§0.Linux 知识补充 - Makefile 文件的作用及编写方法

自行查阅相关资料，动手写测试程序，基于 CentOS 7，给出下列知识点的回答：

1 Makefile 文件的作用

2 Makefile 文件的基本语法

注：此处不必展开太多，关于编译优化的参数/选项等不要，能满足下面几种常用情况的要求就行

3 给出下列几种常用情况的 Makefile 文件的写法

3.1 每个人的目录结构要求如下(假设学号为 1651234，**各人按实修改**)：首先建立"学号-000103"子目录（可位于任意子目录下），下面再建立若干空的子目录，示例如下：

```
1651234-000103
  |-- 01
  |-- 02
  |-- 03
```

3.2 在子目录 01 下建立三个源程序文件，分别是 test1.c/test2.c/test3.c，形式如下（填入自己的学号及姓名即可），要求写一个满足下列要求的 makefile 文件

```
//test1.c
#include <stdio.h>
int fun1()
{
    printf();//打印你的学号
    return 0;
}
```

```
//test2.c
#include <stdio.h>
int fun2()
{
    printf();//打印你的姓名
    return 0;
}
```

```
//test3.c
#include <stdio.h>
int fun1();
int fun2();

int main()
{
    fun1();
    fun2();
    return 0;
}
```

- 执行 make 后即可将 test1.c/test2.c/test3.c 编译后形成一个可执行文件 test
- 任何一个.c 变化后，再次 make，只编译这个改变的.c，其余两个不需要重新编译（根据.o 的生成时间来判断是否重新编译过）
- 执行 make clean 后即可将.o 及可执行文件都清除掉，仅保留.c 源程序

3.3 在子目录 02 下建立四个源程序文件+头文件，分别是 test1.c/test2.c/test3.c/test.h,，形式如下（填入自己的学号及姓名即可），要求写一个满足下列要求的 makefile 文件

```
//test1.c
#include "test.h"
int fun1()
{
    printf();//打印你的学号
    printf();//打印 a 的值
    return 0;
}
```

```
//test2.c
#include "test.h"
int fun2()
{
    printf();//打印你的姓名
    printf();//打印 a*10 的值
    return 0;
}
```

```
//test3.c
#include "test.h"
int main()
{
    fun1();
```

```
//test.h
#include <stdio.h>
int fun1();
int fun2();
#define a 10
```

```
    fun2();
    return 0;
}
```

- 执行 make 后即可将 test1.c/test2.c/test3.c 编译后形成一个可执行文件 test
- 改变 test.h 中的 a 值后，再次 make，会重新编译这三个.c（根据.o 的生成时间来判断是否重新编译过）
- 执行 make clean 后即可将.o 及可执行文件都清除掉，仅保留.c 源程序

3.4 在子目录 03 下建立三个源程序文件，分别是 test1.c/test2.c/test3.c，形式如下（填入自己的学号及姓名即可），要求写一个满足下列要求的 makefile 文件

```
//test1.c
#include <stdio.h>
int main()
{
    //打印你的学号
    return 0;
}
```

```
//test2.c
#include <stdio.h>
int main()
{
    //打印你的姓名
    return 0;
}
```

```
//test3.c
#include <stdio.h>
int main()
{
    //打印你的学号+姓名
    return 0;
}
```

- 执行 make 后即可将 test1.c/test2.c/test3.c 分别编译为三个可执行文件 test1/test2/test3
- 执行 make test1 则只编译 test1.c，生成 test1 可执行文件
  make test2 ... test2.c ... test2 ...
  make test3 ... test3.c ... test3 ...
- 执行 make clean 后即可将.o 及可执行文件都清除掉，仅保留.c 源程序

3.5 在 1651234-000103 目录下写一个满足下列要求的 makefile 文件

- 执行 make 后，依次调用 01/02/03 目录下 makefile，分别编译出 1/1/3 个可执行文件，放在各子目录下
- 执行 make clean 后即可将.o 及可执行文件都清除掉，仅保留.c 源程序
- 本小题的 makefile，要考虑到子目录数量不定，每个子目录的名称不定，但每次 make 后仍能编译所有含 makefile 文件的子目录（测试时将 01/02/03 子目录换名，提交作业时仍换回 01/02/03 的名称）
- 问题：如果本小题的 makefile 被改名为 mymake，如果达到相同效果？（本问题不用提交作业，弄懂即可）

【本次作业的统一批改方法说明：】

1、 首先建立 16-000103 目录（可位于任意目录下）

2、 本次作业，每位同学上交一个 linux-makefile.tar.bz2 文件（注意：*.bz2 用什么命令压缩），截止时间到后，会从每人的交作业目录中复制出来，加学号换名后全部放在 16-000103 目录中
示例如下：

```
16-000103
|-- 1651234-linux-makefile.tar.bz2      （第 1 位同学的作业压缩包）
...
`-- 1654321-linux-makefile.tar.bz2      （最后 1 位同学的作业压缩包）
```

3、 进入到 16-000103 目录下，用 tar -xvjf 1651234-linux-makefile.tar.bz2 解压对应的作业文件，**要求得到**一个“学号-000103”子目录，下面再包含了各个小题的子目录
示例如下（为了简化，**未显示**所有的 tar.bz2 文件）：

```
16-000103
|-- 1651234-000103  （第 1 位同学解压得到的子目录）
|    |-- 01                          （第 3.2 小题对应的子目录）
|    |    |-- ***.c                  （第 3.2 小题的若干源程序文件）
|    |    `-- makefile               （第 3.2 小题对应的 makefile 文件）
|    |...
|    |-- 03                          （第 3.4 小题对应的子目录）
|    |    |-- ***.c                  （第 3.4 小题的若干源程序文件）
|    |    `-- makefile               （第 3.4 小题对应的 makefile 文件）
|    `-- makefile                    （第 3.5 小题对应的 makefile 文件）
……
|-- 1654321-000103  （最后 1 位同学解压得到的子目录）
|    |...
`-- check.sh       （老师事先建好的 shell 文件，预备编译所有同学的本次作业，具体的实
                     现方式是进入到每个学号对应的目录后调用该目录下的总 makefile）
```

4、 进入 16-000103 目录，进行一次./check.sh，就能检查完所有作业（只识别特定学号）
5、 **无法顺利编译则不能得分，对应学号及子目录名错则不能得分**
6、 **作业提交时清除所有的中间文件及生成的可执行文件、源程序备份文件等**
7、 **编译器用 gcc/c++/g++均可**
8、 **有多种 makfile 的写法，在正确的前提下，每位同学根据自己写法的不同得分可能略有差异**

【作业要求：】

1、10 月 10 日前网上提交
2、每题所占平时成绩的具体分值见网页
3、超过截止时间提交作业则不得分