# Linux下项目开发

## 1 SSH终端仿真工具 – secureCRT

### 1.1 连接配置

File->Quick Connect

配置Hostname 为linux主机地址

配置port为SSH端口，通常是22

配置Username为登陆用户名

点击connect，填写Password为该用户对应的密码。

### 1.2 界面显示配置

界面配置分为全局配置和会话配置两种，分别为Option->Global Options和Options->Session Options。

**选择终端类型**

Options->Global Options->General->Default Session->Edit Default Setting

->Terminal->Emulation选择终端

**选择主题**

Options->Global Options->Terminal->Advanced选择主题

### 1.3 上传下载文件

Centos中安装ftp上传下载文件工具: yum install lrzsz

上传文件到linux时，使用命令 rz, 打开文件浏览器，选择相应的文件。

下载文件到windows时，使用命令 sz 文件名。

## 2 编辑器 – vim

### 2.1 vim四种模式

命令模式(command-mode) 输入操作命令

插入模式(insert-mode) 编辑文档

可视模式(visual-mode) 选择文档内容

正常模式(normal-mode)

模式的转换

① 其它模式==>正常模式

按 **Esc**键

② 正常模式==>插入模式

按 **i** 在光标前插入

按 **shit** + **i** 在行首插入

按 **a** 在光标后插入

按 **s** 删除光标所在的字符再插入

按 **shift** + **a** 在行末插入

按 **o** 在当前行之下新建行

按 **shift** + **o** 在当前行之上新建行

按 **shift** +**s** 删除光标所在行再插入

③ 正常模式==>命令模式

按 **shift** + **:** 命令模式

④ 正常模式==>可视模式

按 **v** 可视模式

按 **shift** + **v** 可视块模式

### 2.2 基本操作

**显示行号**

命令模式下输入 **set nu** 显示行号。

命令模式下输入 **set nonu** 取消显示行号。

**光标移动**

正常模式下输入 **k** 光标向上移动，输入 **j** 光标向下移动，输入**h** 光标向左移动，输入 **l**光标向右移动。

**光标跳转**

正常模式下输入 **gg** 跳转到首行，输入 **shift** + **g** 跳转到尾行。

命令模式下输入任意数字**N**，跳转到第N行。

正常模式下输入 **ctrl** + **o** 跳转到上一次光标位置，输入 **ctrl** + **i**跳转到下一次光标位置。

**头文件跳转**

正常模式下输入 **gf** 跳转到光标所在的头文件。

**搜索**

命令模式下输入 **?text** 查找text字符串

正常模式下输入 **shift** + **3** 查找光标所在的单词。

正常模式下输入 **n** 向上查找，输入 **shift** + **n** 向下查找。

**替换**

命令模式下输入 **%s/a/b/g** 将当前文件中的所有a都替换成b。

命令模式下输入 **1,5s/a/b/g** 将当前文件中的第1行到第5行的所有a都替换成b。

正常模式下输入 **shift** + **r** ，输入要替换的内容。

**复制**

可视模块下选择要复制的内容，输入 **y** 复制到引号寄存器上。

可视模块下选择要复制的内容，输入 **”+y** 复制到加号寄存器上。

注：加号寄存器支持系统剪切板，但是必须要使用gvim才能将复制到加号寄存器中的内容在其他应用程序中使用。如果要查看所有寄存器内容可以在命令模式下输入 **reg**。

**粘贴**

在正常模式下输入 **p** 将引号寄存器的内容复制到当前位置上。

在正常模式下输入 **“+p** 将加号寄存器的内容复制到当前位置上。

**插入模式下粘贴**

命名模式输入**set paste**后，在插入模式下粘贴忽略格式。

**内容移动**

可视模块下选择要移动的内容，输入 **>** 向右移动一个tab。

可视模块下选择要移动的内容，输入 **<** 向左移动一个tab。

可视模块下选择要移动的内容，输入**d**，光标移动到位置A，输入**p**，将移动的内容移动到A处。

命令模式下输入**1 mo 2**把第1行移动到第2行。

命令模式下输入**1,3 mo 4**把第1行到第3行移动到第4行。

**批量注释**

**ctrl** + **v**进入可视模式，**shift** + **i** + **//**，然后按**Esc**键。

**取消注释**

**ctrl** +**v** 选择所列的//，然后按**d**。

**补全**

插入模式下输入**ctrl** + **n** 或者**ctrl** + **p**。

**分屏操作**

命令模式下输入 **sp filename**，水平打开文件名为filename的文件。

命令模式下输入 **vsp filename**，垂直打开文件名为filename的文件。

在两个屏幕间切换，可以再正常模式下输入 **ctrl** + **w** + **w** 。

**撤销**

正常模式下输入 **u**。

**保存**

命令模式下输入 **w**。

**退出**

命令模式下输入 **q**。

命令模式下输入 **wq** ，退出并保存。

命令模式下输入 **q!** ，退出不保存。

### 2.3 文件配置

vim的文件配置有两个路径，/etc/vimrc和~/.vimrc，当存在~/.vimrc时加载~/.vimrc文件，当~/.vimrc不存时加载/etc/vimrc文件。

**vimrc文件的简单示例**

"设置语言编码

set encoding=utf-8

set langmenu=zh\_CN.UTF-8

"设置tab为4个空格

set expandtab

set tabstop=4

set softtabstop=4

set shiftwidth=4

"设置c语言风格缩进

set cindent

"设置行号

set number

"设置颜色显示为256

set t\_Co=256

"设置主题

color zellner

### 2.4 插件配置

#### 2.4.1 ctags

ctags工具是用来遍历源代码文件生成tags文件，这些tags文件能被编辑器或其它工具用来快速查找定位源代码中的符号（tag/symbol），如变量名，函数名等。

**安装ctags**

yum install ctags

**配置ctags**

在~./vimrc中添加

let Tlist\_Sort\_Type = "name"

let Tlist\_Use\_Right\_Window = 1 " 在右侧显示窗口

let Tlist\_Compart\_Format = 1 " 压缩方式

let Tlist\_Exist\_OnlyWindow = 1 " 如果只有一个buffer，kill窗口也kill掉buffer

"设置tags

set tags=tags;

set autochdir

**生成ctags**

ctags -R --c++-kinds=+p --fields=+iaS --extra=+q src\_dir

上述选项的含义如下：

--c++-kinds=+p : 为C++文件增加函数原型的标签

--fields=+iaS : 在标签文件中加入继承信息(i)、类成员的访问控制信息(a)、以及函数的指纹(S)

--extra=+q: 为标签增加类修饰符。注意，如果没有此选项，将不能对类成员补全

**使用ctags**

vim 正常模式下输入 **ctrl** + **]** 跳转到定义。

vim 正常模式下输入 **ctrl** +**t** 调回原来的位置。

#### 2.4.2 taglist

taglist基于ctags，它可以在vim中以分割窗口的形式显示当前的代码结构概览，增加代码浏览的便利程度。

**安装taglist**

下载taglist.vim到~/.vim/plugin中

**配置taglist**

let Tlist\_Auto\_Open=0

let Tlist\_Ctags\_Cmd = '/usr/bin/ctags'

let Tlist\_Show\_One\_File = 1 "不同时显示多个文件的tag，只显示当前文件的

let Tlist\_File\_Fold\_Auto\_Close = 1

let Tlist\_Exit\_OnlyWindow = 1 "如果taglist窗口是最后一个窗口，则退出vim

let Tlist\_Use\_Right\_Window = 1 "在右侧窗口中显示taglist窗口

map <F12> :Tlist <CR>

**使用taglist**

vim 正常模式下输入**F12**，打开tlist窗口，按**ctrl** + **w** + **w**在两个窗口中切换

## 3 编译器 – g++

### 3.1 g++本身使用命令

g++编译器根据不同的参数选项可以生成从预处理到生成可执行文件中间的任何一个中间文件。

**生成可执行文件**

g++ **-o** hello hello.cpp 编译hello.cpp文件为可执行文件hello。

**生成动态库文件**

g++ hello.cpp **-fPIC -shared -o** libhello.so 编译hello.cpp文件为libhello.so动态库。

**设置头文件路径**

g++ **-Iinclude** –o hello hello.cpp 指定./include文件夹为头文件路径。

**设置库文件路径，并指定库**

g++ **-Llib –ltest** –o hello hello.cpp 指定./lib文件夹为库文件路径，并链接libtest.so动态库。

### 3.2 Makefile文件

**Makefile内容介绍**

当项目庞大时，每次输入数条g++命令非常麻烦，Makefile像一个脚本一样，将这些g++命令记录在文本中。Makefile定义了一系列的规则来指定，哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作。

简单的Makefile文件如下

all:main.cpp

g++ -o test main.cpp -Iinclude

.PHONY: clean

rm test

install:

cp test /usr/local/bin/

uninstall:

rm /usr/local/bin/test

其中第一行中all是目标，main.cpp是依赖，第二行是命令，命令开头一定要使用tab。

**Makefile使用**

在Makefile同目录下输入make命令进行编译。

输入make clean（需要存在clean规则）进行清理

输入make install（需要存在install规则）安装程序。

输入make uninstall（需要存在uninstall规则）卸载程序。

### 3.3 CMake使用

尽管Makefile把工作简单化了，但Makefile文件的编写却仍旧很繁琐，它有着各种各样的潜在的规则和变量。使用CMake可以生成Makefile文件，使得编译又简单了。值得骄傲的是，CMake是一个跨平台的安装（编译）工具。

#PROJECT (test)

#SET 用来定义变量

#SET(SOURCES main.c)

#AUX\_SOURCE\_DIRECTORY 用来设置使用的源文件的路径，此处表示把"."设置给SOURCES变量，可以设置多个路径给SOURCES变量

AUX\_SOURCE\_DIRECTORY(. SOURCES)

AUX\_SOURCE\_DIRECTORY(./one SOURCES)

AUX\_SOURCE\_DIRECTORY(./two SOURCES)

#MESSAGE用来输出一些信息，PROJECT\_BINARY\_DIR\PROJECT\_SOURCE\_DIR都是cmake的定义好的变量，指向工程目录

MESSAGE(STATUS "This is BINARY dir " ${PROJECT\_BINARY\_DIR})

MESSAGE(STATUS "This is SOURCE dir "${PROJECT\_SOURCE\_DIR})

#INCLUDE\_DIRECTORIES 用来设置头文件的搜索路径，即g++ -I

INCLUDE\_DIRECTORIES(.)

#LINK\_DIRECTORIES 用来设置库的非标准库引用路径，即g++ -L

LINK\_DIRECTORIES(.)

#ADD\_DEFINITIONS设置编译参数

ADD\_DEFINITIONS("-Wall -g")

#这里设置输出路径

#SET(EXECUTABLE\_OUTPUT\_PATH ${PROJECT\_SOURCE\_DIR}/bin)

SET(LIBRARY\_OUTPUT\_PATH ${PROJECT\_SOURCE\_DIR}/lib)

#生成lib库，STATIC表示生成静态库，SHAERED表示生成动态库

ADD\_LIBRARY(test SHARED ${SOURCES})

#生成可执行文件

#ADD\_EXECUTABLE(test ${SOURCES})

#TARGET\_LINK\_LIBRARIES 用来链接定制库，即g++ -l

#TARGET\_LINK\_LIBRARIES(test pthread)

cmake 使用

mkdir build建立单独文件夹build

cd build 进入build文件夹

cmake ../CMakeLists.txt 生成Makefile文件

make 编译代码

## 4 调试– gdb

在g++中使用参数-g可以在生成目标时加载调试信息。如g++ -o hello hello.cpp –g。

使用gdb hello对hello程序进行调试。

**运行**

**r** 或者 **run**

**继续**

**c** 或者 **continue**

**查看代码**

**l** 或者 **list**

**设置断点**

**b main** 设置断点到main函数

**break main** 设置断点到main函数

**b main.cpp:3** 设置断点到main.cpp文件中的第三行

**查看断点**

**info b**

**info break**

**删除断点**

**d** 删除所有断点

**delete** 删除所有断点

**d 2** 删除Num为2的断点

**逐语句下一步**

**s** 或者 **step**

**逐过程下一步**

**n** 或者 **next**

**跳出**

**finish**

**打印变量**

**p 变量名**

注：对于结构体使用p 变量名可能显示不够友好，可以输入set print pretty on设置友好显示

设置变量值

**p i=9** 设置int类型变量i的值为9

**调用函数**

**call printf("%d",i)** 调用printf函数

**调试core文件**

gdb hello core.1234 调试与hello程序对应的core.1234文件

输入**bt**命令，查看堆栈信息

**退出**

**q** 或者**quit**

## 5 Linux常用命令

一个命令后面可能会跟着多个选项，命令和选项之间以空格相隔。

**切换目录** **cd path**

eg: cd /etc

**查看目录结构** **ls –la** 或者**ll**

**更改文件权限** **chmod a**或者**u**或者**o+w**或者**r**或者**x 文件**或者**文件夹**

eg: chmod a+x ~/text

**更改文件用户归属** **chown 用户名:用户组 文件**或者**文件夹**

chown user:user ~/text

**创建文件夹** **mkdir 文件夹**

eg: mkdir myDir

**删除文件** **rm 文件**

eg: rm text

**删除文件夹** **rm 文件夹 –r**

eg: rm myDir –r

**移动文件或文件夹 mv 文件**或**文件夹 新路径**

eg: mv text ~/

**拷贝文件** **cp 文件 新文件**

eg: cp text newText

**拷贝文件夹** **cp 文件夹 新文件夹 –r**

eg: cp myDir ~/ -r

**查找文件** **find 路径 –name ”文件”**

eg: find / -name “\*abc.so”

**查看进程** **ps –ef**

**查看手册** **man 命令**或者**库函数**

eg: man ls