

追根溯源，重走嵌入式之路。



RSS 订阅

+ 加关注 发私信

译文: 0篇 评论: 9条

感想杂谈 (1)

-exec: find命令对匹配的文件执行该参数所给出的shell命令，相应的命令形式为

文章存档

2014年05月 (1)
2014年04月 (1)
2014年02月 (1)
2013年04月 (2)
2013年03月 (1)

↓展开

阅读排行

signal函数 (5894)
【转】ARM获得PC指针 (2035)
网络编程socket之accept (1967)
话说C语言const用法 (1822)
如何修改默认的FTP帐号 (1613)
Hello Qt (在Linux下编写 (1521)
uid_t gid_t等的定义 (1322)
DM9000布线 (1316)
论ARMv7 Thumb-2指令集 (1269)
AVR Mega16的熔丝位用 (1224)

评论排行

mov指令的操作数的取值 (2)
【转】ARM获得PC指针 (2)
QWidget 成员函数 (1)
signal函数 (1)
DM9000布线 (1)
'undefined reference to `__ctype_u010850027: 谢谢分享 学习了 (*^__^*)'
【转】ARM获得PC指针 (1)
关于韦东山书上的裸机程 (0)
assert()函数用法总结 (0)
论ARMv7 Thumb-2指令集 (0)

推荐文章

* Qt Quick里的粒子系统
* Android proguard 详解
* 我的2014---感悟程序员职场
* 前端开发规范之项目架构
* 我的2014年总结 - 一些失败的面试经历
* 最简单的基于FFmpeg的编码器-纯净版

最新评论

'undefined reference to `__ctype_u010850027: 谢谢分享 学习了 (*^__^*)'
QWidget 成员函数ItemAt与dx0829: 表示也是这样
mov指令的操作数的取值范围到底是多少?
@MengShuaiQi520: 0x53是十六进制数, 表示成二进制是0b01010011, 所以需要循...
mov指令的操作数的取值范围到底是多少?
MengShuaiQi520: 0x53循环右移8位, 就得到了0x53000000, 这句话没看懂, 为什么是循环右移8位呢, 不是循环右...
【转】ARM获得PC指针为何PC=a13767414103: ARM7的三级流水线, PC=PC+8, ARM9的五级

'command' { } \; (注意{ }和\之间的空格)

-ok 和 -exec的作用相同, 只不过以一种更为安全的模式来执行该参数所给出的shell命令, 在执行每一个命令之前, 都会给出提示, 让用户来确定是否执行。

options有如下几种:

-name : 按照文件名查找文件

-perm : 按照文件权限来查找文件

-user : 按照文件属主来查找文件

-group : 按照文件所属的组来查找文件

-mtime -n +n 按照文件的更改时间来查找文件, -n表示文件更改时间距现在n天以内, +n表示文件更改时间距现在n天以前。find命令还有-atime 和-ctime选项, 但它们都和-mtime选项相似。

-size n[c]查找文件长度为n块的文件, 带有c时表示文件长度以字节计。

-nogroup 查找无有效所属组的文件, 即该文件所属的组在/etc/groups中不存在

-newer file1 !file2查找更改时间比文件file1新但比文件file2旧的文件

-depth 先查找指定目录有无匹配文件, 若无则再在子目录中查找

-type 查找某一类型的文件, 如

b :块设备文件

d: 目录

e: 字符设备文件

p; 管道文件

l: 符号链接文件

f: 普通文件

(3) find命令举例

find -name "*.txt" -print 查找txt结尾的文件并输出到屏幕上

find /cmd ".sh" -print 查找/cmd目录下所有sh文件, 并输出

find . -perm 755 -print 查找当前目录下权限为755的文件, 并输出

find `pwd` -user root -print 查找当前目录下属主为root的文件, 并输出

find ./ -group sunwill -print 查找当前目录下所属主是sunwill的文件

find /var -mtime -5 -print 查找/var目录下更改时间为5天内的所有文件

find /var -mtime +5 -print 查找/var目录下更改时间为5天以前的所有文件

find /var -newer "myfile1" ! -newer "myfile2" -print 查找/var目录下比myfile1新, 但是比myfile2旧的所有文件。

find /var -type d -print 查找/var目录下所有目录

find /var -type l -print 查找/var目录下所有的符号链接文件。

find . -size +1000000c -print 查找当前目录下大于1000000字节的文件

find / -name "con.file" -depth -print 查找根目录下有无"con.file",若无则在其子目录中查找

find . -type f -exec ls -l {} \; 查找当前目录下是否有普通文件, 若有则执行ls -l

(4) xargs命令

在使用find命令的-exec选项处理匹配到的文件时, find命令将所有匹配到的文件一起传递给exec。不幸的是, 有些系统对能够传递给exec的命令长度有限制, 这样find命令运行几分钟之后就出现溢出错误。错误信息通常是“参数列太长”或“参数列溢出”。这就是xargs的用处所在, 特别是与 find命令一起使用,exec会发起多个进程, 而xargs会多个, 只有一个

find ./ -perm -7 -print | xargs chmod o-w 查找权限为7的文件并传递给chmod处理

3.grep介绍

(1)grep的一般格式为 grep [options] 基本正则表达式 [文件]

字符串参数最好采用是双引号括, 一是以防被误解为shell命令, 二是可以用来查找多个单词组成的字符串

-c: 只输出匹配行的记数

-i: 不区分大小写 (只适用于单个字符)

-h: 查询多个文件时不显示文件名

-H: 只显示文件名

-l: 查询多文件时只输出包含匹配字符的文件名

-n: 只显示匹配行及其行号

-s: 不显示不存在或无匹配文本的错误信息。

-v: 显示不包含匹配文本的所有行。

(2) 举例说明:

grep ^[210] myfile 匹配myfile中以非2、1、0开头的行

grep "[5-8][6-9][0-3]" myfile 匹配myfile中第一位为5|6|7|8, 第二位6|7|8|9, 第三位为0|1|2|3的三个字符的行

grep "4{2,4}" myfile 匹配myfile中含有44,444或4444的行

流水线，也是PC=PC+8，根本的原因是，两者...

【转】ARM获得PC指针为何PC=a13767414103: 其实很简单，一句话PC=PC+(取指第几步-1)*4;4代表4字节。其中的含义大家慢慢去体会吧。

【转】ARM获得PC指针为何PC=a13767414103: 看帖不回贴，这是不好滴。

DM9000布线
ding6078051: 芯片出来的RX+/TX+,RX-/TX-为差分线对（千万别走成RX+/RX-和TX+/TX-，否则你...

signal函数
cl1180: 坑爹！！
#include#include...什么意思代码里面的很多变量都没声明。。。所以，是怎么用...

grep "\?" myfile匹配myfile中含有任意字符的行

(3) grep命令类名

[:upper:] 表示[A-Z]

[:alnum:] 表示[0-9a-zA-Z]

[:lower:] 表示[a-z]

[:space:] 表示空格或者tab键

[:digit:] 表示[0-9]

[:alpha:] 表示[a-zA-Z]

如: grep "5[:digit:][:digit:]" myfile 匹配myfile中含有5开头接下去两位都是数字的行。

4.awk介绍

可以从文件或字符串中基于指定规则浏览和抽取信息，是一种自解释的变成语言。

(1) awk命令行方式 awk [-F filed-spearator] 'command' input-files

awk脚本：所有awk命令插入一个文件，并使awk程序可执行，然后用awk命令解释器作为脚本的首行，以便通过键入脚本名称来调用它。awk脚本是由各种操作和模式组成。

模式部分决定动作语句何时触发及触发事件。(BEGIN,END)

动作对数据进行处理，放在{}内指明 (print)

(2) 分隔符、域和记录

awk执行时，其浏览域标记为\$1,\$2,...\$n.这种方法成为域标识。\$0为所有域。

(3) 举例说明:

awk '{print \$0}' test.txt |tee test.out 输出test.txt中所有行\$0表示所有域

awk -F : '{print \$1}' test.txt |tee test.out' 同上。。只是分隔符为":"

awk 'BEGIN {print "IPDate\n"}{print \$1 "\t" \$4} END{print "end-of-report"}' test.txt

开始时打印"IPDate"结束时打印"end-of-report"中间打印主体信息，比如总共匹配三条信息，则输出如下：

IPDate

1 first

2 second

3 third

end-of-report

(4) 匹配操作符 ~ 匹配,!~ 不匹配

cat test.txt |awk '\$0~/210.34.0.13/' 匹配test.txt中为210.34.0.13的行

awk '\$0!~/210.34.0.13' test.txt 匹配test.txt中不是210.34.0.13的行

awk '{if(\$1=="210.34.0.13") print \$0}' test.txt 匹配 test.txt中第一个域为210.34.0.13的行。

5.sed介绍

sed不与初始化文件打交道，它操作的只是一个拷贝，然后所有的改动如果没有重定向到一个文件，将输出到屏幕。

sed是一种很重要的文本过滤工具，使用一行命令或者使用管道与grep与awk相结合。是一种非交互性文本流编辑。

(1) 调用sed的三种方式

使用sed命令行格式为：sed [options] sed命令 输入文件

使用sed脚本文件格式为：sed[options] -f sed脚本文件 输入文件

sed脚本文件[options] 输入文件

--不管是使用shell命令行方式或脚本文件方式，如果没有指定输入文件，sed从标准输入中接受输入，一般是键盘或重定向结果。

(2) sed 命令的options如下

-n: 不打印

-c: 下一命令是编辑命令

-f: 如果正在调用sed脚本文件

(3) sed在文件中查询文本的方式

--使用行号，可以是一个简单的数字，或是一个行号的范围

--使用正则表达式

(4) 读取文本的方式

x x为一行号

x,y 表示行号范围从x到y

/pattern/ 查询包含模式的行

/pattern/pattern/ 查询包含两个模式的行

pattern/x 在给定的行号上查询包含模式的行

x,/pattern/ 通过行号和模式查询匹配行

x,y! 查询不包含指定行号x和y的行

(5) 基本sed编辑命令

- p 打印匹配行
- d 删除匹配行
- = 显示文件行号
- a\ 在定位行号后附加新文本信息
- i\ 在定位行号后插入新文本信息
- c\ 用新文本替换定位文本
- s 使用替换模式替换相应模式
- r 从另一个文件中读文件
- w 写文本到一个文件
- q 第一个模式匹配完成后推出或立即退出
- l 显示与八禁止ASCII代码等价的控制字符
- { } 在定位行执行的命令组
- n 从另一个文件中读文本下一行，并附加在下一行
- g 将模式2粘贴到/pattern n/
- y 传送字符

(6) 举例说明：

sed -n '2p' test.txt 打印第二行的信息(注意：-n是不打印不匹配的信息，若没加-n，则打印文件的所有信息而不是匹配信息)

sed -n '1,4p' test.txt 打印第一行到第四行的信息

sed -n '/los/p' test.txt 模式匹配los，并打印出来

sed -n '2,/los/p' test.txt 从第二行开始。。知道匹配第一个los

sed -n '/^\$/p' test.txt 匹配空行

sed -n -e '/^\$/p' -e '/^\$/=' test.txt 打印空行及行号

sed -n '/good/a\morning' test.txt 在匹配到的good后面附加morning

sed -n '/good/i\morning' test.txt 在匹配到的good前面插入morning

sed -n '/good/c\morning' test.txt 将匹配到的good替换成morning

sed '1,2d' test.txt 删除第1和2行

sed 's/good/good morning/g' test.txt 匹配good并替换成goodmorning

send 's/good/& hello /p' test.txt 匹配到good就在其后面加上hello

send 's/good/ hello &/p' test.txt 匹配到good就在其前面加上hello

6. 合并与分割 (sort,uniq,join,cut,paste,split)

(1) sort命令

sort [options] files 许多不同的域按不同的列顺序排序

-c 测试文件是否已经排序

-m 合并两个排序文件

-u 删除所有同样行

-o 存储sort结果的输出文件名

-t 域分隔符，用非空格或tab开始排序

+n : n 为列号，使用此列号开始排序

-n 指定排序是域上的数字分类项

-r 比较求逆

sort -c test.txt 测试文件是否分类过

sort -u test.txt 排序并合并一样的行

sort -r test.txt 以相反的顺序排列

sort -t "/" +2 test.txt 以"/"分隔，第二个域开始分类

(2) uniq命令

uniq [options] files 从一个文本文件中去除或禁止重复行

-u 只显示不重复行

-d 只显示有重复数据行，每种重复行只显示其中一行

-c 打印每一重复行出现次数

-f : n为数字，前n个域被忽略

uniq -f 2 test.txt 忽略前2个域

(3) join 命令

join [options] file1 file2 用来将来自两个分类文本文件的行连在一起

-an, n为一数字，用于连接时从文件n中显示不匹配行

-onm, 连接域, n为文件号, m为域号
-jnm, n为文件号, m为域号, 使用其他域作连接域
-t, 域分隔符。用来设置非空格或tab键的域分隔符。

(4) split命令

split -output_file_size input_filename output_filename

用来将大文件分割成小文件。

- b n, 每个分割文件的大小n
 - C n, 每个分割文件一行最多n字节
 - l n, 每个分割文件的行数
 - n, 同-l n
- split -10 test.txt 将test.txt分割成10行的小文件

(5) cut 命令

cut -c n1-n2 filename 显示每行从开头算起 n1 到 n2 的文字。

cut -c 3-5 test.txt 显示test.txt中每行从第3到第5个字符



▲ 上一篇

'undefined reference to `__ctype_tolower' 或 `__ctype_b'

▼ 下一篇

uid_t gid_t等的定义

主题推荐

正则表达式

文件系统

合并

command

标准

猜你在找

查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点, 不代表CSDN网站的观点或立场

核心技术类目

全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker

OpenStack

VPN

Spark

ERP

IE10

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC

WAP

jQuery

BI

HTML5

Spring

Apache

.NET

API

HTML

SDK

IIS

Fedora

XML

LBS

Unity

Splashtop

UML

components

Windows Mobile

Rails

QEMU

KDE

Cassandra

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

SpringSide

Maemo

Compuware

大数据

apttech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

HBase

Pure

Solr

Angular

Cloud Foundry

Redis

Scala

Django

Bootstrap