

**软件与微电子学院**

**集成电路与智能系统系**

**《嵌入式软件开发技术与工具》**

**实验报告**

实验题目： 构建主机端开发环境（上）

小组名称： MGH小组

小组成员： 孟昭泰 龚周闯宇 贺居獒

日 期： 2020.03.14

**原 创 声 明**

本人郑重声明：

所呈交的实验报告，是本人根据实验指导材料的要求，独立进行实验所取得的成果。本文不含任何其他人的实验成果。如果存在抄袭他人成果之情形，一切后果由本人承担。

**一、实验目的**

在建立交叉编译开发平台之前，首先需要建立主 机(虚拟机/PC，或双系统）开发环境；

学会使用本地gcc编译应用程序； 

学会使用Makefile管理应用程序； 

学会通过autotools生成Makefile，学会常用的 make操作； 

学会通过git/github管理团队软件和工作文件

**二、实验内容**

安装主机(虚拟机/PC)Linux开发环境，Fedora，Ubuntu ， Debian均可； 

编写c应用程序，通过本地gcc编译应用程序，如果是基于 x86的主机，gcc输出的执行文件运行的是x86指令集； 

编写Makefile管理应用程序，为更好体现Makefile的作用， 需编写多个c程序，给出所创建的Makefile的内容； 

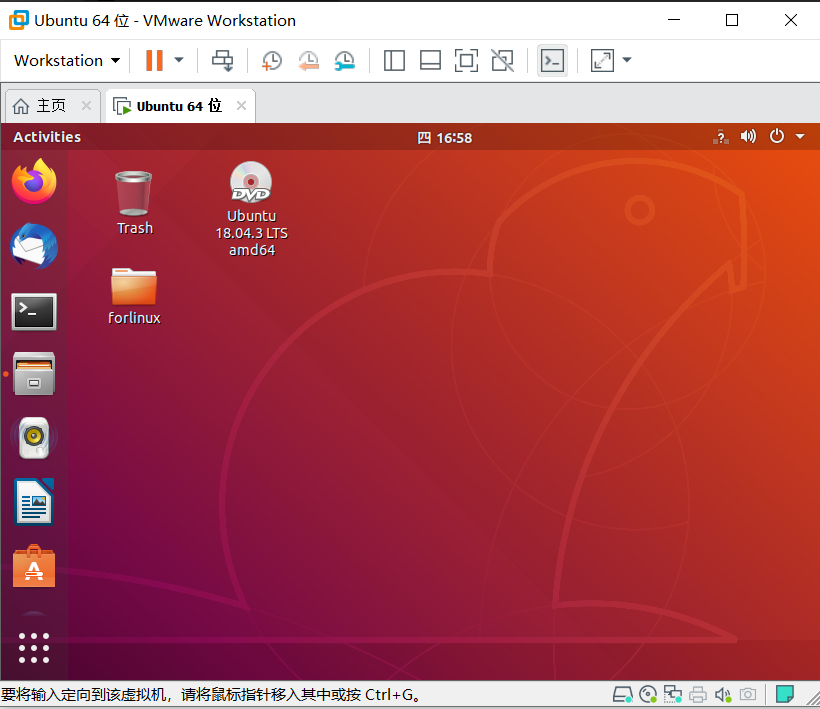
通过autotools生成Makefile，完成常用的make操作(make, make install, make uninstall, make dist)； 

创建小组git仓库，github账号，用来存储小组工作文件以 及小组报告；学习如何构建github文件，如何上传和下载 github文件等

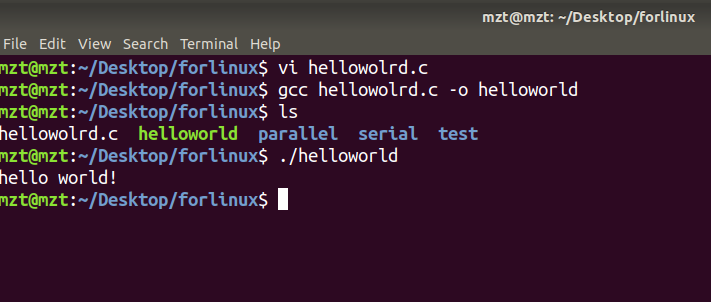
**三、实验过程与结果**

1. **安装linux开发环境**

使用虚拟机Vmware Workstation，Ubuntu 64位，版本：18.04



1. **学会使用本地gcc编译应用程序**
2. 下载gcc： sudo apt-get install gcc
3. 使用编译。运行简单的helloworld程序：gcc helloworld.c -o helloworld

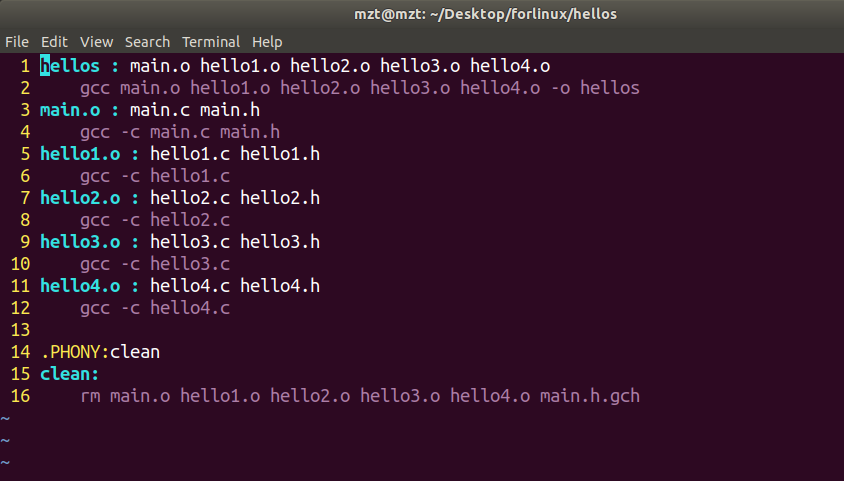


1. **学会使用Makefile管理应用程序**

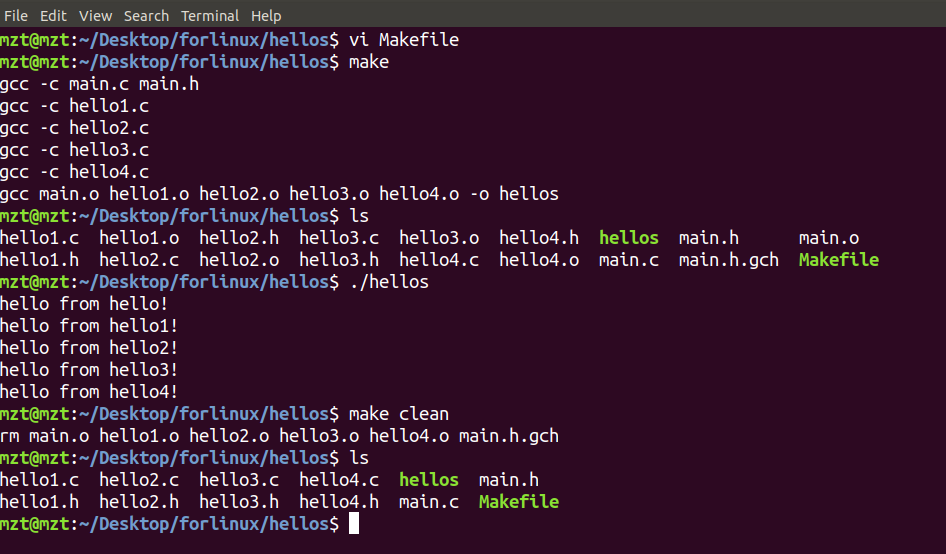
一组程序：有hello1.c,hello2.c,hello3.c,hello4.c4个子程序，功能均为打印一句helloworld。由主程序main.c调用这4个子程序。各程序文件具体内容见（五、附实验源码）

先编写Makefile实现这组程序文件的编译：

Makefile内容：



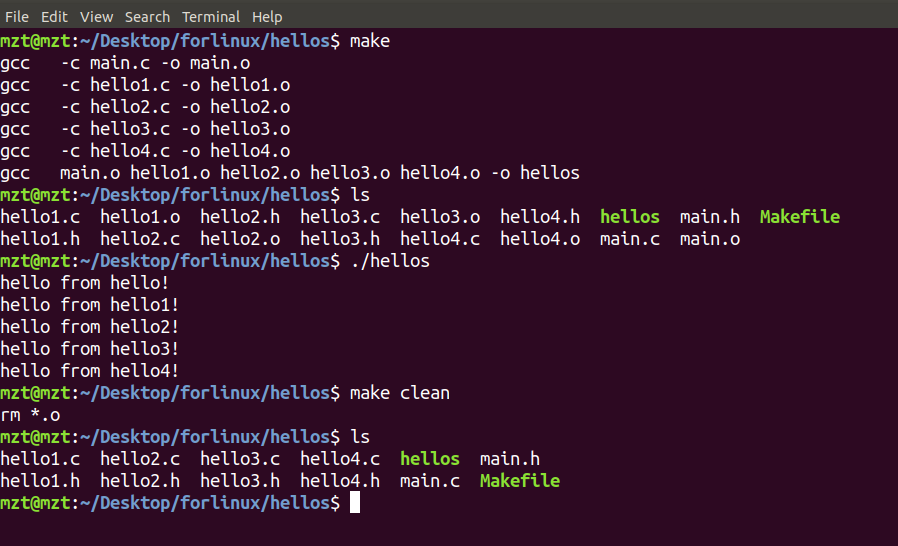
运行结果：



尝试编写更加简洁的Makefile：



运行结果：



1. **通过autotools生成Makefile**

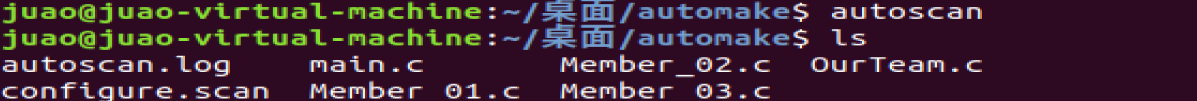
完成常用的make操作(make, make install, make uninstall, make dist)；

一组程序：main.c OurTeam.c Member\_01.c Member\_02.c ,Member\_03.c

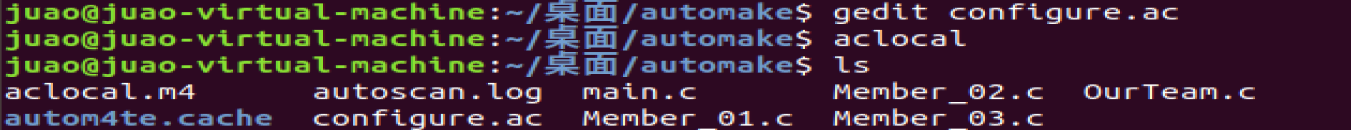
主函数功能为调用4个子程序，4个子程序功能分别为打印队伍名称、打印成员（1.2.3）名字

先基于该组程序完成通过autotools生成Makefile

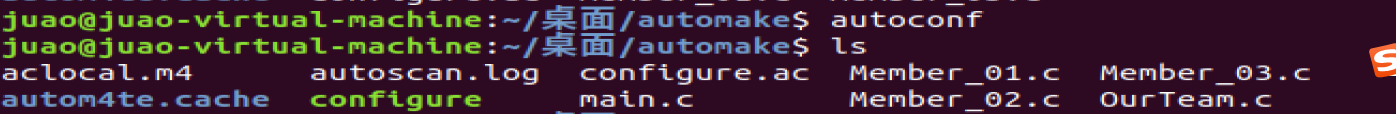
自动扫描目录，运行autoscan命令扫描，生成 configure.scan 文件：



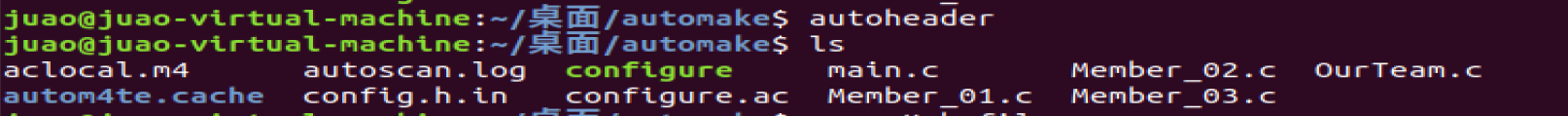
创建configure.ac文件，将configure.scan重命名为configure.ac；使用aclocal命令扫描 configure.ac文件生成 aclocal.m4文件：



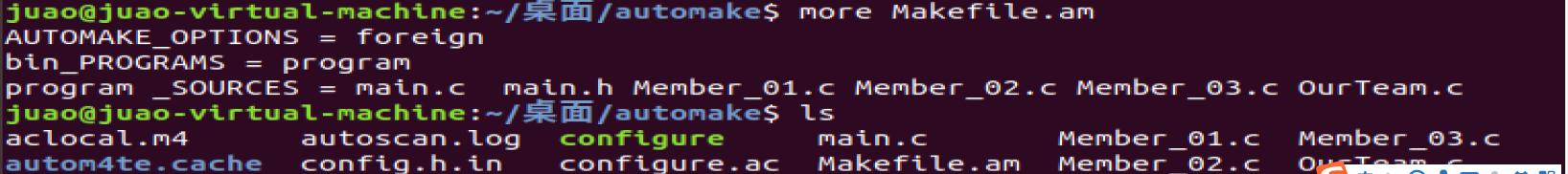
使用autoconf命令读取 configure.ac文件中的宏，生成configure脚本：



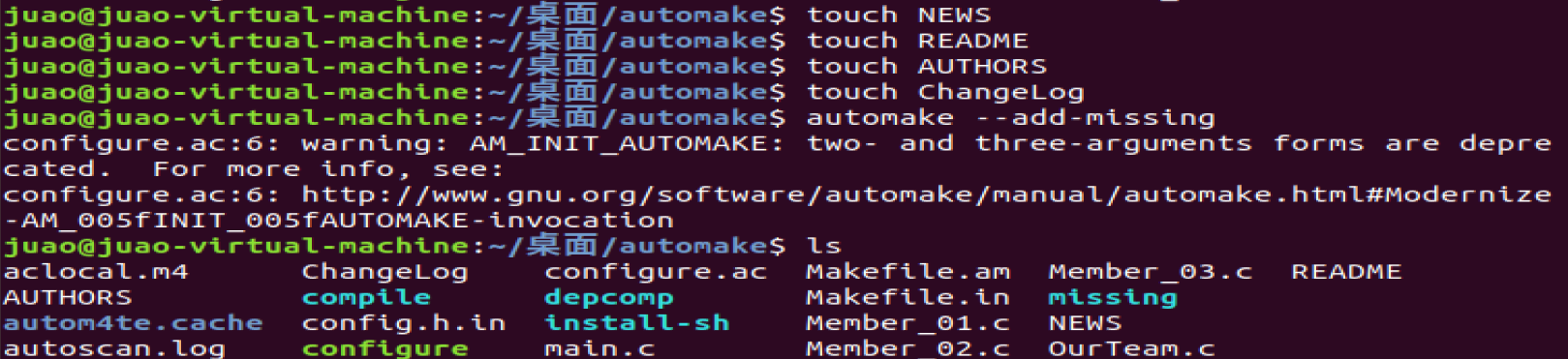
使用autoheader命令生成config.h.in文件：



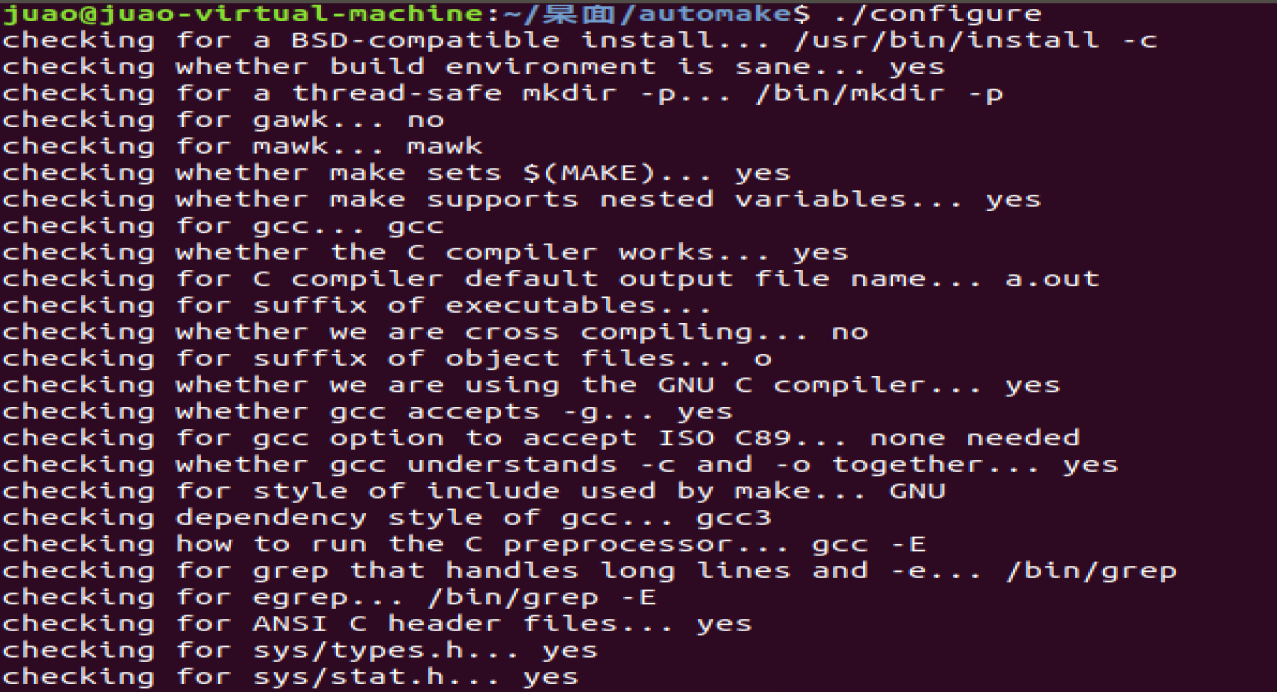
手工创建Makefile.am文件：



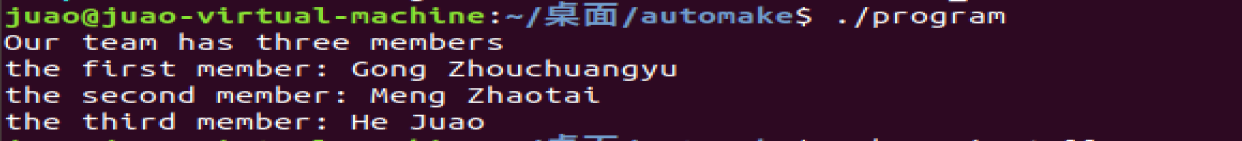
使用automake命令生成Makefile.in文件：



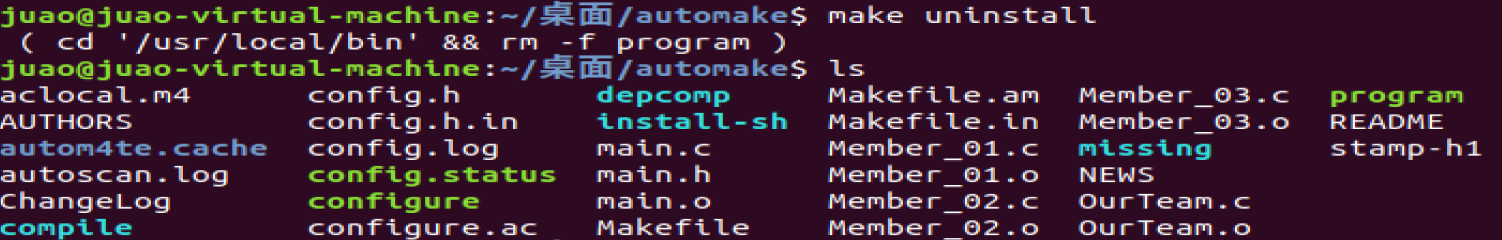
使用 configure 命令把 Makefile.in变成最终的Makefile文件：



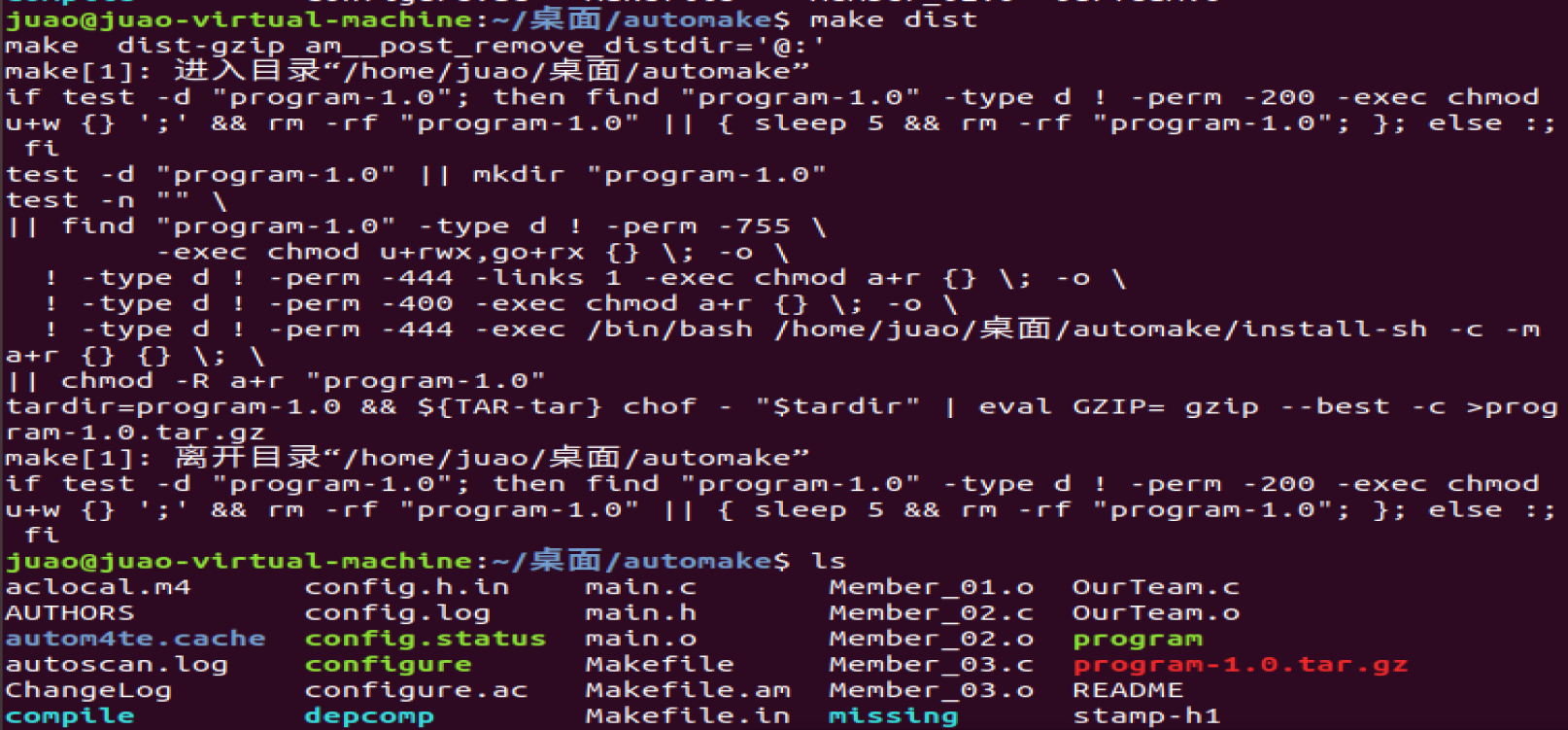
测试make生成的目标文件是否正确：成功！

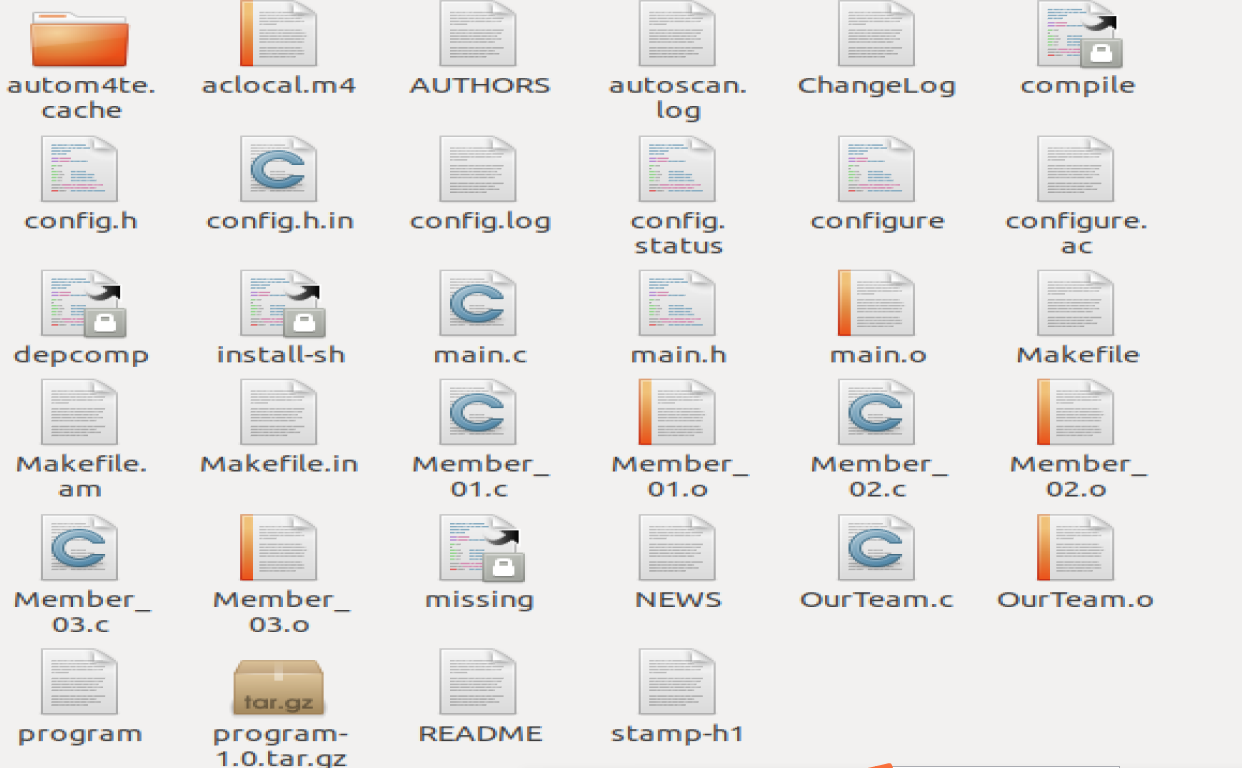


其他操作：make uninstall

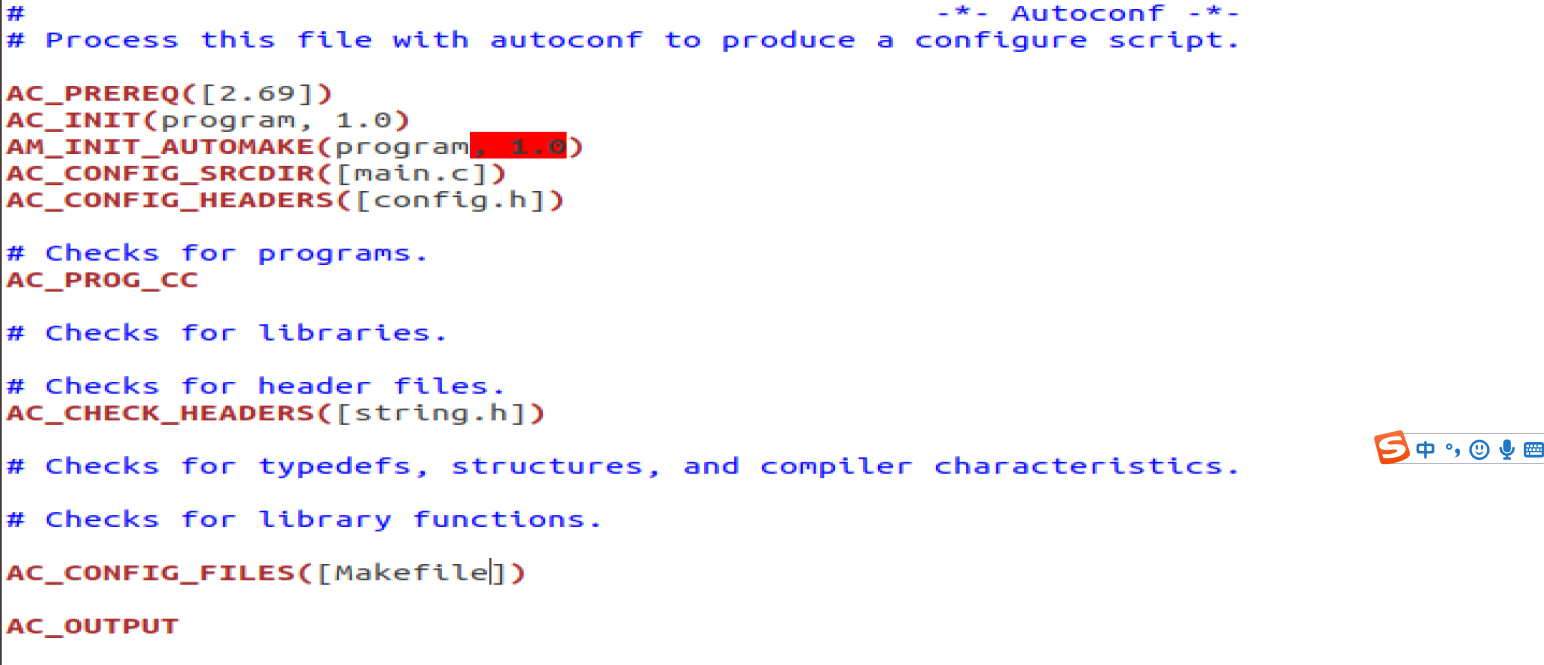


其他操作：make dist用于生成软件压缩包





Configure.ac内容：



Makefile.am内容：



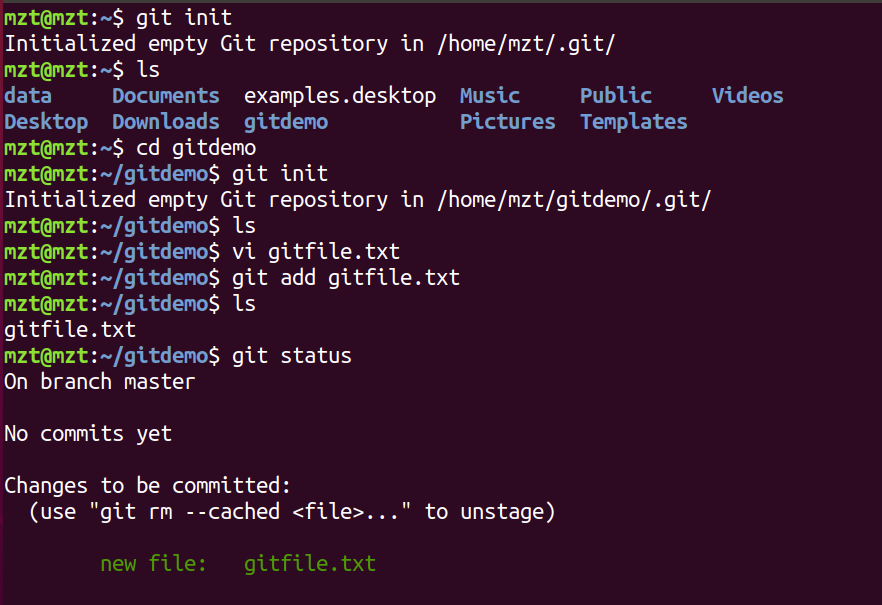
**5. 通过git/github管理团队软件和工作文件**

配置：

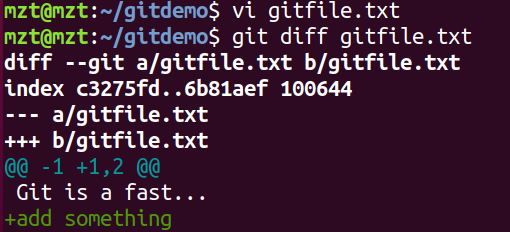


创建版本目录与初始化版本库 

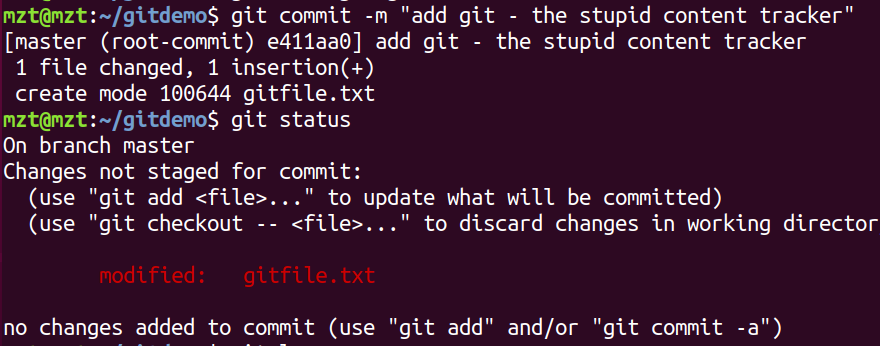
创建文件gitfile.txt 注：常见的版本库文件，如 .txt，codes等



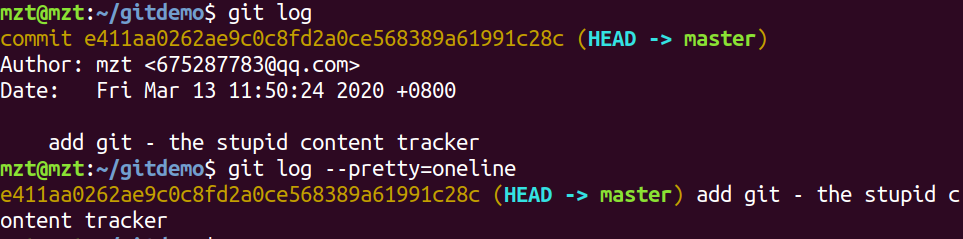
检查文件做了哪些修改：



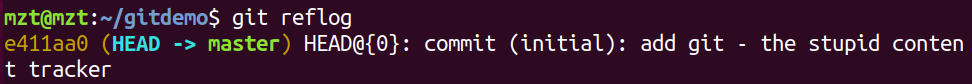
将修改后的文件提交到版本库：（并检查文件入库状态）



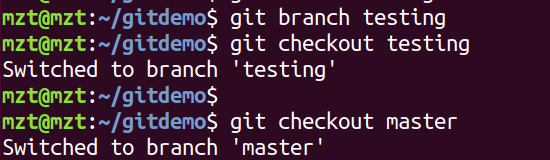
检查文件版本历史：



检查文件版本操作历史：



创建分支与切换分支

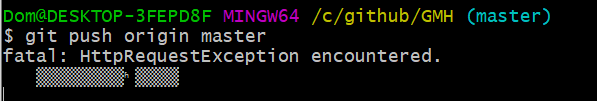


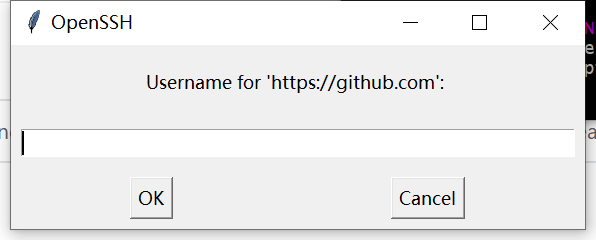
Github账号注册

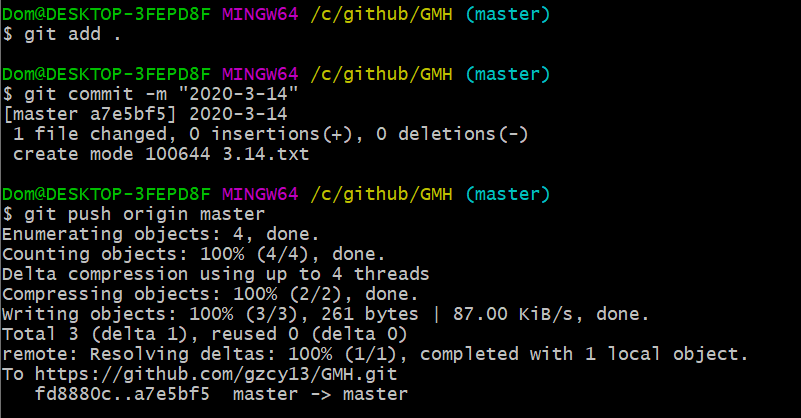
账号：MGHGIT

仓库名：MGH

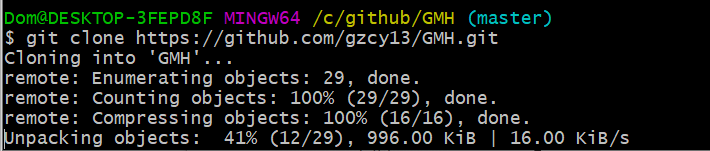
将本地仓库推送到远程







从远程库克隆到本地



**四、实验总结**

* 总结实验收获。

1. 熟悉了linux下的命令行编程。
2. Github不仅可以建立分支，版本管理，还可以邀请多个协作者。刚开始大家合作一个项目，发现建立分支fork的方式很麻烦，后来发现i添加合作者，这样合作者都可以直接对master版本进行修改。属于内部人员合作方式。分支应该属于外部人员的合作方式。

* 未解决的问题。

1、在虚拟机中传文件需要建立共享文件夹。VMware可以通过下载VMwaretool工具实现，但是virtual Box从mac把文件传到ubuntu里，ubuntu里找不到文件夹。

2、有时候更新内容到github仓库，需要多次尝试才会成功。

**五、附实验源码**

1. Main.c:
2. #include "main.h"
4. **void** print\_Member\_01()
5. {
6. printf("the first member: Gong Zhouchuangyu\n");
7. }

1. Main.h:
2. #ifndef  \_MAIN\_H\_
3. #define  \_MAIN\_H\_
4. #include <stdio.h>
5. **void** print\_OurTeam();
6. **void** print\_Member\_01();
7. **void** print\_Member\_02();
8. **void** print\_Member\_03();
9. #endif
11. OurTeam.c:
12. #include "main.h"
13. **void** print\_OurTeam()
14. {
15. printf("Our team has three members\n");
16. }
18. Member\_01.c:
19. #include "main.h"
20. **void** print\_Member\_01()
21. {
22. printf("the first member: Gong Zhouchuangyu\n");
23. }
25. Member\_02.c:
26. #include "main.h"
27. **void** print\_Member\_02()
28. {
29. printf("the second member: Meng Zhaotai\n");
30. }
32. Member\_03.c:
33. #include "main.h"
34. **void** print\_Member\_03()
35. {
36. printf("the third member: He Juao\n");
37. }
38. Makefile:
39. #My makefile
40. OBJS = main.o Member\_01.o Member\_02.o Member\_03.o OurTeam.o
41. HDRS = main.h
42. CFLAGS = -g
43. CC = gcc
44. program: $(OBJS)
45. $(CC) $(CFLAGS) $(OBJS) -o program
46. main.o: main.c $(HDRS)
47. $(CC) $(CFLAGS) -c $< -o $@
48. Member\_01.o: Member\_01.c $(HDRS)
49. $(CC) $(CFLAGS) -c $< -o $@
50. Member\_02.o: Member\_02.c $(HDRS)
51. $(CC) $(CFLAGS) -c $< -o $@
52. Member\_03.o: Member\_03.c $(HDRS)
53. $(CC) $(CFLAGS) -c $< -o $@
54. OurTeam.o: OurTeam.c $(HDRS)
55. $(CC) $(CFLAGS) -c $< -o $@
56. .PHONY:clean
57. clean:
58. rm \*.o