makefile\_shell脚本学习CQIU

2017年4月17日星期一

目 录

[1. 模板标题1 3](#_Toc1684283382)

[1.1 模板标题2 3](#_Toc1267391699)

[1.1.1 模板标题3 3](#_Toc397310112)

[1.1.1.1 模板标题4 3](#_Toc476051819)

## 模板标题1

见下文

### 模板标题2

见下文

#### 模板标题3

见下文

##### 模板标题4

见下文

###### 模板标题5

见下文

模板标题6

见下文

模板标题6

见下文

模板标题7

见下文



对于WPS来说，可以从”文件”-->”页面设置”-->”文档网络”，然后选择“无网络”

## Makefile与shell学习

正文见下

### 背景

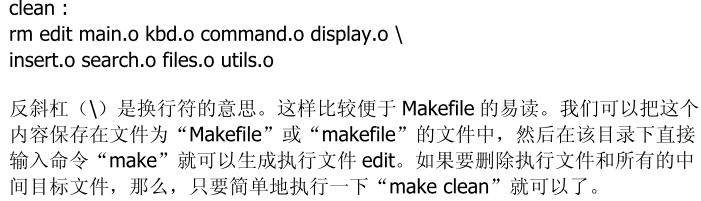
在开发仿真平台过程中，基本原则是能用makefile就用makefile脚本语言，makefile搞不定的地方，可以借助shell脚本进行编程，shell的好处是，可以直接在makefile脚本里面写代码。比调用python，tcl，perl之类的好太多了。

### Makefile

基本介绍见下

#### 几种常见使用技巧

##### 反斜杠用法



#### 局限性

Makefile脚本本身有很多局限性，正则表达式也不够全面，比较烂，不过可以通过借助shell脚本进行编程。

##### 变量不能比较大小

makfile里面定义的变量，不能判断大小哦。

##### 正则表达式支持的很少

Makfile中正则表达式很烂，只有\*,$,^之类的支持，空格目前还不知道怎么支持。

### Shell

这里面介绍的shell，都是能保证shell在makefile中能正常工作的脚本介绍。

#### 几种常见命令配合makefile

##### Grep命令

Linux系统中grep命令是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹 配的行打印出来。grep全称是Global Regular Expression Print，表示全局正则表达式版本，它的使用权限是所有用户。

grep的工作方式是这样的，它在一个或多个文件中搜索字符串模板。如果模板包括空格，则必须被引用，模板后的所有字符串被看作文件名。搜索的结果被送到标准输出，不影响原文件内容。

grep可用于shell脚本，因为grep通过返回一个状态值来说明搜索的状态，如果模板搜索成功，则返回0，如果搜索不成功，则返回1，如果搜索的文件不存在，则返回2。我们利用这些返回值就可进行一些自动化的文本处理工作。

###### 命令格式

grep [option] pattern file

举例说明： grep “^axx” aaa.v

###### 正则表达式

grep的规则表达式，这个表达式在使用时，grep “你的正则表达式” 文件路径

^  #锚定行的开始 如：'^grep'匹配所有以grep开头的行。

$  #锚定行的结束 如：'grep$'匹配所有以grep结尾的行。

.  #匹配一个非换行符的字符 如：'gr.p'匹配gr后接一个任意字符，然后是p。

\*  #匹配零个或多个先前字符 如：'\*grep'匹配所有一个或多个空格后紧跟grep的行。

.\*   #一起用代表任意字符。

[]   #匹配一个指定范围内的字符，如'[Gg]rep'匹配Grep和grep。

[^]  #匹配一个不在指定范围内的字符，如：'[^A-FH-Z]rep'匹配不包含A-R和T-Z的一个字母开头，紧跟rep的行。

\(..\)  #标记匹配字符，如'\(love\)'，love被标记为1。

\<      #锚定单词的开始，如:'\<grep'匹配包含以grep开头的单词的行。

\>      #锚定单词的结束，如'grep\>'匹配包含以grep结尾的单词的行。

x\{m\}  #重复字符x，m次，如：'0\{5\}'匹配包含5个o的行。

x\{m,\}  #重复字符x,至少m次，如：'o\{5,\}'匹配至少有5个o的行。

x\{m,n\}  #重复字符x，至少m次，不多于n次，如：'o\{5,10\}'匹配5--10个o的行。

\w    #匹配文字和数字字符，也就是[A-Za-z0-9]，如：'G\w\*p'匹配以G后跟零个或多个文字或数字字符，然后是p。

\W    #\w的反置形式，匹配一个或多个非单词字符，如点号句号等。

\b    #单词锁定符，如: '\bgrep\b'只匹配grep。

POSIX字符:

为了在不同国家的字符编码中保持一至，POSIX(The Portable Operating System Interface)增加了特殊的字符类，如[:alnum:]是[A-Za-z0-9]的另一个写法。要把它们放到[]号内才能成为正则表达式，如[A- Za-z0-9]或[[:alnum:]]。在linux下的grep除fgrep外，都支持POSIX的字符类。

[:alnum:]    #文字数字字符

[:alpha:]    #文字字符

[:digit:]    #数字字符

[:graph:]    #非空字符（非空格、控制字符）

[:lower:]    #小写字符

[:cntrl:]    #控制字符

[:print:]    #非空字符（包括空格）

[:punct:]    #标点符号

[:space:]    #所有空白字符（新行，空格，制表符）

[:upper:]    #大写字符

[:xdigit:]   #十六进制数字（0-9，a-f，A-F）

###### 命令参数

-a   --text   #不要忽略二进制的数据。

-A<显示行数>   --after-context=<显示行数>   #除了显示符合范本样式的那一列之外，并显示该行之后的内容。

-b   --byte-offset   #在显示符合样式的那一行之前，标示出该行第一个字符的编号。

-B<显示行数>   --before-context=<显示行数>   #除了显示符合样式的那一行之外，并显示该行之前的内容。

-c    --count   #计算符合样式的列数。

-C<显示行数>    --context=<显示行数>或-<显示行数>   #除了显示符合样式的那一行之外，并显示该行之前后的内容。

-d <动作>      --directories=<动作>   #当指定要查找的是目录而非文件时，必须使用这项参数，否则grep指令将回报信息并停止动作。

-e<范本样式>  --regexp=<范本样式>   #指定字符串做为查找文件内容的样式。

-E      --extended-regexp   #将样式为延伸的普通表示法来使用。

-f<规则文件>  --file=<规则文件>   #指定规则文件，其内容含有一个或多个规则样式，让grep查找符合规则条件的文件内容，格式为每行一个规则样式。

-F   --fixed-regexp   #将样式视为固定字符串的列表。

-G   --basic-regexp   #将样式视为普通的表示法来使用。

-h   --no-filename   #在显示符合样式的那一行之前，不标示该行所属的文件名称。

-H   --with-filename   #在显示符合样式的那一行之前，表示该行所属的文件名称。

-i    --ignore-case   #忽略字符大小写的差别。

-l    --file-with-matches   #列出文件内容符合指定的样式的文件名称。

-L   --files-without-match   #列出文件内容不符合指定的样式的文件名称。

-n   --line-number   #在显示符合样式的那一行之前，标示出该行的列数编号。

-q   --quiet或--silent   #不显示任何信息。

-r   --recursive   #此参数的效果和指定“-d recurse”参数相同。

-s   --no-messages   #不显示错误信息。

-v   --revert-match   #显示不包含匹配文本的所有行。

-V   --version   #显示版本信息。

-w   --word-regexp   #只显示全字符合的列。

-x    --line-regexp   #只显示全列符合的列。

-y   #此参数的效果和指定“-i”参数相同。

### 仿真平台脚本

### Caffe中Makefile脚本

#### 模板标题3

见下文

##### 模板标题4

见下文

###### 模板标题5

见下文

模板标题6

见下文

模板标题6

见下文

模板标题7

见下文