

河南工业大学 Linux 基础与应用 实验报告

专业班级: 软件 1305 班 学号: 201316920311 姓名: 田劲锋 指导老师: 赵晨阳 评分: _____

实验题目: Linux 常用工具的使用 (2): 文本编辑器的使用

实验目的: (1) 理解文本编辑器 vi 的工作模式; (2) 掌握文本编辑器的使用方法

实验内容:

- (1) 阐述 vi 编辑器的 3 种工作模式, 以及如何实现工作模式的互相转换?
- (2) 在文本编辑器 vi 中, 实现下列功能, 列举出一个例子即可:
 - 1) 添加单个字符、多个字符
 - 2) 删除单个字符、删除整行文本
 - 3) 文本的替换
 - 4) 文本的复制和粘贴
 - 5) 文本的剪切和粘贴
 - 6) 全文关键字查找
 - 7) 全文字符串替换
 - 8) 保存、另存为、退出
 - 9) 同时打开两个文件, 实现两个窗口的切换
 - 10) 区域复制
 - 11) 与 shell 交互

实验步骤:

这里我使用运行在 Mac OS X 上终端下的 Vim。

- (1) Vim 常用三个模式: 命令模式、插入模式、可视模式。

在命令模式下, 用 `h j k l` 左下上右移动光标, `w e b` 移动到单词首尾, `0 ^ $` 移动到行首尾, `gg G` 移动到文档首尾。

`yy` 复制行, `yw` 复制单词, `p P` 粘贴, `dd dw D x` 剪切。命令前加数字可以重复。

进入插入模式用 `i`, 或者 `a` 从光标后插入, `o O` 另起新行, `I A` 插入到行首尾。

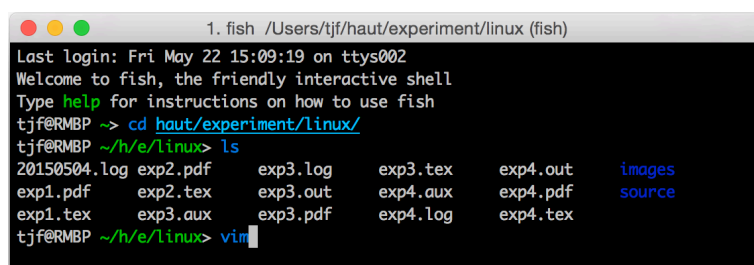
用 `Esc` 退出插入模式。

可视模式不常用, 按 `v` 进入, 可以选择文本。

冒号指令: `w` 保存, `:e` 编辑, `:q` 退出, `:x` 保存并退出。

`/pattern` 查找, `:%s/find/replace/g` 替换。

- (2) 这里我以编辑本次实验报告为示例, 来完成本次实验。



```
1. fish /Users/tjf/haute/experiment/linux (fish)
Last login: Fri May 22 15:09:19 on ttys002
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
tjferMBP ~-> cd haute/experiment/linux/
tjferMBP ~/h/e/linux> ls
20150504.log exp2.pdf exp3.log exp3.tex exp4.out images
exp1.pdf exp2.tex exp3.out exp4.aux exp4.pdf source
exp1.tex exp3.aux exp3.pdf exp4.log exp4.tex
tjferMBP ~/h/e/linux> vim
```

图 1: 启动 Vim

如图1, 首先在终端中键入 `vim` 以启动 Vim。

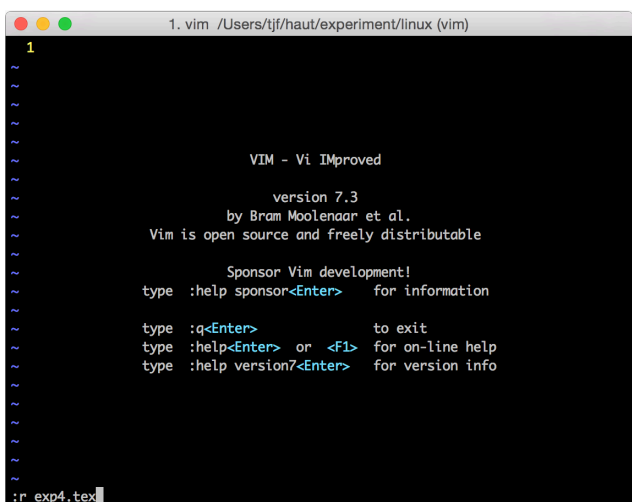


图 2: 读入实验四

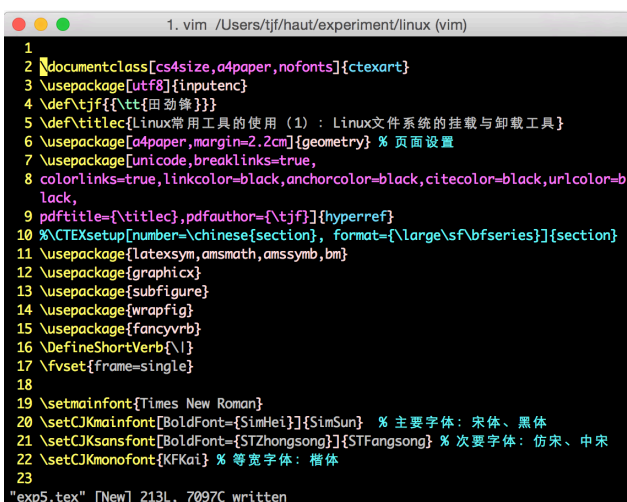


图 3: 保存为实验五

我要编写的是实验五的实验报告，这里直接读入实验四的内容进行修改。如图2，将实验四的 \LaTeX 源文件读入进来。如图3，键入:w exp5.tex，将当前编辑的文稿另存为实验五。



图 4: 从外部粘贴

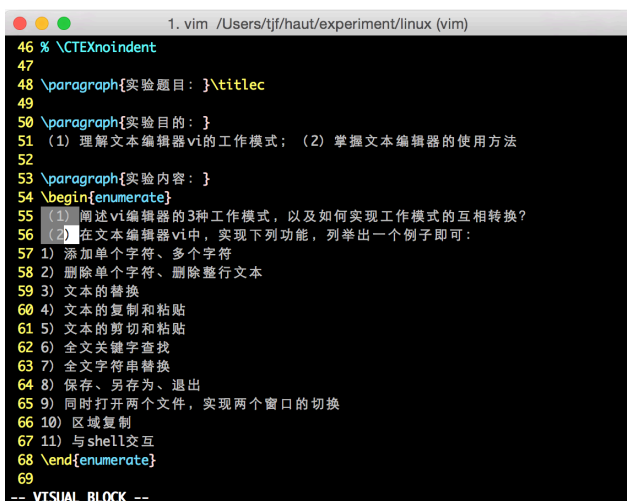


图 5: 自动缩进

接着将光标移动到待修改的地方，这里是实验目的和实验内容，用dd删除这里原有的内容后，按i进入插入模式，从老师给出的Word文档中复制这段文字，粘贴进来。如图4，Vim将直接粘贴进来的文本做了不好的缩进。在命令模式下将光标移动到段首，输入13==将55–67行的文本自动缩进，如图5。

光标定位到58行，按下CTRL-v进入可视块模式，如图6，按11j向下选择11行，按3l向右选择3列。这里我要将这些文本编号改写成 \LaTeX 的格式，键入I进入插入模式，输入\item 之后，连续按两下Esc键，Vim将自动填充到选择的文本块中，如图7，实现了批量插入的功能。

接下来将实验四的实验内容全部删除，按G跳转到文件尾，查看到实验内容最后一行的行号为215，然后跳转到实验内容第一行（这里是第84行），键入d:216从当前光标处删除到第216行，如图8，中间的内容全部被删除。

使用分割窗口的功能，键入:vsplit ../os/exp1.tex打开操作系统实验一的源文件，如图9。打开后如图10是竖屏分割的，光标移动到相应位置，按15yy将操作系统实验中关于

```
1. vim /Users/tjf/haut/experiment/linux (vim)
47
48 \paragraph{实验题目: }\titlec
49
50 \paragraph{实验目的: }
51 (1) 理解文本编辑器vi的工作模式; (2) 掌握文本编辑器的使用方法
52
53 \paragraph{实验内容: }
54 \begin{enumerate}
55 \item 阐述vi编辑器的3种工作模式, 以及如何实现工作模式的互相转换?
56 \item 在文本编辑器vi中, 实现下列功能, 列举出一个例子即可:
57 \begin{enumerate}
58 1) 添加单个字符、多个字符
59 2) 删除单个字符、删除整行文本
60 3) 文本的替换
61 4) 文本的复制和粘贴
62 5) 文本的剪切和粘贴
63 6) 全文关键字查找
64 7) 全文字符串替换
65 8) 保存、另存为、退出
66 9) 同时打开两个文件, 实现两个窗口的切换
67 10) 区域复制
68 11) 与shell交互
69 \end{enumerate}
70 \end{enumerate}
-- VISUAL BLOCK --
```

图 6: 选择文本块

```
1. vim /Users/tjf/haut/experiment/linux (vim)
47
48 \paragraph{实验题目: }\titlec
49
50 \paragraph{实验目的: }
51 (1) 理解文本编辑器vi的工作模式; (2) 掌握文本编辑器的使用方法
52
53 \paragraph{实验内容: }
54 \begin{enumerate}
55 \item 阐述vi编辑器的3种工作模式, 以及如何实现工作模式的互相转换?
56 \item 在文本编辑器vi中, 实现下列功能, 列举出一个例子即可:
57 \begin{enumerate}
58 \item 添加单个字符、多个字符
59 \item 删除单个字符、删除整行文本
60 \item 文本的替换
61 \item 文本的复制和粘贴
62 \item 文本的剪切和粘贴
63 \item 全文关键字查找
64 \item 全文字符串替换
65 \item 保存、另存为、退出
66 \item 同时打开两个文件, 实现两个窗口的切换
67 \item 区域复制
68 \item 与shell交互
69 \end{enumerate}
70 \end{enumerate}
```

图 7: 批量插入

```
1. vim /Users/tjf/haut/experiment/linux (vim)
68 \item 与shell交互
69 \end{enumerate}
70 \end{enumerate}
71
72 \paragraph{实验步骤: }\quad
73
74 \newcommand{\image}[3][height=10cm]{\begin{minipage}[t]{0.5\textwidth}
75 \centering
76 \includegraphics[#1]{images/exp4/#2.png}
77 \caption{#3}
78 \label{fig:#3}
79 \end{minipage}
80 }
81
82 \begin{enumerate}
83
84
85 \end{enumerate}
86
87 \paragraph{实验体会: }\quad
88
89
90 \end{document}
~
131 fewer lines
```

图 8: 删除到指定行

```
1. vim /Users/tjf/haut/experiment/linux (vim)
68 \item 与shell交互
69 \end{enumerate}
70 \end{enumerate}
71
72 \paragraph{实验步骤: }\quad
73
74 \newcommand{\image}[3][height=10cm]{\begin{minipage}[t]{0.5\textwidth}
75 \centering
76 \includegraphics[#1]{images/exp5/#2.png}
77 \caption{#3}
78 \label{fig:#3}
79 \end{minipage}
80 }
81
82 这里我使用运行在 Mac OS X 上终端下的 Vim.
83
84 \begin{enumerate}
85
86 \item
87
88 \end{enumerate}
89
90 \paragraph{实验体会: }\quad
91
92 \end{document}
:vsplit ../os/exp1.tex
```

图 9: 分割窗口

```
1. vim /Users/tjf/haut/experiment/linux (vim)
69 Vim常用三个模式: 命令模式、插入模式、
、可视模式。
70
71 在命令模式下, 用lh j k l左右上下右移>
动光标, lw e b l移动到单词末尾, \verb
+0 ^ $+移动到行首尾, lgg G l移动到文>
档首尾。
72
73 |yy|复制行, |yw|复制单词, |p|粘贴>
, |dd dw D x|剪切。命令前加数字可以>
重复。
74
75 进入插入模式用|il|, 或者|al|从光标后插>
入, |o| |O|另起新行, |I| |A|插入到行首尾
。
76
77 用|Esc|退出插入模式。
78
79 可视模式不常用, 按|v|进入, 可以选择>
文本。
80
81 冒号指令|:w|保存, |:e|编辑, |:q|退出>
, |:x|保存并退出。
..os/exp1.tex exp5.tex [+]
```

图 10: 复制文本

```
1. vim /Users/tjf/haut/experiment/linux (vim)
exp5.tex ../o/e/parent_child.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 int main(int argc, char *argv[])
6 {
7     pid_t pid1, pid2;
8
9     if ((pid1 = fork()) < 0) {
10         fprintf(stderr, "创建子进程1失败\n");
11         exit(1);
12     } else if (pid1 == 0) { /* 子进程1 */
13         printf("This is child1 (pid1=%ld) process: B\n",
14             (long) getpid());
15     } else { /* 父进程 */
16         printf("Parent Process: A\n");
17
18         if ((pid2 = fork()) < 0) {
19             fprintf(stderr, "创建子进程2失败\n");
20             exit(1);
21         } else if (pid2 == 0) { /* 子进程2 */
22             printf("This is child2 (pid2=%ld) process: C\n",
23                 (long) getpid());
24         }
25     }
26 }
```

图 11: 调用Shell

Vim描述的15行文字复制下来，按两下CTRL-w切换窗格，光标跳转到Linux实验中相应位置，按p粘贴。

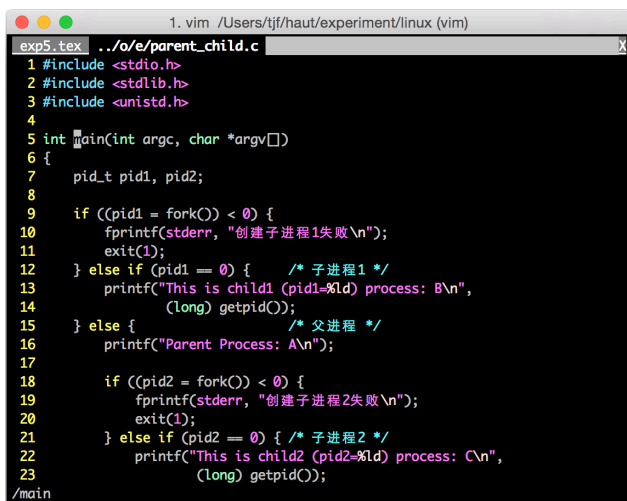
使用:tabnew可以打开一个新的标签页，这里我打开了操作系统实验一的一个C程序源文件。如图11，键入

```
!cc ../os/exp2/parent_child.c
```

来调用Shell解释程序，执行clang编译器，将该源文件编译成可执行文件a.out，Vim会跳转回终端显示相应信息。键入

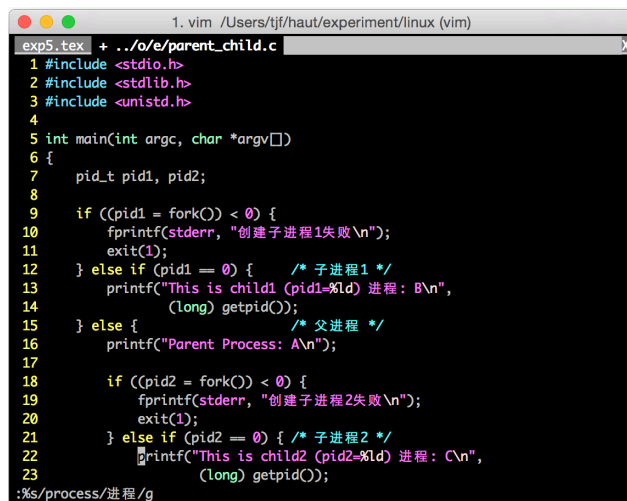
```
!./a.out
```

调用Shell来执行这个编译后的程序。



```
1 vim /Users/tjf/haot/experiment/linux (vim)
exp5.tex  ../o/e/parent_child.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 int main(int argc, char *argv[])
6 {
7     pid_t pid1, pid2;
8
9     if ((pid1 = fork()) < 0) {
10         fprintf(stderr, "创建子进程1失败\n");
11         exit(1);
12     } else if (pid1 == 0) { /* 子进程1 */
13         printf("This is child1 (pid1=%ld) process: B\n",
14             (long) getpid());
15     } else { /* 父进程 */
16         printf("Parent Process: A\n");
17
18         if ((pid2 = fork()) < 0) {
19             fprintf(stderr, "创建子进程2失败\n");
20             exit(1);
21         } else if (pid2 == 0) { /* 子进程2 */
22             printf("This is child2 (pid2=%ld) process: C\n",
23                 (long) getpid());
24         }
25     }
26 }
/main
```

图 12: 查找文本



```
1 vim /Users/tjf/haot/experiment/linux (vim)
exp5.tex  + ../o/e/parent_child.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 int main(int argc, char *argv[])
6 {
7     pid_t pid1, pid2;
8
9     if ((pid1 = fork()) < 0) {
10         fprintf(stderr, "创建子进程1失败\n");
11         exit(1);
12     } else if (pid1 == 0) { /* 子进程1 */
13         printf("This is child1 (pid1=%ld) 进程: B\n",
14             (long) getpid());
15     } else { /* 父进程 */
16         printf("Parent Process: A\n");
17
18         if ((pid2 = fork()) < 0) {
19             fprintf(stderr, "创建子进程2失败\n");
20             exit(1);
21         } else if (pid2 == 0) { /* 子进程2 */
22             printf("This is child2 (pid2=%ld) 进程: C\n",
23                 (long) getpid());
24         }
25     }
26 }
:%s/process/进程/g
```

图 13: 替换文本

如图12，键入

```
/main
```

来查找“main”出现的位置。

如图13，键入

```
:%s/process/进程/g
```

将文件中所有“process”替换成“进程”。

实验体会：

这次实验是关于vi的使用，实际上vi有一些微妙的地方并不是很好用，所以一般都会使用增强版的Vim来代替vi。我这里的Vim是经过配置，有代码高亮和行号显示的。Vim的学习曲线是比较长的，虽然我用Vim也有很多年了，但也并不完全熟悉其强大功能。当然，有什么不会的看看帮助文档和用户手册，问题也就好解决了。