```c
/* Problema del examen de 18/12/07 */

#define _POSIX_C_SOURCE 199506L

#include <unistd.h>
#include <pthread.h>
#include <signal.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

#include "sistema.h"

/* Senal para ciclo de alarma */

#define SIG_CICLO SIGRTMIN+4

/* Variables compartidas */

/* Posiciones de almacenamiento */

int alm[NPA]; pthread_mutex_t mut_alm = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t hay_vacia              = PTHREAD_COND_INITIALIZER;
pthread_cond_t hay_llena              = PTHREAD_COND_INITIALIZER;

/* Pasillo */

pthread_mutex_t mut_pasillo = PTHREAD_MUTEX_INITIALIZER;

/* Temporizador de sobretiempo */

timer_t tim_tout;
#define SIG_TOUT SIGRTMIN+3

/* Fin */

int fin = 0;

/* Funciones */

/* Manejador (no se usa) */

void m(int sig, siginfo_t *info, void *p)
{}

/* Hilos */

void *hilo_veh1(void *p);
void *hilo_veh2(void *p);
void *hilo_alarma(void *p);

/* Main */

int main(int argc, char **argv) {
    sigset_t todas;
    sigset_t sfin;
    struct sigaction sa;
    int i;
    struct sigevent evento;
    pthread_t hveh1, hveh2, alarma;
    struct itimerspec timeout;
    struct timespec min5 = {30, 0L};
    struct timespec cero = {0, 0};
    union sigval valor;

    /* Bloquear senales */

    sigemptyset(&todas);
    sigaddset(&todas, SP1);
    sigaddset(&todas, SP2);
    sigaddset(&todas, SIGTERM);
    sigaddset(&todas, SIGINT);
    sigaddset(&todas, SIG_TOUT);
    sigaddset(&todas, SIG_CICLO);
    pthread_sigmask(SIG_BLOCK, &todas, NULL);

    /* Programar senales como de tiempo real */

    sa.sa_sigaction = m;
    sa.sa_flags = SA_SIGINFO;
    sigemptyset(&sa.sa_mask);
    sigaction(SP1, &sa, NULL);
    sigaction(SP2, &sa, NULL);

    /* Inicializar variables compartidas (almacenes vacios) */

    for(i=0; i<NPA; i++)  alm[i] = 0;

    /* Inicializar temporizador de sobretiempo */
    /* Se activara cuando el almacen se quede vacio,
       y se desactivara cuando contenga alguna pieza */

    evento.sigev_signo = SIG_TOUT;
    evento.sigev_notify = SIGEV_SIGNAL;
    timer_create(CLOCK_REALTIME, &evento, &tim_tout);

    /* Activar temporizador de sobretiempo (inicialmente todo esta vacio...) */

    timeout.it_value = min5;
    timeout.it_interval = cero;
    timer_settime(tim_tout, 0, &timeout, NULL);

    /* Activar hilos */

    pthread_create(&hveh1, NULL, hilo_veh1, NULL);
    pthread_create(&hveh2, NULL, hilo_veh2, NULL);
    pthread_create(&alarma, NULL, hilo_alarma, NULL);

    /* Esperar senales de fin */

    sigemptyset(&sfin);
    sigaddset(&sfin, SIGTERM);
    sigaddset(&sfin, SIGINT);

    sigwaitinfo(&sfin, NULL);
    printf("main: Senal de fin detectada\n");

    /* Parar hilos y esperarlos */

    fin = 1;

    /* Para terminar se generan las condiciones de
       espera que pueden dar problemas */

    valor.sival_int = 1;
    sigqueue(getpid(), SP1, valor);
    pthread_cond_signal(&hay_vacia);
    pthread_join(hveh1, NULL);
    pthread_cond_signal(&hay_llena);
    pthread_join(hveh2, NULL);
    kill(getpid(), SIG_TOUT);
    pthread_join(alarma, NULL);
    printf("main: hilos terminados\n");

    exit(0);
}

/* HIlo de vehiculo 1 */

void *hilo_veh1(void *p) {
    siginfo_t dato;
    sigset_t sp1;
    int lugar;
    int encontrado;
    struct itimerspec desact;
    struct timespec cero = {0, 0L};
    int i;

    /* Preparar prog. para desactivar timeout */
```

```
desact.it_value = cero;
desact.it_interval = cero;

/* Senal de P1, que hay que esperar */
sigemptyset(&sp1);
sigaddset(&sp1, SP1);

/* Bucle de funcionamiento */

while(!fin)
{
   /* Activar M1 y esperar llegada de pieza */

    printf("vehiculo 1 esperando pieza\n"),
    cambia_salida(M1, 1);
    do {
       sigwaitinfo(&sp1, &dato);
    } while(dato.si_value.sival_int == 0 && !fin);
    cambia_salida(M1, 0);

    if(fin) continue;

   /* Cargar pieza */

    cambia_salida(CV1, 1);
    do {
       sigwaitinfo(&sp1, &dato);
    } while(dato.si_value.sival_int == 1);
    cambia_salida(CV1, 0);

    /* Ir a punto de espera */

    printf("vehiculo 1 va a punto de espera de pasillo\n");
    mueve(1, FP, PA0-1);

    /* Esperar lugar vacio */

    printf("vehiculo 1 esperando lugar vacio\n");
    pthread_mutex_lock(&mut_alm);

    /* Condicion: Algun lugar libre
       (alguna posicion de alm vale cero) */

    do {
            for(lugar = 0, encontrado = 0;
                 lugar < NPA && !encontrado; lugar++)
            {
                 if(alm[lugar] == 0) encontrado = 1;
            }

            if(!encontrado && !fin)
              pthread_cond_wait(&hay_vacia, &mut_alm);
            else lugar--;

     } while(!encontrado && !fin);

     pthread_mutex_unlock(&mut_alm);

     if(fin) continue;

     /* Esperar pasillo */

     printf("vehiculo 1 esperando pasillo para descargar en %d\n", lugar);
     pthread_mutex_lock(&mut_pasillo);

     /* Ir a descargar; avisa de que hay pos. llena
         y desactiva el temporizador */

     mueve(1, FP, PA0+lugar);
     cambia_salida(VV1, 1);
     cambia_salida(VV1, 0);

     pthread_mutex_lock(&mut_alm);
     alm[lugar] = 1;
     pthread_cond_signal(&hay_llena);
     timer_settime(tim_tout, 0, &desact, NULL);
     pthread_mutex_unlock(&mut_alm);

     /* Volver a cinta 1 */

     printf("vehiculo 1 ha descargado en %d\n", lugar);
     mueve(1, FP, PA0-1);
     pthread_mutex_unlock(&mut_pasillo);

     mueve(1, FP, RV1);
  }

  printf("vehiculo 1 saliendo\n");

  return NULL;
}

/* Hilo veh2 */

void *hilo_veh2(void *p) {
  int i;
  struct itimerspec activar;
  struct timespec cinco = {30, 0};
  struct timespec cero = {0, 0};
  int lugar;
  int encontrado;
  int cuenta;
  siginfo_t dato;
  sigset_t sp2;

  /* Preparar prog. para activacion de temporizador */

  activar.it_value = cinco;
  activar.it_interval = cero;

  /* Senal SP2 de la cinta */

  sigemptyset(&sp2);
  sigaddset(&sp2, SP2);

  /* Bucle de funcionamiento */

  while(!fin)
  {
      /* Esperar lugar lleno */

      pthread_mutex_lock(&mut_alm);

      /* Condicion: Algun lugar ocupado
         (alguna posicion de alm a 1) o fin  */

      do {

            for(lugar = 0, encontrado = 0;
                  lugar < NPA && !encontrado && !fin; lugar++)
            {
                 if(alm[lugar] == 1) encontrado = 1;
            }

            if(!encontrado && !fin)
              pthread_cond_wait(&hay_llena, &mut_alm);
            else lugar--;

      } while(!encontrado && !fin);

      pthread_mutex_unlock(&mut_alm);

      if(fin) continue;

      printf("vehiculo 2 ha encontrado lugar lleno %d\n", lugar);

      /* Mueve a posicion de espera de pasillo */

      mueve(2, FP, PA0+NPA);
      printf("vehiculo 2 esperando pasillo\n");
```

```
        pthread_mutex_lock(&mut_pasillo);

        /* Va a posicion llena */

        mueve(2, FP, PA0+lugar);

        /* Descarga, avisa de  vacia y
            si todas lo estan, activa temporizador */

       cambia_salida(CV2, 1);
       cambia_salida(CV2, 0);

       pthread_mutex_lock(&mut_alm);
       alm[lugar] = 0;
       pthread_cond_signal(&hay_vacia);
       for(i=0, cuenta = 0; i<NPA; i++)
          if(alm[i] == 0) cuenta++;

       if(cuenta == NPA)
          timer_settime(tim_tout, 0, &activar, 0);

       pthread_mutex_unlock(&mut_alm);

       printf("Vehiculo 2 ha cargado de %d; quedan %d posiciones llenas\n",
                  lugar, NPA-cuenta);

       /* Volver a descargar */

       mueve(2, FP, PA0+NPA);
       pthread_mutex_unlock(&mut_pasillo);

       mueve(2, FP, RV2);

       /* Esperar cinta vacia */

       printf("vehiculo 2 espera cinta vacia\n");
       do {
           sigwaitinfo(&sp2, &dato);
       } while(dato.si_value.sival_int == 1);

       /* Descargar */

       printf("vehiculo 2 descargando\n");
       cambia_salida(VV2, 1);
       do {
           sigwaitinfo(&sp2, &dato);
       } while(dato.si_value.sival_int == 0);
       cambia_salida(VV2, 0);
    }

    printf("vehiculo 2 saliendo\n");

    return NULL;
}


/* HIlo de alarma */

void *hilo_alarma(void *p) {
    sigset_t stout;           /* Senal de timeout */
    sigset_t ciclo;           /* Senal de ciclo */
    timer_t tcic;             /* Temporizador de ciclo */
    struct sigevent evento;
    int alarma;
    int i;
    struct itimerspec cic200;
    struct timespec ms200 = {0, 200000000L};
    struct timespec cero = {0, 0};
    struct itimerspec parar;

    /* Senales de timeout y de ciclo */

    sigemptyset(&stout);
    sigaddset(&stout, SIG_TOUT);
    sigemptyset(&ciclo);
    sigaddset(&ciclo, SIG_CICLO);

    /* Crear temporizador ciclico */

    evento.sigev_notify = SIGEV_SIGNAL;
    evento.sigev_signo = SIG_CICLO;
    timer_create(CLOCK_REALTIME, &evento, &tcic);

    /* Programaciones para parar y arrancar temporizador ciclico */

    cic200.it_value = ms200;
    cic200.it_interval = ms200;

    parar.it_value = cero;
    parar.it_interval = cero;

    /* Bucle, una iteracion por alarma */

    while(!fin)
    {
         /* Esperar alarma de los 5 minutos */

         sigwaitinfo(&stout, NULL);
         printf("alarma de sobretiempo o fin\n");

         if(!fin)
         {
            /* Sobretiempo: Avisar cada 200 ms */

            alarma = 1;
            timer_settime(tcic, 0, &cic200, NULL);

            do
            {
               printf("alarma: Enviando senal\n");
              /* kill(getppid(), SIGRTMIN); */ /* Mejor no la enviamos, porque matariamos el sh
               sigwaitinfo(&ciclo, NULL);

               /* MIrar si hay todavia condicion de alarma */

               pthread_mutex_lock(&mut_alm);
               for(i=0; i<NPA; i++)  if(alm[i] == 1) alarma = 0;
               pthread_mutex_unlock(&mut_alm);

            } while(alarma && !fin);

            timer_settime(tcic, 0, &parar, NULL);

         }

         printf("alarma: Alarma cancelada o fin\n");

    }

    printf("Hilo alarma saliendo\n");

    return NULL;
}
```

```
/* Cabecera sistema.h para problema de 18/12/07 */

/* Constantes simbolicas */

/* Entradas */

#define SP1 SIGRTMIN
#define SP2 SIGRTMIN+1

/* Salidas */

#define M1  0
#define CV1 1
#define CV2 2
#define VV1 3
#define VV2 4

/* Posiciones */

#define FP    10   /* Fila comun */
#define RV1    1   /* Reposo vehiculo 1 */
#define PA0    3   /* Pos. almacenamiento 0 */
#define RV2   20   /* Reposo vehiculo 2 */

/* Numero de posiciones de almacenamiento */

#define NPA 6

/* Funciones disponibles en biblioteca */

void cambia_salida(int out, int valor);
void mueve(int veh, int fila, int col);
```

```
/* Funciones para el problema de 18/12/07 */

#define _POSIX_C_SOURCE 199506L

#include <unistd.h>
#include <signal.h>
#include <time.h>
#include <stdio.h>

#include "sistema.h"

int inic = 0;
int pieza_en_1 = 0;
int pieza_en_2 = 0;

void cambia_salida(int out, int valor) {
    union sigval val;
    struct timespec retraso = {0, 500000000L};
    struct timespec retraso2 = {30, 0L};

/*    printf("cambia_salida: Salida %d cambiada a valor %d\n", out, valor);*/

    /* Primera senal para indicar que la cinta esta vacia */

    if(!inic)
    {
       inic = 1;
        val.sival_int = 0;
       sigqueue(getpid(), SP2, val);
    }

    if(out == M1 && valor == 1)
    {
        nanosleep(&retraso, NULL);
        if(pieza_en_1 == 0)
        {
          pieza_en_1 = 1;
          val.sival_int = 1;
          sigqueue(getpid(), SP1, val);
        }
    }

    if(out == CV1 && valor == 1)
    {
        nanosleep(&retraso, NULL);
        if(pieza_en_1 == 1)
        {
          pieza_en_1 = 0;
          val.sival_int = 0;
          sigqueue(getpid(), SP1, val);
        }
    }

    if(out == VV2 && valor == 1)
    {
        nanosleep(&retraso, NULL);
        /*nanosleep(&retraso2, NULL); *//* Para forzar el lleno total */
        if(pieza_en_2 == 0)
        {
          pieza_en_2 = 1;
          val.sival_int = 1;
          sigqueue(getpid(), SP2, val);
          nanosleep(&retraso, NULL);
          pieza_en_2 = 0;
          val.sival_int = 0;
          sigqueue(getpid(), SP2, val);
        }
   }
}

/* Mover vehiculo. Se limita a esperar un retraso */

void mueve(int veh, int fila, int col) {
    struct timespec retraso = {2, 0};
```

```
    nanosleep(&retraso, NULL);

/*    printf("mueve: vehiculo %d movido a fila %d columna %d\n", veh, fila, col);*/
}
```

```
vehiculo 1 esperando pieza vehiculo 1 va a punto de espera de pasillo
vehiculo 1 esperando lugar vacio
vehiculo 1 esperando pasillo para descargar en 0
vehiculo 1 ha descargado en 0
vehiculo 2 ha encontrado lugar lleno 0
vehiculo 2 esperando pasillo
vehiculo 1 esperando pieza
Vehiculo 2 ha cargado de 0; quedan 0 posiciones llenas
vehiculo 1 va a punto de espera de pasillo
vehiculo 1 esperando lugar vacio
vehiculo 1 esperando pasillo para descargar en 0
vehiculo 2 espera cinta vacia vehiculo 2 descargando
vehiculo 1 ha descargado en 0
vehiculo 2 ha encontrado lugar lleno 0
vehiculo 2 esperando pasillo
vehiculo 1 esperando pieza Vehiculo 2 ha cargado de 0; quedan 0 posiciones llenas
vehiculo 1 va a punto de espera de pasillo
vehiculo 1 esperando lugar vacio
vehiculo 1 esperando pasillo para descargar en 0
main: Senal de fin detectada
vehiculo 2 espera cinta vacia
vehiculo 2 descargando vehiculo 2 saliendo
vehiculo 1 ha descargado en 0
vehiculo 1 saliendo alarma de sobretiempo o fin
alarma: Alarma cancelada o fin
Hilo alarma saliendo
main: hilos terminados
```