If the shell you're using is not intercepting it, you are typing a "Form-feed" character in your terminal. If the terminal application does not intercept or use the keystroke in some way, Ctrl +Letter is translated to the ASCII code of the letter minus 64(1). 65 is the ASCII code of 'A', 'L' is the 12th letter -> code 76. If the shell does not know what to do of the code, it prints it.

Printing a FF char resulted in a new page on a line printer and a clear screen on the terminal (yes, I used a VT-52 back then, at 300 baud).

So Ctrl +L is 12 which is FF. In the same way, Ctrl +I is a TAB, and Ctrl +G rings the bell --- if the terminal or the shell does not intercept it, like Ctrl +C for example.

Notice from the other answer: it seems that bash **do** intercept CTRL-L and do a `clear` . Nice touch that the `bash` authors associated the key with a command which will do more or less the same that the ASCII code did on old terminals!

On the other hand, in my `zsh` the combination CTRL-I works as TAB and CTRL-H as a Backspace (2).

Old nice ASCII... (notice that letter L is at column 4, row 12, it has ASCII code 4*16+12=76).

USASCII code chart

Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL