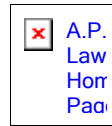


<a href="#">Search</a>
<a href="#">FAQ</a>
<a href="#">New to Sco</a>
<a href="#">New to Unix</a>
<a href="#">New Here</a>
<a href="#">SCO News</a>
<a href="#">Unix Articles</a>
<a href="#">Unixware</a>
<a href="#">Linux</a>
<a href="#">Security</a>
<a href="#">Code</a>
<a href="#">Tests</a>
<a href="#">Links</a>
<a href="#">Books</a>
<a href="#">Reviews</a>
<a href="#">Opinion</a>
<a href="#">Find a Consultant</a>
<a href="#">Find a Job</a>
<a href="#">Site Forum</a>
<a href="#">Donations</a>
<a href="#">Contact Info</a>
<a href="#">Services</a>
<a href="#">Sales</a>
<a href="#">Consulting Rates</a>
<a href="#">Advertising Rates</a>
<a href="#">Site Map</a>



## A.P. Lawrence Home

Over 23,500 Unix/Linux users visited here last month (103,200+ page views)

Note: I am on vacation until July 17th 2001

 <http://www.sysintegrators.com>

# Termcap and Terminfo

[More Articles](#)

See also: [Terminals](#)

Most Unix applications are written so that they can be run on different terminals. The "terminal" may be simply an emulator like Procomm or Tinyterm or IceTen running on a Windows machine, but the running program doesn't know that: if it has been told to use "vt100" emulation, it doesn't matter to it whether you are using Procomm, a Wyse 50 terminal set to VT100 mode, or a real DEC VT100 terminal: the program will expect certain specific capabilities and responses from the "terminal". Usually, the TERM variable determines the expected terminal (though not always- see below). That TERM variable may be set automatically, or you may have to set it yourself. For example, you may see something like this when you login:

```
TERM = (ansi)
```

What's happening there is that the "tset" program in your .profile is telling you that it will assume you are an "ansi" terminal unless you tell it differently. Ordinarily you would just hit ENTER to accept "ansi", but you could type "wyse60" or "vt100" or any other valid terminal name if you were not, in fact, using an ansi terminal.

The "tset" line is in your .profile and looks like this:

```
eval `tset -m ansi:${TERM:-ansi} -m :\?${TERM:-ansi} -e -r -s -Q`
```

If your device line is listed in /etc/ttytype, or if a telnet session has exported the TERM variable, then tset "knows" what your terminal is.

/etc/ttytype is just an ascii file that lists terminal ports and says what type of terminal is connected. Usually dialup terminals will say "dialup" or "unknown" so that tset won't recognize them and you get a chance to say what you have. However, if you are always using the same

terminal type, change the appropriate line to match that.

Local terminals should be set to match what you have connected- if they are wy60's, then your /etc/ttytype might look like:

```
...  
wy60 ttya01  
wy60 ttya02  
...
```

and so on.

However, it needs to be told what to do about what it knows. For example, if you logged in from a vt100 terminal using the .profile that has the above tset line in it, tset recognizes that you are not using an "ansi" terminal, so it stops and gives you a

```
TERM = (vt100)
```

prompt, because it hasn't been told what to do with "vt100". You can fix that by modifying the line:

```
eval `tset -m vt100:vt100 -m ansi:${TERM:-ansi} -m :\?${TERM:-ansi} -e -r -s -Q`
```

which would now just continue on without stopping. See **man tset** for more information.

Note that up to this point, the system has no idea what kind of terminal you are running. Therefore, it should be plain that the "login:" prompt does not depend on any specific terminal setting. I sometimes see people changing their terminals emulation in order to "fix" a terminal that isn't displaying a login. This is useless: that prompt has absolutely nothing to do with emulation.

Some very badly written programs only support specific terminals- perhaps one or two at best. This usually indicates that the programmer did not understand the use of termcap and terminfo. However, in some rare cases, such restrictions are necessary: a program may need capabilities and features that only certain terminals can provide. For example, it may require the ability to switch to a scan-code keyboard mode, which is a feature that only a small number of terminals support. Or, the program may need a large number of function keys which aren't necessarily present on all terminals.

The expectations of the program can sometimes be wrong. For example, I often see programs assume a 25 line display. That's fairly benign, and is usually easily corrected by setting the emulation or the actual terminal to provide 25 lines, but there are more disruptive assumptions that can be made. Programmers sometimes test their work on emulators that aren't necessarily doing a "perfect" emulation; that behave slightly differently than an actual terminal would, or differently than another emulator would. That can cause subtle or even extreme problems if you aren't using the same product the programmer tested on. Sometimes the programmer doesn't test at all; assumptions are made from available documentation, and either the assumptions or the documentation can be incorrect. Or, if the programmer does get everything correct, it may be **your** emulation program that is at fault.

Most programs will just refuse to run if the terminal you use doesn't have the features it wants or if it doesn't know whether the features are present. Some programs, however, assume a "default" terminal, and that's very likely to be a VT100. The advent of the "scoansi" terminal type has brought that behaviour out in the programs that do this: they understand "ansi", but if TERM is set to "scoansi", they'll fall back to a simple VT100 emulation. This means that a lot of things will work correctly (VT100, ansi and scoansi are all very similar), but things like function keys, page-up or page-down and the like will fail. Often a simple solution is to add:

```
[ $TERM = "scoansi" ] && TERM="ansi"
```

to the top of whatever script starts the program. That line tests to see if TERM currently is "scoansi" and, if it is, changes it to "ansi". Note that this change would only affect the programs run from the script that includes this: other programs would still see "scoansi".

Note that nothing really changes with that. Your terminal does not change its personality; it won't work differently- all you have done is tell a particular program that it is running on an "ansi" terminal. If it understands that emulation, your "scoansi" terminal works.

Understanding why that is so is crucial to understanding terminals. Telling a program that your terminal is ansi, or vt100, or whatever is simply informative: it's very much like telling someone that another person speaks French rather than English. If you want to communicate with a French speaking person, you don't use English, and if you want to communicate with a Wyse 60 terminal you don't treat it like a Dec VT100. That's because terminals have commands to do things like like "move the cursor to the third line of the screen and erase that line". These commands are generally short sequences of control characters, letters and numbers, but the commands for a VT100 are different than they are for a Wyse 60. So if your terminal speaks VT100, but your program sends it Wyse 60 commands, the result is a mess. Terminals also have different ideas about keys: an "A" key always sends the same codes for "A" no matter what the emulation is (except "scancode" terminals), but function keys, page up, page down, cursor keys, etc. will be different. For example, the "scoansi" terminal sends Escape followed by { and then M when you press F1. A Wyse60 terminal would send a CTRL-A, a "@" and then a CR (carriage return) instead. Obviously a program that wants to interpret F1 needs to know what kind of terminal you are using.

So both the program and your terminal have to agree on their emulation. But if you have agreement, where does the program get the information it needs to know how a VT100 works? How does it know what CTRL-A@  
 means? There are a few possibilities. It could maintain its own internal tables of terminals. As I noted above, some badly written programs do this: usually they've been ported from some other OS like the ancient multi-user DOS systems where such home-grown tables were necessary. Or, if it has been written by a Unix-aware programmer, it can use termcap or terminfo. The choice of method is up to the programmer: you'll be telling the program what your terminal is by setting your TERM variable (manually or with tset), but the programmer can use either termcap or terminfo to get the specific information needed. See [Termcap and Terminfo](#) for a book that deals with these at great length (if you are programming for Unix systems you'll want this), but if you are just interested in general problem solving, you only need to know that these two methods are possible and where to look to find the tables used.

Unfortunately, it gets more difficult when you move to different OS's. For example, SCO and Linux consoles are radically different- SCO maps the console Backspace key to be CTRL-H,

and Linux maps it to be the same as what SCO maps it DEL key to. So if you telnet from a SCO console to Linux, your Backspace key is seen as the "intr" key. You can change your Linux settings with stty, of course.

A post by Frank da Cruz sums some of this up:

```
Path: news.randori.com!hermes.visi.com!news-out.visi.com!news.maxwell.syr.edu!new
From: fdc@watsun.cc.columbia.edu (Frank da Cruz)
Newsgroups: comp.os.linux.misc,comp.unix.sco.misc
Subject: Re: Linux and SCO
Date: 29 Apr 2000 16:56:28 GMT
Organization: Columbia University
Lines: 66
Message-ID: <8ef47s$qtas$1@newsmaster.cc.columbia.edu>
References: <Pine.LNX.4.21.0004270901200.840-100000@telemachus> <8ecroc$26o$1@per
NNTP-Posting-Host: watsun.cc.columbia.edu
X-Trace: newsmaster.cc.columbia.edu 957027388 27562 128.59.39.2 (29 Apr 2000 16:5
X-Complaints-To: postmaster@columbia.edu
NNTP-Posting-Date: 29 Apr 2000 16:56:28 GMT
Xref: news.randori.com comp.os.linux.misc:203620 comp.unix.sco.misc:59165
X-Mozilla-Status: 8010
X-Mozilla-Status2: 00000000
```

```
In article <Fts9Hz.4s1@wjv.com.remove>,
Bill Vermillion <bill@wjv.com.REMOVE> wrote:
: In article <390ADD9F.7067D850@aplawrence.com>,
: Tony Lawrence <tony@aplawrence.com> wrote:
: >Linus Torvalds wrote:
: >> T.E.Dickey <dickey@shell.clark.net> wrote:
: ...
: I think it would be better to leave the VT100 emulation alone and
: build an alternate emulator than to muck with the VT100 'standard'.
:
```

And your wish came true. PC-based Unixes like Linux and SCO each have their own unique console terminal definition that is definitely not VT100, although it shares the VT100's ANSI X3.64 basis. SCO has SCOANSI and Linux has Linux Console. Also there is AT386 used by Interactive Unix and later (I think) Unixware. The problem with these "emulations" is that they aren't emulations at all, since they aren't emulating anything and there is no physical terminal (like the VT100) to check them against. Their specifications can be hard to come by, and even when found, incomplete and/or inaccurate.

To this day, we discover undocumented sequences used by applications that were designed for SCOANSI, AT386, etc.

As for the Backspace/Delete controversy... This is decades old. The best policy (by the Principle of Least Astonishment) is to treat them the same by default, as many Unix variations now do, and use them to erase the character left of the cursor. People who truly need to distinguish between Rubout (127) and Backspace (8) will have enough of a clue to be able to do so.

Those of us who still use command shells are left with the remnants of two ancient cultures: the original Unix culture and the once-mighty DEC culture. Remember that Unix was first developed in the late 1960s, and predates the DEC heyday. The Unix guys picked certain keys to do certain things. DEC picked other keys to do the same things:

	Unix	DEC
Erase Character	<code>^H</code>	<code>^?</code>
Erase Line	<code>@</code>	<code>^U</code>
Interrupt Process	<code>^?</code>	<code>^C</code>

The differences might largely have been because Unix was originally accessed from Teletypes, which (if memory serves) did not have a proper Ctrl key, whereas DEC systems (PDP-11, PDP-10, etc) were mainly accessed from terminals that did have a Ctrl key (LA34, LA36, VT05, VT50, VT52, VT100, ...) (although it is also true that early DEC systems were delivered with Teletype consoles.)

After the fall of DEC culture, most of its people were absorbed into and influenced by Unix culture, and we began to see DEC usage predominate. Still, many Unixes are delivered with some or all of the default original Unix setups. The result being that when you Telnet to different Unix systems, you never know what Ctrl-H or Del/Rubout will do.

For that reason, terminal emulators always give you a choice for the mapping the Backspace (`<-`) key.

Of course matters are even more confusing when we throw in EMACS/Vi conventions, Function keys, the arrow keypad, and so forth, not to mention keys with promising labels like "Insert", "Page Up", "Find", "Select" and "Help", but that's a different story! Wouldn't it be great if, after all these years, keys actually did what their keytops say they do?

## Termcap

Termcap is `/etc/termcap` (unless you've defined the shell variable `TERMCAP` to point somewhere else). This is the original Unix method of storing terminal information. While termcap isn't as powerful as terminfo, it is simpler to use, and if the programmer doesn't need a lot of fancy features, this is often the choice made. The file consists of entries that describe terminals, both in terms of capabilities (screen dimensions, etc), what special keys send when pressed, and what sequences need to be sent to do things like erase a line or move the cursor. All these are described in **man F termcap**. It is important to understand the difference between a capability (what the program must do in order to make the terminal act a certain way) and a definition (the F1 key will send a certain sequence). If you changed the termcap file for the scoansi terminal so that the "k1" (F1 key) entry was defined as "ESC-J@", pressing your F1 key is NOT going to produce "ESC-J@". All that entry does is tell a program that if it sees "ESC-J@" coming from a scoansi terminal, it should interpret that as being the F1 key.

A program can use the termcap programming routines but read its information from a different termcap file. A programmer can also add new capabilities to that file (or even to `/etc/termcap`). The ubiquitous Filepro database uses both of those techniques to achieve its goals. You can add and change entries for your own purposes, too. For example, a certain program may work fine with your vt100 terminal except that it needs the F1 key but does not recognize it, because it is a "broken" program, badly written, and isn't using termcap or terminfo. Using the setup program within the terminal itself, you can reprogram the F1 key to send what the program expects to see (assuming you can determine this), but now other programs won't work because they are expecting the sequence that termcap (or terminfo) says the terminal will

send. By changing the entry in `/etc/termcap` (or in `terminfo`- see below), you can make both programs happy. Usually the best way to do this is to copy the "mostly working" entry to a new terminal name ("myvt100") and modify that, so that you aren't affecting anything but the terminal(s) you really need to.

As mentioned above, a program can have its own TERMCAP file, located somewhere other than `/etc`. If that's the case, you'd obviously need to look there for any modifications. Sometimes the file provides the functions of TERMCAP but not its syntax: RealWorld has sometimes used files like `ANSI.config`, etc. for this.

## Terminfo

Conceptually, `terminfo` is similar to `termcap`- it is a data base of terminal information. However, it is set up completely differently, and using it in a program is a little more difficult (though the difficulty is offset by increased capability). `Terminfo` keeps its data in a compiled form in subdirectories of `/usr/lib/terminfo`. You'll find Wyse terminals under `/usr/lib/terminfo/w`, vt100 models under `/usr/lib/terminfo/v`, ansi variations under `/usr/lib/terminfo/a`, etc. The source for all these terminals is usually found in `/usr/lib/terminfo/terminfo.src` (see **man M terminfo** for a description of the entries in `terminfo` source files).

Most modern Unix programs use `terminfo`. You can see that by copying the entry for your terminal in `/etc/termcap` to a new entry, and changing it only by modifying its name. For example, you might copy the "vt100" section to "myvt100". You could then set `TERM=myvt100`, `export TERM` (if it isn't already), and programs that use `termcap` would continue to work happily. Attempt to use `vi`, however, and you'd get a strange message:

```
myvt100: Unknown terminal type
I don't know what kind of terminal you are on - all I have is 'myvt100'
[Using open mode]
```

That's because `vi` uses `terminfo`, not `termcap`. You could copy `/usr/lib/terminfo/v/vt100` to `/usr/lib/terminfo/m/myvt100` and then `vi` would be happy.

You could also add a new entry to `terminfo.src` and use "tic" to compile it (which would create the compiled entry in the proper subdirectory). Some programs use `terminfo` files but put them in their own directory and use their own versions of "tic" (Pick based programs often do this, and so did the old Unix Lotus version, some versions of Synchronics, etc.).

## Solving Emulation Related Problems

If the terminal is connected by a serial line, be sure that the `/etc/ttytype` file has the proper entry and that the "tset" line in `.profile` knows how to handle that entry (note that the devices in `/etc/ttytype` are ALWAYS the lower case versions, so even if you are coming in on `/dev/tty2A`, you want to have `tty2a` in `/etc/ttytype`). You can cause `tset` to change the meaning of `/etc/ttytype`:

```
eval `tset -m wy50:wy60-25 -m ansi:${TERM:-ansi} -m :\?${TERM:-ansi} -e -r -s -Q`
```

That line will set `TERM` to `wy60-25` even when `/etc/ttytype` says it is a `wy50`.

If the terminal doesn't work properly, you can sometimes fix it by checking to see what it actually sends. Using "hd", you can type the keys you are having trouble with, and see if they match the proper entries in /etc/termcap or /usr/lib/terminfo.src. For example, if I run "hd" and press the F1 key on my console, followed by ENTER and 2 CTRL\_D's, I'll get back

```
0000      1b 5b 4d 0a      . [M.
```

That shows me that the F1 key generated Escape (that's hex 1b or decimal 27), then a bracket (hex 5b, decimal 91), then an "M" (hex 4d, decimal 77). The 0a is the Enter I typed after the F1.

Checking the ansi entry in /etc/termcap. I see that k1 is defined as "\E[M". As"\E" is termcap's notation for Escape, this is correct. Looking in /usr/lib/terminfo/terminfo.src reveals that the ansi entry defines kf1 (that's terminfo's notation for F1) identically. If these things were not true, I wouldn't expect my terminal to work correctly.

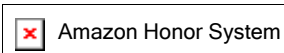
---

[Publish your articles, comments, book reviews or opinions here!](#)

---

Was this information valuable to you? Consider making a [contribution](#) to help pay for the costs of bringing it to you.

[Amazon Honor System](#) it's fast, safe and easy. It's also very private- Amazon doesn't share your information with anyone: I don't even get to find out who contributed!



Related Articles:

[SCO Unix/OSR5 Technical FAQ](#) (Basics)

[Shell Bashing](#) (Administration, Basics, Linux)

[VPN's and other remote access](#) (Basics, Linux, Security, Networking)

[Unix Permissions](#) (Basics, Linux)

[Installing an Equinox SST-64 under Linux](#) (Linux, Modems/Terminals)

[Basic Scripting](#) (Basics, Programming)

[Adding a Hard Drive to Linux](#) (Linux, Basics, Disks/Filesystems)

[Hard Disks](#) (Basics, Disks/Filesystems)

[SCO Unix/OSR5 Installation FAQ](#) (Basics)

[Floppies](#) (Basics, Disks/Filesystems)

[Basics](#) (Basics)

[Monitor users with Peek](#) (Modems/Terminals, Administration)

[Digiboard transparent print used wrong codes](#) (Modems/Terminals, Printing)

[The Chattering Printer](#) (Printing, Modems/Terminals)

[General Trouble Shooting](#) (Basics, Administration)

[Cron, At and Batch](#) (Kernel/Internals, Basics)

[Digiboard Port Server](#) (Reviews, Modems/Terminals)

[AlphaCom3 Terminal Emulator](#) (Windows, Reviews, Modems/Terminals)

[ICE.TEN & ICE.OFFSITE](#) (Modems/Terminals, Windows)

[Understanding RAID](#) (Disks/Filesystems, Basics)

[Understanding SCSI](#) (Disks/Filesystems, Basics)

[Winterm 5000 Terminal](#) (Reviews, Modems/Terminals, Networking)  
[Understanding Dumb Terminals](#) (Basics, Modems/Terminals)  
[Using Tapes and tape drives](#) (Basics, Backup)  
[Configuring SCO's GUI](#) (Misc., Administration, Basics)  
[Networks 101](#) (Networking, Basics)  
[OSR5 Boot Up Messages](#) (Kernel/Internals, Basics)  
[Setting up High Speed Modems](#) (Modems/Terminals, Administration)  
[Using the Korn Shell](#) (Programming, Basics)  
[Vi Primer](#) (Basics)  
[Routing Basics](#) (Networking, Basics)  
[Understanding Serial Wiring](#) (Basics, Modems/Terminals)  
[Unhappy Modems](#) (Modems/Terminals)  
[Qume Serial X-Terminal](#) (Reviews, Modems/Terminals)

© July 1999 A.P. Lawrence. All rights reserved

---

*This article is copyrighted material. You have permission to use it for any purpose, commercial or non-commercial, as long as it is kept intact and no modifications, additions, or deletions are made except as allowed herein.*

*You may publish it in paper or electronic form. That includes magazine, newsletters, and web pages, both internal and external, for profit or not. Banner ads and other graphics may be removed, but all other text, hyperlinks and copyright notices, including this, must remain (but see below also). You may not delete text, alter it, or add to it in any way that does not clearly delineate what is yours and what comes from this site. You may alter fonts, font sizes and the like and reformat text as is appropriate for your use.*

*You may select specific paragraphs or sections, but if you do so, you must include this entire notice also, noting that you have not published the entire article, or simply note that the paragraphs you have published are part of a larger article and give the http address of the actual article.*

*My main concern is that no one would be confused that you wrote something I wrote or vice-versa. If your use meets that concern, and allows people to find the original article here, I have no objection.*

*This general permission specifically does NOT apply to test questions and answers.*

*Some articles at <http://www.aplawrence.com/> and <http://www.pcunix.com/> are copyrighted by other individuals or corporations; these paragraphs do not apply to those articles even if accidentally included.*

*We do appreciate being advised of any such use: Email: [tony@aplawrence.com](mailto:tony@aplawrence.com).*