四 川 大 学 计 算 机 学 院、软 件 学 院

实 验 报 告

学号：_____ 姓名：_____ 专业：软件工程 班级：05 第四周

课程名称	操作系统课程设计	实验课时	2
实验项目	Makefile 编写 GDB 调试	实验时间	2023/10/03
实验目的	1）Linux 下程序开发的过程 2） 了解如何编写 makefile 文件 3） 了解如何使用 GDB 调试程序		
实验环境	ARM64，MacOS, Parallels Desktop 19，Ubuntu Linux 22.04.2		

实验内容（算法、程序、步骤和方法）	**Lab1:编写 LAB03 实验中计算圆柱体的体积和表面积程序相应的 makefile，并且编译运行该程序，并将运行结果截图** 该程序包含 3 个文件，并且完成对圆柱体的表面积和体积进行计算的功能，其中每个文件包含的内容如下： ✓ 第一个文件，包含主函数，提示用户输入半径、高，并计算相应的结果。 ✓ 第二个文件，包含计算圆柱的表面积的函数 ✓ 第三个文件，包含计算圆柱的体积的函数 **Lab2:分析以下的 makefile 文件，并回答以下问题。** CC = gcc OPTIONS = -g -o OBJECTS = main.o input.o compute.o SOURCES = main.c input.c compute.c HEADERS = main.h input.h compute.h **#问题一：以上部分有什么意义** power:main.c $(OBJECTS) $(CC) $(OPTIONS) power $(OBJECTS) -lm **#问题二：上一句命令有什么意义** main.o:main.c $(HEADERS) input.o:input.c input.h compute.o:compute.c compute.h all.tar:$(SOURCES) $(HEADERS) makefile tar -cvf $(SOURCES) $(HEADERS) makefile > all.tar

#问题三：上一句命令有什么意义

```
clean:
    rm *.o
```

#问题四：如何通过 make 执行 clean 的操作(命令格式)

Lab3:使用 gdb 调试以下程序代码

```
#include <stdio.h>
#include <unistd.h>
#define ARRAY_SIZE 10
void natural_numbers (void) {
  int i;
  int array[ARRAY_SIZE];
  i = 1;
  while (i <= ARRAY_SIZE) {
       array[i] = i - 1;
       sleep(1); /* print slowly to see clearly */
       printf("array[%d] = %d\n", i, array[i]);
       i = i + 1;
  }
}
```

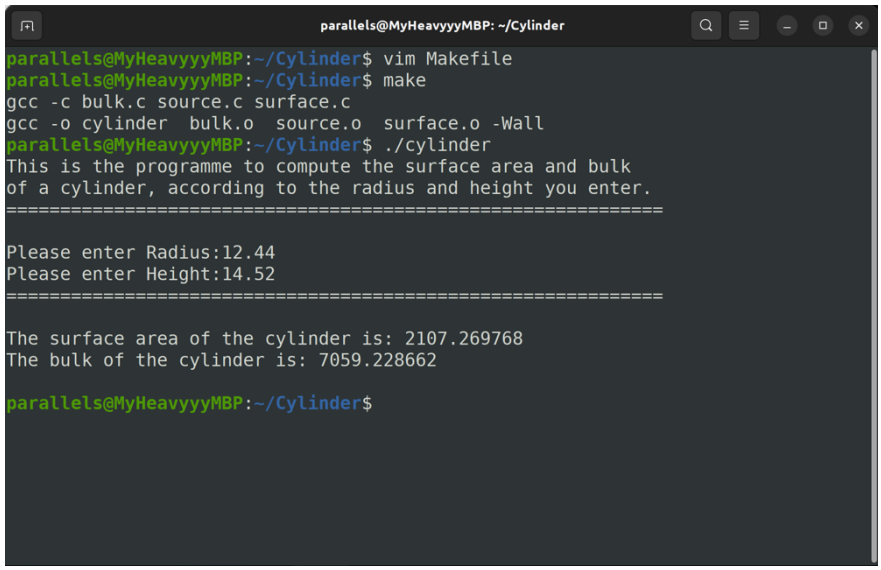
问题 1. 对出现出错信息的部分进行截图

问题 2. 修改完善源代码

	问题 3. 使用 gdb 调试修改后的代码，给出错误修改后部分的调试截图（与原图对比） 问题 4. 说明程序出现错误的原因以及解决方法

实验结果的截图

Lab1 --- makefile 文件内容和运行结果截图



Lab2 --- 回答实验要求中的问题

问题一：

这部分定义了 Makefile 文件的变量：

1） CC：编译器，使用 gcc；
2） OPTIONS：编译选项，-g 生成可执行文件时包含调试信息，-o 指定生成可执行文件的名称；

3） OBJECTS：目标文件；
4） SOURCES：源文件；
5） HEADERS：头文件。

问题二：

1） `power` 是目标，即要生成的可执行文件的名称；
2） `main.c $(OBJECTS)` 是依赖项，即用来输入产生目标的文件，包含 main.c 和 OBJECTS 中定义的所有目标文件；
3） `$(CC) $(OPTIONS) power $(OBJECTS)` 是生成目标文件的命令。使用 CC 定义的编译器（gcc），使用 OPTIONS 中定义的编译选项（-g -o），编译 main.c 和 OBJECTS 中的所有目标文件，输出名为 power 的目标文件；
4） `-lm` 表示在链接过程中将数学库链接到生成的可执行文件中（-l 链接选项前缀， m 表示数学库）。

问题三：

1） `all.tar` 是目标，即要生成的可执行文件的名称；
2） `$(SOURCES) $(HEADERS) makefile` 是依赖项，包含所有的源文件（SOURCES）、所有的头文件（HEADERS）和 Makefile 文件本身
3） `tar -cvf $(SOURCES) $(HEADERS) makefile > all.tar` 是用于创建名为 all.tar 的 tar 归档文件的命令，并将所有的源文件（SOURCES）、所有的头文件（HEADERS）和 Makefile 文件本身，都添加到这个归档文档中。其中：

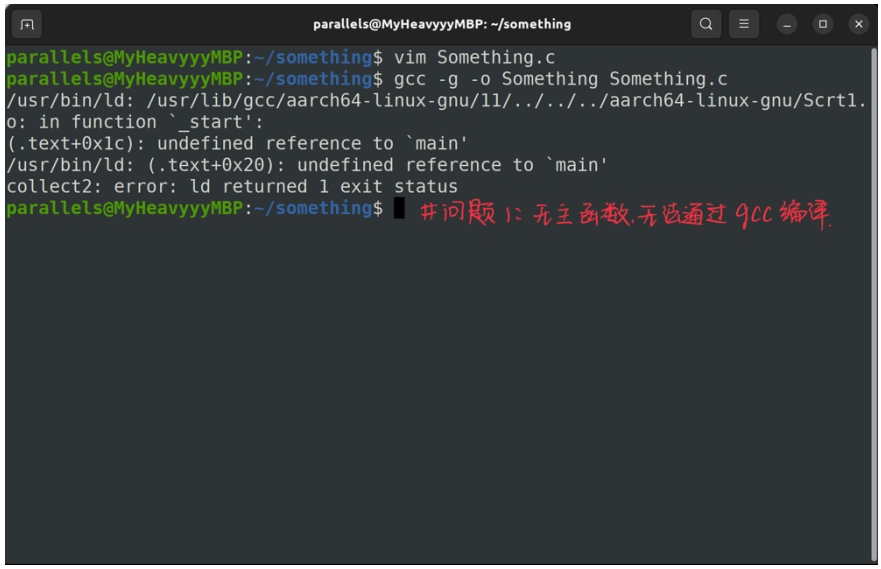
   a) `tar` 是用于创建和管理归档文件的命令行工具
   b) `-cvf` 是 tar 命令的选项和参数的组合。`-c` 表示创建一个新的归档文件，`-v` 表示在创建过程中显示详细的信息，`-f` 表后面会紧跟着创建的归档文件的名称
   c) `$(SOURCES) $(HEADERS) makefile` 是要归档的文件列表
   d) `> all.tar` 表示将`tar`命令的输出结果重定向到名为`all.ta`的文件中，而不是显示在终端上。如果`all.tar`不存在，会创建并包含`tar`命令所列出的文件

问题四：

命令格式为 `make clean`

Lab3 --- 回答实验要求中的问题

问题一：

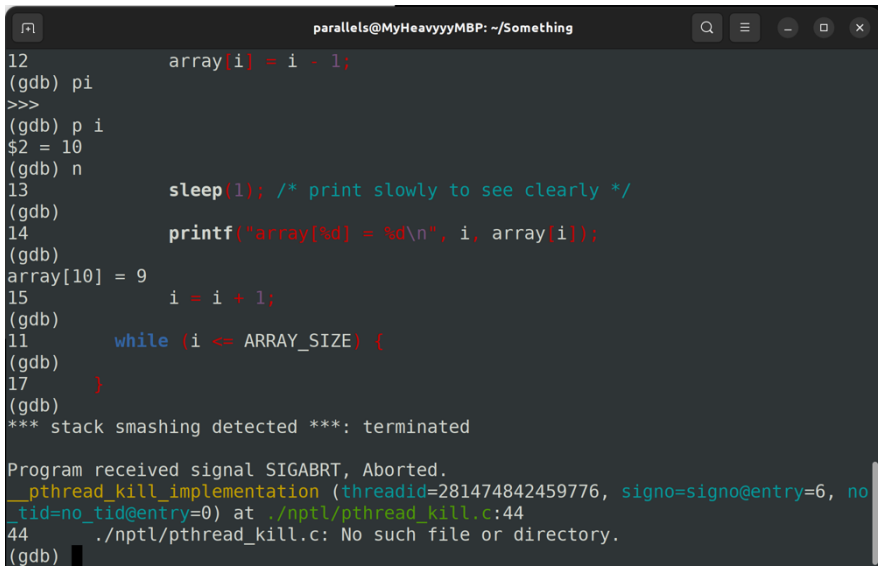


问题二：


parallels@MyHeavyyyMBP: ~/Something
#include <stdio.h>
#include <unistd.h>
#define ARRAY_SIZE 10
void main(void) {
int i;
int array[ARRAY_SIZE];
i = 1;
while (i < ARRAY_SIZE) {
array[i] = i - 1;
sleep(1); /* print slowly to see clearly */
printf("array[%d] = %d\n", i, array[i]);
i = i + 1;
}
}
程序无主函数，将函数名修改为 main 以通过 gcc 编译
数组越界造成栈冲突！
将 "<=" 修改为 "<" 防止溢出。
"Something.c" 18L, 301B
11,12
All

问题三：

修改前 / 修改后：



问题四：

问题：

1）程序无主函数，无法通过 gcc 编译；
2）数组越界造成栈冲突。

解决方案：

1）将程序函数名修改为 `main`；

	2）将 while 循环条件从 `i <= MAX_SIZE` 修改为 `I < MAX_SIZE`，防止数组越界。
小　结	通过本实验，我掌握了： 1）Linux 下程序开发的过程 2）Makefile 文件的编写 3）GDB 调试程序的使用 目前我存在的问题有： 1）对 Makefile 文件高级命令的掌握不深 2）对 GDB 调试程序的使用不熟练
指导老师评　议	成绩评定：　　指导教师签名：