Lab 1: 简单 Shell 实验报告
221300034 高志轩

一些关键的问题：

1. 关于对命令的预处理：要实现这么多要求，涉及了对各类特殊符号的判断与分割，应该按照什么顺序来分割呢?

解决方案：通过对各种功能的综合了解及思考，最终确定需要按照“; & > |”的顺序依次进行分割。

2. 关于 paths 功能的实现：在实验中，一开始对 exec 系列的函数没有深入的理解，对实验的需求没有完全了解，错误的使用了 execvp 函数，导致无论我是否在 paths 数组中添加路径，都能够运行外部程序。

解决方案：通过搜索 exec 系列函数相关的函数的使用方法，了解到 execvp 函数会自动遍历所用的路径，所以导致了不能自己遍历路径。最后通过自己手动遍历自己维护的 paths 数组，恢复出完整的路径，再通过调用 access 函数看对应的文件是否可执行，再使用 execv 函数去执行。

3. 关于 bg 命令及后台运行：究竟什么是放到后台运行？需要我们具体进行什么处理？父进程如何知道子进程的结束信号且进行处理?

解决方案：放到后台运行就是在创建子进程时父进程不需要等待子进程的结束，而是直接执行，通过捕捉子进程结束时发出的 SIGCHLD 信号来进行处理。当我们需要进行后台运行时，父进程需要将子进程的 pid，命令信息存入到自己维护的全局变量 bg_task 中。当子进程结束时，父进程需要将 bg_task 中对应的后台进程删除，从而实现了后台执行的功能。

4. 关于管道的实现：对于多个管道的功能实现，是否需要对每一个子管道命令创建一个子进程？创建子进程后的错误如何处理?

解决方案：如果要支持多个管道，我的实现是通过一个 for 循环，对每一个命令都创建一个子进程，在每一个子进程中通过 execv 来进行执行，如果多个管道中有错误的命令，例如 ls -l | grp sh | wc -l， 一开始我是依次执行，中途记录下命令执行时是否发生错误，如果发生了错误，则记录在 has_error 变量中，最后如果有错误，则一起打印错误信息。但是最后还是先遍历整个管道命令，记录最后一个不可执行的命令(如 grp 是一个不可执行的命令)，如果有不可执行的命令，则从它后面一个命令开始执行，如果没有则从头执行。

5. 关于重定向与管道的关系：究竟是分别单独实现重定向和管道，还是一起实现?

解决方案：经过思考与尝试，最终发现重定向的功能可以被管道的功能涵盖，只需要稍微改动管道处理函数的代码即可。只需要对管道子命令的最后一个进行特殊处理，如果需要重定向，则将最后一个命令的标准输出重定向到目标文件上即可。

印象较深的 bugs：

1. 关于 strtok 函数：

实验过程中，当我尝试依次分割对应的字符（如“|”，“>”等）时，发现对应的字符串发生了变化，令我非常疑惑。之后查阅资料，发现 strtok 函数会更改原字符串，使用 strtok_r 函数可以不改变。但是当我更改成 strtok_r 函数后，还是发现对应位置的字符串没有正确读入，这让我更加迷惑。最后通过单步调试，发现了一个非常低级的错误，nsh 处理完第一个命令后，**忘记将 argc 置为 0**，导致后面的 args[argc]位置始终在正确位置后面，内容甚至为空，所以导致了字符串读取错误。

2. 关于管道与重定向：

一开始我先实现了重定向功能，之后再添加的管道功能。但是当我实现好管道功能之后，发现事实上重定向功能可以在管道功能上修改一点就可以实现，也就是当我们执行到管道的最后一个子命令时，将输出不重定向到管道，而是此时打开目标文件，并且将标准输出重定向到目标文件的文件描述符上即可。

但是，在之后的本地测试中，我发现形如“ls | ”的命令不能正确处理，于是我添加了一个对管道命令个数的特判，即如果只有一个管道命令则判错。但是当我借助处理管道的函数处理重定向的时候，发现对于“ls > 1.txt”这个命令就会报错而不能正常重定向。debug 挺长时间，发现这个命令在进入管道处理函数的时候被上面的**错误特判处理**而报错退出了，导致提前结束。

3. 关于 OJ 测试：

非常无语的是，对于所有的测试样例，包括哪些长期测试不过的样例，一开始在本地全都能够正常运行，就是不明白为什么 OJ 过不了。。。

最后经过非常仔细的检查，终于发现原来是输出命令行提示信息“nsh> ”后**没有及时调用 fflush(stdout)**导致将近一半的测试样例都不过。。。