Shell 挑战性任务

本次 Lab6 挑战性任务需要同学们在 MOS 原有的 shell(mosh) 上实现新的功能，该任务会提供若干任务，完成所有任务后即可参加自动化评测获取挑战性任务分数。

任务要求

- 任务文档：**请务必详细地记录你的实现内容，包括但不限于函数的功能，模块的定义。**
- 参与自动化测试。
- 查重。

参考文档

- Bash Reference Manual
- Linux man pages

实现不带 `.b` 后缀指令

你需要实现不带 `.b` 后缀的指令，但仍需兼容带有 `.b` 后缀的指令，如 `ls` 与 `ls.b` 都应能够正确列出当前目录下的文件。

实现指令条件执行

你需要实现 Linux shell 中的 `&&` 与 `||`。 对于 `command1 && command2`，`command2` 被执行当且仅当 `command1` 返回 0；对于 `command1 || command2`，`command2` 被执行当且仅当 `command1` 返回非 0 值。

> 注: 评测中保证不出现括号。并且需要注意的是，在 bash 中 `&&` 与 `||` 的优先级相同，按照从左到右的顺序求值。
>
> 例如 `cmd1 || cmd2 && cmd3`，若 `cmd1` 返回 0，则 `cmd1` 执行后 `cmd2` 不会被执行，`cmd3` 会被执行；若 `cmd1` 返回非 0 且 `cmd2` 返回非 0，则 `cmd3` 将不会被执行。
>
> 提示：你可能需要修改 MOS 中对用户进程 `exit` 的实现，使其能够返回值。

实现更多指令

你需要实现 `touch`，`mkdir`，`rm` 指令，只需要考虑如下情形：

- `touch`:

> - `touch <file>`：创建空文件 `file`，若文件存在则放弃创建，正常退出无输出。 若创建文件的父目录不存在则输出 `touch: cannot touch '<file>': No such file or directory`。 例如 `touch nonexistent/dir/a.txt` 时应输出 `touch: cannot touch 'nonexistent/dir/a.txt': No such file or directory`。

- `mkdir`:

> - `mkdir <dir>`：若目录已存在则输出 `mkdir: cannot create directory '<dir>': File exists`，若创建目录的父目录不存在则输出 `mkdir: cannot create directory '<dir>': No such file or directory`，否则正常创建目录。

> - `mkdir -p <dir>`：当使用 `-p` 选项时忽略错误，若目录已存在则直接退出，若创建目录的父目录不存在则递归创建目录。

- `rm`：

> - `rm <file>`：若文件存在则删除 `<file>`，否则输出 `rm: cannot remove '<file>': No such file or directory`。
> - `rm <dir>`：命令行输出: `rm: cannot remove '<dir>': Is a directory`。
> - `rm -r <dir>|<file>`：若文件或文件夹存在则删除，否则输出 `rm: cannot remove '<dir>|<file>': No such file or directory`。
> - `rm -rf <dir>|<file>`：如果对应文件或文件夹存在则删除，否则直接退出。

实现反引号

你需要使用反引号实现指令替换。只需要考虑 `echo` 进行的输出，你需要将反引号内指令执行的所有标准输出替换为 `echo` 的参数。例如：

```
echo `ls | cat | cat | cat`
```

实现注释功能

你需要使用 `#` 实现注释功能，例如 `ls | cat # this is a comment meow`，`ls | cat` 会被正确执行，而后面的注释则会被抛弃。

实现历史指令

你需要实现 shell 中保存历史指令的功能，可以通过 Up 和 Down 选择所保存的指令并执行。你需要将历史指令保存到根目录的 `.mosh_history` 文件中（一条指令一行），为了评测的方便，我们设定 `$HISTFILESIZE=20`（bash 中默认为 500），即在 `.mosh_history` 中至多保存最近的 20 条指令。你还需要支持通过 `history` 命令输出 `.mosh_history` 文件中的内容。

> 注：在 bash 中，`history` 为 shell built-in command，我们规定需要将 `history` 实现为 built-in command。

你需要将当前执行的指令先存入 `.mosh_history` 中，例如：

```
echo `ls | cat`
echo meow # comment
history
history | cat
```

当历史指令为空时，依次执行上述四条指令后，后两条指令会分别输出

```
echo `ls | cat`
echo meow # comment
history
```

与

```
echo `ls | cat`
echo meow # comment
history
history | cat
```

使用 Up 能够切换到上一条指令（如果上一条指令存在），使用 Down 能够切换到下一条指令（如果下一条指令存在）。能够选择的指令范围为：用户当前输入的指令与 `.mosh_history` 文件中保存的所有指令。例如在执行了上述四条指令后，用户输入了 `echo`，此时 Up 应该将指令切换至 `history | cat`，再次进行三次 Up 后切换至 `echo`ls | cat `，此时再次 Up 应保留在该指令（因为已经不存在上一条指令）；再进行四次 Down 后将切换回`echo`，此时再次 Down 应保留在该指令（因为不存在下一条指令）。

实现一行多指令

你需要实现使用 `;` 将多条指令隔开从而从左至右依顺序执行每条指令的功能。例如：

```
ls;ls | cat; echo nihao,mosh; echo `ls; echo meow`
```

实现追加重定向

你需要实现 shell 中 `>>` 追加重定向的功能，例如：

```
ls >> file1; ls >> file1
```

最后文件 `file1` 中将会有两次 `ls` 指令的输出。

实现引号支持

你需要实现引号支持，比如 `echo "ls >"`，shell 在解析时需要将双引号内的内容看作是单个字符串。

实现前后台任务管理。

- 你需要支持 mosh 运行后台进程，当命令的末尾添加上 `&` 符号时，该命令应该在后台执行。
- 实现 `jobs` 指令列出当前 shell 中所有后台任务的状态。你需要为任务创建 ID（每次启动 mosh 时，任务从 1 开始编号，每个新增任务编号应加 1），并且通过 `jobs` 指令输出包括：任务 ID（`job_id`）、任务的运行状态（`status`：可能的取值为 `Running`，`Done`）、任务的进程 ID（`env_id`）与运行任务时输入的指令（`cmd`）。请以 `printf("[%d] %-10s 0x%08x %s", job_id, status, env_id, cmd)` 的格式进行输出。
- 实现 `fg` 将后台任务带回前台继续运行，用户通过 `fg <job_id>` 的方式将对应任务带回前台。
- 实现 `kill` 指令，用户通过 `kill <job_id>` 来实现结束后台任务。

在 `fg` 或 `kill` 指令中，若 `job_id` 对应的后台任务不存在则输出 `printf("fg: job (%d) do not exist\n", job_id)`，若 `job_id` 对应的 ID 为 `envid` 的进程状态不为 `Running` 则输出 `printf("fg: (0x%08x) not running\n", envid)`。

例如：

```
sleep 10&
sleep 60 &
jobs
# wait for about 10 seconds...
jobs
```

依次执行上述指令，则第一个 `jobs` 应输出（其中进程 ID 的值可能与你本地运行的输出结果不同）：

```
[1] Running    0x00003805 sleep 10&
[2] Running    0x00005006 sleep 60 &
```

第二个 `jobs` 应输出：

```
[1] Done       0x00003805 sleep 10&
[2] Running    0x00005006 sleep 60 &
```

任务测试

完成后需要在课程网站提交代码进行自动评测。