CSCI 2461, Computer Networking 3 - Linux

Week 05 (Rev. 0)

Wednesday, February 07, 2018

Intro

Star Man Video

Star Man

StarMan Chillin

Figure 1: StarMan Chillin

Don't Panic

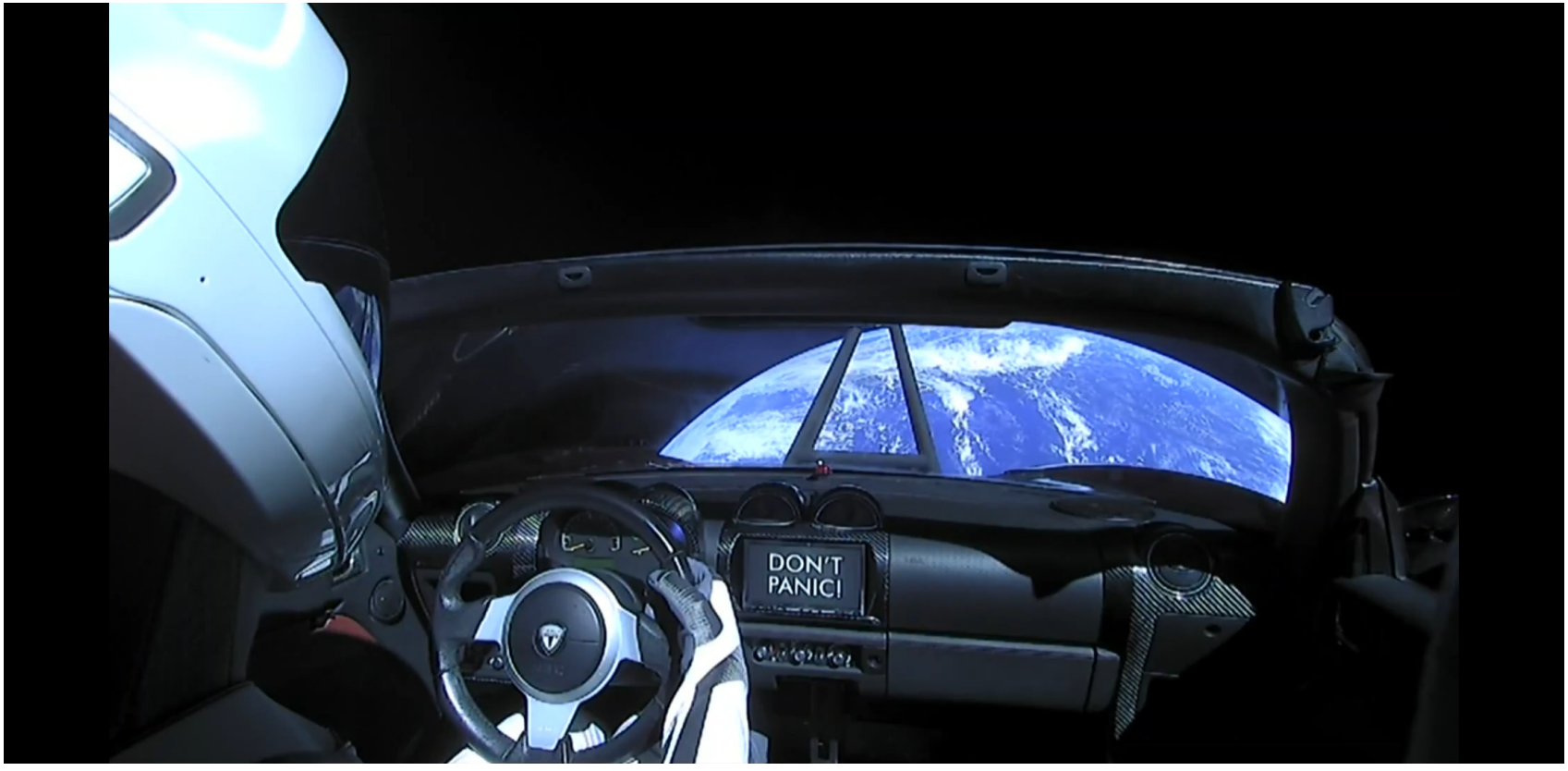

Figure 2: StarMan DontPanic

Due Homework

No Class on Feb 14

As a reminder, there will be **NO CLASS** next week on **Feb 14th** however assignments are *still due.*

Class resumes on **Feb 21st**

Opening

Class Plan

We're going to open up class today making sure your bootable USB's work and we'll troubleshoot any issues you've had looking deep into how the system boots. Then we'll review Chapter 4 "Devices" briefly, move to Chapter 5 "Disks", then if we have time we'll look at user space and how to build your own bootable images with `debootstrap`

1-on-1 Review

During the class lab time, we'll have in class 1-on-1 discussion with me today regarding your progress in class including how you're coming along, we'll review your assignments, and discuss where you need to focus your effort and what you can do to improve. This will give you the next two weeks to catch up before we move into building a physical network and the services on it.

Get Ready

Setup your workstation for class.

* Boot your Debian USB
* If your USB isn't ready, boot your Debian Live VM
* Get your book ready open to Chapter 4

Review: Capter 4

Filesystems & Types

Filesystems exist to allow users a structured way to interact with the kernel to access the hardware.

Windows primarily uses file systems like (ex)FAT and NTFS

Linux general purpose filesystem is called the "extended filesystem" (ext2-4) and has four major versions currently all of which are mostly compatible between each other. Additionally, Linux supports the ISO 9660 CD-ROM standard and Apple's HFS+

Filesystem Creation (make a local image)

Let's start with making some local partitions in files before working on hardware.

Create a 32 MB file with the *dd* command:

- `dd if=/dev/zero of=./32MB.img bs=1M count=32`

Explained:

- `if=/dev/zero` reads /dev/zero as an input file
- `of=./32MB.img` writes to 32MB.img in the current directory (./)
- `bs=1M` means to write with a 1 MegaByte block size
- `count=32` means to write 32 blocks.

Filesystem Creation (look at the image)

Now, let's look at what we just created and then put a filesystem onto it

- `hexdump ./32MB.img | less`

What is in there?

Filesystem Creation (make a filesystem)

Now, write the ext4 filesystem using the mkfs command:

- `mkfs -t ext4 ./32MB.img`

Explained:

- `mkfs` is the "make filesystem" series of commands
- `-t ext4` is the flag to specify the ext4 type of filesystem
- `./32MB.img` is a reference to your file that you just created with *dd*

There are many other filesystems and command shortcuts:

- `ls -l /sbin/mkfs.*`

Filesystem Mounting (mount our image)

We now have a filesystem in an image. Make a mount point, and mount it.

- `mkdir /mnt/tmp`
- `sudo mount ./32MB.img /mnt/tmp`
- `mount; df -h`
- `lsblk`

Now let's put a little "Hello World" file in there

- `cd /tmp/tmp; ls`
- `echo "Hello World" > ./hello.txt`

Now unmount the image

- `sudo umount /mnt/tmp`

Filesystem Storage (look for hello.txt)

Let's look for our file again with hexdump, this time with the –canonical option which will show us hex and the ASCII characters we wrote.

Make sure to pipe it to *less* so you can page through the output.

- `hexdump --canonical ./32MB.img | less`

Use "/" within less to search, e.g. type /ello and / again to go to the next result.

Special Filesystems

- /dev (devfs)
- /proc (proc filesystem)
- /sys (sysfs)
- /run (tmpfs)

Swap Sapce

Swap Space is like Windows pagefiles, it is disk space allocated as RAM type memory.

- `free`
- `dd if=/dev/zero of=./16MB.swap bs=1M count=16`
- `mkswap ./16MB.swap`
- `sudo chmod 0600 16MB.swap; sudo chown root 16MB.swap`
- `swapon ./16MB.swap`
- `swapon -s`

Inside a Filesystem (page 87)

Inode table connects to the data pool (of inodes)

Make some directories and files within those directories and make a *link*.

- `mkdir dir_1 dir_2`
- `echo "a" > dir_1/file_1`
- `echo "b" > dir_1/file_2`
- `echo "c" > dir_1/file_3`
- `echo "d" > dir_2/file_4`
- `ln dir_1/file_3 dir_2/file_5`

Inside a Filesystem (inodes)

Now take a look at the inode IDs

- `ls -iR dir_*`

Compare the far left column of dir_1/file_3 and dir_2/file_5

Ch 5: Linux Kernel

Startup Process

Identify aspects of each of these

1. CPU inspection
2. Memory inspection
3. Device bus discovery
4. Device discovery
5. Networking
6. Root filesystem mount
7. User space start

Kernel Log Sources

Other kernel log sources

- `less /var/log/kern.log`
- `less /var/log/messages`

Boot Parameters

Let's analyze the Kernel Boot Parameters

- `cat /proc/cmdline`

Before the Kernel

1. The PC BIOS or firmware initializes the hardware and searches its boot-order storage devices for boot code.
2. Upon finding the boot code, the BIOS/firmware loads and executes it. This is where GRUB begins.
3. The GRUB core loads.
4. The core initializes. At this point, GRUB can now access disks and filesystems.
5. GRUB identifies its boot partition and loads a configuration there (/boot/grub/grub.cfg)

Before the Kernel (cont)

6. GRUB gives the user a chance to change the configuration.
7. After a timeout or user action, GRUB executes the configuration in `grub.cfg`
8. GRUB may load additional modules in the boot partition.
9. GRUB executes a boot command to load and execute the kernel

Boot Loaders

What is GRUB? What is LILO?

- Interaction between BIOS/UEFI and the direct hardware
- The last direct iteraction before the Kernel

Lab: GRUB Exercise (5.5)

Open your book to Section 5.5 (Page 98), and reboot your USB or Virtual Machine and look at the bootloader options. Move the selector up and down (arrow keys) to interrupt the boot and stop the counter.

- Press `e` and compare the GRUB options
- Lookup the individual items and guess at what they do.
- Exit out of that window and at the `grub>` prompt type `ls`
- type `ls -l` at the `grub>` prompt
- `echo $root`

GRUB Configuration

Reboot into your USB or VM

- Open a terminal window
- `cd /boot; ls; cd grub`
- `less grub.cfg`

grub-mkconfig

Open the manual page for grub-mkconfig

- `man grub-mkconfig`
- `grub-mkconfig -o /boot/grub/grub.cfg`

grub-install

Open the manual page for grub-install

- `man grub-mkconfig`
- `grub-install /dev/sda`

Chainloading (5.7, pg 106)

Chain loading Windows, addition to `grub.cfg`

‘ menuentry “Windows” {

```
insmod chain
insmod ntfs
set root=(hd0,3)
chainloader +1
```

} ‘

Lab

Ending Lab

Use the rest of the class time to:

- Ask questions.
- Start your homework.
- Finish your USB Boot.
- Choose your Debian/Linux distribution for for Chapter 5
- Talk about your script, talk to a peer about their script.

Homework

Reading

- Read and apply “How Linux Works” Chapter 5 “Linux Kernel Booting”
- Read, review, and run peer scripts from GitHub in your VM.
- Installing Debian from Scratch
- QEMU Bootstrap
- Cross Debootstrap

Lab

- Finish setting up and customizing your bootable Debian based image on your USB drive, use *persistence*.
- Write a README.md collaborating with your peers explaining how you built the USB.
- Collaborate on writing about USB booting with your peers scripts.
- Extra Credit: Make your own bootable USB image using debootstrap.

Enjoy Valentines Day!