Linux Live Boot Disk

Contents

0.1 Materials

- USB Drive (Suggest 8GB)
- Linux Live CD (Suggestions below)

The provided USB Disk is 8GB in size.

1 Skills/Concepts

This module will include the following (among other) concepts:

- BIOS/UEFI
- console use
- filesystems
- getting help for commands
- manipulating files
- the boot process

1.1 BIOS

It's the program that resides in the computer's physical hardware and executes when you turn the computer on.

From Wikipedia:

> The BIOS (/ˈbaɪ.ɒs/, an acronym for Basic Input/Output System and also known as the System BIOS, ROM BIOS or PC BIOS) is a type of firmware used to perform hardware initialization during the booting process (power-on startup) on IBM PC compatible computers, and to provide runtime services for operating systems and programs.[1]

1.2 UEFI

It's the replacement for the BIOS and exists in most modern desktop computers.

From Wikipedia:

> The Unified Extensible Firmware Interface (UEFI, pronounced as an initialism U-E-F-I or like "unify" without the n[a]) is a specification that defines a software interface between an operating system and platform firmware. UEFI replaces the Basic Input/Output System (BIOS) firmware interface originally present in all IBM PC-compatible personal computers, with most UEFI firmware implementations providing legacy support for BIOS services. UEFI can support remote diagnostics and repair of computers, even with no operating system installed.[2]

Generally, we will be using the BIOS or the BIOS legacy fall-back mode provided by UEFI in this module. That is, most modern systems support UEFI and will use that to boot, but our various Live systems will rely on the fall-back to BIOS mode for execution.

1.3 Console Use

This module relies heavily on the command-line. You will be using a terminal emulator to execute the following commands.

Command	Description
dd	Copies data from a stream or file to another stream or file
diff	Displays differences
e2label	Labels ext filesystems
fdisk,parted	Manipulate partition tables
man	View on-line manual pages
mkfs.ext4	Creates ext4 filesystems
mkfs.vfat	Creates FAT filesystems
sudo	Execute a command as another user
sync	Flushes/waits for pending disk operations

[1] Retrieved from Wikipedia on 2016-04-13.
[2] Retrieved from Wikipedia on 2016-04-13.

1.4 Filesystems

The filesystem is the layer of abstraction between the programs and the disk where data is stored.

From Wikipedia:

> In computing, a file system (or filesystem) is used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their names is called a "file system".[3]

The specific file systems we will be dealing with in this module are Ext4, FAT32, and ISO. Ext4 is the fourth iteration of the extended file system that replaces ext3, ext2, and ext which originally was created to work around limitations in MINIX.

FAT32 is a Windows-compatible file system that is fairly widely supported and so it makes an okay format to use on a thumb drive or other removable device where its limitations aren't likely to cause too much trouble (no case sensitivity, max file size of one byte less than 4GB).

ISO is commonly used as an abbreviation for the ISO 9660 file system (ISO = international standards organization) that is used as the basic file system on CD and DVD. Generically people will say ISO when they really mean a CD/DVD disk image.

1.5 Getting Help for Commands

Technology these days makes Internet searching just as fast as any built in help, but in addition to googling for stack exchange articles, there is a built-in help or on-line manual for most commands under Linux. Built-in help can typically be accessed by passing `--help` or `-h` to the command of interest and it will typically reply back with a short usage block.

In addition to the built-in help, you can refer to manual pages for more usage information for most commands. These pages are generally terribly formatted, frustrating, and are entirely unhelpful if you're not used to reading them.

Various packages usually have some additional documentation in the form of a README or guide that can be found in `/usr/share/doc/`. Sometimes you will need to install an addition `-doc` package for whatever the package is to get these files.

I suggest you use the following approach to getting help:

- Check for a man page
- Check for documentation in `/usr/share/doc/`
- Google

That aside, the general approach to finding help for a given command should be whatever you are most comfortable with or whatever happens to be available. I recommended the above approach because it will help you get used to reading technical documentation so that in the future it will be easier to comprehend whatever documentation is available.

1.6 Manipulating Files

There's not much to be said about this topic, you'll be manipulating files via the command-line.

1.7 The Boot Process

What does any of this module have to do with the computer boot up process? You'll be creating a boot disk and it will work because of how the system boot up process works. Or if it doesn't work, you'll start debugging[4] by examining the assumptions we've made about the system boot-up process to figure out which assumptions are false.

The traditional BIOS boot-up process involves a series of hardware diagnostics or Power-on Self Test (POST) after which the BIOS copies the first 512K from the default boot device into the computer's RAM, then jumps to that memory address and begins to execute whatever code it finds there.

[3]Retrieved from Wikipedia on 2016-04-13.

[4]Debugging can be considered the process of eliminating false assumptions to find problems or bugs with a given process. The name refers to the act of removing a literal insect from a machine so that it can resume normal operations but it has become synonymous with the process of manually walking through code or an algorithm to understand exactly what it is doing (commonly versus what it is assumed to be doing).

That's approximately what happens when we try to boot our Linux Live drives. The BIOS will copy the first 512K, then within that 512K there is a boot-loader that carries out whatever additional instructions are necessary to continue bootstrapping the actual operating system. For our Live CD this means executing ISOLINUX or SYSLINUX which in turn locates the Linux kernel, copies it into RAM, and then jumps to executing the typical startup process.

If you had any trouble following that, don't worry, an in-depth understanding isn't really necessary at this point.

2 Preparation

We'll be booting from a Live CD in order to provide a common work environment. In order to minimize accidentally destroying data, you may wish to disable hard drive access in your system BIOS. The following Live CD are all fine to use as a starting point for this module:

- Kubuntu (KDE)
- Lubuntu (LXDE)
- Xubuntu (XFCE)

These are all Ubuntu-based distributions that mainly vary in their use of desktop interface. You may also use the stock Ubuntu, it's not a first choice for us because it's slightly more resource hungry than the distributions mentioned above.

3 Standard Live CD

In this module we will create a very simple Live boot USB by copying a source ISO directly to the block device representing the thumb drive. This will destroy the data at the beginning of the disk and replace it with the ISO. If the wrong target disk is chosen during the write, irreversible damage to the data stored on the disk will result.

General Steps:

1. Boot the Live CD/DVD
2. Copy the ISO to the USB disk
3. Reboot and Test

3.1 Boot the Live CD/DVD

Insert the CD or DVD and reboot the computer. There is a key that needs to be pressed in order to boot from the drive, pay attention at power on for a message that will display briefly. For the systems we are using, it should be the F12 key. If that fails, press the `esc` key to pause after POST, then press F10 for the boot menu.

On most systems, this approach will work:

1. Tap the delete key until you get into the UEFI/BIOS.

2. Locate the boot menu, and from the boot menu, select the BIOS entry for the thumb drive (usb hard drive).

3. In the case of the drives distributed with this lesson, it's the SanDisk Cruzer Glide entry.

The system will take quite a bit longer to boot up from the CD/DVD than it does to boot from the hard drive.

3.2 Copy the ISO to the USB disk

3.2.1 CAUTION

Targeting the wrong partition during this tutorial could result in data loss. Be sure to confirm the target drive. For example, unplug and replug the device and then check `dmesg` to see the name of the drive. Or, use `sudo fdisk -l` to scan devices, then plug the drive in, run `sudo fdisk -l` again and compare the output.

For example:

```
sudo fdisk -l >/tmp/none
sudo fdisk -l >/tmp/some
diff /tmp/none /tmp/some | grep Disk
```

```
> Disk /dev/sdd: 58.2 GiB, 62518853632 bytes, 122107136 sectors
```

In this example, `/dev/sdd` is the target device.

3.2.2 Process

The following steps will be carried out in a terminal. Press `ctrl+alt+t` or otherwise open a terminal through whatever means the Live Disk provides.

1. Identify the source device. Note: If you are repeating the steps in this module and experimenting with different ISO images, you will instead use a file name as the source instead of a target device. For this module, you will likely use `/dev/sdc` as the source.
2. Identify the target device. See the example above in the CAUTION section for instructions on how to safely identify the target device. These instructions will use `/dev/sdX` as the target, but you will need to substitute the real target. If you inadvertently type `/dev/sdX` the only error should be a message instead of catastrophic data loss.
3. Use dd to copy from the source CD/DVD to the target USB drive.

```
    sudo dd if=/dev/sdc of=/dev/sdX
```

4. Use `sync` to ensure the device has finished writing.

```
    sync
```

This command will not return until all disks have finished writing out their cached data. This ensures that all of the data from the previous dd command has finished writing.

5. Reboot, remove the CD/DVD, and boot from the thumb drive. This is much the same process as was used to boot from the CD/DVD. When the system comes up, the user name is `root` and the password is `toor`.

3.2.3 Conclusion

A very simple boot disk was created by copying the ISO directly to the thumb drive. This works because the image copied is smaller than the total size of the thumb drive. The extra space is not used, nor is it currently accessible.

4 Persistent Live CD

Create a Persistent Kali Live thumb drive.[5]

You can download a copy of the Kali iso here, we used the 64 bit version at the meeting.

4.1 Using Kali

1. Copy the Kali ISO to the thumb drive:

```
    dd if=kali-linux-2016.1-amd64.iso of=/dev/sdX
```

This step will take quite a while to complete, see the appendix on dd for more background and information on how to get status information.

2. Discover the starting location for the persistent partition:

[5] Kali Linux Live USB Persistence instructions.

```
ls -lh kali-linux-2016.1-amd64.iso
```

```
-rw-rw-r-- 1 user group 2.8G Mar 25 21:19 kali-linux-2016.1-amd64.iso
```

2. Create an additional partition for persistent storage, using the size of the ISO as an offset for the beginning of the partition, and use the remaining space:

```
sudo parted /dev/sdX mkpart primary 2.8g 7g
```

A series of warnings will result regarding not being able to use the exact figures, accept the warnings by entering `Ignore` to each warning.

3. Label and format the new partition:

```
sudo mkfs.ext4 -L persistence /dev/sdX3
```

4. Mount the new partition, and create a configuration file that Kali is able to find:

```
udisksctl mount -b /dev/sdX3
echo "/ union" | sudo tee -a /media/user/persistence/persistence.conf
udisksctl unmount -b /dev/sdX3
```

5. Flush Buffers:

```
sync
```

6. Reboot and test! Be sure to press the down arrow when the Kali menu appears until the entry featuring persistence is selected, then press enter.

5 Appendix - dd

The dd command used in the module will take a long time to run and while it is running, it provides no output as to what it is doing. This is not uncommon for Linux commands. Unless executed in a verbose mode, they tend to only produce output when there is an error, or in the case of dd, when they have completed. Luckily, the dd command does have a mechanism for getting status.

To find it, check the manual page. If you're lucky, you'll find the part of the page that explains that you can use another command (`kill`) to send a special signal to the dd process and then the dd process will yield its current progress and speed.

In order to execute `kill` on the process, you need to know the process identifier (PID) of the dd process. You can find it with the following command:

```
ps -eo pid,args | grep 'dd'
```

The result will be a row of output similar to the following:

```
23424 dd if=/dev/sdc of=/dev/sdX
```

In this example, the PID is 23424. To instruct dd to print usage statistics, you could execute the following command:

```
kill -USR1 23424
```

The terminal where dd is running will yield output like the following:

```
1974272+0 records in
1974272+0 records out
1010827264 bytes (1.0 GB) copied, 322.534 s, 3.1 MB/s
```

Try not to mis-type the command, or you could, at worst, inadvertently terminate the process or a different process.

5.1 Reading the dd man page

Open the man page, that is try the following:

```
man dd
```

You will immediately see the manual page for the dd command. In the top left, you will see the command being described (DD in this case) followed by a numeral in parenthesis. This number is a section number. Some manual pages are actually available in multiple sections. If you know there is a different section for a given command, you can provide the section number in your invocation of man. For example, you could have invoked `man 1 dd` to view the current manual page.

Next you will see labels on the left in all capital letters. Under `NAME` you will typically see the command name followed by a one-line description.

Under `SYNOPSIS` you will see a description of the arguments for the command. Any arguments in brackets are typically optional. Any arguments that may repeat are usually followed by ellipsis. The first entry under synopsis shows that dd can either be invoked with optionally repeating operands or a single option.

Under the `DESCRIPTION` it immediately jumps into describing the operands. Operands are provided on the command-line without any leading hyphens. If you scroll down (space, j, arrows, or ctrl+D,…) you will eventually reach the section on options. `--help` and `--version` are the only options, you may provide either one or the other, as per the synopsis.

For our purposes, we're not currently interested in all of the options or what they do, we just want to know how to get status from an already executing dd command in order to find out how many bytes it has copied.

Man pages support searching, but, rather unhelpfully, there is no good search term we can use in order to find the section that describes how to get status. Some manual pages provide an EXAMPLES sub-section, which makes it easy to locate, but this one does not. Your only option is to scroll and read through the page. Or, if you have looked at the page before, you might know that getting status has something to do with the `kill` command. So, search for "kill" by typing `/kill` and hitting enter.

The manual will reward you with a rather intimidating example:

```
    $ dd if=/dev/zero of=/dev/null& pid=$!
    $ kill -USR1 $pid; sleep 1; kill $pid
```

Directly before the example is a slightly-but-not-really helpful description saying that you can use a USR1 signal to display I/O statistics. It is a rather contrived example but what it is showing is starting a dd to copy a sequence of zeroes from the special `/dev/zero` device and writing the output to the special `/dev/null` device. The command is forked into the background by using the & and then the PID is being saved to a variable. The next command is showing sending the USR1 signal to the PID saved in the variable from the previous line. Then after sleeping for one second, the contrived dd command is sent an actual kill to terminate it.

If you continue scrolling you will see other sub-sections of the manual page, such as `AUTHOR`, `REPORTING BUGS`, etc…