# cxl schedule Documentation

***Release 1.0.0***

**cxl**

**Dec 15, 2017**

# CONTENTS

CHAPTER

# ONE

# INTRODUCE

**good good study ,day day up !**

*good good study ,day day up !*

`good good study ,day day up !`

*good good study ,day day up !*$_{\text{oMygod}}$

*good good study ,day day up !*$^{\text{oMygod}}$

*good good study ,day day up !*

- one
- two three
- onechild
- twochild

1. one
2. two 我是一个 ingnew

   one
   two

企业详情

**企业名称**
南京c0企业
**epcode**
c00001
**端口**
9998
**服务器内网IP**
10.0.0.227
**服务器外网IP**
10.0.0.227
**当前套餐**
C0
**最大用户数**
1
**签单状态**
签单

企业外线号码

| 号码 | 类型 | 归属地 |
| --- | --- | --- |
| 02566669999 | 总机 | 江苏省南京市 |
| 02566669998 | 直线 | 江苏省南京市 |

2 条记录 1/1 页

Table 1.1: Frozen Delights!

| Treat | Quantity | Description |
|---|---|---|
| Albatross | 2.99 | On a stick! |
| Crunchy Frog | 1.49 | If we took the bones out, it wouldn’ t be crunchy, now would it? |
| Gannet Ripple | 1.99 | On a stick! |

## 1.1 read the docs

### 1.1.1 refer

**See also:**

readthedocs 官方网址
sphinx 语法文档
个人博客文档
sphinx 中文语法文档

### 1.1.2 install

1. apt install build-essential
2. apt install python-dev
3. apt install python-pip
4. apt install python-setuptools
5. apt install libxml2-dev
6. apt install libxslt1-dev
7. apt install zlib1g-dev
8. apt install elasticsearch
9. apt install redis-server
10. pip install virtualenv
11. virtualenv rtd && cd rtd
12. source bin/activate
13. apt install git
14. git clone https://github.com/rtfd/readthedocs.org.git
15. cd readthedocs.org
16. pip install -r requirements.txt
17. python manage.py migrate
18. python manage.py collectstatic
19. python manage.py loaddata test_data
20. python manage.py runserver 0.0.0.0:8000
21. apt install texlive-latex-recommended
22. apt install texlive-fonts-recommended

23. apt install texlive-latex-extra
24. apt install latexmk
25. make latexpdf LATEXMKOPTS=" -silent"
26. apt install texlive-xetex
27. apt install latex-cjk-all

### 1.1.3 usage

- install Sphinx

```
apt install python-pip -y
pip install sphinx sphinx-autobuild
pip install --upgrade pip 安装最新版 pip
mkdir doc && cd doc
sphinx-quickstart
make html
```

- **conf.py** latex_engine=xelatex 解决无法生成 pdf 问题

CHAPTER

TWO

# LINUX

## 2.1 point

### 2.1.1 bootloader

#### refer

http://blog.csdn.net/darling757267/article/details/52356354 http://www.cnblogs.com/unicode/archive/2010/06/12/1756755.html

#### ubuntu 启动过程

假如您使用的 Linux 发行版是 Ubuntu，很可能会发现在您的计算机上找不到/etc/inittab 文件了，这是因为 Ubuntu 使用了一种被称为 upstart 的新型 init 系统。

#### 开发 Upstart 的缘由

大约在 2006 年或者更早的时候，Ubuntu 开发人员试图将 Linux 安装在笔记本电脑上。在这期间技术人员发现经典的 sysvinit 存在一些问题：它不适合笔记本环境。这促使程序员 Scott James Remnant 着手开发 upstart。当 Linux 内核进入 2.6 时代时，内核功能有了很多新的更新。新特性使得 Linux 不仅是一款优秀的服务器操作系统，也可以被用于桌面系统，甚至嵌入式设备。桌面系统或便携式设备的一个特点是经常重启，而且要频繁地使用硬件热插拔技术。在现代计算机系统中，硬件繁多、接口有限，人们并非将所有设备都始终连接在计算机上，比如 U 盘平时并不连接电脑，使用时才插入 USB 插口。因此，当系统上电启动时，一些外设可能并没有连接。而是在启动后当需要的时候才连接这些设备。在 2.6 内核支持下，一旦新外设连接到系统，内核便可以自动实时地发现它们，并初始化这些设备，进而使用它们。这为便携式设备用户提供了很大的灵活性。可是这些特性为 sysvinit 带来了一些挑战。当系统初始化时，需要被初始化的设备并没有连接到系统上；比如打印机。为了管理打印任务，系统需要启动 CUPS 等服务，而如果打印机没有接入系统的情况下，启动这些服务就是一种浪费。Sysvinit 没有办法处理这类需求，它必须一次性把所有可能用到的服务都启动起来，即使打印机并没有连接到系统，CUPS 服务也必须启动。还有网络共享盘的挂载问题。在/etc/fstab 中，可以指定系统自动挂载一个网络盘，比如 NFS，或者 iSCSI 设备。在本文的第一部分 sysvinit 的简介中可以看到，sysvinit 分析/etc/fstab 挂载文件系统这个步骤是在网络启动之前。可是如果网络没有启动，NFS 或者 iSCSI 都不可访问，当然也无法进行挂载操作。Sysvinit 采用 netdev 的方式来解决这个问题，即/etc/fstab 发现 netdev 属性挂载点的时候，不尝试挂载它，在网络初始化并使能之后，还有一个专门的 netfs 服务来挂载所有这些网络盘。这是一个不得已的补救方法，给管理员带来不便。部分新手管理员甚至从来也没有听说过 netdev 选项，因此经常成为系统管理的一个陷阱。针对以上种种情况，Ubuntu 开发人员在评估了当时的几个可选 init 系统之后，决定重新设计和开发一个全新的 init 系统，即 UpStart。UpStart 基于事件机制，比如 U 盘插入 USB 接口后，udev 得到内核通知，发现该设备，这就是一个新的事件。UpStart 在感知到该事件之后触发相应的等待任务，比如处理/etc/fstab 中存在的挂载点。采用这种事件驱动的模式，upstart 完美地解决了即插即用设备带来的新问题。此外，采用事件驱动机制也带来了一些其它有益的变化，比如加快了系统启动时间。sysvinit 运行时是同步阻塞的。一个脚本运行的时候，后续脚本必须等待。这意味着所有的初始化步骤都是串行执行的，而实际上很多服务彼此并不相关，完全可以并行启动，从而减小系统的启动时间。在 Linux 大量应用于服务器的时代，系统启动时间也许还不那么重要；然而对于桌面

系统和便携式设备，启动时间的长短对用户体验影响很大。此外云计算等新的 Server 端技术也往往需要单个设备可以更加快速地启动。UpStart 满足了这些需求，目前不仅桌面系统 Ubuntu 采用了 UpStart，甚至企业级服务器级的 RHEL 也默认采用 UpStart 来替换 sysvinit 作为 init 系统。

### Upstart 的特点

UpStart 解决了之前提到的 sysvinit 的缺点。采用事件驱动模型，UpStart 可以：更快地启动系统当新硬件被发现时动态启动服务硬件被拔除时动态停止服务这些特点使得 UpStart 可以很好地应用在桌面或者便携式系统中，处理这些系统中的动态硬件插拔特性。

### Upstart 概念和术语

Upstart 的基本概念和设计清晰明确。UpStart 主要的概念是 job 和 event。Job 就是一个工作单元，用来完成一件工作，比如启动一个后台服务，或者运行一个配置命令。每个 Job 都等待一个或多个事件，一旦事件发生，upstart 就触发该 job 完成相应的工作。

- Job

Job 就是一个工作的单元，一个任务或者一个服务。可以理解为 sysvinit 中的一个服务脚本。有三种类型的工作：task job；service job；abstract job；task job 代表在一定时间内会执行完毕的任务，比如删除一个文件；service job 代表后台服务进程，比如 apache httpd。这里进程一般不会退出，一旦开始运行就成为一个后台精灵进程，由 init 进程管理，如果这类进程退出，由 init 进程重新启动，它们只能由 init 进程发送信号停止。它们的停止一般也是由于所依赖的停止事件而触发的，不过 upstart 也提供命令行工具，让管理人员手动停止某个服务；Abstract job 仅由 upstart 内部使用，仅对理解 upstart 内部机理有所帮助。我们不用关心它。除了以上的分类之外，还有另一种工作（Job）分类方法。Upstart 不仅可以用来为整个系统的初始化服务，也可以为每个用户会话（session）的初始化服务。系统的初始化任务就叫做 system job，比如挂载文件系统的任务就是一个 system job；用户会话的初始化服务就叫做 session job。

- Job 生命周期

Upstart 为每个工作都维护一个生命周期。一般来说，工作有开始，运行和结束这几种状态。为了更精细地描述工作的变化，Upstart 还引入了一些其它的状态。比如开始就有开始之前 (pre-start)，即将开始 (starting) 和已经开始了 (started) 几种不同的状态，这样可以更加精确地描述工作的当前状态。工作从某种初始状态开始，逐渐变化，或许要经历其它几种不同的状态，最终进入另外一种状态，形成一个状态机。在这个过程中，当工作的状态即将发生变化的时候，init 进程会发出相应的事件（event）。表 1.Upstart 中 Job 的可能状态状态名含义 Waiting 初始状态 Starting Job 即将开始 pre-start 执行 pre-start 段，即任务开始前应该完成的工作 Spawned 准备执行 script 或者 exec 段 post-start 执行 post-start 动作 Running interim state set after post-start section processed denoting job is running (But it may have no associated PID!) pre-stop 执行 pre-stop 段 Stopping interim state set after pre-stop section processed Killed 任务即将被停止 post-stop 执行 post-stop 段图 1 展示了 Job 的状态机。

其中有四个状态会引起 init 进程发送相应的事件，表明该工作的相应变化：Starting Started Stopping Stopped 而其它的状态变化不会发出事件。那么我们接下来就来看看事件的详细含义吧。

- 事件 Event

顾名思义，Event 就是一个事件。事件在 upstart 中以通知消息的形式具体存在。一旦某个事件发生了，Upstart 就向整个系统发送一个消息。没有任何手段阻止事件消息被 upstart 的其它部分知晓，也就是说，事件一旦发生，整个 upstart 系统中所有工作和其它的事件都会得到通知。Event 可以分为三类: signal，methods 或者 hooks。Signals Signal 事件是非阻塞的，异步的。发送一个信号之后控制权立即返回。Methods Methods 事件是阻塞的，同步的。

- Hooks

Hooks 事件是阻塞的，同步的。它介于 Signals 和 Methods 之间，调用发出 Hooks 事件的进程必须等待事件完成才可以得到控制权，但不检查事件是否成功。事件是个非常抽象的概念，下面我罗列出一些常见的事件，希望可以帮助您进一步了解事件的含义：系统上电启动，init 进程会发送” start” 事件根文件系统可写时，相应 job 会发送文件系统就绪的事件一个块设备被发现并初始化完成，发送相应的事件某个文件系统被挂载，发送相应的事件类似 atd 和 cron，可以在某个时间点，或者周期的时间点发送事件另外一个 job 开始或结束时，发送相应的事件一个磁盘文件被修改时，可以发出相应的事件一个网络

设备被发现时，可以发出相应的事件缺省路由被添加或删除时，可以发出相应的事件不同的 Linux 发行版对 upstart 有不同的定制和实现，实现和支持的事件也有所不同，可以用 man 7 upstart-events 来查看事件列表。

- Job 和 Event 的相互协作

Upstart 就是由事件触发工作运行的一个系统，每一个程序的运行都由其依赖的事件发生而触发的。系统初始化的过程是在工作和事件的相互协作下完成的，可以大致描述如下：系统初始化时，init 进程开始运行，init 进程自身会发出不同的事件，这些最初的事件会触发一些工作运行。每个工作运行过程中会释放不同的事件，这些事件又将触发新的工作运行。如此反复，直到整个系统正常运行起来。究竟哪些事件会触发某个工作的运行？这是由工作配置文件定义的。

- 工作配置文件

任何一个工作都是由一个工作配置文件（Job Configuration File）定义的。这个文件是一个文本文件，包含一个或者多个小节（stanza）。每个小节是一个完整的定义模块，定义了工作的一个方面，比如 author 小节定义了工作的作者。工作配置文件存放在/etc/init 下面，是以.conf 作为文件后缀的文件。清单 1. 一个最简单的工作配置文件

#This is a simple demo of Job Configure file #This line is comment, start with #

#Stanza 1, The author author “Liu Ming”

#Stanza 2, Description description “This job only has author and description, so no use, just a demo”
上面的例子不会产生任何作用，一个真正的工作配置文件会包含很多小节，其中比较重要的小节有以下几个：

- “expect” Stanza

Upstart 除了负责系统的启动过程之外，和 SysVinit 一样，Upstart 还提供一系列的管理工具。当系统启动之后，管理员可能还需要进行维护和调整，比如启动或者停止某项系统服务。或者将系统切换到其它的工作状态，比如改变运行级别。本文后续将详细介绍 Upstart 的管理工具的使用。为了启动，停止，重启和查询某个系统服务。Upstart 需要跟踪该服务所对应的进程。比如 httpd 服务的进程 PID 为 1000。当用户需要查询 httpd 服务是否正常运行时，Upstart 就可以利用 ps 命令查询进程 1000，假如它还在正常运行，则表明服务正常。当用户需要停止 httpd 服务时，Upstart 就使用 kill 命令终止该进程。为此，Upstart 必须跟踪服务进程的进程号。部分服务进程为了将自己变成后台精灵进程 (daemon)，会采用两次派生 (fork) 的技术，另外一些服务则不会这样做。假如一个服务派生了两次，那么 UpStart 必须采用第二个派生出来的进程号作为服务的 PID。但是，UpStart 本身无法判断服务进程是否会派生两次，为此在定义该服务的工作配置文件中必须写明 expect 小节，告诉 UpStart 进程是否会派生两次。Expect 有两种，” expect fork” 表示进程只会 fork 一次；” expect daemonize” 表示进程会 fork 两次。

- “exec” Stanza 和” script” Stanza

一个 UpStart 工作一定需要做些什么，可能是运行一条 shell 命令，或者运行一段脚本。用” exec” 关键字配置工作需要运行的命令；用” script” 关键字定义需要运行的脚本。清单 2 显示了 exec 和 script 的用法：清单 2.script 例子

# mountall.conf description “Mount filesystems on boot” start on startup stop on starting rcS …script

```
. /etc/default/rcS
[ -f /forcefsck ] && force_fsck=” --force-fsck”
[ “$FSCKFIX” =” yes” ] && fsck_fix=” --fsck-fix”

...

exec mountall -daemon $force_fsck $fsck_fix
```

end script …这是 mountall 的例子，该工作在系统启动时运行，负责挂载所有的文件系统。该工作需要执行复杂的脚本，由” script” 关键字定义；在脚本中，使用了 exec 来执行 mountall 命令。

- “start on” Stanza 和” stop on” Stanza

“start on” 定义了触发工作的所有事件。” start on” 的语法很简单，如下所示：start on EVENT [[KEY=]VALUE]…[and|or…] EVENT 表示事件的名字，可以在 start on 中指定多个事件，表示该工作的开始需要依赖多个事件发生。多个事件之间可以用 and 或者 or 组合，” 表示全部都必须发生” 或者” 其中之一发生即可” 等不同的依赖条件。除了事件发生之外，工作的启动还可以依赖特定的条件，因

此在 start on 的 EVENT 之后，可以用 KEY=VALUE 来表示额外的条件，一般是某个环境变量 (KEY) 和特定值 (VALUE) 进行比较。如果只有一个变量，或者变量的顺序已知，则 KEY 可以省略。“stop on” 和” start on” 非常类似，只不过是定义工作在什么情况下需要停止。代码清单 3 是” start on” 和” stop on” 的一个例子。清单 3. start on/ stop on 例子

#dbus.conf description “D-Bus system message bus”

start on local-filesystems stop on deconfiguring-networking …D-Bus 是一个系统消息服务，上面的配置文件表明当系统发出 local-filesystems 事件时启动 D-Bus；当系统发出 deconfiguring-networking 事件时，停止 D-Bus 服务。

- Session Init

UpStart 还可以用于管理用户会话的初始化。在我写这篇文章的今天，多数 Linux 发行版还没有使用 UpStart 管理会话。只有在 Ubuntu Raring 版本中，使用 UpStart 管理用户会话的初始化过程。首先让我们了解一下 Session 的概念。Session 就是一个用户会话，即用户从远程或者本地登入系统开始工作，直到用户退出。这整个过程就构成一个会话。每个用户的使用习惯和使用方法都不相同，因此用户往往需要为自己的会话做一个定制，比如添加特定的命令别名，启动特殊的应用程序或者服务，等等。这些工作都属于对特定会话的初始化操作，因此可以被称为 Session Init。用户使用 Linux 可以有两种模式：字符模式和图形界面。在字符模式下，会话初始化相对简单。用户登录后只能启动一个 Shell，通过 shell 命令使用系统。各种 shell 程序都支持一个自动运行的启动脚本，比如 ~/.bashrc。用户在这些脚本中加入需要运行的定制化命令。字符会话需求简单，因此这种现有的机制工作的很好。在图形界面下，事情就变得复杂一些。用户登录后看到的并不是一个 shell 提示符，而是一个桌面。一个完整的桌面环境由很多组件组成。一个桌面环境包括 window manager，panel 以及其它一些定义在/usr/share/gnome-session/sessions/下面的基本组件；此外还有一些辅助的应用程序，共同帮助构成一个完整的方便的桌面，比如 system monitors，panel applets，NetworkManager，Bluetooth，printers 等。当用户登录之后，这些组件都需要被初始化，这个过程比字符界面要复杂的多。目前启动各种图形组件和应用的工作由 gnome-session 完成。过程如下：以 Ubuntu 为例，当用户登录 Ubuntu 图形界面后，显示管理器 (Display Manager)lightDM 启动 Xsession。Xsession 接着启动 gnome-session，gnome-session 负责其它的初始化工作，然后就开始了一个 desktop session。图 2. 传统 desktop session 启动过程

```
init
 |- lightdm
 |    |- Xorg
 |    |- lightdm ---session-child
 |         |- gnome-session --session=ubuntu
 |              |- compiz
 |              |- gwibber
 |              |- nautilus
 |              |- nm-applet
 |              :
 |              :
 |
 |- dbus-daemon --session
 |
 :
 :
```

这个过程有一些缺点 (和 sysVInit 类似)。一些应用和组件其实并不需要在会话初始化过程中启动，更好的选择是在需要它们的时候才启动。比如 update-notifier 服务，该服务不停地监测几个文件系统路径，一旦这些路径上发现可以更新的软件包，就提醒用户。这些文件系统路径包括新插入的 DVD 盘等。Update-notifier 由 gnome-session 启动并一直运行着，在多数情况下，用户并不会插入新的 DVD，此时 update-notifier 服务一直在后台运行并消耗系统资源。更好的模式是当用户插入 DVD 的时候再运行 update-notifier。这样可以加快启动时间，减小系统运行过程中的内存等系统资源的开销。对于移动，嵌入式等设备等这还意味着省电。除了 Update-notifier 服务之外，还有其它一些类似的服务。比如 Network Manager，一天之内用户很少切换网络设备，所以大部分时间 Network Manager 服务仅仅是在浪费系统资源；再比如 backup manager 等其它常驻内存，后台不间断运行却很少真正被使用的服务。
用 UpStart 的基于事件的按需启动的模式就可以很好地解决这些问题，比如用户插入网线的时候才启动 Network Manager，因为用户插入网线表明需要使用网络，这可以被称为按需启动。
下图描述了采用 UpStart 之后的会话初始化过程。
图 3. 采用 Upstart 的 Desktop session init 过程

```
init
 |- lightdm
```

```
|   |- Xorg
|   |- lightdm ---session-child
|        |- session-init # <-- upstart running as normal user
|             |- dbus-daemon --session
|             |- gnome-session --session=ubuntu
|             |- compiz
|             |- gwibber
|             |- nautilus
|             |- nm-applet
|             :
|             :
:
:
```

### UpStart 使用

有两种人员需要了解 Upstart 的使用。第一类是系统开发人员，比如 MySQL 的开发人员。它们需要了解如何编写工作配置文件，以便用 UpStart 来管理服务。比如启动，停止 MySQL 服务。另外一种情况是系统管理员，它们需要掌握 Upstart 的管理命令以便配置和管理系统的初始化，管理系统服务。系统开发人员需要了解的 UpStart 知识

系统开发人员不仅需要掌握工作配置文件的写法，还需要了解一些针对服务进程编程上的要求。本文仅列出了少数工作配置文件的语法。要全面掌握工作配置文件的写法，需要详细阅读 Upstart 的手册。这里让我们来分析一下如何用 Upstart 来实现传统的运行级别，进而了解如何灵活使用工作配置文件。Upstart 系统中的运行级别 Upstart 的运作完全是基于工作和事件的。工作的状态变化和运行会引起事件，进而触发其它工作和事件。而传统的 Linux 系统初始化是基于运行级别的，即 SysVInit。因为历史的原因，Linux 上的多数软件还是采用传统的 SysVInit 脚本启动方式，并没有为 UpStart 开发新的启动脚本，因此即便在 Debian 和 Ubuntu 系统上，还是必须模拟老的 SysVInit 的运行级别模式，以便和多数现有软件兼容。虽然 Upstart 本身并没有运行级别的概念，但完全可以用 UpStart 的工作模拟出来。让我们完整地考察一下 UpStart 机制下的系统启动过程。系统启动过程下图描述了 UpStart 的启动过程。

系统上电后运行 GRUB 载入内核。内核执行硬件初始化和内核自身初始化。在内核初始化的最后，内核将启动 pid 为 1 的 init 进程，即 UpStart 进程。Upstart 进程在执行了一些自身的初始化工作后，立即发出”startup”事件。上图中用红色方框加红色箭头表示事件，可以在左上方看到”startup”事件。所有依赖于”startup”事件的工作被触发，其中最重要的是 mountall。mountall 任务负责挂载系统中需要使用的文件系统，完成相应工作后，mountall 任务会发出以下事件：local-filesystem，virtual-filesystem，all-swaps，其中 virtual-filesystem 事件触发 udev 任务开始工作。任务 udev 触发 upstart-udev-bridge 的工作。Upstart-udev-bridge 会发出 net-device-up IFACE=lo 事件，表示本地回环 IP 网络已经准备就绪。同时，任务 mountall 继续执行，最终会发出 filesystem 事件。此时，任务 rc-sysinit 会被触发，因为 rc-sysinit 的 start on 条件如下：start on filesystem and net-device-up IFACE=lo 任务 rc-sysinit 调用 telinit。Telinit 任务会发出 runlevel 事件，触发执行/etc/init/rc.conf。rc.conf 执行/etc/rc$.d/目录下的所有脚本，和 SysVInit 非常类似，读者可以参考本文第一部分的描述。

程序开发时需要注意的事项作为程序开发人员，在编写系统服务时，需要了解 UpStart 的一些特殊要求。只有符合这些要求的软件才可以被 UpStart 管理。规则一，派生次数需声明。很多 Linux 后台服务都通过派生两次的技巧将自己变成后台服务程序。如果您编写的服务也采用了这个技术，就必须通过文档或其它的某种方式明确地让 UpStart 的维护人员知道这一点，这将影响 UpStart 的 expect stanza，我们在前面已经详细介绍过这个 stanza 的含义。规则二，派生后即可用。后台程序在完成第二次派生的时候，必须保证服务已经可用。因为 UpStart 通过派生计数来决定服务是否处于就绪状态。规则三，遵守 SIGHUP 的要求。UpStart 会给精灵进程发送 SIGHUP 信号，此时，UpStart 希望该精灵进程做以下这些响应工作：• 完成所有必要的重新初始化工作，比如重新读取配置文件。这是因为 UpStart 的命令”initctl reload”被设计为可以让服务在不重启的情况下更新配置。• 精灵进程必须继续使用现有的 PID，即收到 SIGHUP 时不能调用 fork。如果服务必须在这里调用 fork，则等同于派生两次，参考上面的规则一的处理。这个规则保证了 UpStart 可以继续使用 PID 管理本服务。规则四，收到 SIGTEM 即 shutdown。• 当收到 SIGTERM 信号后，UpStart 希望精灵进程进程立即干净地退出，释放所有资源。如果一个进程在收到 SIGTERM 信号后不退出，Upstart 将对其发送 SIGKILL 信号。

- 系统管理员需要了解的 Upstart 命令

作为系统管理员，一个重要的职责就是管理系统服务。比如系统服务的监控，启动，停止和配置。UpStart 提供了一系列的命令来完成这些工作。其中的核心是 initctl，这是一个带子命令风格的命令行工具。比如可以用 initctl list 来查看所有工作的概况：$initctl list alsa-mixer-save stop/waiting avahi-daemon start/running, process 690 mountall-net stop/waiting rc stop/waiting rsyslog start/running, process 482 screen-cleanup stop/waiting tty4 start/running, process 859 udev start/running, process 334 upstart-udev-bridge start/running, process 304 ureadahead-other stop/waiting 这是在 Ubuntu10.10 系统上的输出，其它的 Linux 发行版上的输出会有所不同。第一列是工作名，比如 rsyslog。第二列是工作的目标；第三列是工作的状态。此外还可以用 initctl stop 停止一个正在运行的工作；用 initctl start 开始一个工作；还可以用 initctl status 来查看一个工作的状态；initctl restart 重启一个工作；initctl reload 可以让一个正在运行的服务重新载入配置文件。这些命令和传统的 service 命令十分相似。表 2.service 命令和 initctl 命令对照表

Service 命令 UpStart initctl 命令 service start initctl start service stop initctl stop service restart initctl restart service reload initctl reload 很多情况下管理员并不喜欢子命令风格，因为需要手动键入的字符太多。UpStart 还提供了一些快捷命令来简化 initctl，实际上这些命令只是在内部调用相应的 initctl 命令。比如 reload，restart，start，stop 等等。启动一个服务可以简单地调用 start <job> 这和执行 initctl start <job> 是一样的效果。一些命令是为了兼容其它系统 (主要是 sysvinit)，比如显示 runlevel 用/sbin/runlevel 命令：$runlevel N 2 这个输出说明当前系统的运行级别为 2。而且系统没有之前的运行级别，也就是说在系统上电启动进入预定运行级别之后没有再修改过运行级别。那么如何修改系统上电之后的默认运行级别呢? 在 Upstart 系统中，需要修改/etc/init/rc-sysinti.conf 中的 DEFAULT_RUNLEVEL 这个参数，以便修改默认启动运行级别。这一点和 sysvinit 的习惯有所不同，大家需要格外留意。还有一些随 UpStart 发布的小工具，用来帮助开发 UpStart 或者诊断 UpStart 的问题。比如 init-checkconf 和 upstart-monitor 还可以使用 initctl 的 emit 命令从命令行发送一个事件。#initctl emit <event> 这一般是用于 UpStart 本身的排错。

#### Upstart 小结

可以看到，UpStart 的设计比 SysVInit 更加先进。多数 Linux 发行版上已经不再使用 SysVInit，一部分发行版采用了 UpStart，比如 Ubuntu；而另外一些比如 Fedora，采用了一种被称为 systemd 的 init 系统。Systemd 出现的比 UpStart 更晚，但发展迅速，虽然 UpStart 也还在积极开发并被越来越多地应用，但 systemd 似乎发展更快，我将在下一篇文章中再介绍 systemd。

### 2.1.2 defconf-utils

- install

```
apt install debconf-utils
```

### 2.1.3 network

#### 配置固定 ip

# vi /etc/network/interfaces # 配置

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
source /etc/network/interfaces.d/*
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto enp0s3  # 网卡端口
#iface enp0s3 inet dhcp  # dhcp 自动获取 ip
iface enp0s3  inet static  # 静态 ip
address 10.0.0.42          # ip 地址
netmask 255.255.255.0      # 掩码
```

```
gateway 10.0.0.254                  # 网关
dns-nameservers 202.106.0.20 8.8.8.8 # dns 服务器地址
```

# 重启 network

```
/etc/init.d/networking restart
```

## 2.2 apt

### 2.2.1 refer

none refer

### 2.2.2 command example

#### base method

- install package

```
$ apt install package_name  -y
```

- remove package

```
$ apt autoremove --purge package_name -y
$ apt remove --purge package_name -y
```

- check package

```
$ dpkg -l | grep package_name
```

#### install&remove Nginx

- install nginx

```
apt install nginx -y
find /etc -name "*nginx*"
    /etc/logrotate.d/nginx
    /etc/rc1.d/K01nginx
    /etc/rc5.d/S02nginx
    /etc/nginx
    /etc/nginx/nginx.conf
    /etc/init.d/nginx
    /etc/rc3.d/S02nginx
    /etc/ufw/applications.d/nginx
    /etc/rc2.d/S02nginx
    /etc/rc0.d/K01nginx
    /etc/init/nginx.conf
    /etc/default/nginx
    /etc/systemd/system/multi-user.target.wants/nginx.service
    /etc/rc4.d/S02nginx
    /etc/rc6.d/K01nginx
find /usr -name "*nginx*"
    /usr/sbin/nginx
    /usr/share/doc/nginx
    /usr/share/doc/nginx-common
    /usr/share/doc/nginx-core
```

```
    /usr/share/nginx
    /usr/share/vim/registry/nginx.yaml
    /usr/share/vim/addons/indent/nginx.vim
    /usr/share/vim/addons/syntax/nginx.vim
    /usr/share/vim/addons/ftdetect/nginx.vim
    /usr/share/lintian/overrides/nginx-common
    /usr/share/lintian/overrides/nginx-core
    /usr/share/apport/package-hooks/source_nginx.py
find /lib -name "*nginx*"
    /lib/systemd/system/nginx.service
dpkg -l | grep nginx
    ii  nginx           1.10.3-0ubuntu0.16.04.2     all          small,␣
↪powerful, scalable web/proxy server
    ii  nginx-common    1.10.3-0ubuntu0.16.04.2     all          small,␣
↪powerful, scalable web/proxy server - common files
    ii  nginx-core      1.10.3-0ubuntu0.16.04.2     amd64        nginx web/
↪proxy server (core version)
```

- remove nginx

```
apt autoremove --purge nginx-common -y 直接删除 nginx-common
```

会自动卸载依赖 nginx 及配置文件，执行以上命令都为空

### install&remove mysql

- install mysql

```
apt install mysql-server -y
find /etc -name "*mysql*"
    /etc/logrotate.d/mysql-server
    /etc/logcheck/ignore.d.server/mysql-server-5_7
    /etc/logcheck/ignore.d.workstation/mysql-server-5_7
    /etc/logcheck/ignore.d.paranoid/mysql-server-5_7
    /etc/rc1.d/K02mysql
    /etc/rc5.d/S02mysql
    /etc/init.d/mysql
    /etc/rc3.d/S02mysql
    /etc/rc2.d/S02mysql
    /etc/rc0.d/K02mysql
    /etc/init/mysql.conf
    /etc/apparmor.d/usr.sbin.mysqld
    /etc/apparmor.d/local/usr.sbin.mysqld
    /etc/systemd/system/multi-user.target.wants/mysql.service
    /etc/mysql
    /etc/mysql/conf.d/mysql.cnf
    /etc/mysql/conf.d/mysqldump.cnf
    /etc/mysql/mysql.cnf
    /etc/mysql/mysql.conf.d
    /etc/mysql/mysql.conf.d/mysqld_safe_syslog.cnf
    /etc/mysql/mysql.conf.d/mysqld.cnf
    /etc/rc4.d/S02mysql
    /etc/rc6.d/K02mysql
find /usr -name "*mysql*"
    /usr/lib/mysql
    /usr/lib/mysql/plugin/mysql_no_login.so
    /usr/sbin/mysqld
    /usr/bin/mysql_upgrade
    /usr/bin/mysql_embedded
    /usr/bin/mysqlpump
    /usr/bin/mysqldumpslow
```

```
/usr/bin/mysql_secure_installation
/usr/bin/mysqladmin
/usr/bin/mysql_config_editor
/usr/bin/mysqlcheck
/usr/bin/mysqld_multi
/usr/bin/mysqldump
/usr/bin/mysqloptimize
/usr/bin/mysql_tzinfo_to_sql
/usr/bin/mysqlshow
/usr/bin/mysqlimport
/usr/bin/mysqlrepair
/usr/bin/mysql_plugin
/usr/bin/mysqlslap
/usr/bin/mysqlreport
/usr/bin/mysqld_safe
/usr/bin/mysql
/usr/bin/mysqlanalyze
/usr/bin/mysqlbinlog
/usr/bin/mysql_install_db
/usr/bin/mysql_ssl_rsa_setup
/usr/share/doc/mysql-server-5.7
/usr/share/doc/mysql-server-5.7/mysqld.sym.gz
/usr/share/doc/mysql-server
/usr/share/doc/mysql-server-core-5.7
/usr/share/doc/mysql-client-5.7
/usr/share/doc/kamailio/examples/icscf/icscf.mysql.sql.gz
/usr/share/doc/kamailio/examples/kamailio/acc-mysql.cfg.gz
/usr/share/doc/mysql-common
/usr/share/doc/mysql-client-core-5.7
/usr/share/vim/vim74/syntax/mysql.vim
/usr/share/mysql
/usr/share/mysql/mysql-log-rotate
/usr/share/mysql/mysql_security_commands.sql
/usr/share/mysql/mysql_sys_schema.sql
/usr/share/mysql/mysql_test_data_timezone.sql
/usr/share/mysql/mysql-systemd-start
/usr/share/mysql/mysqld_multi.server
/usr/share/mysql/mysql_system_tables.sql
/usr/share/mysql/mysql_system_tables_data.sql
/usr/share/lintian/overrides/mysql-server-5.7
/usr/share/lintian/overrides/mysql-client-5.7
/usr/share/lintian/overrides/mysql-common
/usr/share/man/man1/mysqldumpslow.1.gz
/usr/share/man/man1/mysqloptimize.1.gz
/usr/share/man/man1/mysqlpump.1.gz
/usr/share/man/man1/mysqlshow.1.gz
/usr/share/man/man1/mysql_upgrade.1.gz
/usr/share/man/man1/mysql_config_editor.1.gz
/usr/share/man/man1/mysqlbinlog.1.gz
/usr/share/man/man1/mysqld_safe.1.gz
/usr/share/man/man1/mysqlrepair.1.gz
/usr/share/man/man1/mysqldump.1.gz
/usr/share/man/man1/mysqlanalyze.1.gz
/usr/share/man/man1/mysqlslap.1.gz
/usr/share/man/man1/mysqld_multi.1.gz
/usr/share/man/man1/mysql_secure_installation.1.gz
/usr/share/man/man1/mysqlreport.1.gz
/usr/share/man/man1/mysqlimport.1.gz
/usr/share/man/man1/mysqlcheck.1.gz
/usr/share/man/man1/mysql_plugin.1.gz
/usr/share/man/man1/mysqladmin.1.gz
/usr/share/man/man1/mysql.1.gz
/usr/share/man/man1/mysql_embedded.1.gz
```

```
    /usr/share/man/man1/mysql_install_db.1.gz
    /usr/share/man/man1/mysql_ssl_rsa_setup.1.gz
    /usr/share/man/man1/mysqlman.1.gz
    /usr/share/man/man1/mysql_tzinfo_to_sql.1.gz
    /usr/share/man/man8/mysqld.8.gz
    /usr/share/sosreport/sos/plugins/mysql.py
    /usr/share/sosreport/sos/plugins/__pycache__/mysql.cpython-35.pyc
    /usr/share/apport/package-hooks/source_mysql-5.7.py
    /usr/share/bash-completion/completions/mysqladmin
    /usr/share/bash-completion/completions/mysql
    /usr/share/mysql-common
find /lib -name "*mysql*"
    /lib/systemd/system/mysql.service
dpkg -l | grep mysql
    ii  mysql-client-5.7                  5.7.19-0ubuntu0.16.04.1                 ␣
↪amd64        MySQL database client binaries
    ii  mysql-client-core-5.7             5.7.19-0ubuntu0.16.04.1                 ␣
↪amd64        MySQL database core client binaries
    ii  mysql-common                      5.7.19-0ubuntu0.16.04.1                  all␣
↪          MySQL database common files, e.g. /etc/mysql/my.cnf
    ii  mysql-server                      5.7.19-0ubuntu0.16.04.1                  all␣
↪          MySQL database server (metapackage depending on the latest version)
    ii  mysql-server-5.7                  5.7.19-0ubuntu0.16.04.1                 ␣
↪amd64        MySQL database server binaries and system database setup
    ii  mysql-server-core-5.7             5.7.19-0ubuntu0.16.04.1                 ␣
↪amd64        MySQL database server binaries
```

- remove mysql

```
apt autoremove --purge mysql-common -y
find /etc -name "*mysql*"
    /etc/mysql
find /usr -name "*mysql*"
    /usr/share/doc/kamailio/examples/icscf/icscf.mysql.sql.gz
    /usr/share/doc/kamailio/examples/kamailio/acc-mysql.cfg.gz
    /usr/share/vim/vim74/syntax/mysql.vim
    /usr/share/sosreport/sos/plugins/mysql.py
    /usr/share/sosreport/sos/plugins/__pycache__/mysql.cpython-35.pyc
    /usr/share/bash-completion/completions/mysqladmin
    /usr/share/bash-completion/completions/mysql
这个配置目录没有删除，手动删除
rm /etc/mysql -rf
rm /var/lib/mysql -rf
```

### install&remove redis

- install redis

```
apt install redis-server -y
find /etc -name "*redis*"
    /etc/logrotate.d/redis-server
    /etc/rc1.d/K01redis-server
    /etc/rc5.d/S02redis-server
    /etc/redis
    /etc/redis/redis.conf
    /etc/redis/redis-server.pre-up.d
    /etc/redis/redis-server.post-down.d
    /etc/redis/redis-server.post-up.d
    /etc/redis/redis-server.pre-down.d
    /etc/init.d/redis-server
    /etc/rc3.d/S02redis-server
```

```
    /etc/rc2.d/S02redis-server
    /etc/rc0.d/K01redis-server
    /etc/default/redis-server
    /etc/systemd/system/multi-user.target.wants/redis-server.service
    /etc/systemd/system/redis.service
    /etc/rc4.d/S02redis-server
    /etc/rc6.d/K01redis-server
find /usr -name "*redis*"
    /usr/lib/tmpfiles.d/redis-server.conf
    /usr/bin/redis-check-dump
    /usr/bin/redis-benchmark
    /usr/bin/redis-server
    /usr/bin/redis-cli
    /usr/bin/redis-check-aof
    /usr/share/doc/redis-server
    /usr/share/doc/redis-tools
    /usr/share/doc/redis-tools/examples/redis-trib.rb
    /usr/share/man/man1/redis-server.1.gz
    /usr/share/man/man1/redis-benchmark.1.gz
    /usr/share/man/man1/redis-cli.1.gz
    /usr/share/bash-completion/completions/bash_completion.d/redis-cli
find /lib -name "*redis*"
    /lib/systemd/system/redis-server.service
dpkg -l | grep redis
    ii  redis-server  2:3.0.6-1      amd64        Persistent key-value database with␣
↪network interface
    ii  redis-tools   2:3.0.6-1      amd64        Persistent key-value database with␣
↪network interface (client)
```

- remove redis

```
所有内容都返回空
apt autoremove --purge redis-server -y
```

### install&remove apache

- install apache

```
apt install apache2 -y
find /etc -name "*apache*"
    /etc/logrotate.d/apache2
    /etc/rc1.d/K01apache-htcacheclean
    /etc/rc1.d/K01apache2
    /etc/rc5.d/K01apache-htcacheclean
    /etc/rc5.d/S02apache2
    /etc/init.d/apache2
    /etc/init.d/apache-htcacheclean
    /etc/rc3.d/K01apache-htcacheclean
    /etc/rc3.d/S02apache2
    /etc/ufw/applications.d/apache2
    /etc/ufw/applications.d/apache2-utils.ufw.profile
    /etc/apache2
    /etc/apache2/apache2.conf
    /etc/rc2.d/K01apache-htcacheclean
    /etc/rc2.d/S02apache2
    /etc/rc0.d/K01apache-htcacheclean
    /etc/rc0.d/K01apache2
    /etc/default/apache-htcacheclean
    /etc/cron.daily/apache2
    /etc/rc4.d/K01apache-htcacheclean
    /etc/rc4.d/S02apache2
```

```
    /etc/rc6.d/K01apache-htcacheclean
    /etc/rc6.d/K01apache2
find /usr -name "*apache*"
    /usr/lib/apache2
    /usr/sbin/apachectl
    /usr/sbin/apache2ctl
    /usr/sbin/apache2
    /usr/share/doc/apache2-bin
    /usr/share/doc/apache2
    /usr/share/doc/apache2/examples/apache2.monit
    /usr/share/doc/apache2-data
    /usr/share/doc/apache2-utils
    /usr/share/apache2
    /usr/share/apache2/apache2-maintscript-helper
    /usr/share/apache2/icons/apache_pb.png
    /usr/share/apache2/icons/apache_pb.gif
    /usr/share/apache2/icons/apache_pb.svg
    /usr/share/apache2/icons/apache_pb2.gif
    /usr/share/apache2/icons/apache_pb2.png
    /usr/share/vim/vim74/syntax/apachestyle.vim
    /usr/share/vim/vim74/syntax/apache.vim
    /usr/share/bug/apache2-bin
    /usr/share/bug/apache2
    /usr/share/lintian/overrides/apache2-bin
    /usr/share/lintian/overrides/apache2
    /usr/share/lintian/overrides/apache2-data
    /usr/share/man/man8/apache2.8.gz
    /usr/share/man/man8/apachectl.8.gz
    /usr/share/man/man8/apache2ctl.8.gz
    /usr/share/sosreport/sos/plugins/__pycache__/apache.cpython-35.pyc
    /usr/share/sosreport/sos/plugins/apache.py
    /usr/share/apport/package-hooks/apache2.py
    /usr/share/bash-completion/completions/apache2ctl
find /lib -name "*apache*"
    /lib/systemd/system/apache2.service.d
    /lib/systemd/system/apache2.service.d/apache2-systemd.conf
dpkg -l | grep apache
    ii  apache2          2.4.18-2ubuntu3.4    amd64        Apache HTTP Server
    ii  apache2-bin      2.4.18-2ubuntu3.4    amd64        Apache HTTP Server (modules␣
↪and other binary files)
    ii  apache2-data     2.4.18-2ubuntu3.4    all          Apache HTTP Server (common␣
↪files)
    ii  apache2-utils    2.4.18-2ubuntu3.4    amd64        Apache HTTP Server (utility␣
↪programs for web servers)
```

- remove apache

```
apt autoremove --purge apache2 -y 满足需求
```

### install&remove php

- install php

```
apt install apache2 -y                  安装 apache2
apt install mysql-server -y             安装 mysql
apt install php7.0 -y
find /etc -name "*php*"
    /etc/logrotate.d/php7.0-fpm
    /etc/rc1.d/K01php7.0-fpm
    /etc/rc5.d/S01php7.0-fpm
    /etc/alternatives/php.1.gz
```

```
    /etc/alternatives/php
    /etc/init.d/php7.0-fpm
    /etc/rc3.d/S01php7.0-fpm
    /etc/apache2/conf-available/php7.0-fpm.conf
    /etc/rc2.d/S01php7.0-fpm
    /etc/rc0.d/K01php7.0-fpm
    /etc/php
    /etc/php/7.0/fpm/php-fpm.conf
    /etc/php/7.0/fpm/php.ini
    /etc/php/7.0/cli/php.ini
    /etc/init/php7.0-fpm.conf
    /etc/apparmor.d/abstractions/php5
    /etc/systemd/system/multi-user.target.wants/php7.0-fpm.service
    /etc/rc4.d/S01php7.0-fpm
    /etc/cron.d/php
    /etc/rc6.d/K01php7.0-fpm
find /usr -name "*php*"
    /usr/lib/tmpfiles.d/php7.0-fpm.conf
    /usr/lib/php
    /usr/lib/php/php7.0-fpm-reopenlogs
    /usr/lib/php/php-maintscript-helper
    /usr/lib/php/php7.0-fpm-checkconf
    /usr/lib/php/php-helper
    /usr/lib/php/7.0/php.ini-production
    /usr/lib/php/7.0/php.ini-production.cli
    /usr/lib/php/7.0/php.ini-development
    /usr/sbin/phpenmod
    /usr/sbin/phpdismod
    /usr/sbin/phpquery
    /usr/sbin/php-fpm7.0
    /usr/bin/php7.0
    /usr/bin/php
    /usr/share/doc/php7.0
    /usr/share/doc/php7.0-readline
    /usr/share/doc/php-common
    /usr/share/doc/php7.0-json
    /usr/share/doc/php7.0-cli
    /usr/share/doc/php7.0-fpm
    /usr/share/doc/php7.0-common
    /usr/share/doc/kamailio/examples/web_im/click_to_dial.php
    /usr/share/doc/kamailio/examples/web_im/send_im.php
    /usr/share/doc/kamailio/examples/kamailio/web_im/click_to_dial.php
    /usr/share/doc/kamailio/examples/kamailio/web_im/send_im.php
    /usr/share/doc/php7.0-opcache
    /usr/share/php7.0-readline
    /usr/share/vim/vim74/autoload/phpcomplete.vim
    /usr/share/vim/vim74/indent/php.vim
    /usr/share/vim/vim74/syntax/php.vim
    /usr/share/vim/vim74/ftplugin/php.vim
    /usr/share/vim/vim74/compiler/php.vim
    /usr/share/php7.0-json
    /usr/share/nano/php.nanorc
    /usr/share/bug/php7.0
    /usr/share/bug/php7.0-readline
    /usr/share/bug/php7.0-json
    /usr/share/bug/php7.0-cli
    /usr/share/bug/php7.0-fpm
    /usr/share/bug/php7.0-common
    /usr/share/bug/php7.0-opcache
    /usr/share/mime/application/x-php.xml
    /usr/share/php
    /usr/share/lintian/overrides/php7.0-readline
    /usr/share/lintian/overrides/php7.0-json
```

```
    /usr/share/lintian/overrides/php7.0-cli
    /usr/share/lintian/overrides/php7.0-fpm
    /usr/share/lintian/overrides/php7.0-common
    /usr/share/lintian/overrides/php7.0-opcache
    /usr/share/php7.0-common
    /usr/share/man/man1/php.1.gz
    /usr/share/man/man1/php7.0.1.gz
    /usr/share/man/man8/php-fpm7.0.8.gz
    /usr/share/php7.0-opcache
find /lib -name "*php*"
    /lib/modules/4.4.0-92-generic/kernel/drivers/pci/hotplug/acpiphp_ibm.ko
    /lib/modules/4.4.0-91-generic/kernel/drivers/pci/hotplug/acpiphp_ibm.ko
    /lib/modules/4.4.0-93-generic/kernel/drivers/pci/hotplug/acpiphp_ibm.ko
    /lib/systemd/system/php7.0-fpm.service
dpkg -l | grep php
    ii  php-common                          1:35ubuntu6                                 all␣
↪         Common files for PHP packages
    ii  php7.0                              7.0.22-0ubuntu0.16.04.1                     all␣
↪         server-side, HTML-embedded scripting language (metapackage)
    ii  php7.0-cli                          7.0.22-0ubuntu0.16.04.1                     ␣
↪amd64        command-line interpreter for the PHP scripting language
    ii  php7.0-common                       7.0.22-0ubuntu0.16.04.1                     ␣
↪amd64        documentation, examples and common module for PHP
    ii  php7.0-fpm                          7.0.22-0ubuntu0.16.04.1                     ␣
↪amd64        server-side, HTML-embedded scripting language (FPM-CGI binary)
    ii  php7.0-json                         7.0.22-0ubuntu0.16.04.1                     ␣
↪amd64        JSON module for PHP
    ii  php7.0-opcache                      7.0.22-0ubuntu0.16.04.1                     ␣
↪amd64        Zend OpCache module for PHP
    ii  php7.0-readline                     7.0.22-0ubuntu0.16.04.1                     ␣
↪amd64        readline module for PHP
```

- remove php

```
apt autoremove --purge php-common -y
find /etc -name "*php*"
    /etc/php
    /etc/apparmor.d/abstractions/php5
rm /etc/php -rf
```

### install&remove other

- install

```
apt install php7.0-curl -y
find /etc -name "*curl*"
    /etc/php/7.0/mods-available/curl.ini
    /etc/php/7.0/fpm/conf.d/20-curl.ini
    /etc/php/7.0/cli/conf.d/20-curl.ini
find /usr -name "*php7.0-curl*"
    /usr/share/doc/php7.0-curl
    /usr/share/bug/php7.0-curl
    /usr/share/php7.0-curl
    /usr/share/lintian/overrides/php7.0-curl
find /lib -name "*curl*"
dpkg -l | grep php7.0-curl
    ii  php7.0-curl                         7.0.22-0ubuntu0.16.04.1                     ␣
↪amd64        CURL module for PHP

apt install php7.0-mbstring -y
find /etc -name "*mbstring*"
```

```
    /etc/php/7.0/mods-available/mbstring.ini
    /etc/php/7.0/fpm/conf.d/20-mbstring.ini
    /etc/php/7.0/cli/conf.d/20-mbstring.ini
find /usr -name "*mbstring*"
    /usr/lib/php/20151012/mbstring.so
    /usr/share/doc/php7.0-mbstring
    /usr/share/php7.0-mbstring
    /usr/share/php7.0-mbstring/mbstring
    /usr/share/php7.0-mbstring/mbstring/mbstring.ini
    /usr/share/bug/php7.0-mbstring
    /usr/share/lintian/overrides/php7.0-mbstring
find /lib -name "*mbstring*"
dpkg -l | grep mbstring
    ii  php7.0-mbstring                    7.0.22-0ubuntu0.16.04.1                 ␣
↪amd64        MBSTRING module for PHP

apt install libapache2-mod-php7.0 -y
vi /etc/apache2/mods-enabled/php7.0.load
    LoadModule php7_module /usr/lib/apache2/modules/libphp7.0.so
find /etc -name "*libapache2*"
find /usr -name "*libapache2*"
find /lib -name "*libapache2*"
dpkg -l | grep libapache2
    ii  libapache2-mod-php7.0              7.0.22-0ubuntu0.16.04.1                 ␣
↪amd64        server-side, HTML-embedded scripting language (Apache 2 module)

apt install php7.0-gd -y
find /etc -name "*gd.ini"
    /etc/php/7.0/apache2/conf.d/20-gd.ini
    /etc/php/7.0/mods-available/gd.ini
    /etc/php/7.0/fpm/conf.d/20-gd.ini
    /etc/php/7.0/cli/conf.d/20-gd.ini
find /usr -name "*gd"
    /usr/sbin/rsyslogd
    /usr/share/doc/php7.0-gd
    /usr/share/bug/php7.0-gd
    /usr/share/locale/gd
    /usr/share/lintian/overrides/php7.0-gd
    /usr/share/php7.0-gd
    /usr/share/php7.0-gd/gd
ls /usr/lib/php/20151012/gd*
    /usr/lib/php/20151012/gd.so
dpkg -l | grep php7.0-gd
    ii  php7.0-gd                          7.0.22-0ubuntu0.16.04.1                 ␣
↪amd64        GD module for PHP

apt install php7.0-mysql -y
ls /usr/lib/php/20151012/mysql*
    /usr/lib/php/20151012/mysqli.so  /usr/lib/php/20151012/mysqlnd.so
find /etc -name "mysql*.ini"
    /etc/php/7.0/mods-available/mysqli.ini
    /etc/php/7.0/mods-available/mysqlnd.ini
dpkg -l | grep php7.0-mysql
    ii  php7.0-mysql                       7.0.22-0ubuntu0.16.04.1                 ␣
↪amd64        MySQL module for PHP

apt install php-redis -y
ls /usr/lib/php/20151012/redis*
    /usr/lib/php/20151012/redis.so
find /etc -name "redis*.ini"
    /etc/php/7.0/mods-available/redis.ini
dpkg -l | grep php-redis
    ii  php-redis                          2.2.7-389-g2887ad1+2.2.7-1              ␣
↪amd64        PHP extension for interfacing with Redis
```

```
apt install mcrypt -y
apt install libmcrypt-dev -y

apt install php-mcrypt -y
ls /usr/lib/php/20151012/mcrypt*
    /usr/lib/php/20151012/mcrypt.so
find /etc -name "mcrypt*.ini"
    /etc/php/7.0/mods-available/mcrypt.ini
dpkg -l | grep mcrypt | grep "php"
    ii  php-mcrypt                          1:7.0+35ubuntu6                              all␣
↪          libmcrypt module for PHP [default]
    ii  php7.0-mcrypt                       7.0.22-0ubuntu0.16.04.1                      ␣
↪amd64        libmcrypt module for PHP
```

- remove

```
先删除依赖 mysql, apache 的包, mcrypt 和 libmcrypt-dev 是系统包, 不删除
apt autoremove --purge libapache2-mod-php7.0 -y
ls /etc/apache2/mods-enabled/php*
dpkg -l | grep libapache2
ls /usr/lib/apache2/modules/libphp7.0.so
执行命令均未发现残余
apt autoremove --purge php7.0-mysql -y
执行命令均未发现残余
删除 php
apt autoremove --purge php-common -y
ls /usr/lib/php
    ls: cannot access '/usr/lib/php': No such file or directory
find /etc -name "*php*"
    /etc/php
    /etc/apparmor.d/abstractions/php5
rm /etc/php -rf
```

**Attention:** 以上软件是基于 SPI 系统测试，运行在 ubuntu16.04 上!

## 2.3 shell

### 2.3.1 shell 语法

- 空函数

```
init(){ :; }
```

- case

```
case "$optname" in
   "a")
       # apache 发布端口
      apache_publish_port=$OPTARG
       ;;
    *)
           exit
           ;;
esac
```

### 2.3.2 shell 参数处理

- refer

  http://www.jb51.net/article/48686.htm

- 处理方式

  例如：./test.sh -f config.conf -v –prefix=/home

  - 手工处理

```
* $0 ： ./test.sh，即命令本身，相当于 c/c++ 中的 argv[0]
* $1 ： -f，第一个参数.
* $2 ： config.conf
* $3，$4 ... ：类推。
* $#  参数的个数，不包括命令本身，上例中$# 为 4.
* $@ ：参数本身的列表，也不包括命令本身，如上例为 -f config.conf -v --prefix=/
↪home
* $* ：和$@ 相同，但"$*" 和 "$@"(加引号) 并不同，"$*" 将所有的参数解释成一个字符
串，而"$@" 是一个参数数组。
#!/bin/bash
for arg in "$*"
do
  echo $arg
done
for arg in "$@"
do
 echo $arg
done
```

  - getopts

```
不支持长选项
#!/bin/bash
while getopts "a:bc" arg # 选项后面的冒号表示该选项需要参数
do
        case $arg in
             a)
                echo "a's arg:$optarg" # 参数存在$optarg 中

             b)
                echo "b"

             c)
                echo "c"

             ?)  # 当有不认识的选项的时候 arg 为?
               echo "unkonw argument"
        exit 1

        esac
done
```

  - getopt

```
要点:
1. getopt 用法
语法: getopt [options] [--] optstring parameters
例如: getopt ab:cd -a -b he free cat
输出: -a -b he -- free cat
            getopt 根据 ab:cd 将选项和参数 -a -b he free cat  解析为如下格式:
            -a -b he -- free cat
             其中 -- 将选项与非选项参数分开 free 和 cat 就时非选项参数
```

```
2. set --
-- Do not change any of the flags; useful in setting $1 to -.
set --  主要是影响特殊变量$1 $2 等，其实在上面的脚本中就是将$1 $2 等参数变量重新
组合
例如：
set -- a b c
shell 中的特殊位置变量$1 为 a $2 为 b $3 为 c
3. 如上脚本为 build.sh 用法如下：
./build.sh -a -c -d -e dog
./build.sh -acde  dog
上面两个命令执行结果相同。
```

### 2.3.3 awk

#### example

- 遍历文件相加

```
echo -e "1\n2\n3\n4" > test.txt
awk '{ sum += $1 }; END { print sum }' test.txt
10
```

- 按：分割，打印所有用户名

```
awk -F: '{ print $1 }' /etc/passwd
```

- 读取配置文件, 并输出到变量里

```
echo -e "DB_PWD=123456\nDB_USER=root" > test.txt
cat test.txt |sed "s#//# #g" | sed "s/=/ /g" | awk  '{if(NF>1){printf("%s=\"%s\";",$1,$2)}
↪}'
DB_PWD="123456";DB_USER="root";
eval $(cat test.txt |sed "s#//# #g" | sed "s/=/ /g" | awk  '{if(NF>1){printf("%s=\"%s\";",
↪$1,$2)}}')
echo $DB_PWD
123456
```

- 分析 url

```
param=`echo "$redirectUrl" | awk -F '?' '{print $2}'`;
echo "$param" | awk -F '&' '{i=1;while(i<=NF){n=split($i,array,"=");if(array[1]=="code")
↪{print array[2];break;};i++;}}'
```

- 分析 php 配置文件，找出 session 节 host 配置项

```
awk -F '=>' '{if($0~/\047session\047/){find=1;fl=0;fr=0}if(find){if($0~/\[/){fl++;};if($0~
↪/\]/){fr++;};if($1~/\047host\047/){print $2;exit 0;}if(fl==fr){exit 99}}}' ../
↪application/config.php | awk -F '\047' '{print $2,exit 0}'
'127.0.0.1',
```

### 2.3.4 curl

#### usage

- 静默方式-s，-k 不检查证实 -c 创建 cookie 文件

```
result=`curl -s -k -c cxl_cookie https://10.0.0.42:1066`
 echo $result
{"status":0,"info":" 调用接口成功","data":"home"}
```

- 通过使用 -v 和 -trace 获取更多的链接信息

```
curl  -v  -k -c "" -d "" -H "" https://10.0.0.42:1066
```

- 打印返回的 header（-i）

```
curl  -i  -k -c "" -d "" -H "" https://10.0.0.42:1066
```

### 2.3.5 cut

#### usage

- 按字节剪切

```
echo "test" | cut --b 2-4
est
```

- 按字符剪切

```
echo "test" | cut -c 2-4
est
```

- 多个分段用逗号分开

```
echo "test test"|cut -b 3-5,8
st s
```

- 从文件获取

```
echo -e "星期一\n星期二\n星期三" > test.txt
cut -b 1-3 test.txt
星
星
星
```

- 中文

```
echo "星期四 星期三" | cut -c 1-3
星
```

- 按特殊字符分割

```
echo "root:test:ok" | cut -d : -f 1
root
```

- 综合应用

```
echo "root:test:ok" | cut -d : -f 3 | cut -c 1
o
```

### 2.3.6 find

### 2.3.7 grep

#### usage

- 去不命中的行

```
crontab -l | grep -v "php "
```

- 添加 crontab

```
(crontab -l 2>/dev/null | grep -v "php.*cleanAccesstoken";echo "* 1 * * * php /home/cxl/
↪git-svn/spi-php/think cleanAccesstoken") | crontab -
```

### 2.3.8 jq

#### install

apt install jq -y

#### usage

- get value

```
echo '{"status":0,"info":"we are success","data":{"client":{"client_id":"123456","client_
↪secret":"567890"}}}' | jq ".status"
0
echo '{"status":0,"info":"we are success","data":{"client":{"client_id":"123456","client_
↪secret":"567890"}}}' | jq ".info" | sed 's/"//g'
we are success
echo '{"status":0,"info":"we are success","data":{"client":{"client_id":"123456","client_
↪secret":"567890"}}}' | jq ".data.client.client_id" | sed 's/"//g'
123456
```

- filter data

```
echo '{"status":0,"info":"we are success","data":{"client":{"client_id":"123456","client_
↪secret":"567890"}}}' | jq '.data|.client'
{
  "client_id": "123456",
  "client_secret": "567890"
}
```

### 2.3.9 mktemp

#### usage

- 生成随机文件

```
mktemp ttXXXX.tmp
```

- 生成随机目录

```
mktemp -d tempXXXXXX
```

### 2.3.10 nano

#### usage

### 2.3.11 sed

#### usage

- 替换字符

```
echo "this ReplaceWord success" | sed "s/ReplaceWord/is/"
this is success
```

- 替换文件

```
echo "this ReplaceWord success" > test.txt
cat test.txt
sed -i "s/ReplaceWord/is/" test.txt
cat test.txt
```

- 使用正则

```
echo "this 1234 success" | sed -r "s/[0-9]+/is/"
this is success
```

- -g 表示所有匹配的都执行

```
echo "this 1234 success 1234" | sed -r "s/[0-9]+/is/g"
this is success is
```

- 按行号删除文件内容

```
sed -i '1,50000d' "$cxl_log_file" # 删除 50000 万行
sed -i '1d' a.txt 删首行
sed -i '$d' b.txt 删尾行
sed -i 's/[ ]*//g' c.txt 删空格
sed -i '/^$/d' d.txt 删空行
sed -i  '/^[0-9]*$/d' a.txt 删包含数字的行
sed -i  '1,2d' a.txt 删 2 行
sed -i  '/love/d'  a.txt 删包含 string 的行
```

- 修改制定行内容

  sed -r 's/( '*host*' *)./1" test" /' ../application/config.php

### 2.3.12 service

#### usage

- service –status-all
- service supervisor status

### 2.3.13 tr

#### usage

- 转换为大写

```
echo "hello world" | tr [:lower:] [:upper:]
HELLO WORLD
```

- 删除空格、数字和-号

```
echo "hello-123-world  empty" | tr -d '[:blank:][:digit:]-'
helloworldempty
```

- 删除补级以外的字符，和上一例子正相反

```
echo "hello - 123-world  empty" | tr -d -c '[:blank:][:digit:]-'
- 123-
```

- 去掉连续重复字符

```
echo "helloooo oooo    isssso ookk" | tr -s " os"
hello o iso okk
```

- 删除 window 文件造成 ^M 字符

```
cat file | tr -s "\r" "\n" > new_file
cat file | tr -d "\r" > new_file
```

### 2.3.14 uname

#### usage

- 打印全部

```
uname -a
Linux 1604developer 4.4.0-89-generic #112-Ubuntu SMP Mon Jul 31 19:38:41 UTC 2017 x86_64␣
↪x86_64 x86_64 GNU/Linux
```

- 打印 kernel-name

```
uname -s
Linux
```

- 打印机器网络名

```
uname -n
1604developer
```

- 打印 kernel-release

```
uname -r
4.4.0-89-generic
```

- 打印 kernel-version

```
uname -v
#112-Ubuntu SMP Mon Jul 31 19:38:41 UTC 2017
```

- 打印机器硬件名称

```
uname -m
x86_64
```

- 打印处理器类型

```
uname -p
x86_64
```

- 打印硬件平台

```
uname -i
x86_64
```

- 打印操作系统

```
uname -o
GNU/Linux
```

### 2.3.15 update-rc.d

#### usage

- 添加启动项

```
sudo update-rc.d   apache2 defaults
sudo update-rc.d   nginx defaults
sudo update-rc.d   redis_6379 defaults
```

- 删除启动项

```
sudo update-rc.d -f apache2 remove
sudo update-rc.d -f nginx remove
sudo update-rc.d -f redis_6379 remove
```

### 2.3.16 vi

#### usage

- 翻下页 ctrl+f
- 翻上页 ctrl+b
- 设置行号 set nu

### 2.3.17 xargs

#### usage

1. 当你尝试用 rm 删除太多的文件，你可能得到一个错误信息：/bin/rm Argument list too long. 用 xargs 去避免这个问题

```
find ~ -name  '*.log' -print0 | xargs -0 rm -f
```

2. 获得/etc/ 下所有 *.conf 结尾的文件列表，有几种不同的方法能得到相同的结果，下面的例子仅仅是示范怎么实用 xargs ，在这个例子中实用 xargs 将 find 命令的输出传递给 ls -l

```
find /etc -name "*.conf" | xargs ls -l
```

3. 假如你有一个文件包含了很多你希望下载的 URL, 你能够使用 xargs 下载所有链接

```
cat url-list.txt | xargs wget -c
```

4. 查找所有的 jpg 文件，并且压缩它

```
find / -name *.jpg -type f -print | xargs tar -cvzf images.tar.gz
```

5. 拷贝所有的图片文件到一个外部的硬盘驱动

```
ls *.jpg | xargs -n1 -i cp {} /external-hard-drive/directory
```

6. 查找另一个目录同名文件，-n1 表示每行一个参数 -i 表示用 {} 表示输出的每行记录

```
ls /home/cxl/git-svn/spi/spi-php/bin/res/supervisor | xargs -n1 -i echo {}
```

### 2.3.18 环境变量

#### refer

http://www.cnblogs.com/zhaofeng555/p/4895517.html

#### usage

先将 export LANG=zh_CN 加入/etc/profile，退出系统重新登录，登录提示显示英文。将/etc/profile 中的 export LANG=zh_CN 删除，将 LNAG=zh_CN 加入/etc/environment，退出系统重新登录，登录提示显示中文。用户环境建立的过程中总是先执行/etc/profile 然后在读取/etc/environment。为什么会有如上所叙的不同呢?

应该是先执行/etc/environment，后执行/etc/profile。/etc/environment 是设置整个系统的环境，而/etc/profile 是设置所有用户的环境，前者与登录用户无关，后者与登录用户有关。www.2cto.com 系统应用程序的执行与用户环境可以是无关的，但与系统环境是相关的，所以当你登录时，你看到的提示信息，象日期、时间信息的显示格式与系统环境的 LANG 是相关的，缺省 LANG=en_US，如果系统环境 LANG=zh_CN，则提示信息是中文的，否则是英文的。

对于用户的 SHELL 初始化而言是先执行/etc/profile, 再读取文件/etc/environment. 对整个系统而言是先执行/etc/environment。这样理解正确吗? /etc/enviroment –> /etc/profile –> $HOME/.profile –>$HOME/.env (如果存在)

/etc/profile 是所有用户的环境变量 /etc/enviroment 是系统的环境变量登陆系统时 shell 读取的顺序应该是 www.2cto.com

/etc/profile ->/etc/enviroment –>$HOME/.profile –>$HOME/.env

原因应该是 jtw 所说的用户环境和系统环境的区别了如果同一个变量在用户环境 (/etc/profile) 和系统环境 (/etc/environment) 有不同的值那应该是以用户环境为准了。

(1) /etc/profile: 此文件为系统的每个用户设置环境信息, 当用户第一次登录时, 该文件被执行. 并从/etc/profile.d 目录的配置文件中搜集 shell 的设置。

(2) /etc/bashrc: 为每一个运行 bash shell 的用户执行此文件. 当 bash shell 被打开时, 该文件被读取。www.2cto.com

(3) ~/.bash_profile: 每个用户都可使用该文件输入专用于自己使用的 shell 信息, 当用户登录时, 该文件仅仅执行一次! 默认情况下, 他设置一些环境变量, 执行用户的.bashrc 文件。

(4) ~/.bashrc: 该文件包含专用于你的 bash shell 的 bash 信息, 当登录时以及每次打开新的 shell 时, 该该文件被读取。

(5) ~/.bash_logout: 当每次退出系统 (退出 bash shell) 时, 执行该文件. 另外,/etc/profile 中设定的变量 (全局) 的可以作用于任何用户, 而 ~/.bashrc 等中设定的变量 (局部) 只能继承 /etc/profile 中的变量, 他们是”父子”关系。

(6) ~/.bash_profile 是交互式、login 方式进入 bash 运行的 ~/.bashrc 是交互式 non-login 方式进入 bash 运行的通常二者设置大致相同，所以通常前者会调用后者。

## 2.4 virtualbox

### 2.4.1 install

1、编辑 source.list 添加如下内容 deb http://download.virtualbox.org/virtualbox/debian trusty contrib
2、wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo apt-key add -
3、wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | sudo apt-key add -
4、apt-get update
5、apt-get install virtualbox-5.1
6、wget http://download.virtualbox.org/virtualbox/5.1.0/Oracle_VM_VirtualBox_Extension_Pack-5.1.0-108711.vbox-extpack 下载扩展
7、vboxmanage extpack install Oracle_VM_VirtualBox_Extension_Pack-5.1.0-108711.vbox-extpack 安装扩展
8、vboxmanage list ostypes 查看支持的虚拟机类型
9、vboxmanage list vms 查看已存在的虚拟机

### 2.4.2 example

- create vm

```
vboxmanage createvm --name "ubuntu1204-base" --ostype "Ubuntu_64" --basefolder "/home/cxl/
↪virtualbox/vm/" --register 创建虚拟机
vboxmanage createhd --filename /home/cxl/virtualbox/vm/ubuntu1204-base/ubuntu1204-base --
↪size 8000 创建虚拟硬盘
vboxmanage storagectl "ubuntu1204-base" --add ide  --name "IDE Controller" --bootable on␣
↪创建 ide 接口
vboxmanage storageattach "ubuntu1204-base" --storagectl "IDE Controller" --port 0 --
↪device 0 --type hdd --medium "/home/cxl/virtualbox/vm/ubuntu1204-base/ubuntu1204-base.
↪vdi" 虚拟机关联硬盘
vboxmanage storageattach "ubuntu1204-base" --storagectl "IDE Controller" --port 1 --
↪device 0 --type dvddrive --medium "/home/cxl/virtualbox/iso/ubuntu-12.04.1-server-amd64.
↪iso" 虚拟机关联光驱，加入 iso 安装文件
vboxmanage modifyvm "ubuntu1204-base" --vrde on --vrdeport 5001
vboxmanage startvm "buildserver" --type headless
```

- delete vm

```
vboxmanage unregistervm ubuntu1204-base --delete 删除虚拟机
```

- clone vm

```
vboxmanage clonevm "ubuntu1204-base" --name "buildserver" --register --basefolder "/home/
↪cxl/virtualbox/vm-root/"
vboxmanage startvm "buildserver" --type headless
```

## 2.5 supervisor

### 2.5.1 refer

http://supervisord.org/ http://www.ttlsa.com/linux/using-supervisor-control-program/

### 2.5.2 install

Supervisor: A Process Control System apt install supervisor -y

### 2.5.3 usage

- example

```
cd /etc/supervisor/conf.d
echo_supervisord_conf > /etc/supervisord.conf
[unix_http_server]
file=/tmp/supervisor.sock   ; UNIX socket 文件，supervisorctl 会使用
;chmod=0700                 ; socket 文件的 mode，默认是 0700
;chown=nobody:nogroup       ; socket 文件的 owner，格式： uid:gid
;[inet_http_server]         ; HTTP 服务器，提供 web 管理界面
;port=127.0.0.1:9001        ; Web 管理后台运行的 IP 和端口，如果开放到公网，需要注意安全性
;username=user              ; 登录管理后台的用户名
;password=123               ; 登录管理后台的密码

[supervisord]
logfile=/tmp/supervisord.log ; 日志文件，默认是 $CWD/supervisord.log
logfile_maxbytes=50MB        ; 日志文件大小，超出会 rotate，默认 50MB
logfile_backups=10           ; 日志文件保留备份数量默认 10
loglevel=info                ; 日志级别，默认 info，其它: debug,warn,trace
pidfile=/tmp/supervisord.pid ; pid 文件
nodaemon=false               ; 是否在前台启动，默认是 false，即以 daemon 的方式启动
minfds=1024                  ; 可以打开的文件描述符的最小值，默认 1024
minprocs=200                 ; 可以打开的进程数的最小值，默认 200

; the below section must remain in the config file for RPC
; (supervisorctl/web interface) to work, additional interfaces may be
; added by defining them in separate rpcinterface: sections
[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[supervisorctl]
serverurl=unix:///tmp/supervisor.sock ; 通过 UNIX socket 连接 supervisord，路径与 unix_
↪http_server 部分的 file 一致
;serverurl=http://127.0.0.1:9001 ; 通过 HTTP 的方式连接 supervisord

; 包含其他的配置文件
[include]
files = relative/directory/*.ini    ; 可以是 *.conf 或 *.ini

增加一个管理进程
[program:usercenter]
directory = /home/leon/projects/usercenter ; 程序的启动目录
command = gunicorn -c gunicorn.py wsgi:app  ; 启动命令，可以看出与手动在命令行启动的命令是一
样的
autostart = true     ; 在 supervisord 启动的时候也自动启动
startsecs = 5        ; 启动 5 秒后没有异常退出，就当作已经正常启动了
autorestart = true   ; 程序异常退出后自动重启
startretries = 3     ; 启动失败自动重试次数，默认是 3
user = leon          ; 用哪个用户启动
redirect_stderr = true  ; 把 stderr 重定向到 stdout，默认 false
stdout_logfile_maxbytes = 20MB  ; stdout 日志文件大小，默认 50MB
stdout_logfile_backups = 20     ; stdout 日志文件备份数
; stdout 日志文件，需要注意当指定目录不存在时无法正常启动，所以需要手动创建目录 (supervisord␣
↪会自动创建日志文件)
stdout_logfile = /data/logs/usercenter_stdout.log

; 可以通过 environment 来添加需要的环境变量，一种常见的用法是修改 PYTHONPATH
; environment=PYTHONPATH=$PYTHONPATH:/path/to/somewhere

[program:cat]
command=/bin/cat
process_name=%(program_name)s
```

```
numprocs=1
directory=/tmp
umask=022
priority=999
autostart=true
autorestart=true
startsecs=10
startretries=3
exitcodes=0,2
stopsignal=TERM
stopwaitsecs=10
user=chrism
redirect_stderr=false
stdout_logfile=/a/path
stdout_logfile_maxbytes=1MB
stdout_logfile_backups=10
stdout_capture_maxbytes=1MB
stderr_logfile=/a/path
stderr_logfile_maxbytes=1MB
stderr_logfile_backups=10
stderr_capture_maxbytes=1MB
environment=A="1",B="2"
serverurl=AUTO

;* 为必须填写项
;*[program: 应用名称]
[program:cat]

;* 命令路径, 如果使用 python 启动的程序应该为 python /home/test.py,
; 不建议放入/home/user/, 对于非 user 用户一般情况下是不能访问
command=/bin/cat

; 当 numprocs 为 1 时,process_name=%(program_name)s
; 当 numprocs>=2 时,%(program_name)s_%(process_num)02d
process_name=%(program_name)s

; 进程数量
numprocs=1

; 执行目录, 若有/home/supervisor_test/test1.py
; 将 directory 设置成/home/supervisor_test
; 则 command 只需设置成 python test1.py
; 否则 command 必须设置成绝对执行目录
directory=/tmp

; 掩码:--- -w- -w-, 转换后 rwx r-x w-x
umask=022

; 优先级, 值越高, 最后启动, 最先被关闭, 默认值 999
priority=999

; 如果是 true, 当 supervisor 启动时, 程序将会自动启动
autostart=true

;* 自动重启
autorestart=true

; 启动延时执行, 默认 1 秒
startsecs=10

; 启动尝试次数, 默认 3 次
startretries=3
```

```
; 当退出码是 0,2 时，执行重启，默认值 0,2
exitcodes=0,2

; 停止信号，默认 TERM
; 中断:INT(类似于 Ctrl+C)(kill -INT pid)，退出后会将写文件或日志 (推荐)
; 终止:TERM(kill -TERM pid)
; 挂起:HUP(kill -HUP pid)，注意与 Ctrl+Z/kill -stop pid 不同
; 从容停止:QUIT(kill -QUIT pid)
;KILL，USR1，USR2 其他见命令 (kill -l)，说明 1
stopsignal=TERM

stopwaitsecs=10

;* 以 root 用户执行
user=root

; 重定向
redirect_stderr=false

stdout_logfile=/a/path
stdout_logfile_maxbytes=1MB
stdout_logfile_backups=10
stdout_capture_maxbytes=1MB
stderr_logfile=/a/path
stderr_logfile_maxbytes=1MB
stderr_logfile_backups=10
stderr_capture_maxbytes=1MB

; 环境变量设置
environment=A="1",B="2"

serverurl=AUTO

command：启动程序使用的命令，可以是绝对路径或者相对路径
process_name：一个 python 字符串表达式，用来表示 supervisor 进程启动的这个的名称，默认值是
↪%(program_name)s
numprocs：Supervisor 启动这个程序的多个实例，如果 numprocs>1，则 process_name 的表达式必须包
含%(process_num)s，默认是 1
numprocs_start：一个 int 偏移值，当启动实例的时候用来计算 numprocs 的值
priority：权重，可以控制程序启动和关闭时的顺序，权重越低：越早启动，越晚关闭。默认值是 999
autostart：如果设置为 true，当 supervisord 启动的时候，进程会自动重启。
autorestart：值可以是 false、true、unexpected。false：进程不会自动重启，unexpected：当程序退出
时的退出码不是 exitcodes 中定义的时，进程会重启，true：进程会无条件重启当退出的时候。
startsecs：程序启动后等待多长时间后才认为程序启动成功
startretries：supervisord 尝试启动一个程序时尝试的次数。默认是 3
exitcodes：一个预期的退出返回码，默认是 0,2。
stopsignal：当收到 stop 请求的时候，发送信号给程序，默认是 TERM 信号，也可以是 HUP，INT，QUIT，
↪ KILL，USR1，or USR2。
stopwaitsecs：在操作系统给 supervisord 发送 SIGCHILD 信号时等待的时间
stopasgroup：如果设置为 true，则会使 supervisor 发送停止信号到整个进程组
killasgroup：如果设置为 true，则在给程序发送 SIGKILL 信号的时候，会发送到整个进程组，它的子进
程也会受到影响。
user：如果 supervisord 以 root 运行，则会使用这个设置用户启动子程序
redirect_stderr：如果设置为 true，进程则会把标准错误输出到 supervisord 后台的标准输出文件描述
符。
stdout_logfile：把进程的标准输出写入文件中，如果 stdout_logfile 没有设置或者设置为 AUTO，则
supervisor 会自动选择一个文件位置。
stdout_logfile_maxbytes：标准输出 log 文件达到多少后自动进行轮转，单位是 KB、MB、GB。如果设置
为 0 则表示不限制日志文件大小
stdout_logfile_backups：标准输出日志轮转备份的数量，默认是 10，如果设置为 0，则不备份
stdout_capture_maxbytes：当进程处于 stderr capture mode 模式的时候，写入 FIFO 队列的最大
bytes 值，单位可以是 KB、MB、GB
stdout_events_enabled：如果设置为 true，当进程在写它的 stderr 到文件描述符的时候，PROCESS_
↪LOG_STDERR 事件会被触发
```

stderr_logfile: 把进程的错误日志输出一个文件中，除非 redirect_stderr 参数被设置为 true
stderr_logfile_maxbytes: 错误 log 文件达到多少后自动进行轮转，单位是 KB、MB、GB。如果设置为 0 则表示不限制日志文件大小
stderr_logfile_backups: 错误日志轮转备份的数量，默认是 10，如果设置为 0，则不备份
stderr_capture_maxbytes: 当进程处于 stderr capture mode 模式的时候，写入 FIFO 队列的最大 bytes 值，单位可以是 KB、MB、GB
stderr_events_enabled: 如果设置为 true，当进程在写它的 stderr 到文件描述符的时候，PROCESS_LOG_STDERR 事件会被触发
environment: 一个 k/v 对的 list 列表
directory: supervisord 在生成子进程的时候会切换到该目录
umask: 设置进程的 umask
serverurl: 是否允许子进程和内部的 HTTP 服务通讯，如果设置为 AUTO，supervisor 会自动的构造一个 url
eg:

```
[program:import]
directory=/home/cxl/git-svn/spi/spi-php
command=php think queue:work --queue="importQueue" --tries=1 --daemon
process_name=import_%(process_num)s
numprocs=2
numprocs_start=1
autostart=true
startsecs=5
autorestart=true
startretries=3
user=www-data
;redirect_stderr=true
;stdout_logfile_maxbytes = 20MB
;stdout_logfile_backups = 20
;stdout_logfile = /data/logs/usercenter_stdout.log
;environment=PYTHONPATH=$PYTHONPATH:/path/to/somewhere
```

导入队列，使用两个进程处理，循环处理

CHAPTER

THREE

# PROGRAM

## 3.1 php

### 3.1.1 php7 功能点

1. 花括号 {}

```
1. 表示{} 里面的是一个变量  ，执行时按照变量来处理
2. 在字符串中引用变量使用的特殊包括方式，这样就可以不使用. 运算符，从而减少代码的输入量了。
$s = "Di, ";
echo ("${s}omething");  == echo $s."omething";
//Output: Di, omething

PHP 变量后面加上一个大括号{}，里面填上数字，就是指 PHP 变量相应序号的字符。
例如:
$str = 'hello';
echo $str{0}; // 输出为 h
echo $str{1}; // 输出为 e
如果要检查某个字符串是否满足多少长度，可以考虑用这种大括号（花括号）加 isset 的方式替代
↪strlen 函数，因为 isset 是语言结构，strlen 是函数，所以使用 isset 比使用 strlen 效率更高。
比如判断一个字符串的长度是否小于 5:
if ( !isset ( $str{5} ) ) 就比 if ( strlen ( $str ) < 5 ) 好。
```

2. call_user_func & call_user_func_array

```
区别 调用方式不同
    call_user_func(array($class,$method),param1,param2);
    call_user_func_array(array($class,$method),array(param1,param2));
注意如果上面$class 为字符串,$method 为非静态方法，则必须要先实例化，比如 $class = new
↪\app\class;
```

3. method_exists & is_callable

```
调用方式不同，is_callable
    if ( is_callable( array( $obj, $method ) ) )
    {
    /* 要操作的代码段 */
    }
    method_exists($obj,$method)
    $result = is_callable(['\app\sms\model\NoDisturbNumber','clearData']);
    dump($result);
    $result = method_exists('appsms\model\NoDisturbNumber', "clearData");
    dump($result);
其他
    php 函数 method_exists() 与 is_callable() 的区别在于在 php5 中，一个方法存在并不意味着它
就可以被调用。对于 private，protected 和 public 类型的方法，method_exits() 会返回 true，但是
is_callable() 会检查存在其是否可以访问，如果是 private，protected 类型的，它会返回 false。
```

4. trait

属性：如果 trait 定义了一个属性，那类将不能定义同样名称的属性，否则会产生一个错误。
如果类的定义是兼容的（同样的可见性和初始值）则错误的级别是 E_STRICT，否则是一个致命错误。

5. php7 新功能

```
1、运算符（NULL 合并运算符）
    $a = $_GET['a'] ?? 1; 相当于 $a = isset($_GET['a']) ? $_GET['a'] : 1;
2、函数返回值类型声明
  # declare(strict_types=1);
    function foo($a):int{
        return $a
    }
    foo(1.0); foo 函数返回 int 1，没有任何错误
    去掉上面代码的注释，采用严格模式，则会出发一个 TypeError 的 Fatal error。
3、标量类型声明
    function sumOfints(int ...$ints){
        return array_sum($ints)
    }
    var_dump(sumOfInts(2, '3', 4.1));
4、use 批量声明
    use some/namespace/{ClassA, ClassB, ClassC as C};
    use function some/namespace/{fn_a, fn_b, fn_c};
    use const some/namespace/{ConstA, ConstB, ConstC};
5、spaceship (<=>) 操作符
    用来比较两个表达式，左边小于、等于、大于右边时分别返回-1,0,1
6、常量 array 可以使用 define 定义
    define('ANIMALS', [
'dog',
'cat',
'bird'
    ]);
    echo ANIMALS[1]; // outputs "cat"
7、匿名类
    interface Logger {
public function log(string $msg);
    }
    class Application {
        private $logger;

        public function getLogger(): Logger {
             return $this->logger;
        }

        public function setLogger(Logger $logger) {
             $this->logger = $logger;
        }
    }
    $app = new Application;
    $app->setLogger(new class implements Logger {
        public function log(string $msg) {
            echo $msg;
        }
    });
    var_dump($app->getLogger());
    The above example will output:
    object(class@anonymous)#2 (0) {
    }
8、闭包 ( Closure) 增加了一个 call 方法
    class A {private $x = 1;}
    // Pre PHP 7 code
    $getX = function() {return $this->x;};
```

```
$getXCB = $getX->bindTo(new A, 'A'); // intermediate closure
echo $getXCB();

// PHP 7+ code
$getX = function() {return $this->x;};
echo $getX->call(new A);
```

### 3.1.2 tp5

#### queue

- refer

```
https://github.com/coolseven/notes/blob/master/thinkphp-queue/README.md
```

- install

```
composer require topthink/think-queue
```

- 例子
  - 配置

```
application/extra/queue.php
return [
        'connector'  => 'Redis',         // Redis 驱动
        'expire'     => 60,             // 任务的过期时间，默认为 60 秒；若要禁用，则设
置为 null
        'default'    => 'default',      // 默认的队列名称
        'host'       => '127.0.0.1',        // redis 主机 ip
        'port'       => 6379,           // redis 端口
        'password'   => '',             // redis 密码
        'select'     => 2,                  // 使用哪一个 db，默认为 db1
        'timeout'    => 0,              // redis 连接的超时时间
        'persistent' => false,          // 是否是长连接
];
```

  - 建立生产者

```
public function test(){
    // 1. 当前任务将由哪个类来负责处理。
    //   当轮到该任务时，系统将生成一个该类的实例，并调用其 fire 方法
    $jobHandlerClassName  = 'app\job\Import';
    // 2. 当前任务归属的队列名称，如果为新队列，会自动创建
    $jobQueueName     = "importQueue";
    // 3. 当前任务所需的业务数据 . 不能为 resource 类型，其他类型最终将转化为 json 形式的
字符串
    //   ( jobData 为对象时，需要在先在此处手动序列化，否则只存储其 public 属性的键值对)
    $jobData          = [ 'ts' => time(), 'bizId' => uniqid() , 'a' => 1 ] ;
    // 4. 将该任务推送到消息队列，等待对应的消费者去执行
    $isPushed = Queue::push( $jobHandlerClassName , $jobData , $jobQueueName );
    // database 驱动时，返回值为 1|false  ;   redis 驱动时，返回值为 随机字符串 |false
    if( $isPushed !== false ){
        echo date('Y-m-d H:i:s') . " a new Hello Job is Pushed to the MQ [$isPushed]
↪"."<br>";
    }else{
        echo 'Oops, something went wrong.';
    }
}
```

  - 查看 queue

```
llen "queues:helloJobQueue" 长度
lpop "queues:helloJobQueue" 出栈
lrange "queues:helloJobQueue" 0 -1 查询队列内容
```

– 建立消费者

```
namespace app\job;
use think\queue\Job;
use think\Log;

class Import{

    /**
     * fire 方法是消息队列默认调用的方法
     * @param Job            $job      当前的任务对象
     * @param array|mixed    $data     发布任务时自定义的数据
     */
    public function fire(Job $job,$data){
        $isJobDone = $this->doHelloJob($data);

        if ($isJobDone) {
            //如果任务执行成功， 记得删除任务
            $job->delete();
            print("<info>Hello Job has been done and deleted."."</info>\n");
        }else{
            if ($job->attempts() > 3) {
                //通过这个方法可以检查这个任务已经重试了几次了
                print("<warn>Hello Job has been retried more than 3 times!"."</warn>
↪\n");
                $job->delete();
                // 也可以重新发布这个任务
                //print("<info>Hello Job will be availabe again after 2s."."</info>\n
↪");
                //$job->release(2); //$delay 为延迟时间，表示该任务延迟 2 秒后再执行
            }
        }
    }

    /**
     * 根据消息中的数据进行实际的业务处理
     * @param array|mixed    $data     发布任务时自定义的数据
     * @return boolean                 任务执行的结果
     */
    private function doHelloJob($data) {
        // 根据消息中的数据进行实际的业务处理...
        print("<info>Hello Job Started. job Data is: ".var_export($data,true)."</
↪info> \n");
        print("<info>Hello Job is Fired at " . date('Y-m-d H:i:s') ."</info> \n");
        print("<info>Hello Job is Done!"."</info> \n");
        return true;
    }

    public function failed($data){

        // ... 任务达到最大重试次数后，失败了
        print("<info>Hello Job is failed!"."</info> \n");
        Log::write("has fail in import:");
    }
}
```

– 全局错误类

```
1. tags.php 标签配置
// 任务失败统一回调，有四种定义方式
'queue_failed'=> [
            ['app\\common\\behavior\\MyQueueFailedLogger', 'logAllFailedQueues']
        ],
2. fail 方法
    namespace app\common\behavior;
    use think\Log;
    class MyQueueFailedLogger{
        const should_run_hook_callback = true;

        /**
         * @param $jobObject   \think\queue\Job   //任务对象，保存了该任务的执行情况和
业务数据
         * @return bool     true                  //是否需要删除任务并触发其 failed()␣
↪方法
         */
        public function logAllFailedQueues(&$jobObject){

            /* $failedJobLog = [
                    'jobHandlerClassName'   => $jobObject->getName(), //
↪'application\index\job\Hello'
                    'queueName' => $jobObject->getQueue(),             //
↪'helloJobQueue'
                    'jobData'   => $jobObject->getRawBody()['data'],  // '{'a': 1 }'
                    'attempts'  => $jobObject->attempts(),            // 3
            ];
            var_export(json_encode($failedJobLog,true)); */
            Log::write("i am in failed method");
            //Log::write("failed in ".json_encode($failedJobLog,true));
            // $jobObject->release();     //重发任务
            //$jobObject->delete();         //删除任务
            //$jobObject->failed();   //通知消费者类任务执行失败

            return self::should_run_hook_callback;
        }
    }
```

– 测试

```
1. https://10.0.0.42:1066/home/index/test
    2017-09-23 10:04:18 a new Hello Job is Pushed to the MQ␣
↪[A25ZVpTcRIDUMyUgwSiRy76ebkLPuck8]
2. php think queue:work --queue="importQueue" --daemon
```

– 使用 supervisor 管理进程

```
[program:import]
directory=/home/cxl/git-svn/spi/spi-php
command=php think queue:work --queue="importQueue" --tries=1 --daemon
process_name=import_%(process_num)s
numprocs=2
numprocs_start=1
autostart=true
startsecs=5
autorestart=true
startretries=3
user=www-data
;redirect_stderr=true
;stdout_logfile_maxbytes = 20MB
;stdout_logfile_backups = 20
;stdout_logfile = /data/logs/usercenter_stdout.log
;environment=PYTHONPATH=$PYTHONPATH:/path/to/somewhere
```

```
导入队列，使用两个进程处理，循环处理
kill 一个进程会自动重启
```

## 3.2 shell

## 3.3 mysql

### 3.3.1 mysqldump

#### refer

https://www.cnblogs.com/chenmh/p/5300370.html

#### example

### 3.3.2 功能点

#### 索引长度限制

- **refer** http://blog.csdn.net/shaochenshuo/article/details/51064685
- **explain** From the manual at http://dev.mysql.com/doc/refman/5.6/en/create-table.html >> 从 5.6 的官方文档中我们能找到如下双引号中解释 "For CHAR, VARCHAR, BINARY, and VARBINARY columns, indexes can be created that use only the leading part of column values, using col_name(length) syntax to specify an index prefix length. ···Prefixes can be up to 1000 bytes long (767 bytes for InnoDB tables). Note that prefix limits are measured in bytes, whereas the prefix length in CREATE TABLE statements is interpreted as number of characters ···" >>> 对于 myisam 和 innodb 存储引擎，prefixes 的长度限制分别为 1000 bytes 和 767 bytes。注意 prefix 的单位是 bytes，但是建表时我们指定的长度单位是字符。A utf8 character can use up to 3 bytes. Hence you cannot index columns or prefixes of columns longer than 333 (MyISAM) or 255 (InnoDB) utf8 characters. >> 以 utf8 字符集为例，一个字符占 3 个 bytes。因此在 utf8 字符集下，对 myisam 和 innodb 存储引擎创建索引的单列长度不能超过 333 个字符和 255 个字符

#### 修改 mysql root 密码

1. use mysql;update user set password=PASSWORD( "C1oudP8x&2017" ) where user=" root" ;
2. flush privileges;
3. select Host,User,Password,authentication_string from user;

#### 语法

- insert into ···on duplicate key update ···

```
INSERT INTO table (a,b,c) VALUES (1,2,3) ON DUPLICATE KEY UPDATE c=c+1 UPDATE table SET␣
↪c=c+1 WHERE a=1;
```

## 3.4 vue

### 3.4.1 install

### 3.4.2 get_start

#### 创建项目::

vue init webpack vue_test

## 3.5 redis

### 3.5.1 refer

http://doc.redisfans.com/ http://www.redis.net.cn/tutorial/3501.html http://www.cnblogs.com/liuconglin/p/5847568.html 这是一篇讲各种数据类型的应用场景的文章，可做参考：http://blog.csdn.net/gaogaoshan/article/details/41039581/ redislive 是一款 redis 使用状态的软件具体可参考：http://blog.csdn.net/hz_blog/article/details/41822825 这篇文章讲了此工具如何安装具体不做说明地址：http://www.cnblogs.com/hepingqingfeng/p/6107809.html

### 3.5.2 install&research

#### 安装

$ wget http://download.redis.io/releases/redis-3.2.6.tar.gz
$ tar xzf redis-3.2.6.tar.gz
$ cd redis-3.2.6
$ make
$ src/redis-server 运行 redis 服务
$ src/redis-cli
redis> set foo bar
OK
redis> get foo
“bar”

#### 配置

- redis 配置

```
vi redis.conf
bind 10.0.0.23
daemonize yes
logfile "/var/pbx/tmp/Logs/redis/redis_6379.log"
dbfilename dump_6379.rdb
//生产环境
rename-command FLUSHALL ""
rename-command FLUSHDB ""
rename-command KEYS ""
rename-command CONFIG ""
maxmemory 2gb //占用的最大内存 <span style='color:green;font-weight:bold;'>< 一般推荐 Redis 设置内存为最大物理内存的四分之三，设置最大内存后一般需设置过期策略></span>
appendonly yes
```

```
appendfilename "appendonly_6379.aof"
requirepass "prettydogKnockTheDoor" //需要密码才能访问
将 redis 加入到系统自启动的脚本中

Redis 使用超过设置的最大值
打开 debug 模式下的页面，提示错误：OOM command not allowed when used memory > ‘maxmemory’.
设置了 maxmemory 的选项，redis 内存使用达到上限。可以通过设置 LRU 算法来删除部分 key，释放空间。默认是按照过期时间的，如果 set 时候没有加上过期时间就会导致数据写满 maxmemory。
如果不设置 maxmemory 或者设置为 0，64 位系统不限制内存，32 位系统最多使用 3GB 内存。

LRU 是 Least Recently Used 近期最少使用算法。
volatile-lru -> 根据 LRU 算法生成的过期时间来删除。
allkeys-lru -> 根据 LRU 算法删除任何 key。
volatile-random -> 根据过期设置来随机删除 key。
allkeys->random -> 无差别随机删。
volatile-ttl -> 根据最近过期时间来删除（辅以 TTL）
noeviction -> 谁也不删，直接在写操作时返回错误。
```

**Note:** 详细配置可参考文章: 这篇文章主要讲的是配置的含义和优化 *<http://www.tuicool.com/articles/MvMBf2>*

- iptable 配置

```
考虑到安全，不允许外网用户访问 redis

10.0.0.0/24 表示 C 类地址 24 掩码前面的 0
iptables -A INPUT ! -s 10.0.0.0/24 -p tcp --dport 6379 -j DROP //拒绝所有非本网段的机器访问 redis
iptables -A INPUT ! -s 10.0.0.0/24 -p udp --dport 6379 -j DROP
保存 iptables 规则下次启动时自动启动规则
iptables-save > /etc/iptables.up.rules //保存规则
vi /etc/network/interfaces
pre-up iptables-restore < /etc/iptables.up.rules  //恢复规则
iptables -F
重启网络
/etc/init.d/networking restart
iptables -L -v
重启机器
iptables -L -v
可以看到刚刚配置的规则都在
```

### 测试

- redis 数据类型测试

```
设置值，获取值
* set foo "bar"
* get foo
* expire foo 120
* ttl foo 查询过期时间 返回-2 值不存在 -1 值永不过期

自增变量
set connections 10
incr connections
del connections

list 队列
rpush friends "alice" 添加到队尾
rpush friends "bob"
```

```
lpush friends "sam" 添加到队首
lrange friends 0 -1 获取全部队列
lrange friends 0 1 获取 0-1
llen friends 队列长度
lpop friends 弹出首个对象
rpop friends 弹出队尾对象

set 与 list 相似，但没有顺序且对象只能出现一次
sadd superpowers "flight"   添加值
sadd superpowers "x-ray vision"
sadd superpowers "reflexes"
srem superpowers "reflexes"  删除值
sismember superpowers "reflexes"  判断是否存在 0 不存在 1 存在
smembers superpowers
sadd birdpowers "flight"
sadd birdpowers "pecking"
sunion superpowers birdpowers 合并 set

sorted set 可排序 set
zadd hackers 1940 "alan key"
zadd hackers 1980 "Grace hopper"
zadd hackers 1945 "richard stallman"
zadd hackers 1944 "macker"
zrange hackers 0 -1

hashes
hset user:1000 name "john smith"
hset user:1000 email "john.smith@example.com"
hset user:1000 password "s3cret"
hgetall user:1000
HMSET user:1001 name "Mary Jones" password "hidden" email "mjones@example.com"
SET user:1000 visits 10
HINCRBY user:1000 visits 1 加 1
hdel user:1000 visits 删除值
```

**Note:** 测试参考文档 *<http://www.cnblogs.com/silent2012/p/4514901.html>*

- 安全测试
  - 只有使用密码才能访问

```
root@1604developer:/var# telnet 10.0.0.23 6379
Trying 10.0.0.23...
telnet: Unable to connect to remote host: Connection refused
root@1604developer:/var# telnet 10.0.0.23 6379
Trying 10.0.0.23...
Connected to 10.0.0.23.
Escape character is '^]'.
auth prettydogKnockTheDoor
+OK
keys *
*0
```

  - 指定 ip 才能访问

```
iptables -L -v --line-num
配置 iptables 只允许 10.0.0.1-32 的 ip 地址访问 redis
iptables -A INPUT ! -s 10.0.0.0/27 -p tcp --dport 6379 -j DROP
iptables -A INPUT ! -s 10.0.0.0/27 -p udp --dport 6379 -j DROP
在本机访问
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# telnet 10.0.0.23 6379
```

```
Trying 10.0.0.23...
Connected to 10.0.0.23.
Escape character is '^]'.
auth prettydogKnockTheDoor
+OK
keys *
*0
在 10.0.0.237 机器访问
root@php1 16:51:00:~# telnet 10.0.0.23 6379
Trying 10.0.0.23...
如果策略修改为
iptables -A INPUT ! -s 10.0.0.0/27 -p tcp --dport 6379 -j REJECT
iptables -A INPUT ! -s 10.0.0.0/27 -p udp --dport 6379 -j REJECT
再次在 10.0.0.237 机器访问
root@php1 16:53:23:~# telnet 10.0.0.23 6379
Trying 10.0.0.23...
telnet: Unable to connect to remote host: Connection refused
```

- 数据恢复测试
  - kill redis 进程

```
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# src/redis-cli -h 10.0.0.23
10.0.0.23:6379> auth prettydogKnockTheDoor
OK
10.0.0.23:6379> set "first Key" "first Value"
OK
10.0.0.23:6379> get "first Key"
"first Value"
10.0.0.23:6379> quit
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# ps -aux |grep redis
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
root      4418  0.1  0.3  36124  1564 ?        Ssl  17:54   0:00 src/redis-server 10.
↪0.0.23:6379
root      4497  0.0  0.1   9380   936 pts/2    S+   17:57   0:00 grep --color=auto␣
↪redis
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# kill 4418
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# src/redis-server redis.conf
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# src/redis-cli -h 10.0.0.23
10.0.0.23:6379> auth prettydogKnockTheDoor
OK
10.0.0.23:6379> get "first Key"
"first Value"
10.0.0.23:6379> quit
```

  - 重启 redis 进程，查看 redis 数据

```
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# reboot
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# src/redis-server redis.conf
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# src/redis-cli -h 10.0.0.23
10.0.0.23:6379> auth prettydogKnockTheDoor
OK
10.0.0.23:6379> get "first Key"
"first Value"
10.0.0.23:6379>
```

- 性能测试
  - 内存占用测试

```
vi redis.conf
maxmemory 1m //为了测试方便设置为 1mb
vi addKey.sh
```

```
# !/usr/bin/bash

echo "AUTH prettydogKnockTheDoor"
sleep 1

outstr=""
value10="1"
j=700
while [ $j -gt 690 ]
do
for i in {100..199}
do
outstr="${outstr}set key$j-$i ${value10}\r\n"
done
printf "$outstr"
outstr=""
j=`expr $j - 1`
done
这个脚本往 redis 服务器加入 1000 个 key，其中 Key 值 10 个字节，value 值一个字节

10.0.0.23:6379> flushall //清空 redis
10.0.0.23:6379> info
# Memory
used_memory:821552
used_memory_human:803.70K
看出来，redis 初始状态就占用 803.70k，所以设置 1m 最大内存，实际可用应该是 200k 左右
运行脚本插入 1000 个键值

> 方式 1
./addkey.sh | nc 10.0.0.23 6379

> 方式 2
./addkey.sh | ../redis-3.2.6/src/redis-cli -h 10.0.0.23 -a prettydogKnockTheDoor --
↪pipe
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 100001


10.0.0.23:6379> info
# Memory
used_memory:871184
used_memory_human:850.77K

占用内存 47k，所以 11 字节的 key+value 大概占用 47 字节的内存空间
修改一下脚本加入 4000key
-OOM command not allowed when used memory > 'maxmemory'.
10.0.0.23:6379> dbsize
(integer) 3109
只加入了 3109 键值
used_memory:980120
used_memory_human:957.15K
```

– 大并发测试

```
50 个客户端发送 100000 请求每个请求 3 个字节，看起来 redis 很给力
root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# src/redis-benchmark -h 10.0.0.23 -q
(<span style='color:red;font-weight:bold'> 本机测试</span>)
PING_INLINE: 75131.48 requests per second
PING_BULK: 74074.07 requests per second
SET: 76277.65 requests per second
GET: 74349.44 requests per second
INCR: 75815.01 requests per second
```

```
LPUSH: 74074.07 requests per second
RPUSH: 73421.44 requests per second
LPOP: 74794.31 requests per second
RPOP: 74571.22 requests per second
SADD: 74906.37 requests per second
SPOP: 74794.31 requests per second
LPUSH (needed to benchmark LRANGE): 73367.57 requests per second
LRANGE_100 (first 100 elements): 72621.64 requests per second
LRANGE_300 (first 300 elements): 74515.65 requests per second
LRANGE_500 (first 450 elements): 74019.25 requests per second
LRANGE_600 (first 600 elements): 76277.65 requests per second
MSET (10 keys): 73746.31 requests per second

<span style='color:red;font-weight:bold'> 内网跨服务器测试</span>
PING_INLINE: 13596.19 requests per second
PING_BULK: 13696.75 requests per second
SET: 13676.15 requests per second
GET: 13738.15 requests per second
INCR: 13738.15 requests per second
LPUSH: 13664.94 requests per second
RPUSH: 13585.11 requests per second
LPOP: 13730.61 requests per second
RPOP: 13719.30 requests per second
SADD: 13702.38 requests per second
SPOP: 13691.13 requests per second
LPUSH (needed to benchmark LRANGE): 13655.61 requests per second
LRANGE_100 (first 100 elements): 13650.01 requests per second
LRANGE_300 (first 300 elements): 13648.15 requests per second
LRANGE_500 (first 450 elements): 13550.14 requests per second
LRANGE_600 (first 600 elements): 13674.28 requests per second
MSET (10 keys): 13220.52 requests per second
```

**Note:** 结论：本机访问是内网跨服务器访问各类请求每秒执行数量五倍多

- 监控性能

```
>>top 监控
 root@ubuntu1204base:/home/cxl/redis/redis-3.2.6#  top -b -p 16417 -n 2|egrep "16417|PID
↪"|tail -2
 PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 16417 root      20   0 42268 8980 1240 S  0.0  1.8   0:02.70 redis-server

>>redis-cli info 监控

 root@ubuntu1204base:/home/cxl/redis/redis-3.2.6# src/redis-cli -h 10.0.0.23 -a␣
↪prettydogKnockTheDoor info|grep used_memory
 used_memory:7470144
 used_memory_human:7.12M
 used_memory_rss:9023488
 used_memory_rss_human:8.61M
 used_memory_peak:7470144
 used_memory_peak_human:7.12M
 used_memory_lua:37888
 used_memory_lua_human:37.00K
```

### 代码示例

- predis

```
好处是不需要安装 php 扩展，直接 require 就行了
git clone git://github.com/nrk/predis.git
wget https://github.com/nrk/predis/archive/v1.1.1.tar.gz
拷贝到 Thinkphp vendor 目录下

示例代码
<pre>
       require_once VENDOR_PATH."predis/autoload.php";
       try {
           $redis = new Predis\Client("redis://127.0.0.1:6379/");
           $redis->set('library', 'predis1');
           $retval = $redis->get('library1');
           var_export($retval);
       } catch (Exception $e) {
           var_dump($e->getMessage());
       }

</pre>

predis 函数集：http://www.open-open.com/lib/view/open1355830836135.html
```

### 3.5.3 usage

- key

```
1、help set
SET key value [EX seconds] [PX milliseconds] [NX|XX]
summary: Set the string value of a key
since: 1.0.0
group: string

set test 10
set test 10 EX 60 60 秒过期
```

- **list**

  1. **查询 list** lrange queues:importQueue 0 -1
  2. **从队列头 pop 元素，在队列里删除该元素** lpop queues:importQueue
  3. **清空队列** ltrim queues:importQueue 1 0

- **set**

  – zset（sorted set）操作相关

```
1. 查询
    zrange queues:importQueue:reserved 0 -1
2. 删除所有元素
    zrem queues:importQueue:reserved 0 -1
```

  – hash

```
1. 查看 hash 中的值和 value
    hgetall job_result
    1) "importQueue_p0yzx4AQEcVefgUnvZ9nLLiUprwP6ff5"
    2) "{\"job_status\":1}"
2. 删除值
    hDel job_result importQueue_p0yzx4AQEcVefgUnvZ9nLLiUprwP6ff5
    (integer) 1
```

## 3.6 nodejs

### 3.6.1 example

#### 切换到淘宝镜像

npm install -g cnpm –registry=https//registry.npm.taobao.org

CHAPTER

# FOUR

# PROJECT

## 4.1 spi

spi project

CHAPTER

FIVE

# VERSION CONTROL

## 5.1 git

### 5.1.1 refer

### 5.1.2 command example

- 使用远程覆盖本地

```
git fetch --all
git reset --hard origin/master
```

- 错误:fatal: Pathspec ‘shell-baselib/cxl_cli’ is in submodule ‘shell-baselib

```
git rm --cached shell-baselib/
git add shell-baselib/
```

- merge 其他分支（或远程分支），有冲突是使用远程覆盖本地

```
git svn clone svn://192.168.1.66/namtso/branch/web_code/svntest/front -r31772:HEAD
git remote add origin git@gitlab28:chenxuelin/spifront
git fetch origin master
git merge -m "merge from git" -X theirs origin/master
echo xuelin|git svn clone "svn://192.168.1.66/EmicallDev/ApplicationPlatform/sandbox/
↪WebApplication/spi-front" -r3995:HEAD --username chenxl
git remote add origin git@gitlab28:websrc/vuefront

git svn clone svn://192.168.1.66/namtso/branch/web_code/svntest/front -r31772:HEAD
git remote add origin git@gitlab28:chenxuelin/spifront
git fetch origin master
git reset --hard origin/master

git merge -X theirs -m "merge from git when confict use theirs" origin master
sed -i '/^dist\//d' .gitignore
npm config set registry " https://registry.npm.taobao.org "
svn info svn://192.168.1.66/EmicallDev/ApplicationPlatform/sandbox/WebApplication/spi-
↪front | awk '($1=="Last"){if($3=="Rev:"){print $4}}'
```

- 开发使用 git，打版本使用 svn

```
echo xuelin|git svn clone "$svnfront" -r"$svnLastVersion":HEAD --username chenxl
git remote add origin "$gitfront"
git fetch origin master
git merge -m "merge from git" -X theirs origin/master
git svn dcommit
```

- 配置 ssh 方式连接 github 和本地 gitlab

```
# cd ~/.ssh
# ssh-keygen -t rsa -C "303566@qq.com" -f ~/.ssh/github
# 在目录下有两个文件 github 和 github.pub, cat github.pub 拷贝内容
# 登录 github》》 your profile》》 ssh keys》》 粘贴内容到 sshkey
# ssh-keygen -t rsa -C "chenxuelin@emicnet.com" -f ~/.ssh/gitlib-cxl
# 在目录下有两个文件 gitlib-cxl 和 gitlib-cxl.pub, cat gitlib-cxl.pub 拷贝内容
# 登录 gitlib》》 your profile》》 ssh keys》》 粘贴内容到 sshkey
# 创建~/.ssh/config
#303566-github
  host github
  user git
  hostname github.com
  port 22
  identityfile ~/.ssh/github
  #chenxuelin-gitlib28
  host gitlib28
  user git
  hostname 10.0.0.28
  port 22
  identityfile ~/.ssh/gitlib-cxl
# 测试
## root@1604developer:~/.ssh# ssh -T gitlib28
    Welcome to GitLab, chenxuelin!
## root@1604developer:~/.ssh# ssh -T github
    Hi xuelinchen! You've successfully authenticated, but GitHub does not provide shell␣
↪access.
# 在 github 创建项目
   git clone git@github:xuelinchen/schedule.git
   # 增加 remote 仓库
   git remote add gitlib28 git@gitlib28:chenxuelin/schedule.git
   # 提交结果到远程仓库
   git push gitlib28
```

### 5.1.3 git2svn

# clone git project

```
git clone git@gitlib28:chenxuelin/schedule.git
```

# check out svn project with same directory of git project

```
svn co svn://192.168.1.66/namtso/branch/web_code/svntest schedule --username chenxl --password␣
↪xuelin
```

# add ignore of svn,attention that has a enter after git and has dot in the end

```
svn propset svn:ignore ".git
.gitignore" .
svn status 查看状态
```

# commit to svn

```
svn add *
svn commit -m "result from git"  --username chenxl --password xuelin
```

# commit to git

```
git add *
git commit -m "add .svn path to git res"
git push
```

# add test file in git and check in to svn

```
vi test.md
git add test.md
git commit -m 'add test.md'
svn add test.md
svn commit -m 'check in file from git'  --username chenxl --password xuelin
git commit -m 'delete test.md'
git pull
svn add * --force
svn commit -m 'test add file' --username chenxl --password xuelin
删除文件无法自动提交到 svn 中
```

# git-svn

```
echo xuelin|git svn clone svn://192.168.1.66/namtso/branch/web_code/tp5-project -r29318:HEAD --
↪username chenxl
git branch
git remote add origin git@gitlib28:chenxuelin/schedule.git
git pull origin master ctrl+x 提交
git branch --set-upstream-to=origin/master
强制刷新本地版本为远程仓库
git fetch --all
git reset --hard origin/master
```

# Append svn:ignore settings to the default Git exclude file

```
git svn show-ignore >> .git/info/exclude
git svn dcommit
```

# from git commit to svn

```
1、create new clone project
    git clone git@gitlib28:chenxuelin/schedule
    cd schedule
2、create new svn clone project
    echo xuelin|git svn clone svn://192.168.1.66/namtso/branch/web_code/tp5 -r29318:HEAD --
↪username chenxl
    cd tp5
    git remote add origin git@gitlib28:chenxuelin/schedule.git
```

# git version commit to svn

```
http://blog.csdn.net/zhangskd/article/details/43452627
1. git clone git@gitlib28:chenxuelin/schedule
2. git svn init svn://192.168.1.66/namtso/branch/web_code/tp5
3. git svn fetch
4. git show-ref svn | tail -n 1
f7e97acecbb8098757d9dc451b3c76eb59c4da9d refs/remotes/origin/master
5. git log --pretty=oneline master | tail -n 1
    8041fab27f1ff38f980b9ea00fd335c50005af8c nf:  create svntest project
6. echo "8041fab27f1ff38f980b9ea00fd335c50005af8c 5194b04324e1fa8bcec215ec053c5e707d2e4b83" >>␣
↪.git/info/grafts
    试验失败
```

# another

```
1、echo xuelin|git svn clone svn://192.168.1.66/namtso/branch/web_code/tp5 -r29318:HEAD --
↪username chenxl
2、git remote add origin git@gitlib28:chenxuelin/schedule.git
3、git branch --set-upstream-to=origin/master master
4、git pull origin
5、git svn dcommit
```

```
6、git checkout -b newfeature
7、git checkout master
8、git merge newfeature
9、git push origin
10、git svn dcommit
```

# schedule project

```
1. install git-svn
   apt install git-svn
2. clone svn project,that is empty project
   echo xuelin|git svn clone svn://192.168.1.66/namtso/branch/web_code/schedule -r29318:HEAD --
↪username chenxl
3. update from svn
   git svn rebase
4. find all branch
   git branch -a
5. add remote git repository
   git remote add origin git@gitlab28:chenxuelin/schedule.git
6. set master branch upstream
   git branch --set-upstream-to=origin/master master
7. pull data from remote git
   git pull origin ==》 ctrl+x
8. submit to svn
   git svn dcommit
9. develope in branch
   git checkout -b "cxl"
   git rm test.md
   git commit -m "made some change"
10.return to master for push to remote of git&svn
   git checkout master
   git merge cxl
   git push origin master
   git svn dcommit
11.add ignore from git to svn
   git svn show-ignore >> .git/info/exclude
```

# tp5 project

```
1. clone svn project,that is empty project
   echo xuelin|git svn clone svn://192.168.1.66/namtso/branch/web_code/spi -r29318:HEAD --
↪username chenxl
2. cd spi
3. add remote git repository
   git remote add origin git@gitlab28:websrc/spi.git
4. make some change
5. git add .
6. git commit -m 'add ignore'
7. submit to remote git
        git push -u origin master
8. submit to svn
   git svn dcommit
```

# spi project

```
1. clone svn project spi-front
        echo xuelin|git svn clone svn://192.168.1.66/EmicallDev/ApplicationPlatform/sandbox/
↪WebApplication/spi-front -r3000:HEAD --username chenxl
2. clone svn project spi-php
        echo xuelin|git svn clone svn://192.168.1.66/EmicallDev/ApplicationPlatform/sandbox/
↪WebApplication/spi-php -r3000:HEAD --username chenxl
3. cd spi-php
```

```
4. add remote git repository
   git remote add origin git@gitlab28:websrc/spi.git
5. view remote info
   git remote -v
6. pull data from remote git
        git pull origin
7. git branch -a
   * master
        remotes/git-svn
        remotes/origin/master
8. merge origin/master to local branch master
        git merge remotes/origin/master ==》 ctrl+x
9. add ignore from git to svn
   git svn show-ignore >> .git/info/exclude
10.create branch of "bin"
   git checkout -b "bin"
11.commit data to svn
        git svn dcommit
12.cd spi-front
13.add remote git repository
        git remote add origin git@gitlab28:websrc/vuefront.git
14.pull data from remote git
        git pull origin master => ctrl+x
15.add ignore from git to svn
        git svn show-ignore >> .git/info/exclude
16.commit data to svn
        git svn dcommit
```

## 5.2 svn

### 5.2.1 refer

### 5.2.2 install

- 安装 1.8

```
sudo sh -c 'echo "deb http://opensource.wandisco.com/ubuntu precise svn18" >> /etc/apt/
↪sources.list.d/subversion18.list'
sudo wget -q http://opensource.wandisco.com/wandisco-debian.gpg -O- | sudo apt-key add -
sudo apt-get update
sudo apt-cache show subversion | grep '^Version:'
sudo apt-get install subversion
```

### 5.2.3 usage

- 创建项目目录

```
mkdir svn
```

- 创建仓库

```
svnadmin create /home/svn/project
chmod 777 -R /home/svn/project
```

- 查看 svn 服务器是否支持 cyrus sasl 方式认证

```
svnserve --version
是否存在 Cyrus SASL authentication is available.
```

- 配置权限

```
1、vi /home/svn/project/conf/svnserve.conf
打开 auth-access = write 授权用户可写
password-db = passwd
2、vi /home/svn/project/conf/passwd
添加用户
[users]
# harry = harryssecret
# sally = sallyssecret
cxl = xuelin
wqx = qinxue
3、vi /home/svn/project/conf/authz
admin = cxl
@admin = rw
* =
admin=cxl, cxl 用户属于 admin 权限组, @admin=rw, admin 权限组可以 read, 其他用户不能读取
```

- 守护进程方式启动 svn

```
svnserve -d -r /home/svn
```

- svnsync 同步测试

  1、在源地址创建项目 svn://192.168.1.66/namtso/branch/web_code/svnsynctest 2、vi /home/svn/svnsynctest/hooks/pre_revprop-change.bat 添加一行 exit 0 3、配置权限，同上 4、初始化：svnsync init –source-username chenxl –source-password xuelin –sync-username chenxl –sync-password xuelin svn://10.0.0.21/svnsynctest svn://192.168.1.66/namtso/branch/web_code/svnsynctest 5、同步版本 svnsync sync –source-username chenxl –source-password xuelin –sync-username chenxl –sync-password xuelin svn://10.0.0.21/svnsynctest