UG3 Operating Systems: Course Re-design

Michael O'Boyle & Tom Spink

1 Introduction

This document shall describe the proposed changes to be made to the **UG3 Operating Systems (INFR09015)** (OS) course, in order to increase its ranking from a 10-credit course to a 20-credit course. It shall describe proposed additions to content (Section 2), proposed re-design of the practical coursework (Section 3) and conclude with proposed time and award allocations (Section 4).

2 Course Content

In order to bring OS up to a 20-credit course, it is proposed to increase the number of lecturing hours from 15 to 20. This will allow the following topics to be introduced or extended:

- Boot Loaders
- Virtualisation
- Security and Privacy
- Symmetric Multiprocessing (and by extension lock-free algorithms)
- Real-time Kernels

3 Coursework Re-design

3.1 Background

The current coursework for **Operating Systems** is extremely outdated, using an ancient version of the Linux Kernel as a reference implementation. The coursework itself is based on premises that are no longer valid, and the infrastructure is beginning to age. Additionally, prior to academic year **14/15**, the coursework involved writing an essay about a particular function of an operating system but this was not deemed to be very challenging or valuable to the student, and so it was replaced with a new piece of practical work, in the form of a user-space shell implementation. This piece of coursework helped the students to re-inforce their knowledge of fundamentals that would be beneficial for the latter (unchanged) part of the coursework. Unfortunately, this incremental change suffered logistically, because in **14/15**, **all** the coursework (the shell and the kernel-module) was due to be handed in at the end of the semester, which means the students did not receive feedback about their user-space shell until the end of the course. This actually negates any benefit from having this coursework as an introductory piece, as they can't advance from any feedback. So, in **15/16** the coursework was split into two separate deadlines, with the first (end of week 4) being the user-space shell, and the last (end of week 11) being the original kernel module coursework.

3.2 New Coursework

As the coursework can now be considered to be very outdated, it is therefore proposed to completely re-design the coursework, in both content and structure.

The new coursework would be for the student to build components of an **operating system**, and would be designed to follow the structure of the course content. This would be realised by providing a

skeleton operating system to the student (in binary form), with modules that can be added and removed at compilation time, and a set of headers for students to compile their own modules against. A full (binary) *reference implementation* of the operating system would be provided so that the student can verify the infrastructure works, and it will be up to them to provide their own implementations of certain components as the tasks for each stage.

3.3 Structure

The coursework would be structured in to **three** distinct stages, with **three** separate deadlines:

Stage 1: 15 hours across three weeks (deadline week 4)

Stage 2: 15 hours across three weeks (deadline week 7)

Stage 3: 20 hours across four weeks (deadline week 11)

Total: 50 hours

Each stage could be implemented without predjudice from the other stages, so if a student was unable to complete previous stages, this would have no bearing on their ability to implement the later stages, as they can use the binary reference implementations in place of their buggy/incomplete versions.

All stages of the coursework would be provided to the student from the beginning of the course, giving them the opportunity to start any part of the coursework as early as they wish. It would also be possible to reorder **Stage 2** and **Stage 3**, to more closely follow the lecture structure.

3.3.1 Stage 1 - Introduction/Fundamentals

Stage 1 of the coursework would be an *exploratory* phase, where the student is introduced to the environment they will be working with in the latter two stages. They would be required to set-up the infrastructure, ensure they can boot a working version of the skeleton operating system, and implement a *small* file-system device driver for the operating system to boot a shell, thus ensuring they have the required fundamentals.

This stage would help the student to reinforce their ability to program in C/C++, which is a transferable skill and required for the next stages of the coursework.

It is expected that this stage of the coursework would take 15 hours.

3.3.2 Stage 2 - Applied Knowledge

Stage 2 of the coursework would be an *implementation* phase, where the student is asked to implement a memory allocator component for the operating system. Without a functional memory allocator, the skeleton operating system will not boot.

The memory allocation algorithm would be based on what they are currently learning about as part of the lectures and their own background reading (e.g. a buddy allocator?).

It is expected that this stage of the coursework would take 15 hours.

3.3.3 Stage 3 - Applied Knowledge

Stage 3 of the coursework would be an *implementation* phase, where the student is asked to implement a process scheduler component for the operating system. Without a process scheduler, the skeleton operating system will not be able to run threads/processes.

It is expected that this stage of the coursework would take 20 hours.

3.3.4 Assessment of Stages

These stages would be initially assessed by observing if the system boots as per the *reference* implementation, then by examining the student's code, and providing feedback on their implementation. Simple test cases for the student should be provided, along with more detailed test cases for the marker.

4 Allocations

This section outlines the proposed increase in time and award allocation.

4.1 Temporal

Item	Current	Proposed	Delta
Lectures	15	20	+5
Summative Assessment	2	2	0
Programme Level Learning and Teaching	2	2	0
Directed Learning and Independent Learning	56	126	+70
Coursework	25	50	+25
Total	100	200	+100

4.2 Award

Item	Current	Proposed	Delta
Written Exam	75%	70%	-5%
Coursework	25%	30%	+5%

Coursework Weight Increase The increase in weighting of the coursework is because the coursework now significantly contributes to the student's learning of the internals of operating systems, rather than being a superficial modification of an existing operating system.