DT149G — Building a custom kernel

Lennart Franked[*]

October 31, 2025

Contents

1 Introduction

In this laboratory assignment you will start by first installing linux in a virtual machine, after which you are going to configure and build a customized kernel for your system. Usually one never need to bother with creating a customized kernel, however, when one is given a task of administrating several servers running a unix-like operating system there might be times where a custom kernel is needed, for example if your server is running some special devices that isn't supported in the standard kernel.

2 Aim

After completion of this assignment you will have:

- Knowledge on how to customize the kernel of your system.

[*] E-post: lennart.franked@miun.se.

3 Reading instructions

Before starting this assignment you should have read Nemeth et al., 2011, chapters 13.1-13.3, 13.7–13.9 or Nemeth et al., 2017, chapters 2.1-2.4, 2.6, 2.9, 11.1–11.4, 11.6-11.9 and 24. For task 4.4, consult “Kernel/Compile”, 2024 for a guide on how to compile a kernel.

4 Tasks

4.1 Installing a hypervisor

There are numerous hypervisors available on the market. For this course, it is recommended to use a Type 2 hypervisor for simplicity. See, for example, VirtualBox Oracle, 2024.

Read about the different hypervisors available and install the one you prefer. *Note: if you are performing this lab on the university computers, you must use VirtualBox, since it comes preinstalled on all the computers in L207.*

To answer in your report *Which hypervisor did you choose, and why? Your answer must discuss at least two hypervisors, weighing their advantages and disadvantages. Include relevant references and quotations from the course literature.*

4.2 Installing Linux

Once you have selected your hypervisor, it is time to choose your operating system.

In this course, all labs are written for Ubuntu Server; however, you are free to install any Linux distribution of your choice. Download and install the operating system in your hypervisor.

Once the operating system is installed, ping `www.miun.se` to ensure that your network connection is functioning correctly.

4.3 Before starting

Now that your system is operational, we will begin customizing the kernel. Before proceeding, open a terminal window and run:

```
uname —a
```

To answer in your report *Take a screenshot and include it in your report.* Explain the output of this command.

4.4 Compile the kernel

Compiling a custom kernel is rarely required; however, undertaking this process offers valuable insight into the internal mechanisms of the Linux operating system. Numerous guides describe the procedure for compiling and installing a new kernel in Linux, and most follow the steps outlined below.

- Obtain the kernel source code.
- Specify the components and features to be included in the kernel.
- Compile the kernel.
- Install and boot the newly built kernel.

If the build configuration is not carefully defined, the resulting kernel build files may become excessively large, potentially exhausting available disk space or memory. Even with ample resources, the compilation process can be extremely time-consuming. Moreover, omitting essential modules or drivers required by the system will result in an unusable kernel. Therefore, the configuration phase, in which the components to be compiled are specified, is the most critical step in the process.

4.4.1 Compiling the kernel the Ubuntu way

The guide “How to build an Ubuntu Linux kernel” “Kernel/Compile”, 2024 provides the most up-to-date procedure for building an Ubuntu Linux kernel. It is advisable to adhere to the latest method and begin with this guide.

When reaching the step “Modify kernel configuration”, instead of executing the `editconfig` command, and manually configure what to include in the kernel, it is recommended to base this on the current running kernel, instead. For this, proceed as follows.

Assuming you are located in the kernel source working directory:

- Generate a kernel configuration using `make oldconfig`.
- Copy the resulting `.config` file to your home directory, for example:
  `cp .config ∼/kernconf`.
- Run `make ARCH=‘uname -m‘ mrproper` to remove the `.config` file, since this configuration is not fully compatible with the Ubuntu-specific kernel build process.
- Import the previously saved kernel configuration using:
  `./debian/scripts/misc/annotations --arch ‘uname -m‘ --flavour generic --import ∼/kernconf`.
- Update the configuration so that the build scripts utilize the imported settings:
  `fakeroot debian/rules clean updateconfigs`.

After completing these steps, continue by following the remaining instructions in the guide to compile and install the kernel.

To answer in your report

- What is the purpose of running `make oldconfig`?
- *Take a screenshot of the kernel compilation process.*

- After building and installing the kernel, reboot your system and select your newly built kernel from the GRUB loader. Once you have logged in, run the command again:
  `uname -a`

- *Take a screenshot and include it in your report.*

- Briefly describe any issues you encountered while building and installing your kernel.

4.5 Troubleshooting

When issues arise during the kernel compilation process, the following recommendations may assist in identifying and resolving them:

- Troubleshooting is an iterative process. Carefully study the output produced by the build output. For systematic analysis, redirect the output to a file.

- Keep in mind that using large language models (LLMs) may not always provide the most up-to-date answer or one that is specific to your setup. Their output should therefore always be verified against official documentation before execution.

- Carefully review the output when an error occurs. It is often necessary to scroll upward in the terminal output to locate the initial or primary error message.

- Additional software packages may need to be installed during the build process to satisfy missing dependencies.

- Consult the `makehelp` **makehelp** documentation available on `kernel.org` for further guidance.

  - Begin by compiling a kernel with no configuration options enabled. Although this will produce a largely non-functional kernel, it serves as a useful verification that the build system is operating correctly.
  - Subsequently, repeat the process using `tinyconfig` instead of `allnoconfig`, which enables a minimal yet functional configuration and may simplify troubleshooting.

- Clean the kernel source directory between attempts using `make mrproper` and `fakeroot debian/rules clean`.

5 Examination

In your report, you should, in addition to the three required screenshots, motivate your choice of hypervisor and distribution, and include a description of the process of building your own custom kernel.

References

Kernel/compile. (2024). Retrieved April 15, 2025, from https://canonical-kernel-docs.readthedocs-hosted.com/latest/how-to/develop-customise/build-kernel/

Nemeth, E., Snyder, G., Hein, T. R., & Whaley, B. (2011). *Unix and linux system administration handbook* (4th ed.). Prentice Hall.

Nemeth, E., Snyder, G., Hein, T. R., Whaley, B., & Mackin, D. (2017). *Unix and linux system administration handbook* (Fifth edition.). Addison-Wesley/Pearson.

Oracle. (2024). *Oracle virtualbox.* https://www.virtualbox.org/