Lecture 36

18 October 2018 11:51

③ Creating & executing a child process.

fork

- allocproc
  - find PCB
  - populate w/ basic information
  - allocate the kernel stack

    sp = p → kstack + KTACKSIZE

    └ top of (empty) stack

    sp = sp − size of trapframe

    p → tf = sp ; // initialize trapframe pointer

    sp = sp − 4 ;

    * sp = trapret ;

    } move sp by 4 bytes and store trap return address to emulate return from trap call.

    sp = sp − size of context ; // struct context

    p → context = sp ; // initialize a context pointer on stack

- populate other fields
  - └ ppid, pgdir etc.

*np → tf = *p → tf ; // copy trap frame of ~~new~~ parent process (p) on to new process (np)

np → tf → eax = 0 ; // set return value of fork in new process to zero

np → context → eip = forkret ; // set 'eip' in kernel context of new process to address of function forkret.

np → state = RUNNABLE ;

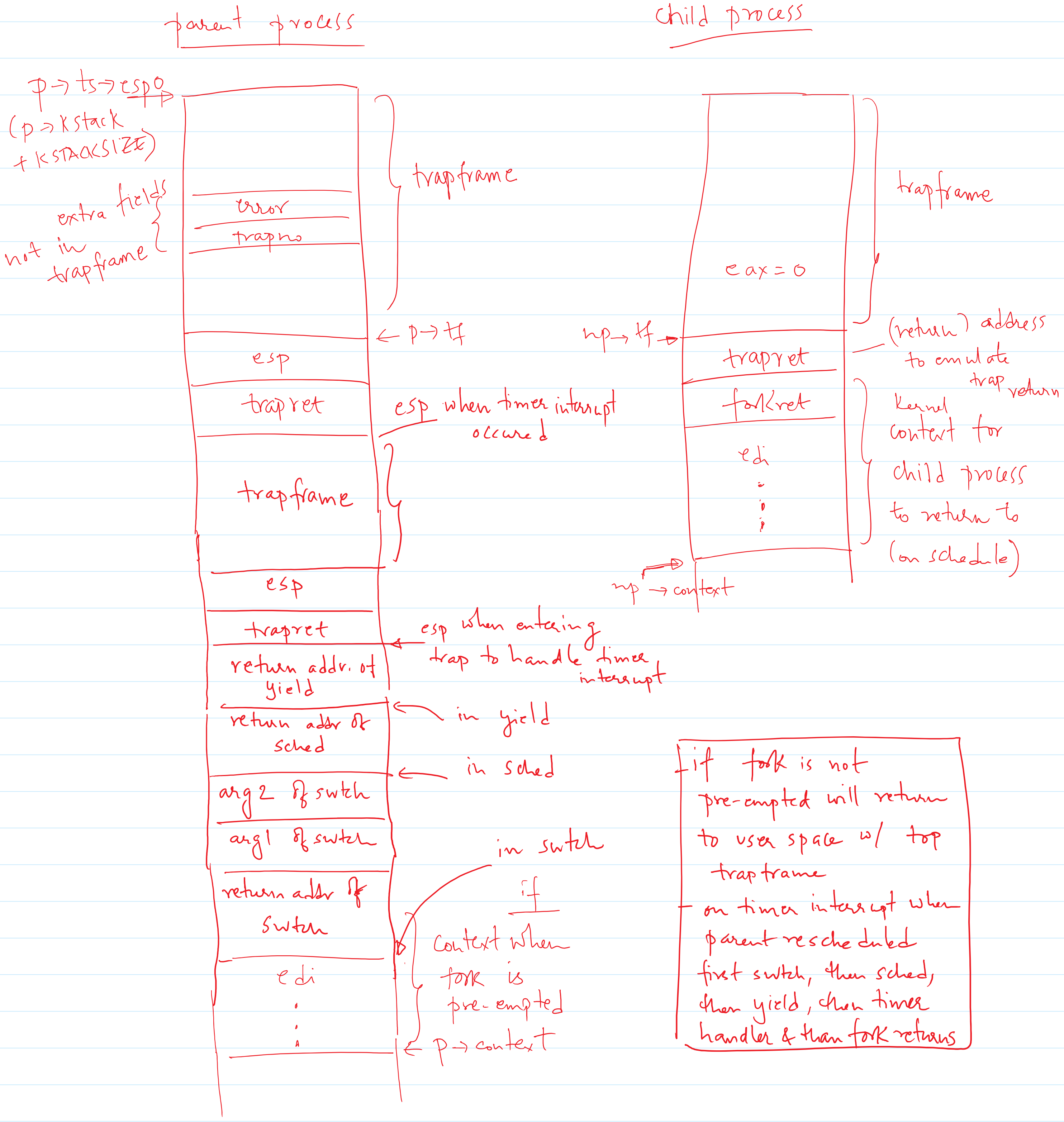
// when child is scheduled via kernel context will execute from forkret.

└ make child runnable (ready to be scheduled)

return pid ;

└ return from fork with pid of (new) child process.

(*) Kernel stacks of parent and child process after the fork system call (before trap handler returns)



if fork is not pre-empted will return to user space w/ top trapframe

- on timer interrupt when parent rescheduled first swtch, then sched, then yield, then timer handler & than fork returns

(*) when scheduler decides to switch/schedule child process, in fn. swtch

- pushes registers on Kstack of scheduler
  switches to Kernel stack of child process
- at this pt. esp = np -> context
- pops 4 registers and returns from swtch

- return from swtch pops address on Kstack to get
  address to return to, which is 'forkret'

- in forkret
  - release ptable lock (swtch is called with ptable lock held)
  - return from forkret

- return from forkret where?
- pop address on kstack ⇒ trapret! — address in generic trap handler code.

- at trapret.
  - pop trapframe from kstack
  - return to userspace via iret.

  || setup same user /process context as parent process (except eax)

⊗ child process see light of the day in user land!

- if first user statement in child is ;
  pid = fork() ; // assignment to variable pid.
- process will be back in Kernel to handle CoW fault
  copy-on-write

  || with CoW implementation. (default xv6 does not employ CoW.

④ Creating the first user process

this works very similar to the fork process.

this works very similar to the fork process,

- alloc proc
  setup of forkret and trapret on Kstack of the process.
- additionally,
  the function userinit does the following,

- allocates page table for process.
- loads a custom binary (initcode) in memory
- set eip in trapframe to zero.
  - on return to user-space execute from address '0'.

- the custom binary/program (initcode.S) is handcrafted (w/ assembly instruction) to do one thing
  - call the exec system call with the real init program ./init