Chapter 1 ICE

1. No answer required.

2. No answer required.

3. On our systems, it is possible to affect the order of booting by pressing a function key before the firmware finds the boot medium. We then enter a “Setup” mode, and can select the order of boot media searched. This is useful, for example, when we want to boot into another system, such as Clonezilla, on a USB thumbdrive, or DVD medium. We can then use Clonezilla to “clone” our system disks and data disks.

4. No answer required, but USB3 is the fastest method of booting.

5. Those programs are insecure, for example ftp sends passwords over the network in plain text. This allows anyone monitoring your session to capture and use the password for your account.

6. **groups**

7. No answer required.

8. No answer required.

9. No answer required.

10. If you don’t remove the old home directory and its contents, nothing changes.

11. It was automounted, and the **findmnt** command listed it as /media/bob/7CF8-00D8 /dev/sdb1. No answer required for the remainder of this In-Chapter Exercise.

12. Because you unmounted it to use **fdisk**.

13. No answer required.

14. No answer required.

15. No answer required.

16. /*data/*datafile20

17. /*data*/text1.txt

18. The pathnames to the mirrored text files would be the same, /data2/textfilename! You can’t edit them separately. This is the salient difference between ZFS mirrored and traditional operating system mirrored disks. On a traditional system, for example, the text files would be mirrored on drives C: and D:, so you could designate different pathnames to the two mirrored files. Not so in ZFS.

19. No, you can’t mix mirrored pairs with single disks.

20. No, as can be seen by what disk usages show in the **zpool iostat -v** command in steps 21. and 22.. data2 is a pool with two mirrors.

21. No answer required.

22. Two assumptions. That you have access to the account on that remote host, via a username and password pair. Also, that both systems have the **rsync** command installed.

23. No answer required.

24. No answer required.

25. With the **gunzip** command, with the proper argument specifying that other directory.

26. No answer required.

27. No answer required.

28. They will restart on reboot, because APT places the correct systemd service unit installation instructions on a Raspberry Pi for them in /usr/lib/systemd/system. You could know this without booting if you examine their service unit files.

29. No answer required.

30. No answer required.

31. No answer required.

32. bob@raspberrypi:~ $ **free**

total used free shared buff/cache available

Mem: 3884368 857128 1266204 203156 1761036 2746864

Swap: 102396 0 102396

When you see numbers like 1024, 3000, or 100 associated with CPU allocation, they likely refer to specific values used for configuring CPU-related settings, but they don't directly correspond to the percentage representation you typically see in **top**. To interpret CPU utilization as a percentage using **top**, you would look at the %CPU column and the other related CPU metrics. We set them to 3000 to give those processes a bigger percentage share of the CPU.

33. Edit /etc/network/interfaces, and delete the assignment there.

34. Edit /etc/network/interfaces, and add new assignments there.

35. Yes.

\* Case 1: Raspberry Pi OS with NetworkManager enabled  
  
If you’ve already switched your Pi 5 to use NetworkManager instead of dhcpcd, then it works the same way as on a desktop Linux box:  
  
Via GUI (if you installed a desktop environment with NetworkManager)  
  
Open Network Settings from the system tray (you may see the little up-down arrow or Wi-Fi icon).  
  
Select your network (e.g. Wired or Wi-Fi).  
  
Click the gear / settings icon.  
  
Go to the IPv4 tab.  
  
Set Method = Manual.  
  
Enter your primary IP (Address, Netmask, Gateway).  
  
Use the + (Add) button to add more IP addresses.  
  
Apply, then reconnect the interface.  
  
On Raspberry Pi OS with GNOME, you can also launch the advanced editor with:  
  
nm-connection-editor  
  
  
That gives you a more complete NetworkManager GUI.  
  
Via CLI (nmcli)  
  
For a wired connection on the Pi 5, typically named Wired connection 1:  
  
nmcli con mod "Wired connection 1" ipv4.addresses "192.168.1.50/24 192.168.1.51/24"  
nmcli con mod "Wired connection 1" ipv4.method manual  
nmcli con up "Wired connection 1"  
  
  
That assigns two static IPs on the same NIC.

\*Case 2: Raspberry Pi OS default (dhcpcd instead of NetworkManager)  
  
If you haven’t switched to NetworkManager, then the Pi won’t show the GNOME-style Network settings. Instead, you edit /etc/dhcpcd.conf.  
  
Example:  
  
interface eth0  
 static ip\_address=192.168.1.50/24  
 static ip\_address=192.168.1.51/24  
 static routers=192.168.1.1  
 static domain\_name\_servers=8.8.8.8  
  
  
Notice you can declare multiple static ip\_address lines for the same interface. Restart with:  
  
sudo systemctl restart dhcpcd  
  
If you enable NetworkManager on your Pi 5 → use GUI (nm-connection-editor) or nmcli just like a desktop.  
  
If you stick with dhcpcd (default) → edit /etc/dhcpcd.conf and add multiple static ip\_address lines.

36. On our Raspberry Pi system-

bob:x:1000:1000:,,,:/home/bob:/bin/bash

username:encrypted password: UID:GID:full name:home directory: login shell

37. The Linux kernel, in conjunction with systemd.

38. No answer required.

39.What Linux capabilities are (and why use them)  
  
Traditionally, a process was either root (UID 0, all powers) or not (very few powers). Linux capabilities split root’s superpowers into small, named privileges (e.g., “bind low ports,” “raw sockets,” “chown arbitrary files”), so you can grant only what’s needed—reducing blast radius compared to setuid-root.  
  
A few commonly used caps:  
  
cap\_net\_bind\_service – bind to TCP/UDP ports <1024  
  
cap\_net\_raw – open raw sockets (classic need for ping)  
  
cap\_dac\_read\_search – bypass file read/search permission checks (read-only)  
  
cap\_sys\_admin – huge, avoid unless you really need it  
  
There are three important per-process sets:  
  
Permitted (P): which caps the process may use.  
  
Effective (E): which caps are currently active.  
  
Inheritable (I): which caps can be passed across exec().  
For file capabilities (set on an executable), you usually set +ep to place the cap in P and mark it E on exec.  
  
How to apply capabilities from the command line  
  
You typically use the setcap / getcap tools (package: libcap2-bin on Debian/Raspberry Pi OS).

Example: Grant the capability to the Python interpreter  
  
Find your Python binary and set the cap on the ELF interpreter, not a symlink:

PY=$(command -v python3)

# Resolve to the real file to avoid setting the cap on a symlink

REALPY=$(readlink -f "$PY")

echo "$REALPY"

sudo setcap 'cap\_net\_bind\_service=+ep' "$REALPY"

40. Examine the sudoers file, if you are allowed to.

41. No answer required.

42. Not ordinarily, but if you have sudo privilege, you can give the command **sudo su -**, and then you become root. Type **exit** to logout of root.

43. Coming from the Internet onto your LAN, and targeting the network stack.

44. No answer required.

45. Install fail2ban. See the current bans are in effect, using the fail2ban-client command **sudo fail2ban-client status service\_name**. Then unblock the IP address with the following command- **sudo fail2ban-client set servie\_name unbanip IP\_ADDRESS** , where IP\_ADDRESS is the address you want to unblock. Finally, to check use the command **sudo fail2ban-client status service\_name** .

46. Wireshark is a popular and widely used network packet analyzer. It is an open-source software that allows users to capture, analyze, and inspect the data packets traveling over your LAN, for example. These packets contain information about network traffic, including details about the source and destination addresses, protocols being used, payload data, and more.

47. **sudo ufw status**

**sudo systemctl status ufw**

Check to see if it’s enabled at boot-

**sudo systemctl is-enabled ufw**

On our Raspberry Pi system, it was not enabled, and no rules were in effect by default.

48. No answer required.

49. No answer required.

50. No answer required.

51. KVM, which stands for Kernel-based Virtual Machine, is an open-source virtualization technology that allows you to create and manage virtual machines (VMs) on Linux. It is a part of the Linux kernel and provides a hypervisor that enables you to run multiple virtual machines with various operating systems on a single physical machine. See Volume 3, Chapter 3 on LXC/LXD and Docker for a better idea of a comparison. Also, KVM is free, compared to Amazon Web Services or Google Cloud.