Makefile 说明

By 13-叶嘉祺

● 简述

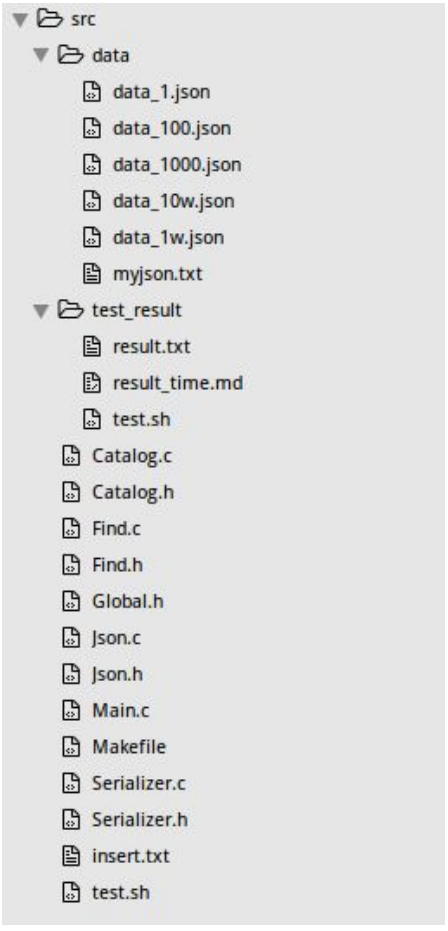
简单通俗地来说就是就是 linux 下的自动编译命令工具。对于用惯了 windows IDE 的人来说，对于这个会很陌生。Makefile 可以批量编译我们的项目，可以高效率地让我们进行项目的编写。会不会写 makefile，写的好不好，从一个侧面说明了一个人是否具备完成大型工程的能力。

● 前提要求

1、使用类 unix 系统（windows 下也可以实现）；
2、要求 linux 下基本的 CLI 操作；
3、要求 gcc，g++等工具链使用。

● 文档说明

1、本教程中使用的实例是数据库实验项目（c 语言项目），文件目录结构如下：



教程使用的 makefile：

```
main:Main.c Global Catalog Json Serializer Find
	gcc Main.c Catalog.o Json.o Serializer.o Find.o -o sinew -g
Global:Global.h
	gcc Global.h -c -g
Catalog:Catalog.h Catalog.c
	gcc Catalog.h Catalog.c -c -g
Json:Json.h Json.c
	gcc Json.h Json.c -c -g
Serializer:Serializer.h Serializer.c
	gcc Serializer.h Serializer.c -c -g
Find:Find.h Find.c
	gcc Find.h Find.c -c -g
100:
	./sinew -i ./data/data_100.json
1000:
	./sinew -i ./data/data_1000.json
1w:
	./sinew -i ./data/data_1w.json
10w:
	./sinew -i ./data/data_10w.json
checkout:
	./sinew -cc
find:
	sh test.sh
clean:
	rm -f sinew *.c~ *.h~ *.o *.h.gch
	rm -f *.*~ *~ *.bin .*.bin.* CATALOG.txt test_JsonSerializer.txt
```

- 运行 Makefile

  直接在终端中输入 make options[target]
  如上述文件运行 make 之后，就生成了 sinew 这个可执行文件
  如果运行 make checkout，就以-cc 参数运行了 sinew 程序

- Makefile 基本规则

target ... : prerequisites ...
[TAB]　　command1
[TAB]　　command2
.....

target：target 可以是一个可执行文件，可以是标签。这里我们通常对此自己命名。
prerequisites：要生成那个 target 所需要的文件或是 target。
command 也就是 make 需要执行的命令。（任意的 Shell 命令，make 命令也可以）
[TAB]:注意这里在写命令之前，必须写一个[TAB]按键，不可以使用空格代替。
例子：

```
Find:Find.h Find.c
	gcc Find.h Find.c -c -g
```

对应上述的 target 和 prerequisites

Command 命令指定，用 gcc 编译这两个文件，并且生成.o 文件

*Makefile 注释：使用#进行注释

● Makefile 变量

如果感觉编译过程中有些 objects 太多，使得整个 makefile 文件太过复杂，可以申明一些变量，简化结构，降低维护成本。
我们可以用这个结构简化上述代码：

```
objects = Catalog.o Json.o Serializer.o Find.o

main:Main.c Global Catalog Json Serializer Find
        gcc Main.c $(objects) -o sinew -g
```

● 依赖关系

在上述的过程中，当我们运行了 make 命令之后，会找到我们所有的 prerequisites，第一个找到 Main.c 这个文件，然后找到所有的依赖关系，如，第一个找到 Global 这个 target，然后 makefile 就会 Global 这个 target，先将它执行，依次类推（类似堆栈过程）。
执行结果如下：

```
gcc Global.h -c -g
gcc Catalog.h Catalog.c -c -g
gcc Json.h Json.c -c -g
gcc Serializer.h Serializer.c -c -g
gcc Find.h Find.c -c -g
gcc Main.c Catalog.o Json.o Serializer.o Find.o -o sinew -g
```

● 扩展学习

1、makefile 的自动依赖关系:

于是我们就没必要去在每一个[.o]文件后都写上类似的命令，因为，我们的 make 会自动识别，并自己推导命令。只要 make 看到一个[.o]文件，它就会自动的把[.c]文件加在依赖关系中，并且 cc -c whatever.c 也会被推导出来。
程序可以简化成这样：

```
objects = Catalog.o Json.o Serializer.o Find.o

main:Main.c Global.o Catalog.o Json.o Serializer.o Find.o
        gcc Main.c $(objects) -o sinew -g
Global.o:Global.h
Catalog.o:Catalog.h
Json.o:Json.h
Serializer.o:Serializer.h
Find.o:Find.h
```

2、伪命令

target ... :
[TAB] command1
[TAB] command2
.....

伪命令也就是省去 prerequisites。直接运行

```
10w:
        ./sinew -i ./data/data_10w.json
```

这个就运行了./sinew 程序，并且插入了 10w 数据。

Clean 命令

```
clean:
        rm -f sinew *.c~ *.h~ *.o *.h.gch
        rm -f *.*~ *~ *.bin .*.bin.* CATALOG.txt test_JsonSerializer.txt
```

我们运行 make clean 之后就可以清除所有编译过的文件和临时产生的数据文件，相当于可以把整个项目复位。这是一个很实用的命令。

● 思考题

1、写一个 makefile 文件，编译整个 Agenda 项目，并且只生成.o 文件，不生成可执行文件。
2、用简化的自动依赖关系来编写第一题。
3、写一个 makefile，当 make 的时候执行更新系统软件源命令：
    sudo apt-get update
4、写一个 makefile 文件，编译一个文件夹中，n 个不同学号的 agenda 子文件夹项目。

● 参考博客

http://blog.csdn.net/hanchaoman/article/details/5395491
http://blog.csdn.net/ruglcc/article/details/7814546/