```c
#define _POSIX_C_SOURCE 199309L

#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <errno.h>

/* Se utiliza fflush para que la salida de fichero sea identica
   a la obtenida por terminal */

/* Las llamadas de espera, sleep(<n>), permiten que sea concurrente
   la actividad de ambos procesos */

int main(int argc, char **argv)
{
  pid_t pid;
  int estado;
  int codigo = 77;
  int i;

  printf("padre -> Hola mundo, soy %d\n", (int)getpid());
  printf("padre -> Soy %d Voy a desdoblarme!\n", (int)getpid());
  fflush(stdout);

  pid = fork();
  if(!pid)      /* Bloque activo en el proceso hijo */
  {
    sleep(1);
    printf("soy mi otro yo, el proceso %d creado por %d\n",
           (int)getpid(), (int)getppid());

    printf("para %d codigo vale %d\n", (int)getpid(), codigo);

    for(i=0; i<10; i++)
    {
      printf("%d Cuenta: %d\n", (int)getpid(), i);
      fflush(stdout);
      sleep(2);
    }

    printf("%d Acaba\n", (int)getpid(), i);
    fflush(stdout);
    exit(3);
  }
  else /* Bloque activo en el proceso padre */
  {
    printf("para %d codigo vale %d\n", (int)getpid(), codigo);

    for(i=0; i<10; i++)
    {

      printf("%d Cuenta: %d\n", (int)getpid(), i);
      fflush(stdout);
      sleep(1);
    }

    printf("voy a esperar al %d\n", (int)pid);
    fflush(stdout);
    if(waitpid(pid, &estado, 0) > 0)
    {
      printf("espera concluida\n");
      if(WIFEXITED(estado))
      {
        printf("%d salio con estado %d\n", (int)pid, WEXITSTATUS(estado));
      }
    }
    else printf("Error en waitpid: %s\n", strerror(errno));

    printf(" el padre acaba\n");
  }
  return 1;
}
```