M1522.000800: System Programming
L4. Shell lab: Session 1

Computer Systems and Platforms Laboratory
School of Computer Science and Engineering
Seoul National University

Outline

- Goal
- General Overview of Unix Shells
- The tsh Specification
- How to test your code
- How to submit & Evaluation

Goal

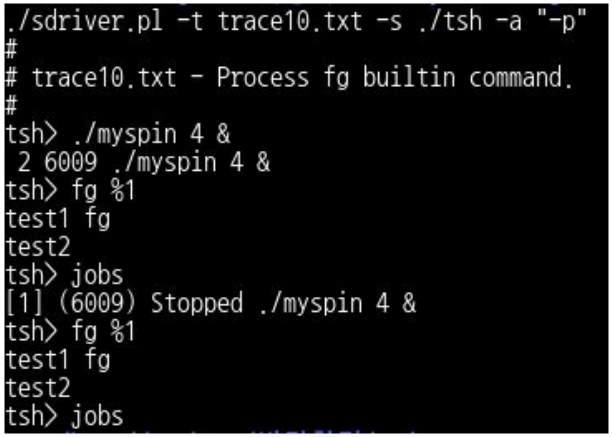
Goal

- Understand how the shell lab works, and what exactly is expected from you and your submission.
- Understand the concepts of process control and signaling used in this lab.
  - How to “make” this tiny shell
- How to test your tiny shell implementation

General Overview of Unix Shells

General Overview of Unix Shells

- A **shell** is an interactive *command-line* interpreter.
  - Runs programs on behalf of the user.
  - Repeatedly prints a prompt
  - Wait a command line on ***stdin*** and then carries out some action.



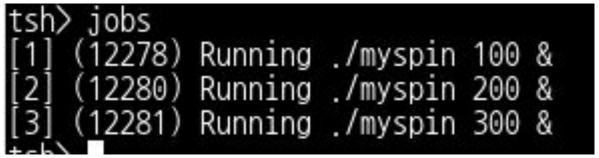
```
./sdriver.pl -t trace10.txt -s ./tsh -a "-p"
#
# trace10.txt - Process fg builtin command.
#
tsh> ./myspin 4 &
 2 6009 ./myspin 4 &
tsh> fg %1
test1 fg
test2
tsh> jobs
[1] (6009) Stopped ./myspin 4 &
tsh> fg %1
test1 fg
test2
tsh> jobs
```

- Command line (CL)
  - A sequence of ASCII text words delimited by whitespace.
    - 4 The first word is a built-in command or the pathname of an executable file.
    - 4 The remaining words are command-line arguments.
  - Shell executes the built-in commands in the current process.
  - Shell forks a child process directed by the pathname of an executable program.
    - 4 A process and its child processes are known collectively as a *job*.

General Overview of Unix Shells (Cont.)

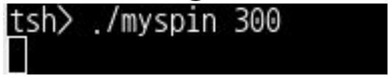
- Background & Foreground
  - If the command line ends with an <u>ampersand “&”</u>, then the job run in the <u>***background***</u>.
    - <u>The shell does **not wait for the job**</u> to terminate before printing the prompt and awaiting the next command line.
    - An arbitrary number of jobs can run in the background.

```
tsh>./myspin 100 &
```



```
tsh> jobs
[1] (12278) Running ./myspin 100 &
[2] (12280) Running ./myspin 200 &
[3] (12281) Running ./myspin 300 &
```

  - Otherwise, the job runs in the ***foreground***.
    - <u>The shell **waits for the job**</u> to terminate before awaiting the next command line.
    - At most one job can be running in the <u>foreground.</u>

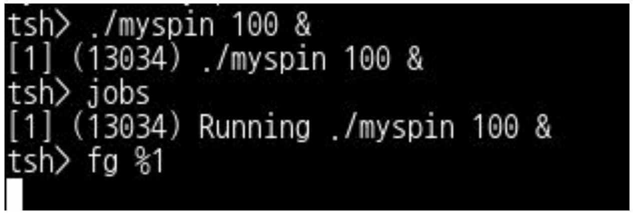


```
tsh> ./myspin 300
```

```
tsh>./myspin 100
```

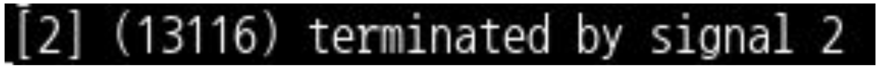
General Overview of Unix Shells (Cont.)

- Unix shells support the notion of ***job control***.
  - Allows users to move jobs back and forth between background and foreground.
  - Allows users to change the process state (running, stopped, or terminated) of the processes in a job.



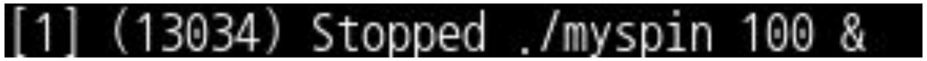
```
tsh> ./myspin 100 &
[1] (13034) ./myspin 100 &
tsh> jobs
[1] (13034) Running ./myspin 100 &
tsh> fg %1
```

- Signal commands
  - **Ctrl-C** : causes a SIGINT signal to be delivered to each process in the foreground job.



```
[2] (13116) terminated by signal 2
```

    4 The default action for SIGINT is **to terminate the process.**
  - **Ctrl-Z** : causes a SIGTSTP signal to be delivered to each process in the foreground job.



```
[1] (13034) Stopped ./myspin 100 &
```

    4 The default action for SIGTSTP is **to place a process in the stopped state,** where it remains until it is awakened by the receipt of a SIGCONT signal.

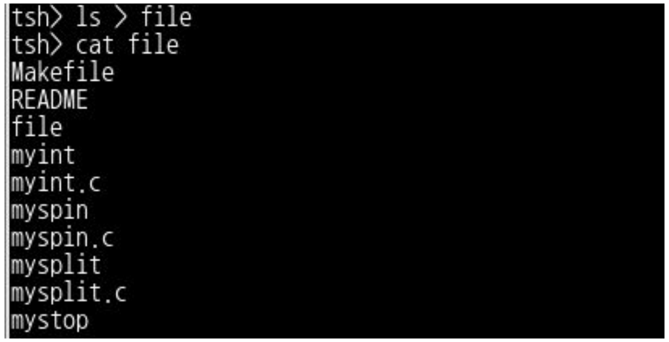
General Overview of Unix Shells (Cont.)

- Examples of built-in commands supporting job control.
  - *jobs*: List the running and stopped background jobs.
  - *bg <job>* : Change a stopped background job to a running background job.
  - *fg <job>* : Change a stopped or running background job to a running in the foreground.
  - *kill <job>* : Terminate a job.

The *tsh* Specification

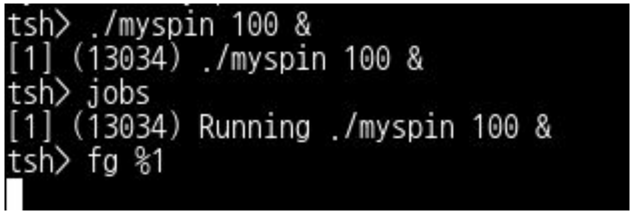
The *tsh* Specification

- Prompt string "tsh> ".
- Command line should consist of a *name* and optional arguments.
  - *name* : built-in command or the name path of an executable file.



```
tsh> ls > file
tsh> cat file
Makefile
README
file
myint
myint.c
myspin
myspin.c
mysplit
mysplit.c
mystop
```

- I/O redirection(>).
- Signal handling
  - Ctrl-C : cause a **SIGINT**
  - Ctrl-Z : cause a **SIGTSTP**
  - To be sent to the current foreground job, as well as any descendants of that job.
- If the command line ends with an ampersand &, ***tsh*** should run the job in the background. Otherwise, it should run the job in the foreground.

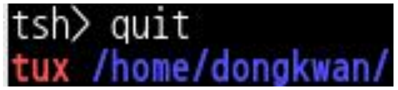


```
tsh> ./myspin 100 &
[1] (13034) ./myspin 100 &
tsh> jobs
[1] (13034) Running ./myspin 100 &
tsh> fg %1
```

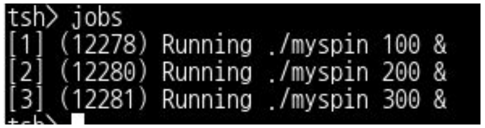
- A process ID (PID) and a job ID (JID)
  - Assigned by ***tsh***.
  - JIDs should be denoted on the command line by the prefix '%' (e.g. "%5")

The *tsh* Specification (Cont.)

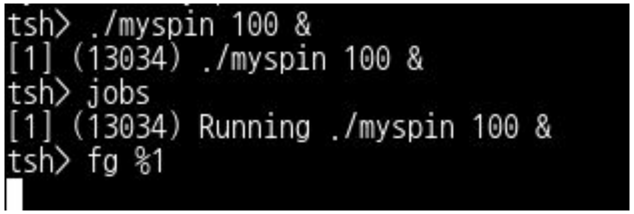
- ***tsh*** should support the following built-in commands.
  - *quit* : terminates the shell.
  - *jobs* : lists all background jobs.
  - *bg <PID or JID>* : restarts <PID or JID> by sending it a SIGCONT signal, and then runs it in the background.
  - *Fg <PID or JID>* : restarts <PID or JID> by sending it a SIGCONT signal, and then runs it in the foreground.



```
tsh> quit
tux /home/dongkwan/
```



```
tsh> jobs
[1] (12278) Running ./myspin 100 &
[2] (12280) Running ./myspin 200 &
[3] (12281) Running ./myspin 300 &
```



```
tsh> ./myspin 100 &
[1] (13034) ./myspin 100 &
tsh> jobs
[1] (13034) Running ./myspin 100 &
tsh> fg %1
```

- ***tsh*** should reap all of its zombie children.

The *tsh* Specification (Cont.)

- Function list of what do you need to implement in this lab with approximate number of lines in our reference solution code.
  - **eval** : Main routine that parses and interprets the command line. [70 lines]
  - **builtin_cmd** : Recognizes and interprets the built-in commands. [25 lines]
    4 *quit*, *fg*, *bg* and *jobs*.
  - **do_bgfg** : Implements the *bg* and *fg* built-in commands. [50 lines]
  - **waitfg** : Waits for a foreground job to complete. [20 lines]
  - **sigchld_handler** : Catches SIGCHILD signals. [80 lines]
  - **sigint_handler** : Catches SIGINT(ctrl-c) signals. [15 lines]
  - **sigtstp_handler** : Catches SIGTSTP(ctrl-z) signals. [15 lines]
- To run your shell, type ***tsh*** to the command line.

```
$:./tsh
tsh> [type commands to your shell here]
```

How to test your code

How to Test Your Code

- Reference Solution
  - ***tshref*** is the reference solution for the shell.
  - Your shell should emit output that is identical to the reference solution.
    - 4 Except for PIDs, of course, which change from run to run.
- Shell driver
  - ***Sdriver.pl*** executes a shell as a child process, sends it commands and signals as directed by a trace file, and captures and displays the output form the shell.

```
$: ./sdriver.pl -h
Usage: ./sdriver.pl [-hv] -t <trace> -s
<shellprog> -a <args>
Options:
  -h            Print this message
  -v            Be more verbose
  -t <trace>    Trace file
  -s <shell>    Shell program to test
  -a <args>     Shell arguments
  -g            Generate output for autograder
```

How to Test Your Code (Cont.)

- 18 trace files (*trace{01-18}.txt*) provided
  - From very simple tests to more complicated tests.
- To compare your result with the reference shell using trace driver

```
$: ./sdriver.pl –t trace01.txt –s ./tsh –a “-p”
```

```
$: make test01
```

- To compare your result with the reference shell using trace driver

```
$: ./sdriver.pl –t trace01.txt –s ./tshref –a “-p”
```

```
$: make rtest01
```

How to Test Your Code (Cont.)

- Example

```
$: ./sdriver.pl -t trace10.txt -s ./tsh -a "-p“
$: make test10
./sdriver.pl -t trace10.txt -s ./tsh -a "-p“
#
# trace10.txt - Process fg builtin command.
#
tsh> ./myspin 4 &
[1] (29391) ./myspin 4 &
tsh> fg %1
Job [1] (29391) stopped by signal 20
tsh> jobs
[1] (29391) Stopped ./myspin 4 &
tsh> fg %1
tsh> jobs
```

Evaluation & How to Submit

Evaluation

- Maximum Score : **100 points**
- Programming Parts : **70 points**
  - **63 Points** : Correctness
    - 18 trace files (3.5 points per each).
  - **7 Points** : Style points.
    - **7 points** : useful comments & check the return value of **system call**
- Report : **30 points**
  - Report should include
    - Description of your implementation.
    - Difficulties and thoughts during the implementation of this lab.
  - There is no format for the content of the report but it must be a ***pdf*** file.
  - **Report must be written in English.**

How to Submit

- Make sure you have included your names and student ID in the header comment of tsh.c

```
/*
 * M1522.000800 System Programming
 * Shell Lab
 *
 * tsh - A tiny shell program with job control
 *
 * Name: <fill in>
 * Student id: <fill in>
 *
 */
```

- Push newest tsh.c.
- Upload your report in report directory(XXXX-XXXXX.pdf).

Thank you
Q & A