Timers

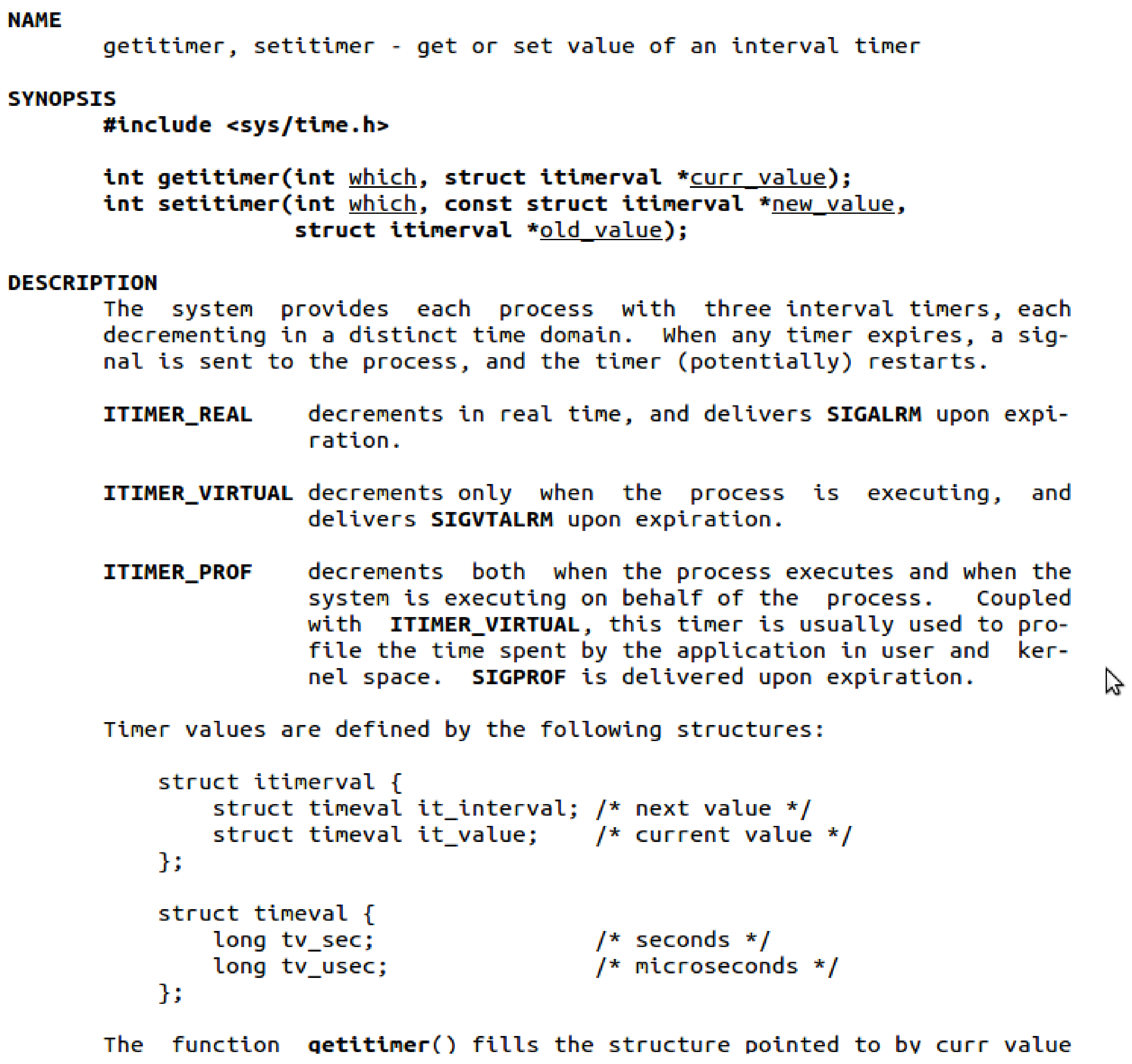
Class Notes

Timers

Timers are events delivered to a process at specific time intervals that allows a process to execute periodic or specific work on a time out.
Linux supports two interfaces:

- BSD timer API
- POSIX timer

BSD timer:



```
NAME
       getitimer, setitimer - get or set value of an interval timer

SYNOPSIS
       #include <sys/time.h>

       int getitimer(int which, struct itimerval *curr_value);
       int setitimer(int which, const struct itimerval *new_value,
                     struct itimerval *old_value);

DESCRIPTION
       The  system  provides  each  process  with  three interval timers, each
       decrementing in a distinct time domain.  When any timer expires, a sig-
       nal is sent to the process, and the timer (potentially) restarts.

       ITIMER_REAL    decrements in real time, and delivers SIGALRM upon expi-
                      ration.

       ITIMER_VIRTUAL decrements only  when  the  process  is  executing,  and
                      delivers SIGVTALRM upon expiration.

       ITIMER_PROF    decrements  both  when the process executes and when the
                      system is executing on behalf of the  process.   Coupled
                      with  ITIMER_VIRTUAL, this timer is usually used to pro-
                      file the time spent by the application in user and  ker-
                      nel space.  SIGPROF is delivered upon expiration.

       Timer values are defined by the following structures:

           struct itimerval {
               struct timeval it_interval; /* next value */
               struct timeval it_value;    /* current value */
           };

           struct timeval {
               long tv_sec;                /* seconds */
               long tv_usec;               /* microseconds */
           };

       The  function  getitimer() fills the structure pointed to by curr_value
```

Sample code:

```
struct  itimerval  itv;
itv.it_value.tv_sec = 1;
 itv.it_value.tv_usec = 10;
 itv.it_interval.tv_sec = 1;
itv.it_interval.tv_usec = 0;
settimer (ITIMER_REAL ,& sa ,NULL);
```

Limitations:

- There is a possibility of missing timer events when application's periodic response takes undeterministic time. [This can fix by setting the handler SA_NODEFER flag]
- Applications designed to use multiple timers will have to be coded to verify timer elapsed before executing response [since all timers generate SIGALRM] .

POSIX timer:

Posix real time timers are optimized to suit the needs of huge applications which may use multiple timers. The following facilities are provided by Posix timer interface:

- Each timer registered by the application is identified using unique ID.
- Application can customize and choose signal to be delivered to notify expiration of timer (need not be SIGALRM).
- Applications can also program for a user level thread to be executed in response to a timer event.
- Applications can assign pre created user level threads in response to timer events.
- Posix interface supports nano seconds resolution.
- API is provided to find information of lost events.

```
#include <signal.h>
#include <time.h>

int timer_create(clockid_t clockid, struct sigevent *sevp,
     timer_t *timerid);
```

Link with -lrt.

timer_create() creates a new per-process interval timer. The ID of the new timer is returned in the buffer pointed to by timerid, which must be a non NULL pointer. This ID is unique within the process, until the timer is deleted. The new timer is initially disarmed.

The clockid argument specifies the clock that the new timer uses to measure time. It can be specified as one of the following values:

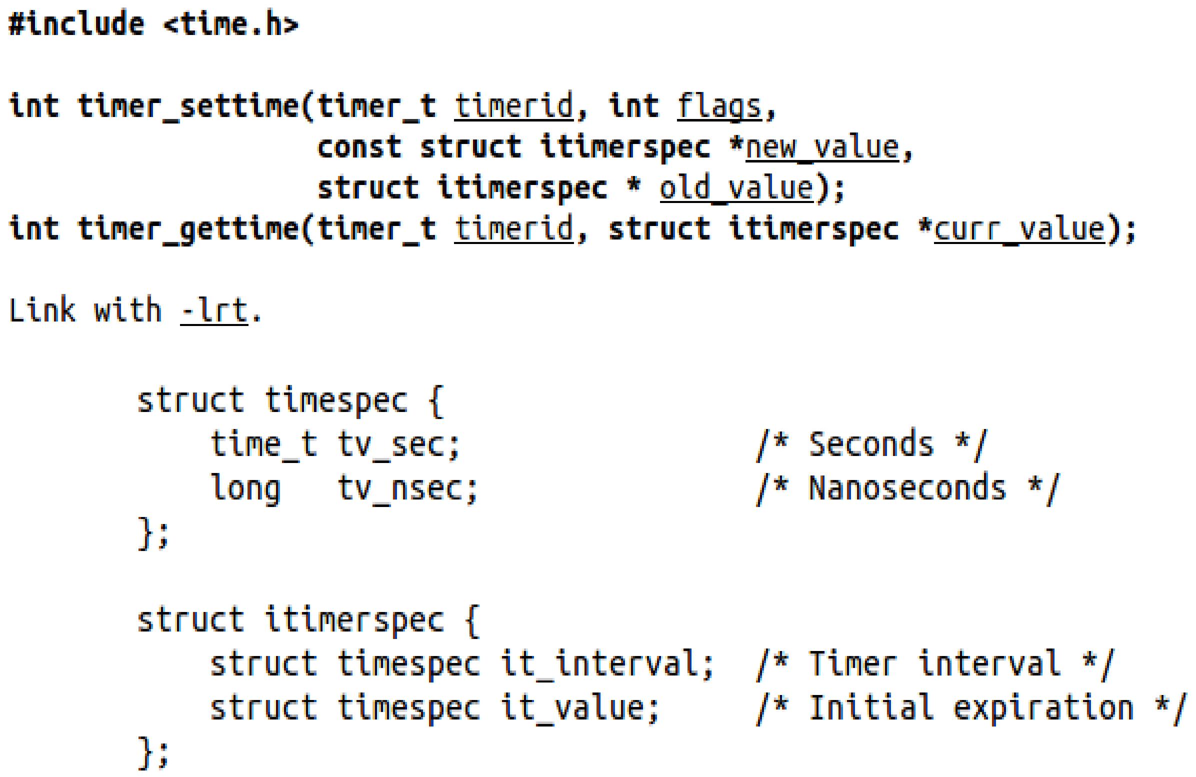
CLOCK_REALTIME
A settable system-wide real-time clock.

CLOCK_MONOTONIC
A nonsettable monotonically increasing clock that measures time from some unspecified point in the past that does not change after system startup.

CLOCK_PROCESS_CPUTIME_ID (since Linux 2.6.12)
A clock that measures (user and system) CPU time consumed by (all of the threads) in the calling process.

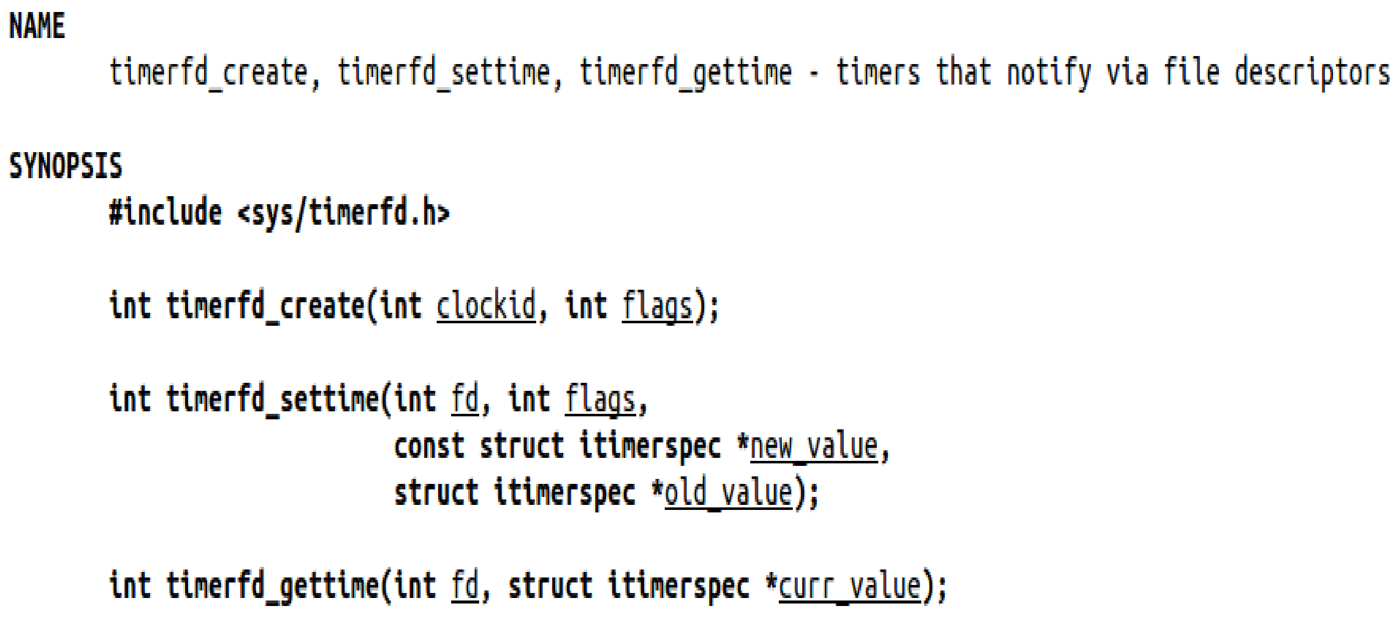
CLOCK_THREAD_CPUTIME_ID (since Linux 2.6.12)
A clock that measures (user and system) CPU time consumed by the calling thread.

Step 2 :



```
#include <time.h>

int timer_settime(timer_t timerid, int flags,
                  const struct itimerspec *new_value,
                  struct itimerspec * old_value);
int timer_gettime(timer_t timerid, struct itimerspec *curr_value);

Link with -lrt.

       struct timespec {
           time_t tv_sec;                /* Seconds */
           long   tv_nsec;               /* Nanoseconds */
       };

       struct itimerspec {
           struct timespec it_interval;  /* Timer interval */
           struct timespec it_value;     /* Initial expiration */
       };
```

- Linux kernel provides a preparatory timer interface that is mapped to a file descriptor.
- Applications designed to wait for timers to elapse and execute response work in the main thread can choose to use this interface.



NAME

timerfd_create, timerfd_settime, timerfd_gettime - timers that notify via file descriptors

SYNOPSIS

```
#include <sys/timerfd.h>

int timerfd_create(int clockid, int flags);

int timerfd_settime(int fd, int flags,
                    const struct itimerspec *new_value,
                    struct itimerspec *old_value);

int timerfd_gettime(int fd, struct itimerspec *curr_value);
```