Operating Systems 18/19 Practical Coursework

Tom Spink

tspink@inf.ed.ac.uk
IF-1.47

Overview

THE UNIVERSITY of EDINBURGH
informatics

The coursework is based on a research operating system called:

InfOS

Deadlines

THE UNIVERSITY of EDINBURGH
informatics

There are three distinct tasks:

1. Implement a round-robin process scheduler
   DUE: Week 3 31/01/2019

2. Implement a physical memory allocator, using the buddy algorithm
   DUE: Week 7 07/03/2019

3. Implement a device driver for a real-time clock
   DUE: Week 10 28/03/2019

All deadlines are **strict** and are on a **Thursday** at **4pm GMT**

Specification Document

THE UNIVERSITY of EDINBURGH
informatics

Available here:

https://www.inf.ed.ac.uk/teaching/courses/os/

Why aren’t we using Linux?

THE UNIVERSITY of EDINBURGH
informatics

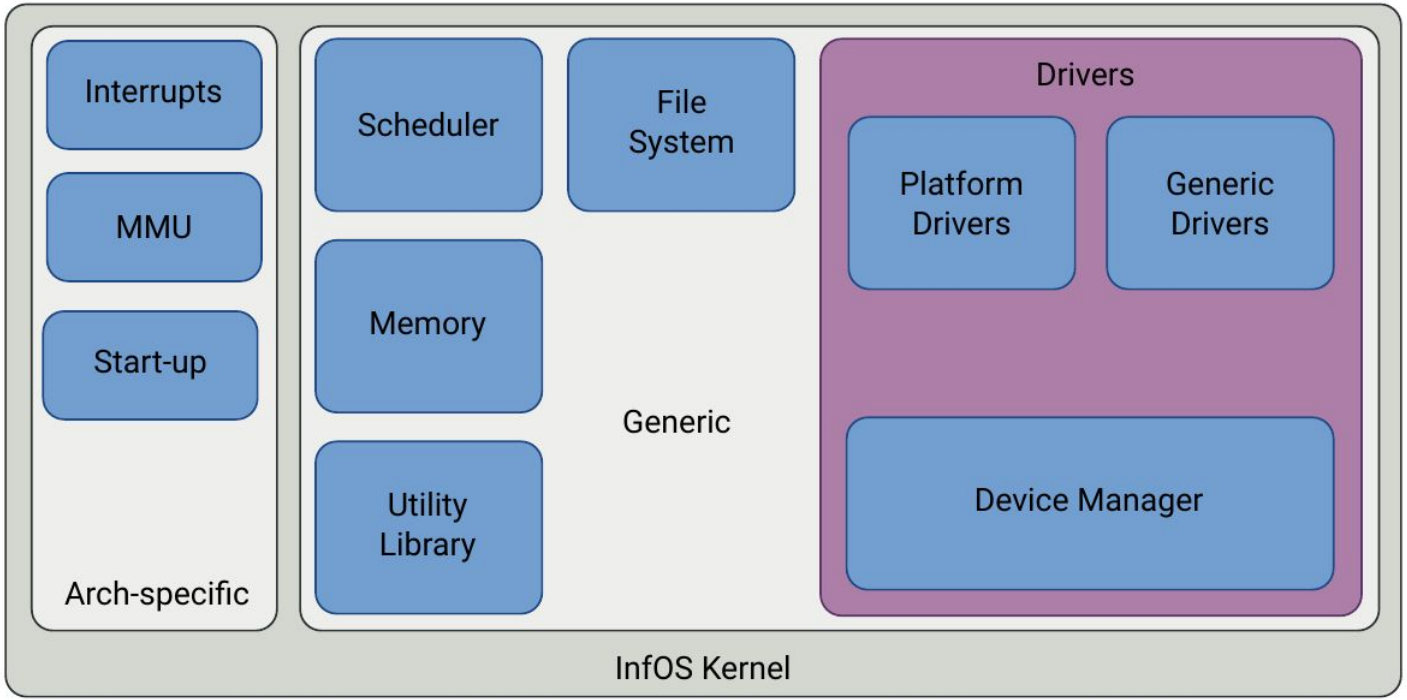
- The Linux kernel is very complex.
- It is semi-object oriented (C structs)
- It cannot be understood in its entirety (17 MLoC as of 4.5.4)
- It is not feasible to swap out fundamental infrastructure

Why are we using InfOS?

THE UNIVERSITY of EDINBURGH
informatics

- Designed to be straightforward to program for, and to understand.
- Written in modern C++, and based on object-oriented principles.
- Can be understood in its entirety (~20 kLoC)
- Easily plug in and out implementations of core infrastructure.
- Not too bothered about performance.

InfOS Overview



Obtaining, Compiling, and Running

On **DICE** machines:

```
$ scl enable devtoolset-7 bash
$ cd
$ /afs/inf.ed.ac.uk/group/teaching/cs3/os/prepare-coursework.sh
$ cd ~/os-coursework
$ ./build-and-run.sh
```

Obtaining, Compiling and Running

Inside the newly created `$HOME/os-coursework` directory:

- `build-and-run.sh`
  - Executes `build.sh`, followed by `run.sh` (if building was successful)
- `build.sh`
  - Compiles the InfOS kernel, the InfOS userspace, and your coursework.
- `coursework/`
  - Initially contains the coursework skeletons, but this will be the directory where you implement your solutions.
- `coursework-skeletons/`
  - Read-only copies of the skeleton files for the coursework tasks.
- `infos/`
  - Contains the checked-out InfOS kernel git repository.
- `infos-user/`
  - Contains the checked-out InfOS userspace git repository.
- `run.sh`
  - Launches the compiled InfOS kernel in Qemu.

Running over SSH


```
$ ssh student.ssh.inf.ed.ac.uk -X
$ ssh student.login -X
$ scl enable devtoolset-7 bash
$ cd ~/os-coursework
$ ./build-and-run.sh
```

Running on the Remote Desktop Service

THE UNIVERSITY of EDINBURGH
informatics

As a student, you have access to a remote desktop service, which will give you a graphical DICE interface.

Follow the instructions here:

http://computing.help.inf.ed.ac.uk/remote-desktop

Other Development Environments

...are not officially supported!

Your coursework solutions **MUST** compile and run on the DICE platform.

By all means, develop in whatever environment you feel comfortable with, but check that your solutions work on DICE.

Coursework that does not compile scores **zero**.

Resetting

If you've broken the InfOS kernel or user-space source-code repository, run the following command:

```
$ reset-repo.sh
```

This will delete any changes you have made to the InfOS kernel and user-space source code repository.

Your coursework will **NOT** be affected. If you need to start again, copy the appropriate skeleton out of the `coursework-skeletons/` directory.

Source Code Organisation

InfOS has two main parts:

- Architecture-specific part
  - x86-64
- Generic part

Source Code Organisation

- `arch/`
  - Architecture-specific code
- `arch/x86/`
  - x86-specific code
- `build/`
  - Build system support
- `drivers/`
  - Device drivers
- `fs/`
  - Virtual Filesystem Subsystem
- `include/`
  - C++ header files
- `kernel/`
  - Core kernel routines
- `mm/`
  - Memory management
- `util/`
  - Utility libraries/functions

Debugging

THE UNIVERSITY of EDINBURGH
informatics

- Execute `run.sh` or `build-and-run.sh` to launch InfOS in Qemu.
- Supports kernel command-line parameters to adjust debugging output.
  - e.g. `./run.sh pgalloc.debug=1`
- InfOS syslog goes to the terminal, and should be coloured.
- Implementation of a `printf`-style logging system, so that you can insert your own debug messages wherever you want.
- Press `Ctrl+C` in the terminal to quit Qemu, and shutdown InfOS.

Standard C++ Library

THE UNIVERSITY of EDINBURGH informatics

There is no standard C++ library

But, there is a limited utility library - and plenty of examples of its usage in the source code!

Generic Containers

- Generic `List<TElem>` container
  - Implemented as a linked-list.
  - Has methods for use as a queue (`enqueue`, `dequeue`)
  - Has methods for use as a stack (`push`, `pop`)

- Generic `Map<TKey, TElem>` container
  - Implemented as a red-black tree.
  - Has methods for insertion (`add`), lookup (`try_get_value`), and removal (`remove`).

- **Note:** These containers use dynamic memory allocation.

String Manipulation

THE UNIVERSITY of EDINBURGH
informatics

- Implementation of a standard C++ string class.
  - (but it's probably not useful for you)
- Standard C-style string manipulation functions.

C++ Programming

THE UNIVERSITY of EDINBURGH
informatics

- Use an IDE to help you
  - Netbeans
  - Eclipse
  - VSCode (more of a fancy text editor than an IDE, but still awesome)
- Use Google for your problems/issues (not for solutions!)
- Use classmates/piazza for general programming discussion

General Note on Solutions

- Insert appropriate error checking and validation.
- Use a clear coding style
  - Take inspiration from the InfOS source code. Use comments to help yourself and the marker understand your thought process.
- No compilation = zero marks
  - You have a compiler that tells you where and what the errors are.
- Efficiency
  - Try to use efficient algorithms in your implementation, so that the system remains responsive.
- Remove debugging statements
  - By all means use debugging print-outs during development, but they can slow the system down (writing to the virtual serial port is slow!) and can make it difficult to see what's happening.
- You **may not** rely on modifications to the InfOS core kernel code for your solution, but feel free to edit any aspect of the skeleton.

Task 1

A Round-robin Process Scheduler

Task 1: Round-robin Process Scheduler

THE UNIVERSITY of EDINBURGH
informatics

- InfOS has an interface for scheduling called the scheduling algorithm.
- Your job is to implement this interface, by creating a round-robin scheduler.

Task 1: Round-robin Process Scheduler

THE UNIVERSITY of EDINBURGH
informatics

- Provided skeleton is: `sched-rr.cpp`
- Implement these three methods, as per the specification:
  - `add_to_runqueue`
  - `remove_from_runqueue`
  - `pick_next_task`

Task 1: Round-robin Process Scheduler

THE UNIVERSITY of EDINBURGH
informatics

- Test by using the `build-and-run.sh` script
- If your implementation is broken, it's likely the system will hang.
- When you get to the shell, try running the scheduler tests:
  - `/usr/sched-test1`
  - `/usr/sched-test2`
- Also run some more programs to make sure starting and stopping processes works.
- Don't worry about output ordering (this is not important), just make sure multiple threads are starting and stopping.

Questions/Clarifications?