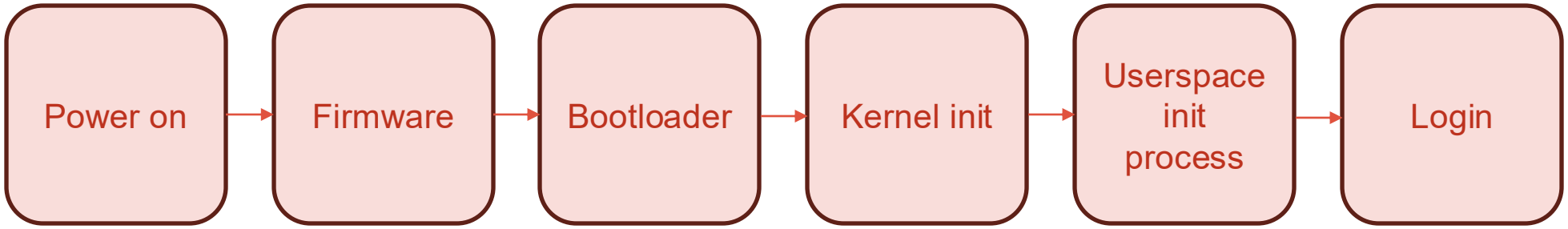
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

Introduction to the Linux Boot Process

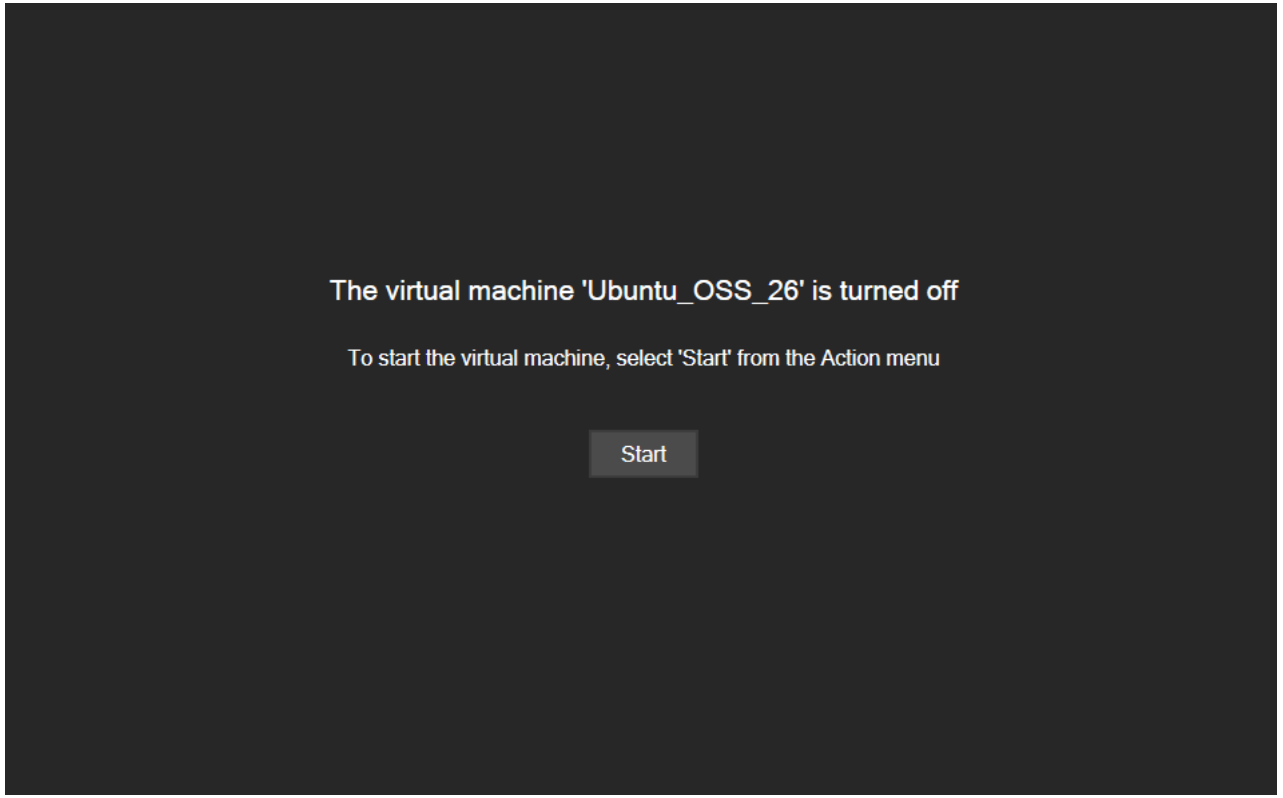
Karissa Sanchez & Angelina Vu

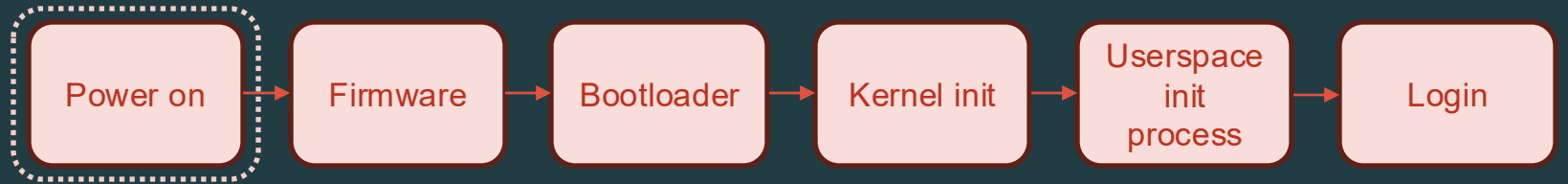
About this talk



Power on

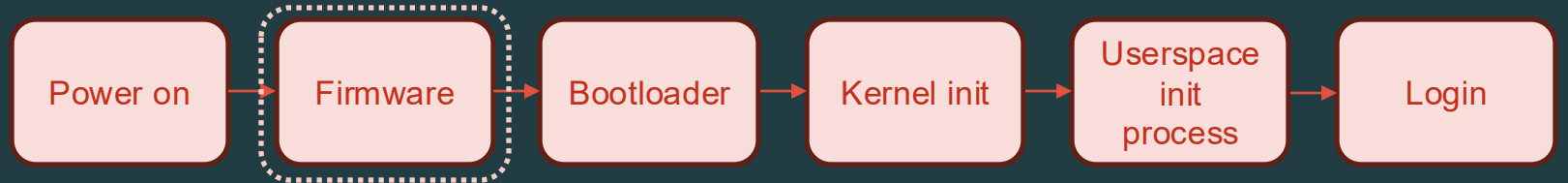
- CPU begins execution at reset vector
- Reads boot code from ROM
- Launches BIOS/UEFI





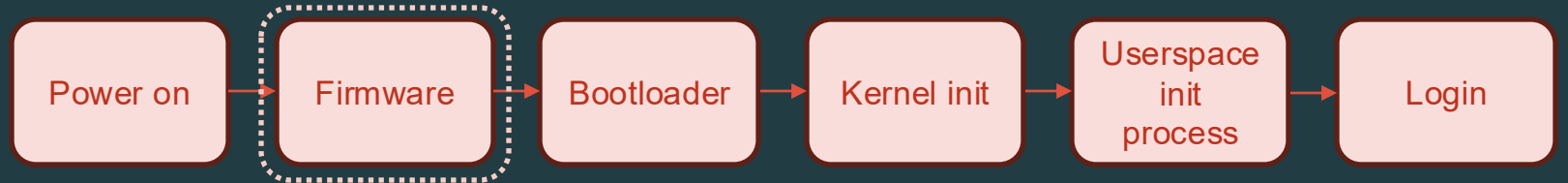
Firmware Phase Overview

- Early hardware init
- Detects and checks hardware
- Sets up boot environment
- Jumps to bootloader



POST (Power-On Self Test)

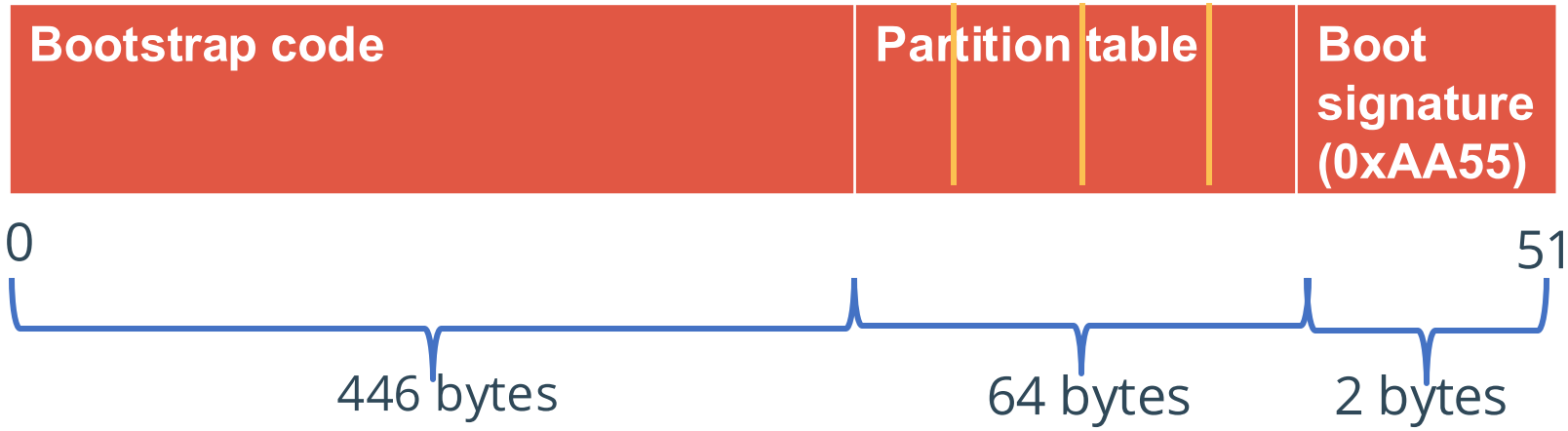
- Performed by BIOS/UEFI
- Early hardware initialization
- Checks functionality of essential hardware components
- System halts if fails or components are missing

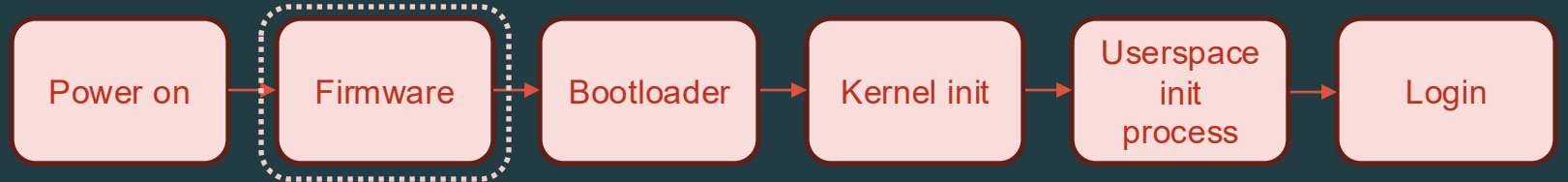


BIOS

- Basic Input/Output System
- Tied to MBR to access bootloader
- Less complex
- Very limited
  - ~2TB max disk size
  - 16-bit real mode
  - Only access 1MB mem
  - 4 primary partitions

Master Boot Record (MBR):

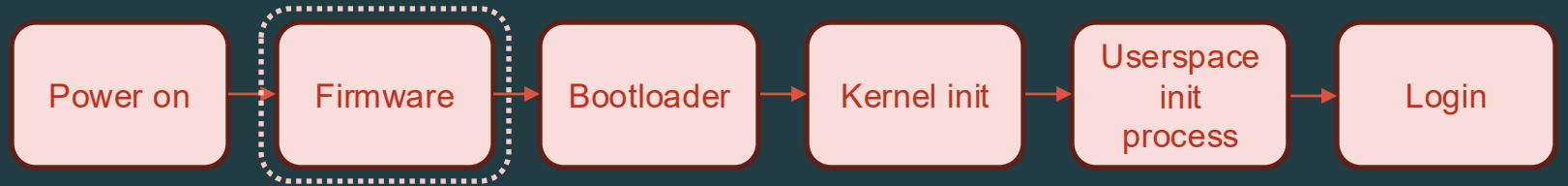




BIOS vs UEFI

- Basic Input/Output System
- Tied to MBR to access bootloader
- Less complex
- Very limited
  - ~2TB max disk size
  - 16-bit real mode
  - Only access 1MB mem
  - 4 primary partitions

- Unified Extensible Firmware Interface
- GPT (GUID Partition Table) supports larger disk size and more partitions
- Faster
  - Direct access to bootloader from ESP (EFI System Partition)
- More secure (Secure Boot, CRC)
- 32-bit or 64-bit mode
- Access to full RAM



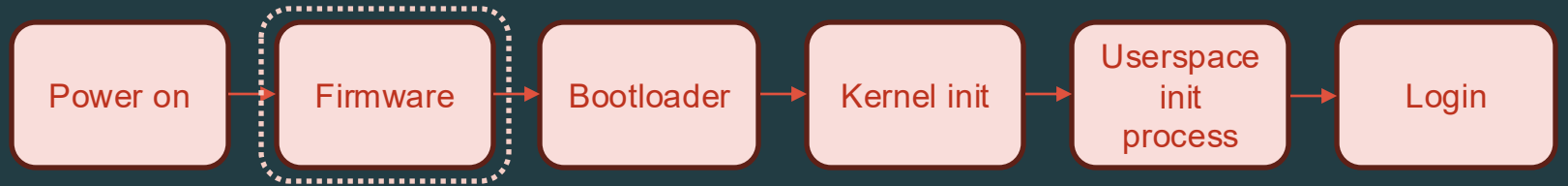
UEFI Internals

- NVRAM stores UEFI variables that define boot entry

```
avu@avu-Virtual-Machine:~$ sudo efibootmgr -v
BootCurrent: 0002
Timeout: 0 seconds
BootOrder: 0002,0000,0001
Boot0000* EFI SCSI Device       AcpiEx(VMBus,,)/VenHw(9b17e5a2-0891-42dd-b653-80b5c22809ba,d96361baa104294db60572e2f
fb1dc7f229c0c34df6c774884753232241cbe48)/SCSI(0,0)
Boot0001* EFI Network   AcpiEx(VMBus,,)/VenHw(9b17e5a2-0891-42dd-b653-80b5c22809ba,635161f83edfc546913ff2d2f965ed0e2
e8cf79c8b6aee45b377bdf28b0e8995)/MAC(000000000000,0)/IPv4(0.0.0.00.0.0.0,0,0)
Boot0002* ubuntu        HD(15,GPT,970642ca-fa8f-4fe2-b978-f6e9b1d2ed25,0x2800,0x35000)/File(\EFI\ubuntu\shimx64.efi)
Boot0003* FrontPage     MemoryMapped(11,0x100000,0x5dfffff)/FvFile(4042708a-0f2d-4823-ac60-0d77b3111889)
```

- GUID Partition Table (GPT)

Protective MBR	GPT Header	Partition Table	Partitions	Partition Table (backup)	GPT Header (backup)
			EFI System Partition (ESP) | ...		

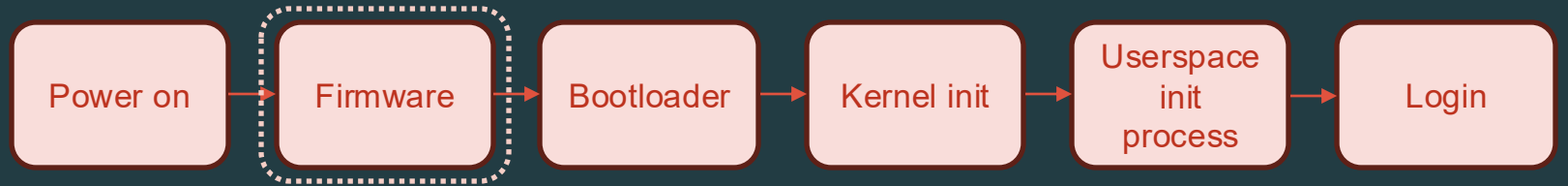


UEFI Internals

- EFI System Partition (ESP)
  - FAT32 format
  - Stores bootloaders, EFI apps, fw updates, diagnostic tools

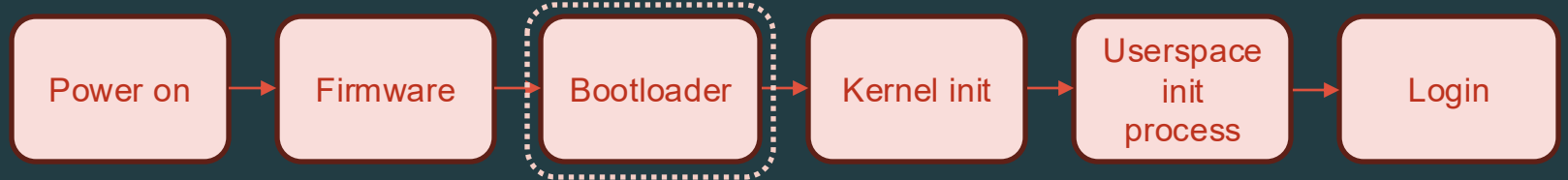
```
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda       8:0    0    12G  0 disk
├─sda1    8:1    0  11.9G  0 part /
├─sda14   8:14   0     4M  0 part
└─sda15   8:15   0   106M  0 part /boot/efi
```

```
/boot/efi/EFI
├── BOOT
│   ├── BOOTX64.EFI
│   ├── fbx64.efi
│   └── mmx64.efi
└── ubuntu
    ├── BOOTX64.CSV
    ├── grub.cfg
    ├── grubx64.efi
    ├── mmx64.efi
    └── shimx64.efi
```

Bootloader

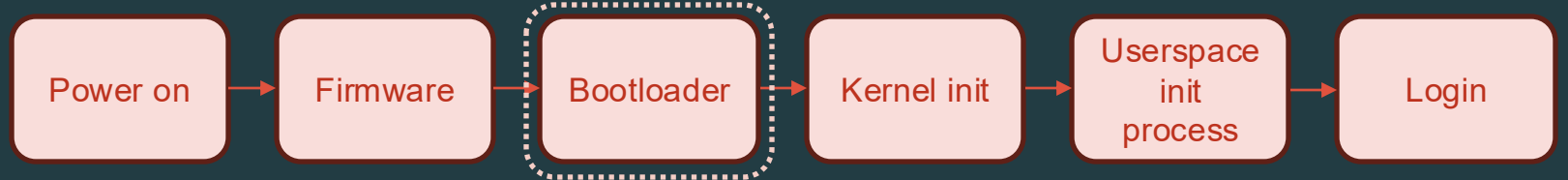
- The bootloader is responsible for **locating and loading the kernel and transferring execution to it**
- Common default bootloader is GNU GRUB2 (GRand Unified Bootloader)
- Supports choosing from multiple images to boot
- Not limited to Linux kernels
- Supports UEFI and BIOS with MBR or GPT partitioning
- Grub command line can be used to recover a failed boot



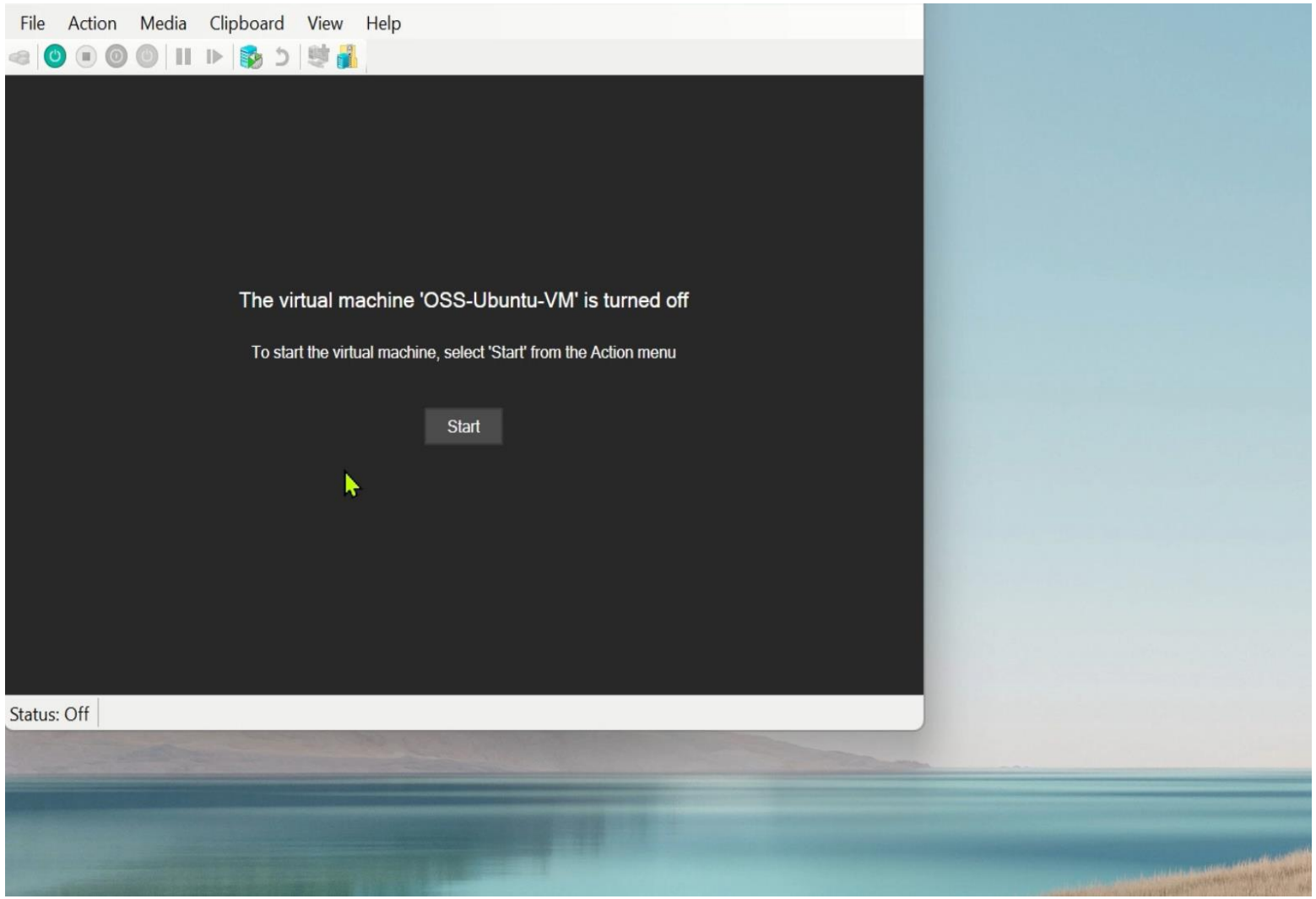
GRUB Settings

- Users can modify their kernel command line using GRUB
- There are many built-in kernel parameters, and custom ones can be added as well
- Users can also modify GRUB settings to specify timeout style and amount
- These changes are made by modifying /etc/default/grub and running update-grub (or grub2-mkconfig)
- /boot/grub/grub.cfg tells GRUB which settings to use, but users **cannot write** to this file

```
avu@avu-Virtual-Machine:~$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-27-generic root=LABEL=desktop-rootfs ro quiet splash vt.handoff=7
```

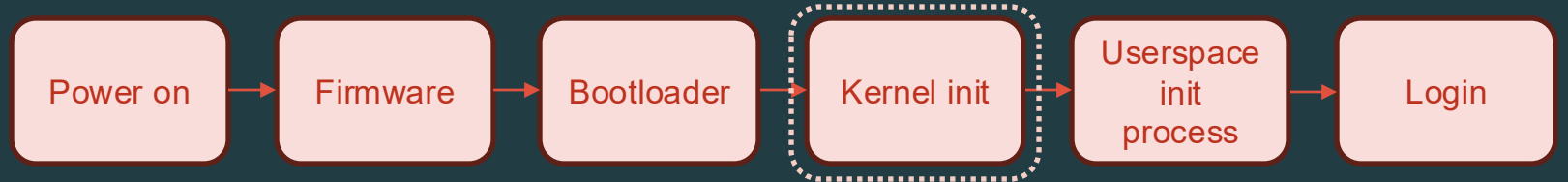
GRUB demo



Power on → Firmware → Bootloader → Kernel init → Userspace init process → Login

Kernel Init Overview

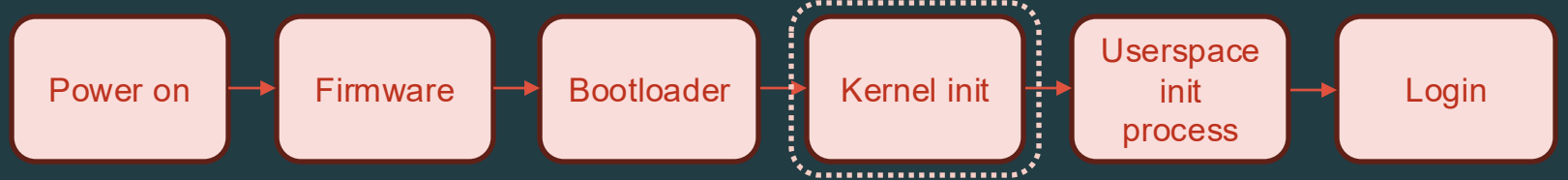
- Kernel decompresses itself
- Early CPU and memory setup
- Core kernel subsystems initialized
- Prepare and mount root fs



Early Kernel Setup

- Arch-specific entry code
  - E,g., arch/x86/boot/compressed/head_64.S, /arch/x86/kernel/head_64.S
- Pure assembly code
- Prepares to run C code

- Prepare stack
- Setup .bss
- Detect CPU features
- Init early memory management
- Call decompress_kernel()

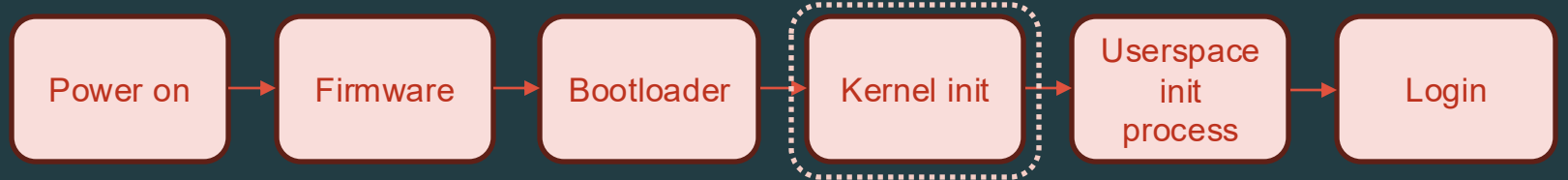


start_kernel()

- /init/main.c

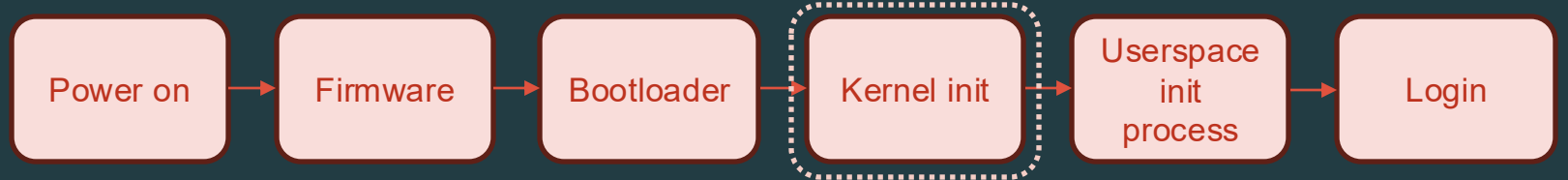
- First C kernel function
- Arch-specific inits
- Init core kernel subsystems
  - Memory management
  - Interrupt/IRQ handling
  - Scheduler and timers
  - Filesystem/VFS
  - Device and driver model

- Unpack initramfs into RAM (populate_rootfs)



End of start_kernel()

```
arch_call_rest_init()
  rest_init()
    pid = user_mode_thread(kernel_init, NULL, CLONE_FS);
```

kernel_init()

kernel_init_freeable()

do_basic_setup()

do_initcalls()

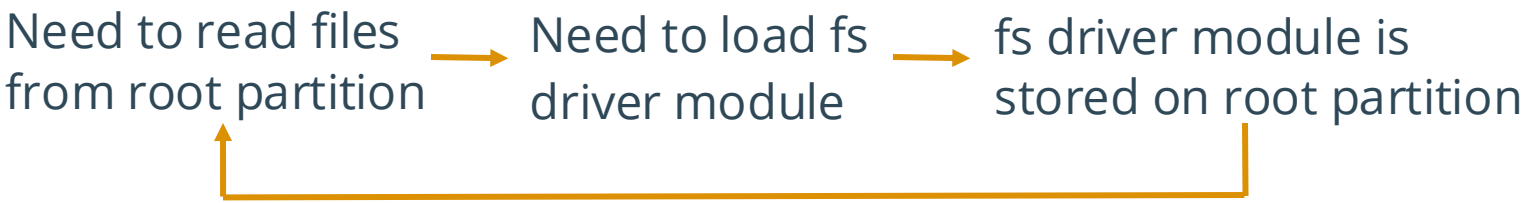
```
for (level = 0; level < ARRAY_SIZE(initcall_levels) - 1; level++) {
        /* Parser modifies command_line, restore it each time */
        strcpy(command_line, saved_command_line);
        do_initcall_level(level, command_line);
}
```

```
#define pure_initcall(fn)               __define_initcall(fn, 0)

#define core_initcall(fn)               __define_initcall(fn, 1)
#define core_initcall_sync(fn)          __define_initcall(fn, 1s)
#define postcore_initcall(fn)           __define_initcall(fn, 2)
#define postcore_initcall_sync(fn)      __define_initcall(fn, 2s)
#define arch_initcall(fn)               __define_initcall(fn, 3)
#define arch_initcall_sync(fn)          __define_initcall(fn, 3s)
#define subsys_initcall(fn)             __define_initcall(fn, 4)
#define subsys_initcall_sync(fn)        __define_initcall(fn, 4s)
#define fs_initcall(fn)                 __define_initcall(fn, 5)
#define fs_initcall_sync(fn)            __define_initcall(fn, 5s)
#define rootfs_initcall(fn)             __define_initcall(fn, rootfs)
#define device_initcall(fn)             __define_initcall(fn, 6)
#define device_initcall_sync(fn)        __define_initcall(fn, 6s)
#define late_initcall(fn)               __define_initcall(fn, 7)
#define late_initcall_sync(fn)          __define_initcall(fn, 7s)
```

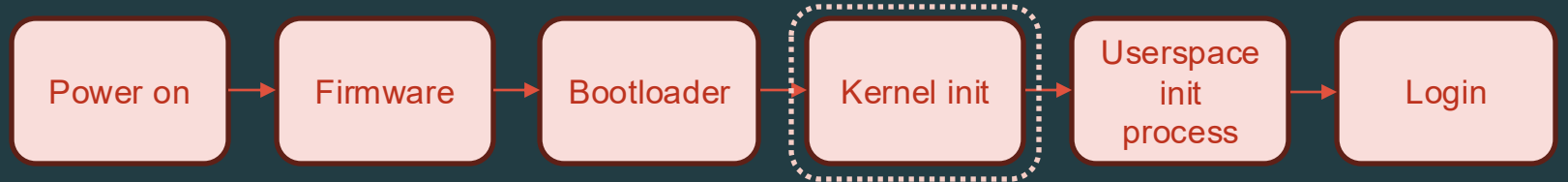
→ `rootfs_initcall(populate_rootfs);`

initramfs

- Why?



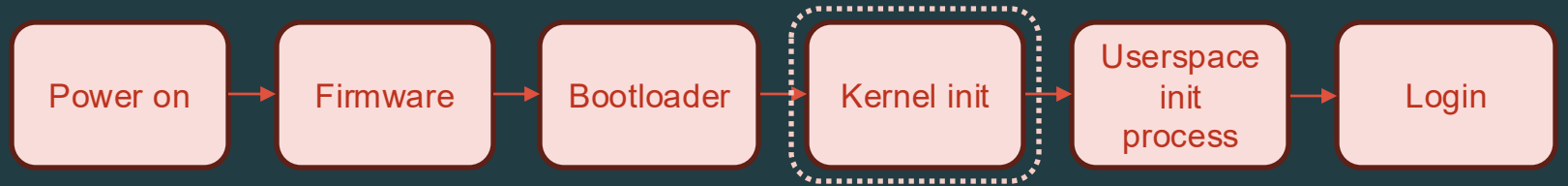
- Temporary root fs
- Launches early user space (/init)
  - Loads necessary drivers and modules
  - Mounts real root fs
- switch_root
  - Switch to real root
  - Starts real init (/sbin/init)



Initrd vs initramfs

- Initial RAM Disk
- Compressed fs image
  - Needs built-in fs driver
- Loaded into RAM as block device
- More overhead and complexity

- Initial RAM Filesystem
- Cpio archive
  - Handled natively in kernel
- Unpacked directly into RAM (rootfs)
- Faster and simpler



Back to kernel_init()

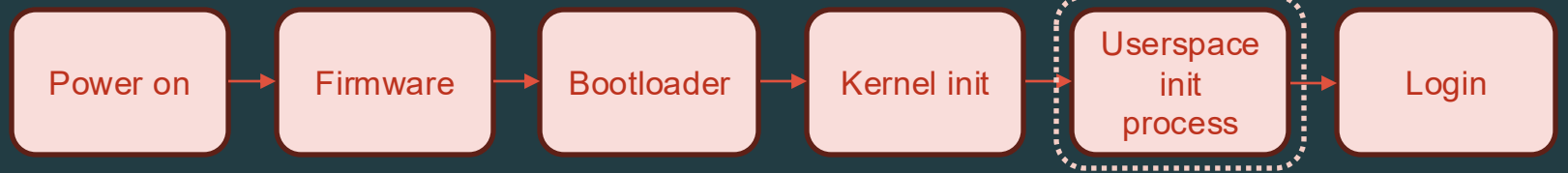
- Launch 1st userspace process (PID 1)

```
if (execute_command) {
        ret = run_init_process(execute_command);
        if (!ret)
                return 0;
        panic("Requested init %s failed (error %d).",
              execute_command, ret);
}

if (CONFIG_DEFAULT_INIT[0] != '\0') {
        ret = run_init_process(CONFIG_DEFAULT_INIT);
        if (ret)
                pr_err("Default init %s failed (error %d)\n",
                       CONFIG_DEFAULT_INIT, ret);
        else
                return 0;
}

if (!try_to_run_init_process("/sbin/init") ||
    !try_to_run_init_process("/etc/init") ||
    !try_to_run_init_process("/bin/init") ||
    !try_to_run_init_process("/bin/sh"))
        return 0;

panic("No working init found.  Try passing init= option to kernel. "
      "See Linux Documentation/admin-guide/init.rst for guidance.");
```

From init= kernel cmd line parameter

Init system

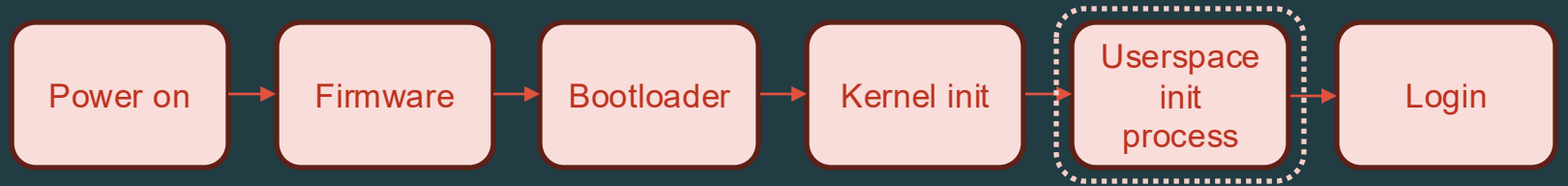
- Initializes processes on the system and prepares environment to achieve desired states
- Configures networking, mounts file systems, hands off to login prompts, etc
- init is the first process to begin on startup and the last to end on shutdown (PID=1)
- All processes are descendants of PID 1
- SysVinit was the primary init until it was replaced by systemd in the early 2010s



SysVinit

- Early linux init
- Uses runlevels 0-6 to define states
- Each runlevel has a series of scripts attached to it
- Does not run parallel processes
- It can be slow or get hung up on a process

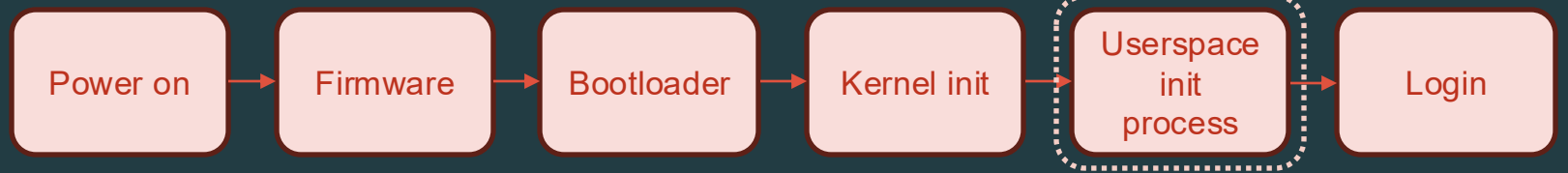
Runlevel	Function
0	Power off
1	Single user
2	Multi-user non-graphical, no network
3	Multi-user non-graphical
4	User-defined
5	Multi-user graphical
6	Reboot



Systemd

- Modern mainstream init process
- More capabilitites than SysVinit and more dynamic
- Uses units for processes
  - .socket, .service, .path, .target, etc
- Targets replace runlevels, whose requirements are fulfilled depending on the status of units
- Has awareness of dependencies and conflicts
- Feature bloat, can raise security concerns

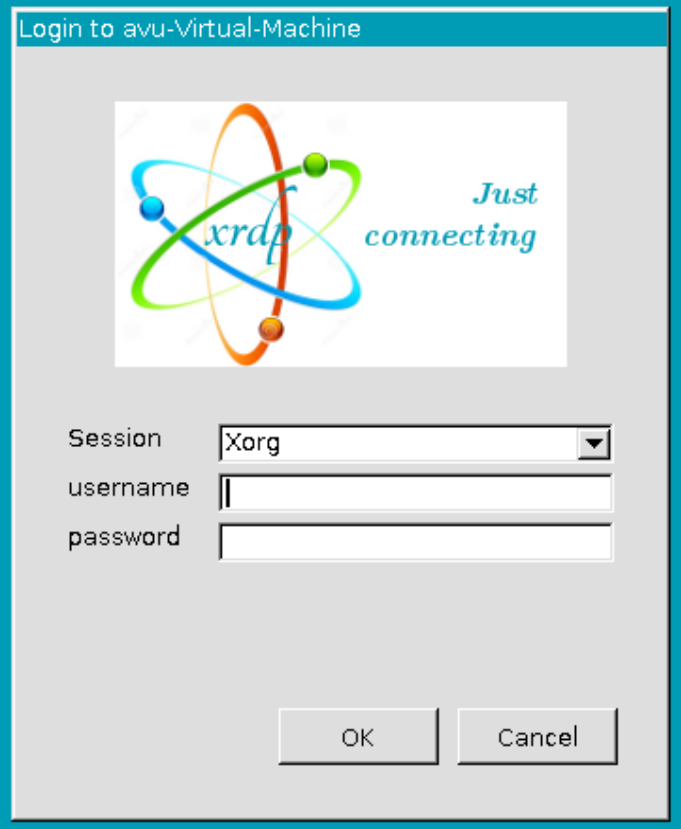
```
[    1.304075] xor: automatically using best checksumming function   avx
[    1.351676] Btrfs loaded, crc32c=crc32c-intel, zoned=yes, fsverity=yes
[    1.487314] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null). Quota mode: none.
[    1.582839] systemd[1]: Inserted module 'autofs4'
[    1.601655] systemd[1]: systemd 249.11-0ubuntu3.1 running in system mode (+PAM +AUDIT +SELINUX +APPARMOR
TLS -OPENSSL +ACL +BLKID +CURL +ELFUTILS -FIDO2 +IDN2 -IDN +IPTC +KMOD +LIBCRYPTSETUP -LIBFDISK +PCRE2 -PWQ
Z4 +XZ +ZLIB +ZSTD -XKBCOMMON +UTMP +SYSVINIT default-hierarchy=unified)
[    1.620443] systemd[1]: Detected virtualization microsoft.
[    1.620453] systemd[1]: Detected architecture x86-64.
[    1.620942] systemd[1]: Hostname set to <karsanchez-Virtual-Machine>.
[    1.747297] systemd[1]: Queued start job for default target Graphical Interface.
[    1.747304] systemd[1]: Unnecessary job was removed for /sys/devices/virtual/misc/vmbus!hv_fcopy.
[    1.747308] systemd[1]: Unnecessary job was removed for /sys/devices/virtual/misc/vmbus!hv_vss.
[    1.748627] systemd[1]: Created slice Slice /system/modprobe.
[    1.748801] systemd[1]: Created slice Slice /system/systemd-fsck.
[    1.748904] systemd[1]: Created slice User and Session Slice.
[    1.748939] systemd[1]: Started Forward Password Requests to Wall Directory Watch.
[    1.749036] systemd[1]: Set up automount Arbitrary Executable File Formats File System Automount Point.
[    1.749072] systemd[1]: Reached target Remote File Systems.
[    1.749080] systemd[1]: Reached target Slice Units.
[    1.749088] systemd[1]: Reached target Swaps.
[    1.749099] systemd[1]: Reached target Local Verity Protected Volumes.
[    1.749139] systemd[1]: Listening on Device-mapper event daemon FIFOs.
[    1.749186] systemd[1]: Listening on LVM2 poll daemon socket.
[    1.749233] systemd[1]: Listening on Syslog Socket.
[    1.749267] systemd[1]: Listening on fsck to fsckd communication Socket.
[    1.749290] systemd[1]: Listening on initctl Compatibility Named Pipe.
[    1.749372] systemd[1]: Listening on Journal Audit Socket.
[    1.749406] systemd[1]: Listening on Journal Socket (/dev/log).
[    1.749451] systemd[1]: Listening on Journal Socket.
[    1.749749] systemd[1]: Listening on udev Control Socket.
[    1.749796] systemd[1]: Listening on udev Kernel Socket.
[    1.750239] systemd[1]: Mounting Huge Pages File System...
[    1.750803] systemd[1]: Mounting POSIX Message Queue File System...
[    1.751340] systemd[1]: Mounting Kernel Debug File System...
[    1.751877] systemd[1]: Mounting Kernel Trace File System...
[    1.754018] systemd[1]: Starting Journal Service...
[    1.754172] systemd[1]: Finished Availability of block devices.
```

```
● apparmor.service - Load AppArmor profiles
     Loaded: loaded (/lib/systemd/system/apparmor.service; enabled; vendor preset: enabled)
     Active: active (exited) since Wed 2026-05-20 12:59:30 EDT; 4min 6s ago
       Docs: man:apparmor(7)
             https://gitlab.com/apparmor/apparmor/wikis/home/
    Process: 501 ExecStart=/lib/apparmor/apparmor.systemd reload (code=exited, status=0/SUCCESS)
   Main PID: 501 (code=exited, status=0/SUCCESS)
        CPU: 233ms

May 20 12:59:30 karsanchez-Virtual-Machine systemd[1]: Starting Load AppArmor profiles...
May 20 12:59:30 karsanchez-Virtual-Machine apparmor.systemd[501]: Restarting AppArmor
May 20 12:59:30 karsanchez-Virtual-Machine apparmor.systemd[501]: Reloading AppArmor profiles
May 20 12:59:30 karsanchez-Virtual-Machine apparmor.systemd[529]: Skipping profile in /etc/apparmor.d/disable: usr.sbin.rsyslogd
May 20 12:59:30 karsanchez-Virtual-Machine apparmor.systemd[530]: Warning: found usr.sbin.sssd in /etc/apparmor.d/force-complain, forcing co>
May 20 12:59:30 karsanchez-Virtual-Machine apparmor.systemd[530]: Warning from /etc/apparmor.d (/etc/apparmor.d/usr.sbin.sssd line 60): Cach>
May 20 12:59:30 karsanchez-Virtual-Machine systemd[1]: Finished Load AppArmor profiles.
```

```
[Unit]
Description=Load AppArmor profiles
DefaultDependencies=no
Before=sysinit.target
After=local-fs.target
After=systemd-journald-audit.socket
RequiresMountsFor=/var/cache/apparmor
AssertPathIsReadWrite=/sys/kernel/security/apparmor/.load
ConditionSecurity=apparmor
Documentation=man:apparmor(7)
Documentation=https://gitlab.com/apparmor/apparmor/wikis/home/

# Don't start this unit on the Ubuntu Live CD
ConditionPathExists=!/rofs/etc/apparmor.d

# Don't start this unit on the Debian Live CD when using overlayfs
ConditionPathExists=!/run/live/overlay/work

[Service]
Type=oneshot
ExecStart=/lib/apparmor/apparmor.systemd reload
ExecReload=/lib/apparmor/apparmor.systemd reload
ExecStop=/bin/true
RemainAfterExit=yes

[Install]
WantedBy=sysinit.target
```

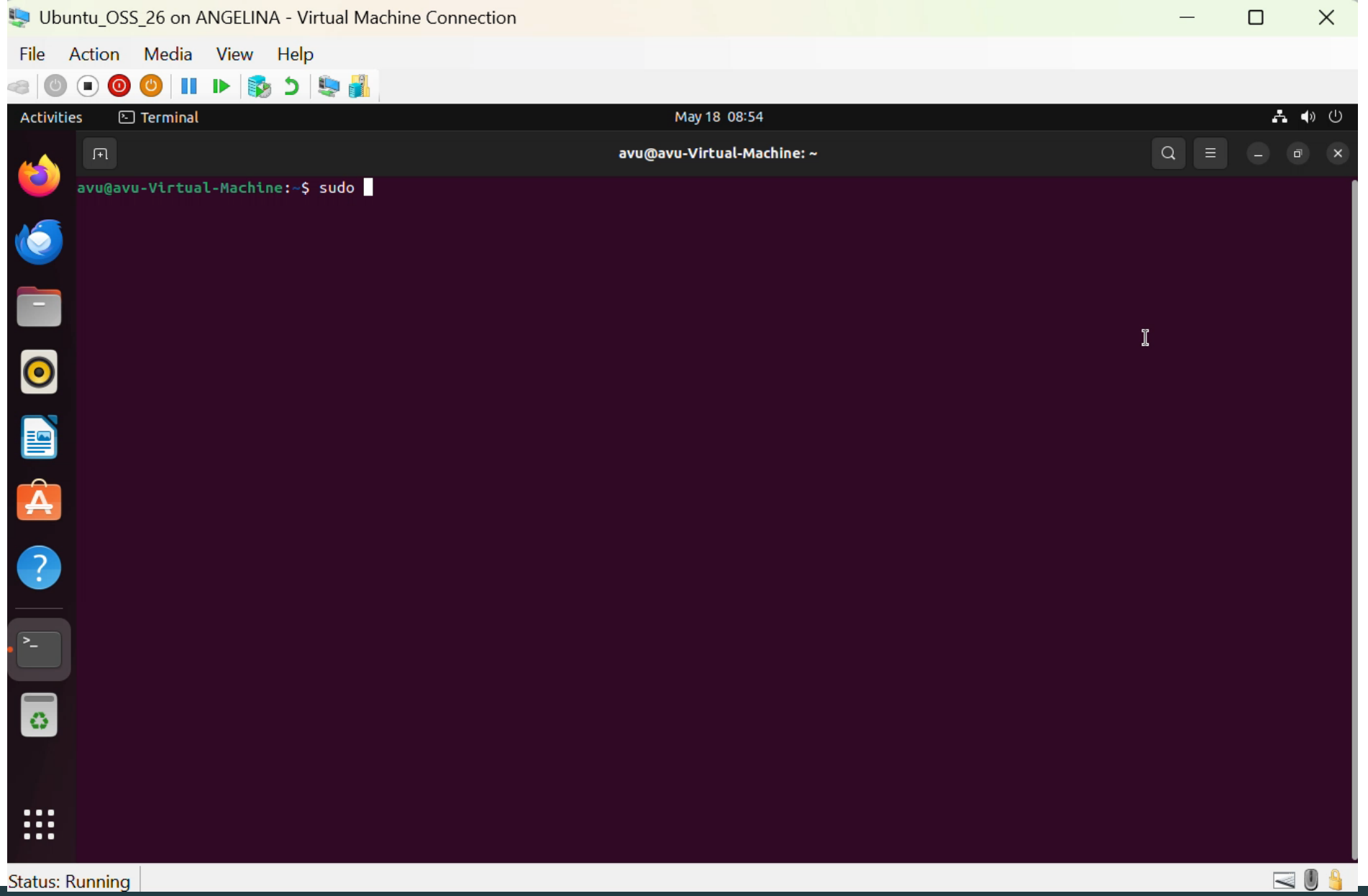
```
karsanchez@karsanchez-Virtual-Machine:~$ ls /lib/apparmor/
apparmor.systemd  profile-load  rc.apparmor.functions
```


Login to avu-Virtual-Machine
xrdp
Just
connecting
Session
Xorg
username
password
OK
Cancel

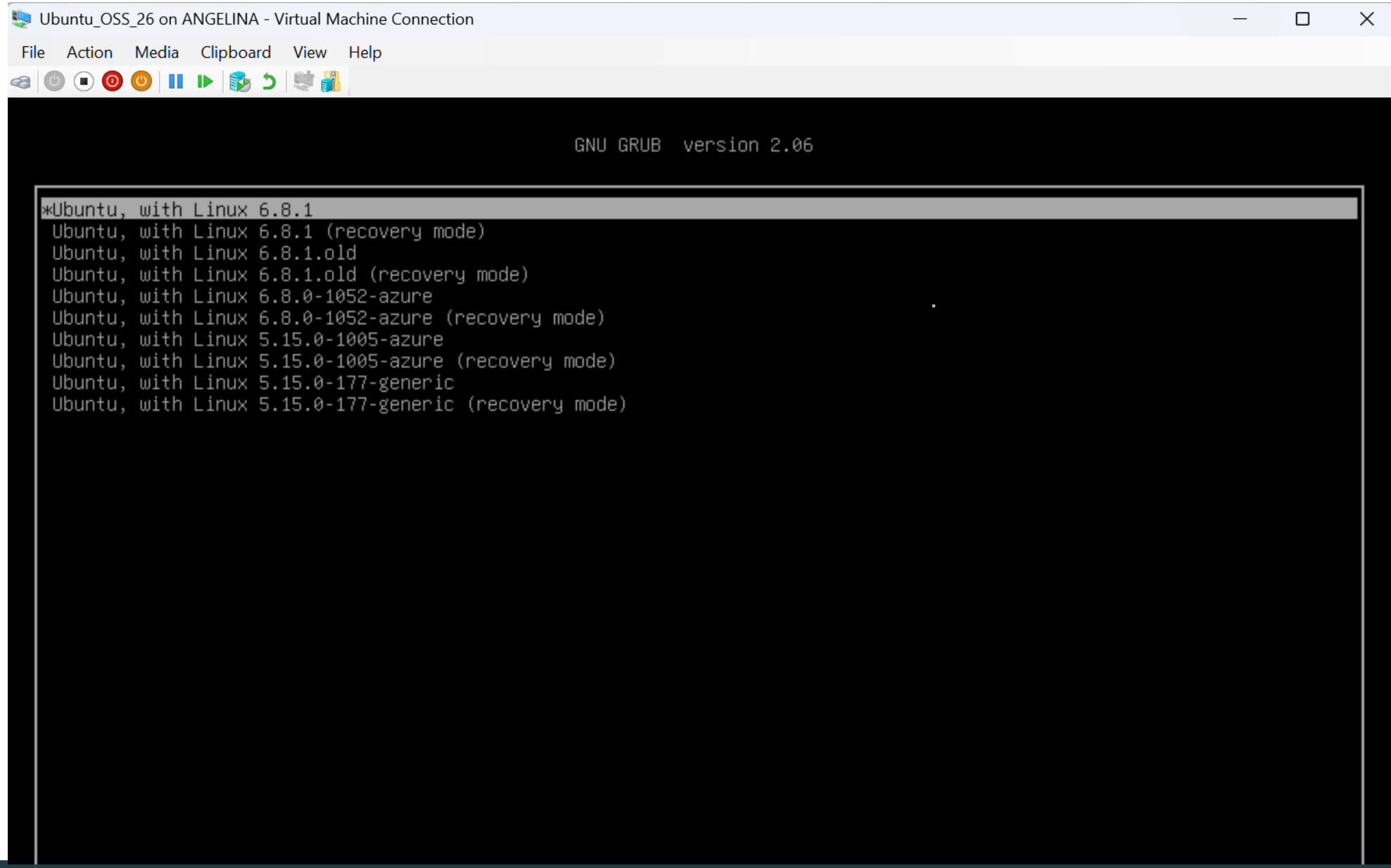
dmesg

- Kernel ring buffer logs

```
Supported log levels (priorities):
   emerg - system is unusable
   alert - action must be taken immediately
    crit - critical conditions
     err - error conditions
    warn - warning conditions
  notice - normal but significant condition
    info - informational
   debug - debug-level messages
```


Ubuntu_OSS_26 on ANGELINA - Virtual Machine Connection
File Action Media View Help
Activities Terminal May 18 08:54
avu@avu-Virtual-Machine: ~
avu@avu-Virtual-Machine:~$ sudo
Status: Running

Debug demo


Ubuntu_OSS_26 on ANGELINA - Virtual Machine Connection
File Action Media Clipboard View Help
GNU GRUB version 2.06
*Ubuntu, with Linux 6.8.1
Ubuntu, with Linux 6.8.1 (recovery mode)
Ubuntu, with Linux 6.8.1.old
Ubuntu, with Linux 6.8.1.old (recovery mode)
Ubuntu, with Linux 6.8.0-1052-azure
Ubuntu, with Linux 6.8.0-1052-azure (recovery mode)
Ubuntu, with Linux 5.15.0-1005-azure
Ubuntu, with Linux 5.15.0-1005-azure (recovery mode)
Ubuntu, with Linux 5.15.0-177-generic
Ubuntu, with Linux 5.15.0-177-generic (recovery mode)

Boot-time optimizations/Best practices

- systemd-analyze
- systemd-analyze blame

- Disable unnecessary services
- Optimize kernel command line
- Install /boot and root / on fast storage device

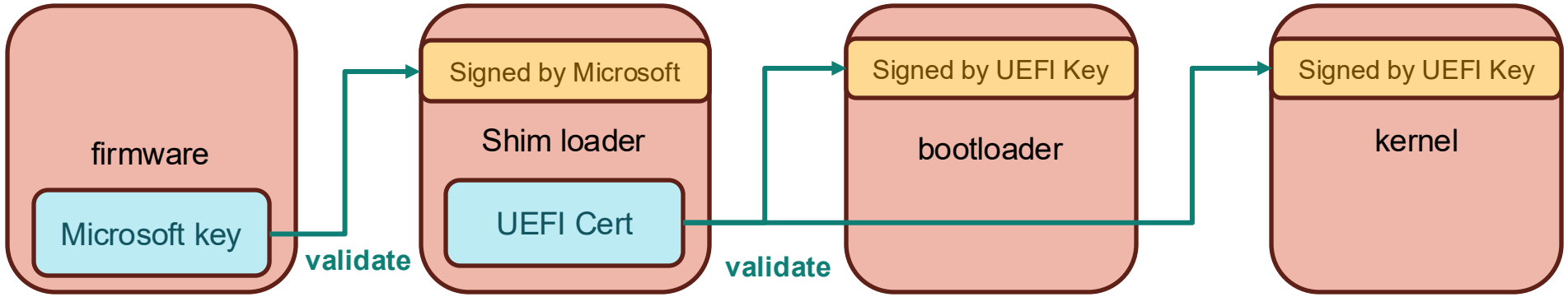
- Enable secure boot
- Password protect GRUB

```
avu@avu-Virtual-Machine:~$ systemd-analyze
Startup finished in 2.389s (kernel) + 13.088s (userspace) = 15.477s
graphical.target reached after 13.081s in userspace
```

```
avu@avu-Virtual-Machine:~$ systemd-analyze blame
12.249s plymouth-quit-wait.service
 1.059s xrdp.service
  303ms snapd.service
  238ms dev-sda1.device
  174ms networkd-dispatcher.service
  170ms udisks2.service
  153ms apparmor.service
  151ms NetworkManager-wait-online.service
  143ms dev-loop4.device
```

Secure Boot

- Ensure that binaries loaded during boot are trustworthy
- Establish a chain of trust
- Shim is signed by Microsoft, and trusting this is a critical assumption
- Attempting to load unvalidated binaries will cause boot to fail



Thank you

Questions?

THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

Embedded Linux Conference