

**学 生 实 验 报 告**

**（理工类）**



课程名称：Linux程序开发基础 专业班级： 22软件工程z班

学生学号： 2212001182 学生姓名： 孙澄宇

所属院部： 软件工程学院 指导教师： 孙菲艳

**2024 ——2025 学年 第 1 学期**

金陵科技学院教务处制

**实验报告书写要求**

实验报告原则上要求学生手写，要求书写工整。若因课程特点需打印的，要遵照以下字体、字号、间距等的具体要求。纸张一律采用A4的纸张。

格式要求：全文大标题为黑体，四号，1.5倍行距。

**实验报告书写说明**

实验报告中一至四项内容为必填项，包括实验目的和要求；实验仪器和设备；实验内容与过程；实验结果与分析；填写注意事项

（1）细致观察，及时、准确、如实记录。

（2）准确说明，层次清晰。

（3）尽量采用专用术语来说明事物。

（4）外文、符号、公式要准确，应使用统一规定的名词和符号。

（5）应独立完成实验报告的书写，严禁抄袭、复印，一经发现，以零分论处。

**实验报告批改说明**

实验报告的批改要及时、认真、仔细，一律用红色笔批改。实验报告的批改成绩采用百分制，具体评分标准由各院部自行制定。

**实验成绩评价及分析**

实验成绩表中实验项目名称、学时与实验大纲、实验计划表中信息保持一致；实验课结束后学生对该门实验课就自身能力产出展开评价；任课教师结合该课对应的能力要求、毕业指标给予评语。

**实验报告装订要求**

实验批改完毕后，任课老师将每门课程的每个实验项目的实验报告以自然班为单位、按学号升序排列，装订成册。

实验项目名称： Makefile 实验学时： 4

同组学生姓名： 无 实验地点： 科技楼1 **-** 2120

实验日期： 2024.9.4 实验成绩：

批改教师： 孙菲艳 批改时间： 2024.9.9

一、基于OBE模式的实验目的和要求

1、实验目的：

1. 理解 Makefile 规则和依赖的概念，掌握基本规则写法，包括伪规则。
2. 理解 Make 自动化的工作原理
3. 熟练掌握Makefile 提供的 “隐式推导” 机制、“变量” 机制、Makefile 提供的 “wildcard 和 patsubst” 函数
4. 熟练掌握采用 VPATH 和 vpath 两种方式实现项目的编译构造
5. 熟练掌握采用嵌套方式实现项目的编译构造

2、实验要求：

按照实验题目的要求，编写程序并上机调试

3、实验内容

见附件PPT

子任务1：

 任务描述：

 创建 hello.c文件，编写一个简单的 helloworld 程序

 使用命令行的方法手动编译 hello.c文件，经由预处理到编译，再到汇编，最后到链接的过程，分步执行，生成中间文件和最终的可执行程序，并确定每一步的依赖关系。

 编写 Makefile文件，完成以下要求：

* 编写显示规则自动执行并完成第 2 步的执行过程，生成中间文件和最终的可执行程序。
* 编写伪目标，清除生成的中间文件和最终的可执行程序。

 审核要求：

 正确编写源程序以及 Makefile文件， Makefile文件运行正常，包括生成结果（.i，.s，.o，a.out）正常，以及清除操作正常。

 提交编写的源程序以及 Makefile 文件，并记录在编写 Makefile 文件时遇到的错误和修改方法（如果有的话），代码思路清晰，编写规范。

子任务2：

 任务描述：

 分析项目文件的依赖关系，并参考 任务1 的解析方式画出 任务2 的项目文件依赖关系图。

 编写 Makefile文件，完成以下要求：

* 编写显示规则自动执行并完成对项目的编译和链接，生成最终的可执行程序（不要求生成中间文件 .i, .s） 。
* 编写伪目标，清除生成的中间文件和最终的可执行程序。

 审核要求：

 正确编写源程序以及 Makefile文件， Makefile文件运行正常，包括生成结果（.o，a.out）正常，以及清除操作正常。

 提交编写的源程序以及 Makefile 文件，并记录在编写 Makefile文件时遇到的错误和修改方法（如果有的话），代码思路清晰，编写规范。

子任务3：

 任务描述：

 利用 Makefile 提供的 “隐式推导” 机制对“任务2”的 Makefile 进行优化改造

 审核要求：

 1.完成项目构造的优化

 2.能正确运行优化后的 Makefile 文件

 3.对比“任务2” 和 “任务3” 的代码，体会采用 “隐式推导” 后代码的优化改进。

 4.提交全部程序代码文件。

子任务4：

 任务描述：

 利用 Makefile 提供的 “变量” 机制对“任务 2”的 Makefile 进行优化改造

* 尽量应用到课程中学到的各种变量类型，并进一步理解其含义和使用方法。

 审核要求：

 完成项目构造的优化

 能正确运行优化后的 Makefile 文件

 在 Makefile 中使用变量替换的地方用注释说明采用的是哪种变量类型

 对比“任务 2” 和 “任务 4” 的代码，体会采用 “变量” 后代码的优化改进。

 提交全部程序代码文件。

子任务5：

 任务描述：

 在 “任务 4” 的基础上，利用 “自动变量” 和 “规则通配” 技术对 Makefile 进一步优化。

 审核要求：

 按要求完成项目构造的优化

 能正确运行优化后的 Makefile 文件

 对比“任务 4” 和 “任务 5” 的代码，体会采用 “自动变量” 和 “规则通配” 技术的优化改进效果。

 提交全部程序代码文件。

 思考

 如何自动处理头文件的依赖关系？

子任务6：

 任务描述：

 利用 Makefile 提供的 “wildcard 和 patsubst” 函数对“任务 5”的 Makefile 进行优化改造，自动列出项目下的所有 .c 文件对应的 .o 文件。

 审核要求：

 按要求完成项目构造的优化

 能正确运行优化后的 Makefile 文件

 对比“任务 5” 和 “任务 6” 的代码，体会采用 “wildcard 和 patsubst” 后代码的优化改进。

 提交全部程序代码文件。

子任务7：

 任务描述：

 在 “任务 6” 的基础上进一步优化，在项目目录下创建一个新的 project.mk 文件，将 Makefile 中的变量定义部分抽取出来，放到该 project.mk 文件中，然后利用 Makefile 提供的 “include” 语句在 Makefile 中包含 project.mk，实现类似 c 语言中 “#include” 头文件的效果。

 审核要求：

 按要求完成项目构造的优化

 能正确运行优化后的 Makefile 文件

 对比“任务 6” 和 “任务 7” 的代码，体会采用 “include” 后代码的优化改进。

 提交全部程序代码文件。

子任务8 & 9：

 任务描述：

 在 “实训7” 的基础上，项目文件的目录存放方式发生改变如下，所有的源文件（包括头文件）不再存放在同一个目录下，而是将所有的 .h 文件存放在 “include” 目录下，所有的 .c 文件存放在 “source” 目录下，具体目录安排如右图所示：

 要求分别采用 VPATH 和 vpath 两种方式实现项目的编译构造。

 审核要求：

 按要求完成项目构造， VPATH 方式对应 “实训8”，vpath 方式对应 “实训9” 。

 能正确运行优化后的 Makefile 文件

 提交全部程序代码文件。

子任务10：

 任务描述：

 在 “实训8” 的基础上，“source” 目录拆分为两个 “source1” 和 “source2”。新增一个 “build” 目录，该目录用于存放所有编译中间生成的 .o 文件以及最终的可执行程序 edit。具体目录安排如右图所示：

 要求采用嵌套方式实现项目的编译构造。

 审核要求：

 按要求完成项目构造 。

 能正确运行优化后的 Makefile 文件

 提交全部程序代码文件。

二、实验仪器和设备

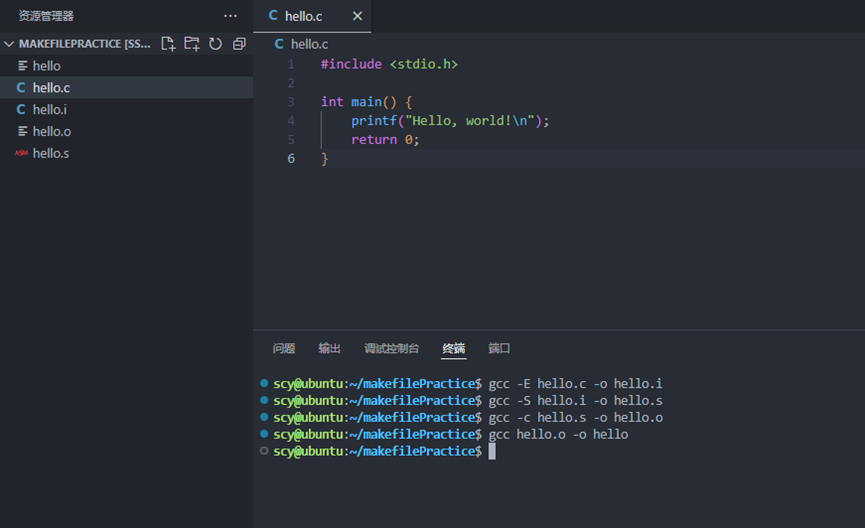
计算机、Linux、Gcc

1. 实验过程、结果与分析

（此处包括实验结果分析与截图，请注意截图务必和代码相吻合，截图需要显示用户名，标明题号下同）

子任务1：

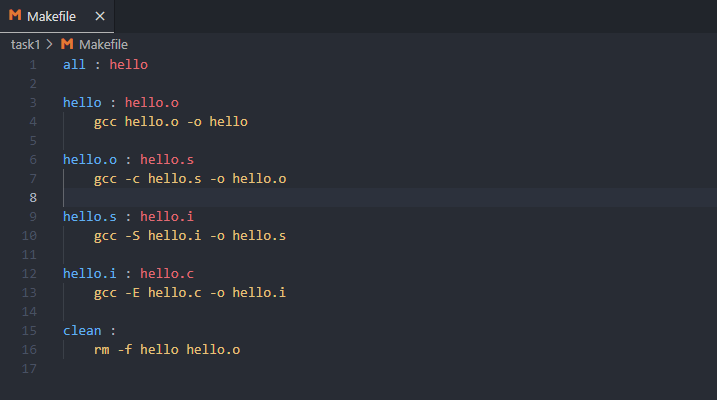
* 任务描述：
  + 创建 hello.c文件，编写一个简单的 helloworld 程序
  + 使用命令行的方法手动编译 hello.c文件，经由预处理到编译，再到汇编，最后到链接的过程，分步执行，生成中间文件和最终的可执行程序，并确定每一步的依赖关系。
  + 编写 Makefile文件，完成以下要求：
    - 编写显示规则自动执行并完成第 2 步的执行过程，生成中间文件和最终的可执行程序。
    - 编写伪目标，清除生成的中间文件和最终的可执行程序。
* 审核要求：
  + 正确编写源程序以及 Makefile文件， Makefile文件运行正常，包括生成结果（.i，.s，.o，a.out）正常，以及清除操作正常。
  + 提交编写的源程序以及 Makefile 文件，并记录在编写 Makefile 文件时遇到的错误和修改方法（如果有的话），代码思路清晰，编写规范。

创建hello.c，使用gcc完成手动编译

创建Makefile文件



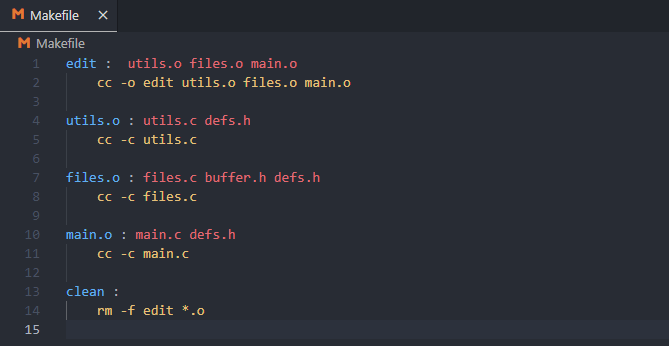
Makefile：



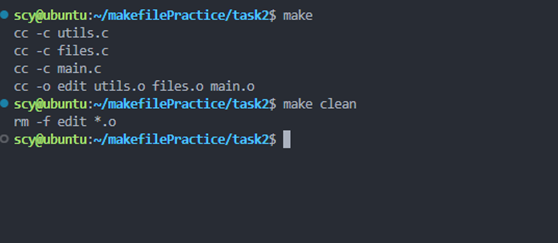
子任务2：

* 任务描述：
  + 分析项目文件的依赖关系，并参考 任务1 的解析方式画出 任务2 的项目文件依赖关系图。
  + 编写 Makefile文件，完成以下要求：
    - 编写显示规则自动执行并完成对项目的编译和链接，生成最终的可执行程序（不要求生成中间文件 .i, .s） 。
    - 编写伪目标，清除生成的中间文件和最终的可执行程序。
* 审核要求：
  + 正确编写源程序以及 Makefile文件， Makefile文件运行正常，包括生成结果（.o，a.out）正常，以及清除操作正常。
  + 提交编写的源程序以及 Makefile 文件，并记录在编写 Makefile文件时遇到的错误和修改方法（如果有的话），代码思路清晰，编写规范。

代码：



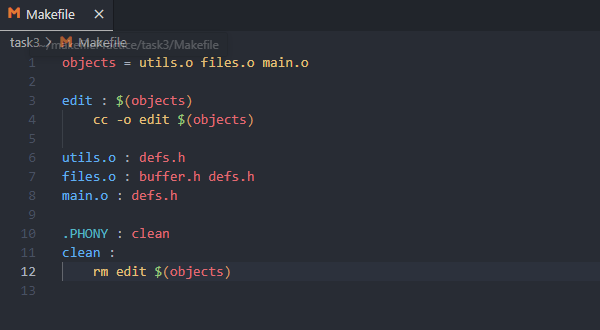
效果：



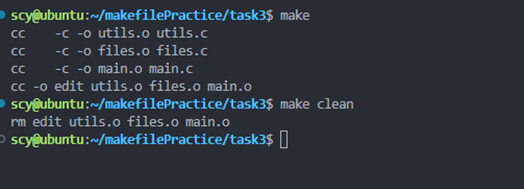
子任务3：

* 任务描述：
  + 利用 Makefile 提供的 “隐式推导” 机制对“任务2”的 Makefile 进行优化改造
* 审核要求：
  + 1.完成项目构造的优化
  + 2.能正确运行优化后的 Makefile 文件
  + 3.对比“任务2” 和 “任务3” 的代码，体会采用 “隐式推导” 后代码的优化改进。
  + 4.提交全部程序代码文件。

代码：



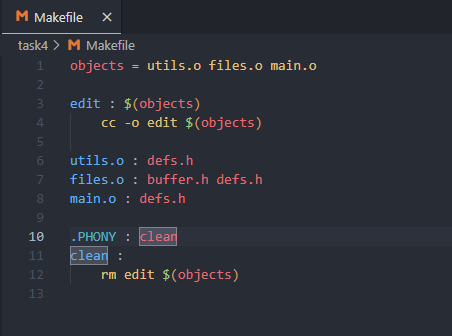
执行：



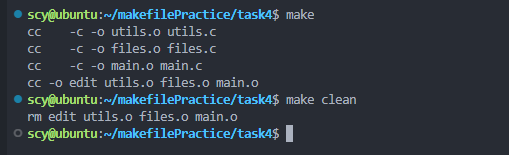
子任务4：

* 任务描述：
  + 利用 Makefile 提供的 “变量” 机制对“任务 2”的 Makefile 进行优化改造
    - 尽量应用到课程中学到的各种变量类型，并进一步理解其含义和使用方法。
* 审核要求：
  + 完成项目构造的优化
  + 能正确运行优化后的 Makefile 文件
  + 在 Makefile 中使用变量替换的地方用注释说明采用的是哪种变量类型
  + 对比“任务 2” 和 “任务 4” 的代码，体会采用 “变量” 后代码的优化改进。
  + 提交全部程序代码文件。

代码：



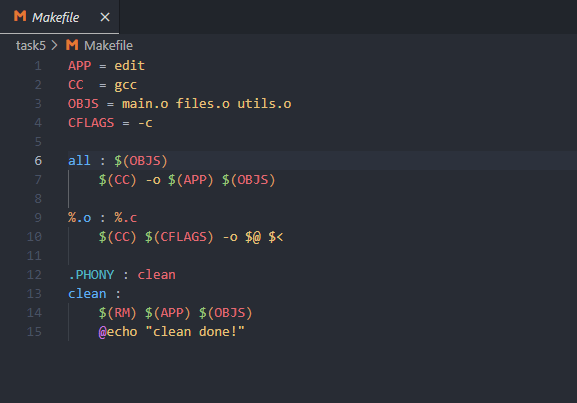
效果：



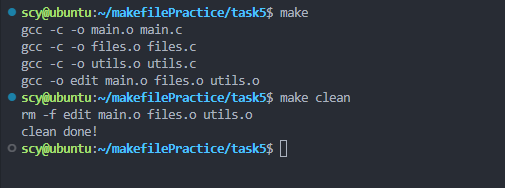
子任务5：

* 任务描述：
  + 在 “任务 4” 的基础上，利用 “自动变量” 和 “规则通配” 技术对 Makefile 进一步优化。
* 审核要求：
  + 按要求完成项目构造的优化
  + 能正确运行优化后的 Makefile 文件
  + 对比“任务 4” 和 “任务 5” 的代码，体会采用 “自动变量” 和 “规则通配” 技术的优化改进效果。
  + 提交全部程序代码文件。
* 思考
  + 如何自动处理头文件的依赖关系？

代码；



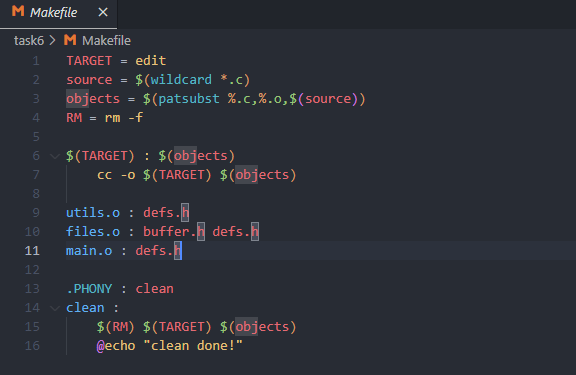
效果：



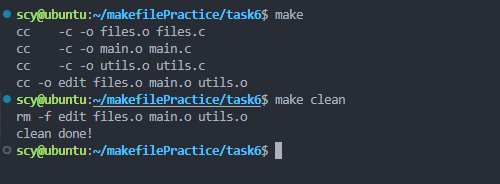
子任务6：

* 任务描述：
  + 利用 Makefile 提供的 “wildcard 和 patsubst” 函数对“任务 5”的 Makefile 进行优化改造，自动列出项目下的所有 .c 文件对应的 .o 文件。
* 审核要求：
  + 按要求完成项目构造的优化
  + 能正确运行优化后的 Makefile 文件
  + 对比“任务 5” 和 “任务 6” 的代码，体会采用 “wildcard 和 patsubst” 后代码的优化改进。
  + 提交全部程序代码文件。

代码：



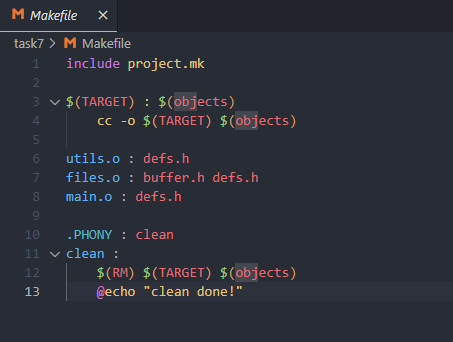
效果：

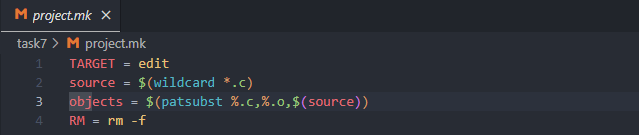


子任务7：

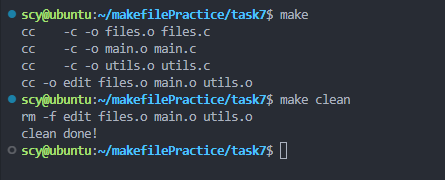
* 任务描述：
  + 在 “任务 6” 的基础上进一步优化，在项目目录下创建一个新的 project.mk 文件，将 Makefile 中的变量定义部分抽取出来，放到该 project.mk 文件中，然后利用 Makefile 提供的 “include” 语句在 Makefile 中包含 project.mk，实现类似 c 语言中 “#include” 头文件的效果。
* 审核要求：
  + 按要求完成项目构造的优化
  + 能正确运行优化后的 Makefile 文件
  + 对比“任务 6” 和 “任务 7” 的代码，体会采用 “include” 后代码的优化改进。
  + 提交全部程序代码文件。

代码；





效果：

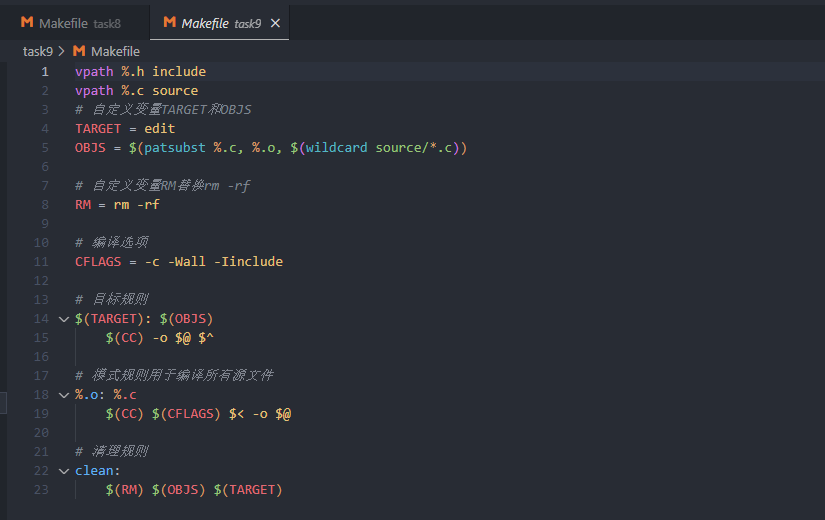


子任务8 & 9：

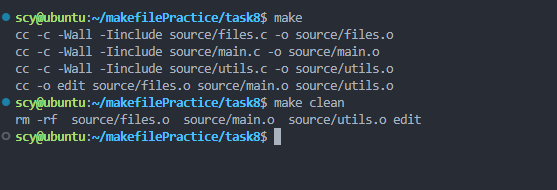
* 任务描述：
  + 在 “实训7” 的基础上，项目文件的目录存放方式发生改变如下，所有的源文件（包括头文件）不再存放在同一个目录下，而是将所有的 .h 文件存放在 “include” 目录下，所有的 .c 文件存放在 “source” 目录下，具体目录安排如右图所示：
  + 要求分别采用 VPATH 和 vpath 两种方式实现项目的编译构造。
* 审核要求：
  + 按要求完成项目构造， VPATH 方式对应 “实训8”，vpath 方式对应 “实训9” 。
  + 能正确运行优化后的 Makefile 文件
  + 提交全部程序代码文件。

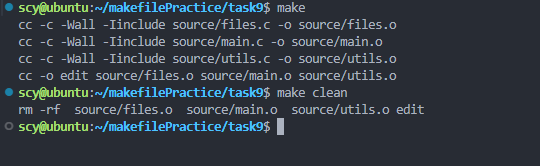
代码；





效果：



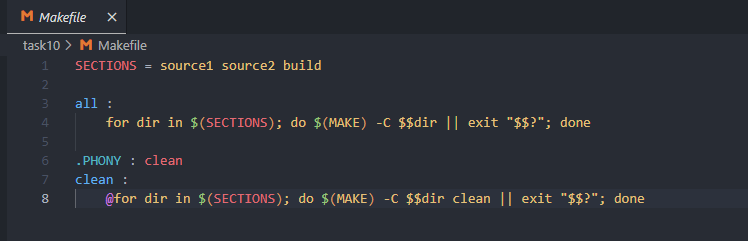


子任务10：

* 任务描述：
  + 在 “实训8” 的基础上，“source” 目录拆分为两个 “source1” 和 “source2”。新增一个 “build” 目录，该目录用于存放所有编译中间生成的 .o 文件以及最终的可执行程序 edit。具体目录安排如右图所示：
  + 要求采用嵌套方式实现项目的编译构造。
* 审核要求：
  + 按要求完成项目构造 。
  + 能正确运行优化后的 Makefile 文件
  + 提交全部程序代码文件。

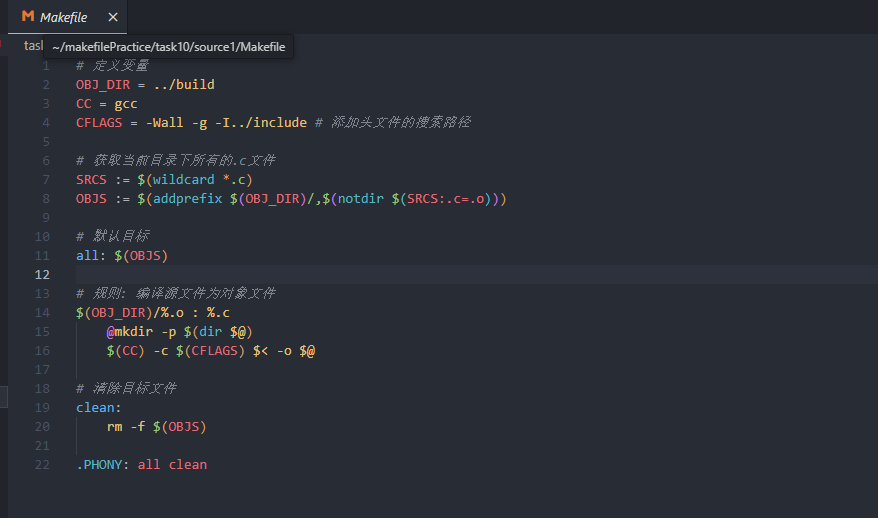
代码：

/Makefile

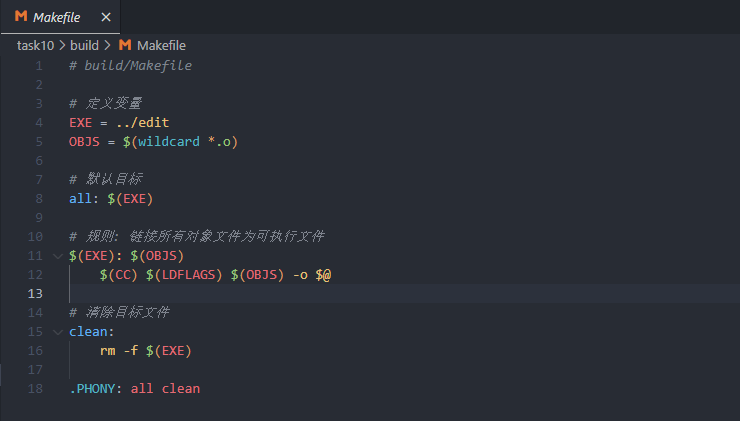


/source1/makefile

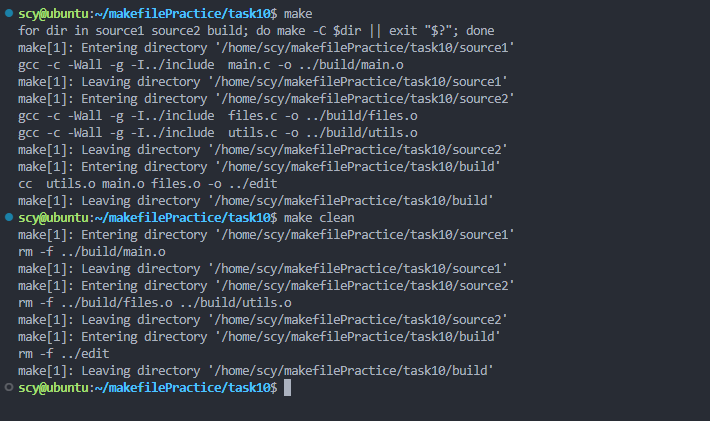
/source2/makefile



/build/Makefile



效果：



1. 基于OBE模式的学生自我评价与体会

在此次Makefile实训过程中，我完成了从需求分析到最终实现的所有步骤。通过这次实训，我不仅巩固了对Makefile基本概念的理解，还提升了自主解决问题的能力。以下是我对本次实训的一些个人体会和自我评价：

首先，这次实训让我深刻认识到了Makefile在软件项目构建中的作用。通过自己动手编写Makefile脚本，我掌握了如何定义目标文件、设置依赖关系以及编写编译命令等基本技能。这使我能够更好地管理和自动化软件构建流程，从而提高开发效率。

其次，独立完成实训任务的过程让我遇到了不少挑战。比如，在处理复杂的依赖关系时，我需要仔细考虑各个模块之间的相互作用；在解决编译错误时，我也必须耐心调试。这些经历锻炼了我的逻辑思维能力和问题解决技巧，让我在面对困难时能够保持冷静并找到合适的解决方案。

另外，通过这次实训，我还学会了如何利用网络资源进行自我学习。在遇到问题时，我积极查阅官方文档、论坛帖子以及相关教程，这不仅帮助我解决了实训中遇到的技术难题，也为我今后的自学提供了宝贵的经验。

最后，我认为这次实训对我的技术成长有着重要意义。它不仅增强了我的动手能力，还让我意识到持续学习和实践的重要性。我相信这些经验将为我未来的学习和职业生涯打下坚实的基础。

总的来说，尽管这次实训过程中充满了挑战，但我还是成功地独立完成了所有任务，并从中获得了许多宝贵的实践经验。这次经历不仅提升了我的专业技能，也增强了我的自信心，我相信这将是我前进道路上的一笔财富。