jake: Just a Kernel

First time setup for the practicals

Computer Engineering · 227-0013-00L

Spring Semester 2022

Before you start

The installation instructions that follow assume a very basic knowledge of the command line: you need to know how to execute commands, specify files, and move between directories.

Have you never used a terminal before?

- For **macOS users** there is this excellent guide from Apple. Open the Table of Contents and read the section **Work in Terminal windows** up to and including the subsection **Specify files and folders**. This should not take more than 10 minutes.
- We assume that **Linux users** will know how to use the command line.

- **Windows users** should use the Windows Subsystem for Linux (WSL 2). This will allow you to use the same kind of terminal used by macOS and Linux. However, to install it we need to use Windows' own terminal, the Command Prompt or `cmd.exe`, which works in a different way and uses different commands than the terminals used by macOS and Linux. On the next slide we will explain how to install the WSL 2.

Installing the WSL 2 (Windows users only)

- Install the WSL 2 using the official instructions from Microsoft here
- The instructions say that “by default, the installed Linux distribution will be Ubuntu” which is perfectly fine, no need to change it

Installation

We have prepared a user environment for you that has everything you need for working on the practicals. Please use it! It will also guarantee that all students have exactly the same version of each tool, which will make both your life as well as our lives much easier when we help you debug your kernel.

You have two options for installing the user environment. **Please note that both will consume around 8 GB of disk space.**

We recommend macOS users to go with Docker (Option 1), and Windows users to go with Nix (Option 2). For Linux users, both options should be relatively easy and straightforward, pick one!

Finally, `any text that looks like this is a command, meant for a terminal.` For **Windows users**, that is the Ubuntu terminal inside the WSL 2, which you can open by typing `wsl` in the Command Prompt.

Option 1: Docker

Recommended for macOS users that have not yet installed Nix

This will isolate the user environment from the rest of your system, and only requires you to install Docker. However, it does mean that you will have to work inside a Docker container, which is not always that convenient.

1. Install Docker (Desktop) using the official instructions here. Alternatively, if you are **using macOS** and have Homebrew installed, you can install Docker using Homebrew: `brew install --cask docker`
2. Download the image here
3. Load it using `docker load < jake.env.tar.gz` (you may need to have opened Docker at least once before this works)
4. Start a container with `docker run -it -v $(pwd):/workdir computer-engineering/jake`. This will mount your working directory at `/workdir` inside the container. Make sure that it contains the latest homework assignment (i.e., `workdir` should contain `env`, `README.md`, etc.). Exit your container by typing `exit`.
5. If you do `docker ps -a` you should see a container there, with some funny name, in my case `dazzling_torvalds`
6. You can now open a shell inside that container by issuing `docker start -i dazzling_torvalds`

What you need to know about Docker

The large `jake.env.tar.gz` image that we gave you is a Docker image. By typing `docker run` (see previous slide) you have started a container based off that image. In other words, a container is an instance of an image. You can start many different containers all based on the same image.

Why is this relevant? If you create a new file inside a container, and then remove that container using `docker container rm`, the file will be lost forever. By typing `exit` your do not remove the container, however, you only stop it. Calling `docker start -i` restarts it.

The directory that you mount inside a container using `-v` does not get removed if you remove the container. After all, that directory is not really part of the container, but instead resides on your own file system, outside of Docker. For this reason, make sure to do all your work inside `workdir`, in case you accidentally remove your container. . .

Optional: Installing additional packages

Our Docker container is based on NixOS. If you would like to install additional packages, use `nix-env -qaP <some_package>` to list available packages and `nix-env -iA <some_package>` to install a package.

Option 2: Local installation using Nix

Recommended for Windows users

This will install the user environment locally. It requires you to install Nix and **may interfere with your current environment**, for example, it may temporarily override the version of Git that you are using.

1. Install the Nix package manager (not NixOS) using the official instructions here. It does not matter whether you perform a single- or multi-user installation. **Using Windows?** Perform the single-user installation inside the WSL 2 as recommended here
2. Install the environment with `bash ./env/nix/install.sh`. This might take a while (20 min. on my machine)
3. Install the man pages with `bash ./env/nix/install.sh --manpages`. You will have to do this again when a new assignment becomes available (see below)

To uninstall the environment, simply do `bash ./env/nix/install.sh --uninstall` and optionally remove Nix using the official instructions. (If you want to free space but keep Nix around, you may additionally want to run `nix-collect-garbage -d`.)

Verify your installation

To verify that you have installed the environment correctly, run the following command, either in your Docker container, WSL 2, or in your shell of choice if you are using Linux:

```
riscv64-unknown-linux-gnu-gcc --version
```

You should get... (shortened)

```
riscv64-unknown-linux-gnu-gcc (GCC) 11.1.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the ...
```

Documentation

Each homework assignment has its own man page, which you can read using for example `man 1 asm` (which will open `asm(1)`). The following man pages **will become available over time**:

- `general(0)`
- `asm(1)`
- `boot(2)`
- `buddy(3)`
- `paging(4)`
- `cache(5)`
- `user(6)`
- `multiprocessing(7)`

The special page `general(0)` contains general instructions for the kernel assignments. `boot(2)` is special in the sense that it is not a homework assignment: there's nothing for you to hand in that week.

Distribution of new assignments

Each homework assignment will be made available on Moodle in the form of a tarball and through Git at `git@gitlab.ethz.ch:comsec-internal/jake-fs22.git`.

There is a difference: the tarballs on Moodle contain the solutions of previous assignments, whereas the assignments made available through Git will only add new code and possibly bug fixes for old code that you should not have touched. In this way, you will be able to reuse you own work, and we prevent our solutions from conflicting with yours. **It is important to realize that each assignment builds on top of the next, especially the kernel assignments.**

In other words, you have a choice to make:

Option 1: Moodle + using our code + one directory per assignment

You can unpack the tarballs from Moodle using `tar -xvzf <homework_assignment>.tar.gz`. Simply unpack each assignment in **its own directory** and “start over” each time. In this way, you will not touch the code that you’ve written for previous assignments. It also means that you will automatically use our code (the solutions) for previous assignments.

Option 2: Git + using your own code + all assignments in one directory

Writing your own kernel is cool and we therefore encourage you to try and reuse the code you have written for earlier assignments, at least partially. **However, this does require you to use Git.** Or rather, it serves as a great way to learn Git!

Start by cloning our repository: `git clone git@gitlab.ethz.ch:comsec-internal/jake-fs22.git`. Do you work, submit it, and then, when a new assignment becomes available, simply pull using `git pull` and possibly deal with merge conflicts. We will do our best to minimize the chances of you getting a merge conflict, but avoiding them is also your responsibility: try to not change code that we do not ask you to change (i.e., that is not part of a **TODO**). Adding code should be fine, however.

Finally, if you go for this option, you are of course free to look at and use the solutions that we make available on Moodle. For example, when you did not finish an assignment, or simply if you would like to make your kernel more robust.

Installing new man pages

With each homework assignment comes a new man page. You can find them in `./env/man`. You can reinstall the man pages in that directory using `bash ./env/nix/install.sh --manpages`.

Docker users will have to run this command inside their container. Make sure to use the `env` directory in your `workdir` and not `/env` i.e., for Docker users the command is `bash /workdir/env/nix/install.sh --manpages`. You will also have to run this command again if you accidentally (or intentionally) remove your container.

Optional: Text editors

To write code you need a proper text editor. We recommend to use either Vim or Visual Studio Code, but you are of course free to use any other editor as well. Our Docker container comes with Vim preinstalled.

If you have never used Vim before: it takes a while to get used to it. To start learning Vim, check out `vimtutor`.

Windows users that would like to use Visual Studio Code may want to read this, to set up Visual Studio Code with WSL 2.

Known issues

- Installing `man-db` using Nix (natively) breaks `apropos`. Probably related to this. If this bothers you, and you already had `man` installed anyway, simply uninstall it using `nix-env -e man-db-2.9.4`
- You may get this warning while working on `asm(1)`: `<stdin>:20.39-24.9: Warning (interrupt_provider): /cpus/cpu@0/interrupt-controller: Missing #address-cells in interrupt provider`. Simply ignore it
- You may get this warning when starting a Docker container on macOS: `WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested`. Simply ignore it

- Do you get a `No space left on device` error? This might happen if you are using Linux and going for Option 2. **It should not be a problem for Windows users. For Linux users, the easiest solution is therefore to with Docker instead of Nix.** If you would really like to use Nix, however, you may try the following: The problem is probably caused by a lack of space on either your / (root) partition or `/tmp`, and not necessarily because your disk is full, so removing "normal files" won't help. You should have around 30 GB available on your root partition, which you can verify using `df -H`. (After the installation completes, it will clean up, so that in the end our environment consumes "only" around 8 GB.) What could help is either removing unused Docker images and/or containers (see `docker container prune` and `docker rmi`) or running `nix-collect-garbage -d`. **WARNING:** `nix-collect-garbage -d` will remove old environments, make sure that you know what you're doing, and otherwise talk to a teaching assistant first