

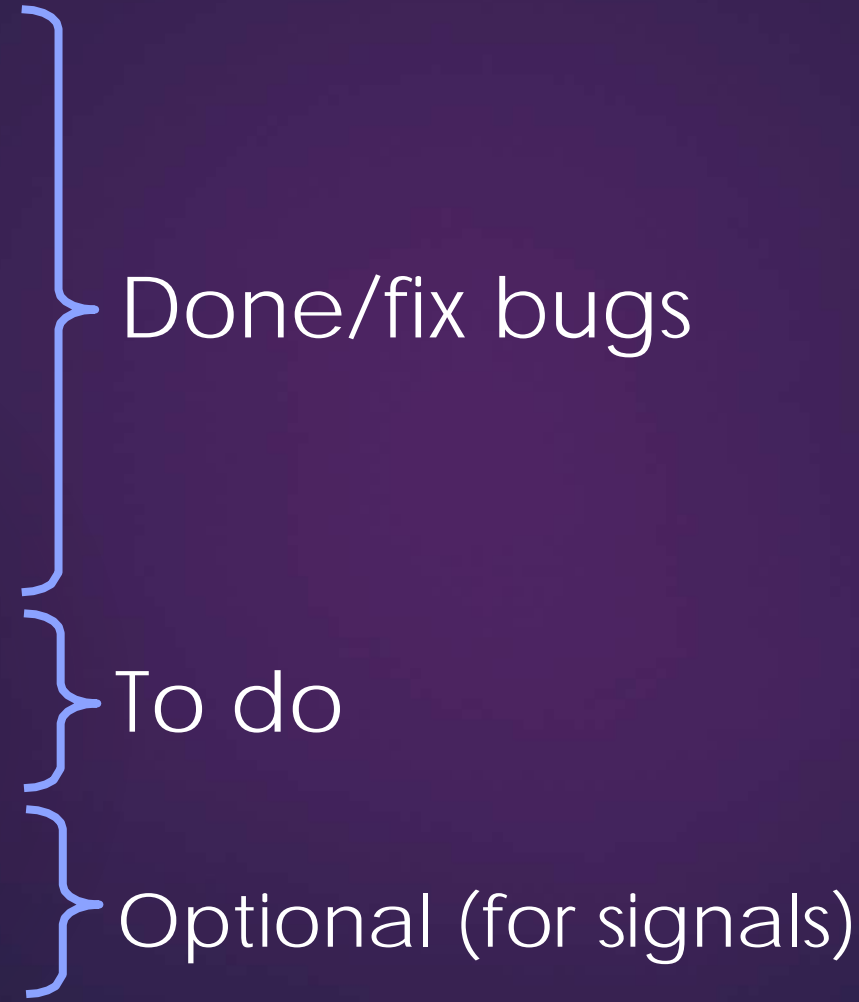
ECE 391 Discussion

Week 12

Announcements & Reminders

- u MP3.4 due the Tuesday after break (November 27) at 6pm
- u 4 weeks left (including break)
- u CP4 and CP5 Gradesheets and Final Demo Gradesheet will be posted today
- u Extra Credit Demo/Final Competition is on reading day

MP3.4: Finish Syscall Implementation

- 
- A diagram showing a list of 10 syscall implementation tasks. The tasks are numbered 1 through 10. To the right of the list, three curly braces group the tasks into three categories: 'Done/fix bugs' (tasks 1-6), 'To do' (tasks 7-8), and 'Optional (for signals)' (tasks 9-10). The tasks 'Set_handler' and 'Sigreturn' are highlighted in green.
- 1. Halt
 - 2. Execute
 - 3. Read
 - 4. Write
 - 5. Open
 - 6. Close
 - 7. Getargs
 - 8. Vidmap
 - 9. Set_handler
 - 10. Sigreturn
- Done/fix bugs
- To do
- Optional (for signals)

MP3.4: New Syscalls Quiz

```
int32_t getargs ( ?? );
```

- u What is buf?
- u What is nbytes?
- u What does getargs do?

```
int32_t vidmap ( ?? );
```

- u What is screen_start?
- u What does vidmap do?

MP3.4: Tips

- u Fix all bugs you might have
 - u Small bugs will cause you big trouble in MP3.5
- u Make sure all the user level programs listed below works
 - u testprint/hello/counter – terminal read/write
 - u syserr – bad system calls
 - u ls – opendirdirectory
 - u cat – open/read files and also use get_args()
 - u grep – open/close all files
 - u pingpong – RTC
 - u fish – vidmap, RTC, open/read files

MP3.4: given executables

- u Read the source! You must understand how each works for efficient testing/debugging
- u Summary:
 - u testprint/hello/counter – all print to the terminal
 - u Hello also does a read, counter runs for a long time
 - u syserr – series of malformed system calls
 - u Your kernel should handle these “gracefully” and pass the tests
 - u ls, cat, grep – do what they do in Linux, minus the bells and whistles
 - u pingpong – runs forever (might want to have a ctrl-c functionality)
 - u fish – animated fish (only executable using vidmap)
 - u Multi-block executable..

Task State Segment (TSS) and Halt

- u One TSS per CPU
- u GDT stores a pointer to the TSS
- u TSS contains all task-specific information (ss0/esp0)
 - u ss0 and esp0 are used when moving from user to kernel space
 - u ss0 = KERNEL_DS
 - u esp0 = Start of this process's kernel stack
 - u Save original value in PCB
- u Find TSS details in Intel manual Vol.3
- u In the halt system call, remember to close/cleanup open file descriptors

Have a nice break!

