Trabajo Práctico 0: Infraestructura básica

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

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

Resumen

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. `setStdinBuffer`

`setStdinBuffer` reserva espacio en memoria para almacenar la información ingresada por entrada estandar (STDIN).

2.1.18. `TODO`

`TODO` es una funció

2.1.19. `TODO`

`TODO` es una función

2.1.20. `TODO`

`TODO` es una funció

2.1.21. `Especificaciones`

FALTA, TODO.

2.2. Dificultades

FALTA, TODO.

3. Compilación

FALTA, TODO.
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

4. Resultados

FALTA, TODO.

5. Conclusiones

FALTA, TODO.

Referencias

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