```
/* Problema de examen del 11/9/08 */

#define _POSIX_C_SOURCE 199506

#include <unistd.h> #include <mqueue.h> #include <signal.h> #include
<errno.h> #include <string.h> #include <time.h> #include <stdio.h>
#include <sys/wait.h>

#include "proceso.h"

#define TIMEOUT               100 #define RETRASO_MINIMO    20
#define RETRASO_MAXIMO   60

/* Manejador: No hace nada, salvo romper la espera de mq_receive
    si es necesario */

void manej(int sig) { }

int main(int argc, char **argv) {
    mqd_t cola;
    struct mq_attr atr;
    int parar = 0;
    pid_t pb;
    int prio;
    int dato;
    int cfallos;
    int res;
    timer_t tempo;
    struct timespec cero = {0, 0};
    struct timespec cienmilis = {0, 100000000L}; /* 500 en vez de 100 */
    struct itimerspec progtempo;
    struct itimerspec desactivar;
    struct sigevent evento;
    struct timespec antes;
    struct timespec ahora;
    long espera;
    sigset_t set;
    struct sigaction acc;

    if(argc < 3)
    {
       printf("argc menor que 3\n");
       exit(1);
    }

    /* Crear y abrir cola */

    atr.mq_maxmsg = LCOLA;
    atr.mq_msgsize = sizeof(int);

    mq_unlink(argv[2]);
    cola = mq_open(argv[2], O_RDONLY | O_CREAT | O_EXCL, S_IRWXU, &atr);
    if(cola == -1)
    {
       printf("Error en apertura de cola\n");
       exit(1);
    }

    /* Crear temporizador */

    evento.sigev_signo = SIGRTMIN;
    evento.sigev_notify = SIGEV_SIGNAL;
    timer_create(CLOCK_REALTIME, &evento, &tempo);

    /* Preparar programaciones */

    progtempo.it_value = cienmilis;
    progtempo.it_interval = cero;

    desactivar.it_value = cero;
```



```
    desactivar.it_interval = cero;

    /* Preparar manejador de la senal de aviso del temporizador */

    acc.sa_flags = 0;
    acc.sa_handler = manej;
    sigemptyset(&acc.sa_mask);
    sigaction(SIGRTMIN, &acc, NULL);

    /* Desbloquear senal de aviso del temporizador */

    sigemptyset(&set);
    sigaddset(&set, SIGRTMIN);
    sigprocmask(SIG_UNBLOCK, &set, NULL);

    /* Bucle: Una iteracion por reintento */

    while(!parar)
    {
       /* Arrancar proceso B teniendo como argumento 1 el argumento  2
           del proceso A (el nombre de la cola)  */

       pb = fork();
       if(!pb)
       {
          execl(argv[1], argv[1], argv[2], NULL);
          printf("Problemas abriendo %s\n", argv[1]);
       }

       cfallos = 0;                              /* Al empezar, 0 fallos */
       clock_gettime(CLOCK_REALTIME, &antes);   /* Referencia inicial para retrasos */

        /* Iterar leyendo mensajes hasta que el proceso B termine correctamente
           o falle  */

       while(!parar && cfallos < N_MAX_ERR)
       {
          /* Programar temporizador de 100 ms */

          timer_settime(tempo, 0, &progtempo, NULL);

          /* Esperar mensaje o sobretiempo.
              Si el temporizador avisa antes de la recepcion, el manejador salta y
              la espera se rompe */

          res = mq_receive(cola, (unsigned char *)&dato, sizeof(int), &prio);

          /* Desactivar temporizador y tomar medida de tiempo para medir
              retraso */

          timer_settime(tempo, 0, &desactivar, NULL);
          clock_gettime(CLOCK_REALTIME, &ahora);

          /* Si ha habido error en la recepcion suponemos que la espera se ha
              roto por culpa del temporizador; si no, la recepcion ha sido correcta */

          if(res == -1)
          {
              /* Sobretiempo. Abortar proceso B y terminar bucle */

              printf("Sobretiempo en recepcion; abortando proceso B\n"); fflush(stdout);

              kill(pb, SIGTERM);
              wait(&res);
              cfallos = N_MAX_ERR;
          }
          else
          {
            /* Comprobar si el retraso es inadecuado */
```

```
            espera = (ahora.tv_sec - antes.tv_sec)*1000 +
                            (ahora.tv_nsec - antes.tv_nsec)/1000000L;

            if(espera < RETRASO_MINIMO || espera > RETRASO_MAXIMO)
            {
               /* Fuera de limites: Incrementar fallos y avisar a proceso B.
                    Si se alcanza el limite, abortar proceso B y terminar
                    el bucle para rearrancarlo */

              cfallos++;
              printf("Espera %ld: fallo %d\n", espera, cfallos); fflush(stdout);
              if(cfallos < N_MAX_ERR) kill(pb, SIGRTMIN);
              else
              {
                 printf("Demasiados fallos; rearrancando proceso %s\n", argv[1]); fflush(stdout
                 kill(pb, SIGTERM);
                 wait(&res);
              }
            }
            else
            {
                 printf("Recepcion correcta %d con espera %ld\n", dato, espera); fflush(stdout)
                 if(dato == FIN)
                 {
                     printf("Fin de proceso b\n");
                     wait(&res);
                     parar = 1;
                 }
             }
          }

          /* Tomar como referencia futura el tiempo medido en esta iteracion */

          antes = ahora;
       }
    }

    printf("Proceso A acabando\n"); fflush(stdout);
    return 0;
}
```



```
/* Cabecera */

#define LCOLA       5
#define N_MAX_ERR   5
#define ACTIVO      0
#define FIN         1
```

```
#define _POSIX_C_SOURCE 199506L

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

#include "proceso.h"

/* Hilo de recepcion de senales */

void *recibe(void *p);

int main(int argc, char **argv)
{
   mqd_t cola;
   int n;
   struct timespec t = {0, 40000000L};
   int dato;
   sigset_t set;
   int espera [] = {40, 50, 80, 80, 8, 50, 80, 80, 50};
   //int espera [] = {40, 50, 80, 80, 8, 50, 80, 200, 50};
   int nt = sizeof(espera)/sizeof(int);
   pthread_t hilo;

   if(argc < 2)
   {
      printf("proceso B con menos de 2 args\n");
      exit(1);
   }

   /* El argumento 1 es el nombre de la cola */

   cola = mq_open(argv[1], O_WRONLY, 0, NULL);
   printf("proceso b ha abierto la cola\n");

   /* Bloquear senal SIGRTMIN */

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

   /* Crear hilo de recepcion de senales */

   pthread_create(&hilo, NULL, recibe, NULL);

   /* Iteraciones para enviar datos por la cola utilizando un retraso variable */

   t.tv_sec = 0;
   for(n = 0; n<nt; n++)
   {
      t.tv_nsec = espera[n]*1000000L;
      nanosleep(&t, NULL);
      if(n < nt-1) dato = ACTIVO;
      else         dato = FIN;
      printf("proceso B envia con %d ms de retraso\n",espera[n]); fflush(stdout);
      mq_send(cola, (unsigned char *)&dato, sizeof(int), 0);
   }

   printf("proceso B acabando correctamente\n");

   return 0;
}

/* Hilo de recepcion de senales */

void *recibe(void *p)
{
```



```
    sigset_t senal;

    sigemptyset(&senal); sigaddset(&senal, SIGRTMIN);

    while(1)
    {
        sigwaitinfo(&senal, NULL);
        printf("Hilo recibe: Recibida senal\n"); fflush(stdout);
    }
    return NULL;
}
```

```
proceso b ha abierto la cola
proceso B envia con 40 ms de retraso
Recepcion correcta 0 con espera 44
proceso B envia con 50 ms de retraso
Recepcion correcta 0 con espera 51
proceso B envia con 80 ms de retraso
Espera 81: fallo 1
Hilo recibe: Recibida senal
proceso B envia con 80 ms de retraso
Espera 81: fallo 2
Hilo recibe: Recibida senal
proceso B envia con 8 ms de retraso
Espera 9: fallo 3
Hilo recibe: Recibida senal
proceso B envia con 50 ms de retraso
Recepcion correcta 0 con espera 52
proceso B envia con 80 ms de retraso
Espera 81: fallo 4
Hilo recibe: Recibida senal
proceso B envia con 80 ms de retraso
Espera 81: fallo 5
Demasiados fallos; rearrancando proceso ./procesob
proceso b ha abierto la cola
proceso B envia con 40 ms de retraso
Recepcion correcta 0 con espera 44
proceso B envia con 50 ms de retraso
Recepcion correcta 0 con espera 51
proceso B envia con 80 ms de retraso
Espera 81: fallo 1
Hilo recibe: Recibida senal
proceso B envia con 80 ms de retraso
Espera 81: fallo 2
Hilo recibe: Recibida senal
proceso B envia con 8 ms de retraso
Espera 9: fallo 3
Hilo recibe: Recibida senal
proceso B envia con 50 ms de retraso
Recepcion correcta 0 con espera 51
proceso B envia con 80 ms de retraso
Espera 81: fallo 4
Hilo recibe: Recibida senal
proceso B envia con 80 ms de retraso
Espera 81: fallo 5
Demasiados fallos; rearrancando proceso ./procesob
proceso b ha abierto la cola
proceso B envia con 40 ms de retraso
Recepcion correcta 0 con espera 43
proceso B envia con 50 ms de retraso
Recepcion correcta 0 con espera 51
proceso B envia con 80 ms de retraso
Espera 81: fallo 1
Hilo recibe: Recibida senal
proceso B envia con 80 ms de retraso
Espera 81: fallo 2
Hilo recibe: Recibida senal
proceso B envia con 8 ms de retraso
Espera 9: fallo 3
Hilo recibe: Recibida senal
proceso B envia con 50 ms de retraso
Recepcion correcta 0 con espera 52
proceso B envia con 80 ms de retraso
Espera 81: fallo 4
Hilo recibe: Recibida senal
proceso B envia con 80 ms de retraso
Espera 81: fallo 5
Demasiados fallos; rearrancando proceso ./procesob
proceso b ha abierto la cola
proceso B envia con 40 ms de retraso
Recepcion correcta 0 con espera 44
proceso B envia con 50 ms de retraso
```



```
Recepcion correcta 0 con espera 51
proceso B envia con 80 ms de retraso
Espera 81: fallo 1
Hilo recibe: Recibida senal
proceso B envia con 80 ms de retraso
Espera 81: fallo 2
Hilo recibe: Recibida senal
proceso B envia con 8 ms de retraso
Espera 9: fallo 3
Hilo recibe: Recibida senal
proceso B envia con 50 ms de retraso
Recepcion correcta 0 con espera 51
```