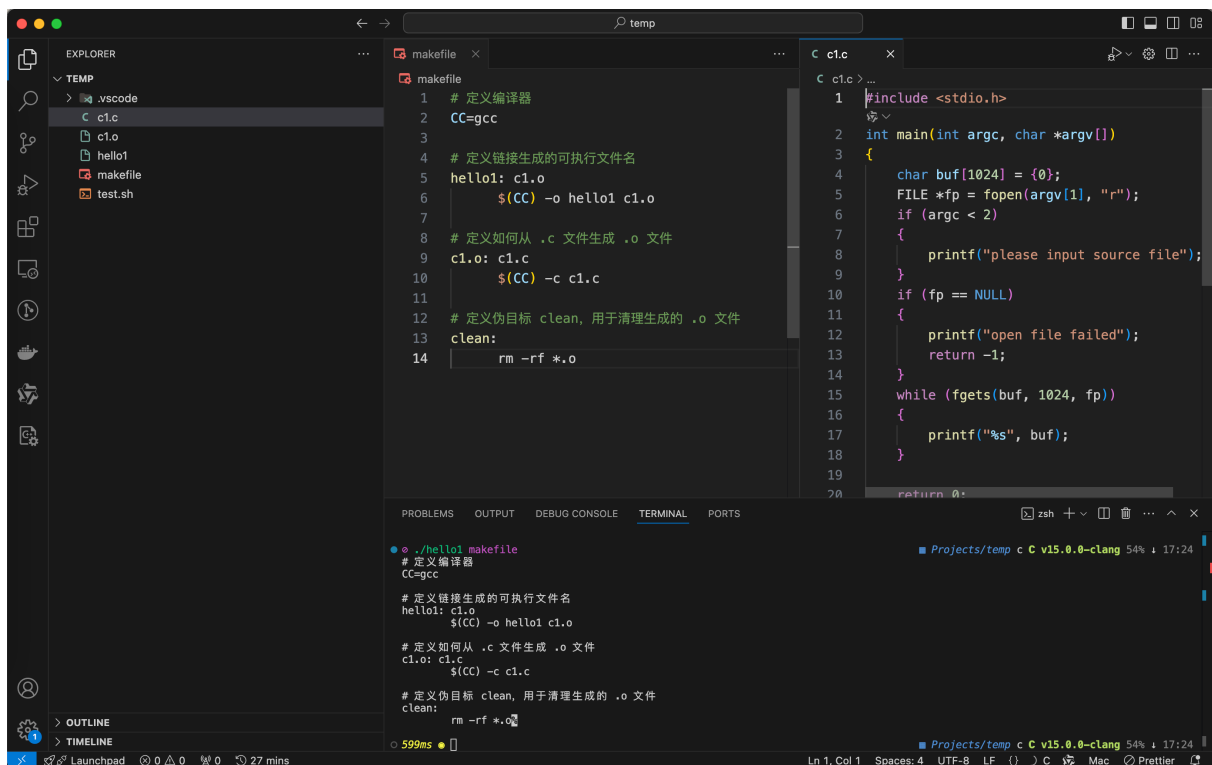


B2205106卜庆鑫实验3

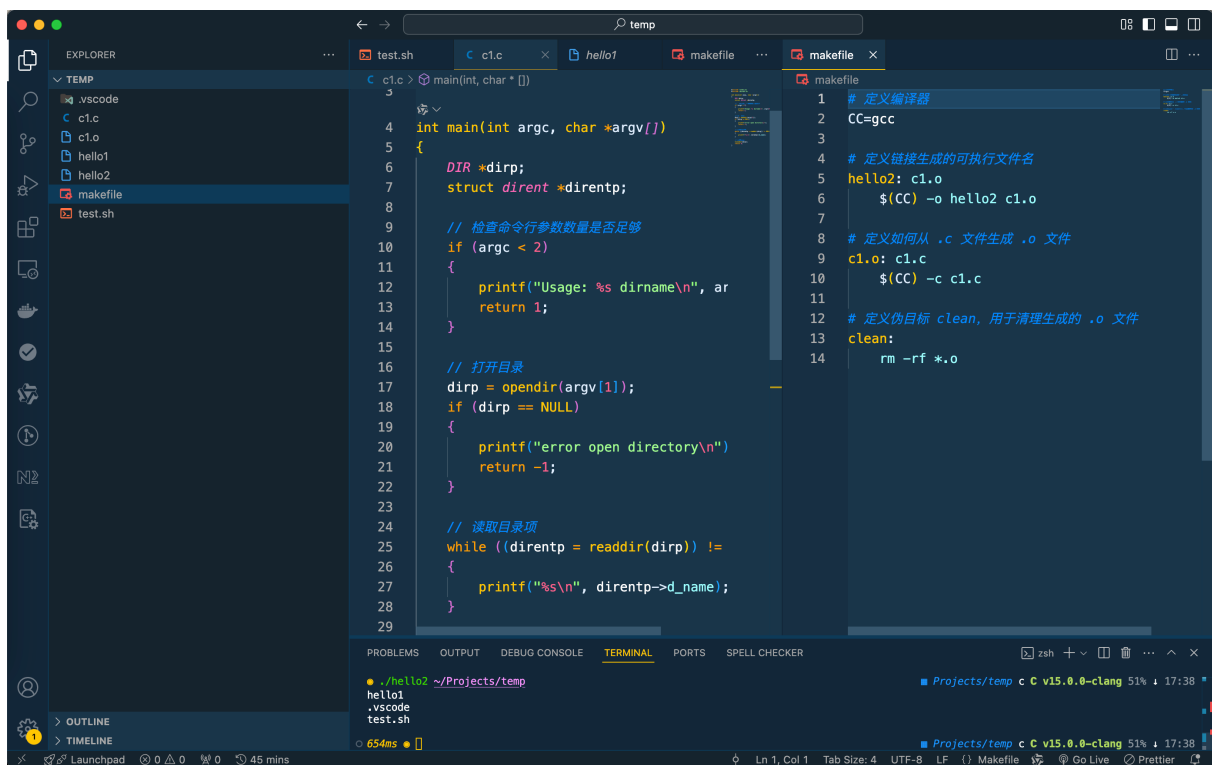


This screenshot shows the VS Code interface with a project named 'temp'. The Explorer panel on the left shows the file structure: `.vscode`, `c1.c`, `hello1`, `makefile`, and `test.sh`. The main editor displays two files: `makefile` and `c1.c`.

```
makefile
1 # 定义编译器
2 CC=gcc
3
4 # 定义链接生成的可执行文件名
5 hello1: c1.o
6     $(CC) -o hello1 c1.o
7
8 # 定义如何从 .c 文件生成 .o 文件
9 c1.o: c1.c
10    $(CC) -c c1.c
11
12 # 定义伪目标 clean, 用于清理生成的 .o 文件
13 clean:
14     rm -rf *.o
```

```
c1.c
1 #include <stdio.h>
2 int main(int argc, char *argv[])
3 {
4     char buf[1024] = {0};
5     FILE *fp = fopen(argv[1], "r");
6     if (argc < 2)
7     {
8         printf("please input source file");
9     }
10    if (fp == NULL)
11    {
12        printf("open file failed");
13        return -1;
14    }
15    while (fgets(buf, 1024, fp))
16    {
17        printf("%s", buf);
18    }
19    return 0;
20 }
```

The TERMINAL panel at the bottom shows the output of running `./hello1 makefile`, which displays the contents of the `makefile` and the compilation command `gcc -c c1.c`.



This screenshot shows the VS Code interface with a project named 'temp'. The Explorer panel on the left shows the file structure: `.vscode`, `c1.c`, `hello1`, `hello2`, `makefile`, and `test.sh`. The main editor displays two files: `test.sh` and `makefile`.

```
test.sh
1 #!/bin/bash
2
3 int main(int argc, char *argv[])
4 {
5     DIR *dirp;
6     struct dirent *direntp;
7
8     // 检查命令行参数数量是否足够
9     if (argc < 2)
10    {
11        printf("Usage: %s dirname\n", argv[0]);
12        return 1;
13    }
14
15    // 打开目录
16    dirp = opendir(argv[1]);
17    if (dirp == NULL)
18    {
19        printf("error open directory\n");
20        return -1;
21    }
22
23    // 读取目录项
24    while ((direntp = readdir(dirp)) != NULL)
25    {
26        printf("%s\n", dirent->d_name);
27    }
28
29 }
```

```
makefile
1 # 定义编译器
2 CC=gcc
3
4 # 定义链接生成的可执行文件名
5 hello2: c1.o
6     $(CC) -o hello2 c1.o
7
8 # 定义如何从 .c 文件生成 .o 文件
9 c1.o: c1.c
10    $(CC) -c c1.c
11
12 # 定义伪目标 clean, 用于清理生成的 .o 文件
13 clean:
14     rm -rf *.o
```

The TERMINAL panel at the bottom shows the output of running `./hello2 ~/Projects/temp`, which displays the contents of the `makefile` and the compilation command `gcc -c c1.c`.

The screenshot shows a code editor with two tabs: `c1.c` and `makefile`. The `c1.c` tab contains a C program that includes `<stdio.h>` and `<unistd.h>`, defines `main()`, and uses `printf` to print the current working directory. It also includes comments in Chinese. The `makefile` tab contains a Makefile with the following content:

```
1 # 定义编译器
2 CC=gcc
3
4 # 定义链接生成的可执行文件名
5 hello3: c1.o
6     $(CC) -o hello3 c1.o
7
8 # 定义如何从 .c 文件生成 .o 文件
9 c1.o: c1.c
10     $(CC) -c c1.c
11
12 # 定义伪目标 clean, 用于清理生成的 .o 文件
13 clean:
14     rm -rf *.o
```

The bottom of the editor shows a terminal window with the following output:

```
705ms ● make
gcc -o hello3 c1.o
● ● ./hello3
/Users/boqingxin/Projects/temp
success
/System/Volumes/Data/home
```

The terminal also shows the command `zsh` and the file `Projects/temp c C v15.0.0-clang 50% 17:42`.