IEEE P1003.2 Draft 11.2 – September 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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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]

to support (in particular, not to 1, 0, or negative bases, if an implementation supports them as an extension).

4.4 `cat` — Concatenate and print files

4.4.1 Synopsis

`cat [-u] [`*file* `...]`

4.4.2 Description

The `cat` utility reads files in sequence and writes their contents to the standard output in the same sequence.

4.4.3 Options

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

The following option shall be supported by the implementation:

`-u` Write bytes from the input file to the standard output without delay as each is read.

4.4.4 Operands

The following operand shall be supported by the implementation:

file A pathname of an input file. If no *file* operands are specified, the standard input is used. If a *file* is -, the `cat` utility shall read from the standard input at that point in the sequence. The `cat` utility shall not close and reopen standard input when it is referenced in this way, but shall accept multiple occurrences of - as a *file* operand.

4.4.5 External Influences

4.4.5.1 Standard Input

The standard input is used only if no *file* operands are specified, or if a *file* operand is -. See Input Files.

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

4.4.5.2 Input Files

The input files can be any file type.

4.4.5.3 Environment Variables

The following environment variables shall affect the execution of `cat`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

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.

4.4.5.4 Asynchronous Events

Default.

4.4.6 External Effects

4.4.6.1 Standard Output

The standard output shall contain the sequence of bytes read from the input file(s). Nothing else shall be written to the standard output.

4.4.6.2 Standard Error

Used only for diagnostic messages.

4.4.6.3 Output Files

None.

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

4.4.7 Extended Description

None.

4.4.8 Exit Status

The `cat` utility shall exit with one of the following values:

0 All input files were output successfully.

>0 An error occurred.

4.4.9 Consequences of Errors

Default.

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

Examples, Usage

Historical versions of the `cat` utility include the options `-e`, `-t`, and `-v`, which permit the ends of lines, <tab>s, and invisible characters, respectively, to be rendered visible in the output. The working group omitted these options because they provide too fine a degree of control over what is made visible, and similar output can be obtained using a command such as:

```
sed -n -e 's/$/$/' -e l pathname
```

The `-s` option was omitted because it corresponds to different functions in BSD and System V-based systems. The BSD `-s` option to squeeze blank lines will be handled by `more -s` in the UPE. The System V `-s` option to silence error messages can be accomplished by redirecting the standard error. An alternative to `cat -s` is the following shell script using `sed`:

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

```
sed -n '
# Write non-empty lines.
/./     {
        p
        d
        }
# Write a single empty line, then look for more empty lines.
/^$/    p
# Get next line, discard the held <newline> (empty line),
# and look for more empty lines.
:Empty
/^$/    {
        N
        s/.//
        b Empty
        }
# Write the non-empty line before going back to search
# for the first in a set of empty lines.
        p
'
```

Note that the BSD documentation for `cat` uses the term ‘‘blank line’’ to mean the same as the POSIX ‘‘empty line’’; a line consisting only of a `<newline>`.

The BSD `–n` option is omitted because similar functionality can be obtained from the `–n` option of the `pr` utility.

The `–u` option is included here for its value in prototyping nonblocking reads from FIFO s. The intent is to support the following sequence:

```
mkfifo foo
cat –u foo > /dev/tty13 &
cat –u > foo
```

It is unspecified whether standard output is or is not buffered in the default case. This is sometimes of interest when standard output is associated with a terminal, since buffering may delay the output. The presence of the `–u` option guarantees that unbuffered I/O is available. It is implementation dependent whether the `cat` utility buffers output if the `–u` option is not specified. Traditionally, the `–u` option is implemented using the BSD *setbuffer* () function, the System V *setbuf* () function, or the C Standard {7}*setvbuf* () function.

The following command

```
cat myfile
```

writes the contents of the file `myfile` to standard output.

The following command

```
cat doc1 doc2 > doc.all
```

concatenates the files `doc1` and `doc2` and writes the result to `doc.all`.

Because of the shell language mechanism used to perform output redirection, a command such as this:

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

```
cat doc doc.end > doc
```

causes the original data in `doc` to be lost.

Due to changes made to subclause 2.11.4 in Draft 11, the description of the *file* operand now states that – must be accepted multiple times, as in historical practice. This allows the command:

```
cat start - middle - end > file
```

when standard input is a terminal, to get two arbitrary pieces of input from the terminal with a single invocation of `cat`. Note, however, that if standard input is a regular file, this would be equivalent to the command:

```
cat start - middle /dev/null end > file
```

because the entire contents of the file would be consumed by `cat` the first time – was used as a *file* operand and an end-of-file condition would be detected immediately when – was referenced the second time.

History of Decisions Made

None.

4.5 `cd` — Change working directory

4.5.1 Synopsis

`cd` [*directory*]

4.5.2 Description

The `cd` utility shall change the working directory of the current shell execution environment; see 3.12.

When invoked with no operands, and the **HOME** environment variable is set to a nonempty value, the directory named in the **HOME** environment variable shall become the new working directory. If **HOME** is empty or is undefined, the default behavior is implementation defined.

4.5.3 Options

None.

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