CS 1217 Operating Systems

Lecture 2 – The Process Abstraction

Logistics

- Course caps: Have asked the OAA to increase it to 80; still waiting on them
- TAs
  - Ahlah Husain
  - Soham Bagchi
  - Karan Handa
  - Neil Chowdhary
  - Aaryann Mavani
  - Bhavesh Neekhra
- Contacting course staff: cs1217-staff@googlegroups.com **only**

Logistics

- Partner signup
  - Should have done by now.
  - If you have questions/concerns, chat with us.
- Assignment # 1 – super low weightage – will be sent out today
  - Mostly designed to get you started
  - Also, the easiest one that you will get this semester
- Office hours
  - I will be available for discussions after each class
- TA Office Hours
  - Hound them mercilessly during assigned times
  - Other times, leave them alone : they also have stuff to do
  - **Absolutely NO CALLING the TAs**

Logistics

- Partners
  - Communication
  - Patience
  - Commitment
- Questions / Email
  - Use discussions on classroom for assignment related questions
    - Other students might have the same doubts; reduces effort duplication on our side
  - If have to use email, use the staff email list
    - One of us will respond
- Remember: TAs enforce policy; instructor decides it

Screen Policy Reminder

- Put your screens away, if haven’t already
- Last of the “gentle” reminders

Recap

- OS is a ___
  - computer program that eases the use of a computing system
  - Does so by providing a number of abstractions, managing hardware resources
- Abstractions : what and why?
  - Hide implementation details, makes it easier to reason about certain problems
- Policy vs Mechanism
  - In case of Operating Systems, abstractions help decide **policy**, without having to worry about the **mechanism**.
  - Mechanisms for the same policy might be different across systems

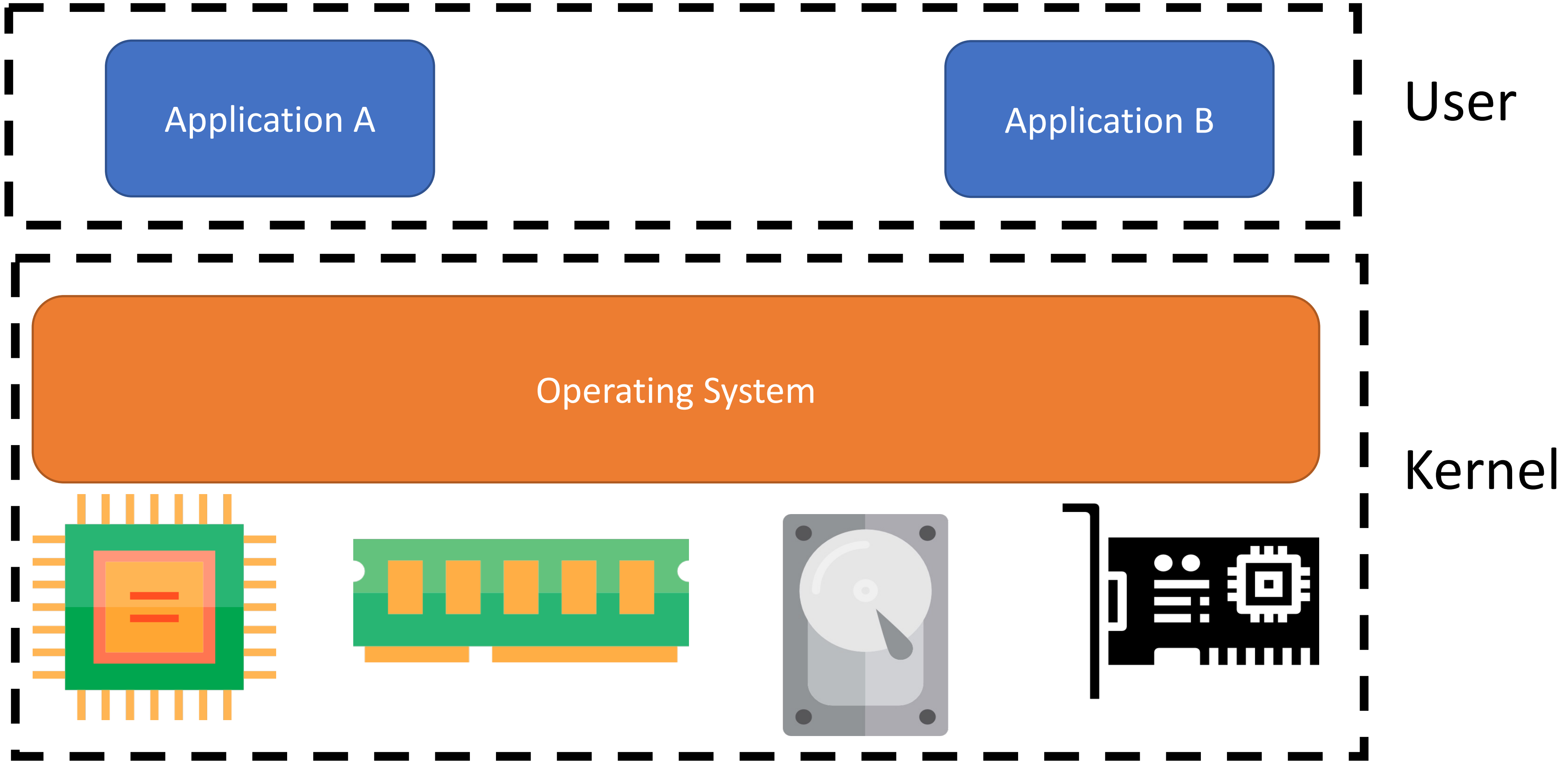
Various OS Abstractions

- Processes (and threads) abstract the **CPU**
- Address spaces and virtual memory abstract the **memory**
- Files (and less so, directories) abstract **storage**

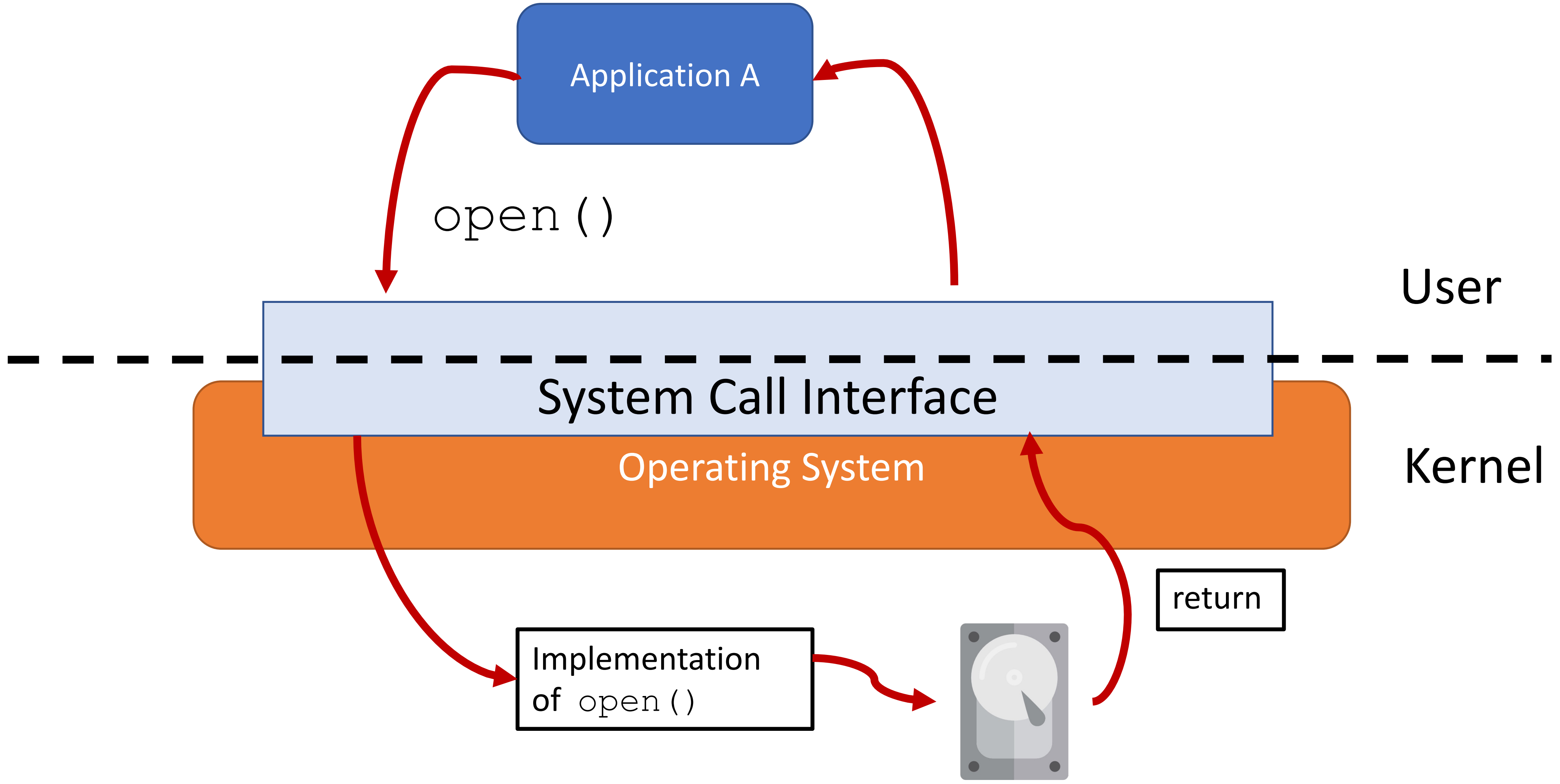
Example of Abstraction

- Files and File systems
- Hides?
- Makes simpler?

OS Structure – User vs Kernel Mode



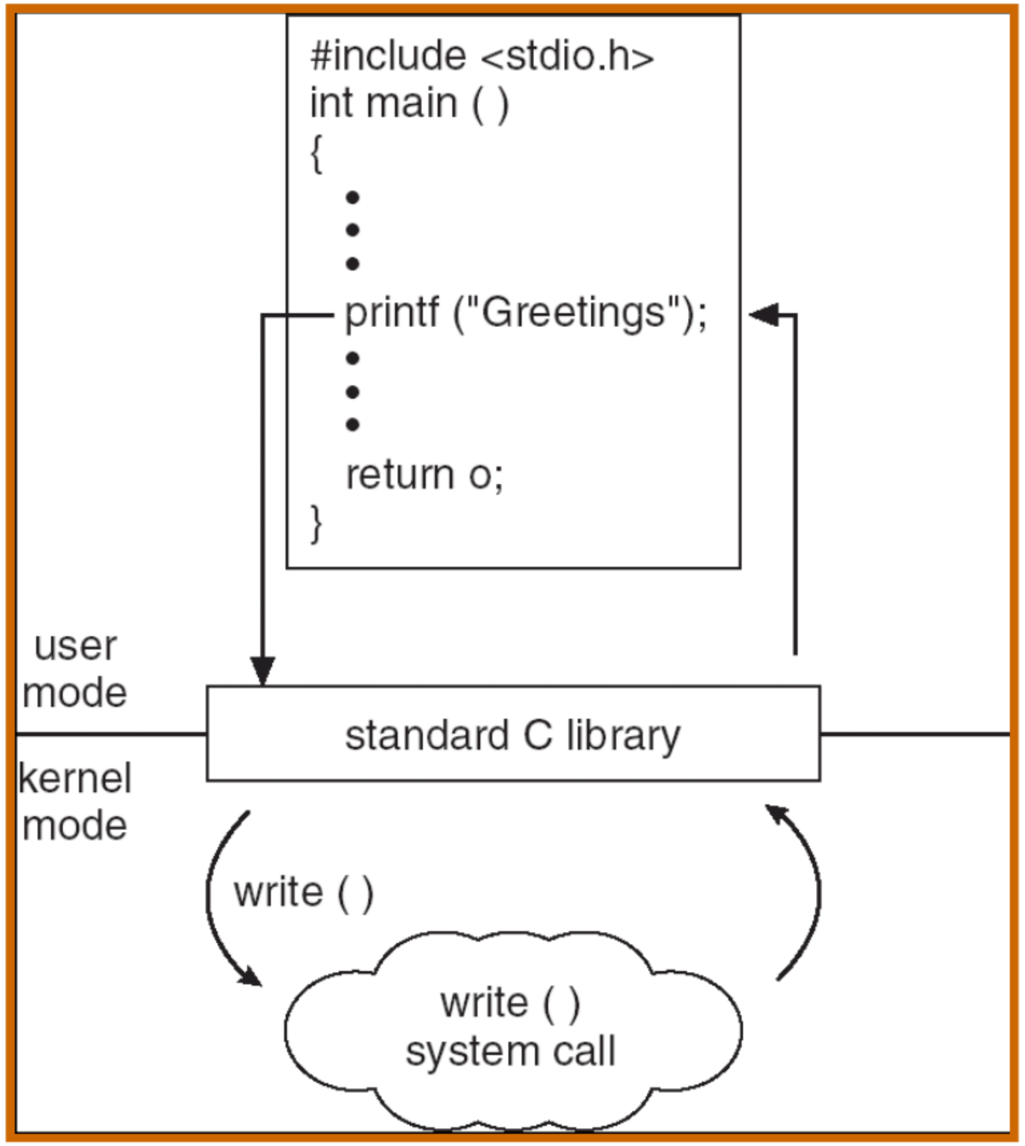
Availing OS Services : System Call Interface



Why?

- Goal: Do things application can’t do in unprivileged mode
  - Very similar to a library call, but into a privileged kernel code
- Kernel supplies **well-defined** *system call* interface
  - Applications set up syscall arguments and *trap* to kernel
  - Kernel performs operation and returns result
- Syscalls are an API; provides higher level, library-like functions
  - *printf*, *scanf*, *fgets*, etc. all user-level code
- Example: POSIX/UNIX interface
  - *open, close, read, write, ...*

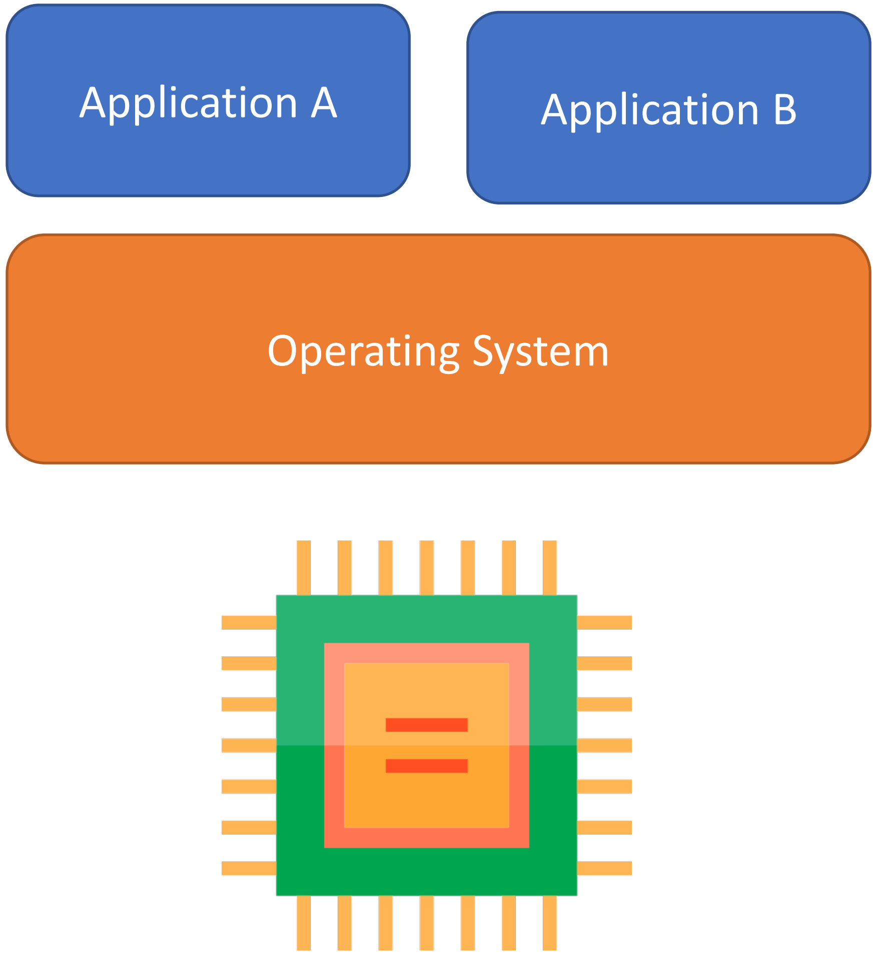
System call example



```
#include <stdio.h>
int main ( )
{
   •
   •
   •
   printf ("Greetings");
   •
   •
   •
   return o;
}
```

- Standard library implemented in terms of syscalls
  - **printf** – in libc, has same privileges as application
  - calls **write** – in kernel, which can send bits out serial port

The Process Abstraction

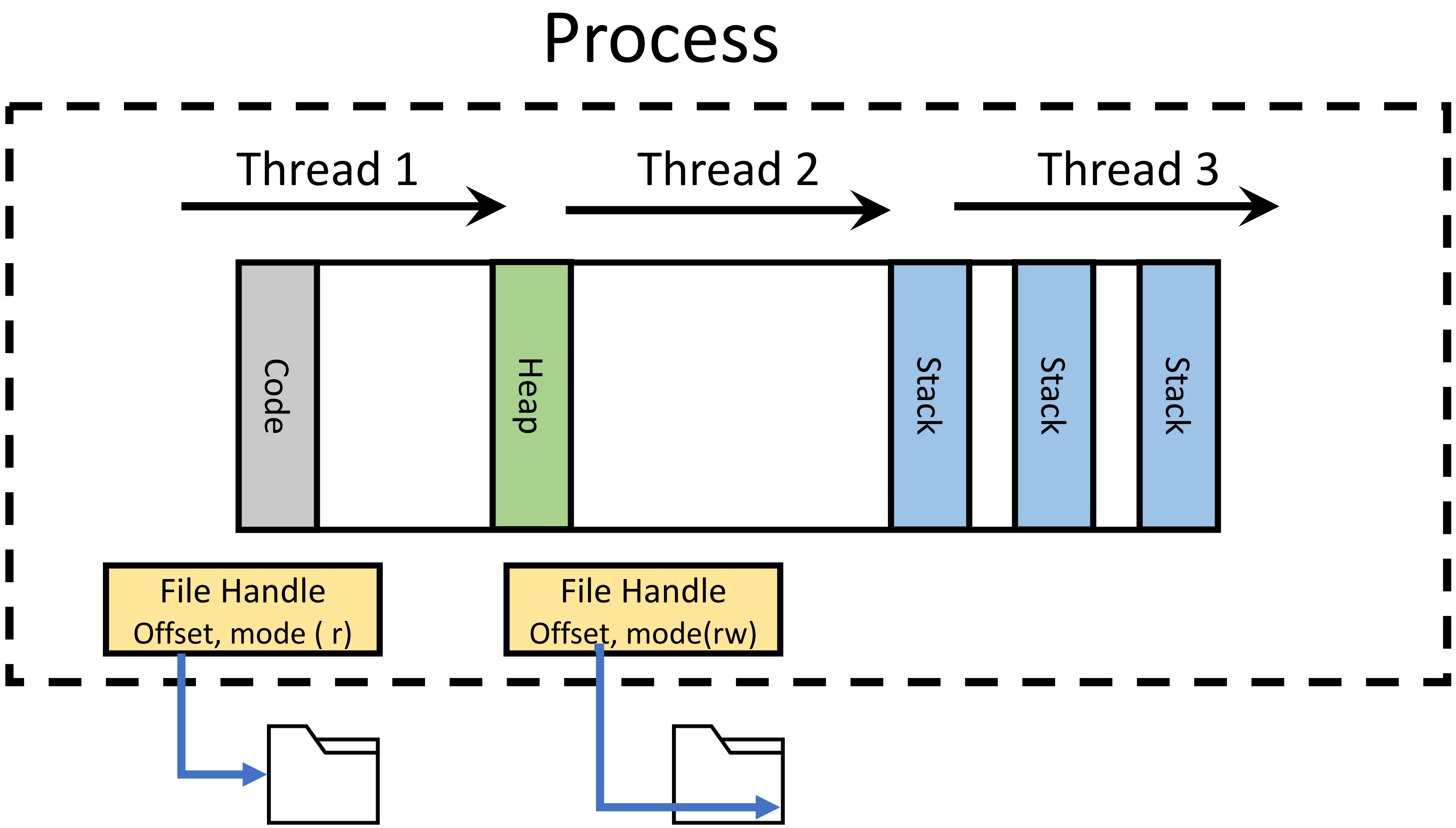


- A binary is a dead entity; process is the alive version
- Process: Abstraction that encapsulates this live entity
- Provide "illusion" to processes that they are the only ones executing
- Isolate programs from one another

Why do you need this abstraction?

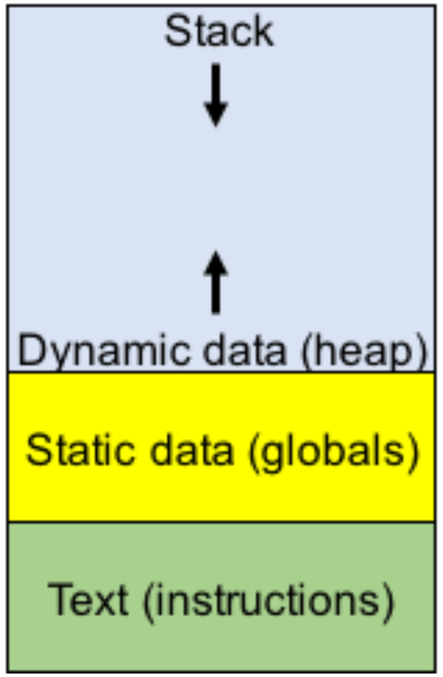
- Each process gets its own view of machine
  - Its own address space
  - Its own open files
  - Its own virtual CPU (through preemptive multitasking)
- Greatly simplifies programming model
  - Powerpoint does not care that chrome is ***also*** running
- Sometimes want interaction between processes
  - Simplest is through **files**: emacs edits file, gcc compiles it
  - More complicated: **error codes**, **pipes**, **signals**, **shared memory**

The Process Abstraction



What constitutes a process?

- What does a running program do?
  - Reads/writes data, maintains its “structure” in memory
  - Fetches and executes instructions
    - Maintains content in registers
  - Read and writes to files
- A running process has to maintain this state
  - The OS helps the process do that



Process Information

- Processes are a **collection** of other abstractions and contain
- Threads
  - One or more
- Address Spaces
  - One or more
- List of open files
  - Zero or more

If a program is generating random strings and writing them out to the terminal, what file does it **definitely** have open?