CSED211: Lab 9

Shell Lab

Postech

김정우: jwkim0417@postech.ac.kr

Table of Contents

Shell

What is shell?

- An interactive command-line interpreter that runs programs on behalf of user
- Command line: a sequence of ASCII text words
- Common exmples: Bash(Bourne-again shell)
  - Linux default
- Most applications in linux (command line) are run through shell

Basic functions of shell

- jobs: List the running and stopped background jobs
- bg <job>: Change a stopped background job to a running backgroun job
- fg <job>: Change a stopped or running background job to a running in the foreground
- ctrl+c: deliver SIGINT signal to each process in the foreground job
- ctrl+z: deliver SIGTSTP signal to each process in the foreground job

Basic functions of shell

```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]   Stopped                 vi 111
[2]-  Stopped                 vi 222
[3]+  Stopped                 vi 333
```

Job id, spec status

Job name

```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ fg %1
vi 111
```

```
^Z
[1]+  Stopped                 vi 111
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]+  Stopped                 vi 111
[2]   Stopped                 vi 222
[3]-  Stopped                 vi 333
```

Basic functions of shell

```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ python count_10.py
start
^Z
[1]+  Stopped                 python count_10.py
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]+  Stopped                 python count_10.py
```

```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ bg %1
[1]+ python count_10.py &
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]+  Running                 python count_10.py &
```

```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ python count_10.py
start
^CTraceback (most recent call last):
  File "count_10.py", line 5, in <module>
    time.sleep(1)
KeyboardInterrupt
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$
```

Homework

Lab Homework 9

- Due: 12/02 23:59 (midnight)
- Upload your source file and report
  - Explain your answer in the report
  - File name format : [student_#]-tsh.c / [student_#].docx

- Refer to ‘writeup_lab9’ and following description.

Shell Lab

- Write a simple shell (‘tsh’)
- Only need to write seven specified functions
- Helper function already provided in source file

Tsh basic functionality

- Run a command or application on shell
  - E.g. tsh>/bin/ls -l -h
    - Shell forks child process
    - Executes “/bin/ls” with arguments “-l” and “-h”
  - Argv[0] = “/bin/ls”, argv[1] = “-l”, argv[2] = “-h”

* Tsh manages running application as child process

Tsh basic functionality

- Foreground job management
  - Runs application in foreground and waits for its ending
  - E.g. tsh>/bin/ls -l -h
    - Shell executes “/bin/ls” with “-l –h”
    - Wait for it to finish before other application runs

* Every application run is foreground by default

Tsh basic functionality

- Background job management
  - Runs application in background
  - Many simultaneous background jobs possible
  - “&” added to end of command/application name
  - E.g. tsh>./myprogram &

* Tsh can run many jobs in the background

Tsh basic functionality

- Background/Foreground management
  - Changes job status (bg to fg / fg to bg)
  - E.g. tsh>fg <job_id>
    - Makes a background job run as foreground
  - E.g. tsh>bg <job_id>
    - Makes a foreground job run as background

* Tsh can move jobs between foreground and background

Tsh basic functionality

- Prints list of jobs
  - E.g. tsh> jobs
    - Prints list of jobs including both running and stopped.

- Quit tsh and return to bash
  - E.g. tsh> quit
  - Or press ctrl+d

Work in this assignment

- Many helper function coded for you
  - Parseline
  - addjob, deletejob, clearjob
  - Fqpid
  - getjobpid, getjobid, …
- 4 programs used by tsh
  - myint
  - myspin
  - mysplit
  - mystop

Work in this assignment

- eval: Main routine that parses and interprets the command line.
- builtin_cmd: Recognizes and interprets the built-in commands: quit, fq, bg, and jobs
- do_bgfg: Implements the bg and fg built-in commands.
- waitfg: Waits for a foreground job to complete.
- sigchld handler: Catches SIGCHLD signals.
- sigint handler: Catches SIGINT (ctrl-c) signals.
- sigtstp handler: Catches SIGTSTP (ctrl-z) signals.

* Reuse already coded functions in tsh.c to write specified functions

How to evaluate your code

- Use the provided ‘reference tsh’ binary & 16 traces
- Test each tsh feature with a trace & provided ‘sdriver.pl’
- Your tsh output must match ‘tshref.out’

1. ./sdriver.pl -t trace01.txt -s ./tsh -a “-p”
   or make test01
2. ./sdriver.pl -t trace01.txt -s ./tshref -a “-p”
   or make rtest01
3. If the result of (1) and (2) match, then your code is correct.

Hints

- You should use kill() function well.
  - kill(pid, SIG) => sent SIG signal to job which has pid.
- fork() return 0 in child process, but return pid of child process in parent process.
  - You can distinguish child or parent by checking the return value.
  - fork() also copies signal mask from parent to child.
- You can use waitpid() function.
  - You can wait child process and check child's status.

Assignment Submission

- Complete tsh.c
- Rename file to “student_id-tsh.c”
  E.g. 20212927-tsh.c

- Submit your renamed tsh code and report to PLMS
- Due: 12/02 23:59 (midnight)