# COMPSCI4011 OS Coursework

Jeremy Singer

Mon 26 Feb 2024

| | |
|---|---|
| Submission deadline | **Mon 11 March, 1630** |
| Submit at | **Moodle file upload** |
| Coursework weighting | **20%** |

## Introduction

This document specifies the final coursework exercise for COMPSCI4011 Operating Systems (H). This is a code development exercise in C. As you know, the course has changed somewhat in scope and content over the past year, so this is a completely new coursework assignment.

I expect there will be some bugs to iron out — please inform me via email `mailto:Jeremy.Singer@glasgow.ac.uk` if you find any errors, inconsistencies or lack of clarity and I will do my best to issue updates and fixes as rapidly as possible.

In at least five of the OS labs, we have been working with the open-source MentOS system[1]. While I recognise this is experimental software, I believe it is a suitable basis for a coursework assessment—indeed, this is what the OS is explicitly designed for!

If you cannot get MentOS compiling and running on your device, do reach out to lab tutors or myself and we will help you out.

There are three tasks to complete for this coursework exercise, although the final task is marked as optional. It should be possible to get a very good 'B' grade without attempting the final task.

[1] `https://github.com/mentos-team/MentOS`

## Task 1

*I recommend you should spend no more than three hours on this task.*

Define a C function

```
char *get_cpu_info(void)
```

that returns a C string of information about the current CPU. You will probably want to use the x86 `CPUID` instruction as inline assembler. Alternatively you could use utility functions in mentos/src/hardware/cpuid.c — however please do not use `libcpuid` or similar, although you could consult such open-source code for inspiration.

What information would it be sensible to include about the current CPU? (After all, remember it is only an emulated CPU in QEMU.) Check `/proc/cpuinfo` on a running Linux system for a possible template.

Once you have a working C function, you should hook this up to the `procs_do_cpuinfo` function in the file `mentos/src/io/proc_system.c`

so you can execute the command

```
cat /proc/cpuinfo
```

and get sensible output.

## Task 2

*I recommend you should spend no more than five hours on this task.*

The Linux file `/proc/loadavg` records the number of runnable processes averaged over 1 minute, 5 minutes, and 15 minutes.

See details of the calculation at `https://archive.org/details/Linux-Journal-2006-12/page/n81/mode/2up`.

There is some other metadata relating to the number of processes in the system, and the PID of the most recent process.

Can you replicate the `/proc/loadavg` facility in MentOS? You will need to add an extra file in /proc/ and you will need to perform extra calculations in the scheduler to update the count of running/runnable processes.

Then add the relevant info to `/proc/loadavg` when it is read, e.g. with `cat /proc/loadavg`.

## Task 3 (optional, extension task)

*I recommend you should spend no more than three hours on this task.*

Write a MentOS utility program called `observe` that takes as command-line argument(s) a command to be executed every 2 seconds, and the output displayed on the screen. So, effectively, the `observe` function runs the command and displays the standard output every 2 seconds until you kill observe (with CTRL+c or similar).

What might be an interesting (changing) thing to observe in MentOS? Give a suggestion and add an output dump of the observe strings printed.

## Submission Requirements

Submit the output of git diff against a clean version of MentOS, master branch (git hash: `1832c77`). This should be a *single file* named `diffs.txt`.

Also submit a `changes.txt` file outlining what changes you have made to the source code and why - i.e. which changes were needed for each task.

If you had a go at Task 3, please also submit a `task3.txt` file - to explain how you attempted Task 3, and what you observed and any comments about it (no more than 400 words in total).

So, this makes a total of three text files to upload to Moodle:

1. diffs.txt
2. changes.txt
3. task3.txt

## Marking

**Total marks: 50**

### Task 1: worth 15 marks

- 5 marks for sensible, compilable, idiomatic C code
- 5 marks for useful CPU information, sensibly collected and presented

- 5 marks for successful integration with the existing procfs file named/proc/cpuinfo

## Task 2: worth 20 marks

- 5 marks for sensible, compilable, idiomatic C code
- 7 marks for appropriate calculation and presentation of CPU load average data
- 8 marks for successful integration with new procfs file named /proc/loadavg

## Task 3: (optional) worth 15 marks

- 7 marks for correct implementation of `observe` command
- 8 marks for evidence and comments regarding an interesting dynamic observation in MentOS