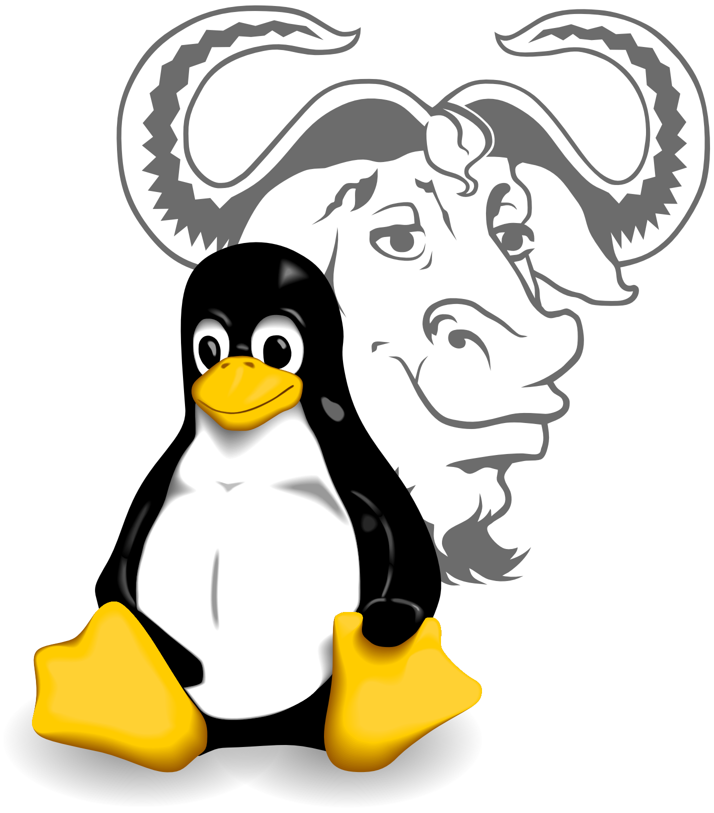
Document 1

What is Linux?



From smartphones to cars, supercomputers and home appliances, home desktops to enterprise servers, the Linux operating system is everywhere.

Linux has been around since the mid-1990s and has since reached a user-base that spans the globe. Linux is actually everywhere: It's in your phones, your thermostats, in your cars, refrigerators, Roku devices, and televisions. It also runs most of the Internet, all of the world's top 500 supercomputers, and the world's stock exchanges.

But besides being the platform of choice to run desktops, servers, and embedded systems across the globe, Linux is one of the most reliable, secure and worry-free operating systems available.

Here is all the information you need to get up to speed on the Linux platform.

What is Linux?

Just like Windows, iOS, and Mac OS, Linux is an operating system. In fact, one of the most popular platforms on the planet, Android, is powered by the Linux operating system. An operating system is software that manages all of the hardware resources associated with your desktop or laptop. To put it simply, the operating system manages the communication between your software and your hardware. Without the operating system (OS), the software wouldn't function.

The Linux operating system comprises several different pieces:

1. **Bootloader** – The software that manages the boot process of your computer. For most users, this will simply be a splash screen that pops up and eventually goes away to boot into the operating system.
2. **Kernel** – This is the one piece of the whole that is actually called 'Linux'. The kernel is the core of the system and manages the CPU, memory, and peripheral devices. The kernel is the lowest level of the OS.
3. **Init system** – This is a sub-system that bootstraps the user space and is charged with controlling daemons. One of the most widely used init systems is systemd, which also happens to be one of the most controversial. It is the init system that manages the boot process, once the initial booting is handed over from the bootloader (i.e., GRUB or GRand Unified Bootloader).
4. **Daemons** – These are background services (printing, sound, scheduling, etc.) that either start up during boot or after you log into the desktop. Graphical server – This is the sub-system that displays the graphics on your monitor. It is commonly referred to as the X server or just X.
5. **Desktop environment** – This is the piece that the users actually interact with. There are many desktop environments to choose from (GNOME, Cinnamon, Mate, Pantheon, Enlightenment, KDE, Xfce, etc.). Each desktop environment includes built-in applications (such as file managers, configuration tools, web browsers, and games).
6. **Applications** – Desktop environments do not offer the full array of apps. Just like Windows and macOS, Linux offers thousands upon thousands of high-quality software titles that can be easily found and installed. Most modern Linux distributions (more on this below) include App Store-like tools that centralize and simplify application installation. For example, Ubuntu Linux has the Ubuntu Software Center (a rebrand of GNOME Software) which allows you to quickly search among the thousands of apps and install them from one centralized location.

Why use Linux?

This is the one question that most people ask. Why bother learning a completely different computing environment, when the operating system that ships with most desktops, laptops, and servers works just fine?

To answer that question, I would pose another question. Does that operating system you're currently using really work "just fine"? Or, do you find yourself battling obstacles like viruses, malware, slow downs, crashes, costly repairs, and licensing fees? If you struggle with the above, Linux might be the perfect platform for you. Linux has evolved into one of the most reliable computer ecosystems on the planet. Combine that reliability with zero cost of entry and you have the perfect solution for a desktop platform.

That's right, zero cost of entry… as in free. You can install Linux on as many computers as you like without paying a cent for software or server licensing.

Let's take a look at the cost of a Linux server in comparison to Windows Server 2016. The price of the Windows Server 2016 Standard edition is $882.00 USD (purchased directly from Microsoft). That doesn't include Client Access License (CALs) and licenses for other software you may need to run (such as a database, a web server, mail server, etc.). For example, a single user CAL, for Windows Server 2016, costs $38.00. If you need to add 10 users, for example, that's $388.00 more dollars for server software licensing. With the Linux server, it's all free and easy to install. In fact, installing a full-blown web server (that includes a database server), is just a few clicks or commands away (take a look at Easy LAMP Server Installation to get an idea how simple it can be).

If zero cost isn't enough to win you over–what about having an operating system that will work, trouble free, for as long as you use it? I've used Linux for nearly 20 years (as both a desktop and server platform) and have not had any issues with ransomware, malware, or viruses. Linux is generally far less vulnerable to such attacks.

As for server reboots, they're only necessary if the kernel is updated. It is not out of the ordinary for a Linux server to go years without being rebooted. If you follow the regular recommended updates, stability and dependability are practically assured.

Open source

Linux is also distributed under an open source license. Open source follows these key tenets:

- The freedom to run the program, for any purpose.
- The freedom to study how the program works, and change it to make it do what you wish.
- The freedom to redistribute copies so you can help your neighbor.
- The freedom to distribute copies of your modified versions to others.

These points are crucial to understanding the community that works together to create the Linux platform. Without a doubt, Linux is an operating system that is "by the people, for the people". These tenets are also a main factor in why many people choose Linux. It's about freedom and freedom of use and freedom of choice.

What is a "distribution?"

Linux has a number of different versions to suit any type of user. From new users to hard-core users, you'll find a "flavor" of Linux to match your needs. These versions are called distributions (or, in the short form, "distros"). Nearly every distribution of Linux can be downloaded for free, burned onto disk (or USB thumb drive), and installed (on as many machines as you like).

Popular Linux distributions include:

- LINUX MINT
- MANJARO
- DEBIAN
- UBUNTU

- ANTERGOS
- SOLUS
- FEDORA
- ELEMENTARY OS
- OPENSUSE

Each distribution has a different take on the desktop. Some opt for very modern user interfaces (such as GNOME and Elementary OS's Pantheon), whereas others stick with a more traditional desktop environment (openSUSE uses KDE).

And don't think the server has been left behind. For this arena, you can turn to:

- Red Hat Enterprise Linux
- Ubuntu Server
- Centos
- SUSE Enterprise Linux

Some of the above server distributions are free (such as Ubuntu Server and CentOS) and some have an associated price (such as Red Hat Enterprise Linux and SUSE Enterprise Linux). Those with an associated price also include support.

Which distribution is right for you?

Which distribution you use will depend on the answer to three simple questions:

- How skilled of a computer user are you?
- Do you prefer a modern or a standard desktop interface?
- Server or desktop?

If your computer skills are fairly basic, you'll want to stick with a newbie-friendly distribution such as Linux Mint, Ubuntu (Figure 3), Elementary OS or Deepin. If your skill set extends into the above-average range, you could go with a distribution like Debian or Fedora. If, however, you've pretty much mastered the craft of computer and system administration, use a distribution like Gentoo. If you really want a challenge, you can build your very own Linux distribution, with the help of Linux From Scratch.

If you're looking for a server-only distribution, you will also want to decide if you need a desktop interface, or if you want to do this via command-line only. The Ubuntu Server does not install a GUI interface. This means two things your server won't be bogged down loading graphics and you'll need to have a solid understanding of the Linux command line. However, you can install a GUI package on top of the Ubuntu Server with a single command like sudo apt-get install ubuntu-desktop. System administrators will also want to view a distribution with regards to features. Do you want a server-specific distribution that will offer you, out of the box, everything you need for your server? If so, CentOS might be the best choice. Or, do you want to take a desktop distribution and add the pieces as you need them? If so, Debian or Ubuntu Linux might serve you well.

Linux Distribution Comparison

Criteria	Ubuntu	Debian	Fedora	Gentoo
Package Management	APT (deb)	APT (deb)	DNF (rpm)	Portage (source-based)
Release Cycle	6 months (LTS: 2 years)	~2 years (stable)	~6 months	Rolling release
Target Audience	Beginners to intermediate	Intermediate to advanced	Intermediate to advanced	Advanced users
System Reqs	Moderate	Low to moderate	Moderate to high	Variable
Customization	Limited	Moderate	Moderate	Unlimited
Installation Difficulty	Very Easy	Easy	Easy	Hard
Default Desktop	GNOME	None	GNOME	None

Installing Linux

For many people, the idea of installing an operating system might seem like a very daunting task. Believe it or not, Linux offers one of the easiest installations of all operating systems. In fact, most versions of Linux offer what is called a Live distribution, which means you run the operating system from either a CD/DVD or USB flash drive without making any changes to your hard drive. You get the full functionality without having to commit to the installation. Once you've tried it out, and decided you wanted to use it, you simply double-click the "Install" icon and walk through the simple installation wizard.

Typically, the installation wizards walk you through the process with the following steps (We'll illustrate the installation of Ubuntu Linux):

- Preparation: Make sure your machine meets the requirements for installation. This also may ask you if you want to install third-party software (such as plugins for MP3 playback, video codecs, and more).
- Wireless setup (if necessary): If you are using a laptop (or machine with wireless), you'll need to connect to the network, in order to download third-party software and updates.
- Hard drive allocation (Figure 4): This step allows you to select how you want the operating system to be installed. Are you going to install Linux alongside another operating system (called "dual booting"), use the entire hard drive, upgrade an existing Linux installation, or install over an existing version of Linux.
- Location: Select your location from the map.
- Keyboard layout: Select the keyboard for your system.
- User setup: Set up your username and password.

That's it. Once the system has completed the installation, reboot and you're ready to go. For a more in-depth guide to installing Linux, take a look at "How to Install and Try Linux the Absolutely Easiest and Safest Way" or download the Linux Foundation's PDF guide for Linux installation.

Installing software on Linux

Just as the operating system itself is easy to install, so too are applications. Most modern Linux distributions include what most would consider an app store. This is a centralized location where software can be searched and installed. Ubuntu Linux (and many other distributions) rely on GNOME Software, Elementary OS has the AppCenter, Deepin has the Deepin Software Center, openSUSE has their AppStore, and some distributions rely on Synaptic.

Regardless of the name, each of these tools do the same thing: a central place to search for and install Linux software. Of course, these pieces of software depend upon the presence of a GUI. For GUI-less servers, you will have to depend upon the command-line interface for installation.

Let's look at two different tools to illustrate how easy even the command line installation can be. Our examples are for Debian-based distributions and Fedora-based distributions. The Debian-based distros will use the apt-get tool for installing software and Fedora-based distros will require the use of the yum tool. Both work very similarly. We'll illustrate using the apt-get command. Let's say you want to install the wget tool (which is a handy tool used to download files from the command line). To install this using apt-get, the command would like like this:

```
sudo apt-get install wget
```

The sudo command is added because you need super user privileges in order to install software. Similarly, to install the same software on a Fedora-based distribution, you would first su to the super user (literally issue the command su and enter the root password), and issue this command:

```
yum install wget
```

That's all there is to installing software on a Linux machine. It's not nearly as challenging as you might think. Still in doubt? Recall the Easy Lamp Server Installation from earlier. With a single command:

```
sudo taskel
```

You can install a complete LAMP (Linux Apache MySQL PHP) server on either a server or desktop distribution. It really is that easy.

More resources

If you’re looking for one of the most reliable, secure, and dependable platforms for both the desktop and the server, look no further than one of the many Linux distributions. With Linux you can assure your desktops will be free of trouble, your servers up, and your support requests minimal.

For more information to help guide you through your lifetime with Linux, check out the following resources:

Linux.com: Everything you need to know about Linux (news, tutorials and more) Howtoforge: Linux tutorials Linux Documentation Project: How-tos, guides, and FAQs Linux Knowledge Base and Tutorial: Plenty of tutorials and in-depth guides LWN.net: Linux kernel news and more

Document 2

Python

Learn the Basics

Hello, World!

To print a string in Python 3, just write:

```
print("Hello, World!")
```

Variables and Types

There are two types of numbers in python, integers (whole numbers) and floating point numbers (decimals).

To define an integer, use the following syntax:

```
myint = 7
print(myint)
```

To define a floating point number you may use one of the following:

```
myfloat = 7.0
print(myfloat)
myfloat = float(7.0)
print(myfloat)
```

Strings are defined with single or double quotes, double quotes making it easier to include apostrophes.

```
mystring = 'hello'
print(mystring)
mystring = "hello, who's there"
print(mystring)
```

Simple operators can be executed on numbers and strings:

```
one = 1
two = 2
three = one + two
print(three)

hello = "hello"
world = "world"
helloworld = hello + " " + world
print(helloworld)
```

Assignments can be done on more than one variable "simultaneously" on the same time like this:

```
a, b = 3, 4
print(a, b)
```

Mixing operators between numbers and strings is not supported.

Lists

Lists are very similar to arrays. They can contain any type of variable, and they can contain as many variables as you wish. Lists can also be iterated over in a very simple manner. Here is an example of how to build a list:

```
mylist = []
mylist.append(1)
mylist.append(2)
mylist.append(3)
print(mylist[0]) # prints 1
print(mylist[1]) # prints 2
print(mylist[2]) # prints 3

# prints out 1,2,3
for x in mylist:
    print(x)
```

Basic Operators

Just as any other programming languages, the addition, subtraction, multiplication, and division operators can be used with numbers. Another operator available is the modulo (%) operator, which returns the integer remainder of the division. dividend % divisor = remainder:

```
number = 1 + 2 * 3 / 4.0
print(number)

# Modulo
remainder = 11 % 3
print(remainder)

# Using two multiplication symbols makes a power relationship.

squared = 7 ** 2
cubed = 2 ** 3
print(squared)
print(cubed)
```

Python supports concatenating strings using the addition operator, as well as multiplying strings to create a repeating sequence:

```
helloworld = "hello" + " " + "world"
print(helloworld)

lotsofhellos = "hello" * 10
print(lotsofhellos)
```

String Formatting

The "%" operator is used to format a set of variables enclosed in a "tuple" (a fixed size list), together with a format string, which contains normal text together with "argument specifiers", special symbols like "%s" and "%d".

```
# This prints out "Hello, John!"
name = "John"
print("Hello, %s!" % name)
# This prints out "John is 23 years old."
name = "John"
age = 23
print("%s is %d years old." % (name, age))
# This prints out: A list: [1, 2, 3]
mylist = [1,2,3]
print("A list: %s" % mylist)
```

Basic argument specifiers:

`%s - String (or any object with a string representation, like numbers)`

`%d - Integers`

`%f - Floating point numbers`

`%.<number of digits>f - Floating point numbers with a fixed amount of digits to the right of the dot.`

`%x/%X - Integers in hex representation (lowercase/uppercase)`

Basic String Operations

String Creation

```
text = "Hello world!"
text2 = 'Hello world!'
```

Basic Operations

```
len(text)               # Length: 12
text.index("o")         # First occurrence position: 4
text.count("l")         # Count occurrences: 3
text[3:7]               # Slice from index 3 to 6: "lo w"
text[::-1]              # Reverse string
text.upper()            # Uppercase
text.lower()            # Lowercase
text.startswith("H")    # True
text.endswith("!")      # True
text.split(" ")         # Split into list: ["Hello", "world!"]
```

Conditions

The syntax for comparisons are:

```
x == 2    # Equal to
x != 3    # Not equal to
x < 3     # Less than
x >= 2    # Greater than or equal to
```

Boolean operators have two states, True or False.

```
# Logical operators
name == "John" and age == 23
name == "John" or name == "Rick"
name in ["John", "Rick"]
not False

# Identity vs equality
x is y     # Same object
x == y     # Same value
```

If statements are conditional, only allowing the block of code to be executed if the condition is True.

```
if condition:
    # do something
elif another_condition:
    # do something else
else:
    # default action
    pass
```

Loops

For loops iterate over a sequence:

```
# Iterate over list
for item in [1, 2, 3]:
    print(item)

# Range iterations
for i in range(5):        # 0 to 4
for i in range(3, 6):     # 3 to 5
for i in range(0, 10, 2): # 0,2,4,6,8
```

While loops repeat as long as a boolean condition is met.

```
count = 0
while count < 5:
    print(count)
    count += 1
```

To control loops, you may use `break` to exit the loop or `continue` to skip to the next iteration.

Document 3

Tables

Table 1

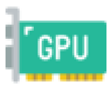
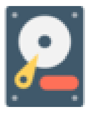
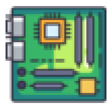
ID	Name	Email	Investments
231	AlBert Master	albert.master@gmail.com	Bonds
210	Alfred Alan	aalan@gmail.com	Stocks
256	Alison Smart	asmart@biztalk.com	Residential Property
211	Ally Emery	allye@easymail.com	Stocks
248	Andrew Phips	andyp@mycorp.com	Stocks
234	Andy Mitchel	andym@hotmail.com	Stocks
226	Angus Robins	arobins@robins.com	Bonds
241	Ann Melan	ann_melan@iinet.com	Residential Property
225	Ben Bessel	benb@hotmail.com	Stocks
235	Bensen Romanolf	benr@albert.net	Bonds

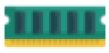
Table 2

Grade Distribution

Letter Grade	Percentage
A	90 - 100%
B	80 - 89.99%
C	70 - 79.99%
D	60 - 69.99%
F	< 60%

Table 3

Part	Name
	Graphics Card
	Hard Drive
	Motherboard

Part	Name
	RAM