Lab3:Shell Lab

目标

- 完成一个自己的Unix Shell程序，支持一般的工作调度，进程信号控制和前后台程序切换等。

要求

1. 请不要抄袭，可以与同学讨论，但不要直接抄袭同学的代码和实验报告。
2. 请认真完成实验报告，请说明你认为本次实验的**重点**、**难点**和**你这样认为的原因**，以及**你遇到的问题和你解决的方式**，以及**你认为可能成为你遇到的问题但被你提前解决并以此自得的点**，并可以插入屏幕截图辅助说明。
3. 请在截至日期（初定为2020.5.17日23：55）前将实验报告提交至Unicourse+上，**不必提交代码。**

指导

1、基本步骤

- 登录服务器
  - 地址：192.144.229.231
  - 用户名密码：学号
  - 登录后尽快使用passwd 命令修改密码。
- 使用`tar xvf shlab-handout.tar`命令解压。
- 仔细阅读README 文件。
  - `tsh.c` 你需要完成的代码，其中已经完成一些基本的框架。
  - `tshref` 供你参考的二进制文件，你编写完成的Shell程序应与该程序表现完全一致。
  - `trace*.txt`用来测试你的Shell程序正确性的输入文件。
  - `*sdriver.pl` 用来测试你的Shell程序正确性的驱动脚本。
  - `tshref.out` 16个测试文件所期望的正确输出。
  - `my*.c` 一些测试时被调用的小程序。
- **认真阅读并理解tsh.c 中的基本框架。**
- 完成并测试你自己的Shell程序。（你可以使用make 命令来编译你的程序。）
- 提交你的代码。
  - 将`Makefile` 文件中`STUID` 后 2018000000 改为你的学号。
  - 确认正确无误后使用`make handin` 命令提交你的代码。
  - 如果你想再次提交，需要将`Makefile` 文件中的`VERSION` 后的数字增大1。**每个人最多可以提交两次，如果你提交次数大于两次，将对你的成绩产生负面影响，所以请谨慎对待你的每次提交。**

2、你需要完成的函数

- `eval` Shell命令执行的主路线。
- `builtin_cmd` 执行Shell程序中的内建命令。
- `do_bgfg` 执行前台后台任务切换及调度相关的内建命令。
- `waitfg` 等待一个前台任务结束。

- `sigchld_handler` 捕获并处理SIGCHILD信号。
- `sigint_handler` 捕获并处理SIGINT（ctrl-c）信号。
- `sigtstp_handler` 捕获并处理SIGTSTP（ctrl-z）信号。

3、tsh应具有的特性

- 引导提示信息应为tsh> （已在框架中完成）。
- 每一行输入的命令由一个name和若干个argument组成，它们之间有一个或多个空格分隔。如果name是一条内建命令，tsh应该立即执行它并等待输入下一行命令；否则tsh应该认为name是一个可执行文件的路径，并应该在一个子进程中加载并执行它（我们称之为一个任务/job）。
- tsh不必支持管道符(|)和输入输出重定向(<,>)。
- 当你输入ctrl-c(ctrl-z)时，tsh应该立即向当前的前台任务（以及所有它的子进程）发送一个SIGINT(SIGTSTP)信号，如果此时没有前台任务，那么ctrl-c(ctrl-z)将没有任何效果。**如果你想退出Shell程序，可以输入ctrl-d。**
- 如果一行命令以一个& 符号结尾，那么tsh应在后台执行这个任务，否则tsh应在前台执行这个任务，此时应打
  印出这个任务的相关信息和命令。
- 每个任务/job可以以两种方式表示，进程号(PID)和任务号(JID)，这两个ID都为正整数，其中任务号是tsh内部设定的标识。任务号在命令中应有前缀% ，如%5 表示JID为5的任务， 5 表示PID为5的任务。（框架中已经实现你所需的PID与JID的转换函数。）
- tsh应该支持如下的内建命令：
  - `quit` 退出tsh（退出并回收所有任务）。
  - `jobs` 列出所有的后台任务。
  - `bg <job>` 通过向`<job>`发送一个SIGCONT信号使其在**后台**继续执行， `<job>`可以是PID或JID，此时应
    打印出这个任务的相关信息和命令。
  - `fg <job>` 通过向 `<job>` 发送一个SIGCONT信号使其在**前台**继续执行， `<job>` 可以是PID或JID。
- tsh应该回收所有僵尸状态(zombie)的子进程。如果一个任务因为收到了一个信号而暂停或退出，tsh应该识别
  出这种事件并打印一条包含任务PID和该信号的描述。

4、验证你的程序

实验提供了一些工具来验证你的程序。

- `tshref` 一个参考二进制文件，如果你有任何关于你的Shell程序的问题，你可以运行该文件来获得解答，你的
  程序应与该程序表现完全一致（除了如PID这样的每次运行会发生改变的值，和-p 参数下输出的调试信息）。
- `sdriver.pl` 一个以子进程运行你的Shell程序的验证脚本，它会向你的Shell根据`trace*.txt` 发送命令和信
  号，然后从你的Shell捕获输出。
  - 使用`-h` 参数来获得该脚本的帮助信息。
  - 你可以使用`make` 命令来运行该脚本（输入更简便）：
    - 你需要首先使用`make` 命令编译你的Shell程序。
    - `make test*`以`trace*.txt`作为输入来运行你的Shell程序，并使用verbose模式（更详细的输
      出）。比如，你可以输入`make test01`。

    - `make rtest*`以`trace*.txt`作为输入来运行参考Shell程序，并使用verbose模式（更详细的输
      出）。比如，你可以输入`make rtest15`。

5、一些有用的提示

- 你可以循序渐进地使用`trace*.txt`来帮助你完成这个实验，比如先完成`trace01.txt`，再完成`trace02.txt`，以此类推。
- 一些类似`more`、`less`、`vi`、`emacs`之类的程序会根据终端的设置进行一些奇怪的操作，所以尽量不要用你的Shell程序运行这些程序；你可以用一些简单的，基于文本的小程序来试验你的Shell，如`/bin/ls`、`/bin/ps`、`/bin/echo`等。
- 默认框架已经实现在tsh程序中输入ctrl-d退出程序，如果你的程序死循环，你可以使用SIGKILL信号强行终止程序。
- 当你实现SIGINT和SIGTSTP信号的处理程序时，要确保信号被发送给指定任务的所有子进程，你可以在`kill`函
  数中使用`-pid`来代替`pid`作为参数。
- 当你等待前台程序执行完毕的时候，如何在`waitfg`函数和`sigchld_handler`函数之间分配任务值得考虑。我
  推荐下面这个做法：
    - 在`waitfg`函数中使用`sleep`函数并在其附近采用忙等待的策略。
    - 在`sigchld_handler`函数中只调用一次`waitpid`函数。

  当然其它的解决方式也是可行的，比如在waitfg 和sigchld_handler 中都调用waitpid 函数，但在sigchld_handler 函数中进行判断，但这非常复杂而且容易出错，把所有的回收工作放在handler里更容易一
  些。
- 在`sigchld_handler`中，你使用`waitpid`函数时，一些可选参数如`WNOHANG`、`WUNTRACED`、`WCONTINUED`可能对你很重要，同时你要考虑多个SIGCHLD信号并发的情况。
- 你应该考虑在`fork`函数附近使用`sigprocmask`函数（`sigset_t`与之匹配使用）屏蔽一些信号，当把一个任务加进任务列表时同样需要屏蔽一些信号。但记住子进程从你的Shell主进程继承信号屏蔽列表，记得解除屏蔽。**考虑一下需要屏蔽哪些信号，为什么？请把你的思考结果写入实验报告。**
- 由于你的Shell程序是执行在一个标准的Unix Shell程序，参见第4条提示，Unix Shell的编写者会把ctrl-c和ctrl-z信号发送到所有你的Shell程序的子进程，而这显然不是我们想要的。我们希望只由你的Shell程序接收这些信号再继续向子进程分发，所以在`exec`族函数被执行之前，你需要使用`setpgid`函数为你的子进程重新分配进程
  组ID。

评分要求

- 代码（64%）：
    - 每个测试点4分，共16个测试点（已发给大家），共计64分。
    - 代码风格（注释，缩进，变量名），若不好，视情况扣分。
- 报告（36%）：
    - 你认为的重难点。
    - 你遇到的问题。
    - 你认为你优秀的地方。
    - 反思总结。
- 按时提交：

- 晚交酌情减分。