Algoritmos y Programación II (75.41) Trabajo Práctico Nº2

22 de octubre de 2014

1. Consigna

Evan Henshaw-Plath y tres amigos suyos decidieron crear un *start-up*. El proyecto se trataba de una nueva forma de interacción entre personas, basada en mensajes cortos de longitud menor a 140 caracteres. Así nació Twitter. Con el tiempo, esta red de *microblogging* serviría para múltiples usos: desde alertas de emergencia en tiempo real[1], hasta las pequeñas nimiedades de la vida[2].

Sin embargo, Evan viajo a Uruguay en el año 2006, se casó y radicó en ese país, para despues vender su parte de la empresa por 7000 dólares.

Grave error. Ahora Twitter vale miles de millones, y Evan quiere recuperar el tiempo perdido haciendo un nuevo Twitter, más genial que el anterior: ¡150 caracteres cómo límite!

Para éste nuevo proyecto, Evan nos pidió ayuda para programar el módulo de ingreso y búsqueda de twits.

1.1. Comandos

El programa a realizar debe contar con las siguientes funcionalidades:

Nuevo twit: Ingresa un nuevo twit al sistema y le asigna un identificador único.

Favorito: Incrementa la popularidad de un twit.

[1] http://xkcd.com/723/
[2] http://survivingtheworld.net/Lesson132.html

Buscar twits: Busca una cantidad dada de twits filtrando por `#hashtag` o por `@autor`. Permite ordenar los resultados por fecha de publicación o por popularidad, y permite establecer una cantidad máxima de resultados.

1.2. Especificación

1.2.1. Comando: `twittear`

Formato: `twittear @autor mensaje`

`mensaje` contiene el contenido completo del twit. En las pruebas proporcionadas, el mensaje contiene únicamente caracteres alfanuméricos, espacios y los caracteres `@` y `#`. [3]

Descripción: Guarda en el sistema el nuevo twit.

Salida: `OK id`, siendo `id` el identificador que le asigna el sistema al twit registrado. Los identificadores deben partir de 0 y ser incrementales.

Ejemplo:

```
comando: twittear @diego hola mundo
salida:  OK 0
comando: twittear @javi implementando el #tp2 con @fedeB
salida:  OK 1
comando: twittear @fedeB usando #heap al infinito y mas
   alla
salida:  OK 2
```

1.2.2. Comando: `favorito`

Formato: `favorito id`

Descripción: Incrementa en uno la popularidad del twit cuyo identificador es `id`.

Salida: `OK id`, siendo `id` el identificador del twit modificado.

Ejemplo:

```
comando: favorito 2
salida:  OK 2
```

[3] El comportamiento ante otros caracteres no está determinado; es decir que cualquier comportamiento será considerado como válido.

1.2.3. Comando: `buscar`

Formato 1: `buscar #hashtag orden cantidad`

Formato 2: `buscar @autor orden cantidad`

Descripción: Permite buscar twits que contienen un `#hashtag` o `@autor` determinado.

`orden` puede tomar dos valores posibles:

`cronologico`: Los resultados deben imprimirse en orden **inverso** al que fueron creados (el twit más reciente primero).

`popular`: Los resultados deben imprimirse en orden de popularidad (el twit más popular primero). La popularidad de un twit se determina según cuántas veces fue marcado como favorito.

`cantidad` es la cantidad máxima de twits que se espera obtener en el resultado de la búsqueda. Si es 0, se debe buscar e imprimir todos los resultados posibles.

Salida: `OK cantidad`, siendo `cantidad` la cantidad de resultados obtenidos en la búsqueda.

Luego se deben imprimir todos los twits encontrados con el siguiente formato: `id @autor mensaje`.

Ejemplo: [4]

```
comando: buscar @daniR cronologico 1
salida:  OK 0
comando: twittear @daniR #hola @nacho
salida:  OK 0
comando: buscar @daniR cronologico 1
salida:  OK 1
         0 @daniR #hola @nacho
comando: buscar @nacho cronologico 0
salida:  OK 1
         0 @daniR #hola @nacho
comando: buscar #hola popular 10
salida:  OK 1
         1 @daniR #hola @nacho
```

[4] Ver más ejemplos en las pruebas.

1.2.4. Detección de errores

En todos los comandos puede ocurrir que el sistema no sea capaz de procesar el pedido. En ese caso, se debe informar imprimiendo un código de error apropiado. En las pruebas se detallan cada uno de los casos que deben estar implementados con el código de error correspondiente. Aquí se detalla la lista de códigos de error:

`ERROR_COMANDO_INVALIDO`: El comando recibido o los parámetros son inválidos.

`ERROR_TWIT_ID_INVALIDO`: No existe un twit con el identificador ingresado.

`ERROR_TWIT_DEMASIADO_LARGO`: El contenido del twit excede los 150 caracteres.

2. Pruebas

Para facilitar la implementación del programa se dispone de **pruebas automáticas**, que serán de utilidad para revisar que el sistema cumpla con la especificación.

Una vez descomprimido el archivo zip con las pruebas[5], se debe efectuar los siguientes pasos para correr las pruebas:

1. Compilar el programa (supongamos que se llama `tp2`)
2. Ejecutar[6]:

```
$ bash pruebas/correr-pruebas.sh ./tp2
```

Cada una de las pruebas está especificada en un archivo dentro de la carpeta de pruebas. El script `correr-pruebas` ejecutará el programa una vez por cada prueba, y verificará que la salida estándar del programa sea exactamente igual a la salida esperada.

[5] Puede descomprimirse en cualquier lugar; para el ejemplo suponemos que se guardó en la misma carpeta que el TP. Es decir, el ejecutable `tp2` quedaría al mismo nivel que la carpeta `pruebas` (que contiene el archivo `correr-pruebas.sh`).

[6] Para correr las pruebas es necesario disponer de un entorno con línea de comandos Bash. En Ubuntu y Mac OSX lo más probable es que ya esté instalado. Existen varias implementaciones para Windows; por ejemplo MSYS o Cygwin.

2.1. Pruebas de volumen

Se espera que el sistema funcione eficientemente ante un volumen numeroso de datos. Para facilitar esta prueba, incluimos el script `pruebas-volumen.py`. Lo que hace el script es imprimir en la salida estándar los comandos que generan twits aleatorios, los marca como favoritos y generan búsquedas. Ejemplo de uso:

```
$ python pruebas/pruebas-volumen.py 10000 | ./tp2
```

3. Criterios de aprobación

A continuación describimos criterios y lineamientos que deben respetarse en el desarrollo del trabajo:

3.1. Utilización de estructuras de datos

Para la realización de este trabajo es necesario utilizar al menos una de las estructuras de datos vistas en clase: tablas de hash, árboles binarios de búsqueda y heaps; además de crear las estructuras adicionales que se consideren necesarias.

Todos los tipos de datos deben estar implementados de la forma más genérica posible, correctamente documentadas, y con sus correspondientes pruebas unitarias.

3.2. Informe

El informe deberá consistir en una descripción del **diseño** del programa.

Recordar que la etapa de diseño es **anterior a la implementación**, por lo tanto lo que debe estar explicado en esta sección, utilizando texto y/o diagramas, es cómo se va a estructurar el código para cumplir con las especificaciones de la consigna.

Algunas preguntas que deberían responderse:

- A grandes rasgos, ¿cómo será el flujo del programa?
- ¿Qué estructuras de datos se utilizarán para guardar en memoria los twits?
- ¿Qué operaciones se efectuarán para cada comando?
- ¿Cuál es el orden de ejecución en tiempo y memoria de cada comando?

3.3. Programa

Además de satisfacer la especificación de la consigna, el programa debe cumplir los siguientes requerimientos:

- Debe estar adecuadamente estructurado y modularizado, utilizando funciones definidas en forma lo más genérica posible.
- El código debe ser claro y legible.

- Todas las funciones deben estar adecuadamente documentadas, y donde sea necesario, el código debe estar acompañado de comentarios.

4. Entrega

La entrega del trabajo consiste en:

- El informe y código fuente impresos. Para el código fuente utilizar una tipografía `monoespaciada`.
- El informe digital, en formato `pdf`
- Una versión digital de todos archivos de código fuente, separados del informe, en un archivo comprimido `.zip` o `.tar.gz` ¡no `.rar`!

Se aclara que por código fuente se entiende todos los archivos `.h` y `.c`, el archivo `Makefile` para poder compilar, y todos los archivos adicionales que sean necesarios para ejecutar el programa. No deben entregarse nunca archivos `.o` u otros archivos compilados.

El informe impreso debe entregarse en clase. Los dos últimos (PDF y código fuente) deben enviarse a la dirección electrónica `tps.7541rw@gmail.com` con el asunto `TP2 - <Padrón 1><Alumno 1>- <Padrón 2><Alumno 2>`

El plazo de entrega vence el **lunes 10 de noviembre de 2014**.