
Commit log:
8ec56c2f8ced6bbcabb6a946a24ef50d3f9d9278: Remove strip.
9544fc6320a3c26672147d31f8112c8e1315b90c: feat: -e optional parameter
```

5.3 Compilation

The compilation process will then be shown. If you're missing marks for build quality, please look to this section.

Common issues:

- `make clean` is not implemented, or fails with non-zero status code
- There are dirty files committed to the repository, or `make clean` is non-functional
- Running `make` does not produce the required executable
- Code files were marked with executable bit, use `chmod -x <file>` to remove

```
make -B && make clean (output suppressed)

make clean
rm -f allocate process

make
gcc -c allocate.c -Wall -O2
gcc -c util.c -Wall -O2
gcc -o allocate allocate.o util.o -O2

OK -- ./allocate found. Copying to clean working directory.
```

5.4 Test case execution

```
Task 1 simple (0.5): Passed
Task 2 two processes (0.5): Passed
```

Each test will:

- Come under a task in the marking criteria
- Have a unique name (this will be reflect the names of sample test cases, if given)
- Have a weight
- Pass or fail
- (Possibly) have diffs or error messages

5.5 Results table

Finally, there is the results table.

```
=============== START RESULTS TABLE ====================
Task 1: Shortest Job First                    1.0
Task 2: Round Robin                           1.5
Task 3: Best Fit Memory Allocation            1.5
Task 4: Controlling Real processes            1.5
Task 5: Performance statistics computation    1.0
Task 6: Build quality                         1
Task 7: Quality of software practices         #CODE_QUALITY#
Project 1 (Total):                            #TOTAL_MARKS#
================ END RESULTS TABLE =====================
```

Rows with `##` are manually marked or excluded from CI.
This may include reports, challenge tasks, code quality etc.

Additionally, note that hidden cases are not included in the CI. Marks indicated are for visible cases only.

6 Conclusion

This concludes our brief introductory guide to COMP30023 projects.

Please let us know if you find any issues with our infrastructure and feel free to ask for clarification on any confusing aspects.