IEEE P1003.2a Draft 8 – December 1991

breaking many historical scripts relying on the 512-byte units.

It was suggested that `df` and the various related utilities be modified to access a **BLOCKSIZE** environment variable to achieve consistency and user acceptance. Since this is not existing practice on any system, it is left as an interesting area for system extensions and will be re-evaluated in a future revision if it is widely implemented.

5.9 du — Estimate file space usage

5.9.1 Synopsis

`du` [−a | −s] [−kx] [*file* . . .]

5.9.2 Description

The `du` utility, by default, shall write to standard output the size of the file space 7
allocated to, and the size of the file space allocated to each subdirectory of, the file 7
hierarchy rooted in each of the specified files. The size of the file space allocated 7
to a file of type directory shall be defined as the sum total of space allocated to all 7
files in the file hierarchy rooted in the directory plus the space allocated to the 7
directory itself. 7

When `du` cannot stat [see *stat* () in POSIX.1 {8} 5.6.2] files, or stat or read directories, it shall report an error condition and the final exit status shall be affected. Files with multiple links shall be counted and written for only one entry. The
directory entry that is selected in the report is unspecified. By default, file sizes 7
shall be written in 512-byte units, rounded up to the next 512-byte unit.

5.9.3 Options

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

The following options shall be supported by the implementation:

−a In addition to the default output, report the size of each file not of 7
type directory in the file hierarchy rooted in the specified file. 7
Regardless of the presence of the −a option, nondirectories given as *file* operands shall always be listed.

−k Write the files sizes in units of 1024 bytes, rather than the default 512-byte units.

`−s` Instead of the default output, report only the total sum for each of 7
the specified files. 7

`−x` When evaluating file sizes, evaluate only those files that have the same device as the file specified by the *file* operand.

5.9.4 Operands

The following operand shall be supported by the implementation:

file The pathname of a file whose size is to be written. If no *file* is specified, the current directory shall be used.

5.9.5 External Influences

5.9.5.1 Standard Input

None.

5.9.5.2 Input Files

None.

5.9.5.3 Environment Variables

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

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.

5.9.5.4 Asynchronous Events

Default.

5.9.6 External Effects

5.9.6.1 Standard Output

The output from `du` shall consist of the amount of the space allocated to a file and the name of the file, in the following format: 7

`"%d %s\n"`, *<size>*, *<pathname>*

5.9.6.2 Standard Error

Used only for diagnostic messages.

5.9.6.3 Output Files

None.

5.9.7 Extended Description

None.

5.9.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.9.9 Consequences of Errors

Default.

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

Usage, Examples

The use of 512-byte units is historical practice and maintains compatibility with `ls` and other utilities in the base standard and the UPE. This does not mandate that the file system itself be based on 512-byte blocks. The `−k` option was added as a compromise measure. It was agreed in the working group that 512 was the best default unit because of its complete historical consistency on System V (versus the mixed 512/1024 usage on BSD systems), and that a `−k` option to switch to 1024-byte units was a good compromise. Users who prefer the 1K quantity can easily alias `du` to `du -k` without breaking the many historical scripts relying on the 512-byte units.

History of Decisions Made

The −b option was added to an earlier draft to provide a resolution to the situation where System V and BSD systems give figures for file sizes in *blocks*, which is an implementation-defined concept. (In common usage the block size is 512 bytes for System V and 1024 bytes for BSD systems.) However, −b was later deleted, since the default was eventually decided as 512-byte units.

Historical file systems provided no way to obtain exact figures for the space allocation given to files. There are two known areas of inaccuracies in traditional file systems: cases of *indirect blocks* being used by the file system or *sparse* files yielding incorrectly high values. An indirect block is space used by the file system in the storage of the file, but that need not be counted in the space allocated to the file. A *sparse* file is one in which an *lseek*() (see 6.5.3 in POSIX.1 {8}) call has been made to a position beyond the end of the file and data has subsequently been written at that point. A file system need not allocate all the intervening zero-filled blocks to such a file. It is up to the implementation to define exactly how accurate its methods are.

The −a and −s options were mutually exclusive in the original version of du. The 7
POSIX.2 description is implied by the language in the *SVID*, where −s is described 7
as causing ‘‘only the grand total’’ to be reported. Some systems may produce out- 7
put for −sa, but a strictly conforming POSIX.2 application cannot use that combi- 7
nation. 7

The −a and −s options were adopted from the *SVID*, except that System V's behavior of not listing nondirectories explicitly given as operands, unless the −a option is specified, was considered a bug; the BSD-based behavior (report for all operands) is mandated. The default behavior of du in the *SVID* with regard to reporting the failure to read files (it produces no messages) was considered counter-intuitive, and thus it was specified that the POSIX.2 default behavior shall be to produce such messages. These messages can be turned off with shell redirection to achieve the System V behavior.

The −x option is existing practice on recent BSD systems. It has been adopted by POSIX.2 because there was no other historical method of limiting the du search to a single file hierarchy. This limitation of the search is necessary to make it possible to obtain file space usage information about a file system on which other file systems are mounted, without having to resort to a lengthy find and awk script.

5.10 `ex` — Text editor

5.10.1 Synopsis

`ex` [−rR] [−s | −v] [−c *command*] [−t *tagstring*] [−w *size*] [*file* ...]

Obsolescent Version :

`ex` [−rR] [− | −v] [+*command*] [−t *tagstring*] [−w *size*] [*file* ...] 8

5.10.2 Description

The `ex` utility is a line-oriented text editor that shall support both line and full-screen editing (see `vi` in 5.35).

Certain block-mode terminals do not have all the capabilities necessary to support the complete `ex` definition, such as the full-screen editing commands (*visual* 7
mode) or *open mode* . When these commands cannot be supported on such terminals, this condition shall not produce an error message such as ''not an editor command'' nor report a syntax error. The implementation may either accept the commands and produce results on the screen that are the result of an unsuccessful attempt to meet the requirements of this standard or report an error describing the terminal-related deficiency. The affected commands are noted as they occur later in this clause. 7

5.10.3 Options

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

The following options shall be supported by the implementation:

−c *command*

+*command* (Obsolescent.)

Begin editing by executing the specified `ex` command-mode command(s). As with normal editing command-line entries, the *command* option-argument can consist of multiple `ex` commands separated by vertical-line characters (|). The use of commands that enter input or visual modes in this manner produces undefined results.

−r Recover the named files after an editor or system crash, after the editor has been terminated by a signal, or after the use of a `preserve` editor command. A ''crash'' in this context is an unexpected failure of the system or utility that requires restarting the failed system or utility. A system crash implies that any utilities running at the time also crash. In the case of an editor or system crash, the degree of recovery (the number of changes to the buffer since the most recent `preserve` command) available is