Assignment 4 - Morse Code Blinky Part 2

Nickoli Londura & Benjamin Martin - Group 7
Fall 2018
CS 444
Oregon State University

November 26, 2018

Abstract

This is the assignment 4 write-up for group 7 of CS 444. We explore how to build a kernel for the Raspberry PI on the os2 server, as well as learning how to modify the drivers for LEDS on the PI. We can make the LEDS present any string in morse code, be able to change the speed at which they blink, and have the string blink once through or keep repeating.

1 Write Up

1. What do you think the main point of this assignment is? Being able to build a kernel for the Raspberry PI, and also being able to modify the LED driver to have them blink words in morse code, be able to change the speed at which they blink, and have the string blink once through or keep repeating. It is important to know how to create and modify drivers when working with kernels.

2. How did you personally approach the problem? Design decisions, algorithm, etc. We took the contents from the heartbeat trigger and created our own trigger that uses morse code. We had to modify the Makefile and Kconfig in order to acommodate our morse trigger when building the kernel. We also had to create a file called `ledtrig-morse.c` which was the morse trigger code itself.

3. How did you ensure your solution was correct? Testing details, for instance. We started out by using print statements to see if the kernel was built and running properly. We then realized that we couldn't get anything to print to the console, even though we used the right function, but later on we figured out how to. When it came to testing if the LEDs are blinking correctly, we had to build the kernel every time to check and see if the morse code for inputs strings was correct, as well as the speeds were reflected by changes requested to the SYSfs system, and the amount of times the string is being blinked were also reflected. We were unable to fully complete the project as there was an issue with the kernel crashing after displaying the message once. This means that we have the 'once-off' settings permanently set. Sorry about that!

4. What did you learn? We learned how to build a Raspberry PI kernel. Specifically, we were able to create a morse code from the any input string and have the LEDs display it. We also learned how to have the user change the speed at which the LEDs blink, as well as having the user change the amount of times the string is blinked. Another thing we learned was how to create a character device that allows any kind of input string to be blinked in morse code, rather than have a hardcoded one.

5. How should the TA evaluate your work? Provide detailed steps to prove correctness.

1. Download the Raspbian OS by going to `https://www.raspberrypi.org/downloads/raspbian/` and choosing "Raspbian Stretch Lite"
2. Download Etcher by going to `https://www.balena.io/etcher/` and choosing one for your OS
3. Flash the Raspbian OS onto the SD card for the Raspberry PI using Etcher
4. Open the root directory of the SD card (by plugging it into its USB thingy and into your machine) and add `kernel=kernel8.img` and `enable_uart=1` to the config.txt file.
5. Ensure that the "code" directory is in the same directory as the "linux" directory. If no linux directory exists, it will get created in the directory above the present working directory.
6. Run the "builder" script to build and patch the kernel. If you get any prompts from the patcher, answer no ("n"). This means that you have previously built and are trying to build again, but the patcher will think to reverse the changes.
7. The image will be outside the linux directory (in the same directory as where "linux" and "code" are housed). It will be called "kernel8.img"
8. Copy the "kernel8.img" to the SD card

Here is how we created the patches (but you DO NOT NEED TO DO THIS TO RUN OUR CODE!)

1. After finishing our work on the `ledtrig-morse.c` file, we did `diff -u ledtrig-heartbeat.c ledtrig-morse.c >` to produce the `morse.patch` file.
2. We made a copy of the Makefile (in the trigger directory) and called it Makefileold. We modified the Makefile as necessary. Then we did `diff -u Makefileold Makefile > Makefile.patch` to produce the `Makefile.patch` file.
3. We made a copy of the Kconfig (in the trigger directory) and called it Kconfigold. We modified the Kconfig as necessary. Then we did `diff -u Kconfigold Kconfig > Kconfig.patch` to produce the `Kconfig.patch` file.

2 Version Control Log

acronym	meaning
V	`version`
tag	`git tag`
MF	Number of `modified files`.
AL	Number of `added lines`.
DL	Number of `deleted lines`.

V	tag	date	commit message	MF	AL	D
1		57351b4b4272e2e202f9c37236e6aa529d57d48d 2018-10-04	Initial test	4	118	
2		457cb43d9a5ffd145692bf8048d9f67837ef5b67 2018-10-04	Create README.md	1	1	
3		a0d31694c3643232027d1603ce90157f0aa4f7d0 2018-10-04	Create README.md	1	1	
4		7ed32fd0a03d4cd228a052dc6e82d18036585619 2018-10-04	Small change	0	0	
5		789c79dd6550916482399238babfdb32c9a9bc26 2018-10-04	Added assignment folders	6	111	
6		73e30d4da7ba8a16076476ad9dbc6762dabe5842 2018-10-04	Added assignment folders	6	111	
7		f61afcd4df3637188959edd4269d9964eb01196a 2018-10-04	Organized	10	4402	
8			0	0	0	
9			0	0	0	
10			0	0	0	
11		fa83ec19fc9d6e1c06b20b5edc4ee75f4a0d7b3a 2018-10-04	There we go	0	0	
12		06539913bd0fab683fe6cde3bd06cde97e9431f3 2018-10-10	Added concurrency code	1	35	
13		b5310109d285d732ccc2527efe50f124dc0415bc 2018-10-15	Added stuff	6	57	4
14		840df6290e155004b32584878ee1d12810aeff5e 2018-10-15	Finished concurrency assignment	11	2102	
15		35987c988887860b9d57168407068aa8864ea9d6 2018-10-15	Changed int in main and spaced code	1	19	
16		50b05095e10db4eee4bbd92773c721d7803f9f2d 2018-10-25	Added writing and things	13	505	1
17		e0c136046374c324bbd47f0723b1055ea4290abe 2018-10-25	modified code	6	477	
18		a87023fe93a19dbfe84e84a7374ded43bb4662db 2018-10-25	Finished concurrency project	2	156	
19		9ff7580e7ac32d8522936600a6cd6650cda6d7b5 2018-10-29	made the writeup latex	6	6829	
20		9036481e1bcef26af90e9b91e33e6f562c8177c1 2018-10-29	Mostly done?	10	670	
21		4729168de8c9cce8acb77f5b9baf6c5713864d04 2018-10-29	After assignment 2	14	2830	4
22		5f4d44aba72909a2a4a9d072166de25b30111c24 2018-11-12	Moved code and patch files into repository	4	506	
23		7ec298a2f7318ab4eb0c183c2af6aa13497319d7 2018-11-12	made latex file	4	511	
24		7295b1e68a2f97aca4fdde5a2d4fbab20bbbd444 2018-11-12	Added how to tar	14	2850	
25		1971a4e862078681cf793638f2f03a4d4399ea94 2018-11-26	Assignment4	7	675	

3 Work log

The majority of the project was completed on Tuesday, November 20, 2018. We saved the writeup and the character device portion on the due date (Monday, November 26, 2018). That made our work log real spicy for the final dayickoli did most of the work for the write-up and Ben did most of the coding for LED driver. This worked out really well because Ben hates writing. Our group will continue this work pattern in the future.