# 基于VSCode和CMake实现C/C++开发 <Linux篇>

#### 本次课程适合的对象:

- Linux零基础,想了解最常用的高频Linux命令的小伙伴
- 只会在Windows开发C/C++,需要转Linux开发的小伙伴
- 想深入了解并掌握GCC编译器编译语法和规则的小伙伴
- 想深入了解并掌握GDB调试器命令行调试的小伙伴
- 想学习使用CMake构建C/C++工程的小伙伴
- 想学习Linux下使用VSCode进行C/C++开发的小伙伴

#### 总而言之,本课程将从零开始,教会你如何在Linux开发C/C++,带领你一起打开新世界的大门~

系统环境: Ubuntu18.04 LTS 虚拟机

开发语言: C++

开发IDE: VSCode

课程宗旨:无干货,不视频。分享让生活更美好。

### 基于VSCode和CMake实现C/C++开发<Linux篇> 第一讲: 1.1 目录结构 1.2 指令与选项 1.3 重要指令讲解 + 【实战】命令行演练 1.4 文件编辑 第二讲:开发环境搭建 2.1 编译器,调试器安装 2.2 CMake安装 第三讲: GCC编译器 3.1 编译过程 3.2 g++重要编译参数 3.3【实战】g++命令行编译 3.3.1 直接编译 3.3.2 生成库文件并编译 3.3.3 运行可执行文件 第四讲: GDB调试器 4.1 常用调试命令参数 4.2【实战】命令行调试 第五讲: IDE - VSCode 5.1 界面介绍 5.2 插件安装 5.3 快捷键 5.4【实战】2个小项目 5.4.1 高频使用技巧 配套视频bilibili首发:https://b23.tv/tWqKrC

版权所有 违者必究

xiaobing

5.4.2 代码编写 5.4.3 编译并运行 第六讲: CMake 6.1 Cross-platform development 6.2 语法特性介绍 6.3 重要指令和CMake常用变量 6.3.1 重要指令 6.3.2 CMake常用变量 6.4 CMake编译工程 6.4.1 编译流程 6.4.2 两种构建方式 6.5【实战】CMake代码实践 6.5.1 最小CMake工程 6.5.2 多目录工程 - 直接编译 6.5.3 多目录工程 - 生成库编译 第七讲: 【实战】使用VSCode进行完整项目开发 案例: 士兵突击 需求: 开发: 7.1 合理设置项目目录 7.2 编写项目源文件 7.3 编写CMakeLists.txt构建项目编译规则 7.4 使用外部构建,手动编译CMake项目 7.5 配置VSCode的json文件并调试项目

第一讲:

#### 什么是linux系统?

• Linux是开源的操作系统

#### 多用户多任务:

- 单用户:一个用户,在登录计算机(操作系统),只能允许同时登录一个用户;
- 单任务: 一个任务, 允许用户同时进行的操作任务数量;
- 多用户:多个用户,在登录计算机(操作系统),允许同时登录多个用户进行操作;
- 多任务:多个任务,允许用户同时进行多个操作任务;

#### Windows属于:单用户、多任务。

Linux属于:多用户、多任务。

#### Linux一切皆文件

对于文件的操作的都有哪些种类?

• 创建文件、编辑文件、保存文件、关闭文件、重命名文件、删除文件、恢复文件。

### 1.1 目录结构

xiaobing



#### 目录结构:

- **Bin**: 全称binary, 含义是二进制。该目录中存储的都是一些二进制文件, 文件都是可以被运行 的。
- Dev: 该目录中主要存放的是外接设备,例如盘、其他的光盘等。在其中的外接设备是不能直接被 使用的,需要挂载 (类似window下的分配盘符)。
- Etc: 该目录主要存储一些配置文件。
- Home: 表示"家",表示**除了root用户以外**其他用户的家目录,类似于windows下的User/用户目录。
- Proc: 全称process, 表示进程, 该目录中存储的是Linux运行时候的进程。
- Root: 该目录是root用户自己的家目录。
- **Sbin**: 全称super binary, 该目录也是存储一些可以被执行的二进制文件, 但是必须得有super权 限的用户才能执行。
- Tmp: 表示"临时"的, 当系统运行时候产生的临时文件会在这个目录存着。
- Usr:存放的是用户自己安装的软件。类似于windows下的program files。
- Var:存放的程序/系统的日志文件的目录。
- Mnt: 当外接设备需要挂载的时候, 就需要挂载到mnt目录下。



#### 微信公众号 : VSCode bilibili : xiaobing1016



## 1.2 指令与选项

- 指令含义:
- Linux的指令是指在Linux终端(命令行)中输入的内容。
- 指令格式:
  - 完整指令的标准格式: 命令(空格) [选项](空格) [操作对象]
  - 选项和操作对象都可以没有,也可以是多个

```
1 # 指令示例: 以下两条指令等价
2 1s -1 -a -h /home ./
3 ls -lah /home ./
```

## 1.3 重要指令讲解 + 【实战】命令行演练

- pwd Print current working directory
   作用: 打印当前终端所在的目录

  - 用法: pwd

# 打印当前目录
 pwd

• Is - List directory contents

- 作用:列出当前工作目录下的所有文件/文件夹的名称
- **用法1**: Is

含义:列出当前工作目录下的文件/文件夹的名称

1 **]s** 

○ 用法2: Is [路径]

含义:列出指定路径下的所有文件/文件夹的名称

- 绝对路径:相对根目录的路径;
- 相对路径:相对当前目录的路径;

```
    # 1s 相对路径
    ls ./ #【表示当前目录下】
    ls ../ #【上一级目录下】
    # 1s 绝对路径
    5 ls /home
```

• 用法3: ls [选项] [路径]

含义: 在列出指定路径下的文件/文件夹的名称, 并以指定的格式进行显示。

```
    # ls 选项 路径
    ls -lah /home
    # 选项解释:

            -1:表示list,表示以详细列表的形式进行展示
            -a:表示显示所有的文件/文件夹(包含了隐藏文件/文件夹)
            -h:表示以可读性较高的形式显示

    # ls -l 中 "-"表示改行对应的文档类型为文件, "d"表示文档类型为文件夹。
    # 在Linux中隐藏文档一般都是以"."开头
```

• cd - change directory

- 作用: 切换当前的工作目录
- 用法1: cd; cd~

```
    # 以下两条命令等价,示直接进入当前用户的家目录下【很常用】
    cd
    cd ~
```

• 用法2: cd [相对路径]

```
    # 进入到上级目录下
    cd ...
    # 进入到上级目录中的local目录下
    cd .../local
```



xiaobing

- 1 # 进入到/usr/local目录下
- 2 cd /usr/local

#### • mkdir - make directories

- 作用: 创建目录
- 用法1: mkdir 路径
- 1 # 在当前路径下创建出目录"myfolder"
- 2 mkdir myfolder

○ 用法2: mkdir -p 路径

含义: 一次性创建多层不存在的目录

1 # 创建 ~/a/b/c 目录 2 mkdir -p ~/a/b/c

#### ○ 用法3: mkdir 路径1 [路径2] [路径3]

含义: 一次性创建多个目录

```
    # 在当前目录分别创建a、b、c三个文件夹
    mkdir a b c
```

#### • touch - change file timestamps

• 作用: 创建新文件

#### ○ 用法1: touch [路径]

```
1 # 在当前目录下创建linux.txt文件
2 touch linux.txt
3
4 # 在上级目录下创建linux文件
5 touch ../linux
6
7 # 在/home/bing/目录下创建myfile文件
8 touch /home/bing/myfile
0 用法2: touch [路径1] [路径2]
1 # 在当前目录下创建file file.txt 两个文件
2 touch file file.txt
```

#### rm - remove files or directories

- 作用:删除文件/目录
- 用法1: rm [选项] 需要移除的文件路径
- 1 # 删除当前路径下的myfile文件

2 rm myfile

- 3 # 删除/usr路径下的myfile文件
- 4 rm /usr/myfile

### ○ 用法2: rm [选项] 需要移除的目录

- 1 # 删除当前路径下的abc文件
- 2 rm -rf myfolder
- 3 # 删除/usr路径下的abc文件
- 4 rm -rf /usr/myfolder

### • cp - copy files and directories

- 作用:复制文件/文件夹到指定的位置
- 用法1: cp [被复制的文件路径] [文件被复制到的路径]

```
1 # cp命令来复制一个文件
2 cp /home/bing/myfile ./
```

○ 用法2: cp -r [被复制的文件夹路径] [文件夹被复制到的路径]\*\*

含义: -r 表示递归复制, 复制文件夹的时候需要加 -r

- 1 # 复制/home/bing/myfolder文件夹到根目录/下
- 2 cp -r /home/bing/myfolder /
- 用法3: mkdir [路径1] [路径2] [路径3]

含义: 一次性创建多个目录

- mv move (rename) files
  - 作用: 移动文件到新的位置, 或者重命名文件
  - 用法: mv [需要移动的文件路径] [需要保存的位置路径]

```
1 # 移动当前目录下myfile文件到根目录/下
2
  m∨ myfile /myfile
3
  # 移动当前目录下myfolder文件夹到根目录/下
4
5
  mv myfolder /myfolder
6
7
  # 移动当前目录下myfile到根目录/下,并重命名为myfile007
  mv myfile myfile007
8
```

- man an interface to the system reference manuals iaobing
  - 作用: 包含了Linux中全部命令手册
  - 用法: man [命令]
  - 含义: 查看命令使用手册, 按 q 退出

- 1 # 查看1s命令的手册
- 2 man ls
- 3 # 查看**cd**命令的手册
- 4 man cd
- 5 # 查看man命令的手册
- 6 man man
- reboot reboot the machine
  - 作用: 重启linux系统
  - 用法: reboot

1 # 立即重启 2 reboot

• shutdown - power-off the machine

- 作用:关机
- 用法: shut -h [时间]

1 # 立即关机 2 shutdown -h now

# 1.4 文件编辑

- Vim [file]
  - 。 所有的 Linux系统都会内建 Vi/Vim编辑器,其他的编辑器则不一定会存在
  - 。 Vim是所有Unix及Linux系统下标准的编辑器
  - Vim 具有程序开发的能力,也可以用来对文件进行简单的编辑

Vim具有"编辑器之神"的称号,学会Vim便可在Linux的世界里畅行无阻,尤其是在终端中。

♀Vim操作终极图片♀

xiaobing

微信公众号 :VSCode bilibili :xiaobing1016

[operator][count][motion] g	1 up-dow-motions	:h cmd Normal mode and help	<cr> ^m \r</cr>
	ne ts sw sts st tabstop ts Columns per tabstop	:h i cmd Insert mode cad help	
C operator. Marks and searches v WORD v WDRD count as motions, tool gu	use spaces only a a e on shiftwidth sw Columns per Bill use tabs only a a 0 off softabstop sts Spaces per tab	:h v cmd Visual mode and help	$\langle C-n \rangle$ in
Construction of the next instance [, ] ] block of "foo", giff will yank from (, , ) () block of	Set + to desired tab width (default 8) expandeab et #Zaba inserts spaces	:h g and Command-line solding and help	
yenk/copy also come before operators: ( ) how	MIXING TABS AND SPACES IS RIGHT OUT.		Jump to tag under oursor,
C Stad will delete five lines.	Replace all tabs with spaces according to current		including (tags) in help files (ESC)
(use text-objects)	i Ieccip tabatop setting	:n option option hap	T sumpleace up the tag-list (BS) ~n \D
	fileformat If Try changing this if your line-endings are messed up	:helpgrep Search through all help does!	g^] match; else list matching tags <del></del>
i(1W	et list Display whitespace visibly according to listchars		
terreter O 102 - annue D server b annue b	and I want - beginne an antit I beginne but want the	7 words	h word-motions
or and O university V WORD B strength D strength U	churactur I were e of nast W word E of nast W live 3	http://www.vimcheats	heet_com
in pattern-searches		1 WORD	
SEARCHING	D p soste after P paste tefore ^ return to formal mode	:h option	
Prev Next Forward Backward Matches	na u undo ^r redo , repret	:set opt? View current value of opt	hidden hid Lets you switch buffers without saving out a of 1 w Institution II. Show status line never (0) shares (2) or with 24 windows (1) here a status
N n * # word under cursor	d gf and file under current in path dd delete current yy yank current line	:set noopt Turn off Rag opt	hisearch his Highlight search matches. Also see 'highlight'
tx Tx upto x	and delete character % jump to replace char after cursor % metching peren <b>T</b> under cursor	:set opt Turn on flag opt	number nu Showline numbers or sur white
i i fx Fx find x	f nG jump to los A Ao jump beck Ai jump forward	:set opt=val Overwrite value of opt	showcmd sc Show commands as you type them For es di ( v
25 BARS-BOLLDIS	Jup and an Jup and an		plies ou Shawline and column number of the curror
mm set mark # mM (A-2) across * [ jump to first dow	and align top of align bottom	:set ont+=val Append to value of opt	(100) he for the the control interference which the rate address "()? It
mm set mark a (a-d) in the mM field of the set of table	align top of align bottom align bottom with a shift current line between with a shift current line between with a shift current line between a shift current line	:set opt+=val Append to value of opt :echo & opt Access opt as a variable	backspace bs Set to '2' to make backspace work like same editors '(100) with a discussion of the same editors '(1)'s b
mm         serial and track with the series         mm         set rank with the series         1         serial track with the series         1         set rank with the series         1         series         1         set rank with the series         1         series         1         set rank with the series         1         1         set rank with the series         1         set rank with the series         1         1         set rank with the series         1         1         1         set rank with the series         1         1         1         1         1         1         1         1	Bit     ZZ conterbance on zt     zt data to all conterbance on zt     all a to all conterbance on zt       Bit     all a to all conterbance on zt     all a to all conterbance on zt     all a to all conterbance on zt       Bit     all a to all conterbance on zt     all a to all conterbance on zt     all a to all conterbance on zt       Bit     all a to all conterbance on zt     all a to all all all all all all all all all al	:set opt+=val Append to value of opt :echo &opt Access opt as a variable	Vision         Safety mining of the committed of the carbon         (100)           Inackspace         Bs         Set to 21 to make backspace work like same editors         wit 2           wrap         Cantrol like empiring         Damond the empiring         Damond the empiring           backspace         Bg         Set to Yark' If you have a dark color scheme         weet.
mm retrieval. mm re	Bit         ZZ contervorsen         Zt         Als to set contervorsen         Als to set contervorsen         Zt         Als to set contervorsen         Als to set contervorsen <td>:set opt+=val Append to value of opt :echo &amp;opt Access opt as a variable</td> <td>socialize is let by the make setting wave setting and the set of the set</td>	:set opt+=val Append to value of opt :echo &opt Access opt as a variable	socialize is let by the make setting wave setting and the set of the set
Important         Important <t< td=""><td>Image: State State</td><td>:set opt+=val Append to value of opt :echo &amp;opt Access opt as a variable :ls Lot al open files Lot al open files</td><td>the sector is at the 1° in main inclusion mutual on the same efforts     the sector is at the 1° in main inclusion work loss area efforts     the sector is at the sector i</td></t<>	Image: State	:set opt+=val Append to value of opt :echo &opt Access opt as a variable :ls Lot al open files Lot al open files	the sector is at the 1° in main inclusion mutual on the same efforts     the sector is at the 1° in main inclusion work loss area efforts     the sector is at the sector i
Image: Section 2016         Image: Section 2016         1         Section 2016	2.2 million with a product of the second	:set opt+=val         Avenue to value d'out           :echo & opt         Avenue opt a value           :ls         Data avenue           :bp path         New opt avenue monthing path. Men See on other on the opt avenue monthing path. Men See on the opt avenue monthing	Implementation     implementation     implementation     implementation       <
mm         testing         mf         testing         t         testing         t         testing         t         testing         testing <thtesting< th="">         testing         te</thtesting<>	22     augusta marking and angle a	Sector opt+=val Arrow to would got Control to the sector of the sector	The sector of the sec
mm         Mm/	22     automation     2     automation     2     automation       33     automation     c     automation     but automation     but automation       4     automation     automation     but automation     but automation     but automation       5     automation     automation     but automation     but automation     but automation       6     automation     automation     but automation     but automation     but automation       6     automation     automation     automation     but automation     but automation       7     material     automation     automation     automation     automation       7     material     automation     automation     automation     automation       8     automation     automation     automation     automation     automation	:set opt+=val     Arrend to would dot       :echo &opt     Arrend to would dot       :ls     Arrend to would dot       :bp path     Arrend to would dot       :bn     Arrend to would dot       :bnext     Arrend to would dot	Image:
mm         with the second	• C finances         • C	:set opt+=val     Averde to used opt       :set opt+=val     Averde to used opt       :set opt & average     Average opt is a variable       :ls     Gara average       :bp path     Average opt is average       :bn     Average opt is average       :bn     Average opt is average       :bn     Average opt is average       :bpx:event     Average opt is average       :bpy:event     Average opt is average	The second secon
mm     time     <	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	Set opt+=val A word is word of an er of a constraint of a con	The second secon
mm     mile     <	$ \begin{array}{c} zz & we want \\ zz & z$	Set opt+=val A word to would dott     echo & opt         Accord to would dott     echo & opt         Accord to would dott     echo & opt	Image:
	ZZ     Warm (M)     ZY     Warm (M)     ZY     Warm (M)       ZZ     Warm (M)     Warm (M)     Warm (M)     Warm (M)       ZZ     Warm (M)     Warm (M)     Warm (M)       Mark (M)     Mark (M)     Mark (M)     Warm (M)       Mark (M)     Mark (M)     Mark (M)     Mark (M)       Mark (M)     Mark (M)	Sect opt+=val A word to ward of all     Sector & Sec	Import Term     The State Term Term Term Term Term Term     The State Term
mm     minimum     minimum     i. j.	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	iset opt+=val     Averdet twind end       iset opt+=val     Averdet twind end       iset opt-average     Average twind end       isb     Average twind end       ibp     Average twind end       ibdelette     Average twind end end       ibdelette     Average twind end       ibdelette     Average twind end       ibmed     Average twind end	Image: 1 and 1 and 1 and and include on which item and the second on
Image: Normal state in the	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	Set opt+=val A word to would of all control to the set of the set	The result of the second secon
mm     mem     mem <td>2 2 main with a product of the state of</td> <td><pre>:set opt+=val Aver to ward of and to the set opt and the</pre></td> <td>Import Top Top Top Top Top Top Top Top Top Top</td>	2 2 main with a product of the state of	<pre>:set opt+=val Aver to ward of and to the set opt and the</pre>	Import Top
Image: Margin and Margin an	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	<pre>:set opt+=val Aver to ward of all :set opt+=val Aver to ward of all :set opt+=val Aver to ward of all :set opt+=val Aver to the aver to all :set opt+=val Aver to the aver to all :bp ath Aver to the aver to all :bn Aver to the aver to all :bd aver to all aver to all :come W Over + New Aver to all :vsplit Soft over the Aver to all why is all W hill New Aver to all aver to all .exect to all aver to all aver to all .exect to all aver to all aver to all .exect to all aver to all .exect</pre>	Image in the standard method metho
Image: Barrier and State and	$ \begin{array}{cccc} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 $	<pre>:set opt+=val Aver to ward of all :set opt+=val Aver to ward of all :b path Aver to ward of all of</pre>	Import Table Table State
Image: Barrier and State and	$ \begin{array}{c} 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 3 \\ \mathbf$	<pre>:set opt+=val A word build got :echo &amp; pot :lb path characteristic :b path characteristic :b path characteristic :bnext App or ceft :bprev App or ceft :bprev App or ceft :bdelete America for mote suffer an :edit one af for a table :edit one af for a table :enew Ore store and for words; :vsplit Soft ourse tables are for an :vsplit Soft ou</pre>	Import
Image: Normal state         Image: Normal state       I	$ \begin{array}{c} \mathbf{Z} \mathbf{Z} & \operatorname{wind} \mathbf{Y} \\ \mathbf{Z} \mathbf{Z} & \operatorname{wind} \mathbf{Z} \mathbf{Z} \\ \mathbf{Z} & \operatorname{wind} \mathbf{Z} \\ \mathbf{Z} \mathbf{Z} \mathbf{Z} \mathbf{Z} \\ \mathbf{Z} \mathbf{Z} \mathbf{Z} \mathbf{Z} \\ \mathbf{Z} \mathbf{Z} \mathbf{Z} \mathbf{Z} \\ \mathbf{Z} \mathbf{Z} \mathbf{Z} \\ \mathbf{Z} \mathbf{Z} \mathbf{Z} \\ \mathbf{Z} \mathbf{Z} \mathbf{Z} \\ \mathbf{Z} $	<pre>:set opt+=val Aver twind ref :set opt+=val Aver twind ref :set opt+=val Aver twind ref :set opt+=val Aver twind ref :set opt+=val Aver twind ref :b path Aver twind ref aver twind :bn Aver twind ref aver twind ref :bnext Aver twind ref twind ref aver twind :bdelete Aver the for a data :enew Over + New Twind ref :vsplit Set overse added here twind :vsplit Set overse added here twind :vsplit Aver twind the twind ref aver Aver the Aver twind the twind ref aver Aver the Aver twind the twind ref :vsplit Set overse added here twind Aver the Aver twind the twind ref aver Aver the Aver twind the twind ref aver Aver the Aver twind the twind ref aver Aver the Aver twind ref aver twind the twind ref aver Aver the Aver twind ref aver twind the twind the twind ref Aver the Aver twind ref aver twind the twind the twind Aver the Aver twind ref aver twind the twind</pre>	1     1     Market in a last standard st
Image: Normal state         Image: Normal state       I	Image: Section of the section of t	<pre>:set opt+=val Aver to ward and red :echo &amp; opt+=val Aver to ward and red :set opt+=val Aver to aver and aver :set opt+=val Aver to aver aver aver :set opt+=val Aver to aver aver aver aver :bp ath Aver to aver the aver aver aver aver aver aver aver ave</pre>	Import the set of the distribution work that and the set of the distribution work the distribution work that and the set of the distribution work that and the set of the distribution work the dis
Image: Marrie	$ \begin{array}{c} \mathbf{Z} \\ \mathbf$	<pre>:set opt+=val Aver to ward and and and and average to an exact a set of a set o</pre>	
Image: Normal state in the	$ \begin{array}{c} \mathbf{Z} \mathbf{Z} & \operatorname{window} \mathbf{X} \mathbf{Z} \mathbf{U} & \operatorname{window} \mathbf{Z} \mathbf{Z} \mathbf{U} & \operatorname{window} \mathbf{Z} \mathbf{U} \\ \mathbf{Z} \mathbf{U} & \operatorname{window} \mathbf{U} \\ \mathbf{Z} \mathbf{U} & \operatorname{window} \mathbf{U} \\ \mathbf{Z} \mathbf{U} & \operatorname{window} \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \mathbf{U} \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \\ \mathbf{U} \mathbf{U} \\ \mathbf{U} \\$	<pre>set opt+=val Meet build get :echo &amp; bopt :set opt+=val Meet build get :set opt+=val Meet build get :set opt+=val Meet build get :bp path Aere build meet build get :bp ath Aere build and meet build are and a :bp ath Aere build are and a :bot ath Aere build are and a :set ath Aere and a and a and a :set ath Aere and a and a :set</pre>	Import the set of the set o

- gedit [file]
  - Linux 下的一个纯文本编辑器
  - 。 可以根据不同的语言高亮显现关键字和标识符。
- nano [file]
  - nano 是一个小巧的文本编辑器
  - 。 它比vi/vim要简单得多, 比较适合Linux初学者使用。
  - 。 某些Linux发行版的默认编辑器就是nano。

# 第二讲:开发环境搭建

### 2.1 编译器,调试器安装

- 安装GCC, GDB
- 1 sudo apt update
- 2 # 通过以下命令安装编译器和调试器
- 3 sudo apt install build-essential gdb

#### • 安装成功确认

- 1 # 以下命令确认每个软件是否安装成功
- 2 # 如果成功,则显示版本号
- 3 gcc --version
- 4 g++ --version
- 5 gdb --version

### 2.2 CMake安装

• 安装cmake

xiaobing

微信公众号 :VSCode bilibili :xiaobing1016

- 1 # 通过以下命令安装编译器和调试器
- 2 sudo apt install cmake

#### • 安装成功确认

- 1 # 确认是否安装成功
- 2 # 如果成功,则显示版本号
- 3 cmake --version

# 第三讲: GCC编译器

#### 前言:

- 1. GCC 编译器支持编译 Go、Objective-C, Objective-C ++, Fortran, Ada, D 和 BRIG (HSAIL) 等程序;
- 2. Linux 开发C/C++ 一定要熟悉 GCC
- 3. VSCode是通过调用GCC编译器来实现C/C++的编译工作的;

实际使用中:

- 使用 gcc 指令编译 C 代码
- 使用 g++指令编译 C++ 代码

### 3.1 编译过程

1. 预处理-Pre-Processing //.i文件

1 # -E 选项指示编译器仅对输入文件进行预处理

- 2 g++ -E test.cpp -o test.i //.i文件
- 2. 编译-Compiling // .s文件

```
    # -S 编译选项告诉 g++ 在为 C++ 代码产生了汇编语言文件后停止编译
    # g++ 产生的汇编语言文件的缺省扩展名是 .s
    g++ -S test.i -o test.s
```

3. 汇编-Assembling // .o文件

1 # -c 选项告诉 g++ 仅把源代码编译为机器语言的目标代码

2 # 缺省时 g++ 建立的目标代码文件有一个 .o 的扩展名。

```
3 g++ -c test.s -o test.o
```

4. 链接-Linking // bin文件

1 # -o 编译选项来为将产生的可执行文件用指定的文件名

2 g++ test.o -o test

## 3.2 g++重要编译参数

1. -g 编译带调试信息的可执行文件

1	# -g 选项告诉 GCC 产生能被 GNU 调试器GDB使用的调试信息,以调试程序。
2	
3	# 产生带调试信息的可执行文件test
4	g++ -g test.cpp

#### 2. -O[n] 优化源代码

```
## 所谓优化,例如省略掉代码中从未使用过的变量、直接将常量表达式用结果值代替等等,这些操作
1
   会缩减目标文件所包含的代码量,提高最终生成的可执行文件的运行效率。
2
3
  # -O 选项告诉 g++ 对源代码进行基本优化。这些优化在大多数情况下都会使程序执行的更快。 -O2
  选项告诉 q++ 产生尽可能小和尽可能快的代码。 如-02, -03, -0n (n 常为0-3)
4
  # -0 同时减小代码的长度和执行时间,其效果等价于-01
5
  # -00 表示不做优化
  # -01 为默认优化
6
7
  # -02 除了完成-01的优化之外,还进行一些额外的调整工作,如指令调整等。
8
  # -03 则包括循环展开和其他一些与处理特性相关的优化工作。
  # 选项将使编译的速度比使用 -0 时慢, 但通常产生的代码执行速度会更快。
9
10
  # 使用 -O2优化源代码,并输出可执行文件
11
12 g++ -02 test.cpp
```

#### 3. - I 和 - L 指定库文件 | 指定库文件路径

```
# -1参数(小写)就是用来指定程序要链接的库, -1参数紧接着就是库名
1
2
   # 在/lib和/usr/lib和/usr/local/lib里的库直接用-l参数就能链接
3
4
   # 链接glog库
5
   g++ -lglog test.cpp
6
7
   # 如果库文件没放在上面三个目录里,需要使用-L参数(大写)指定库文件所在目录
   # -L参数跟着的是库文件所在的目录名
8
9
10
  # 链接mytest库, libmytest.so在/home/bing/mytestlibfolder目录下
11
   g++ -L/home/bing/mytestlibfolder -lmytest test.cpp
```

#### 4. -I 指定头文件搜索目录

1 # -I

- 2 # /usr/include目录一般是不用指定的,gcc知道去那里找,但是如果头文件不在/usr/icnclude 里我们就要用-I参数指定了,比如头文件放在/myinclude目录里,那编译命令行就要加上-I/myinclude 参数了,如果不加你会得到一个"xxxx.h: No such file or directory"的错误。-I参数可以用相对路径,比如头文件在当前目录,可以用-I.来指定。上面我们提到的-cflags参数就是用来生成-I参数的。
- 3
- 4 g++ -I/myinclude test.cpp

5. - Wall 打印警告信息

- 1 # 打印出gcc提供的警告信息
- 2 g++ -Wall test.cpp
- 6. -w 关闭警告信息



- 1 # 关闭所有警告信息
- 2 g++ -w test.cpp

#### 7. -std=c++11 设置编译标准

# 使用 c++11 标准编译 test.cpp
 g++ -std=c++11 test.cpp

#### 8. -o 指定输出文件名

# 指定即将产生的文件名
 # 指定输出可执行文件名为test
 g++ test.cpp -o test

#### 9. -D 定义宏



示例代码:



注:使用 man gcc 命令可以查看gcc英文使用手册,见下图

xiaobing

微信公众号 :VSCode bilibili :xiaobing1016



## 3.3【实战】g++命令行编译

作者:xiaobing

案例: 最初目录结构: 2 directories, 3 files

└── <b>include</b> └── Swap.h └── main.cpp └── <b>src</b> └── Swap.cpp		
2 directories, 3 files		
1	# 最初日录结构	
2		
3	—— include	
4	Swap.h	
5	⊢ main.cpp	
6	L- src	
7	└── Swap.cpp	
8		
9	2 directories, 3 files	

#### 3.3.1 直接编译

最简单的编译,并运行

1 # 将 main.cpp src/Swap.cpp 编译为可执行文件
2 g++ main.cpp src/Swap.cpp -Iinclude
3 # 运行a.out
4 ./a.out

增加参数编译,并运行

```
      1
      # 将 main.cpp src/Swap.cpp 编译为可执行文件 附带一堆参数

      2
      g++ main.cpp src/Swap.cpp -Iinclude -std=c++11 -02 -wall -o b.out

      3
      # 运行 b.out

      4
      ./b.out
```

#### 3.3.2 生成库文件并编译

#### 链接静态库生成可执行文件①:

```
1
   ## 进入src目录下
 2
   $cd src
 3
   # 汇编, 生成Swap.o文件
 4
 5
   g++ Swap.cpp -c -I../include
   # 生成静态库libSwap.a
 6
 7
   ar rs libSwap.a Swap.o
8
9
   ## 回到上级目录
10
   $cd ..
11
12
   # 链接, 生成可执行文件: staticmain
13 g++ main.cpp -Iinclude -Lsrc -lSwap -o staticmain
```

链接动态库生成可执行文件②:

```
## 进入src目录下
 1
 2
   $cd src
 3
 4
   # 生成动态库libSwap.so
 5
   g++ Swap.cpp -I../include -fPIC -shared -o libSwap.so
 6
   ## 上面命令等价于以下两条命令
 7
   # gcc Swap.cpp -I../include -c -fPIC
8
   # gcc -shared -o libSwap.so Swap.o
9
   ## 回到上级目录
10
11
   $cd ..
12
13 # 链接, 生成可执行文件: sharemain
   g++ main.cpp -Iinclude -Lsrc -lSwap -o sharemain
14
```

#### 编译完成后的目录结构

最终目录结构: 2 directories, 8 files



### 3.3.3 运行可执行文件

#### 运行可执行文件①

1 # 运行可执行文件

2 ./staticmain

#### 运行可执行文件②

```
1 # 运行可执行文件
```

2 LD\_LIBRARY\_PATH=src ./sharemain

# 第四讲: GDB调试器

#### 前言:

- GDB(GNU Debugger)是一个用来调试C/C++程序的功能强大的调试器,是Linux系统开发 C/C++最常用的调试器
- 程序员可以使用GDB来跟踪程序中的错误,从而减少程序员的工作量。
- Linux 开发C/C++ 一定要熟悉 GDB
- VSCode是通过调用GDB调试器来实现C/C++的调试工作的;

Windows 系统中,常见的集成开发环境(IDE),如 VS、VC等,它们内部已经嵌套了相应的调试器

#### GDB主要功能:

- 设置断点(断点可以是条件表达式)
- 使程序在指定的代码行上暂停执行,便于观察
- 单步执行程序,便于调试
- 查看程序中变量值的变化
- 动态改变程序的执行环境
- 分析崩溃程序产生的core文件

### 4.1 常用调试命令参数

调试开始:执行gdb [exefilename],进入gdb调试程序,其中exefilename为要调试的可执行文件名

1 ## 以下命令后括号内为命令的简化使用,比如run (r),直接输入命令 r 就代表命令run

```
作者:xiaobing
商务邮箱:zqh0102@foxmail.com
```

-		
3 4	<pre>\$(gdb)help(h)</pre>	# 查看命令帮助,具体命令查询在gdb中输入help + 命令
5	<b>\$(gdb)run(r)</b> 件)	# 重新开始运行文件(run-text: 加载文本文件, run-bin: 加载二进制文
6		
7	\$(gdb)start	# 单步执行,运行程序,停在第一行执行语句
8		
9	<b>\$(gdb)list(l)</b> 数)	# 查看原代码(list-n,从第n行开始查看代码。list+ 函数名: 查看具体函
10		
11	\$(gdb)set	# 设置变量的值
12		
13	<pre>\$(gdb)next(n)</pre>	# 单步调试(逐过程,函数直接执行)
14		
15	<pre>\$(gdb)step(s)</pre>	# 单步调试 (逐语句: 跳入自定义函数内部执行)
16		
17	\$(gdb)backtrace	(bt) # 查看函数的调用的栈帧和层级关系
10	¢(adb)£aama(£)	4 171 42
19	\$(gob)Trame(T)	# 切供函数的代则
20	<pre>\$(adb)info(i)</pre>	# 杏看函数内部局部亦量的数值
22	(gub) mo(l)	" 已有四双门即内即又主时双臣
23	\$(adb)finish	# 结束当前函数, 返回到函数调用点
24		
25	<pre>\$(gdb)continue(</pre>	c) # 继续运行
26		
27	<pre>\$(gdb)print(p)</pre>	# 打印值及地址
28		
29	<pre>\$(gdb)quit(q)</pre>	# 退出gdb

1	<pre>\$(gdb)break+num(b)</pre>	# 在第num行设置断点
2 3	<pre>\$(gdb)info breakpoints</pre>	# 查看当前设置的所有断点
4		
5	<pre>\$(gdb)delete breakpoints nu</pre>	m(d) # 删除第num个断点
6		
7	\$(gdb)display	# 追踪查看具体变量值
8		
9	\$(gdb)undisplay	# 取消追踪观察变量
10		
11	<pre>\$(gdb)watch</pre>	# 被设置观察点的变量发生修改时,打印显示
12		
13	\$(gdb)i watch	# 显示观察点
14		
15	<pre>\$(gdb)enable breakpoints</pre>	# 启用断点
16		
17	<pre>\$(gdb)disable breakpoints</pre>	# 禁用断点
18		
19	\$(gdb)x	# 查看内存x/20xw 显示20个单元, 16进制, 4字节每单元
20		
21	\$(gdb)run argv[1] argv[2]	# 调试时命令行传参 7 7 7 7
22		
23	<pre>\$(gdb)set follow-fork-mode</pre>	child#Makefile项目管理:选择跟踪文子进程(fork())

作者:xiaobing 商务邮箱:zqh0102@foxmail.com

Tips:

- 1. 编译程序时需要加上-g,之后才能用gdb进行调试:gcc-g main.c-o main
- 2. 回车键: 重复上一命令

## 4.2【实战】命令行调试

给出一段简单代码,准备调试。

```
1
    #include <iostream>
 2
    using namespace std;
 3
 4
    int main(int argc,char **argv)
 5
    {
        int N = 100;
 6
 7
        int sum = 0;
        int i = 1;
 8
9
    // calculate sum from 1 to 100
10
11
        while (i <= N)
12
        {
13
             sum = sum + i;
             i = i + 1;
14
15
        }
16
17
        cout << "sum = " << sum << endl;</pre>
18
        cout << "The program is over." << endl;</pre>
19
20
        return 0;
21 }
```

# 第五讲: IDE - VSCode

## 5.1 界面介绍

- 1. 侧边栏
- 2. 菜单栏
- 3. 编辑区
- 4. 状态栏



#### 微信公众号 : VSCode bilibili : xiaobing1016



### 5.2 插件安装

以下三款插件是在Linux下开发C/C++的三款必备插件~

• C/C++

作者:xiaobing

- CMake
- CMake Tools

### 5.3 快捷键

#### X Visual Studio Code

Keyboard shortcuts for Linux

#### Ge

General	
Ctrl+Shift+P, F1	Show Command Palette
Ctrl+P	Quick Open, Go to File
Ctrl+Shift+N	New window/instance
Ctrl+W	Close window/instance
Ctrl+,	User Settings
Ctrl+K Ctrl+S	Keyboard Shortcuts
Basic editing	

Ctrl+X	Cut line (empty selection)
Ctrl+C	Copy line (empty selection)
Alt+ 1 / 1	Move line down/up
Ctrl+Shift+K	Delete line
Ctrl+Enter / Ctrl+Shift+Enter	Insert line below/ above
Ctrl+Shift+\	Jump to matching bracket
Ctrl+] / Ctrl+[	Indent/Outdent line
Home / End	Go to beginning/end of line
Ctrl+ Home / End	Go to beginning/end of file
Ctrl+ 1 / 1	Scroll line up/down
Alt+ PgUp / PgDn	Scroll page up/down
Ctrl+Shift+ [ / ]	Fold/unfold region
Ctrl+K Ctrl+ [ / ]	Fold/unfold all subregions
Ctrl+K Ctrl+0 / Ctrl+K Ctrl+J	Fold/Unfold all regions
Ctrl+K Ctrl+C	Add line comment
Ctrl+K Ctrl+U	Remove line comment
Ctrl+/	Toggle line comment
Ctrl+Shift+A	Toggle block comment
Alt+Z	Toggle word wrap

#### **Rich languages editing**

trl+Space	Trigger suggestion
trl+Shift+Space	Trigger parameter hints
trl+Shift+I	Format document
trl+K Ctrl+F	Format selection
12	Go to Definition
trl+Shift+F10	Peek Definition
trl+K F12	Open Definition to the side
trl+.	Quick Fix
hift+F12	Show References
2	Rename Symbol
trl+K Ctrl+X	Trim trailing whitespace
trl+K M	Change file language

#### 高频使用快捷键:

Multi-cursor and selection		
Alt+Click	Insert cursor*	
Shift+Alt+ 1 / 1	Insert cursor above/below	
Ctrl+U	Undo last cursor operation	
Shift+Alt+I	Insert cursor at end of each line selected	
Ctrl+L	Select current line	
Ctrl+Shift+L	Select all occurrences of current selection	
Ctrl+F2	Select all occurrences of current word	
Shift+Alt + →	Expand selection	
Shift+Alt + ⊷	Shrink selection	
Shift+Alt + drag mouse	Column (box) selection	
Display		

F11	Toggle full screen
Shift+Alt+0	Toggle editor layout (horizontal/vertical)
Ctrl+ = / -	Zoom in/out
Ctrl+B	Toggle Sidebar visibility
Ctrl+Shift+E	Show Explorer / Toggle focus
Ctrl+Shift+F	Show Search
Ctrl+Shift+G	Show Source Control
Ctrl+Shift+D	Show Debug
Ctrl+Shift+X	Show Extensions
Ctrl+Shift+H	Replace in files
Ctrl+Shift+J	Toggle Search details
Ctrl+Shift+C	Open new command prompt/terminal
Ctrl+K Ctrl+H	Show Output panel
Ctrl+Shift+V	Open Markdown preview
Ctrl+K V	Open Markdown preview to the side
Ctrl+K Z	Zen Mode (Esc Esc to exit)

#### Search and replace

F11 Shift Ctrl+ Ctrl+ Ctrl+

Ctrl+F	Find
Ctrl+H	Replace
F3 / Shift+F3	Find next/previous
Alt+Enter	Select all occurrences of Find match
Ctrl+D	Add selection to next Find match
Ctrl+K Ctrl+D	Move last selection to next Find match

#### Navigation

Ctrl+T	Show all Symbols
Ctrl+G	Go to Line
Ctrl+P	Go to File
Ctrl+Shift+O	Go to Symbol
Ctrl+Shift+M	Show Problems panel
F8	Go to next error or warning
Shift+F8	Go to previous error or warning
Ctrl+Shift+Tab	Navigate editor group history
Ctrl+Alt+-	Go back
Ctrl+Shift+-	Go forward
Challe Md	Toggla Tab mayor focus

#### Editor management

trl+W	Close editor
trl+K F	Close folder
trl+\	Split editor
trl+1/2/3	Focus into 1st, 2nd, 3rd editor group
trl+K Ctrl + ⊷	Focus into previous editor group
trl+K Ctrl + →	Focus into next editor group
trl+Shift+PgUp	Move editor left
trl+Shift+PgDn	Move editor right
trl+K ⊷	Move active editor group left/up
trl+K →	Move active editor group right/down

#### File management

··- ·····		
Ctrl+N	New File	
Ctrl+O	Open File	
Ctrl+S	Save	
Ctrl+Shift+S	Save As	
Ctrl+W	Close	
Ctrl+K Ctrl+W	Close All	
Ctrl+Shift+T	Reopen closed editor	
Ctrl+K Enter	Keep preview mode editor open	
Ctrl+Tab	Open next	
Ctrl+Shift+Tab	Open previous	
Ctrl+K P	Copy path of active file	
Ctrl+K R	Reveal active file in Explorer	
Ctrl+K O	Show active file in new window/instance	

#### Debug

F9	Toggle breakpoint
F5	Start / Continue
F11 / Shift+F11	Step into/out
F10	Step over
Shift+F5	Stop
Ctrl+K Ctrl+I	Show hover

#### Integrated terminal

Ctrl+'	Show integrated terminal
Ctrl+Shift+`	Create new terminal
Ctrl+Shift+C	Copy selection
Ctrl+Shift+V	Paste into active terminal
Ctrl+Shift+ ↑/↓	Scroll up/down
Shift+ PgUp / PgDn	Scroll page up/down
Shift+ Home / End	Scroll to top/bottom

\* The Alt+Click gesture may not work on some Linux distributions. You can change the modifier key for the Insert cursor command to Ctrl+Click with the "editor.multiCursorModifier" setting.



功能	快捷键	功能	快捷键
转到文件 / 其他常用操作	Ctrl + P	关闭当前文件	Ctrl + W
打开命令面板	Ctrl + Shift + P	当前行上移/下移	Alt + Up/Down
打开终端	Ctrl +`	变量统一重命名	F2
关闭侧边栏	Ctrl + B	转到定义处	F12
复制文本	Ctrl+C	粘贴文本	Ctrl+V
保存文件	Ctrl+S	撤销操作	Ctrl+Z

#### 在 Ctrl+P 窗口下还可以:

- 直接输入文件名, 跳转到文件
- ? 列出当前可执行的动作
- ! 显示 Errors 或 Warnings , 也可以 Ctrl+Shift+M
- : 跳转到行数,也可以 Ctrl+G 直接进入
- @ 跳转到 symbol (搜索变量或者函数),也可以 Ctrl+Shift+O 直接进入
- @ 根据分类跳转 symbol , 查找属性或函数, 也可以 Ctrl+Shift+O 后输入:进入
- # 根据名字查找 symbol, 也可以 Ctrl+T

#### 快捷键:编辑器与窗口管理

- 1. 打开一个新窗口: Ctrl+Shift+N
- 2. 关闭窗口: Ctrl+Shift+W
- 3. 同时打开多个编辑器 (查看多个文件)
- 4. 新建文件 Ctrl+N
- 5. 文件之间切换 Ctrl+Tab
- 6. 切出一个新的编辑器(最多 3 个) Ctr1+\, 也可以按住 Ctr1 鼠标点击 Explorer 里的文件名
- 7. 左中右 3 个编辑器的快捷键 Ctrl+1 Ctrl+2 Ctrl+3
- 8.3 个编辑器之间循环切换 Ctrl+
- 9. 编辑器换位置, Ctrl+k 然后按 Left 或 Right

#### ↓代码编辑相关的快捷键↓

#### 快捷键: 格式调整

- 1. 代码行缩进 Ctrl+[ 、 Ctrl+]
- 2. Ctrl+C 、 Ctrl+V 复制或剪切当前行/当前选中内容
- 3.代码格式化: Shift+Alt+F,或 Ctrl+Shift+P 后输入 format code
- 4. 上下移动一行: Alt+Up 或 Alt+Down
- 5. 向上向下复制一行: Shift+Alt+Up 或 Shift+Alt+Down
- 6. 在当前行下边插入一行 Ctrl+Enter
- 7. 在当前行上方插入一行 Ctrl+Shift+Enter

#### 快捷键:光标相关

- 1. 移动到行首: Home
- 2. 移动到行尾: End
- 3. 移动到文件结尾: Ctrl+End
- 4. 移动到文件开头: Ctrl+Home
- 5. 移动到定义处: F12

xiaobing

- 6. 定义处缩略图:只看一眼而不跳转过去 Alt+F12
- 7. 移动到后半个括号: Ctrl+Shift+]
- 8. 选择从光标到行尾: Shift+End
- 9. 选择从行首到光标处: Shift+Home
- 10. 删除光标右侧的所有字: Ctrl+Delete
- 11. 扩展/缩小选取范围: Shift+Alt+Left 和 Shift+Alt+Right
- 12. 多行编辑(列编辑): Alt+Shift+鼠标左键, Ctrl+Alt+Down/Up
- 13. 同时选中所有匹配: Ctrl+Shift+L
- 14. Ctrl+D 下一个匹配的也被选中 (在 sublime 中是删除当前行,后面自定义快键键中,设置与 Ctrl+shift+K 互换了)
- 15. 回退上一个光标操作: Ctrl+U

#### 快捷键: 重构代码

- 1. 找到所有的引用: Shift+F12
- 2. 同时修改本文件中所有匹配的: Ctrl+F12
- 3. 重命名:比如要修改一个方法名,可以选中后按 F2,输入新的名字,回车,会发现所有的文件都 修改了
- 4. 跳转到下一个 Error 或 Warning:当有多个错误时可以按 F8 逐个跳转
- 5. 查看 diff: 在 explorer 里选择文件右键 Set file to compare, 然后需要对比的文件上右 键选择 Compare with file\_name\_you\_chose

#### 快捷键:查找替换

- 1. 查找 Ctrl+F
- 2. 查找替换 Ctrl+H
- 3. 整个文件夹中查找 Ctrl+Shift+F

#### 快捷键:显示相关

- 1. 全屏: F11
- 2. zoomln/zoomOut: Ctrl +/-
- 3. 侧边栏显/隐: Ctrl+B
- 4. 显示资源管理器 Ctrl+shift+E
- 5. 显示搜索 Ctrl+Shift+F
- 6.显示Git Ctrl+Shift+G
- 7. 显示 Debug Ctrl+Shift+D
- 8. 显示 Output Ctrl+Shift+U

## 5.4【实战】2个小项目

通过手写2个小项目,呈现出基本的C++工程建立的过程,并编译运行这两个小项目

- 项目1: Hello world
- 项目2: Swap with class

### 5.4.1 高频使用技巧

- 左右分屏
- 固定打开的文件
- 格式化代码
- Rename Symbol
- 显示minimap

xiaobing

- 侧边栏查看OUTLINE
- 全屏
- 关闭多个打开的文件
- 5.4.2 代码编写
- 5.4.3 编译并运行

# 第六讲: CMake

#### 前言:

- CMake是一个跨平台的安装编译工具,可以用简单的语句来描述所有平台的安装(编译过程)。
- CMake可以说已经成为**大部分C++开源项目标配**

## 6.1 Cross-platform development

Let's assume you have some cross-platform project with C++ code shared along different platforms/IDEs. Say you use Visual Studio on Windows, xcode on OSX and Makefile for Linux:



What you will do if you want to add new bar.cpp source file? You have to add it to every tool you use:

xiaobing



To keep the environment consistent you have to do the similar update several times. And the most important thing is that you have to do it **manually** (arrow marked with a red color on the diagram in this case). Of course such approach is error prone and not flexible.

CMake solve this design flaw by adding extra step to development process. You can describe your project in CMakeLists.txt file and use CMake to generate tools you currently interested in using cross-platform CMake code:





Same action - adding new bar.cpp file, will be done in **one step** now:





Note that the bottom part of the diagram was not changed. I.e. you still can keep using your favorite tools like visual Studio/msbuild, Xcode/xcodebuild and Makefile/make!

## 6.2 语法特性介绍

- 基本语法格式:指令(参数1参数2...)
  - 参数使用**括弧**括起
  - 参数之间使用**空格**或**分号**分开
- 指令是大小写无关的,参数和变量是大小写相关的

```
1 set(HELLO hello.cpp)
```

- 2 add\_executable(hello main.cpp hello.cpp)
- 3 ADD\_EXECUTABLE(hello main.cpp \${HELLO})
- ・ 变量使用\${}方式取值,但是在 IF 控制语句中是直接使用变量名
   3 重要指令和CMake常用变量

6.3 重要指令和CMake常用变量

#### 6.3.1 重要指令

- cmake\_minimum\_required 指定CMake的最小版本要求
  - 语法: cmake\_minimum\_required(VERSION versionNumber [FATAL\_ERROR])

```
1 # CMake最小版本要求为2.8.3
2 cmake_minimum_required(VERSION 2.8.3)
```

- project 定义工程名称,并可指定工程支持的语言
  - 语法: project(projectname [CXX] [C] [Java])

```
1 # 指定工程名为HELLOWORLD
2 project(HELLOWORLD)
```

#### • **set** - 显式的定义变量

○ 语法: set(VAR [VALUE] [CACHE TYPE DOCSTRING [FORCE]])

1 # 定义SRC变量,其值为sayhello.cpp hello.cpp

- 2 set(SRC sayhello.cpp hello.cpp)
- include\_directories 向工程添加多个特定的头文件搜索路径 --->相当于指定g++编译器的-l参数
  - 语法: include\_directories([AFTER|BEFORE] [SYSTEM] dir1 dir2 ...)

1 # 将/usr/include/myincludefolder 和 ./include 添加到头文件搜索路径

- 2 include\_directories(/usr/include/myincludefolder ./include)
- link\_directories 向工程添加多个特定的库文件搜索路径 --->相当于指定g++编译器的-L参数
  - 语法: link\_directories(dir1 dir2...)

```
1 # 将/usr/lib/mylibfolder 和 ./lib 添加到库文件搜索路径
```

2 link\_directories(/usr/lib/mylibfolder ./lib)

• add\_library - 生成库文件

• 语法: add\_library(libname [SHARED|STATIC|MODULE] [EXCLUDE\_FROM\_ALL] source1 source2 ... sourceN)

```
1 # 通过变量 SRC 生成 libhello.so 共享库
```

- 2 add\_library(hello SHARED \${SRC})
- add\_compile\_options 添加编译参数
  - 语法: add\_compile\_options(
  - 1 # 添加编译参数 -Wall -std=c++11 -02 2 add\_compile\_options(-wall -std=c++11 -02)
- add\_executable 生成可执行文件
- dd\_executable 生成可执行文件 。 语法:add\_executable(exename source1-source2....sourceN)

- 1 # 编译main.cpp生成可执行文件main
- 2 add\_executable(main main.cpp)
- target\_link\_libraries 为 target 添加需要链接的共享库 --->相同于指定g++编译器-l参数
  - 语法: target\_link\_libraries(target library1<debug | optimized> library2...)

```
1 # 将hello动态库文件链接到可执行文件main
2 target_link_libraries(main hello)
```

- add\_subdirectory 向当前工程添加存放源文件的子目录,并可以指定中间二进制和目标二进制 存放的位置
  - 语法: add\_subdirectory(source\_dir [binary\_dir] [EXCLUDE\_FROM\_ALL])

```
1 # 添加src子目录, src中需有一个CMakeLists.txt
2 add_subdirectory(src)
```

- aux\_source\_directory 发现一个目录下所有的源代码文件并将列表存储在一个变量中,这个指 令临时被用来自动构建源文件列表
  - 语法: aux\_source\_directory(dir VARIABLE)

```
1 # 定义SRC变量,其值为当前目录下所有的源代码文件
```

```
2 aux_source_directory(. SRC)
```

```
3 # 编译SRC变量所代表的源代码文件,生成main可执行文件
```

```
4 add_executable(main ${SRC})
```

### 6.3.2 CMake常用变量

- CMAKE\_C\_FLAGS gcc编译选项
- CMAKE\_CXX\_FLAGS g++编译选项

```
1 # 在CMAKE_CXX_FLAGS编译选项后追加-std=c++11
```

```
set( CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
2
```

• CMAKE\_BUILD\_TYPE 编译类型(Debug, Release)

```
1 # 设定编译类型为debug,调试时需要选择debug
```

```
2 set(CMAKE_BUILD_TYPE Debug)
```

- 3 # 设定编译类型为release,发布时需要选择release
- 4 **set**(CMAKE\_BUILD\_TYPE Release)
- CMAKE\_BINARY\_DIR

### PROJECT\_BINARY\_DIR

### \_BINARY\_DIR

- 这三个变量指代的内容是一致的。
   如果是 in source build, 指的就是工程顶层目录。
   如果是 out-of-source 编译,指的是工程编译发生的目录。

4. PROJECT\_BINARY\_DIR 跟其他指令稍有区别,不过现在,你可以理解为他们是一致的。

### • CMAKE\_SOURCE\_DIR

PROJECT\_SOURCE\_DIR \_SOURCE\_DIR

- 1. 这三个变量指代的内容是一致的,不论采用何种编译方式,都是工程顶层目录。
- 2. 也就是在 in source build时,他跟 CMAKE\_BINARY\_DIR 等变量一致。
- 3. PROJECT\_SOURCE\_DIR 跟其他指令稍有区别,现在,你可以理解为他们是一致的。
- CMAKE\_C\_COMPILER: 指定C编译器
- CMAKE\_CXX\_COMPILER: 指定C++编译器
- EXECUTABLE\_OUTPUT\_PATH: 可执行文件输出的存放路径
- LIBRARY\_OUTPUT\_PATH: 库文件输出的存放路径

## 6.4 CMake编译工程

CMake目录结构:项目主目录存在一个CMakeLists.txt文件

### 两种方式设置编译规则:

- 1. 包含源文件的子文件夹**包含**CMakeLists.txt文件,主目录的CMakeLists.txt通过add\_subdirectory 添加子目录即可;
- 2. 包含源文件的子文件夹**未包含**CMakeLists.txt文件,子目录编译规则体现在主目录的 CMakeLists.txt中;

### 6.4.1 编译流程

### 在 linux 平台下使用 CMake 构建C/C++工程的流程如下:

- 手动编写 CMakeLists.txt。
- 执行命令 cmake PATH 生成 Makefile (PATH 是顶层CMakeLists.txt 所在的目录)。
- 执行命令 make 进行编译。

1	# impor	tant tips
2		# 表示当前目录
3	./	# 表示当前目录
4		
5		# 表示上级目录
6	/	# 表示上级目录

### 6.4.2 两种构建方式

• **内部构建(in-source build)**:不推荐使用 内部构建会在同级目录下产生一大堆中间文件,这些中间文件并不是我们最终所需要的,和工程源 文件放在一起会显得杂乱无章。

```
    ## 内部构建
    # 在当前目录下,编译本目录的CMakeLists.txt,生成Makefile和其他文件
    cmake .
    # 执行make命令,生成target
    make
```

• 外部构建(out-of-source build): 推荐使用

将编译输出文件与源文件放到不同目录中

```
1 ## 外部构建
2
3 # 1. 在当前目录下,创建build文件夹
4 mkdir build
5 # 2. 进入到build文件夹
6 cd build
7 # 3. 编译上级目录的CMakeLists.txt,生成Makefile和其他文件
8 cmake ..
9 # 4. 执行make命令,生成target
10 make
```

## 6.5【实战】CMake代码实践

针对第五章写的两个小项目来写对应的CMakeLists.txt

### 6.5.1 最小CMake工程

```
1 # Set the minimum version of CMake that can be used
2 cmake_minimum_required(VERSION 3.0)
3 
4 # Set the project name
5 project (HELLO)
6 
7 # Add an executable
8 add_executable(hello_cmake main.cpp)
```

### 6.5.2 多目录工程 - 直接编译

```
# Set the minimum version of CMake that can be used
 1
 2
    cmake_minimum_required(VERSION 3.0)
 3
 4
    #project name
 5
    project(SWAP)
 6
 7
    #head file pat
    include_directories( include )
 8
 9
10
    #source directory files to var
11
    add_subdirectory( src DIR_SRCS )
12
   #add executable file
add_executable(swap_02 ${TEST_MATHE}) 10000102
#add link library
13
14
15
16
   #add link library
```

17 target\_link\_libraries(\${FS\_BUILD\_BINARY\_PREFIX}sqrt \${LIBRARIES})

### 6.5.3 多目录工程 - 生成库编译

```
# Set the minimum version of CMake that can be used
1
2
   cmake_minimum_required(VERSION 3.0)
3
4
   #project name
5
   project(SWAP_LIBRARY)
6
   #add compile options
7
   add_compile_options("-wall -std=c++11")
8
9
10
   #set CMAKE_BUILD_TYPE
11
   set( CMAKE_BUILD_TYPE Debug )
12
13
   # set output binary path
   set(EXECUTABLE_OUTPUT_PATH ${PROJECT_BINARY_DIR}/bin)
14
15
   *****
16
17
   # Create a library
18
   ***********
19
20
   #Generate the static library from the library sources
21
   add_library( swap_library STATIC src/Swap.cpp )
22
23
   target_include_directories( swap_lib PUBLIC ${PROJECT_SOURCE_DIR}/include )
24
25
   *****
   # Create an executable
26
   27
28
   # Add an executable with the above sources
29
30
   add_executable( swap_01 main.cpp )
31
32
  # link the new swap_01 target with the swap_lib target
33 target_link_libraries( swap_01 swap_liby )
```

# 第七讲:【实战】使用VSCode进行完整项目开发

### 案例: 士兵突击

#### 需求:

- 1. **士兵 许三多** 有一把枪, 叫做 AK47
- 2. **士兵** 可以 开火
- 3. 士兵 可以 给枪装填子弹
- 4. 枪能够发射子弹
- 5. 枪能够装填子弹 —— 增加子弹数量

### 开发:

• 开发枪类

<b>•</b>	1 A A	
Sol	IC	er

string \_name; Gun\* \_ptr\_gun;

Solider(string name); addBulletToGun(int num); fire();

## 7.1 合理设置项目目录

7.2 编写项目源文件

- 7.3 编写CMakeLists.txt构建项目编译规则
- 7.4 使用外部构建,手动编译CMake项目
- 7.5 配置VSCode的json文件并调试项目

## Gun

string \_type; Int \_bullet\_count;

Gun(string type); addBullet(int num); shoot();



第 30 页,共 30 页