UNIX & LINUX

What is the exact difference between a 'terminal', a 'shell', a 'tty' and a 'console'?

Asked 8 years, 11 months ago Active 4 months ago Viewed 333k times

1240

871

I think these terms almost refer to the same thing, when used loosely:

- terminal
- shell
- tty
- console

What exactly does each of these terms refer to?

shell terminal console tty terminology

edited Nov 7 '15 at 3:55
Navin
179 1 12

asked Nov 16 '10 at 20:06
Lazer
13.7k 20 63 73

86 The TTY demystified – firo Mar 7 '13 at 7:54

28 I'd like to add 'command line' to that :-) – teeks99 Sep 7 '14 at 13:32

1 The command line is simply the language used to send commands to the command-line interpreter running in a shell from the terminal/terminal emulator. – Marty Fried Sep 7 '14 at 18:03

1 The **teletypewriter (TTY)** was first put in operation and exhibited at the Mechanics Institute in New York in **1844.** en.wikipedia.org/wiki/Teleprinter – Serge Stroobandt Dec 10 '15 at 20:28

Two more useful links - feyrer.de/NetBSD/ttys.html and quora.com/… – Nishant Dec 25 '18 at 19:58

1201

A terminal is at the end of an electric wire, a shell is the home of a turtle, tty is a strange abbreviation and a console is a kind of cabinet.

Well, etymologically speaking, anyway.

In unix terminology, the short answer is that

- terminal = tty = text input/output environment
- console = physical terminal
- shell = command line interpreter

Console, terminal and tty are closely related. Originally, they meant a piece of equipment through which you could interact with a computer: in the early days of unix, that meant a teleprinter-style device resembling a typewriter, sometimes called a teletypewriter, or “tty” in shorthand. The name “terminal” came from the electronic point of view, and the name “console” from the furniture point of view. Very early in unix history, electronic keyboards and displays became the norm for terminals.

In unix terminology, a **tty** is a particular kind of device file which implements a number of additional commands (ioctls) beyond read and write. In its most common meaning, **terminal** is synonymous with tty. Some ttys are provided by the kernel on behalf of a hardware device, for example with the input coming from the keyboard and the output going to a text mode screen, or with the input and output transmitted over a serial line. Other ttys, sometimes called **pseudo-ttys**, are provided (through a thin kernel layer) by programs called **terminal emulators**, such as Xterm (running in the X Window System), Screen (which provides a layer of isolation between a program and another terminal), Ssh (which connects a terminal on one machine with programs on another machine), Expect (for scripting terminal interactions), etc.

The word terminal can also have a more traditional meaning of a device through which one interacts with a computer, typically with a keyboard and display. For example an X terminal is a kind of thin client, a special-purpose computer whose only purpose is to drive a keyboard, display, mouse and occasionally other human interaction peripherals, with the actual applications running on another, more powerful computer.

A **console** is generally a terminal in the physical sense that is by some definition the primary terminal directly connected to a machine. The console appears to the operating system as a (kernel-implemented) tty. On some systems, such as Linux and FreeBSD, the console appears as several ttys (special key combinations switch between these ttys); just to confuse matters, the name given to each particular tty can be “console”, ”virtual console”, ”virtual terminal”, and other variations.

A **shell** is the primary interface that users see when they log in, whose primary purpose is to start other programs. (I don't know whether the original metaphor is that the shell is the home environment for the user, or that the shell is what other programs are

running in.)

In unix circles, **shell** has specialized to mean a command-line shell, centered around entering the name of the application one wants to start, followed by the names of files or other objects that the application should act on, and pressing the Enter key. Other types of environments don't use the word “shell”; for example, window systems involve “window managers” and “desktop environments”, not a “shell”.

There are many different unix shells. Popular shells for interactive use include Bash (the default on most Linux installations), zsh (which emphasizes power and customizability) and fish (which emphasizes simplicity).

Command-line shells include flow control constructs to combine commands. In addition to typing commands at an interactive prompt, users can write scripts. The most common shells have a common syntax based on the Bourne_shell. When discussing “**shell programming**”, the shell is almost always implied to be a Bourne-style shell. Some shells that are often used for scripting but lack advanced interactive features include the Korn shell (ksh) and many ash variants. Pretty much any Unix-like system has a Bourne-style shell installed as `/bin/sh`, usually ash, ksh or bash.

In unix system administration, a user's **shell** is the program that is invoked when they log in. Normal user accounts have a command-line shell, but users with restricted access may have a restricted shell or some other specific command (e.g. for file-transfer-only accounts).

The division of labor between the terminal and the shell is not completely obvious. Here are their main tasks.

- Input: the terminal converts keys into control sequences (e.g. Left → `\e[D`). The shell converts control sequences into commands (e.g. `\e[D` → `backward-char`).
- Line editing, input history and completion are provided by the shell.
  - The terminal may provide its own line editing, history and completion instead, and only send a line to the shell when it's ready to be executed. The only common terminal that operates in this way is `M-x shell` in Emacs.
- Output: the shell emits instructions such as “display `foo`”, “switch the foreground color to green”, “move the cursor to the next line”, etc. The terminal acts on these instructions.
- The prompt is purely a shell concept.
- The shell never sees the output of the commands it runs (unless redirected). Output history (scrollback) is purely a terminal

Shift + Insert). The shell may have its own internal copy-paste mechanism as well (e.g. Meta + W and Ctrl + Y).

- Job control (launching programs in the background and managing them) is mostly performed by the shell. However, it's the terminal that handles key combinations like Ctrl + C to kill the foreground job and Ctrl + Z to suspend it.

edited Nov 16 '17 at 14:06

answered Nov 16 '10 at 22:31

Gilles 'SO- stop being evil'

584k 142 1209 1721

50 Only quibble: I would say that both kinds of ttys are “provided by” the kernel. The difference I would emphasize is that hardware ttys (e.g. serial lines and the built-in, text-mode console) have one end connected to hardware and one end connected to software (e.g. login programs and/or shells) while pseudo-ttys have both ends connected to software (e.g. a terminal emulator on one end and shell on the other). – Chris Johnsen Nov 17 '10 at 4:04

12 @phunehehe: Right, that's a different meaning of “shell”, in common use in operating system design: the shell is the outer part of the kernel. It's not unix terminology: Unix kernels don't tend to have a component that one could call a shell. – Gilles 'SO- stop being evil' Nov 17 '10 at 19:27

24 This is the image in my mind for the shell metaphor. – ændrük Dec 7 '10 at 19:00

12 There is also another meaning of "console" under Linux. The console (there is only one) is where `printk` of sufficient priority goes (e.g., kernel panics). It is set by passing `console=DEVICE,...` on the kernel command line (e.g., `console=ttyS0,115200` for a the first serial port, at 115,200 bps). Normally it defaults to the virtual-terminal, but that can be changed when the kernel is compiled. – derobert Aug 29 '11 at 21:12

19 “…the terminal…handles key combinations like Ctrl+C to kill the foreground job and Ctrl+Z to suspend it” Not quite: the terminal still merely sends control characters, it’s the tty device that decides how to handle them, and it’s configurable. By default the tty device converts the control characters into signals sent to the shell (and other processes). – Chris Page Mar 10 '12 at 20:34

194

A **terminal** or a **console** is a piece of hardware, using which a user can interact with a host. Basically a keyboard coupled with a text screen.
Nowadays nearly all terminals and consoles represent "virtual" ones.

The file that represents a terminal is, traditionally, called a **tty** file. If you look under the "/dev" directory of a UNIX system, you'll find a lot of **tty** files connected to virtual consoles (e.g. tty1 on linux), virtual terminals (e.g. pts/0) or physically connected hardware (e.g. ttyS0 is the physical serial terminal, if any, attached on first serial port of the host)

point to access a system for maintenance and some special operation can be done only from a console (e.g. see `single user mode`). A **terminal** can be, and usually is, a remote piece of hardware.

Last, but not the least, a **shell** is a special program that interacts with a user through a **controlling tty** and offers, to the user, the way of launching other programs (e.g. bash, csh, tcsh).

A **terminal emulator** is a program that emulates a physical terminal (e.g. xterm, gnome-terminal, minicom).

So when you look to a "text window" on your linux system (under X11) you are looking to: a *terminal emulator*, connected to a *virtual terminal*, identified by a *tty* file, inside which runs a *shell*.

edited Feb 21 '14 at 19:22

crisron
314 4 16

answered Nov 16 '10 at 21:53

andcoz
13.5k 3 31 41

2 Any desktop computer has system console (in my 2015 or poster's 2010, don't matter). As it was correctly stated, it's a piece of hardware. But stating "Nowadays nearly all… consoles represent "virtual" ones" is nearly contradictory and obviously not good. – Incnis Mrsi Sep 6 '15 at 7:20

1 "A terminal or a console is a piece of hardware, using which a user can interact with a host. Basically a keyboard coupled with a text screen." awesome explication by its concrete aspect – Webwoman Sep 9 '18 at 11:45

@andcoz - What do you mean by "text window"? Isn't a terminal emulator a virtual terminal? If i run at the command `tty` in a terminal emulator such as KDE's Konsole, the output is /dev/pts/0. – Motivated Jan 16 at 6:43

1 @IncnisMrsi - Isn't a desktop computer the *console*? If not, what do you mean by it has a system console? – Motivated Jan 16 at 6:45

2 @Motivated `/dev/pts/0` is a tty file, an handler to a programmatic interface exposed by the kernel. Through this handler, a program (e.g. the shell) can interact with a terminal (real or virtual). A terminal emulator is a software that emulates a terminal. The emulator asks the kernel to create an handler to let programs to interact with itself (see `man openpty`). So information flows from the terminal (emulator) to the kernel tty handler, to the program (and vice versa). Programs and terminals do not talk each other directly but only through the tty file (the handler). – andcoz Jan 16 at 13:45

48

SHORT explanation:

The console is a terminal. A system has got one console and potentially multiple terminals. The console is typically the primary interface for managing a computer, eg while it is still booting up.

A terminal is a session which can receive and send input and output for command-line programs. The console is a special case of these.

The shell is a program which is used for controlling and running programs. It is often used interactively, via a terminal. Several Shell programs exist, Bash being arguably the most commonly used shell today. Other shells, in no particular order, includes Bourne Shell, C-shell, Dash, Tsch, Ksh, and the increasingly popular zsh. There are many more.

When you have a GUI, you can use a terminal program to draw a nice resizeable border, add scroll bars, and format the text, and so on, for a terminal session. Often these are called terminal emulators, and sometimes they can handle multiple sessions via a TAB concept. A Terminal Emulator often starts a Shell to allow you to interactively work on a command line.

edited Mar 13 at 5:33

Prajwal Dhatwalia
420 4 10

answered Mar 19 '13 at 9:22

Johan
2,932 2 19 29

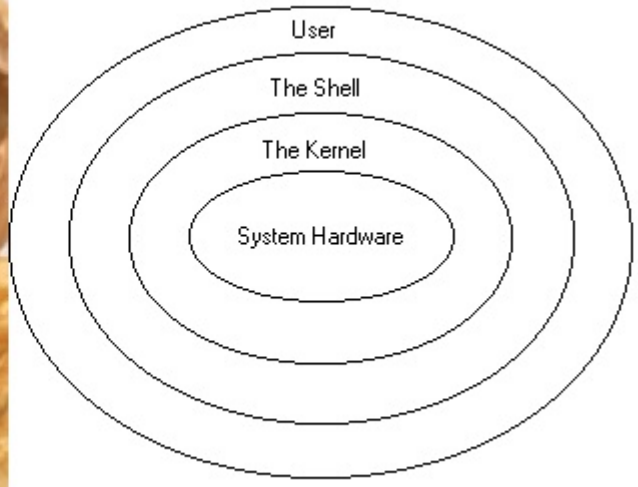
1 PTY is a pseudo TTY. TTY can be, but ins't essentially virtual (either pseudo) terminal. – Luciano Jun 18 '15 at 21:23

35

A **TTY** (i.e. **T**ele**TY**pewriter) is a special device that lets people who are deaf, hard of hearing, or speech-impaired use the telephone to communicate, by allowing them to type text messages. A TTY is required at both ends of the conversation in order to communicate.
OR
TTY is **terminal** which is used to type text message.

Shell :the outside protective covering part of a seed i.e. kernel.
OR

framework or exterior structure to central or essential part of a system.



Console means the keyboard and monitor physically attachements to a computer.

edited Dec 20 '15 at 1:53

answered Dec 20 '15 at 1:44

Premraj
1,346 1 13 20

28

There are already two great answers, but I'd like to add information about the phrase **"virtual terminal"**. Generally, it means something that provides appearance/functionality of a terminal, i. e. a terminal-emulator in broad sense. But in early days of Linux (1994–95) is was used synonymously with "virtual console" (several unrelated user interfaces), by some developers. This usage persists in documentation; two different terms were (and are) used to refer to tty1, tty2… thingies. Nowadays (since ≈ 1996) "virtual terminal" may also refer to pty -based terminal emulators.

engineers consistently use the word “consoles” to denote tty1, tty2… and used “vc_” prefix for them. For example, there is a `vc_allocate` function. On the other hand, developers of such user-space tools as `kbd` and `console-tools` used “virtual console” (VC) and “virtual terminal” (VT) interchangeably. I contacted Andries E. Brouwer and asked him to clarify terminology used by early

developers (1994–95). Andries kindly provided some answers. He states that VT and VC are synonymous and “indivisible” abbreviations. --> In general, a virtual console is a virtual terminal, but converse isn’t true. Those “virtual terminals” that are not virtual consoles are indeed *pseudoterminals* (as Andries states, these *are not VT*). Unlike virtual consoles, where the kernel provides terminal functionality for a console application, pseudoterminals use PTY “devices” to arrange communication between console applications and the terminal-making program that runs in userspace. Examples are X-based terminal emulators and `sshd`, that allocates a pseudotty for each login session. A pseudotty may not be called “console” – it’s a mistake.

edited Apr 13 '17 at 12:36
Community ♦
1

answered Sep 7 '15 at 13:14
Incnis Mrsi
1,446 11 23

14

- Terminal = An interface that provides a display for output and a key board for input to a shell session .
- Shell = Interpreter that executes commands typed as string
- Console: Actually two types of console we use
  - Physical console=The hardware display and keyboard used to interact with a system
  - Virtual console= One of multiple logical consoles that can each support an independent login session.
- tty(teletype ie terminal). = A terminal is a basically just a user interface device that uses text for input and output.message.

edited Jul 3 '17 at 9:34

answered Apr 12 '17 at 8:23
Rakib
1,224 1 10 14

What is a tty ? -- otherwise, your answer is the only one I read – loxaxs Jul 2 '17 at 21:39

You need to dive into history.

There were typewriter-like devices with paper and keyboard. They were called teletypes (which means "type remotely," since "tele"

Any computer need some way to report its status and errors (and, probably, accept commands). It is done through **console** which is almost always connected directly to the computer. So, there are 2 meanings for **console**: something that is used to report status and something that is connected directly.

UNIX is an interactive system: several users may connect to it and start applications. First computers used teletypes (**tty**) for that: each user had teletype connected to machine with serial line connection. Such teletype is called **terminal**. UNIX also got special subsystem to handle "users sitting behind terminals" which is also called **tty** because first terminals were teletypes. Each process could be connected to tty in Unix. That means there is a user somewhere sitting near terminal. See http://www.linusakesson.net/programming/tty/ for more info.

Users need some way to tell kernel to run application. **shell** (sh, bash, csh, ksh, etc.) is used for that. **shell** runs on **tty**, accepts commands from user and asks kernel to run some app.

But terminals are not always physically connected to the machine. There may be some application that "emulates" terminal accepting keystrokes from user and sending them somewhere (xterm and ssh are good examples). There is an API in Kernel called **pseudo terminal** for that. So your **tty** may really be connected to some application instead of real terminal. Xterm uses X11 to display text and ssh uses network connection for it.

IBM PC has keyboard and video card (they are also called **console** sometimes). Linux can do different things with it:

- Use it as "engine to report errors and status": Linux console. If you pass console=/dev/ttyS0 to kernel it will use something connected to COM1 as console, and if you do not it will use PC console.
- Use it to emulate terminal, so called **virtual terminal** (vty).

It also may stop emulating terminal on console and give it to some app. App may switch its video mode and use it exclusively (X11 or svgalib may do that).

So, here are modern meanings:

- terminal: Something with real user sitting behind it. Could be physical terminal (rare) or pseudo terminal (xterm, ssh) or virtual terminal (vty in Linux)
- shell: application (bash, tcsh, etc) that helps user to interact with system.
- tty: either terminal or kernel subsystem to support terminals.
- console: something where status and errors are reported (`/dev/console`) or physical keyboard and video display connected to computer.

7

Here is the short answer -

Kernel - the innermost part of any modern operating system which directly talks to actual hardware.

Shell - wrapper around the actual Kernel. Whenever we run command, we actually talk to shell which in turn invokes appropriate Kernel instructions. Apart from this, the shell is capable of performing some other stuffs like finding appropriate program while having commands, some file name short hand, piping commands etc.

Terminal - in the era of earlier computing, computers (known as Mainframe) were giant. So, it was easy to have a single processing unit and connect it from many places. Terminal is the actual hardware with keyboard and output devices connected to mainframe.

Console - Special type of terminal which is directly connected to Mainframe for the purpose of OS Administration.

tty - TeleTypewriter used to send and receive data to and from Mainframe. Used before Video Terminals were available. But conventionally it has been still named as tty. Even the coommand `stty`

The long detailed answer is here - Terminal, Console, Shell, Kernel, Commands - Different parts of a Computer

edited Jul 7 '17 at 18:01

answered Jul 7 '17 at 16:47

Palash Kanti Kundu
109 1 6

thanks but basically if the terminal exist, why exist still tty also in ubuntu system for personal computers, accessible with alt + f-1/6 please, I can't figure out their utility above the fact they can be accessed without graphics system usage if I have well understood – Webwoman Sep 9 '18 at 11:59

1 @Webwoman - I have my system configured to only allow access to the `root` account through `sudo` or through a console login. Consoles often have a special place privilege-wise as someone who has access to them necessarily has physical access to the computer they communicate with. They are also the access method of last resort. If the OS is in a partially broken state (like the ethernet driver is broken) you can still access the console. It's the one human interface device that should ALWAYS be available, no matter what state the system is in. – Omnifarious Jul 12 at 21:35

@Omnifarious thanks for your answer "Consoles often have a special place privilege-wise as someone who has access to them necessarily has physical access to the computer they communicate with" you meant TTY often have a special place privilege-wise ? – Webwoman Jul 13 at 1:41

@Webwoman - Nope, console. A TTY is frequently not directly connected to a computer. And a console frequently isn't a TTY in the traditional sense. When I went to the U of MN in the late 80s, there was a campus-wide specialized network that I don't was running the Internet Protocol (aka IP) for connecting random TTYs to random computers. IBM mainframes have a similar thing going on and it was one the big reasons IBM created SNA. And a normal PC, even when it's only showing text, is very unlike a TTY in many ways, so a PC console isn't really a TTY exactly. – Omnifarious Jul 13 at 3:21

6

Apart from the accepted answer and The TTY demystified article, I really loved reading these two articles:

This one is based on NetBSD.

> Back in the stone ages of Unix, computer systems consisted of a mainframe, a big box of blinking lights which had memory, mass storage and computing units, and that run processes started by users or operators. As the hardware was very expensive, the systems were used as true multiuser systems, with many people interacting with the system at the same time. What it usually didn't have - unlike today's Unix workstations - was a fixed monitor and keyboard. Instead, issuing commands to the machine and retrieving output was done over serial lines, using teletypers first, and CRT (cathode ray tube) terminals later. Teletypers - that's where the "ttys" in Unix come from - are electronic typewriters that send keys pressed over the serial line to the host, and replies were sent back to the teletyper char by char over the serial line, with the built-in printer putting the reply on paper, much like a typewriter.

This one is based on Linux.

> Terminals are devices that provide enhanced input/output capabilities beyond what could be achieved with only regular files, pipes, and sockets. These features are designed to make it easier for humans to interact with computers, and are useless for programs trying to talk to each other.

answered Dec 25 '18 at 20:10

Nishant
269 2 11

1

Let me take a crack at this... I will use Unix and Linux more or less synonymously in this. If I'm referring to something historical that predates the existence of Linux, I will usually write "Unix", and if I'm talking about something more recent, or something specific to the Linux flavor of Unix, I will usually write "Linux".

Shell

The only thing in your list that is a discrete concept that has no overlap with the others is the 'shell'. The shell is a program who's purpose is to communicate with a user and carry out operating system operations on their behalf.

In Unix, if someone calls something a 'shell' they almost certainly mean some form of command line interface as I just described. And it is very odd in the Unix world to refer to anything as a 'shell' if it isn't communicating to a user using the tty model I describe further on.

TTY

This is a confusing one because it can refer to a few different kinds of things.

In Linux, there is a kind of device called a 'tty'. It is an abstract device that is expected to be used for bi-directional communication with something that either is a user, or is taking input from a user in some way. Sometimes that abstract device may correspond directly to some physical device. Sometimes it may be a program that is presenting someone with a window in which the communication appears and into which the user can type.

But, the reason this abstract device exists and the reason it is called a 'tty' is that 'tty' is short for 'teletype', which was an actual physical device that had a printer that printed on paper combined with a keyboard. The model the the abstract 'tty' device presents to programs that are using it is basically that there is a teletype on the other end. You send it characters and those characters appear on the teletype. When you read characters from it, those characters represent keys that were typed on a keyboard.

The old paper-printer based ttys were quickly supplanted with video ttys. On those, of course, there is no roll of paper. And, in fact, it is possible overwrite any character on the screen. But, rather than present some kind of abstract 'screen' interface to programs, programs are instead expected to send special streams of characters called escape sequences that accomplish a variety of tasks. Usually there is an abstract thing called a 'cursor' that can be moved around the screen, and any character sent will replace whatever is at the cursor and the cursor will move one character further on. Often you can change the color of a character that's about to be printed with escape sequences as well.

There are 'glass ttys' that do not follow this model and consequently are handled poorly in the Unix world. The IBM 3270 family of video terminals fall into this category.

What Linux/Unix people typically call a 'shell window' is an emulation of a glass tty using a graphical user interface. Internally, programs running inside of a shell window are talking to a virtual tty device that is sometimes called a pseudo-tty or pseudo-terminal (aka a pty).

Terminal

A terminal is just a place where computer and human are supposed to interface. Terminals may be completely graphical and not follow the tty model in any way, even though a program may use their capabilities to emulate this. All actual physical ttys (glass or otherwise) are terminals.

Console

operating system is running on.

In Linux, the console is virtualized in a small way which allows you to use a special keystroke to switch between the virtual consoles. But this virtualization is done with a real piece of hardware by software in the kernel.

There are ways to use Linux through what's called a 'serial console' which is a console that's attached to the computer through a serial port like a USB port (or, on some very small and/or very old computers, an RS-232 port of some kind) and follows the old teletype model in a fairly strict way.

Again, the idea is that this console is connected in a direct physical way to the computer instead of through some sort of network that might allow anybody to connect.

answered Jul 13 at 3:50

Omnifarious

996 9 18

---

protected by Anthon Jul 7 '17 at 22:02

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?