# **<实验三> Linux常用开发工具**

班级学号： 姓名：

≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡≡

|  |  |
| --- | --- |
| 实验目的和内容 | 实验目的：  熟悉和掌握C语言源程序的编译、链接与调试过程，学会Linux常用工具 GCC（编译） GDB（调试）和Make（维护）的基本使用。   1. 练习vi或gedit 文本编辑器的使用   2．掌握gcc编译器的使用。  3．掌握gdb调试器的使用。  4．掌握make工具的使用。 |
| 实验环境 | 微机、WINDOWS XP、 LINUX操作系统、VMWARE虚拟机软件 |
| 实验重点及难点 | 实验重点：GCC对源程序编译、链接的过程；GDB调试程序的方法，  MakeFile 对程序编译过程的控制  实验难点：GDB和Make工具使用方法 |
| 实验过程（实验过程、实验结果及实验分析） | |
| **一、**使用vi编辑器完成以下4个文件的内容输入：  [操作步骤]  (1)在/home目录下创建experiment3子目录，其命令： cd /home (enter) mkdir experiment3 。  (2)在该目录下，创建以下四个文件，文件名：hello.h、starfun.h、hello.c、star.c, 其命令是： geidt hello.h gedit starfun.h gedit hello.c gedit star.c 。  (3)使用vi编辑器分别输入以下内容，如图1、图2所示：    图1 使用gedit编辑以下内容  图2 使用gedit所建立的4个文件  A. 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  **B. 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  **C. hello.c 文件内容如下：**  void showhello() {  hello();  }  **D. star.c文件内容如下：**  #include "starfun.h"  #include "hello.h"  #include <stdio.h>  int main() {  star1();  star2();  showhello();  return 0;  }  **二．使用gcc编译器，编译程序**  [操作步骤]  (1)第一种方法：分步进行  A．由star.c starfun.h 文件生成star.o 目标文件：  gcc -c star.c -o star.o  B．由hello.c hello.h starfun.h生成hello.o目标文件  gcc -c hello.c -o hello.o  C．由hello.o star.o 生成应用程序myprog  gcc star.o hello.o -o myprog  D．执行应用程序myprog  [root@localhost 01\_hello]# ./myprog  (2) 一条命令完成以上操作:  A.执行清屏命令： clear  B．输入命令生成可执行程序myprog ： gcc star.c hello.c -o myprog  C．执行程序myprog,其命令是： ./myprog  **三. GDB调试器的使用**  **(1)．启动gdb调试程序**  1、创建文件greeting.c 如下  #include  <stdio.h> int main() { void my\_print(char \*string); void my\_print2(char \*string); char my\_string[]="hello there"; my\_print(my\_string); my\_print2(my\_string); }  void my\_print(char \*string) { printf("The string is %sn",string); }  void my\_print2(char \*string) { char \*string2; int size,size2,i; // int size,i; size=strlen(string); size2=size-1; string2=(char \*)malloc(size+1); for (i=0;i<=size;i++) string2[size2-i]=string[i]; // string2[size-i]=string[i]; string string2[size+1]='0'; printf("The string printed backward is %sn",string2); }  3、创建Makefile文件如下：  greeting:greeting.o gcc -g -o greeting greeting.o greeting.o:greeting.c gcc -g -c greeting.c clean: rm \*.o  4、执行make 5、执行gdb greeting 进入gdb调试    **(2) gdb相关操作**  A. 查看文件，在gdb中键入“l”（list）就可以查看所载入的文件  B.设置断点 ，在gdb中设置断点非常简单，只需在“b”后加入对应的行号即可  C.查看断点情况，在设置完断点之后，用户可以键入“info b”来查看设置断点情况，在gdb中可以设置多个断点  D.运行代码，gdb默认从首行开始运行代码，可键入“r”（run）即可  E.查看变量值，在gdb中只需键入“p”＋变量值即可  F.单步运行，单步运行可以使用命令“n”（next）或“s”（step）  G. 恢复程序运行，可以使用命令“c”（continue）恢复程序的正常运行了.  四．维护工具Make的使用  用Make自动实现多个.c文件的编译  1、创建文件hello1.c、hello2.c、hello.h和makefile  //hello1.c  //written by Emdoor  #include <stdio.h>  int main()  {  printf("Welcome Emdoor!\n");  test2();  return 1;  }  //hello2.c  //written by Emdoor  ＃include "hello2.h"  #include <stdio.h>  void test2(void)  {  printf("Welcome Emdoor! –hello2\n");  }  //hello2.h  //written by Emdoor  void test2(void);  # makefile test for multi files program  #written by Emdoor  CC=gcc  CFLAGS=  OBJS=hello1.o hello2.o  all: hello  hello: $(OBJS)  $(CC) $(CFLAGS) $^ -o $@  hello1.o: hello1.c  $(CC) $(CFLAGS) –c $< -o $@  hello2.o: hello2.c  $(CC) $(CFLAGS) –c $< -o $@  clean:  rm –rf hello \*.o  2、先后执行如下命令  [root@local]$make  [root@local]$ls  [root@local]$./hello  查看并记录所生成的文件和运行的结果,如下图图14所示并写出你的结论。  图14 多个.c文件的make工具的编译  3、修改makefile文件（斜黑体表示修改部分）  # makefile test for multi files program  #written by Emdoor  CC=gcc  CFLAGS=  ***CFILES=$(wildcard \*.c)***  OBJS=***$(CFILES:%.c=%.o)***  all: hello  hello: $(OBJS)  $(CC) $(CFLAGS) ***–o hello $(OBJS)***  ***.c.o:***  ***$(CC) –c $<***  clean:  rm –rf hello \*.o   1. 重复第2步操作，查看并记录所生成的文件和运行的结果, 如下图图15所示并写出你的结论。并指出wildcard、.c.o的含义和变量CFILES代表的内容。     图15 修改makefile文件再次编译情况  结论：  含义： | |
| 实验总结 | |
| 通过实验，，， | |