Process management in xv6

Mythili Vutukuru
CSE, IIT Bombay

PCB in xv6: struct proc

```
2334 enum procstate { UNUSED, EMBRYO, SLEEPING, RUNNABLE, RUNNING, ZOMBIE };
2335
2336 // Per-process state
2337 struct proc {
2338   uint sz;                     // Size of process memory (bytes)
2339   pde_t* pgdir;                // Page table
2340   char *kstack;                // Bottom of kernel stack for this process
2341   enum procstate state;        // Process state
2342   int pid;                     // Process ID
2343   struct proc *parent;         // Parent process
2344   struct trapframe *tf;        // Trap frame for current syscall
2345   struct context *context;     // swtch() here to run process
2346   void *chan;                  // If non-zero, sleeping on chan
2347   int killed;                  // If non-zero, have been killed
2348   struct file *ofile[NOFILE];  // Open files
2349   struct inode *cwd;           // Current directory
2350   char name[16];               // Process name (debugging)
2351 };
2352
```

struct proc: page table

- Every instruction or data item in the memory image of process (code/data, stack, heap, etc.) has an address
  - Virtual addresses, starting from 0
  - Actual physical addresses in memory can be different (all processes cannot store their first instruction at address 0)
- Page table of a process maintains a mapping between the virtual addresses and physical addresses
- Page table used to translate virtual addresses to physical addresses

struct proc: kernel stack

- Stack to store CPU context when process jumps to kernel mode from user mode, or when process is context switched out
  - Why separate stack? OS does not trust user stack
  - Separate area of memory in the kernel, not accessible by regular user code
  - Linked from struct proc of a process

struct proc: list of open files

- Array of pointers to open files
  - When user opens a file, a new entry is created in this array, and the index of that entry is passed as a file descriptor to user
  - Subsequent read/write calls on a file use this file descriptor to refer to the file
  - First 3 files (array indices 0,1,2) open by default for every process: standard input, output and error
  - Subsequent files opened by a process will occupy later entries in the array

Process table (ptable) in xv6

- Ptable in xv6 is a fixed-size array of all processes
- Real kernels have dynamic-sized data structures

```
2409 struct {
2410   struct spinlock lock;
2411   struct proc proc[NPROC];
2412 } ptable;
```

CPU scheduler in xv6

- The OS loops over all runnable processes in ptable, picks one, and sets it running on the CPU

```
2768    // Loop over process table looking for process to run.
2769    acquire(&ptable.lock);
2770    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
2771      if(p->state != RUNNABLE)
2772        continue;
2773
2774      // Switch to chosen process.  It is the process's job
2775      // to release ptable.lock and then reacquire it
2776      // before jumping back to us.
2777      c->proc = p;
2778      switchuvm(p);
2779      p->state = RUNNING;
```