Operating Systems: Intro

Shankar

September 3, 2013

Overview

- Computer-system hardware
  - processors, memory, IO devices (display, keyboard, disks, ...)
  - connected by buses
  - IO device: device + adaptor
- Operating system
  - Software: runs directly on the hardware, always running
  - Provides a more convenient virtual machine
    - processes, threads; communication, protection
    - virtual address space, filesystem, high-level IO, users
  - Shares hardware among processes and OS
    - sharing mechanisms + scheduling policies

Course outline – 1

- Hardware review
- OS overview
- ToyOS
- Booting and kernel initialization
- Process: instance of an executing program
  - resources: processor, memory, files, IO devices, ...
  - scheduling: short/medium/long term
- IPC: pipes, signals, shared memory, ...
- Threads: active agents of a process
  - user, kernel, kernel-mode user

Course outline – 2

- Multi-threaded programs
  - synchronization constructs: semaphores, locks, swap, ...
  - critical section, producer-consumer, ...
- Deadlocks
- Memory management
  - swapping, segmentation, paging, allocation, ...
- Filesystem:
  - interface, implementation, ...
  - GOSFS, UFS, log-structured, distributed, ...
- Block devices: disks, SSD, ...

- Throughout, relevant parts of GeekOS will be discussed

Hardware Review

Processor – 1

- Executes (machine) instructions from memory
- State
  - general-purpose registers (gpr)
  - instruction pointer (ip) // aka program counter
  - stack pointer (sp)
  - processor status (ps)
    - arith/logic flags: overflow, carry, zero, ...
    - mode: user/kernel
    - intrpts on/off
    - paging on/off
    - ...
  - address-translation stuff
    - segment/page table base address, associative maps
  - ...

Processor – 2

- Instructions
  - move
  - io (in/out)
  - arith/logic
  - jmp[cond] *addr*
  - push reg: mem[sp–] ← reg
  - pop reg: reg ← mem[+sp]
  - call *addr*: push ip; ip ← *addr*
  - ret: pop ip

Processor – 3

- Instructions
  - sw-intrpt *n* // aka traps, exceptions; from cpu
    - push ip, ps
      ip ← mem[*n*]
      ps ← intrpt-off, kernel-mode
    - rti: pop ps, ip
  - hw-intrpt *n* // from external, adaptor
    - same action as swi
  - ...
- Privileged instr: io, set kernel mode, clear cache, ...
  - user-mode execution → exception
- user-mode → kernel-mode: only via sw/hw intrpts

Adaptors (aka controllers)

- Adaptors (aka controllers)
  - processor/memory ⟷ adaptor ⟷ device

- Disk adaptor
  - disk: holds blocks at surface/track/sector
  - data register: holds input/output data
  - pcontrol register:
    - operation: r, w, seek, ...
    - location: in disk
    - addrress: of buffer in memory
    - intrpt on/off
    - dma on/off
    - busy: on/off // for non-interrupt IO

- Adaptors: display, keyboard, mouse, USB, Ethernet, WLAN, ...
- Varying data unit size, transfer bandwidth, latency

Memory

- Ideal
  - single-level memory
  - accessible to all processors and dma-capable adaptors
  - fast enough to handle simultaneous requests
  - unrealistic
- Reality
  - multiple levels: caches, memories
  - small/fast ⟶ large/slow
  - caches: local to a processor
  - local memories: accessible by a subset of processors/adaptors
  - global memory: accessible by all processors/adaptors

Active agents

- Active agents: processors + adaptors
- Execute independently
- Interact via
    - io instructions
        - processor reads/writes adaptor registers
    - hw-interrupts
        - adaptor makes processor execute io code
    - shared memroy
        - buffers accessed by processor and by adaptor via dma

Operating System Overview

Introduction

- Provides a more convenient virtual machine
  - processes + threads
    - protection for each process
    - synchronization (IPS): semaphores, locks, signals, ...
    - communication (IPC): sockets, pipes, shared memory, ...
  - virtual structured address space
  - filesystem, high-level IO
  - users

- Shares hardware among processes and OS
  - sharing mechanisms + scheduling policies

- Active agents: processes + threads
  - execute independently
  - interact via synchronization/communication constructs

Processes + Threads

- Process: executing instance of a program
  - Life: start, execute, terminate (perhaps)
  - Address space: text segment (code) + data segment
  - Resources: files, sockets, ...
  - Threads: each executes code; has its own stack
- Traditional programs: process has exactly one thread
  - address space: text, data, stack
- Multi-threaded programs: one or more threads per process
  - address space: text, data, stack_1, stack_2, ...
- OS makes all processes and threads execute concurrently
  - gives each process/thread a share of the hardware resources
  - sharing done in time and/or space (depends on resource)

PCB (Process Control Block)

- PCB per process: holds enough state to resume the process
  - address-space: text/data locations; memory or disk
  - for each thread: processor state; stack location
  - IO state
  - accounting info
  - ...
  - status
    - running: executing on a processor
    - ready (aka runnable): waiting for a processor
    - waiting: for a non-processor resource (eg, memory, IO, ...)
    - swapped-out: its process holds no memory

- running ↔ ready: timer intrpt/short-term scheduler
  running → waiting → ready: io request/completion
  ready/waiting ↔ swapped-out: medium-term scheduler

Virtual Address Space

- Address space of a process
- Structured into segments/pages
  - attributes: size, allowed access, ...
  - checked during execution
- OS maps each virtual address to
  - address in physical memory (accessible to processor)
  - location in disk (processor access → exception)
- Mapping: segment/page tables, associative maps, ...
- Allocation of physical memory to process
  - maximize multi-processing w/o thrashing

Filesystem

- Non-volatile structure of directories and files
- Tree/acyclic structure
- Each node is a directory or a file
  - file: holds data; variable size
  - directory: pointers to directories and files
  - attributes: owner, access rights, creation time, ...
- Processes can create/delete/read/modify/execute nodes
- Executable file: code + data segments, loading/linking info
- OS implements filesystem on block devices (disks, ...)
  - each node is mapped to one or more blocks
  - pointer structure to locate blocks of any node
  - use free blocks to expand nodes

System Calls

- swi-syscall *n*: like a function call except
  - function ("syscall handler") is in kernel
  - *n* is not address but an index to a kernel table of addresses
- Classes of system calls
  - Process management
    - create/terminate a process/thread (including self)
  - Filesystem and IO
    - create, delete, open, read, write, close, modify attributes
  - Information
    - time, process information, hardware, IO devices, ...
  - Communication
    - connect, send, receive, terminate