Trabajo práctico integrador

Procesador de subtítulos

Seminario de lenguajes opción C

2013

El trabajo de este año consiste en el desarrollo en C de una aplicación que manipule archivos de subtítulos en formato SubRip.

El formato SubRip es uno de los formatos de subtítulos para vídeos más difundidos y compatibles. Normalmente los archivos SubRip tienen extensión .srt.

1. El formato

Un archivo SubRip está compuesto de una secuencia de subtítulos, cada subtítulo se almacena con el formato:

Figura 1: Formato SubRip

```
índice
tiempo_inicio --> tiempo_fin
Texto del subtítulo (una o más líneas)
línea_en_blanco
```

El índice es un número entero positivo representando el orden de los subtítulos, así el primer subtítulo siempre tendrá índice 1 y el tiempo se representa con formato HH:MM:SS,mmm donde:

1. HH son las horas relativas al comienzo del vídeo.
2. MM los minutos.
3. SS los segundos.
4. mmm las milésimas de segundo.

Figura 2: Ejemplo de subtítulo

```
1
00:00:20,000 --> 00:00:24,400
Altocumulus clouds occur between six thousand

2
00:00:24,600 --> 00:00:27,800
and twenty thousand feet above ground level.
```

2. Modalidad de la entrega

El trabajo se divide en 2 etapas, la entrega es individual y todo lo entregado se someterá a un testeo de similitudes usando el similarity tester de Dick Grune [1] seguido de una inspección visual de los casos sospechosos para detectar posibles copias, cualquier copia detectada es motivo suficiente para que los involucrados desaprueben la cursada.

Todas las etapas son obligatorias para aprobar la cursada.

La reentrega de la etapa 2 solamente está disponible para quienes entreguen la etapa 2 en la fecha original del 07/07 y asistan al coloquio correspondiente (desaprobando por errores o porque el código está incompleto y asistiendo al coloquio).

El calendario de entregas para lo que resta de la cursada es el siguiente:

Fecha	**Instancia de evaluación**
Semana del 26/05	Ejercicio de la práctica 3 + Etapa 1 del trabajo integrador + Coloquios
Jueves 13/06, Martes 18/06 y Miércoles 19/06	Autoevaluación 2 en los horarios de práctica.
Semana del 07/07	Entrega de la etapa 2 + coloquios.
Semana del 04/08	Recuperatorio de la etapa 2 y coloquio de quienes reentregaron.

Observación 1: El día Jueves 20/06 es feriado por lo que **la autoevaluación se hará el jueves anterior (es decir el 13/06)**.

Observación 2: Notar que en la semana del 07/07 el día martes es feriado, por lo que **el coloquio correspondiente al martes 09/07 se tomará el día lunes 08/07 a las 15:30hs en el aula 1-1**.

[1] http://dickgrune.com/Programs/similarity_tester/

2.1. Etapa 1

Consiste en un archivo de nombre README con detalles cómo el alumno planea modularizar e implementar la **etapa 2**. Este archivo es un archivo de texto plano sin formato (como los .txt).

Este documento deberá nombrar las estructuras de datos a usar, los prototipos de las funciones a implementar, los nombres de los archivos donde estarán las declaraciones y las definiciones, y una descripción de la funcionalidad de cada función a implementar.

Esta etapa se deberá entregar hasta el 26/05 a las 23:55hs a través del servidor SVN de la cátedra.

La entrega se considerará aprobada si se cumplen las siguientes condiciones:

1. Entrega subida en tiempo y forma en el path /tpintegrador/tags/etapa1/README del repositorio SVN del alumno.

2. El alumno asiste al coloquio que será en la semana de la entrega, en el horario de práctica elegido en Marzo.

3. El archivo README respeta el formato y nivel de detalle del ejemplo mostrado en la siguiente sección.

4. El archivo README entregado describe una forma razonable de solucionar la etapa 2 de este trabajo integrador.

5. El archivo README así como la defensa del alumno demuestran que el alumno sabe modularizar de forma que otro desarrollador pueda entender fácilmente la estructura del programa.

2.1.1. Ejemplo del nivel de detalle esperado

Si la aplicación en lugar de leer subtítulos debiera leer números por teclado e imprimirlos en orden inverso el archivo README debería tener una forma parecida al de la figura 3 (página 4).

Figura 3: Ejemplo de README de una aplicación distinta a esta entrega

```
Estructuras de datos a usar:
    - Pila (implementada con una lista enlazada)
Módulos:
    - pila.c -> operaciones para pila_t
     define todo lo declarado en pila.h

    - pila.h -> interfaz y tipo de datos pila_t
      declara el tipo pila_t y las siguientes funciones:

      pila_t pila_crear(); -> Retorna una pila vacía

      int pila_pop(pila_t *); -> Retorna el elemento
                                 del tope

      void pila_push(pila_t, int); -> Agrega un
                                      elemento
                                      en el tope

      int pila_empty(pila_t); -> Devuelve 1 si la pila
                                 está vacía

    - lista.c -> operaciones para lista_t
                 define todo lo declarado en lista.h

    - lista.h -> interfaz y tipo de datos lista_t
      declara el tipo lista_t y las siguientes funciones:

      lista_t lista_crear(); -> Retorna una lista vacía

      void lista_append(lista_t *, int); -> Agrega un valor
                                            al final de la lista

      int lista_last(lista_t); -> Retorna el último valor de
     la lista eliminándolo de la misma

Programa principal:
    - main.c define: int main(int argc, char **argv) que
          lee de teclado e invoca a las funciones declaradas
          en pila.h para invertir los números leídos y luego
          imprimirlos.
```

2.2. Etapa 2

Implementación de la funcionalidad completa de la aplicación como se detalla más abajo.

Se deberá entregar la etapa 2 hasta el 07/07 en el directorio /tpintegrador/tags/etapa2/ del repositorio SVN del alumno. Esa misma semana se harán los coloquios correspondientes a la entrega.

La entrega se considerará aprobada si se cumplen las siguientes condiciones:

- Se realiza la entrega por SVN en tiempo y forma.
- El programa funciona como es especificado en este documento sin fallar con ningún subtítulo de prueba ni combinación de parámetros (excepto que venga más de un *-f* o *-o*).
- El programa permite cargar archivos con líneas arbitrariamente largas sin fallar.
- El alumno asiste al coloquio y demuestra conocimiento sobre el código desarrollado.
- Se libera toda la memoria alguna vez alocada.
- Se lee correctamente desde archivos.
- Se hace un manejo correcto de arreglos y punteros.
- El programa está correctamente modularizado utilizando compilación separada y usando correctamente los `.h`.
- El programa es legible por cualquier desarrollador y compila sin warnings.

2.2.1. Funcionalidad

El programa deberá permitir manipular archivos SubRip. Para ello aceptará los siguientes parámetros:

- -d tiempo ← desplaza todos los subtítulos *tiempo* milisegundos (*tiempo* puede ser negativo).
- -s ← separa subtítulos que estén solapados.
- -f filein ← *file* es el archivo a procesar.
- -o fileout ← *fileout* es el archivo donde se guardarán los cambios.
- -b index ← borra el subtítulo número *index*.
- -i startmilis endmillis texto ← inserta el subtítulo *texto* en *startmilis* con duración hasta *endmillis*, corrige todos los índices de los subtítulos siguientes.

- -m millis ← establecer duración mínima de los subtítulos, cualquier subtítulo que dure menos se extenderá hasta ocupar *millis* o hasta que esté a 75 milisegundos del subtítulo siguiente (lo que suceda primero).
- -c ← aumenta la duración de los subtítulos que tengan demasiados caracteres por segundo hasta que tengan 25 cps o menos o hasta que esté a 75 milisegundos del subtítulo siguiente (lo que suceda primero, se puede reutilizar el código de -m).
- -M millis ← reduce el tiempo de los subtítulos que duren más de *millis* milisegundos.
- -v ← valida el subtítulo buscando los siguientes errores:
  - Los números de índice dentro del archivo no son números consecutivos ordenados.
  - Los números de índice no arrancan en 1.
  - El subtítulo i termina antes que empiece el subtítulo i + 1.
  - Dura menos de 1 segundo.
  - Dura más de 7 segundos.
  - Tiene líneas demasiado largas (más de 36 caracteres).
  - Tiene demasiados caracteres por segundo (más de 25).
  - Tiene más de 2 líneas.
  - Hay menos de 75 milisegundos entre el final del subtítulo i y el principio del i + 1.

  Si la validación encuentra algún error programa deberá imprimir el número de índice del subtítulo y una descripción del error, y deberá seguir validando el archivo en busca de más errores. Luego deberá seguir procesando el resto de los parámetros por más que alguna validación falle.

Modo de procesar los parámetros:

- El parámetro *-f* es obligatorio y no se puede repetir.
- El parámetro *-o* es obligatorio cuando hay opciones que modifican al archivo (todas excepto *-v*) y no se puede repetir.
- El resto de los parámetros son opcionales, pueden repetirse y las acciones que representan deben ejecutarse en orden.
- Si un parámetro se repite, la acción indicada por el mismo deberá repetirse.
- Los parámetros se reciben en cualquier orden (es conveniente hacer 2 pasadas, en la primera se procesan *-f* y *-o*, y en la segunda se procesa el resto de los parámetros).
- Si se recibe más de un *-f* o *-o* el programa deberá mostrar un mensaje de ayuda con los parámetros válidos y terminar con código de salida 1.

- Si no se recibe un parámetro obligatorio deberá mostrar la ayuda anterior y terminar con código de salida 2.
- Si se recibe un parámetro inválido (por ejemplo: *-z* o *-cM*) deberá mostrar la ayuda anterior y terminar con código de salida 3.

Si no se puede abrir el archivo de entrada o el de salida, el programa deberá mostrar un error indicando el nombre del archivo que no pudo abrir y terminar con código de salida 4.

Por ejemplo la siguiente invocación al programa verifica los errores, luego reduce el tiempo de los subtítulos que duran más de 7 segundos, verifica el resultado, borra el subtítulo 3 y verifica el resultado:

```
./tpintegrador -v -f "los_simpsons.srt" -o "salida.srt" -M 7000 -v -b 3 -v
```

Subtítulos para probar: http://www.subdivx.com/X2X-subtitulos.html

Nota 1: Se puede usar cualquier funcionalidad incluida en GNU libc.

Nota 2: Oficialmente la película más larga de la historia "La cura para el insomnio" [2] dura 5.220 minutos, unos 313.200.000 milisegundos, tener esto en cuenta a la hora de almacenar los tiempos y duraciones.

[2] http://es.wikipedia.org/wiki/The_Cure_for_Insomnia