The tsh.c program is to provides a basic command-line interface for users. This report summarizes my thought process and some key elements used to implement all those functions.

**eval function:**

* eval is for evaluating and executing user commands.
* It parses the command line and determine if the input should run.

**Implementation:**

* Use the parseline to parse the command line and extract command arguments.
* Check if the command is a built-in command using builtin\_cmd under two circumstances if it’s a built-in command or not.
* Properly manage signal masks using sigprocmask.
* setpgid to set the process group ID to handle fg and bg processes.

**builtin\_cmd function:**

* Uses to determine if a command is a built-in command and executes it if it is.

**Implementation:**

* Compare the first argument of the command (argv[0]) to a list of known built-in commands.
* Check the argument to execute if the argument matches.

**do\_bgfg function:**

* Uses to handle the bg and fg built-in commands.

**Implementation:**

* Parse the command.
* Use getjobpid or getjobjid to find the job provided.
* Send the appropriate signal to bring it to the foreground or background.

**waitfg function:**

* waitfg waits till the fg is done.

**Implementation:**

* Properly handle signals and signal masks to avoid race conditions.

**sigchld\_handler function:**

* It handles the signal when a child process state change.

**Implementation:**

* Call to waitpid to reap terminating child processes.
* Remove terminated job.

**sigint\_handler function:**

* Handles the signal which is sent when user calls to foreground job.

**Implementation:**

* Sends signal to foreground to process.

**sigtstp\_handler function:**

* Handles the signal when the user suspends a foreground job.

**Implementation:**

* Call to kill to send the signal to the foreground job process.
* Update the job list.