Informática Industrial

4º Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Práctica 1

Procesos y señales

1. Introducción

Se realizará una aplicación en C sobre QNX 6.3 que controla los pulsadores, posiciones de almacenamiento y vehículos simulados de la Figura 1. En el gráfico hay dos conjuntos de pulsadores que permiten iniciar operaciones de carga y descarga. La sincronización y comunicación entre los procesos del sistema se realiza mediante señales.

Para el desarrollo de este ejercicio se proporcionará a los alumnos el código C de los diferentes módulos de la aplicación, aunque con funcionalidad errónea o limitada. Es responsabilidad del alumno modificar y/o completar el código fuente y compilarlo para obtener un comportamiento satisfactorio.

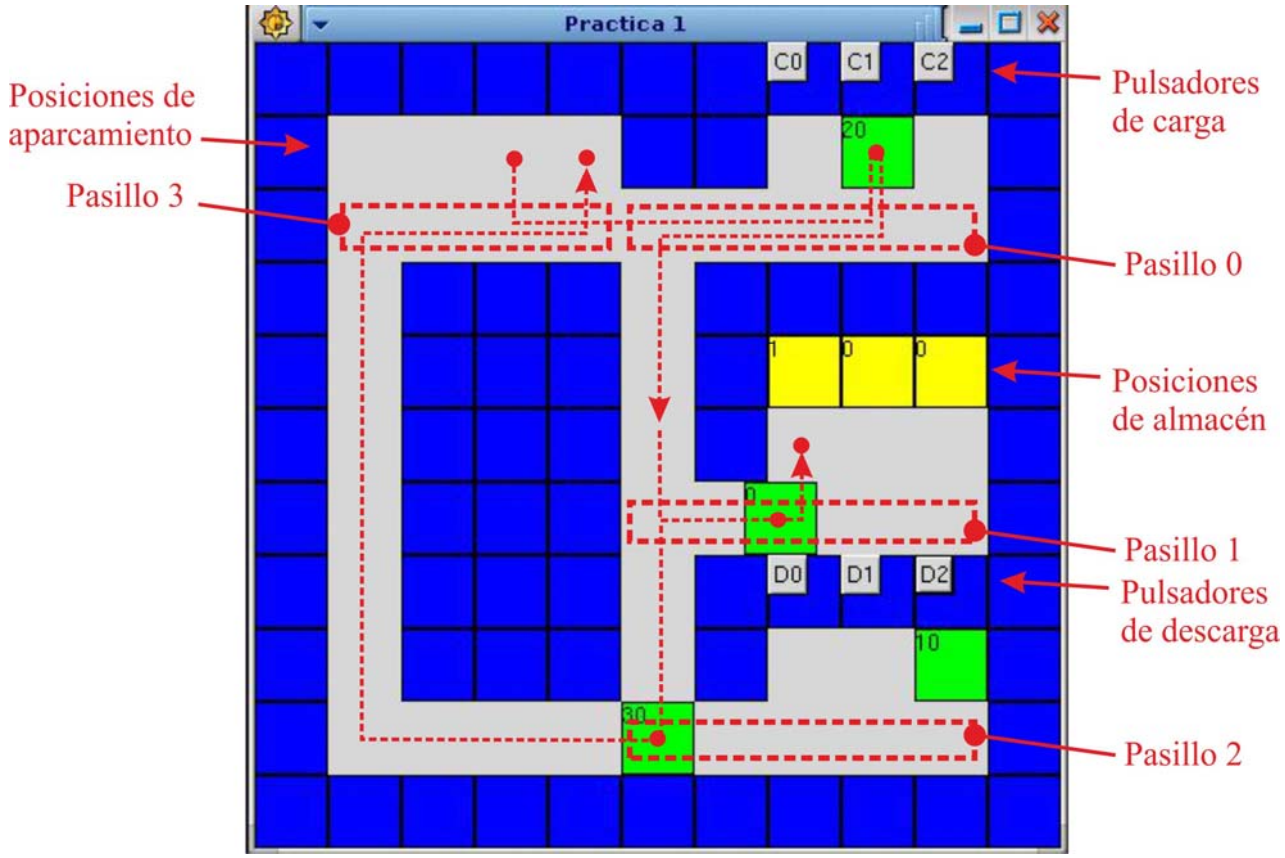


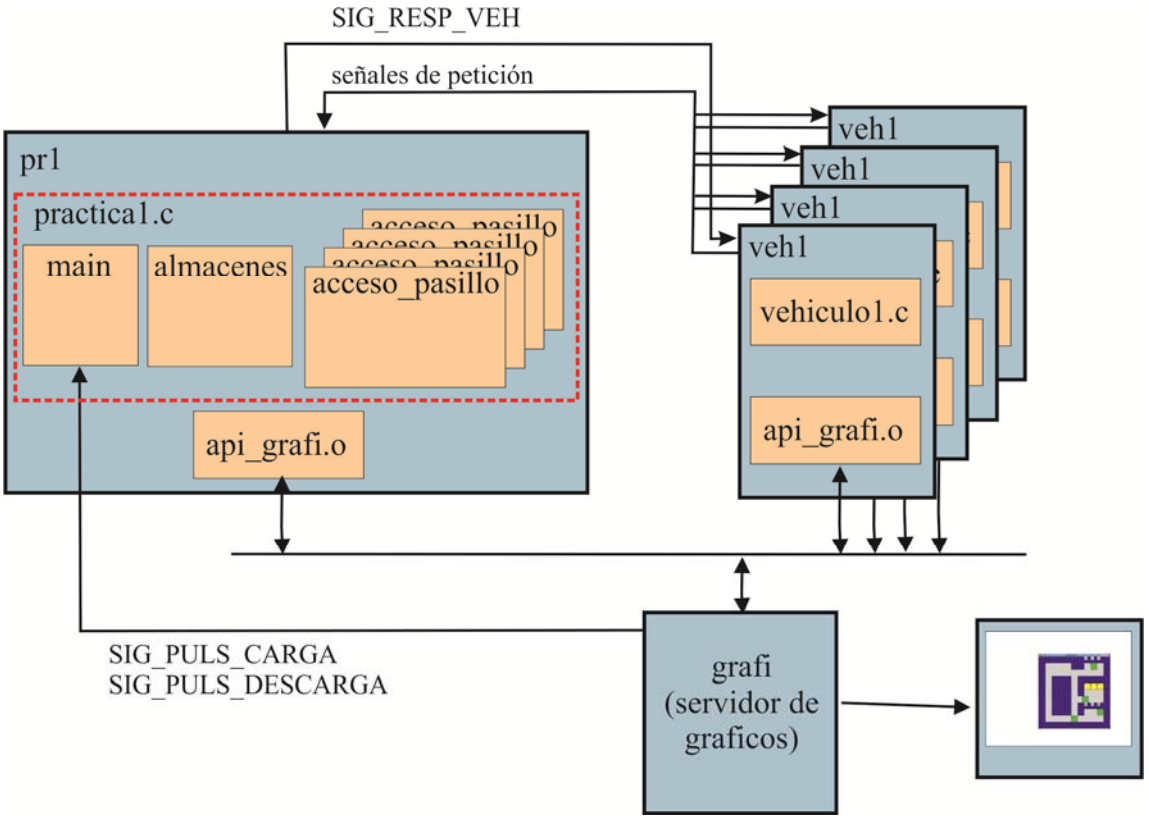
Figura 1: Representación gráfica

2. Funcionamiento del sistema

Los vehículos pueden realizar dos tipos de operaciones:

- Si se pulsa un pulsador de carga (C0 a C2) un vehículo carga una pieza del punto de carga donde se encuentra el pulsador y la descarga en una posición de almacenamiento.
- Si se pulsa un pulsador de descarga (D0 a D2) el vehículo toma una pieza de una posición de almacenamiento y la descarga en el punto de descarga del pulsador.

Por simplificar a la primera se le llamará "**CARGA**" y a las segunda "**DESCARGA**". Cada vez que un vehículo acaba su operación vuelve al aparcamiento. En la Figura 1 el vehículo 0 se dirige a la posición de almacenamiento 1 para cargar, el vehículo 2 está cargando en el pulsador C1, y el el vehículo 3 vuelve al aparcamiento. Siempre que un vehículo recibe un encargo intenta reservar una posición de almacenamiento con el estado adecuado (ocupada para **DESCARGA** o libre para **CARGA**); si no lo consigue no realiza el encargo. También debe pedir acceso a los pasillos 0, 1, 2 y 3 antes de entrar en ellos para evitar colisiones y liberarlo cuando salga de ellos.



s

Figura 2: Procesos del sistema

3. Arquitectura software

La aplicación está formada por

- El proceso **pr1** crea al resto de procesos del sistema, recibe señales de activación de los pulsadores y varias señales de petición de los vehículos, a los que responde con la señal **SIG_RESP_VEH**. Se compone de los siguientes hilos:

    - **main**: Recibe y ejecuta las peticiones de los pulsadores.
    - **almacenes**: Gestiona las posiciones de almacenamiento ejecutando peticiones de reserva, carga y descarga.
    - **N_PASILLOS** hilos **acceso_pasillo**, cada uno de los cuales gestiona el acceso a uno de los pasillos para que los vehículos consigan acceso exclusivo a los mismos.

    El ejecutable **pr1** se compone de los módulos **practica1.c** (a modificar por el alumno) y **api_grafi.o** (interfaz de gráficos).

- **N_VEH** procesos **veh1**, uno por vehículo. Todos comparten el mismo ejecutable. Los procesos **veh1** controlan el movimiento del vehículo interaccionando con el proceso **pr1** para solicitar encargo de operación a realizar, acceder a los pasillos y utilizar las posiciones de almacenamiento. Los procesos **veh1** intentan evitar colisiones utilizando los sónares y reintentan el movimiento cuando a pesar de ello se producen. Sólo se proporciona el ejecutable **veh1**, que ya funciona correctamente.
- El proceso **grafi**, que actúa como servidor de gráficos. Las funciones de **grafi** se utilizan por medio de **api_grafi.o**.

El alumno sólo debe completar el fichero **practica1.c**.

Señales de pulsadores: Los pulsadores de carga y descarga envían a **pr1** señales **SIG_PULS_CARGA** y **SIG_PULS_DESCARGA** respectivamente cuando se pulsan, acompañadas de un índice de pulsador entre **0** y **N_PULS-1**.

Algunos detalles sobre los procesos veh1:

- Cada proceso **veh1** recibe los siguientes argumentos:
    - Argumento 1: Identificador del vehículo (entero entre **0** y **N_VEH-1**).
    - Argumento 2: Fila del aparcamiento, donde también se crea el vehículo.
    - Argumento 3: Columna del aparcamiento, donde también se crea el vehículo.
- Los procesos **veh1** intenta conectarse con la ventana gráfica “Practica 1”.
- Los procesos **veh1** envían las siguientes señales de petición a **pr1**:
    - **SIG_PEDIR_ENCARGO**:
        - Enviada cuando el vehículo llega al aparcamiento. Dato: Identificador del vehículo entre **0** y **N_VEH-1**.
        - Respuesta esperada: Dos señales **SIG_RESP_VEH** consecutivas, una con el código de la operación a realizar (**CARGA** o **DESCARGA**) y otra con el número del pulsador donde debe dirigirse a realizarla.
    - **SIG_RESERVAR_VACIO** y **SIG_RESERVAR_LLENO**:
        - Enviadas para reservar una posición de almacenamiento vacía o llena, respectivamente. Dato: Identificador del vehículo entre **0** y **N_VEH-1**.
        - Respuesta esperada: Una señal **SIG_RESP_VEH** con el índice de la posición de almacenamiento concedida (entre **0** y **N_ALM-1**), o bien -1 si no hay ninguna disponible. Las posiciones reservadas no deben volver a concederse hasta que se realice una operación de carga o descarga sobre ellas.
    - **SIG_CARGAR** y **SIG_DESCARGAR**:
        - Enviadas para realizar una operación de carga o descarga sobre una posición de almacenamiento. Dato: Índice de la posición entre **0** y **N_ALM-1**.
        - Respuesta esperada: Ninguna.

- **SIG_PEDIR_P0** a **SIG_PEDIR_P3**:
  - Enviadas para pedir acceso a un pasillo del 0 al 3; tienen asignados números consecutivos de señal. Dato: Identificador del vehículo entre **0** y **N_VEH-1**.
  - Respuesta esperada: señal **SIG_RESP_VEH**. Su dato se ignora.

- **SIG_LIB_P0** a **SIG_LIB_P3**:
  - Enviadas para liberar un pasillo del 0 al 3; tienen asignados números consecutivos de señal. Dato: Identificador del vehículo entre **0** y **N_VEH-1**.
  - Respuesta esperada: Ninguna.

Explicación del arranque del sistema de soporte gráfico:

El arranque del sistema gráficos (función **inic_graficos**) incluye la creación de varios hilos en el proceso **pr1** que se utilizan para que funcione la comunicación con el servidor de gráficos. Por otro lado en QNX no es posible utilizar **fork** en un proceso multihilo, así que el proceso que realiza el arranque no puede crear procesos hijos posteriormente. Por eso es necesario crear los procesos de control de vehículos antes de ejecutar **inic_graficos**.

4. Trabajo del alumno

El trabajo a realizar consta de dos partes:

- **Parte obligatoria**: A realizar en el laboratorio en horario de prácticas. Consiste en completar y corregir el código que se proporciona para conseguir la funcionalidad descrita en los puntos anteriores.
- **Parte voluntaria**: El alumno podrá realizar mejoras y desarrollos adicionales de la práctica, que influirán positivamente en la calificación final. Como ejemplo se proponen las siguientes modificaciones:

  - Añadir una condición de sobretiempo que permita activar los pulsadores cuando ha pasado un cierto intervalo de tiempo desde el comienzo o desde que se activaron por última vez.
  - Añadir un pulsador para parar y realizar una parada ordenada del sistema de modo que todos los vehículos acaben la operación en curso y terminen aparcados.
  - Cambiar la gestión de reservas del almacén de modo que si no hay posiciones que reservar **pr1** se limite a no responder al vehículo hasta que las haya, en lugar de denegar la reserva (-1).
  - Evitar la condición de bloqueo de vehículos que puede aparecer cuando se pulsa un pulsador varias veces sin dar tiempo a que se realice la primera operación (el vehículo que sale puede impedir el paso al que entra).
  - Realizar una implementación de los procesos de control de vehículos **veh1**.

Deberá entregarse la siguiente documentación:

- Memoria del trabajo realizado, incluyendo errores detectados y acciones para corregirlos, junto con una explicación detallada del funcionamiento del programa. **No se trata de repetir el enunciado**.
- Ficheros fuentes, de manera que pueda comprobarse el funcionamiento de la aplicación.

5. Ficheros disponibles

- **practica1.c**: Fichero fuente para el proceso **pr1** (a completar y corregir).
- **veh1**: Fichero ejecutable para los procesos **veh1**.
- **practica1.h**: Cabecera común para los programas de la práctica 1.
- **api_grafi.o**: Fichero objeto con las funciones de interfaz para el servidor de gráficos.
- **api_grafi.h**: Cabecera para las funciones de **api_grafi.o**.
- **grafi**: Ejecutable del servidor de gráficos.
- **ej_referencia/pr1**: Ejecutable de **pr1** sin fallos, para utilizar como referencia.

Para compilar el ejecutable **pr1** pueden utilizarse los siguientes comandos en QNX 6.3.2:

gcc –o pr1 practica1.c api_grafi.o

En Ubuntu el comando es algo distinto:

gcc –o pr1 practica1.c api_grafi.o –lrt -lpthread

6. Cabeceras

6.1. Cabecera practica1.h

```
/* Cabecera para practica 1 INFI (11/2015) */
/* Copyright (C) Joaquin Ferruz Melero 2015 */

#ifndef   __PRACTICA1_H__
#define   __PRACTICA1_H__

/* Acceso a interfaz grafica */

#include "api_grafi.h"

/* Senales de pulsadores recibidas por pr1 */

#define SIG_PULS_CARGA      SIGRTMIN   /* Senal generada por los pulsadores de carga */
#define SIG_PULS_DESCARGA   SIGRTMIN+1 /* Senal generada por los pulsadores de descarga */

/* Senales procedentes de vehiculos y recibidas por pr1 */

#define SIG_PEDIR_ENCARGO   SIGRTMIN+2   /* Pedir encargo de operacion
                                            de carga o descarga */
#define SIG_RESERVAR_VACIO  SIGRTMIN+3   /* Reservar posicion de almacen vacia */
#define SIG_RESERVAR_LLENO  SIGRTMIN+4   /* Reservar posicion de almacen llena */
#define SIG_DESCARGAR       SIGRTMIN+5   /* Descargar en posicion de almacen */
#define SIG_CARGAR          SIGRTMIN+6   /* Cargar en posicion de almacen */
#define SIG_PEDIR_P0        SIGRTMIN+7   /* Pedir pasillo 0 */
#define SIG_PEDIR_P1        SIGRTMIN+8   /* Pedir pasillo 1 */
#define SIG_PEDIR_P2        SIGRTMIN+9   /* Pedir pasillo 2 */
```

```
#define SIG_PEDIR_P3         SIGRTMIN+10  /* Pedir pasillo 3 */
#define SIG_LIB_P0           SIGRTMIN+11  /* Liberar pasillo 0 */
#define SIG_LIB_P1           SIGRTMIN+12  /* Liberar pasillo 1 */
#define SIG_LIB_P2           SIGRTMIN+13  /* Liberar pasillo 2 */
#define SIG_LIB_P3           SIGRTMIN+14  /* Liberar pasillo 3 */

/* Senal de respuesta enviada por pr1 y recibida por el vehiculo */

#define SIG_RESP_VEH         SIGRTMIN     /* Respuesta comun para cualquier peticion */

/* Tipo de operacion para enviar con SIG_RESP_VEH */

enum tipo_op {  CARGA =      0, /* Cargar en zona de pulsadores de carga */
                DESCARGA =   1  /* Descargar en zona de pulsadores de descarga */
             };

/* Numero de elementos */

#define N_PULS               3  /* Numero de pulsadores de carga y descarga */
#define N_VEH                4  /* Numero de vehiculos */
#define N_ALM                3  /* Numero de posiciones de almacenamiento */
#define N_PASILLOS           4  /* Numero de pasillos */

/* Dimensiones del dibujo */

#define ANCHO                11
#define ALTO                 11
#define ESCALA_DIB           40

/* Filas y columnas */

#define FILA_PULS_CARGA      0
#define COL_PULS_CARGA       7
#define FILA_APARCAMIENTO    1
#define COL_APARCAMIENTO     1
#define FILA_ALMACEN         4
#define COL_ALMACEN          7
#define FILA_PULS_DESCARGA   7
#define COL_PULS_DESCARGA    7

#define COL_IDA              5
#define FILA_VUELTA          ALTO-2
#define COL_VUELTA           1

/* Mapa:
    C: camino
    P: Pared
 */

#define P PARED                    /* Definido en api_grafi.h */
#define C PASILLO                  /*   "              "           "     */

int mapa[ALTO][ANCHO] =
                  {
                    {P, P, P, P, P, P, P, P, P, P, P},
                    {P, C, C, C, C, P, P, C, C, C, P},
                    {P, C, C, C, C, C, C, C, C, C, P},
                    {P, C, P, P, P, C, P, P, P, P, P},
                    {P, C, P, P, P, C, P, C, C, C, P},
                    {P, C, P, P, P, C, P, C, C, C, P},
                    {P, C, P, P, P, C, C, C, C, C, P},
                    {P, C, P, P, P, C, P, P, P, P, P},
                    {P, C, P, P, P, C, P, C, C, C, P},
                    {P, C, C, C, C, C, C, C, C, C, P},
                    {P, P, P, P, P, P, P, P, P, P, P}
                  };

#endif
```