1. VMware 12不支持Ubuntu 18.04, 使用VMware 14.1.1安装成功
2. 安装source code pro字体，下载好预编译的字体

在用户目录下新建.fonts文件夹，拷贝TTF,OTF文件夹至该目录下，执行命令

fc-cache –fv

1. Ubuntu 16 Terminal 字体外观中有 Allow bold text选项，可以使得终端中没有粗体字。但是在Ubuntu 18的Terminal中没有发现此选项。

* 命令ls下显示的粗体字（比如文件夹）

$ dircolors -p > ~/.dircolors

修改DIR 01；34 为 00；34 // 01 bold 00 normal

* 命令提示符（PS1）粗体

修改.bashrc文件PS1的定义

01的地方改为00 // 01 bold 00 normal

1. 修改gnome主题

$ sudo apt install gnome-tweak-tool

$ gnome-tweaks

默认gnome-shell加锁的

$ sudo apt install gnome-shell-extensions

$ reboot

打开shell theme开关，就可以切换shell主题了

1. Ubuntu Dock自动隐藏

Settings -> Dock -> Auto-hide the Dock (enable it)

1. Install Clang

<http://clang.llvm.org/get_started.html>

Note: cmake -G “Unix Makefiles” -DCMAKE\_BUILD\_TYPE=Release ../llvm

默认是debug模式

* Check out LLVM

$ cd ~/Downloads

$ svn co <http://llvm.org/svn/llvm-project/llvm/trunk> llvm

* Check out Clang

$ cd llvm/tools

$ svn co <http://llvm.org/svn/llvm-project/cfe/trunk> clang

$ cd ../..

* Check out extra Clang tools

$ cd llvm/tools/clang/tools

$ svn co <http://llvm.org/svn/llvm-project/clang-tools-extra/trunk> extra

$ cd ../../..

* Check out Compiler-RT

$ cd llvm/projects

$ svn co <http://llvm.org/svn/llvm-project/compiler-rt/trunk> compiler-rt

$ cd ../..

* Check out libcxx (libc++ is a new implementation of the C++ standard library)

$ cd llvm/projects

$ svn co <http://llvm.org/svn/llvm-project/libcxx/trunk> libcxx

$ cd ../..

* Check out libcxxabi

$ cd llvm/projcts

$ svn co <http://llvm.org/svn/llvm-project/libcxxabi/trunk> libcxxabi

$ cd ../..

* Check out LLD

$ cd llvm/tools

$ svn co <http://llvm.org/svn/llvm-project/lld/trunk> lld

$ cd ../..

* Check out LLDB

$ cd llvm/tools

$ svn co <http://llvm.org/svn/llvm-project/lldb/trunk> lldb

$ cd ../..

Note:

See <http://lldb.llvm.org/build.html>

Preliminaries: Swig, libedit, Python

$ sudo apt install swig libedit-dev libncurses5-dev python-dev

* Build LLVM and Clang

$ mkdir build

$ cd build

$ cmake -G “Unix Makefiles” -DCMAKE\_BUILD\_TYPE=Release ../llvm

$ make

$ sudo make install

Note:

默认安装目录/usr/local

【 update: SVN deprecated 2020.5.28 】

* Check out the LLVM project

$ cd ~/Downloads

$ git clone https://github.com/llvm/llvm-project.git llvm

* Configure and build LLVM

$ cd llvm

$ mkdir build

$ cd build

$ cmake -G "Unix Makefiles" -DLLVM\_ENABLE\_PROJECTS="clang;clang-tools-extra;libcxx;libcxxabi;libunwind;lldb;compiler-rt;lld;polly;pstl" -DCMAKE\_BUILD\_TYPE=Release ../llvm

// Starting with LLVM 12.0.0, the minimum version of CMake required to build LLVM will become 3.13.4

$ make

$ sudo make install

遇到的问题

* + - 1. Not found epydoc

$ sudo apt install python-epydoc

* + - 1. /usr/bin/ld: cannot find –lz

Ans:

$ sudo apt install zlib1g-dev

* + - 1. LibXml2 cannot found

$ sudo apt install libxml2-dev

* + - 1. Z3 cannot found

$ git clone <https://github.com/Z3Prover/z3.git>

$ cd z3

$ python scripts/mk\_make.py

$ cd build

$ make

$ sudo make install

* + - 1. Could not find Ocaml

$ sudo apt install ocaml

* + - 1. C++: fatal error: Killed signal terminated program cc1plus

因为内存不足，在虚拟机设置中增大内存

1. Using Clang

* Using libc++

See：<https://libcxx.llvm.org/docs/UsingLibcxx.html>

$ clang++ -stdlib=libc++ test.cpp

Clang寻找libc++的头文件路径/usr/include/c++/v1

Clang寻找libc++的库路径/usr/lib

* 当安装的libc++库不在标准路径中时

$ clang++ -stdlib=libc++ -nostdinc++ \

-I /usr/local/include/c++/v1 \

-L /usr/local/lib \

-Wl,-rpath, /usr/local/lib

Test.cpp

Note:

Libc++安装路径是/usr/local

-Wl,-rpath,/usr/local/lib 指定运行库查找路径

实例：

在/usr/lib中建立软链接libc++.so.1, libc++abi.so.1至/usr/local/lib中对应的文件

$ clang++ -stdlib=libc++ -nostdinc++ \

-I /usr/local/include/c++/v1 \

Test.cpp

这时可以不使用-Wl,-rpath,<libcxx-install-prefix>

或者

$ clang++ -stdlib=libc++ Test.cpp

-nostdinc++ 表示不在标准路径中找头文件，需要在-I指定目录中找

而标准路径有/usr/include, /usr/local/include, 所以我安装的头文件是在标准路径中

1. Install Chrome

$ sudo wget <http://www.linuxidc.com/files/repo/google-chrome.list> -P /etc/apt/sources.list.d/

$ wget -q -O – <https://dl.google.com/linux/linux_signing_key.pub> | sudo apt-key add –

$ sudo apt upate

$ sudo apt install google-chrome-stable

1. Install Chinese input method

* $ sudo apt install ibus-pinyin
* Settings - Region&Language - Input Sources

Click ‘Manage Installed Language’, 添加中文字体，重启

Click ‘+’ , 选择‘Chinese Intelligent Pinyin’

1. Install vim-plug

* Download

$ curl –fLo ~/.vim/autoload/plug.vim –create-dirs \

https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim

* Configure

$ mkdir ~/.vim/plugged

$ vim ~/.vimrc

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

call plug#begin(‘~/.vim/plugged’)

Plug ‘fatih/vim-go’

call plug#end()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Install boost

$sudo apt install libboost-all-dev

查看安装的版本：

$ dpkg –S /usr/include/boost/version.hpp

<https://www.boost.org/doc/libs/1_73_0/more/getting_started/unix-variants.html>

* Download boost\_1\_73\_0.tar.bz2
* In the directory where you want to put the Boost installation, extract it
* Most Boost libraries are header-only

使用Boost的时候只要指明boost路劲即可

Eg: clang++ -I path/to/boost\_1\_73\_0 example.cc

* If you want to use any of the separately-compiled Boost libraries

Build and Install

$ cd path/to/boost\_1\_73\_0

$ ./bootstrap.sh –prefix=…

$ sudo ./b2 install

1. Install libstdc++ man

$ sudo apt install libstdc++-7-doc

1. Install Rust

<https://www.rust-lang.org/tools/install>

$ curl <https://sh.rustup.rs> –sSf | sh

1. Install GCC (Build from source)

<https://gcc.gnu.org/>

* Download the source

Download form one of the available mirrors at <https://gcc.gnu.org/mirrors.html>

***gcc-9.2.0.tar.xz***

* Decompress

$ cd ~/Downloads

$ xz –d gcc-9.2.0.tar.xz

$ tar –xvf gcc-9.2.0.tar

* Download GCC’s prerequisites

$ cd gcc-9.2.0

$ ./contrib/download\_prerequisites

--- snip ---

gmp-6.1.0.tar.bz2: OK

mpfr-3.1.4.tar.bz2: OK

mpc-1.0.3.tar.gz: OK

isl-0.18.tar.bz2: OK

All prerequisites downloaded successfully.

* Configure GCC

$ cd ../

$ mkdir gcc-9.2.0-build

$ cd gcc-9.2.0-build

$ ../gcc-9.2.0/configure –disable-multilib

* Build GCC

$ make

* Test whether GCC binaries built correctly

$ make –k check

[

Optional:

$ sudo apt install autogen

]

* Install GCC

$ sudo make install

$ sudo vi /etc/ld.so.conf.d/x86\_64-linux-gnu.conf

--- snip ---

/usr/local/lib64

…

$ sudo ldconfig

1. Install Julia 1.2.0

See <https://julialang.org/downloads/platform.html>

* Download the official binaries and extract the file to a folder
* Add Julia’s bin folder to your system PATH environment variable

1. Install latest CMake

<https://cmake.org/download/>

* Download binary distributions
* Extract to /opt
* Write path into $PATH

$ vi .profile

export PATH=”/opt/cmake-3.17.3-Linux-x86\_64/bin:$PATH”

1. Install GCC (package manager)

* 将ubuntu-toolchain-r/test PPA添加到系统

$ sudo apt install software-properties-common

$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test

* 需要安装什么版本就输入什么版本

$ sudo apt install gcc-9 g++-9

* 设定每个版本优先级

$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 70 --slave /usr/bin/g++ g++ /usr/bin/g++-7

$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 90 --slave /usr/bin/g++ g++ /usr/bin/g++-10

* 更改默认版本

$ sudo update-alternatives --config gcc

1. Reserved