Thi Nguyen
Fall 2017
CS 475 – A1

1. Does your program include any code for parsing the ctrl-z character? If not, then how is the ctrl-z being processed?
My program doesn't include any code for parsing the ctrl-z character. The ctrl-z was interpret as a stop signal from terminal.

2. The user's intent in typing ctrl-z is to stop the foreground process. What does the OS do when the user types in a ctrl-z?
The OS suspends the most recent foreground process by sending SIGTSTP signals to the entire foreground process group.

3. What action should the shell take when it receives the SIGTSTP signal? Which function is invoked when the shell receives a SIGTSTP signal?
When the shell receives the SIGTSTP signal, it will stop the process that sends that signal to the shell.
The shell sends SIGTSTP to the entire foreground process group to stop the foreground job and will then invoke getjobpid(jobs, pid)->state = ST to change the state of that process to stop.

4. What actions are taken in the function sigchld handler? Go over each case that is handled by your code and describe the actions taken.
There are three actions taken in the function sigchld handler after it reap the terminated or stopped child with waitpid.

First, if child is terminated normally via exit or return, that is if (WIFEXITED(status)), it removes the child from the job list

Second, if the child is terminated when user type Ctrl-C, that is if(WIFSIGNALED(status)), it will also removes the child from the job list and print the message to announce that action.

Third, if the child is stopped by Ctrl-Z, that is (WIFSTOPPED(status)), the sigtstp_handler // send SIGTSTP to foreground job and the sigchld_handler will change that job state to ST.

5. Why do you need to use the WNOHANG and WUNTRACED options when invoking waitpid?
WNOHANG and WUNTRACED options are needed because I need to the waitpid() to return immediately with a return value of 0 if none of the children in the wait set has stopped or terminated, or with a return value equal to the PID of one of the stopped or terminated children.

6. Which function contains the code for reaping foreground jobs? How (and why) is this different from the code in your textbook (Figure 8.24 on page 755 of Computer Systems: A Programmer's Perspective, 3rd edition)?
The sigchld handler contains the code for reaping foreground jobs.

In the textbook, the wait is

```
 if (waitpid(pid, &status, 0) < 0)
          unix_error("waitfg: ;waitpid<~rror");
```

in my case, the wait is

```
while((pid = waitpid(-1, &status,(WNOHANG|WUNTRACED))) >0)
```

The difference is the while loop that keeps reaping all the terminated or stopped child and the options WNOHANG|WUNTRACED so that the waitpid() return immediately with a return value of 0 if none of the children in the wait set has stopped or terminated, or with a return value equal to the PID of one of the stopped or terminated children.

7. By examining the file /usr/include/wait.h, explain what the macros WIFEXITED and WEXITSTATUS do, i.e., look at the code for each macro and explain what it does.