Lecture 35

14 October 2018 21:06

process mgmt. mechanisms (with xv6)

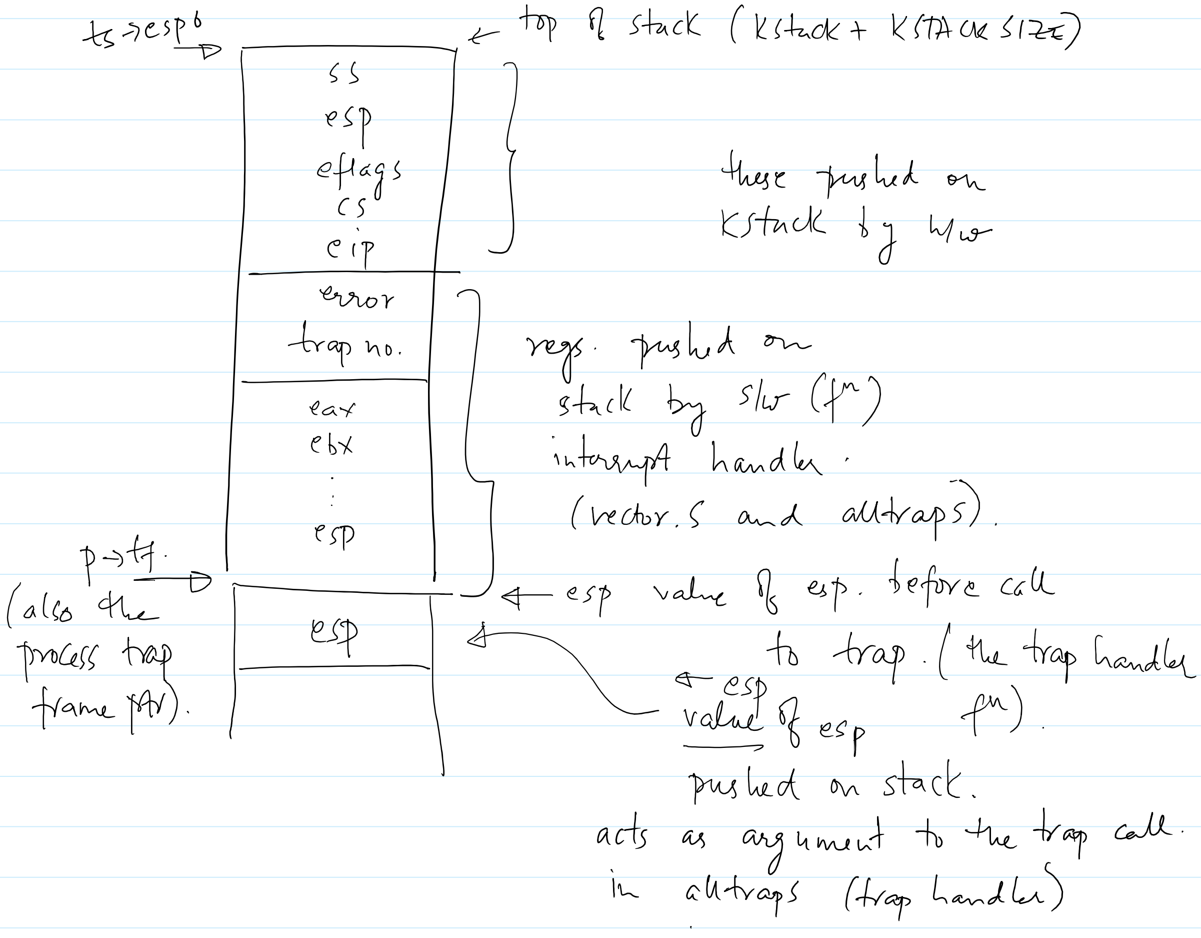
1. the interrupting process. (e.g. the int instruction)
2. the context switch mechanism
3. the child process creation & execution mechanism
4. the first process and its execution.

1. the 'int' instruction.

(i) check privilege levels to execute int.
(ii) find address of handler for interrupt
(iii) find address of process Kstack (p → kstack + KSTACKSIZE) from the task segment. regs. (which holds loc^n. of the task segment descriptor) ⇑ initialized with max. address of Kstack of current process.
(iv) push user stack and execution info./state on Kstack of process.
(v) jump to interrupt handler.

// steps (i), (ii), (iii) & (iv) happen in hardware as part of 'int' execution.

following is the Kstack of a process on/after an 'int'



in alltraps (trap handler)

push esp
call trap — trap (struct trapframe *tf)
mov esp+4, esp
pop regs.
iret

on return adjust esp back to trapframe → mov esp+4, esp

restore regs. → pop regs.

eax is updated with return value

return to user space & change privilege level.

esp value is tf ptr.

trap handler uses tf struct for syscall

syscall no. is stored in tf → eax.

lookup syscall()

lookup argint, ptrarg etc.

to get trap no. & system call parameters via user stack (tf → esp).

(2) context switch mechanism.

- a process will not voluntarily give up the CPU, usually. it may 'YIELD' at some times (yield, sleep etc.)
- or CPU is switched to another process by the scheduler on some interrupt processing (interrupts h/w, system calls etc.)

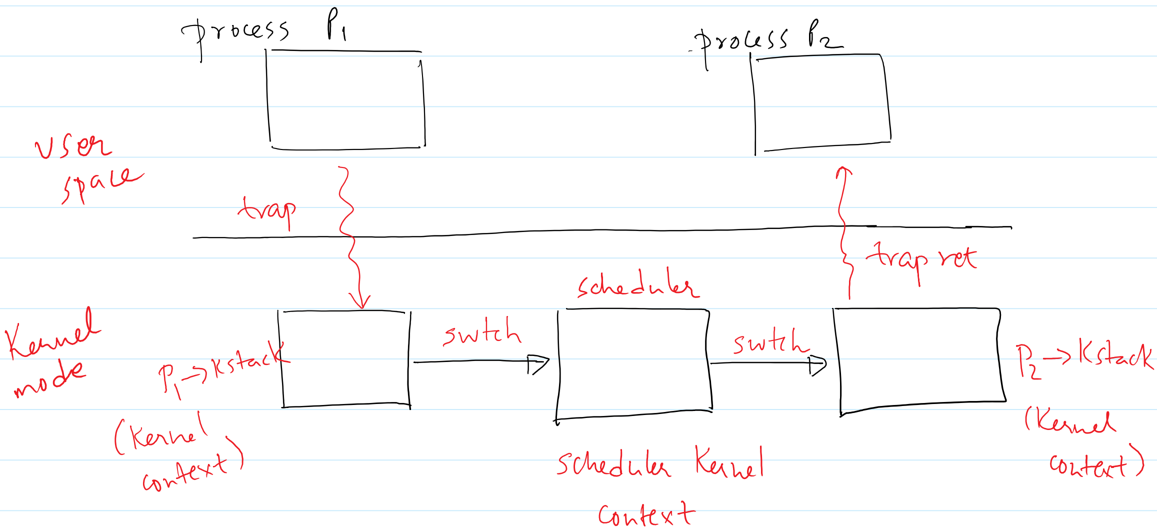
- the calls that are important are **sched** and **swtch**.
  - sched: to initiate the scheduler
  - swtch: to initiate context switch between processes / contexts

- **Sched** is called in exit, sleep, yield
  - sched in turn calls **swtch**.
  - additionally, the cpu scheduler calls **swtch**.

- with xv6, process preemption is a two step process.

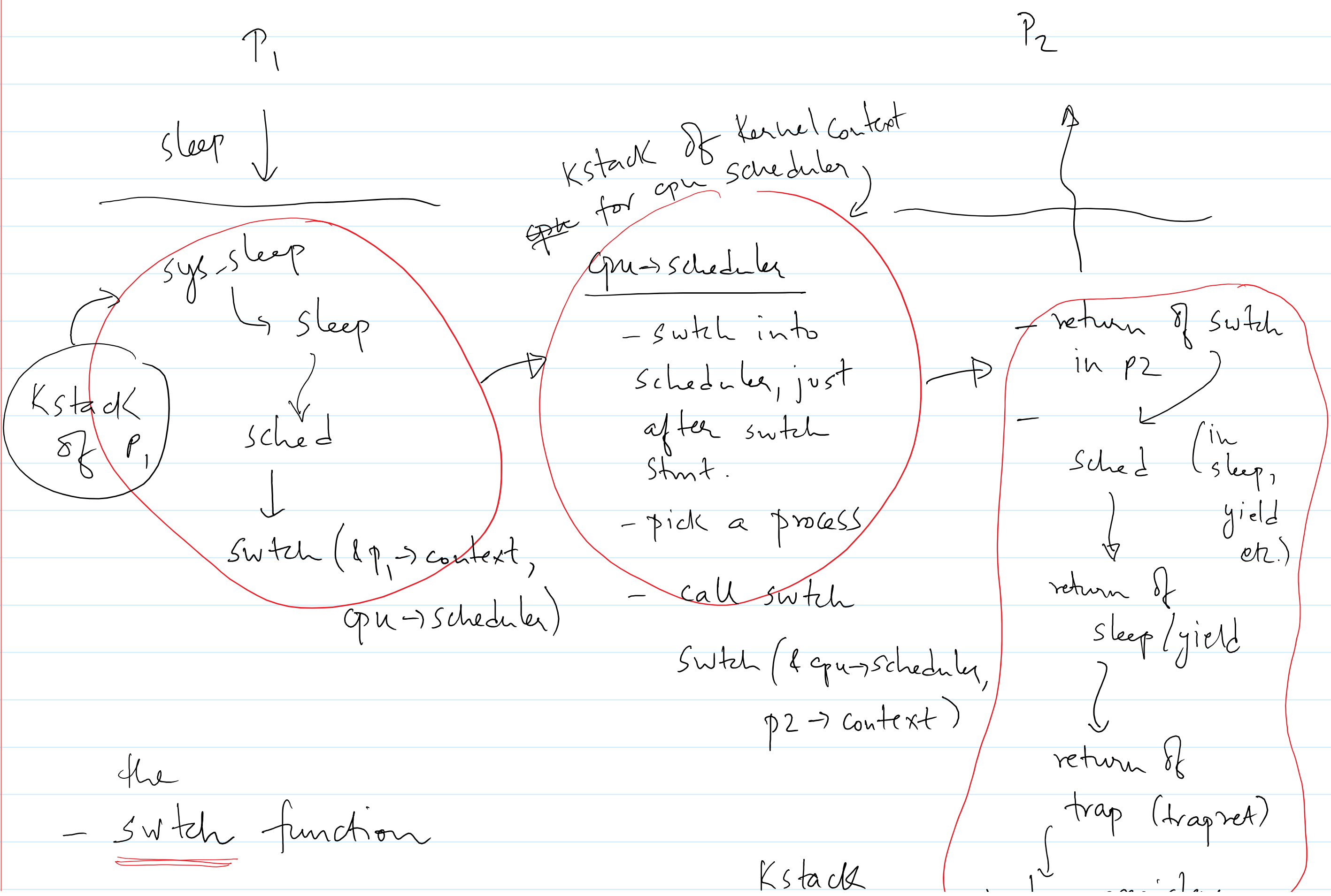
  step 1: process is switched out & kernel scheduler (with different kernel context) is switched in.

step 2: Kernel scheduler picks a process and switches to kernel context of the new process.



~ assume a process 'p_1' calls the sleep system call, and its state is updated in the sleep function and sched is called.

after sched, the (a two-step) context switch results in different process context for execution.



the
- switch function

Kstack

- switch function

(1) switch(&p→context,
cpu → scheduler)

Kstack of P_2 → trap (trapret) ↓ restore registers and return to user space.

- both arguments related to addresses of kernel stack.
- &p→context - address of context variable in PCB
  cpu → scheduler - value of top of stack pointer of the scheduler

~ how does switch switch?

switch

```
mov 4(esp), eax      ~ store 1st argument (at (esp+4)) in eax
mov 8(esp), edx      ~ store 2nd argument (at esp+8) in edx.

push registers
mov esp, (% eax)     ~ store current esp in address stored in eax.
                       i.e: p → context = esp
mov edx, esp         ~ switch kstack (to cpu→ scheduler)
pop registers        ~ pop registers on new stack
iret                   and return from switch (in new context).
```

this is where switch happens!