目录

第 1 章　linux 配置

1.1　CMake 安装和常用库的 CMakeLists.txt 的写法

CMake 是一个跨平台的生成工具，主要使用平台无关的 CMakeLists.txt 生成平台相关的 Makefile 文件。

CMake 的安装：

```
tar -zvxf cmake.tar.gz
cd cmake
./configure --prefix=to/install/path --qt-gui
make -j4
make install
```

1.1.1　Boost

Boost 库的 CMakeList.txt 的写法：

```
cmake_minimum_required(VERSION 3.0)

set(CMAKE_PREFIX_PATH "/opt/boost/install")
set(Boost_USE_STATIC_LIBS OFF)
set(Boost_USE_MULTITHREADED ON)
set(Boost_STATIC_RUNTIME OFF)

find_package(Boost)
include_directories(\${Boost_INCLUDE_DIRS})

```

```
message(STATUS \${Boost_INCLUDE_DIRS})
message(STATUS \${Boost_LIBRARIES})

add_subdirectory(src)
```

其中 Boost_INCLUDE_DIRS 必须显示的添加到头文件的搜索路径中。

1.1.2 OpenCV

OpenCV 库的 CMakeLists.txt 的写法：

```
file(GLOB SRC_FILES ${CMAKE_CURRENT_SOURCE_DIR}/*.cpp)

# Declare the executable target built from your sources
add_executable(${PROJECT_NAME} ${SRC_FILES})

# Link your application with OpenCV libraries
target_link_libraries(${PROJECT_NAME} ${OpenCV_LIBS})

# 安装
install(TARGETS ${PROJECT_NAME}
        RUNTIME DESTINATION ${PROJECT_SOURCE_DIR}/bin)
```

注意 OpenCV 的库的搜索结果是放在 OpenCV_LIBS 这个变量中的，且头文件无需添加到搜索路径中。最好的行是就是看下 OpenCV 库自带例子的写法。

1.1.3 ROOT

ROOT 库的 CMakeLists.txt 的写法：

```
# 生成files
file(GLOB SRC "${PROJECT_SOURCE_DIR}/src/*.cpp")

```

```
# 添加目标文件
add_executable(${PROJECT_NAME} ${SRC})

# 添加依赖库
target_link_libraries(${PROJECT_NAME} ${ROOT_LIBRARIES})

# 安装
install(TARGETS ${PROJECT_NAME}
        RUNTIME DESTINATION ${PROJECT_SOURCE_DIR}/bin)
```

注意 ROOT 的库的搜索结果是放在 ROOT_LIBRARIES 这个变量中的，且头文件无需添加到搜索路径中。

1.1.4 Qt5

Qt5 库的 CMakeLists.txt 的写法：

```
# 最低需要2.8.11版本
cmake_minimum_required(VERSION 3.0)

# 设定Qt的安装位置
set(CMAKE_PREFIX_PATH "/opt/qt/install/")

# include的目录包含当前目录
set(CMAKE_INCLUDE_CURRENT_DIR ON)

# 工程名字
project(QtUseCMake)

# 查找需要的Qt库文件
find_package(Qt5Widgets REQUIRED)
find_package(Qt5Core REQUIRED)
find_package(Qt5Gui REQUIRED)
```

```
17
18  # 设定头文件和UI的搜索路径
19  set(HEADER_DIR "${CMAKE_CURRENT_SOURCE_DIR}/include")
20  set(GUI_DIR "${CMAKE_CURRENT_SOURCE_DIR}/ui")
21  set(RESOURCES_DIR "${CMAKE_CURRENT_SOURCE_DIR}/resources")
22
23  # 添加头文件的搜索路径
24  include_directories(${HEADER_DIR})
25
26  # 添加子目录
27  add_subdirectory(src)
28
29  # 安装
30  install(FILES COPYRIGHT README.md
31        DESTINATION doc)
```

20 其中 src 目录下的 CMakeLists.txt 的写法如下：

```
1   # 生成files
2   file(GLOB SRC "${PROJECT_SOURCE_DIR}/src/*.cpp")
3   file(GLOB MOC_HEADER "${PROJECT_SOURCE_DIR}/include/*.h")
4   file(GLOB UIC_UI "${PROJECT_SOURCE_DIR}/ui/*.ui")
5   file(GLOB QRC "${PROJECT_SOURCE_DIR}/resources/action.qrc")
6
7   # 通过UI文件生成对应的头文件
8   qt5_wrap_cpp(MOC_SRC ${MOC_HEADER})
9   qt5_wrap_ui(UIC_SRC ${UIC_UI})
10  qt5_add_resources(QRC_SRC ${QRC})
11
12  # 添加目标文件
13  add_executable(${PROJECT_NAME} ${SRC} ${MOC_SRC} ${UIC_SRC}
    ↪  ${QRC_SRC})
14
```

```
# 添加依赖库
target_link_libraries(${PROJECT_NAME} Qt5::Widgets Qt5::Core
    ↪ Qt5::Gui)

# 安装
install(TARGETS ${PROJECT_NAME}
        RUNTIME DESTINATION bin)
```

库的依赖方式为 target_link_libraries(target Qt5::Core Qt5::Widgets Qt5::Gui Qt5:: 其它模块)。

1.2 OpenCV 的安装

这里我们主要给出 Ubuntu 下 OpenCV 的安装例子。首先我们要先安装 OpenCV 所需要的依赖，打开 terminal 输入以下的命令：

```
sudo apt-get -y install libopencv-dev build-essential cmake
    ↪ libdc1394-22 libdc1394-22-dev libjpeg-dev libpng12-dev
    ↪ libtiff-dev libjasper-dev libavcodec-dev libavformat-dev
    ↪ libswscale-dev libxine-dev libgstreamer0.10-dev
    ↪ libgstreamer-plugins-base0.10-dev libv41-dev libtbb-dev
    ↪ libqt4-dev libmp3lame-dev libopencore-amrnb-dev
    ↪ libopencore-amrwb-dev libtheora-dev libvorbis-dev
    ↪ libxvidcore-dev x264 v41-utils
```

接着我们从官网或者 github 下载自己需要版本的源代码，并且下载 opencv_contrib 的测试模块。分别解压缩两个源代码。

```
cd opencv-版本
mkdir build
cd build
```

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D
    ↪ CMAKE_INSTALL_PREFIX=/full/path/to/opencv/install/dir -D
    ↪ INSTALL_C_EXAMPLES=ON -D BUILD_EXAMPLES=ON -D OPENCV_EXTRA_M
    ↪ ODULES_PATH=/full/path/to/opencv-contrib/modules
    ↪ ../
make -j4
sudo make install
```

剩下只需要修改下 PATH 和 LD_LIBRARY_PATH 环境变量的值。

1.3 GNOME 主题桌面配置

一个好看的 GNOME 桌面可以使做事情时感到赏心悦目，因此倒腾一个好看的桌面主题是势在必然的。各种下载主题配置的工具的网址肯定是 gnome 的官网了。这里只记录下我用了什么主题：

Appearance 的控制：

- GTK+：Sierra-compact-light-thin
- Icons：Flat-Remix-Light
- Cursor：Adwaita(default)
- Shell theme：Sierra-compact-light-thin
- Enable animations：打开

Desktop 的控制：

- Mode：两个模式全部的打开
- 墙纸：这个这个看着找吧

Extensions：

- 打开 User themes 即可，剩下的自己看着办吧。

Fonts：

- Window Titles : Cantarell Bold 13

- Interface : Cantarell Regular 13
- Monospace : Monospace Regular 13
- Hinting : Slight
- Antialiasing : Grayscale
- Scaling Factor : 1.00

Keyboard and Mouse

- Show All Input Sources : close
- Key theme : Emacs
- Switch between overview and desktop : Left super
- Acceleration profile : Flat
- Show location of pointer : open
- Middle-click Paste : open
- Click method : default

Power

- Action : Suspend
- Don't suspend on lid close : close

Top Bar

- Show Application Menu : on
- Show date : checked
- Show seconds : checked
- Show week numbers : checked

第 2 章　我的 emacs 配置

2.1　基本的设置

```
  ;;------语言环境字符集设置结束------------
(set-language-environment 'utf-8)

;;缩进
(setq tex-basic-offset 4)
(setq latex-basic-offset 4)
(setq php-basic-offset 4)

;;去行尾空格
;; (add-hook 'before-save-hook 'delete-trailing-whitespace)

;;取消滚动栏
(set-scroll-bar-mode nil)
;;取消工具栏
(tool-bar-mode nil)
;;启动设置
(setq default-frame-alist
      '((vertical-scroll-bars)
^^I(tool-bar-lines . 0)
^^I(menu-bar-lines . 0)
^^I(right-fringe)
^^I(left-fringe)))

;;显示时间设置
(display-time-mode 1);;启用时间显示设置，在minibuffer上面的那个杠上
```

```
26
27 ;;设置打开文件的缺省路径
28 (setq default-directory "/home/xiaohai")
29
30 ;;关闭烦人的出错时的提示声
31 (setq visible-bell t)
32
33 ;;关闭emacs启动时的画面
34 (setq inhibit-startup-message t)
35
36 ;;关闭gnus启动时的画面
37 (setq gnus-inhibit-startup-message t)
38
39 ;; 改变 Emacs 固执的要你回答 yes 的行为。按 y 或空格键表示 yes，n
   ↪  表示 no。
40 (fset 'yes-or-no-p 'y-or-n-p)
41
42 ;; 显示行列号(缓冲区的)
43 ;; (global-linum-mode 1) ; always show line numbers
44 (setq column-number-mode t)
45 (setq line-number-mode t)
46
47
48 ;;设置粘贴缓冲条目数量.用一个很大的kill ring(最多的记录个数).
   ↪  这样防止我不小心删掉重要的东西
49 (setq kill-ring-max 200)
50
51 ;; Autofill in all modes;;
52 (setq-default auto-fill-function 'do-auto-fill)
53
54 ;;把 fill-column 设为 60. 这样的文字更好读
55 (setq default-fill-column 300)
56
```

```
57 ;;tab键为8个字符宽度
58 (setq default-tab-width 8)
59
60 ;;不用 TAB 字符来indent, 这会引起很多奇怪的错误。编辑 Makefile
   ↪ 的时候也不用担心，因为 makefile-mode 会把 TAB 键设置成真正的 TAB
   ↪ 字符，并且加亮显示的。
61 (setq tab-stop-list ())
62
63 ;;设置 sentence-end 可以识别中文标点。不用在 fill
   ↪ 时在句号后插入两个空格。
64 (setq sentence-end "\\([。！ ？]\\|……\\|[.?!][]\"')}]*\\($\\|[
   ↪ \t]\\)\\)[ \t\n]*")
65 (setq sentence-end-double-space nil)
66
67 ;;可以递归的使用 minibuffer
68 (setq enable-recursive-minibuffers t)
69
70 ;;防止页面滚动时跳动， scroll-margin 3
   ↪ 可以在靠近屏幕边沿3行时就开始滚动，可以很好的看到上下文。
71 (setq scroll-margin 3 scroll-conservatively 10000)
72
73 ;;设置缺省主模式是text,
   ↪ ,并进入auto-fill次模式.而不是基本模式fundamental-mode
74 (setq default-major-mode 'text-mode)
75 (add-hook 'text-mode-hook 'turn-on-auto-fill)
76
77 ;;打开括号匹配显示模式
78 (show-paren-mode t)
79 ;;括号匹配时可以高亮显示另外一边的括号,
   ↪ 但光标不会烦人的跳到另一个括号处。
80 (setq show-paren-style 'parenthesis)
81
82 ;;光标靠近鼠标指针时，让鼠标指针自动让开，别挡住视线。
```

```
83 (mouse-avoidance-mode 'animate)
84
85 ;;在标题栏显示buffer的名字，而不是 emacs@wangyin.com
   ↪ 这样没用的提示。
86 (setq frame-title-format "emacs@%b")
87
88 ;;在行首 C-k 时，同时删除该行。
89 (setq-default kill-whole-line t)
90
91 ;;设定不产生备份文件
92 (setq make-backup-files nil)
93
94 ;;自动保存模式
95 (setq auto-save-mode nil)
96
97 ;;允许屏幕左移
98 (put 'scroll-left 'disabled nil)
99
100 ;;允许屏幕右移
101 (put 'scroll-right 'disabled nil)
102
103
104 ;;允许emacs和外部其他程序的粘贴
105 ;; (setq x-select-enable-clipboard t)
106
107 ;;设置有用的个人信息,这在很多地方有用。
108 (setq user-full-name "jinxiaohai")
109 (setq user-mail-address "xiaohaijin@outlook.com")
110
111 ;;Non-nil if Transient-Mark mode is enabled.
112 (setq-default transient-mark-mode t)
113
114 ;; 当光标在行尾上下移动的时候，始终保持在行尾。
```

```
(setq track-eol t)

;; 当浏览 man page 时，直接跳转到 man buffer。
(setq Man-notify-method 'pushy)

;;设置home键指向buffer开头，end键指向buffer结尾
(global-set-key [home] 'beginning-of-buffer)
(global-set-key [end] 'end-of-buffer)

;; C-f5，设置编译命令；f5，保存所有文件然后编译当前窗口文件
(defun du-onekey-compile ()
  "Save buffers and start compile"
  (interactive)
  (save-some-buffers t)
  (switch-to-buffer-other-window "*compilation*")
  (compile compile-command))
(global-set-key [C-f5] 'compile)
(global-set-key [f5] 'du-onekey-compile)

;; 设置时间戳，标识出最后一次保存文件的时间。
(setq time-stamp-active t)
(setq time-stamp-warn-inactive t)
(setq time-stamp-format "%:y-%02m-%02d %3a %02H:%02M:%02S
  ↪  jinxiaohai")

;;设置M-g为goto-line
(global-set-key (kbd "M-g") 'goto-line)

;; ;;取消control+space键设为mark
;; (global-set-key (kbd "C-SPC") 'nil)

;;代码折叠
(load-library "hideshow")
```

```
147 (add-hook 'c-mode-hook 'hs-minor-mode)
148 (add-hook 'c++-mode-hook 'hs-minor-mode)
149 (add-hook 'java-mode-hook 'hs-minor-mode)
150 (add-hook 'perl-mode-hook 'hs-minor-mode)
151 (add-hook 'php-mode-hook 'hs-minor-mode)
152 (add-hook 'emacs-lisp-mode-hook 'hs-minor-mode)
153 (add-hook 'f90-mode-hook 'hs-minor-mode)
154
155 ;;能把一个代码块缩起来，需要的时候再展开
156 ;; C-c @ C-M-s 现实所有的代码
157 ;; C-c @ C-M-h 折叠所有的代码
158 ;; C-c @ C-s   显示当前代码区
159 ;; C-c @ C-h   折叠当前代码区
160 ;; C-c @ C-c   折叠/显示当前代码区
161
162
163 ;;---------------模式的强制转换--------------
164 (add-to-list 'auto-mode-alist '("\\.h\\'" . c++-mode))
165 ;; (add-to-list 'auto-mode-alist '("\\.f\\'" . f90-mode))
```