IEEE P1003.2 Draft 11.2 − September 1991

History of Decisions Made

It was suggested that this utility cannot be used portably, since the format of the symbols is implementation defined. The POSIX.1 {8} working group could not achieve consensus on defining these formats in the underlying *uname*() function and there is no expectation that POSIX.2 would be any more successful. In any event, some applications may still find this historical utility of value. For example, the symbols could be used for system log entries or for comparison with operator or user input.

4.69 `uniq` — Report or filter out repeated lines in a file

4.69.1 Synopsis

`uniq [−c|−d|−u][−f` *fields* `][−s` *chars* `][`*input_file* `[`*output_file* `]]`

Obsolescent Version:

`uniq [−c|−d|−u][−`*n*`][+`*m*`][`*input_file* `[`*output_file* `]]`

4.69.2 Description

The `uniq` utility shall read an input file comparing adjacent lines, and write one copy of each input line on the output. The second and succeeding copies of repeated adjacent input lines shall not be written.

Repeated lines in the input shall not be detected if they are not adjacent.

4.69.3 Options

The `uniq` utility shall conform to the utility argument syntax guidelines described in 2.10.2; the obsolescent version does not, as one of the options begins with `+` and the −*m* and +*n* options do not have option letters.

The following options shall be supported by the implementation:

`−c` Precede each output line with a count of the number of times the line occurred in the input.

`−d` Suppress the writing of lines that are not repeated in the input.

`−f` *fields* Ignore the first *fields* fields on each input line when doing comparisons, where *fields* shall be a positive decimal integer. A field is the maximal string matched by the basic regular expresssion:

```
[[:blank:]]*[^[:blank:]]*
```

If the *fields* option-argument specifies more fields than appear on an input line, a null string shall be used for comparison.

`-s` *chars*	Ignore the first *chars* characters when doing comparisons, where *chars* shall be a positive decimal integer. If specified in conjunction with the `-f` option, the first *chars* characters after the first *fields* fields shall be ignored. If the *chars* option-argument specifies more characters than remain on an input line, a null string shall be used for comparison.
`-u`	Suppress the writing of lines that are repeated in the input.
$-n$	(Obsolescent.) Equivalent to `-f` *fields* with *fields* set to n.
$+m$	(Obsolescent.) Equivalent to `-s` *chars* with *chars* set to m.

4.69.4 Operands

The following operands shall be supported by the implementation:

input_file	A pathname of the input file. If the *input_file* operand is not specified, or if the *input_file* is –, the standard input shall be used.
output_file	A pathname of the output file. If the *output_file* operand is not specified, the standard output shall be used. The results are unspecified if the file named by *output_file* is the file named by *input_file* .

4.69.5 External Influences

4.69.5.1 Standard Input

The standard input shall be used only if no *input_file* operand is specified or if *input_file* is –. See Input Files.

4.69.5.2 Input Files

The input file shall be a text file.

4.69.5.3 Environment Variables

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

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 and input files) and which characters constitute a `<blank>` in the current locale.

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

4.69.5.4 Asynchronous Events

Default.

4.69.6 External Effects

4.69.6.1 Standard Output

The standard output shall be used only if no *output_file* operand is specified. See Output Files.

4.69.6.2 Standard Error

Used only for diagnostic messages.

4.69.6.3 Output Files

If the `−c` option is specified, the output file shall be empty or each line will be of the form:

`"%d %s"`, *<number of duplicates>* , *<line>*

otherwise, the output file will be empty or each line will be of the form:

`"%s"`, *<line>*

4.69.7 Extended Description

None.

4.69.8 Exit Status

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

0 The utility executed successfully.

>0 An error occurred.

4.69.9 Consequences of Errors

Default.

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

Examples, Usage

Some historical implementations have limited lines to be 1080 bytes in length, which will not meet the implied {LINE_MAX} limit.

The `sort` utility (see 4.58) can be used to cause repeated lines to be adjacent in the input file.

The following input file data (but flushed left) was used for a test series on `uniq`:

```
#01 foo0 bar0 foo1 bar1
#02 bar0 foo1 bar1 foo1
#03 foo0 bar0 foo1 bar1
#04
#05 foo0 bar0 foo1 bar1
#06 foo0 bar0 foo1 bar1
#07 bar0 foo1 bar1 foo0
```

What follows is a series of test invocations of the `uniq` utility that use a mixture of `uniq`'s options against the input file data. These tests verify the meaning of *adjacent* . The `uniq` utility views the input data as a sequence of strings delimited by `\n`. Accordingly, for the *fields* th member of the sequence, `uniq` interprets unique or repreated adjacent lines strictly relative to the *fields* +1th member.

This first example tests the line counting option, comparing each line of the input file data starting from the second field:

```
uniq −c −f 1 uniq_0I.t
  1 #01 foo0 bar0 foo1 bar1
  1 #02 bar0 foo1 bar1 foo0
  1 #03 foo0 bar0 foo1 bar1
  1 #04
  2 #05 foo0 bar0 foo1 bar1
  1 #07 bar0 foo1 bar1 foo0
```

The number `2`, prefixing the fifth line of output, signifies that the `uniq` utility detected a pair of repeated lines. Given the input data, this can only be true when `uniq` is run using the `−f 1` option (which causes `uniq` to ignore the first field on each input line).

The second example tests the option to suppress unique lines, comparing each line of the input file data starting from the second field:

```
uniq −d −f 1 uniq_0I.t
#05 foo0 bar0 foo1 bar1
```

This test suppresses repeated lines, comparing each line of the input file data starting from the second field:

```
uniq -u -f 1 uniq_0I.t
#01 foo0 bar0 foo1 bar1
#02 bar0 foo1 bar1 foo1
#03 foo0 bar0 foo1 bar1
#04
#07 bar0 foo1 bar1 foo0
```

This suppresses unique lines, comparing each line of the input file data starting from the third character:

```
uniq -d -s 2 uniq_0I.t
```

In the last example, the `uniq` utility found no input matching the above criteria.

History of Decisions Made

The `-f` and `-s` options were added to replace the obsolescent −*n* and +*m* options so that `uniq` could meet the syntax guidelines in an upward-compatible way.

The output specifications in Output Files do not show a terminating `<newline>` because they both specify <*line*>, which includes its own `<newline>` (because of the definition of *line*).

4.70 `wait` — Await process completion

4.70.1 Synopsis

`wait` [*pid* . . .]

4.70.2 Description

When an asynchronous list (see 3.9.3.1) is started by the shell, the process ID of the last command in each element of the asynchronous list shall become known in the current shell execution environment; see 3.12.

If the `wait` utility is invoked with no operands, it shall wait until all process IDs known to the invoking shell have terminated and exit with a zero exit status.

If one or more *pid* operands are specified that represent known process IDs, the `wait` utility shall wait until all of them have terminated. If one or more *pid* operands are specified that represent unknown process IDs, `wait` shall treat them as if they were known process IDs that exited with exit status 127. The exit status returned by the `wait` utility shall be the exit status of the process requested by the last *pid* operand.

The known process IDs are applicable only for invocations of `wait` in the current shell execution environment.