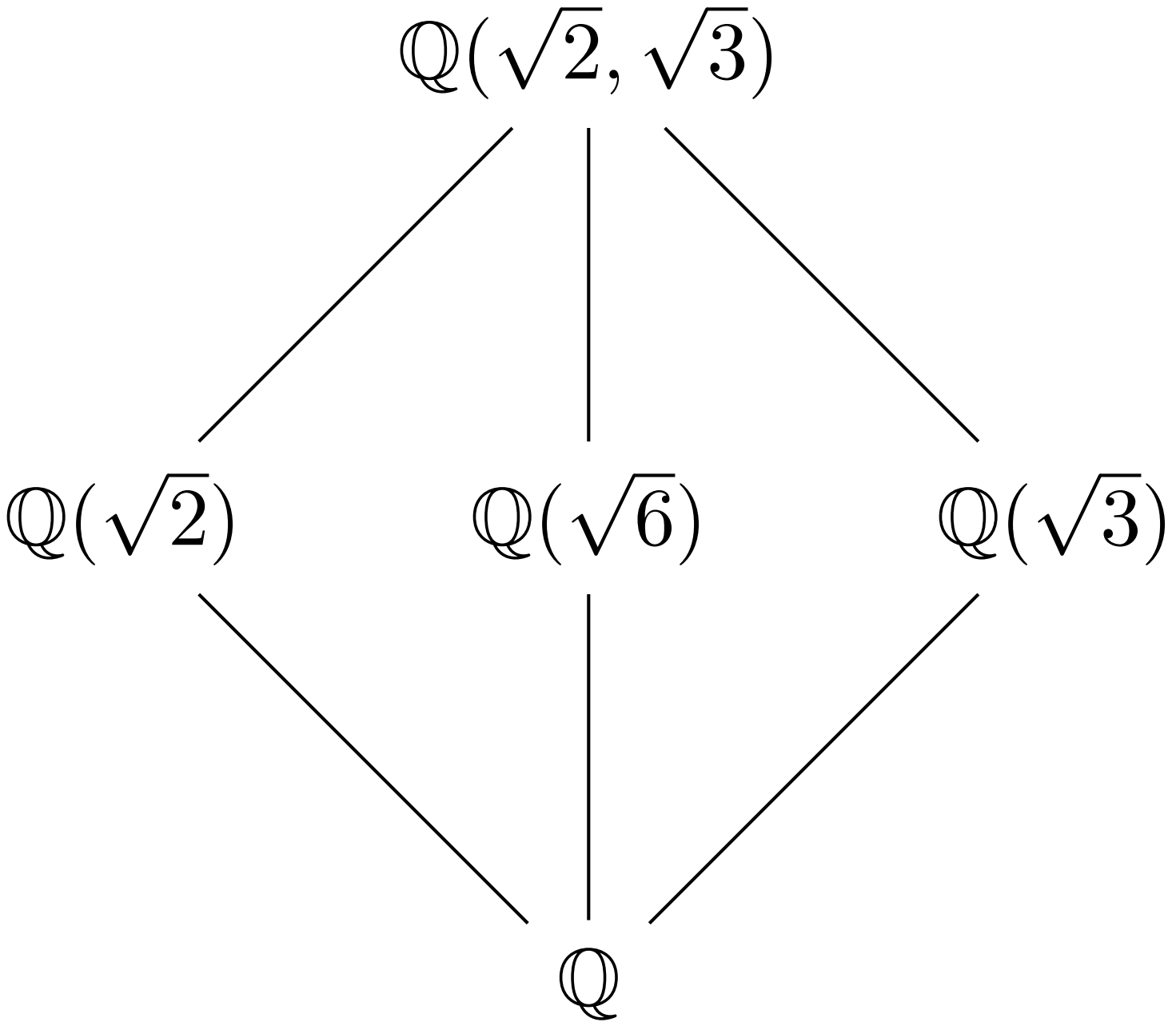

使用 C 语言构建终端文本编辑器

作者：左元

目录

第一章 设置

第 1 步: *kilo.c*

```c
int main() {
  return 0;
}
```

编写 Makefile 文件。

第 2 步: *Makefile*

```
kilo: kilo.c
	$(CC) kilo.c -o kilo -Wall -Wextra -pedantic -std=c99
```

Makefile 中一定要使用制表符. 命令中的参数:

- `$(CC)` 是一个 make 会展开的变量, 默认是 cc.
- `-Wall` 代表 “所有警告”, 并让编译器在看到程序中的代码时向你发出警告, 这些代码在技术上可能没有错误, 但被认为是 C 语言的错误或有问题的用法, 例如在初始化变量之前使用变量.
- `-Wextra` 和 `-pedantic` 会打开更多警告. 对于本教程中的每个步骤, 如果你的程序能够编译通过, 除了在某些情况下出现 “未使用的变量” 警告外, 它不应产生任何警告. 如果你收到任何其他警告, 请检查以确保你的代码与该步骤中的代码完全匹配.
- `-std=c99` 指定我们正在使用的 C 语言标准的确切版本, 即 C99. C99 允许我们在函数内的任何地方声明变量, 而 ANSI C 要求所有变量都在函数或块的顶部声明.

使用 make 命令来编译程序. 运行 `./kilo` . 然后使用命令 `echo $?` 查看程序的返回值.

第二章 进入原始模式

> 原始模式: raw mode

接下来读取用户的按键操作.

第 3 步: *kilo.c*

```c
#include <unistd.h>

int main() {
  char c;
  while (read(STDIN_FILENO, &c, 1) == 1);

  return 0;
}
```

`read()` 和 `STDIN_FILENO` 来自 <unistd.h> . `read()` 从标准输入中读取 1 个字节到变量 c 中, 在 while 循环中一直读取, 直到没有可以读取的字节. read() 返回读取的字节数, 并在到达文件末尾时返回 0.

当运行 `./kilo` 时, 终端会连接到标准输入, 因此键盘的输入会被读取到变量 c 中. 但是, 默认情况下终端以 **规范模式**[1] 启动. 在规范模式下, 键盘输入仅在用户按下 **回车键**时发送到我们的程序. 这对许多程序都很有用: 它允许用户输入一行文本, 这样可以使用 **退格键**来修复错误, 直到文本完全按照想要的方式输入, 最后按 **回车键**将其发送到程序. 但它不适用于具有更复杂用户界面的程序, 如文本编辑器. 我们希望在每个按键输入时都对其进行处理, 以便我们可以立即做出响应.

我们想要的是 **原始模式**. 不幸的是, 没有简单的开关可以将终端设置为原始模式. 原始模式是通过关闭终端中的许多标志位来实现的, 我们将在本章的过程中逐渐做到这一点.

要退出上述程序, 请按 Ctrl-D 以告知 read() 它已到达文件末尾. 或者我们始终可以按 Ctrl-C 以发出立即终止进程的信号.

[1] canonical mode