Linux Kernel Introduction

•••

WHAT TO EXPECT

HISTORY

LINUX KERNEL vs LINUX OS

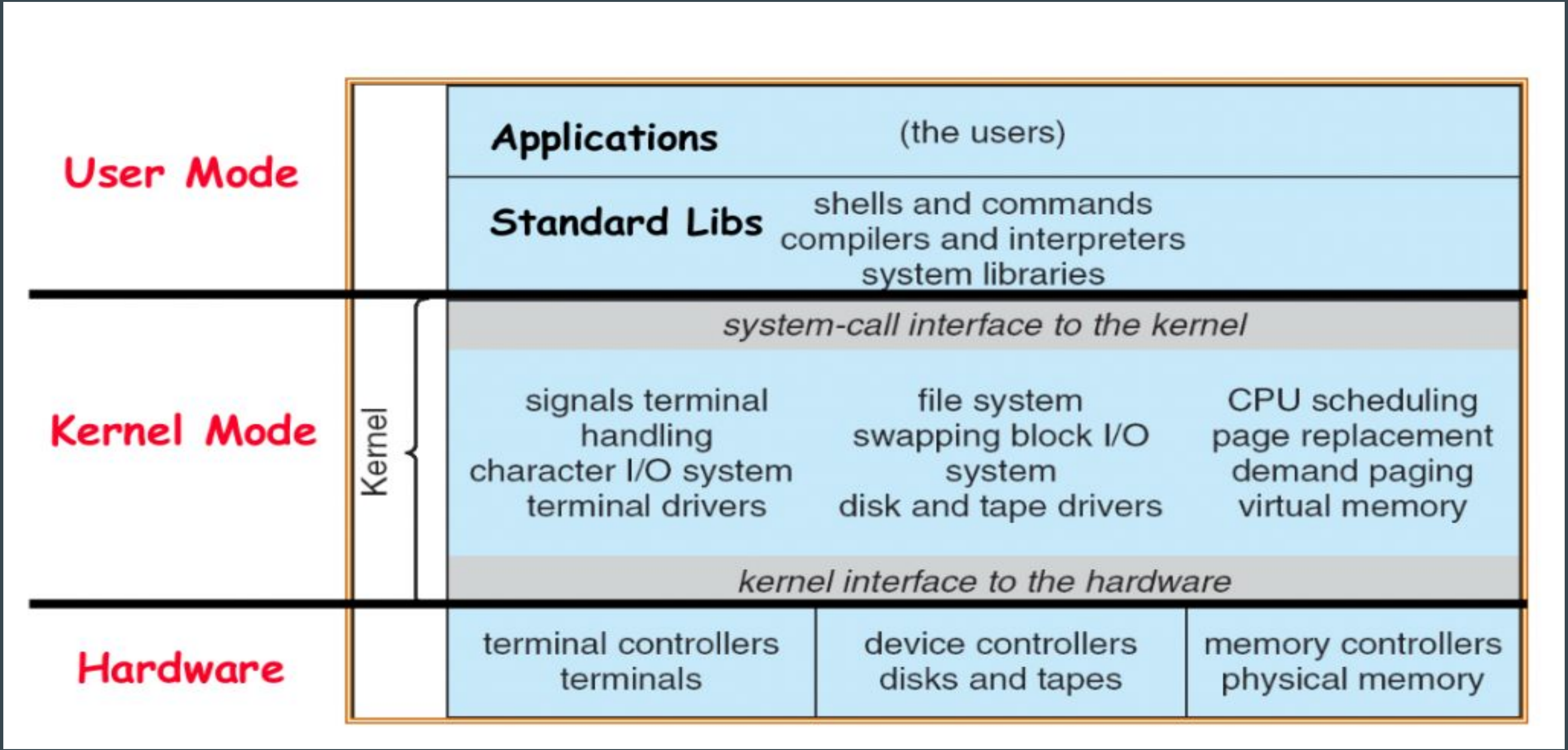


Image taken from: https://www.linuxfordevices.com/tutorials/linux/linux-kernel

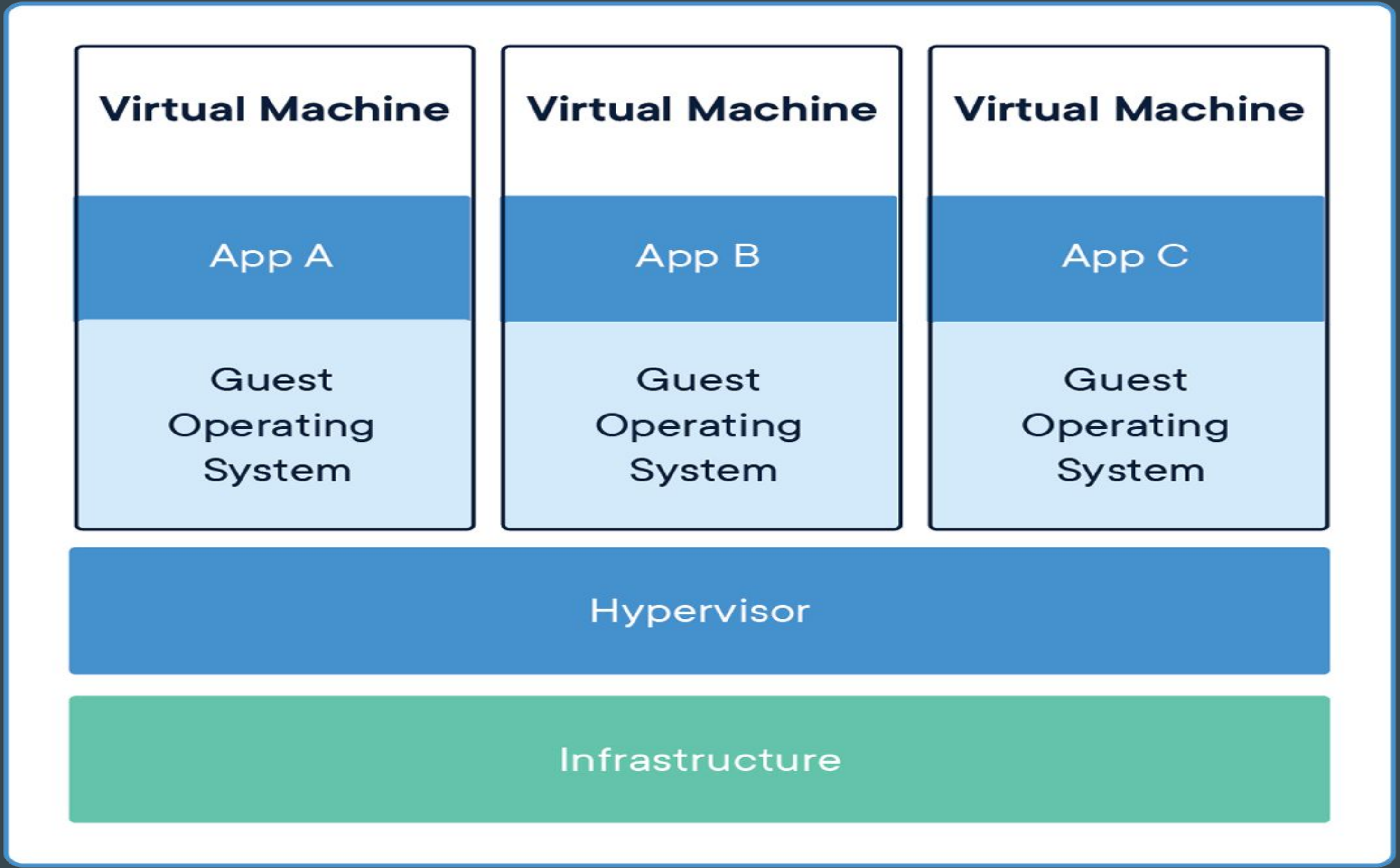
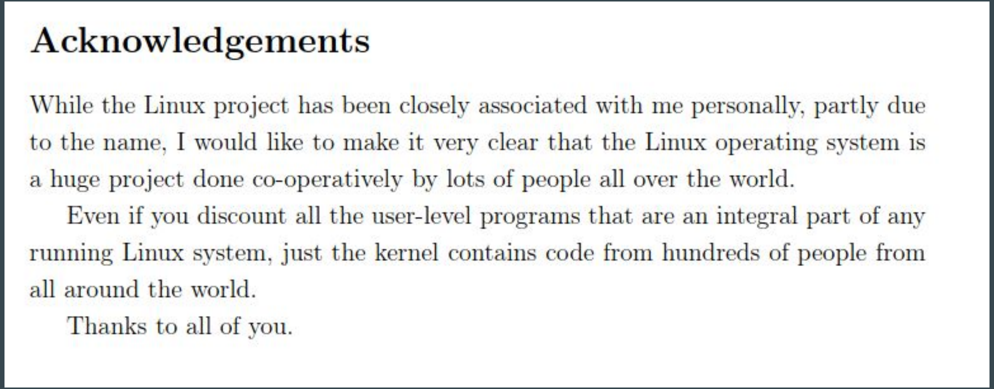


Image taken from: https://commons.wikimedia.org/wiki/File:Container-vm-whatcontainer_2.png

LINUX KERNEL IS

THE open source project



Acknowledgements

While the Linux project has been closely associated with me personally, partly due to the name, I would like to make it very clear that the Linux operating system is a huge project done co-operatively by lots of people all over the world.

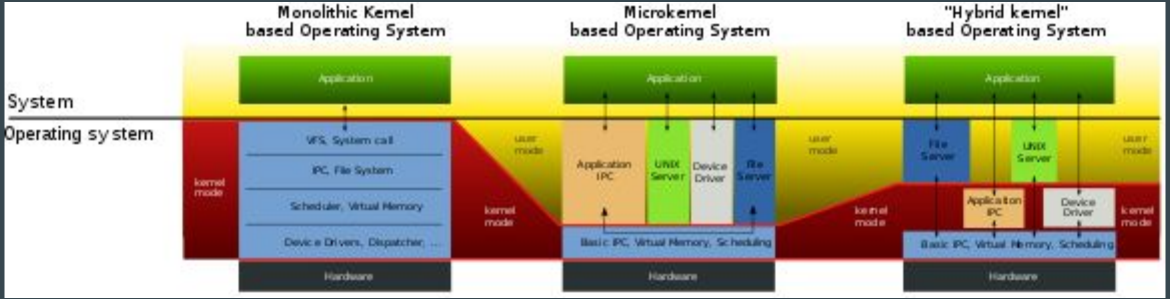
Even if you discount all the user-level programs that are an integral part of any running Linux system, just the kernel contains code from hundreds of people from all around the world.

Thanks to all of you.

https://www.cs.helsinki.fi/u/kutvonen/index_files/linus.pdf

UNIX (kinda)

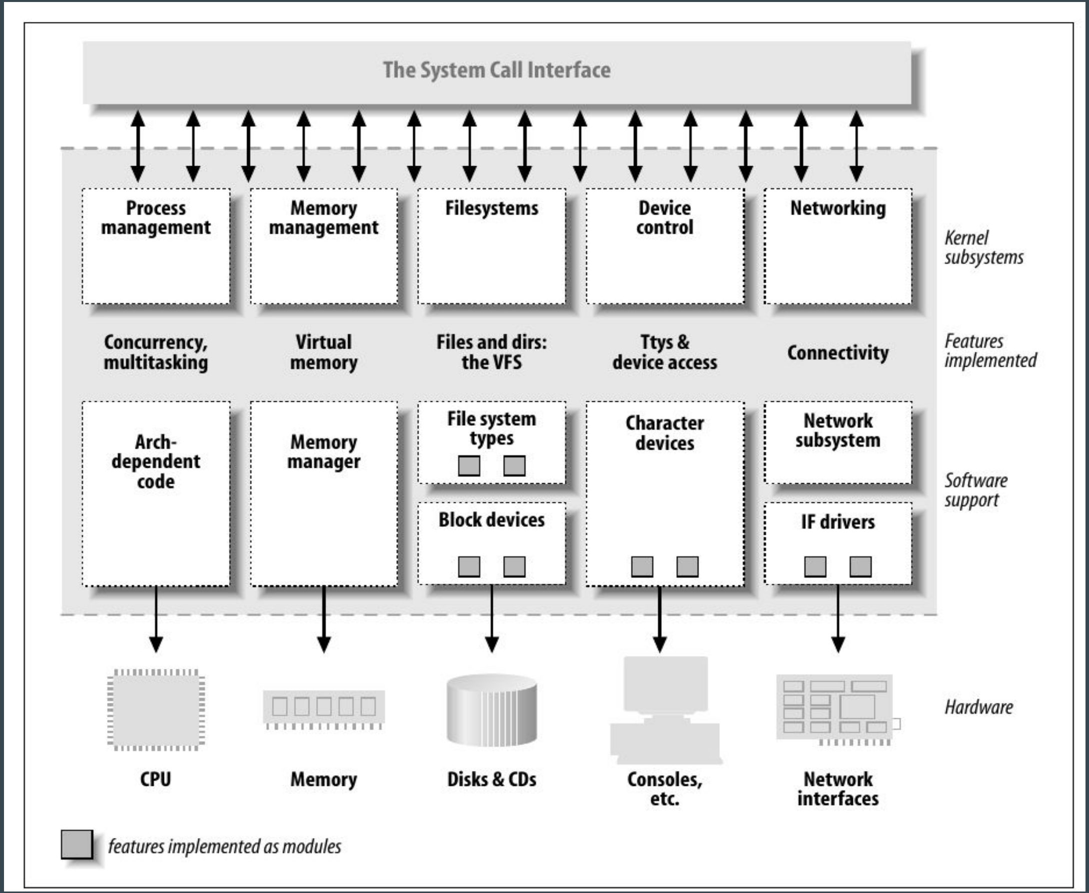
Monolithic...



Monolithic Kernel based Operating System
Microkernel based Operating System
"Hybrid kernel" based Operating System
System
Operating system
Application
VFS, System call
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
kernel mode
user mode
Application IPC
UNIX Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling
Hardware

Taken from:
https://upload.wikimedia.org/wikipedia/commons/thumb/d/d0/OS-structure2.svg/580px-OS-structure2.svg.png

… but with Modules



Taken from: https://lwn.net/Kernel/LDD3/

Symmetric Multiprocessors

Preemptive

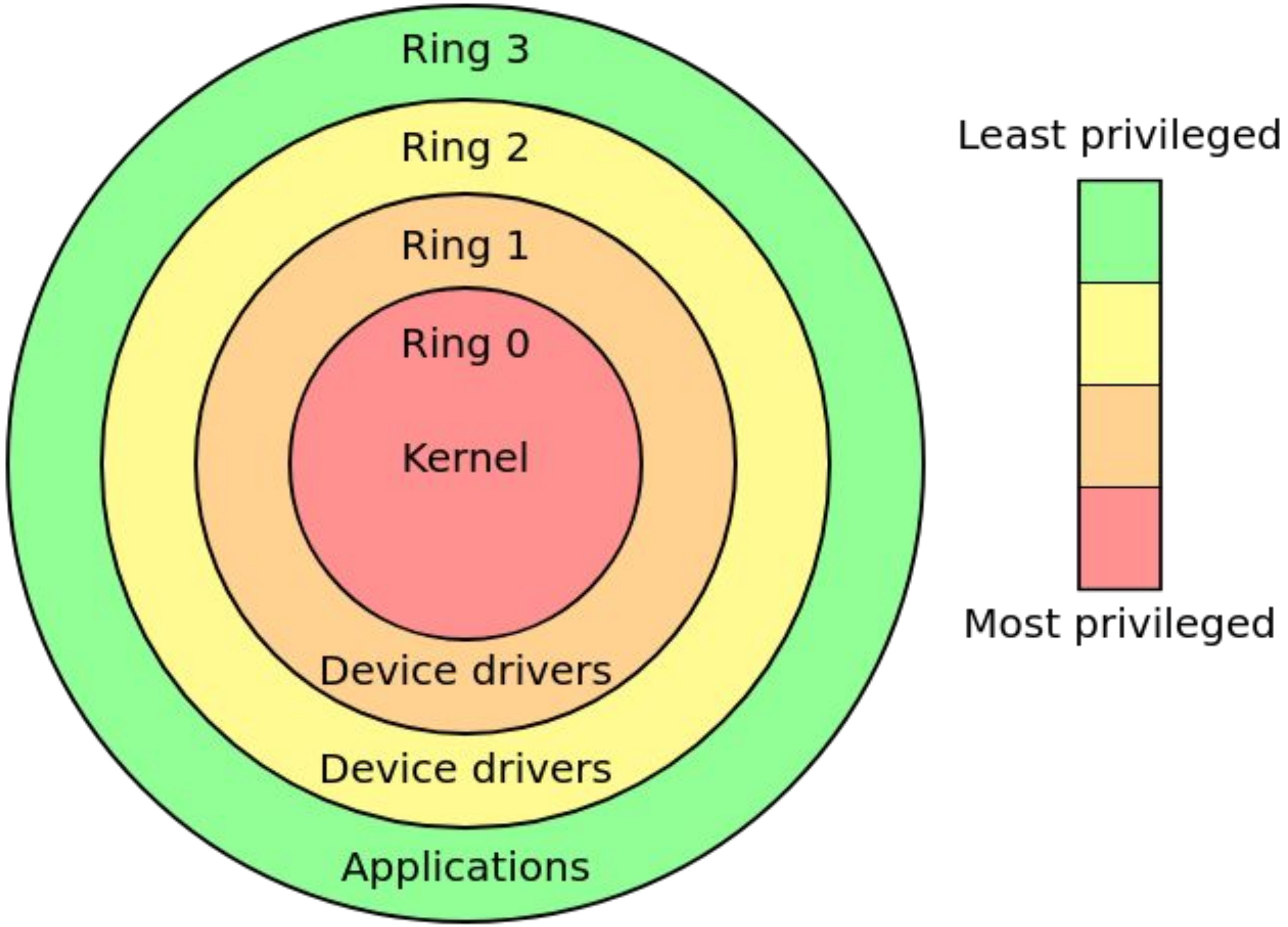
Reentrant

HOW LINUX WORKS

Process vs Thread

~~Process vs Thread~~
Task

(Lord of the) Rings


Ring 3
Ring 2
Ring 1
Ring 0
Kernel
Device drivers
Device drivers
Applications
Least privileged
Most privileged

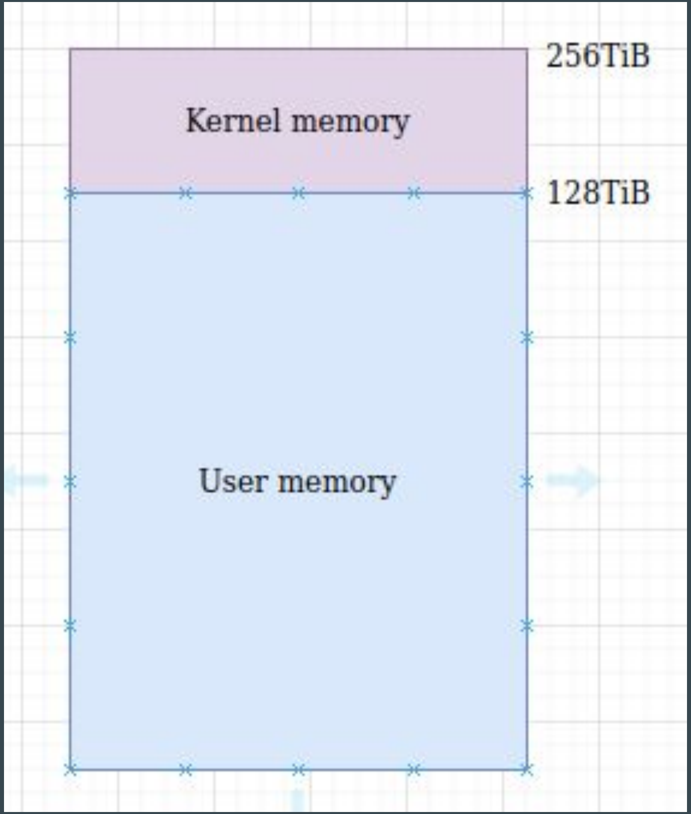
Traps

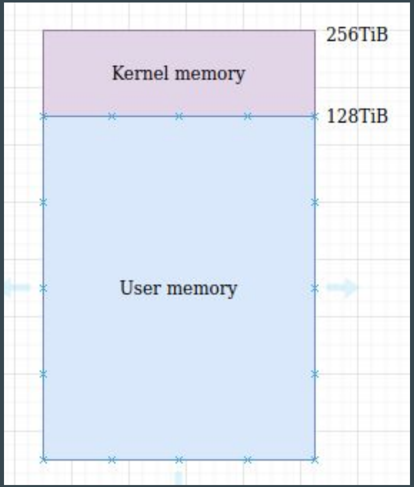
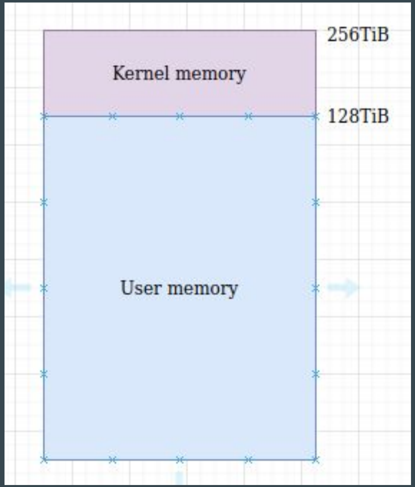
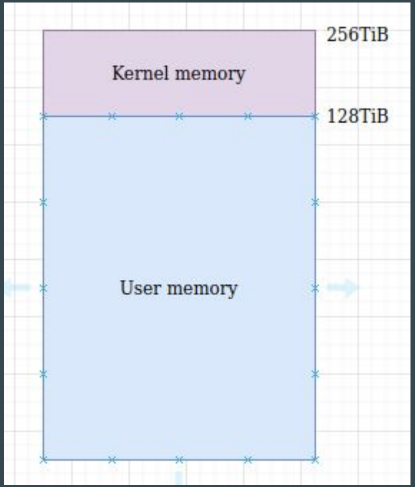
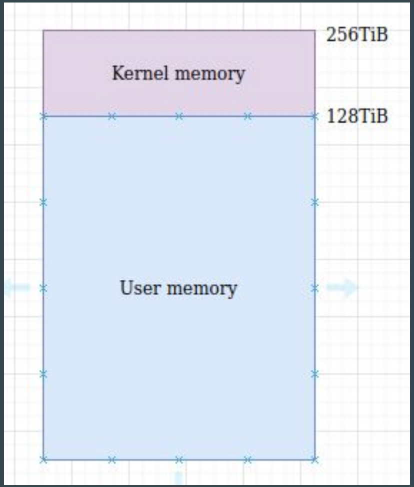
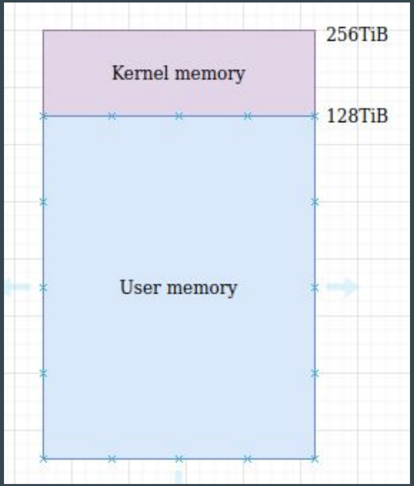
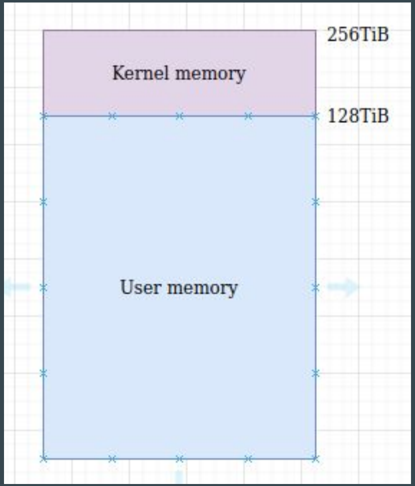
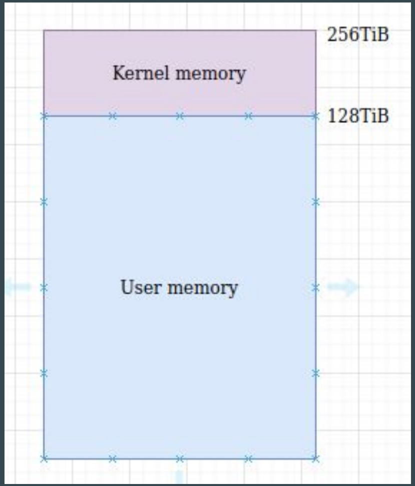
Cutest traps in anime

Scheduling...

… Or how you can write on the terminal without freezing your entire machine.

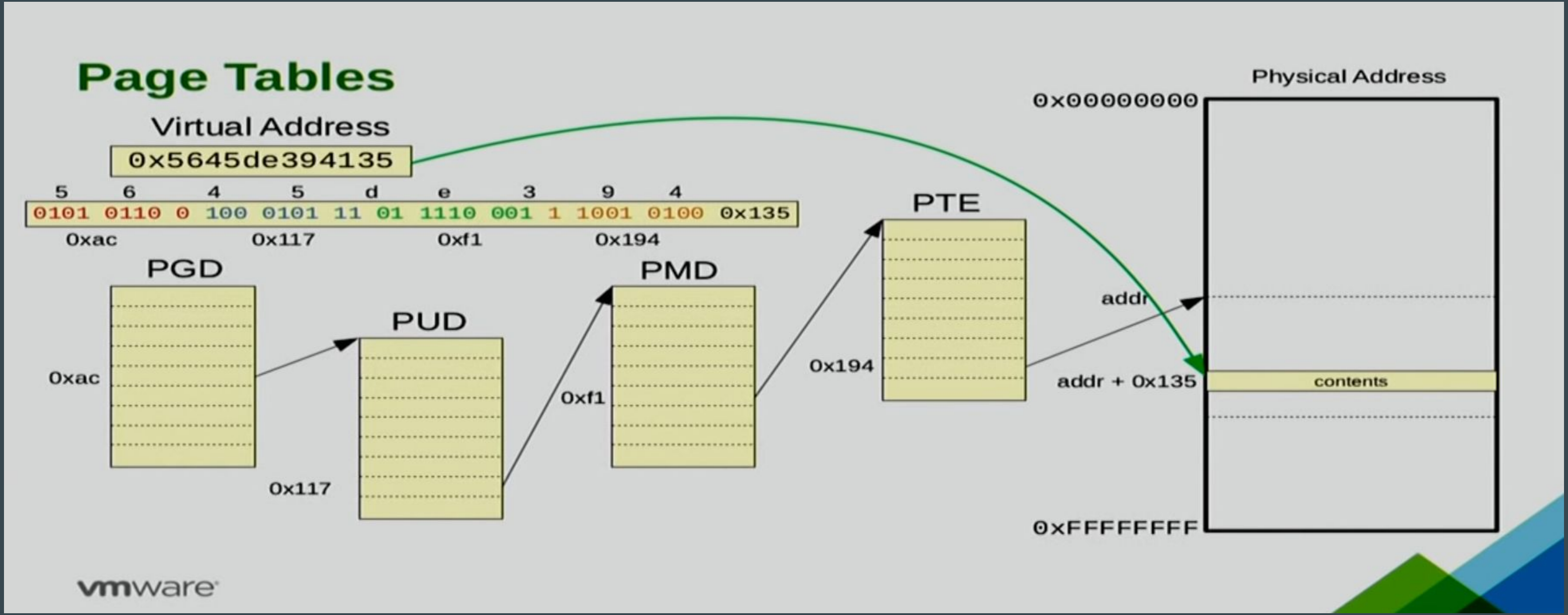
Userland vs Kernelland


256TiB
Kernel memory
128TiB
User memory


256TiB
Kernel memory
128TiB
User memory

256TiB
Kernel memory
128TiB
User memory

256TiB
Kernel memory
128TiB
User memory

256TiB
Kernel memory
128TiB
User memory

256TiB
Kernel memory
128TiB
User memory

256TiB
Kernel memory
128TiB
User memory

256TiB
Kernel memory
128TiB
User memory

?????

Everything in OS can be solved with one more layer of indirection



Steven Rostedt - Learning the Linux Kernel with tracing

https://www.youtube.com/watch?v=JRyrhsx-L5Y&t=3128s

HOW TO TALK TO THE KERNEL

Syscalls

HACKING THE KERNEL

No libc

No floats

(almost) No stack

No swapping

No userland memory

Lots of spinlocks

Oops & Panic

```
printk(KERN_ALERT "Hello, world\n");
```

kalloc()

copy_from_user()
copy_to_user()

Clone Torvalds tree

or

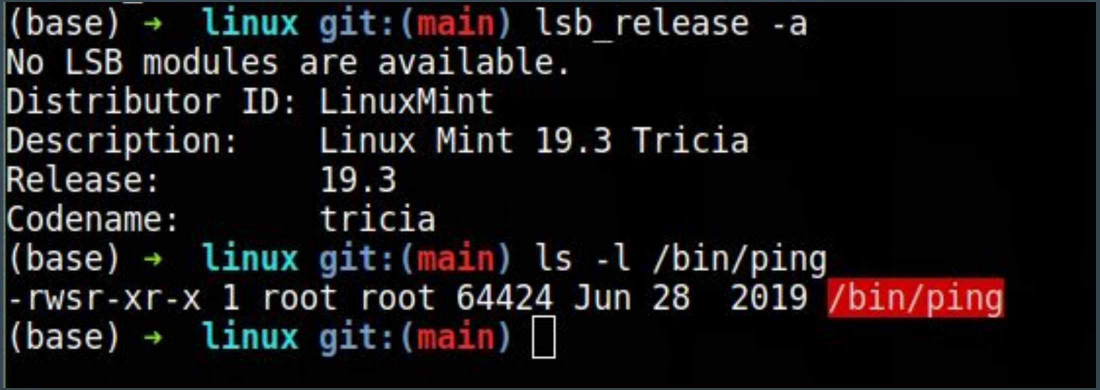
Download the sources from kernel.org

elixir.bootlin.com

COMPILING

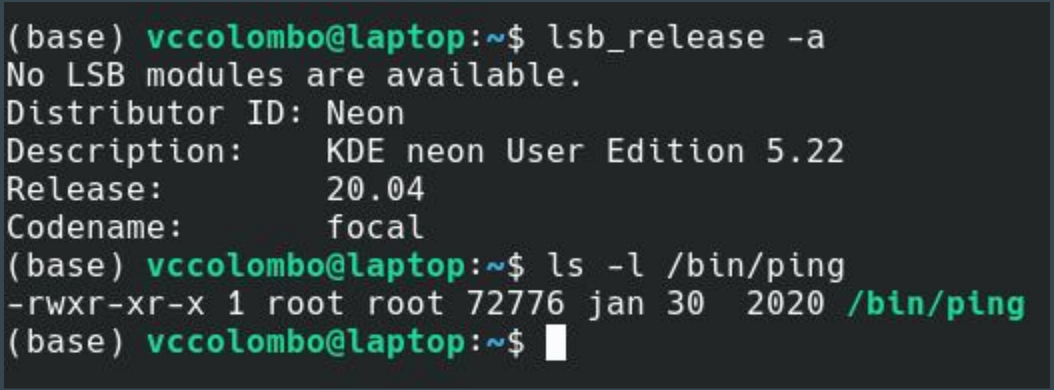
ANATOMY OF A BUG

Capabilities



```
(base) ➜  linux git:(main) lsb_release -a
No LSB modules are available.
Distributor ID: LinuxMint
Description:    Linux Mint 19.3 Tricia
Release:        19.3
Codename:       tricia
(base) ➜  linux git:(main) ls -l /bin/ping
-rwsr-xr-x 1 root root 64424 Jun 28  2019 /bin/ping
(base) ➜  linux git:(main) 
```

VS



```
(base) vccolombo@laptop:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Neon
Description:    KDE neon User Edition 5.22
Release:        20.04
Codename:       focal
(base) vccolombo@laptop:~$ ls -l /bin/ping
-rwxr-xr-x 1 root root 72776 jan 30  2020 /bin/ping
(base) vccolombo@laptop:~$ 
```

CVE-2016-9793

CVE-2016-9793

The sock_setsockopt function in net/core/sock.c in the Linux kernel before 4.8.14 mishandles negative values of sk_sndbuf and sk_rcvbuf, which allows local users to cause a denial of service (memory corruption and system crash) or possibly have unspecified other impact by leveraging the CAP_NET_ADMIN capability for a crafted setsockopt system call with the (1) SO_SNDBUFFORCE or (2) SO_RCVBUFFORCE option.

WHERE TO GO FROM HERE

Working with the Kernel

Contributing to the Kernel

Exploiting the Kernel

printk(KERN_NOTICE “Thanks for watching”);