COMP435: *SECURITY CONCEPTS!*

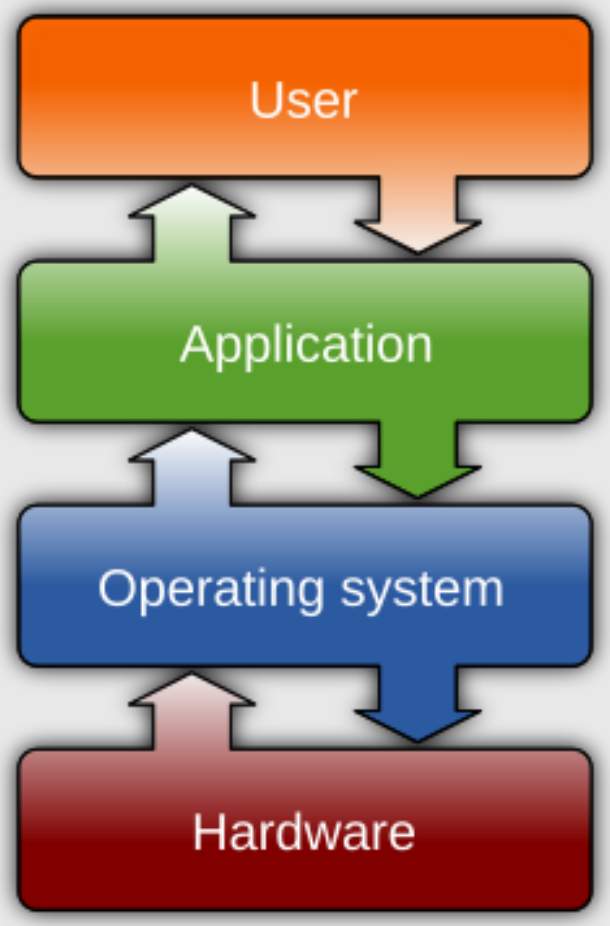
Lecture 12: Operating Systems Security Cont.

tinyurl.com/comp435-fa25

OPERATING SYSTEMS

What is the Operating System??

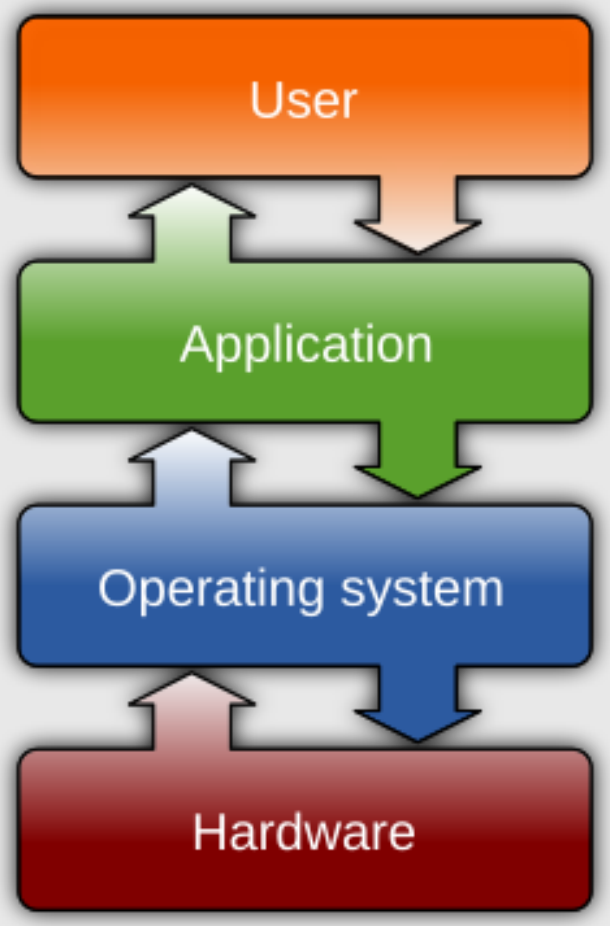
Software that **manages** hardware and software **resources**, and **provides common services** for software programs



Source: https://en.wikipedia.org/wiki/Operating_system

What is a System Call??

A system call (syscall) is the mechanism by which a program in user space asks the operating system to do something on it's behalf.



Source: https://en.wikipedia.org/wiki/Operating_system

OS Responsibilities

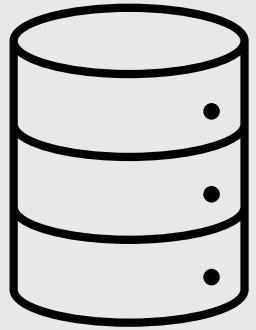
- manage resources
  - *fork(), mmap()*
- provide interface to HW
  - *read(), write()*
- provide user authentication
  - *getuid(), setuid()*
- mediate interprocess communication
  - *pipe()*
- protect itself
  - *kill(), mprotect()*

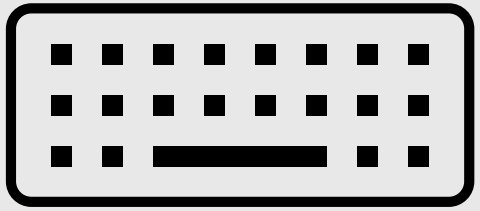
Manage Resources

- allocation
- sharing
- protection

Provide Interface to Hardware

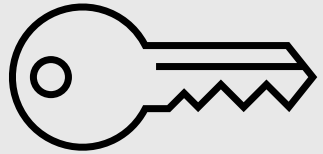
- memory
- file system
- device I/O

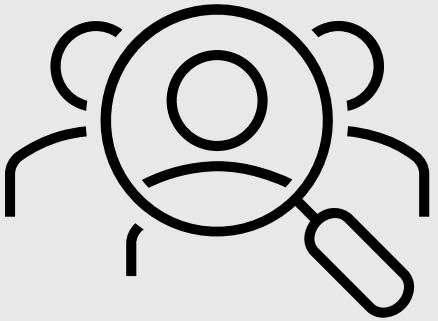




Provide User Authentication

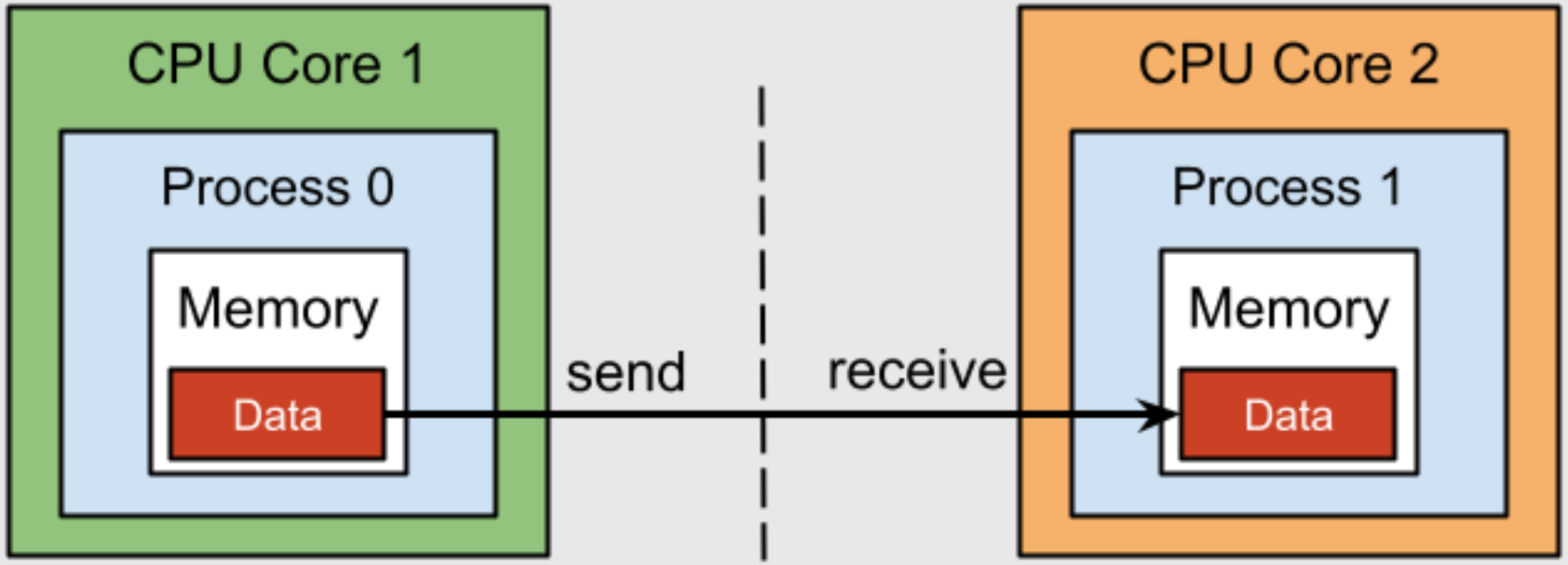
- user accounts
- roles
- passwords





Mediate Interprocess Communication

- shared memory
- message passing



Protect the Operating System

- data
- code
- permissions

RACE CONDITIONS

Race Condition

Def’n: Concurrent access of a resource is not serializable

Occurs when:

- Two or more processes have access to **the same resource**
- The final state of the resource depends on the **relative timing** of the **two processes**

ATM

Bank

balance = 900

withdraw $100

get_balance()

balance = 900

$100

set_balance(800)

balance = 800

ATM

Bank

ATM

balance = 900

w/d $100

w/d $50

get_bal()

get_bal()

bal = 900

bal = 900

$100

$50

set_bal(800)

set_bal(850)

balance = ??

ATM

Bank

ATM

balance = 900

w/d $100

get_bal()

get_bal()

w/d $50

bal = 900

bal = 900

$100

$50

set_bal(800)

set_bal(850)

balance = ??

```
filename = “tmpname”;
```

```
// Write to the file
fd = open(filename, O_CREATE|O_RDWR);
```

`Symlink` System Call

- creates a symbolic link (soft link) in the filesystem.
- Ex: `symlink(target, linkpath);`
    - If you open and read the linkpath, now the system is going to follow the link and read target
    - After running this you have,
      linkpath ⟶ target

```
filename = “tmpname”;
           symlink(“/etc/passwd”, “tmpname”);

// Write to the file
fd = open(filename, O_CREATE|O_RDWR);
```

Program A: Privileged Program!

```
1.  filename = "tmpname";
    // Write to the file
2.  fd = open(filename,O_CREATE|O_RDWR);
```

Program B: Attacker!

```
1.  symlink("/etc/passwd", "tmpname");
```

Problem: TOCTTOU
Race condition!

Time of Check to Time of Use (TOCTTOU)

Def’n: a particular type of race condition in which the gap between checking permissions and using permissions can be exploited

Under the hood of the open syscall

```
if(access(fname,W_OK) == 0) {


   fd = open(fname,O_WRONLY);
}
```

```
if(access(fname,W_OK) == 0) {
                    unlink(fname);
                    symlink("\etc\passwd", fname);
   fd = open(fname,O_WRONLY);
}
```

MEMORY SEPARATION

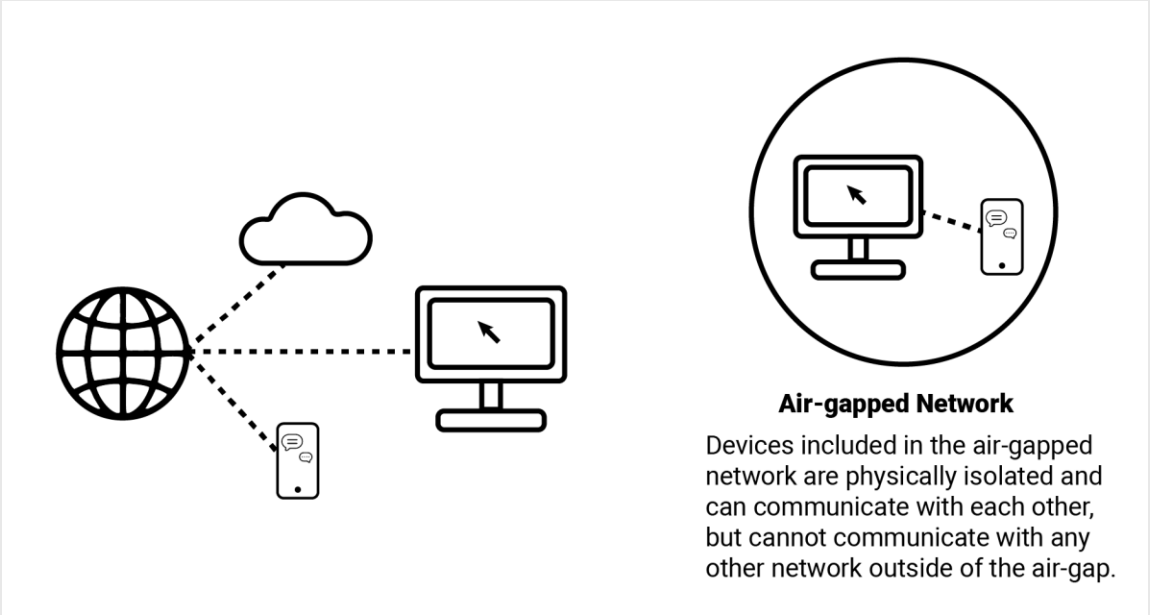
Styles of Separation

- Physical
- Temporal
- Logical
- Cryptographic

Styles of Separation

- ***Physical***
- Temporal
- Logical
- Cryptographic

not efficient, not complex, secure



Source: https://www.belden.com/blog/network-air-gapping

Styles of Separation

- Physical
- ***Temporal***
- Logical
- Cryptographic

more efficient, more complex, less secure

Object reuse requires sanitization!

Styles of Separation

- Physical
- ***Temporal***
- Logical
- Cryptograp

Objec
s

c

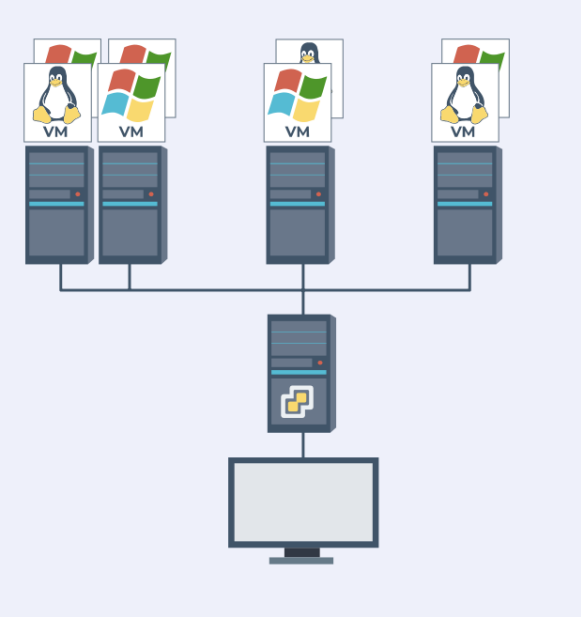
Copy code

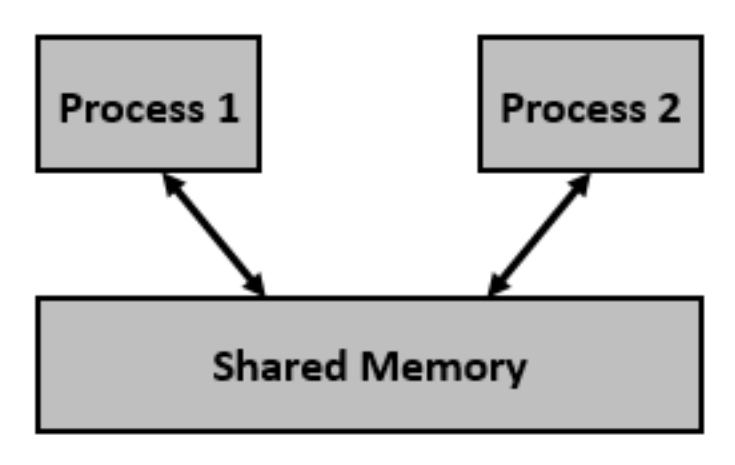
```c
#include <string.h>
#include <stdlib.h>

int main() {
    char *password = malloc(100);
    if (!password) return 1;

    // password is used here...
    strcpy(password, "SuperSecret123!");

    // ✅ Object sanitization: overwrite before freeing
    memset(password, 0, 100);

    free(password);
    return 0;
}
```

Styles of Separation

- Physical
- Temporal
- ***Logical***
- Cryptographic

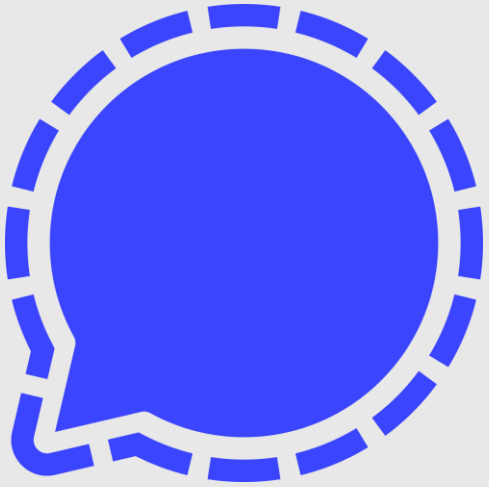
more efficient, more complex, less secure





Styles of Separation

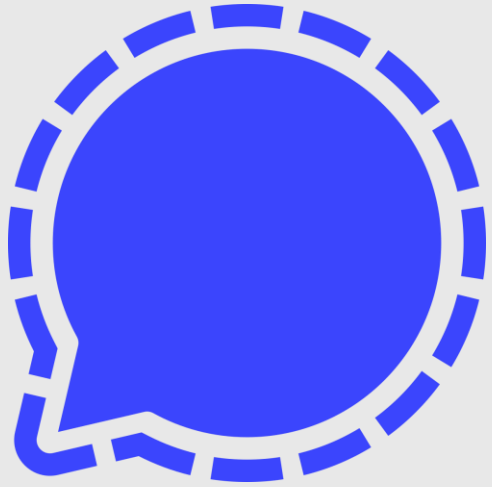
- Physical
- Temporal
- Logical
- Cryptographic

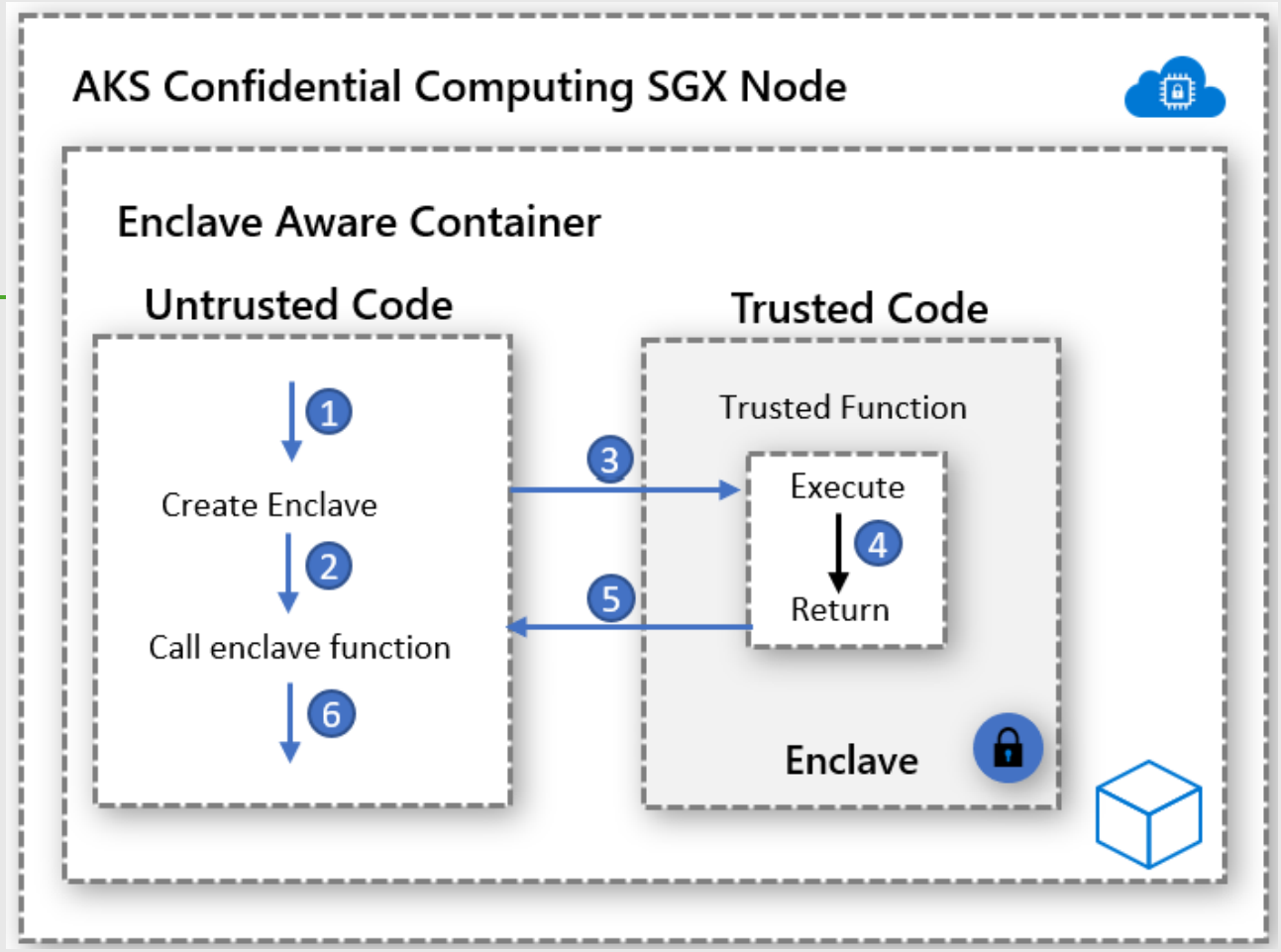


issues of key management

Styles of Separation

- Physical
- Temporal
- Logical
- Cryptographic





Some examples! (Q1)

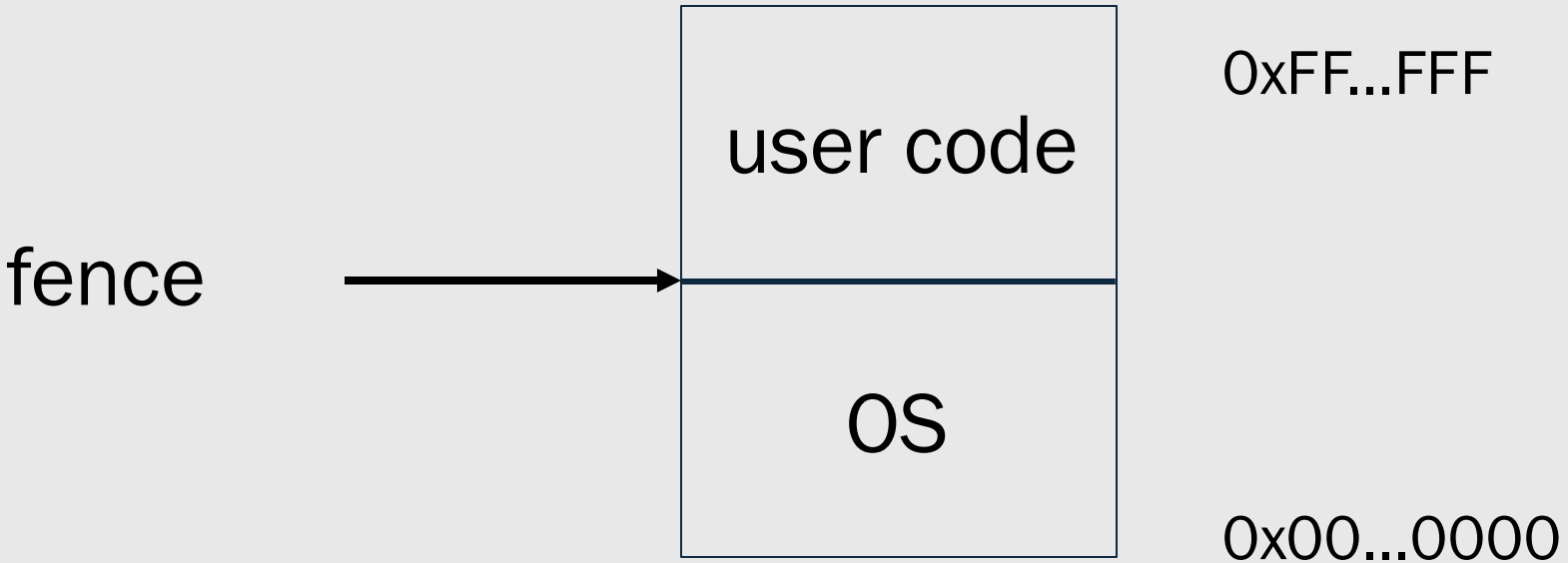
MEMORY PROTECTION MECHANISMS

OS Provided Memory Protection Mechanisms

- fence
- base--bounds registers
- segment addressing

Memory Fence

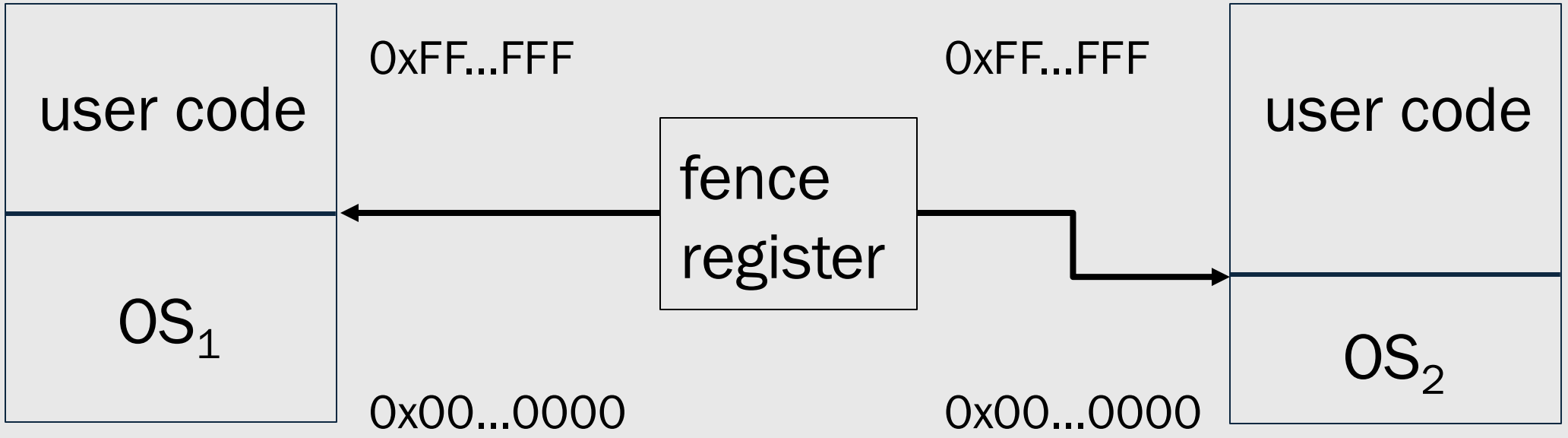
Physical Separation



Fence: hardcoded address in processor that denotes starting point of OS code

Memory Fence

Logical Separation

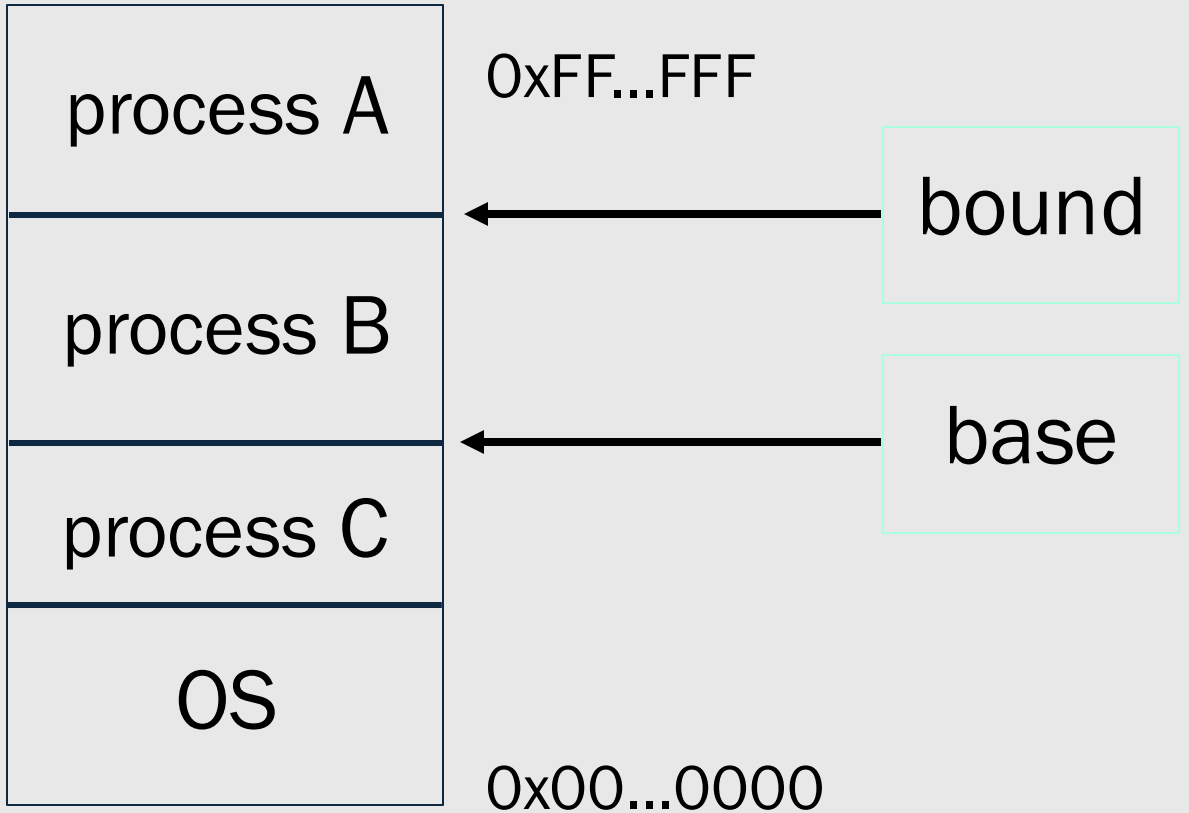


configuration 1

configuration 2

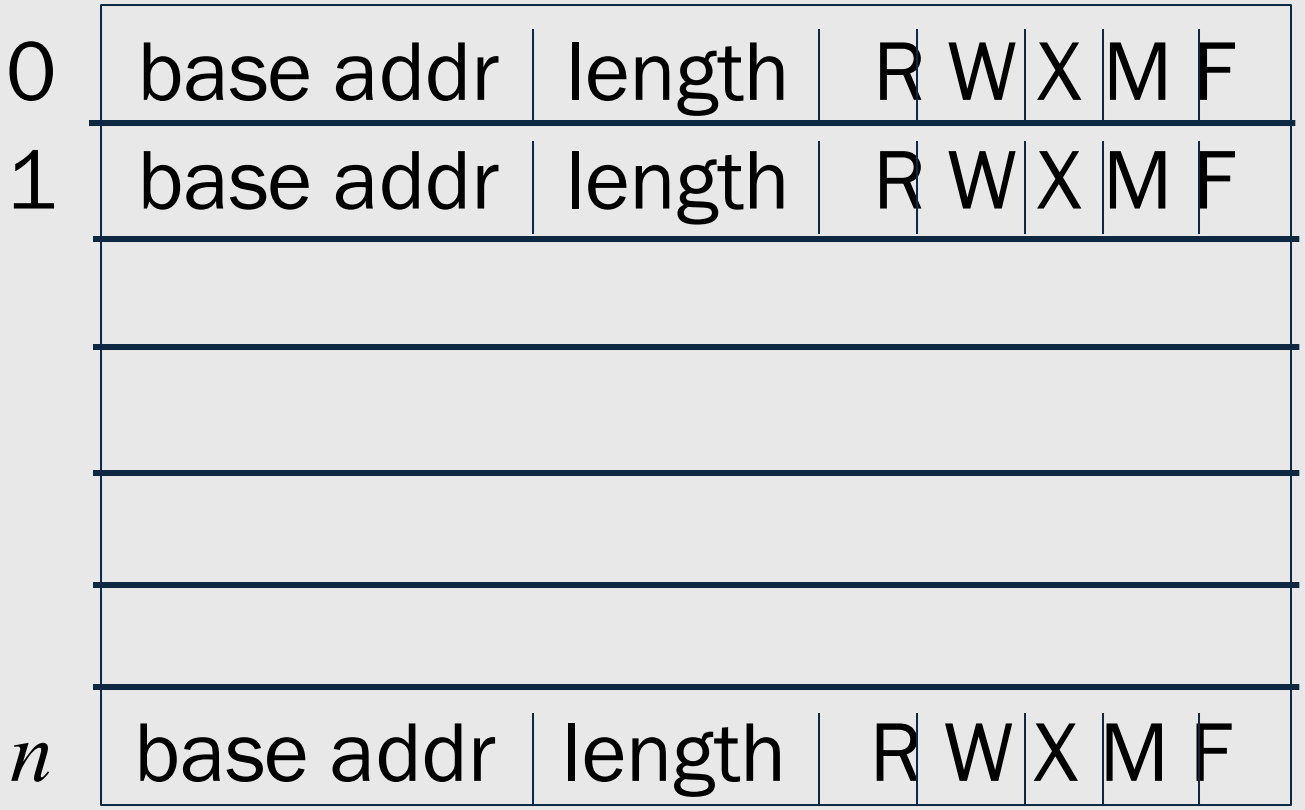
Fence now configurable

Base–Bounds Registers



Base & Bounds Registers: prevent code from one process from interfering with w/ another

Segment Addressing

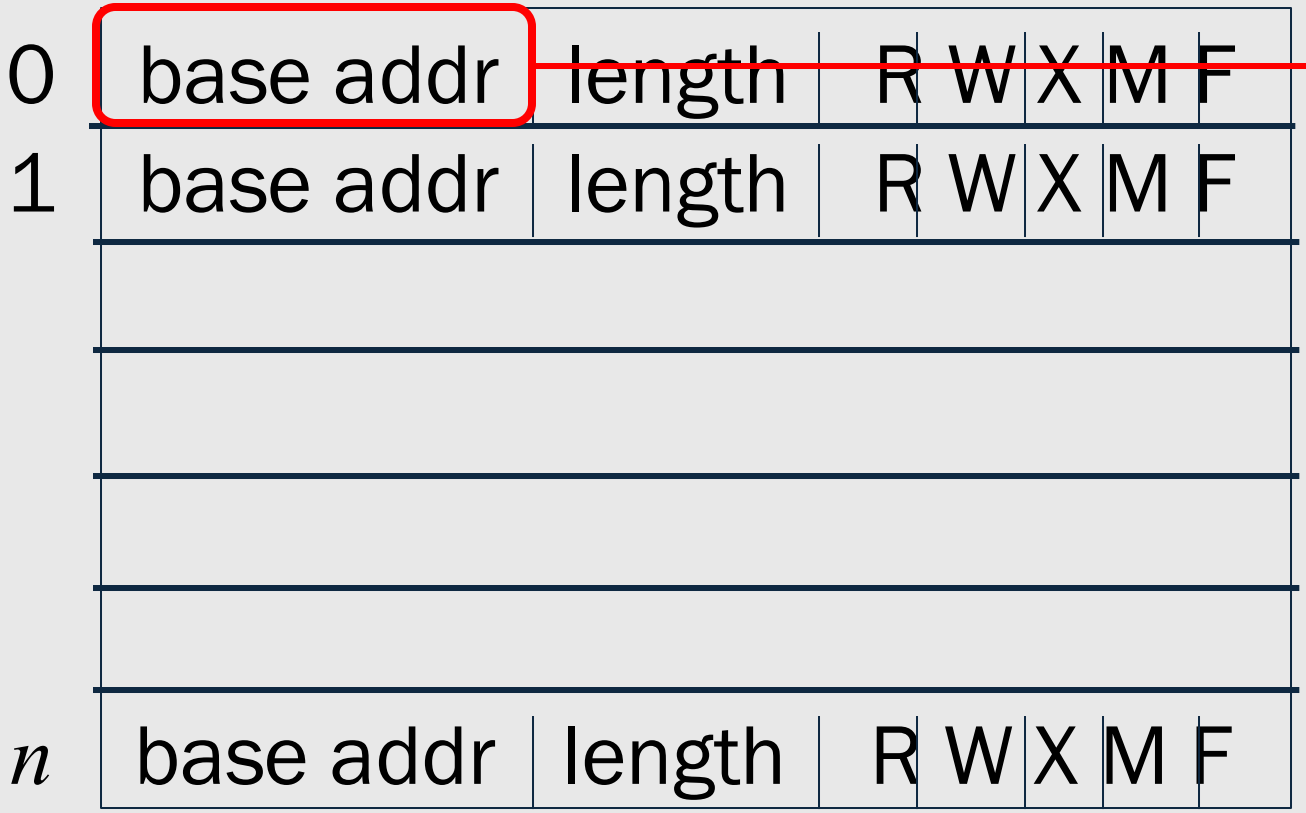


segment descriptors
for process A

Introduces Virtual Memory: gives each process an isolated view of memory

Allows for separate access permissions (read, write, execute) for a specified memory region

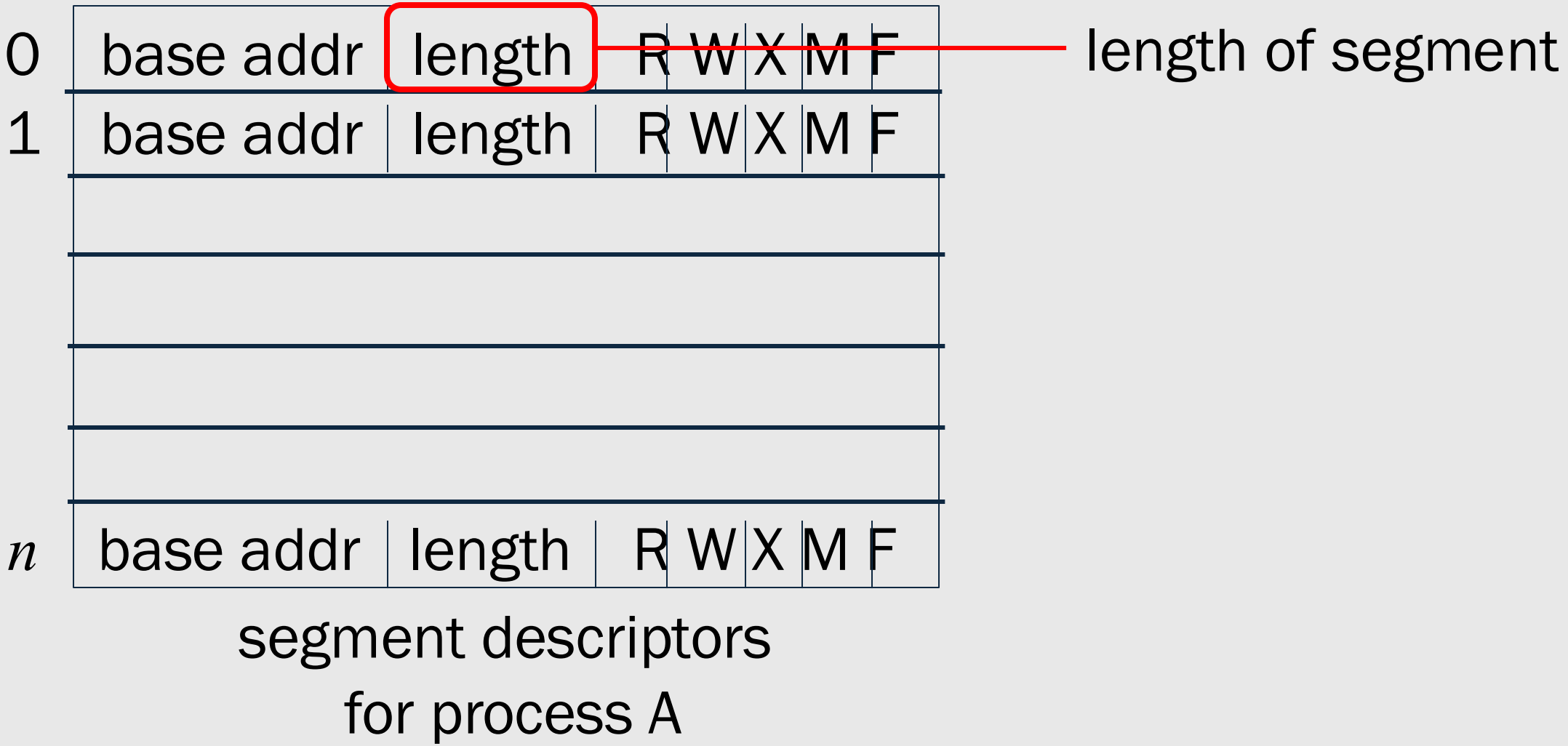
Segment Addressing



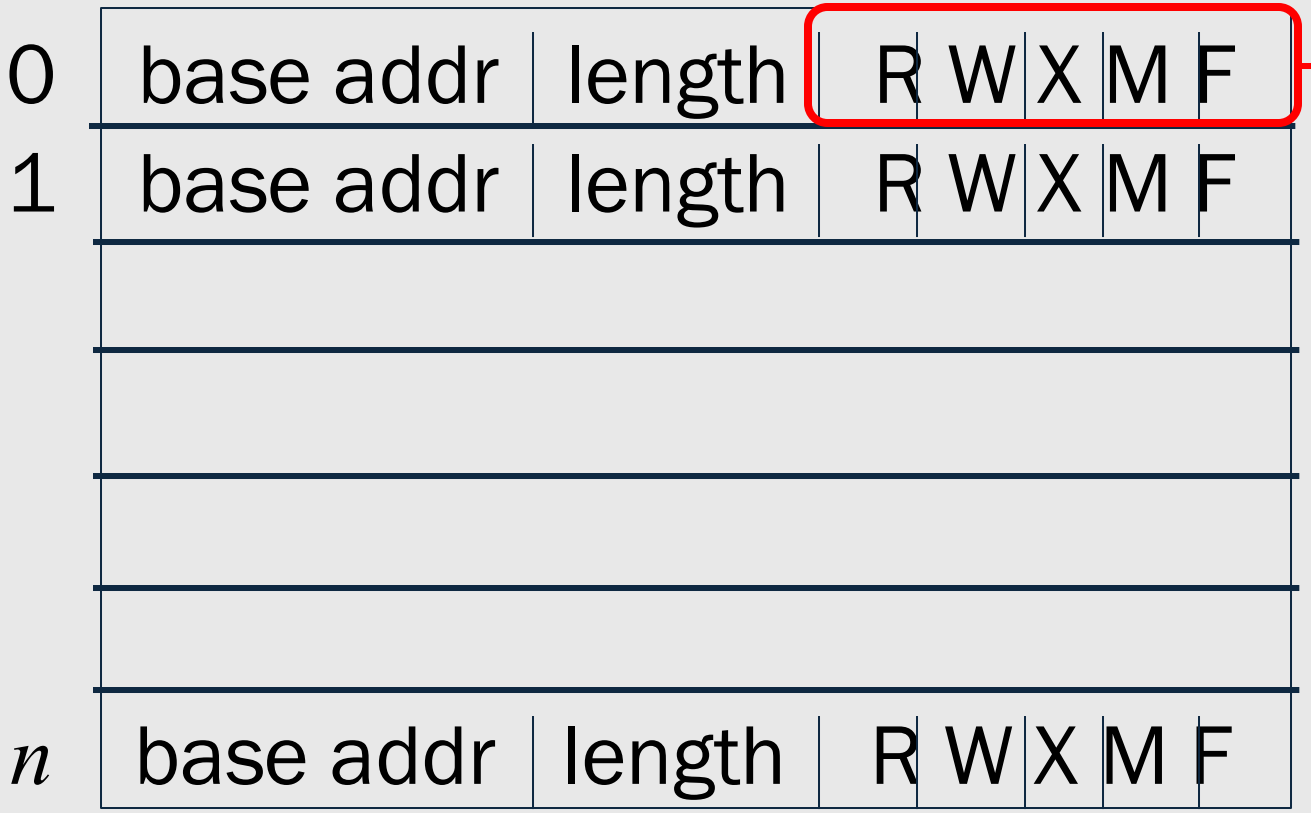
physical starting address of segment

segment descriptors for process A

Segment Addressing



Segment Addressing

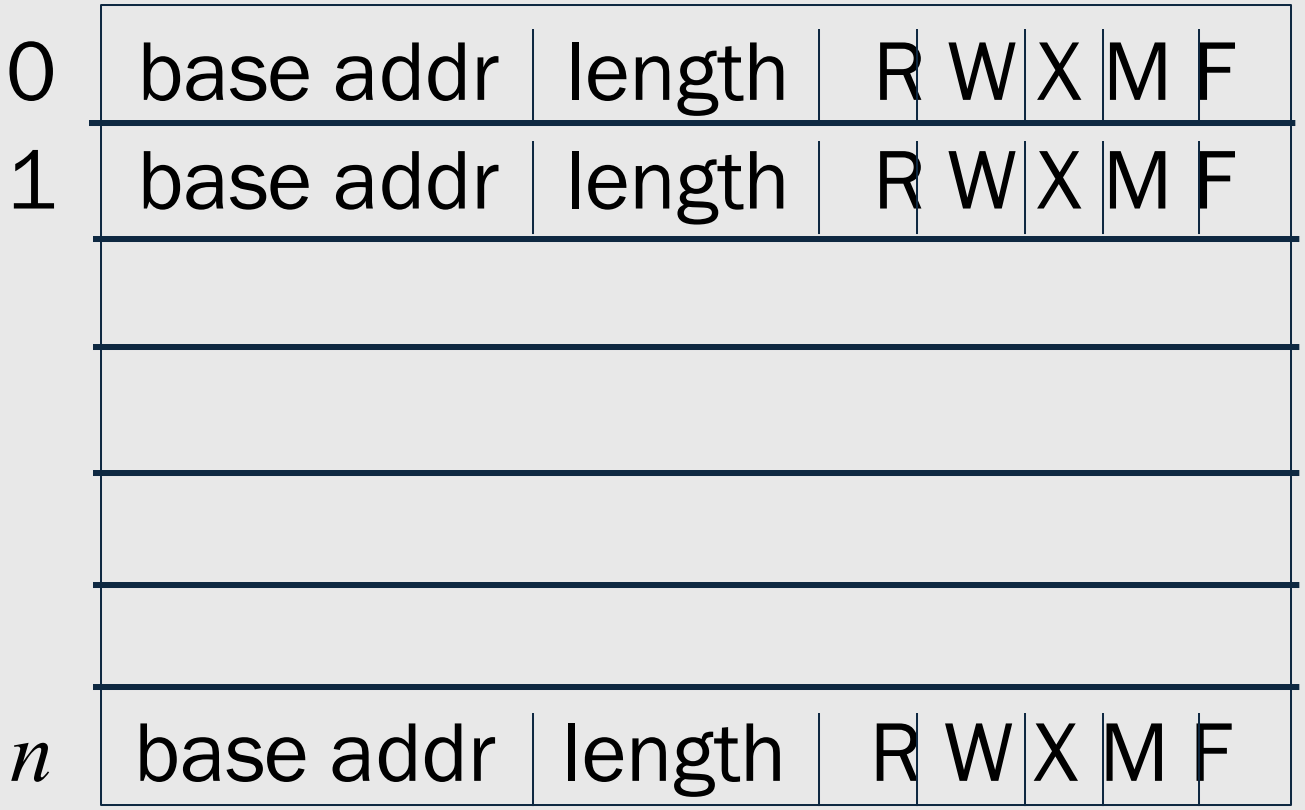


access control indicators

segment descriptors for process A

There is a bit for read, write, execute, mode (execute as supervisor?), fault

Segment Addressing



segment descriptors
for process A

<segment #, offset>

addressing

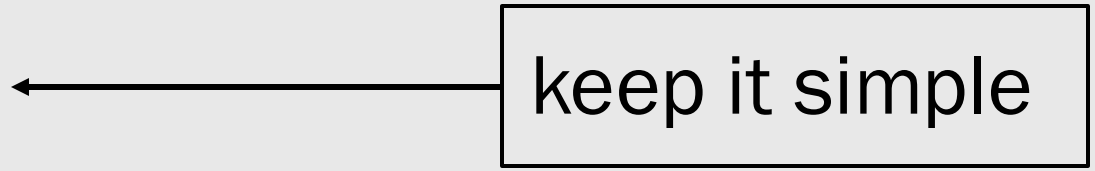
DESIGN PRINCIPLES

Design Principles (Saltzer and Schroeder)

- economy of mechanism
- fail safe defaults
- complete mediation
- open design
- separation of privilege
- least privilege
- least common mechanism
- psychological acceptability

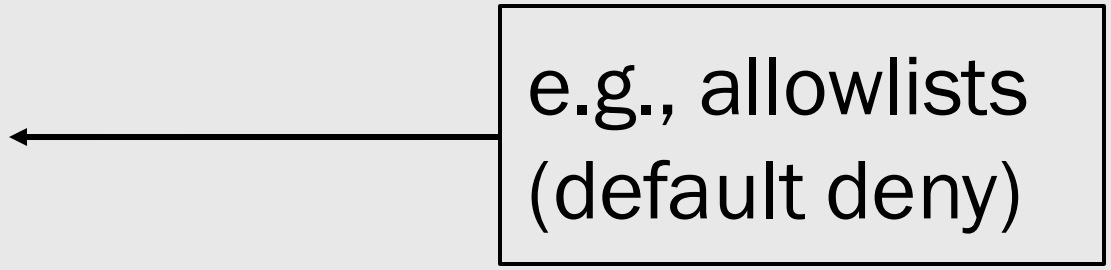
Design Principles (Saltzer and Schroeder)

- Economy of mechanism
- fail safe defaults
- complete mediation
- open design
- separation of privilege
- least privilege
- least common mechanism
- psychological acceptability



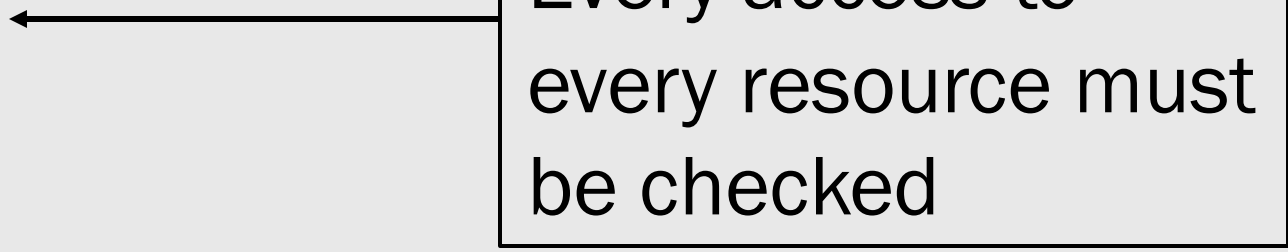
Design Principles (Saltzer and Schroeder)

- economy of mechanism
- fail safe defaults
- complete mediation
- open design
- separation of privilege
- least privilege
- least common mechanism
- psychological acceptability



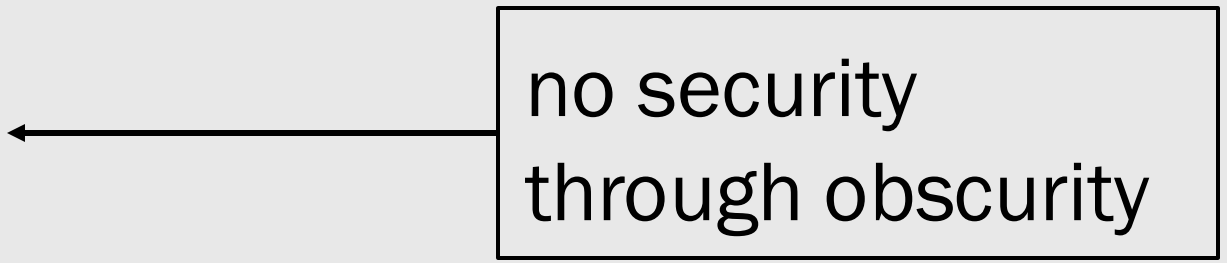
Design Principles (Saltzer and Schroeder)

- economy of mechanism
- fail safe defaults
- complete mediation
- open design
- separation of privilege
- least privilege
- least common mechanism
- psychological acceptability



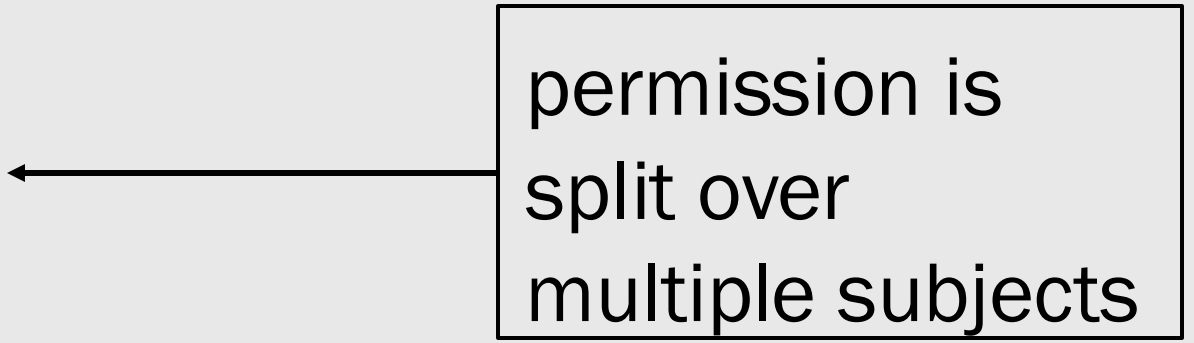
Design Principles (Saltzer and Schroeder)

- economy of mechanism
- fail safe defaults
- complete mediation
- open design
- separation of privilege
- least privilege
- least common mechanism
- psychological acceptability



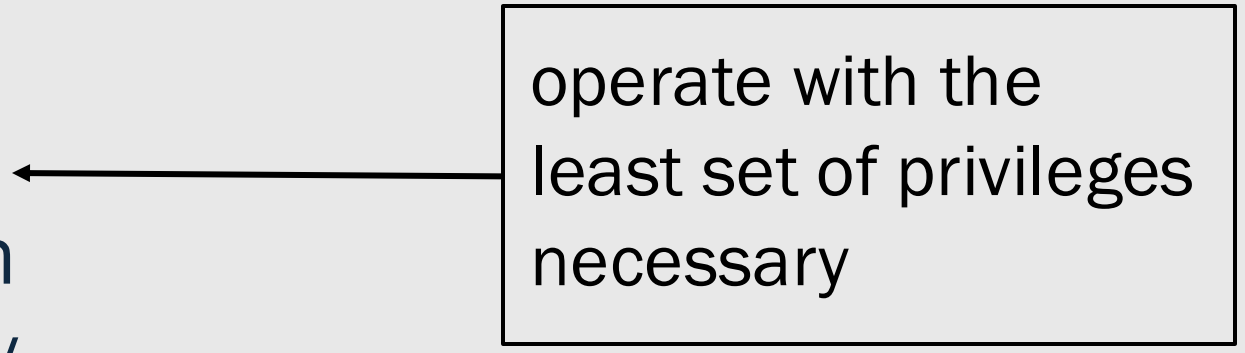
Design Principles (Saltzer and Schroeder)

- economy of mechanism
- fail safe defaults
- complete mediation
- open design
- separation of privilege
- least privilege
- least common mechanism
- psychological acceptability



Design Principles (Saltzer and Schroeder)

- economy of mechanism
- fail safe defaults
- complete mediation
- open design
- separation of privilege
- least privilege
- least common mechanism
- psychological acceptability



Design Principles (Saltzer and Schroeder)

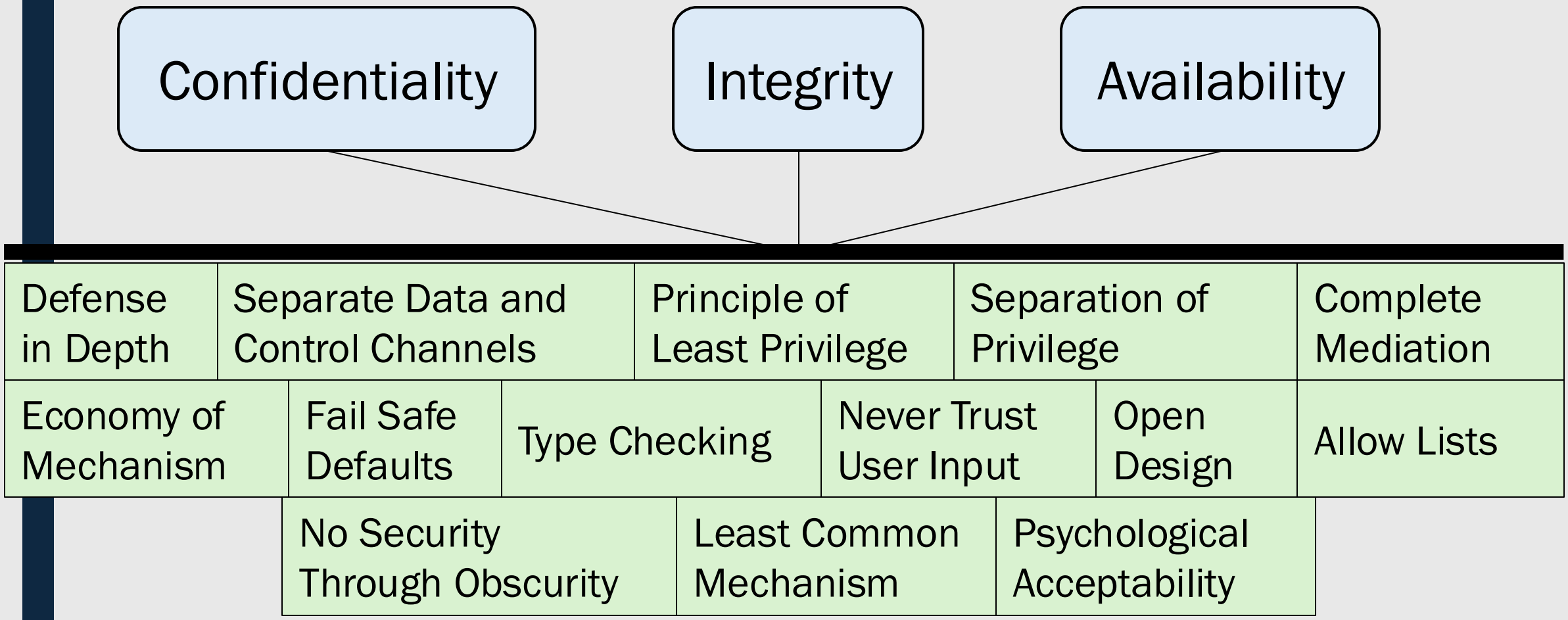
- economy of mechanism
- fail safe defaults
- complete mediation
- open design
- separation of privilege
- least privilege
- least common mechanism
- psychological acceptability

Limit shared and trusted mechanisms

Design Principles (Saltzer and Schroeder)

- economy of mechanism
- fail safe defaults
- complete mediation
- open design
- separation of privilege
- least privilege
- least common mechanism
- psychological acceptability

User interface should be intuitive and easy


Confidentiality
Integrity
Availability
Defense in Depth
Separate Data and Control Channels
Principle of Least Privilege
Separation of Privilege
Complete Mediation
Economy of Mechanism
Fail Safe Defaults
Type Checking
Never Trust User Input
Open Design
Allow Lists
No Security Through Obscurity
Least Common Mechanism
Psychological Acceptability

Some examples! (Q4)