Universidad Autónoma de Madrid

Departamento de Informática

# Fundamentos de Bases de Datos
# Práctica - 2

Jorge de las Peñas Plana

# Registro de Cambios

| **Versión**[1] | **Fecha** | **Autor** | **Descripción** |
|---|---|---|---|
| 1.0 | 01.09.2025 | JPP | Primera versión. |

[1]La asignación de versiones se realizan mediante 2 números $X.Y$. Cambios en $Y$ indican aclaraciones, descripciones más detalladas de algún punto o traducciones. Cambios en $X$ indican modificaciones más profundas que o bien varían el material suministrado o el contenido de la práctica.

# Índice

# 1. Objetivo

El usuario final nunca accede directamente a la base de datos (no lanza peticiones SQL), si no que interactúa con una interfaz de usuario que a su vez utiliza una capa lógica (software) que construye las consultas y las lanza. En esta práctica experimentaremos con el acceso a bases de datos desde un programa escrito en lenguaje C. En particular, usaremos la librería ODBC para crear un programa que actuará sobre la base de datos `flights` que hemos proporcionado en la primera práctica. Todo lo aprendido en esta práctica es casi directamente aplicable a cualquier lenguaje de programación que disponga de una implementación de tipo ODBC, JDBC o similar.

# 2. Acceso a la base de datos desde C

Como ya se comentó, en esta práctica usaremos ODBC para crear un programa en C que ejecute ciertas operaciones en la base de datos. ODBC es una librería para enviar consultas a bases de datos y gestionar los resultados. Podéis encontrar documentación relacionada con ODBC en el URL `http://www.unixodbc.org/`. En particular, es interesante el "Programming Manual tutorial" (`http://www.unixodbc.org/doc/ProgrammerManual/Tutorial/`). Otro enlace a resaltar es "ODBC from C Tutorial Part 1" (`https://www.easysoft.com/developer/languages/c/odbc_tutorial.html`).

Adicionalmente, se os proporcionan los ficheros `odbc.h` y `odbc.c` que contienen las funciones más comunes de conexión, desconexión e impresión de errores, así como una colección de ejemplos que usan estas funciones (véase fichero *Ejemplos odbc* accesible en *Moodle*).

# 3. Programa a implementar

Se requiere escribir en lenguaje C un programa que usando un sistema de menús y formularios permita: (1) buscar vuelos entre dos ciudades y (2) solicitar una tarjeta de embarque para todos los vuelos relacionados con una compra (*bookings*).

## 3.1. Sistema de menús

Para esta práctica os proporcionamos un interfaz de usuario implementado usando la biblioteca *ncurses* y vosotros tendréis que encargaros de realizar las consultas a la base de datos. *Ncurses* es una biblioteca de programación que provee un API que permite al programador escribir interfaces basadas en texto y crear menús y formularios de forma sencilla. El interfaz proporcionado divide la terminal en cuatro grandes áreas tal y como se muestra en la figura 1. Como en la práctica anterior los ficheros conteniendo el código de ejemplo, se encuentran en Moodle .

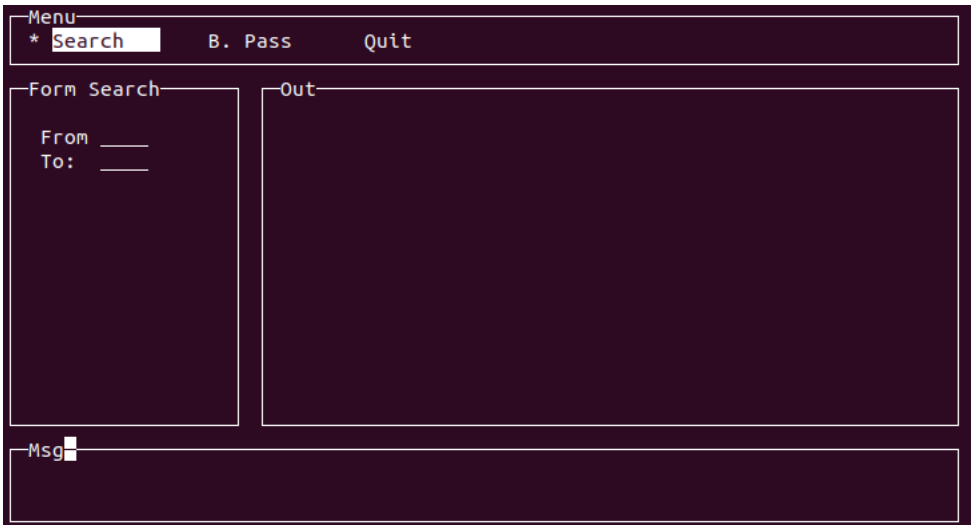


Figura 1: Captura de pantalla mostrando el interfaz de menús a utilizar en esta práctica.

A continuación se describe el interfaz (que ya está implementado) y posteriormente detallaremos las consultas a implementar (las cuales tendréis que implementar e integrar en el interfaz). El interfaz puede generarse ejecutando la utilidad *make*.

### Parte superior: menú

La parte superior contiene un menú que permite seleccionar la operación a realizar, bien sea buscar un vuelo entre dos aeropuertos (*Search*) o solicitar una tarjeta

de embarque (*B. Pass*). Mediante el uso de las flechas del teclado *izquierda* y *derecha* es posible navegar por las opciones del menú. La aplicación se cierra cuando se selecciona la opción *Quit* y se presiona la tecla *Enter*.

En el resto de este documento nos referiremos a la parte de la pantalla como *menu_window*.

### Parte izquierda: formulario

En la parte izquierda aparece el formulario donde se solicita la información necesaria para buscar los vuelos (aeropuerto de origen y destino) o emitir la tarjeta de embarque (identificador de reserva -*booking_ref*-). En cada momento aparece el formulario relacionado con la opción seleccionada en el menú situado en *menu_window*.

Las flechas arriba y abajo del teclado posibilitan el movimiento entre los diferentes campos del formulario. Finalmente, al presionar la tecla *Enter* el interfaz lee los datos contenidos en el formulario e invoca una función donde se debe realizar la consulta oportuna a la base de datos y muestra el resultado en la parte derecha de la terminal. En la versión que os entregamos no se realiza ninguna consulta sino que se muestra un listado con los Diez Mandamientos tal y como aparecen en la Biblia del Rey Jorge.

En el resto de este documento nos referiremos a esta parte de la pantalla como *form_window* y a la parte derecha de la pantalla como *out_window*.

### Parte derecha: resultado de la consulta

En esta zona se mostrará el resultado de las consultas realizadas a la base de datos, las cuales se detallarán en la siguiente sección. Presionando el tabulador es posible mover el foco de la ventana *form_window* a *out_window* y viceversa. Si el foco se encuentra en la parte derecha, las teclas arriba y abajo permiten navegar entre las diferentes respuestas y si se presiona *Enter* aparece en la zona inferior información especifica relacionada con la respuesta seleccionada.

### Zona inferior

Esta parte de la terminal muestra mensajes de error o información adicional.

En el resto de este documento nos referiremos a esta parte de la pantalla como *message_window*.

#### Mapeo del teclado

El menú usa el teclado como entrada. Los caracteres especiales tales como los generados por las flechas y página arriba y abajo pueden ser conflictivos. En general, si antes de lanzar el programa, se teclea en la terminal la orden `export TERM=gnome` el mapeo del teclado será correcto. De todas formas para evitar posibles problemas, hemos mapeado las teclas potencialmente peligrosas a caracteres ASCII tal y como se ve en la tabla 2. Por ejemplo, si la tecla *fecha arriba* no funciona podéis usar el carácter *+*.

| | |
|---|---|
| ↑ | + |
| ↓ | − |
| ← | < |
| → | > |

Tabla 2: Tabla mostrando los caracteres especiales (izquierda) y sus caracteres ASCII equivalentes (derecha).

## 3.2. Consultas a realizar

El sistema debe ser capaz de buscar vuelos y emitir tarjetas de embarque.

### 3.2.1. Realización de búsquedas

Se solicita que modifiquéis el código de forma que cuando se seleccione en el menú la opción *search* y se rellenen los campos *From* y *To* con los identificadores de los aeropuertos de origen y destino (*airport_code*), aparezca un listado con todos los vuelos disponibles entre ambos aeropuertos.

Para poder realizar la consulta apropiadamente además de los aeropuertos de origen y destino es necesario conocer la fecha de salida del primer vuelo. Este campo

no existe en el formulario así que tendréis que añadirlo, usad una cadena con el formato 'YYYY-MM-DD' (por ejemplo 2021-03-21).

El resultado de la consulta debe:

- Contener tanto los vuelos directos como los que se puedan realizar usando hasta un transbordo (esto es, hasta dos vuelos encadenados) siempre que la duración total del viaje no sea superior a 24 horas.
- Mostrar la fecha de salida del primer vuelo y de llegada del último vuelo (*scheduled_departure*, *scheduled_arrival*).
- Mostrar el número de vuelos de conexión.
- Mostrar el número de asientos libres (si el trayecto requiere del uso de dos aviones se mostrarán los datos relacionados con el avión con MENOS plazas libres)
- No se mostrarán los vuelos que no tengan asientos disponibles.
- Los vuelos se ordenarán de forma ascendente por tiempo transcurrido desde la salida del aeropuerto de origen hasta la llegada al aeropuerto de destino.

Como se describió en la sección anterior es posible cambiar el foco entre la ventana que contiene los formularios (*form_window*) y la que muestra los resultados (*out_window*). Si el foco está en esta última usando las flechas arriba/abajo del teclado debe ser posible navegar entre las diferentes opciones de vuelo. Al presionar la tecla *Enter* debe aparecer en la ventana *message_window* los detalles del itinerario solicitado. Estos detalles incluirán la fecha de salida y llegada de cada uno de los vuelos así como su identificador (*flight_id*) y el código de la nave (*aircraft_code*).

Igualmente se usará la ventana *message_window* para mostrar al usuario cualquier mensaje informativo o de error como pueda ser la inexistencia del código de aeropuerto utilizado, la falta de datos en el formulario, la inexistencia de vuelos, etc.

### 3.2.2. Emisión de tarjetas de embarque

Esta opción posibilita la asignación de tarjetas de embarque para una compra dada. El usuario introducirá en el formulario el identificador de compra (*book_ref*) y el sistema asignará un asiento a cada uno de los vuelos asociados a esta compra. Solo se asignarán asientos a los vuelos que todavía no tengan tarjeta de embarque asignada. Para simplificar el proceso la asignación se realiza de forma automática eligiéndose el primer asiento disponible. En particular, se ordenarán todos los billetes (*ticket_flights*) de forma ascendente usando el atributo *ticket_no* y para cada vuelo se ordenarán los asientos disponibles de forma ascendente usando los atributos (*aircraft_code*, *seat_no*) eligiéndose el primer asiento disponible. La asignación de asiento debe consignarse en la base de datos actualizando la tabla *boarding_passes* y también deben mostrarse en la ventana *out_window* mostrando una línea por asiento reservado. Cada linea contendrá los atributos (*passenger_name*, *flight_id*, *scheduled_departure* and *seat_no*), el nombre del pasajero se truncará a los primeros 20 caracteres.

Puesto que es posible que dos operadores soliciten el mismo asiento de forma concurrente diseña tu aplicación para que no sea posible asignar la misma plaza dos veces a dos pasajeros distintos.

**Importante**: por motivos de seguridad cualquier consulta que incorpore un parámetro introducido por el usuario se realizará usando "prepared statements" (vease odbc-ejemplo4.c)

**Importante:** para obtener una calificación superior a 8 deberéis implementar paginación en la ventana *out_window*. Esto es, en caso de que el número de resultados sea superior a los que se pueden mostrar en la ventana de la derecha, se mostrarán solo aquellos que quepan. Utilizando las teclas "avance página" y "retroceso página", el usuario debe poder moverse a la página siguiente o a la página anterior siempre que estén disponibles. (Nótese que en la actualidad el menú no está configurado para procesar los "avance página" y "retroceso página"). Igualmente vuestra implementación de las consultas preparadas debe ser óptima, esto es, definid una consulta preparada para la búsqueda de vuelos y otra para la gestión de tarjetas de embarque y reusadlas. No crees una consulta preparada cada vez que se ejecuta un comando de SQL.

# 4. Material a entregar

Junto con el programa se debe entregar un fichero makefile que permita:

1. Compilar el código (make compile).
2. Borrar, crear y poblar la base de datos (make all).

Igualmente, se solicita que entreguéis el resultado de ejecutar el comando:

```
valgrind ./menu
```

y esperamos que todos los problemas reportados por esta utilidad y que estén relacionados con el código que vosotros habéis escrito se hayan subsanado. Guardad la salida de valgrind en un fichero llamado `valgrind.log` y adjuntar este fichero al material entregado. Nota: las librerías de *ncurses* contienen código que provoca errores del tipo *still reachable: XXXX bytes in YYY block* estos errores los podéis ignorar.

Si bien un programa escrito en C se podría hacer en un único fichero de texto, cualquier proyecto serio requiere que el código fuente de un programa se divida en varios ficheros para que sea manejable. Igualmente, se espera que el tamaño de las funciones creadas sea moderado (en número de líneas), que estas funciones estén comentadas y que incluyan el nombre de su creador.

En esta práctica se realizarán dos entregas, la primera a la mitad de la práctica y la segunda al finalizar la misma. Los requerimientos de cada entrega se describen a continuación:

### 4.1. Resumen del material a entregar al final de la práctica

Subid a *Moodle* el fichero practica2.zip que debería contener:

1. Fichero `Makefile` y todos los ficheros necesarios para crear el programa solicitado. IMPORTANTE: **NO incluyáis el fichero** `flights.sql`. Incluid el fichero `valgrind.log`
2. Memoria en formato pdf que, por cada opción implementada en el programa, incluya una breve explicación de la lógica implementada, así como del comando/s SQL utilizado.

## 5. Criterios de corrección

Para obtener una nota en el rango 5-5.9 es necesario:

- Que el código subido a *Moodle* se pueda compilar con el fichero makefile suministrado en los ordenadores de los laboratorios de prácticas.
- Haber implementado en su totalidad la primera consulta (búsqueda de vuelos). No es necesario implementar la parte que requiere paginación.
- La consulta deben funcionar para cualquier instancia de la base de datos.

Para obtener una nota en el rango 6-6.9 es necesario:

- Satisfacer todos los requerimientos del apartado anterior.
- Haber implementado ambas consultas solicitadas (búsqueda de vuelos y asignación de tarjetas de embarque). No es necesario implementar la parte que requiere paginación.
- Estas consultas deben funcionar para cualquier instancia de la base de datos.

Para obtener una nota en el rango 7-7.9 es necesario:

- Satisfacer todos los requerimientos del apartado anterior.
- La implementación de las consultas debe ser robusta. Por ejemplo, el programa debe responder adecuadamente cuando se le proporcionan identificadores inexistentes o el resultado de la consulta sea el conjunto vacío.
- Presentar una memoria que esté correctamente redactada, demuestre vuestra comprensión de la práctica y se ajuste a lo solicitado en la práctica.
- El código, tanto en C como en SQL, debe ser legible y estar comentado.
- las consultas deben realizarse usando "prepared statements"
- Se reusan los "prepared statements"
- El programa debe estar bien estructurado y estar distribuido en varios ficheros.
- Cuando se ejecuta el código usando valgrind no debe aparecer ningún error atribuible a vuestro código.

Para obtener una nota en el rango 8-10 es necesario:

- Satisfacer todos los requerimientos del apartado anterior.
- El código, tanto en C como en SQL debe estar optimizado. Por ejemplo: (1) no se deben realizar dos conexiones a la base de datos cuando el mismo resultado puede obtenerse con una única conexión y (2) las consultas deben ser resueltas por la base de datos, esto es, no os traigáis todas las entradas de una tabla y luego implementéis la búsqueda en el programa en C.
- Haber implementado la paginación

NOTA: Cada día (o fracción) de retraso en la entrega de la práctica se penalizará con un punto.
NOTA: El código usado en la corrección de la práctica será el entregado en *Moodle*.
NOTA: Si no se puede compilar en la imagen de la partición linux de los ordenadores de los laboratorios con el comando `make` el programa solicitado en la práctica la calificación de la práctica será cero.