Linux Kernel Fundamentals: Chapter 2, Booting

You need to get to the system console. Using virt-manager and a VM is a handy way to do this.

1. Run `dmesg` and look to see if each output line starts with a time stamp in the form of `[     0.1234]`. Reboot. From your system console, interrupt GRUB and add the option `printk.time=1` to the kernel line (if you do not have a time stamp) or add `printk.time=0` (if you do have a time stamp). Continue with the bootup. After booting, log in and see if `dmesg` output now looks different. Were time stamps added or are they gone?

   Using https://wiki.centos.org/HowTos/Grub2, or another appropriate distro, make a custom GRUB entry that is the same as your current kernel's entry, but with some changes:

   a. Make the title say Custom Linux Boot Entry.

   b. Add the kernel command-line option `initcall_debug` to the end of the kernel line.

   c. For your distro, determine the `grub.cfg file` to use, and then make a new one with `grub2-mkconfig`. For example: `grub2-mkconfig -o /boot/grub/grub.cfg`

   d. Reboot, pick your new GRUB entry, and after it boots, `grep initcall` from the output of `dmesg.`

2. Interrupt GRUB, and choose your original kernel entry. At the end of the `vmlinuz` line, add `init=/bin/bash` and boot. What happened? Turn the power off and on, interrupt GRUB again, and this time, put `rdinit=/bin/sh` at the end and boot. What happens now?

   Reboot back into your full Linux environment.

3. Using `pstree`, can you determine which processes are direct descendants of PID 1?

4. Does your system have a program called `init`? Is `init` a soft link? Is PID 1 running `init`?