Linux系统及Shell编程

东北林业大学

卢洋

第三章

Linux C 开发工具

1. 编辑器vi/vim的使用

2. 编译器gcc的使用

3. 调试器gdb的使用

4. 构建工具make的使用

3.4

工程管理器make的使用

前置条件(prerequisites)

- 前置条件通常是一组文件名，之间用空格分隔。它指定了“目标”是否重新构建的判断标准：只要一个前置文件不存在，或者更新过，“目标”就需要重新构建。

```
result.txt: source.txt

    cp source.txt result.txt
```

- 构建result.txt的前置条件时source.txt。如果当前路径下，source.txt已经存在，那么make result.txt可以正常运行；否则必须再写一条规则，用以生成source.txt。

- 再写一条规则:

source.txt:

```
echo "this is the source" > source.txt
```

- source.txt没有前置条件，意味着跟其它文件无关，只要source.txt不存在，每次调用make source.txt，都会生成source.txt。

- 连续执行两次make result.txt。
- 第一次会先创建source.txt，再创建result.txt。
- 第二次执行，make发现source.txt没有更新，就会不执行任何操作。

命令(command)

- 命令(command)表示如何更新目标文件，由一行或多行Shell命令组成；
- 是构建"目标"的具体指令；
- 结果通常是生成目标文件。

- 每行命令之前必须有一个tab键。如果想用其它键，可以用内置变量.RECIPEPREFIX声明。

```
.RECIPEPREFIX=>
all:
>echo 123
```

- 用.RECIPEPREFIX指定大于号(>)替代tab键。

- 每行命令都是在一个单独运行的Shell中执行的，这些Shell间没有继承关系。

var:

```
export foo=bar

echo “foo=[$$foo]”
```

- 解决方法：

(1)；

(2) \

(3) .ONESHELL