Informe TP1

Cotarelo Rodrigo, *Padrón Nro. XXXXX*
cotarelorodrigo@gmail.com

Echegaray Rodrigo, *Padrón Nro. XXXXX*
XXXXXXX

Longo Nicolás, *Padrón Nro. 98271*
longo.gnr@hotmail.com

1er. Cuatrimestre de 2019
66.20 Organización de Computadoras – Práctica Martes s
Facultad de Ingeniería, Universidad de Buenos Aires

poner la fecha correspondiente

1. Enunciado

1. Objetivos
   Familiarizarse con las herramientas de software que usaremos en los siguien- tes trabajos, implementando un programa (y su correspondiente documenta- ción) que resuelva el problema piloto que presentaremos más abajo.

2. Alcance
   Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos
   El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes. Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 7), la presentación de los resultados obtenidos, explicando, cuando co- rresponda, con fundamentos reales, las razones de cada resultado obtenido.

4. Recursos
   Usaremos el programa GXemul [1] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD [2]. En la clase del 12/3 hemos repasado, brevemente, los pasos necesarios para la instalación y configuración del entorno de desarrollo.

5. Programa
   Se trata de escribir en lenguaje C dos programas que permitan convertir archivos de texto desde y hacia plataformas UNIX como las que usamos en los trabajos prácticos.

   En particular, buscamos generar dos programas: unix2dos para transformar archivs de texto de UNIX a Windows y dos2unix para hacer la operación inversa.

   Por defecto, en ambos casos, tomaremos la entrada estándar stdin y salida por stdout. Las opciones -i y -o nos permiten además indicar explícitamente los archivos de entrada y salida usando - para indicar los flujos estándar que ya mencionmos.

   A continuación se presenta un ejemplo de codificación:

   $ (echo Uno; echo Dos; echo Tres) |unix2dos |od -t c

   0000000 U n o \r \n D o s \r \n T r e s \r \n

   0000020

Podemos ver, entonces, que por defecto unix2dos opera usando los streams estandar de entrada/salida, generando los newlines con la convencion adoptada por Windows. Notar ademas el uso del programa od, que nos permite ver caracteres que de otra forma no podrían ser observados.

A continuación, usamos dos2unix para hacer una conversión en el sentido inverso:

$ od -t c /tmp/dos.txt

0000000 U n o \r \n D o s \r \n T r e s \r \n

0000020

$ dos2unix -i /tmp/dos.txt -o - |od -t c

0000000 U n o \n D o s \n T r e s \n

0000015

Aqui /tmp/dos.txt contiene un archivo de texto usando la convención CR+LF para los caracteres newline, y hemos usado el programa dos2unix para transformarlo a la convensión LF adoptada en sistemas UNIX.

6. Casos de prueba
   Se deberá verificar el correcto comportamiento del programa, por lo que los alumnos deberán proponer casos de prueba que crean convenientes, indicando el motivo de la elección de cada caso, indicando el método utilizado para verificar que el programa responde correctamente en cada caso.

7. Portabilidad
   Como es usual, es necesario que la implementación desarrollada provea un grado minímo de portabilidad. Para satisfacer esto, el programa deberá funcio- nar al menos en NetBSD/pmax (usando el simulador GXemul [1]) y la versión de Linux usada para correr el simulador.

8. Informe
   El informe deberá incluir:

   - Documentación relevante al diseño, implementación, validación y utiliza- ción del programa.
   - La documentación necesaria para generar los binarios a partir del código fuente suministrado.
   - Las corridas de prueba, con los comentarios pertinentes.
   - El código fuente, el cual también deberá entregarse en formato digital compilable (incluyendo archivos de entrada y salida de pruebas).

- Este enunciado.

9. Fechas Fecha de vencimiento: martes 16/4 de 2019.

2. Implementacion

Se implementaron dos funciones para poder convertir un archivo de DOS a UNIX y viceversa. La implementación se basó en un ciclo while que recorre el archivo caracter por caracter mientras consulta si está parado sobre un '\r' o '\n'. En caso de ser alguno de estos caracteres, se efectua la modificación correspondiente en el archivo resultado o salida y sino, simplemente se escribía el mismo caracter en el documento de salida.

3. Errores que surgieron

Una de las complicaciones que surgieron durante la implementación fue la aparación de un caracter de más en la conversión del archivo. Esto sucedía porque nuestra implementación verificaba si estábamos en el End Of File (EOF) como condición de corte del while para una vez dentro del mismo efectuar la lectura de un nuevo caracter. Entonces pasaba que al leer el último caracter, c sigue apuntando a este, entonces el while corria una vez mas debido a que no estabamos en el EOF y se producia una lectura mas lo que implicaba leer un EOF y escribir un caracter vacio en el archivo. Esto se soluciono moviendo la lectura del caracter adentro de la condicion del while, lo que implico que se lea el EOF y luego realize la verificacion el while. !!!!!Esto no se entiende bien. Pero no supe parafrasearlo!!!!

4. Validacion

Las primeras pruebas básicas que se hicieron, fueron convertir archivos de texto de un formato a otro chequeando que el archivo de salida tuviese el mismo aspecto que el archivo de entrada. Luego, se hicieron comparaciones entre los archivos de entrada y salida usando el programa OD, para tener una vista más detallada de los documentos y sus caracteres. Por ultimo se generaron algunas pruebas en codigo que comprobasen el correcto funcionamiento de los programas desarrollados.???? Qué dice????

5. Utilizacion

Para ejecutar los programas se utiliza la consola o shell (convencionalmente), donde se corren ambos ejecutables. Dependiendo de qué tipo de conversión de archivo se desee, se usará alguno de los dos programas. La ejecución de cualquiera de estos puede recibir dos parámetros: el archivo de entrada y el archivo de salida. Si no recibe el primero, entonces se setea por defecto la entrada estándar (stdin); si no recibe el segundo, se setea por defecto la salida estándar (stdout).

Antes del nombre del archivo de entrada se debe especificar el flag '-i' y antes del nombre del archivo de salida, el flag '-o'.

Por ejemplo, para convertir un archivo de DOS a UNIX se ejecuta:

```
./dos2unix -i archivoDeEntradaWindows.txt -o archivoDeSalidaUnix.txt
```