

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 | | |

| | |
|-------------------------------|--|
| <p>实验内容 (算法、程序、步骤和方法)</p> | <p>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</p> <p>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:</p> <ul style="list-style-type: none"> ✓ 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 <p>Lab2:分析以下的 makefile 文件, 并回答以下问题。</p> <pre>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</pre> <p>#问题一：以上部分有什么意义</p> <pre>power:main.c \$(OBJECTS) \$(CC) \$(OPTIONS) power \$(OBJECTS) -lm</pre> <p>#问题二：上一句命令有什么意义</p> <pre>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</pre> |
|-------------------------------|--|

#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

| | |
|--|--|
| | <p>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）</p> |
|--|--|

问题 4. 说明程序出现错误的原因以及解决方法

Lab1 --- makefile 文件内容和运行结果截图

实验结果
的截图

```
parallels@MyHeavyyyyMBP: ~/Cylinder
1 CC = gcc
2
3 TAR = cylinder
4 SRC = $(wildcard *.c)
5 OBJ = $(patsubst %.c, %.o, $(SRC))
6
7 $(TAR): $(OBJ)
8 | $(CC) -o $(TAR) $(OBJ) -Wall
9
10 $(OBJ): $(SRC)
11 | $(CC) -c $(SRC)
12
13 all:
14 | make $(TAR)
15
16 .PHONY:clean
17 clean:
18 | rm -f *.o
19 | rm -f $(TAR)
20
21
"Makefile" 21L, 228B 1,1 All
```

```
parallels@MyHeavyyyyMBP: ~/Cylinder$ vim Makefile
parallels@MyHeavyyyyMBP:~/Cylinder$ make
gcc -c bulk.c source.c surface.c
gcc -o cylinder bulk.o source.o surface.o -Wall
parallels@MyHeavyyyyMBP:~/Cylinder$ ./cylinder
This is the programme to compute the surface area and bulk
of a cylinder, according to the radius and height you enter.
=====
Please enter Radius:12.44
Please enter Height:14.52
=====
The surface area of the cylinder is: 2107.269768
The bulk of the cylinder is: 7059.228662
parallels@MyHeavyyyyMBP:~/Cylinder$
```

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) $(HEADERS) makefile > all.tar` 是用于创建名为 `all.tar` 的 `tar` 归档文件的命令, 并将所有的源文件 (`SOURCES`)、所有的头文件 (`HEADERS`) 和 `Makefile` 文件本身, 都添加到这个归档文档中。其中:
 - a) `tar` 是用于创建和管理归档文件的命令行工具
 - b) `-cvf` 是 `tar` 命令的选项和参数的组合。`-c` 表示创建一个新的归档文件, `-v` 表示在创建过程中显示详细的信息, `-f` 表示后面会紧跟着创建的归档文件的名称
 - c) `$(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:

```
parallels@MyHeavyyyyMBP: ~/something
parallels@MyHeavyyyyMBP:~/something$ vim Something.c
parallels@MyHeavyyyyMBP:~/something$ gcc -g -o Something Something.c
/usr/bin/ld: /usr/lib/gcc/aarch64-linux-gnu/11/../../../../aarch64-linux-gnu/Scrt1.o: in function `_start':
(.text+0x1c): undefined reference to `main'
/usr/bin/ld: (.text+0x20): undefined reference to `main'
collect2: error: ld returned 1 exit status
parallels@MyHeavyyyyMBP:~/something$ #问题1: 无主函数,无法通过gcc编译.
```

```
parallels@MyHeavyyyyMBP: ~/Something
12         array[i] = i - 1;
(gdb) pi
>>>
(gdb) p i
$2 = 10
(gdb) n
13         sleep(1); /* print slowly to see clearly */
(gdb)
14         printf("array[%d] = %d\n", i, array[i]);
(gdb)
array[10] = 9
15         i = i + 1;
(gdb)
11         while (i <= ARRAY_SIZE) {
(gdb)
17     }
(gdb)
*** stack smashing detected ***: terminated #问题2: 数组越界造成栈冲突
Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (threadid=281474842459776, signo=signo@entry=6, no_tid=no_tid@entry=0) at ./nptl/pthread_kill.c:44
44     ./nptl/pthread_kill.c: No such file or directory.
(gdb)
```

问题二:

```

1 #include <stdio.h>
2 #include <unistd.h>
3
4 #define ARRAY_SIZE 10
5
6 void main(void) { 程序无主函数. 将函数名修改为 main 以通过 gcc 编译
7     int i;
8     int array[ARRAY_SIZE];
9     i = 1;
10
11     while (i <= ARRAY_SIZE) { 数组越界造成栈冲突!
12         array[i] = i - 1; 将 "<=" 修改为 "<" 防止溢出.
13         sleep(1); /* print slowly to see clearly */
14         printf("array[%d] = %d\n", i, array[i]);
15         i = i + 1;
16     }
17 }
18

```

"Something.c" 18L, 301B 11,12 All

问题三:

修改前 / 修改后:


```

parallels@MyHeavyvyyMBP: ~/Something
12      array[i] = i - 1;
(gdb) pi
>>>
(gdb) p i
$2 = 10
(gdb) n
13      sleep(1); /* print slowly to see clearly */
(gdb)
14      printf("array[%d] = %d\n", i, array[i]);
(gdb)
array[10] = 9
15      i = i + 1;
(gdb)
11      while (i <= ARRAY_SIZE) {
(gdb)
17      }
(gdb)
*** stack smashing detected ***: terminated

Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (threadid=281474842459776, signo=signo@entry=6, no
_tid=no_tid@entry=0) at ./nptl/pthread_kill.c:44
44      ./nptl/pthread_kill.c: No such file or directory.
(gdb)

```

```

parallels@MyHeavyvyyMBP: ~/Something
(gdb)
Continuing.

array[6] = 5

Breakpoint 1, main () at Something.c:12
12      array[i] = i - 1;
(gdb)
Continuing.
array[7] = 6

Breakpoint 1, main () at Something.c:12
12      array[i] = i - 1;
(gdb)
Continuing.
array[8] = 7

Breakpoint 1, main () at Something.c:12
12      array[i] = i - 1;
(gdb)
Continuing.
array[9] = 8
[Inferior 1 (process 58524) exited with code 0130]
(gdb)

```

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``；

| | |
|-------------|--|
| | <p>2) 将 while 循环条件从 <code>i <= MAX_SIZE`</code> 修改为 <code>I < MAX_SIZE`</code>, 防止数组越界。</p> |
| 小 结 | <p>通过本实验，我掌握了：</p> <ol style="list-style-type: none"> 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 <p>目前我存在的问题有：</p> <ol style="list-style-type: none"> 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 | <p>成绩评定：</p> <p>指导教师签名：</p> |