# 6.S913 Fundamentals of Linux Systems

Juni Kim (junickim)

January 29, 2026

This class is intended as a crash course for students to understand the basics of how modern Linux systems operate and for students to construct one from scratch.

## Staff

**Instructor** - Juni Kim (junickim), https://junic.kim

**Teaching Assistant** - Jacky Zhao (icecream)

## Prerequisites

Students are expected to be comfortable working in Unix/Linux environments and using the command line as their primary interface for development.

It would be strongly preferred that students can:

- Navigate and manipulate the filesystem using shell commands
- Write and debug simple shell scripts
- Understand environment variables, PATH resolution, and program invocation
- Understand how to compile software from source using tools such as `make`

Prior coursework in operating systems or systems programming is helpful but not required.

## Logistics

Please join the piazza at https://piazza.com/mit/spring2026/6s913. Announcements and any other class discussion will take place there.

Lectures and lab hours will take place at **34-301**. We will try our best to record lectures and allow for remote participation.

The course will move quite fast, so lecture and lab hour participation is strongly recommended. Lectures 1-4 come with an associated handout that students are expected to read to complete the assignment.

## Assignment

The assignment starter source code is available at https://github.com/junikimm717/fls-assignment-public.

Students will be responsible for reading the setup instructions in the README. To receive credit, students must submit a tarball with their project to the portal at https://6s913.mit.junic.kim/.

The assignment for this course has four main parts, named `busybox`, `kernel`, `user`, `image`, which represent different stages of an OS bootstrapping build system. All parts must be completed and integrated together to receive credit for the course. The parts all rely on one another. The assignment must be submitted by **Friday, January 30, 2026 11:59PM GMT-5**.

A significant part of this course is learning to maintain clean, reproducible filesystem state. Projects will be evaluated on a fresh system with no network connection, and any reliance on undeclared or residual state (e.g. working directory, undeclared environment variables, artifacts that weren't cleaned up) will be treated as a correctness issue.

The assignment should be done on sufficiently modern hardware (for reference, you should expect to compile the linux kernel). Docker is required on all systems, and Windows users are strongly advised to use wsl.

There will be multiple lab hours designed to help students with the checkpoints.

## Policies

### Alternate Grading

The primary and expected path to receiving credit for this course is to submit a project that passes the automated grading infrastructure. Students are strongly encouraged to aim to complete the full assignment and pass the autograder.

For students who have attended the first four lectures and have made a serious, good-faith effort on the assignment, an alternative evaluation path may be available. In this case, students may request a meeting with course staff to demonstrate a working system and explain their design decisions, debugging process, and understanding of the material.

Credit under this alternative path is granted at staff discretion and is intended for students who clearly understand the core concepts of the course but were unable to complete the full build pipeline. This option is not intended to replace the autograder and will not be offered to students who have not engaged with the course.

Details about requesting such a meeting, including eligibility and expectations, will be announced later.

### AI Use

Use of AI tools is permitted. We encourage using AI in a similar way to a search engine, reference manual, or debugging assistant (e.g. clarifying error messages, understanding documentation, or recalling command syntax).

However, use of agentic IDEs or tools that substantially offload reasoning, design decisions, or understanding is strongly discouraged. Submitting work you do not understand defeats the purpose of the course.

A central goal of this course is learning to reason about Linux systems rather than cargo culting commands or configuration (i.e. copying commands or options without understanding why they are needed). You are expected to be able to explain and justify the commands and design choices used in your submission.

## Integrity

Students are expected to submit work that they have honestly authored and understand.

Course staff reserve the right to investigate and adjust grades for submissions that attempt to subvert the assignment, grading process, or course infrastructure. This includes, but is not limited to:

- Injecting malicious or intentionally disruptive code into build scripts or artifacts
- Attempting to interfere with the grading environment or its execution
- Hardcoding outputs or behaviors specifically to pass the grader without correctly implementing the required system
- Submitting build pipelines that do not genuinely construct a working system, even if they appear to pass automated checks

While the grading infrastructure includes safeguards against many such behaviors, these safeguards are not exhaustive. Submissions are expected to reflect a genuine, correct build process rather than exploit assumptions or weaknesses in the grader.

If a submission is found to violate the spirit or intent of the assignment, course staff may adjust grades accordingly.

## Schedule

All lectures and lab hours are expected to take place at **34-301**.

### Lecture 1 - Tuesday 1/20 1-3PM

- Course structure, expectations, and assignment overview
- Writing correct shell scripts in hostile environments
- Working directory, filesystem state, and environment variables
- Host vs target systems
- Docker as host-system standardization

### Lecture 2 - Wednesday 1/21 1-3PM

- Build pipelines and required invariants
- Source vs artifact distinction
- C build systems: configure, make, install
- Compiler and linker flags, libc concerns
- Using QEMU and why kernels need an initramfs

### Lab 1 - Wednesday 1/21 End of Lecture-5PM

Help will be available for all parts of the lab, but we will prioritize those with questions about lab setup and the `busybox` section.

### Lecture 3 - Friday 1/23 1-3PM

- Kernel role during boot and early system bring-up
- Manual kernel configuration with `make menuconfig`
- Kernel command line and serial console configuration
- Initramfs construction and pseudo-filesystems (/proc, /dev, /sys, ...)
- `switch_root` and transition to real userspace

### Lab 2 - Friday 1/23 End of Lecture-5PM

Help will be available for all parts of the lab, but we will prioritize those with questions regarding the `kernel` and `image` section.

By this point, students should have a compiled and working kernel binary and the ability to boot up a kernel into an initramfs.

### Lecture 4 - Tuesday 1/27 1-3PM

- Persistent userspace and filesystem hierarchy
- Init systems and BusyBox init configuration
- Users, groups, and permission-critical files
- Essential system daemons (getty, eudev, dhcpcd, chrony)
- Disk images, filesystems, and EFI boot flow
  - GUID Partition Table

### Lab 3 - Tuesday 1/27 3-5PM

Help will focus on the `image` and `user` sections of the lab.

### Lecture 5 - Thursday 1/29 1-3PM

- Review of core system-building concepts
- Additional topics:
  - SUID binaries and controlled privilege escalation
  - User and group configuration
  - Time, certificates, and other system-level configuration

## Lab 4 - Thursday 1/29 End of Lecture-5PM

Help will focus on the `image` and `user` sections of the lab.

## Assignment DUE - Friday 1/30 11:59PM ET