Signals Activity

The purpose of this exercise is to explore the use of signal and signal handlers.

Task 1.

1. Write a program called `greeting` that **efficiently** waits in an infinite loop for the next signal to be delivered.
2. Run your program in a terminal window.
3. Kill it with `SIGINT` by typing `ctrl+c` in the same window.

Task 2.

1. Run your `greeting` program again in the terminal, but in the background.
2. Run `ps` to confirm it is still running.
3. Run `fg` to move it to the foreground.
4. Type `ctrl+z` to stop it with `SIGTSTP`.
5. Run `bg` to continue executing the most-recently stopped process (with `SIGCONT`) in the background.

Task 3.

1. Open another terminal window and run `ps` again to determine the PID for this process. Do you need to pass any command-line arguments to `ps`? (Hint: `man ps`).
2. Use the `kill` command (from your new terminal window) to kill your `greeting` program.

Task 4.

1. Add a prototype to `greeting` for a signal handler called `sing` to your code.
2. Write the `sing` function so that it prints the "Happy birthday" song to `stdout`.

Task 5. When you run your `greeting` program it does not sing the song because the `sing` function never gets called. Write the code to install `sing` as the handler for the `SIGUSR1` signal.

Task 6.

1. Run `greeting` from one window.
2. Look up the PID from another window and send it a `SIGUSR1` signal.

Task 7.

1. Change `greeting` so that it expects a single command-line argument that will hold the name of the birthday boy or girl.

2. Change `sing` to use the name from the command-line argument. Note you cannot change the signature for `sing` (why?) — use a global variable instead.

Task 8.

1. Add an invocation to `sleep(10)` in the middle of your "Happy birthday" song in the `sing` function.
2. Compile and run your program.
3. Send your program a good number (say, 5-6) of `SIGUSR1` signals from another window, in quick succession.

How many times does the program sing? Why?

Task 9.

1. Run your program again, and send it a `SIGUSR1` signal from another window.
2. Before it finishes the singing, send a `SIGINT` (either from the other window, or from the same window using `ctrl+c`).

What happened? Why?

Task 10.

1. Change your program so that the `SIGINT` signal is **not** delivered to the program in the middle of the birthday song.
2. Repeat the previous task to confirm the song is finished before the program is killed.

Task 11. Now, let's rewrite our signal handler without using unsafe functions—`stdio` functions are unsafe to use in signal handlers (see lecture slides and `man signal-safety`, which includes a list of async-signal-safe functions). Here is one possible solution:

1. Add a global variable `sing_song`, initialized to `0`.
2. Rewrite the `sing` function so that it sets `sing_song` to `1`.
3. Modify your infinite loop in `main` so that whenever it resumes execution, it checks the value of `sing_song` and sings "Happy birthday" if it was `1`, and then sets `sing_song` back to `0`.
4. Test your program by running it one window, and sending it `SIGUSR1` signals from another window.
5. Now, try sending it `SIGINT` while it is in the middle of singing a song (as before, you will need to insert a `sleep` invocation in the middle of the song). Does our previous solution still work, to delay killing the program until after it has sung the song? Why?