CS 3214: Computer Systems
Lecture 2: Processes

Instructor: Huaicheng Li

Aug 25 2022

VIRGINIA TECH™

Some Updates

- Syllabus quiz released, deadline: 8/31 11:59pm
- Lectures (Please bookmark them)
  - https://courses.cs.vt.edu/cs3214/fall2022/lecturesli/
  - Course schedule: https://tinyurl.com/cs3214-schedule
- Office hours posted
  - TA office hours: Google Calendar (Course website -> MORE INFO -> Staff)
  - Huaicheng Li's office hours
    - Fridays, **McBryde 122-B**, 1:30-3:30pm (starting 9/2/22)
    - Zoom for appointment-based meetings, sign up here
- Exercise 0 will be released soon…

Recap

- Systems Architecture: Applications / OS / Hardware
- Dual mode operation: Applications <-> OS (protection, isolation, performance)
  - System calls as OS APIs for applications to use
  - Dual mode operations: User/kernel mode, CPU privilege levels (Ring 3/0)
- Processes
  - Virtual resources including CPU share, address space, file descriptors, etc.
- **Time-sharing:** *N applications on 1 CPU ➔ 1/n^{th} CPU for each application*

Mode Switching

- User -> Kernel mode
  - Explicit:
    - Call system calls to enter kernel mode
    - Fault/exceptions (e.g, division by zero, attempt to execute privileged instructions
    - Synchronous
  - Implicit: (external events, e.g., hardware interrupts or preemption)
    - Preemption: higher priority kernel-level process needs to run
    - Interrupts: What is it?
    - What types of interrupts? Timer, keyboard, mouse, disk, network, etc.
    - Asynchronous
- Kernel -> User mode
  - Via special privileged instruction (e.g., Intel *iret*)
  - A return from interrupt

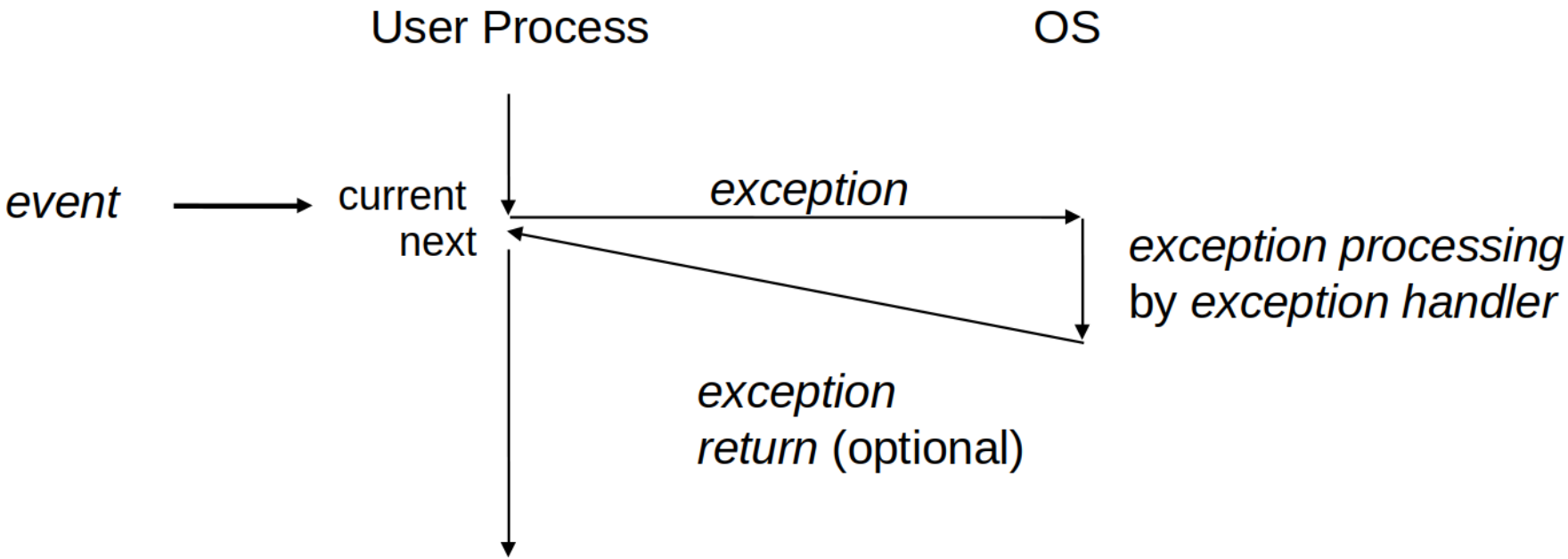
Processes

- **Process definition:** An instance of a program that is being executed (aka, a running programming)
- Abstractions provided to a process
  - Virtual CPU: illusion of many CPUs
  - Address space – machine state
  - Files, etc.
- Time-sharing to enable multi-programming
- Context switches
  - **Context:** the state of the running program, which includes the current program text, the location within the program text (PC/IP), and all associated state: variables (global, heap, stack, CPU registers)
  - Switches – Dual Mode operations

Context vs. Mode Switches

- Mode switch guarantees that kernel gains control when needed
  - To react to external events
  - To handle error situations
  - Entry into kernel is controlled
- Not all mode switches lead to context switches
- Kernel decides if/when – subject to process state transitions and scheduling policies
- Mode switch does not change the identity of current process/thread

A Bottom-Up View of "Exceptions"



Process Struct in Linux

- Check Linux code: ***struct task_struct*** { … }
  - *https://elixir.bootlin.com/linux/v5.19.3/source/include/linux/sched.h#L726*
  - struct mm_struct *mm;
  - struct files_struct *files;
  - struct sched_info sched_info;

System Calls

- How it works:
  - The "syscall" instruction
  - Syscall table (ID, parameters)
  - Some example system calls
  - Syscalls in Linux: "*arch/x86/entry/syscalls/syscall_64.tbl*"
- A demo with "printf()"
  - GDB cheatsheet: http://csapp.cs.cmu.edu/3e/docs/gdbnotes-x86-64.pdf
  - Linux kernel code: ***https://elixir.bootlin.com/linux/v5.19.3/source***
  - Help command: "man strace"

- Program: binary/executable, on-disk set of instructions + static data
- How does OS convert a program to a running process?
  - Load program into memory (disk/file read)
  - Parse parameters (registers, PC)
  - Main()
  - Address space (stack, heap, etc.)
  - I/O accesses
- Process APIs
  - Create
  - Destroy
  - Wait
  - Misc control
  - Status

We ended here on 8/25.

Slides 7&8 not covered yet.