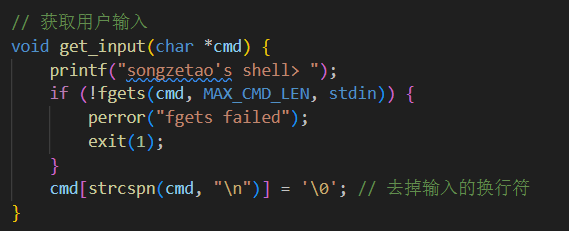
Operation System Project

1. **实现一个自己的shell**
   1. 获取用户输入

Shell 需要从用户获取输入，可以通过 fgets() 或 scanf() 函数获取输入的命令。通常 fgets() 比较常用，因为它能读取整行输入并处理换行符。



* 1. 解析用户输入

用户输入的命令一般是一个字符串，需要将命令和参数分开。我们可以使用 strtok() 函数将输入字符串分割成一个个单独的命令和参数。



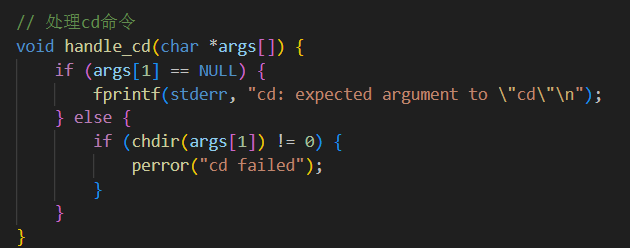
* 1. 执行命令

命令执行可以使用 fork() 来创建子进程，再通过 execvp() 在子进程中执行命令。父进程使用 wait() 等待子进程完成。



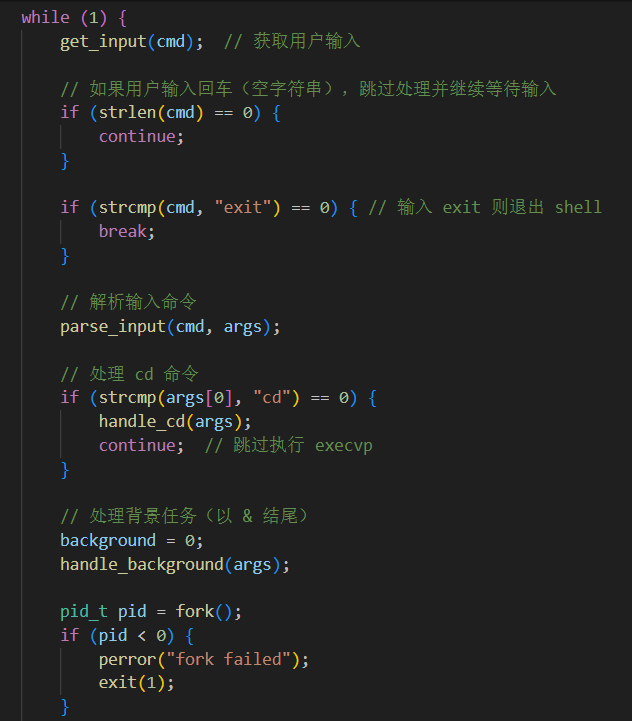
* 1. 处理内建命令

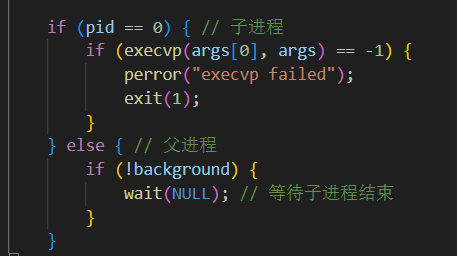
对于一些内建命令（如 cd），需要在 Shell 中单独处理，而不是通过 execvp() 调用外部命令。



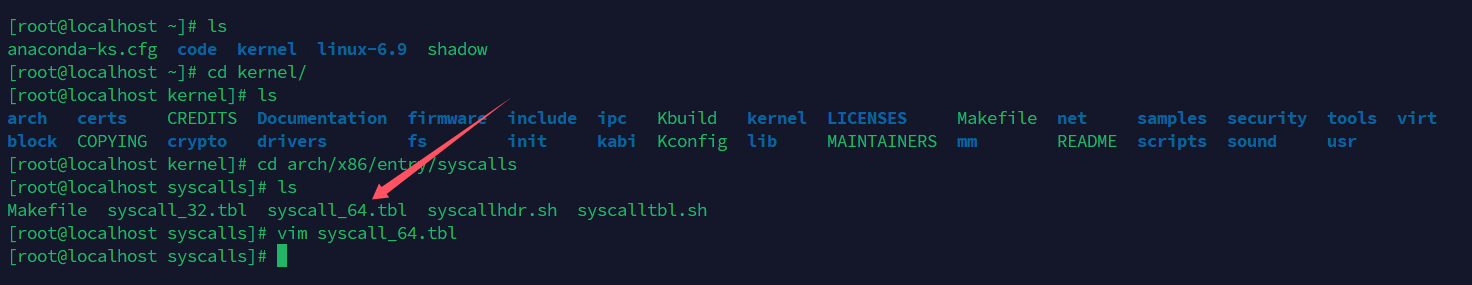
* 1. 主循环

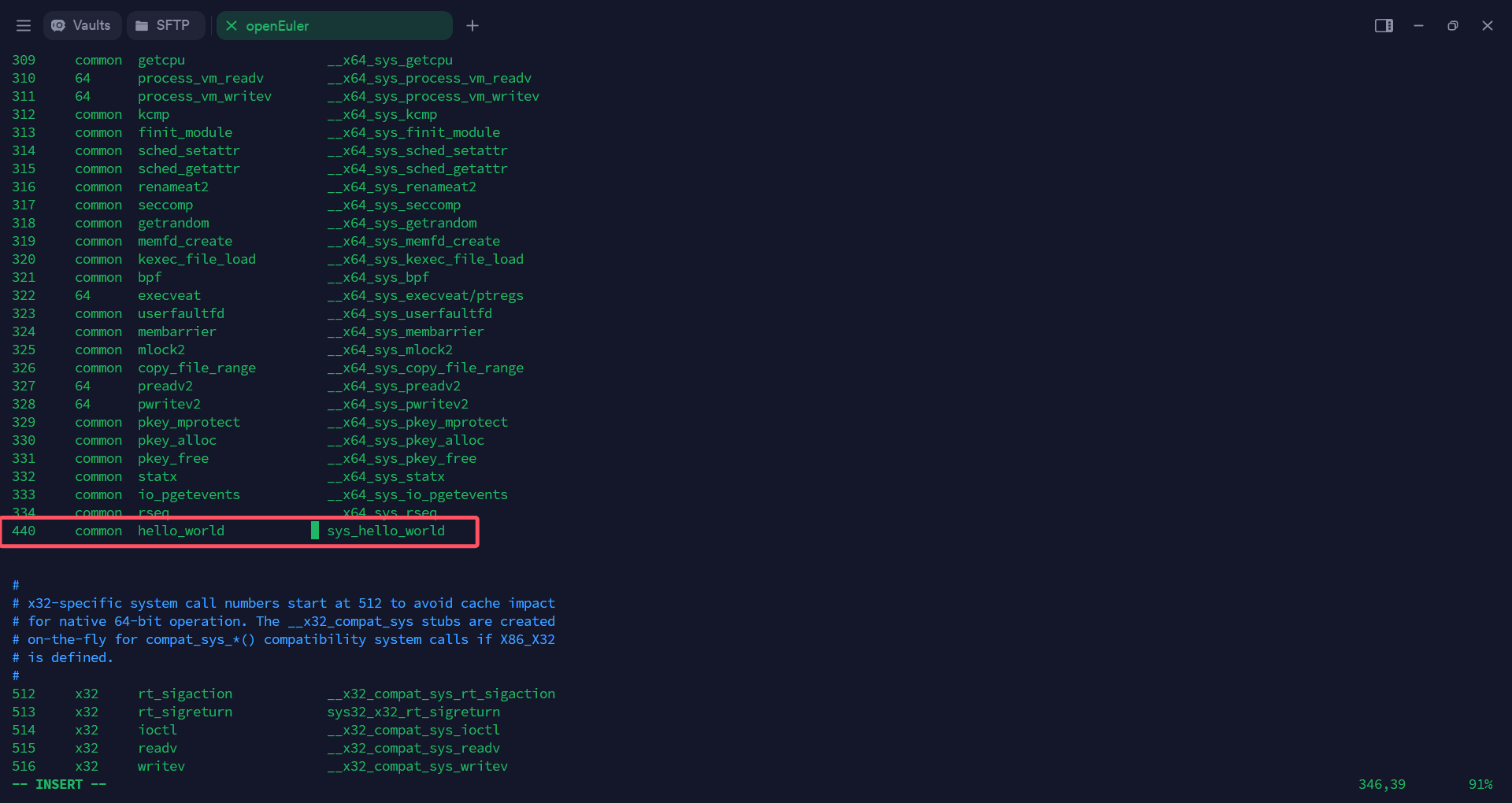
将所有部分组合起来，在主函数中实现一个主循环，不断获取用户输入、解析并执行命令。



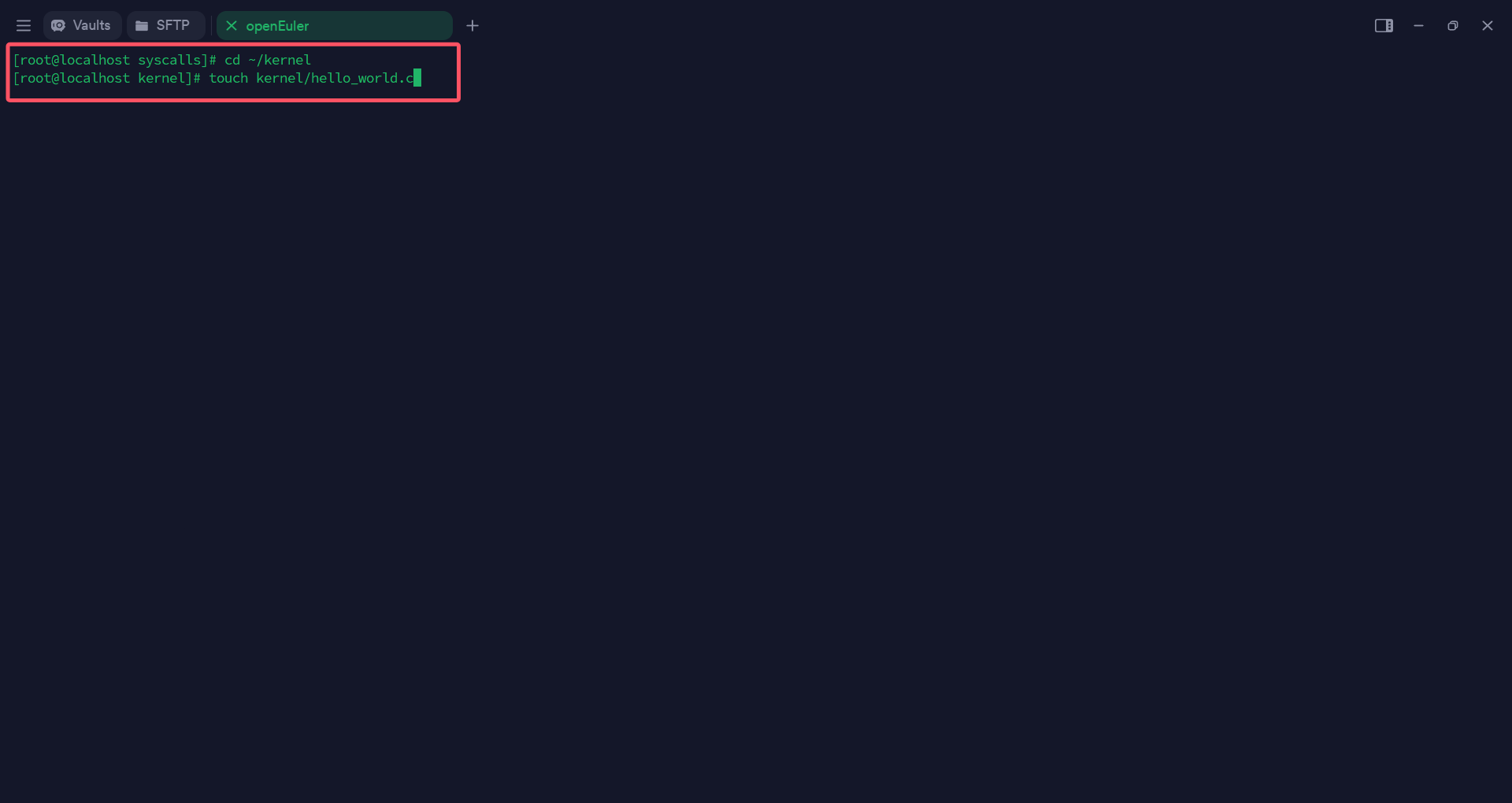


1. **向linux系统中添加一个系统调用，该系统调用向用户空间返回一个字符串“hello world！”**
   1. 从 openEuler 的 GitHub 上下载源代码。openEuler 是基于 Linux 的，因此内核源码和配置与 Linux 大致相同。官方 GitHub 地址：<https://gitee.com/openeuler/kernel>
   2. 定位到kernel/arch/x86/entry/syscalls文件夹，找到syscall\_64.tbl文件，在这个文件中添加一个新的系统调用



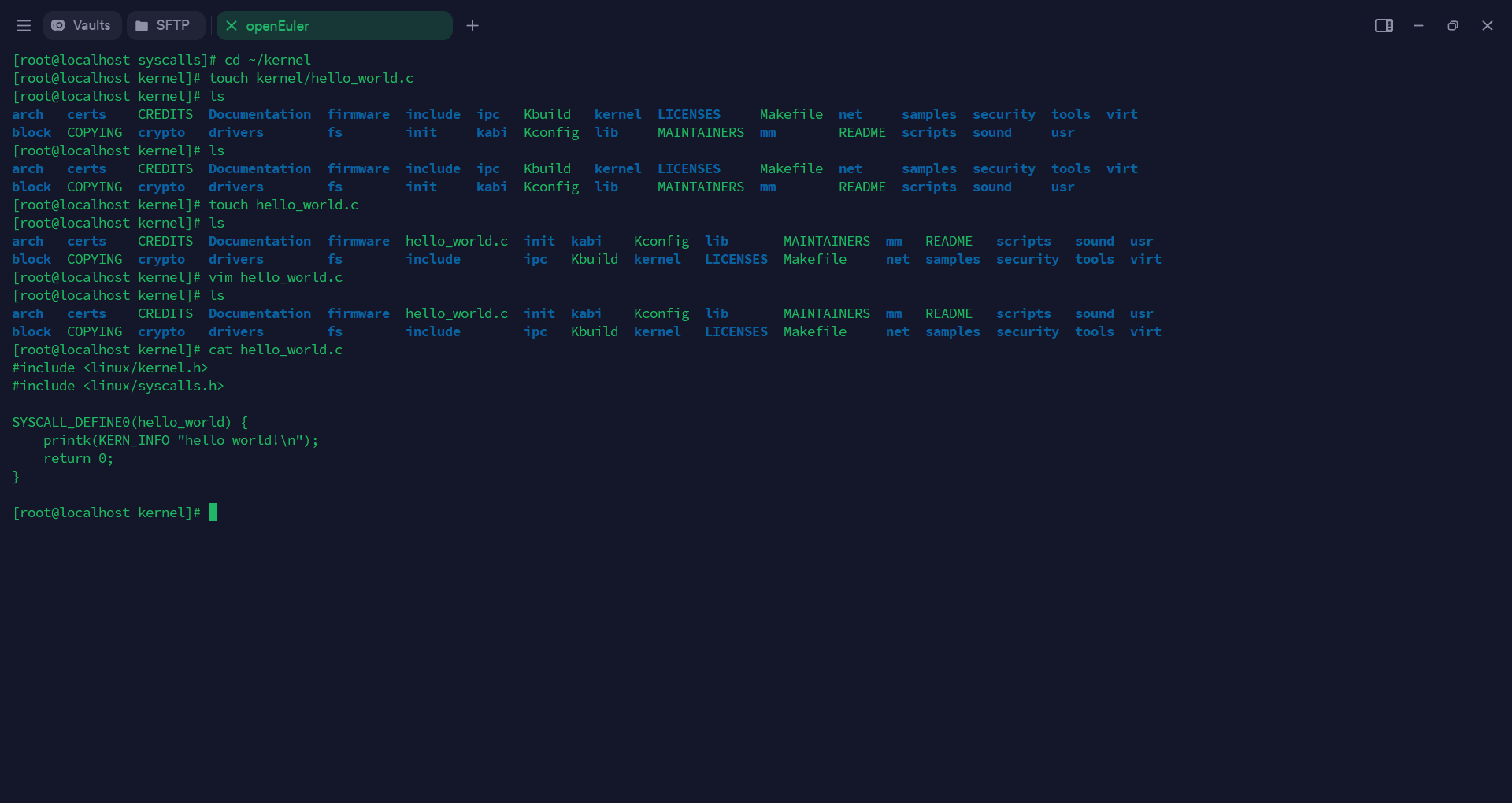


* 1. 接下来添加系统调用的实现，首先回到内核源码的kernel目录，然后创建一个新的C文件来实现刚刚添加的系统调用。

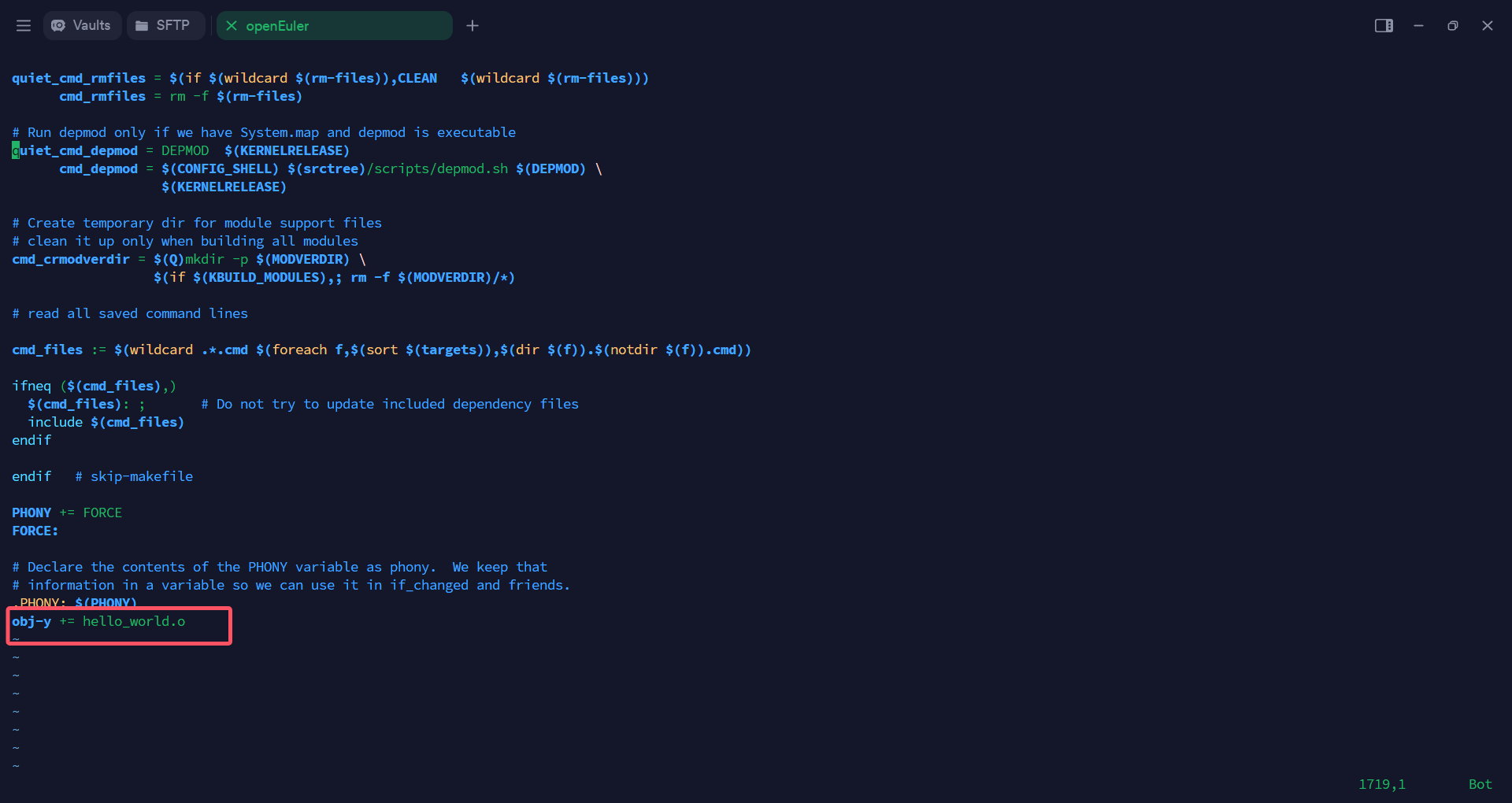


2.4 在新建的C文件中填写以下内容，其中SYSCALL\_DEFINE0宏定义了一个不带参数的系统调用，printk用于将日志打印到内核日志中





* 1. 更新内核的makefile，首先打开kernel目录中的Makefile，在文件末尾添加：obj-y += hello\_world.o



2.5 然后进行内核编译