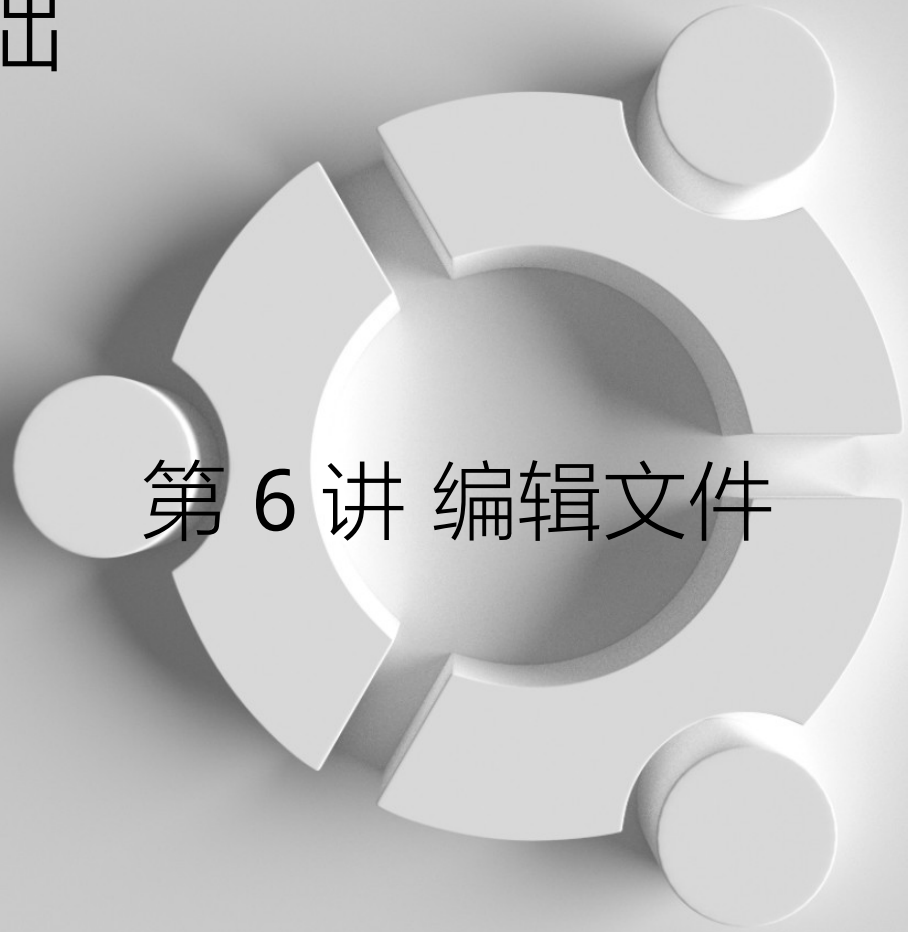
Linux 基础



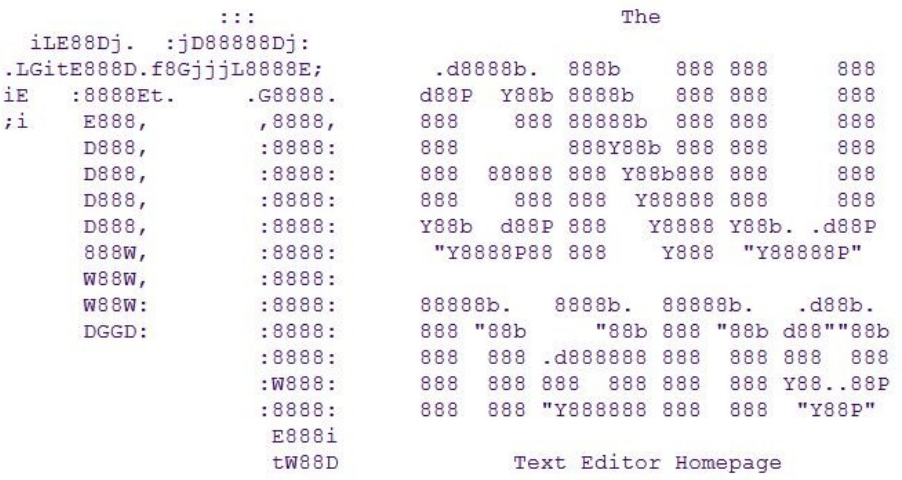
第 6 讲 编辑文件

目标

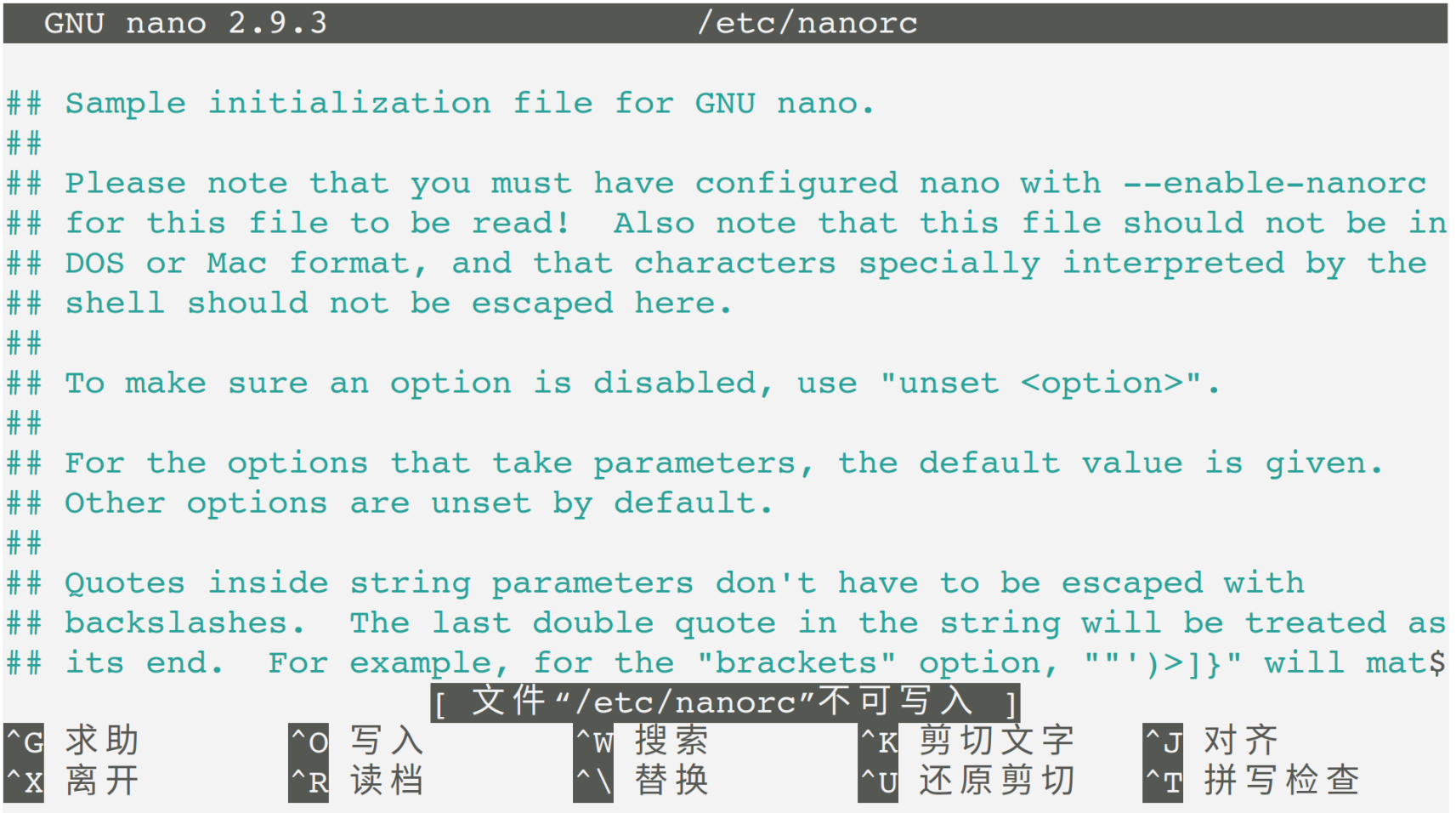
- 能够熟练使用 nano 编辑文本。
- vim 短时间内不容易上手，使用 nano 主要应对需要终端操作的场景。
- 对于桌面环境，可以使用 vscode 、 gedit 、 geany 等 GUI 工具。
- 对于 GUI 工具，自己可以尝试安装使用。

nano

- nano 是 Debian/Ubuntu 系列的发行版默认的文本编辑工具。
- nano 属于 GNU 软件。
- nano 是终端模式的简单易用的文本编辑器，但是也提供了代码高亮、字符匹配，指定跳转等一些高级编辑功能。



nano 启动界面


GNU nano 2.9.3 /etc/nanorc
Sample initialization file for GNU nano.
##
Please note that you must have configured nano with --enable-nanorc
for this file to be read! Also note that this file should not be in
DOS or Mac format, and that characters specially interpreted by the
shell should not be escaped here.
##
To make sure an option is disabled, use "unset <option>".
##
For the options that take parameters, the default value is given.
Other options are unset by default.
##
Quotes inside string parameters don't have to be escaped with
backslashes. The last double quote in the string will be treated as
its end. For example, for the "brackets" option, ""')>]}" will mat$
[文件“/etc/nanorc”不可写入]
^G 求助 ^O 写入 ^W 搜索 ^K 剪切文字 ^J 对齐
^X 离开 ^R 读档 ^\ 替换 ^U 还原剪切 ^T 拼写检查

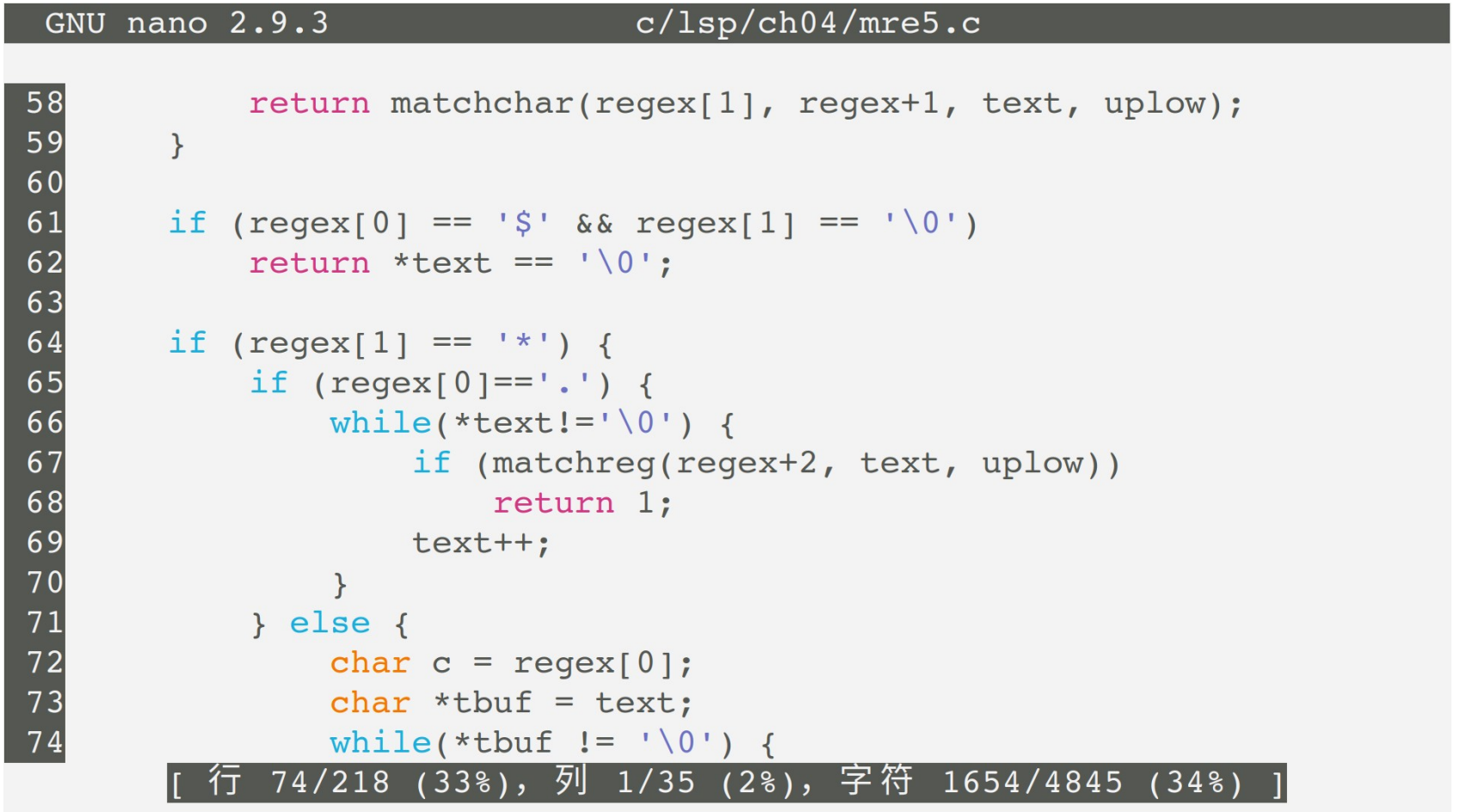
nano 基本使用

- 打开文件：`nano [ 文件名 ]`
- 保存文件：`Ctrl+S`
- 另存为：`Ctrl+O`，这时候底部会提示输入文件名，默认为当前文件名。
- 退出：`Ctrl+X`

nano 配置文件

- 配置文件位置： `/etc/nanorc`
- 配置文件选项一般使用 `set **`， `#` 开头表示注释，比如 `set linenumbers` 表示显示行号。
- 每个选项前使用注释说明了其作用。

nano 修改配置后截图



vim

- vi 是最古老的文本编辑器之一。vim 是其加强版。
- 几乎所有的 Linux 发行版都会内置 vi，但是 vim 一般需要安装。
- 尽管 vim 非常古老，但是因其强大的功能，灵活的扩展性，高效的编辑能力吸引了大批极客。

- 尽管 vim 有几十年的历史，但是其生命力依然强大。也因为其特点，导致很多开发者不会去使用它。
- 无论现代软件开发方式如何，我都像你推荐它。在任何时候，当你遇到 Linux/Unix 的一些极端情况，有了 vim 则可以从容应对。

vim 入门

- Ubuntu 系统上可能只提供了 vi，这是早期的版本，可以使用 apt 安装：

  ```
  sudo apt install vim
  ```

- 打开文件：vim [文件名]。
- 如果文件不存在则会在保存时创建。

vim 基本使用

- vim 的使用和其他编辑器不同， vim 使用了 3 种不同的模式，为了区分，给它们命名为
  - 命令模式
  - 输入模式（底部显示 --INSERT-- 或 -- 插入 --）
  - 底行模式（底部显示：）

vim 基本使用：开始编辑

- 启动 vim 默认是命令模式。这时候不能直接编辑文件。
- 输入 i,a,o 都可以切换到输入模式，此时底部显示：

  --INSERT-- 或 -- 插入 --
- i 是在当前位置输入， a 会把光标向右移动一个， o 是在下一行输入。

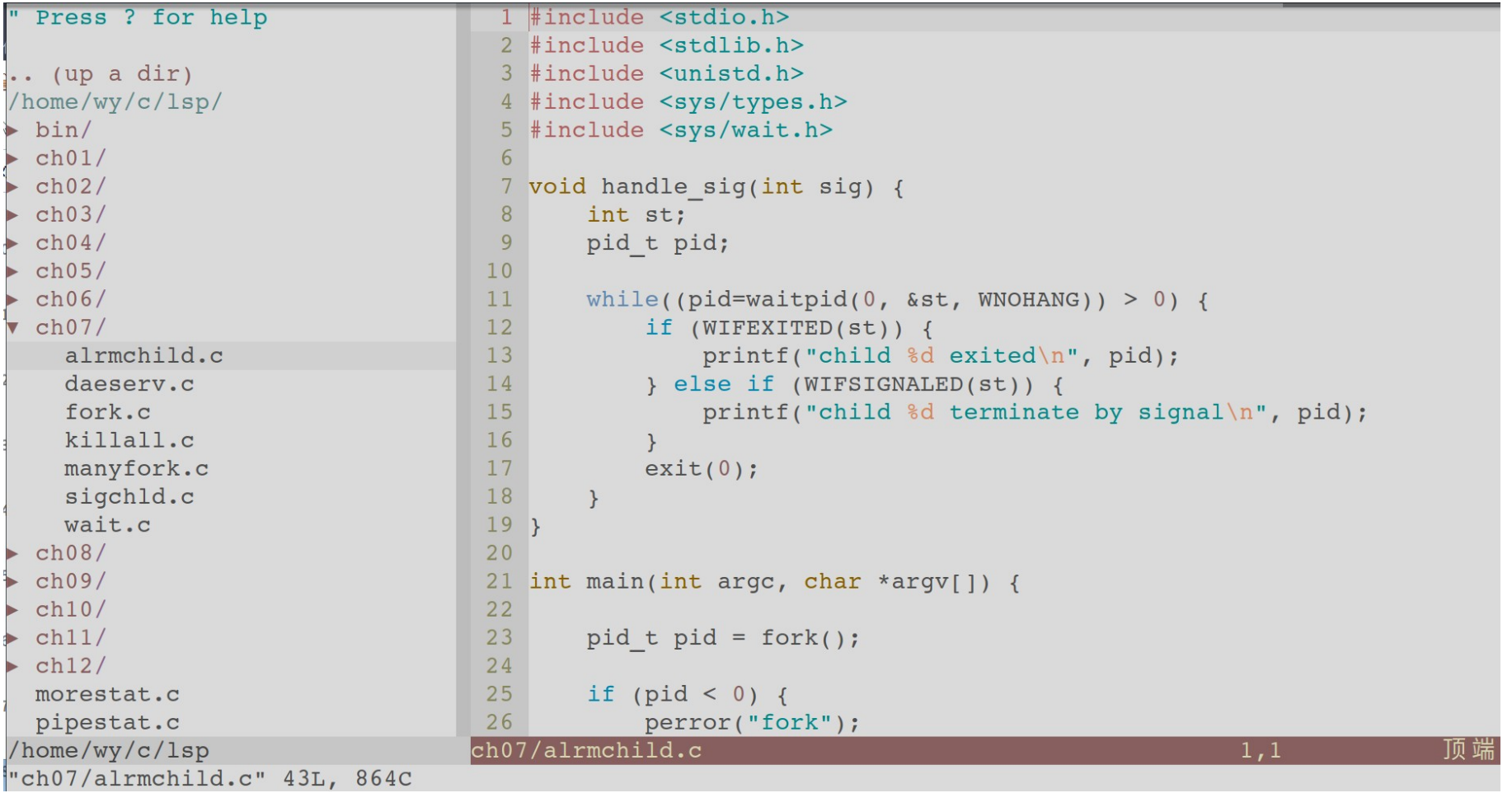
vim 基本使用：保存并退出

- 编辑完文件之后，在 vim 中保存退出需要以下操作：
  - 按 ESC 切换回 ‘命令模式’
  - 输入 ： 切换到 ‘底行模式’
  - 输入 w 写入文件，输入 q 退出，可以连用，输入 wq 保存并退出

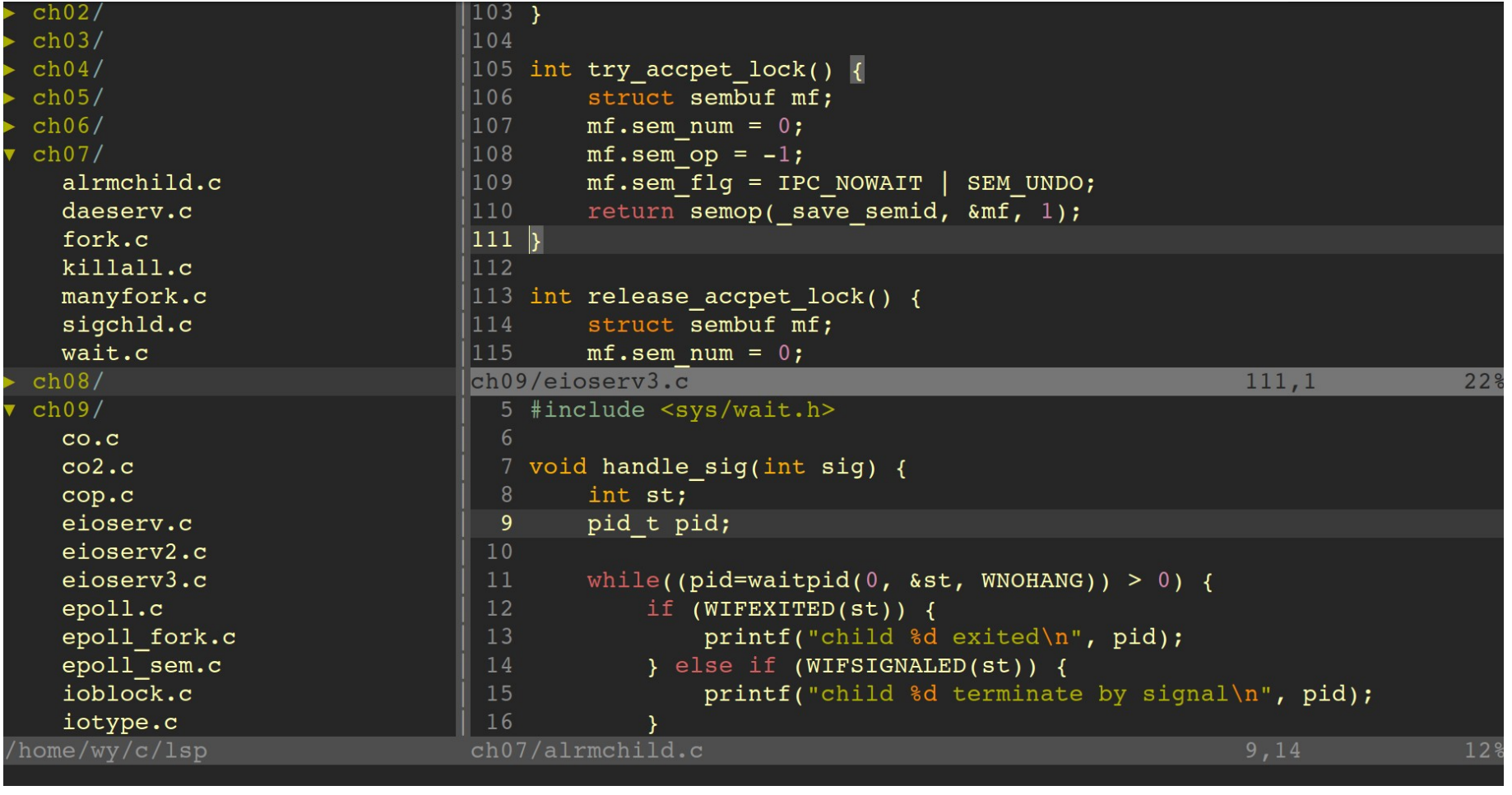
vim 基本使用：不保存退出

- 如果想要放弃本次修改，可以按照以下步骤：
  - 按 ESC 回到 ‘命令模式’
  - 输入 ：切换到 ‘底行模式’
  - 输入 q! 不保存退出

Vim 截图


" Press ? for help
.. (up a dir)
/home/wy/c/lsp/
▶ bin/
▶ ch01/
▶ ch02/
▶ ch03/
▶ ch04/
▶ ch05/
▶ ch06/
▼ ch07/
alrmchild.c
daeserv.c
fork.c
killall.c
manyfork.c
sigchld.c
wait.c
▶ ch08/
▶ ch09/
▶ ch10/
▶ ch11/
▶ ch12/
morestat.c
pipestat.c
/home/wy/c/lsp
"ch07/alrmchild.c" 43L, 864C
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
void handle_sig(int sig) {
int st;
pid_t pid;
while((pid=waitpid(0, &st, WNOHANG)) > 0) {
if (WIFEXITED(st)) {
printf("child %d exited\n", pid);
} else if (WIFSIGNALED(st)) {
printf("child %d terminate by signal\n", pid);
}
exit(0);
}
}
int main(int argc, char *argv[]) {
pid_t pid = fork();
if (pid < 0) {
perror("fork");
ch07/alrmchild.c
1,1
顶端

Vim 截图


▸ ch02/
▸ ch03/
▸ ch04/
▸ ch05/
▸ ch06/
▾ ch07/
alrmchild.c
daeserv.c
fork.c
killall.c
manyfork.c
sigchld.c
wait.c
▸ ch08/
▾ ch09/
co.c
co2.c
cop.c
eioserv.c
eioserv2.c
eioserv3.c
epoll.c
epoll_fork.c
epoll_sem.c
ioblock.c
iotype.c
103 }
104
105 int try_accpet_lock() {
106 struct sembuf mf;
107 mf.sem_num = 0;
108 mf.sem_op = -1;
109 mf.sem_flg = IPC_NOWAIT | SEM_UNDO;
110 return semop(_save_semid, &mf, 1);
111 }
112
113 int release_accpet_lock() {
114 struct sembuf mf;
115 mf.sem_num = 0;
ch09/eioserv3.c 111,1 22%
5 #include <sys/wait.h>
6
7 void handle_sig(int sig) {
8 int st;
9 pid_t pid;
10
11 while((pid=waitpid(0, &st, WNOHANG)) > 0) {
12 if (WIFEXITED(st)) {
13 printf("child %d exited\n", pid);
14 } else if (WIFSIGNALED(st)) {
15 printf("child %d terminate by signal\n", pid);
16 }
/home/wy/c/lsp ch07/alrmchild.c 9,14 12%

桌面环境复制粘贴内容

- 在 gnome 桌面环境的虚拟终端里，默认使用快捷键 Shift+Ctrl+C 复制选中内容，使用 Shift+Ctrl+V 粘贴内容。
- 这可以把内容复制粘贴到 nano/vim 中。
- 因为 nano/vim 这种终端模式的编辑器其复制的内容使用自己独立的暂存区，不和桌面环境在复制时的暂存区域冲突，所以要把网页或其他文件中的内容复制到 nano/vim 中，可以在复制后使用 Shift+Insert 键。

Shift+Insert 是虚拟终端软件的快捷键