
Table of Contents

Introduction	1.1
git	1.2
gitbook	1.3
dos	1.4
python	1.5
error-code	1.6
cache	1.7

Notebook

How to use git in Linux?

在同一台电脑上配置多个git账户

1.首先在`~/.ssh`目录下执行

```
ssh-keygen -t rsa -C "miaoying.new@qq.com"
```

其中`-C "miaoying.new@qq.com"`可以不加。如果加上，则在最后生成的`myself_id_rsa.pub`文件内容的末尾会带上`miaoying.new@qq.com`；如果不加，则`myself.id.rsa.pub`文件内容的末尾会加上当前设备的登录用户名和设备名。

根据提示输入文件名（我输入的是`myself_id_rsa`，文件名随意取），之后可以看到生成了两个文件：

```
myself_id_rsa  myself_id_rsa.pub
```

其中，`myself_id_rsa`存放的是私钥，`myself_id_rsa.pub`存放的是公钥。

2.将公钥添加到github的SSH keys列表里，即表示该github账户可以允许含有该SSH的设备进行读写操作，把该SSH文件拷贝到其他设备上，其他设备也可以对项目进行读写操作。

3.配置好后，该设备上就有两个github账户，需要对项目进行账户指定，即允许哪些用户对项目进行git操作，例如项目Demo，只允许用户名为zhangsan，邮箱为`zhangsan@qq.com`进行操作，那么在Demo项目根目录下执行（用户名和邮箱随意取，因为git项目信任的是SSH key，而不是用户名）

```
git config user.name zhangsan  
git config user.email zhangsan@qq.com
```

另外，同一台设备上可以生成多个SSH，也就是说以上操作可重复执行多次。

How to use gitbook in Linux?

安装

1. 首先安装nodejs sudo apt-get install nodejs sudo apt-get install npm npm install gitbook-pdf -g
2. 由于生成pdf文件依赖于ebook-convert，故首先安装ebook-convert：<http://calibre-ebook.com/download>
3. 安装gitbook npm install gitbook-cli -g
4. 安装pdf转换工具 https://calibre-ebook.com/download_linux

使用

1. 将项目clone到本地后，在项目根目录下执行：gitbook init, 则会生成相应的文件：SUMMARY.md 和 README.md
其中，SUMMARY.md是用来存放书的目录的，可以对SUMMARY.md进行编写，但是，对于目录的每一个链接必须有实体文件，否则点击无效。
2. 对md文件进行编辑保存后，使用gitbook pdf命令生成书即可。

DOS命令

- DOS简介
- 批处理（Batch）
- 批处理语法（不间断更新）
 - echo
 - rem
 - pause
 - call
 - start
 - goto
 - set
 - 管道符号 |
 - 逻辑命令符

DOS简介

1.dos是磁盘操作系统（Disk Operation System）的缩写，是个人计算机上的一类单用户单任务的操作系统。微软所有的后续版本中，磁盘操作系统仍然被保留着。微软图形界面操作系统Windows NT问世以来，DOS就是以一个后台程序的形式出现，可以通过点击运行CMD进入运行。

2.dos操作系统用户指令不区分大小写。

3.完整的dos组成（5个部分）

- a.引导程序（BOOT）：由格式化程序直接写入磁盘初始扇区。
- b.基本输入/输出管理程序（PC-DOS为IBMBIO.COM、MS-DOS为IO.SYS）。
- c.文件管理和系统调用程序（PC-DOS为IBMDOS.COM、MS-DOS为MSDOS.SYS）。
- d.命令处理程序（COMMAND.COM）。
- e.各种外部命令：完成各种辅助功能的可执行文件

4.dos是命令模式下的人机交互界面，人通过这个界面来运行和控制计算机。另外，dos作为操作系统能有效地管理、调度、运行个人计算机各种软件和硬件资源。

5.Windows在“附件”中有一个“命令提示符”（CMD），其模拟了一个DOS环境，可以使用相关的命令来对计算机和网络进行操作。

批处理（Batch）

1.批处理也称为批处理脚本，也称作宏，是对某对象进行批量的处理，通常被认为是一种简化的脚本语言，应用于DOS和Windows系统中。类似于Unix的Shell脚本，由DOS和Windows系统内的命令解释器（COMMAND.COM 或者 CMD.EXE）解释运行。批处理文件的扩展名为bat。目前常见的批处理包含两类：DOS批处理（基于DOS命令，用来自动地批量地执行DOS命令以实现特定操作的脚本）和PS批处理（基于图片编辑软件PhotoShop，用来批量处理图片的脚本）。

2.批处理程序虽然是在命令行环境中运行，但不仅仅能使用命令行软件，任何当前系统下可运行的程序都可以放在批处理文件中运行。

3.系统在解释运行批处理程序时，首先扫描整个批处理程序，然后从第一行代码开始向下逐句执行所有的命令，直至程序结尾遇见exit命令或出错意外退出。

批处理语法（不间断更新）

echo

打开回显或关闭回显功能，或显示消息

```
echo [{on|off}] [message]
```

sample:

```
:: 关闭回显  
echo off  
  
:: 开启回显  
echo on  
  
:: 默认开启  
echo  
  
:: 显示消息 "hello world"  
echo "hello world"
```

在实际应用中，我们会把这条命令和重定向符号（也称为管道符号，一般用>>>^）结合起来实现输入一些命令到特定的文件中。

rem

注释命令，类似于在java中的//，但它并不会被执行，只是起到一个注释作用，只有在编辑批处理时才会被看到，主要便于修改。

```
rem [注释消息]
```

sample:

```
rem 这是我的注释消息
```

::也有rem的功能，以下是区别：

```
在关闭回显时，rem和::后的內容都不会显示  
打开回显时，rem后的內容回显示出来，然而::后的內容仍然不会显示出来
```

pause

暂停命令，运行pause命令时，将显示：Press any key to continue...（或：请按任意键继续...）

sample:

```
@echo off  
echo 接下来会显示“请按任意键继续...”  
pause
```

call

从一个批处理命令调用另一个批处理命令，并且不终止父批处理程序。如果在脚本或批处理外使用call，他将不会在命令行起作用。

```
:: 指定要调用的批处理的位置和名称
call [路径文件名]
```

start

调用外部程序，所有的dos命令和命令行程序都可以由start命令来调用。

sample:

```
::打开Windows的计算器
start calc.exe
```

start命令的常用参数：

min	开始时窗口最小化
high	在high优先级类别开始应用程序
separate	在分开的空间内开始16位Windows程序
realtime	在realtime优先级类别开始应用程序
wait	启动应用程序并等候它结束
parameters	传送到命令/程序的参数

执行的应用程序是32位 GUI 应用程序时，CMD.EXE 不等应用程序终止就返回命令提示。如果在命令脚本内执行，则新行为则不会发生。

goto

跳转命令。程序指针跳转到指定的标签，从标签后的第一条命令开始继续执行批处理。

sample:

```
:: 跳转到test
goto test

:test
```

set

显示、设置或删除变量。 1.显示变量：

```
set 或 set s 前者显示批处理当前已定义的所有变量及其值，后者显示所有以s开头的变量及值。
```

2.设置变量：

```
set aa=abcd
```

3.删除变量：

```
:: 若变量已被定义，则删除变量；若aa尚未被定义，则此命令无实际意义
```

```
set aa=
```

批处理中的变量是不区分类型的，比如执行set aa=345后，变量aa的值既可以被视为数字345，也可以被视为字符串345。set命令具有扩展功能，如用作交互输入、字符串处理、数值计算等，属于高级命令范畴。

管道符号 |

将管道符号前面命令的输出结果重定向输出到管道符号后面的命令中去，作为后面命令的输入。使用格式为：
command1 | command2

sample:

```
:: 在询问是否删除文件a.txt时，直接显示y（即表示同意删除）
@echo off
echo aaa>a.txt
echo y|del /p a.txt
```

逻辑命令符

逻辑命令符包括：& && ||

```
& -- 用来连接n个DOS命令，并把这些命令按顺序执行，而不管是否有命令执行失败
&& -- 当前面的命令执行成功时才会执行后续的命令，否则不执行
|| -- 当前面的命令失败时才会执行后续的命令，否则不执行
```

python

- [简介](#)

简介

python 是一种高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。

1. 解释性：在开发过程中没有了编译环节。
2. 交互性：可以在一个Python提示符直接互动执行写程序。
3. 面向对象：支持面向对象的风格或代码封装在对象的编程技术。

优点

易于学习
易于维护
易于阅读
一个广泛的标准库
互动模式
可移植
可扩展
数据库（python提供所有主要的商业数据库的接口）
GUI编程
可嵌入（python可嵌入到c/c++程序中，使程序的用户获得脚本化的能力）

error code

记录遇到的错误码

502 Bad Gateway

情况描述 项目发布时使用同事编写的脚本进行release发布到服务器，由于项目配置文件中端口号一直是8084,(某个地方配置了项目发布到外网的端口一定是8084，具体未知)，然后由于本地有个服务占用了8084，所以把该项目端口换成了8083，发布到服务器后就报错502 Bad Gateway.

错误码描述 502 Bad Gateway是指错误网关，无效网关，在互联网中表示一种网络错误，表现在web浏览器中给出的页面反馈。但这不意味上游服务器已关闭（无响应网关/代理），而是上游服务器和网关/代理不同意的协议交换数据，往往意味着一个或两个机器已不正确或不完全编程。

一般出现了这个问题是由于不良的IP之间的沟通后端计算机，包括可能尝试访问的在web服务器上的网站。
(分析问题前需要完全清除浏览器缓存)

cache

刷新

1. 基本刷新

点击刷新或者使用F5快捷键

基本刷新有可能只是从本地的硬盘重新拿取数据到浏览器，并不一定重新向服务器发出请求。

2. 从服务器刷新

重新直接点击链接（快捷键Ctrl + F5）

实际上会从服务器重新下载数据。