```c
#define _POSIX_C_SOURCE 199506L

#include <string.h>

#include <unistd.h>
#include <pthread.h>
#include <stdio.h>
#include <time.h>
#include <errno.h>
#include <string.h>

int comun;

struct timespec tim = {0, 40000000L}; /* 40 milisegundos */
void *hilo1(void *p);
void *hilo2(void *p);

int main(int argc, char **argv)
{
  pthread_t th;
  int dato = 20;
  int res;
  int cnt;
  void *result;
  int error;

  /* Arranca primer hilo */

  res = pthread_create(&th, NULL, hilo1, (void *)dato);

  /* Ejemplo de comprobacion de errores -no se usa errno- */

  if(res)
  {
    printf("Error en create: %s\n", strerror(res));
    exit(1);
  }


  /* Espera a que termine el primer hilo */

  pthread_join(th, &result);
  printf("Soy main. Resultado del primer hilo: %s\n", (char *)result);

  /* Generacion y deteccion de un error */

  if(res = pthread_join(th, &result))
    printf("Error (previsto) en join: %s\n", strerror(res));

  /* Arranca el segundo hilo */

  pthread_create(&th, NULL, hilo2, (void *)dato);
  cnt = 10;
  while(cnt--)
  {
    nanosleep(&tim, NULL);  /* Espera 40 milisegundos */
    printf("Soy main. Var. compartida: %d\n", comun);
  }
  printf("Soy main. Voy a esperar a Hilo2.\n");
  res = pthread_join(th, &result);

  printf("Soy main. Hilo2 ha acabado.");
  if(res != 0)
       printf("Error en join %s\n", strerror(res));
  else if(result == (void *)PTHREAD_CANCELED)
       printf(" Ha sido cancelado.\n");
  else printf(" Ha generado resultado %d\n", (int)result);

  /* Otra vez hilo2; ahora sera cancelado */

  printf("main: prueba de cancelacion\n");
  pthread_create(&th, NULL, hilo2, (void *)dato);

  /* Main deja a hilo2 que se ejecute por un tiempo */
```

```
  cnt = 4;
  while(cnt--) nanosleep(&tim, NULL);

  /* Cancelacion de hilo2 */

  {
     int a;
     a = pthread_cancel(th);
     if(a) printf("error en pthread_cancel %s\n", strerror(a));
  }

/*  pthread_detach(th);  */

  printf("Soy main. Voy a esperar a Hilo2.\n");

  res = pthread_join(th, (void *)&result);

  printf("Soy main. Hilo2 ha acabado.");

  if(res != 0)
       printf("Error en join %s\n", strerror(res));
  else if(result == (void *)PTHREAD_CANCELED)
       printf(" Ha sido cancelado.\n");
  else printf(" Ha generado resultado %d\n", (int)result);

  /* Otra vez hilo2; ahora acabara solo porque main termina - y llama a
     exit */

  pthread_create(&th, NULL, hilo2, (void *)dato);

  /* Main deja a Hilo2 que se ejecute por un tiempo */
  cnt = 6;
  while(cnt--) nanosleep(&tim, NULL);
  printf("Soy main. Voy a acabar.\n");

}

/* Rutina del hilo1 */

void *hilo1(void *p)
{
  char *mensaje = "hola";
  char *pc;

  printf("Soy hilo2\n");

  pc = (char *)malloc(5);
  strcpy(pc, mensaje);

  /* Devuelve un puntero, pero apunta a memoria reservada con malloc,
     no a la pila */

  return (void *)pc;
}

/* Rutina del hilo2 */

void *hilo2(void *arg)
{
  int cnt = (int)arg;       /* Toma arg por un entero */
  int r = 77;               /* Este es el valor que se devuelve */

  printf("soy hilo2. Argumento: %d\n", (int)arg);
/*
  pthread_setcancelstate(PTHREAD_CANCEL_ENABLE, NULL);
  pthread_setcanceltype(PTHREAD_CANCEL_DEFERRED, NULL);
*/
/*  pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS, NULL); */

  while(cnt--)
  {
     /* En QNX 6.2.0 una cancelacion durante un printf no funciona
         adecuadamente, por eso se deshabilita (es un problema del sistema) */
    pthread_setcancelstate(PTHREAD_CANCEL_DISABLE, NULL);
    printf("soy hilo2. Contador: %d\n", cnt);
```

```
    pthread_setcancelstate(PTHREAD_CANCEL_ENABLE, NULL);
    comun = cnt;            /* El contador (local) se copia en comun (global) */
    pthread_testcancel();   /* En realidad no es necesario si esta el nanosleep */
/*     pthread_setcancelstate(PTHREAD_CANCEL_DISABLE, NULL);  */
     nanosleep(&tim, NULL);  /* Espera 40 milisegundos (incluye p. de canc.) */
/*     pthread_setcancelstate(PTHREAD_CANCEL_ENABLE, NULL);  */
  }
 sleep(15);
  return ((void *)r);
}
```