1. 问题：配置VS Code时无法下载VS Code Server

解决办法：VS Code版本过旧，卸载并下载新版本

2.根据对AI的搜索，得知Make工具的核心功能如下：

构建项目

- 它能根据项目的依赖关系和规则，自动构建软件项目。比如在C语言项目中，它可将多个源文件编译链接成可执行文件。通过读取Makefile中定义的目标、依赖和命令，按顺序执行编译命令，完成项目构建。

管理依赖关系

- 清晰描述项目中文件的依赖关系。例如，一个源文件可能依赖头文件，可在Makefile中指定这种关系。当依赖的头文件修改，Make能自动识别并重新编译相关源文件，保证项目的一致性和正确性。

提高编译效率

- 仅重新编译修改过的文件及其依赖文件，避免不必要的重新编译。如大型项目中只修改一个源文件，Make只会编译该文件及依赖它的目标文件，大大节省编译时间。

定制构建过程

- 可在Makefile中定义各种规则和命令，实现对构建过程的定制。如添加编译选项、链接库，执行自定义脚本等，满足不同项目的特殊构建需求。

3.Makefile 有以下主要语法规则：

基本结构

makefile

target: dependencies

command

 -  target 是要生成的目标文件或执行的目标动作，如可执行文件、目标文件等。

-  dependencies 是生成目标所依赖的文件或其他目标，多个依赖之间用空格分隔。

-  command 是生成目标需要执行的命令，需以一个Tab键开头。

变量定义与使用

- 定义变量： VAR\_NAME = value ，如 CC = gcc 。

- 使用变量： $(VAR\_NAME) ，如 $(CC) -o main main.c 。

模式规则

- 用于匹配一类文件的规则，如 %.o: %.c 表示所有 .c 文件对应生成 .o 文件的规则。

伪目标

- 用 .PHONY 声明，如 .PHONY: clean ，表示 clean 是伪目标，即使有同名文件，也会执行其定义的命令。

条件判断

makefile

ifeq ($(DEBUG), yes)

CFLAGS += -g

else

CFLAGS += -O2

endif

 - 可以根据条件来决定执行哪些命令或定义哪些变量。

函数调用

- 例如 $(wildcard \*.c) ， wildcard 函数用于获取当前目录下所有 .c 文件的列表。

4.Makefile语法规则示例解析：

- 第1行 app: main.c utils.c  定义了目标 app ，其依赖 main.c  和  utils.c  两个源文件。

- 第2行 gcc -o app main.c utils.c 是生成目标 app 的命令，使用 gcc 编译器，将 main.c  和  utils.c  编译链接成可执行文件 app 。这里命令前需有一个Tab键（图中未明显体现，但语法要求如此）。

5.make工具使用方法：

在终端进入包含Makefile的项目目录，直接输入 make  命令即可根据Makefile规则构建项目；若要执行特定伪目标，如 clean ，则输入 make clean  。

6.C语言编译流程（4个过程）：

- 预处理（Preprocessing）：处理 #include 、 #define 等预处理指令，生成.i文件。

- 编译（Compilation）：将预处理后的文件翻译成汇编代码，生成.s文件。

- 汇编（Assembly）：把汇编代码转化为机器码，生成.o文件。

- 链接（Linking） ：将多个目标文件和库文件链接成可执行文件。

7.C语言多文件开发方法：

将不同功能模块分别写在不同.c源文件中，用.h头文件声明函数和变量，在主文件中 #include 相应头文件，最后通过Makefile管理编译和链接过程。

1. 问题：在WSL:Ubuntu中的VS Code中运行代码时出现”launch:program’/root/projects/helloworld’does not exist”错误

原因：将Linux系统中的文件保存至Windows系统中去

9.个人学到WSL中的Ubuntu中的一些指令：

①mkdir 创建文件夹

②cd 移动当前所处文件夹中的位置

cd .. 返回上一级文件所在位置

cd \_ 返回下一级文件所在位置

③code . 打开Linux系统中的VS Code

④gcc XXX.c -o XXX 将XXX.c编译成一个可执行文件，该可执行文件的名称为XXX

⑤./ 用于运行当前目录下的可执行文件或脚本，后面增加内容，为argv[]，即命令行参数的内容;包括程序的名称，把所有的argv[]算在一起，可得到命令行参数的总数，即argc

⑥argc的计算规则为：只运行程序名称，则argc的值为1；

中间有n个空格，则argc为n+1;如果参数中带空格，则需要用引号将参数括起来

10.个人对阶段一的任务一的理解：

分三种情况：只运行程序的名称；参数中有”--help”；有参数但为其他参数。通过字符串的比较来参数检查，来把程序的运行归类，从而实现识别”--help”的存在和其他不正确的参数

1. Windows系统中的路径为“\”,Linux系统中的路径为“/”
2. 将Windows系统中的文件导入Linux系统的方法：

①确定Windows路径

如\Users\20682\Desktop\1.1.c

②在Windows系统中的VS Code中使用Ubuntu（WSL）的终端

（1）创建挂载点目录：

输入:sudo mkdir -p /mnt/c

（2）挂载C:盘：

输入:sudo mount -t drvfs C: /mnt/c

③访问目标文件

1. 进入目标路径:

如输入:cd /mnt/c/Users/20682/Desktop

1. 查看文件:

输入ls

1. 编辑文件：

如输入nano 1.1.c

④创建符号链接（可选）

为了方便访问，可以在WSL的家目录下创建一个符号链接:

输入ln -s /mnt/c/Users/20682/Desktop/1.1.c ~/

1. 个人对阶段一的任务二的理解：

利用process\_makefile来逐行读取makefile文件；

通过line[strcspn(line,”\n”)] =’\0’;来将换行换成空格；

通过if(strlen(line)==0||line[0]==’#’){  
 continue;

}来识别空行和注释并跳过读取

1. ./指令中间不要空格，直接接上要运行的可执行文件
2. 个人对阶段一的任务三的理解：

设置一个最长长度为1024的数组来规定一行的最多字数，用line\_number来记录行数，再遍历文件每一行来检查预处理后的Minimake\_claered.mk目标行是否包含冒号，命令行是否以tab键开头

1. 个人对阶段二的任务一、二的理解：

对于rule\_parser.c：在rule\_parser.h中定义了存储Makefile规则的结构体MakefileRule及相关常量。parse\_rules函数读取预处理后的文件，解析目标、依赖和命令。每行目标行中，提取目标名，分割依赖文件名并存储，再读取下一行作为命令存储。解析结果保存在全局数组rules中，rule\_count记录规则数量。

对于dependency\_checker.c：遍历已解析的规则数组，找到与传入目标匹配的规则。对该规则中的每个依赖文件，使用stat函数检查其是否存在于当前目录。若有依赖文件不存在，输出错误信息并返回0；所有依赖文件都存在则返回1，用于判断构建目标的依赖是否满足。

对于rebuild\_checker.c：通过stat函数获取目标文件的状态信息。若目标文件不存在，返回1表示需要重新构建。目标文件时，遍历其依赖规则，获取每个依赖文件的状态信息，比较依赖文件与目标文件的修改时间。只要有一个依赖文件比目标文件新，就返回1；否则返回0，以此判断目标是否需要重新构建。

对于command\_executor.c：找到与传入目标匹配的规则后，使用fork创建子进程。在子进程中，分割命令字符串为参数数组，使用execvp执行命令。若execvp执行失败，输出错误信息并退出子进程。父进程使用waitpid等待子进程结束，检查子进程的退出状态，若执行失败则输出错误信息并退出，确保命令执行的正确性和稳定性。

对于rule\_parser.h：定义了存储Makefile规则的结构体MakefileRule，包含目标名、依赖文件名数组、依赖数量和命令字符串。还声明了全局变量rules和rule\_count，以及parse\_rules函数原型，为rule\_parser.c及其他相关模块提供数据结构和函数声明，便于各模块间共享和使用。

1. 对于Git的学习：  
   （1）Git与github交互的流程：

①在github上创建一个远程仓库；

②把远程仓库和本地代码库进行绑定：

方法一：打开命令行，输入命令git clone [远程仓库链接（地址）]，若远程仓库有内容，还会把所有内容下载下来，此方法适合于本地没有代码的情况；

方法二：git remote add <远程仓库名称><远程仓库地址>把一个远程仓库绑定到本地仓库，先在本地新建一个文件夹，在文件夹中打开”Open Git Bash here”,输入命令git init来创建一个本地代码仓库，再输入git remote add<远程仓库名称（可随意取名）><远程仓库地址>

③使用git add \*把代码存到暂存区，使用git commit -m “备注”来提交代码

④把代码提交到github上：

打开命令行，执行git push origin master命令（注意，github默认分支为main，使用github当远程仓库时，请把master修改为main）

⑤登录并校验权限：

**第一种方式：**打开github--->点击自己的头像--->选择settings--->在左下角找到Developer Settings--->打开之后选择Personer access token，Fine-grained tokens--->点击Generate new token按钮--->设置令牌：设置名称、设置到期时间、调整权限（All repositories）--->点开Repository permissions--->找到contents--->修改为Read and write--->拉到最下面，选择Generate Token--->复制得到的字符，保存到本地（用来代替密码）

**第二种方式（SSH链接+密钥）**：在前边使用git remote add命令绑定远程仓库时，不使用HTTPS链接，而是在项目主页的code中找到SSH链接，之后，在本地电脑上打开Git Bash，执行命令ssh-keygen-t rsa -C加上自己的邮箱，看到冒号就按回车，看到yes or no就输入y，再按回车，看到一大串很混乱字符，就代表已经创建成功了，在命令行的输出中，找到public key一行，按路径打开其后边的文件地址，全选复制里面的内容，打开github，点击自己的头像，选择Settings，找到SSH and GPG keys选项，点击New SSH key，取名并把刚复制的字符粘贴到Key输入框中，之后生成密钥的电脑就和github绑定了，以后在这台电脑上只要是使用SSH链接，就不需要再校验权限。

1. github中的readme文件用于介绍项目；.gitignore文件在Git提交时，会忽略需要被忽略的文件或文件夹，用法为把需要忽略的文件名或文件夹名添加到.gitignore文件中；需选择开源许可证，不同的许可证会对用户有不同的道德约束以及法律约束。
2. Git的命令：

①git push（推送）：把本地代码上传到远程仓库

格式：git push <远程仓库名><本地分支名>:<远程分支名>

例如：git push origin master即把代码上传至origin仓库的master分支

若不加参数，即git push，则会把代码上传到默认仓库中和本地分支对应的分支里

②git pull（拉取）：把远程代码，下载到本地

若直接执行这条命令，即git pull，则会把默认远程仓库中对应远程分支的最新代码拉取下来

类似于git pull origin master的，则可以用来指定想要拉取的代码

1. 单人开发流程

①创建远程仓库和本地仓库并把两个仓库绑定在一起（github默认会自动创建main分支（master），此分支一般用来保存稳定的代码）

②进行开发则执行命令git checkout -b develop，切换到develop分支并编写代码

③开发完成之后，执行git add \*命令，把所有代码提交到暂存区，然后执行git commit -m “备注，如增加了一个功能”，完成一次代码提交

④执行命令git checkout master，切换回master分支，执行命令git merge develop，完成代码合并

⑤最后，执行命令git push origin master，把代码推送到github

好处：1’每次留下修改记录

2’方便随时查看修改内容

3’可以随时切回旧版本代码

4’保存多份，代码安全

1. 多人使用流程

①在一个项目中，一般会有两个长期存在的分支（保存稳定代码的master分支和保存开发代码的develop分支），一般不能在这两个分支中直接修改，而是需要基于develop分支，创建一个新的功能分支，执行命令git checkout develop,切换到develop

②执行命令git checkout -b dianzan（dianzan为一个例子，这个可任意命名），创建并切换到点赞分支

③完成代码开发后，执行git add \*和git commit -m “”命令，提交代码

④提交完成后，执行命令git checkout develop,回到develop

⑤执行git pull origin develop，把远程代码拉取到本地，如果自己在开发点赞功能期间同事向github上传了代码，此时，Git就会把他开发好的代码下载到自己的电脑上，并合并到develop分支里

⑥执行命令git merge dainzan，则刚刚开发好的点赞功能就会和此前的代码完成合并

⑦最后，执行git push origin develop，把代码上传到github上

（6）现在已经存在拥有界面的Git客户端