Lab 9 : Shell Lab

What is shell?

- A program **running applications** on behalf of user and managing them

- Common example: **Bash(**Bourne-again shell**)**
  - Linux default

- Most applications in linux(command line) are run through shell

What is your assignment?

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

Tsh basic functionality

1. 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 processes

Tsh basic functionality

2. 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

3. 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

- 4. Background/foreground management
  - Change job status (bg to fg/ fg to bg)
  - E.g. tsh> fg <job_id>

    -> Makes a background job run as foreground

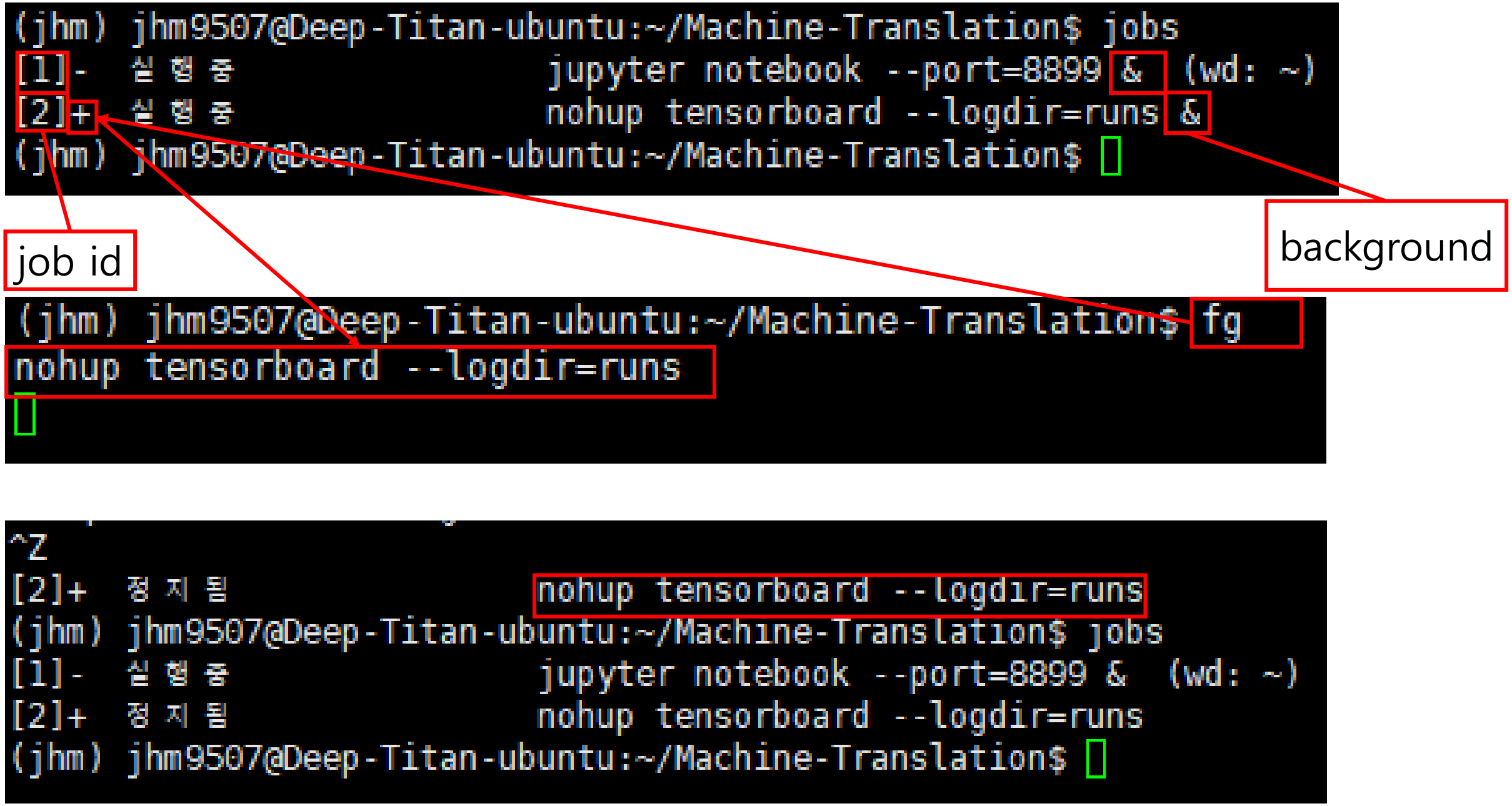
*Tsh can move jobs between foreground and background

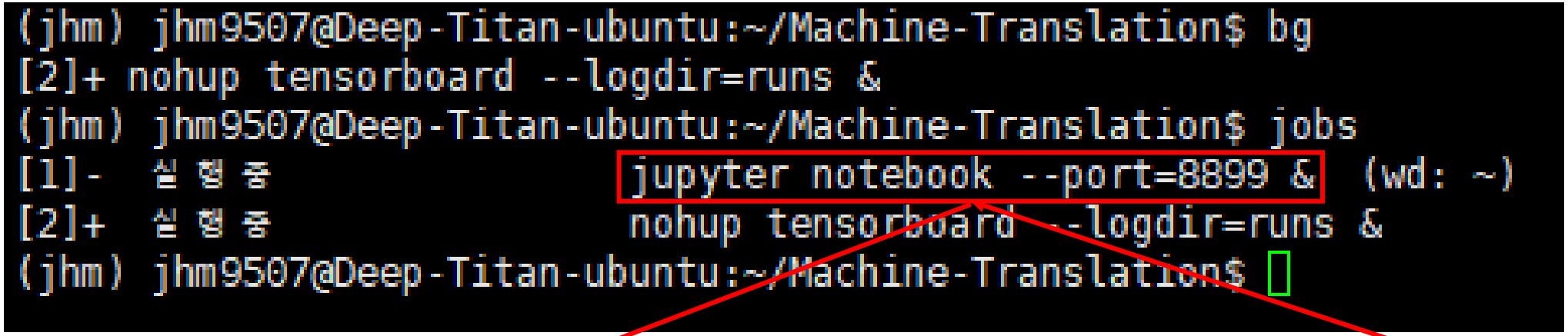
Tsh basic functionality

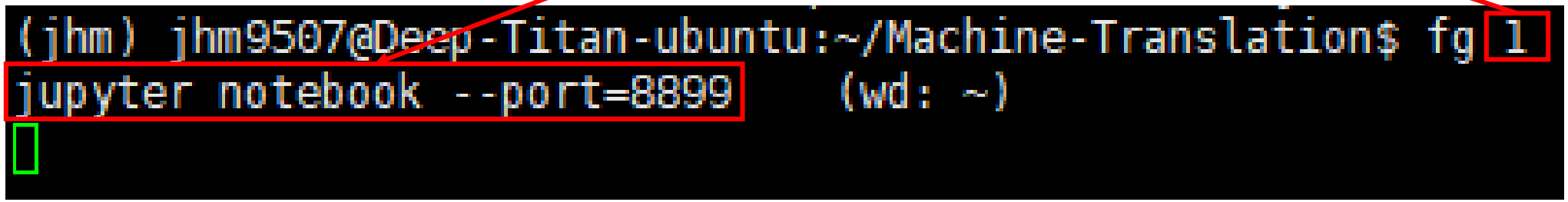
- 5. Prints list of jobs
  - E.g. tsh> jobs

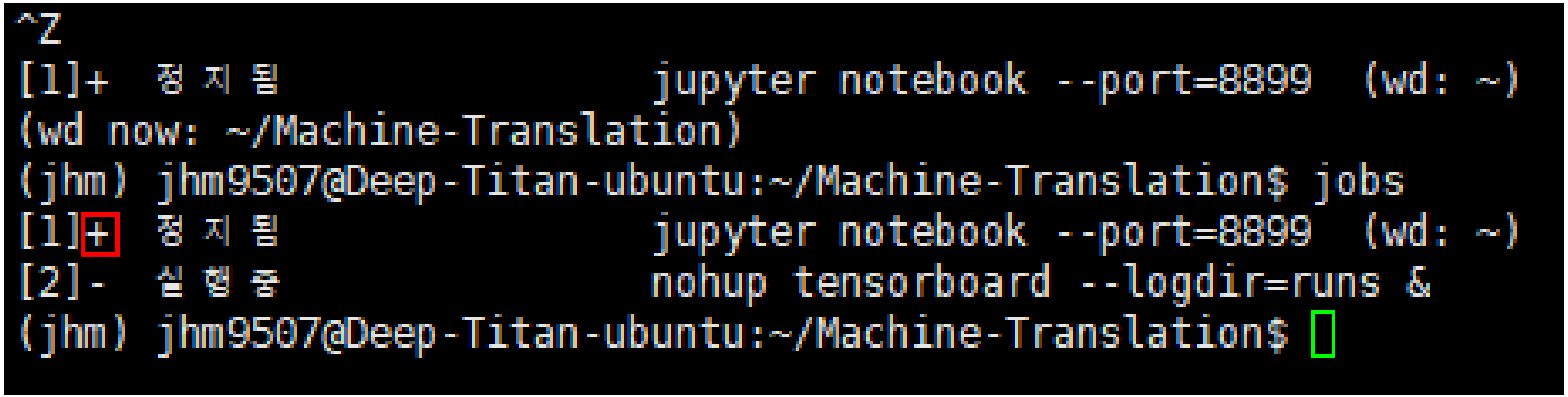
    -> Prints list of jobs including both running and stopped.

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


(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$ jobs
[1]- 실행 중 jupyter notebook --port=8899 & (wd: ~)
[2]+ 실행 중 nohup tensorboard --logdir=runs &
(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$
job id
background
(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$ fg
nohup tensorboard --logdir=runs
^Z
[2]+ 정지됨 nohup tensorboard --logdir=runs
(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$ jobs
[1]- 실행 중 jupyter notebook --port=8899 & (wd: ~)
[2]+ 정지됨 nohup tensorboard --logdir=runs
(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$


(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$ bg
[2]+ nohup tensorboard --logdir=runs &
(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$ jobs
[1]- 실행 중 jupyter notebook --port=8899 & (wd: ~)
[2]+ 실행 중 nohup tensorboard --logdir=runs &
(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$


(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$ fg 1
jupyter notebook --port=8899 (wd: ~)


^Z
[1]+ 정지됨 jupyter notebook --port=8899 (wd: ~)
(wd now: ~/Machine-Translation)
(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$ jobs
[1]+ 정지됨 jupyter notebook --port=8899 (wd: ~)
[2]- 실행 중 nohup tensorboard --logdir=runs &
(jhm) jhm9507@Deep-Titan-ubuntu:~/Machine-Translation$

Work in this assignment

- Many helper function coded for you
  - Parseline
  - Addjob, deletejob, clearjob
  - fgpid
  - getjobpid, getjobjid…
- 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, fg, 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. Unix> ./sdriver.pl –t trace01.txt –s ./tsh –a “-p”
   or Unix> make test01
2. Unix> ./sdriver.pl –t trace01.txt –s ./tshref –a “-p”
   or Unix> make rtest01
3. If the result of (1) and (2) match, then your code is correct.

```
(jhm) jhm9507@Deep-Titan-ubuntu:~/TA/CMU_Source/shlab/shlab-handout$ make rtest04
./sdriver.pl -t trace04.txt -s ./tshref -a "-p"
#
# trace04.txt - Run a background job.
#
tsh> ./myspin 1 &
[1] (7851) ./myspin 1 &
(jhm) jhm9507@Deep-Titan-ubuntu:~/TA/CMU_Source/shlab/shlab-handout$ make test04
./sdriver.pl -t trace04.txt -s ./tsh -a "-p"
#
# trace04.txt - Run a background job.
#
tsh> ./myspin 1 &
[1] (7857) ./myspin 1 &
```

You must type

Unix> make clean

Unix> make

After writing some codes in tsh.c

Tutorial

- Unix> tar –xvf shlab-handout.tar
- Unix> make
- Unix> ./tshref # You can test tsh
- tsh> /bin/ls
- tsh> /bin/echo “hello world”
- tsh> myint 1
- tsh> myspin 1
- tsh> mysplit 1
- tsh> mystop 1
- tsh> jobs
- tsh> fg %1
- tsh> quit # or Ctrl d

Tutorial

- Unix> make rtest01
- Unix> make test01 # the results should be same

- Unix> make rtest02
- Unix> make test02 # should be stopped

Assignment Submission

- Complete your tsh.c
- Rename to “student id”-tsh.c
  - Ex) 20192482-tsh.c
- Submit your renamed tsh code to LMS
- Due: Wednesday, November 27, 11:59PM

Hints

- Take a look at the textbook chapter 8.
- 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.

Any Questions?