实验一 Linux下C编程

**一、实验目的**

1、熟悉Linux常用命令。

2、掌握gcc编译器、gdb调试器的使用方法。

3、会编写多文件的C程序，并使用gcc编译。

4、熟悉makefile的结构和编制，利用make命令编译程序。

**二、实验原理**

1、linux不以文件名后缀区分文件类型，但是通常将C源代码以.c结尾，事实证明，在使用GCC编译器时，是要求以.c结尾的。

2、在linux下，编写好的C程序需要编译为二进制的可执行文件才能运行；得到可执行文件要经过以下几个步骤：

预处理🡪汇编🡪编译🡪链接🡪二进制可执行文件。

3、make工具，是程序的自动编译器。利用时间戳检测文件修改的时间，查看当前哪些文件是更新的，只编译改变了的程序，是一个项目管理工具。用于大型程序的编制，可节省时间，提高效率。

Makefile的基本格式：

（1）目标体：目标文件/可执行文件。

（2）依赖文件。

（3）运行的命令（必须以Tab开头）。

Makefile具有隐含规则和模式规则。隐含规则：同名的.c文件生成同名的.o文件，不用写main.o、print.o等的编译语句。模式规则：相同名称的.c生成相同名称的.o文件，这种方式更加显式。

**三、实验内容**

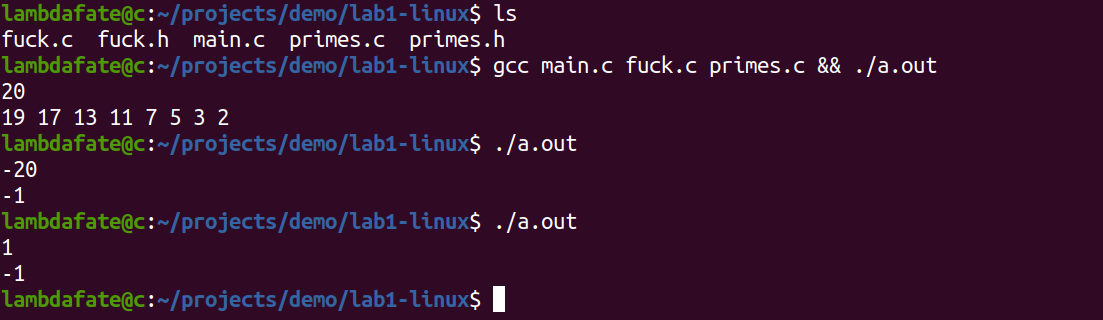
1、编写包含多文件的.c源码，通过调用自定义函数，实现功能：用户输入一个数字，程序计算并输出介于1至此数之间所有个位数为1的素数；如果不存在，则打印-1。

2、直接使用gcc编译多个源文件并运行结果。

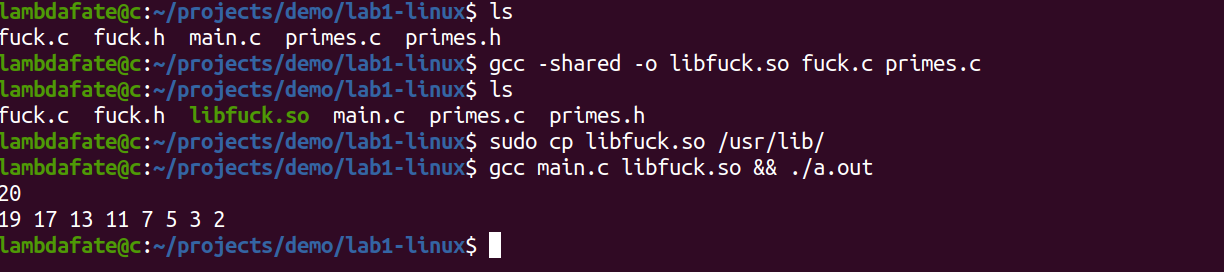
3、通过创建动态链接库lib\*\*\*.so，使得 main 函数调用自定义函数时，可使用动态链接库，编译生成运行结果。

4、编写Makefile文件，使用make编译并运行。

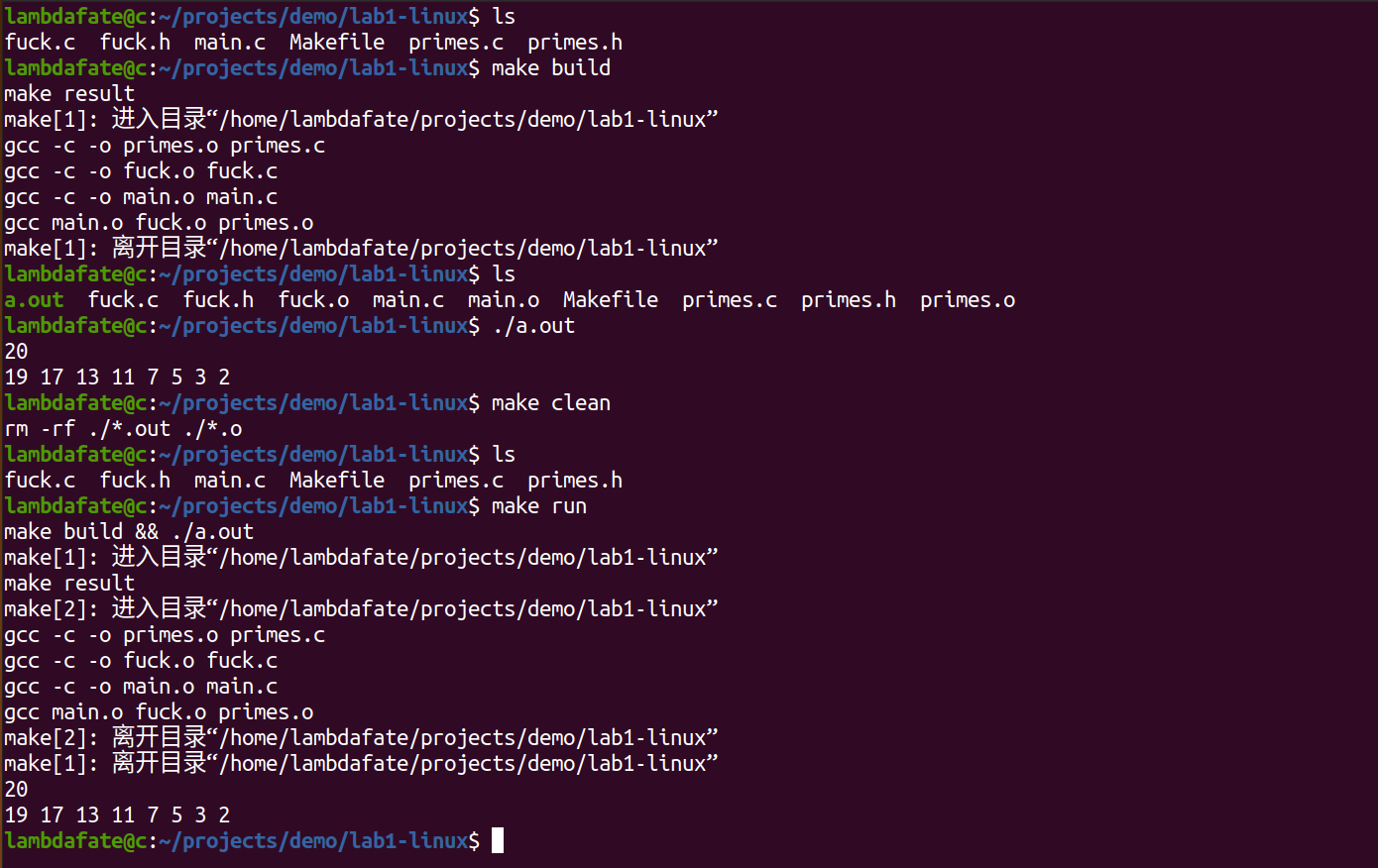
使用GCC直接编译多个源文件查看结果



通过创建动态链接库lib\*\*\*.so，使得 main 函数调用自定义函数时，可使用动态链接库，编译生成运行结果。



编写Makefile文件，使用make编译并运行。



四 程序代码

Main.c

#include<stdio.h>

#include"fuck.h"

int main(int argc, char \*argv[]){

fuck();

return 0;

}

Fuck.h 和 fuck.c

#ifndef \_FUCK\_H

#define \_FUCK\_H

void fuck();

#endif

#include<stdio.h>

#include"primes.h"

void fuck(){

int num = 0;

scanf("%d", &num);

if(num < 2){

printf("%d\n", -1);

return;

}

int count = print\_primes(num);

if(count == 0){

printf("%d\n", -1);

}

}

Primes.h 和 primes.c

#ifndef \_PRIMES\_H

#define \_PRIMES\_H

unsigned int print\_primes(int num);

#endif

#include<stdio.h>

int check\_prime(int num){

if(num < 2){

return 0;

}

for (int i = num-1; i > 1; i--){

if(num % i == 0){

return 0;

}

}

return 1;

}

unsigned int print\_primes(int num){

if(num < 2){

return 0;

}

unsigned int count = 0;

while(num > 1){

if(check\_prime(num)){

printf("%d ", num);

count++;

}

num--;

}

if(count){

printf("\n");

}

return count;

}

Makefile

primes.o: primes.c

gcc -c -o primes.o primes.c

fuck.o: fuck.c

gcc -c -o fuck.o fuck.c

main.o: main.c

gcc -c -o main.o main.c

OBJECTS = main.o fuck.o primes.o

result: $(OBJECTS)

gcc $(OBJECTS)

.PHONY: build, run, clean

build:

make result

run:

make build && ./a.out

clean:

rm -rf ./\*.out ./\*.o