Group Name: Snoop-Diggity-dOS
Members: Nathan Beauchamp (beaucha3), Rishi Thakkar (rrthakk2), David Desberg (desberg2), Matthew Faust (mhfaust2)

Buglog

Checkpoint 1:

- **Bug #1**
  - **Characterization:** Interrupts are not being handled correctly.
  - **Cause:** Pointers to the interrupt handlers are not being loaded correctly into the IDT. The reason for this is that we were incorrectly using C to reference a label in x86 assembly language. Labels are treated as raw values when referencing them in C. We were treating them as a pointer and due to this we were storing the incorrect address to the functions.
  - **Date Fixed:** 10/15/16 10:47 p.m.
  - **Solution:** By applying the address-of operator (&) to the label, we were able to store the correct function address.
- **Bug #2**
  - **Characterization:** Page fault occurs when we turn paging on.
  - **Cause:** In our page directory entries, we are setting the incorrect bit in order to indicate the Page Size. Specifically, we set bit 6 when we wanted to set bit 7.
  - **Date Fixed:** 10/16/16, 3:26 a.m.
  - **Solution:** Set the correct bit in the page directory entries.
- **Bug #3**
  - **Characterization:** Interrupts cannot be interrupted by higher priority interrupts, but they can grab processor control from exception handlers.
  - **Cause:** Incorrect usage of interrupt gates and trap gates within in the IDT entries for all interrupt handlers and exception handlers.
  - **Date Fixed:** 10/20/16, 9:44 p.m.
  - **Solution:** Using interrupt gates for exceptions and trap gates for system calls and interrupts.

Checkpoint 2:

- **Bug #1**
  - **Characterization:** Once 80 characters are printed on a line, the line doesn't wrap around and continues to print on the same line.
  - **Cause:** In putc(), screen_x is being updated before screen_y and updating screen_y relies on the old value of screen_x.
  - **Date Fixed:** 10/22/16, 6:06 a.m.

    - **Solution:** Change the order in which the updates occur.
- **Bug #2**
    - **Characterization:** Upon reaching the end of a line, the cursor doesn't wrap around correctly.
    - **Cause:** Updating the cursor location in multiple places. This caused conflicts in where the cursor should actually be.
    - **Date Fixed:** 10/23/16, 4:09 p.m.
    - **Solution:** Removed the redundant code that was setting the cursor for a second time. We changed it such that the cursor would only be set in putc_kbd().
- **Bug #3**
    - **Characterization:** When we switch to the ls, read_file_by_name, or read_file_by_index tests, the file/directory contents are continuously printed.
    - **Cause:** In our test code, there was no check to see if we had already printed the contents once.
    - **Date Fixed:** 10/23/16, 12:18 a.m.
    - **Solution:** We added flags to enforce the expected behavior of only printing the contents once. Specifically, we checked the flag to determine if we can print the contents- after the first time we had printed the contents, we cleared the flag so that we would not be continuously printing. This flag would then be set once we switch to a new test so that when we come back to the current test we can print the contents once more.

Checkpoint 3:

- **Bug #1**
    - **Characterization:** read_file() reads the same data on consecutive calls, with the same file descriptor as input.
    - **Cause:** The file position was not updated correctly in the read_file() function.
    - **Date Fixed:** 11/10/16, 1:22 p.m.
    - **Solution:** We added code to update the file position.
- **Bug #2**
    - **Characterization:** Executing a user-level program fails due to a page fault.
    - **Cause:** A logical error: In the execute() function, we didn't accounting for return address being on stack before performing IRET.
    - **Date Fixed:** 11/11/16, 12:00 a.m.
    - **Solution:** Added 4 to the ESP to account for this return address being on the stack.
- **Bug #3**
    - **Characterization:** If you ran ls 6 times in a row in the same shell, then it would not be able to run again. The ls user program would print "directory open failed".

    - **Cause:** This occurred because we were not clearing the fd entries for a program on either execute or halt. Since ls would have the same kernel stack as the previous ls, it would have the ability to open one less file every time we ran ls. Due to this, once we had run ls 6 times, we were not able to open any more files.
    - **Date Fixed:** 11/11/16, 12:53 p.m.
    - **Solution:** To fix this, we added a clear_fd_array() function that would be called in execute() to clear the fd entries associated with a program at the start of its execution. This way a program would always have a clear array of file_info_t structs to work with.
- **Bug #4**
    - **Characterization:** A TSS fault is generated upon the completion of a user-level program. This leads to an infinite page fault.
    - **Cause:** The code for marking a PID as available was incorrect.
    - **Date Fixed:** 11/14/16, 12:25 a.m.
    - **Solution:** We fixed logical errors in the functions that handle PIDs and process management.
- **Bug #5**
    - **Characterization:** The value returned by execute()- i.e., the value passed to it from the corresponding halt() system call- is incorrect.
    - **Cause:** The goto keyword in C places the jump address in EAX and then performs a JMP to the address stored in EAX. This would overwrite the value that we had put into EAX as the return address.
    - **Date Fixed:** 11/14/16, 4:51 p.m.
    - **Solution:** We stored the return address in the PCB of the corresponding program that is halting. Based on this, we were able to access the value upon returning to the correct location in the execute() function for the halting process.

Checkpoint 4:

- **Bug #1**
    - **Characterization:** A page fault exception is generated upon a user program call to getargs().
    - **Cause:** The pointer to the argument string (i.e., the string for a single argument) was not being advanced correctly relatively to the input command string.
    - **Date Fixed:** 11/11/16, 6:39 p.m.
    - **Solution:** We now advance the pointer to the argument string each time we walk down the argument string, as opposed to only when a space character is encountered (this is a problem as an empty space might never be seen).
- **Bug #2**

    - **Characterization:** A page fault exception is generated in syserr for the err_big_fd test.
    - **Cause:** We were not doing bounds checking on the fd input in read() and write().
    - **Date Fixed:** 11/14/16, 5:36 p.m.
    - **Solution:** Add the correct bounds checking to read() and write().
- **Bug #3**
    - **Characterization:** A page fault exception is generated when fish is executed.
    - **Cause:** In vidmap(), we only set the page table entry to be present, and not the page directory entry that points to that page table.
    - **Date Fixed:** 11/16/16, 4:47 p.m.
    - **Solution:** Set the present bit of the correct page directory entry.

Checkpoint 5:

- **Bug #1**
    - **Characterization:** After ALT key handling was added- once CTRL is pressed, no other keyboard inputs are processed.
    - **Cause:** Forgot to insert a "break" statement in the code for processing "command" keys in process_sent_scancode().
    - **Date Fixed:** 11/17/16, 9:49 p.m.
    - **Solution:** Insert a break statement in the switch-case.
- **Bug #2**
    - **Characterization:** When the user switches between terminals, the video memory does not update.
    - **Cause:** The TLB still contains the old video memory mapping, which causes the same text to remain on the screen.
    - **Date Fixed:** 11/19/16, 1:24 a.m.
    - **Solution:** After flushing the TLB entry associated with the video memory page via the invlpg instruction, the bug was fixed.
- **Bug #3**
    - **Characterization:** Keyboard input is being printed in the incorrect terminal.
    - **Cause:** Once scheduling is started, the page directories- and as a result, the video memory mapping- are constantly changing as context switches occur.
    - **Date Fixed:** 11/23/16, 12:07 p.m.
    - **Solution:** In process_sent_scancode(), we make sure to change the page directory to that associated with the process running in the current terminal. Then, once the printing is complete, we switch back to the previously loaded page directory.
- **Bug #4**
    - **Characterization:** After clearing the screen via CTRL-L, printing of STDIN occurs starting from an incorrect screen location.

    - **Cause:** Our CTRL-L handling in process_sent_scancode() was not modified to account for multiple terminals,
    - **Date Fixed:** 11/23/16, 4:13 p.m.
    - **Solution:** Create a small critical section and handle the currently selected terminal.
- **Bug #5**
    - **Characterization:** Shells associated with different terminals reading from the same buffer.
    - **Cause:** In read_terminal(), we were using the STDIN buffer associated with the currently selected terminal instead of the buffer associated with the terminal for the currently running process.
    - **Date Fixed:** 11/23/16, 5:35 p.m.
    - **Solution:** In read_terminal(), we read from the terminal associated with the current process- using the function get_terminal_of_current_process() - rather than using the current terminal state.
- **Bug #6**
    - **Characterization:** The cursor changes location when a print operation is executed by a user program running in a different terminal.
    - **Cause:** The code for setting the cursor location is a part of the putc_kbd() function. This causes a problem because it will set the cursor location regardless of which terminal the process making the write() system call is associated with.
    - **Date Fixed:** 11/23/16, 7:04 p.m.
    - **Solution:** Remove the code for setting the cursor from the putc_kbd() function, and place it in the write_terminal() function. This way, the cursor location can be set correctly on the screen regardless of which process is making the write() system call.
- **Bug #7**
    - **Characterization:** The system no longer functions correctly due to lack of interrupt generation.
    - **Cause:** Sending EOI in incorrect location for timer chip. Specifically, in IRQ0_handler_sub(), we checked for edge cases and returned before sending the EOI to the PIC.
    - **Date Fixed:** 11/23/16, 7:04 p.m.
    - **Solution:** Sending EOI before checking for edge cases fixed this problem.