



Operating Systems Meetup

Policy Against Harassment & CoC

<https://www.acm.org/about-acm/policy-against-harassment>

OS Meetup wants to encourage and preserve this open exchange of ideas, which requires an environment that enables all to participate without fear of personal harassment. We define harassment to include specific unacceptable factors and behaviors listed in the ACM's policy against harassment. Unacceptable behavior will not be tolerated.

Code of Conduct

Systems Gossip Meetup

Introduction

- Diversity and inclusion make our community strong. We encourage participation from the most varied and diverse backgrounds possible and want to be very clear about where we stand.
- Our goal is to maintain a safe, helpful and friendly community for everyone, regardless of experience, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, religion, nationality, or other defining characteristic.
- This code and related procedures also apply to unacceptable behavior occurring outside the scope of community activities, in all community venues— online and in-person— as well as in all one-on-one communications, and anywhere such behavior has the potential to adversely affect the safety and well-being of community members.

Expected Behavior

- Be welcoming.
- Be kind.
- Look out for each other.

Unacceptable Behavior

- Conduct or speech which might be considered sexist, racist, homophobic, transphobic, ableist or otherwise discriminatory or offensive in nature.
- Do not use unwelcome, suggestive, derogatory or inappropriate nicknames or terms.
- Do not show disrespect towards others. (Jokes, innuendo, dismissive attitudes.)
- Intimidation or harassment (online or in-person). Please read the [Citizen Code of Conduct](#) for how we interpret harassment.
- Disrespect towards differences of opinion.
- Inappropriate attention or contact. Be aware of how your actions affect others. If it makes someone uncomfortable, stop.
- Not understanding the differences between constructive criticism and disparagement.
- Sustained disruptions.
- Violence, threats of violence or violent language.
- Sharing messages outside of this slack **without** permission of the poster.

<https://docs.google.com/document/d/1EBDdpXlev-ChxPrSueFWAMKmj51jbd5ESX8V5qYuWI8/edit?usp2sharing>

The Format

- One hour meetup per week on Saturday 5:00 p.m. PST
- Volunteers signing up for topics to present
- 1 lecture per week. Customized schedule
- Each talk will be recorded, and uploaded to a private YouTube playlist
- Guest speakers
- We have a Slack group

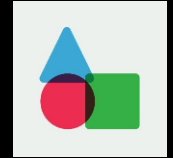
Organizational Change

- New organizer team: Mossaka and Wen Lin
- New org name: Systems Gossip¹ Meetup
- We will still maintain all previous meetups artifacts, including GitHub Repos, Slack group, and websites, etc.
- All repos are licensed under Apache 2.0

1. <https://www.notion.so/Meetup-2-Amazon-Dynamo-2a1ece24a27e433a99d32dfce6d2a28c>

About Me (mossaka, @jiaxiao_zhou)

I'm a software engineer at Deis Labs. Microsoft Azure



I work on open-source WebAssembly and Rust projects



Graduated from University of California, San Diego



Rust is my favorite programming language, and I ❤️ Haskell

I co-organized systems meetup, summer paper reading
meetup and other public events

About Me (Wen Lin)



Distributed Systems Meetup

Database Systems Meetup

Operating Systems Meetup

Virtualized Systems Meetup

Computer Networking Meetup

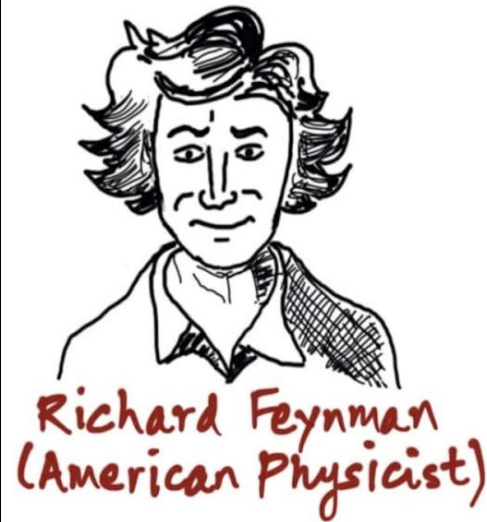
Why join us?

The Philosophy:

1. Reading
2. Listening
3. Demonstrating
4. Teaching

This meetup should serve as an opportunity for you to teach us something challenging; as a place to discuss or debate about something in detail; as a motivation to learn by doing. Everything served in this meetup is open to everyone

THE FEYNMAN TECHNIQUE



STEP 1 - Pick and study
a topic



STEP 2 - Explain the topic to
someone, like a child,
who is unfamiliar with
the topic... and at their
level of understanding.
Use simple language.



STEP 3 - Identify any gaps in
your understanding



STEP 4 - Return to the literature
to understand better



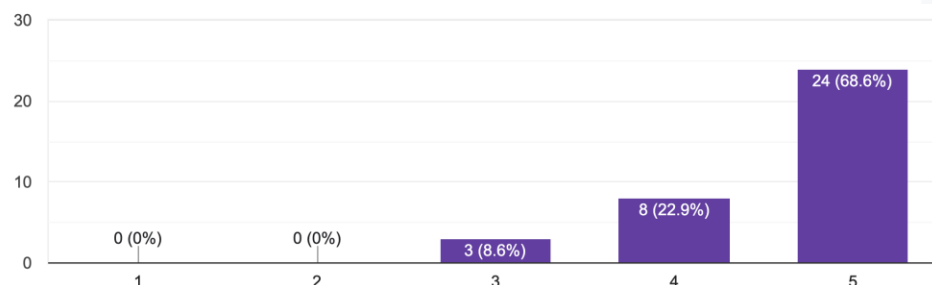
The goal of the system meetup

- **Know the concepts**
- **Know the algorithms**
- **Know the system design trade-offs**
-
- **This is both for system architect and system applicants.**
-

Feedback from previous participants

SPLVM database meetup对我学习数据库系统有帮助

35 responses



建议暂时没有想到，但是还是想表达一下感谢。多谢我们的Organizer和Presenter，作为一个普普通通的搬砖党我深知组织这样的学习小组需要私下花费大量的精力。所以很感谢各位的无私分享精神！！

We got a very supportive, inclusive and smart community!

特别有组织，而且大家互相尊重，积极讨论，受益匪浅，真的非常感谢host！

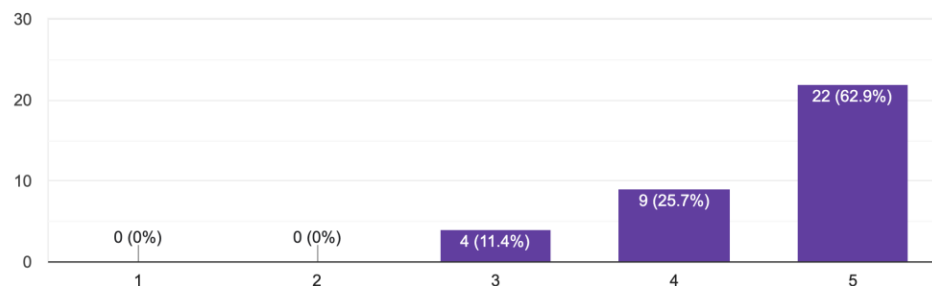
看到有这么多陌生人一起学一个东西，还是挺好的，希望自己也继续加油









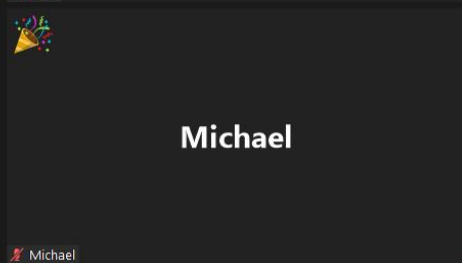
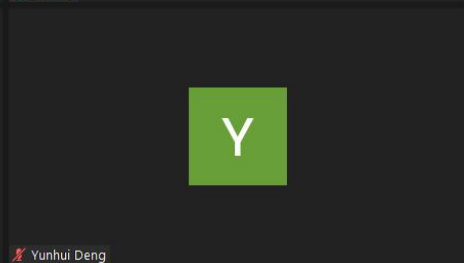
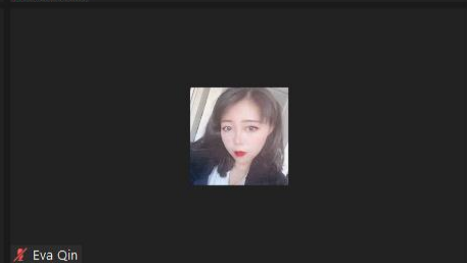
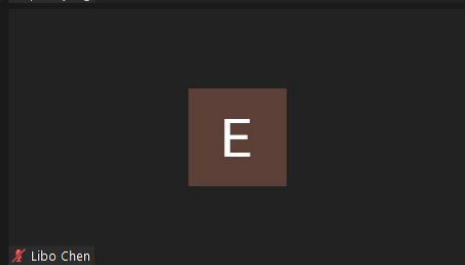
感谢组织者和讲者贡献自己的宝贵时间和大家分享！

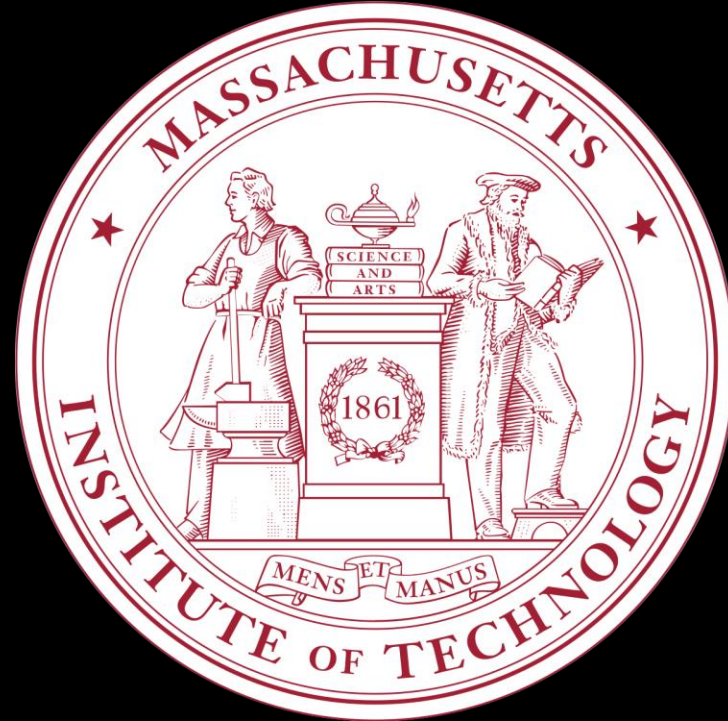
非常幸运能加入组织，希望能一直持续下去！以后还能线下活动。

SPLVM database meetup的presenter解释的很清楚

35 responses



 <p>Verity Chu</p>	 <p>Mossaka</p>	 <p>Wen Lin</p>	 <p>Chengshuo Xu</p>	 <p>Richard Tu</p>
 <p>r12f</p>	 <p>Xun Peng</p>	 <p>jwt</p>	 <p>Wilbur Liu</p>	 <p>Xuefeng Zhu</p>
<p>xtray</p> <p>xtray</p>	 <p>r12f</p>	<p>Gordon</p> <p>Gordon</p>	<p>Ping Lu</p> <p>Ping Lu</p>	<p>Yao Xiao</p> <p>Yao Xiao</p>
 <p>Michael</p> <p>Michael</p>	<p>Chi</p> <p>Chi</p>	<p>jasonyang</p> <p>jasonyang</p>	 <p>Yunhui Deng</p>	 <p>Eva Qin</p>
		 <p>Libo Chen</p>		



MIT 6.S081

Schedule

- [6.S081 / Fall 2021 \(mit.edu\)](#)
- [https://splvm.github.io/os-meetup/](#)

Honor Code

- <https://pdos.csail.mit.edu/6.828/2021/general.html>

"Do not post your lab or homework solutions on publicly accessible web sites (such as GitHub) or file spaces (such as your Athena Public directory)."

All links you will need

- [Course Website](#)
- [Meetup Website](#)
- [YouTube Link](#)
- [Zoom Link](#)
- [Textbook](#)
- Github Education

Let's Get Started!

What is even an operating system?

Abstraction + Multiplexing

- Abstract away hardware details (virtual hardware?)
- Support a wide range of applications (still a need?)
- Allow multiple apps to share the same hardware (multi-core)
- Isolation (rings)
- Sharing to allow cooperation (IPC)
- High performance

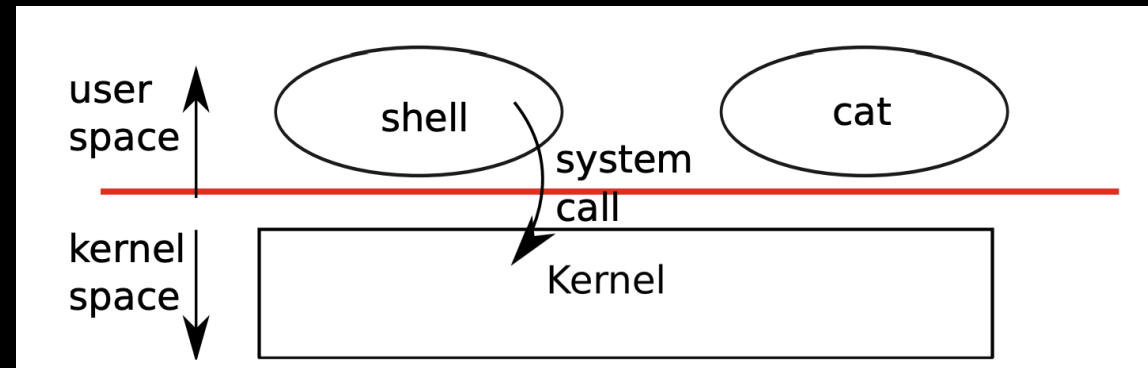
What is even an operating system?

- Operating system \neq kernel
- Application benefits from OS because they become portable.
- Except if you want to deploy your app to multiple OSes
- Linux, MacOS, Windows
- Different OS has different backward compatibility story
 - Your app that runs on Win XP can probably run on Win 10

Kernel

Kernel is a service provider that happens to have the highest privilege

- when a program/process requests a service, it must invoke a syscall and let the kernel process it and return
- Syscalls interface contains a collection of services a kernel provides



Application / syscall interface

- It's a misconception to say that C gives direct access to filesystem.
- Accessing filesystem on disk is a privilege, so only kernel can do it
- How does C program communicate with kernel services?
- C provides a standard library for accessing the filesystem. That library, when compiled, uses the actual OS syscall to access the resources.
- glibc vs. musl libc

glibc

By far the most widely used C library on Linux is the GNU C Library (<http://www.gnu.org/software/libc/>), often referred to as *glibc*. This is the C library that is nowadays used in all major Linux distributions. It is also the C library whose details are documented in the relevant pages of the *man-pages* project (primarily in Section 3 of the manual). Documentation of glibc is also available in the glibc manual, available via the command *info libc*. Release 1.0 of glibc was made in September 1992. (There were earlier 0.x releases.) The next major release of glibc was 2.0, at the beginning of 1997.

The pathname */lib/libc.so.6* (or something similar) is normally a symbolic link that points to the location of the glibc library, and executing this pathname will cause glibc to display various information about the version installed on your system.

Linux libc

In the early to mid 1990s, there was for a while *Linux libc*, a fork of glibc 1.x created by Linux developers who felt that glibc development at the time was not sufficing for the needs of Linux. Often, this library was referred to (ambiguously) as just "libc". Linux libc released major versions 2, 3, 4, and 5, as well as many minor versions of those releases. Linux libc4 was the last version to use the a.out binary format, and the first version to provide (primitive) shared library support. Linux libc 5 was the first version to support the ELF binary format; this version used the shared library soname *libc.so.5*. For a while, Linux libc was the standard C library in many Linux distributions.

However, notwithstanding the original motivations of the Linux libc effort, by the time glibc 2.0 was released (in 1997), it was clearly superior to Linux libc, and all major Linux distributions that had been using Linux libc soon switched back to glibc. To avoid any confusion with Linux libc versions, glibc 2.0 and later used the shared library soname *libc.so.6*.

Since the switch from Linux libc to glibc 2.0 occurred long ago, *man-pages* no longer takes care to document Linux libc details. Nevertheless, the history is visible in vestiges of information about Linux libc that remain in a few manual pages, in particular, references to *libc4* and *libc5*.

Other C libraries

There are various other less widely used C libraries for Linux. These libraries are generally smaller than glibc, both in terms of features and memory footprint, and often intended for building small binaries, perhaps targeted at development for embedded Linux systems. Among such libraries are *uClibc* (<http://www.uclibc.org/>), *dietlibc* (<http://www.fefe.de/dietlibc/>), and *musl libc* (<http://www.musl-libc.org/>). Details of these libraries are covered by the *man-pages* project, where they are known.

xv6

- A simplified teaching UNIX OS
- Runs on Risc-V processor
- Labs run on QEMU machine emulation
- Lab 1: Xv6 and Unix utilities
- <https://pdos.csail.mit.edu/6.S081/2021/labs/util.html>

File descriptor

- a file descriptor is a handle (opaque type, unique identifier) for I/O resources, such as file, pipe/sockets, or devices
- 0 – stdin
- 1 – stdout
- 2 – stderr

File descriptor in xv6

- Xv6 kernel uses the file descriptor as an index into a per-process table, so that every process has a private space of file descriptors starting at zero.
- `read (fd, buf, n)`
- `write (fd, buf, n)`
- `close (fd)`

```
char buf[512];
int n;
for(;;){
    n = read(0, buf, sizeof buf);
    if(n == 0)
        break;
    if(n < 0){
        fprintf(2, "read error\n");
        exit();
    }
    if(write(1, buf, n) != n){
        fprintf(2, "write error\n");
        exit();
    }
}
```

THE **LINUX** PROGRAMMING INTERFACE

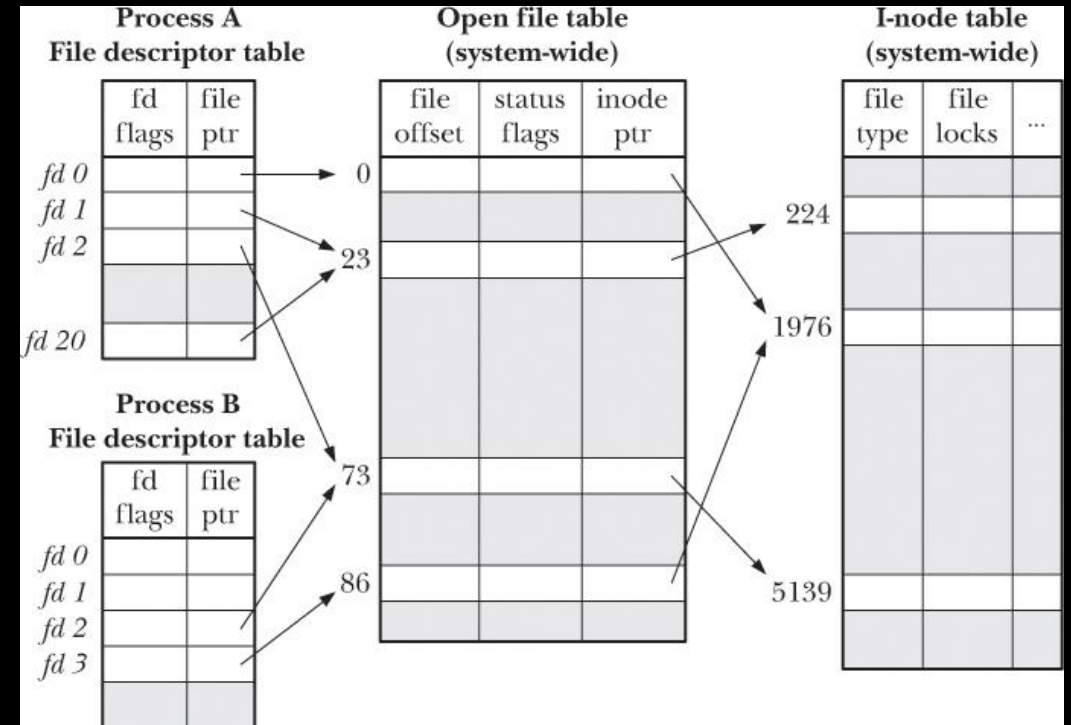
A Linux and UNIX® System Programming Handbook

MICHAEL KERRISK



File descriptor in Linux

- Linux kernel maintains 3 data structures (Kerrisk p90)
 - The per-process file descriptor table
 - The system-wide table of open file descriptions
 - The file system I-node table



And many more!

- In lecture, Robert shows many more examples of syscalls
- Shell
- Pipes
- Fork
- Exec
- Redirect

Next time

- Lecture 2: OS organization and system calls
- Read chapter 2
- Lab util due
- Lab syscall starts
- Don't forget to signup (pull request) presenters!