Operating Systems 20/21 Practical Coursework

Tom Spink

tspink@inf.ed.ac.uk

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 FOUR distinct tasks:

1. Implement a device driver
   DUE: Week 4 04/02/2021
2. Implement process scheduling algorithms
   DUE: Week 6 25/02/2021
3. Implement a physical memory allocator, using the buddy algorithm
   DUE: Week 8 11/03/2021
4. Implement a file-system driver for a TAR-based file-system
   DUE: Week 9 19/03/2019

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

Specification Document and Submission

THE UNIVERSITY of EDINBURGH
informatics

Available on the course Learn page:

http://course.inf.ed.ac.uk/os

Why aren’t we using Linux?

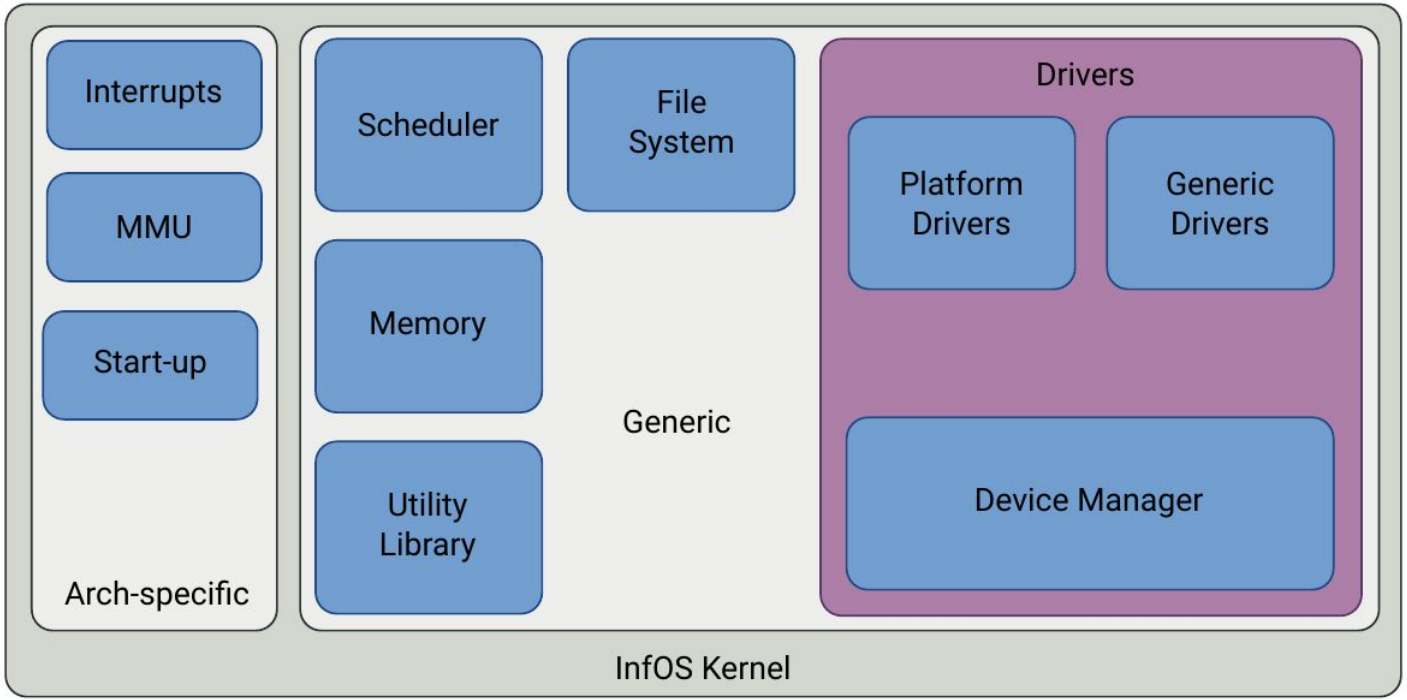
THE UNIVERSITY of EDINBURGH
informatics

- The Linux kernel is very complex.
- It is semi-object oriented (C structs)
- It cannot be understood in its entirety (**27.8 MLoC** as of 2020!)
- It is not feasible to swap out fundamental infrastructure

Why are we using InfOS?

- 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


```
$ git clone --recurse-submodules https://github.com/tspink/infos-coursework
$ cd ~/infos-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

Running on Remote DICE

THE UNIVERSITY of EDINBURGH informatics

Terminal 1

```
$ git clone --recurse-submodules \
https://github.com/tspink/infos-coursework
$ cd ~/infos-coursework
$ ./build-and-run.sh
```

Terminal 2

```
$ vncviewer localhost
```

Obtaining, Compiling and Running

Inside the newly created `infos-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.
- `infos/`
  - Contains the checked-out InfOS kernel git repository.
- `infos-user/`
  - Contains the checked-out InfOS userspace git repository.
- `reset-repo.sh`
  - Removes any changes to the `infos/` and `infos-user/` repositories, but does not touch the `coursework/` directory.
  - This action **CANNOT** be undone.
- `run.sh`
  - Launches the compiled InfOS kernel in Qemu.

Other Development Environments

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 **IRREVOCABLY 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, use the checkout command to revert a file to its original contents:

```
$ git checkout coursework/sched-rr.cpp
```

Source Code Organisation

InfOS is on GitHub:

- https://github.com/tspink/infos
- https://github.com/tspink/infos-user

I encourage you to explore!

InfOS has two main parts:

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

Source Code Organisation

THE UNIVERSITY of EDINBURGH
informatics

- `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.
  - You need to use `vncviewer` to see the graphics on remote DICE.
- 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

THE UNIVERSITY of EDINBURGH
informatics

- 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)
  - (and it uses dynamic memory allocation)
- Standard C-style string manipulation functions.

C++ Programming

THE UNIVERSITY of EDINBURGH
informatics

- Use an IDE to help you
  - Netbeans
  - Eclipse
  - VSCode (I use this, and there is a workspace file that might help)
- 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.