



Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'alarm.3p' command

\$ man alarm.3p

ALARM(3P) POSIX Programmer's Manual ALARM(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

alarm ? schedule an alarm signal

SYNOPSIS

```
#include <unistd.h>

unsigned alarm(unsigned seconds);
```

DESCRIPTION

The alarm() function shall cause the system to generate a SIGALRM signal for the process after the number of realtime seconds specified by seconds have elapsed. Processor scheduling delays may prevent the process from handling the signal as soon as it is generated.

If seconds is 0, a pending alarm request, if any, is canceled.

Alarm requests are not stacked; only one SIGALRM generation can be scheduled in this manner. If the SIGALRM signal has not yet been generated, the call shall result in rescheduling the time at which the SIGALRM signal is generated.

Interactions between alarm() and setitimer() are unspecified.

RETURN VALUE

If there is a previous alarm() request with time remaining, alarm() shall return a non-zero value that is the number of seconds until the previous request would have generated a SIGALRM signal. Otherwise, alarm() shall return 0.

ERRORS

The alarm() function is always successful, and no return value is reserved to indicate an error.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The fork() function clears pending alarms in the child process. A new process image created by one of the exec functions inherits the time left to an alarm signal in the image of the old process.

Application developers should note that the type of the seconds argument and the return value of alarm() is unsigned. That means that a Strictly Conforming POSIX System Interfaces Application cannot pass a value greater than the minimum guaranteed value for {UINT_MAX}, which the ISO C standard sets as 65535, and any application passing a larger value is restricting its portability. A different type was considered, but historical implementations, including those with a 16-bit int type, consistently use either unsigned or int.

Application developers should be aware of possible interactions when the same process uses both the alarm() and sleep() functions.

RATIONALE

Many historical implementations (including Version 7 and System V) allow an alarm to occur up to a second early. Other implementations allow alarms up to half a second or one clock tick early or do not allow them to occur early at all. The latter is considered most appropriate, since it gives the most predictable behavior, especially since the signal can always be delayed for an indefinite amount of time due to scheduling. Applications can thus choose the seconds argument as the minimum amount of time they wish to have elapse before the signal.

The term "realtime" here and elsewhere (`sleep()`, `times()`) is intended to mean "wall clock" time as common English usage, and has nothing to do with "realtime operating systems". It is in contrast to virtual time, which could be misinterpreted if just time were used.

In some implementations, including 4.3 BSD, very large values of the seconds argument are silently rounded down to an implementation-specific maximum value. This maximum is large enough (to the order of several months) that the effect is not noticeable.

There were two possible choices for alarm generation in multi-threaded applications: generation for the calling thread or generation for the process. The first option would not have been particularly useful since the alarm state is maintained on a per-process basis and the alarm that is established by the last invocation of `alarm()` is the only one that would be active.

Furthermore, allowing generation of an asynchronous signal for a thread would have introduced an exception to the overall signal model. This requires a compelling reason in order to be justified.

FUTURE DIRECTIONS

None.

SEE ALSO

`alarm()`, `exec`, `fork()`, `getitimer()`, `pause()`, `sigaction()`, `sleep()`

The Base Definitions volume of POSIX.1-2017, `<signal.h>`, `<unistd.h>`

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at <http://www.opengroup.org/unix/online.html>.

Any typographical or formatting errors that appear in this page are

most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.kernel.org/doc/man-pages/reporting_bugs.html.

IEEE/The Open Group

2017

ALARM(3P)