

实验二

Part1 GCC使用

首先用vim，编写了一个简单的C程序：

[illegible]

进行了GCC的各种使用:

1. GCC -E: 进行预处理, 将源文件转换为.i文件:

```

# 0 "main.c"
# 0 "<built-in>"
# 0 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 0 "<command-line>" 2
# 1 "main.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 1 3 4
# 33 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 392 "/usr/include/features.h" 3 4
# 1 "/usr/include/features-time64.h" 1 3 4
# 20 "/usr/include/features-time64.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 21 "/usr/include/features-time64.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 1 3 4
# 19 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 20 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 2 3 4
# 22 "/usr/include/features-time64.h" 2 3 4
# 393 "/usr/include/features.h" 2 3 4
# 486 "/usr/include/features.h" 3 4
--More--(5%)

```

2. GCC -S: 进行编译, 将源文件转换为汇编文件:

```
baijy@baijy-virtual-machine: ~/Desktop/exercise/e2
.file "main.c"
.text
.section .rodata
.LC0:
.string "this is a test "
.text
.globl main
.type main, @function
main:
.LFB0:
.cfi_startproc
endbr64
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
leaq .LC0(%rip), %rax
movq %rax, %rdi
call puts@PLT
movl $0, %eax
popq %rbp
.cfi_def_cfa 7, 8
--More--(52%)
```

3. 以及 -c, -o, -g 等选项的使用:

```
baijy@baijy-virtual-machine: ~/Desktop/exercise/e2$ ls
main main.c main.i main.o main.s power.c power.h test test.c
baijy@baijy-virtual-machine: ~/Desktop/exercise/e2$
```

Part2 makefile的编写

主函数 test.c:

```
#include<stdio.h>
#include"power.h"
int main(){
    printf("hello world!\n");
    int a=5;
    printf("%d",power(a));
    return 0;}
```

"test.c" 7L, 121B

6,21-28

All

power.h:

```
#include<stdio.h>
int power(int a);
```

power.c:

```
#include<stdio.h>
#include"power.h"
int power(int a){
    return a*a;}
```

定义编译器和编译选项

CC = gcc

```
CFLAGS = -g -Wall
```

定义目标文件和可执行文件

TARGET = myprogram

OBJS = test.o power.o

默认目标（第一个目标）为可执行文件

```
all: $(TARGET)
```

生成可执行文件的规则

$\$(TARGET) : \(OBS)

```
$ (CC) $(CFLAGS) -o $(TARGET) $(OBJS)
```

生成目标文件的规则

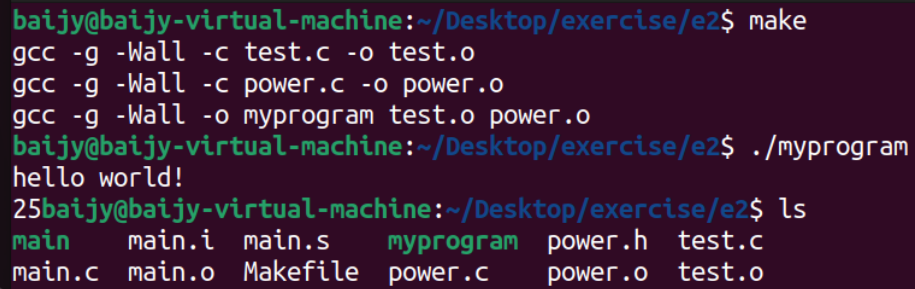
```
test.o: test.c power.h
```

```
$(CC) $(CFLAGS) -c test.c -o test.o
```

```
power.o: power.c power.h
    $(CC) $(CFLAGS) -c power.c -o power.o

# 清理规则
clean:
    rm -f $(OBJS) $(TARGET)
```

输出结果:



```
baijy@baijy-virtual-machine:~/Desktop/exercise/e2$ make
gcc -g -Wall -c test.c -o test.o
gcc -g -Wall -c power.c -o power.o
gcc -g -Wall -o myprogram test.o power.o
baijy@baijy-virtual-machine:~/Desktop/exercise/e2$ ./myprogram
hello world!
25baijy@baijy-virtual-machine:~/Desktop/exercise/e2$ ls
main  main.i  main.s  myprogram  power.h  test.c
main.c  main.o  Makefile  power.c  power.o  test.o
```