Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'trap.1p' command

$ man trap.1p

TRAP(1P) POSIX Programmer's Manual TRAP(1P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

trap ? trap signals

SYNOPSIS

trap n [condition...]

trap [action condition...]

DESCRIPTION

If the first operand is an unsigned decimal integer, the shell shall treat all operands as conditions, and shall reset each condition to the default value. Otherwise, if there are operands, the first is treated as an action and the remaining as conditions.

If action is '-', the shell shall reset each condition to the default value. If action is null (""), the shell shall ignore each specified condition if it arises. Otherwise, the argument action shall be read and executed by the shell when one of the corresponding conditions arises. The action of trap shall override a previous action (either de? fault action or one explicitly set). The value of "$?" after the trap action completes shall be the value it had before trap was invoked.

The condition can be EXIT, 0 (equivalent to EXIT), or a signal speci? fied using a symbolic name, without the SIG prefix, as listed in the tables of signal names in the <signal.h> header defined in the Base Definitions volume of POSIX.1?2017, Chapter 13, Headers; for example, HUP, INT, QUIT, TERM. Implementations may permit names with the SIG prefix or ignore case in signal names as an extension. Setting a trap for SIGKILL or SIGSTOP produces undefined results.

The environment in which the shell executes a trap on EXIT shall be identical to the environment immediately after the last command exe? cuted before the trap on EXIT was taken.

Each time trap is invoked, the action argument shall be processed in a manner equivalent to:

```
eval action
```

Signals that were ignored on entry to a non-interactive shell cannot be trapped or reset, although no error need be reported when attempting to do so. An interactive shell may reset or catch signals ignored on en? try. Traps shall remain in place for a given shell until explicitly changed with another trap command.

When a subshell is entered, traps that are not being ignored shall be set to the default actions, except in the case of a command substitu? tion containing only a single trap command, when the traps need not be altered. Implementations may check for this case using only lexical analysis; for example, if `trap` and $(trap --) do not alter the traps in the subshell, cases such as assigning var=trap and then using $($var) may still alter them. This does not imply that the trap command cannot be used within the subshell to set new traps.

The trap command with no operands shall write to standard output a list of commands associated with each condition. If the command is executed in a subshell, the implementation does not perform the optional check described above for a command substitution containing only a single trap command, and no trap commands with operands have been executed since entry to the subshell, the list shall contain the commands that were associated with each condition immediately before the subshell en?

vironment was entered. Otherwise, the list shall contain the commands currently associated with each condition. The format shall be:

"trap -- %s %s ...\n", <action>, <condition> ...

The shell shall format the output, including the proper use of quoting, so that it is suitable for reinput to the shell as commands that achieve the same trapping results. For example:

```
save_traps=$(trap)
...
eval "$save_traps"
```

XSI-conformant systems also allow numeric signal numbers for the condi? tions corresponding to the following signal names:

1 SIGHUP

2 SIGINT

3 SIGQUIT

6 SIGABRT

9 SIGKILL

14 SIGALRM

15 SIGTERM

The trap special built-in shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines.

OPTIONS

None.

OPERANDS

See the DESCRIPTION.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

None.

ASYNCHRONOUS EVENTS

Default.

STDOUT

See the DESCRIPTION.

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

If the trap name or number is invalid, a non-zero exit status shall be returned; otherwise, zero shall be returned. For both interactive and non-interactive shells, invalid signal names or numbers shall not be considered a syntax error and do not cause the shell to abort.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

None.

EXAMPLES

Write out a list of all traps and actions:

```
trap
```

Set a trap so the logout utility in the directory referred to by the HOME environment variable executes when the shell terminates:

```
trap "'$HOME"/logout' EXIT
```

or:

```
trap "'$HOME"/logout' 0
```

Unset traps on INT, QUIT, TERM, and EXIT:

```
trap - INT QUIT TERM EXIT
```

RATIONALE

Implementations may permit lowercase signal names as an extension. Im? plementations may also accept the names with the SIG prefix; no known historical shell does so. The trap and kill utilities in this volume of POSIX.1?2017 are now consistent in their omission of the SIG prefix for signal names. Some kill implementations do not allow the prefix, and

kill -l lists the signals without prefixes.

Trapping SIGKILL or SIGSTOP is syntactically accepted by some histori? cal implementations, but it has no effect. Portable POSIX applications cannot attempt to trap these signals.

The output format is not historical practice. Since the output of his? torical trap commands is not portable (because numeric signal values are not portable) and had to change to become so, an opportunity was taken to format the output in a way that a shell script could use to save and then later reuse a trap if it wanted.

The KornShell uses an ERR trap that is triggered whenever set -e would cause an exit. This is allowable as an extension, but was not mandated, as other shells have not used it.

The text about the environment for the EXIT trap invalidates the behav? ior of some historical versions of interactive shells which, for exam? ple, close the standard input before executing a trap on 0. For exam? ple, in some historical interactive shell sessions the following trap on 0 would always print "--":

```
trap 'read foo; echo "-$foo-"' 0
```

The command:

```
trap 'eval " $cmd"' 0
```

causes the contents of the shell variable cmd to be executed as a com? mand when the shell exits. Using:

```
trap '$cmd' 0
```

does not work correctly if cmd contains any special characters such as quoting or redirections. Using:

```
trap " $cmd" 0
```

also works (the leading <space> character protects against unlikely cases where cmd is a decimal integer or begins with '-'), but it ex? pands the cmd variable when the trap command is executed, not when the exit action is executed.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.14, Special Built-In Utilities

The Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syn?

tax Guidelines, <signal.h>

COPYRIGHT