```c
/* Proceso hijo: Recibe senales de tiempo real */

#define _POSIX_C_SOURCE 199309L

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

int acabar = 0;  /* Indicador de terminacion */

/* No siempre se activara el manejador */

void manejador(int signo, siginfo_t *datos, void *pa_na)
{
  /* El dato que lleva la senal determina si se acaba o no */

  fprintf(stderr, "hijo -> el manejador recibe %d\n",
          datos->si_value.sival_int);
  if(datos->si_value.sival_int == 9) acabar = 1;
}

void main(int argc, char **argv)
{
  struct sigaction sa;
  siginfo_t info;
  sigset_t la_de_tr;
  int retorno;
  int i;

 /* Accion para SIGRTMIN: manejador de POSIX.4 */

  sa.sa_handler = NULL;
  sa.sa_sigaction = manejador;
  sigemptyset(&sa.sa_mask);
  sa.sa_flags = SA_SIGINFO;
  retorno = sigaction(SIGRTMIN, &sa, NULL);
  if(retorno == -1) fprintf(stderr, "hijo -> problema en sigaction\n");

 /* BLOQUEO de la senal que van a enviar, para reconocimiento sincrono */

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

  fprintf(stderr, "hijo -> primero espero con sigwaitinfo.\n");

  for(i=0; i<3; i++)
  {
    retorno = sigwaitinfo(&la_de_tr, &info);

    if(retorno != -1)
    {
      if(retorno == SIGRTMIN) fprintf(stderr, "hijo -> ha llegado SIGRTMIN\n");
      else                    fprintf(stderr, "hijo -> no es SIGRTMIN!\n");
      switch(info.si_code)
      {
        case SI_QUEUE:
         fprintf(stderr, "hijo -> llega el %d enviado por sigqueue\n",
                 info.si_value.sival_int);
```



```c
         break;
        case SI_USER:
         fprintf(stderr, "hijo -> llega senal pero por un kill\n");
         break;
        default:
         fprintf(stderr, "hijo -> llega senal por causas inesperadas\n");
         break;
      }
    }
    else
    {
      fprintf(stderr, "hijo -> error en sigwaitinfo:\n");

      switch(errno)
      {
        case EINTR:
          fprintf(stderr, "hijo -> ha llegado una senal no esperada\n");
          break;

        default:
          fprintf(stderr, "hijo -> fallo desconocido\n");
          break;
      }
    }
  }

  /* Ahora espera con manejador */

  fprintf(stderr, "hijo: ahora  espero con manejador.\n");

  /* Desbloqueo de SIGRTMIN */

  sigprocmask(SIG_UNBLOCK, &la_de_tr, NULL);

  while(!acabar);

  fprintf(stderr, "hijo -> acabando\n");

}
```