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 'nice.1p' command

$ man nice.1p

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

nice ? invoke a utility with an altered nice value

SYNOPSIS

nice [-n increment] utility [argument...]

DESCRIPTION

The nice utility shall invoke a utility, requesting that it be run with a different nice value (see the Base Definitions volume of POSIX.1?2017, Section 3.244, Nice Value). With no options, the exe? cuted utility shall be run with a nice value that is some implementa? tion-defined quantity greater than or equal to the nice value of the current process. If the user lacks appropriate privileges to affect the nice value in the requested manner, the nice utility shall not affect the nice value; in this case, a warning message may be written to stan? dard error, but this shall not prevent the invocation of utility or af? fect the exit status.

OPTIONS

The nice utility shall conform to the Base Definitions volume of

POSIX.1?2017, Section 12.2, Utility Syntax Guidelines.

The following option is supported:

-n increment

A positive or negative decimal integer which shall have the same effect on the execution of the utility as if the utility had called the nice() function with the numeric value of the increment option-argument.

OPERANDS

The following operands shall be supported:

utility The name of a utility that is to be invoked. If the utility operand names any of the special built-in utilities in Sec? tion 2.14, Special Built-In Utilities, the results are unde? fined.

argument Any string to be supplied as an argument when invoking the utility named by the utility operand.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of nice:

LANG Provide a default value for the internationalization vari? ables that are unset or null. (See the Base Definitions vol? ume of POSIX.1?2017, Section 8.2, Internationalization Vari? ables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

LC_MESSAGES

Determine the locale that should be used to affect the format

and contents of diagnostic messages written to standard er?

ror.

NLSPATH Determine the location of message catalogs for the processing

of LC_MESSAGES.

PATH Determine the search path used to locate the utility to be

invoked. See the Base Definitions volume of POSIX.1?2017,

Chapter 8, Environment Variables.

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

If utility is invoked, the exit status of nice shall be the exit status

of utility; otherwise, the nice utility shall exit with one of the fol?

lowing values:

1?125 An error occurred in the nice utility.

126 The utility specified by utility was found but could not be in?

voked.

127 The utility specified by utility could not be found.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

The only guaranteed portable uses of this utility are:

nice utility

Run utility with the default higher or equal nice value.

nice -n <positive integer> utility

Run utility with a higher nice value.

On some implementations they have no discernible effect on the invoked utility and on some others they are exactly equivalent.

Historical systems have frequently supported the <positive integer> up to 20. Since there is no error penalty associated with guessing a num? ber that is too high, users without access to the system conformance document (to see what limits are actually in place) could use the his? torical 1 to 20 range or attempt to use very large numbers if the job should be truly low priority.

The nice value of a process can be displayed using the command:

ps -o nice

The command, env, nice, nohup, time, and xargs utilities have been specified to use exit code 127 if an error occurs so that applications can distinguish ``failure to find a utility'' from ``invoked utility exited with an error indication''. The value 127 was chosen because it is not commonly used for other meanings; most utilities use small val? ues for ``normal error conditions'' and the values above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen in a similar manner to indicate that the utility could be found, but not invoked. Some scripts produce meaningful error messages differ? entiating the 126 and 127 cases. The distinction between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to exec the utility fail with [ENOENT], and uses 126 when any attempt to exec the utility fails for any other reason.

EXAMPLES

None.

RATIONALE

The 4.3 BSD version of nice does not check whether increment is a valid decimal integer. The command nice -x utility, for example, would be treated the same as the command nice --1 utility. If the user does not have appropriate privileges, this results in a ``permission denied'' error. This is considered a bug.

When a user without appropriate privileges gives a negative increment,

System V treats it like the command nice -0 utility, while 4.3 BSD writes a ``permission denied'' message and does not run the utility. The standard specifies the System V behavior together with an optional BSD-style ``permission denied'' message.

The C shell has a built-in version of nice that has a different inter? face from the one described in this volume of POSIX.1?2017.

The term ``utility'' is used, rather than ``command'', to highlight the fact that shell compound commands, pipelines, and so on, cannot be used. Special built-ins also cannot be used. However, ``utility'' in? cludes user application programs and shell scripts, not just utilities defined in this volume of POSIX.1?2017.

Historical implementations of nice provide a nice value range of 40 or 41 discrete steps, with the default nice value being the midpoint of that range. By default, they raise the nice value of the executed util? ity by 10.

Some historical documentation states that the increment value must be within a fixed range. This is misleading; the valid increment values on any invocation are determined by the current process nice value, which is not always the default.

The definition of nice value is not intended to suggest that all pro? cesses in a system have priorities that are comparable. Scheduling pol? icy extensions such as the realtime priorities in the System Interfaces volume of POSIX.1?2017 make the notion of a single underlying priority for all scheduling policies problematic. Some implementations may im? plement the nice-related features to affect all processes on the sys? tem, others to affect just the general time-sharing activities implied by this volume of POSIX.1?2017, and others may have no effect at all.

Because of the use of ``implementation-defined'' in nice and renice, a wide range of implementation strategies are possible.

Earlier versions of this standard allowed a -increment option. This form is no longer specified by POSIX.1?2008 but may be present in some implementations.

None.

SEE ALSO

Chapter 2, Shell Command Language, renice

The Base Definitions volume of POSIX.1?2017, Section 3.244, Nice Value, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guide? lines

The System Interfaces volume of POSIX.1?2017, nice()

COPYRIGHT