```c
/* Pruebas de temporizadores */

#define _POSIX_C_SOURCE 199309L

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <signal.h>
#include <errno.h>
#include <time.h>

#define NTST  10
#define NTST1 20

/* Manejador, que no se utiliza */

void manejador(int signo, siginfo_t *datos, void *pa_na)
{
  printf("segnal llega al manejador!!!\n");
}

/* Tiempo transcurrido en segundos */

double lapso(struct timespec antes, struct timespec ahora)
{
  return (double)(ahora.tv_sec - antes.tv_sec) +
         ((double)(ahora.tv_nsec - antes.tv_nsec))/1000000000.;
}

/* Programa principal */

int main(int argc, char **argv)
{
  sigset_t la_de_tim;
  struct sigaction accion;
  int i;
  int over;
  struct timespec interv = { 1, 0};            /* Un segundo */
  struct timespec interv1 = {1, 0};
  struct timespec parada = { 10, 500000000L}; /* 10.5 segundos */
  struct timespec doscmseg = { 0, 200000000L}; /* 200 milisegundos */
  struct timespec cero = { 0, 0};
  struct timespec ahora;
  struct timespec antes;
  struct timespec inicial;
  struct timespec resolu;
  struct itimerspec tempo;
  int res;
  float media;
  float intrv;

  timer_t  mitimer;
  struct sigevent evento;
  siginfo_t info;

  sigset_t launo;

  /* Observando caracteristicas del reloj */

  clock_getres(CLOCK_REALTIME, &resolu);

  printf("El reloj tiene resolucion de %ld seg %ld nanoseg\n",
         (long)resolu.tv_sec, (long)resolu.tv_nsec);

  /* Temporizador periodico construido con retrasos */

/*
  interv1.tv_sec = 0;
  interv1.tv_nsec = 1000000000L - 1200000;
*/
  media = 0.;

  res = clock_gettime(CLOCK_REALTIME, &antes);
  inicial = antes;
```

```
for(i=0; i<NTST; i++)
{
  nanosleep(&interv1, NULL);
  clock_gettime(CLOCK_REALTIME, &ahora);
  intrv = lapso(antes, ahora);

  /* Perdida de tiempo intermedia */

  nanosleep(&doscmseg, NULL);

  /* Trazadores */

  media += intrv;
  printf("Intervalo %f sg. Total tiempo %f sg. Desviacion desde comienzo: %f sg.\n",
         intrv,
         lapso(inicial, ahora),
         lapso(inicial, ahora) - (double)(i+1));

  antes = ahora;
}

printf("Media de intervalos %f\n", media/NTST);

/* Probando temporizadores */

/* Primero funcionamiento de un solo disparo */

printf("probando temporizadores de un solo disparo\n");

tempo.it_value = interv;
tempo.it_interval = cero;

/* Le asocio la segnal SIGRTMIN */

/* 1: La bloqueo en la mascara de proceso */

sigemptyset(&la_de_tim);
sigaddset(&la_de_tim, SIGRTMIN);
sigprocmask(SIG_BLOCK, &la_de_tim, NULL);

/* 2: La programo de tiempo real */

sigemptyset(&accion.sa_mask);
accion.sa_flags = SA_SIGINFO;
accion.sa_sigaction = manejador;
sigaction(SIGRTMIN, &accion, NULL);

/* 3: Preparo sigevent para ponerla como evento de temporizador */

evento.sigev_signo = SIGRTMIN;
evento.sigev_notify = SIGEV_SIGNAL;
evento.sigev_value.sival_int = 77;

/* Creo el temporizador */

i = timer_create(CLOCK_REALTIME, &evento, &mitimer);
if(i==-1) printf("error en timer_create\n");

/* Lo programo de un disparo */

i = timer_settime(mitimer, 0, &tempo, NULL);

/* Espero la segnal sincronamente */

sigemptyset(&launo);
sigaddset(&launo, SIGRTMIN);
sigwaitinfo(&launo, &info);

printf("vencido temporizador de un disparo; dato de la segnal: %d\n",
        info.si_value.sival_int);

/* Ahora ciclico */
/* No hace falta destruir el temporizador para reprogramarlo */
```

```
  printf("probando temporizador ciclico con intervalo de 1 segundo\n");

  tempo.it_interval = interv;

  timer_settime(mitimer, 0, &tempo, NULL);
  nanosleep(&doscmseg, NULL);
  timer_gettime(mitimer, &tempo);

  printf("Prueba de timer_gettime: Quedan %ld sg %ld ms, Ciclo programado: %ld sg % ld nsg\n",
         (long)tempo.it_value.tv_sec,
         tempo.it_value.tv_nsec/1000000L,
         (long)tempo.it_interval.tv_sec,
         tempo.it_interval.tv_nsec);

  /* Espero la segnal */

  /* Se activa varias veces sin que se detecte segnal  -> overrun */

  nanosleep(&parada, NULL);

  /* Ahora se esperan las senales generadas y se mide el intervalo */

  media = 0.;

  clock_gettime(CLOCK_REALTIME, &antes);

  for(i=0; i<NTST1; i++)
  {
    if(i==2) inicial = ahora;
    sigwaitinfo(&launo, &info);
    clock_gettime(CLOCK_REALTIME, &ahora);
    over = timer_getoverrun(mitimer);
    intrv = lapso(antes, ahora);

    /* Perdida de tiempo intermedia */

//    nanosleep(&doscmseg, NULL);

    /* Trazadores */

    /* Se ignoran los dos primeros ciclos en los promedios,
       ya que son excepcionales por el overrun */

    if(i>=2)
    {
      if(over > 0) printf("Overrun de %d!\n", over);
      media += intrv;
      printf("Intervalo %f seg. Total de %f seg. Desv. desde el com.: %f\n",
                  intrv,
                  lapso(inicial, ahora),
                  lapso(inicial, ahora) - (i-1));
    }
    else
    {
      printf("Intervalo %f, ovr.: %d, valor de segnal: %d\n",
              intrv, over, info.si_value.sival_int);
    }

    antes = ahora;
  }

  printf("Media de intervalos %f\n", media/(NTST1-2));

  timer_delete(mitimer);
  return 0;
}
```