
# TP2

# Stack Frame

## Debianitas

Carreras: Ingeniería en computación , Ingeniería Electrónica

Profesor:- Javier Jorge
- Solinas Miguel Angel

Alumnos:
- Landaeta Ezequiel Matias, Legajo: 41092007
- Agüero David Gabriel, Legajo: 39621362
- Cherino Macarena, Legajo: 41809793

# Índice

# 1 Consigna

Para aprobar el TP#2 se debe diseñar e implementar una Calculadora de Cotización de Criptomonedas. La capa superior recuperará la cotización de al menos dos criptomonedas que pueden ser obtenidas de alguna de "Las 12 mejores API del mercado de valores para crear productos financieros". Se recomienda el uso de API Rest y python. Los datos de consulta realizados deben ser entregados a un programa en C que convocará rutinas en ensamblador para que hagan los cálculos de conversión y devuelvan las cotizaciones en pesos, u$s y euros. Luego el programa en C o python mostrará los cálculos obtenidos.

Se debe utilizar el stack para convocar, enviar parámetros y devolver resultados. O sea utilizar las convenciones de llamadas de lenguajes de alto nivel a bajo nivel.

# 2 Desarrollo

## 2.1 Código

En el github https://github.com/TheRabbitt/Sistemas-de-Computacion se encuentra el código de un programa escrito en lenguaje C que obtiene el precio de dos criptomonedas (Bitcoin y Ethereum) usando la API de Binance y luego imprime el precio de estas criptomonedas en diferentes monedas (ARS, USD y EUR).

El programa utiliza la biblioteca libcurl para realizar solicitudes HTTP a la API de Binance y obtener los precios de las criptomonedas. También utiliza la función "write_data" como una función de retorno de llamada para procesar los datos recibidos de la API.

Se define una variable llamada BTC y se le da el valor 0. Por otro lado, se define una variable llamada ETH y se le da el valor 1. Estas dos variables tienen el propósito de facilitar el manejo de las diferentes monedas. Luego se definen dos matrices de caracteres btc_price y eth_price, que se utilizarán para almacenar los precios de Bitcoin y Ethereum, respectivamente. current_currency es una variable entera que se utilizará para indicar qué moneda se está procesando en un momento dado. Por último, se definen las variables valorArsDolar y valorEurDolar, que se utilizarán como factores de conversión para convertir los precios de Bitcoin y Ethereum en diferentes monedas.

A continuación, se declara la función calc_conversion, la cual se implementa en un archivo de ensamblador externo y se utilizará para realizar la conversión de precios de USD a otras monedas.

Después de la declaración de la función externa, se define la función write_data(). Esta función, es utilizada para procesar los datos recibidos de la API en la solicitud HTTP GET. Esta

función se llamará automáticamente por la biblioteca libcurl cada vez que reciba datos de la API.

La función write_data() toma cuatro argumentos:

- ptr: un puntero al bloque de memoria que contiene los datos recibidos.
- size: el tamaño en bytes de cada elemento recibido.
- nmemb: el número de elementos recibidos.
- stream: un puntero al objeto que llama a la función.

La función devuelve el número total de bytes recibidos. Si la moneda actual es BTC, los datos recibidos se copiarán en btc_price; si es ETH, los datos se copiarán en eth_price.

A continuación, comienza la función main().

En esta función, se inicializan las variables, se configuran las opciones de CURL, se realiza la solicitud a la API, se procesa la respuesta y se imprimen los resultados.

Aquí está la explicación detallada de lo que hace cada línea de la función main():

1. Se declara un puntero de tipo CURL, que se usará para realizar las solicitudes HTTP a la API.
2. Se declara una variable de tipo CURLcode, que se utiliza para verificar si la solicitud se realizó con éxito.
3. Se inicializa el puntero CURL con la función curl_easy_init().
4. Se verifica si el puntero CURL se inicializó correctamente.
5. Se establece la solicitud HTTP GET para obtener el precio de BTC con la función curl_easy_setopt(). Esto se realiza configurando las opciones CURLOPT_URL y CURLOPT_WRITEFUNCTION. La opción CURLOPT_URL especifica la URL de la API que proporciona el precio de BTC. La opción CURLOPT_WRITEFUNCTION establece la función de retorno de llamada que procesará los datos recibidos de la API.
6. Se realiza la solicitud HTTP GET para obtener el precio de BTC con la función curl_easy_perform(). Si la solicitud se realizó con éxito, se copian los datos recibidos en "btc_price" usando la función de retorno de llamada write_data().
7. Se establece la solicitud HTTP GET para obtener el precio de ETH con la función curl_easy_setopt(). Esto se realiza configurando las opciones CURLOPT_URL y CURLOPT_WRITEFUNCTION. La opción CURLOPT_URL especifica la URL de la API que proporciona el precio de ETH. La opción CURLOPT_WRITEFUNCTION establece la función de retorno de llamada que procesará los datos recibidos de la API.
8. Se realiza la solicitud HTTP GET para obtener el precio de ETH con la función curl_easy_perform(). Si la solicitud se realizó con éxito, se copian los datos recibidos en "eth_price" usando la función de retorno de llamada write_data().
9. Se imprime el precio de BTC en diferentes monedas usando la función obtenerCotizacion(). La función obtiene la cotización actual del BTC en pesos

argentinos y en euros, y utiliza la función calc_conversion() (externa en archivo .asm) para convertir el precio de BTC a las diferentes monedas.

10. Se imprime el precio de ETH en diferentes monedas usando la función obtenerCotizacion(). La función obtiene la cotización actual del ETC en pesos argentinos y en euros, y utiliza la función calc_conversion() (externa en archivo .asm) para convertir el precio de ETH a las diferentes monedas.
11. Se libera la memoria y los recursos utilizados por CURL con la función curl_easy_cleanup().

La función main() devuelve 0, lo que indica que el programa finalizó correctamente.

La función "obtenerCotizacion" se utiliza para convertir el precio de una criptomoneda en diferentes monedas. Esta función toma dos parámetros: moneda, que es la cotización actual de la moneda de referencia en relación a una tercera moneda (USD en este caso), y monedaDigital, que es un arreglo de caracteres que contiene el precio de una criptomoneda en relación a USD en formato de cadena.

La función comienza buscando la letra 'e' en monedaDigital utilizando la función strchr(). Si se encuentra la letra 'e', ya que en el formato que retorna la API hay una ‘e’ antes de que comience el valor de la cotización de la moneda digital respectiva. Si no se encuentra la letra 'e', la función termina con un error.

Luego, la función avanza cuatro caracteres más allá de la letra 'e' y convierte el resto de la cadena en un número de punto flotante utilizando la función strtod(). Este número representa el precio de la criptomoneda en USD.

La función luego llama a la función calc_conversion() que se define en otro archivo .asm y toma como parámetros el valor del precio de la criptomoneda en USD y la cotización actual de la moneda de referencia en relación a USD. La función calc_conversion() devuelve el valor de la criptomoneda en la moneda de referencia.

## 2.2 Comandos para compilar

**1-** nasm -f elf64 -d ELF_TYPE -o Conversor5.o -d NO_STACK_PROTECTOR Conversor5.asm
**2-** gcc -o simpleget simplev3.c Conversor5.o -lcurl -z noexecstack

El primer comando empleado (1), utiliza el ensamblador NASM para compilar un archivo de código fuente llamado "Conversor5.asm" en un archivo objeto de 64 bits con formato ELF llamado "Conversor5.o", con la adición de dos macros definidas en tiempo de compilación: "ELF_TYPE" y "NO_STACK_PROTECTOR".

**-f elf64:** establece el formato de salida a ELF de 64 bits, que es un formato de archivo binario comúnmente utilizado en sistemas operativos basados en Unix/Linux.

**-d ELF_TYPE:** define una macro llamada "ELF_TYPE" con un valor vacío. Esta macro podría usarse en el código fuente de "Conversor5.asm" para realizar operaciones condicionales o para diferenciar entre diferentes tipos de ELF.

**-o Conversor5.o:** establece el nombre de salida del archivo objeto a "Conversor5.o".

**-d NO_STACK_PROTECTOR:** establece una macro de preprocesador llamada NO_STACK_PROTECTOR con un valor vacío durante la compilación del código fuente NASM. Esta macro puede ser utilizada en el código fuente para desactivar la protección de pila en el programa final compilado.

La protección de pila es una técnica de seguridad utilizada en la programación para detectar y prevenir los desbordamientos de pila. Los desbordamientos de pila ocurren cuando un programa escribe datos en una ubicación de la pila que está más allá del final de la región de memoria reservada para la pila, lo que puede llevar a comportamientos inesperados o vulnerabilidades de seguridad.

Sin embargo, en algunos casos, la protección de pila puede interferir con el funcionamiento de un programa, especialmente si se está escribiendo código de bajo nivel o se está interactuando con hardware. Por lo tanto, la opción -d NO_STACK_PROTECTOR permite desactivar la protección de pila en el programa final compilado si se considera que es necesario en un contexto particular. Es importante tener en cuenta que desactivar la protección de pila puede aumentar el riesgo de explotación de vulnerabilidades de seguridad en el programa, por lo que esta opción debe usarse con precaución y solo en casos donde se justifique.

El segundo comando empleado (2), utiliza el compilador GCC para compilar y vincular dos archivos de código fuente: "simplev3.c" y "Conversor5.o", generando un archivo ejecutable llamado "simpleget". Además, la línea de comando especifica que el archivo ejecutable no debe tener una pila ejecutable, y que debe vincularse con la biblioteca estática "curl" durante la compilación.

**-o simpleget:** establece el nombre de salida del archivo ejecutable a "simpleget".

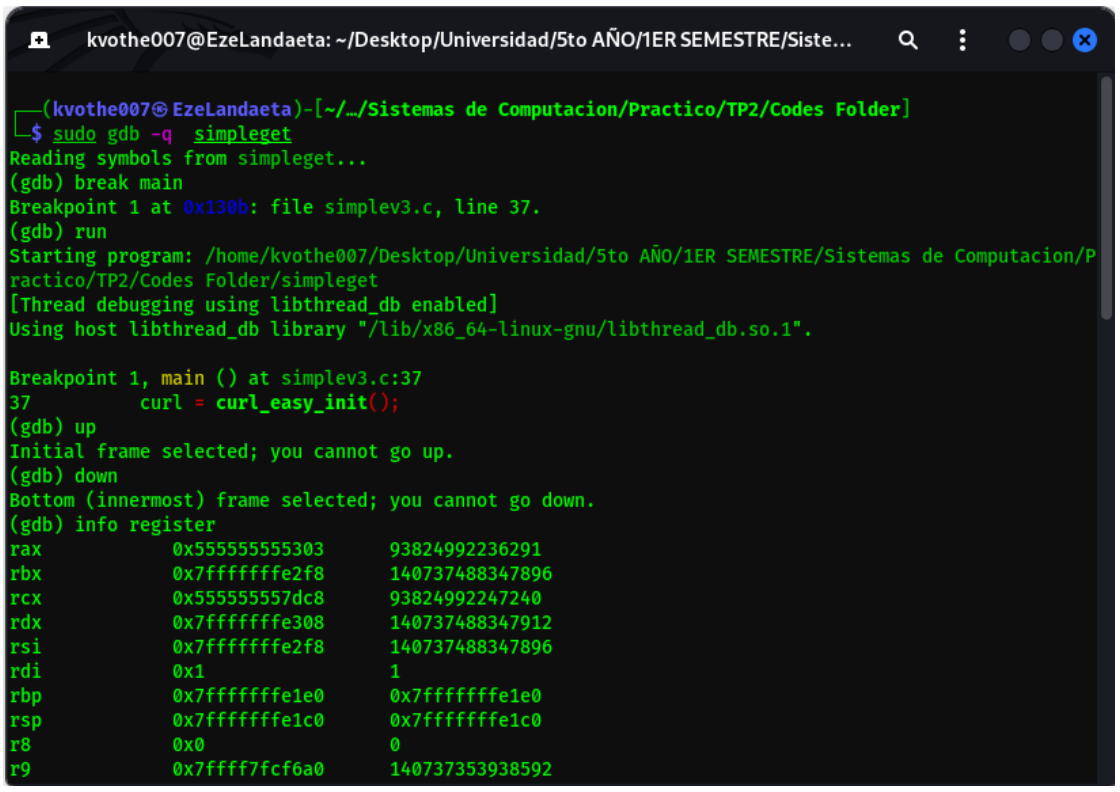
**simplev3.c:** es el nombre del archivo de código fuente de C que se compilará y vinculará con el archivo objeto "Conversor5.o" para crear el archivo ejecutable "simpleget".

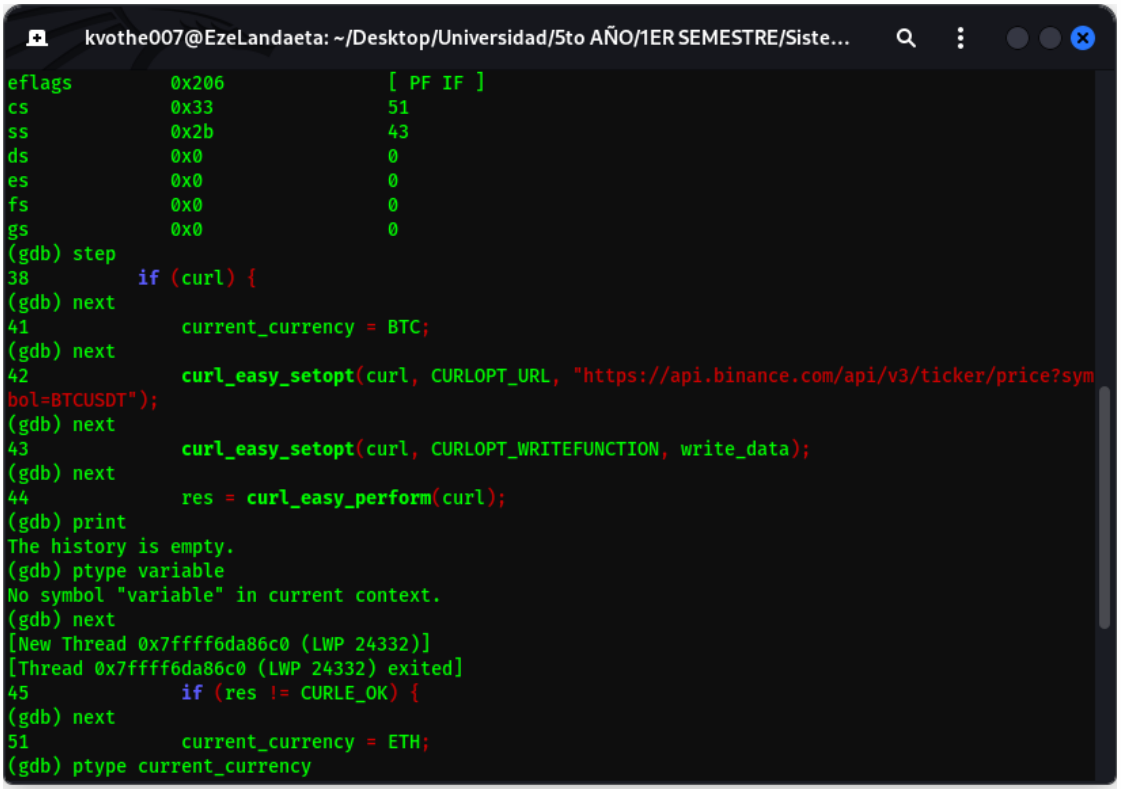
**Conversor5.o:** es el nombre del archivo objeto compilado previamente que se vinculará con el archivo de código fuente "simplev3.c" para crear el archivo ejecutable "simpleget".

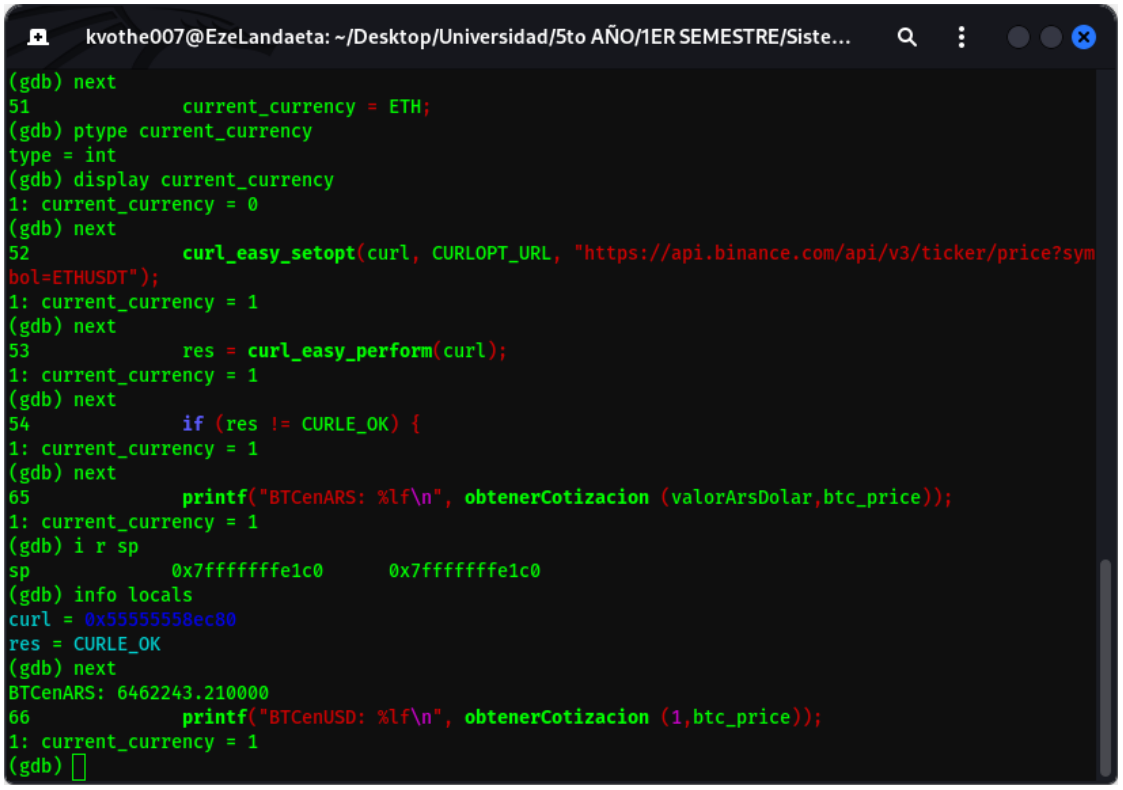
**-lcurl:** especifica que la biblioteca estática "curl" debe vincularse con el archivo ejecutable durante la compilación. Esto significa que el archivo ejecutable tendrá acceso a las funciones y variables definidas en la biblioteca estática "curl" durante su ejecución.

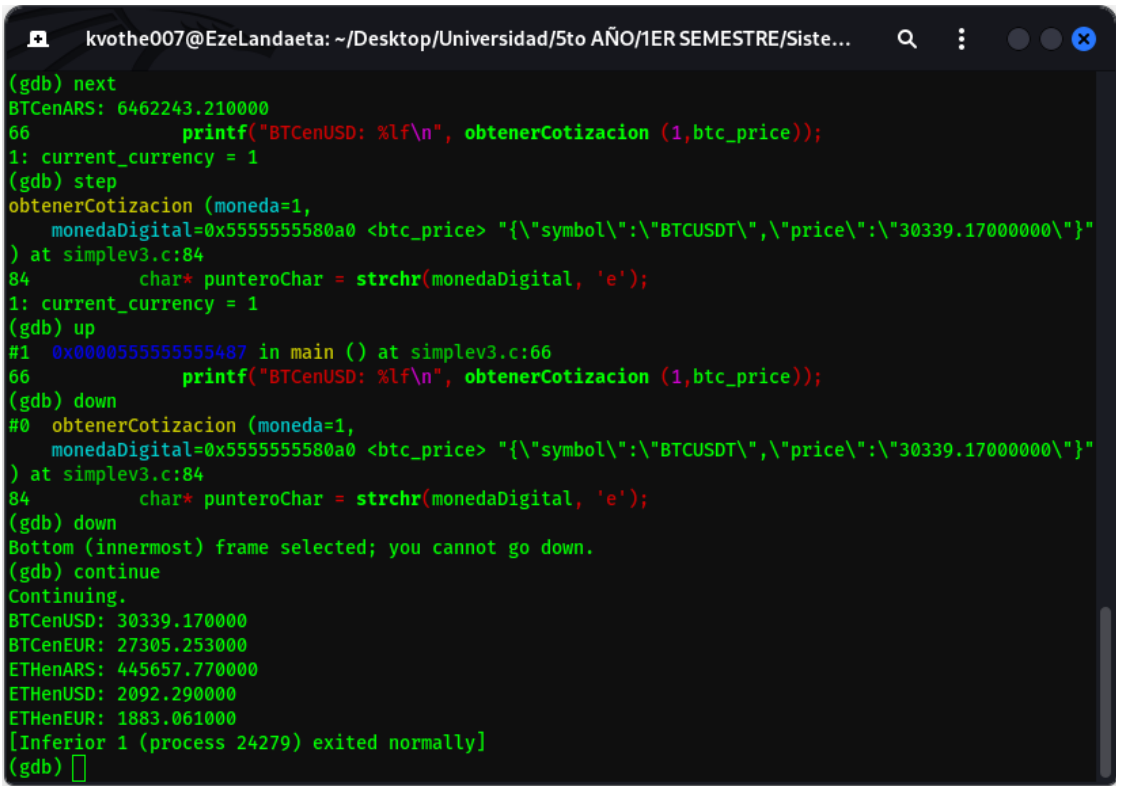
**-z noexecstack:** establece la pila del programa para que no sea ejecutable. Esto significa que se evita que la pila del programa se use para ejecutar código, lo que puede ayudar a prevenir ciertos tipos de ataques de seguridad.

## 2.3 Depuración con gdb




kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Siste...
(gdb) next
51 current_currency = ETH;
(gdb) ptype current_currency
type = int
(gdb) display current_currency
1: current_currency = 0
(gdb) next
52 curl_easy_setopt(curl, CURLOPT_URL, "https://api.binance.com/api/v3/ticker/price?sym
bol=ETHUSDT");
1: current_currency = 1
(gdb) next
53 res = curl_easy_perform(curl);
1: current_currency = 1
(gdb) next
54 if (res != CURLE_OK) {
1: current_currency = 1
(gdb) next
65 printf("BTCenARS: %lf\n", obtenerCotizacion (valorArsDolar,btc_price));
1: current_currency = 1
(gdb) i r sp
sp 0x7fffffffe1c0 0x7fffffffe1c0
(gdb) info locals
curl = 0x55555558ec80
res = CURLE_OK
(gdb) next
BTCenARS: 6462243.210000
66 printf("BTCenUSD: %lf\n", obtenerCotizacion (1,btc_price));
1: current_currency = 1
(gdb)


kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Siste...
(gdb) next
BTCenARS: 6462243.210000
66 printf("BTCenUSD: %lf\n", obtenerCotizacion (1,btc_price));
1: current_currency = 1
(gdb) step
obtenerCotizacion (moneda=1,
monedaDigital=0x5555555580a0 <btc_price> "{\"symbol\":\"BTCUSDT\",\"price\":\"30339.17000000\"}"
) at simplev3.c:84
84 char* punteroChar = strchr(monedaDigital, 'e');
1: current_currency = 1
(gdb) up
#1 0x0000555555555487 in main () at simplev3.c:66
66 printf("BTCenUSD: %lf\n", obtenerCotizacion (1,btc_price));
(gdb) down
#0 obtenerCotizacion (moneda=1,
monedaDigital=0x5555555580a0 <btc_price> "{\"symbol\":\"BTCUSDT\",\"price\":\"30339.17000000\"}"
) at simplev3.c:84
84 char* punteroChar = strchr(monedaDigital, 'e');
(gdb) down
Bottom (innermost) frame selected; you cannot go down.
(gdb) continue
Continuing.
BTCenUSD: 30339.170000
BTCenEUR: 27305.253000
ETHenARS: 445657.770000
ETHenUSD: 2092.290000
ETHenEUR: 1883.061000
[Inferior 1 (process 24279) exited normally]
(gdb)

# 3 Bibliografía

**Documentación Api Rest:**

https://binance-docs.github.io/apidocs/spot/en/#general-api-information

**Ejemplos de Paul Carter:**

http://pacman128.github.io/pcasm/

**Link del repo:**

https://github.com/TheRabbitt/Sistemas-de-Computacion