Graphviz и Sed : ПОСТРОЕНИЕ СХЕМЫ ИЕРАРХИИ НАСЛЕДОВАНИЯ

ПИСКУНОВ А.Г., ПЕТРЕНКО С.М.

21 июня 2006 г.

АННОТАЦИЯ

Данный документ обсуждает использование open source [7] утилиты dot из пакета Graphviz [3] и потокоориентированного редактора Sed [6] из набора утилит [5] для построения схемы иерархии наследования на примере классов C#. Использование dot обсуждалось в [1]

По адресу [2] находится введение в sed на русском языке от А.Соловьева. По адресу [4] находится обсуждение пакета Graphviz от CustisWiki.

Дата создания документа: 21.06.2006

ПОСТАНОВКА ЗАДАЧИ

Утилита dot может из текстового описания графа построить файл в одном из популярных графических форматов, из которых привлекают внимание такие как:

- GIF;
- JPEG;
- png - Portable Network Graphics format;
- PS - PostScript.

Далее, построенный файл может включаться в документацию. Полный список форматов, поддерживаемых пакетом Graphviz , можно найти на странице

http://www.graphviz.org/cvs/doc/info/output.html

Поэтому, для построения иерархии наследования классов осталось построить текстый файл на языке утилиты dot в такой форме

```
digraph _Имя_графа_ {
  общий_атрибут_графа1;
  общий_атрибут_графа2;
  ...
  вершина1 -> вершина2
  ...
}
```

В качестве вершиныX надо подставить имя класса. Слева будет записан потомок, справа - предок. Для примера будем использовать измененный пример программы, использованной в [1] , в которой усложним наследование классов b, d и e. В качестве одного из предков класса b, для иллюстрации множественного наследования испльзован интерфейс IComparer. Поддержка множественного наследования означает, что приведенные скрипты могут быть использованы для построения иерархии наследования C++. Определение классов разделяется символами новой строки и комментариями для целей тестирования.

---- File:./csharp/lib.cs

```
/** \file
    \brief файл с библиотекой
*/
using System;
using System.IO;   // foo
namespace adm_w{
 class a{
  static public void    wrLn (   string s   ) {        /// \callgraph
       Console.WriteLine (s);
  }
 }
 class b: a,
  /* multiple inheritance */ IComparer   {     int dummy;  }
 class c2
       : b   {     int dummy2; }
 class d2:
        app {     int dummy;  }
 class   // last class
     e : d2  {
   public int dummy;
   public d   foo;
 }
}
```

---- End Of File:./csharp/lib.cs

ПРЕОБРАЗОВАНИЕ ФАЙЛА С ПРОГРАММОЙ C#

Так как sed хорошо работает со строками сначала выполним следующую пред обработку c# :

- удалим однострочные комментарии;
- многострочные комментарий гарантированно растянем на несколько строк. Для этого, до и после каждой лексемы /* и */ поставим по символу новой строки;
- До и после каждой лексемы из набора:
  - двоеточие - ':',
  - запятая - ',',
  - фигурная открывающая скобка - '{'

  поставим по символу новой строки;
- несколько пробелом и табуляций заменим символом новой строки.

---- File:./csharp/newline.sed

```
# #           #   1)  delete single line comment
s/\/\/.*//
            #   2)  put myltiline comment to multi line
s/\/\*/\
&\
/g
s/\*\//\
&\
/g
            #  3)  surround different lexemas (:,{) with  new line
s/[:,{]/\
&\
/g

            #   4)  map blanks to new line
s/[  ][  ]*/\
/g
p
```

---- End Of File:./csharp/newline.sed

В скриптах редактора Sed (см. [2])

- символ '#' - играет роль комментария.

- буква s/reg_exp/rpls/ - команда замены цепочки регулярных выражений reg_exp на цепочку rpls. В регулярных выражения символ бэкслеш '\' экранирует специальные символы. К таковым, очевидно, относятся символы слеш '/'. Цепочка rpls может быть многострочной, при этом в выходной поток попадают символы новой строки. Символ '&' означает все_что_распознала_reg_exp.

- буква p - команда печати строки.

Командная строка для запуска Sed следующая:

```
sed -nf newline.sed lib.cs  >.lib.cs.1
```

Далее, удалим пустые строки из полученного файла .lib.cs.1:

```
grep . <.lib.cs.1  >>.lib.cs.2
```

Следующим шагом из потока примерно такого вида

```
  ...
class
b
:
a
,
/*
multiple
inheritance
*/
IComparer
{
int
dummy;
}
  ...
```

будем извлекать пары предок - потомок. Для этого

- удалим многострочные комментарии. Перед командной d - находятся два регулярных выражения задающих диапазон применение команды - от сих до сих;

- удалим строчки с символами ':'и ','. Перед командой d находится одно регулярное выражение. команда применяется ко всем строчкам содержащих заданную цепочку символов;

- для строк, находящихся между строчками с лексемами class и открывающаяся фигурная скобка '{' (/\bclass\b/,/{/ - регулярные выражения, задающие диапазон применение последующей группы команд в фигурных скобках) выполним следующую группу команд:

  - для строк, начиная с со строки с лексемой class и первой строчкой содержащей алфавитно-цифровую лексему распознанную \w\w* (непустая последовательность букв, цифр или подчерков):
  - удаляем строку с лексемой class;
  - заносим алфавитноцифровую лексему из буфера регулярного выражения в буфер обмена - команда h. Как не трудно догадаться это и есть имя класса потомка;
  - удаляем все остальные строчки.

- удаляем строку с символом открывающей фигурной скобки - '{';
- для каждой прочитанной строки входного файла (которая содержит имя класса родителя) в буфер регулярного выражения читаем строку с именем класса потомка, записанного в буфер обмена - команда G;
- символ новой строки заменяем на двоеточие;
- выводим полученную пару в выходной поток.

---- File:./csharp/mkpair.sed

```
#               1)     delete multiline comment
/\/\*/,/\*\//d
#               2)     delete : and ,
/:/d
/,/d

#               3)     between class and {
/\bclass\b/,/{/{
                       # class
                       # name1
                       # name2
                       # ...
                       # {    ------ or
                       # class
                       # name1
                       # {
                       # if class name1 +> remember  name1
 /\bclass\b/,/\b\w\w*\b/{
                       #   1) delete class
  /\bclass\b/d
                       #   2) save class- child
  /\b\w\w*\b/h
                       #   3) delete all the lines
  d
 }
 /{/d
```

```
                    #   4) restore name1

  G
                    #   5)  2 lines +> 1 line
  s/\n/: /
                    #   6) print pair (parent, child)
  p
}
```

---- End Of File:./csharp/mkpair.sed

Командная строка для запуска Sed следующая:

```
sed -nf mkPair.sed .lib.cs.2  >.lib.cl
```

Получаем поток такого вида

---- File:./csharp/.lib.cl

```
a: b
IComparer: b
b: c2
app: d2
d2: e
```

---- End Of File:./csharp/.lib.cl

Осталось переделать список пар потомок : ребенок в требуемую программу на языке dot:

- перед первой строчкой файла вставить название графа и его общие атрибуты. Команда вставить перед - 1i, 1 - первая строка входного потока; ;
- для каждой строчки входного потока создать строчку выходного потока, поменяв родителя и потомка местами и разделив их стрелкой ->. Регулярные выражения из reg_exp берутся в специальные скобки \(и \). Символы \2 и \1 в rpls - обозначают соответственно цепочку, распознанную выражением во второй группе скобок, цепочку, распознанную выражением в первой группе скобок;
- после последней строки файла поставить закрывающую скобку. Команда добавить после - $a, $ - означает последнюю строка входного потока.

---- File:./csharp/mkdot.sed

```
#
#     classParent: classChild +> to make dot graph
#
                        # 1) begin of graph
1i\
digraph X { \
  rankdir=LR;\
  node[shape=box]\

                        # 2) make  next arc
s/\([^ :]*\):[  ]*\(\w\w*\).*/ "\2" -> "\1"/p

                        # 3) end of graph
$a\
}
```

---- End Of File:./csharp/mkdot.sed

Командная строка для запуска Sed следующая:

```
sed -nf mkdot.sed <.lib.cl >hrrchy.dot
```

Требуемый файл получен:

---- File:./csharp/hrrchy.dot

```
digraph X {
  rankdir=LR;
  node[shape=box]

 "b" -> "a"
 "d2" -> "app"
 "c2" -> "b"
 "e" -> "d2"
 "b" -> "IComparer"
}
```

---- End Of File:./csharp/hrrchy.dot

Осталось применить утилиту dot, чтобы получить иерархию наследования:

```
dot -Tpng hrrchy.dot -ohrrchy.png
```

Полученная иерархия представлена на рисунке 1 .

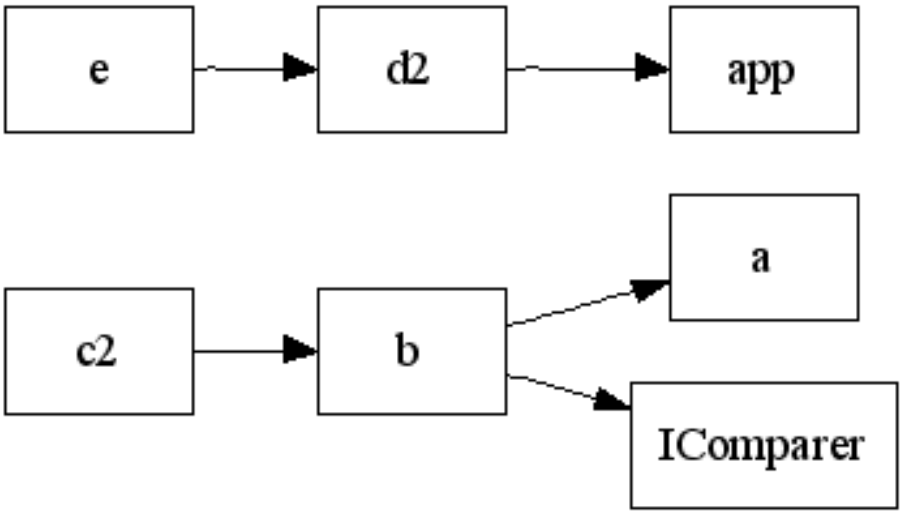

Рис. 1: схема иерархии наследования

НЕКОТОРЫЕ ВЫВОДЫ

Таким образом, мы убедились, что рассмотренные утилиты являются очень полезными, стабильными инструментами программиста широкого назначения, многократно окупающими усилия потраченные на их изучение. Sed , к примеру, был создан более четверти века назад.

ССЫЛКИ

[1] A.G. Piskunov, A.S.Gorban. Doxygen И Graphviz: documentaion of C projects. http://users.iptelecom.net.ua/~agp1/ru/dgIntro.html. 1, 2

[2] A.Solov'ev. SED: The stream text editor. http://www.lissyara.su/?id=1076. 1, 3

[3] ATT and Lucent Bell Labs. Graphviz - Graph Visualization Software. http://www.graphviz.org/. 1

[4] CustisWiki. Graphviz. http://lib.custis.ru/index.php?title=Graphviz&redirect=no. 1

[5] Karl M. Syring and Paul Falstad and so on. GNU utilities for Win32. http://unxutils.sourceforge.net/. 1

[6] Lee E. McMahon. SED: A Non-interactive Text Editor. http://sed.sourceforge.net/#docs. 1

[7] OpenSourceCommunity. SourseForge. http://sourceforge.net/. 1