NunyaOS

Week 10

Updates

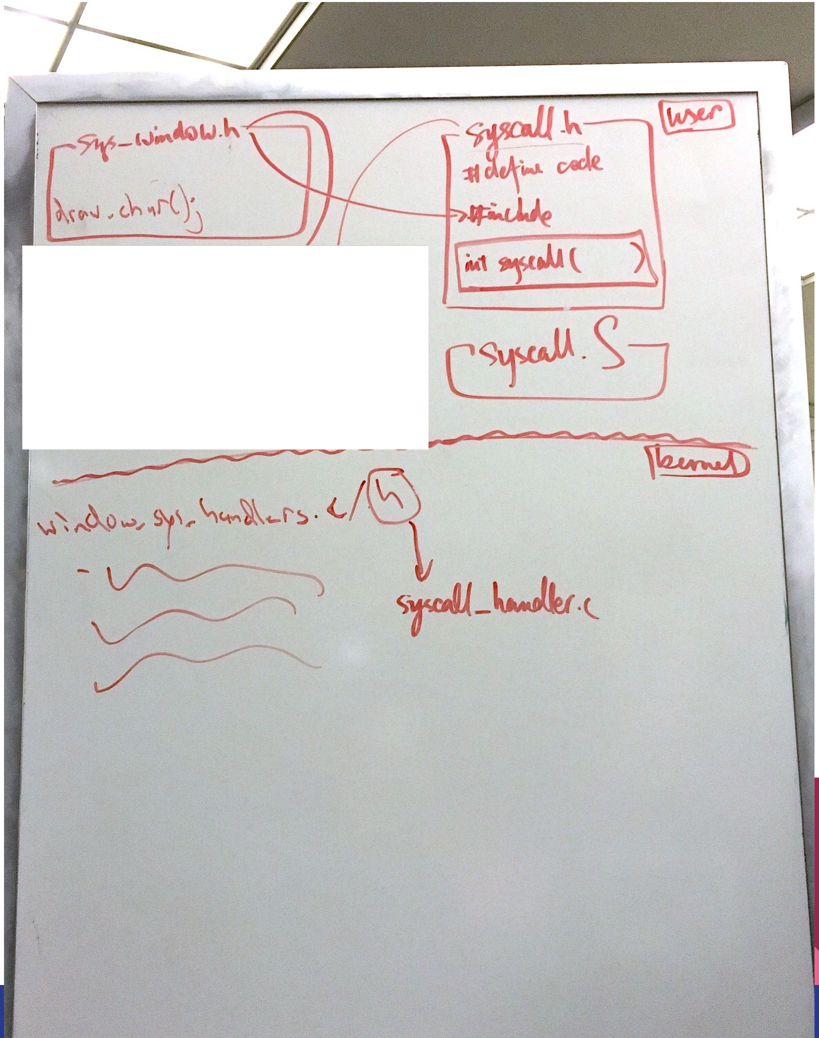
- Changed mouse cursor to a pointer design (Alex)
- Design run() syscall (everyone)
- Implemented run() and other process lifecycle syscalls (Zach, Shuyang)
- Decided on file organization for syscalls (Shuyang, Kyle, Ryan)
- Implemented file system syscalls (Ryan, Jesse)
- Discussed Window concurrency issues (Kyle, Alex)
- Implemented User event distribution (Kyle, Alex)
- Bug fixes (Jesse, Alex)

Run() syscall

- One syscall that does everything
- Takes a descriptor for a kernel-managed permissions template
- Todo: implement a suite of syscalls to create, edit these templates
- Use a global kernel descriptor table to keep track of the templates
    - Question on how to implement this – see end
- Permissions for each module:
    - Memory: a number of allowed pages
    - Fs: a list of directories/files that are allowed
    - Window: { max_width, max_height, origin (offset_x, offset_y) }

File organization

- Syscall.h contains all syscall numbers and the syscall prototype, and #include’s all sys_module.h files
- Sys_module.h contains inline syscall functions the user can access for the module
- syscall_handler_module.h/c contains handlers for syscalls per module



FS Syscalls

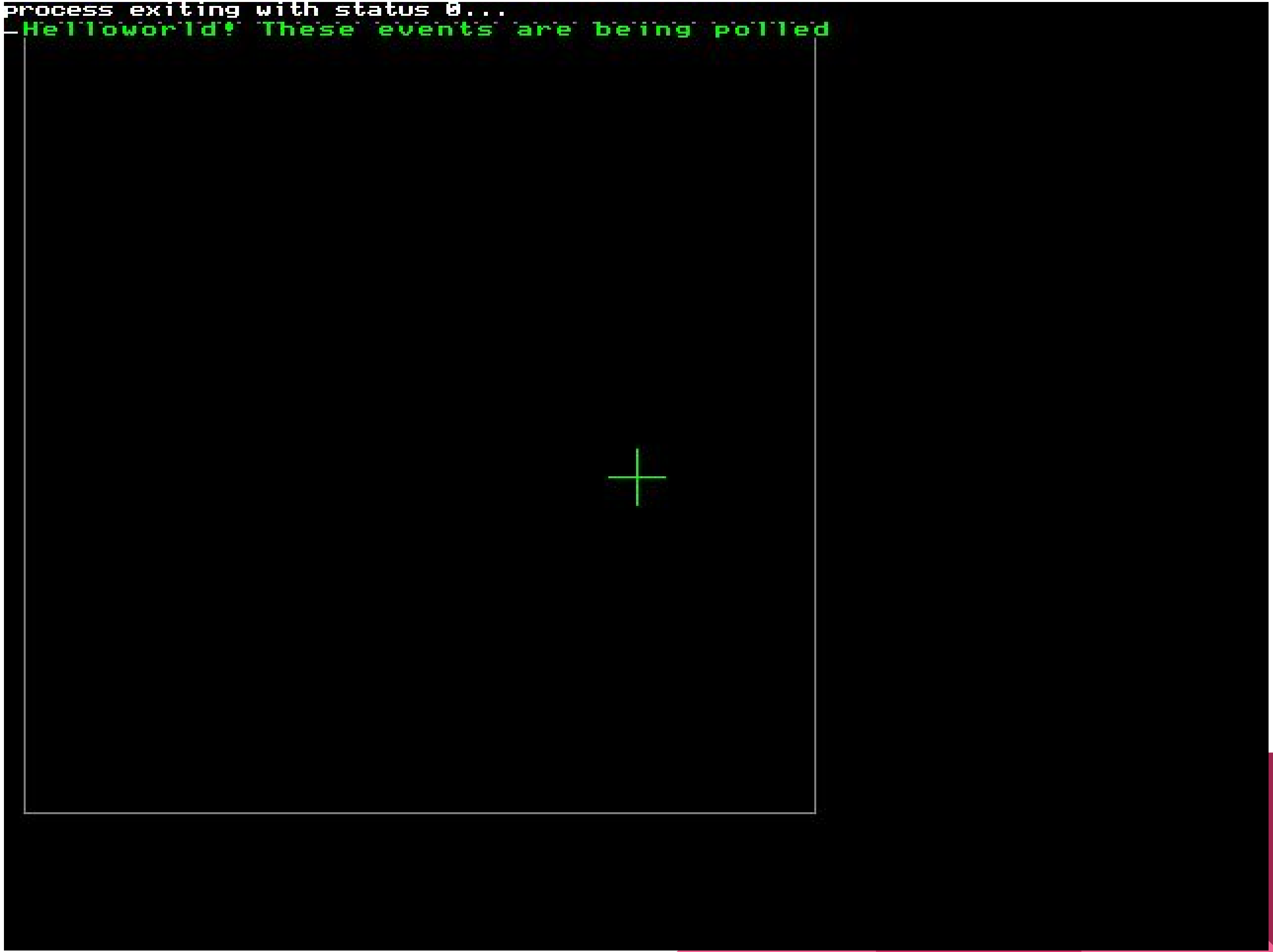
- PR is up, blocked due to other recent design decisions
  - File structure of code
  - Current implementation of run
- Lots of participation in CRs, design decision
- Jesse begins work on OS open file table
- Ryan begins work on code re-organization, interface with run

Window concurrency issues

- Decision between making all draw calls atomic and tightly coupling plot_pixel to a window
- Performance concerns with atomic draws -- unresponsive system?
- Leaning towards coupling plot_pixel to a window
    - Windows can track their offset and absolute bounding box

Event queue

- Keyboard and Mouse Interrupts create events
- Currently, mouse move and keyboard press
- These events are passed into a queue in each window, which each process reads from
- Processes poll events from their queue to react to



```
process exiting with status 0...
_Helloworld! These events are being polled
```

Bug fixes

- [NUN-20] CAPS LOCK simulates shift
- [NUN-31] 'W' cannot be typed
- Basekernel -> Nunya
- Mouse bug when redrawing near edge

Future Plan

- Finish run() syscall (and permissions templates syscalls)
- Implement syscalls for each module
- Write a few user programs

Questions

- Avoiding circular #includes in syscall.h - is this the best approach?
  - Syscall.h #include’s sys_process.h at the end
  - Sys_process.h uses syscall() but never includes syscall.h
  - Users only #include’s syscall.h, never sys_process.h
- How to implement permissions descriptor table
  - Dynamically or statically allocate?
  - If dynamic, how to do so without it counting toward a process’ limit?