

(3) 隐式规则：指出何时以及如何根据名称重新编译或更新一类文件。隐式规则描述了目标如何依赖一个与目标名称相似的文件，并且给出了创建或更新目标的命令。

(4) 指令：当使用 make 读取 Makefile 文件时，指令用来告诉 make 执行一些特殊活动，例如：

- ① 读取其他 Makefile 文件；
- ② 根据变量的值决定是忽略还是使用 Makefile 文件中的部分内容；
- ③ 定义多行变量。

(5) 注释：Makefile 文件中的注释以“#”开头，表示该行将在执行时被忽略。

前文所述的 Makefile 文件可以根据变量定义、自动推导和隐式规则进行修改，例如改写成如下内容：

```
obj= main.o hello1.o hello2.o    #变量定义
main: $(obj)
    gcc -o main $(obj)
hello1.o: hello1.h                #自动推导和隐式规则
hello2.o: hello2.h
clean:
    rm $(obj)
```

Makefile 文件的编写相对复杂，包括对规则编写、指令使用、函数调用等的详细说明等。读者可参阅具体的帮助文档加以了解。

2.4 实验 2.1：Linux 常用命令的使用

一、实验目的

了解 Linux 操作系统的 Shell 命令格式，熟练掌握常用命令和选项的功能。

二、实验内容

练习常用的 Linux Shell 命令及命令选项，包括文件目录命令、备份压缩命令、重定向及管道命令等。要求熟练掌握下列命令的使用。

- (1) 改变及显示目录命令：cd、pwd、ls。
- (2) 文件及目录的创建、复制、删除和移动命令：touch、cp、mv、rm、mkdir、rmdir。
- (3) 显示文件内容命令：cat、more、less、head、tail。
- (4) 文件查找命令：find、whereis、grep。
- (5) 文件和目录权限改变命令：chmod。
- (6) 备份和压缩命令：tar、gzip、bzip2。



三、实验指导

具体开展实验时, 将上述实验内容中的命令均练习一遍, 并查看结果。为此, 实验步骤可分为两大步:

- (1) 打开终端, 在提示符下输入命令;
 - (2) 执行每一条命令后, 分析结果, 修改选项后再次执行, 查看并记录结果的变化。
- 上述命令的部分示例用法如表 2.2 所示。

表 2.2 Linux 常用命令用法示例

(1) <code>cd /home</code> <code>cd ..</code> <code>pwd</code> <code>ls -la</code>	(4) <code>find /home/usr1 -name myfile</code> <code>whereis java</code> <code>locate *file*</code> <code>grep smith phonebook</code>
(2) <code>touch file1</code> <code>cp file1 /home/user/stu</code> <code>mv file2 /tmp</code> <code>rm -fr /tmp/*</code> <code>mkdir stu1</code> <code>rmdir stu1</code>	(5) <code>chmod g+rw, o+r file1</code> <code>chmod 764 file1</code>
(3) <code>cat file1</code> <code>cat file1 > file2</code> <code>more file2</code> <code>less file1</code> <code>head -n 4 /etc/passwd</code> <code>tail -n 4 file2</code>	(6) <code>tar -czvf usr.tar.gz /home/usr1/lib32</code> <code>tar -xzvf usr.tar.gz</code>

四、实验结果

本实验部分 Linux 常用命令的执行结果如图 2.1 所示。

```
[hlwang@localhost Desktop]$ pwd
/home/hlwang/Desktop
[hlwang@localhost Desktop]$ cd ..
[hlwang@localhost ~]$ pwd
/home/hlwang
[hlwang@localhost ~]$ whereis java
java: /usr/bin/java /etc/java /usr/lib/java /usr/lib64/java /usr/share/java
/usr/share/man/man1/java.1.gz
[hlwang@localhost ~]$ mkdir stu1
[hlwang@localhost ~]$ cd stu1
[hlwang@localhost stu1]$ touch file1
[hlwang@localhost stu1]$ ls -la
total 8
drwxrwxr-x. 2 hlwang hlwang 4096 Sep 26 22:16 .
drwx----- 42 hlwang hlwang 4096 Sep 26 22:16 ..
-rw-rw-r-- 1 hlwang hlwang 0 Sep 26 22:16 file1
[hlwang@localhost stu1]$ cd ..
[hlwang@localhost ~]$ cp ./stu/file1 .
cp: cannot stat './stu/file1': No such file or directory
[hlwang@localhost ~]$ cp ./stu1/file1 .
[hlwang@localhost ~]$ ls file1 -la
-rw-rw-r-- 1 hlwang hlwang 0 Sep 26 22:17 file1
[hlwang@localhost ~]$
```

图 2.1 部分 Linux 常用命令的执行结果



需要说明的是，由于本实验涉及多个 Linux 常用命令的使用，且由于每个系统所含内容不同，因此命令执行后在终端输出的内容也不同。在此，本书不再给出所有命令的执行结果，请读者自行对照命令的使用说明验证命令执行的正确性。

五、实验思考

- (1) 在 Linux 中，图形界面与终端控制台以及各终端控制台之间应如何切换？
- (2) 练习上面没有列出的其他 Linux 常用命令。

2.5 实验 2.2：Linux 下 C 程序的编写

一、实验目的

- (1) 掌握 Linux 下 C 程序的编写、编译与运行方法。
- (2) 掌握 gcc 编译器的编译过程，熟悉编译的各个阶段。
- (3) 熟悉 Makefile 文件的编写格式和 make 编译工具的使用方法。

二、实验内容

练习使用 gcc 编译器编译 C 程序并执行，编写 Makefile 文件，使用 make 工具编译程序并执行。具体要求如下。

(1) 编写简单的 C 程序，功能为在屏幕上输出“Hello gcc!”。利用该程序练习使用 gcc 编译器的 E、S、c、o、g 选项，观察不同阶段所生成的文件，即*.c、*.i、*.s、*.o 文件和可执行文件。

(2) 编写一个由头文件 greeting.h、自定义函数文件 greeting.c、主函数文件 myapp.c 构成的 C 程序，并根据这三个文件的依赖关系编写 Makefile 文件。程序如下：

```
/*-----myapp.c-----*/
#include <stdio.h>
#include "greeting.h"
#define N 10
int main()
{
    char name[N];
    printf("your name, please: ");
    scanf("%s", name);
    greeting(name);
    exit(0);
}
/*-----greeting.h-----*/
#ifndef _GREETING_H
#define _GREETING_H
```




```

void greeting(char *name);
#endif
/*-----greeting.c-----*/
#include <stdio.h>
#include "greeting.h"
void greeting(char *name)
{
    printf("Hello %s", name);
}

```

三、实验指导

对于实验内容 (1), 可将其分为三个步骤: ①创建空文档, 修改名称为 `myhello.c`, 输入程序代码, 保存并退出; ②打开终端, 用 `gcc` 命令对 `myhello.c` 程序进行分阶段编译; ③利用 `ls` 命令查看编译过程中所产生的各个文件, 即 `myhello.i`、`myhello.s`、`myhello.o` 文件和可执行文件 (如 `myhello.c`)。

`myhello.c` 中的示例代码如下:

```

/*-----myhello.c-----*/
#include <stdio.h>
int main()
{
    printf("Hello gcc!\n");
    exit(0);
}

```

对于实验内容 (2), 除了三个源代码文件之外, 最重要的是 `Makefile` 文件的编写, 示例如下:

```

myapp: greeting.o myapp.o
    gcc myapp.o greeting.o -o myapp
greeting.o: greeting.c greeting.h
    gcc -c greeting.c
myapp.o: myapp.c greeting.h
    gcc -c myapp.c
clean:
    rm -rf *.o

```

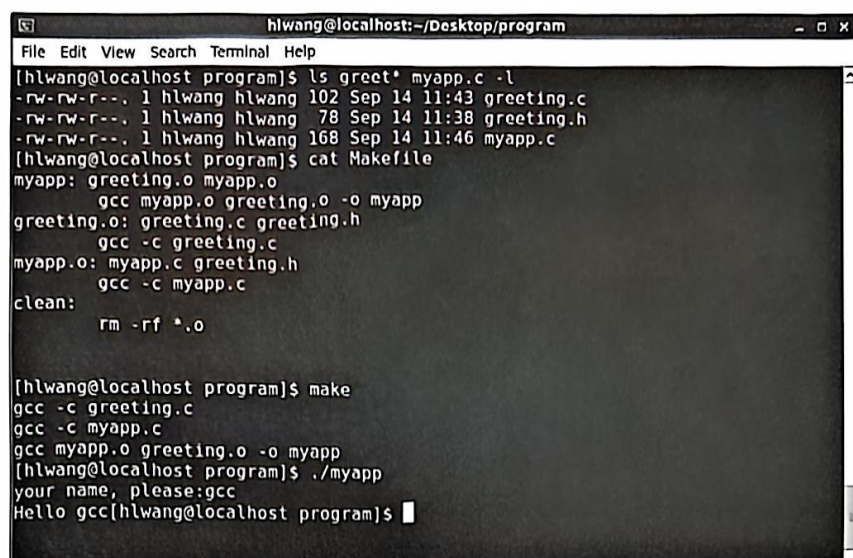
最后使用 `make` 工具编译程序, 即在终端提示符的后面输入 “`make`”, 并按 `Enter` 键。

四、实验结果

实验内容 (1) 的结果可通过 `ls` 命令列出所生成的文件来查看。

实验内容 (2) 的实验结果如图 2.2 所示。





```
hlwang@localhost:~/Desktop/program
File Edit View Search Terminal Help
[hlwang@localhost program]$ ls greet* myapp.c -l
-rw-rw-r--. 1 hlwang hlwang 102 Sep 14 11:43 greeting.c
-rw-rw-r--. 1 hlwang hlwang  78 Sep 14 11:38 greeting.h
-rw-rw-r--. 1 hlwang hlwang 168 Sep 14 11:46 myapp.c
[hlwang@localhost program]$ cat Makefile
myapp: greeting.o myapp.o
    gcc myapp.o greeting.o -o myapp
greeting.o: greeting.c greeting.h
    gcc -c greeting.c
myapp.o: myapp.c greeting.h
    gcc -c myapp.c
clean:
    rm -rf *.o

[hlwang@localhost program]$ make
gcc -c greeting.c
gcc -c myapp.c
gcc myapp.o greeting.o -o myapp
[hlwang@localhost program]$ ./myapp
your name, please: gcc
Hello gcc[hlwang@localhost program]$
```

图 2.2 示例代码的执行结果

五、实验思考

- (1) make 工具的编译原理是什么?
- (2) 如何直接使用 gcc 命令完成 myapp.c、greeting.h、greeting.c 三个文档的编译?

