This project was a struggle. This project was a pain. This project taught me a great many valuable things that did not come easy.  
I’ll admit that the things I struggled most with were things I had learned before: pointer behaviors, basic string manipulation, how to declare structs, etc. I learned all this a while ago but coming back to it meant I had to relearn a good chunk.  
Beyond that, there were plenty of system calls I had never made use of before (or if I had, I was just following procedure and didn’t quite understand what I was doing). The next paragraph will be more detail on specific cases where I struggled, overcame, and learned in the process.

I didn’t know that there were system calls for managing the current program’s working directory. I had my own solution for managing the current directory under construction before I found out that there were already system calls (getcwd and chdir) that did what I wanted and managed everything under the hood. Using them was easy, but I had to learn of their existence first. Yes, I know they’re in the lab doc. I’m not good at paying attention.  
Ironically, the part I struggled with most might have been completely unnecessary (but it is very cool). I was bothered by the fact that I couldn’t pass quoted arguments and have them count as one argument. In what was the programming equivalent of scratching a hard-to-reach itch, I spent almost an entire day trying to work out a way to split a string by spaces EXCEPT when protected by double quotes. My early attempts had the mark of a Java programmer about them (guilty as charged), and none of that worked. My eventual solution came from taking a few steps back and really thinking about what a string IS. By adding to pointers and inserting null characters, I was able to get what I wanted. I still miss Java/C# strings though.

Now for my implementation of the extra credit. It’s quite simple, actually. I first implemented a linked list with two structs: one for the list and one for the node. Each node stores the process name, pid, and a pointer to the previous and next nodes (it’s a doubly linked list). The list stores the head node, tail node, and the number of total nodes. To manage this system I created three functions: addBGProcess, findBGProcess, and removeBGProcess. I won’t go into too much detail on these because they essentially follow the standard implementation of a doubly linked head+tail list. To use this system, all I do is add a new node at the end for each new background process, find and remove the corresponding node for each reaped background process, and designed a print function that goes through the entire list until it finds a null next pointer.