# 实验一：Makefile实验

## 实验目的

1. 了解make
2. 了解makefile
3. 掌握makefile编程

## 实验仪器

PC机，飞凌6410开发板

## 实验内容

1. Make

Make是大型程序的维护工具。

1. makefile

当运行make命令时，make命令会在当前目录下按顺序寻找文件名为“GUNmakefile"、"Makefile"、"makefile"的文件，找到后解释这些文件。所以说make是一个解释makefile中指令的命令工具。Makefile或makefile:告诉make维护一个大型程序，该做什么。Makefile说明了组成程序的各模块间的相互关系及更新模块时必须进行的动作，make按照这些说明自动地维护这些模块。简单的说makefile就像批处理的脚本文件，里边写好了一些命令的集合，当运行make命令时，便会按着makefile提供的命令及顺序来完成编译。Makefile文件包含了五部分内容：显示规则、隐式规则、变量定义、文件指示和注释。

1. Mkefile中的变量

Makefile里的变量就像一个环境变量。事实上，环境变量在make中也被解释成make的变量。这些变量对大小写敏感，一般使用大写宇母。几乎可以从任何地方引用定义的变量。Makefile中的变量是用一个文本串在Makefile中定义的，这个文本串就是变量的值。只要在一行的开始写下这个变量的名字，后面跟一个“二”号，以及要设定这个变量的值即可定义变量。

1. Make工作时的执行步骤
2. 读入所有的makefile文件
3. 读入被include包括的其他的makefile文件
4. 初始化文件中的变量
5. 推到隐式规则，并分析所有规则
6. 为所有的目标文件创建依赖关键链
7. 根据依赖关系，决定哪些目标要重新生成
8. 执行生成命令

## 实验程序

使用Makefile对多个程序进行本机编译：

实验代码如下：

code.c

*#include*"code.h"

*#include*<stdio.h>

void circle(float *r*)

{

    printf(format\_circle,2\*PI\*r);

}

prog.c

*#include*"prog.h"

*#include*"code.h"

*#include*<stdio.h>

area(float *r*)

{

    printf("r=%f\n",*r*);

    printf(format\_area,PI\**r*\**r*);

    printf("thank yoi\n");

}

test.c

*#include*<stdio.h>

*extern* area(float);

*extern* circle(float);

main()

{

    printf("make file test start\n");

    area(2.5);

    circle(2.5);

*return* 0;

}

code.h

*#define* PI 3.141926

*#define* format\_circle "circle=%f\n"

prog.h

*#define* format\_area "area=%f\n"

Makefile

#CC=/usr/local/arm/4.3.2/bin/arm-linux-gcc

CC=gcc

test: prog.o code.o main.o

*$*(CC) -o test prog.o code.o main.o

main.o: test.c

*$*(CC) -o main.o -c test.c

prog.o: prog.c prog.h code.h

*$*(CC) -c prog.c -o prog.o

code.o: code.c code.h

*$*(CC) -c code.c -o code.o

*clean*:

    rm -f \*.O

## 实验步骤

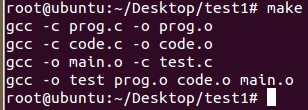
1. 本机编译

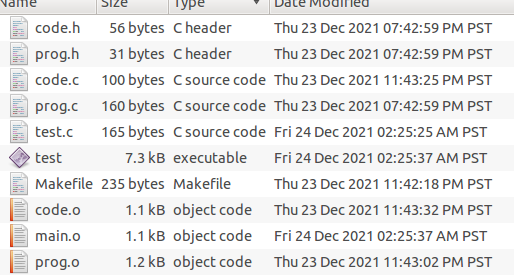
本机编译需要使用本机编译器，修改Makefile文件：也就是将CC改成gcc” CC=gcc”,将交叉编译器屏蔽掉（用#屏蔽）

#CC=/usr/local/arm/4.2.2-eabi/usr/bin/arm-linux-gcc

CC=gcc

保存退出





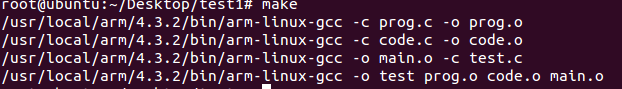
1. 交叉编译

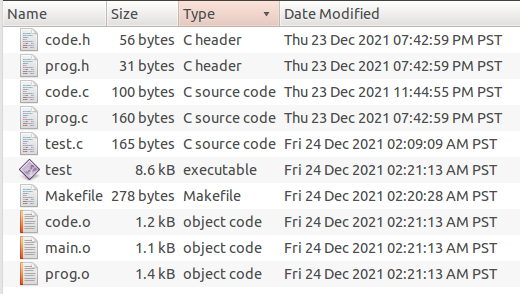
修改Makefile文件，屏蔽掉Makefile文件中的#CC=gcc （用#屏蔽），这样使编译器指向arm-linux-gcc

CC=/usr/local/arm/4.3.2/bin/arm-linux-gcc

#CC=gcc

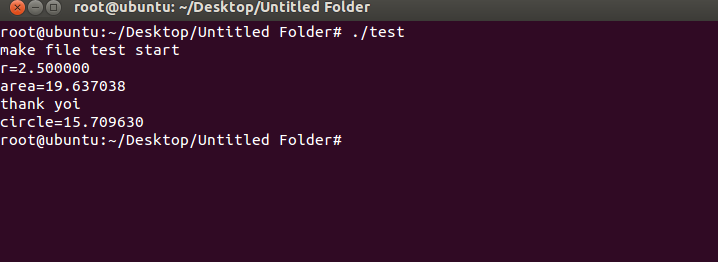
保存退出



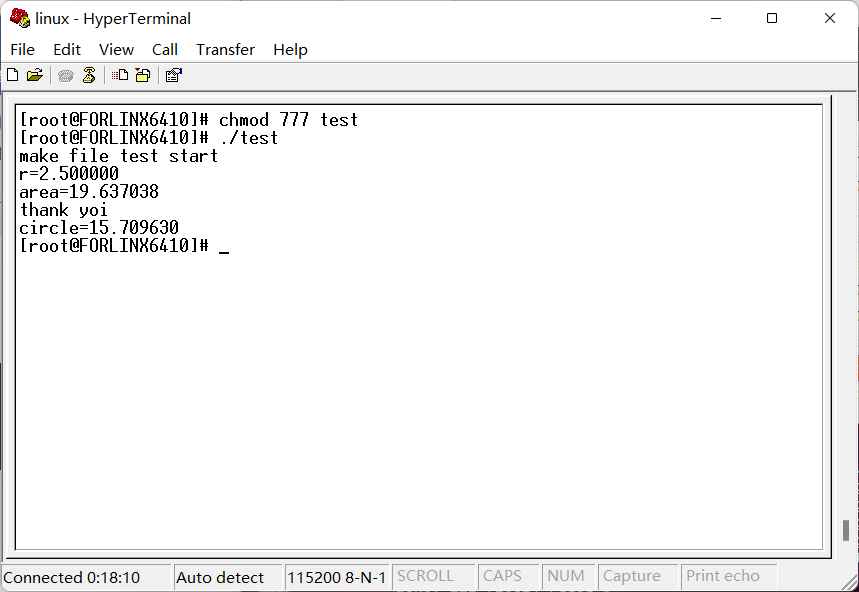


## 运行结果

1. 本机编译



1. 交叉编译



## 实验心得

通过本次实验我了解了Makefile文件的编写，本地编译与交叉编译的区别。使用make命令不需要gcc一条一条的编译,方便操作.