ST0254 – Organización de computadores
Práctica 3: Procesamiento paralelo de cadenas de caracteres

I.S. José Luis Montoya Pareja
Especialista en Telecomunicaciones
Departamento de Informática y Sistemas
Universidad EAFIT
Medellín, Colombia, Suramérica

RESUMEN
El presente documento contiene de forma corta y simple, el enunciado de tercera y última práctica que realizarán los estudiantes del curso ST0254 – Organización de computadores en el semestre 2014-1.

PALABRAS CLAVE
Assembler, strings, Intel, Linux, Assembler, Bash.

ENUNCIADO

"Cadenas de caracteres

En programación, una cadena de caracteres, palabra, ristra de caracteres o frase (string en inglés) es una secuencia ordenada de longitud arbitraria (aunque finita) de elementos que pertenecen a un cierto lenguaje formal o alfabeto análogas a una fórmula o a una oración. En general, una cadena de caracteres es una sucesión de caracteres (letras, números u otros signos o símbolos).

Desde un punto de vista de la programación, si no se ponen restricciones al alfabeto, una cadena podrá estar formada por cualquier combinación finita de todo el juego de caracteres disponibles (las letras de la 'a' a la 'z' y de la 'A' a la 'Z', los números del '0' al '9', el espacio en blanco ' ', símbolos diversos '!', '@', '%', etc). En este mismo ámbito (el de la programación), se utilizan normalmente como un tipo de dato predefinido, para palabras, frases o cualquier otra sucesión de caracteres. En este caso, se almacenan en un vector de datos, o matriz de datos de una sola fila (array en inglés). Las cadenas se pueden almacenar físicamente:

- Seguidas.
- Enlazados letra a letra.

Generalmente son guardados un carácter a continuación de otro por una cuestión de eficiencia de acceso.

Un caso especial de cadena es la que contiene cero caracteres, a esta cadena se la llama cadena vacía; en teoría de autómatas es común denotar a la misma por medio de la letra griega ε" [1]

"**Bash Comando tr**

```
TR(1) User Commands TR(1)

NAME
       tr  -  translate  or  delete
characters

SYNOPSIS
       tr [OPTION]... SET1 [SET2]

DESCRIPTION
       Translate,  squeeze,  and/or
delete  characters  from  standard
input, writing to standard output.

       -c, -C, --complement
first complement SET1

       -d, --delete
delete  characters  in  SET1,  do  not
translate
```

-s, --squeeze-repeats
replace each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character

-t, --truncate-set1
first truncate SET1 to length of SET2

--help
display this help and exit

--version
output version information and exit

SETs are specified as strings of characters. Most represent themselves. Interpreted sequences are:

\NNN character with octal value NNN (1 to 3 octal digits)
\\ backslash
\a audible BEL
\b backspace
\f form feed
\n new line
\r return
\t horizontal tab
\v vertical tab
CHAR1-CHAR2 all characters from CHAR1 to CHAR2 in ascending order
[CHAR*] in SET2, copies of CHAR until length of SET1
[CHAR*REPEAT]
REPEAT copies of CHAR, REPEAT octal if starting with 0
[:alnum:]
all letters and digits
[:alpha:]
all letters
[:blank:]
all horizontal whitespace
[:cntrl:]
all control characters
[:digit:]
all digits
[:graph:]
all printable characters, not including space
[:lower:]
all lower case letters
[:print:]
all printable characters, including space
[:punct:]
all punctuation characters
[:space:]
all horizontal or vertical whitespace
[:upper:]
all upper case letters
[:xdigit:]
all hexadecimal digits
[=CHAR=]
all characters which are equivalent to CHAR

Translation occurs if -d is not given and both SET1 and SET2 appear.
-t may be used only when translating. SET2 is extended to length of SET1 by repeating its last character as necessary. Excess characters of SET2 are ignored. Only [:lower:] and [:upper:] are guaranteed to expand in ascending order; used in SET2 while translating, they may only be used in pairs to specify case conversion. -s uses SET1 if not translating nor deleting; else squeezing uses SET2 and occurs after translation or deletion.
” [2]

Ejemplo de uso del comando tr

```
jlmontoy@linux(/home/jlmontoy)# y="This is a test"
jlmontoy@linux(/home/jlmontoy)# echo $y | tr '[:lower:]' '[:upper:]'
THIS IS A TEST
jlmontoy@pbmdeapnim02(/home/jlmontoy)# cat prueba.txt
Bienvenidos

Este es un texto de prueba, para validar que se cumpla lo requerido de pasar, caracter por caracter, el texto de:

- Mayusculas a Minusculas
- Minusculas a Mayusculas

Felicitaciones por intentarlo!
jlmontoy@pbmdeapnim02(/home/jlmontoy)# tr '[:lower:]' '[:upper:]' < prueba.txt > pruebaMAY.txt
jlmontoy@pbmdeapnim02(/home/jlmontoy)# cat pruebaMAY.txt
BIENVENIDOS

ESTE ES UN TEXTO DE PRUEBA, PARA VALIDAR QUE SE CUMPLA LO REQUERIDO DE PASAR, CARACTER POR CARACTER, EL TEXTO DE:

- MAYUSCULAS A MINUSCULAS
```

```
- MINUSCULAS A MAYUSCULAS

FELICITACIONES POR INTENTARLO!
jlmontoy@pbmdeapnim02(/home/jlmontoy)#
```

Uno de los problemas típicos de las cadenas de texto es convertir un texto a mayúsculas o minúsculas. Lo normal es que, por ejemplo, usando el comando tr, se haga el proceso de manera secuencial carácter por carácter. Que tal si se plantea un modelo que permita realizar mediante instrucciones MMX, el proceso de pasar de mayúsculas a minúsculas o viceversa? Y que podamos comparar los tiempos de ejecución con el comando tr y con el algoritmo que ustedes diseñen?

OBJETIVOS

1. Objetivo General

Realizar el proceso de construcción de un programa que realice el procesamiento mediante comandos MMX del proceso de convertir de mayúsculas a minúsculas o de minúsculas a mayúsculas un texto dado.

2. Objetivos Específicos

Crear un programa en assembler para Linux que realice las siguientes tareas:

1. Lea mediante línea de comandos, la instrucción a realizar. Por ejemplo:

```
trmmx upper lower < archivo.txt > nuevoarchivo.txt
```

Este comando convierte a minúscula todo el contenido del archivo "archivo.txt" y deja la respuesta en el archivo "nuevoarchivo.txt"

```
trmmx lower upper < archivo.txt > nuevoarchivo.txt
```

Este comando convierte a minúscula todo el contenido del archivo "archivo.txt" y deja la respuesta en el archivo "nuevoarchivo.txt"

2. Abra el archivo especificado en la línea de comandos.
3. Lea el archivo y mediante instrucciones MMX convierta dicho archivo en mayúsculas o minúsculas según sea la instrucción dada.
4. Guarde en un archivo nuevo, la salida.
5. Si hay errores, estos deben mostrarse en la salida estándar de errores.
6. Validar el tiempo de ejecución de la nueva rutina comparada con el comando tr para un archivo de texto grande.

CONSIDERACIONES GENERALES

1. Cada semana de la práctica se debe entregar un informe de avance de la misma. El informe debe incluir:
    a. El estado actual de la práctica
    b. Problemas encontrados
    c. Acciones tomadas para resolverlos

2. El desarrollo de la práctica puede ser individual o en grupos de máximo tres personas.

3. El informe final de la práctica se entregará en un archivo siguiendo el formato de la IEEE para la publicación de artículos; el enunciado de la práctica estará escrito en dicho formato.

4. Cualquier entrega relacionada con la práctica se realizará por recepción de trabajos de Eafit Interactiva (cualquier otro medio no será admitido).

5. Si hay cambios en los grupos, enviar al profesor a más tardar el jueves 8 de mayo a las 11:59 a.m. el nombre de las personas que conforman cada grupo.

6. El informe final deberá contener una breve descripción de cómo funciona el programa, que dificultades debieron superar para el desarrollo de la práctica y posibles mejoras que consideran, se puede hacer a la misma.

7. La práctica se debe realizar en Linux usando la máquina virtual que se podrá descargar desde Google Drive. Cada grupo deberá notificar al profesor cual es el estudiante que tendrá acceso a descargar la

máquina virtual (debe tener correo de Google).

8. El ensamblador preferido para Linux será el nasm y el linker será el ld. También se puede usar el linker de GCC.

9. Forma de evaluación:
    a. Entrega de los integrantes de la práctica (5%)
    b. Entrega a tiempo de la práctica (5%)
    c. Cumplimiento del primer objetivo (5%)
    d. Cumplimiento del segundo objetivo (5%)
    e. Cumplimiento del tercer objetivo (20%)
    f. Cumplimiento del cuarto objetivo (20%)
    g. Cumplimiento del quinto objetivo (5%)
    h. Cumplimiento del sexto objetivo (5%)
    i. Cumplimiento en el envío de avances (5%)
    j. Sustentación de la práctica (25%)

FECHA DE ENTREGA

Viernes 30 de Mayo, hasta las 11:50 p.m. a través de Eafit Interactiva). Las sustentaciones serán el 3 y 4 de Junio a medio día.

REFERENCIAS

[1] http://es.wikipedia.org/wiki/Cadena_de_caracteres

[2] http://unixhelp.ed.ac.uk/CGI/man-cgi?tr+1