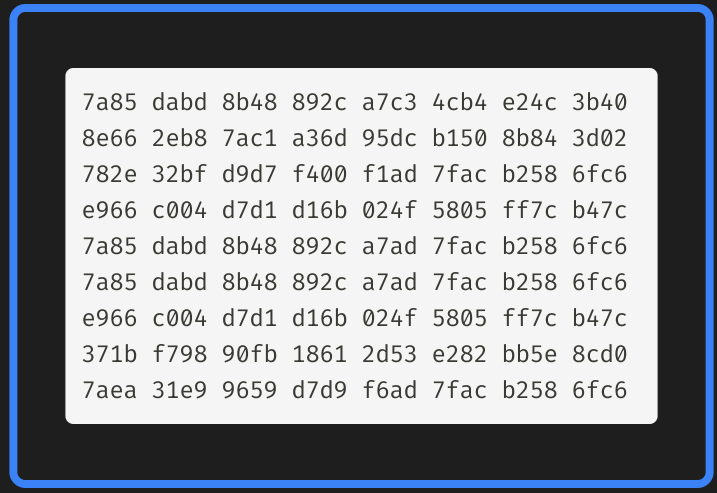
Platform and Code Analysis

Identifying system components and properties

Start Week 2 →



Review

Any questions on what we covered last week?

Platforms

Recent years has seen a proliferation of many different devices and platforms. The first step to reverse engineering one is understanding the platform.

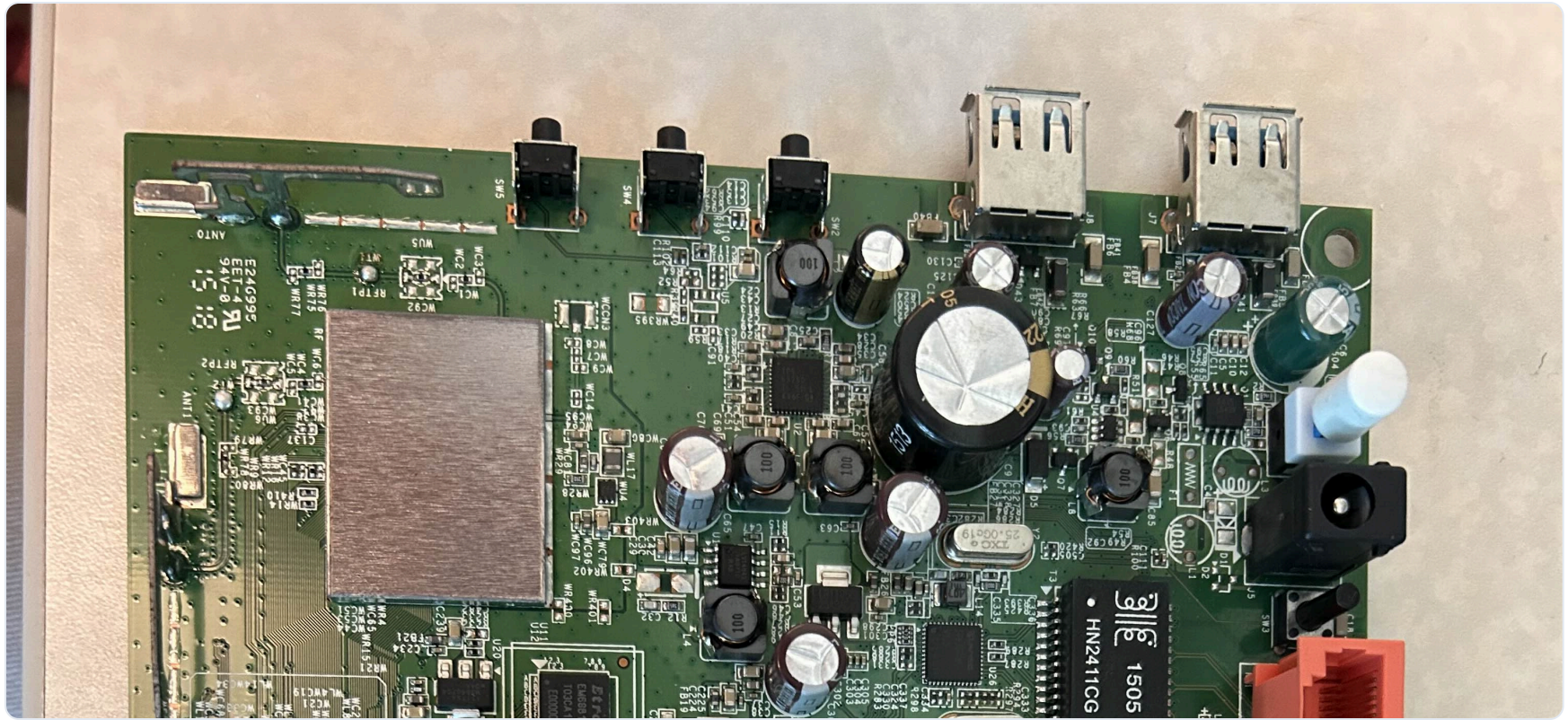


When analyzing a device we ask a variety of questions.

- **Hardware**: What kind of device is it? What hardware components does it have?
- **Operating System**: What's the operating system? What version is it? Has it been modified?
- **Applications**: What services does it run? What ports? What dependencies?

Hardware

The first step is usually understanding the hardware.



Operating System

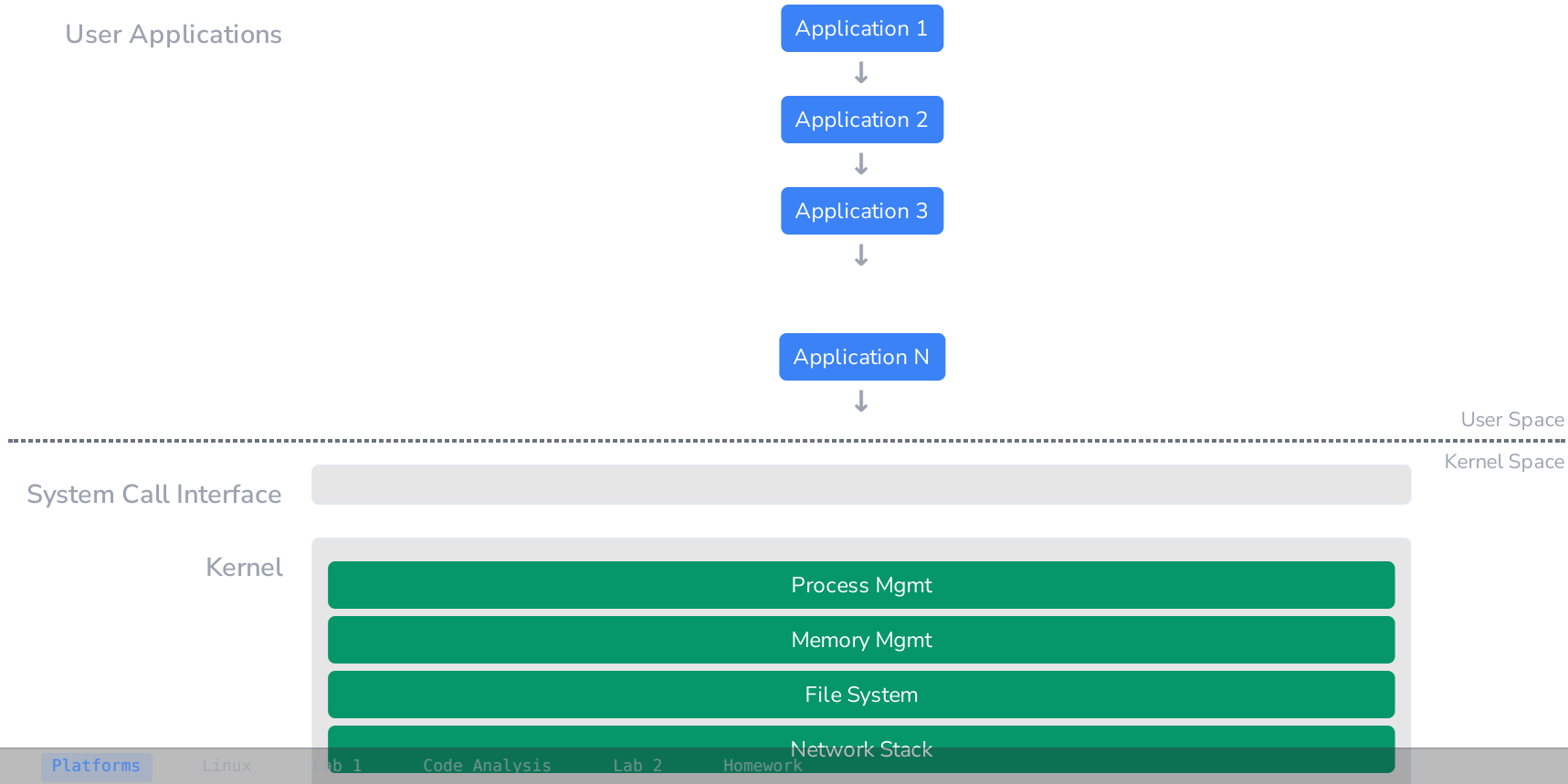
Some devices run the firmware directly (Arduino, Network Card, Motherboard, SSD, some IOT devices), but these days most have an operating system, whose primary purpose is providing an interface between user applications and the hardware.





Operating System Architecture

Most operating systems follow this general architecture.



Test Your Comprehension

Here's some questions.

- What's the *difference* between user and kernel mode?
- What's the *interface* between user and kernel mode?
- Which of these are handled by the user processes vs. the kernel?
  - `printf` formatting
  - Command parsing
  - Process creation
  - Heap management
  - Device drivers

Linux

Since most customized platforms run a linux variant, we'll focus on it as our operating system for platform analysis.

We'll look at

- Versioning
- File system locations to triage
- Application triage

System Versions

One critical piece of information for most platforms is its *version*.

Where can I find the operating system version?

```
$ cat /etc/issue
Ubuntu 24.04.1 LTS \n \l
```

```
$ cat /etc/os-release
NAME="Ubuntu"
VERSION="24.04.1 LTS (Noble Numbat)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 24.04.1 LTS"
VERSION_ID="24.04"
```

Where can I find the kernel version?

```
$ cat /proc/version
Linux version 5.9.0-41-generic (buildd@... ) (gcc (Ubuntu 13.3.0-6ubuntu2) 13.3.0) #41-Ubuntu SMP PREEMPT_DYNAMIC Fri Aug 16
```

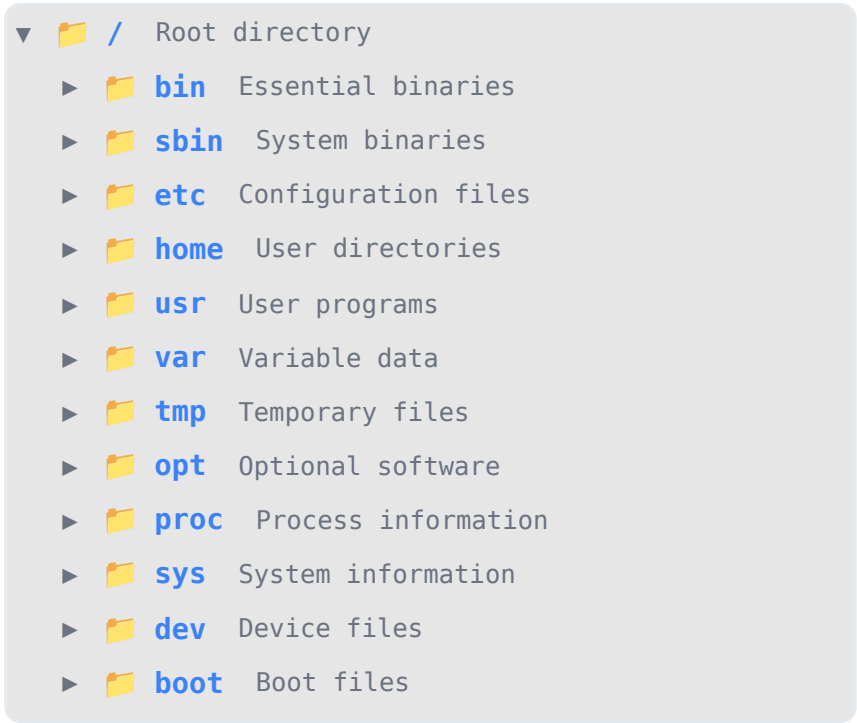
What can I do with this information?

Linux Directory Structure

When analyzing a linux platform, it's useful to have a general understanding of the directory structure.

Some things to look for.

- Where are user files (documents, downloads, etc)?
- Where are the executables?
- Where are libraries like libc?
- Where's the list of users/groups?
- Where are password hashes?
- Where might I find other key material?



Application Versions

What are some ways we might get an application's version?

For a Binary File

```
$ strings launchctl | grep -i "version "
Darwin Bootstrapper Control Interface Version 7.0.0: Fri Jul 11 20:01:57 PDT 2025; root:libxpc_executables-2894.140.12~26/l
[%s]: entitlements blob has unexpected version %lld
```

```
$ strings bash | grep -i version
GNU bash, version %s-%s
display-shell-version
    version, type `enable -n test'.  On systems supporting dynamic
BASH_VERSION
Version information for this Bash.
The type of CPU this version of Bash is running under.
```

Configuration Files

- Sometimes have useful information
- No rules here

Startup Scripts

To get a comprehensive understanding of a system's services, it's useful to start with the system boot scripts.

systemd

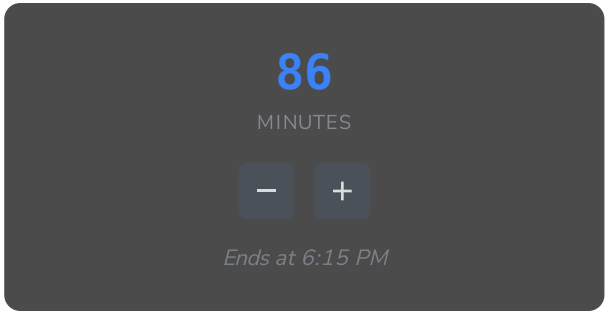
```
# /etc/systemd/system/myapp.service
[Unit]
Description=MyApp service
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
```

initd

```
#!/bin/sh
# /etc/init.d/myapp
### BEGIN INIT INFO
# Provides:          myapp
# Required-Start:    $remote_fs $syslog $network
# Required-Stop:     $remote_fs $syslog $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
```

Lab 1

Firmware startup analysis.



What to look for

You may want to further analyze an application you discover. In the rest of this course we'll mostly be analyzing compiled applications, but sometimes you'll have source available if it's written in a scripting language or is open source.

- Languages
- Frameworks
- Dependencies
- Versions
  - Public vulnerabilities

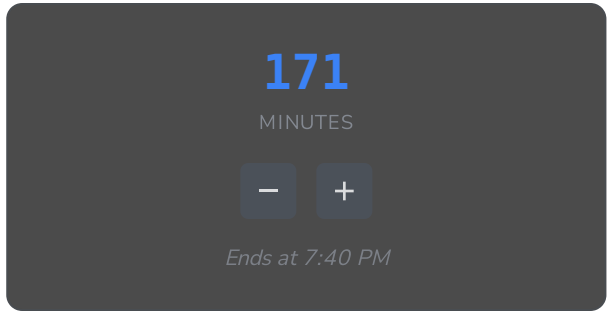
Analysis Strategies

There's a variety of strategies for analyzing source code.

- Reading through the code: open it in `vscode`
- Running the application
  - `python3 -m http.server`
  - `npm run dev`
- Code analysis tools
  - Large language model
  - Query langauges like `CodeQL`

Lab 2

https://hacs408e.umd.edu/schedule/week-02/lab-2/



Homework

First homework is due next week.

I will be in Nevada wilderness from Thursday through next Monday.