Worksheet 6: C Terminal I/O

Updated: 25th February, 2020

Note: You are encouraged to keep notes on your observations and answers to the worksheet exercises and lecture examples.

Assistance is only given when students can demonstrate their own attempt of resolving any problems.

Make sure you include things that went wrong. Your notes will serve as evidence of attempt, and often you can learn a lot analysing things that did not work.

Your notes should include answers to the following questions:

- What programs or websites did you use in the practical today?
- What do you need to remember about your practical work today?
- How does your program work?

You should consult the lecture notes whilst attempting these exercises.

1. `stty`

A user from the shell can control the terminal settings by using the stty command.

Read the manual page and then try it out.

Note: Make sure you read about **`stty sane`** first - you will probablyneed it at some stage during these exercises!

2. `terminos`

A program can control the terminal settings by using **`tcgetattr(3)`** and **`tcsetattr(3)`** Most realistic programs that process one character at a time do not echo input back to the user, and capture/process some or all of the control characters.

(a) Inspect, compile and run `rotate.c` to determine what the program does. What happens when you hit `CTRL-C`? Is your input echoed? Is your input buffered?

(b) Try using `stty` to control the terminal process one character at a time and not echo input back to the user. Run rotate again.

> **Note:** Have a look at flags `icanon` and `echo` on the manual page

(c) Using `stty` is not an acceptable solution for most programs, it is better if the program itself controls the terminal. Revise `rotate.c` to eliminate editing and buffering so the program now processes each character as it is typed. Make sure your program restores initial terminal settings.

(d) Add the appropriate line to `rotate.c` to disable echoing as well. Examine the man page for `termio` to find the right flag name.

(e) Now modify your latest version of the program so that it does not wait for input. Specify an interval of about two or three seconds.

(f) The last thing is to handle is an interruption. When a user types `CTRL-C`, the default action is for the process to just halt without performing any cleanup. Now modify `rotate.c` to add a signal handler to catch the `INT` signal (`CTRL-C`); print the string `"Exiting..."` and do any clean-up: restore the terminal; close relevant files, etc before **`exit()`**.

3. (n)curses

`ncurses` replace the discontinued UNIX `curses` library. All programs using `ncurses` must call `initscr()` before using any of the library functions. This function can fail, so always check the return value.

Most other library functions do not actually directly modify the display; they just modify data structures held in memory. To actually reflect our changes on the screen, one of the `refresh()` functions must be called.

When finished the terminal must be returned to a usable state before the program exits. First, delete all of the created windows using `delwin()`, including the one returned by `initscr()`. Then, call `endwin()` and finally `refresh()` again, before exiting safely.

When you are learning curses development, you will almost certainly foul up your terminal with alarming regularity. On UNIX, you can type `stty sane` at the command prompt to return the terminal to a usable state without needing to log out. You will need to link to the ncurses library in order to use the functions in it. In UNIX, this is typically done using the -l compiler switch, e.g.:

```
[user@pc]$ gcc -Wall -o hello hello.c -lncurses
```

(a) Inspect, compile and run **`helloWorld09.c`**

(b) Revise **`helloWorld09.c`** so that a character will bounce across the screen rather than a message

(c) Modify **`helloWorld09.c`** so that the character does not get erased at every movement, but instead is replaced by a new character. That is there will be a trail everywhere the character moves

End of Worksheet