The GNU `time` Command

Measuring Program Resource Usage

1.1 Using the `time` command

The format of the `time` command is:

```
time [-apvV] [-f format] [-o file] [--append] [--portability] [--verbose]
[--format=format] [--output-file=file] [--version]
command [arg...]
```

`time` first runs the program *command*. When *command* finishes, `time` displays information about resources used by *command*, on the standard error output by default. If *command* exits with non-zero status, `time` displays a warning message and the exit status.

`time` determines which information to display about the resources used by the *command* from a *format string* (see Section 1.2 [Format], page 2). If no format is specified on the command line, but the `TIME` environment variable is set, its value is used as the format. Otherwise, a default format built into `time` is used (see Section 1.2 [Format], page 2).

Options to `time` must appear on the command line before *command*. Anything on the command line after *command* is passed as arguments to *command*.

The long-named options can be introduced with '`+`' as well as '`--`', for compatibility with previous releases. Eventually support for '`+`' will be removed, because it is incompatible with the POSIX.2 standard.

'`-o file`'
'`--output-file=file`'
: Write the resource usage statistics to *file* instead of to the standard error stream. By default, this *overwrites* the file, destroying the file's previous contents. This option is useful for collecting information on interactive programs and programs that produce output on the standard error stream.

'`-a`'
'`--append`'
: *Append* the resource usage information to the output file instead of overwriting it. This option is only useful with the '`-o`' or '`--output-file`' option.

'`-f format`'
'`--format=format`'
: Use *format* as the format string that controls the output of `time`.

'`-p`'
'`--portability`'
: Use the following format string, for conformance with POSIX 1003.2:

```
real %e
user %U
sys %S
```

'`-v`'
'`--verbose`'
: Use the built-in verbose format (different from the built-in default format), which displays every available piece of information on the program's resource usage, on its own line with an English description of its meaning.

'-V'
'--version'
: Print the version number of time.

1.2 Formatting The Output

The *format string* controls the contents of the time output. The format string can be set using the '-f' or '--format' options, or a built-in verbose format can be selected using the '-v' or '--verbose' options; if they are not given, but the TIME environment variable is set, its value is used as the format string. Otherwise, a built-in default format is used. The default format is:

```
%Uuser %Ssystem %Eelapsed %PCPU (%Xtext+%Ddata %Mmax)k
%Iinputs+%Ooutputs (%Fmajor+%Rminor)pagefaults %Wswaps
```

The format string usually consists of *resource specifiers* interspersed with plain text. A percent sign ('%') in the format string causes the following character to be interpreted as a resource specifier, which is similar to the formatting characters in the C printf function.

A backslash ('\') introduces a *backslash escape*, which is translated into a single printing character upon output. '\t' outputs a tab character, '\n' outputs a newline, and '\\' outputs a backslash. A backslash followed by any other character outputs a question mark ('?') followed by a backslash, to indicate that an invalid backslash escape was given.

Other text in the format string is copied verbatim to the output. time always prints a newline after printing the resource usage information, so normally format strings do not end with a newline character (or '\n').

There are many resource specifications. Not all resources are measured by all versions of Unix, so some of the values might be reported as 0. Any character following a percent sign that is not listed in the table below causes a question mark ('?') to be output, followed by that character, to indicate that an invalid resource specifier was given.

The resource specification characters are:

'%'
: A literal '%'.

'C'
: The name and command line arguments of the command being timed.

'D'
: The average size of the process's unshared data area, in Kilobytes.

'E'
: The elapsed real (wall clock) time used by the process, in [hours:]minutes:seconds.microseconds.

'F'
: Number of major, or I/O-requiring, page faults that occurred while the process was running. These are faults where the page has actually migrated out of primary memory.

'I'
: Number of file system inputs by the process.

'K'
: The average total memory usage of the process, in Kilobytes.

'M'
: The maximum resident set size of the process during its lifetime, in Kilobytes.

'O'
: Number of file system outputs by the process.

'P'
: The percentage of the CPU that this job got. This is just user + system times divied by the total running time.

'R' Number of minor, or recoverable, page faults. These are pages that are not valid (so they fault) but which have not yet been claimed by other virtual pages. Thus the data in the page is still valid but the system tables must be updated.

'S' Total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds:microseconds.

'U' Total number of CPU-seconds that the process used directly (in user mode), in seconds:microseconds.

'W' Number of times the process was swapped out of main memory.

'X' Average amount of shared text in the process, in Kilobytes.

'Z' The system's page size, in bytes. This is a per-system constant, but varies between systems.

'c' Number of times the process was context-switched involuntarily (because the time slice expired).

'e' The elapsed real (wall clock) time used by the process, in seconds.microseconds.

'k' Number of signals delivered to the process.

'p' The average unshared stack size of the process, in Kilobytes.

'r' Number of socket messages received by the process.

's' Number of socket messages sent by the process.

't' The average resident set size of the process, in Kilobytes.

'w' Number of times that the program was context-swapped voluntarily, for instance while waiting for an I/O operation to complete.

'x' The exit status of the process.

The resource specification characters are a superset of those recognized by the `tcsh` builtin `time` command.

1.3 Examples

These examples assume that your command interpreter is `bash`.

To run the command '`wc /etc/hosts`' and show the default information:

```
time wc /etc/hosts
```

To run the command '`ls -Fs`' and show just the user, system, and total time:

```
time -f "\t%E real,\t%U user,\t%S sys" ls -Fs
```

To edit the file *bork* and have `time` append the elapsed time and number of signals to the file `log`, reading the format string from the environment variable `TIME`:

```
export TIME="\t%E,\t%k"
time -a -o log emacs bork
```

1.4 Accuracy

The elapsed time is not collected atomically with the execution of the program; as a result, in bizarre circumstances (if the `time` command gets stopped or swapped out in between when the program being timed exits and when `time` calculates how long it took to run), it could be much larger than the actual execution time.

When the running time of a command is very nearly zero, some values (e.g., the percentage of CPU used) may be reported as either zero (which is wrong) or a question mark.

Most information shown by `time` is derived from the `wait3` system call. The numbers are only as good as those returned by `wait3`. On systems that do not have a `wait3` call that returns status information, the `times` system call is used instead. However, it provides much less information than `wait3`, so on those systems `time` reports the majority of the resources as zero.

The '`%I`' and '`%O`' values are allegedly only "real" input and output and do not include those supplied by caching devices. The meaning of "real" I/O reported by '`%I`' and '`%O`' may be muddled for workstations, especially diskless ones.

Table of Contents