Lab 3

by Jarred Parr and Alexander Fountain

1. The order is waiting, received an interrupt, outta here
2. The program starts and prints that it is waiting, from there it pauses the main thread and puts a signal handler on the program. The program is now paused until it detects a system interrupt which, in this case, is handled via a `ctrl-c` on the keyboard. From there, it catches the interrupt and says that it was received. This leads to the final print statement of "outta here" which leads the system to exit the program gracefully.
3. The child process will output to `stdout`. This is because `dup2` was called before the child was initialized, so it will print to the default, which is `stdout`.
4. The child process will now output to `tmp`. This is because the dup2 remapped the output after the child was initialized and all of the subprocesses will print as a result of that as well.
5. The program starts and awaits input on `stdin`. It then sends it across the pipe to the waiting file descriptor to be read. After the data is read, it prints the data. Both of the pipe ends then close their descriptors and store the output to a `char*` variable which is then printed to `stdout` at the end. The parent process then prints an empty string because it did not catch the input from the `stdin` that the child received as the descriptor was not pointed at it properly.
6.

keyboard

display

Process

Child

stdin

stdout

W

R

W

R

File Descriptors
0 stdin
1 stdout
2 stderr

R-pipe
W-pipe

keyboard

display

Process

Child

File Descriptors
0 stdin
1 stdout
2 stderr

R-pipe
W-pipe

stdin

stdout

keyboard

display

Process

Child

stdin

stdin

stdout

stdout

File Descriptors
0 stdin
1 stdout
2 stderr

R-pipe
W-pipe