POSIX timers

How

System Calls

Compile with `cc -lrt` to link against real-time library

System calls

- `timer_create()`
- `timer_settime()`
- `timer_gettime()`
- `timer_getoverrun()`
- `timer_delete()`

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

Creates a timer.

- ***clockid*** specifies **clock used for measuring time**; typical is `CLOCK_REALTIME` (wallclock time).
- ***timerid*** **returns unique timer ID** (integer).
- ***evp*** specifies **how process should be notified** when timer expires.

```
struct sigevent {   /* Abridged structure defn */
    int          sigev_notify;  /* Notification method */
    int          sigev_signo;   /* Timer expiration signal */
    union sigval sigev_value;   /* Value passed to signal
                                   handler or thread function */
    void (*sigev_notify_function) (union sigval);
                                /* Thread notification function */
    ...
};
```

- **Various notification methods** possible, e.g., `SIGEV_SIGNAL` (send signal) or `SIGEV_THREAD` (call a function in a new thread).
- ***sigev_value*** specifies **data** to be sent to signal handler, or passed to thread function.

```
union sigval {
    int   sival_int;   /* Integer value for accompanying data */
    void *sival_ptr;   /* Pointer value for accompanying data */
};
```

```
int timer_settime(timer_t timerid, int flags,
      const struct itimerspec *value, struct itimerspec *oldvalue);
```

- Starts timer ***timerid***.
- ***value*** specifies an **initial timer expiration**, and a **repeat interval**:

```
struct itimerspec {
    struct timespec it_interval;   /* Interval for periodic timer */
    struct timespec it_value;      /* First expiration */
};

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

- ***oldvalue*** returns previous timer settings.
- By default (***flags == 0***), timer is **relative**.
- Specify ***flags*** as **TIMER_ABSTIME** for **absolute** timer (measured since *Epoch*).

```
int timer_getoverrun(timer_t timerid)
```

Returns timer overrun value.

- Useful because timer might expire multiple times before signal is delivered or thread function is started.
- Reset to 0 each time we receive the timer signal, or thread function is started.

```
int timer_gettime(timer_t timerid, struct itimerspec *value)
```

Retrieve current settings of a timer.

```
int timer_delete(timer_t timerid)
```

Delete a timer, allowing the associated resources to be re-used.