CIT 371 Lab 19: The Booting and Initialization Process

This lab can be done with SSH/PuTTY or with the Web Console. See the Student VM Access document for information on accessing your VMs.

Read chapter 11 sections 1-3 before working on this lab!

1. GRUB - RedHat 7 has moved from GRUB to GRUB2 as the boot loader process. Its hard to explore what GRUB2 does while booting, but we can explore parts of it once booted. After your system comes up, log in as yourself, open a terminal window and cd to /boot.
   1. As yourself, type **ls grub** and **ls grub2**. The contents of grub are limited to a compressed file as we now use grub2. *What happens when you type ls grub2?*
      1. **Our permission is denied.**
   2. **su** to **root** and type **cd grub2/i386-pc**. There are a number of .mod files. These are not text files but contain modules that the Linux kernel loads on demand. Examine the file **grub.cfg** and you will see a number of insmod instructions. Type **man insmod**. *What does this instruction do?* To see loaded modules, type **lsmod**. To remove modules, use rmmod. You can also add/remove modules with modprobe. Read about modules in section 8.2.4 of the textbook. For the most part, you do not want to remove modules.
      1. **This instruction tells us that this is a simple program to insert a module into a Linux kernel.**
   3. The boot process is recorded in the log file /var/log/dmesg. You can view this using the dmesg command. Type **dmesg | less** and scroll through the first portion of this file. You will see messages about cgroups pertaining to subsys, locating the boot image (kernel), examination of BIOS, VM-related messages, MTRR (memory type range register), ACPI (advanced configuration power interface), SRAT (static resource affinity table), NUMA (non-uniform memory access), additional cgroups initialization, PCI (peripheral control interface) interfaces and the bus, and other hardware initialization. This goes on for many screens. Eventually, systemd is launched. Type **q** to exit dmesg. From here, we move on to part 2 of this lab. There are no questions to answer from this part.
2. System initialization – GRUBs main task is to locate the Linux kernel in the /boot portion of the file system and load it into memory.
   1. cd back to /boot. The kernel is named vmlinuz followed by a string of numbers and letters. Locate this file in /boot. *What color is the file when you do an ls –l? How big is this file?*
      1. **The color is green. The file size is 6080. Or about 6.0 M.**
   2. After the kernel is running, Linux initializes the system using systemd (which is highly different from RedHat 6, read 16.4.1 about systemd instead of 11.4 about RedHat 6’s init process). Type **ps aux**. systemd is the first process run (PID of 1). Once running, it remains running the entire time your system is up, usually in the background. Type **cd /usr/lib/systemd**. Type **ls –l**, you will find a number of executable programs (green font) and directories (blue font). Of particular note are the subdirectories of *system* and *user*. cd to *system* and type **ls**. Many files have extensions of .target or .service (and a few others like .socket). These are configuration files to specify how systemd should start. The file default.target is read first to determine what .target file to use; type **ls –l default.target** and you will see this is a symbolic link which points to one of two main .target files: graphical.target or multi-user.target. Graphical includes operations to start the GUI but also runs multi-user.target so graphical.target does everything multi-user.target does but more. These two are roughly equivalent to RedHat 6’s runlevels of 5 and 3 respectively. View the contents of graphical.target (use cat, more, less). The Requires statement indicates that to run, this target file needs multi-user.target to run. View that file for its Requires, it needs basic.target. Look at basic.target to see what it requires, and continue to do this until you reach a target file that has does not require another .target file. *What specific order do these target files run (default.target would be first)?*
      1. **The order that is required to go after the graphical target in order is, multi-user.traget, rescue.service, rescue.target, display-manager.service .**
   3. To update default.target, you can change the symbolic link to another .target file but its safer to issue **systemctl set-default *newtarget*.target** where *newtarget* is the name of the target you want to use as the starting point. We explore systemctl in the next lab, but for now, type systemctl get-default. *What is the response?*
      1. **The response is graphical.target**
   4. Target files run in a particular order based on *dependencies* of the order that services must start. Target files cause .service files to run. Look at firewalld.service to view a .service file. .service files are similar to .target file but contain two additional sections: [Service] and [Install]. [Service] specifies how the service should run using ExecStart, and where the service’s configuration file(s) is. *How does firewalld start (what command)?* The Type entry indicates *how* the process starts. For firewalld, it indicates dbus. *What does dbus mean?* (hint: see chapter 16.8.3)
      1. **Systemctl start firewall. Dbus means desktop bus and is a software bus.**
   5. Re-examine /usr/lib/systemd/system. Aside from .target and .service files, *what other types of file extensions do you find?* Each one of these types is a type of unit. systemd supports a number of unit types, each with its own file extension. For each type of unit file, there are different sections (such as [Service] for .service files and [Timer] for .timer files). Look at a [Timer] section of a .timer file, *what types of information are specified in that section?*
      1. **.socket, .mount, .wants, .timer, and more. Looking into a timer file we can see specific commands for timed events such as on unit active sec = 1 day.**
   6. Services and other units have dependencies. These are indicated using directives Requires, Wants, After and Before. *What do After and Before denote?* (hint: see table 16.2).
      1. **After denoting a list of targets/services that must start before this one, and after denoting List of targets/services that must start after this one.**
   7. cd to /etc/systemd/system and you will find a number of *wants* directories. One such directory exists for every .target file. These directories contain symbolic links to .target and .service files required by this .target file. For instance, if a target requires foo.service, then there is a symbolic link from this target’s .wants directory in /etc to /usr/lib/systemd/system/foo.service. In this way, you can determine the dependencies. *What file(s) does default.target want? What file(s) does basic.target want? How many files does multi-user.target want?*
      1. **Systemd-readahead-collect.service, Systemd-readahead-replay.service . Basic.target want microcode.service, rhel-dmesg.service. Multi-user target want is in the picture below since there was so many.**
      2. Timeline

         Description automatically generated
3. RedHat 6 used the Upstart initialization process, which invoked script files that date back to Unix System V. Services would start based on a runlevel. Some of these files still exist.
   1. In older versions of Linux/Unix, the first process to run was called init. Type **ls –l /usr/sbin/init**. init still exists but is now a symbolic link. *What file is it pointing to?* The first thing init did was inspect the file /etc/inittab. This file still exists. Look at its contents, it consists only of comments now but previous it indicated the runlevel. By default, we start graphcial.target, *what runlevel does this correspond?* 
      1. **../lib/systemd/systemd. This run level is 5 since we are accessing graphical displays.**
   2. Given the runlevel, init executed start-up scripts in directories /etc/rc.d/rc#.d (where # was the runlevel 0-6). These directories contained symbolic links to service startup scripts in /etc/init.d. The symbolic links were named S##service or K##service (S for start, K for kill or stop) where ## was a two-digit number to indicate the order that services should be started/stopped (this was how dependencies were handled rather than through Wants/Requires statements). cd to /etc/init.d and type **ls -l**. Only two such script files remain: netconsole and network (these are still here as backup services to the new network service called NetworkManager). Aside from starting up services, init would also assign various environment variables, load modules and take care of other aspects of initialization. There are no questions to answer in this part.

Shut down your VM if desired, disconnect from the VPN if you are using it, and submit your lab report. Make sure you read 16.4.1 if you have not already done so.