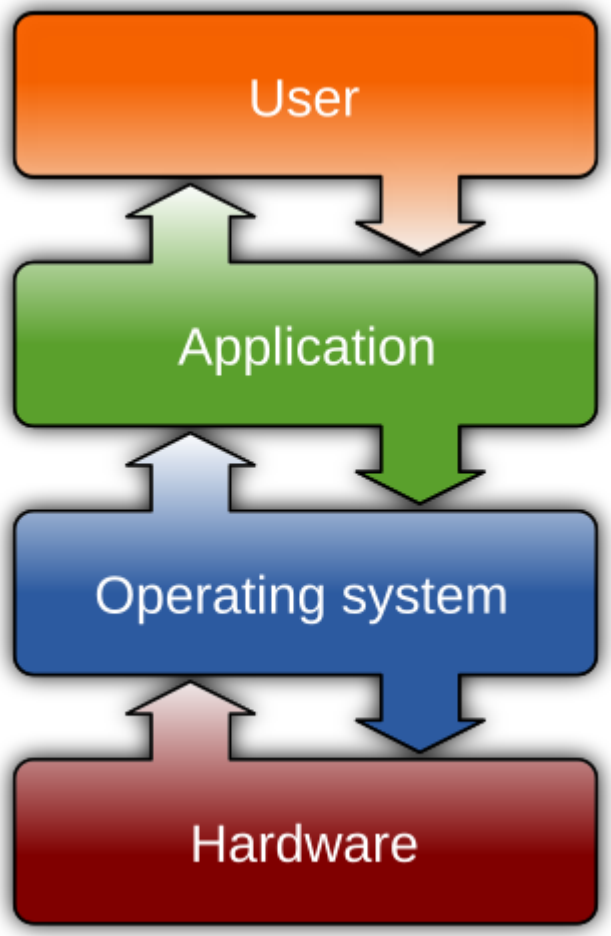
Operating System and UNIX File System

COS 316 Precept 2

Today's Agenda

- What is OS
- UNIX File System

What is OS?



Functionalities of OS

Process management

Memory management

File system management

Networking

Security

...

Functionalities of OS

Process management

Create, terminate processes using system calls**.**

Multi-tasking: even with only 1 CPU core, processes can run as if they were running simultaneously

Scheduling: which process gets the CPU and for how long

How: context switching

When: scheduling policy

Functionalities of OS

Memory management

Allocate memory for processes to run.

Isolation:

Process A cannot see memory of Process B

How: virtual memory

Main difference between process and thread

	Virtual memory space	Sharing memory
Process	Each process has its own virtual memory space	Not allowed across processes
Thread	Threads in the same process exist in the same virtual memory space	Shared

Functionalities of OS

File system management

The OS organizes and retrieves files, managing directories, file naming and permissions.

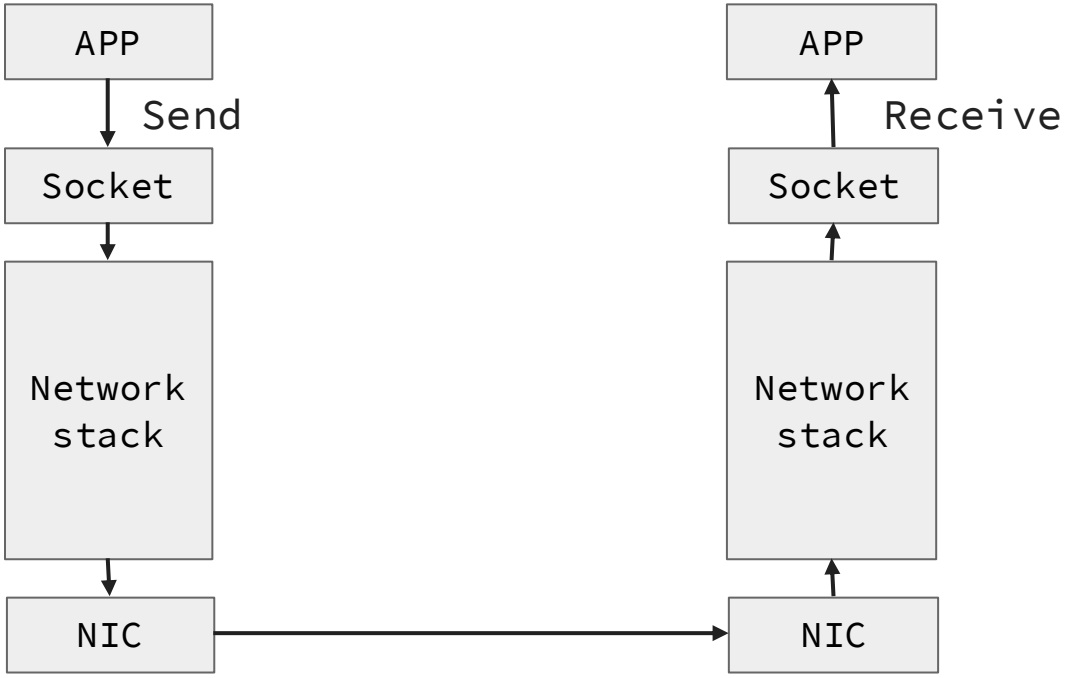
It ensures data integrity through mechanisms such as data validation, checksums and error-correcting codes.

Functionalities of OS

Networking

Handle network communications.

Socket programming provides an interface to send/receive messages.



Functionalities of OS

Security

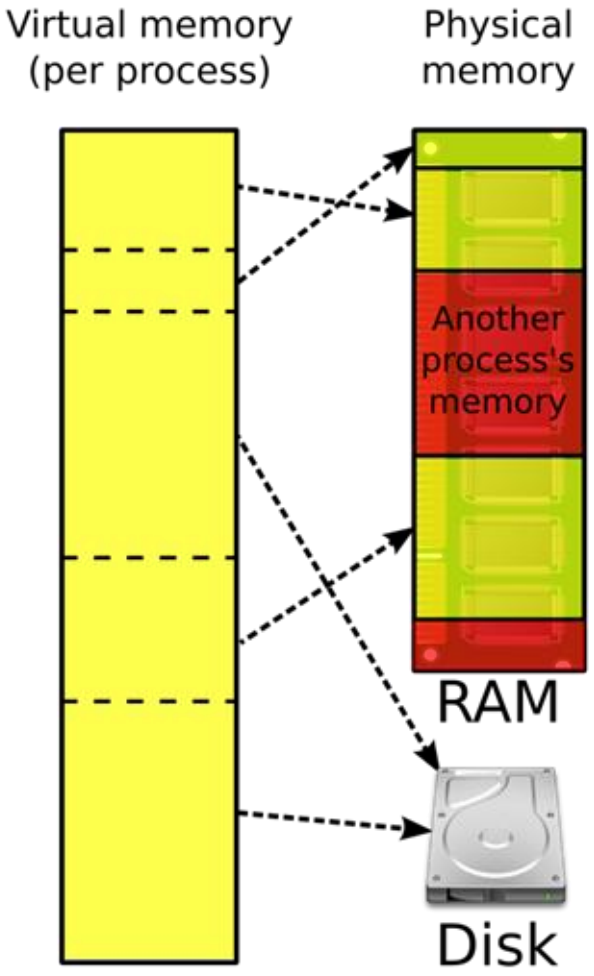
- Authentication, encryption, etc.

Virtual Memory

Purpose: provide an abstract of memory to applications

- Illusion of having a large, continuous block of memory
  In reality, data may be fragmented across physical RAM and even stored on disk.
- Isolation.
  Different progresses cannot access others' memory.

Virtual Memory



Virtual Memory

Naming: 64 bit address & process ID

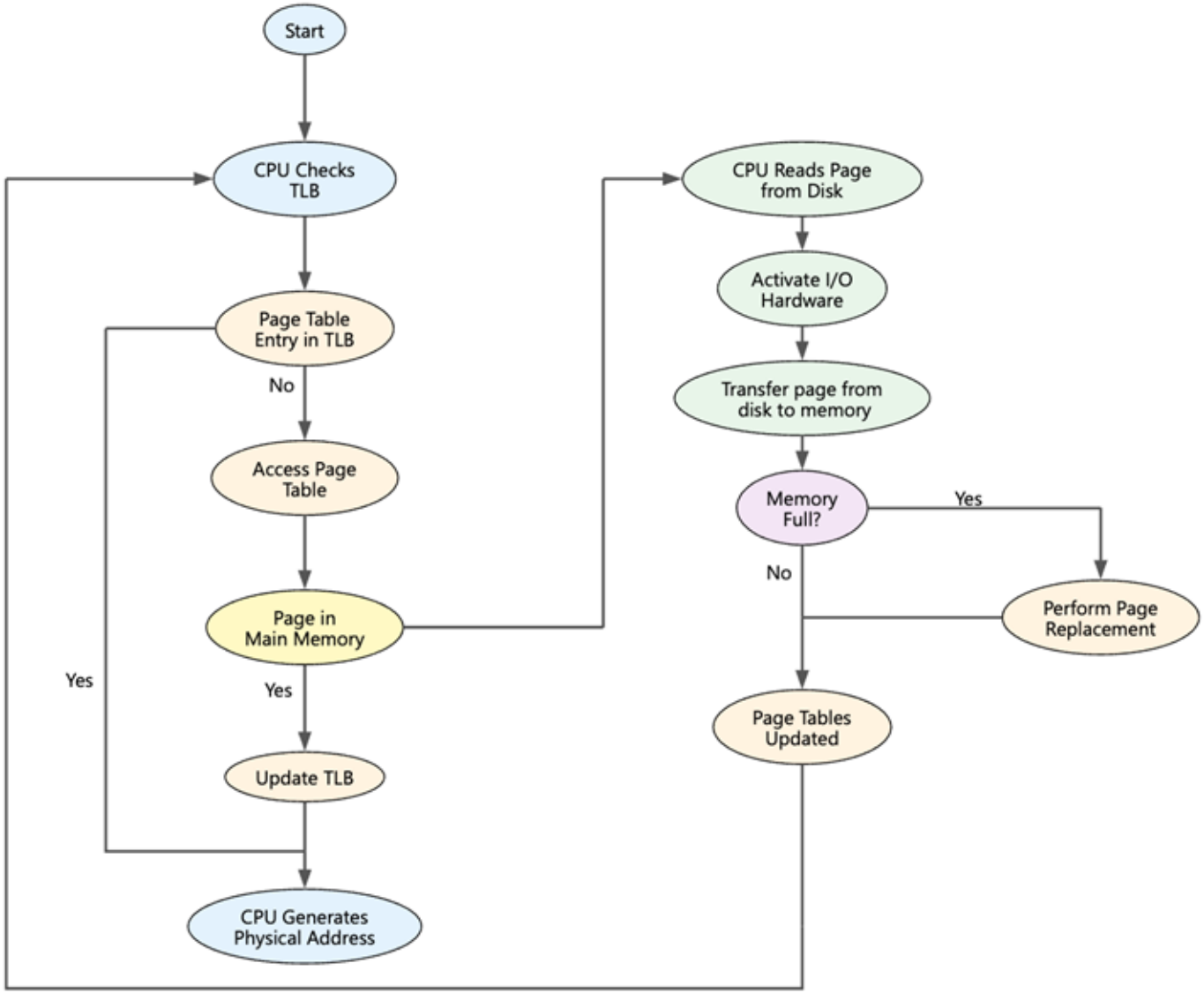
Lookup: page table (per process)

- Mappings between virtual memory addresses to physical memory addresses.
- Stored in memory.

Speedups: TLB

- Cached page table entries.
- Stored in a cache between CPU and memory.
- Accessing TLB is faster than accessing memory.

Virtual Memory



UNIX File System

Why a file system?

- Common themes in UNIX systems:
  - User oriented
  - Multiple applications
  - Time sharing
- Need a way to store and organize persistent data

Key question: how to let users organize and locate their data on persistent storage?

UNIX File System

Key abstraction

- Data is organized into “files”
  - A linear array of bytes of arbitrary length
  - Metadata about the bytes (modification and creation time, owner, permissions)
- Files organized into “directories”
  - A list of other files or sub-directories
- Common root directory named "/"
- Contrast with drive letters in Windows

UNIX File System Layers

Block layer Purpose: organizes persistent storage into fix-sized blocks
Names: integer block numbers
Values: fix-sized “blocks” of contiguous persistent memory

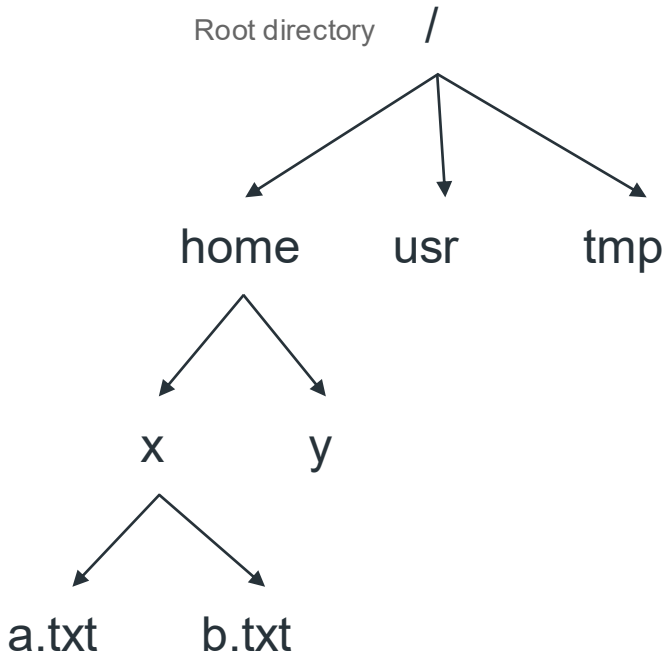
File layer Purpose: organizes blocks into arbitrary-length files
Names: Inode struct
Values: Arrays of bytes up to size N

Inode number layer Purpose: names files as uniquely numbered inodes
Names: Inode numbers
Values: Inode struct

Directory layer Purpose: human-readable names for files in a directory
Names: Human readable names with “directory”
Values: Inode numbers

Absolute path name layer Purpose: a global root directory
Usually assigned with a hardcoded inode number

Example:



Task 1: Check if “a.txt” is in “/home/x/” (lookup)

Suppose we’ve already known that the inode number of “x” is #1000. (**Directory layer**)

1. Investigate the inode struct whose inode number is #1000

    a. #1000 → Inode struct ***I***#1000

    b. Inode number layer

2. Get list of physical block numbers in ***I***#1000

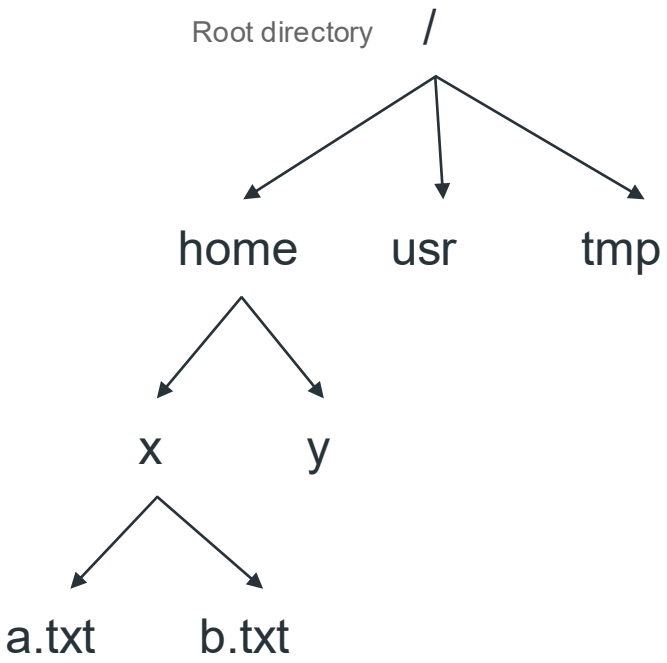
    a. File layer

3. Read the byte array stored in those blocks based on the given block numbers

    a. …| a.txt #1101 | b.txt #1102 |...

    b. Block layer

Example:



Task 2: Read /home/x/a.txt (read)

Assume inode number of “/” is #1. (**Absolute path name**)

1. Start from “/”, get the byte array from the blocks following the previous procedure shown in task 1

   a. #1 -> …| home #2 | usr #4 | tmp #6 |...

2. The sub-dir “home” is in “/”. Its inode number is 2. Repeat the previous procedure

   a. #2 -> …| x #20 | y #24 |...

3. The sub-dir “x” is in “/home/. It is #20. Repeat..

   a. #20 -> …| a.txt #35 | b.txt #2484 |...

4. Finally, we find “a.txt”

   a. #35 -> …| b”welcome to” | b”the precept” |...