Andreas Palsson - palssoa@rose-hulman.edu
Andrew Davidson - davidsac@rose-hulman.edu

Bare Metal Operating System Reflection

1. Known bugs in our project,

There are currently only two known bugs in our project, but fortunately neither of them critically effect performance. The first bug allows users to try to copy a file that does not exist. If the name of the source file is invalid, the kernel still creates a new, empty file with the correct given output file name. This “feature” allows our users to quickly create a new empty file.

We were having repeated memory problems with process blocking during the last milestone, so we implemented a simple solution. Instead of being able to set the current process (whichever process it happened to be) to waiting with the execforeground command, the shell command is set to waiting by default. In this way, the shell and only the shell is always blocked by execforeground. This is not a current problem since only the shell can make command calls at the moment anyways, and is thus the only process that can be blocked in the first place.

2. Special features

The special features we implemented are the following:

- help command

Typing ‘help’ into the shell will show the user a list of available commands and examples on how the commands are used.

- dir command

Our command to print directory also prints the file sizes in sector numbers to the console.

- ls/cp - linux style commands

Instead of only letting the user use “copy” and “dir”, he or she can also use the more Linux-like commands ls and cp.

- bg

This command changes the background color to green and the text color to blue. However, this change cannot be reverted. If the user chooses to change his or her background and text color, they cannot change it back to black and white.

- <enter>

The user can hit enter repeatedly from the command line to repeadely type shell. This is handy for clearing the terminal.

Andreas Palsson - palssoa@rose-hulman.edu
Andrew Davidson - davidsac@rose-hulman.edu

3. Implementation techniques

One problem that initially slowed us down was our inability to tell the value of integers. We would normally use printline statements, but they were unavailable. To translate integers to characters, we would get the first digit from a custom modulus method and then add 48 to its value and cast it as a character by ASCII value, since ASCII 48 to 57 represent 0 through 9. We also needed to use this technique to go the opposite direction for the killtask command, where we parsed an integer from a command argument. In this case, we subtracted 48 from the ascii value and only kept going if the value was in range.

On a different note, we departed from the standard solution for writing a file to the disk. Instead, we checked to make sure that there were enough open sectors in the map before checking the directory for an opening. This was helpful so that we wouldn't need to undo anything if we got to the end and found out that it wouldn't work out. It was also helpful that during our initial pass through the map, we collected the sector numbers of free sectors as we went and put them in an array. After we found a free directory opening, we went back to the map and set the sectors to "occupied" and wrote to the disk.

To further overcome our recurring problem of debugging and testing, we also created a massive text file and added it to the make file in order to test that our readSector method was working. This was necessary since all of our other files were just one sector up to that point. On a similar note, we quickly added the extra credit sector count feature for our dir command. Our reasoning, which paid off, was that we would need it in order to debug our project and make sure that it was working.

We also had issues with comparing file names of to each other when they were shorter than 6 characters. In order to overcome this, we found it helpful to replace all junk characters at the end of a word like spaces, dots, or new lines with the null character, 0x0.

Also, in order to handle a user pressing delete while we were reading a string, we printed the backspace in the terminal to go back to the previous character, then printed a whitespace to delete the character, and then printed another backspace to bring the cursor back again.

There was a problem with having complete strings in our code. We fixed this by creating char arrays with the exact size that we needed and then added each character to the array individually.

We also got stuck with our cursor not being reset to the left of the page after hitting enter. However, we were able to fix this by adding a '\r' after each new line character.

Andreas Palsson - palssoa@rose-hulman.edu
Andrew Davidson - davidsac@rose-hulman.edu

4, 5. Things learned
WE ARE VERY GRATEFUL FOR METHODS. We wish we would have paid more attention to abstraction and cohesion. We should have focused earlier on keeping our code cleaner and easier to debug by breaking off large routines into their own separate, reusable methods. Tech debt rapidly became a bigger issue in our project. As a result, a couple of routines had to be re-written.

This project taught both of us a number of new things when it comes to understanding the most fundamental software written for computers. Instead of only learning theory, this way of actually implementing features discussed in class improved our understanding of the technical details and difficulties of creating an operating system. It also gave us an understanding of ANSI C and how to work around problems when we're not given the standard libraries that C offers.

It also taught us to respect memory allocation and how to handle pointers, since we were never given any error messages or warnings. We encountered several problems where we accidentally overwrote other variables' memory. Due to this, we are now a lot more careful when handling pointers, as to not overwrite anything in memory.

Since we were working a lot with svn, our usage and understanding of the importance of version control has improved. This will without a doubt be a valuable skill in the future.

We both definitely feel that working as a team has been a lot easier towards the end of the project, since we learned how to cooperate and how to take advantage of each others' strengths. We also found that it was helpful for us both to work individually but in the same room since we both worked at varying speeds depending on the problem. It helped to meet after working independently on it in order to check one another's code. This made us both understand the problems and difficulties of a milestone before meeting to solve the problems given. It also helped since we would usually each take a different approach and encounter different problems; by combining our solutions, we created something that actually worked. Additionally, we worked faster by ourselves. In cases where one of us arrived with more preaparation, the other would quickly fall behind, get lost, and be unable to help debug.

This class and project differs from the classes I've taken in Sweden in a number of ways. The greatest difference is the importance of always getting to practice what we've been taught through labs and milestones, whereas classes in Sweden tend to be a bit more theory-oriented and the students are not expected to practice as much as they are here. Thanks to this teaching approach, I feel like I have gained an increased understanding of the course material, and I feel like I know both how and an operating system works and how to implement some of its basic features. -- Andreas