

# Proyecto Final Programación Bajo Plataformas abiertas

## IE-0117

Roberto Sánchez Cárdenas B77059  
Jeremy Caldwell Chacón B61330

19 de julio del 2019

## 1. Introducción

El presente proyecto tiene como fin poner a prueba los conocimientos adquiridos durante el semestre en el curso de Programación Bajo Plataformas Abiertas en el lenguaje de programación C. Para este proyecto se espera implementar los temas de entrada y salida de datos, memoria dinámica y el manejo de archivos por medio del lenguaje C. El programa va a ser diseñado desde el sistema operativo Linux y también se hará uso del software de Cmake para simplificar la compilación de cada una de las secciones de código.

Para lograr la implementación de todos los temas mencionados previamente se decidió realizar una simulación de un cajero automático de un banco. El programa debe tener acceso a archivos en los que se almacena la información de cada usuario y otro en el que se encuentra la información de las transacciones. A partir de los archivos mencionados previamente se llevan todos los controles de dinero y usuarios.

## 2. Nota Teórica

Para empezar esta nota teórica se van a definir los siguientes conceptos básicos:

**Puntero:** Un puntero es una variable que contiene la dirección de otra variable. También podríamos decir que un puntero es una variable que representa la posición (más que el valor) de otro dato, tal como una variable o un elemento de un array. Cuando una variable puntero es definida, el nombre de la variable debe ir precedido de un asterisco (\*). Este identifica que la variable es un puntero. Los punteros se pueden comparar entre ellos y se pueden asignar direcciones de memoria. Además, se pueden decrementar o incrementar. El incremento o el decremento varía según el tipo de dato al que apunten. Los punteros son usados con frecuencia en C, y tienen gran cantidad de aplicaciones:

- Proporcionan una forma de devolver varios datos desde una función mediante los argumentos de la función.
- Permiten igualmente, que referencias a otras funciones puedan ser especificadas como argumentos de una función (pasar funciones como argumentos en una función determinada).

**Arreglo:** Existe una estrecha relación entre apuntadores y arreglos, tanto que pueden ser usados en forma casi indistinta. Los arrays son variables estructuradas, donde cada elemento se almacena de forma consecutiva en memoria. Las cadenas de caracteres son declaradas en C como arrays de caracteres y permiten la utilización de un cierto número de notaciones y de funciones especiales. Los arreglos tienen

un tamaño que es la cantidad de objetos del mismo tipo que pueden almacenar. Los arreglos son entidades estáticas debido a que se declaran de un cierto tamaño y conservan este a lo largo de la ejecución del programa en el cual fue declarado.

Una vez vistas las definiciones anteriores es posible continuar con el desarrollo del proyecto el cual se lista como una serie de imágenes las cuales se dan en un orden secuencial de pasos para posteriormente ser analizado.

### 3. Planteamiento del problema

Para emular de la mejor manera a un cajero automático, se deben crear perfiles de usuario en un archivo .csv, en este se debe tener la siguiente información:

- Nombre del usuario
- Número de tarjeta
- Número de pin
- Actividad

Se debe crear una función de ingreso; esta función pedir el número de tarjeta del usuario y su pin, debe ser capaz de recorrer todo el archivo .csv en busca de el número de tarjeta y cuando lo encuentre, sin ambos datos coinciden, le permite seguir al programa. Es importante que esta función realice lo que se desea, debido a que es el unico paso de seguridad que se implementa para usuarios.

Para manejar a cantidad de memoria que se le debe dar a ciertas funciones y cuantas iteraciones se deben realizar en algunos casos, la mejor opción que se encontró fue implementar una función que cuenta las líneas de un archivo. Esta función se va a implementar en gran parte de las funciones, por lo que muchas dependerán de esta. También se planea crear una función que imprime la fecha u hora.

Se necesitan funciones que lean el archivo en el que se llevan todas las transacciones, esto con diferentes fines. Se necesita una que sume todas las transacciones una persona para dar un balance total. Otra que muestre todos los movimientos, otra para todos los movimientos de un usuario. Una vez listas las funciones anteriores, se planea desarrollar una función que genere estados de cuenta en archivos aparte.

Otro tipo de funciones necesarias son de añadir líneas a los archivos de control. Esto con diferentes fines como lo son: realizar movimientos de dinero, depositarle a alguien más y añadir usuarios, como una función para administradores.

Se planea que el cajero tenga una opción de ingreso de administradores, quienes pueden acceder a los archivos completos y añadir usuarios. Para acceder a este 'perfil' se deberá ingresar una contraseña y para cada opción se deberá ingresar el número de tarjeta del administrador y su pin.

### 4. Desarrollo

Como se ha mencionado previamente, el fin de este proyecto es desarrollar un programa en el lenguaje de programación C que implemente diferentes temas, lo cual se logró al finalizar con todas las funcionalidades mencionadas previamente. Para que este programa se utilizaron las siguientes bibliotecas:

- stdio.h
- stdlib.h

- string.h
- time.h

Cada una de las librerías vienen incluidas en el sistema operativo Linux y tienen diferentes finalidades. La biblioteca stdio trabaja lo que son entradas y salidas que se le pasan al programa, stdlib lleva el control de otro tipo de procesos como lo son la memoria dinámica y la aritmética de enteros. String.h trabaja con el manejo de strings. Y la biblioteca time se utilizó con el fin de mostrar la fecha y hora en el programa. Para la separación del programa en varias partes, se creó un archivo de cabecera llamado archivo.h, que contiene las estructuras utilizadas y todos los defines que se crearon.

## 4.1. Entrada y salida

La entrada y salida de datos es un tema muy importante y uno de los más recurrentes, este se utiliza cada vez que se requiere que el usuario ingrese datos a la computadora o bien, cada vez que se espera que la computadora retorne algo. En el programa esto se utiliza repetidamente, un pequeño ejemplo es el siguiente:

```
1 int ingreso_admin(void){
2     char tarjeta_ad[LEN_TARJETA];
3     char pin[LEN_PIN];
4     printf("Ingrese su número de tarjeta -> ");
5     scanf("%s", tarjeta_ad);
6     printf("Ingrese su número de pin -> ");
7     scanf("%s", pin);
8     if(!strcmp(TARJETA_ADMIN, tarjeta_ad) && !strcmp(PIN_ADMIN, pin)){
9         return 1;
10    }
11    else{
12        return 0;
13    }
14 }
```

Se utiliza scanf para pedirle cosas al usuario, que son las entradas y como salida se tienen dos return, dependiendo de lo que suceda con la información brindada por el usuario.

## 4.2. Memoria dinámica

La memoria dinámica es un tema realmente importante que nos permite usar únicamente la memoria, ya sea física o virtual, de la mejor manera, ya que nos permite destinar únicamente lo necesario para ciertos casos. En el programa fue implementado de la siguiente manera:

```
1 int balance(balance_t**tamano, int no_tarjetas){
2     int num_balances = countlines(FBALANCES); //contamos líneas para dar tamaño
3     balance_t balance[num_balances];
4
5     //char buffer_b[len_buffer];
6     *tamano = (balance_t*) malloc(num_balances * sizeof(balance_t)); //Damos tamaño
7     exacto
8
9     FILE *f_p = fopen(FBALANCES, "r");
10    //memset(buffer_b, 0, sizeof(char *len_buffer));
11    printf("Estos son los movimientos de su cuenta: \n");
12    for(int i; i < num_balances ; i++){ //Contamos hasta la cantidad de líneas exacta
```

```

13     int var = fscanf(f_p, "%[^,], %[^,], %f", balance[i].nombre, balance[i].
no_tarjeta, &balance[i].transaccion); //& solo para int y float, ya char es un
puntero
14
15     if(atoi(balance[i].no_tarjeta) == no_tarjeta1){
16         printf("%s -> %.2f", balance[i].nombre, balance[i].transaccion);
17     }
18     //printf("%s, %.2f", balance[i].no_tarjeta, balance[i].transaccion);
19 }
20 printf("\n");
21 rewind(f_p);
22 fclose(f_p);
23 }

```

En el código anterior, se utilizó malloc para darle el tamaño necesario a la estructura.

### 4.3. Archivos

El tema de archivos fue el más fuerte del proyecto, debido a que todo lo requería de la lectura de archivos para correr. Una de las implementaciones que se hicieron fue para leer todo el archivo de balances.

```

1 void estados_todos(void){
2     balance_est_t balance1[NUM_USUARIOS];
3     char monto_final[LEN_BUFFER];
4     char buffer[LEN_BUFFER];
5     FILE *fp_balances = fopen(F_BALANCES, "r");
6     memset(balance1, 0, sizeof(balance_est_t)* NUM_USUARIOS);
7     memset(buffer, 0, sizeof(char)*LEN_BUFFER);
8
9     //Manejo de errores
10    if((fp_balances) == NULL){
11        printf("Hubo un error con los archivos, contacte con el centro de atención.");
12        exit(0);
13    }
14
15    //Lo que se va a imprimir
16    while (!balance1[0].encontrado && fgets(buffer, LEN_BUFFER, fp_balances) != NULL){
17        char *p_nombre_b = strtok(buffer, ",");
18        char *p_no_tarjeta_b = strtok(NULL, ",");
19        char *p_transaccion_b = strtok(NULL, ",");
20
21        strcpy(balance1->nombre, p_nombre_b);
22        strcpy(balance1->no_tarjeta, p_no_tarjeta_b);
23        strcpy(balance1->transaccion, p_transaccion_b);
24        printf("%s, %s, %s", balance1->nombre, balance1->no_tarjeta, balance1->
transaccion);
25    }
26    fseek(fp_balances, 0, SEEK_SET);
27    fclose(fp_balances);
28 }

```

## 5. ¿Cómo utilizar el programa?

El código desarrollado se configuró para ser compilado con el software Cmake, para ello se creó un CMakeLists.txt, que viene adjunto con el programa. Es necesario que en la carpeta donde se tiene todo,

estén los archivos: info.csv y balance.csv, de otro modo, el programa no funciona de nada.

En una terminal de Linux se debe ir a la dirección en la que se encuentra el programa y una vez en esta se deben ingresar los siguientes comandos:

- `cmake .`
- `cmake --build .`

Para correr el programa se debe escribir:

- `./cajero_automático`



```
roberto@robertosc: ~/proyecto_plataformas/proyecto_plataformas/entregable
(base) roberto@robertosc:~/proyecto_plataformas/proyecto_plataformas/entregable$ cmake .
-- Configuring done
-- Generating done
-- Build files have been written to: /home/roberto/proyecto_plataformas/proyecto_plataformas/entregable
(base) roberto@robertosc:~/proyecto_plataformas/proyecto_plataformas/entregable$ cmake --build .
(base) roberto@robertosc:~/proyecto_plataformas/proyecto_plataformas/entregable$ ./cajero_automático

Desea ingresar como:
1. Cliente
2. Administrador
3. Salir
Opción -> 
```

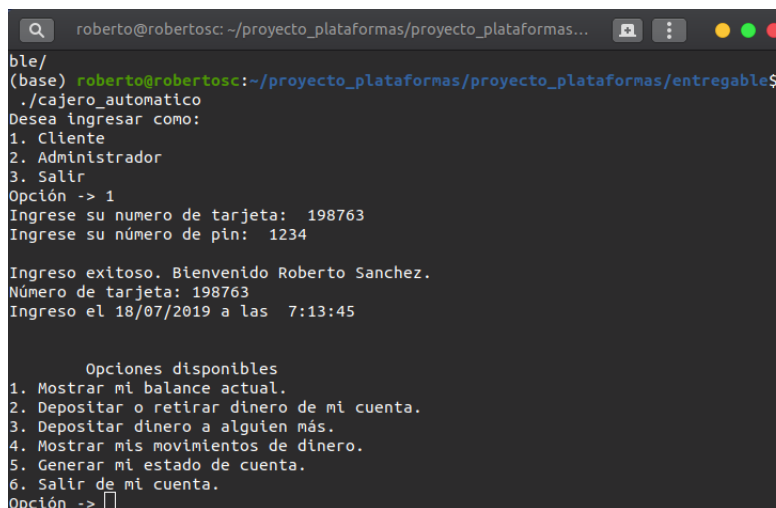
Figura 1: Captura de como se compila y como se ve las opciones iniciales

Una vez en la pantalla anterior, se elige una opción ingresando el número de opción como entrada.

## 5.1. Ingreso como cliente

Para probar esta opción se recomienda ingresar con los datos que se muestran a continuación, debido a que esta cuenta posee la mayor cantidad de información, sin embargo se puede con cualquiera de las disponibles en el archivo info.csv.

- Número de tarjeta: 198763
- Pin: 1234



```
roberto@robertosc: ~/proyecto_plataformas/proyecto_plataformas/entregable/
(base) roberto@robertosc:~/proyecto_plataformas/proyecto_plataformas/entregable$ ./cajero_automático
Desea ingresar como:
1. Cliente
2. Administrador
3. Salir
Opción -> 1
Ingrese su numero de tarjeta: 198763
Ingrese su número de pin: 1234

Ingreso exitoso. Bienvenido Roberto Sanchez.
Número de tarjeta: 198763
Ingreso el 18/07/2019 a las 7:13:45

Opciones disponibles
1. Mostrar mi balance actual.
2. Depositar o retirar dinero de mi cuenta.
3. Depositar dinero a alguien más.
4. Mostrar mis movimientos de dinero.
5. Generar mi estado de cuenta.
6. Salir de mi cuenta.
Opción -> 
```

Figura 2: Opciones que se muestran al ingresar como cliente.

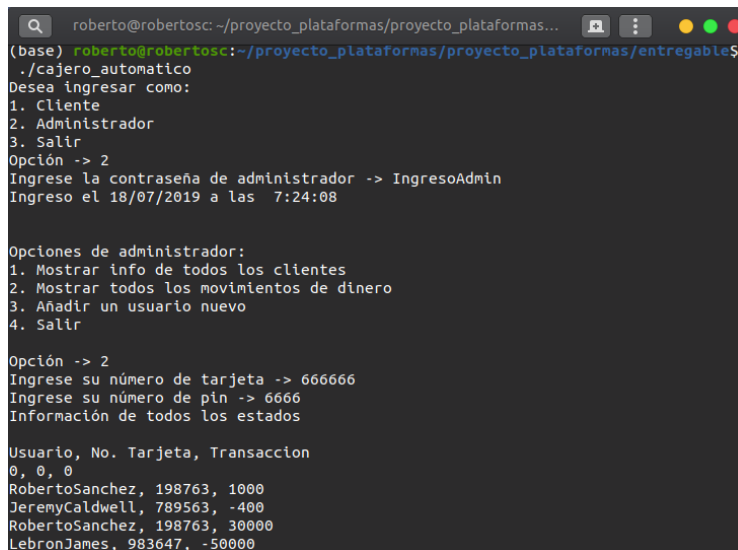
En la Figura 2 se muestran las opciones disponibles en el cajero. Para usar cada una se ingresa el número de opción como entrada.

## 5.2. Ingreso como administrador

El ingreso como administrador tiene más pasos para simular mayor seguridad. Cuando se selecciona la opción 2 en la pantalla que se muestra en la figura 1, se pide una contraseña. Y cuando se selecciona una opción se piden más datos de seguridad. Toda las contraseñas de administrador se listan a continuación:

- Contraseña de ingreso admin: IngresoAdmin
- Número de tarjeta admin: 666666
- PIN: 6666

La pantalla de administrador tiene opciones diferentes a las de un cliente, estas se muestran a continuación junto con como se ve el ingreso a este tipo de cuenta:



```
roberto@robertosc: ~/proyecto_plataformas/proyecto_plataformas...
(base) roberto@robertosc:~/proyecto_plataformas/proyecto_plataformas/entregables$ ./cajero_automatico
Desea ingresar como:
1. Cliente
2. Administrador
3. Salir
Opción -> 2
Ingrese la contraseña de administrador -> IngresoAdmin
Ingreso el 18/07/2019 a las 7:24:08

Opciones de administrador:
1. Mostrar info de todos los clientes
2. Mostrar todos los movimientos de dinero
3. Añadir un usuario nuevo
4. Salir
Opción -> 2
Ingrese su número de tarjeta -> 666666
Ingrese su número de pin -> 6666
Información de todos los estados

Usuario, No. Tarjeta, Transaccion
0, 0, 0
RobertoSanchez, 198763, 1000
JeremyCaldwell, 789563, -400
RobertoSanchez, 198763, 30000
LebronJames, 983647, -50000
```

Figura 3: Opciones e ingreso a la cuenta de administrador

## 6. Conclusiones

El desarrollo de este proyecto fue muy enriquecedor a nivel intelectual ya que permitió llegar a tener un mejor manejo del lenguaje de programación C. Por la extensión del mismo se logró abarcar todos los temas requeridos. El programa diseñado tiene como objetivo único emular un cajero automático, sin embargo, por la variedad de funciones que se crearon y la variedad de las mismas en el manejo de listas en formato .csv, este programa podría ser modificado para otros propósitos. Un ejemplo de programa que se podría derivar de este es el manejo de inventario en un establecimiento.

A pesar de que el lenguaje C no es el mejor para el tipo de programa que se deseaba crear, se llegó a ver que tampoco es extremadamente complicado el uso del mismo y que existen diferentes opciones para resolver un solo problema. En este programa se tuvo que utilizar más de una manera de solucionar cierto problema debido a que no siempre se lograba lo deseado con cierta solución que realizaba algo similar.

El lenguaje de programación C tiene su fuerte en el manejo de memoria, lo cual lo hace irremplazable hasta el día de hoy. Este tema fue muy importante a la hora de desarrollar el código, esto debido a que se requirió de mucho cuidado a la hora de definir la memoria que se dispone para cada función.

## Referencias

- [1] Tutorials Point, *Pointers in C*. Recuperado de [https://www.tutorialspoint.com/cprogramming/c\\_pointers#](https://www.tutorialspoint.com/cprogramming/c_pointers#)
- [2] Choudhary, V., *Array - C Programming*. Developer Insider. Recuperado de <https://developerinsider.co/array-c-programming/>
- [3] cplusplus.com, *cstdlib (stdlib.h)*. Recuperado de <http://www.cplusplus.com/reference/cstdlib/>