SYSTEMS
GOSSIP
MEETUP

Learning large systems using peer-to-peer gossip

Policy Against Harassment at ACM Activities

https://www.acm.org/about-acm/policy-against-harassment

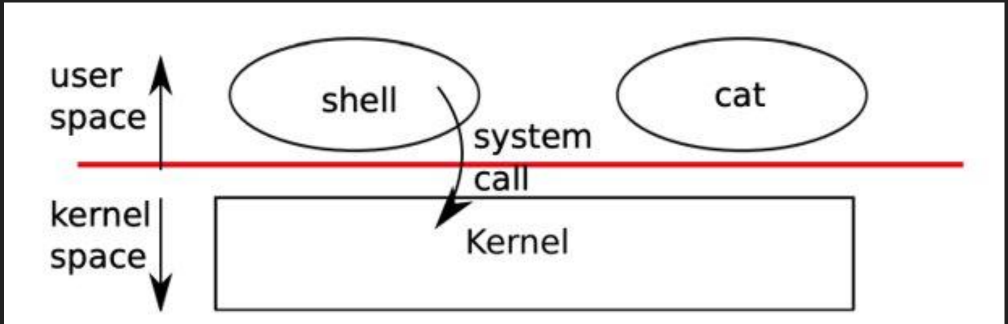
OS Meetup wants to encourage and preserve this open exchange of ideas, which requires an environment that enables all to participate without fear of personal harassment. We define harassment to include specific unacceptable factors and behaviors listed in the ACM’s policy against harassment. Unacceptable behavior will not be tolerated.

🎉Lab Utils🎉

Previously…

What is an Operating System?

1. Multiplexing (time-sharing)
2. Strong Isolation
3. Interaction (IPC)



Kernel, syscalls, and user space

`open`, `write`, `read`, `fork`, `wait`, `exec`, `pipe`

File descriptors (0 stdin, 1 stdout, 2 stderr) and per-process fd table

Previously…

Xv6 - a UNIX like teaching operating system

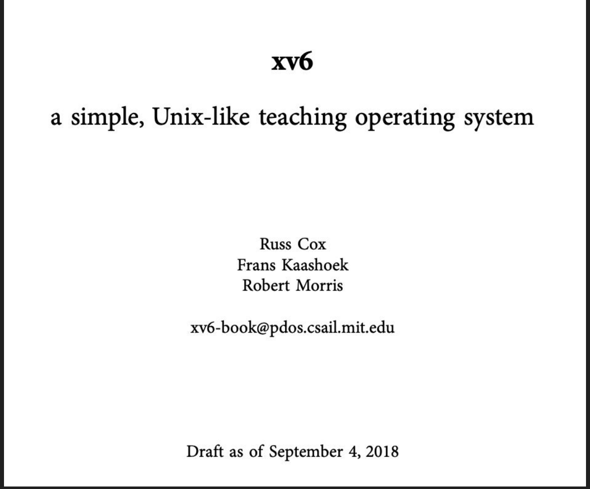
Xv6-risc-v is Xv6 running on RISC-V ISA

RISC reduced instruction set computing, e.g. ARM

CISC complex instruction set computing, e.g. X86

POSIX: portable operating system interface

“...POSIX-like Semantics…” “POSIX API…” “pthread…”



xv6

a simple, Unix-like teaching operating system

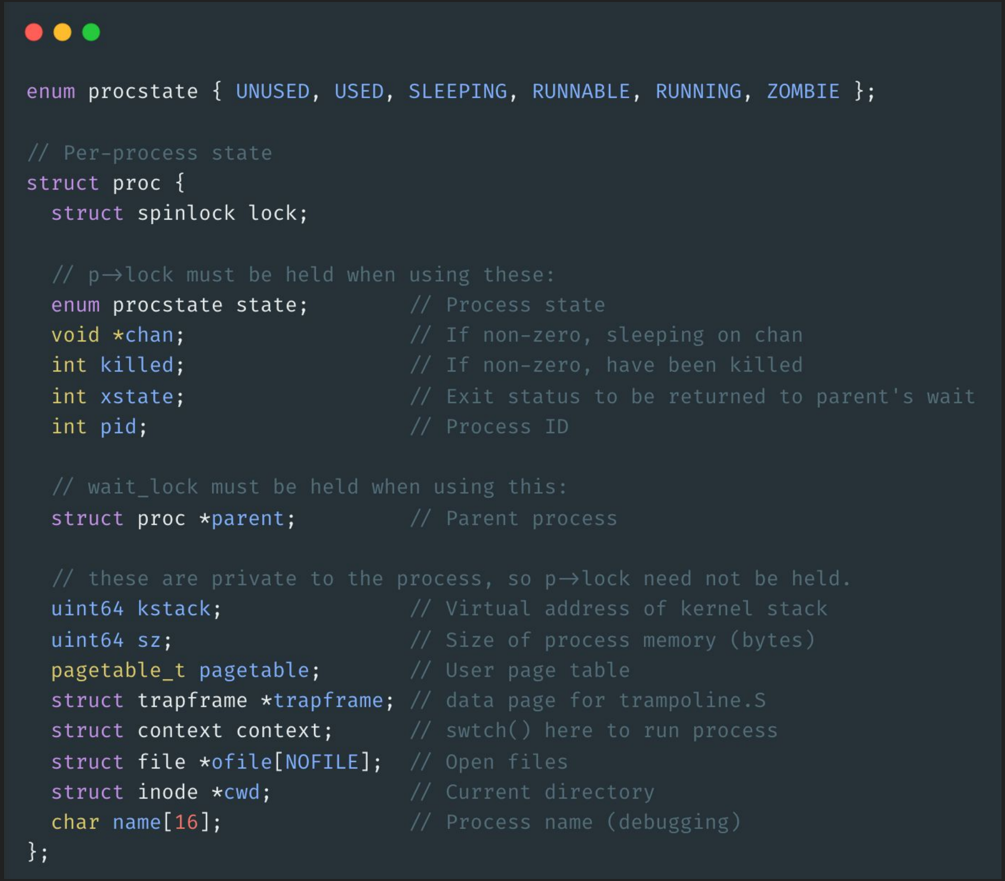
Russ Cox
Frans Kaashoek
Robert Morris

xv6-book@pdos.csail.mit.edu

Draft as of September 4, 2018

Process

- “The unit of isolation in Xv6 is a process”
- Virtualized CPU
- How can we virtualize CPU to create an illusion that we have many CPUs running at the same time?
- Time sharing (multiplexing): we run a process for a few milliseconds, and then another process for a while, and so forth



```c
enum procstate { UNUSED, USED, SLEEPING, RUNNABLE, RUNNING, ZOMBIE };

// Per-process state
struct proc {
  struct spinlock lock;

  // p->lock must be held when using these:
  enum procstate state;        // Process state
  void *chan;                  // If non-zero, sleeping on chan
  int killed;                  // If non-zero, have been killed
  int xstate;                  // Exit status to be returned to parent's wait
  int pid;                     // Process ID

  // wait_lock must be held when using this:
  struct proc *parent;         // Parent process

  // these are private to the process, so p->lock need not be held.
  uint64 kstack;               // Virtual address of kernel stack
  uint64 sz;                   // Size of process memory (bytes)
  pagetable_t pagetable;       // User page table
  struct trapframe *trapframe; // data page for trampoline.S
  struct context context;      // swtch() here to run process
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
};
```

Time Sharing

- How does OS switch one process to another?
- When a process is running, OS is not running, so is it even possible for OS to regain control, and how?

Approach A

Cooperative scheduling - OS waits for processes to make syscalls

What happens if one process runs in an infinite loop?

Reboot the machine!

Time Sharing

Approach B

- Non-cooperative scheduling - A timer device raises interrupt periodically
    - This requires some hardware help
    - What happens if timer interrupts during a system call?
    - What happens if timer interrupts during previous interrupt handler?
    - Concurrency problem! Needs locking…

Execution

How does a process enters kernel?

- Invoke a system call
- Is a system call a procedure call? Yes and no
- There must be an agreed calling convention between syscalls and syscall library in user space (libc)
- Trap instruction and return-from-trap instructions

User/Kernel Mode

0 - user mode: unprivileged instr

1 - kernel mode: privileged instr

? - machine mode (execute a few lines in entry.S): full privileged

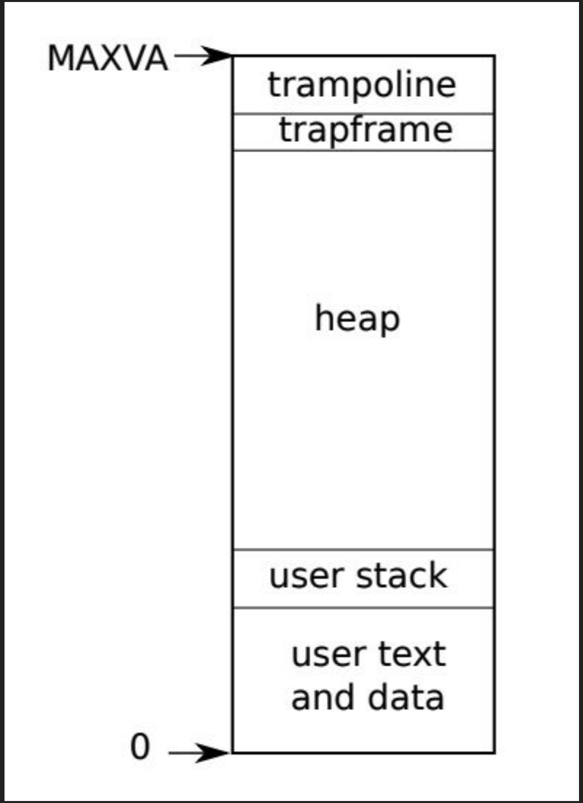
What are considered to be privileged instr?

1. editing TLB (translation lookaside buffer)
2. access page tables
3. exec OS code

Address Space

Address space includes

- Instructions
- The data to read and write to memory
- PC (program counter): instruction to execute next
- SP (stack pointer): manage the stack
- A list of file descriptors
- Runtime stack: local vars, function params, and return address
- Heap: dynamically allocated memory (`malloc`



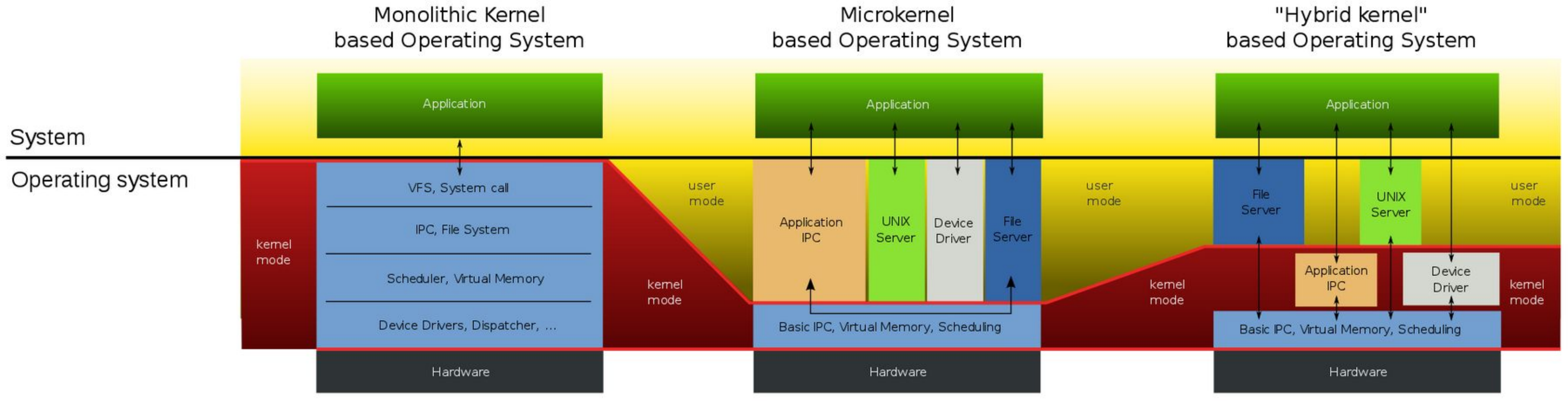
Entering kernel

Environment call (`ecall`)

- switches the sp (stack pointer) to the kernel stack
- saves the user process Context
- saves the old privilege mode
- sets the new privilege mode to 1
- sets the new PC to the kernel syscall handler

Kernel Design

- Kernel must have no bugs
- Kernel must treat processes as malicious



Monolithic kernel - Wikipedia

Next time

1. Don’t forget to sign up as presenters!
2. Lecture 4: page tables
3. Read chapter 3: page tables
4. Lab syscalls starts today!