NAME
> time_namespaces – overview of Linux time namespaces

DESCRIPTION
> Time namespaces virtualize the values of two system clocks:

> • **CLOCK_MONOTONIC** (and likewise **CLOCK_MONOTONIC_COARSE** and **CLOCK_MONO-TONIC_RAW**), a nonsettable clock that represents monotonic time since—as described by POSIX—"some unspecified point in the past".

> • **CLOCK_BOOTTIME** (and likewise **CLOCK_BOOTTIME_ALARM**), a nonsettable clock that is identical to **CLOCK_MONOTONIC**, except that it also includes any time that the system is suspended.

> Thus, the processes in a time namespace share per-namespace values for these clocks. This affects various APIs that measure against these clocks, including: **clock_gettime**(2), **clock_nanosleep**(2), **nanosleep**(2), **timer_settime**(2), **timerfd_settime**(2), and */proc/uptime*.

> Currently, the only way to create a time namespace is by calling **unshare**(2) with the **CLONE_NEW-TIME** flag. This call creates a new time namespace but does *not* place the calling process in the new namespace. Instead, the calling process's subsequently created children are placed in the new namespace. This allows clock offsets (see below) for the new namespace to be set before the first process is placed in the namespace. The */proc/*pid*/ns/time_for_children* symbolic link shows the time namespace in which the children of a process will be created. (A process can use a file descriptor opened on this symbolic link in a call to **setns**(2) in order to move into the namespace.)

/proc/PID/timens_offsets
> Associated with each time namespace are offsets, expressed with respect to the initial time namespace, that define the values of the monotonic and boot-time clocks in that namespace. These offsets are exposed via the file */proc/PID/timens_offsets*. Within this file, the offsets are expressed as lines consisting of three space-delimited fields:

> ```
> <clock-id> <offset-secs> <offset-nanosecs>
> ```

> The *clock-id* is a string that identifies the clock whose offsets are being shown. This field is either *monotonic*, for **CLOCK_MONOTONIC**, or *boottime*, for **CLOCK_BOOTTIME**. The remaining fields express the offset (seconds plus nanoseconds) for the clock in this time namespace. These offsets are expressed relative to the clock values in the initial time namespace. The *offset-secs* value can be negative, subject to restrictions noted below; *offset-nanosecs* is an unsigned value.

> In the initial time namespace, the contents of the *timens_offsets* file are as follows:

> ```
> $ cat /proc/self/timens_offsets
> monotonic          0          0
> boottime           0          0
> ```

> In a new time namespace that has had no member processes, the clock offsets can be modified by writing newline-terminated records of the same form to the *timens_offsets* file. The file can be written to multiple times, but after the first process has been created in or has entered the namespace, **write**(2)s on this file fail with the error **EACCES**. In order to write to the *timens_offsets* file, a process must have the **CAP_SYS_TIME** capability in the user namespace that owns the time namespace.

> Writes to the *timens_offsets* file can fail with the following errors:

> **EINVAL**
>> An *offset-nanosecs* value is greater than 999,999,999.

> **EINVAL**
>> A *clock-id* value is not valid.

> **EPERM**
>> The caller does not have the **CAP_SYS_TIME** capability.

ERANGE

An *offset-secs* value is out of range. In particular;

- *offset-secs* can't be set to a value which would make the current time on the corresponding clock inside the namespace a negative value; and

- *offset-secs* can't be set to a value such that the time on the corresponding clock inside the namespace would exceed half of the value of the kernel constant **KTIME_SEC_MAX** (this limits the clock value to a maximum of approximately 146 years).

In a new time namespace created by **unshare**(2), the contents of the *timens_offsets* file are inherited from the time namespace of the creating process.

NOTES

Use of time namespaces requires a kernel that is configured with the **CONFIG_TIME_NS** option.

Note that time namespaces do not virtualize the **CLOCK_REALTIME** clock. Virtualization of this clock was avoided for reasons of complexity and overhead within the kernel.

For compatibility with the initial implementation, when writing a *clock-id* to the */proc/*pid*/timens_offsets* file, the numerical values of the IDs can be written instead of the symbolic names show above; i.e., 1 instead of *monotonic*, and 7 instead of *boottime*. For readability, the use of the symbolic names over the numbers is preferred.

The motivation for adding time namespaces was to allow the monotonic and boot-time clocks to maintain consistent values during container migration and checkpoint/restore.

EXAMPLES

The following shell session demonstrates the operation of time namespaces. We begin by displaying the inode number of the time namespace of a shell in the initial time namespace:

```
$ readlink /proc/$$/ns/time
time:[4026531834]
```

Continuing in the initial time namespace, we display the system uptime using **uptime**(1) and use the *clock_times* example program shown in **clock_getres**(2) to display the values of various clocks:

```
$ uptime --pretty
up 21 hours, 17 minutes
$ ./clock_times
CLOCK_REALTIME : 1585989401.971 (18356 days +  8h 36m 41s)
CLOCK_TAI      : 1585989438.972 (18356 days +  8h 37m 18s)
CLOCK_MONOTONIC:       56338.247 (15h 38m 58s)
CLOCK_BOOTTIME :       76633.544 (21h 17m 13s)
```

We then use **unshare**(1) to create a time namespace and execute a **bash**(1) shell. From the new shell, we use the built-in **echo** command to write records to the *timens_offsets* file adjusting the offset for the **CLOCK_MONOTONIC** clock forward 2 days and the offset for the **CLOCK_BOOTTIME** clock forward 7 days:

```
$ PS1="ns2# " sudo unshare -T -- bash --norc
ns2# echo "monotonic $((2*24*60*60)) 0" > /proc/$$/timens_offsets
ns2# echo "boottime  $((7*24*60*60)) 0" > /proc/$$/timens_offsets
```

Above, we started the **bash**(1) shell with the **−−norc** option so that no start-up scripts were executed. This ensures that no child processes are created from the shell before we have a chance to update the *timens_offsets* file.

We then use **cat**(1) to display the contents of the *timens_offsets* file. The execution of **cat**(1) creates the first process in the new time namespace, after which further attempts to update the *timens_offsets* file produce an error.

```
ns2# cat /proc/$$/timens_offsets
monotonic        172800           0
```

```
        boottime      604800           0
ns2# echo "boottime $((9*24*60*60)) 0" > /proc/$$/timens_offsets
bash: echo: write error: Permission denied
```

Continuing in the new namespace, we execute **uptime**(1) and the *clock_times* example program:

```
ns2# uptime --pretty
up 1 week, 21 hours, 18 minutes
ns2# ./clock_times
CLOCK_REALTIME : 1585989457.056 (18356 days +  8h 37m 37s)
CLOCK_TAI      : 1585989494.057 (18356 days +  8h 38m 14s)
CLOCK_MONOTONIC:     229193.332 (2 days + 15h 39m 53s)
CLOCK_BOOTTIME :     681488.629 (7 days + 21h 18m  8s)
```

From the above output, we can see that the monotonic and boot-time clocks have different values in the new time namespace.

Examining the */proc/*pid*/ns/time* and */proc/*pid*/ns/time_for_children* symbolic links, we see that the shell is a member of the initial time namespace, but its children are created in the new namespace.

```
ns2# readlink /proc/$$/ns/time
time:[4026531834]
ns2# readlink /proc/$$/ns/time_for_children
time:[4026532900]
ns2# readlink /proc/self/ns/time   # Creates a child process
time:[4026532900]
```

Returning to the shell in the initial time namespace, we see that the monotonic and boot-time clocks are unaffected by the *timens_offsets* changes that were made in the other time namespace:

```
$ uptime --pretty
up 21 hours, 19 minutes
$ ./clock_times
CLOCK_REALTIME : 1585989401.971 (18356 days +  8h 38m 51s)
CLOCK_TAI      : 1585989438.972 (18356 days +  8h 39m 28s)
CLOCK_MONOTONIC:      56338.247 (15h 41m  8s)
CLOCK_BOOTTIME :      76633.544 (21h 19m 23s)
```

SEE ALSO

nsenter(1), **unshare**(1), **clock_settime**(2), **setns**(2), **unshare**(2), **namespaces**(7), **time**(7)