

# 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`

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, 一旦监视内容值变化, 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`