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 'break.1p' command

$ man break.1p

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

break ? exit from for, while, or until loop

SYNOPSIS

break [n]

DESCRIPTION

If n is specified, the break utility shall exit from the nth enclosing for, while, or until loop. If n is not specified, break shall behave as if n was specified as 1. Execution shall continue with the command im? mediately following the exited loop. The value of n is a positive deci? mal integer. If n is greater than the number of enclosing loops, the outermost enclosing loop shall be exited. If there is no enclosing loop, the behavior is unspecified.

A loop shall enclose a break or continue command if the loop lexically encloses the command. A loop lexically encloses a break or continue command if the command is:

* Executing in the same execution environment (see Section 2.12, Shell Execution Environment) as the compound-list of the loop's do-

group (see Section 2.10.2, Shell Grammar Rules), and

* Contained in a compound-list associated with the loop (either in the compound-list of the loop's do-group or, if the loop is a while or until loop, in the compound-list following the while or until reserved word), and

* Not in the body of a function whose function definition command (see Section 2.9.5, Function Definition Command) is contained in a compound-list associated with the loop.

If n is greater than the number of lexically enclosing loops and there is a non-lexically enclosing loop in progress in the same execution en? vironment as the break or continue command, it is unspecified whether that loop encloses the command.

OPTIONS

None.

OPERANDS

See the DESCRIPTION.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

None.

ASYNCHRONOUS EVENTS

Default.

STDOUT

Not used.

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

0 Successful completion.

>0 The n value was not an unsigned decimal integer greater than or equal to 1.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

None.

EXAMPLES

```
for i in *
do
   if test -d "$i"
   then break
   fi
done
```

The results of running the following example are unspecified: there are two loops in progress when the break command is executed, and they are in the same execution environment, but neither loop is lexically en? closing the break command. (There are no loops lexically enclosing the continue commands, either.)

```
foo() {
   for j in 1 2; do
      echo 'break 2' >/tmp/do_break
      echo "  sourcing /tmp/do_break ($j)..."
      # the behavior of the break from running the following command
      # results in unspecified behavior:
      . /tmp/do_break
      do_continue() { continue 2; }
      echo "  running do_continue ($j)..."
      # the behavior of the continue in the following function call
      # results in unspecified behavior (if execution reaches this
      # point):
      do_continue
```

```
        trap 'continue 2' USR1
        echo "  sending SIGUSR1 to self ($j)..."
        # the behavior of the continue in the trap invoked from the
        # following signal results in unspecified behavior (if
        # execution reaches this point):
        kill -s USR1 $$
        sleep 1
    done
}
for i in 1 2; do
    echo "running foo ($i)..."
    foo
done
```

RATIONALE

In early proposals, consideration was given to expanding the syntax of break and continue to refer to a label associated with the appropriate loop as a preferable alternative to the n method. However, this volume of POSIX.1?2017 does reserve the name space of command names ending with a <colon>. It is anticipated that a future implementation could take advantage of this and provide something like:

```
outofloop: for i in a b c d e
do
    for j in 0 1 2 3 4 5 6 7 8 9
    do
        if test -r "${i}${j}"
        then break outofloop
        fi
    done
done
```

and that this might be standardized after implementation experience is achieved.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.14, Special Built-In Utilities

COPYRIGHT