NAME

gettytab - terminal configuration data base

SYNOPSIS

gettytab

DESCRIPTION

The **gettytab** file is a simplified version of the termcap(5) data base used to describe terminal lines. The initial terminal login process getty(8) accesses the **gettytab** file each time it starts, allowing simpler reconfiguration of terminal characteristics. Each entry in the data base is used to describe one class of terminals.

There is a default terminal class, *default*, that is used to set global defaults for all other classes. (That is, the *default* entry is read, then the entry for the class required is used to override particular settings.)

CAPABILITIES

Refer to termcap(5) for a description of the file layout. The *default* column below lists defaults obtained if there is no entry in the table obtained, nor one in the special *default* table.

Name	Type	Default	Description
ac	str	unused	expect-send chat script for modem answer
al	str	unused	user to auto-login instead of prompting
ap	bool	false	terminal uses any parity
bk	str	0377	alternate end of line character (input break)
c0	num	unused	tty control flags to write messages
c1	num	unused	tty control flags to read login name
c2	num	unused	tty control flags to leave terminal as
ce	bool	false	use crt erase algorithm
ck	bool	false	use crt kill algorithm
cl	str	NULL	screen clear sequence
co	bool	false	console - add ‘\n’ after login prompt
ct	num	10	chat timeout for *ac* and *ic* scripts
dc	num	0	chat debug bitmask
de	num	0	delay secs and flush input before writing first prompt
df	str	%+	the strftime(3) format used for %d in the banner message
ds	str	‘^Y’	delayed suspend character
dx	bool	false	set DECCTLQ
ec	bool	false	leave echo *OFF*
ep	bool	false	terminal uses even parity
er	str	‘^?’	erase character

et	str	‘^D’	end of text (EOF) character
ev	str	NULL	initial environment
fl	str	‘^O’	output flush character
hc	bool	false	do *NOT* hangup line on last close
he	str	NULL	hostname editing regular expression
hn	str	hostname	hostname
ht	bool	false	terminal has real tabs
hw	bool	false	do cts/rts hardware flow control
i0	num	unused	tty input flags to write messages
i1	num	unused	tty input flags to read login name
i2	num	unused	tty input flags to leave terminal as
ic	str	unused	expect-send chat script for modem initialization
if	str	unused	display named file before prompt, like /etc/issue
ig	bool	false	ignore garbage characters in login name
im	str	NULL	initial (banner) message
iM	str	NULL	execute named file to generate initial (banner) message
in	str	‘^C’	interrupt character
is	num	unused	input speed
kl	str	‘^U’	kill character
l0	num	unused	tty local flags to write messages
l1	num	unused	tty local flags to read login name
l2	num	unused	tty local flags to leave terminal as
lm	str	login:	login prompt
ln	str	‘^V’	‘‘literal next’’ character
lo	str	*/usr/bin/login*	program to exec when name obtained
mb	bool	false	do flow control based on carrier
nc	bool	false	terminal does not supply carrier (set clocal)
nl	bool	false	terminal has (or might have) a newline character
np	bool	false	terminal uses no parity (i.e., 8-bit characters)
nx	str	default	next table (for auto speed selection)
o0	num	unused	tty output flags to write messages
o1	num	unused	tty output flags to read login name
o2	num	unused	tty output flags to leave terminal as
op	bool	false	terminal uses odd parity
os	num	unused	output speed
pc	str	‘\0’	pad character
pe	bool	false	use printer (hard copy) erase algorithm
pf	num	0	delay between first prompt and following flush (seconds)
pl	bool	false	start PPP login program unconditionally if *pp* is specified
pp	str	unused	PPP login program

ps	bool	false	line connected to a MICOM port selector
qu	str	‘^\’	quit character
rp	str	‘^R’	line retype character
rt	num	unused	ring timeout when using *ac*
rw	bool	false	do *NOT* use raw for input, use cbreak
sp	num	unused	line speed (input and output)
su	str	‘^Z’	suspend character
tc	str	none	table continuation
to	num	0	timeout (seconds)
tt	str	NULL	terminal type (for environment)
ub	bool	false	do unbuffered output (of prompts etc)
we	str	‘^W’	word erase character
xc	bool	false	do *NOT* echo control chars as ‘^X’
xf	str	‘^S’	XOFF (stop output) character
xn	str	‘^Q’	XON (start output) character
Lo	str	C	the locale name used for %d in the banner message

The following capabilities are no longer supported by getty(8):

bd	num	0	backspace delay
cb	bool	false	use crt backspace mode
cd	num	0	carriage-return delay
f0	num	unused	tty mode flags to write messages
f1	num	unused	tty mode flags to read login name
f2	num	unused	tty mode flags to leave terminal as
fd	num	0	form-feed (vertical motion) delay
lc	bool	false	terminal has lower case
nd	num	0	newline (line-feed) delay
uc	bool	false	terminal is known upper case only

If no line speed is specified, speed will not be altered from that which prevails when getty is entered. Specifying an input or output speed will override line speed for stated direction only.

Terminal modes to be used for the output of the message, for input of the login name, and to leave the terminal set as upon completion, are derived from the boolean flags specified. If the derivation should prove inadequate, any (or all) of these three may be overridden with one of the *c0*, *c1*, *c2*, *i0*, *i1*, *i2*, *l0*, *l1*, *l2*, *o0*, *o1*, or *o2* numeric specifications, which can be used to specify (usually in octal, with a leading ’0’) the exact values of the flags. These flags correspond to the termios *c_cflag*, *c_iflag*, *c_lflag*, and *c_oflag* fields, respectively. Each these sets must be completely specified to be effective.

Should getty(8) receive a null character (presumed to indicate a line break) it will restart using the table indicated by the *nx* entry. If there is none, it will re-use its original table.

Delays are specified in milliseconds, the nearest possible delay available in the tty driver will be used. Should greater certainty be desired, delays with values 0, 1, 2, and 3 are interpreted as choosing that particular delay algorithm from the driver.

The *cl* screen clear string may be preceded by a (decimal) number of milliseconds of delay required (a la termcap). This delay is simulated by repeated use of the pad character *pc*.

The initial message, login message, and initial file; *im*, *lm* and *if* may include any of the following character sequences, which expand to information about the environment in which getty(8) is running.

%d	The current date and time formatted according to the *Lo* and *df* strings.
%h	The hostname of the machine, which is normally obtained from the system using gethostname(3), but may also be overridden by the *hn* table entry. In either case it may be edited with the *he* POSIX "extended" regular expression, which is matched against the hostname. If there are no parenthesized subexpressions in the pattern, the entire matched string is used as the final hostname; otherwise, the first matched subexpression is used instead. If the pattern does not match, the original hostname is not modified.
%t	The tty name.
%m, %r, %s, %v	The type of machine, release of the operating system, name of the operating system, and version of the kernel, respectively, as returned by uname(3).
%%	A "%" character.

When getty execs the login process, given in the *lo* string (usually "*/usr/bin/login*"), it will have set the environment to include the terminal type, as indicated by the *tt* string (if it exists). The *ev* string, can be used to enter additional data into the environment. It is a list of comma separated strings, each of which will presumably be of the form name=value.

If a non-zero timeout is specified, with *to*, then getty will exit within the indicated number of seconds, either having received a login name and passed control to login(1), or having received an alarm signal, and exited. This may be useful to hangup dial in lines.

Output from getty(8) is even parity unless *op* or *np* is specified. The *op* string may be specified with *ap*

to allow any parity on input, but generate odd parity output. Note: this only applies while getty is being run, terminal driver limitations prevent a more complete implementation. The getty(8) utility does not check parity of input characters in RAW mode.

If a *pp* string is specified and a PPP link bring-up sequence is recognized, getty will invoke the program referenced by the *pp* option. This can be used to handle incoming PPP calls. If the *pl* option is true as well, getty(8) will skip the user name prompt and the PPP detection phase, and will invoke the program specified by *pp* instantly.

Getty provides some basic intelligent modem handling by providing a chat script feature available via two capabilities:

ic	Chat script to initialize modem.
ac	Chat script to answer a call.

A chat script is a set of expect/send string pairs. When a chat string starts, **getty** will wait for the first string, and if it finds it, will send the second, and so on. Strings specified are separated by one or more tabs or spaces. Strings may contain standard ASCII characters and special ’escapes’, which consist of a backslash character followed by one or more characters which are interpreted as follows:

\a	bell character.
\b	backspace.
\n	newline.
\e	escape.
\f	formfeed.
\p	half-second pause.
\r	carriage return.
\S, \s	space character.
\t	tab.
\xNN	hexadecimal byte value.
\0NNN	octal byte value.

Note that the ‘\p’ sequence is only valid for send strings and causes a half-second pause between sending the previous and next characters. Hexadecimal values are, at most, 2 hex digits long, and octal values are a maximum of 3 octal digits.

The *ic* chat sequence is used to initialize a modem or similar device. A typical example of an init chat script for a modem with a hayes compatible command set might look like this:

```
:ic="" ATE0Q0V1\r OK\r ATS0=0\r OK\r:
```

This script waits for nothing (which always succeeds), sends a sequence to ensure that the modem is in the correct mode (suppress command echo, send responses in verbose mode), and then disables auto-answer. It waits for an "OK" response before it terminates. The init sequence is used to check modem responses to ensure that the modem is functioning correctly. If the init script fails to complete, **getty** considers this to be fatal, and results in an error logged via syslogd(8), and exiting.

Similarly, an answer chat script is used to manually answer the phone in response to (usually) a "RING". When run with an answer script, **getty** opens the port in non-blocking mode, clears any extraneous input and waits for data on the port. As soon as any data is available, the answer chat script is started and scanned for a string, and responds according to the answer chat script. With a hayes compatible modem, this would normally look something like:

```
:ac=RING\r ATA\r CONNECT:
```

This causes the modem to answer the call via the "ATA" command, then scans input for a "CONNECT" string. If this is received before a *ct* timeout, then a normal login sequence commences.

The *ct* capability specifies a timeout for all send and expect strings. This timeout is set individually for each expect wait and send string and must be at least as long as the time it takes for a connection to be established between a remote and local modem (usually around 10 seconds).

In most situations, you will want to flush any additional input after the connection has been detected, and the *de* capability may be used to do that, as well as delay for a short time after the connection has been established during which all of the connection data has been sent by the modem.

SEE ALSO

login(1), gethostname(3), uname(3), termcap(5), getty(8)

HISTORY

The **gettytab** file format appeared in 4.2BSD.

BUGS

The special characters (erase, kill, etc.) are reset to system defaults by login(1). In *all* cases, '#' or '^H' typed in a login name will be treated as an erase character, and '@' will be treated as a kill character.

The delay stuff is a real crock. Apart form its general lack of flexibility, some of the delay algorithms are not implemented. The terminal driver should support sane delay settings.

The termcap(5) format is horrid, something more rational should have been chosen.