Module02 - Linux 开发环境

Linux 开发环境 - 课程目标

通过单元的学习，我们将熟悉以下 6 部分内容：

- 使用 vi/vim 文本编辑器
- 运用 GNU C/C++ 编译器： gcc/g++ 编译器编译 C/C++ 代码，生成可执行文件或库文件
- 用 make 来构建 (build)C/C++ 项目
- 使用 GNU 调试器： gdb 来调试 C/C++ 程序或库
- 熟悉版本控制工具 CVS 的常用操作
- 安装、配置并使用一个 IDE： Eclipse CDT

Module02-01
Linux 开发环境 : vi 编辑器

Linux 开发环境 - vim 编辑器

- vim
- 使用 gcc/g++
- make 和 makefile
- 使用 gdb
- CVS
- Eclipse CDT

vi 编辑器

在本单元中，我们将了解 vi 编辑器的以下几个内容：

- 进入、退出 vi ，保存文件
- 在 vi 中移动光标
- 编辑文本，如插入、修改、删除、复制、粘贴等
- 搜索并替换文本
- 简单的代码补全
- vi 编辑器与系统 Shell 的交互
- 一些常用的设置

vi 编辑器 - 简介

- 简介
  - 在很多 Linux 发行版中 vi 命令是指向 vim 命令的连接
  - vi 的四种模式：
    - 正常模式 (Normal)：

      进入 vi 后的默认状态是正常模式
    - 命令模式 (Command)：

      一般是执行 ex 命令，在正常模式下输入 :、 /、 ？等字符就进入了命令模式
    - 插入模式 (Insert)：

      编辑命令可以进入插入模式，如在正常模式下输入 i, a, o, I, O,P 等字符进入插入模式
    - 可视模式 (Visual)：

      用于选择文本块，如在正常模式下输入 v ， 按字符选择；输入 V ，按行选择；输入 Ctrl-V ， 按块选择

- 简介（续）
  - 一些说明
    - 查看 vim 入门指南的命令： vimtutor
    - ESC 键：结束任何编辑动作，如插入 (i,a,o) 、 修改 (c) 、 覆盖 (R) 等操作，从插入模式进入正常模式
    - 很多 vi 命令前可以加数字，达到批量操作的效果，如：
      - dd 表示删除当前行，而 3dd(d3d) 表示删除当前行在内的 3 行
      - x 表示删除当前字符，而 12x 表示删除当前和随后的共 12 个字符
      - 2j 表示光标向下移动 2 个字符（ 2 行）
      - .....
    - 尽量使用合适的命令来编辑，而不是通过重复同样的操作达到目的

vi 编辑器 - 打开 vi

- 打开 vi

命令	功能
`vi file`	打开或新建文件 `file`
`vi file1 file2 ...`	打开或新建多个文件
`vi -R file`	以只读方式打开文件 `file`
`vi -r file`	修复最近异常关闭的文件 `file`
`view file`	以只读方式打开文件 `file`
`vi + file`	打开文件 `file`，将光标定位到文件最后一行
`vi +n file`	打开文件 `file`，将光标定位到文件的第 *n* 行
`vi +/pattern file`	打开文件 `file`，将光标定位到第一个匹配 *pattern* 的行

vi 编辑器 - 保存文件、退出 vi

- 保存文件、退出 vi

命令	功能
:wq, ZZ, :x	保存文件，退出 vi
:w, :w!	保存当前文件，！表示忽略写保护
:w newfile	将文件另存为 newfile
:w %.new	将文件另存为原文件名 +.new 后缀名的文件
:20,100w newfile	将当前文件的 20-100 行的内容存入新文件 newfile
:20,100w>>afile	将当前文件的 20-100 行的内容追加到文件 afile 末尾
:q, :q!	不保存退出，不保存强行退出
Q	退出 vi，进入 ex 模式
:vi	退出 ex 模式，进入 vi
:r afile	将文件 afile 的内容读入、追加到当前行下
:*n*r afile	将文件 afile 的内容读入、追加到第 *n* 行下
:e newfile	不离开 vi，编辑新文件 newfile
:n	编辑下一个（n-next）文件（针对多文档编辑的情况）

- 移动光标

命令	功能
字符	
h, j, k, l	光标分别向左、下、上、右移动一个字符，而： 5h 光标向左移动 5 个字符
文字	
w, W, b, B	光标向前、向后移动一个单词，而： 2w 光标向前移动 2 个单词
e, E	光标移到下个单词的词尾
), (	光标移到下句、上句的句首
}, {	光标移到下个、上个段落开始之处
]], [[	光标移到下个、上个章节开始之处
指定行号	
:*n*	光标移到第 *n* 行
*n*G	光标移到第 *n* 行
G	光标移到文件末行
[CTRL-G]	显示文件的总行数、当前行号、列号等信息

vi 编辑器 - 移动光标

- 移动光标（续）

命令	功能
通过搜索定位（命令 less 同样支持）	
/pattern， ?pattern	由前往后、从后往前查找 pattern
/， ?	由前往后、从后往前继续上一次查找（注意与上行的操作对应）
n， N	按上一次的顺序、按上一次相反的顺序继续上次查找
行	
[RETURN]	光标移到下行行首第一个非空白字符
0， $	光标移到当前行首、行尾
^	光标移到当前行第一个非空白字符
+， -	光标移到下一行、上一行行首第一个非空白字符
n|	光标移到当前行第 *n* 列
H	光标移到当前屏幕第一行
M	光标移到当前屏幕中间
L	光标移到当前屏幕最后一行

- 编辑

命令	功能
插入	
i, a	在光标之前、之后插入文字
I, A	在当前行的行首、行尾插入文字
o, O	在当前行之下、之上新建一行，插入文字
修改	
~	转换光标所在位置的字符的大小写
r	替代单个字符
R	替代（覆盖）文字
cw	修改单词，如： 3cw 修改 3 个单词
cc	修改当前行
C	修改光标所在位置的字符到行尾的内容
s	替代：删除当前字符，插入新的字符
S	替代：删除当前行，插入新内容

- 编辑（续）

命令	功能
删除、移动	
x, X	删除光标所在位置、光标之前的字符，而： 5x 删除 5 个字符
dw	删除光标所在位置的单词，而： 2dw 或 d2w 删除 2 个单词
dd	删除当前行，而： 5dd 或 d5d 删除 5 行
D	删除光标所在位置到行尾的所有内容
复制 (yank)	
yw	复制单词，而： 3yw 或 y3w 复制 3 个单词
yy	复制当前行，而： 2yy 或 y2y 复制 2 行
粘贴	
p （小写）	将删除、复制的内容插入到光标所在位置之后
P （大写）	将删除、复制的内容插入到光标所在位置之前

- 编辑（续）

命令	功能
其它命令	
.	重复上一次编辑命令
u, U	撤销上一次操作、恢复当前行
J	合并临近的 2 行（当前行和下一行）
ex 编辑命令	
:d	删除行，:d 删除当前行；:12d 删除第 12 行；:5,8d 删除 5-8 行
:m	移动行，:m12 将当前行移到 12 行下；:4m12 将第 4 行移到 12 行下；:5,8m16 将 5-8 行移到 16 行下
:co 或 :t	复制行，:co12 将当前行复制到 12 行下；:4co12 将第 4 行复制到 12 行下；:5,8co16 将 5-8 行复制到 16 行下
:.,$d	删除当前行到文件末尾所有内容
:30,60m0	将第 30 行至第 60 行移动到文件头部
:.,/*pattern*/co$	将当前行至下面第一个匹配 *pattern* 的行复制到文件末尾

vi 编辑器 - 样式匹配及替换

- 样式匹配及替换

命令	功能
当前行匹配	
:s/*pattern*/new-str/	将当前行中第一个匹配 *pattern* 的字符串替换为 new-str
:s/*pattern*/new-str/g (g：表示全部，下同)	将当前行中所有匹配 *pattern* 的字符串替换为 new-str
在指定范围中匹配	
:5,26s/*pattern*/new-str/g	将 5-26 行中所有匹配 *pattern* 的字符串替换为 new-str
全局范围中匹配	
:%s/*pattern*/new-str/g	将整个文件中所有匹配 *pattern* 的字符串替换为 new-str，%代表所有行(1,$)
:g/*pattern*/s//new-str/g	同上
:g/*pattern*/s/old-str/new-str/g	在整个文件中，将匹配 *pattern* 的行中的所有 old-str 替换成 new-str

- 样式匹配及替换（续）
  - 影响匹配和替换的 3 个选项：
    - g：表示匹配行中的所有匹配项，而非仅仅第一项，如：

      行 01： `int n = 9; int m = 0;`

      `:s/int/INT/` → 只将 int n 中的 int 替换为 INT

      `:s/int/INT/g` → 替换本行中所有 int 为 INT
    - c：每次替换前有确认提示，如针对行 01 内容：

      `:s/int/INT/gc` → 每次替换前都提示是否要替换
    - i：样式匹配过程中忽略大小写，如

      行 02： `int n = 9; INt m = 0;`

      `:s/int/INT/gi` → 替换本行中 int、INt 为 INT，忽略大小写
  - 上述 3 个选项可单独使用，也可组合使用（如 3 者皆用或两两组合）

vi 编辑器 - 代码补全

- 代码补全： CTRL-P
  - 说明：在编辑如 C/C++ 代码的时候，我们可以使用 vim 的代码补全功能，加快键入的速度：
  - 示例：

```
/* hello.c */
#include <stdio.h>

int main() {
    pri
}
```

在键入 pri 后，按 CTRL-P ，可以补全为 printf

- 打开相关源文件： gf
  - 说明：在正常模式下，将光标移到单词 stdio ，然后按 gf ，可以打开头文件 stdio.h ，通过 CTRL-O 切换回当前文档

vi 编辑器 - 与 Shell 交互

- 与 Shell 交互

命令	功能
:sh	切换到 shell
^D	在 shell 环境中按 ctrl-D 返回 vi
:! command	在 vi 中临时执行 shell 命令 command
:n,m! command	将 shell 命令 command 的输出替换 n-m 行的内容
:r !command	将 shell 命令 command 的输出追加到当前行下

vi 编辑器 - 常用的设置

- 常用的设置

命令	功能
:set nu	显示行号
:syntax on	开启语法高亮（编辑各类代码很有用）
:set autoindent	自动缩进
:set smartindent	智能选择对齐方式
:set shiftwidth=4	缩进 4 字符
:set tabstop=4	tab 键的宽度设为 4 字符
:set softtabstop=4	移动光标时 tab 键的字符数，不影响插入 tab 键的宽度
:set expandtab	将 tab 转换成指定数量的空格
:set showmatch	高亮显示匹配的括号

如果想让这些设置能在下一次打开 vi 中使用，我们可以将这些设置写入 ~/.vimrc 文件，写入文件时不要冒号（ : ）

vi 编辑器 - 总结

- 本单元涉及 vi 编辑器的基础部分，包括打开、保存、退出、编辑、搜索与替换等常规操作
- vi 编辑器中，有着非常庞大的 ex 命令集，可以完成各式各样的复杂操作，如果大家有兴趣，可以通过有关书籍作进一步的了解
- 结合 ctags 等工具，可以将 vi 配置成一个完整、高效的集成开发环境 (IDE)，大家可以查看相关的资料，尝试配置
- 最后，作为 UNIX 系统中默认一定会存在的一个编辑器，我们不论作为 UNIX 软件开发人员，还是系统管理员等，都有必要熟练掌握它