-- --

IEEE P1003.2 Draft 11.2 – September 1991

Section 6: Software Development Utilities Option

This section describes utilities used for the development of applications, including compilation or translation of source code, the creation and maintenance of library archives, and the maintenance of groups of interdependent programs.

The utilities described in this section may be provided by the conforming system; however, any system claiming conformance to the **Software Development Utilities Option** shall provide all of the utilities described here.

6.1 `ar` — Create and maintain library archives

6.1.1 Synopsis

`ar −d [−v]` *archive file* ...

`ar −p [−v]` *archive* [*file* ...]

`ar −r [−cuv]` *archive file* ...

`ar −t [−v]` *archive* [*file* ...]

`ar −x [−v]` *archive* [*file* ...]

6.1.2 Description

The `ar` utility can be used to create and maintain groups of files combined into an archive. Once an archive has been created, new files can be added, and existing files can be extracted, deleted, or replaced. When an archive consists entirely of valid object files, the implementation shall format the archive so that it is usable as a library for link editing (see A.1 and C.2). When some of the archived files are not valid object files, the suitability of the archive for library use is undefined.

All *file* operands can be pathnames. However, files within archives shall be named by a filename, which is the last component of the pathname used when the file was entered into the archive. The comparison of *file* operands to the names of files in archives shall be performed by comparing the last component of the operand to the name of the archive file.

It is unspecified whether multiple files in the archive may be identically named. In the case of such files, however, each *file* operand shall match only the first archive file having a name that is the same as the last component of the *file* operand.

6.1.3 Options

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

The following options shall be supported by the implementation:

`−c` Suppress the diagnostic message that is written to standard error by default when the archive file *archive* is created.

`−d` Delete *file* (s) from *archive.*

`−p` Write the contents of the *file* (s) from *archive* to the standard output. If no *file* (s) are specified, the contents of all files in the archive shall be written in the order of the archive.

`−r` Replace or add *file* (s) to *archive* . If the archive named by *archive* does not exist, a new archive file shall be created and a diagnostic message shall be written to standard error (unless the `−c` option is specified). If no *file* (s) are specified and the *archive* exists, the results are undefined. Files that replace existing files shall not change the order of the archive. Files that do not replace existing files shall be appended to the archive.

`−t` Write a table of contents of *archive* to the standard output. The files specified by the *file* operands shall be included in the written list. If no *file* operands are specified, all files in *archive* shall be included in the order of the archive.

`−u` Update older files. When used with the `−r` option, files within the archive will be replaced only if the corresponding *file* has a modification time that is at least as new as the modification time of the file within the archive.

`−v` Give verbose output. When used with the option characters `−d`, `−r`, or `−x`, write a detailed file-by-file description of the archive creation and maintenance activity, as described in 6.1.6.1.

When used with `−p`, write the name of the file to the standard output before writing the file itself to the standard output, as described in 6.1.6.1.

When used with `−t`, include a long listing of information about the files within the archive, as described in 6.1.6.1.

`−x` Extract the files named by the *file* operands from *archive* . The contents of the archive file shall not be changed. If no *file* operands are given, all files in the archive shall be extracted. If the filename of a file extracted from the archive is longer than that supported in the directory to which it is being extracted, the results are undefined. The modification time of each file extracted shall be set to the time the file is extracted from the archive.

6.1.4 Operands

The following operands shall be supported by the implementation:

archive A pathname of the archive file.

file A pathname. Only the last component shall be used when comparing against the names of files in the archive. If two or more *file* operands have the same last pathname component (basename), the results are unspecified. The implementation's archive format shall not truncate valid filenames of files added to, or replaced in, the archive.

6.1.5 External Influences

6.1.5.1 Standard Input

None.

6.1.5.2 Input Files

The input file named by *archive* shall be a file in the format created by `ar -r`.

6.1.5.3 Environment Variables

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

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.

LC_TIME This variable shall determine the format and content for date and time strings written by `ar`.

6.1.5.4 Asynchronous Events

Default.

6.1.6 External Effects

6.1.6.1 Standard Output

If the `−d` option is used with the `−v` option, the standard output format is:

`"d − %s\n"`, *<file>*

where *file* is the operand specified on the command line.

If the `−p` option is used with the `−v` option, `ar` shall precede the contents of each file with:

`"\n<%s>\n\n"`, *<file>*

where *file* is the operand specified on the command line, if *file* operands were specified, and the name of the file in the archive if they were not.

If the `−r` option is used with the `−v` option, and *file* is already in the archive, the standard output format is:

`"r − %s\n"`, *<file>*

where *file* is the operand specified on the command line.

If *file* is being added to the archive with the `−r` option, the standard output format is:

`"a − %s\n"`, *<file>*

where *file* is the operand specified on the command line.

If the `−t` option is used, `ar` writes the names of the files to the standard output in the format:

`"%s\n"`, *<file>*

where *file* is the operand specified on the command line, if *file* operands were specified, or the name of the file in the archive if they were not.

If the `−t` option is used with the `−v` option, the standard output format is:

`"%s %u/%u %u %s %d %d:%d %d %s\n"`, *<member mode>*, *<user ID>*, *<group ID>*, *<number of bytes in member>*, *<abbreviated month>*, *<day-of-month>*, *<hour>*, *<minute>*, *<year>*, *<file>*

Where:

file	shall be the operand specified on the command line, if *file* operands were specified, or the name of the file in the archive if they were not.
<*member mode*>	shall be formatted the same as the <*file mode*> string defined in 4.39.6.1 (Standard Output of `ls`), except that the first character, the <*entry type*>, is not used; the string represents the file mode of the archive member at the time it was added to, or replaced in, the archive.

The following represent the last-modification time of a file when it was most recently added to or replaced in the archive:

<*abbreviated month*>	shall be equivalent to the `%b` format in `date` (see 4.15).
<*day-of-month*>	shall be equivalent to the `%e` format in `date`.
<*hour*>	shall be equivalent to the `%H` format in `date`.
<*minute*>	shall be equivalent to the `%M` format in `date`.
<*year*>	shall be equivalent to the `%Y` format in `date`.

When **LC_TIME** does not specify the POSIX Locale, a different format and order of presentation of these fields relative to each other may be used in a format appropriate in the specified locale.

If the `−x` option is used with the `−v` option, the standard output format is:

`"x − %s\n"`, <*file*>

where *file* is the operand specified on the command line, if *file* operands were specified, or the name of the file in the archive if they were not.

6.1.6.2 Standard Error

Used only for diagnostic messages. The diagnostic message about creating a new archive when `−c` is not specified shall not modify the exit status.

6.1.6.3 Output Files

Archives are files with unspecified formats.

6.1.7 Extended Description

None.

6.1.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

6.1.9 Consequences of Errors

Default.

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

Examples, Usage

The archive format is not described. It is recognized that there are several known `ar` formats, which are not compatible. The `ar` utility is being included, however, to allow creation of archives that are intended for use only on the same machine. The archive file is specified as a file and it can be moved as a file. This does allow an archive to be moved from one machine to another machine that uses the same implementation of `ar`.

Utilities such as `pax` (and its forebears `tar` and `cpio`) also provide portable 1
''archives.'' This is a not a duplication; the `ar` interface is included in the standard to provide an interface primarily for `make` and the compilers, based on a historical model.

In historical implementations, the `−q` option is known to execute quickly because `ar` does not check whether the added members are already in the archive. This is useful to bypass the searching otherwise done when creating a large archive piece-by-piece. The remarks may or may not hold true for a brand-new POSIX.2 implementation; and hence, these remarks have been moved out of the specification and into the Rationale.

Likewise, historical implementations maintain a symbol table to speed searches, particularly when the archive contains object files. However, future implementors may or may not use a symbol table, and the `−s` option was removed from this clause to permit implementors freedom of choice. Instead, the requirement that archive libraries be suitable for link editing was added to ensure the intended functionality. Systems such as System V maintain the symbol table without requiring the use of `−s`, so adding `−s` (even if it were worded as allowing a no-op) would essentially require all portable applications to use it in all invocations involving libraries.

The Operands subclause requires what might seem to be true without specifying it: the archive cannot truncate the filenames below {NAME_MAX}. Some historical implementations do so, however, causing unexpected results for the application. Therefore, POSIX.2 makes the requirement explicit to avoid misunderstandings.

According to the System V documentation, the options `-dmpqrtx` are not required to begin with a hyphen (−). POSIX.2 requires that a conforming application use the leading hyphen.

When extracting files with long filenames into a file system that supports only shorter filenames, an undefined condition occurs. Typical implementation actions might be one of the following:

— Extract and truncate the filename only when an existing file would not be overlaid.

— Extract and truncate the filename and overlay an existing file only if some extension such as another command-line option were used to override this safety feature.

— Refuse to extract any files unless an extension overrode the default.

The archive format used by the 4.4BSD implementation is documented in the rationale as an example:

> A file created by `ar` begins with the ‘‘magic’’ string ‘‘`!<arch>\n`’’. The rest of the archive is made up of objects, each of which is composed of a header for a file, a possible filename, and the file contents. The header is portable between machine architectures, and, if the file contents are printable, the archive is itself printable.
>
> The header is made up of six ASCII fields, followed by a two-character trailer. The fields are the object name (16 characters), the file last modification time (12 characters), the user and group IDs (each 6 characters), the file mode (8 characters) and the file size (10 characters). All numeric fields are in decimal, except for the file mode, which is in octal.
>
> The modification time is the file *st_mtime* field. The user and group IDs are the file *st_uid* and *st_gid* fields. The file mode is the file *st_mode* field. The file size is the file *st_size* field. The two-byte trailer is the string ‘‘`‘<newline>`’’.
>
> Only the name field has any provision for overflow. If any filename is more than 16 characters in length or contains an embedded space, the string ‘`#1/`’’ followed by the ASCII length of the name is written in the name field. The file size (stored in the archive header) is incremented by the length of the name. The name is then written immediately following the archive header.
>
> Any unused characters in any of these fields are written as `<space>` characters. If any fields are their particular maximum number of characters in length, there will be no separation between the fields.
>
> Objects in the archive are always an even number of bytes long; files that are an odd number of bytes long are padded with a `<newline>` character, although the size in the header does not reflect this.

History of Decisions Made

The `ar` utility description requires that (when all its members are valid object files) `ar` produce an object code library, which the linkage editor can use to extract object modules. If the linkage editor needs a symbol table to permit random access to the archive, `ar` must provide it; however, `ar` does not require a symbol table. The historical `−m` and `−q` positioning options were omitted, as were the positioning modifiers formerly associated with the `−m` and `−r` options, because the two functions of positioning are handled by the `ranlib`-style (a utility found on some historical systems to create symbol tables within the archive) symbol tables and/or the ability of portable applications to create multiple archives instead of loading from a single archive.

Earlier drafts had elaborate descriptions in the Asynchronous Events subclause about how signals were caught and then resent to itself. These were removed in favor of the default case because they are essentially implementation details, unnecessary for the application. Similarly, information about where (and if) temporary files are created was removed from earlier drafts.

The BSD `−o` option was omitted. It is a rare portable application that will use `ar` to extract object code from a library with concern for its modification time, since this can only be of importance to `make`. Hence, since this functionality is not deemed important for applications portability, the modification time of the extracted files is set to the current time.

There is at least one known implementation (for a small computer) that can accommodate only object files for that system, disallowing mixed object and other files. The ability to handle any type of file is not only existing practice for most implementations, but is also a reasonable expectation.

Consideration was given to changing the output format of `ar −tv` to the same format as the output of `ls −l`. This would have made parsing the output of `ar` the same as that of `ls`. This was rejected in part because the current `ar` format is commonly used and changes would break existing usage. Second, `ar` gives the user ID and group ID in numeric format separated by a slash. Changing this to be the user name and group name would not be right if the archive were moved to a machine that contained a different user database. Since `ar` cannot know whether the archive file was generated on the same machine, it cannot tell what to report.

The text on the `−ur` option combination is historical practice—since one filename can easily represent two different files (e.g., `/a/foo` and `/b/foo`), it is reasonable to replace the member in the archive even when the modification time in the archive is identical to that in the file system.

6.2 `make` — Maintain, update, and regenerate groups of programs

6.2.1 Synopsis

```
make [−einpqrst][−f makefile ]... [ −k | −S ][macro=name ]...
     [target_name ... ]
```

6.2.2 Description

The `make` utility can be used as a part of software development to update files that are derived from other files. A typical case is one where object files are derived from the corresponding source files. The `make` utility examines time relationships and updates those derived files (called targets) that have modified times earlier than the modified times of the files (called prerequisites) from which they are derived. A description file (''makefile'') contains a description of the relationships between files, and the commands that must be executed to update the targets to reflect changes in their prerequisites. Each specification, or rule, shall consist of a target, optional prerequisites, and optional commands to be executed when a prerequisite is newer than the target. There are two types of rules:

— Inference rules, which have one target name with at least one period (`.`) and no slash (`/`)

— Target rules, which can have more than one target name

In addition, `make` shall have a collection of built-in macros and inference rules that infer prerequisite relationships to simplify maintenance of programs.

To receive exactly the behavior described in this clause, a portable makefile shall:

— Include the special target `.POSIX` (see 6.2.7.3)

— Omit any special target reserved for implementations (a leading period followed by uppercase letters) that has not been specified by this clause.

The behavior of `make` is unspecified if either or both of these conditions are not met.

6.2.3 Options

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

The following options shall be supported by the implementation:

`−e` Cause environment variables, including those with null values, to override macro assignments within makefiles.