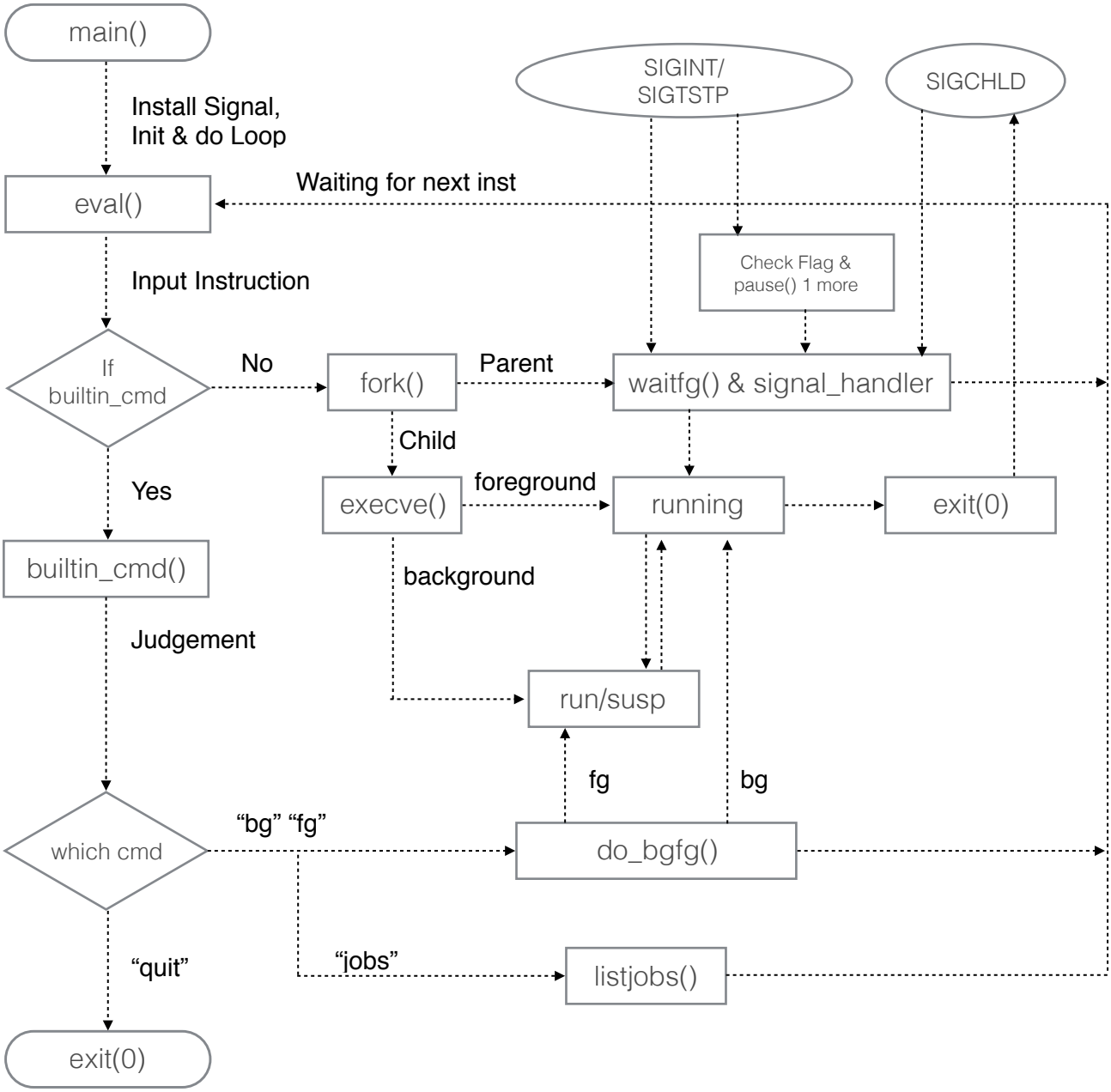
Shell Lab Report

Introduce

Name: CUI HANGQI (최항기)
Student-ID: 2016-27542
Lab: Shell Lab

Main Flow Chart



Process Introduction

In this lab, program start with signal installation, initialization and a while loop which waiting for a command line input.

When the shell received a command the eval function will first check whether the command is a built-in command or a path of file. If name is a built-in command, then shell will handle it immediately. Otherwise, the shell will fork a child process and execute the path by execve().

The parent process - shell, will waiting for foreground job until it gets a signal by pause(). If it gets the SIGCHLD, then waiting for next command. Or if it gets the SIGTSTOP, the child process didn't terminated, so the parent should pause 1 more time.

Implementation

eval()

This function will do such things:

1. execute parseline() to get whether the command is background or foreground.
2. execute builtin_cmd() to check whether the command is built-in command or path of an executable file.
3. If it is Not a built-in command, then fork a child process and run execve().
4. If the command run in background, add job to the jobs by addjob(). Or add job then keep waiting by waitfg().

builtin_cmd()

In this function, 3 if-else expression will judge the first argument of the command.

1. If it is "quit" then exit(0).
2. Or if it is "jobs", execute listjobs(jobs).
3. Else it is "bg" or "fg" execute do_bgfg(argv).

do_bgfg()

For test14, this function needs some error handling.

1. Check arguments exist by (NULL == argv[1]).
2. Check the first character of the argument is '%' or a number.
3. If the argument start with '%' thus it is searching job ID.
4. Else the argument is searching for process ID.

5. Finally, send SIGCONT signal to the process group, lei it continue. and set state to BG or FG.If the command is “fg” parent should wait.

waitfg()

This function execute pause() function, waiting for child process terminate and send SIGCHLD signal to parent process, then waiting for next command line.

If the parent process received a SIGTSTOP signal before the child process terminate, then the parent process will pause() 1 more time.

sigchld_handler()

This function is a handler for SIGCHLD which installed in main function.

First, the function should get the pid by waitpid.

Then, check the status type.

1. If it’s stop signal, then print “Job [$jid] ($pid) stopped by signal $sig”.

2. Else if it is interrupt signal, then print terminated information and delete the job.

3. Otherwise, delete the job.

sigint_handler()

In this function, when parent process (the shell) get SIGINT, this handler will handle the signal and send SIGINT to the foreground process group.

sigtstp_handler()

In this function, when parent process (the shell) get SIGTSTP, this handler will handle the signal and send SIGTSTP to the foreground process group. And then set the job state to ST.

Problems & Difficulty

During the programming I got several confusing problems. Most problem solved in Hints, like set process group ID and kill -pid to send signal to the shell. The following are other problems.

1. At the beginning I’m used pause() function to waiting child process when I implement the waitfg() function. But I get a problem that every time `/bin/echo tsh> jobs` then `jobs’ after ctrl+z the echo output sometimes show behind the jobs output and the echo job finished last.

    - Reason: I’m not sure, but I think it is the following resone. In test08, when spinning ./myspin 5, the parent process (shell) is waiting for the next SIGCHLD, but SIGTSTP comes first. The shell deal with next command and wait, this time the previous process terminated and send SIGCHLD signal, when echo command didn’t finished. And the jobs command is faster. So the echo output show behind the jobs output.

- Solution: There are two solution. First, add a flag variable when received SIGTSTP signal pause 1 more time. The second is replace the pause() function with busy loop.

2. The second problem met when I treat with test16. I already set the stop signal and interrupt signal, but there are nothing output. After reading the source code, I found the reason.

   - Reason: The program mystop and myint send stop and interrupt signal to themselves. Hence the signal didn't send to the parent process (shell), so there is no output about the stop or interrupt.

   - Solution: The solution is that get the status of waitpid in sigchld_handler, and check what kind of status is it. If it is stop signal then do not delete the job, just change the state. Otherwise, delete the job, and handle the stop and terminate output.