软件技术开发通用知识

版本：v1.3

Crifan Li

摘要

本文主要介绍了软件技术领域内，做软件开发期间，常见的，通用的，共同的，一些知识。

本文提供多种格式供：

在线阅读	HTML [1]	HTMLs [2]	PDF [3]	CHM [4]	TXT [5]	RTF [6]	WEBHELP [7]
下载（7zip压缩包）	HTML [8]	HTMLs [9]	PDF [10]	CHM [11]	TXT [12]	RTF [13]	WEBHELP [14]

HTML版本的在线地址为：

http://www.crifan.com/files/doc/docbook/soft_tech_common/release/html/soft_tech_common.html

有任何意见，建议，提交bug等，都欢迎去讨论组发帖讨论：

http://www.crifan.com/bbs/categories/soft_tech_common/

修订历史		
修订 1.3	2015-05-22	crl

1. 添加技术框架的Host和Device
2. 添加技术中的HS是高速的意思
3. profile某种领域的协议和技术
4. 技术说明Technical Notes，Image镜像，前提条件和辅助条件
5. 添加了协议栈介绍
6. 添加了通用的做事情的想法，方法
7. 添加CLI和GUI
8. 添加portfolio

[1] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/html/soft_tech_common.html
[2] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/htmls/index.html
[3] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/pdf/soft_tech_common.pdf
[4] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/chm/soft_tech_common.chm
[5] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/txt/soft_tech_common.txt
[6] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/rtf/soft_tech_common.rtf
[7] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/webhelp/index.html
[8] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/html/soft_tech_common.html.7z
[9] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/htmls/index.html.7z
[10] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/pdf/soft_tech_common.pdf.7z
[11] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/chm/soft_tech_common.chm.7z
[12] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/txt/soft_tech_common.txt.7z
[13] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/rtf/soft_tech_common.rtf.7z
[14] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/webhelp/soft_tech_common.webhelp.7z

9. 添加协议分析工具

软件技术开发通用知识:

Crifan Li

版本：v1.3

出版日期 2015-05-22

[15] http://www.crifan.com/files/doc/docbook/soft_dev_basic/release/html/soft_dev_basic.html#cc_by_nc

目录

插图清单

前言

1. 本文目的

本文目的在于，介绍软件技术开发方面的通用知识和概念。

第 1 章 通用软件开发之技术点

此处整理一些，软件开发期间，会遇到的一些通用的技术概念或名词。

1.1. 主机Host和设备Device

很多技术的框架都是有Host和Device（Controller）

比如：

- 蓝牙
  蓝牙技术中，也有主机端Host和设备Device端，的概念。其中蓝牙中将设备端，有时候也叫做Controller端，指的是设备控制器，设备端。

  详见：蓝牙协议详解[1]

- USB
  USB协议中，也有主机端和设备端的概念。

  详见：USB基础知识概论 - USB系统的核心是Host[2]

1.2. profile

profile，作为英文单词，本意是：侧面；轮廓；外形；剖面；简况

而在软件技术领域内profile的的含义，可以翻译为：

不同的协议层次，档次，应用领域划分，对应不同的协议和规范

再白话点说就是：（某种技术）根据应用领域不同，而专门划分出来的一部分协议或规范的总和

比如：

- DVB中的profile
  卫星通讯，DVB，方面也有相关不同profile的划分

  DVB，根据应用领域不同，可以分为：

  - DVB-S：S指的是Satellite，卫星。用于卫星领域的DVB技术
  - DVB-T：T指的是Terrestrial television，用于地面无线广播中的DVB技术
  - DVB-C：C指的是Cable，用于有线领域的DVB技术
- PROFINET中的profile
  PROFINET也是有多个profile的协议：
  - PROFINET ：用于编码器方面的
  - PROFIdrive：用于运动控制的
  - PROFIsafe：用于功能安全领域
  - PROFIenergy：用于能源需求领域
  - 用于火车领域

[1] http://www.crifan.com/files/doc/docbook/bluetooth_intro/release/html/bluetooth_intro.html
[2] http://www.crifan.com/files/doc/docbook/usb_basic/release/html/usb_basic.html#core_is_host

详见：【整理】现场总线技术：PROFINET | 在路上[3]

- 蓝牙中的profile
  蓝牙协议中的profile的概念更明显：

  蓝牙中，根据应用领域不同，针对不同领域，定制了很多profile，相当于子协议。

  相关的子协议，非常多。具体的解释参见协议的详细介绍：

  详见：蓝牙协议详解[4]

1.3. portfolio产品组合

在软件开发期间，涉及到软硬件产品时，常常会听说到一个词：portfolio

其英文原意是：n. 公文包；文件夹；证券投资组合；部长职务

而实际上此处的含义，更接近于：产品组合，产品线，一系列的产品，产品家族

此处列举很多例子来解释其含义，如下：

- i.MX portfolio
  [ZT]Freescale’s i.MX and Micrium’s µC/OS-II | 在路上[5]

  中的：

  The i.MX portfolio is a central feature of Freescale"s i.Smart smartphone reference design, providing power performance to our Innovative Convergence platforms.

- ACS portfolio
  New English Words v2011-10-31 | 在路上[6]

  中的：

  XXX discussed the four key growth technologies pervasive across the ACS portfolio.

- SPI NOR portfolio
  Micron Technology, Inc. - Serial NOR Flash[7]

  中的：

  The broadest SPI NOR portfolio in the market

- Micron Flash Portfolio
  Micron Technology, Inc. - Serial NOR Flash[8]

  中的：

  Micron Flash Portfolio

  Micron has one of the broadest portfolios of NOR and NAND Flash solutions in the industry.

  We have products that are tailored to meet your application requirements, including a strong product offering for automotive applications, which require more reliability than others.

[3] http://www.crifan.com/summary_filedbus_profinet_overview/
[4] http://www.crifan.com/files/doc/docbook/bluetooth_intro/release/html/bluetooth_intro.html
[5] http://www.crifan.com/zt_freescale39s_imx_and_micrium39s_c__os-ii/
[6] http://www.crifan.com/new_english_words_v2011-10-31/
[7] http://www.micron.com/products/nor-flash/serial-nor-flash
[8] www.micron.com/~/media/Documents/Products/Product%20Flyer/NOR_NAND_Flash_Guide_lo.pdf

1.4. HS=High Speed=高速

HS==High Speed==高速（传输速率）

有些技术，涉及到传输速度的话，会提到高速，High Speed。

比如USB 2.0的HS，Bluetooth 3.0+HS。

高速的高，是相对而言的，是相对于更低一些的速度来说的：

- USB的HS
  USB 2.0的HS是480Mb/s，是相对于USB 1.1的Low Speed（1.5Mbs/s）和Full Speed（12Mb/s）而言更高一些。

  不过后来的USB 3.0的Super Speed（5.0Gb/s），则是比HS更高。

- 蓝牙的HS
  之前版本的蓝牙的速度就相对较低，而后来的标准Bluetooth 3.0，支持更高的速度，所以就叫做HS，意思是更高的速度，高速了。

1.5. 技术说明Technical Notes

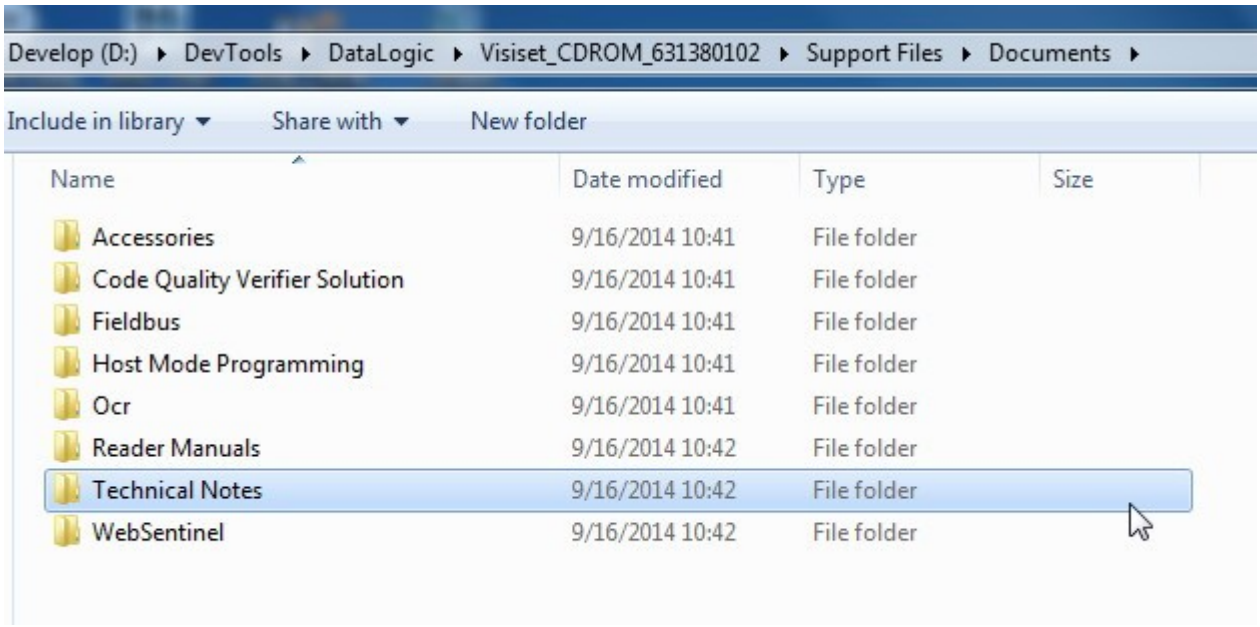
很多设备或产品的手册中，都会出现一个：Technical Notes，中文暂翻译为技术说明

此技术说明的主要作用是：针对此处涉及的技术，尤其是产品相关的技术，如何使用，如何将其应用到产品，需要哪些注意事项，等等这方面的内容，都专门写个说明，让技术的使用者更快的上手使用。

比如：

- DataLogic的扫描枪的手册中就有Technical Notes

图 1.1. DataLogic扫描枪手册中的Technical Notes



1.6. Image==镜像==镜像文件

在计算机领域，包括计算机软件的应用和开发，都可能会遇到Image这个词汇。

Image，本意是：图像，影像。

在计算机领域内，往往指的是，对应的一个文件，该文件包含了相关的软件的二进制文件。

而相关的软件，往往指的是：操作系统。

而操作系统，有的指的是普通的桌面级的，比如Win7，也有的指的是嵌入式领域内的，比如嵌入式Linux

所以，你可能会看到这样的说法：

- 给我个Win7镜像，我拿去刻盘
  此处的镜像Image，指的是就是：

  对于微软开发的Win7这个桌面操作系统来说，往往都是对应的.iso文件

  是可以刻录到光盘（或者启动U盘中模拟出来的光盘CDROM）中。

  然后用词光盘，就可以启动笔记本电脑，然后按照正常的操作步骤去安装Win7这个操作系统了。

  即：

  桌面级操作系统的（往往是ISO的，可刻录用于启动和安装系统的）镜像文件
- 通过其他把Linux的image烧录到板子上去
  此处的image，就是指的是：

  在嵌入式Linux系统开发过程中，已经把Linux源码编译成为Linux的二进制文件了。

  ->这个二进制文件，就是对应某嵌入式开发板的真正运行时候所需要去运行的嵌入式Linux系统。

  ->即嵌入式版本的Linux镜像文件。

1.7. pre-requisite前提条件和辅助条件co-requisites

除了前提条件，还有个词是：辅助条件

co-requisites是辅助条件的意思。

在写教程解释说明某东西之前,往往有个pre-requisite

prerequisite，表示前提条件：表示你要学习下面将要介绍的内容之前，本身需要掌握何种背景知识，相关基础知识，才能看得懂。

举个最简单的例子：

要是你的小学老师在叫你数学中的乘法之前，你还没学过加减法的话，那自然很难，无法学会乘法

此时，在老师教你乘法之前，就需要你掌握，加减法，这个背景的基础知识。

也就叫做：

在学习乘法之前，要先有前提条件：已经学会掌握了加减法。加减法，就是乘法的前提条件。

1.8. 协议栈

协议栈往往指的是：

相对于普通的某个技术的单个的协议而言，是一个协议的集合

就像计算机领域内的栈的概念一样

是有层次划分的，是多个协议一点一点地堆起来的

其中的单个的协议，设计的初衷就是用于该协议栈的某一层

比如：

- TCP/IP协议栈
  是最典型的代表

- USB的协议栈
  USB也有协议栈

  详见：USB协议概览[9]

- 蓝牙也有协议栈
  虽然各种协议主要是依据profile被划分成不同应用领域的，但是也是分层次的，所以也叫做蓝牙的协议栈

  详见：蓝牙协议的架构[10]

1.9. 协议分析工具

很多技术，协议都有专门的协议分析工具，便于开发调试

比如：

- USB协议分析工具
  有很多个USB协议分析工具，包括国内公司做的，和国外公司做的。

  TODO：详见：

  其他一些USB测试和协议分析等软件[11]

- 蓝牙协议分析工具
  在：

  CC2541发送超长字节内容的处理方式？（500~1000字节） - 蓝牙Bluetooth 技术 - 德州仪器在线技术支持社区[12]

  中看到的：

图 1.2. 蓝牙协议分析工具截图示例

- HART的协议分析工具：FrameAlyst
  TODO：详见【记录】下载试用HART协议分析工具FrameAlyst[13]

[9] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/html/soft_tech_common.html#ch03_protocol_overview
[10] http://www.crifan.com/files/doc/docbook/bluetooth_intro/release/html/bluetooth_intro.html#ch02_bt_arch_profile
[11] http://crifan.com/files/doc/docbook/usb_basic/release/html/usb_basic.html#usb_analysis_tool
[12] http://www.deyisupport.com/question_answer/wireless_connectivity/bluetooth/f/103/t/53706.aspx
[13] http://www.crifan.com/try_hart_protocol_debug_tool_framealyst/

- 网络协议分析工具WireShark（旧称Ethereal）
  TODO：详见【记录】折腾WireShark（旧称Ethereal）[14]

此处之所以整理各种不同的协议都有专用分析工具的意思是：

不论做哪方面的技术和协议，一般来说，都是有对应的专用工具的。

在折腾对应的技术的时候，找到对应专用工具，并合理利用，可以大幅度地提供做事情的效果：

对于抓取出来的某协议的数据，无需手动地极其费力去分析，并且人工也很难分析，而改用工具分析数据，立马就so easy的节奏。

1.10. UUID通用唯一识别码

UUID==Universally Unique IDentifier

主要用来标示，某种系统或环，中的唯一性

- PROFINET中的UUID
  PROFINET CBA Protocol Overview[15]

  中就有解释到PROFINET中也用到UUID

- Linux系统中磁盘分区时也涉及到UUID

1.11. Brochure宣传手册

作为某种技术来说，官网去宣传该技术，往往都是制作一个技术类的宣传手册，往往叫做：Brochure

比如：

- HART的Brochure
  HART Communication Protocol and Foundation - Home Page[16]

  中的：

  HART Communication Protocol (brochure, PDF)[17]

  就是个PDF文件：HART_Protocol_Brochure.pdf

  就是对应的用于宣传HART技术的Brochure

1.12. cheat sheet

cheat sheet，本意是，小抄

[14] http://www.crifan.com/try_with_wireshark/
[15] http://www.rtaautomation.com/technologies/profinet-cba/
[16] http://en.hartcomm.org/
[17] http://www.hcf-files.com/webasyst/published/DD/2.0/file_link.php?sl=b1f501197481c80f429a61cb87c45ec1&DB_KEY=V0VCRklMRVM=

不过在技术领域内，其含义是：

用一个小抄的形式，去简明介绍某个技术

往往表现形式都是一个简明扼要的，图片，图表，等方式，去整理出某个技术的核心的东西。

以便可以起到，想要学习使用某个技术，带上这个，小抄，随时可以方便的“抄写”，查看到自己所需要某个点的用法。

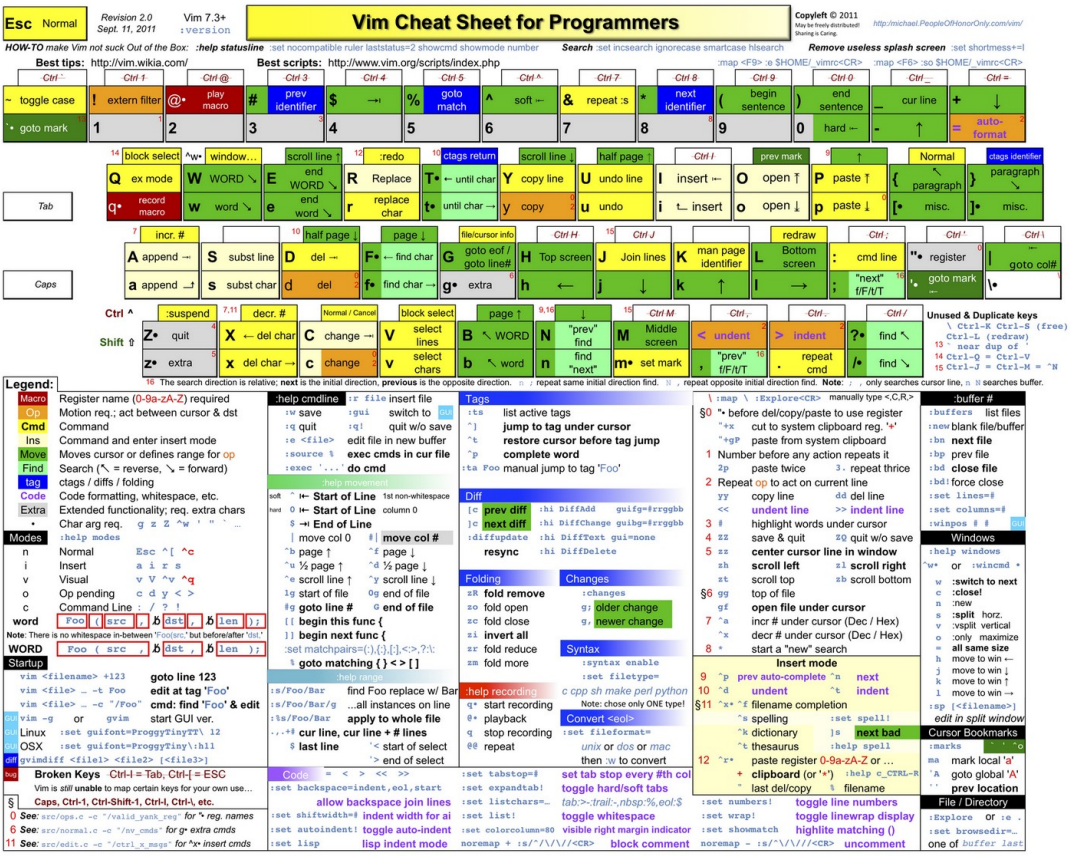
- vim的cheat sheet
  vim中最核心的就是各个快捷键及其含义

  所以就有人去以键盘的方式，去整理出vim中有哪些快捷键，及其对应含义：

  What a wonderful world ! - Coding~[18]

  中的vim的cheat sheet：

图 1.3. Vim的cheat sheet



- git的cheat sheet
  Git Cheat Sheet[19]

  中的git的cheat sheet：

[18] http://www.love67.net/?p=711
[19] http://www.git-tower.com/blog/git-cheat-sheet/

图 1.4. Git的cheat sheet

fournova

GIT CHEAT SHEET

presented by **TOWER** › Version control with Git - made easy

CREATE

Clone an existing repository
```
$ git clone ssh://user@domain.com/repo.git
```
Create a new local repository
```
$ git init
```

LOCAL CHANGES

Changed files in your working directory
```
$ git status
```
Changes to tracked files
```
$ git diff
```
Add all current changes to the next commit
```
$ git add .
```
Add some changes in <file> to the next commit
```
$ git add -p <file>
```
Commit all local changes in tracked files
```
$ git commit -a
```
Commit previously staged changes
```
$ git commit
```
Change the last commit
Don't amend published commits!
```
$ git commit --amend
```

COMMIT HISTORY

Show all commits, starting with newest
```
$ git log
```
Show changes over time for a specific file
```
$ git log -p <file>
```
Who changed what and when in <file>
```
$ git blame <file>
```

BRANCHES & TAGS

List all existing branches
```
$ git branch -av
```
Switch HEAD branch
```
$ git checkout <branch>
```
Create a new branch based on your current HEAD
```
$ git branch <new-branch>
```
Create a new tracking branch based on a remote branch
```
$ git checkout --track <remote/branch>
```
Delete a local branch
```
$ git branch -d <branch>
```
Mark the current commit with a tag
```
$ git tag <tag-name>
```

UPDATE & PUBLISH

List all currently configured remotes
```
$ git remote -v
```
Show information about a remote
```
$ git remote show <remote>
```
Add new remote repository, named <remote>
```
$ git remote add <shortname> <url>
```
Download all changes from <remote>, but don't integrate into HEAD
```
$ git fetch <remote>
```
Download changes and directly merge/integrate into HEAD
```
$ git pull <remote> <branch>
```
Publish local changes on a remote
```
$ git push <remote> <branch>
```
Delete a branch on the remote
```
$ git branch -dr <remote/branch>
```
Publish your tags
```
$ git push --tags
```

MERGE & REBASE

Merge <branch> into your current HEAD
```
$ git merge <branch>
```
Rebase your current HEAD onto <branch>
Don't rebase published commits!
```
$ git rebase <branch>
```
Abort a rebase
```
$ git rebase --abort
```
Continue a rebase after resolving conflicts
```
$ git rebase --continue
```
Use your configured merge tool to solve conflicts
```
$ git mergetool
```
Use your editor to manually solve conflicts and (after resolving) mark file as resolved
```
$ git add <resolved-file>
$ git rm <resolved-file>
```

UNDO

Discard all local changes in your working directory
```
$ git reset --hard HEAD
```
Discard local changes in a specific file
```
$ git checkout HEAD <file>
```
Revert a commit (by producing a new commit with contrary changes)
```
$ git revert <commit>
```
Reset your HEAD pointer to a previous commit
...and discard all changes since then
```
$ git reset --hard <commit>
```
...and preserve all changes as unstaged changes
```
$ git reset <commit>
```
...and preserve uncommitted local changes
```
$ git reset --keep <commit>
```

30-day free trial available at
www.git-tower.com

TOWER
Version control with Git - made easy

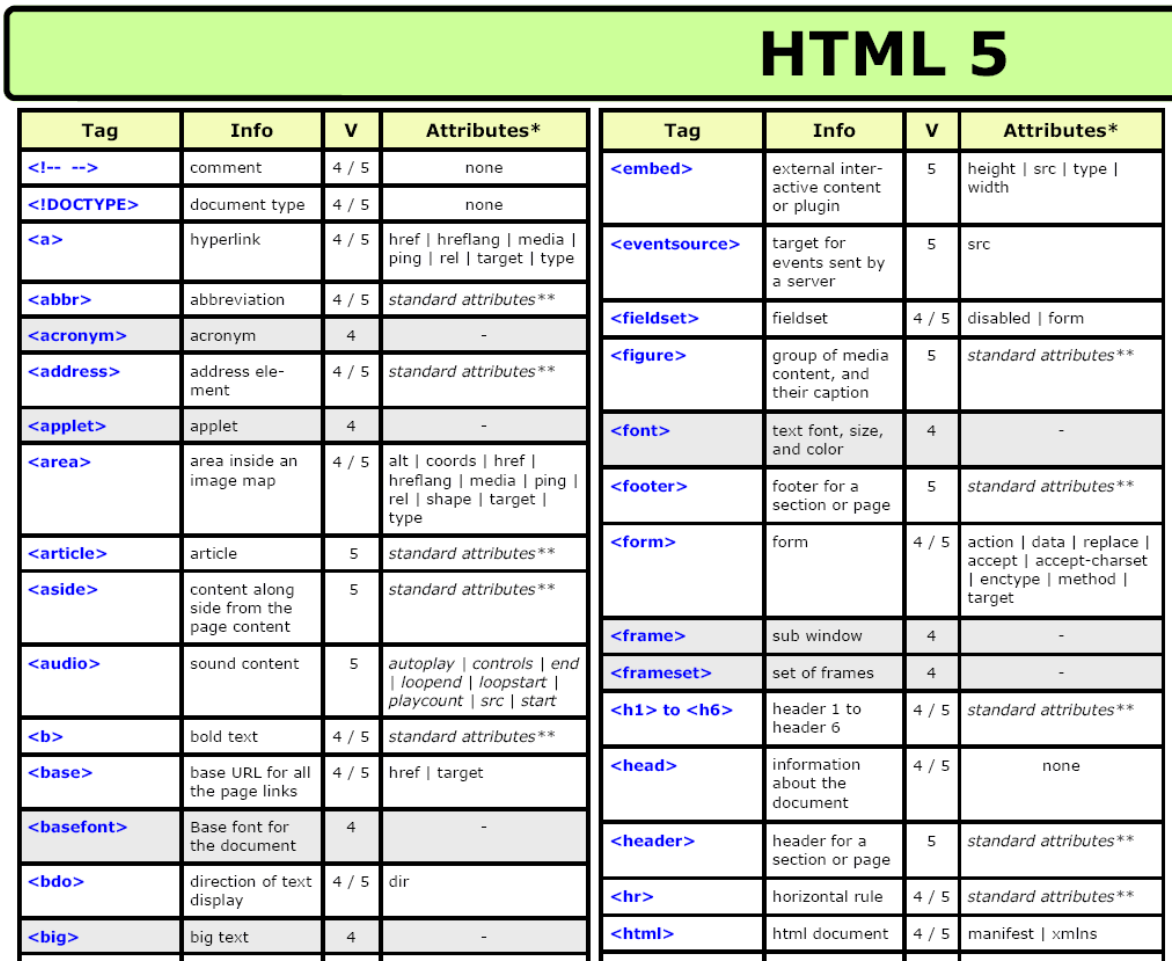
不过，另外也看到了，好像cheat sheet只是表示总结摘要，所以也可以有其他形式，比如网页：

Git Cheat Sheet-Zh.md at master GitHub[20]

- HTML5的cheat sheet
  HTML 5 Cheat Sheet (PDF) – Smashing Magazine[21]

  中的HTML5的cheat sheet：

图 1.5. HTML5的cheat sheet



HTML 5

Tag	Info	V	Attributes*
<!-- -->	comment	4 / 5	none
<!DOCTYPE>	document type	4 / 5	none
<a>	hyperlink	4 / 5	href | hreflang | media | ping | rel | target | type
<abbr>	abbreviation	4 / 5	*standard attributes***
<acronym>	acronym	4	-
<address>	address element	4 / 5	*standard attributes***
<applet>	applet	4	-
<area>	area inside an image map	4 / 5	alt | coords | href | hreflang | media | ping | rel | shape | target | type
<article>	article	5	*standard attributes***
<aside>	content along side from the page content	5	*standard attributes***
<audio>	sound content	5	*autoplay* | *controls* | *end* | *loopend* | *loopstart* | *playcount* | *src* | *start*
<b>	bold text	4 / 5	*standard attributes***
<base>	base URL for all the page links	4 / 5	href | target
<basefont>	Base font for the document	4	-
<bdo>	direction of text display	4 / 5	dir
<big>	big text	4	-

Tag	Info	V	Attributes*
<embed>	external interactive content or plugin	5	height | src | type | width
<eventsource>	target for events sent by a server	5	src
<fieldset>	fieldset	4 / 5	disabled | form
<figure>	group of media content, and their caption	5	*standard attributes***
<font>	text font, size, and color	4	-
<footer>	footer for a section or page	5	*standard attributes***
<form>	form	4 / 5	action | data | replace | accept | accept-charset | enctype | method | target
<frame>	sub window	4	-
<frameset>	set of frames	4	-
<h1> to <h6>	header 1 to header 6	4 / 5	*standard attributes***
<head>	information about the document	4 / 5	none
<header>	header for a section or page	5	*standard attributes***
<hr>	horizontal rule	4 / 5	*standard attributes***
<html>	html document	4 / 5	manifest | xmlns

1.13. retro-spec先有实现后有标准

计算机领域内的技术，尤其是软件，协议等东西，正常的普通的逻辑是：

先有人（或组织）设计了某个技术的标准，然后再有众多人去实现了这个标准，将技术应用到某个现实领域内。

不过现实往往有些特例：

开始某种技术的应用，并没有统一的标准，此时是百花齐放，但是也是诸侯混战，没有统一。

最后，由各方出面，大家达成共识，统一采用某个标准。

[20] https://github.com/flyhigher139/Git-Cheat-Sheet/blob/master/Git%20Cheat%20Sheet-Zh.md
[21] http://www.smashingmagazine.com/2009/07/06/html-5-cheat-sheet-pdf/

比如，HTML就是这样一个先有了实现后才由标准的retro-spec，详见：HTML4，HTML5，XHTML之间有什么区别？[22]

1.14. 国际标准IEC

国际标准，尤其是IEC，是工业领域方面的国际标准。

每个IEC标准，对应的是IEC加上编号，对应的是某个协议，标准等等。

比如：IO-Link im Durchblick[23]中的IO-Link

IO-Link是工业领域用的一种协议，对应的IEC中的标准是：IEC 61131-9

类比来解释就是：

IEC 61131-9 就像一个学生的学号；

学号背后，对应的是某个真实存在的，具有名字的学生：IO-Link，就像该学生的名字。

1.15. 有些库函数是相通的但有不同版本

比如：

- log日志的库
  - 针对C语言的log4c：【记录】Ubuntu下用arm-xscale-linux-gnueabi交叉编译log4c[24]
  - 针对java语言的log4j：【记录】尝试用android-logging-log4j去实现log输出内容到sd卡中的文件的功能[25]
  - 针对go语言的log4g：【记录】go语言中通过log4go实现同时输出log信息到log文件和console[26]

1.16. 命令行界面（CLI==Command Line Interface） vs 图形界面（GUI==Graphical User Interface）

命令行界面，图形界面，是两种不同形式

两者的区别和联系是：

相同点：

对于很多事情，或者说实现很多功能，都可以既有命令行方式，也可以有图形界面方式。

只是，实现目的的方式，不太一样而已。

但往往底层所依赖的东西，内部实现的原理，是一致的。

[22] http://www.zhihu.com/question/19818208
[23] http://www.io-link.com/en/Technology/what_is_IO-Link.php?thisID=73
[24] http://www.crifan.com/under_ubuntu_use_arm_xscale_linux_gnueabi_gcc_cross_compile_log4c/
[25] http://www.crifan.com/android_try_use_android_logging_log4j_to_output_log_to_sd_card_file/
[26] http://www.crifan.com/go_language_output_log_to_both_file_and_console_meantime_via_log4go

- 娱乐方面的例子
  假如是Linux类的Ubuntu系统，在电脑上去放一首歌听，则可以有：
  - 命令行方式：mplayer
    通过mplayer[27]的命令行方式去播放一首歌
  - 图形界面方式：ExMplayer或Rhythmbox
    也可以通过图形界面，比如ExMplayer[28]

    或那个Ubuntu自带的Rhythmbox[29]

    去播放，都是可以的。

    而对应的图形界面的工具ExMplayer，其内部本身就是调用的是mplayer去播放歌曲的。

    与此对应的Rhythmbox，倒不是基于mplayer的播放工具。

    但是本质上，都是调用底层的系统所提供的声卡驱动，去将音频数据传递给声卡驱动，最终通过声卡去将歌曲的声音播放出来的。
- 软件开发方面的例子
  比如之前介绍的Python的开发，有命令行方式，也是各种GUI工具，即IDE的方式

  但是本质上，底层都是调用Python的解析器，去解析和执行Python程序的。

  详见：Python的IDE和Python代码编辑器，Windows的cmd，等的关系[30]

1.17. 软件发布领域内通用概念

软件发布领域内的常见的概念和名词：

1.17.1. 版本号命名规则

在我们进行计算机领域内的开发时，往往会涉及到某个软件、硬件、某个源代码文件等等的版本号的命名。

这些版本号的命名，原先并没有什么规律。但是随着系统的复杂，就需要一个行之有效的，版本号的命名的规则。

然后就诞生了，对应的，多数人都采纳的一种版本号的命名的规则。

TODO：将请问python2.7.x各个版本之间的差异[31]的回复中提到的：语义化版本号[32]整理过来。

1.17.2. nightly夜晚编译

nightly，每天晚上编译一次，编译出来一个可以工作的版本

这个是软件开发过程中的，软件发布的一种方式。

[27] http://wiki.ubuntu.com.cn/MPlayer
[28] http://ubuntuhandbook.org/index.php/2013/11/install-exmplayer-ubuntu-ppa/
[29] http://www.howtogeek.com/howto/19043/how-to-make-ubuntu-play-mp3-files/
[30] http://www.crifan.com/files/doc/docbook/python_beginner_tutorial/release/htmls/ch04_python_dev_env.html#python_ide_relation_with_windows_cmd
[31] http://segmentfault.com/q/1010000000317198
[32] http://semver.org/

这样每天一个新版本，每个版本都包含了最新的功能，但是相对于正式发布的版本，可能存在一定的bug缺陷，但是不妨碍想要尝新的用户去试用。

http://www.worldhello.net/doc/nightlybuild/ar01s01s01.html

也有提及该概念。

1.17.3. RC==release candidate

软件正式发布之前，往往会其他几个版本，比如RC版本等等。

对应的RC的意思是：

> Release Candidate（简称RC）指可能成为最终产品的候选版本，如果未出现问题则可发布成为正式版本。
>
> 在此阶段的产品通常包含所有功能、或接近完整，亦不会出现严重问题。
>
> 多数开源软件会推出两个RC版本，最后的RC2则成为正式版本。
>
> 闭源软件较少公开使用，微软公司在Windows 7上应用此名称。
>
> 苹果公司把在这阶段的产品称为“Golden Master”（简称GM），而最后的GM即成为正式版本。

不同时期的版本的叫法分别是：

- Pre-alpha
- Alpha
- Beta
- RC1
- RC2
- RTM
- Stable

举例：

http://www.python.org/中就有提到RC：

> Python 3.3.4 release candidate has been released
>
> The first rc for Python 3.3.4, Python 3.3.4rc1, has been released.

1.18. 软件开发中的IDE

在软件开发领域内，随着经验积累你会发现，在不同计算机语言进行开发期间，都会遇到各种不同的IDE。以及：

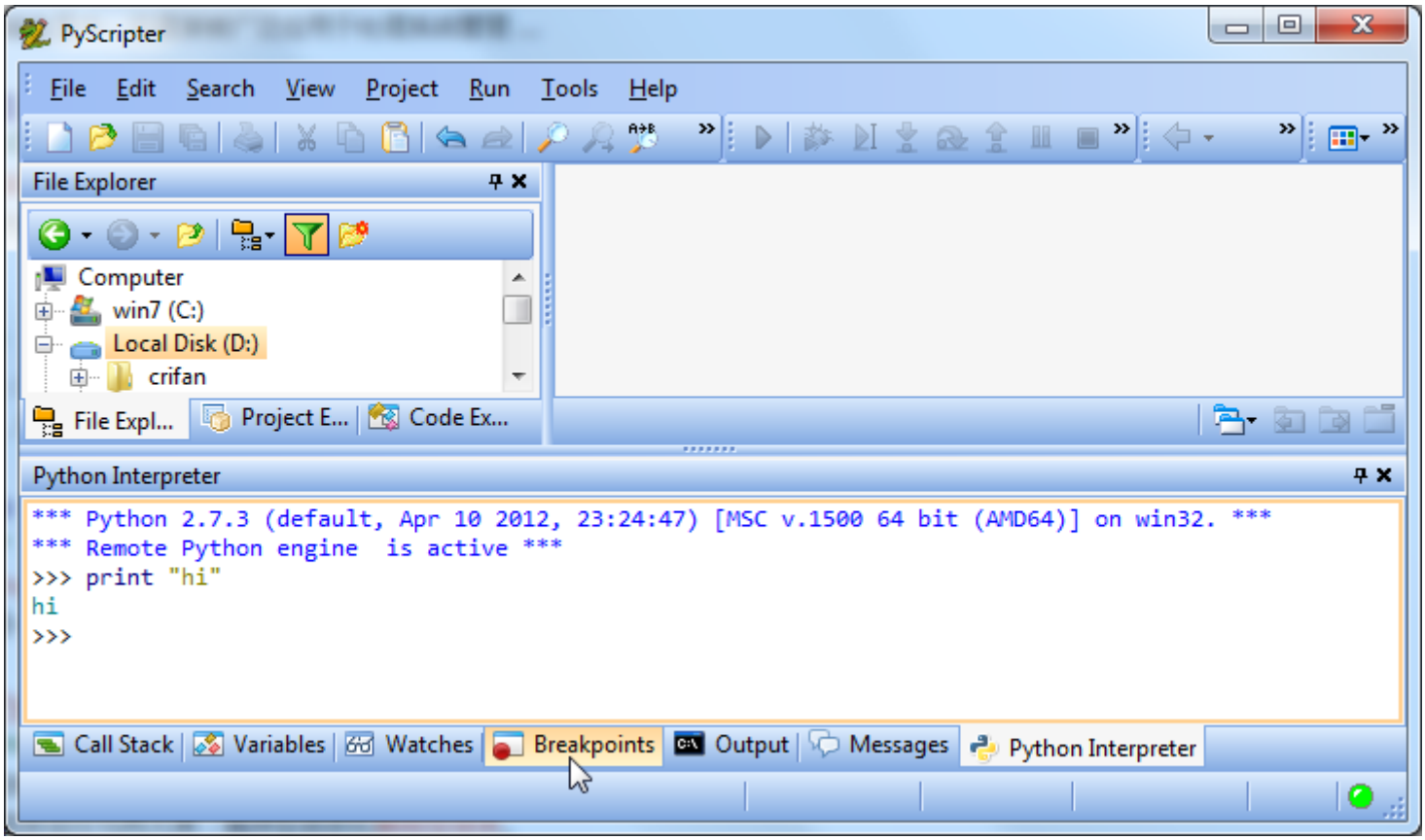
1.18.1. IDE中的通用功能

比如调试功能，调用堆栈，变量值等等。

比如：

- Python的IDE：PyScripter
  Python的IDE：PyScripter中，就也支持调试相关的功能。

图 1.6. Python的IDE：PyScripter中的Breakpoints



包括：

- Call Stack
- Variables
- Watches
- Breakpoints

等等

TODO：将请问python2.7.x各个版本之间的差异[33]的回复中提到的：语义化版本号[34]整理过来。

1.18.2. 常见IDE：VS(Visual Studio)

TOOD：整理VS（Visual Studio）中的各种调试功能。

[33] http://segmentfault.com/q/1010000000317198
[34] http://semver.org/

1.18.3. 常见IDE：Eclipse

TODO：整理Eclipse中的调试功能。

以及基于Eclipse的ADT中的，同样的调试功能

1.18.4. 常见IDE：IntelliJ IDEA

TODO：IntelliJ IDEA中的功能。

第 2 章 通用软件开发之态度和想法

此处整理一些，软件开发期间，会遇到和用到的，通用的思想，想法，做事情的态度，方式。

目的是，更加高效的，去搞开发，搞软件开发。

2.1. 软件开发风格之思路清晰有条理

很多时候，做软件开发，和做其他事情，道理都是相同的：

即使是临时文件，也要清晰的命名，避免后续可能的，潜在的错误发生，并且还可以帮助我们快速分析，调试出现的错误。

或者是快速恢复思路：自己之前做过的开发，即使过了相对较长的时间之后，比如3,5天，3,5个月，甚至3，5年，自己回头再看自己的软件或代码，仍旧可以看懂，并且明白自己当时的意图。

举例：

之前看到这个新手玩python偶遇坑爹错误，求开导[1]

中的我的回复：

不论错误的原因是否是由于测试文件名和模块同名。但是这样的做事情的方式，都是偷懒的做法，不好的做法，都属于很不好的习惯。

而针对该特定问题：其由于第一次，刚开始就没有给临时文件一个清晰的名字，而导致了后续的问题。

而正常的思路，好的做法，应该是：

去测试该requests模块，需要写个测试文件，目的是用来测试此模块，

那么该测试文件的命名，就不应该很随意的、轻易的去起个无意义的名字，比如：

- a.py或abc.py
- 1.py或123.py
- test.py或demo.py

而是应该要保持思路清晰，起个见名知意的名字，比如：

- requestsDemo.py或requests_demo.py
- requestsTest.py或requests_test.py

如此，才可以达到写代码的目的：

- 不论是，别人看你的文件名，看你的代码
- 还是，过了很长时间之后，你回来看你自己的代码

都可以快速的看懂你所写的代码，能读懂你的每个文件所要实现的功能。

否则就很容易变成，很多人所遇到的情况：

- 不论是，自己写完代码，没过多长时间

[1] http://bbs.chinaunix.net/thread-4099920-1-1.html

- 还是，别人过来看你的代码

都是：

- 单独看文件名，不知道该文件的作用
- 再去看代码，也是逻辑不够清晰，逻辑混乱，无法快速准确的明白你的代码的含义，所要实现是什么功能

如此：

就变成了典型的：

- 坏的软件开发的习惯
- 坏的写代码的风格

这就和之前的软件开发的重中之重的要注意的那一点写代码是给别人看的所冲突了。

看到这里，估计有人会问：

不就是简单的给测试文件命个名，写点测试代码吗？

又不是真正的去搞软件开发，写详细实现功能的代码，用得着这么严谨，甚至说死板吗？

对此，我的回答是：需要这么严谨。

因为：

- 第一次就做对的事情
  把这个问题，上升到做事情的态度来看，那么你需要明白一个做事情原则中，很重要的一条：第一次，就做对的事情

  因为，如果第一次就马虎，尤其而带来的以后的潜在的错误，以及解决该问题的错误所需要花费的精力和成本，

  远大于你为了第一次做对的事情，仅仅比你随意的就下手做了，而多思考了几秒，几十秒，甚至几十分钟

  这个逻辑，也非常类似于：

  解决软件的bug的时间，永远是越早越好，最好是想办法找到合理的机制去预防软件的bug。

  因为随着时间往后推移，软件的bug所导致的问题会越严重。

- 只有好的习惯，才会产生更好的结果
  只有严谨的，认真的态度，养成好的习惯，才能从好的习惯中获益：减少后续发生问题的概率

  最终实现我们的终极目的：更加高效的做事情，更加高效的搞软件开发。

  而高质量的软件，和低质量的差距，也就是从第一天的做事情、搞软件开发的态度上，就决定了。

2.2. 软件开发的效率、成本和投资

在http://zh.wikipedia.org/wiki/%E8%84%9A%E6%9C%AC%E8%AF%AD%E8%A8%80中看到的：

在很多案例中，如编写一些数十行的小脚本，它所带来的编写优势就远远超过了运行时的劣势，尤其是在当前程序员工资趋高和硬件成本趋低时。

由此想到：

- 计算机发展初期：硬件成本高，软件成本低：程序员工资相对硬件成本要低
- 计算法发展到现在：硬件成本相对很低，软件成本相对较高：程序员工资相对来说要比硬件要高

所以为了提升软件开发效率，不论是公司还是个人，都要舍得适当的投资：

- 买配置好的电脑，方便软件开发
  不论是公司还是个人，都是要舍得在升级电脑硬件配置方面舍得投入

  当然前提是，升级了配置，对于软件开发的速度，效率方面有所提升

  此种情况，往往是在发生了，电脑硬件配置不够用，比如Windows下跑Linux的虚拟机，编译项目代码占用资源较高

  此时才考虑，是否值得花钱，升级内存，还是把机械硬盘换成SSD硬盘，甚至是购买新电脑

  由此，保证硬件资源够用，方便软件开发，不会把多余的时间和精力消耗在低硬件配置无法解决软件方面的高性能的要求上。

  关于SSD固态硬盘，对于电脑性能的提升，的确很大，不了解的，可以参考：

  TODO：添加换了SSD硬盘后电脑性能提升对比的帖子

- 为了更方便的利用google等国外网站资源而购买翻墙方面的服务
  搞软件开发，往往是国外的资源更有参考价值

  而如何找到这类国外资源，往往需要利用google

  而google被封，被墙，导致基本无法使用

  想要方便的，稳定的用google，及查看其它国外技术资料，往往就需要翻墙

  而目前免费的可以翻墙的资源，往往效果不够好

  所以，在能接受的前提下，可以购买翻墙方面的服务，比如VPN，Shadowsocks等等服务。

  比如我放弃了GoAgent，而花钱99元买了一年的shadowsocks，使得可以稳定高效的翻墙用google。

  TOOD：添加购买shadowsocks的帖子。

- 为了方便开发而买必要的有价值的软件
  比如搞特定领域的软件开发，而需要一些工具类的软件。

  为了支持国内外的精品软件，在可以支付的起的情况下，可以考虑购买正版软件，而不要一直都用盗版。

  比如我觉得一些做的好的软件有，Source Insight，Beyond Compare等等。

  其他值得推荐的，详见：crifan推荐软件[2]

[2] http://www.crifan.com/files/doc/docbook/crifan_rec_soft/release/html/crifan_rec_soft.html

第 3 章 通用软件开发之方法

此处整理一些，软件开发期间，所涉及到的，通用的学习方法。

3.1. 软件开发方法之充分利用互联网资源

尤其是，在查技术资料时，利用好谷歌（google），而不是百度。

当然，还有stackoverflow，维基百科（wikipedia）等有价值的网站。

TODO：把之前写的，以某些技术点为例，说明如何利用google去搜资料的过程整理过来。

3.2. 软件开发方法之如何学习某个技术点

对于在软件开发期间，学习某个，你之前不懂，听都没听过的，技术点的时候，这时候，往往最有用的网络资源是：维基百科

TODO：整理之前在学习某个技术点，如何利用wiki去搞懂概念，并搞懂该技术点，在技术体系中的位置，以便于更清楚技术点的所属层次，的过程。

参考书目

[1] USB基础知识概论 - USB系统的核心是Host[1]

[2] 【整理】现场总线技术：PROFINET | 在路上[2]

[3] 蓝牙协议详解[3]

[4] PROFINET - Wikipedia, the free encyclopedia[4]

[5] HTML4，HTML5，XHTML 之间有什么区别？[5]

[6] 请问python2.7.x各个版本之间的差异[6]

[7] 新手玩python偶遇坑爹错误，求开导[7]

[8] 软件版本周期 - 维基百科，自由的百科全书[8]

[9] mplayer[9]

[10] ExMplayer[10]

[11] Rhythmbox[11]

[12] 蓝牙协议的架构[12]

[13] USB协议概览[13]

[14] Python的IDE和Python代码编辑器，Windows的cmd，等的关系[14]

[1] http://www.crifan.com/files/doc/docbook/usb_basic/release/html/usb_basic.html#core_is_host
[2] http://www.crifan.com/summary_filedbus_profinet_overview/
[3] http://www.crifan.com/files/doc/docbook/bluetooth_intro/release/html/bluetooth_intro.html
[4] http://en.wikipedia.org/wiki/PROFINET
[5] http://www.zhihu.com/question/19818208
[6] http://segmentfault.com/q/1010000000317198
[7] http://bbs.chinaunix.net/thread-4099920-1-1.html
[8] http://zh.wikipedia.org/zh-cn/%E8%BB%9F%E4%BB%B6%E7%89%88%E6%9C%AC%E9%80%B1%E6%9C%9F
[9] http://wiki.ubuntu.com.cn/MPlayer
[10] http://ubuntuhandbook.org/index.php/2013/11/install-exmplayer-ubuntu-ppa/
[11] http://www.howtogeek.com/howto/19043/how-to-make-ubuntu-play-mp3-files/
[12] http://www.crifan.com/files/doc/docbook/bluetooth_intro/release/html/bluetooth_intro.html#ch02_bt_arch_profile
[13] http://www.crifan.com/files/doc/docbook/soft_tech_common/release/html/soft_tech_common.html#ch03_protocol_overview
[14] http://www.crifan.com/files/doc/docbook/python_beginner_tutorial/release/htmls/ch04_python_dev_env.html#python_ide_relation_with_windows_cmd