ICS-OS Lab 01: Building ICS-OS

Objectives

At the end of this activity, you should be able to:

1. set up the build environment;
2. build the ICS-OS kernel binary image;
3. build the ICS-OS distribution floppy disk image; and
4. boot ICS-OS and run two commands.

1 Introduction

ICS-OS is an instructional operating system that can be used for understanding different operating system concepts. An operating system is no different from other software in that it is written in a programming language, such as C.

The build process creates the compressed kernel binary image (`vmdex`) and the floppy disk image (`ics-os-floppy.img`). Since ICS-OS has a relatively large number of source files, the `Make` utility is used for the build. You can examine the contents of `Makefile` to learn more of the details how this is done.

To start ICS-OS, the floppy disk image is set as the boot device in Qemu, an emulator for various machine microarchitectures. The floppy disk image contains the GRUB bootloader which transfers control to the ICS-OS kernel (`vmdex`). After the boot process, a prompt is provided for users to enter commands.

2 Prerequisites

The recommended development environment is Ubuntu 16.04. Tasks described here may or may not work on other Linux distributions. Familiarity with the command line is also needeed. Install the following packages.

```
$sudo apt update
$sudo apt install qemu-system-x86 git
```

3 Deliverables and Credit

Perform the tasks below and capture screenshots while you do them. Submit a PDF file containing the screenshots with captions. Do not forget to put your name and laboratory section. Credit is five (5) points.

4 Tasks

Task 1: Install Docker and Docker-Compose

`docker`[1] and `docker-compose`[2] should be installed to build in the latest versions of Ubuntu (see footnote for links to the guides).

[1] https://docs.docker.com/engine/install/ubuntu/
[2] https://docs.docker.com/compose/install/

Task 2: Clone the repository and explore the source tree

ICS-OS[3] is open source and is hosted on Github. Run the following command to checkout the source code and explore the source tree.

```
$git clone https://github.com/srg-ics-uplb/ics-os.git
```

Take note where the source was cloned. We will refer to this directory in this document as `$ICSOS_HOME`.

Task 3: Build

The build process must be done inside a docker container. There is a `$ICSOS_HOME/ics-os/docker-compose.yml` and a `$ICSOS_HOME/ics-os/Dockerfile` that describe the build environment.

Open a new terminal. Start and enter the container using the commands below.

```
$cd $ICSOS_HOME/ics-os
$docker-compose run ics-os-build
```

You will be dropped to a root shell inside the container where you can perform the build. The `$ICSOS_HOME/ics-os` directory is mapped to `/home/ics-os` inside the container. Thus, you can perform the edits outside the container(in another terminal) and the changes will be reflected inside the build environment. The following steps will build the kernel image.

```
:/#cd /home/ics-os
:/#make clean
:/#make
```

Note that the details of the steps are in the `$ICSOS_HOME/ics-os/Makefile`.

Task 4: Boot

Open a new terminal. Build the boot floppy then start Qemu with the floppy image as boot device using the commands below.

```
$cd $ICSOS_HOME/ics-os
$make floppy
$make boot-floppy
```

You should now see the Grub boot menu. Simply press enter to boot ICS-OS. Note that the details of the steps are in the `$ICSOS_HOME/ics-os/Makefile`.

Task 5: Run ICS-OS commands

Once the ICS-OS command prompt (`%`) appears, type `help`. Examine the list of commands and run two of these commands. Do not forget to capture screen shots of the outputs.

[3] https://github.com/srg-ics-uplb/ics-os/

Task 6: Cleanup

To exit the build containerb.

```
:/#exit
```

You can go back to the build container by performing `Task 3` above.

