15213-ICS-tshlab

2017年8月27日 16:17

这次的实验感觉好牛逼的样子，直接生撸一个简易版的Shell程序。
这个程序每一行会输出 tsh>， 然后等待用户输入。 用户的输入包括name加上对应参数，如果name是内置命令，直接执行，否则需要新建一个子进程，并在子进程中完成具体的工作。这个程序需要支持重定向功能 > 和 <。对于前台任务，当用户输入ctrl-z 或者ctrl-c的时候，能有相应反应。如果命令以&结尾，转为后台任务。

对于这个程序来说，第一步当然是解析命令，这个程序本来是提供了解析函数的，但是我看大神们把它给修改了。

```
/* Parsing states */
#define ST_NORMAL   0x0   /* next token is an argument */
#define ST_INFILE   0x1   /* next token is the input file */
#define ST_OUTFILE  0x2   /* next token is the output file */

struct cmdline_tokens {
    int argc;               /* Number of arguments */
    char *argv[MAXARGS];    /* The arguments list */
    char *infile;           /* The input file */
    char *outfile;          /* The output file */
    enum builtins_t {       /* Indicates if argv[0] is a builtin command */
        BUILTIN_NONE,
        BUILTIN_QUIT,
        BUILTIN_JOBS,
        BUILTIN_BG,
        BUILTIN_FG} builtins;
};
```

这样的话，在解析函数中判断是不是内置命令，如果是，是什么，是前台任务还是后台任务，方便后续工作。
内置工作相对简单一些，先搞定它们。对于quit来说，一个exit(0)就搞定：

```
hao@hao:~/Documents/CMU/cmu-15213/Shell Lab/shlab-handout$ ./tsh
tsh> quit()
hao@hao:~/Documents/CMU/cmu-15213/Shell Lab/shlab-handout$
```

然后对于jobs而言，就是把当前所有的job显示出来，并且判断重定向。直接调用listjobs函数即可。

```
case BUILTIN_JOBS:
    /* print the job list */
    if (tok.outfile == NULL)
        listjobs(jobs, 1);
    return;
```

接下来实现BG和FG两个命令，改变相关job状态值。我们需要注意的地方有两个，一个是先根据判断传入的是 JID 还是 PID，然后发送信号之后需要等待进程完成（这里注意使用 sigsuspend）。

对于其他命令，新建子进程运行：

```
if ((pid = fork()) == 0) {
 if (execve(tok.argv[0], tok.argv, environ) < 0)
```

同样需要注意重定向的处理。
如果是前台程序，suspend，等待当前的前台程序结束。
在整个处理过程中应该上锁，避免中断进入。

```
sigprocmask(SIG_BLOCK, &mask, NULL);
sigprocmask(SIG_UNBLOCK, &mask, NULL);
```

最后结果：

```
hao@hao:~/Documents/CMU/cmu-15213/Shell Lab/shlab-handout$ ./tsh
tsh> ./myspin 2 &
[1] (22652) ./myspin 2 &
tsh> ./myspin 3 &
[2] (22657) ./myspin 3 &
tsh> jobs
[1] (22652) Running    ./myspin 2 &

[2] (22657) Running    ./myspin 3 &

tsh>
```

后面的mark一下，后续再写。