Funcionamiento de los Juegos de Prueba

Rosa María Jiménez & Bernardino Casas

1. Juegos de Prueba

La siguiente documentación informal describe de forma suscinta el formato de los ficheros usados por el driver genérico que prueba vuestra práctica. Esta información es necesaria tanto para crear vuestros propios juegos de prueba como para interpretar el resultado de usar los nuestros.

Los ficheros de texto que intervienen en cada juego de prueba son:

- Un fichero de extensión **.in**, que es el fichero de entrada (el que lee el driver).
- Un fichero de extensión **.out**, que recoge el resultado de la ejecución.
- Un fichero de extensión **.res**, que os proporcionamos para nuestros juegos de pruebas, que es el ".out esperado".

1.1. Estructura típica de un fichero .in

```
##1 Titulo del bloque 1
##! comentario general
##! comentario general
...
##! comentario general
...
operacion arg1 arg2 ... argk
#output esperado
operacion arg1 arg2 ... argk
#output esperado
operacion arg1 arg2 ... argk
#output esperado
...
##1.1 Titulo del subbloque 1.1
##! comentario general
##! comentario general
```

```
...
##! comentario general
...
operacion arg1 arg2 ... argk
#output esperado
operacion arg1 arg2 ... argk
#output esperado
operacion arg1 arg2 ... argk
#output esperado
...
```

Existen pues tres tipos de "comentario":

Headings: ##n (o ##n.m, ##n.m.p, etc.)
destinados a estructurar el juego de pruebas en bloques, subbloques, subsubbloques, etc.

Comentarios: ##! comentarios de tipo general

Output: #output esperado

Los dos primeros tipos de comentario pueden aparecer en cualquier punto del fichero .in, excepto entre una operación y su output esperado. Los comentarios de output deben aparecer siempre justo a continuación de la orden correspondiente.

En el fichero .out los dos primeros tipos de comentario se caracterizan por los prefijos ### y ###! respectivamente. Las operaciones del fichero .in aparecen como comentarios en el .out (con solo un simbolo # como prefijo) y los outputs esperados aparecen justo a continuación del output real, marcados con el prefijo ##.

1.2. Ejemplo

IN	OUT
##1 donde	###1 donde
##! testea la operacion donde	###! testea la operacion donde
init 1 10 3 FIRST_FIT	#init 1 10 3 FIRST_FIT
i A 10	#i A 10
i B 10	#i B 10
i C 10	#i C 10
i D 30	#i D 30
i E 20	#i E 20
donde F	#donde F
#<-1,-1,-1>	<-1,-1,-1>
	##<-1,-1,-1>
donde D	#donde D
#<0,1,0>	<0,1,0>
	##<0,1,0>
donde B	#donde B
#<0,0,1>	<0,0,1>
	##<0,0,1>

1.3. Como ejecutar los juegos de prueba

Nosotros os proporcionamos un driver (programa principal) hecho específicamente para probar los módulos de la práctica, con el código fuente, y uno o más juegos de prueba en el formato arriba descrito. Para utilizarlos, primero deberéis compilar el driver y montar un ejecutable con él, usando todos los archivos .o de los distintos módulos que habéis implementado, y asegurándoos de incluir la librería ***libesin*** (agregando la opción *-lesin* a la línea de comandos con que montáis el ejecutable).

Por cada juego de prueba os proporcionaremos dos ficheros: uno con extensión **.in** y el otro **.res**. Para ejecutar, por ejemplo, el juego de prueba jp1, (suponiendo que vuestro ejecutable se llame Driver), deberéis hacer:

```
./Driver < jp1.in >& jp1.out
```

El juego de pruebas se ha superado con éxito si el fichero **.out** que obtenéis es idéntico al **.res** que os hemos proporcionado, cosa que podéis comprobar ejecutando

```
diff jp1.out jp1.res
```

Recordad que para que una práctica o sesión de laboratorio sea evaluada positivamente tiene que pasar los juegos de pruebas públicos (*jp_public.in*), o sea, el fichero de salida que genere vuestra práctica debe ser exactamente igual al que os proporcionamos (*jp_public.res*).

2. Uso de DIFF

Para la comprobación de los juegos de pruebas, podéis usar el comando *diff* de unix para comparar los ficheros **.out** con los **.res**.

Ejemplo:

```
diff fichero1 fichero2
```

El formato de la salida del diff indica las diferencias que el programa ha encontrado entre los dos ficheros. En general hay 2 tipos básicos de diferencias:

1. alguno de los 2 ficheros contiene algo que no sale en el otro
2. alguno de los 2 ficheros contiene algo que no sale en el otro, pero a su vez este último tiene algo que no sale en el primero en su lugar.

En el primer caso la diferencia se describirá así:

```
nº linea(fichero1) a  nº lineas(fichero2)
> "lo que hay en fichero2 que no tiene fichero1"
```

o bien

```
nº lineas(fichero1) d nº linea(fichero2)
< "lo que hay en fichero1 que no tiene fichero2"
```

En el segundo caso:

```
nº lineas(fichero1) c nº lineas(fichero2)
< "lo que tiene fichero1"
---
> "lo que tiene fichero2 en su lugar"
```

La numeración de lineas indica en que lineas se encuentran las diferencias. Por ejemplo: 3,7 significa las lineas de la 3 a la 7.

IMPORTANTE: Si los dos ficheros son iguales, diff no devolverá nada.

Para más ayuda, más precisión, parámetros, etc. podéis obtenerla usando el comando '**man diff**'.

3. Uso de STLFILT

Stlfilt es un programa que os ayudará a interpretar los mensajes de error del compilador g++, sobre todo cuando tratéis con clases genéricas y strings. Es un parser que traduce parte de los mensajes de error, haciéndolos más claros.

3.1. Uso en la universidad

Para usarlo desde la escuela tenéis que utilizar el comando:

```
/home/public/esin/bin/gfilt [y a continuación todas las
         opciones y parámetros que usaríais en el
         compilador g++]
```

Por ejemplo:

```
/home/public/esin/bin/gfilt -ansi -Wall cj_enters.cpp
```

3.2. Uso en casa

Para usarlo en casa seguid los siguientes pasos:

1. bajaros **gstlfilt.tar** de:

   ```
   http://www.bdsoft.com/tools/stlfilt.html
   ```

2. desempaquetadlo con:

   ```
   tar (-)xvf gstlfilt.tar
   ```

3. para compilar, usad el comando

   ```
   gfilt [y a continuación todas las opciones y
          parámetros que usaríais en el compilador g++]
   ```