Informática Industrial

4º GIERM, curso 2016-17

Práctica 4

Colas de mensajes

1. Introducción

El trabajo consiste en completar una aplicación en C utilizando las llamadas POSIX para controlar el sistema de la Figura 1. La comunicación y la sincronización entre los hilos y procesos de la aplicación se realizará mediante **colas de mensajes POSIX**.

Para el desarrollo de la práctica 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. La práctica puede realizarse sobre QNX 6.3, Kubuntu 12.04 o Ubuntu 14.04 utilizando los ficheros adecuados.

Figura 1: Representación gráfica

2. Funcionamiento del sistema

En el sistema de la Figura 1 existen dos tipos de almacenes: entrada y salida. Sólo hay un almacén de salida, pero existen **N_ALM_IN** almacenes de entrada. Los almacenes de entrada tienen capacidad para **una sola pieza**, mientras que el de salida tiene capacidad para varias, hasta un máximo de **MAX_PIEZAS_OUT**. Para evitar conflictos en el acceso a las piezas de la entrada y a los lugares para piezas de la salida existirá un procedimiento de reserva para ambos. Existen también **N_VEH** vehículos que permanecerán en su aparcamiento mientras no puedan realizar una operación de transporte:

- Si el vehículo está vacío y en el aparcamiento, espera hasta reservar una pieza en alguno de los almacenes de entrada.
- Si el vehículo contiene una pieza y está en el aparcamiento, espera hasta reservar un lugar en el almacén de salida.

Por simplificar a la primera se le llamará "**carga**" y a las segunda "**descarga**":

- En una operación de carga el vehículo reserva una pieza disponible, se dirige al almacén de entrada correspondiente, carga y luego toma el camino de vuelta (ver Figura 1). Cuando llega a la bifurcación determina si es posible descargar la pieza en el almacén de salida; si lo es, reserva un lugar, avanza por el camino del almacén de salida, descarga y vuelve al aparcamiento vacío. Si no lo es sigue su camino y vuelve al aparcamiento con su carga.
- En una operación de descarga el vehículo sigue el mismo camino que en una operación de carga, pero ahora no utiliza ningún almacén de entrada, y se dirige directamente al de salida, donde ya ha reservado un lugar para descargar. Luego vuelve vacío a su aparcamiento.

Los pulsadores **IN0** a **IN3** permiten aportar piezas a los almacenes de entrada (nunca más de una), mientras que el pulsador **OUT** permite retirar piezas del almacén de entrada.

3. Arquitectura software

La aplicación está formada por

- El proceso **pr4**, cuyo ejecutable se compone de los módulos **practica4.c**, y **api_grafi.o** (interfaz de gráficos). Este proceso envía y recibe mensajes de petición y envía mensajes de respuesta utilizando las colas de mensajes POSIX de la figura. En el proceso **pr4** existen lo siguientes hilos:
    - **main**: Arranca la interfaz de gráficos de la ventana principal, crea las colas de mensajes, los procesos **veh_pr4** y el hilo **hilo_pet**. Finalmente se dedica a tratar las señales que generan los pulsadores y de enviar los correspondientes comandos a la cola **/cola_pet_pr4**.
    - **hilo_pet**: Hilo de recepción y ejecución de peticiones. Actúa como un servidor que lee peticiones de la cola **/cola_pet_pr4** y las ejecuta, o bien las deja pendientes hasta el momento adecuado si no es posible ejecutarlas inmediatamente. También envía respuestas a las peticiones que lo requieren utilizando las colas **/cola_resp_pr4_0** a la **/cola_resp_pr4_<N_VEH-1>**.

- **N_VEH** procesos de control de vehículo creados con el ejecutable **veh_pr4**. Las diferencias en su funcionamiento proceden de los diferentes argumentos que reciben. Cada hilo controla un vehículo distinto y se sincroniza y comunica con el resto enviando peticiones y recibiendo respuestas de **pr4**. Toda esta funcionalidad está ya resuelta, pero depende de que el intercambio de mensajes se realice adecuadamente.
- El proceso **grafi**, que actúa como servidor de gráficos. Las funciones de **grafi** se utilizan por medio de **api_grafi.o**.

El alumno debe completar el fichero **practica4.c** para que la funcionalidad sea la descrita y se corresponda con el ejecutable de referencia del que también se dispone (**ej_referencia/pr4**). El código de los procesos de vehículo no se suministra, sólo el ejecutable **veh_pr4**.

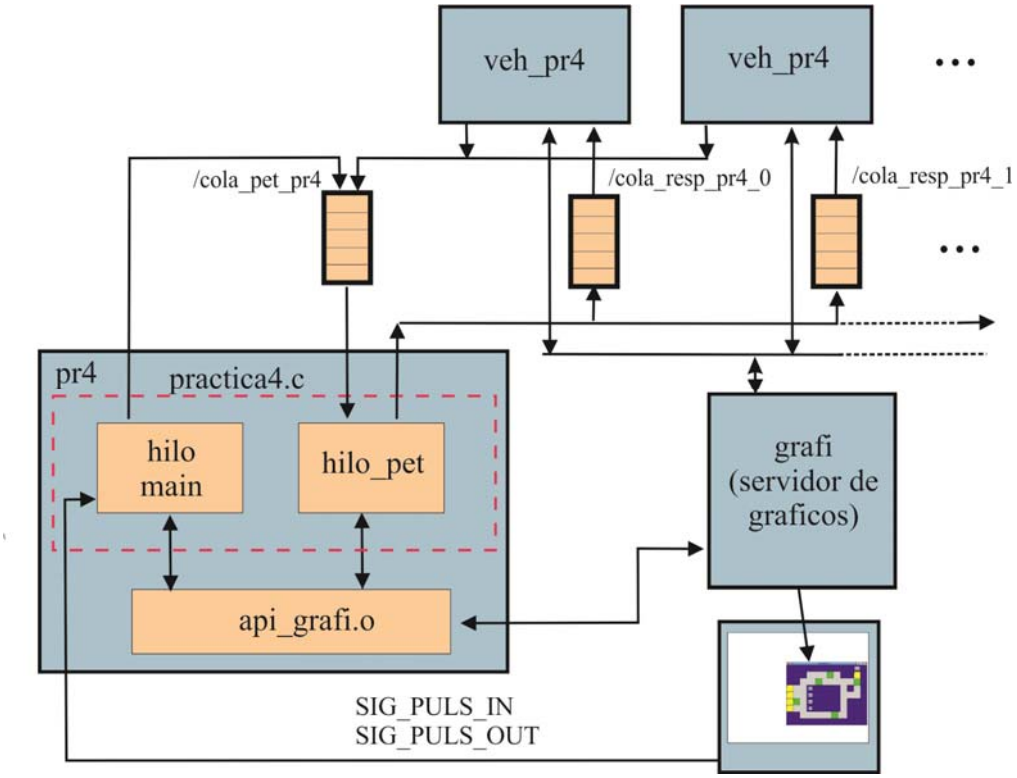

Figura 2: Procesos e hilos del sistema

Señales de pulsadores:

Los pulsadores para incrementar o decrementar los almacenes de entrada y salida envían a **pr4** señales **SIG_PULS_IN** y **SIG_PULS_OUT** respectivamente cuando se pulsan. Las señales **SIG_PULS_IN** van acompañadas de un índice de pulsador entre **0** y **N_ALM_IN-1**.

Argumentos de los procesos veh_pr4:

- Cada proceso **veh_pr4** recibe los siguientes argumentos de línea de comandos:
    - Argumento 1: Fila inicial y de aparcamiento (número entero).
    - Argumento 2: Columna inicial y de aparcamiento (número entero).
    - Argumento 3: Identificador del vehículo (entero entre **0** y **N_VEH-1**).
    - Argumento 4: Nombre de la cola de mensajes de peticiones (cadena de caracteres).
    - Argumento 5: Nombre de la cola de mensajes de respuestas (cadena de caracteres).

Mensajes de petición y respuesta:

- Los mensajes enviados por la cola de peticiones consisten en estructuras de tipo **msg_pet**, y los recibidos por la cola de respuestas son estructuras de tipo **msg_resp**, ambas definidas en la cabecera **practica4.h**. El tipo de la respuesta (cuando es necesaria) es

siempre **idéntico al tipo de petición a la que pretende responder**. A continuación se describe el intercambio de mensajes.

- **PET_PEDIR_PASILLO**:
  - Enviada por **veh_pr4** para pedir acceso a uno de los tres pasillos.
  - Otros campos válidos: **nveh** y **npasillo**.
  - Respuesta esperada: Del mismo tipo que la petición, cuando el pasillo se haya reservado.

- **PET_LIB_PASILLO**:
  - Enviada para liberar un pasillo de los tres pasillos
  - Otros campos válidos: **nveh** y **npasillo**.
  - Respuesta esperada: Ninguna.

- Tipo **PET_ESP_ENTRADA**:
  - Enviada por **veh_pr4** cuando el vehículo quiere esperar a reservar una pieza en los almacenes de entrada.
  - Otros campos válidos: **id_veh**.
  - Respuesta esperada: Cuando se ha realizado la reserva, incluyendo número de almacén de entrada donde está la pieza reservada (**nalm**).

- Tipo **PET_RES_SALIDA**:
  - Enviada por **veh_pr4** para intentar reservar una posición de almacenamiento en el almacén de salida.
  - Otros campos válidos: **id_veh**.
  - Respuesta esperada: inmediata, indicando si se ha podido reservar o no la posición (**sal_reservada**).

- Tipo **PET_ESP_SALIDA**:
  - Enviada por **veh_pr4** para esperar a reservar una posición de almacenamiento en el almacén de salida.
  - Otros campos válidos: **id_veh**.
    Respuesta esperada: Cuando se ha realizado la reserva, incluyendo el indicador de salida reservada (**sal_reservada**) que siempre estará a 1.

- **PET_CARGA**:
  - Enviada por **veh_pr4** para cargar una pieza de un almacén de entrada.
  - Otros campos válidos: **id_veh** y **nalm**.
  - Respuesta esperada: Ninguna.

- **PET_DESCARGA**:
  - Enviada por **veh_pr4** para descargar una pieza en el almacén de salida.
  - Otros campos válidos: **id_veh**.
  - Respuesta esperada: Ninguna.

- **PET_INC_IN**:
  - Enviada por el hilo **main** de **pr4** para incrementar un almacén de entrada.
  - Otros campos válidos: **id_veh** (siempre a **N_VEH**) y **nalm**.
  - Respuesta esperada: Ninguna.

- **PET_DEC_OUT**:
  - Enviada por el hilo **main** de **pr4** para decrementar el almacén de salida.
  - Otros campos válidos: **id_veh** (siempre a **N_VEH**).
  - Respuesta esperada: Ninguna.

Arranque del sistema de soporte gráfico:

El arranque del sistema gráficos (función **abrir_graficos** en **inic_graficos**) incluye la creación de varios hilos en el proceso **pr4** 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 vehículo antes de ejecutar **inic_graficos**. Por otra parte la función **ini_estado** utiliza los gráficos, y por tanto hay que ejecutarla después de **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. En particular es necesario que ningún hilo se quede realizando continuamente operaciones durante un tiempo prolongado de modo que lleve el procesador al 100%.
- **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, realizándolas con los medios de comunicación y sincronización utilizados en esta práctica:

  - Crear un nuevo proceso que abra un teclado en una ventana gráfica separada, y envíe las pulsaciones de las teclas a **pr4** por medio de una cola de mensajes. El teclado tendría al menos las mismas teclas que la ventana actual.
  - Cambiar el método de asignación de piezas de modo que se escojan con algún criterio lógico si hay más de una, por ejemplo dando preferencia a los almacenes que menos se han usado.
  - Añadir un pulsador para realizar una parada ordenada del sistema de modo que todos los vehículos acaben la operación en curso y terminen aparcados en su aparcamiento y vacíos antes de cerrar los gráficos y terminar la aplicación.
  - Modificar la aplicación para que se admita más de una pieza en los almacenes de entrada, sin dar lugar a colisiones entre vehículos.
  - Modificar la gestión del acceso a pasillos para minimizar las paradas de los vehículos sin dar lugar a colisiones.
  - Realizar una implementación de los procesos **veh_pr4**.

Deberá entregarse la siguiente documentación:

- Memoria del trabajo realizado, incluyendo
  - Explicación detallada del funcionamiento general del programa. **No se trata de repetir el enunciado**.
  - Explicación de los errores detectados y las acciones realizadas para corregirlos.

- Ficheros fuentes, de manera que pueda comprobarse el funcionamiento de la aplicación.

5. Ficheros disponibles

- **practica4.c**: Fichero fuente para **main** del proceso **pr4** (a completar y corregir).
- **practica4.h**: Cabecera común para los programas de la práctica 4.
- **veh_pr4**: Ejecutable para el proceso de control de vehículo.
- **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/pr4**: Ejecutable de **pr4** sin fallos, para utilizarlo como referencia.

Para compilar el ejecutable **pr4** pueden utilizarse los siguientes comandos en QNX 6.3.2:

gcc –o pr4 practica4.c api_grafi.o

Si en el directorio sólo están los ficheros de la práctica basta con hacer: **gcc –o pr4 *.c *.o**

En Ubuntu el comando es algo distinto:

gcc –o pr4 practica4.c api_grafi.o –lrt -lpthread

Del mismo modo, si en el directorio sólo están los ficheros de la práctica este comando podría abreviarse a **gcc –o pr4 *.c *.o –lrt -lpthread**

6. Cabecera practica4.h

```
/* Cabecera para practica 4 de INFI-GIERM (12/2016)  */
/* Copyright (C) Joaquin Ferruz Melero 2016 */

#ifndef   __PRACTICA4_H__
#define   __PRACTICA4_H__

/* Acceso a interfaz grafica */

#include "api_grafi.h"

#define N_VEH           5 /* Numero de vehiculos */
#define N_ALM_IN        4 /* Numero de posiciones de almacenamiento */
#define N_PASILLOS      3 /* Numero de pasillos */
#define MAX_PIEZAS_OUT  3 /* Maximo numero de piezas en almacen de salida */

/* Pasillos (ver figura) */

enum tipo_pasillo {APARCAMIENTO, ENTRADA, SALIDA};

/* Senales de pulsadores */

#define SIG_PULS_IN  SIGRTMIN    /* Senal para los pulsadores de entrada */
#define SIG_PULS_OUT SIGRTMIN+1  /* Senal para los pulsadores de salida */

/* Mensaje de peticion (recibido por pr4) */
```

```
enum tipo_pet {
     PET_PEDIR_PASILLO, /* Pedir acceso al pasillo */
     PET_LIB_PASILLO,   /* Liberar pasillo */
     PET_ESP_ENTRADA,   /* Pedir reserva de entrada */
     PET_RES_SALIDA,    /* Pedir reserva de salida no bloqueante */
     PET_ESP_SALIDA,    /* Pedir reserva de salida */
     PET_CARGA,         /* Cargar de almacen de entrada */
     PET_DESCARGA,      /* Descargar en almacen de salida */
     PET_INC_IN,        /* Incrementar almacen de entrada */
     PET_DEC_OUT        /* Decrementar almacen de salida */
    };

struct msg_pet {
   enum tipo_pet tipo;  /* Tipo de peticion */
    int npasillo;       /* Numero de pasillo (solo con PET_LIB_PASILLO y
                           PET_PEDIR_PASILLO) */
   int nalm;            /* Numero de almacen (solo con PET_CARGA y
                           PET_INC_IN */
   int id_veh;          /* ID del vehiculo que envia la peticion */
};

/* Mensaje de respuesta (enviado por pr4) */

struct msg_resp {
   enum tipo_pet tipo;  /* Tipo de peticion a la que se responde */
   int nalm;            /* Numero del almacen reservado
                           (solo con PET_ESP_ENTRADA) */
   int sal_reservada;   /* Indicador de salida reservada (1) o no reservada
                           (0) (solo con PET_RES_SALIDA o PET_ESP_SALIDA) */
};

/* Dimensiones del dibujo */

#define ANCHO        12
#define ALTO         9
#define ESCALA_DIB   40

/* Filas y columnas */

#define FILA_PULS_IN      3.25   /* Fila de los pulsadores de entrada */
#define COL_PULS_IN       3.25   /* Columna del primer pulsador de entrada */
#define FILA_PULS_OUT     0.25   /* Fila del pulsador de salida */
#define COL_PULS_OUT      10.15  /* Columna del pulsador de salida */

#define FILA_APARCAMIENTO 1      /* Fila de los aparcamientos */
#define COL_APARCAMIENTO  3      /* Columna del aparcamiento 0 */
#define FILA_ALM_IN       3      /* Fila del primer almacen de entrada */
#define COL_ALM_IN        0      /* Col. de los almacenes de entrada */
#define FILA_ALM_OUT      1      /* Fila del almacen de salida */
#define COL_ALM_OUT       10     /* Col. del almacen de salida */
#define FILA_DESV_OUT     4      /* Fila de la desviacion para ir a la salida */
#define COL_DESV_OUT    ANCHO-2  /* Col. de la desviacion para ir a la salida */
#define FILA_VUELTA     ALTO-2   /* Fila del pasillo inferior para volver al
                                    aparcamiento */
#define COL_VUELTA      ANCHO-4  /* Col. del pasillo de la derecha para volver al
                                    aparcamiento */
#endif
```