Core Functionality Bugs

1. Bug: System calls page fault every time.

Solution: Redesigned the system call linkage.

1. Bug: Shell prints “391OS> ” prompt non-stop.

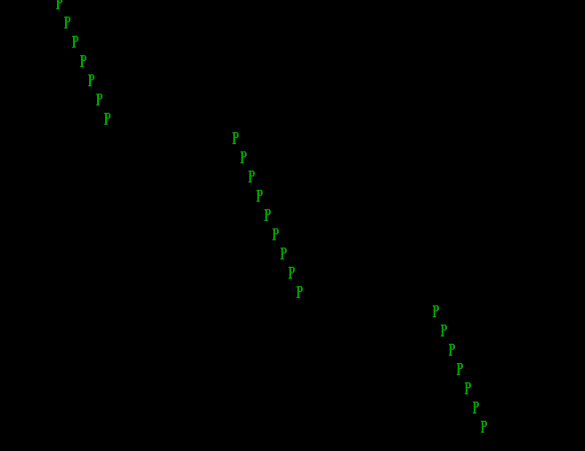
Solution: Used GDB and found that “terminal\_read” function is not blocking to wait for an “ENTER” press, added a loop to block.

1. Bug: Fish is not running correctly, prints “/” non-stop.

Solution: Used GDB and found that “read\_data” function is returning a wrong value when the file reached the end.

1. Bug: Terminal switch sometimes generates page fault.

Solution: Redesigned the switching algorithm and added a few condition checks.

1. Bug: Scheduler is not working correctly, a lot of weird problem happen randomly. 

Solution: Redesigned the whole scheduler using a new round-robin style design but didn’t use a run queue, specially optimized for our system only.

1. Bug: In execute syscall, we used a pointer in the inline assembly part, but it caused page fault when processes halts.

Solution: After using GDB and found the problem, we decided to use local variables to get pass the bug.

Miscellaneous/Extra Functionality Bugs

1. Bug: System some time stop functioning.

Solution: Some of the extra stuff are RTC invoked or special key invoked, but we forgot to send EOI.

1. Bug: The history commands are not stored and displayed correctly after the scheduler started working.

Solution: We forgot to switch to the correct terminal to store and display the history commands, so we added a flag to make sure it is the currently displaying terminal.

1. Bug: Backspace and delete only clears the VGA memory but not the terminal buffer when buffer is full.

Solution: Used GDB and found that the buffer index was not moving in sync with left/right keys press when buffer is full. So we re-designed the part to make them move together.

1. Bug: Ctrl + C stops processes randomly (i.e. may be the process in the background).

Solution: Same reason as the history commands, but because we are not using signals for this feature, we spent a long time to redesign the invoke method.

1. Bug: We are getting random page fault after displaying an image in Mode X and returning to text mode.

Solution: The array that stores the entire image data was too big (320 \* 200 Bytes), and sometimes it may change the data it shouldn’t be changing. So we changed the array to only store one row (320 Bytes) of the image data.

1. Bug: Status bar caused a few problems with our existing features.

Solution: We added a lot more condition checks to make sure things displaying on the terminal and status bar does not interfere with each other.

1. Bug: Matrix code rain had a few bugs, such as the back ground color was not set correctly due to the “terminal\_change\_color” function. And a few others.

Solution: We designed two functions that will save/restore the current state of all three terminals and disabled the scheduler.

1. Bug: Insert and overwrite cursor was not displaying correctly.

Solution: We tried a few values and found a perfect combination.

1. Bug: Cursor disappeared in terminal 2 and 3.

Solution: The cursor scan line of terminal 2 and 3 does not start with 0, have to count the scan lines of terminal 1.

1. Bug: The images are not displaying correctly in Mode X.

Solution: There were quite a few things related to this part. The first thing is we forgot to set the plane to write, so everything got messed up. Then it was the bmp file read, the bmp file has a weird data order: the palette data were in the order of BGR, so we have to flip the order, and there’s a null between two palette colors; the data starts from the last row of the image, so we have to start displaying our image from the bottom.

1. Bug: Writable file system crashed the whole file system.

Solution: We shifted the data blocks one by one instead of all together.

1. Bug: Write to file always writes an extra space.

Solution: It was the user level program we made that sent an extra count by accident.