CSC3150 – Operating System

Review

Prof. Wei Chung Hsu

School of Data Science

Chinese University of Hong Kong, Shenzhen

Operating System Definition

Old definition:

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Users want convenience, **ease of use** and **good performance,** they don't care about **resource utilization**

Operating System Definition

Broader definition:

- OS provides **User friendly Interface** and API for **system calls**
- OS is a **resource allocator and manager**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS provides a **File system** to assist I/O functions
- OS is a **control** program to manage who can access the system and what they're allowed to do, with security features like firewall and anti-virus scanning.

Computer-System Architectures

- **Single CPU systems**: Older systems use a single general-purpose processor
- **Multiprocessors systems** growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a specie task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks
- **Clustered systems** multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** has multiple nodes running applications, monitoring each other
  - Some clusters are for **high-performance computing (HPC)**
    - Applications must be written to use **parallelization**

Operating System Structures

- **Multiprogramming** (**Batch system**) needed for efficiency
  - A single program cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job
- **Timesharing** (**multitasking**) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨**process**
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory

Operating-System Operations

- **OS is Interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
    - Device controller informs OS that I/O operations are compete
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - Memory access violations
    - Request for operating system service (**system calls**)
    - Other process problems include infinite loop, processes modifying each other or the operating system

Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction

  some interrupts only need to save PC and a few registers

  some interrupts may cause a full context switch which is a lot more costly
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request

  trap or exception are internal events (e.g. fp underflow, page faults, system calls), which must be handled immediately. Interrupts are external events, and may be delayed to be handled at a more convenient time.

User Mode and Kernel Mode

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **Kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - **Root mode (Hypervisor mode), Kernel mode (Supervisor mode) and User mode.**
  - Hypervisor runs in root mode, Guest VM kernel runs in supervisor mode, and applications run in user mode.

Operating System Services

- Operating systems provide an environment for execution of programs and services to programs and users
- One set of operating-system services provides functions that are helpful to the user:
  - **User interface** -
    - Varies between **Command-Line (CLI)**, **Graphics User Interface (GUI)**, **Batch**
  - **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device
  - **File Systems:** for open/read/write services
  - **Communications:** IPC via shared memory or message passing
  - **Error detection**: OS needs to be constantly aware of possible errors

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
  - Many system calls are available in Clib
- Mostly accessed by programs via a high-level **Application Programming Interface (API)** rather than direct system call use
- Three most common system call APIs are
  - Win32 API for Windows,
  - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and
  - Java API for the Java virtual machine (JVM)

Types of System Calls

- Process control
  - create process, terminate process
  - end, abort
  - load, execute
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
  - Dump memory if error
  - **Debugger** for determining **bugs, single step** execution
  - **Locks** for managing access to shared data between processes

Types of System Calls

- File management
  - create file, delete file
  - open, close file
  - read, write, reposition
  - get and set file attributes
- Device management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices

Types of System Calls (Cont.)

- Information maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get and set process, file, or device attributes
- Communications
  - create, delete communication connection
  - send, receive messages if **message passing model** to **host name** or **process name**
    - From **client** to **server**
  - **Shared-memory model** create and gain access to memory regions
  - transfer status information
  - attach and detach remote devices

OS Services

- Interacting with the OS
  - CLI
  - GUI
  - Touch screen
- Services are provided by
  - System calls via API
  - A set of system programs
- System calls in six major category: process control, file management, device management, information maintenance, communication, protection

OS Design and Implementation

- There are some different OS designs:
  - monolithic,
  - layered, and
  - micro-kernel based.

  Layered and micro-kernel approach have lower performance due to communication and mode switch delays.

- Loadable kernel module (LKM) is more commonly adopted to work with monolithic kernels.

- Performance can be monitored using counters or tracing. Lots of tracing tools (Strace, Dtrace) available today.

Process Management

- A process is a program in execution.
- The status of a process is represented by the PC, the registers, and a set of flags. (and a pointer to page table)
- The virtual address space of a process has four sections: text, data, heap, stack
- As a process execute, it changes stats:
  - New, Ready, Running, Waiting, and Terminated
- PCB (Process Control Block) is the kernel data structure that represents a process.
- Process scheduler manages which queue a process should be moved to. It happens when some event occurred, such as a new process created, an I/O finished, system overloaded, …
- When a running process is switched out, a context switch is required.
- Fork() and CreateProcess() system calls are used to create processes on Unix and Windows.
- IPC can use shared memory or message passing

Process Management (cont.)

- A pipe can be used for two processes to communicate.
  - Ordinary pipes are for parent-children communications. In Unix, it is used via the pipe() system call.
  - Named pipes are more general and can be used for several processes to communicate. In Unix, it is called FIFOs.

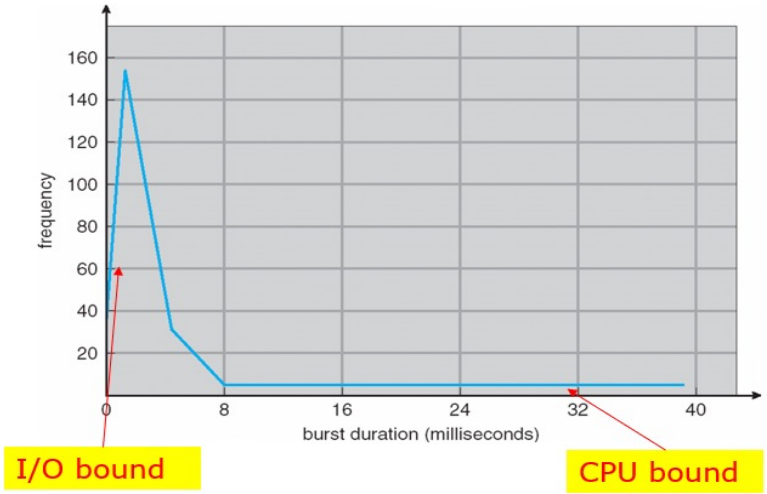
- Client-server communications can use sockets and RPC (Remote Procedure Call). Sockets allow two processes on different machines to communicate over a network. RPC allows a function to be invoked on another process that may reside on a separate computer. Android uses RPC as a form of IPC. During a RPC call, the kernel finds a port from a matchmaker, and use the port number to send out RPC call.

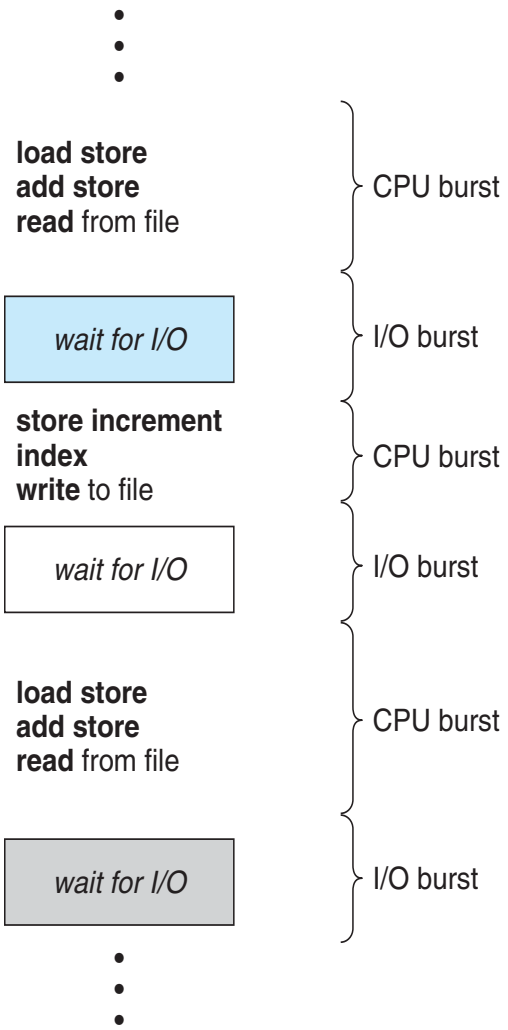
Threads

- A thread represents a basic unit of CPU utilization.
- Threads belong to the same process share many of the process resources, such as opened files, code, and data.
- Multithreading challenges:
  - Dividing and balancing the work
  - Dividing the data
  - Identifying data dependences
  - Hard to test and debug
- User-level threads must be mapped to kernel-level threads to be executed in parallel on multi-processors. Three different mappings: Many-to-one,, one-to-one, many-to-many. One-to-one gives best parallel performance, but may require too many kernel level threads.
- A thread library provides API to create and manage threads. Pthread is the library for Unix, Linux, and MacOS.
- Implicit threading allows users to identify tasks – not threads – and allowing languages or API to create and manage threads. Examples are MS GCD (Grand Central Dispatch), OpenMP, Intel TBB, thread pools, ..

CPU Scheduling

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle – Process execution consists of a **cycle** of CPU execution and I/O wait
- **CPU burst** followed by **I/O burst**
- CPU burst distribution is of main concern





Scheduling Criteria and Algorithms

- **Criteria**
  - **CPU utilization** – keep the CPU as busy as possible
  - **Throughput** – # of processes that complete their execution per time unit
  - **Turnaround time** – amount of time to execute a particular process
  - **Waiting time** – amount of time waiting in the ready queue
  - **Response time** – amount of time it takes from when a request was submitted until the first response is produced, for time-sharing environment
- Scheduling Algorithm
  - FCFS
  - SJF (Shortest Job First) – best for turnaround time and average wait time
  - SRTN (Shortest Remain Time Next)
  - RR (Round Robin) – best for response time, but context switch overhead may be high
  - Priority scheduling

Multilevel Queue and Feedback

- Multi-level queue
  - Ready queue is partitioned into separate queues, eg:
    - **foreground** (interactive)
    - **background** (batch)
  - Process permanently in a given queue
  - Each queue has its own scheduling algorithm:
    - foreground – RR (*for best response time*)
    - background – FCFS (*for simplicity, lowest overhead*)
- Multi-level feedback queue
  - A process can move between the various queues; aging can be implemented this way
  - Determine when a process comes, which queue to enter
  - When to promote or demote a job

Process Scheduling vs CPU Scheduling

- CPU scheduling and process scheduling are closely related concepts in operating systems, but they operate at different levels of granularity.

	CPU Scheduling	Process Scheduling
Focus	Which individual process will have access to the CPU	Managing larger-scale transitions between process states (e.g., ready, running, blocked).
Time scale	Extremely short-term	Only invoked when a process is created, blocked, moved, woke up..
Decisions	Which one to run? For how long?	Determines which queue a process goes to
Algorithms	FCFS, Round Robin, Shortest Job First, Priority Scheduling	System loads may also be considered, multiple level queues, multi- feedback queue, …

Synchronization Tools

- Race condition: concurrent access to shared data makes the data non-deterministic
- Critical section is a section code where shared data may be manipulated and race conditions may occur.
- Critical sections must be protected and the solutions must meet three requirements:
  - Mutual exclusion
  - Progress
  - Bounded waiting
- Peterson's solution meets all three requirements, but may not work on modern computers due to memory operation reordering (HW and SW).
- Hardware support such as (barrier, test-and-set, compare-and-swap) are required.
- Mutex locks can ensure mutual exclusion
- Semaphore is more flexible than mutex locks, because it supports integer values rather than just 0 and 1. Can be implemented without busy waiting
- A Monitor is an abstract data type that support process synchronizations

Memory Management and Virtual Memory

- Addresses generated by the CPU are logical address (Virtual Address). MMU translates them to Physical Address.
- Memory Allocation can be done with contiguous section of varying size using first-fit, best-fit, or worst-fit algorithms. Due to fragmentations, this is not used. Modern OSes use paging to manage memory.
- A page table maps VA to PA. A virtual address is divided into two parts: a page number and a page offset. The virtual page number is used to index the page table, and the page table entry (PTE) points to the mapped physical page number.
- Most operating systems support demand paging: a page is mapped only when it is referenced. When a virtual page is referenced but not mapped, page fault occurs. A page fault traps to OS, and OS needs to locate a physical page and update the PTE in the page table.
- To speed up PTE reference, TLB is used. TLB is a cache for the page table.
- For 64bit virtual address space (48bit are used today) page table may be too large to be used.

- Page Table
  - Multi-level index: such as the 4-level index table for x86-64 and the ARM-v9 architectures
  - Hash table approach used in Solaris with two level caching (TLB and TSB)
  - Inverted table
    - Minimal table size, but searching may be slower, and is not good for memory sharing (virtual address aliasing)
- TLB miss
  - Can be handled in HW by the HW table walker
  - Can be handled in SW by the TLB miss handler

Virtual Memory

- Benefit of Virtual Memory
  - Programs can have an address space much greater than physical memory
  - Demand paging: programs do not need to be entirely in memory
  - Processes can share memory
  - Process can be created more efficiently (e.g. COW)
- Page replacement is needed when the available memory runs low. Page replacement algorithms include FIFO, Optimal (Belady's algorithm), LRU, LRU approximation, second chance, enhanced second chance, counting based (LFU and MFU).
- Global replacement select a page from other processes. Local replacement selects a page from the faulting process.
- Thrashing happens when a system spends more time paging than execution.
- A working set is defined as a set of pages currently in use by a process.
- Memory compression is one alternative to paging, and is used in some mobile systems.

Virtual Memory (cont.)

- TLB reach refers to the amount of memory accessible from the TLB.

Since the cache size is kept increasing, now we have cases that the TLB reach is insufficient to cover the cache size. One way is to increase the page size. Large page size may be unfriendly to small, interactive jobs, so many modern operating systems start to adopt variable sized pages.

- Linux, Windows, and Solaris use a variation of LRU approximation as the clock algorithm. The clock algorithm is similar to the second chance algorithm (circular FIFO and a reference bit). However, the enhanced second chance algorithm (a reference bit and a dirty bit) works better.

File System Interface

- A file is an abstract data type defined and implemented by the OS. It is composed of a sequence of logical records (bytes, lines, structs,..)
- The OS is responsible to map the logical file onto physical storage devices (HDD or SDD). The logical record size may be different from the physical record size.
- Within a file system, directories are often used to organize data. Multi-level directories are used to avoid naming conflicts.
- A generalization of the multi-level directory is a tree-based structure. Acyclic-graph based directory enable users to share subdirectories and files.

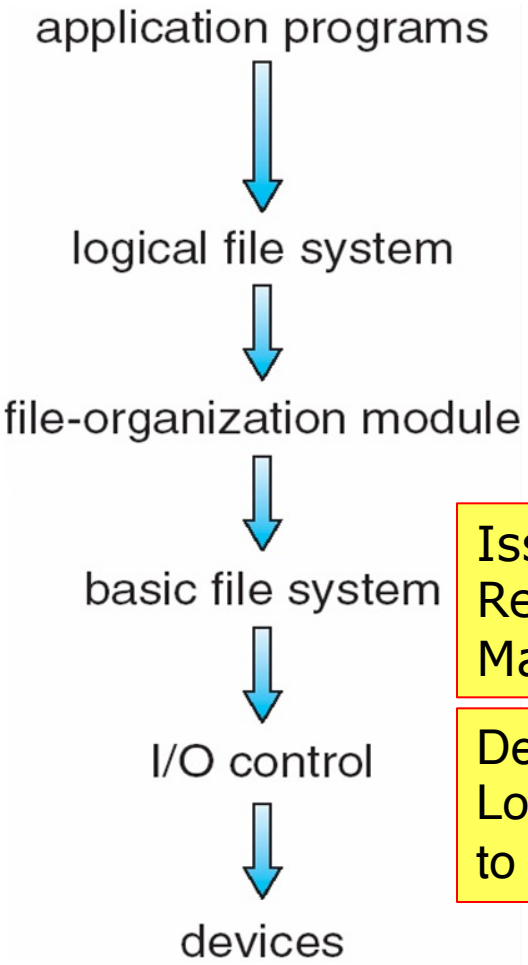
File System Interface

- A file is an abstract data type defined and implemented by the OS. It is composed of a sequence of logical records (bytes, lines, structs,..)
- The OS is responsible to map the logical file onto physical storage devices (HDD or SDD). The logical record size may be different from the physical record size.
- Within a file system, directories are often used to organize data. Multi-level directories are used to avoid naming conflicts.
- A generalization of the multi-level directory is a tree-based structure. Acyclic-graph based directory enable users to share subdirectories and files.

File System Implementation

- Most file systems reside on secondary storage: HDD and SSD. HDD for low cost, SSD for high performance.
- Storage devices are often partitioned, and mounted to differenn logic file systems.
- File systems are implemented in layers.
- Space allocation:
  - Contiguous
    - fragmentation
  - Linked
    - Direct access is a problem
  - Indexed
    - Indexing overhead may be high

Layered File System



Manage metadata such as directory
FCB (inode in Unix)

Mapping from logical address to physical block address
Free space manager

Issues commands to I/O control
Request scheduling
Manage buffers and caches

Device driver and interrupt handlers
Low-level hardware specific commands to hardware controller

File System Implementation (cont.)

- Free space allocation:
  - Bit vector
  - Linked list
- Optimizations include grouping, counting, FAT.
- Directory often uses hash table searching approach
- File system must be reliable, cannot afford data loss
  - A consistency checker to repair damaged file systems
  - Regular backup is often needed
  - Database system mused logging approach to ensure the atomicity of write transactions
- The inode structure can support up to 4TB of files. However, to access to the late records, we must go through the three level of indirections. How can we speed up such references?
- In the basic file system layer, disk buffer cache and page caches are often used to speed up file accesses.

About Final Exam

- Time: 05/11, Saturday, 8:30 to 10:30
- Room: TA 301, 302
- Seat Assignment: See your mail message from registry.
- Style: Open Book, Open Notes, Open Laptop/Tablet (no communications)
- Scope: Textbook Chapter 1-6, 9-10, 13-14 and all lecture notes
- Format: Three sections: single choice (10), multiple choices (10), short essay questions (6-7).
- Hints:
  - Know the synchronization tools provided
  - How to deal with the page table problems?
  - How to implement a file greater than 4GB and with good performance
  - Have you done your homework?