1.eval:
The fromat of the input is:
tsh> first_part + space + argument1,2,3,...

Firstly,recognize the first_part is a executable file path or a buil_in command.
Executable file path is like: ***/***/***/name

buil_in command: quit,fg,bg,jobs (the command be recognized is only 4,"and" is not "&", just a word)

In the code, I forgot to mask the signal set.
There is an error in trace08. because I didn't mask the signals in this part, so sometimes it is right, but sometimes it is wrong.

In the code of the fork(), if the child finished firstly, it will sent the SIGCHLD to the parent, and the parent will delete the job. At this time, there isn't a job, because we didn't "addjob"

So, before the fork(), we should mask the signal set:
sigset_t mask: set a mask_signal_set named "mask"
sigemptyset(&mask): empty the mask set
sigaddset(&mask,SIGNAL_NAME): add the signal "SIGNAL_NAME" to the mask set
sigprocmask(SIG_B;OCK,&mask,NULL): set the mask as the current process set's mask list

Secondly, in the child:
I don't know why to setpgid(0,0), and the processes will be killed at once.
setpgid(0,0): set the current processes setgpid as the current pid.

When we fork(), the child and parent will have the same gpid. If we don't change the pid of the child, the SIGINT will sent to the parent and children,so the process will be killed.

2.builtin_cmd:
In this part, we should recognize 4 types of built_in command: quit,bg,fg,jobs
if it is "quit": exit(0) and process will quit,
if jobs: we want to print the joblist,
if bg or fg: we want to execute the do_bgfg function,
otherwise, it isn't a built_in command, return 0 to tell eval().

3 types return value: exit(0),0,1.

3.do_bgfg:
In this part, we handle the command bg and fg.
The command bg or fg needs the arguments(argv[]),so firstly identificate whether the arguments legal.
Secondly, identify the BG or FG.(srtcmp())
Then, recognize whether the pid or jid: the jid is like: "%+num" (argc[1] [0]=='%')
Set the state of the process, and call the relative function.

I was confused with the kill(pid,signal) and kill(-pid,signal):
kill(-pid,SIGNAL): -pid means that we will send the signal to all the processed in the process set gpid=="pid".

4.watifg:
Waiting for the forehead process.
sigsuspend(&mask):block the signals in the mask set. During this time, the backgroud process will be waiting.

5.sigchld:
In this part, the parents receive the signal from there children, and handle it.
Firstly, the parents identify the return number from there children:
WIFSTIOOED(status): the child is stopped by signal. So, set the state ST;
SIFSIGNALED(status): the child is quit by signal, we delete the job from the joblist and print the information.
otherwise, the child is quit normally, delete the job.

6.sigint:
In this part, send the signal to all the forehead process.
Firstly, recognize whether there is a forehead process,
If there is, kill(-pid,SIGINT)

7.sigstp:
(It is the easiest part in this lab)
In this part, we stop the forehead process
For each process, we recognize whether it is already stopped,
if it is already stopped(state==ST), ignore it,
otherwise, send the SIGTSTP to it.