GIT CHEAT SHEET

CREATE

Clone 远程仓库

```
$ git clone ssh://user@domain.com/xx.git
```

初始化本地 git 仓库（新建仓库）

```
$ git init
```

LOCAL CHANGES

查看当前版本状态（是否修改）

```
$ git status
```

显示所有未添加至 index 的变更

```
$ git diff
```

比较与上一个版本的差异

```
$ git diff HEAD^ / HEAD -- ./lib
```

增加更改过的文件至 index

```
$ git add . / add * ...
```

提交

```
$ git commit -m  'xxx'
```

合并上一次提交（用于反复修改）

```
$ git commit -amend -m  'xxx'
```

将 add 和 commit 合为一步

```
$ git commit -am  'xxx'
```

COMMIT HISTORY

显示日志

```
$ git log
```

显示某个提交的详细内容

```
$ git show <commit>
```

在每一行显示 commit 号,提交者,最早提交日期

```
$ git blame <file>
```

BRANCHES & TAGS

显示本地分支

```
$ git branch
```

切换分支

```
$ git checkout <branch>
```

新建分支

```
$ git branch <new-branch>
```

创建新分支跟踪远程分支

```
$ git branch --track <new> <remote>
```

删除本地分支

```
$ git branch -d <branch>
```

给当前分支打标签

```
$ git tag <tag-name>
```

UPDATE & PUBLISH

列出远程分支详细信息

```
$ git remote -v
```

显示某个分支信息

```
$ git remote show <remote>
```

添加一个新的远程仓库

```
$ git remote add <remote> <url>
```

获取远程分支 ,但不更新本地分支 ,另需 merge

```
$ git fetch <remote>
```

获取远程分支，并更新本地分支

```
$ git pull <remote> <branch>
```

推送本地更新到远程分支

```
$ git push <remote> <branch>
```

删除一个远程分支

```
$ git push <remote> --delete <branch>
```

推送本地标签

```
$ git push --tags
```

MERGE & REBASE

合并分支到当前分支，存在两个

```
$ git merge <branch>
```

合并分支到当前分支，存在一个

```
$ git rebase <branch>
```

回到执行 rebase 之前

```
$ git rebase --abort
```

解决矛盾后继续执行 rebase

```
$ git rebase --continue
```

使用 mergetool 解决冲突

```
$ git mergetool
```

使用冲突文件解决冲突

```
$ git add <resolve-file>
$ git rm <resolved-file>
```

UNDO

将当前版本重置为 HEAD（用于 merge 失败）

```
$ git reset --hard HEAD
```

将当前版本重置至某一个提交状态（慎用！）

```
$ git reset --hard <commit>
```

将当前版本重置至某一个提交状态，代码不变

```
$ git reset <commit>
```

重置至某一状态，保留版本库中不同的文件

```
$ git reset --merge <commit>
```

重置至某一状态，重置变化的文件，代码改变

```
$ git reset --keep <commit>
```

丢弃本地更改信息并将其存入特定文件

```
$ git checkout HEAD <file>
```

撤销提交

```
$ git revert <commit>
```

by Kevins Bobo http://kevins.pro

VIM CHEAT SHEET

Cursor Movement

左，下，上，右

h j k l

上/下翻页，半翻页，翻一行

^-f/b , ^-u/d , ^-y/e

将第 n 行滚至屏幕顶部，当前行

nz , z

跳至下个单词尾部/头部 以空格区分

e / w , E / W

跳至上个单词头部/以空格区分

b / B

跳至行首/首字母，行尾

0 / ^ , $

跳至首行/某行/尾行

gg / [N]G / G

跳至最后一次编辑的地点

'.

句首 / 句尾 , 段首 / 段尾

(/) , { / }

屏幕 首行 / 中间行 / 底行

H / W / l

跳至书签 a

'a

向右/向左跳至本行字符 x 处 ,右前/左后

fx / Fx , tx / Tx 使用 ; 重复

括号匹配

%

Macros

开始/停止录制名为 a 的宏

qa / q

执行名为 a 的宏 / 重复执行

@a / @@

Exiting

保存 / 保存退出 / 退出 / 不保存退出

:w / :wq , :x , ZZ / :q / :q!

:syntax on 语法高亮

Insert Mode

字 前 / 后 插入，行 前 / 后 插入

i / a , I / A

行 下 / 上 插入

o / O

匹配下 / 上一个关键字（自动补全）

^-n / ^-p

Editing

替换单个字符 / 替换模式

r / R

合并下面的行到当前行

J

删除并重新编辑 行/ 单词/ 当前至行尾

cc / cw / c$

删除 字符 / 行 并进入编辑模式

s / S

交换字符

xp

撤销 / 重做 / 修正之前对改行的操作

u / ^-r / U

重复最后一个命令

.

大小写转换 字符 / 单词 ,单词 大写/ 小写

~ / g~iw , gUiw / guiw

左 / 右 / 自动 缩进

<< / >> / ==

Search & Replace

从首行 / 向前 / 向后查找当前单词

gd / # / *

向下 / 向上 查找， 下一个 / 上一个

/pattern / ?pattern , n / N

行内替换不询问 ,询问(特殊字符需转译)

:s/old/new , :s/old/new/gc

替换 1 至 2 行，替换文件中所有

:1,2s/old/new , g/old/s//new

:set nu/number 显示行号

Multi-File

打开 / 关闭文件，下 / 上 一个文件

:e / :bd , :bn / :bp

打开文件并分屏/垂直分屏

:sp fn / :vsp fn

切换分屏

h j k l

分屏/垂直分屏，切换，关闭

<^-w> s / v , w , q

最大化尺寸，等分尺寸，增加/减小尺寸

<^-w> _ / | , = , + , -

标签 创建 ,前/后/最前/最后切换 ,移动

:tabe fn,gt/gT/:tabr/l,:tabm [N]

Visual Mode

开启选择模式 字符 / 行 / 块

v / V / ^-v

跳至区域边缘/角 , 区域大/小写切换

o / O , U / u

选择一个单词,()/{}内,不包括左括号

aw , ab / aB , ib / iB

多行同时插入 --

^-v ^-d I -- [ESC]

Cut & Paste

剪切 行 / 到行尾 / 单词 / 字符 / 前字符

dd / D / dw / x / X

复制 行 / 多行 / 单词 / 到行尾

yy / 2yy / yw / y$

粘贴 后 / 前 / 自动缩进

p / P /]p

寄存器 查看(0 是复制 ,1-9 是删除的行)

:reg

新建 / 追加 / 粘贴 [a] 寄存器

"a / "A / "ap

:set tabstop/shiftwidth=4 缩进

:set cindent C/C++语法缩进

:set nocompatible 去掉一致性

注：^- 代表 Ctrl-

by Kevins Bobo http://kevins.pro

SHELL & GDB CHEAT SHEET

SHELL : Cursor Movement

前 / 后 跳一个字符 backward/forward

^-b / ^-f

前后跳一个单词

alt-b / alt-f

跳至 行首 / 行尾 a/end

^-a / ^-e

删除当前光标到行首的字符

^-x ←

SHELL : Editing

粘贴最后一次命令最后的参数

alt-.

删除光标到右边单词开始(符号空格区分)

alt-d

删除光标到左边单词结束(仅空格区分)

^-w

删除光标 前 / 后 一个字符

^-h / ^-d

删除光标 左 / 右 所有

^-u / ^-k

清屏

^-l

复制 / 粘贴

^-shift-c/v

SHELL : Other

上 / 下 一条命令

^-p / ^-n

在历史中向上 / 下寻找最近一条命令

alt-p / alt-n

向 上 / 下 翻页

shift-PageUp/PageDown

从历史中查找命令 ,重复按代表下一匹配

^-r

注 : ^- 代表 Ctrl-

GCC

一步编译

```
$ gcc xxx.c -o xxx
```

预处理

```
$ gcc -E xxx.c (-o xxx.i)-可选
```

预处理后编译为汇编代码

```
$ gcc -S xxx.i -o xxx.s
```

将汇编代码汇编

```
$ gcc -c xxx.s -o xxx.o
```

汇编后连接

```
$ gcc xxx.o -o xxx
```

多文件编译

```
$ gcc xxx1.c xxx2.c test
```

检错

```
$ gcc -padantic xxx.c -o xxx
$ gcc -Wall/Werror xxx.c -o xxx
```

调用 GDB 编译

```
$ gcc -g xxx.c -o xxx
```

GDB

启动 GDB

```
$ gdb <file>
```

从第一行开始列出源码

```
(gdb) l
```

在 第 n 行 / func 函数入口 设置，取消断点

```
(gdb) break n / func , clear n
```

删除 / 暂停 /开启 断点 n

```
(gdb) delete / disable / enable n
```

运行/ 单步/ 单步进函数/ 继续/ 退出函数

```
(gdb) r / n / step / c / finish
```

单步跟踪机器指令

```
(gdb) stepi / nexti
```

每次单步之后输出设置的表达式及值

```
(gdb) display a
```

GDB

打印变量 a 值 / 调用后打印

```
(gdb) p a / xxx(a) / xxx(22)
```

指定运行时的参数 / 查看设置好的参数

```
(gdb) set args / show args
```

查看程序运行路径

```
(gdb) show paths
```

设置环境变量

```
(gdb) set environment varname [=v]
```

查看环境变量

```
(gdb) show environment [varname]
```

打开 / 显示 目录

```
(gdb) cd / pwd
```

查看程序是否运行，进程号，暂停原因

```
(gdb) info program
```

查看函数堆栈

```
(gdb) bt
```

退出循环（退出前循环会自动运行完）

```
(gdb) until
```

运行 shell 命令行

```
(gdb) shell
```

设置监视点，一但监视内容值变化，调试终止

```
(gdb) watch a
```

强行终止

```
(gdb) kill
```

调用函数，并传参

```
(gdb) call xxx(a)
```

源码/反汇编/CPU 寄存器/源码和反汇编 分窗口

```
(gdb) layout src/ asm/ regs/ split
```

分割窗口刷新

^-L

查看内存的值(参数/oxdtficsbhwg)

```
(gdb) x /x [ADDRESS]
```

修改内存的值

```
(gdb) set *ADDRESS=VALUE
```

by Kevins Bobo http://kevins.pro