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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

```
*/
%token inc_number
/*
  The inc_number is the number of times the succeeding increment
  period is to be added to the specified date and time.
*/
%token timezone_name
/*
  The name of an optional time zone suffix to the time                8
  field, in an implementation-defined format.                         8
*/
%token month_name
/*
  One of the values from the "mon" or "abmon" keywords in
  the LC_TIME locale category.
*/
%token day_of_week
/*
  One of the values from the "day" or "abday" keywords in
  the LC_TIME locale category.
*/
%token am_pm
/*
  One of the values from the "am_pm" keyword in
  the LC_TIME locale category.
*/
%start timespec
%%
timespec     : time
             | time date
             | time increment
             | time date increment
             | nowspec                                                8
             ;
nowspec      : "now"
             | "now" increment
             ;
time         : hr24clock_hr_min
             | hr24clock_hr_min timezone_name
             | hr24clock_hour ":" minute
             | hr24clock_hour ":" minute timezone_name
             | wallclock_hr_min am_pm
             | wallclock_hr_min am_pm timezone_name
             | wallclock_hour ":" minute am_pm
             | wallclock_hour ":" minute am_pm timezone_name
             | "noon"
             | "midnight"
                                                                      8
```

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

```
                ;
date            : month_name day_number
                | month_name day_number "," year_number
                | day_of_week
                | "today"
                | "tomorrow"
                ;
increment       : "+" inc_number inc_period
                : "next" inc_period
                ;
inc_period      : "minute" | "minutes"
                | "hour" | "hours"
                | "day" | "days"
                | "week" | "weeks"
                | "month" | "months"
                | "year" | "years"
                ;
```

5.2.5 External Influences

5.2.5.1 Standard Input

The standard input shall be a text file consisting of commands acceptable to the shell command language described in Section 3. The standard input shall only be used if no `-f` *file* option is specified.

5.2.5.2 Input Files

See Standard Input.

5.2.5.3 Environment Variables

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

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 interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files).

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

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

LC_TIME This variable shall determine the the format and contents for date and time strings written and accepted by `at`.

SHELL This variable shall be interpreted as a name of a command interpreter to be used to invoke the at-job. If the variable is unset or null, `sh` (see 4.56) shall be used. If it is set to a value other than a name for `sh`, the implementation shall do one of the following: use that shell; use `sh`; use the login shell from the user database; any of the preceding accompanied by a warning diagnostic about which was chosen.

TZ This variable shall specify a time zone. The job shall be submitted for execution at the time specified by *timespec* or `-t` *time* relative to the time zone specified by the **TZ** variable. If *timespec* specifies a time zone, it shall override **TZ**. If *timespec* does not specify a time zone and **TZ** is unset or null, an unspecified default time zone shall be used.

5.2.5.4 Asynchronous Events

Default.

5.2.6 External Effects

5.2.6.1 Standard Output

When standard input is a terminal, prompts of unspecified format for each line of the user input described in 5.2.5.1 may be written to standard output.

In the POSIX Locale, the following shall be written to the standard output for each job when jobs are listed in response to the `-l` option:

`"%s\t%s\n"`, *at_job_id* , *<date>*

where *<date>* shall be equivalent in format to the output of:

```
date +"%a %b %e %T %Y"
```

The date and time written shall be adjusted so that they appear in the time zone of the user (as determined by the **TZ** variable).

5.2.6.2 Standard Error

The following shall be written to standard error when a job has been successfully submitted:

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

398 "job %s at %s\n", *at_job_id* , *<date>*

399 where *<date>* shall have the same format as is described in Standard Output.
400 Neither this, nor warning messages concerning the selection of the command
401 interpreter, shall be considered a diagnostic that changes the exit status.

402 Diagnostic messages, if any, shall be written to standard error.

403 **5.2.6.3 Output Files**

404 None.

405 **5.2.7 Extended Description**

406 None.

407 **5.2.8 Exit Status**

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

409 0 The `at` utility successfully submitted, removed, or listed a job or jobs.

410 >0 An error occurred.

411 **5.2.9 Consequences of Errors**

412 The job shall not be scheduled, removed, or listed.

413 **5.2.10 Rationale.** *(This subclause is not a part of P1003.2a)*

414 **Usage, Examples**

415 The `at` utility reads from standard input the commands to be executed at a later
416 time. It may be useful to redirect standard output and standard error within the 7
417 specified commands. 7

418 The format of the `at` command line shown here is guaranteed only for the POSIX 8
419 Locale. Other cultures may be supported with substantially different interfaces, 8
420 although implementations are encouraged to provide comparable levels of func- 8
421 tionality. 8

422 This sequence can be used at a terminal:

```
423 at -m 0730 tomorrow
424 sort < file >outfile
425 EOT
```

426 This sequence, which demonstrates redirecting standard error to a pipe, is useful
427 in a command procedure (the sequence of output redirection specifications is
428 significant):

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

```
at now + 1 hour <<!
diff file1 file2 2>&1 >outfile | mailx mygroup
!
```

To have a job reschedule itself, `at` can be invoked from within the at-job.

The spacing of the three portions of the POSIX Locale *timespec* is quite flexible as long as there are no ambiguities. Examples of various times and operand presentation include:

```
at 0815am Jan 24
at 8 :15amjan24
at now "+ 1day"
at 5 pm FRIday
at ’17
        utc+
        30minutes’
```

Since the commands run in a separate shell invocation, running in a separate process group with no controlling terminal, open file descriptors, traps, and priority inherited from the invoking environment are lost.

Some implementations do not allow substitution of different shells using **SHELL**. System V systems, for example, have used the login shell value for the user in `/etc/passwd`. To reliably select another command interpreter, the user must include it as part of the script, such as:

```
$ at 1800
myshell myscript
job ... at ...
$
```

History of Decisions Made

Due to various implementations, and since the POSIX system administration group is dealing with administrative issues, the location or use of the files `at.allow` and `at.deny` were omitted. The following, if using these files, is a suggested implementation:

> Users shall be permitted to use `at` if their name appears in the file `at.allow`. If that file does not exist, the file `at.deny` shall be checked to determine if the user shall be denied access to `at`. If neither file exists, only root shall be allowed to submit a job. If only `at.deny` exists and is empty, global usage shall be permitted. The allow/deny files shall consist of one user name per line.

The `−t` *time* option was added as a new capability to support an internationalized way of specifying a time for execution of the submitted job.

Early drafts added a ‘‘jobname’’ concept as a way of giving submitted jobs names that are meaningful to the user submitting them. The historical, system-specified *at_job_id* gives no indication of what the job is. Upon further reflection, it was decided that the benefit of this was not worth the change in historical interface. It is anticipated that considerably more sophisticated ways of controlling

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

background or batch work will be the subject of future revisions to this standard.

The –q option historically has been an undocumented option, used mainly by the `batch` utility.

The System V –m option was added to provide a method for informing users that an at-job had completed. Otherwise, users are only informed when output to standard error or standard output are not redirected. This information is mailed to the user when the –m option is provided. When –m is not given, most systems mail the standard output and standard error to the user, but this is not required; some other method, such as a log file, could be used.

The behavior of `at` <*now*> was changed in an earlier draft from being unspecified to submitting a job for potentially immediate execution. Historical BSD `at` implementations support this. Historical System V implementations give an error in that case, but a change to the System V versions should have no backward compatibility ramifications.

On BSD-based systems, a –u *user* option has allowed those with appropriate privileges to access the work of other users. Since this is primarily a system administration feature and is not universally implemented, it has been omitted. Similarly, a specification for the output format for user with appropriate privileges viewing the queues of other users has been omitted.

The –f *file* option from System V is used instead of the BSD method of using the last operand as the pathname. The BSD method is ambiguous—does

```
at 1200 friday
```

mean the same thing if there is a file named `friday` in the current directory?

The *at_job_id* is composed of a limited character set in historical practice and it is mandated here to invalidate systems that might try using characters that require shell quoting or could not be easily parsed by shell scripts.

The `at` utility varies between System V and BSD systems in the way time zones are used. On System V systems, the **TZ** variable affects the at-job submission times and the times displayed for the user. On BSD systems, **TZ** is not taken into account. The BSD behavior is easily achieved with the current specification. If the user wishes to have the time zone default to that of the system, she merely needs to issue the `at` command immediately following an unsetting or null assignment to **TZ**, For example,

```
TZ= at noon . . .
```

will give the desired BSD result.

A previous draft indicated that various signals would be ignored. What it meant to say was that the at-job would not be affected by the various keyboard-related signals the user could initiate. The current wording about no controlling terminal should now suffice.

While the `yacc`-like grammar specified in 5.2.4 is lexically unambiguous with respect to the digit strings, a lexical analyzer would probably be written to look for and return digit strings in those cases. The parser could then check whether

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

the digit string returned is a valid `day_number`, `year_number`, etc., based on the context.

5.3 `batch` — Execute commands when the system load permits

5.3.1 Synopsis

```
batch
```

5.3.2 Description

The `batch` utility shall read commands to be executed at a later time. It shall be the equivalent of the command:

```
at -q b -m now
```

where queue `b` is a special `at` queue, specifically for batch jobs. Batch jobs shall be submitted to the batch queue with no time constraints and shall be run by the system using algorithms, based on unspecified factors, that may vary with each invocation of `batch`.

5.3.3 Options

None.

5.3.4 Operands

None.

5.3.5 External Influences

5.3.5.1 Standard Input

See 5.2.

5.3.5.2 Input Files

None.

5.3.5.3 Environment Variables

See 5.2.

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

5.3.5.4 Asynchronous Events

See 5.2.

5.3.6 External Effects

5.3.6.1 Standard Output

See 5.2.

5.3.6.2 Standard Error

See 5.2.

5.3.6.3 Output Files

None.

5.3.7 Extended Description

None.

5.3.8 Exit Status

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

0 The `batch` utility successfully submitted a job.

>0 An error occurred.

5.3.9 Consequences of Errors

The job shall not be scheduled.

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

Usage, Examples

The `batch` utility reads from standard input the commands to be executed at a later time. It may be useful to redirect standard output within the specified commands.

This sequence can be used at a terminal:

```
batch
sort < file >outfile
EOT
```

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

This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

```
batch <<!
diff file1 file2 2>&1 >outfile | mailx mygroup
!
```

History of Decisions Made

Early drafts described `batch` in a manner totally separated from `at`, even though the historical model treated it almost as a synonym for `at -q b`. A number of features were added to list and control batch work separately from those in `at`. Upon further reflection, it was decided that the benefit of this was not worth the change in historical interface. It is anticipated that considerably more sophisticated ways of controlling background or batch work will be the subject of future revisions to this standard.

The `–m` option was included on the equivalent `at` command because it is historical practice to mail results to the submitter, even if all job-produced output is redirected. As explained in the rationale for `at`, the `now` keyword submits the job for immediate execution (after scheduling delays), despite some historical systems where `at now` would have been considered an error. 7

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

5.4 bg — Run jobs in the background

This utility shall be provided on systems that support both the User Portability Utilities Option and the POSIX.1 {8} job control option. On other systems it is optional.

5.4.1 Synopsis

```
bg [job_id ...]
```

5.4.2 Description

If job control is enabled (see the description of `set –m` in 3.14.11), the `bg` utility shall resume suspended jobs from the current environment (see 3.12) by running them as background jobs. If the job specified by *job_id* is already a running background job, the `bg` utility shall have no effect and shall exit successfully.

Using `bg` to place a job into the background shall cause its process ID to become ''known in the current shell execution environment,'' as if it had been started as an asynchronous list; see 3.9.3.1.

5.4.3 Options

None.

5.4.4 Operands

The following operand shall be supported by the implementation:

job_id Specify the job to be resumed as a background job. If no *job_id* operand is given, the most recently suspended job shall be used. The format of *job_id* is described in 2.2.2.206.

5.4.5 External Influences

5.4.5.1 Standard Input

None.

5.4.5.2 Input Files

None.

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

5.4.5.3 Environment Variables

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

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.

5.4.5.4 Asynchronous Events

Default.

5.4.6 External Effects

5.4.6.1 Standard Output

The output of `bg` shall consist of a line in the format:

`"[%d] %s\n"`, *<job-number>* , *<command>*

where the fields are as follows:

<job-number > A number that can be used to identify the job to the `wait`, `fg`, and `kill` utilities. Using these utilities, the job can be identified by prefixing the job number with `%`.

<command > The associated command that was given to the shell.

5.4.6.2 Standard Error

Used only for diagnostic messages.

5.4.6.3 Output Files

None.

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

5.4.7 Extended Description

None.

5.4.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.4.9 Consequences of Errors

If job control is disabled, the `bg` utility shall exit with an error and no job shall be placed in the background.

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

Usage, Examples

The `bg` utility will not work as expected when it is operating in its own utility execution environment because that environment will have no suspended jobs. In the following examples:

```
... | xargs bg
(bg)
```

each `bg` operates in a different environment and will not share its parent shell's understanding of jobs. For this reason, `bg` is generally implemented as a shell regular built-in.

History of Decisions Made

The extensions to the shell specified in POSIX. 2 and POSIX. 2a have mostly been based on features provided by the KornShell. The job control features provided by `bg`, `fg`, and `jobs` are also based on the KornShell. The developers of the standard examined the characteristics of the C-Shell versions of these utilities and found that differences exist. Despite widespread use of the C-Shell, the KornShell versions were selected for this standard to maintain a degree of uniformity with the rest of the KornShell features selected (such as the very popular command-line editing features).

8

The `bg` utility is expected to wrap its output if the output exceeds the number of display columns.

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

5.5 `crontab` — Schedule periodic background work

5.5.1 Synopsis

```
crontab [file]
crontab [ -e | -l | -r ]
```

5.5.2 Description

The `crontab` utility shall create, replace, or edit a user's *crontab* entry; a crontab entry is a list of commands and the times at which they are to be executed. The new crontab entry can be input by specifying *file* or input from standard input if no *file* operand is specified, or by using an editor, if `-e` is specified.

Upon execution of a command from a crontab entry, the implementation shall supply a default environment, defining at least the following environment variables:

HOME A pathname of the user's home directory.

LOGNAME The user's login name.

PATH A string representing a search path guaranteed to find all of the standard utilities.

SHELL A pathname of the command interpreter. When `crontab` is invoked as specified by this standard, the value shall be a pathname for `sh` (see 4.56).

The values of these variables when `crontab` is invoked as specified by this standard shall not affect the default values provided when the scheduled command is run.

If standard output and standard error are not redirected by commands executed from the crontab entry, any generated output or errors shall be mailed, via an implementation-defined method, to the user.

5.5.3 Options

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

The following options shall be supported by the implementation:

`-e` Edit a copy of the invoking user's crontab entry, or create an empty entry to edit if the crontab entry does not exist. When editing is complete, the entry shall be installed as the user's crontab entry.

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

`-l` (The letter ell.) List the invoking user's crontab entry.

`-r` Remove the invoking user's crontab entry.

5.5.4 Operands

The following operand shall be supported by the implementation:

file The pathname of a file that contains specifications, in the format defined in 5.5.5.2, for crontab entries.

5.5.5 External Influences

5.5.5.1 Standard Input

See Input Files.

5.5.5.2 Input Files

In the POSIX Locale, a crontab entry shall be a text file consisting of lines of six fields each. The fields shall be separated by <blank>s. The first five fields shall be integer patterns that specify the following:

(1) Minute (0–59)

(2) Hour (0–23)

(3) Day of the month (1–31)

(4) Month of the year (1–12)

(5) Day of the week (0–6 with 0=Sunday).

Each of these patterns can be either an asterisk (meaning all valid values), an element, or a list of elements separated by commas. An element shall be either a number or two numbers separated by a hyphen (meaning an inclusive range). The specification of days can be made by two fields (day of the month and day of the week). If month, day of month, and day of week are all asterisks, every day shall be matched. If either the month or day of month is specified as an element or list, but the day of week is an asterisk, the month and day of month fields shall specify the days that match. If both month and day of month are specified as asterisk, but day of week is an element or list, then only the specified days of the week shall match. Finally, if either the month or day of month is specified as an element or list, and the day of week is also specified as an element or list, then any day matching either the month and day of month, or the day of week, shall be matched.

The sixth field of a line in a crontab entry is a string that shall be executed by `sh` (see 4.56) at the specified times. A percent-sign character in this field shall be translated to a <newline> character. Any character preceded by a backslash (including the %) shall cause that character to be treated literally. Only the first

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

line (up to a % or end of line) of the command field shall be executed by the command interpreter. The other lines shall be made available to the command as standard input.

Blank lines and those whose first non-<blank> character is # shall be ignored.

5.5.5.3 Environment Variables

The following environment variables shall affect the execution of crontab:

EDITOR This variable shall determine the editor to be invoked when the −e option is specified. The default editor shall be vi.

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 interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files).

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

5.5.5.4 Asynchronous Events

Default.

5.5.6 External Effects

5.5.6.1 Standard Output

If the −l option is specified, the crontab entry shall be written to the standard output.

5.5.6.2 Standard Error

Used only for diagnostic messages.

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

5.5.6.3 Output Files

None.

5.5.7 Extended Description

None.

5.5.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.5.9 Consequences of Errors

The user's crontab entry is not submitted, removed, edited, or listed.

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

Usage, Examples

As an example of specifying the two types of days,

```
0 0 1,15 * 1
```

would run a command on the first and fifteenth of each month, as well as on every Monday. To specify days by only one field, the other field should be set to * (for example,

```
0 0 * * 1
```

would run a command only on Mondays).

The default settings of the **HOME** , **LOGNAME** , **PATH** , and **SHELL** variables that are given to the scheduled job are not affected by the settings of those variables when `crontab` is run; as stated, they are defaults. The text about ''invoked as specified by this standard'' means that the implementation may provide extensions that allow these variables to be affected at runtime, but that the user has to take explicit action in order to access the extension, such as give a new option flag or modify the format of the crontab entry.

History of Decisions Made

All references to a `cron` daemon and to `cron` *files* have been removed. Although historical implementations have used this arrangement, there is no reason to limit future implementations.

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

This description of `crontab` is designed to support only users with normal privileges. The format of the input is based on the System V `crontab`; however, there is no requirement here that the actual system database used by the `cron` daemon (or similar mechanism) use this format internally. For example, systems derived from BSD will likely have an additional field appended that indicates the user identity to be used when the job is submitted.

The `-e` option was adopted from the *SVID* as a user convenience, although it does not exist in all historical implementations.

5.6 `csplit` — Split files based on context

5.6.1 Synopsis

`csplit [-ks][-f` *prefix* `][-n` *number* `]` *file arg1 ... argn*

5.6.2 Description

The `csplit` utility shall read the file named by the *file* operand, write all or part of that file into other files as directed by the *arg* operands, and write the sizes of the files.

5.6.3 Options

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

The following options shall be supported by the implementation:

`-f` *prefix* Name the created files *prefix* `00`, *prefix* `01`, ..., *prefixn* . The default shall be `xx00` ... `xx`*n*. If the *prefix* argument would create a filename exceeding {NAME_MAX} bytes, an error shall result, `csplit` shall exit with a diagnostic message and no files shall be created.

`-k` Leave previously created files intact. By default, `csplit` shall remove created files if an error occurs.

`-n` *number* Use *number* decimal digits to form filenames for the file pieces. The default shall be 2.

`-s` Suppress the output of file size messages.

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

827 ### 5.6.4 Operands

828 The following operands shall be supported by the implementation:

829 *file* The pathname of a text file to be split. If *file* is −, the standard
830 input shall be used.

831 The operands *arg1* ... *argn* can be a combination of the following:

832 /*rexp* /[*offset*]
833 Create a file using the content of the lines from the current line
834 up to, but not including, the line that results from the evaluation
835 of the regular expression with *offset* , if any, applied. The regular
836 expression *rexp* shall follow the rules for basic regular expres-
837 sions described in 2.8.3. The optional *offset* shall be a positive or
838 negative integer value representing a number of lines. The
839 integer value shall be preceded by + or −. If the selection of lines
840 from an offset expression of this type would create a file with zero
841 lines, or one with greater than the number of lines left in the
842 input file, the results are unspecified. After the section is created,
843 the current line shall be set to the line that results from the
844 evaluation of the regular expression with any offset applied. The
845 pattern match of *rexp* shall be always applied from the current
846 line to the end of the file.

847 %*rexp* %[*offset*]
848 This operand is the same as /*rexp* /[*offset*], except that no file shall
849 be created for the selected section of the input file. 7

850 *line_no* Create a file from the current line up to (but not including) the
851 line number *line_no* . Lines in the file shall be numbered starting
852 at one. The current line becomes *line_no* .

853 {*num* } Repeat operand. This operand can follow any of the operands
854 described previously. If it follows a *rexp* type operand, that
855 operand shall be applied *num* more times. If it follows a *line_no*
856 operand, the file shall be split every *line_no* lines, *num* times,
857 from that point.

858 An error shall be reported if an operand does not reference a line between the
859 current position and the end of the file.

860 ### 5.6.5 External Influences

861 #### 5.6.5.1 Standard Input

862 See Input Files.

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

5.6.5.2 Input Files

The input file shall be a text file.

5.6.5.3 Environment Variables

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

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_COLLATE This variable shall determine the locale for character collation information in regular expressions.

LC_CTYPE This variable shall determine the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files), and the behavior of character classes within regular expressions.

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

5.6.5.4 Asynchronous Events

If the `-k` option is specified, created files shall be retained. Otherwise the default action occurs.

5.6.6 External Effects

5.6.6.1 Standard Output

Unless the `-s` option is used, the standard output shall consist of one line per file created, with a format as follows:

`"%d\n"`, *<file size in bytes>*

5.6.6.2 Standard Error

Used only for diagnostic messages.

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

894 #### 5.6.6.3 Output Files

895 The output files shall contain portions of the original input file, otherwise
896 unchanged.

897 ### 5.6.7 Extended Description

898 None.

899 ### 5.6.8 Exit Status

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

901 0 Successful completion.

902 >0 An error occurred.

903 ### 5.6.9 Consequences of Errors

904 By default, created files shall be removed if an error occurs. When the `-k` option
905 is specified, created files shall not be removed if an error occurs.

906 ### 5.6.10 Rationale. *(This subclause is not a part of P1003.2a)*

907 **Usage, Examples**

908 This example creates four files, `cobol00` ... `cobol03`:

909 ```
csplit -f cobol file '/procedure division/' /par5./ /par16./
```

910 After editing the split files, they can be recombined as follows:

911 ```
cat cobol0[0-3] > file
```

912 Note that this example overwrites the original file.

913 This example would split the file after the first 99 lines, and every 100 lines there 7
914 after, up to 9 999 lines; this is because lines in the file are numbered from 1 rather 7
915 than zero, for historical reasons. 7

916 ```
csplit -k file 100 {99}
```

917 Assuming that `prog.c` follows the C-language coding convention of ending rou-
918 tines with a `}` at the beginning of the line, this example will create a file contain-
919 ing each separate C routine (up to 21) in `prog.c`.

920 ```
csplit -k prog.c '%main(%' '/^}/+1' {20}
```

921 The use of repeated regular expressions with negative offsets produces bizarre
922 results in historical implementations.

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