CSSE332 Final Project Report

Team 1E

Donglai Guo
guod@rose-hulman.edu
Zesun Yang
yangz@rose-hulman.edu
Zhihong Zhai
zhaiz@rose-hulman.edu

Date: Feb.04.2018

Section 1: Known Bugs and Deficiencies

When performing extra feature “clear”, shell does not show up in the first line, but show up in the second line. We have tried to fix that issue, however, we still cannot remove/fill the first empty line.

When changing foreground color, although text color will be changed, the screen will be cleared due to the type of interrupt we used.

Section 2: Special Features and Instructions

- Dir with file size printed
    - When user launches shell, and type “dir”, then not only available file names will display on the screen, file sizes in sectors will be printed on the screen as well.
- Clear screen command
    - When use launches shell and types “clear” , then the screen will be cleared, and the all text will changed to have red color (cool feature). The user can still use shell to implement commands.
- Help screen
    - When user launches shell and type “help”, then a table of all available commands will be printed out as well as instructions on how to use these commands.
- Quit command
    - When user launches shell and type “ quit”, then the screen will print “bye” and no longer able to execute command until a new shell is started.
- Background color changing
    - When user launches shell and type “changebg + color code), it will change the background color of shell to user specified color. Right now, our system supports 7 colors, blue, black, magenta, green, red, yellow, white. The color code is two characters. The color code is listed as below:
        - Blue: bl
        - Black : bk
        - Magenta: lm
        - Green : gn
        - Red : rd
        - Yellow : yw
        - White : wh
- Show simple graphics
    - Can show a simple pink square if user type “block”
- Foreground color changing

    - When user launches shell and type “changefg +color code”, similar to background color changing, the user then can change the text color.
- Snake
    - When user launches shell and type “snake”, a snake will show up in the shell. After the snake stops, type any key to return to shell.
- Scroll
    - User is able to scroll up and down using “w” and “s”, “w” for scrolling up, “s” for scrolling down. “q” to quit the help menu.
- Wait
    - In shell, we can type “wait” to wait for 1 second. In kernel, we can call delay(s) to delay for s seconds.
- Time
    - Type “time” in shell will return the time tick in hex of our internal clock.

Section 3: Interesting Implementations

❖ We added an integer argument served to store the sector number in ReadFile, and by doing so to help with our copy file. We can just use the sector number we stored in readFile, pass it to writeFile to make a copy of the targeted file. That way we do not need to go back and find the sectors, and thus save work and time.

Section 4: Lessons Learned

❖ Donglai Guo

➢ One of the most crucial things I learned is that it is really important to fully understand how the system flows and how subsystems depend on each other before actually start writing codes. The biggest problem we ran into in this project is debugging and we clearly recognized that smaller the bugs, harder to debug. Sometimes when bug happens, we could not do anything to it but setting up all sorts of print statement to check variables. However, this is not always efficient. If we have a decent system-level understanding, we could easily understand the bug and make proper modification to it. Thus, it is necessary to organize and get things in order beforehand because otherwise you don’t even know where the bug comes from. In addition, it is really amazing to apply what we learned in this course into a project and see how they work out. This hands-on experience helped a lot for me in understanding operating systems.

❖ Zesun Yang

➢ Before I started the project, when I thought about building my own operating system, I felt it would be hard to implement. However, when I actually started on it with my teammates, everything just flowed. I have learned how to make use of

interrupts to do multiple tasks (like printString, writefile, readSectors and etc.), and I have also learned to write my own shell. Everything we coded for this operating system is highly correlated to class materials, and it was a good enhancement to me. In addition to that, I really appreciate my operating system now.

- Zhihong Zhai
  - I learned that teamwork makes the dream work. Cooperation with team members is important to complete a great project. Arranging the workload among the teammates can improve the efficiency.

Section 5: Technical Knowledge Learned

- Donglai Guo
  - In this project, we built our own operating system from scratch, which helped me understand the bare bone structure of operating systems. The most import thing I learned from this project is Memory. Everything is based on memory including programs, file systems, system calls and shell commands. Memory(disk image) is also useful for debugging. When bugs occur, it is always not a bad idea to look into the image and see which sector of memory has been modified. Memory is also import in multiprocessing because it could determine how many processes could run concurrently.
- Zesun Yang
  - Through the project, I learned how important interrupt is in operating systems. Basically what we have done so far is indispensable from interrupts. I learned that interrupt 0x10 will call BIOS to perform I/O operations; interrupt 0x16 reads from keyboard; interrupt 0x13 reads/writes to disk... I have learned the concept of memory management and file systems in this project as well. The most important thing is the basic idea of multitasking I learned, which is finding new process to do and time out the current process. Creating a process table is a fast and efficient way for multitasking.
- Zhihong Zhai
  - I learned much about BIOS interrupts and how kernel is to interact with user programs. This project also helped me review the knowledge of assembly.