# 目录

# Linux使用和开发心得

- 最新版本：`v1.0`
- 更新时间：`20210717`

# 简介

介绍Linux系统的基本概念，以及常用命令，开发相关的一些心得。

# 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

## Gitbook源码

- crifan/linux_usage_dev_summary: Linux使用和开发心得

## 如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

## 在线浏览

- Linux使用和开发心得 book.crifan.com
- Linux使用和开发心得 crifan.github.io

## 离线下载阅读

- Linux使用和开发心得 PDF
- Linux使用和开发心得 ePub
- Linux使用和开发心得 Mobi

# 版权说明



# 鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

# 更多其他电子书

本人 `crifan` 还写了其他 `100+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

# Linux系统概述

TODO:

grep command in Unix/Linux - GeeksforGeeks https://www.geeksforgeeks.org/grep-command-in-unixlinux/ The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for globally search for regular expression and print out).

grep(1) - Linux manual page https://man7.org/linux/man-pages/man1/grep.1.html grep, egrep, fgrep - print lines that match patterns

sed, a stream editor https://www.gnu.org/software/sed/manual/sed.html sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). While in some ways similar to an editor which permits scripted edits (such as ed), sed works by making only one pass over the input(s), and is consequently more efficient. But it is sed's ability to filter text in a pipeline which particularly distinguishes it from other types of editors.

Sed Command in Linux/Unix with examples - GeeksforGeeks https://www.geeksforgeeks.org/sed-command-in-linux-unix-with-examples/ SED command in UNIX is stands for stream editor and it can perform lot's of function on file like, searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX is for substitution or for find and replace. By using SED you can edit files even without opening it, which is much quicker way to find and replace something in file, than first opening that file in VI Editor and then changing it.

SED is a powerful text stream editor. Can do insertion, deletion, search and replace(substitution). SED command in unix supports regular expression which allows it perform complex pattern matching.

Viewing text files on Linux - cat, head, tail, more and less · 2buntu https://2buntu.com/articles/1491/viewing-text-files-on-linux-cat-head-tail-more-and-less/

Short version head - View the top few lines of a file - - Displays the first number of lines of a file tail - View the bottom few lines of a file - - Displays the last number of lines of a file -f : continually watch for any additions at the end of the file -f --pid=PID - continually display any additions until process with PID terminates -f -s - continually display any additions at intervals of seconds cat - View the whole file - n : line-numbered output -b : line-numbered output with no line numbers for blank lines -s : multiple blank lines compressed into a single blank line more - View the whole file, one screenful at a time. spacebar : View next screen b : View previous screen d : View next half-screen Enter : View next line = : Current line number in file v : Start vi editor on current line /string : Search for string in file n : Go to next occurrence of string ' : Go to first occurrence of string less - Same as more, but

with many more features. Displays the portion of the file without waiting for the entire file to be read by it. Accepts most of the more commands. Pg Dn : View next screen Pg Up : View previous screen Up arrow : View previus line Down arrow : View next line

Manage Files Effectively using head, tail and cat Commands in Linux https://www.tecmint.com/view-contents-of-file-in-linux/

14 tail and head commands in Linux/Unix - Linux.com https://www.linux.com/training-tutorials/14-tail-and-head-commands-linuxunix/

tail(1) - Linux manual page https://man7.org/linux/man-pages/man1/tail.1.html tail - output the last part of files

head(1) - Linux manual page https://man7.org/linux/man-pages/man1/head.1.html

```
    head – output the first part of files
```

The head and tail commands in LINUX | Baeldung on Linux https://www.baeldung.com/linux/head-tail-commands

Coreutils - GNU core utilities https://www.gnu.org/software/coreutils/ Coreutils - GNU core utilities The GNU Core Utilities are the basic file, shell and text manipulation utilities of the GNU operating system. These are the core utilities which are expected to exist on every operating system.

2 Common options https://www.gnu.org/software/coreutils/manual/coreutils.html#toc-Common-options-1

‘--help’ Print a usage message listing all available options, then exit successfully.

‘--version’ Print the version number, then exit successfully.

‘--’ Delimit the option list. Later arguments, if any, are treated as operands even if they begin with ‘-’. For example, ‘sort -- -r’ reads from the file named -r.

‘-S suffix’ ‘--suffix=suffix’ Append suffix to each backup file made with -b. If this option is not specified, the value of the SIMPLE_BACKUP_SUFFIX environment variable is used. And if SIMPLE_BACKUP_SUFFIX is not set, the default is ‘~’, just as in Emacs.

GNU Coreutils https://www.gnu.org/software/coreutils/manual/coreutils.html#toc-Introduction-1

27.2 Symbolic Modes https://www.gnu.org/software/coreutils/manual/coreutils.html#Symbolic-Modes

Symbolic modes represent changes to files’ mode bits as operations on single-character symbols. They allow you to modify either all or selected parts of files’ mode bits, optionally based on their previous values, and perhaps on the current umask as well (see Umask and Protection).

The format of symbolic modes is:

[ugoa…][-+=]perms…[,…] where perms is either zero or more letters from the set ‘rwxXst’, or a single letter from the set ‘ugo’.

27.2.1 Setting Permissions

users operation permissions

u the user who owns the file;

g other users who are in the file’s group;

o all other users;

a all users; the same as ‘ugo’.

The operation part tells how to change the affected users’ access to the file, and is one of the following symbols:

+ to add the permissions to whatever permissions the users already have for the file;

- to remove the permissions from whatever permissions the users already have for the file;

= to make the permissions the only permissions that the users have for the file.

The permissions part tells what kind of access to the file should be changed; it is normally zero or more of the following letters. As with the users part, the order does not matter when more than one letter is given. Omitting the permissions part is useful only with the ‘=’ operation, where it gives the specified users no access at all to the file.

r the permission the users have to read the file;

w the permission the users have to write to the file;

x the permission the users have to execute the file, or search it if it is a directory.

For example, to give everyone permission to read and write a regular file, but not to execute it, use:

a=rw To remove write permission for all users other than the file’s owner, use:

go-w The above command does not affect the access that the owner of the file has to it, nor does it affect whether other users can read or execute the file.

To give everyone except a file’s owner no permission to do anything with that file, use the mode below. Other users could still remove the file, if they have write permission on the directory it is in.

go= Another way to specify the same thing is:

og-rwx

The head and tail commands in LINUX | Baeldung on Linux
https://www.baeldung.com/linux/head-tail-commands

Coreutils - GNU core utilities https://www.gnu.org/software/coreutils/ GNU coreutils - Core GNU utilities - GNU Project - Free Software Foundation https://www.gnu.org/software/coreutils/manual/ -> 所有的GNU的core util命令: https://www.gnu.org/software/coreutils/manual/coreutils.html#toc-Introduction-1

```
3 Output of entire files
3.1 cat: Concatenate and write files
3.2 tac: Concatenate and write files in reverse
3.3 nl: Number lines and write files
3.4 od: Write files in octal or other formats
3.5 base32: Transform data into printable data
3.6 base64: Transform data into printable data
3.7 basenc: Transform data into printable data
4 Formatting file contents
4.1 fmt: Reformat paragraph text
4.2 pr: Paginate or columnate files for printing
4.3 fold: Wrap input lines to fit in specified width
5 Output of parts of files
5.1 head: Output the first part of files
5.2 tail: Output the last part of files
5.3 split: Split a file into pieces.
5.4 csplit: Split a file into context-determined pieces
6 Summarizing files
6.1 wc: Print newline, word, and byte counts
6.2 sum: Print checksum and block counts
6.3 cksum: Print CRC checksum and byte counts
6.4 b2sum: Print or check BLAKE2 digests
6.5 md5sum: Print or check MD5 digests
6.6 sha1sum: Print or check SHA-1 digests
6.7 sha2 utilities: Print or check SHA-2 digests
7 Operating on sorted files
7.1 sort: Sort text files
7.2 shuf: Shuffling text
7.3 uniq: Uniquify files
7.4 comm: Compare two sorted files line by line
7.5 ptx: Produce permuted indexes
7.5.1 General options
7.5.2 Charset selection
7.5.3 Word selection and input processing
7.5.4 Output formatting
7.5.5 The GNU extensions to ptx
7.6 tsort: Topological sort
7.6.1 tsort: Background
8 Operating on fields
8.1 cut: Print selected parts of lines
8.2 paste: Merge lines of files
8.3 join: Join lines on a common field
8.3.1 General options
8.3.2 Pre-sorting
8.3.3 Working with fields
8.3.4 Controlling join's field matching
8.3.5 Header lines
8.3.6 Union, Intersection and Difference of files
9 Operating on characters
9.1 tr: Translate, squeeze, and/or delete characters
9.1.1 Specifying sets of characters
9.1.2 Translating
9.1.3 Squeezing repeats and deleting
9.2 expand: Convert tabs to spaces
9.3 unexpand: Convert spaces to tabs
10 Directory listing
10.1 ls: List directory contents
10.1.1 Which files are listed
10.1.2 What information is listed
10.1.3 Sorting the output
10.1.4 General output formatting
10.1.5 Formatting file timestamps
10.1.6 Formatting the file names
10.2 dir: Briefly list directory contents
10.3 vdir: Verbosely list directory contents
10.4 dircolors: Color setup for ls
11 Basic operations
11.1 cp: Copy files and directories
11.2 dd: Convert and copy a file
11.3 install: Copy files and set attributes
```

```
11.4 mv: Move (rename) files
11.5 rm: Remove files or directories
11.6 shred: Remove files more securely
12 Special file types
12.1 link: Make a hard link via the link syscall
12.2 ln: Make links between files
12.3 mkdir: Make directories
12.4 mkfifo: Make FIFOs (named pipes)
12.5 mknod: Make block or character special files
12.6 readlink: Print value of a symlink or canonical file name
12.7 rmdir: Remove empty directories
12.8 unlink: Remove files via the unlink syscall
13 Changing file attributes
13.1 chown: Change file owner and group
13.2 chgrp: Change group ownership
13.3 chmod: Change access permissions
13.4 touch: Change file timestamps
14 Disk usage
14.1 df: Report file system disk space usage
14.2 du: Estimate file space usage
14.3 stat: Report file or file system status
14.4 sync: Synchronize cached writes to persistent storage
14.5 truncate: Shrink or extend the size of a file
15 Printing text
15.1 echo: Print a line of text
15.2 printf: Format and print data
15.3 yes: Print a string until interrupted
16 Conditions
16.1 false: Do nothing, unsuccessfully
16.2 true: Do nothing, successfully
16.3 test: Check file types and compare values
16.3.1 File type tests
16.3.2 Access permission tests
16.3.3 File characteristic tests
16.3.4 String tests
16.3.5 Numeric tests
16.3.6 Connectives for test
16.4 expr: Evaluate expressions
16.4.1 String expressions
16.4.2 Numeric expressions
16.4.3 Relations for expr
16.4.4 Examples of using expr
17 Redirection
17.1 tee: Redirect output to multiple files or processes
18 File name manipulation
18.1 basename: Strip directory and suffix from a file name
18.2 dirname: Strip last file name component
18.3 pathchk: Check file name validity and portability
18.4 mktemp: Create temporary file or directory
18.5 realpath: Print the resolved file name.
18.5.1 Realpath usage examples
19 Working context
19.1 pwd: Print working directory
19.2 stty: Print or change terminal characteristics
19.2.1 Control settings
19.2.2 Input settings
19.2.3 Output settings
19.2.4 Local settings
19.2.5 Combination settings
19.2.6 Special characters
19.2.7 Special settings
19.3 printenv: Print all or some environment variables
19.4 tty: Print file name of terminal on standard input
20 User information
20.1 id: Print user identity
20.2 logname: Print current login name
20.3 whoami: Print effective user ID
20.4 groups: Print group names a user is in
20.5 users: Print login names of users currently logged in
20.6 who: Print who is currently logged in
21 System context
```

```
21.1 date: Print or set system date and time
21.1.1 Time conversion specifiers
21.1.2 Date conversion specifiers
21.1.3 Literal conversion specifiers
21.1.4 Padding and other flags
21.1.5 Setting the time
21.1.6 Options for date
21.1.7 Examples of date
21.2 arch: Print machine hardware name
21.3 nproc: Print the number of available processors
21.4 uname: Print system information
21.5 hostname: Print or set system name
21.6 hostid: Print numeric host identifier
21.7 uptime: Print system uptime and load
22 SELinux context
22.1 chcon: Change SELinux context of file
22.2 runcon: Run a command in specified SELinux context
23 Modified command invocation
23.1 chroot: Run a command with a different root directory
23.2 env: Run a command in a modified environment
23.2.1 General options
23.2.2 -S/--split-string usage in scripts
Testing and troubleshooting
23.2.3 -S/--split-string syntax
Splitting arguments by whitespace
Escape sequences
Comments
Environment variable expansion
23.3 nice: Run a command with modified niceness
23.4 nohup: Run a command immune to hangups
23.5 stdbuf: Run a command with modified I/O stream buffering
23.6 timeout: Run a command with a time limit
24 Process control
24.1 kill: Send a signal to processes
25 Delaying
25.1 sleep: Delay for a specified time
26 Numeric operations
26.1 factor: Print prime factors
26.2 numfmt: Reformat numbers
26.2.1 General options
26.2.2 Possible units:
26.2.3 Examples of using numfmt
26.3 seq: Print numeric sequences
```

TODO：整理出 常用的命令， 不常用的，就暂时不整理了。

cron - Wikipedia https://en.wikipedia.org/wiki/Cron

Cron - 维基百科，自由的百科全书 https://zh.wikipedia.org/wiki/Cron

Linux Shell Commands https://docs.cs.cf.ac.uk/notes/linux-shell-commands/

rsync - Wikipedia https://en.wikipedia.org/wiki/Rsync

Use Similar to cp, rcp and scp, rsync requires the specification of a source and of a destination, of which at least one must be local.[19]

Generic syntax:

rsync [OPTION] … SRC … [USER@]HOST:DEST rsync [OPTION] … [USER@]HOST:SRC [DEST]

rsync(1) - Linux man page https://linux.die.net/man/1/rsync rsync -- a fast, versatile, remote (and local) file-copying tool

Rsync (Remote Sync): 20 Helpful Examples in Linux
https://phoenixnap.com/kb/rsync-command-linux-examples

# 典型使用场景

# Linux基础概念

# 目录结构

# 启动脚本

Linux类系统中，典型的启动脚本文件

- `bash`：`~/.bash_profile`
- `zsh`：`~/.zshrc`
- `ksh`：`~/.kshrc`
- `other`：`~/.profile`

# .bashrc

# .zshrc

# 发行版

# Linux常用命令

# Linux命令概览

此处，给出部分，相对常用的Linux命令的简要介绍：

- `ar` — archive library builder
- `bzip2` — bzip2 command for distribution generation
- `bunzip2` — bunzip2 command for distribution checking
- `chmod` — change permissions on a file
- `cat` — output concatenation utility
- `cp` — copy files
- `date` — print the current date/time
- `echo` — print to standard output
- `egrep` — extended regular expression search utility
- `find` — find files/dirs in a file system
- `grep` — regular expression search utility
- `gzip` — gzip command for distribution generation
- `gunzip` — gunzip command for distribution checking
- `install` — install directories/files
- `mkdir` — create a directory
- `mv` — move (rename) files
- `ranlib` — symbol table builder for archive libraries
- `rm` — remove (delete) files and directories
- `sed` — stream editor for transforming output
- `sh` — Bourne shell for make build scripts
- `tar` — tape archive for distribution generation
- `test` — test things in file system
- `unzip` — unzip command for distribution checking
- `zip` — zip command for distribution generation

# 系统信息

# uname

# arch

- `arch`
  - 功能：显示机器的处理器架构
  - 举例
    - Mac

```
~ □ arch
i386
```

# date

- `date`
  - 作用：在终端中显示当前时间
  - 举例
    - 显示时间

      ```
      # date
      Wed Aug 28 17:01:10 CST 2019
      ```

    - 加上格式

      ```
      # date +%Y%m%d_%H:%M:%S
      20201031_12:04:49
      ```

    - 如果格式中间有空格，则需要加上双引号

      ```
      # date "+%Y%m%d %H:%M:%S"
      20201031 12:03:57
      ```

# 用户

# who

# whoami

# users

# 帮助信息

# man

# info

# 文件和文件夹

# ls

# tree

# cp

# touch

# mv

# rm

# find

- `find`
  - 功能：查找字符串

# 用途举例

## Mac中统计代码行数

```
find . -name '*.md' | xargs wc -m
```

效果：可以统计出，每一个 `md` 文件的字符 `character` 个数，并且给出总数。

此方法，也适用于其他Linux类系统。

# 权限

# chown

# chmod

# chgrp

# 文件夹

# pwd

# cd

# mkdir

# rmdir

# 文件

# wc

# file

# install

# 显示

把文件内容输出显示，用于查看文件的相关内容。Linux中有很多工具。

- 查看文件的开始或末尾的内容： `head` 和 `tail`

# cat

# head

举例：

查看文本文件最开始的内容，默认行数：`10` 行

```
head file_name
```

实际用途举例：

当文件改动后，去看文件最开头的几行的内容是否变化了：

```
# head uploadToOSS.py
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# Function: Upload to OSS for course video(.mp4) & subtitle(.srt) & cover(.jpg
#   Note:
#       1. also need update Mongod after upload OSS
#       2. video: before upload to OSS, need remove water mark
#       3. TODO: add support for user show
# Updated: 20190926


import time
```

->此处确认内容的确从

`Updated: 20190924`

变化为刚修改的：

`Updated: 20190926`

如果要制定输出的函数，用：`-n`

举例：

```
head -n 40
```

输出效果：

```
# tail -n 30 uploadToOSS.py
    processCourseVideo()
    processAllVideoTime = calcTimeEnd("processAllVideo")
    logging.debug("processAllVideoTime=%s", processAllVideoTime)
    allVideoFileSize = getFileFolderSize(CourseRootFolder)
    logging.debug("allVideoFileSize=%s", allVideoFileSize)


    gSummaryInfo["all"] = gCurNum
    gSummaryInfo["processed"]["total"] = {
        "videoNum": gTotalVideoNum,
        "processTime": processAllVideoTime,
        "fileSize": allVideoFileSize,
        "fileSizeStr": formatSize(allVideoFileSize),
    }


    averageFileSize = 0
    averageProcessTime = 0.0
    if gTotalVideoNum > 0:
        averageFileSize = allVideoFileSize/gTotalVideoNum
        logging.debug("averageFileSize=%s", averageFileSize)
        averageProcessTime = processAllVideoTime/gTotalVideoNum
        logging.debug("averageProcessTime=%s", averageProcessTime)
    gSummaryInfo["processed"]["average"] = {
        "processTime": averageProcessTime,
        "fileSize": averageFileSize,
        "fileSizeStr": formatSize(averageFileSize),
    }


    saveJsonToFile(gSummaryInfoFullPath, gSummaryInfo)
    gSummaryInfoStr = jsonToPrettyStr(gSummaryInfo)
    logging.debug("gSummaryInfoStr=\n%s", gSummaryInfoStr)
```

# tail

查看文本文件的最末尾的内容，默认行数： `10` 行

```
tail file_name
```

举例：

```
# tail uploadToOSS.py
        logging.debug("averageProcessTime=%s", averageProcessTime)
    gSummaryInfo["processed"]["average"] = {
        "processTime": averageProcessTime,
        "fileSize": averageFileSize,
        "fileSizeStr": formatSize(averageFileSize),
    }


    saveJsonToFile(gSummaryInfoFullPath, gSummaryInfo)
    gSummaryInfoStr = jsonToPrettyStr(gSummaryInfo)
    logging.debug("gSummaryInfoStr=\n%s", gSummaryInfoStr)
```

如果要指定函数，用 `-n`

```
tail -n 40
```

# less

# more

# 显示输出

# echo

# 过滤器

# grep

# egrep

# 流处理

# sed

# 打包和压缩

# tar

# zip

# unzip

# bzip2

# bunzip2

# gzip

# gunzip

# 进程

# ps

# top

# kill

# killall

# 磁盘

# 空间

# du

# df

# 网络

# 状态

# netstat

# 下载

# curl

# wget

# 同步和拷贝

# scp

# rcp

# rsync

# ssh

# ftp

# tftp

# sftp

# telnet

# 编译和链接

# make

# gcc

# ar

# ranlib

# strip

# nm

# 工具类

# test

# Linux通用逻辑

# 命令

# 参数

# --help

# --version

# --xxx和-X

# 常见命令组合

# ps+grep

# chown+chmod

# Linux开发心得

# Shell

Linux中的shell，是命令行交互环境，用来操作和控制系统的。

可以在shell中运行各种命令。

其中shell自带很多命令，其中常见的有：

- `command &`：让进程在后台运行
- `jobs -l`：查看后台运行的进程
- `fg %n`：让后台运行的进程 `n` 到前台来
- `bg %n`：让进程 `n` 到后台去
- `ps`
  - `n` 为 `jobs` 查看到的进程编号

其他一些快捷键：

- `ctrl-z`：将任务丢到后台暂停
- `jobs -l`：查看后台所有任务状态
-

# cron

`cron` 用来实现定时任务，其中的配置文件是 `crontab`

- `cron` 文件
  - root用户一般是
    - `/var/spool/cron`
      - 可以看到 `crontab -l` 列出的任务对应的原始文件配置
  - 系统级别的还有
    - `/etc/crontab`
    - `/etc/cron.d` 下面的子文件

crontab相关命令：

- 列表当前任务
  - `crontab -l`
- 编辑任务
  - `crontab -e`
- 查看cron状态
  - `service cron status`
  - `/etc/init.d/cron status`
  - `ps -ef | grep cron`
    - 输出举例
      - `root 899 1 0 May07 ? 00:01:28 /usr/sbin/cron -f`
- 重启cron
  - `/etc/init.d/cron restart`
- 启动cron
  - `/etc/init.d/cron start`
- 查看cron的log日志：
  - `tailf /var/log/syslog | grep cron`

# cron文件格式(语法)

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7)
# |  |  |  |  |
# *  *  *  *  *   command to be executed
```

说明：

- 每一个域中，可以包含 `*` 或者 `逗号` 分割的数字，或者 `-` 连接的数字
  - `*` 星号：表示任意
  - `,` 逗号：分割表示时刻， separator
  - `-` 短横线：连接，表示时间段， range of values

- `/` 斜杠：表示间隔， 如果第一个域为 `/2`，则表示每隔两分钟， step value

举例：

```
例子:
# 每月的最后1天
0 0 L * * *


说明:

Linux
*    *    *    *    *
-    -    -    -    -
|    |    |    |    |
|    |    |    |    +----- day of week (0 - 7) (Sunday=0 or 7) OR sun,mon,tue,
|    |    |    +---------- month (1 - 12) OR jan,feb,mar,apr ...
|    |    +--------------- day of month (1 - 31)
|    +-------------------- hour (0 - 23)
+------------------------- minute (0 - 59)
```

以及：

| Field Name 字段 | Mandatory? 是否必填 | Allowed Values 允许值 | Allowed special Characters 允许特殊字符 | 备注 |
|---|---|---|---|---|
| Seconds | Yes | 0-59 | `,` `-` `*` `/` | 标准实现不支持此字段 |
| Minutes | Yes | 0-59 | `,` `-` `*` `/` | |
| Hours | Yes | 0-23 | `,` `-` `*` `/` | |
| Day of the Month | Yes | 1-31 | `,` `-` `*` `?` `/` `L` `W` | 只有部分软件实现了：`?` `L` `W` |
| Month | Yes | 1-12 or JAN-DEC | `,` `-` `*` `/` | |
| Day of the Week | Yes | 1-7 OR SUN-SAT | `,` `-` `*` `?` `/` `L` `#` | 只有部分软件实现了：`?` `L` `#` |
| Year | No | EMPTY, 1970-2099 | `,` `-` `*` `/` | 标准实现不支持此字段 |

说明：

- 标准字段
  - `逗号` ： 用于分隔列表。例如，在第5个字段(星期几)中使用 MON,WED,FRI 表示周一、周三和周五

    - `连字符` ： 定义范围。例如，2000-2010 表示2000年至2010年期间的每年，包括2000年和2010年
        - 除非用 `反斜杠` = `\` 转义，否则命令中的 `百分号` = `%` 会被替换成换行符，第一个百分号后面的所有数据都会作为标准输入发送给命令
- 非标准字段
    - `L` = `Last`
        - 当在星期几字段中使用的时候，可以指定给定月份的结构
            - 例如
                - `最后一个星期五`` = 5L`
            - 在月日字段中，可以指定一个月的最后一天
    - `W` = `day of month`
        - 指定最接近给定日期的工作日（星期一-星期五）
            - 例如
                - `15W` = `最接近该月15日的工作日`
                    - 所以
                        - 如果15号是星期六，触发器在14号星期五触发
                        - 如果15日是星期天，触发器在16日星期一触发。
                        - 如果15号是星期二，那么它在15号星期二触发。
                - `1W`
                    - 如果这个月的第一天是星期六，不会跨到上个月，触发器会在这个月的第三天（也就是星期一）触发。
            - 只有指定一天（不能是范围或列表）的时候，才能指定 `W` 字符
    - `#` = `星期几`
        - 后面必须跟一个介于 `1` 和 `5` 之间的数字。
            - 例如
                - `5#3` =每个月的第 `三` 个星期 `五`
    - 在某些实现中， `?` 用来代替 `*` 以将月中的某一天或周中的某一天留空
        - 其他cron的实现是替换 `?` 为cron守护进程的启动时间
            - 例如
                - `？？＊＊＊＊`
                    - 如果cron在上午8:25启动，将更新为 `25 8 * * * *` ，并在每天的这个时间运行，直到再次重新启动
    - 分钟字段设置 `*/5` 表示每5分钟一次
        - 注意：这里指的是能被 `5` 整除的分钟数

特殊字符:

| Operator | Purpose | Example |
|---|---|---|
| asterisk `*` | Specifies all possible values for a field | An asterisk in the hour time field is equivalent to “every hour.” |
| question mark `?` | A question mark ( ? ) is allowed in the day-of-month and day-of-week fields. It is used to specify “no specific value,” which is useful when you need to specify something in one of these two fields, but not in the other. | If you want a trigger to fire on a particular day of the month (for example, the 10th), but you don't care what day of the week that is, enter 10 in the day-of-month field, and ? in the day-of-week field |
| dash `-` | Specifies a range of values | 2-5, which is equivalent to 2,3,4,5 |
| comma `,` | Specifies a list of values | 1,3,4,7,8 |
| slash `/` | Used to skip a given number of values | `*/3` in the hour time field is equivalent to 0,3,6,9,12,15,18,21. The asterisk `*` specifies `every hour`, but the `/3` means only the first, fourth, seventh.<br>You can use a number in front of the slash to set the initial value. For example, `2/3` means 2,5,8,11, and so on. |
| `L` `last` | The `L` character is allowed for the day-of-month and day-of-week fields. Specifies either the last day of the month, or the last xxx day of the month. | The value L in the day-of-month field means “the last day of the month,” which is day 31 for January, or day 28 for February in non-leap years. If you use L in the day-of-week field by itself, it simply means 7 or SAT. But if you use it in the day-of-week field after another value, it means “the last xxx day of the month.” For example, 6L means “the last Friday of the month.” **HINT**:When you use the L option, be careful not to specify lists or ranges of values. Doing so causes confusing results. |

| Operator | Purpose | Example |
| --- | --- | --- |
| `W weekday` | The W character is allowed for the day-of-month field. Specifies the weekday (Monday-Friday) nearest the given day. | If you specify 15W as the value for the day-of-month field, the meaning is “the nearest weekday to the 15th of the month.” So if the 15th is a Saturday, the trigger fires on Friday the 14th. If the 15th is a Sunday, the trigger fires on Monday the 16th. If the 15th is a Tuesday, it fires on Tuesday the 15th. However, if you specify 1W as the value for day-of-month, and the 1st is a Saturday, the trigger fires on Monday the 3rd, because it does not “jump” over the boundary of a month’s days. The W character can only be specified when the day-of-month is a single day, not a range or list of days.<br>HINT:You can combine the L and Wcharacters for the day-of-month expression to yield LW, which translates to “last weekday of the month.” |
| pound sign `#` | The pound sign ( # ) character is allowed for the day-of-week field. This character is used to specify “the nth” xxx day of the month. | The value of 6#3 in the day-of-week field means the third Friday of the month (day 6 = Friday and #3 = the 3rd one in the month).<br>**Other Examples**: 2#1 specifies the first Monday of the month and 4#5 specifies the fifth Wednesday of the month. However, if you specify #5 and there are fewer than 5 of the given day-of-week in the month, no firing occurs that month. |

# 在线工具

网上有些在线网站，可以帮助你计算和生成cron规则，比如

crontab执行时间计算 - 在线工具

可以在线计算出cron规则的含义

举例：

cron规则是：

```
0 4 1 */2 0
```

当前时间：`2019-11-08 23:55`

网站会帮你计算出：

接下来7次的执行时间：

```
2019-12-01 04:00:00
2021-08-01 04:00:00
2023-10-01 04:00:00
2024-12-01 04:00:00
```

-》从而得知是：

之后每个偶数的月份的1号的04:00:00 去执行命令

频率是：每2个月一次

# 常见问题

## 命令本身运行没问题，但cron执行任务不执行不起效果

```
sudo sh /root/xxx-ssl/renew_cert.sh
```

虽然命令单独执行没问题，但是cron不执行任务，不起效，则可能是脚本权限问题

解决办法：

需要加上可执行权限

```
chmod +x /root/cy-ssl/renew_cert.sh
```

才可以。

# 立即生效

# 包管理器

# 注意事项

## 安装必要的依赖的包

如果需要编译安装，则除了要安装对应工具，还需要安装对应库的开发包，其中包括头文件等文件，是别的编译安装期间需要用到的

比如，Python3在编译安装期间，开启了 `ssl` 的时候，需要用到ssl的库： `openssl`

则除了要安装对应的库：

```
apt-get install openssl
```

之外，还要安装对应的头文件：

```
apt-get install libssl-dev
```

才能被正确编译

# 基础开发工具

此处介绍，Linux类系统中的开发工具集合库

Linux类系统中，如果后续需要源码编译等开发，往往需要安装一个开发包的集合：

- `Ubuntu` 中叫做：`build-essential`
  - 解释
    - 中文直译：编译相关的基础的（工具，库等）
  - 如何安装：
    - `apt-get install build-essential`
- `CentOS` 中叫做：`Development Tools`
  - 解释
    - 其中最核心的是这几个库：`gcc` 、`gcc-c++` 、`make`
      - 如何安装：`yum install gcc gcc-c++ make`
  - 如何安装：
    - `yum groupinstall "Development Tools"`

# 数据恢复

Linux中，万一不小心数据被删，借助相关数据恢复工具，也是有一定机会恢复误删数据的。

Linux中的数据被删恢复工具：

- `ext3grep`
  - 安装：
    - `sudo apt install ext3grep`
- `extundelete`
  - 能指定恢复单个文件
  - 支持系统格式：`ext3`、`ext4`
- 其他
  - `foremost`
    - 支持系统格式：`ext2`、`ext3`、`vfat`、`NTFS`、`ufs`、`jfs`等
  - `debugfs`

# 模拟Linux系统

# 常见问题

此处整理一些Linux开发期间，相对通用的一些问题，以及对应解决办法。

# 安装新版后使用仍是旧版

当安装了一个新的软件/库/工具后，虽然which查看到已经是新的，但是实际上运行该命令，使用该工具，则还是旧的，导致报错。

举例：

（1）Ubuntu中升级了pip3

```
pip3 install --upgrade pip
```

后，虽然

```
which pip3
```

已经是最新的版本：`/usr/local/bin/pip3`

但是实际上去运行pip3，结果还是用的旧的：`/usr/bin/pip3`

解决办法：重启shell终端 = 关闭旧的当前的shell，重新打开一个新的shell，即可

（2）基于Debian的树莓派中，用源码编译和安装了最新版Python 3.7.3

虽然

```
which python3
```

已经是新的 `/usr/local/bin/python3`

且通过版本号可以确认的确是新版：

```
# python3 --version
Python 3.5.3
```

但实际运行的 `python3` 还是旧的版本：`/usr/bin/python3`

解决办法：重启shell终端，即可

(3) 在安卓上模拟Linux的Termux

```
pkg -y install wget
```

后，wget仍然是旧的

解决办法：

还要：

```
hash -r
```

-》目的：清除bash的缓存

-》命令行中找到的wget是最新的

-》wget就支持https地址了

# 附录

下面列出相关参考资料。

# 参考资料


- 【已解决】Mac中统计某个目录下所有文件中的字符个数
- 【已解决】Linux命令显示当前系统时间
- 【整理】如何读懂cron的值的真正含义即计算出命令执行的具体时间
- 【或许解决】Ubuntu中crontab的脚本任务没有执行
-
- linux ar命令_繁城落叶-CSDN博客
- linux的ar命令： 目标文件(.o)<===> 静态库(.a)_认知 行动 坚持-CSDN博客
- ar(1) - Linux man page
- ar command in Linux with examples - GeeksforGeeks
- Linux命令大全: cat命令 – 在终端设备上显示文件内容_谭青海-CSDN博客
- Linux中的13个基本Cat命令示例 - 知乎
- ar、ranlib、nm命令详解 - 简书
- Sed Command in Linux/Unix with examples - GeeksforGeeks
- sed命令详解 - 简书
- Linux sed 命令 | 菜鸟教程
- 一个开发的Linux使用心得总结 - 马一特 - 博客园
- Getting Started with the LLVM System — LLVM 12 documentation
- Home · pyenv/pyenv Wiki
- Shell 前后台进程切换 - 陈斌彬的技术博客
- Mac终端增强技能 - 简书
- crontab执行时间计算 - 在线工具
- Understanding Cron Syntax in the Job Scheduler - Cloud Manager Administrator Reference
- 每天学习一个命令：crontab 定时任务 | Verne in GitHub
- 19. crontab 定时任务 — Linux Tools Quick Tutorial
- crontab执行时间计算 - 在线工具
- crontab - Cron Job Not Running? - Ask Ubuntu
- linux - Why is my crontab not working, and how can I troubleshoot it? - Server Fault
- 程序员的崩溃瞬间：一个rm -rf把服务器数据删没了...
- 使用 Linux 文件恢复工具
- ext3grep - Recover Deleted Files on Debian and Ubuntu
- ext3文件系统反删除利器-ext3grep | 《Linux就该这么学》
-