Algoritmos y Programación I - 75.02
Guía de Ejercicios Nº 12
Archivos

1) Escribir un programa de comandos en línea de órdenes que convierta un archivo de texto a minúsculas o mayúsculas, dependiendo de la selección hecha por el usuario:

   Uso: convertidor <archivo origen> <archivo destino> <formato>

   en donde el formato puede ser:

   -u : a mayusculas (UUPERCASE)
   -l : a minusculas (LOWERCASE)

2) Escribir un programa de comando en línea de órdenes que cuente la cantidad de líneas de un archivo de texto. Consultar la documentación del comando `wc` de Unix en las `man pages`, en particular para el comando: `wc -l`.

3) Escribir un fragmento de código que permita imprimir datos directamente sobre la impresora, a través del flujo `stdprn` .

4) Escribir un programa de comando en línea de órdenes que permita numerar las líneas de un archivo de texto:
   Uso: `numerador <numero maximo> <archivo de salida>`

5) Se desea disponer de una función para depurar un programa, volcando la información sobre un archivo de texto histórico (log file), de acuerdo al siguiente prototipo:

```
status LogDebug(char archivo [], string datos);
```

   en donde “archivo” contiene la ruta al archivo sobre el cual *añadir* los datos (sin borrar los ya existentes), “datos” es la cadena de caracteres con la información a guardar, y “status” un tipo enumerativo con los símbolos ERROR y OK.
   Se pide: a) Definir los tipos `status` y `string`. b) Escribir el código de la función, teniendo en cuenta que el puntero “datos” puede ser nulo, producto de información faltante. c) Dar un ejemplo de invocación de la función.

6) Escribir un programa que abre dos archivos, en modo binario, uno para leer caracteres y otro para escribir los caracteres leídos del primero. Es decir que copia el archivo fuente en el destino. Analice el contenido binario de ambos archivos (por ejemplo con Norton Commander, View file, F3 ó F4).

7) Escribir un programa tal que lea cadenas con gets() y con fputs() las escriba en un archivo de nombre “prueba”. Para que finalice el programa, hay que ingresar una línea en blanco.

8) Escriba un programa que lea todas y cada una de las cadenas generadas por el programa del ejercicio anterior.

9) Escribir un programa que borre tabulaciones de un archivo de texto y las sustituya por una cantidad de espacios.

10) Escribir un programa de prueba de la función remove().

11) Escribir un programa que escriba un double, un int y un long en un archivo de disco y luego los lea del archivo. Usar fread(), fwrite(). Analizar el contenido con Norton Commander o similar.

12) Explicar el significado de las siguientes macros: SEEK_SET, SEEK_CUR o SEEK_END .

13) Escribir un programa de comandos en línea de órdenes que genere un archivo de texto con información de clientes morosos (saldo negativo) contenida en un archivo binario (entrada). El archivo binario de entrada contiene una secuencia de estructuras de longitud desconocida, del siguiente tipo:

```
typedef struct { char calle[MAX_LARGO]; int numero; int piso, char depto; } StDomicilio;
```

```
typedef struct { char CBU[MAX_LARGO]; double saldo; char categoria; } StCuentaBancaria;
typedef struct { char razonsocial[MAX_LARGO]; StDomicilio domicilio; StCuentaBancaria cuenta; }StCliente;
```

El programa debe ser invocado como: Morosos.exe <archivo de datos> <archivo a generar>

y el formato de una línea del archivo de texto a generar debe ser el siguiente, con valores delimitados por pipes ("|"): <Razon Social> | <CBU> | <Saldo> \n

Ejemplo:

```
Juan Perez|123456789|-141.24
Margarita Sosa|258654|-4.68
José Atomo|987654321|-278.25
```

14) Escribir un programa de comandos en línea de órdenes que permita extraer un campo contenido en un archivo de texto de tipo CSV (valores separados por comas), que sea invocado como:

extractor.exe <archivo de entrada> <número de campo> <archivo de salida>

Ejemplo: Si el archivo de entrada, “in.txt” está compuesto de los siguientes campos:

```
Juan, Perez, Otamendi 2274 PB, 4123-4567
Lucas, Pato, Cucha Cucha 373 4ºB, 4765-4321
```

al invocar al programa como:

extractor.exe in.txt 3 out.txt

se habrá de generar un segundo archivo de texto, “out.txt”, con el siguiente contenido:

```
Otamendi 2274 PB
Cucha Cucha 373 4ºB
```

Se debe validar los argumentos del programa e informar al usuario en caso de error. Debe contemplarse los casos en que el archivo esté vacío, y el que no exista el campo seleccionado.
Nota: Para la resolución del ejercicio puede suponerse que cada renglón del archivo de texto no contiene más de 16K caracteres, incluídos todos los delimitadores y el retorno de carro.

15) Se requiere disponer de un programa que permita convertir el contenido de un archivo de texto al "jeringoso", y que guarde el texto convertido sobre un segundo archivo de texto. Se pide: a) Escribir el código del programa C utilizando argumentos en línea de órdenes, de forma que el programa pueda ser utilizado mediante la siguiente invocación:

jeringo <archivo entrada> <archivo salida>

en donde <archivo entrada> es el nombre (con su ruta) del archivo con el contenido original, y <archivo salida> es el nombre (con su ruta) del archivo a ser creado con el contenido original traducido al jeringoso.

Se deben validar los argumentos recibidos por el programa, y alertar al usuario si no se puede abrir alguno de los archivos involucrados en la conversión.

Ejemplo: hola mundo -→ hopolapa mupundopo

16) Escribir un programa de comando en línea de órdenes que permita generar un lote de números aleatorios sobre un archivo de texto, de acuerdo a la siguiente forma de uso:

random <cantidad> <minimo> <maximo> <archivo destino>

en donde <cantidad> es el tamaño del lote, <mínimo> y <máximo> los extremos del intervalo al que pertenecen los números generados, y <archivo destino> el archivo destino donde han de volcarse las muestras.

17) Escribir un programa de comando en línea de órdenes que genere un archivo CSV con las muestras de una

función senoidal de la forma:

$$S(t) = A \cdot sin(2 \cdot p \cdot f \cdot t + f)$$

La invocación del programa debe responder a:

sample <forma > <cantidad> <amplitud> <frecuencia> <inicio> <fin> <fase> <archivo> <delimitador>

en donde:

<forma>	forma de onda, cadena literal "SENOIDAL".
<cantidad>	Cantidad de muestras a generar.
<amplitud>	Amplitud de pico de la senoidal.
<frecuencia>	frecuencia de la senoidal.
<inicio>	Valor inicial de la variable independiente.
<fin>	Valor final de la variable independiente.
<archivo>	Path del archivo CVS de destino.
<delimitador>	Secuencia de caracteres delimitadores (ej. |, "|", <coma>, <TAB>, etc.).

Comprobar el contenido del archivo resultante con una planilla de cálculo (importación de archivos CSV).

18) [*] Escribir un programa de comandos en línea de órdenes que convierta un archivo de texto de tipo CSV (comma-separated values) a formato HTML.
Uso: csv2html <input file> <output file> <delimitter> <web page title>

En donde <delimitter> es el carácter delimitador del arvchivo CSV y <web page title> el título a darle a la página HTML resultante.

19) [*] Idem a XML sin caracteres extendidos.

19) Escribir un programa que lee un archivo de texto e imprime en pantalla las primeras n líneas que recibe en la línea de comandos. Si n es mayor que el número de líneas presentes en el archivo de entrada, imprime hasta la última que encuentre en el archivo de entrada.

21) Escribir un programa similar al del ejercicio anterior, que imprime las últimas n líneas leídas desde el archivo de entrada.
Nota: Comparar este ejercicio y el anterior con los comandos `more` y `less` de Unix (consultar las `man pages`) .

22) Escribir un programa que concatena dos archivos de texto cuyos nombres recibe desde la línea de comandos.

23) Escribir un programa que busque una cadena de caracteres que recibe desde la línea de comandos, en un archivo de texto y determine si esa cadena existe o no. En caso de encontrarse dicha cadena, también deberá informar en qué línea se encuentra.

24) Documente los siguientes prototipos de funciones, indicando el significado de los parámetros formales, del tipo de dato que devuelve y cómo se invoca:

fgetc(), ungetc(), puts(), fseek(), ftell(), freopen(), fcloseall(), y rewind().

25) Un archivo llamado "polar.dat" contiene las coordenadas polares necesarias en un programa de gráficas. Actualmente este archivo contiene los siguientes datos:

Distancia (en cm.)	Angulo (grados)
2.0	45.0
6.0	30.0
10.0	45.0
4.0	60.0

12.0	55.0
8.0	15.0

Escribir un programa que cree este archivo.

26) Escribir un programa que lea el archivo creado en el ejercicio anterior y cree otro de nombre "xycord.dat". Las entradas al nuevo archivo deben contener las coordenadas rectangulares que corresponden a las coordenadas polares en el archivo "polar.dat".
Las coordenadas polares se convierten a rectangulares mediante las ecuaciones:

x = r*coseno(t)
y = r*seno(t)

donde r es la coordenada de distancia y t es el equivalente en radianes a la coordenada del ángulo en el archivo "polar.dat."

27) Escribir un programa invocable en línea de órdenes que permita restaurar un archivo de texto transmitido por FTP en un modo incorrecto. a) Desde plataforma UNIX a plataforma Windows™ b) Viceversa.

28) [*] Escribir un programa invocable por línea de órdenes que convierta un archivo CVS a formato HTML (tabla) y XML, en donde el formato de salida es también indicado como argumento. Por simplicidad, no traducir el juego de caracteres.

29) [Optativo, es extenso] Se tiene dos archivos con la misma estructura, es decir, ambos contienen información acerca de ciudades de la República Argentina: código de la ciudad (único para cada ciudad), nombre de la ciudad, nombre de la provincia a la que pertenece, nombre del departamento al que pertenece la ciudad, si es ciudad capital de la provincia correspondiente o no, cantidad de habitantes en la ciudad, fecha de fundación de la ciudad, sueldo actual promedio de un empleado público típico que trabaja en la ciudad.
Empleado público típico es un empleado administrativo de planta que cobra según el escalafón general, en cualquier repartición oficial, en la ciudad correspondiente. Esta información debe estar ordenada por provincia y por código de ciudad. Se debe generar un tercer archivo que contenga los datos de los dos archivos anteriores, ordenado por provincia y por código de ciudad.

30) [Optativo, es extenso] El dueño de una librería llamada “El Incunable” quiere automatizar su sistema de control de inventario. Por eso lo
ha contratado a usted para que desarrolle el sistema.
Después de algunas reuniones con el propietario para determinar las funciones que el sistema debe proporcionara a la librería, se plasman las condiciones que debe reunir el sistema de control de inventario, que son las siguientes:
a. El sistema debe estar funcionando mientras la librería esté abierta. Cuando la librería abra por la mañana, el programa leerá el inventario desde el archivo de inventario, "entrada.dat" .
Cuando, al finalizar el día, se cierre la librería, el inventario actualizado deberá grabarse en el mismo archivo.
b. Cuando la librería reciba un envío de libros, el inventario deberá actualizarse bien, mediante el aumento del número de libros en depósito o bien mediante la entrada de un nuevo libro, si aún no está contenido en el inventario.
c. Cuando los libros estén agotados, el inventario deberá actualizarse mediante el borrado de la entrada, pero sólo cuando el número de libros en reserva sea cero.
d. Cada vez que un libro sea vendido, el inventario deberá actualizarse.
e. El propietario quiere tener un listado completo del inventario, ordenado alfabéticamente por el título de los libros.
f. También quiere poder mostrar la información bibliográfica referente a un libro en concreto que haya en el inventario.
g. El programa debe generar una lista de los títulos de todos lo libros que haya en el inventario e indicar cuántas copias de cada libro se venden cada día.
En las reuniones han acordado la información referente a cada libro que se guardará en el inventario. La información siguiente es la que se almacenará en el archivo de inventario "entrada.dat" para cada libro:

- Título (50 caracteres como máximo)

- Apellido del autor (30 caracteres como máximo)
- Nombre del autor(20 caracteres como máximo)
- Precio
- Editorial (30 caracteres como máximo)
- ISBN
- Fecha de copyright
- Número de ejemplares disponibles
- Estado (1 para libros que pueden solicitarse a la editorial y 0 para los libros agotados en la editorial y que no se reimprimen)