Trabajo Práctico 0: Infraestructura básica

Leandro Huemul Desuque, *Padrón Nro. 95836*
desuqueleandro@gmail.com
Cristian Nicolas Gonzalez, *Padrón Nro. 94719*
cristian3629@gmail.com

1er. Cuatrimestre de 2017
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

Abstract

Se desarrolló un programa en C que codifica y decodifica información en formato base 64. El objetivo del presente trabajo fue familiarizarse con las herramientas básicas, tales como GXEmul y LaTeX.

1 Introducción

Se implementó en C un codificador/decodificador de información en base 64. Base 64 es un sistema de numeración posicional que usa 64 como base. Es la mayor potencia de dos que puede ser representada usando únicamente los caracteres imprimibles de ASCII. Esto ha propiciado su uso para codificación de correos electrónicos, PGP y otras aplicaciones. Como puede entenderse, se trata de un sistema simple por lo que resulta una buena elección si el objetivo final no es construir un sistema elaborado sino familiarizarse con el software necesario para ello.

2 Desarrollo

2.1 Documentación del código C

La documentacion de las funciones se detalla por orden de aparición en el código fuente.

2.1.1 `help`

`help` despliega la ayuda para el usuario final.

2.1.2 `version`

`version` informa la version del código fuente.

2.1.3 `CharToBinary`

`CharToBinary` convierte cada caracter que ingresa a una cadena de bits.

2.1.4 `encodeBase64`

`encodeBase64` se encarga de codificar a base 64 utilizando la posicion que se le envía y comparándola con una tabla de códigos.

2.1.5 `BinaryToDecimal`

`BinaryToDecimal` convierte la cadena parcial de bits a la posicion decimal de mi tabla de codigos base64.

2.1.6 `fileOpen`

`fileOpen` se encarga de abrir archivos.

2.1.7 `fileClose`

`fileClose` cierra el archivo abierto.

2.1.8 `fileGetSize`

`fileGetSize` calcula el tamaño del archivo ingresado.

2.1.9 `fileRead`

`fileRead` se encarga de parsear el archivo y cargar los datos a un buffer.

2.1.10 `fileProcessing`

`fileProcessing` se encarga de procesar el archivo de texto, haciendo el llamado para la apertura, lectura y eventual cierre del fichero.

2.1.11 `bufferOpen`

`bufferOpen` es una función que reserva espacio en memoria de un tamaño arbitrario.

2.1.12 `bufferClose`

`bufferClose` es una función que libera el espacio en memoria pedido por el usuario.

2.1.13 `PosicionToBinary`

`PosicionToBinary` se encarga de convertir una posicion de la tabla de códigos base 64 a su correspondiente cadena binaria.

2.1.14 `decodeBase64`

`decodeBase64` devuelve la posicion correspondiente a la tabla de códigos según el caractere ingresado.

2.1.15 `decode`

`decode` se encarga de traducir una cadena en base 64.

2.1.16 `grabarArchivo`

`grabarArchivo` guarda la información procesada en un archivo de salida.

2.1.17 `leerArchivo`

`leerArchivo` obtiene el tamaño del archivo, se encarga de abrir el buffer y de procesarlo.

2.1.18 `readSTDIN`

`readSTDIN` lee la cadena de entrada estandar STDIN.

2.1.19 `charCopy`

`charCopy` copia cadenas de caracteres.

2.1.20 `calculateLen`

`calculateLen` calcula la longitud de una cadena de caracteres que previamente venía en formato de bits.

2.1.21 `encode`

`encode` se encarga de convertir cualquier caracter a binario, para luego convertirlo en la posición de codificación en la tabla de códigos base 64.

2.2 Especificaciones

El funcionamiento del algoritmo de encoding/decoding es el siguiente:

* Para codificar: Se lee un caracter, el cual se convierte a su representación binaria. La misma se divide en cadenas de 6 bits, los cuales se transforman en su representación numérica. Esta última representación se compara con una tabla de códigos que no es más que todas las posibles secuencias de caracteres que puede tomar un texto en base 64.

* Para decodificar: El camino es análogo a la codificación, se lee un caracterer codificado, el cual se transforma a su representación binaria. Éste y los demás caracteres se unen en una nueva cadena binaria, la cual se divide en subcadenas de 8 bits. Estas subcadenas se convierten a representación ASCI obteniendo el texto decodificado.

2.3 Dificultades

Se podría remarcar el tiempo de aprendizaje de las nuevas herramientas, aunque no se encontraron grandes dificultades en el desarrollo del trabajo.

3 Compilación

Los argumentos utilizados para la compilación son los siguientes:

-c Compila el código fuente pero no corre el linker. Genera el código objeto.

-o Especifica el archivo de salida (ya sea un archivo objeto, ejecutable, ensamblado).

-Wall Activa los mensajes de warning.

-I Agrega el directorio especificado a la lista de directorios buscados para los archivos header

Los argumentos para la generación del código assembly con los siguientes:

-Wall Activa los mensajes de warning.

-std=c99 Se selecciona el estándar C99.

-O0 No se aplica optimizaciones por parte del compilador

-S detiene al compilador luego de generar el assembly.

-mrnames indica al compilador que genere la salida utilizando nombre de registro en lugar de número de registro

4 Resultados

4.1 Código Assembly

Se adjunta una porción del código MIPS al final del informe.

4.2 Pruebas

A continuación se muestran las pruebas realizadas con sus respectivas salidas por consola y/o archivos.

```
   ./tp0
Man
TWFu

   ./tp0
M
TQ==

   ./tp0
Ma
TWE=

   ./tp0
any carnal pleasure.
YW55IGNhcm5hbCBwbGVhc3VyZS4=
```

```
   ./tp0
any carnal pleasure
YW55IGNhcm5hbCBwbGVhc3VyZQ==

   ./tp0
any carnal pleasur
YW55IGNhcm5hbCBwbGVhc3Vy

   ./tp0
any carnal pleasu
YW55IGNhcm5hbCBwbGVhc3U=

   ./tp0
any carnal pleas
YW55IGNhcm5hbCBwbGVhcw==

   ./tp0 -i emptyFile.txt


   ./tp0 -i example1-encode.txt
TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ
1dCBieSB0aGlzIHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLC
B3aGljaCBpcyBhIGx1c3Qgb2YgdGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlc
mFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGludWVkIGFuZCBpbmRlZmF0aWdh
YmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRoZSBzaG9ydCB
2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=
```

5 Conclusiones

Luego de ejecutar ambas versiones, tanto la realizada con compilación normal como la realizada sobre el emulador, podemos concluir que se han comprendido los usos generales de las herramientas propuestas por la materia y se han afianzado las técnicas de programación necesarias para resolver este tipo de problemas.

References

[1] Sitio web de GXemul http://gxemul.sourceforge.net/

[2] Sitio web de GNU https://www.gnu.org/

[3] Base 64 https://en.wikipedia.org/wiki/Base64