Lab5: Shelllabs

CSE4009: System Programming

Woong Sul

Overview

- Fetch the handout from hconnect
- Place and extract the handout file
- Complete your assignment
- Push your final source files to hconnect

Goal

- Understanding how a shell
  - Interprets and processes commands
  - Sends a signals to the target processes
  - Switches jobs between foreground and background

parseline bg면 1 return
           fg면 0 return

if(!bg) waitpid(pid, &status, 0);

1. Download the handout file

- Check the handout file assigned to you

  $ git pull origin

```
wsul@splab2022012345:~/Projects/evals/12843/2022_cse4009_201220789$ git pull origin
Already up to date.
wsul@splab2022012345:~/Projects/evals/12843/2022_cse4009_201220789$
```

2. Extract handout

- Check your files
  - README
  - Makefile
  - sdriver.pl: testing program
  - **tsh.c:** your tiny shell (incomplete)
  - tsh-ref: the reference binary for tsh.c
  - trace01.txt ~ trace16.txt: tests to validate your tsh
  - tshref.out: example output for tshref
  - myspin.c, mysplit.c, mystop.c, myint.c

3. Complete your tiny shell: tsh

- test01 is already done, and move on to test02
  - Your tsh should exit with “quit” command.
  - To do so you need to complete eval()
    - converts a command line (cmdline) to char*[] (argv)
    - validates the command line with builtin_cmd(char ** argv)

```
161 /*
162  * eval - Evaluate the command line that the user has just typed in
163  *
164  * If the user has requested a built-in command (quit, jobs, bg or fg)
165  * then execute it immediately. Otherwise, fork a child process and
166  * run the job in the context of the child. If the job is running in
167  * the foreground, wait for it to terminate and then return.  Note:
168  * each child process must have a unique process group ID so that our
169  * background children don't receive SIGINT (SIGTSTP) from the kernel
170  * when we type ctrl-c (ctrl-z) at the keyboard.
171  */
172 void eval(char *cmdline)
173 {
174   return;
175 }
```

3. Complete your tiny shell: tsh

- test03
  - Your tsh should run an external program
  - To do so you need to use fork() and exec()
    - fork() creates a child process
      - The child process keeps running (to call exec())
      - The parent process waits
    - exec() replaces the current process image with new one
  - You may add a job with addjob() later

```
173 void eval(char *cmdline)
174 {
175   pid_t pid;
176   char *argv[MAXARGS];
177   parseline(cmdline, argv);
178   if (!builtin_cmd(argv)) {
179     if ((pid = fork()) == 0) {
180       execve(argv[0], argv, environ);
181
182     }
183   }
184   return;
185 }
```

4. Some useful hints

- Job is not a Linux Process
  - Process is a linux data structure created by fork()
  - Job is a tsh own data structure created by addjob()
    - Job has three states

```c
/* addjob - Add a job to the job list */
int addjob(struct job_t *jobs, pid_t pid, int state, char *cmdline)
{
  int i;

  if (pid < 1)
    return 0;

  for (i = 0; i < MAXJOBS; i++) {
    if (jobs[i].pid == 0) {
      jobs[i].pid = pid;
      jobs[i].state = state;
      jobs[i].jid = nextjid++;
      if (nextjid > MAXJOBS)
        nextjid = 1;
      strcpy(jobs[i].cmdline, cmdline);
      if(verbose){
        printf("Added job [%d] %d %s\n", jobs[i].jid, jobs[i].pid, jobs[i].cmdline);
      }
      return 1;
    }
  }
  printf("Tried to create too many jobs\n");
  return 0;
}
```

```c
/*
 * Jobs states: FG (foreground), BG (background), ST (stopped)
 * Job state transitions and enabling actions:
 *     FG -> ST  : ctrl-z
 *     ST -> FG  : fg command
 *     ST -> BG  : bg command
 *     BG -> FG  : fg command
 * At most 1 job can be in the FG state.
 */
```

4. Some useful hints

fprintf(stderr, "%s\n", cmdline);
strcmp() 동일하면 0 return
ctrl D 강제종료

- Job is not a Linux Process
- Job has three states and signals can change the state
  - SIGSTP: (BG or FG) to ST
  - SIGCONT: ST to (BG or FG)

```
29 /*
30  * Jobs states: FG (foreground), BG (background), ST (stopped)
31  * Job state transitions and enabling actions:
32  *     FG -> ST  : ctrl-z
33  *     ST -> FG  : fg command
34  *     ST -> BG  : bg command
35  *     BG -> FG  : fg command
36  * At most 1 job can be in the FG state.
37  */
```

4. Some useful hints

- Job is not a Linux Process
- Job has three states and signals can change the state
- You need to call setpgid(0, 0)
  - Any child process belongs to the same process group with tsh
  - Ctrl + c sends SIGINT to all processes created from tsh (including tsh)
  - So you need to separate the child process from tsh by setting new process group

4. Tests

- make rtest# (01 to 16)
  - Check how your tsh behaves with each trace file
- make test# (01 to 16)
  - test your tsh with each trace file
- make rtests / make tests
  - test with all trace files from trace01.txt to trace16.txt

6. Submission

- Submit **tsh.c** file to hconnect

Good Luck!