# 实验题目：构建主机端开发环境（上）

小组名称：嵌入式 Debug 小组

小组成员：伊亚玎、周泽园、谢豪

一、 实验目的

1. 搭建主机开发环境
2. 学会使用本地 gcc 编译应用程序
3. 学会使用 Makefile 管理应用程序
4. 学会使用 autotools 生成 Makefile，学会常用 make 操作
5. 学会使用 git/github 管理团队软件和工作文件

二、 实验内容

1. 安装 ubuntu 开发环境
2. 编写 c 应用程序，通过本地 gcc 编译应用程序
3. 编写 Makefile 管理应用程序
4. 通过 autotools 生成 Makefile
5. 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. 我们组使用的时云服务器，可直接装载 ubuntu 系统。若本地安装，可选择 VMware 虚拟机或者 EFI 双系统安装，这两种安装方式我们组都实验过。但考虑到操作的便捷性，我们组最后选择使用云服务器来作为开发环境。
2. 本地使用 gcc 编译，只需创建一个 .c 文件，并使用 gcc filename -o outname 即可实现编译。
3. 编写 Makefile 管理应用程序

    1) 手工创建 Makefile 文件并执行 make

```
#My makefile
OBJS = main.o dec_bin.o bin_dec.o dec_oct.o oct_dec.o dec_hex.o hex_dec.o
HDRS = funs.h
CFLAGS = -g
CC = gcc

program: $(OBJS)
	$(CC) $(CFLAGS) $(OBJS) -o program
main.o: main.c $(HDRS)
	$(CC) $(CFLAGS) -c $< -o $@
dec_bin.o: dec_bin.c
	$(CC) $(CFLAGS) -c $< -o $@
bin_dec.o: bin_dec.c
	$(CC) $(CFLAGS) -c $< -o $@
dec_oct.o: dec_oct.c
	$(CC) $(CFLAGS) -c $< -o $@
oct_dec.o: oct_dec.c
	$(CC) $(CFLAGS) -c $< -o $@
dec_hex.o: dec_hex.c
	$(CC) $(CFLAGS) -c $< -o $@
hex_dec.o: hex_dec.c
	$(CC) $(CFLAGS) -c $< -o $@
```

2) 报错：.c 文件被用于生成 .o 文件。

3) 解决：创建所有.c 文件，并执行 make

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test3# ls
funs.h  main.c  main.o  Makefile
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test3# make
make: *** No rule to make target 'dec_bin.c', needed by 'dec_bin.o'.  Stop.
```

4) 报错：在“_stare”中找不到 main 函数

5) 解决：在 main.c 中声明 main 函数

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test3# make
gcc -g -c main.c -o main.o
gcc -g -c dec_bin.c -o dec_bin.o
gcc -g -c bin_dec.c -o bin_dec.o
gcc -g -c dec_oct.c -o dec_oct.o
gcc -g -c oct_dec.c -o oct_dec.o
gcc -g -c dec_hex.c -o dec_hex.o
gcc -g -c hex_dec.c -o hex_dec.o
gcc -g main.o dec_bin.o bin_dec.o dec_oct.o oct_dec.o dec_hex.o hex_dec.o -o program
/usr/lib/gcc/x86_64-linux-gnu/7/../../../x86_64-linux-gnu/Scrt1.o: In function `_start':
(.text+0x20): undefined reference to `main'
collect2: error: ld returned 1 exit status
Makefile:8: recipe for target 'program' failed
make: *** [program] Error 1
```

```
#include<stdio.h>

int main(void){

        return 0;
}
```

6) 执行 make，生成 target 文件 program

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test3# make
gcc -g -c main.c -o main.o
gcc -g main.o dec_bin.o bin_dec.o dec_oct.o oct_dec.o dec_hex.o hex_dec.o -o program
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test3# ls
bin_dec.c  dec_bin.c  dec_hex.c  dec_oct.c  funs.h     hex_dec.o  main.o    oct_dec.c  program
bin_dec.o  dec_bin.o  dec_hex.o  dec_oct.o  hex_dec.c  main.c     Makefile  oct_dec.o
```

4. 通过 autotools 生成 Makefile，并完成 make 操作

1) 创建所有需要用到的 .c 文件，并使用

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# ls
bin_dec.c  dec_bin.c  dec_hex.c  dec_oct.c  funs.h  hex_dec.c  main.c  oct_dec.c
```

2) autoscan 扫描目录生成 configure.scan 和 autoscan.log 文件

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# autoscan
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# ls
autoscan.log  configure.scan  dec_hex.c  funs.h     main.c
bin_dec.c     dec_bin.c       dec_oct.c  hex_dec.c  oct_dec.c
```

3) 修改 configure.scan 内容并将其重命名为 configure.ac

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# mv configure.scan configure.ac
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# ls
autoscan.log  configure.ac  dec_hex.c  funs.h     main.c
bin_dec.c     dec_bin.c     dec_oct.c  hex_dec.c  oct_dec.c
```

```
#                                               -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ([2.69])
AC_INIT(program, 1.0)
#AM_INIT_AUTOMAKE需手动添加，是automake必须用到的宏，创建发布压缩包时会用到
AM_INIT_AUTOMAKE(program, 1.0)
#AC_CONFIG_SCRDIR检查指定源码目录中源码文件是否存在，可以指定所有相关源文件
AC_CONFIG_SRCDIR([main.c])
AC_CONFIG_HEADERS([config.h])

# Checks for programs.
AC_PROG_CC                      #AC_PROG_CC指定编译器
# Checks for libraries.
# Checks for header files.
AC_CHECK_HEADERS([string.h])
# Checks for typedefs, structures, and compiler characteristics.
# Checks for library functions.
AC_CONFIG_FILES([Makefile])     #AC_CONFIG_FILES需要手动添加，用于生成最终的Makefile文件
AC_OUTPUT
```

4) 使用 aclocal 命令扫描 configure.ac 文件自动生成 aclocal.m4 和 automa4te.cache

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# aclocal
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# ls
aclocal.m4      autoscan.log  configure.ac  dec_hex.c  funs.h     main.c
autom4te.cache  bin_dec.c     dec_bin.c     dec_oct.c  hex_dec.c  oct_dec.c
```

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

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# autoconf
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# ls
aclocal.m4      autoscan.log  configure     dec_bin.c  dec_oct.c  hex_dec.c  oct_dec.c
autom4te.cache  bin_dec.c     configure.ac  dec_hex.c  funs.h     main.c
```

6) 使用 autoheader 命令生成 config.h.in 文件

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# autoheader
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# ls
aclocal.m4      autoscan.log  config.h.in  configure.ac  dec_hex.c  funs.h     main.c
autom4te.cache  bin_dec.c     configure    dec_bin.c     dec_oct.c  hex_dec.c  oct_dec.c
```

7) 手工创建 Makefile.am 文件

```
AUTOMAKE_OPTIONS=foreign        # 用来设置软件等级，选项为foreign，GUN和gnits；foreign只检测必要的文件

bin_PROGRAMS=program            # 如果定义多个执行文件，之间用空格分开

# 如果定义多个执行文件，需要建立每个相应的file_SOURCES
program_SOURCES=main.c funs.h dec_bin.c bin_dec.cdec_oct.c oct_dec.c dec_hex.c hex_dec.c
```

8) automake --add-missing 命令生成 Makefile.in 文件

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# ls
aclocal.m4      bin_dec.c     configure     dec_hex.c  funs.h      main.c       Makefile.in
autom4te.cache  compile       configure.ac  dec_oct.c  hex_dec.c   Makefile.    missing
autoscan.log    config.h.in   dec_bin.c     depcomp    install-sh  Makefile.am  oct_dec.c
```

9) 使用./configure 生成 Makefile 文件

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# ./configure^C
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# ls
aclocal.m4      compile      config.status  dec_hex.c  hex_dec.c   Makefile.    oct_dec.c
autom4te.cache  config.h     configure      dec_oct.c  install-sh  Makefile.am  stamp-h1
autoscan.log    config.h.in  configure.ac   depcomp    main.c      Makefile.in
bin_dec.c       config.log   dec_bin.c      funs.h     Makefile    missing
```

10) 运行 make 测试生成的目标文件是否正确，报错

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/test4# make
make  all-am
make[1]: Entering directory '/study/embedded_study/test4'
gcc -DHAVE_CONFIG_H -I.     -g -O2 -MT main.o -MD -MP -MF .deps/main.Tpo -c -o main.o main.c
mv -f .deps/main.Tpo .deps/main.Po
gcc -DHAVE_CONFIG_H -I.     -g -O2 -MT dec_bin.o -MD -MP -MF .deps/dec_bin.Tpo -c -o dec_bin.o dec_bin.c
mv -f .deps/dec_bin.Tpo .deps/dec_bin.Po
make[1]: *** No rule to make target 'bin_dec.cdec_oct.c', needed by 'bin_dec.cdec_oct.o'.  Stop.
make[1]: Leaving directory '/study/embedded_study/test4'
Makefile:279: recipe for target 'all' failed
make: *** [all] Error 2
```

11) 解决：Makefile.am 文件中少了一个空格；并且 main.c 文件中需要声明 main 函数

```
1 AUTOMAKE_OPTIONS=foreign        # 用来设置软件等级，选项为foreign，GUN和gnits；foreign只检测必要的文件
2
3 bin_PROGRAMS=program            # 如果定义多个执行文件，之间用空格分开
4
5 # 如果定义多个执行文件，需要建立每个相应的file_SOURCES
6 program_SOURCES=main.c funs.h dec_bin.c bin_dec.cdec_oct.c oct_dec.c dec_hex.c hex_dec.c
```

12) make 编译，并执行 program

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug# ls
aclocal.m4      compile      configure     dec_hex.o  hex_dec.c   Makefile     oct_dec.o
autom4te.cache  config.h     configure.ac  dec_oct.c  hex_dec.o   Makefile.am  program
autoscan.log    config.h.in  dec_bin.c     dec_oct.o  install-sh  Makefile.in  stamp-h1
bin_dec.c       config.log   dec_bin.o     depcomp    main.c      missing
bin_dec.o       config.status  dec_hex.c   funs.h     main.o      oct_dec.c
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug# program
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug# make install
```

13) make install and make uninstall

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug# make install
make[1]: Entering directory '/study/embedded_study/ErrorDebug'
 /bin/mkdir -p '/usr/local/bin'
  /usr/bin/install -c program '/usr/local/bin'
make[1]: Nothing to be done for 'install-data-am'.
make[1]: Leaving directory '/study/embedded_study/ErrorDebug'
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug# program
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug#
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug# make uninstall
 ( cd '/usr/local/bin' && rm -f program )
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug#
```

14) make dist 生成压缩包

```
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug# make dist
make  dist-gzip am__post_remove_distdir='@:'
make[1]: Entering directory '/study/embedded_study/ErrorDebug'
if test -d "program-1.0"; then find "program-1.0" -type d ! -perm -200 -exec chmod u+w {} ';' && rm -rf
gram-1.0" || { sleep 5 && rm -rf "program-1.0"; }; else :; fi
test -d "program-1.0" || mkdir "program-1.0"
test -n "" \
|| find "program-1.0" -type d ! -perm -755 \
        -exec chmod u+rwx,go+rx {} \; -o \
  ! -type d ! -perm -444 -links 1 -exec chmod a+r {} \; -o \
  ! -type d ! -perm -400 -exec chmod a+r {} \; -o \
  ! -type d ! -perm -444 -exec /bin/bash /study/embedded_study/ErrorDebug/install-sh -c -m a+r {} {} \;
|| chmod -R a+r "program-1.0"
tardir=program-1.0 && ${TAR-tar} chof - "$tardir" | eval GZIP= gzip --best -c >program-1.0.tar.gz
make[1]: Leaving directory '/study/embedded_study/ErrorDebug'
if test -d "program-1.0"; then find "program-1.0" -type d ! -perm -200 -exec chmod u+w {} ';' && rm -rf
gram-1.0" || { sleep 5 && rm -rf "program-1.0"; }; else :; fi
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug# ls
aclocal.m4      compile      configure     dec_hex.o  hex_dec.c   Makefile     oct_dec.o
autom4te.cache  config.h     configure.ac  dec_oct.c  hex_dec.o   Makefile.am  program
autoscan.log    config.h.in  dec_bin.c     dec_oct.o  install-sh  Makefile.in  program-1.0.tar.gz
bin_dec.c       config.log   dec_bin.o     depcomp    main.c      missing      stamp-h1
bin_dec.o       config.status  dec_hex.c   funs.h     main.o      oct_dec.c
root@iZm5ehbv33s9rnh3h2rth3Z:/study/embedded_study/ErrorDebug#
```

5、git 工具版本控制

（1）再 GitHub 上创建 repo，git clone 到本地

```
root@ubuntu:/home/xiehao/embed# git clone https://github.com/yiyading/Embedded-
software
Cloning into 'Embedded-software'...
remote: Enumerating objects: 64, done.
remote: Counting objects: 100% (64/64), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 64 (delta 16), reused 61 (delta 16), pack-reused 0
Unpacking objects: 100% (64/64), done.
```

（2）在本地对程序进行开发，下图所示是编写了一个排序程序

```
root@ubuntu:/home/xiehao/embed/Embedded-software/first-experiment-2020314/Makef
ile3/barrel_sort# ls
barrel.c  barrel.o    char2int.h  main.c  makefile
barrel.h  char2int.c  char2int.o  main.o  myproj
```

（3）将文件添加到本地版本库

```
root@ubuntu:/home/xiehao/embed/Embedded-software/first-experiment-2020314/Makef
ile3# git add ./barrel_sort
root@ubuntu:/home/xiehao/embed/Embedded-software/first-experiment-2020314/Makef
ile3# git commit
[master 49449a1] "barrel_sort edited by XieHao"
 10 files changed, 102 insertions(+)
 create mode 100644 first-experiment-2020314/Makefile3/barrel_sort/barrel.c
 create mode 100644 first-experiment-2020314/Makefile3/barrel_sort/barrel.h
 create mode 100644 first-experiment-2020314/Makefile3/barrel_sort/barrel.o
 create mode 100644 first-experiment-2020314/Makefile3/barrel_sort/char2int.c
 create mode 100644 first-experiment-2020314/Makefile3/barrel_sort/char2int.h
 create mode 100644 first-experiment-2020314/Makefile3/barrel_sort/char2int.o
 create mode 100644 first-experiment-2020314/Makefile3/barrel_sort/main.c
 create mode 100644 first-experiment-2020314/Makefile3/barrel_sort/main.o
 create mode 100644 first-experiment-2020314/Makefile3/barrel_sort/makefile
 create mode 100755 first-experiment-2020314/Makefile3/barrel_sort/myproj
```

（4）将本地仓库推送到远程

```
root@ubuntu:/home/xiehao/embed/Embedded-software/first-experiment-2020314/Makef
ile3# git push origin master
Username for 'https://github.com': un-cmd
Password for 'https://un-cmd@github.com':
Counting objects: 15, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 6.34 KiB | 3.17 MiB/s, done.
Total 15 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), completed with 2 local objects.
To https://github.com/yiyading/Embedded-software
   6cab546..49449a1  master -> master
```

## 四、 实验总结

1. 实验最大的收获就是在实验的过程中使用 word 文档记录每一步的操作，当出现报错时，可以明确的定位到出错的步骤
2. 未解之谜：在使用 aclocal 时，使用不同的主机会出现 .m4 文件无法生

成的问题。

3. git clone 之后，只要 repo 中加有本地 key，则可以直接 push：git push origin master

## 五、 实验源码

两个主要实验的关键源码就是 Makefile 文件内的源码，第一个实验 Makefile 源码简单，在实验过程中给出了截图。第二个实验使用 autotools 工具生成的 Makeifle 文件内的源码有 700+行，在 ErrorDebug_autotools 中可以查看。