IEEE P1003.2a Draft 8 – December 1991

Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

message to tell the user that the environment changed, it would be inappropriate to require this change to some historical implementations.

The password mechanism is allowed in the group file, but how this would be implemented is not specified.

The `newgrp` utility was retained in POSIX.2 even given the existence of the multiple group permissions feature in POSIX.1 {8}, because it allows files to be created with a specific group ownership. In some systems, the group ownership is determined by the group of the parent directory in which a file is created, as allowed by POSIX.1 {8}; on such systems, `newgrp` does not affect the group ownership in directories where the parent directory dictates it. However, POSIX.1 {8} allows systems to omit this feature; therefore, the `newgrp` functionality is still required on those systems.

5.20 `nice` — Invoke a utility with an altered system scheduling priority

5.20.1 Synopsis

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

Obsolescent Version:

`nice [-`*increment* `]` *utility* `[`*argument* `...]`

5.20.2 Description

The `nice` utility shall invoke a utility, requesting that it be run with a different system scheduling priority (see 2.2.2.212). With no options and only if the user has appropriate privileges, the executed utility shall be run with a system scheduling priority that is some implementation-defined quantity less than or equal to the system scheduling priority of the current process. If the user lacks appropriate privileges to affect the system scheduling priority in the requested manner, the `nice` utility shall not affect the system scheduling priority; in this case, a warning message may be written to standard error, but this shall not prevent the invocation of *utility* or affect the exit status.

5.20.3 Options

The `nice` utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that the obsolescent version allows a multidigit decimal integer as an option name.

The following option shall be supported by the implementation:

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

4165 –n *increment*
4166 -*increment* (Obsolescent.)
4167 Specify how the system scheduling priority of the executed utility
4168 shall be adjusted. The *increment* option-argument shall be a posi-
4169 tive or negative decimal integer that shall be used to modify the
4170 system scheduling priority of the executed utility in an
4171 implementation-defined manner.

4172 Positive *increment* values shall cause a lower or unchanged sys-
4173 tem scheduling priority. Negative *increment* values may require
4174 appropriate privileges and shall cause a higher or unchanged sys-
4175 tem scheduling priority.

4176 The system scheduling priority shall be bounded in an
4177 implementation-defined manner. If the requested *increment*
4178 would raise or lower the system scheduling priority of the exe-
4179 cuted utility beyond implementation-defined limits, then the limit
4180 whose value was exceeded shall be used.

4181 ### 5.20.4 Operands

4182 The following operands shall be supported by the implementation:

4183 *utility* The name of a utility that is to be invoked. If the *utility* operand
4184 names any of the special built-in utilities in 3.14, the results are
4185 undefined.

4186 *argument* Any string to be supplied as an argument when invoking the util-
4187 ity named by the *utility* operand.

4188 ### 5.20.5 External Influences

4189 #### 5.20.5.1 Standard Input

4190 None.

4191 #### 5.20.5.2 Input Files

4192 None.

4193 #### 5.20.5.3 Environment Variables

4194 The following environment variables shall affect the execution of `nice`:

4195 **LANG** This variable shall determine the locale to use for the
4196 locale categories when both **LC_ALL** and the correspond-
4197 ing environment variable (beginning with **LC_**) do not
4198 specify a locale. See 2.6.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments).

LC_MESSAGES This variable shall determine the language in which messages should be written.

PATH This variable shall determine the search path used to locate the utility to be invoked. See 2.6.

5.20.5.4 Asynchronous Events

Default.

5.20.6 External Effects

5.20.6.1 Standard Output

None.

5.20.6.2 Standard Error

Used only for diagnostic messages.

5.20.6.3 Output Files

None.

5.20.7 Extended Description

None.

5.20.8 Exit Status

If the *utility* 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 following values:

1–125 An error occurred in the `nice` utility.

126 The utility specified by *utility* was found but could not be invoked.

127 The utility specified by *utility* could not be found.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.20.9 Consequences of Errors

Default.

5.20.10 Rationale. *(This subclause is not a part of P1003.2a)*

Usage, Examples

Note that, in the obsolescent version, `–5` is a positive *increment*, while `––5` is a negative *increment*. While this might appear counterintuitive, it is how historical versions of `nice` work.

The only guaranteed portable uses of this utility are:

`nice` *utility* Run *utility* with the default lower system scheduling priority.

`nice -n` ‹*positive integer*› *utility*
Run *utility* with a lower system scheduling priority.

On some systems they will have no discernible effect on the invoked utility and on others they will be exactly equivalent.

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

The system scheduling priority 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 values 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 differentiating 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. In System V, the exit status from `nice`, when `nice` detects an error, is 2. In 4.3 BSD the status is 1.

History of Decisions Made

Due to the text about the limits of the system scheduling priority being implementation defined, `nice` is not required to actually change the priority of the executed command; the limits could be zero differences from the system default, although

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

the implementor is required to document this fact in the conformance document. Because of this, including this utility in POSIX.2 does not require implementations to support something like the traditional *nice*() system call. In particular, a POSIX.2 implementation could run on a nonextended POSIX.1 {8} implementation simply by supplying the following `nice` utility. (The example supports only the nonobsolescent form for illustrative purposes.)

```
#include <stdio.h>
extern int      optind;
extern char     *optarg;
#define USAGE   "Usage: nice [-n incr] <utility> [<arg> ...]\n"

int main(int argc, char *argv[]);
{
        int     i;

        while ((i = getopt(argc, argv, "n:")) != EOF) {
                switch (i) {
                    default:
                        fprintf(stderr, USAGE);
                        exit(127);
                    case 'n':  /* ignore increment */
                        break;
                }
        }
        argc -= optind;
        argv += optind;
        if (argc > 0) {
                execvp(argv[0], &argv[0]);
                perror(argv[0]);
        } else {
                fprintf(stderr, USAGE);
        }
        exit(127);
}
```

On systems that choose this approach, `nice` is a benign no-op. On the vast majority of systems that support `nice` historically, it becomes more useful. The developers of the standard considered this a quality of implementation issue and resisted calls to omit `nice` unless it could be guaranteed to have a discernible effect on all systems. The efficacy of `nice` is a legitimate procurement consideration.

The 4.3 BSD version of `nice` does not check if *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 will result 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. Neither was considered clearly superior, so the behavior was left unspecified.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

The C-shell has a built-in version of `nice` that has a different interface from the one described here.

The term ‘‘utility’’ is used, rather than ‘‘command,’’ to highlight the fact that shell compound commands, pipelines, etc., cannot be used. Special built-ins also cannot be used. However, ‘‘utility’’ includes user application programs and shell scripts, not just utilities defined in this standard.

Historical implementations of `nice` provide a priority range of 40 or 41 discrete steps, with the default priority being the midpoint of that range. By default they lower the priority of the executed utility 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 priority, which is not always the default.

The two limits in a previous draft, {POSIX 2_MAX_NICE } and {POSIX 2_MIN _NICE }, were removed when the `nice` description was reworded to allow users to exceed the system’s range with impunity.

The definition of *system scheduling priority* is not intended to suggest that all processes in a system have priorities that are comparable. Scheduling policy extensions such as the realtime priorities in POSIX. 4 make the notion of a single underlying priority for all scheduling policies problematic. Some systems may implement the `nice`-related features to affect all processes on the system, others to affect just the general time-sharing activities implied by POSIX. 2, 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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.21 nm — Write the name list of an object file

This utility shall be provided on systems that support both the User Portability Utilities Option and the Software Development Utilities Option. On other systems it is optional.

5.21.1 Synopsis

```
nm [-APv][ -g| -u ][-t format ]file ...
```

5.21.2 Description

The nm utility shall display symbolic information appearing in the object file, executable file, or object-file library named by *file*. If no symbolic information is available for a valid input file, the nm utility shall report that fact, but not consider that an error condition.

The default base used when numeric values are written is unspecified.

5.21.3 Options

The nm utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

-A Write the full pathname or library name of an object on each line.

-g Write only external (global) symbol information.

-P Write information in a portable output format, as specified in 5.21.6.1.

-t *format* Write each numeric value in the specified format. The format shall be dependent on the single character used as the *format* option-argument:

- **d** The offset shall be written in decimal.
- **o** The offset shall be written in octal.
- **x** The offset shall be written in hexadecimal.

-u Write only undefined symbols.

-v Sort output by value instead of alphabetically.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.