Operating System CH3

Susie Glitter

2025 年 7 月 7 日

注：本次实验使用了 VMware 中的 ubuntu-16.04.6-desktop

1 任务一：实现 Shell 接口 osh>

1.1 一些准备（命令解析）

为更加方便地管理命令，我们尝试用面向对象方式编程（尽管这是 c 语言），我们使用`command`类储存一条指令，其中包含命令参数，输入输出文件，是否同步执行（$），以及由管道连接的下一条指令。接下来需要实现这个类的初始化、拷贝、清空、解析、执行等行为。

1.2 创建子进程并在子进程中执行命令

我们需要使用`fork()`创建一个子进程，在其中执行`comm_execute()`，在其中使用`execvp()`执行对应命令即可，而父进程需要根据命令是否含有 $ 决定是否等待子进程返回，这里使用的是`waitpid()`函数。exit 指令需要特别判断以退出主循环。

1.3 提供历史记录功能

为实现历史记录功能，我们使用两个`command`类的指针`currnet`和`history`来指向存有当前与历史命令的位置，每个循环开始时将当前的命令改为储存到历史，若有输入!!，则可以将历史拷贝至当前命令（其实可以交换指针），实现历史命令。这里需要注意内存安全，及时释放被覆盖的命令的空间。

1.4 提供输入输出重定向功能

解析命令时我们将输入输出文件名保存了下来，只需要在命令执行前使用`dup2()`进行重定向，即可实现此功能。注意输入输出完成后关闭文件释放资源。

我的程序额外实现了同时重定向输入与输出的功能。

1.5 允许父子进程通过管道进行通信

使用管道可以实现父子进程的通信，使用递归可以让子进程再创建子进程，获取上一条命令的输出并且将输出传递给下一条命令。为了兼容单命令，将判断是否有下一条命令决定输出到管道或是标准输出。

我的程序额外实现了多重管道、管道与输入输出重定向混合、每一段程序独立确定是否等待三个功能。

2 任务二：通过/proc/pid 获取进程信息

相比于之前的任务，我们需要额外实现`proc_write()`函数，通过 echo 输入获取需要查看的进程，`file_operations`中也需要添加操作符`.write`进行绑定。

为保证目标进程存在，使用 ps 命令获得当前正在运行的进程，再使用 awk 截取得到当前 bash（大概率是）的进程进行查看。

3 任务三：匿名管道与命名管道的差异

1. 匿名管道只可以用于两个父子进程的单向通信，命名管道可以用于任意进程之间的通信。
2. 匿名管道在文件系统中不可见，命名管道在文件系统中可见。
3. 匿名管道可以直接使用，命名管道需要像文件一样先打开才能进行操作。
4. 匿名管道有两个文件描述符，分别指向读端与写端，命名管道只有一个文件描述符，通过打开文件时指定打开方式确定读写功能。