PROYECTO
DE
PROGRAMACIÓN EN ASSEMBLER

Profesor: Leonardo De Mateis
Asistente: Gabriela Díaz

UNIVERSIDAD NACIONAL DEL SUR

10/10/2017
Giampietri Gonzalo – Zárate Tomás

1. Definición y especificación de requerimientos.

El desarrollo del proyecto consiste en implementar un programa que contabilice, enumere y muestre o escriba las líneas de un archivo de entrada dependiendo de los argumentos ingresados por el usuario.

Se considera que una línea está numerada, cuando antes de mostrar/escribir la misma, se agrega el número de línea seguido de ": " (dos puntos y un espacio).

Con el objetivo de cumplir los requerimientos, se implementó todo el sistema en lenguaje ensamblador: un programa principal que cumple con los objetivos propuestos.

Aspectos técnicos y tecnologías empleadas.

Lenguaje/s y herramientas de Programación utilizado/S:

- Ensamblador Yasm: lenguaje de programación de bajo nivel para sistemas operativos GNU/linux bajo arquitectura Intel i386.

2. Procesos y servicios ofrecidos por el sistema.

Programa Principal – enum

Enum contabiliza, enumera y muestra/escribe la líneas de un archivo de entrada dependiendo de los argumentos ingresados por el usuario.

El programa se invoca a través de la consola. Recibe como argumento, por línea de comandos, el nombre de un archivo de texto compuesto por todo tipo de caracteres. Enum se invoca de la siguiente manera:

enum [-h] | archivo entrada [archivo salida]

Las opciones separadas por "|" (barra vertical) denotan posibles alternativas a llamadas por línea de comandos, y los parámetros entre [] (corchetes) denotan parámetros opcionales.

Operaciones ofrecidas por *enum.*
Enum ofrece tres tipos de operaciones dependiendo del parámetro/s ingresado/s por el usuario :

- Si se especifica únicamente el parámetro -h, el programa debe ofrece una pequeña ayuda por pantalla la cual refleja un breve resumen del propósito general del programa junto con la especificación de las diferentes posibilidades que ofrece para ser invocado.

- Si se especifica únicamente el parámetro archivo entrada, el programa lee las líneas (línea por línea) contenidas en el archivo entrada, y muestra cada una de éstas líneas de forma numerada en la consola.

- Si se especifica tanto el nombre del archivo entrada como el nombre del archivo salida, el programa lee las líneas (línea por línea) contenidas en el archivo entrada, y escribe cada una de estas líneas de forma numerada en el archivo salida.

3. Documentación técnica

Secciones

- **section .data**: Sección que contiene los datos inicializados.

- **section .bss**: Sección que contiene los datos no inicializados.

- **section .text**: Sección que contiene el programa.

Datos inicializados

- **contador**: Mantiene el la cantidad de líneas contadas.
- **limite**: Determina la cantidad límite de líneas que es capaz de leer el programa desde el archivo de entrada.
- **cantidadDigitos**: Determina la cantidad de dígitos que tendrá el número que numera cada línea.
- **mhelp**: Contiene el mensaje de ayuda a mostrar cuando se especifica el argumento -h.
- **tHelp**: Determina el tamaño del mensaje de ayuda mHelp.
- **mContadorLineas**: Contiene el mensaje a motrar/escribir cuando se termina de enumerar todas las líneas del archivo de entrada.
- **tContadorLineas**: Determina el tamaño del mensaje mContadorLineas.

Datos no inicializados

- **contadorChar**: especifica la cantidad de dígitos que tendrá cada número de líneas.
- **buffer**: Espacio donde se almacenan los caracteres a escribir/imprimir.
- **cantBuffer**: especifica el tamaño del buffer.
- **archEntrada**: rita del archivo de entrada especifica por el usuario.
- **archSalida**: rita del archivo de salida especifica por el usuario.
- **descriptorArchivo1**: File descriptor del archEntrada.
- **descriptorArchivo2**: File descriptor del archSalida.
- **entrada**: File descriptor del archivo de entrada. Establece desde donde se leen líneas.
- **salida**: File descriptor del archivo de salida. Establece desde donde se leen líneas.

Etiquetas utilizadas

- **_start**: Establece el inicio del programa

- **unArgumento**: El usuario ha ingresado un solo argumento. Se procede a determinar si se trata del argumento -h o del nombre de un archivo de entrada.

- **posibleAyuda**: Se ha ingresado un “-” como parte del primer argumento, se procede a determinar si se trata de argumento -h. En el caso de que no se trate del argumento -h, el programa termina con error.

- **mostrarAyuda**: Se ha ingresado -h como primer argumento, se procede a mostrar el contenido del texto de ayuda por consola.

- **mostrarXConsola**: Procesa el archivo, enumera sus líneas y muestra el resultado por consola.

- **dosArchivos**: Se han ingresado dos argumentos, que corresponderían a nombres de los archivos de entrada y salida. Se abren ambos archivos, y se procede a numerar las líneas del archivo de entrada y luego a copiarlas en el archivo de salida con sus respectivas numeración.

- **pasarNumAChar**: mapea el contador de líneas, de número a su código ASCCI correspondiente. Para luego imprimirlo/escribirlo.

- **obtenerASCII**: Calcula el códio ASCCI del contador de líneas.

- **enumerarLineas**: lee el archivo de entrada, numera las líneas y las rescribe (con su respecivo número de línea).
- **loopArchivo**: Ciclo que permite leer el archivo de entrada y numerar sus líneas.
- **loopEnter**: ; Ciclo que lee una línea caracter por caracter hasta leer salto de línea (enter).
- **leerEnter**: ; Se leyó un salto de línea, se agrega otro a los procesado para compensar el salto de línea leido.
- **escribirResultadoFinal**: Escribe la cantidad de líneas enumeradas.
- **abrirArchivo**: Abre el archivo (de entrada o de salida) especificado.
- **cerrarArchivo**: Cierra el archivo (de entrada o de salida) especificado en EBX.
- **saltoDeLinea**: Escribe/imprime un salto de línea (enter).
- **leer**: Lee un carácter y lo coloca en el buffer.
- **escribir**: escribe lo que hay en el buffer en el archivo de salida o en la consola.
- **agregarContador**: Escribe el contador de la línea. Con dos puntos y espacio.
- **agregarContadorSolo**: Escribe el contador de línea sin dos puntos y espacio.
- **errorDeArchivoEntrada**: Terminación anormal del programa producida por algún error en el archivo de entrada.
- **errorDeArchivoSalida**: Terminación anormal del programa producida por algún error en el archivo de salida.
- **salidaExitosa**: Terminación normal del programa.
- **salidaError**: Terminación anormal del programa correspondiente a otros tipos de errores.

4. Alcance y limitaciones

1. Toda vez que el programa termine su ejecución, éste debe informar sobre la situación de terminación a quien haya invocado al mismo. Para esto debe utilizar la llamada al sistema sys_exit, respetando la siguiente convención:

EBX	Detalle
0	Terminación normal.
1	Terminación anormal por error en el archivo de entrada.
2	Terminación anormal por error en el archivo de salida.
3	Terminación anormal por otras causas.

2. Se considera que una línea está numerada, cuando antes de mostrar/escribir la misma, se agrega el número de línea seguido de “: ” (dos puntos y un espacio).
3. El programa enumera hasta un máximo de 9999 líneas. Una vez pasada las 9999 líneas, el programa para de enumerar e imprimir en consola o escribir en el archivo de salida.
4. Al ingresar un archivo de salida cuyo contenido es no vacío, el programa sobre-escribe las primeras n+2 líneas (un espaciado + el resultado final), donde n es la cantidad de líneas del archivo de entrada.

5. Detalles y Conclusiones

Con tan solo ingresar el argumento -h, el programa muestra el texto de ayuda. Es decir, el programa muestra el texto de ayuda independientemente de que si se haya colocado o no más caracteres después de la “h”. Por ejemplo: Con el argumento “-hola”, el programa muestra el texto de ayuda.