实验报告

一、实验目标

本次实验的目标是通过编写一个简单的支持作业控制（job control）的 Unix shell 程序来熟悉进程控制和信号的概念。我们需要通过填充eval、builtin\_cmd、do\_bgfg等函数，自己实现一个简易的shell，使其具备实验要求中的功能，并得到与参考程序相同的输出。

二、实验过程

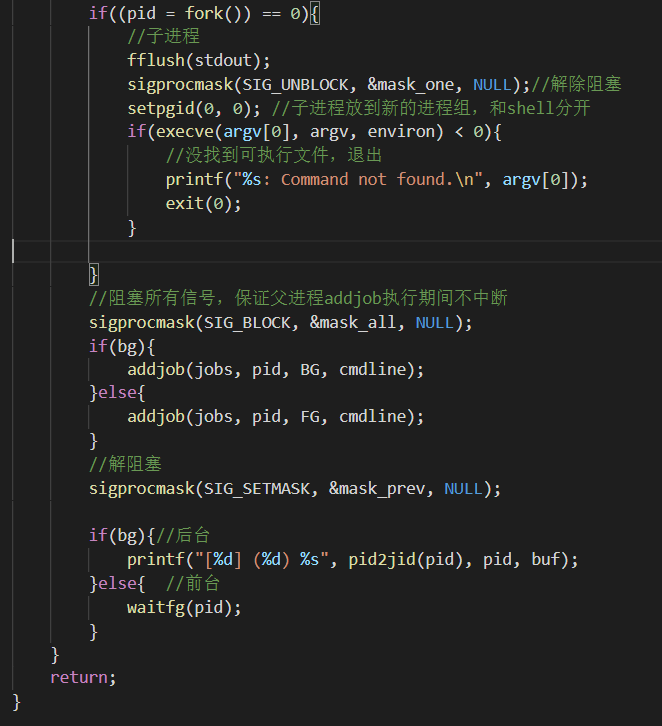
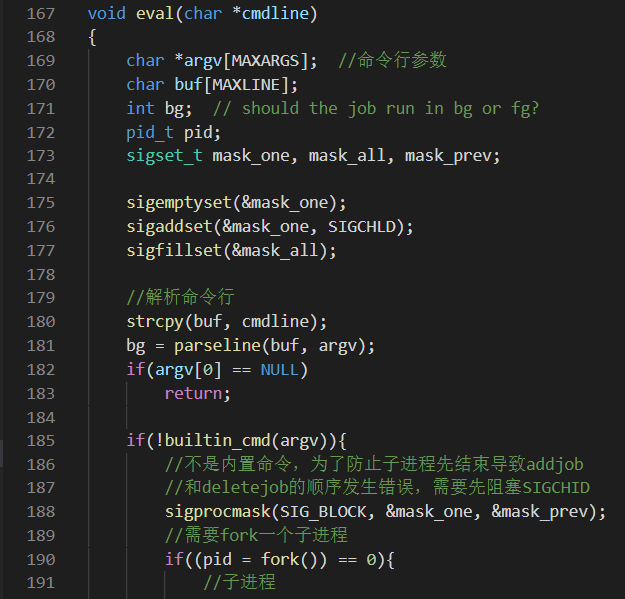
1. eval函数

根据说明，eval函数负责**解析并解释命令行的主要部分**。是内建命令则立即执行；否则fork一个子进程并在其上下文中运行该任务。且对前台任务，要等待它运行结束才能返回。

需要注意的是，父进程**fork子进程之前需要先屏蔽SIGCHLD信号**，以防止子进程在父进程addjob之前就已经运行结束被回收。并且我们还需要**为子进程重新分配进程组id**，以防止ctrl+c、ctrl+z命令的信号被发送给我们的shell和其创建的每一个进程。此外，**访问全局变量(如jobs)时需要阻塞所有信号。**（这一点在之后也会用到，届时不再赘述）

通过以上分析，首先初始化各变量并设置信号阻塞合集。其次，通过parseline函数解析命令行，以判断是否为后台命令，并将结果存放在bg中。然后builtin\_cmd函数判断是否为内建命令，是则由builtin\_cmd函数立即执行，否则由eval函数阻塞信号后fork子进程：子进程调用执行命令文件后退出；父进程执行addjob添加子进程后解阻塞，如果是前台任务则调waitfg函数等待，后台任务直接打印消息即可。

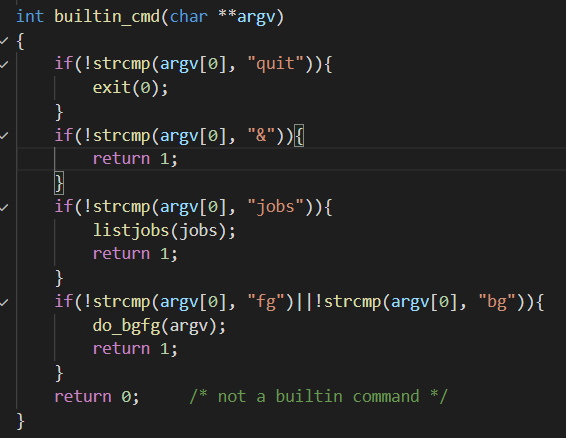
因此，eval函数实现如下：



2.builtin\_cmd函数

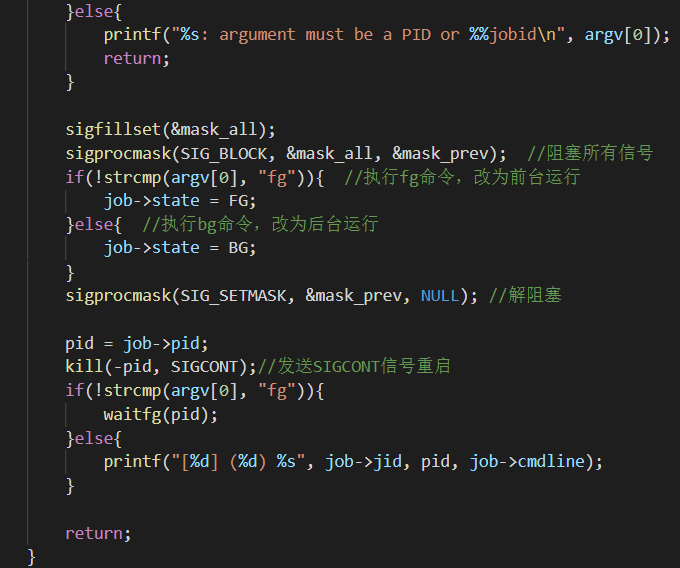
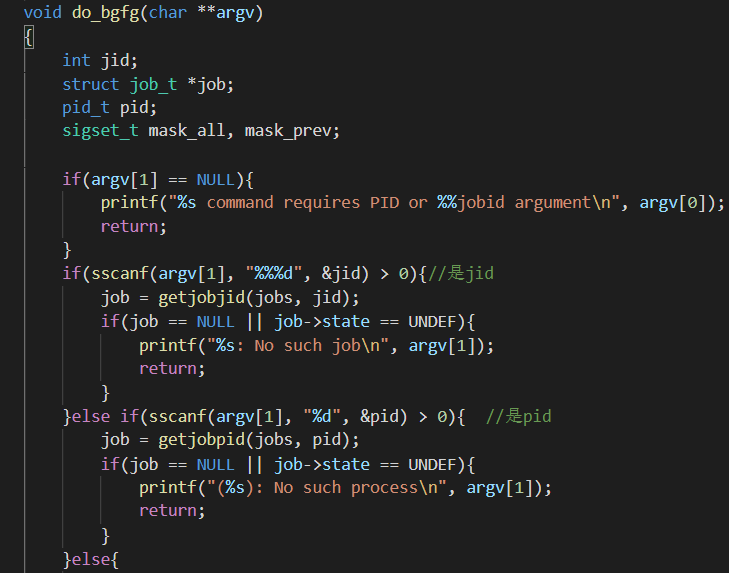
该函数的任务是识别并解释内建命令(quit、fg、bg以及jobs)。所以只需要**判断是哪个内建命令**，分别执行操作然后return 1即可。若与每个内建命令都不匹配，则不需要执行操作，return 0即可。

其中，对quit命令直接exit(0)退出，对单独的&指令直接无视，对jobs命令调用listjobs函数，对bg或fg命令则调用do\_bgfg函数即可。



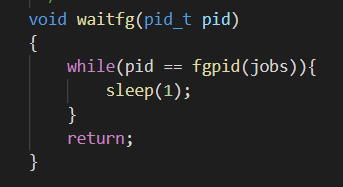
3.do\_bgfg函数

该函数的任务是实现内建命令bg <job>和fg <job>，既要**对bg和fg命令进行区分**，又要**对传入的参数是pid还是jid进行区分**。前者直接根据argv[0]判断即可，后者则利用它们格式上的不同，尝试用sscanf以不同格式读入（sscanf在正确读取时返回读取的参数个数，返回值大于0；读取失败时返回值为-1）。此外，还要注意对参数数量不够、重启不存在的进程等等**错误进行处理**。最后使用**kill发送SIGCONT信号**，并根据前后台状态采取不同操作。



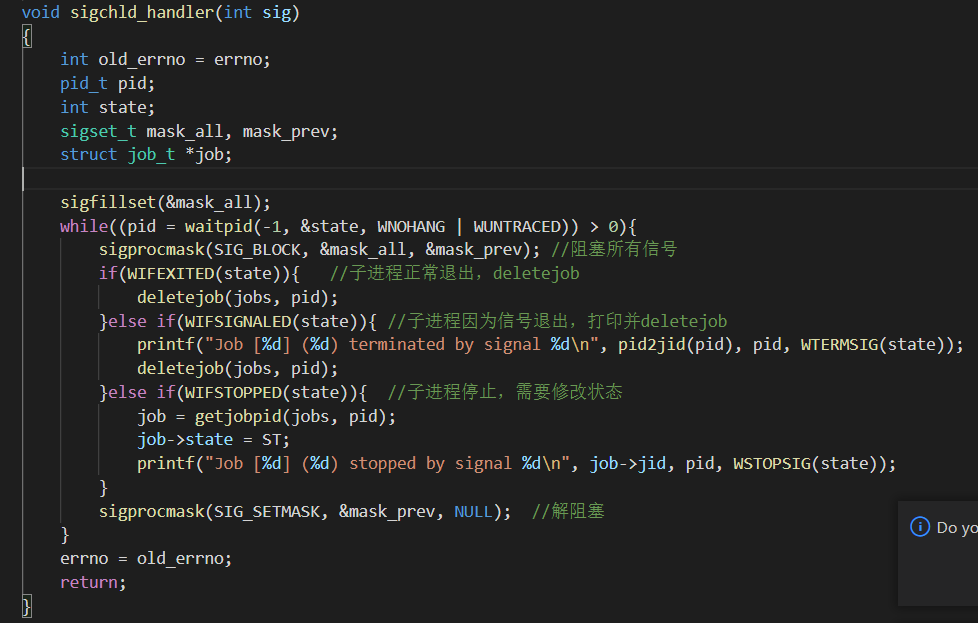
4.waitfg函数

该函数等待一个前台作业结束，只需要重复**调用sleep函数**直至当前子进程的pid与前台进程的pid不再相等即可。



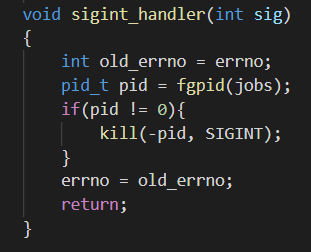
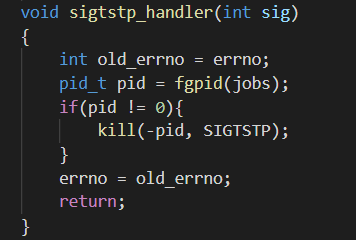
5. sigchld\_handler函数

该函数捕捉SIGCHLD信号，并作出响应。首先需要注意**使用while循环**而不是if来接收信号，因为不可靠信号不支持排队，一个未处理信号可能对应多个信号。然后waitpid函数回收子进程时，要将**参数设置为WNOHANG | WUNTRACED**，在当前进程中不存在已经停止或者终止的进程时立即返回，以防止卡在while循环中。最后则需要通过state参数和WIFEXITED等函数来**判断返回的进程终止或暂停的原因**，并进行相应的不同处理。



6. sigint\_handler函数和sigtstp\_handler函数

这两个函数的功能类似，分别捕捉SIGINT(ctrl-c)和SIGTSTP(ctrl-z)信号，并将其发送给所有前台程序。只需要获得前台进程的pid，然后使用kill发送信号即可。过程中注意保存和恢复errno。注意：根据提示，使用-pid作为kill的参数。

三、思维扩展

实现管道和输入输出重定向功能的思路：在解析命令行时，检测其中是否包含“< > |”标识符，如果不存在则正常执行命令即可。如果存在，对于输入输出重定向的情况，要保存标识符之后的文件名，通过dup2函数将标准输入或输出替换为相应文件；对于管道功能，要在当前子进程中再fork一个子进程，用新建的子进程执行管道符之前的命令，并重定向标准输出，待其完成后再由之前的子进程重定向标准输入。

四、正确性检验

对比test与rtest的01-16共16个案例的输出，除PID这类每次运行都会改变的值外，输出均与参考程序相同，证明了程序的正确性。

（以下随机展示几个样例）

