

嵌入式系统实验报告

学生姓名:

学 号:

专 业:

班 级:

指导教师:

完成日期: 2019 年 12 月 22 日

目 录

[嵌入式系统实践 实验报告1（使用GCC编译C语言程序） 3](#_Toc27907030)

[嵌入式系统实践 实验报告2（Linux 中通过minicom串口下载程序） 16](#_Toc27907031)

# 嵌入式系统实践 实验报告1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称：** | 使用GCC编译C语言程序 | | | | |
| **班 级：** | 物联网工程17-2班 | **姓 名：** | 文华 | **学 号：** | 2017218007 |
| **实验地点：** |  | **日 期：** | 2019年11月18日 | | |

|  |
| --- |
| 一、实验目的：   1. 了解GNU gcc编译器 2. 掌握使用GCC编译C语言程序的方法 |
| 二、实验环境：  CentOS |
| 三、实验内容和要求：   1. 使用vi或其它文本编辑器，输入C语言程序，并保存为test.c。 2. 在Linux shell下，输入命令gcc –o test test.c 3. 编译正确后，输入命令./test运行程序，观察程序运行结果 4. 若编译错误，根据提示信息，进入程序查错，再回到第二步，直至程序语法无误。 5. 输入后面第八条中的4个程序。按照下面的要求进行。   使用gcc编译器，编译程序  第一种方法：分步进行   1. 由star.c 和 starfun.h 文件生成star.o 目标文件   gcc -c star.c -o star.o   1. 由hello.c， hello.h和 starfun.h生成hello.o目标文件   gcc -c hello.c -o hello.o   1. 由hello.o 和star.o 生成应用程序myprog   gcc star.o hello.o -o myprog   1. 执行myprog   [root@localhost 01\_hello]# ./myprog  观察程序myprog运行的结果。  \*  \*\*\*  \*\*\*\*\*  \*\*\*\*\*\*\*  \*\*\*\*\*  \*\*\*  \*    \*  \*\*\*  \*\*\*\*\*  \*\*\*\*\*\*\*  hello,my friends  第二种方法：一条命令完成以上操作  gcc star.c hello.c -o myprog1  结合不同的选项，观察编译的过程和得到的结果是否有所不同。  [root@localhost 01\_hello]# gcc star.c hello.c -o myprog1  [root@localhost 01\_hello]# gcc -w star.c hello.c -o myprog2  [root@localhost 01\_hello]# gcc -Wall star.c hello.c -o myprog3  体会-Wall 和 -w选项的作用  查阅当前的gcc版本命令  [root@localhost 01\_hello]# gcc -v   1. 使用动态库   [root@localhost 01\_hello]# gcc -c -fpic hello.c  [root@localhost 01\_hello]# ls  amake hello.c hello.h hello.o makefile\_01 makefile\_02 makefile\_03 Makefile\_rule star.c starfun.h  [root@localhost 01\_hello]# gcc -shared -s -o libhello.so hello.o  [root@localhost 01\_hello]# ls  amake hello.c hello.h hello.o libhello.so makefile\_01 makefile\_02 makefile\_03 Makefile\_rule star.c starfun.h  注意libhello.so库文件的命名格式，1）2）也可以用下边命令替代  gcc -fpic -shared -s hello.c -o libhello.so  [root@localhost 01\_hello]# cp libhello.so /usr/lib  注意/usr/lib为用户库自动搜索路径  [root@localhost 01\_hello]# gcc -lhello star.c -o mystar  [root@localhost 01\_hello]# ldd mystar  libhello.so => /usr/lib/libhello.so (0x4002d000)  libc.so.6 => /lib/tls/libc.so.6 (0x42000000)  /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)  [root@localhost 01\_hello]# ./mystar  \*  \*\*\*  \*\*\*\*\*  \*\*\*\*\*\*\*  \*\*\*\*\*  \*\*\*  \*    \*  \*\*\*  \*\*\*\*\*  \*\*\*\*\*\*\*  hello,my friends   1. 使用静态库   [root@localhost 01\_hello]# rm \*.o  rm：是否删除一般文件‘hello.o’? y  [root@localhost 01\_hello]# rm mystar  rm：是否删除一般文件‘mystar’? y  [root@localhost 01\_hello]# rm libhello.\*  [root@localhost 01\_hello]# gcc -c hello.c -o hello.o  [root@localhost 01\_hello]# ar -rc libhello.a hello.o  [root@localhost 01\_hello]# gcc star.c libhello.a -o mystar  [root@localhost 01\_hello]# ./mystar  \*  \*\*\*  \*\*\*\*\*  \*\*\*\*\*\*\*  \*\*\*\*\*  \*\*\*  \*    \*  \*\*\*  \*\*\*\*\*  \*\*\*\*\*\*\*  hello,my friends   1. 上面所用到的4个文件hello.h，starfun.h，hello.c，star.c，内容参考如下：   Starfun.h 文件内容如下：  /\*\*\*\*\*starfun.h\*\*\*\*\*/  #ifndef STARFUN\_H  #define STARFUN\_H  #define NUM 4  #define NUMBER 3  int star1() {  int i,j,k;  for(k=1;k<=NUM;++k) {  for(i=1;i<=(NUM-k);++i)  printf(" ");  for(j=1;j<=(2\*k-1);++j)  printf("\*");  printf("\n");  }  return 0;  }  int star2() {  int i,j,k;  for(k=NUMBER;k>=0;--k) {  for(i=1;i<=(NUMBER-k+1);++i)  printf(" ");  for(j=1;j<=(2\*k-1);++j)  printf("\*");  printf("\n");  }  return 0;  }  #endif  hello.h文件内容如下：  /\*\*\*\*\*hello.h\*\*\*\*\*/  #ifndef HELLO\_H  #define HELLO\_H  void hello() {  star1();  printf("hello,my friends\n");  }  #endif  hello.c文件内容如下：  /\*\*\*\*\*hello.c\*\*\*\*\*/  void showhello() {  hello();  }  star.c文件内容如下：  /\*\*\*\*\*star.c\*\*\*\*\*/  #include "starfun.h"  #include "hello.h"  #include <stdio.h>  int main() {  star1();  star2();  showhello();  return 0;  }  **附：GCC使用方法和常用选项**  使用GCC编译C程序生成可执行文件需要经历4个步骤：   1. 预处理，这一步需要分析各种命令，如#define、#include、#ifdef等。Gcc调用cpp程序来进行预处理 2. 编译，这一步将根据输入文件产生汇编语言，gcc调用ccl进行编译工作 3. 汇编，这一步中将汇编语言作为输入，产生具有.o扩展名的目标文件，gcc调用as进行汇编工作 4. 连接，这一步中各目标文件.o被放在可执行文件的适当位置上，该程序引用的函数也放在可执行文件中，gcc调用ld来完成   gcc命令的基本用法为：gcc [option] [filename]，命令行选项指定的操作将在命令行上每个给出的文件上执行。例如：  gcc –o prog main.c test1.c test2.c  其中，“-o prog”指定输出的可执行文件名为prog，如果没有指定-o参数，gcc将使用默认的可执行文件名a.out  gcc的命令选项有许多项，但经常使用的几个选项是：   1. -c：只预处理，编译和汇编源程序，不进行连接 2. -o exefile，确定输出文件为exefile，如果没有该选项，默认输出为可执行文件a.out 3. -Dmacro或-Dmacro=defn，其作用类似于源程序代码中的#define 4. -O，对程序编译进行优化，编译后可执行文件的长度和执行时间缩短，但编译过程耗时变长，对主机性能要求较高。 5. -O2，比-O更好地优化 6. -g，告诉gcc产生能被GNU调试器使用的调试信息以便调试程序 7. -Idir，将dir目录加到搜寻头文件的目录列表中。并优先于在gcc中默认的搜寻目录 |
| 四、实验步骤：  按照“三、实验内容和要求”分步操作。 |
| **五、 实验结果与分析（**含程序、数据记录及分析和实验总结等**）：**    图 5-1 实验结果截图1    图 5-2 实验结果截图2    图 5-3 实验结果截图3    图 5-4 实验结果截图4    图 5-5 实验结果截图5    图 5-6 实验结果截图6    图 5-7 实验结果截图7    图 5-8 实验结果截图8    图 5-9 实验结果截图9    图 5-10 实验结果截图10  本次实验按照实验步骤进行，完全符合实验要求，达到了实验预期。 |
| 六、教师评语：  实验成绩： 教师：（签名要全称） 年 月 日 |

# 嵌入式系统实践 实验报告2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称：** | Linux 中通过minicom串口下载程序 | | | | |
| **班 级：** | 物联网工程17-2班 | **姓 名：** | 文华 | **学 号：** | 2017218007 |
| **实验地点：** |  | **日 期：** | 2019年12月3日 | | |

|  |
| --- |
| 一、实验目的：  熟悉Linux开发环境，学会基于Mini6410的Linux开发环境的配置和使用。使用Linux的arm-linux-gcc编译，minicom串口方式下载调试。 |
| 二、实验环境：  硬件：Mini6410嵌入式实验平台。  软件：PC机操作系统CentOS＋Minicom＋Arm-Linux 交叉编译环境。    图 2-1 嵌入式开发板顶视图 |
| 三、实验内容和要求：  本次实验使用CentOS6.7 操作系统环境,安装ARM-Linux 的开发库及编译器。创建一个新目录，并在其中编写hello.c 和Makefile 文件。学习在Linux 下的编程和编译过程，以及ARM 开发板的使用和开发环境的设置。下载已经编译好的文件到目标开发板上运行。 |
| 四、实验步骤：   1. 建立工作目录     图 4-1  此时我们新建的hello工作目录，在home目录下，已出现，说明，我们此次操作成功**(这里我们要注意，记清楚自己在创建目录时，所在的位置)**，如下图所示：    图 4-2  编写程序源代码  在Linux 下的文本编辑器有许多，常用的是vi 和Xwindow界面下的gedit 等，我们在开发过程中推荐使用vi。  hello.c 源代码较简单，如下：   |  | | --- | | /\*\*\*\*\*hello.c\*\*\*\*\*/  #include <stdio.h>  int main**()**  **{**  printf**(**"Hello,World!\n"**);**  **return** 0**;**  **}** |   我们可以是用下面的命令来编写hello.c 的源代码，进入hello 目录使用vi 命令来编辑代码(**如果不会使用vi命令来编辑，我们也可以使用gedit命令来编辑hello.c文件，命令为：gedit hello.c)**  **vi命令中常用命令有**： esc i :wq :q!    图 4-3  按“i”或者“a”进入编辑模式，将上面的代码录入进去，完成后按Esc 键进入命令状态，再用命令“：wq”保存并退出。这样我们便在当前目录下建立了一个名为hello.c 的文件。   1. 编写Makefile   要使上面的hello.c 程序能够运行，我们必须要编写一个Makefile 文件，Makefile 文件定义了一系列的规则，它指明了哪些文件需要编译，哪些文件需要先编译，哪些文件需要重新编译等等更为复杂的命令。使用它带来的好处就是自动编译，你只需要敲一个“make” 命令整个工程就可以实现自动编译，当然我们本次实验只有一个文件，它还不能体现出使用Makefile 的优越性，但当工程比较大文件比较多时，不使用Makefile 几乎是不可能的。下面我们介绍本次实验用到的Makefile 文件。  /\*\*\*\*\*Makefile\*\*\*\*\*/  CC=arm-linux-gcc  EXEC=armhello  OBJS=hello.o  CFLAGS+=  LDFLAGS+=  all: $(EXEC)  $(EXEC): $(OBJS)  <TAB>$(CC) $(LDFLAGS) -o $@ $(OBJS)  clean：  <TAB>rm -f $(EXEC) \*.o  在shell环境下，运行make，则自动编译程序，生成可执行程序armhello，执行armhello。  [root@localhost home]# make  [root@localhost home]# ls  [root@localhost home]# ./armhello   1. 编译应用程序   在上面的步骤完成后，我们就可以在hello 目录下运行“make”来编译我们的程序了。如果进行了修改，重新编译则运行:    图 4-4  注意：编译、修改程序都是在宿主机（本地PC 机）上进行，不能在MINICOM 下进行。   1. 下载调试   由于采用的是arm编译器，所以编译后的程序是不能直接在PC机上运行的。需要下载到开发板上才能运行。通过minicom串口通信软件可以进行下载。  需要在CentOS中先添加串口连接，然后才能通过minicom串口通信软件进行串口下载。如下图所示。同时要将开发板最边上的串口连接到计算机后面的串口上，开发板最边上的电源连接接口也要连上电源线。开发板靠近电源连接线边上的黑色的开关是电源开关，推下开关，开发板上电源指示灯亮，表示开发板电源连接正常。    图 4-5    图 4-6    图 4-7    图 4-8    图 4-9    图 4-10    图 4-11   1. 进入root。终端输入su root，再输入密码即可。     图 4-12   1. 终端输入minicom -s，出现设置串口的界面，进行相应的设置，保存设置后，然后按ctrl+a,就进入了minicom下载模式。等待1分钟后，可能要重新启动开发板，以及要按几下回车键，才会出现下面的登录开发板成功的图形。   minicom –s  点击串口设置，选中ttyS1，保存退出。      图 4-13  输入minicom ，回车后出现上面的图形后，才能进行正常的下载。   1. 先按下ctrl+a键,再按下键盘上的s键，出现下图     图 4-14  选择第一个zmodem或者xmodem，回车。出现下图    图 4-15  回车，出现下图  此处 单击空格代表选中.双击空格代表回到上一级目录或者是进入当前目录内部.最好将用户要下载的程序拷贝到home目录下,然后选中进行下载.    图 4-16  输入绝对路径，出现下图    图 4-17  回车，出现下图    图 4-18  从上图可以看出，文件未被下载，原因是mini6410板子已经有了armhello可执行文件（之前下载的），若需要下载，则需要删除之前的armhello文件，rm armhello即可。删除之后，继续下载，出现下图    图 4-19  下载成功。   1. 运行程序   在开发板对应的终端上输入./armhello，执行armhello程序，观察执行的结果。    图 4-20 |
| **五、 实验结果与分析（**含程序、数据记录及分析和实验总结等**）：**    图 5-1 实验截图1    图 5-2 实验截图2    图 5-3 实验截图3  本次实验按照实验步骤进行，完全符合实验要求，达到了实验预期。 |
| 六、教师评语：  实验成绩： 教师：（签名要全称） 年 月 日 |