IEEE P1003.2a Draft 8 – December 1991

5.29 `time` — Time a simple command

5.29.1 Synopsis

`time [−p]` *utility* `[`*argument* ...`]`

5.29.2 Description

The `time` utility shall invoke the utility named by the *utility* operand with arguments supplied as the *argument* operands and write a message to standard error that lists timing statistics for the utility. The message shall include the following information:

— The elapsed (''real'') time between invocation of *utility* and its termination.

— The User CPU time, equivalent to the sum of the *tms_utime* and *tms_cutime* fields returned by the POSIX.1 {8} *times*() function for the process in which *utility* is executed.

— The System CPU time, equivalent to the sum of the *tms_stime* and *tms_cstime* fields returned by the POSIX.1 {8} *times*() function for the process in which *utility* is executed.

The precision of the timing shall be no less than the granularity defined for the size of the clock tick unit on the system, but the results shall be reported in terms of standard time units (e.g., 0.02 seconds, 00:00:00.02, 1m33.75s, 365.21 seconds), not numbers of clock ticks.

When `time` is used as part of a pipeline, the times reported are unspecified, 8
except when it is the sole command within a grouping command (see 3.9.4.1) in 8
that pipeline. For example, the commands on the left are unspecified; those on 8
the right report on utilities `a` and `c`, respectively. 8

```
time a | b | c          { time a } | b | c        8
a | b | time c          a | b | (time c)          8
```

5.29.3 Options

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

The following option shall be supported by the implementation:

`−p` Write the timing output to standard error in the format shown in 7
5.29.6.2. 7

5.29.4 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.

5.29.5 External Influences

5.29.5.1 Standard Input

None.

5.29.5.2 Input Files

None.

5.29.5.3 Environment Variables

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

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_NUMERIC This variable shall determine the locale for numeric formatting.

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

5.29.5.4 Asynchronous Events

Default.

5.29.6 External Effects

5.29.6.1 Standard Output

None.

5.29.6.2 Standard Error

The standard error shall be used to write the timing statistics described in 5.29.2. If `-p` is specified, the following format shall be used in the POSIX Locale:

`"real %f\nuser %f\nsys %f\n"`, *<real seconds>* , *<user seconds>* , *<system seconds>*

where each floating point number shall be expressed in seconds. The precision used may be less than the default six digits of `%f`, but shall be sufficiently precise to accommodate the size of the clock tick on the system (for example, if there were 60 clock ticks per second, at least two digits shall follow the radix character). The number of digits following the radix character shall be no less than one, even if this always results in a trailing zero. The implementation may append white space and additional information following the format shown here.

5.29.6.3 Output Files

None.

5.29.7 Extended Description

None.

5.29.8 Exit Status

If the *utility* utility is invoked, the exit status of `time` shall be the exit status of 7
utility; otherwise, the `time` utility shall exit with one of the following values: 7

1−125	An error occurred in the `time` utility. 7
126	The utility specified by *utility* was found but could not be invoked. 7
127	The utility specified by *utility* could not be found. 7

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

Examples, Usage

It is frequently desirable to apply `time` 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 `time` applies to everything in the file.

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

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

However, note that on some historical systems that do not implement job control, 8
not all of the processes in a pipeline have to be owned by the same parent process 8
(they are sometimes owned by `init`). Some of the process times in a pipeline are 8
therefore invisible to the POSIX.1 {8} *times*() function and to the `time` utility. 8

The `command`, `env`, `nice`, `nohup`, `time`, and `xargs` utilities have been specified to 7
use exit code 127 if an error occurs so that applications can distinguish ''failure to 7
find a utility'' from ''invoked utility exited with an error indication.'' The value 7
127 was chosen because it is not commonly used for other meanings; most utilities 7
use small values for ''normal error conditions'' and the values above 128 can be 7
confused with termination due to receipt of a signal. The value 126 was chosen in 7
a similar manner to indicate that the utility could be found, but not invoked. 7
Some scripts produce meaningful error messages differentiating the 126 and 127 7
cases. The distinction between exit codes 126 and 127 is based on KornShell 8
practice that uses 127 when all attempts to *exec* the utility fail with [ENOENT], 8
and uses 126 when any attempt to *exec* the utility fails for any other reason. 8

History of Decisions Made

The `time` utility was originally proposed for the base POSIX.2 standard, but it was rejected because it was not useful for portable applications:

- The underlying CPU definitions from POSIX.1 {8} are somewhat vague, so the numeric output could not be compared accurately between systems, or even between invocations.
- The creation of portable benchmark programs was outside the POSIX.2 scope of work.

However, `time` does fit in the scope of user portability. Human judgment can be applied to the analysis of the output and it could be very useful in hands-on debugging of applications or in providing subjective measures of system performance.

The default output format has been left unspecified because historical implementations differ greatly in their style of depicting the numbers. The `–p` was invented to provide scripts a common means of obtaining this information.

The KornShell `time` is shell reserved word that can be used to time an entire 8
pipeline, rather than just a simple command. The POSIX definition has been 8

worded to allow this implementation. Consideration was given to invalidating 8
this approach because of the historical model from the C-shell and System V shell. 8
However, since the System V `time` historically has not produced accurate results 8
in pipeline timing (because the constituent processes are not all owned by the 8
same parent process, as allowed by POSIX), it did not seem worthwhile to break 8
existing KornShell usage. 8

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.

5.30 `tput` — Change terminal characteristics

5.30.1 Synopsis

`tput [−T` *type* `]` *operand* ...

5.30.2 Description

The `tput` utility shall display terminal-dependent information. The manner in which this information is retrieved is unspecified. The information displayed shall clear the terminal screen, initialize the user's terminal, or reset the user's terminal, depending on the operand given. The exact consequences of displaying this information are unspecified.

5.30.3 Options

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

The following option shall be supported by the implementation:

`−T` *type* Indicate the type of terminal. If this option is not supplied and
the **TERM** variable is unset or null, an unspecified default termi- 8
nal type shall be used. The setting of *type* shall take precedence 8
over the value in **TERM**.

5.30.4 Operands

The following strings shall be supported as operands by the implementation in the POSIX Locale:

`clear` Display the clear-screen sequence.