IEEE P1003.2 Draft 11.2 – September 1991

destination, seemed just as inconsistent. The semantics of the *rename* () function were preserved as much as possible. For example, mv is not permitted to ‘‘rename’’ files to or from directories, even though they might be empty and removable.

Historic implementations of mv did not exit with a nonzero exit status if they were unable to duplicate any file characteristics when moving a file across file systems, nor did they write a diagnostic message for the user. The former behavior has been preserved to prevent scripts from breaking; a diagnostic message is now required, however, so that users are alerted that the file characteristics have changed.

The exact format of the interactive prompts is unspecified. Only the general nature of the contents of prompts are specified, because implementations may desire more descriptive prompts than those used on historical implementations. Therefore, an application not using the −f option or using the −i option relies on the system to provide the most suitable dialogue directly with the user, based on the behavior specified.

4.44 nohup — Invoke a utility immune to hangups

4.44.1 Synopsis

nohup *utility* [*argument* . . .]

4.44.2 Description

The nohup utility shall invoke the utility named by the *utility* operand with arguments supplied as the *argument* operands. At the time the named *utility* is invoked, the SIGHUP signal shall be set to be ignored.

If the standard output is a terminal, all output written by the named *utility* to its standard output shall be appended to the end of the file nohup.out in the current directory. If nohup.out cannot be created or opened for appending, the output shall be appended to the end of the file nohup.out in the directory specified by the **HOME** environment variable. If neither file can be created or opened for appending, *utility* shall not be invoked. If a file is created, the file’s permission bits shall be set to S_IRUSR | S_IWUSR instead of the default specified in 2.9.1.4.

If the standard error is a terminal, all output written by the named *utility* to its standard error shall be redirected to the same file descriptor as the standard output.

4.44.3 Options

None.

4.44.4 Operands

The following operands shall be supported by the implementation:

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

argument Any string to be supplied as an argument when invoking the utility named by the *utility* operand.

4.44.5 External Influences

4.44.5.1 Standard Input

None.

4.44.5.2 Input Files

None.

4.44.5.3 Environment Variables

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

HOME This variable shall determine the pathname of the user's home directory: if the output file `nohup.out` cannot be created in the current directory, the `nohup` utility shall use the directory named by **HOME** to create the file.

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.

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

4.44.5.4 Asynchronous Events

The `nohup` utility shall take the standard action for all signals (see 2.11.5.4), except that SIGHUP shall be ignored.

4.44.6 External Effects

4.44.6.1 Standard Output

If the standard output is not a terminal, the standard output of `nohup` shall be the standard output generated by the execution of the *utility* specified by the operands. Otherwise, nothing shall be written to the standard output.

4.44.6.2 Standard Error

If the standard output is a terminal, a message shall be written to the standard error, indicating the name of the file to which the output is being appended. The name of the file shall be either `nohup.out` or `$HOME/nohup.out`.

4.44.6.3 Output Files

If the standard output is a terminal, all output written by the named *utility* to the standard output and standard error is appended to the file `nohup.out`, which is created if it does not already exist.

4.44.7 Extended Description

None.

4.44.8 Exit Status

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

126 The utility specified by *utility* was found but could not be invoked. 1

127 An error occurred in the `nohup` utility or the utility specified by *utility* 1
could not be found. 1

Otherwise, the exit status of `nohup` shall be that of the utility specified by the *utility* operand.

4.44.9 Consequences of Errors

Default.

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

Examples, Usage

It is frequently desirable to apply `nohup` to pipelines or lists of commands. This can be done by placing pipelines and command lists in a single file; this file can then be invoked as a utility, and the `nohup` applies to everything in the file.

Alternatively, the following command can be used to apply `nohup` to a complex command:

`nohup sh -c '`*complex-command-line*`'`

The 4.3BSD version ignores SIGTERM and SIGHUP, and if `./nohup.out` cannot be used, it fails instead of trying to use `$HOME/nohup.out`.

The `command`, `env`, `nohup`, 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.

History of Decisions Made

The `csh` utility has a built-in version of `nohup` that acts differently than this.

The term *utility* is used, rather than *command*, to highlight the fact that shell compound commands, pipelines, special built-ins, etc., cannot be used directly. However, *utility* includes user application programs and shell scripts, not just the standard utilities.

Historical versions of the `nohup` utility use default file creation semantics. Some more recent versions use the permissions specified here as an added security precaution.

Some historical implementations ignore SIGQUIT in addition to SIGHUP; others ignore SIGTERM. An earlier draft allowed, but did not require, SIGQUIT to be ignored. Several members of the balloting group objected, saying that `nohup` should only modify the handling of SIGHUP as required by this specification.

4.45 od — Dump files in various formats

4.45.1 Synopsis

od [−v] [−A *address_base*] [−j *skip*] [−N *count*] [−t *type_string*] . . . [*file* . . .]

4.45.2 Description

The od utility shall write the contents of its input files to standard output in a user-specified format.

4.45.3 Options

The od utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that the order of presentation of the −t options is significant.

The following options shall be supported by the implementation:

−A *address_base*
: Specify the input offset base (see 4.45.7). The *address_base* option argument shall be a character. The characters d, o, and x shall specify that the offset base shall be written in decimal, octal, or hexadecimal, respectively. The character n shall specify that the offset shall not be written.

−j *skip*
: Jump over *skip* bytes from the beginning of the input. The od utility shall read or seek past the first *skip* bytes in the concatenated input files. If the combined input is not at least *skip* bytes long, the od utility shall write a diagnostic message to standard error and exit with a nonzero exit status.

: By default, the *skip* option-argument shall be interpreted as a decimal number. With a leading 0x or 0X, the offset shall be interpreted as a hexadecimal number; otherwise, with a leading 0, the offset shall be interpreted as an octal number. Appending the character b, k, or m to offset shall cause it to be interpreted as a multiple of 512, 1024, or 1 048 576 bytes, respectively.

−N *count*
: Format no more than *count* bytes of input. By default, *count* shall be interpreted as a decimal number. With a leading 0x or 0X, *count* shall be interpreted as a hexadecimal number; otherwise, with a leading 0, it shall be interpreted as an octal number. If *count* bytes of input (after successfully skipping, if −j *skip* is specified) are not available, it shall not be considered an error; the od utility shall format the input that is available.