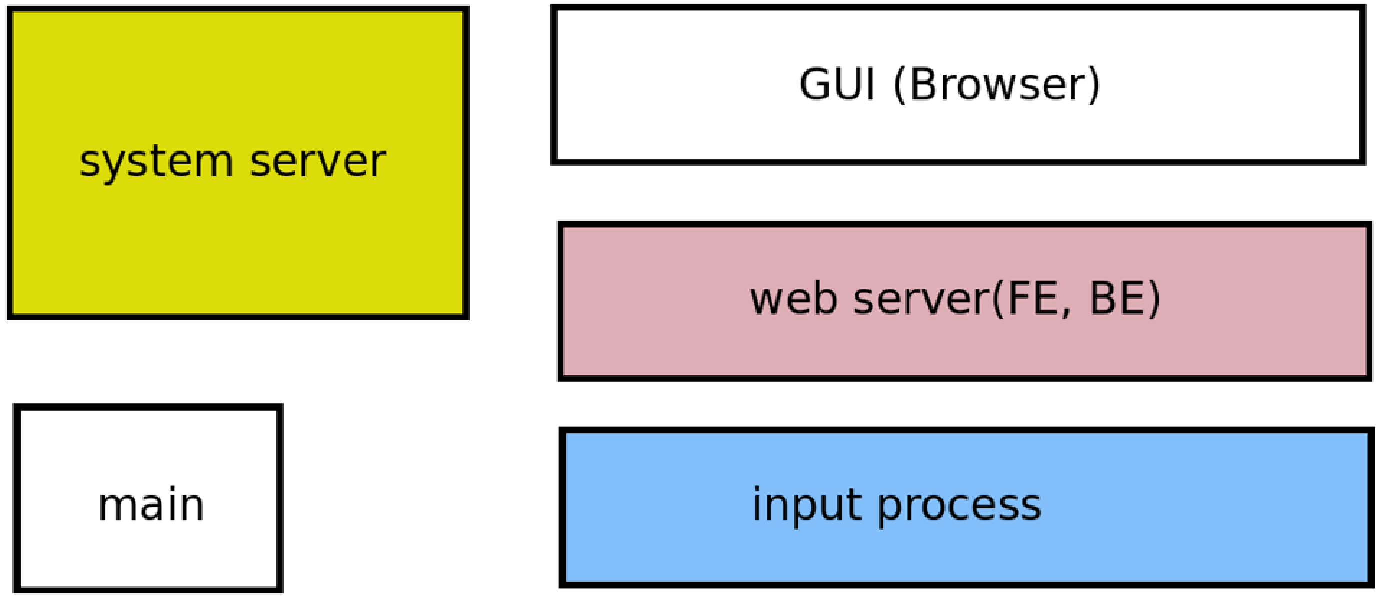
타이머

지난 토이 프로젝트 시그널 과제



타이머

- 설정한 시간에 시그널 발생 -> 핸들러 호출
- 현재 진행 중인 작업을 멈추고 타이머 시그널 핸들러 호출
- SW 개발에 굉장히 유용하게 활용됨
  - 딜레이(슬립), 시간 제어, 뒷처리 등등

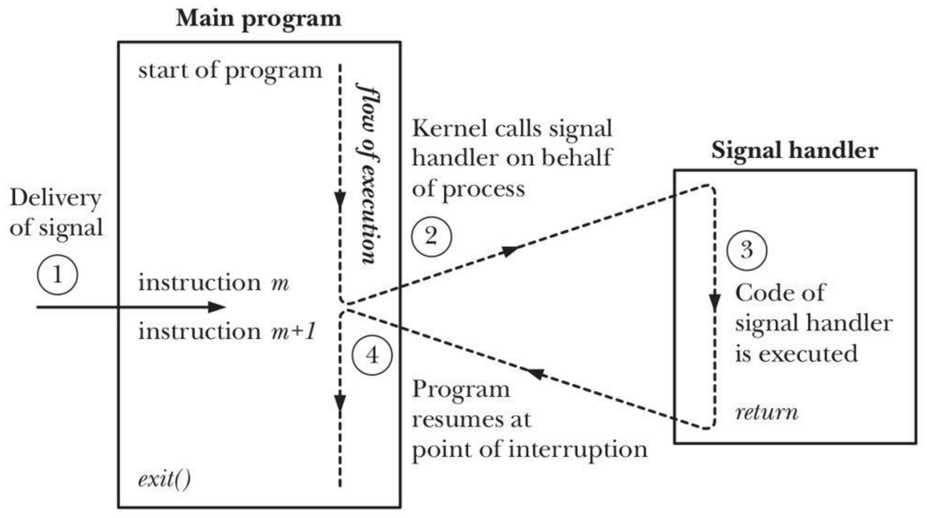
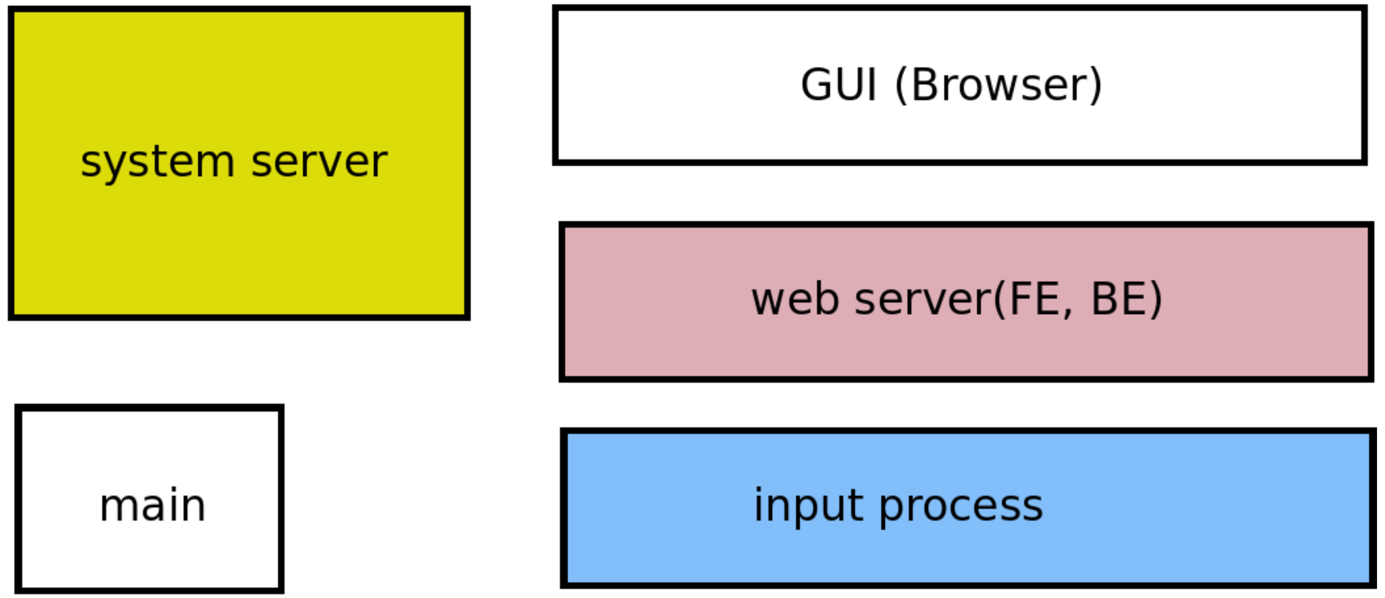

Main program
start of program
flow of execution
Kernel calls signal handler on behalf of process
Signal handler
Delivery of signal
1
2
3
instruction m
instruction m+1
4
Code of signal handler is executed
Program resumes at point of interruption
return
exit()

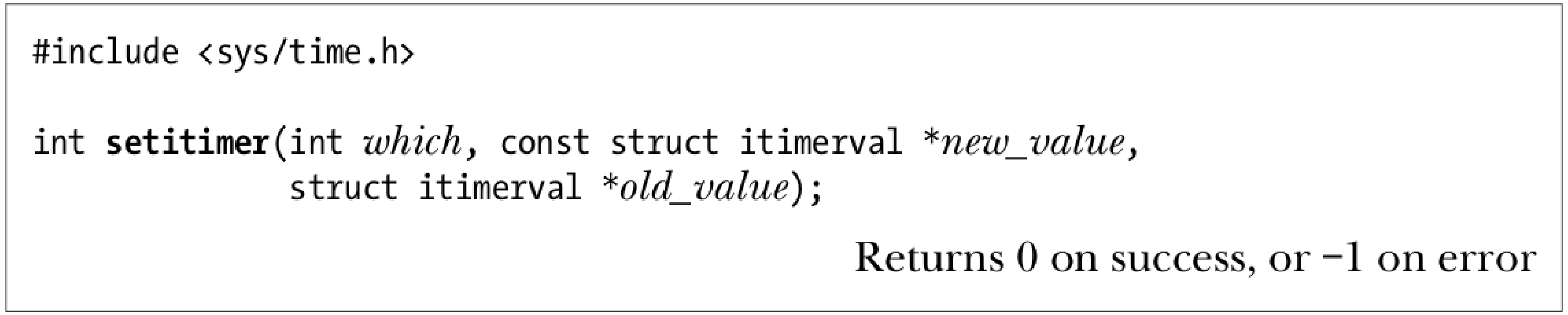
Figure 20-1: Signal delivery and handler execution

토이 프로젝트 - 1 sec 타이머 추가



시간 간격 타이머

- 전통적인 UNIX API
  - setitimer() and alarm()

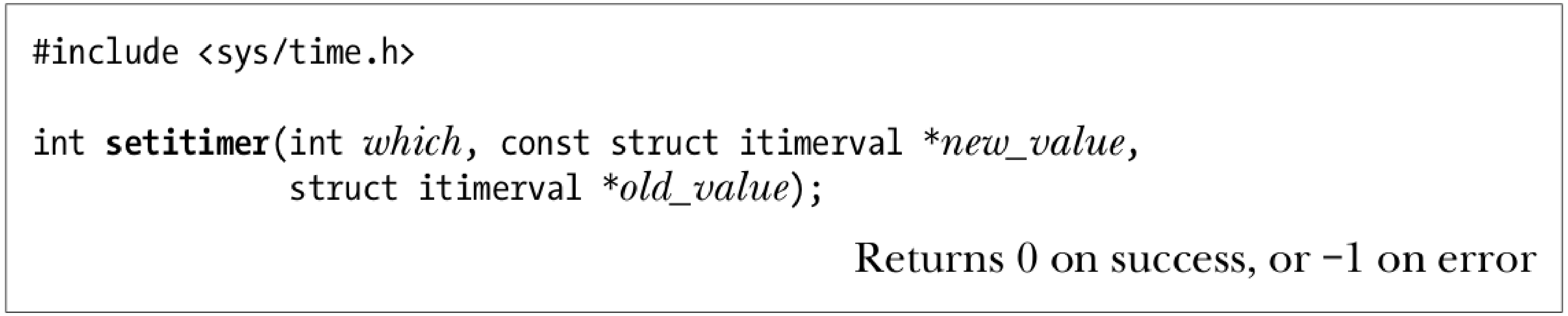


```
#include <sys/time.h>

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

Returns 0 on success, or –1 on error

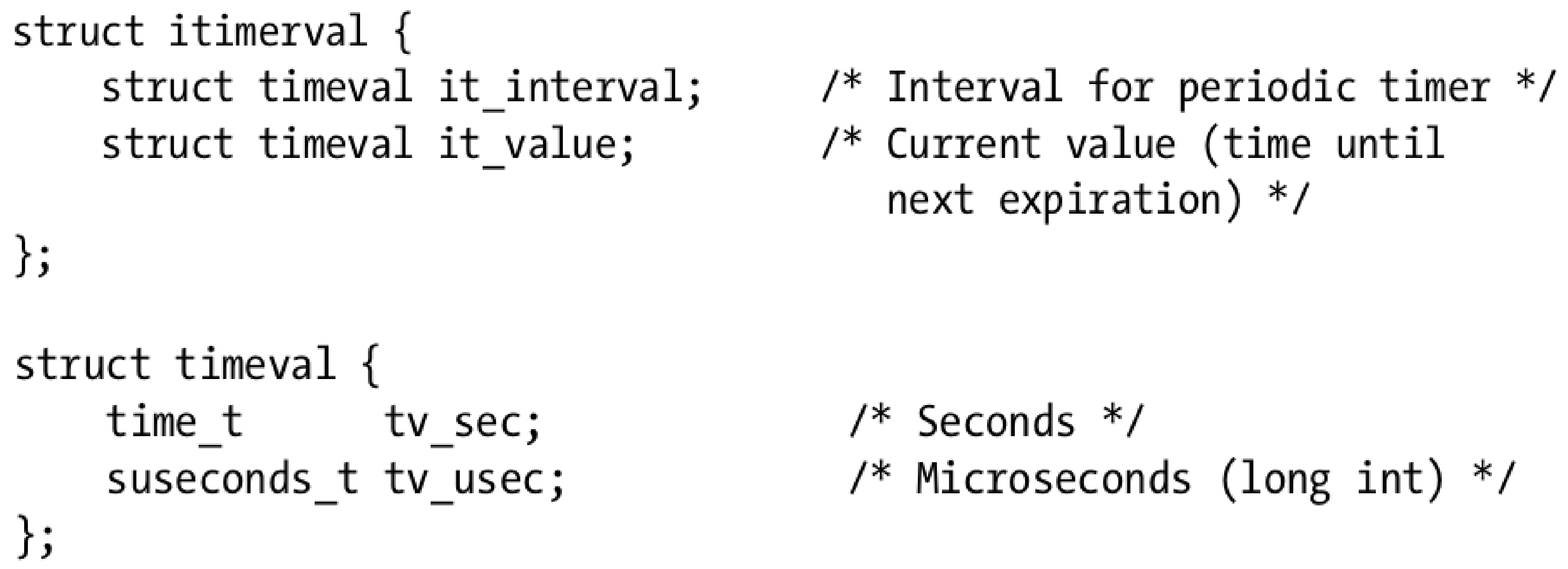
- which
  - ITIMER_REAL
  - ITIMER_VIRTUAL
  - ITIMER_PROF

setitimer



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

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

Returns 0 on success, or –1 on error



```
struct itimerval {
    struct timeval it_interval;     /* Interval for periodic timer */
    struct timeval it_value;        /* Current value (time until
                                       next expiration) */
};

struct timeval {
    time_t      tv_sec;             /* Seconds */
    suseconds_t tv_usec;            /* Microseconds (long int) */
};
```

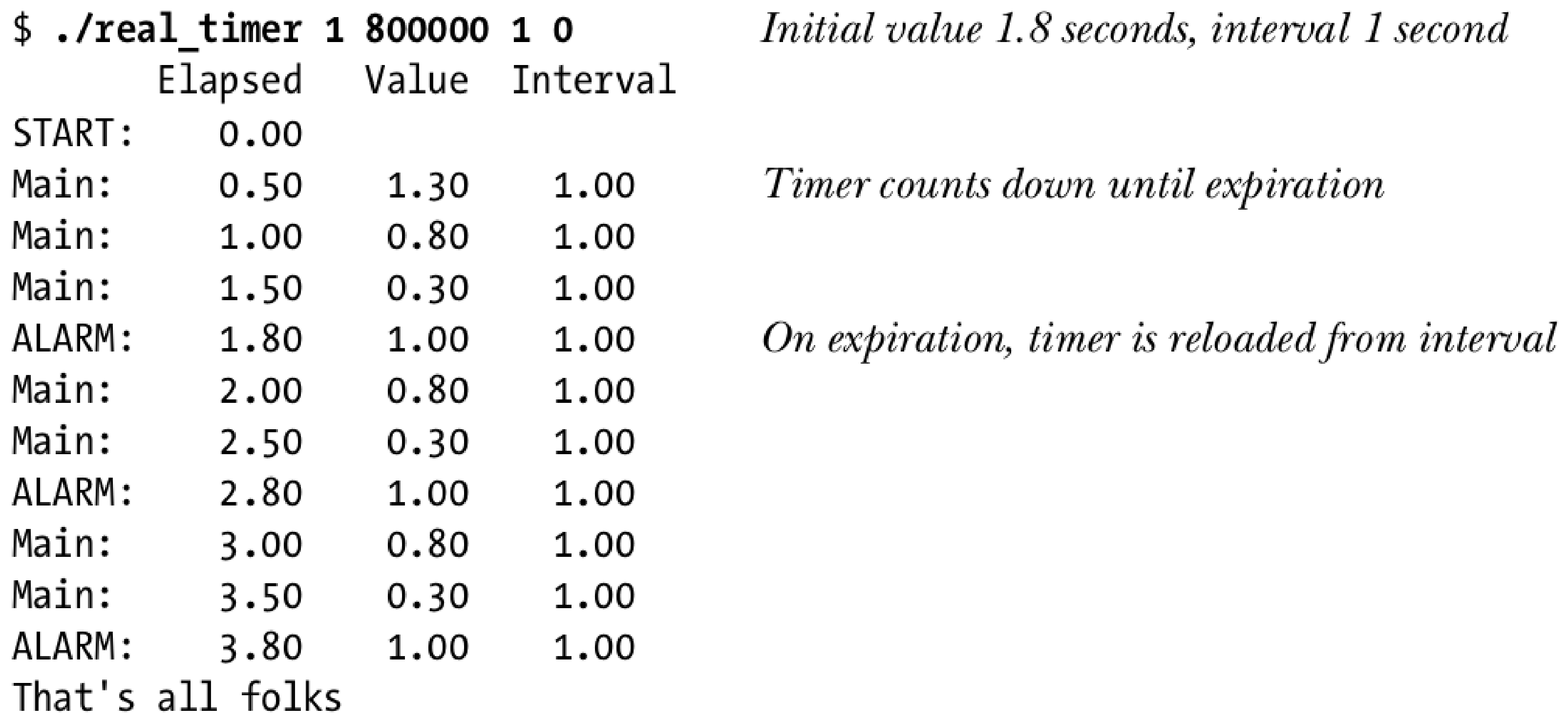
getitimer

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

int getitimer(int which, struct itimerval *curr_value);
```

Returns 0 on success, or −1 on error

```
struct itimerval {
    struct timeval it_interval;     /* Interval for periodic timer */
    struct timeval it_value;        /* Current value (time until
                                       next expiration) */
};
```

real_timer



```
$ ./real_timer 1 800000 1 0          Initial value 1.8 seconds, interval 1 second
       Elapsed   Value  Interval
START:    0.00
Main:     0.50    1.30    1.00       Timer counts down until expiration
Main:     1.00    0.80    1.00
Main:     1.50    0.30    1.00
ALARM:    1.80    1.00    1.00       On expiration, timer is reloaded from interval
Main:     2.00    0.80    1.00
Main:     2.50    0.30    1.00
ALARM:    2.80    1.00    1.00
Main:     3.00    0.80    1.00
Main:     3.50    0.30    1.00
ALARM:    3.80    1.00    1.00
That's all folks
```

real_timer

```
static volatile sig_atomic_t gotAlarm = 0;
                        /* Set nonzero on receipt of SIGALRM */


static void
sigalrmHandler(int sig)
{
②    gotAlarm = 1;
}
```

```
int
main(int argc, char *argv[])
{
    struct itimerval itv;
    clock_t prevClock;
    int maxSigs;                /* Number of signals to catch before exiting */
    int sigCnt;                 /* Number of signals so far caught */
    struct sigaction sa;

    if (argc > 1 && strcmp(argv[1], "--help") == 0)
        usageErr("%s [secs [usecs [int-secs [int-usecs]]]]\n", argv[0]);

    sigCnt = 0;

    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
    sa.sa_handler = sigalrmHandler;
③   if (sigaction(SIGALRM, &sa, NULL) == -1)
        errExit("sigaction");

    /* Exit after 3 signals, or on first signal if interval is 0 */

    maxSigs = (itv.it_interval.tv_sec == 0 &&
                itv.it_interval.tv_usec == 0) ? 1 : 3;

    displayTimes("START:", FALSE);

    /* Set timer from the command-line arguments */

    itv.it_value.tv_sec = (argc > 1) ? getLong(argv[1], 0, "secs") : 2;
    itv.it_value.tv_usec = (argc > 2) ? getLong(argv[2], 0, "usecs") : 0;
```

```
    itv.it_interval.tv_sec = (argc > 3) ? getLong(argv[3], 0, "int-secs") : 0;
    itv.it_interval.tv_usec = (argc > 4) ? getLong(argv[4], 0, "int-usecs") : 0;

④   if (setitimer(ITIMER_REAL, &itv, 0) == -1)
        errExit("setitimer");

    prevClock = clock();
    sigCnt = 0;

⑤   for (;;) {

        /* Inner loop consumes at least 0.5 seconds CPU time */

        while (((clock() - prevClock) * 10 / CLOCKS_PER_SEC) < 5) {
⑥           if (gotAlarm) {                     /* Did we get a signal? */
                gotAlarm = 0;
                displayTimes("ALARM:", TRUE);

                sigCnt++;
⑦               if (sigCnt >= maxSigs) {
                    printf("That's all folks\n");
                    exit(EXIT_SUCCESS);
                }
            }
        }

        prevClock = clock();
        displayTimes("Main: ", TRUE);
    }
}
```

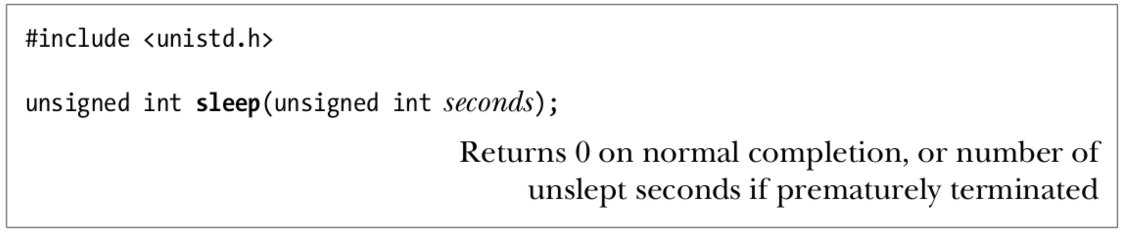
timers/real_timer.c

real_timer

```
    static void
①  displayTimes(const char *msg, Boolean includeTimer)
    {
        struct itimerval itv;
        static struct timeval start;
        struct timeval curr;
        static int callNum = 0;             /* Number of calls to this function */

        if (callNum == 0)                   /* Initialize elapsed time meter */
            if (gettimeofday(&start, NULL) == -1)
                errExit("gettimeofday");

        if (callNum % 20 == 0)              /* Print header every 20 lines */
            printf("       Elapsed   Value Interval\n");

          if (gettimeofday(&curr, NULL) == -1)
              errExit("gettimeofday");
          printf("%-7s %6.2f", msg, curr.tv_sec - start.tv_sec +
                             (curr.tv_usec - start.tv_usec) / 1000000.0);

          if (includeTimer) {
              if (getitimer(ITIMER_REAL, &itv) == -1)
                  errExit("getitimer");
              printf("  %6.2f  %6.2f",
                      itv.it_value.tv_sec + itv.it_value.tv_usec / 1000000.0,
                      itv.it_interval.tv_sec + itv.it_interval.tv_usec / 1000000.0);
          }

          printf("\n");
          callNum++;
      }
```

Sleep

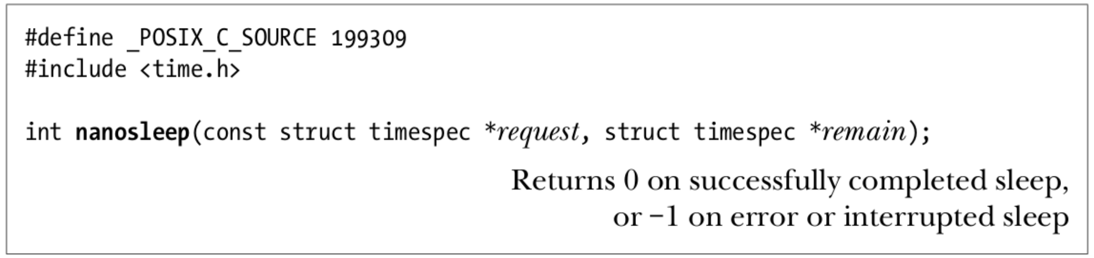
- 저해상도 수면
  - sleep()



```
#include <unistd.h>

unsigned int sleep(unsigned int seconds);
```

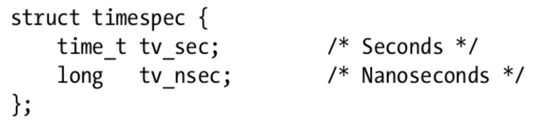
Returns 0 on normal completion, or number of
unslept seconds if prematurely terminated

- 고해상도 수면
  - nanosleep()



```
#define _POSIX_C_SOURCE 199309
#include <time.h>

int nanosleep(const struct timespec *request, struct timespec *remain);
```

Returns 0 on successfully completed sleep,
or −1 on error or interrupted sleep

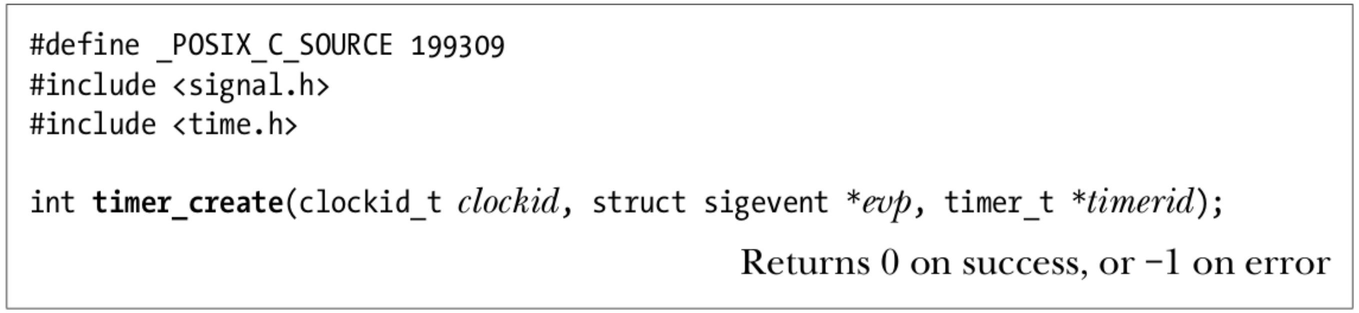


```
struct timespec {
    time_t tv_sec;          /* Seconds */
    long   tv_nsec;         /* Nanoseconds */
};
```

POSIX Interval 타이머

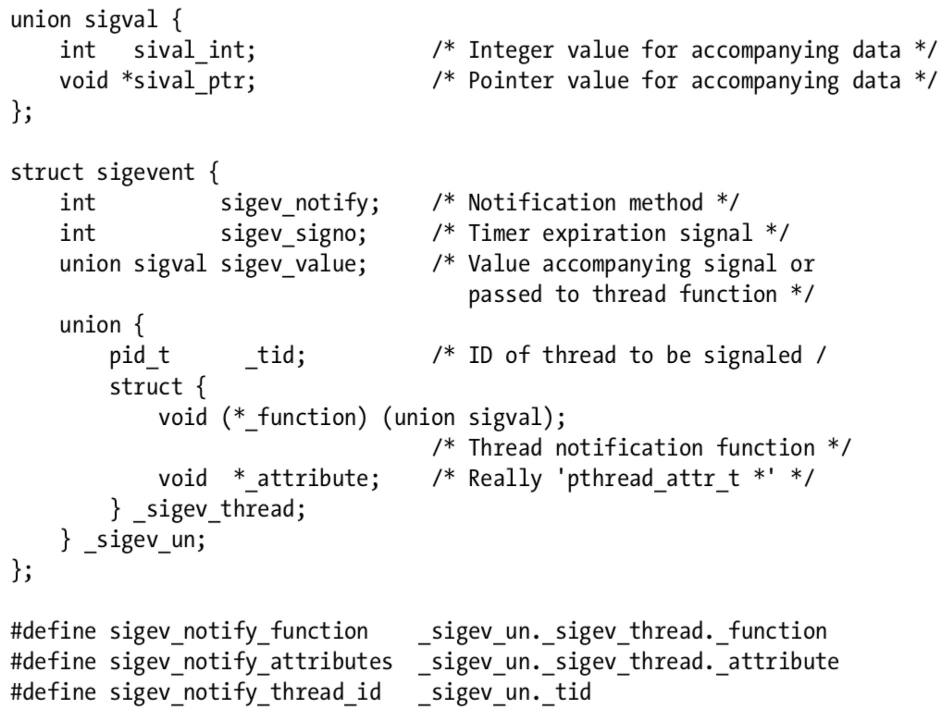
- setitimer의 문제점
  - 타이머 만료를 전달받는 유일한 방법이 시그널
  - 시그널이 수행 중 타이머 만료가 여러번이면?
    - 무시 - 타이머 오버런(timer overrun) 발생
    - 방법 없음

POSIX Interval 타이머



```
#define _POSIX_C_SOURCE 199309
#include <signal.h>
#include <time.h>

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

Returns 0 on success, or –1 on error



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

struct sigevent {
    int          sigev_notify;    /* Notification method */
    int          sigev_signo;     /* Timer expiration signal */
    union sigval sigev_value;     /* Value accompanying signal or
                                     passed to thread function */
    union {
        pid_t      _tid;          /* ID of thread to be signaled /
        struct {
            void (*_function) (union sigval);
                                  /* Thread notification function */
            void  *_attribute;    /* Really 'pthread_attr_t *' */
        } _sigev_thread;
    } _sigev_un;
};

#define sigev_notify_function    _sigev_un._sigev_thread._function
#define sigev_notify_attributes  _sigev_un._sigev_thread._attribute
#define sigev_notify_thread_id   _sigev_un._tid
```

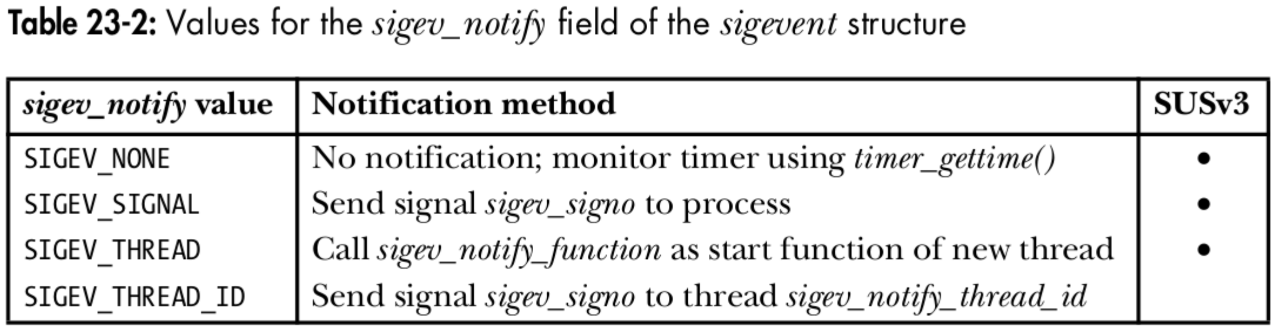


Table 23-2: Values for the *sigev_notify* field of the *sigevent* structure

sigev_notify value	Notification method	SUSv3
SIGEV_NONE	No notification; monitor timer using *timer_gettime()*	•
SIGEV_SIGNAL	Send signal *sigev_signo* to process	•
SIGEV_THREAD	Call *sigev_notify_function* as start function of new thread	•
SIGEV_THREAD_ID	Send signal *sigev_signo* to thread *sigev_notify_thread_id*	

타이머 시작과 중지

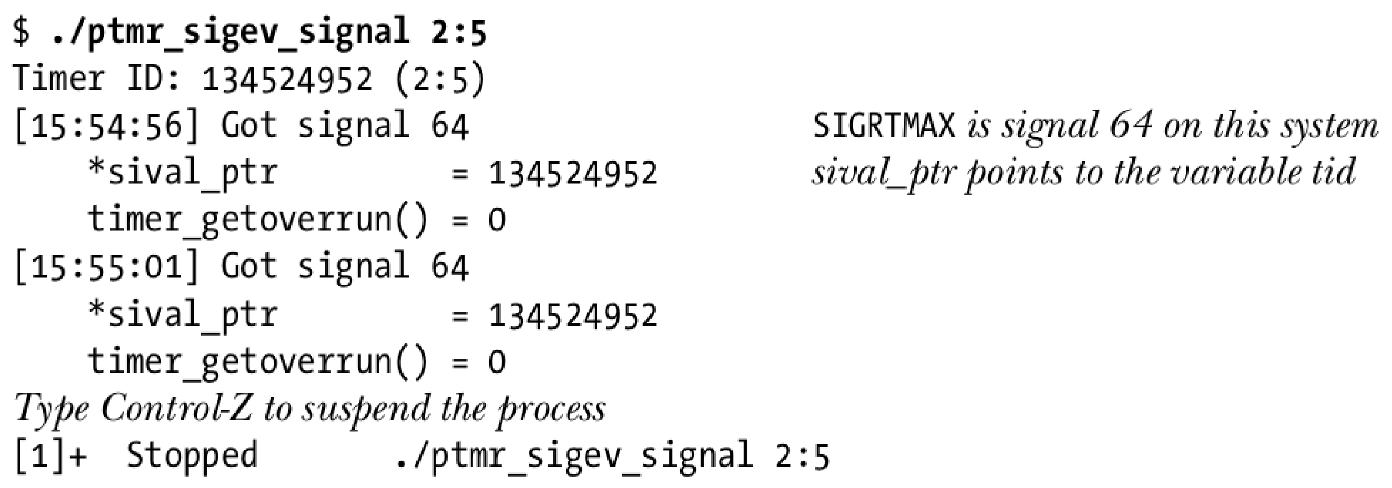
```
#define _POSIX_C_SOURCE 199309
#include <time.h>

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

Returns 0 on success, or –1 on error

```
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 */
};
```

ptmr_sigev_signal



```
$ ./ptmr_sigev_signal 2:5
Timer ID: 134524952 (2:5)
[15:54:56] Got signal 64                   SIGRTMAX is signal 64 on this system
    *sival_ptr         = 134524952         sival_ptr points to the variable tid
    timer_getoverrun() = 0
[15:55:01] Got signal 64
    *sival_ptr         = 134524952
    timer_getoverrun() = 0
Type Control-Z to suspend the process
[1]+  Stopped       ./ptmr_sigev_signal 2:5
```

시그널을 통한 타이머 알림

```
  static void
① handler(int sig, siginfo_t *si, void *uc)
  {
      timer_t *tidptr;

      tidptr = si->si_value.sival_ptr;

      /* UNSAFE: This handler uses non-async-signal-safe functions
         (printf(); see Section 21.1.2) */


    printf("[%s] Got signal %d\n", currTime("%T"), sig);
    printf("    *sival_ptr         = %ld\n", (long) *tidptr);
    printf("    timer_getoverrun() = %d\n", timer_getoverrun(*tidptr));
 }
```

```
      sa.sa_flags = SA_SIGINFO;
      sa.sa_sigaction = handler;
      sigemptyset(&sa.sa_mask);
②     if (sigaction(TIMER_SIG, &sa, NULL) == -1)
          errExit("sigaction");

      /* Create and start one timer for each command-line argument */

      sev.sigev_notify = SIGEV_SIGNAL;    /* Notify via signal */
      sev.sigev_signo = TIMER_SIG;        /* Notify using this signal */

      for (j = 0; j < argc - 1; j++) {
③         itimerspecFromStr(argv[j + 1], &ts);

          sev.sigev_value.sival_ptr = &tidlist[j];
                  /* Allows handler to get ID of this timer */

④         if (timer_create(CLOCK_REALTIME, &sev, &tidlist[j]) == -1)
              errExit("timer_create");
          printf("Timer ID: %ld (%s)\n", (long) tidlist[j], argv[j + 1]);

⑤         if (timer_settime(tidlist[j], 0, &ts, NULL) == -1)
              errExit("timer_settime");
      }

⑥     for (;;)                            /* Wait for incoming timer signals */
          pause();
  }
```

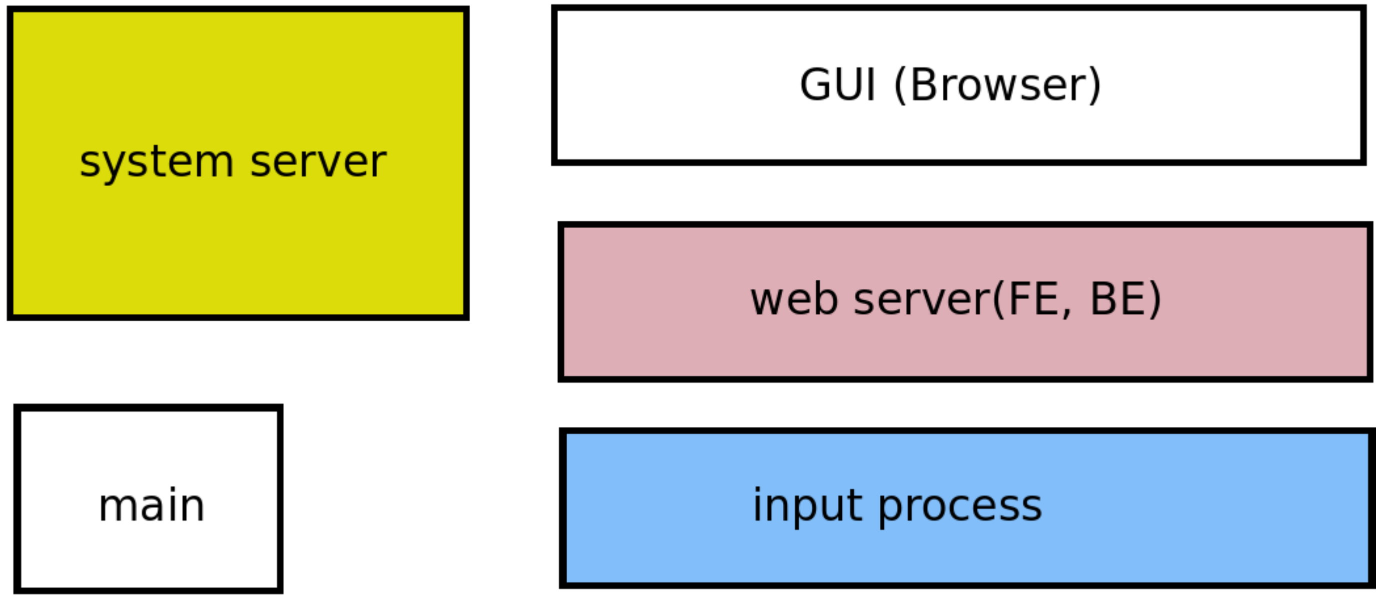
timers/ptmr_sigev_signal.c

실습 코드 분석

- vscode debugger로 디버깅
- tlpi-dist/timers/real_timer.c
  - 코드 분석 및 실행
- tlpi-dist/timers/ptmr_sigev_signal.c
  - 코드 분석 및 실행

토이 프로젝트 - 1 sec 타이머 추가

- system server에서 global timer 1개 구현
  - setitimer 이용
- 1 sec 마다 핸들러 호출 후 틱 값 출력


system server
GUI (Browser)
web server(FE, BE)
main
input process