informatics

Operating Systems

Practical Coursework

Tom Spink
tspink@inf.ed.ac.uk

January 2017

Coursework

The coursework this year is based on a new research operating system called InfOS.

Coursework

There are three distinct tasks:

1. Implement a round-robin process scheduler (**due:** Week 3: 02/02/17)
2. Implement a physical memory allocator (**due:** Week 7: 09/03/17)
3. Implement a TAR file-system driver (**due:** Week 10: 30/03/17)

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

Visit the course webpage to get the specification document:
http://www.inf.ed.ac.uk/teaching/courses/os/

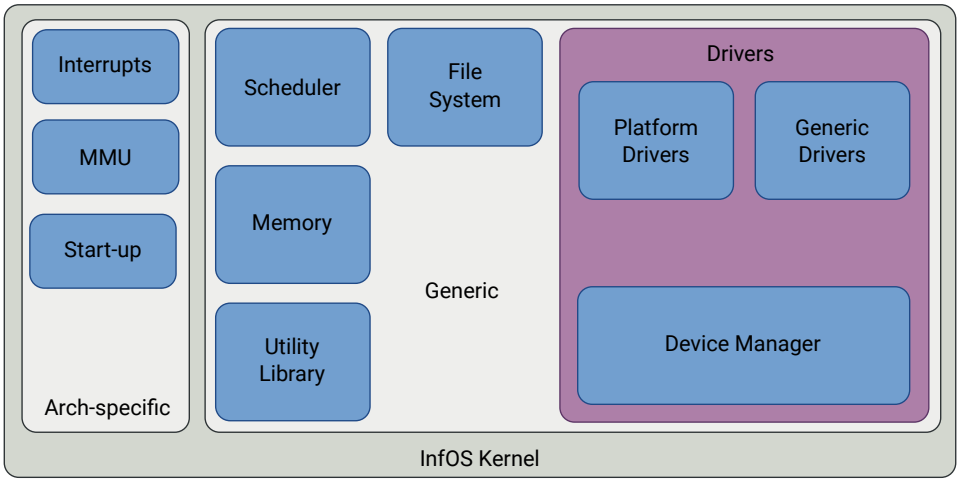
Problems using Linux

- The Linux kernel is very complex.
- 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.

Solution using InfOS

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

Overview


Interrupts
MMU
Start-up
Arch-specific
Scheduler
File System
Memory
Utility Library
Generic
Drivers
Platform Drivers
Generic Drivers
Device Manager
InfOS Kernel

Obtaining, Compiling and Running

On DICE machines:

```
$ cd
$ /afs/inf.ed.ac.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`, follwed by `run.sh` if compilation succeeded.

`build.sh` Compiles InfOS, the InfOS userspace and your coursework answers.

`coursework/` Initially contains the coursework skeletons, but this will be the directory where you implement your coursework solutions.

`coursework-skeletons/` Read-only copies of the skeleton files for the coursework tasks.

`infos/` Contains the checked-out InfOS git respository.

`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
$ cd ~/os-coursework
$ ./build-and-run.sh
```

Resetting

- If you've broken the InfOS source-code repository, run `./reset-repo.sh` to **delete** any changes you've made to the InfOS source-code, or the InfOS user-space source code.
  - This will **NOT** affect your coursework directory.
- If you need to start again, copy the appropriate skeleton out of the `coursework-skeletons` directory.

Source code Organisation

Two main parts:

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

Source code Organisation

`arch/`	Arch-specific code
`arch/x86/`	x86-specific code
`build/`	Build system support
`drivers/`	Device Drivers
`fs/`	Virtual file-system Subsystem
`include/`	C++ Header Files
`kernel/`	Kernel
`mm/`	Memory management
`util/`	Utility libraries/functions

Debugging

- 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 debugging output wherever you want.
- Press `Ctrl+C` in the terminal to quit QEMU, and shutdown InfOS.

Standard C++ Library

There is no standard C++ library.
But, there is a limited utility library—and plenty of examples of its usage in the source code.

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`).

String Manipulation

- Implementation of a standard C++ String class—but it's probably not useful for you.
- Standard C-style string manipulation functions.
  - `strlen`
  - `strncmp`
  - `strncpy`
  - `memcpy`
  - `memset`
  - `bzero`

C++ Programming

- Use an IDE to help you
  - Netbeans
  - Eclipse
- Use Google for your problems/issues (not for solutions!)
- Use classmates/piazza for general programming discussion

Task 1

Round-robin Process Scheduler

Task 1: Round-robin Process Scheduler

- 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

- Provided skeleton is `sched-rr.cpp`
- Implement these three methods:
  - `add_to_runqueue`
  - `remove_from_runqueue`
  - `pick_next_task`

Task 1: Round-robin Process Scheduler

- Test by using the `build-and-run.sh` script
- If your implementation is completely broken, it's likely that the system will hang.
- When you get to the shell, try running the scheduler tests:
  - `/usr/sched-test1`
  - `/usr/sched-test2`

Questions/Clarifications?