ShellLab-Report

21302010042
侯斌洋

本次 lab 要求实现一个简易的 shell，且 shell 的框架已经给出，故只需要实现其中的一些关键函数即可。这些关键函数其实在 CSAPP 课本中也给出了大致的思路，不过需要做一些修改来进行完善。

此外，对于系统调用返回值的错误检查，用到了很多包装函数，这些函数也是 CSAPP 书中提到的 csapp.c 文件中的函数，相关的说明均已给出了注释。

下面介绍各个函数实现的大致思路

eval：

整体框架参考 CSAPP 中图 8-24 和图 8-40 的程序，首先通过调用已经给出的 parseline 函数来将命令行转化为参数列表，之后调用 builtin_cmd 函数来区分第一个参数是否为内置命令，若为内置命令则在 builtin_cmd 函数内部执行相应的操作，若不为内置命令则把第一个参数当作一个可执行程序进行执行，在这个过程中要创建新的子进程并在子进程中运行该程序。关于函数中的对于信号的阻塞以及前台后台程序的区分等详细内容不在此赘述，函数中都有详细的注释。

builtin_cmd：

该函数主要是通过 if 语句来判断命令的类型，若为内置命令则执行相应的操作，这些操作集成在不同的函数中，之后返回 1 或者直接退出（quit）；若不为内置命令则返回 0。函数整体思路较为简单明确。

do_bgfg：

该函数主要实现对于前台以及后台进程的一些操作。由于参数既可以为 JID 也可以为 PID，故需要进行区分并根据参数的类型进行不同的操作来获取整个作业的信息，包括 PID、JID 和 job 的地址。之后对于 bg 命令，要在后台重启进程，则利用 kill 函数发送 SIGCONT 信号并设置作业的状态；对于 fg 命令，要在前台重启进程，则利用 kill 函数发送 SIGCONT 信号并设置作业的状态，之后还需要一直等待该前台的结束。函数中的各种错误检测不在此赘述，请看详细代码以及注释。

waitfg：

该函数较为简单，用一个循环来等待前台作业的结束，用 fgpid()作为循环检测条件。当前台作业运行结束，被回收并从 jobs 中删除后，fgpid()返回 0 结束等待。

sigchld_handler：

框架参考 CSAPP 中图 8-40 的程序，通过循环来回收所有的僵死子进程。由于需要对停止的或因为 SIFINT 终止的子进程进行提示，故在回收时需要利用 WIFEXITED 等函数来检查子进程的退出状态。详细内容见代码。

sigtstp_handler：

该函数较为简单由于 SIGTSTP 信号是针对前台进程组的，故只需要获取前台作业 PID 并利用 kill 函数将该信号发送到该进程所在进程组的每一个进程即可。

sigint_handler：

该函数与 sigtstp_handler 思路基本相同。