SER 334 A Session

SI Session

Sunday, February 4th 2024

7:00 pm - 8:00 pm MST

Agenda

- Critical Section & Solutions
  - Hardware Solutions
  - Peterson’s Solution
  - Mutexes
  - Semaphores
  - Monitors

SI Session Expectations

Thanks for coming to the **SER 334** SI session. We have a packed agenda and we are going to try to get through as many of our planned example problems as possible. This session will be recorded and shared with others.

- If after this you want to see additional examples, please visit the drop-in tutoring center.
- We will post the link in the chat now and at the end of the session.
  - tutoring.asu.edu
- Please keep in mind we are recording this session and it will be made available for you to review 24-48 hours after this session concludes.
- Finally, please be respectful to each other during the session.

Interact with us:

Zoom Features

Zoom Chat

- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

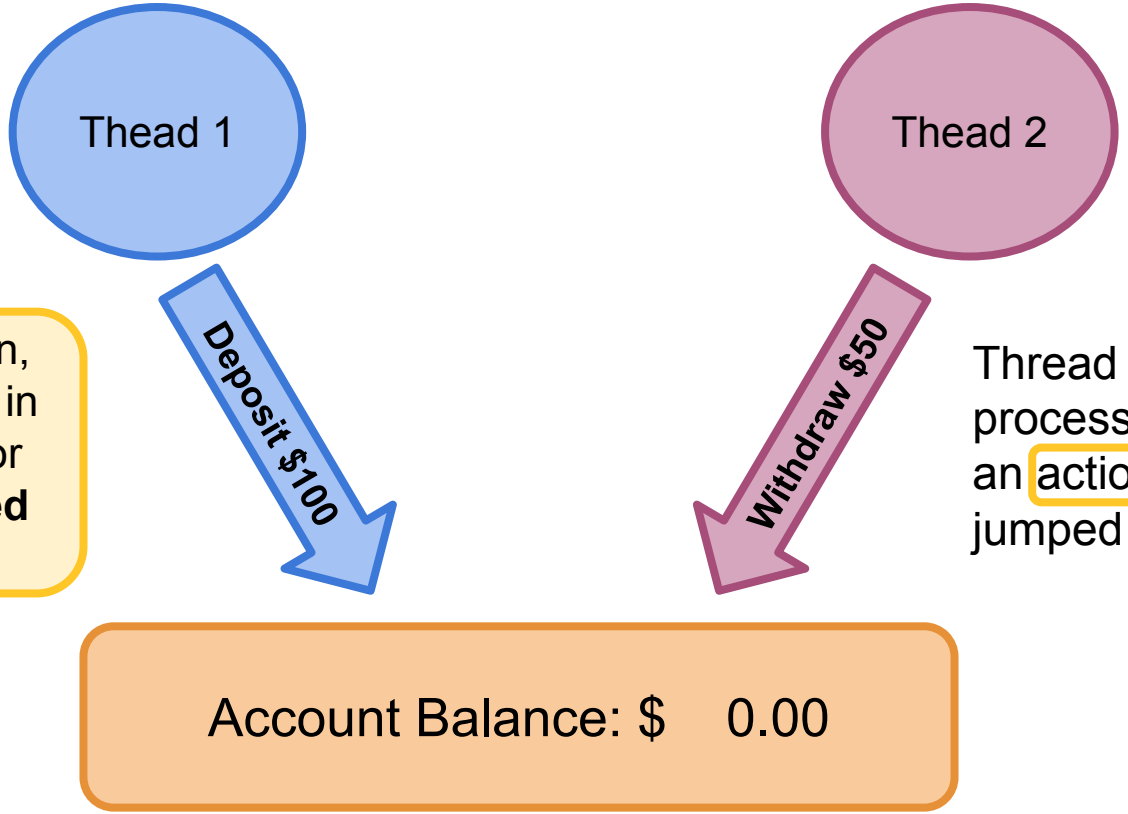
SER 334
Critical Section

Stepping on each other's toes

What is the Critical Section?

Area of code where thread execution can impact other threads

SER 334
Critical Section



Not just any action, actions that result in a **state change** or manipulate **shared resources**.

Thread 1 was in the process of performing an action, and Thread 2 jumped in the middle!

SER 334
Critical Section

Race Condition

Starvation

Deadlock

The solution needs to ensure…

-
-
-

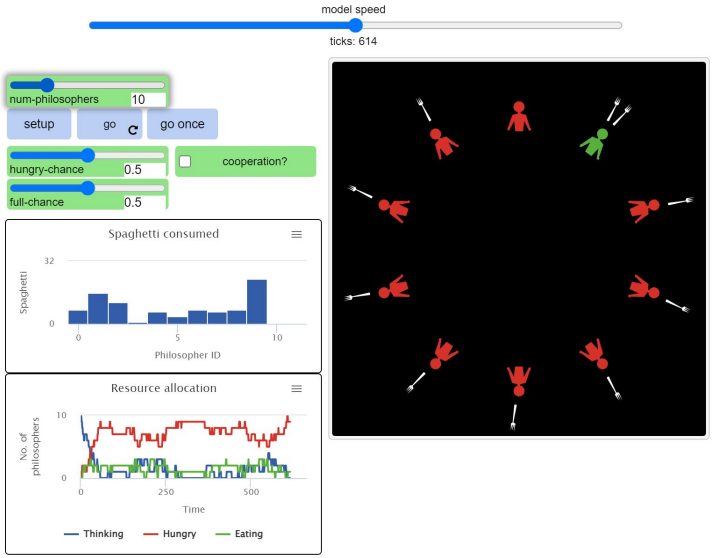
SER 334
Critical Section

Dining Philosophers

Race Condition

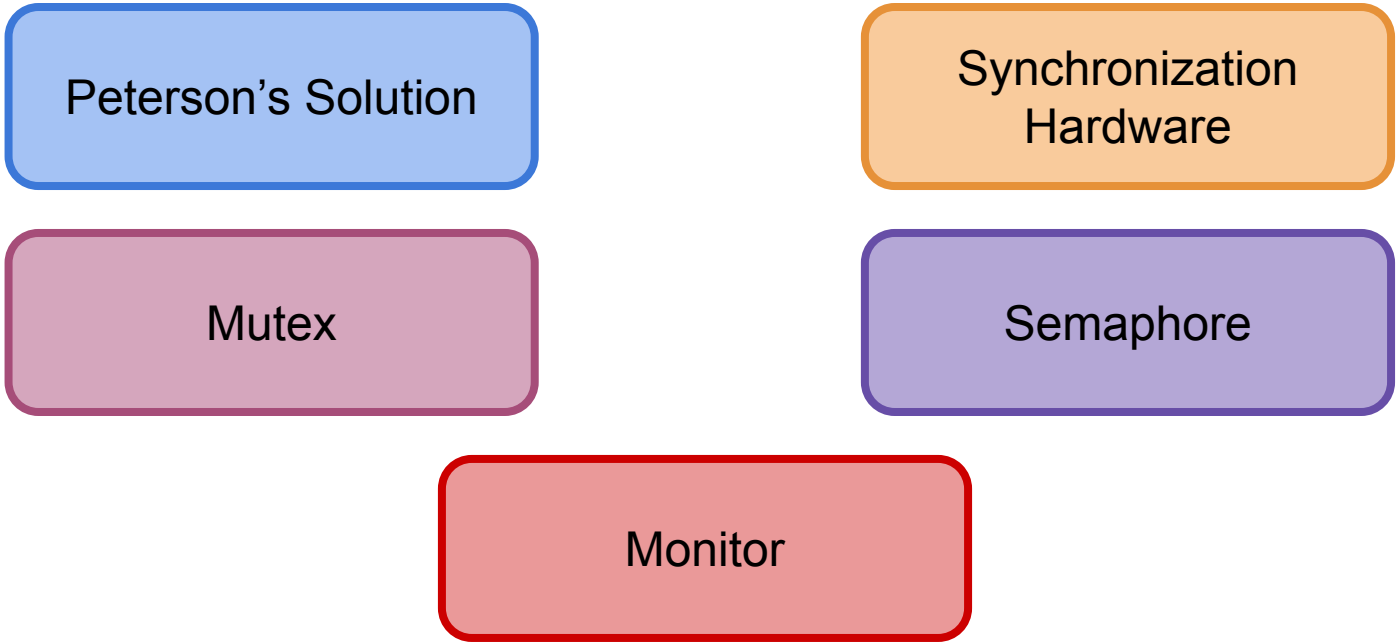
Deadlock

Starvation



SER 334

Critical Section Solutions



SER 334

Peterson's Solution

Peterson's Solution

```
//shared memory
int turn = 0;
bool flag[2] = { false, false };


//for some process i
do {
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j);
    //critical section
    flag[i] = false;
    //remainder section
} while (true);
```

I *want* to go

SER 334

Peterson's Solution

```
//shared memory
int turn = 0;
bool flag[2] = { false, false };
```

```
//for some process i
do {
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j);
    //critical section
    flag[i] = false;
    //remainder section
} while (true);
```

I *want* to go

But you can go first…

SER 334

Peterson's Solution

```
//shared memory
int turn = 0;
bool flag[2] = { false, false };


//for some process i
do {
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j);
    //critical section
    flag[i] = false;
    //remainder section
} while (true);
```

I *want* to go

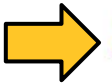
But you can go first…

Wait for my turn

SER 334

Peterson's Solution

```
//shared memory
int turn = 0;
bool flag[2] = { false, false };


//for some process i
do {
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j);
    //critical section
    flag[i] = false;
    //remainder section
} while (true);
```



I *want* to go

But you can go first…

Wait for my turn

Danger Zone

SER 334

Peterson's Solution

```
//shared memory
int turn = 0;
bool flag[2] = { false, false };


//for some process i
do {
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j);
    //critical section
    flag[i] = false;
    //remainder section
} while (true);
```

I *want* to go

But you can go first…

Wait for my turn

Danger Zone

You can go now

SER 334

Synchronization Hardware

Synchronization Hardware

Hardware that has special ***atomic*** actions

```
//shared data
boolean lock = false;

do {
    while (test_and_set(&lock));
    // critical section
    lock = false;
    // remainder section
} while (true);
```

T → T : ⟳

F → T : ⤸

```
//shared data
int lock = 0;

do {
    while (compare_and_swap(&lock, 0, 1)
           != 0);
    // critical section
    lock = 0;
    // remainder section
} while (true);
```

Expected

New Value

If lock is currently zero, set lock to 1

SER 334
Mutex

A Mutex is sort of like a lock on a door

Lock the door behind you

```
//thread safe!
void add_front(int data) {
    struct data_node* node = (struct data_node*)malloc(sizeof(struct data_node));

    node->data = data;

    pthread_mutex_lock(&list_lock);
        node->next = list_head;
        list_head = node;
    pthread_mutex_unlock(&list_lock);
}
```

Already Locked? → In use!

Unlock door as you leave

SER 334
Semaphores

What is the difference between a mutex and a *semaphore*?

Many stalls…

```
//shared data
semaphore lock = 1;

do {
    wait(&lock);
    // critical section
    sign(&lock);
    // remainder section
} while (true);
```

Wait in line for open stall

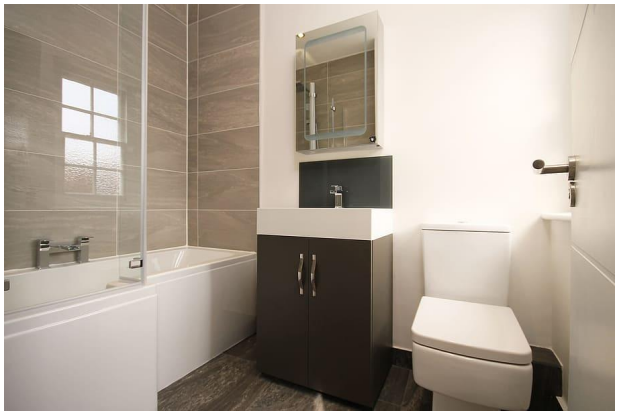
Unlock when leaving

Semaphores support ***more than one*** resource

SER 334
Monitors

Monitor expands the concept to cover a ***class as a whole***.

Only one process can execute within the class at one time



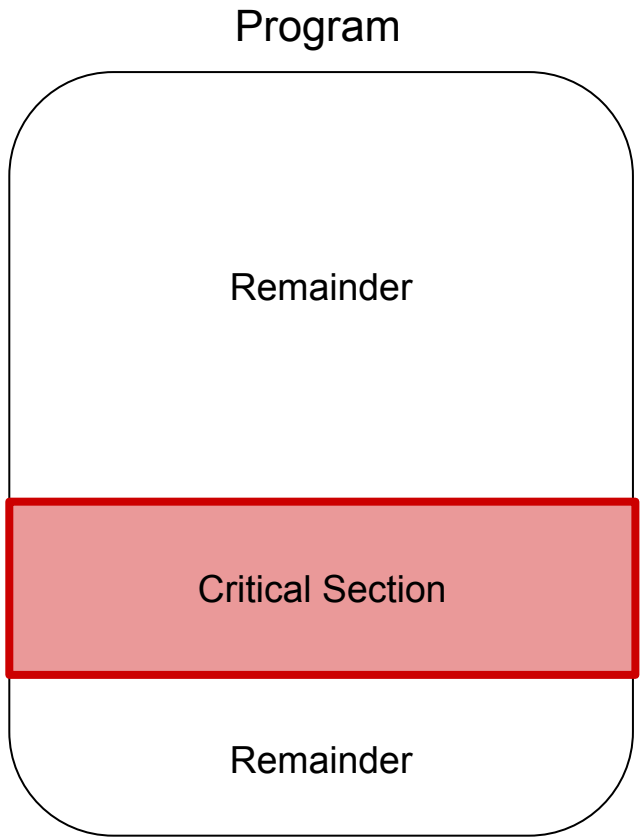
```
monitor class Account {
    int balance;

    Account(int opening) {
        balance = opening;
    }

    void deposit(int amount) {
        balance = balance + amount;
    }

    void withdraw(int amount) {
        balance = balance - amount;
    }
};
```

Deposit $100

Withdraw $50

SER 334
Samples

4. [Acuña] Explain how it would be possible to have a situation where programs are making progress but do not have bounded waiting time.



SER 334

Scratch Space

Upcoming Events

SI Sessions:

- Monday, February 5th at 7:00 pm MST

- ~~Sunday, February 11th at 7:00 pm MST~~ **Cancelled - Good luck on Exam 2!**
- Monday, February 12th at 7:00 pm MST

Review Sessions:

- Exam 2 Review: Thursday, February 8th 7:00 pm - 9:00 pm MST
- Exam 3 Review: TBD

Questions?

Survey:

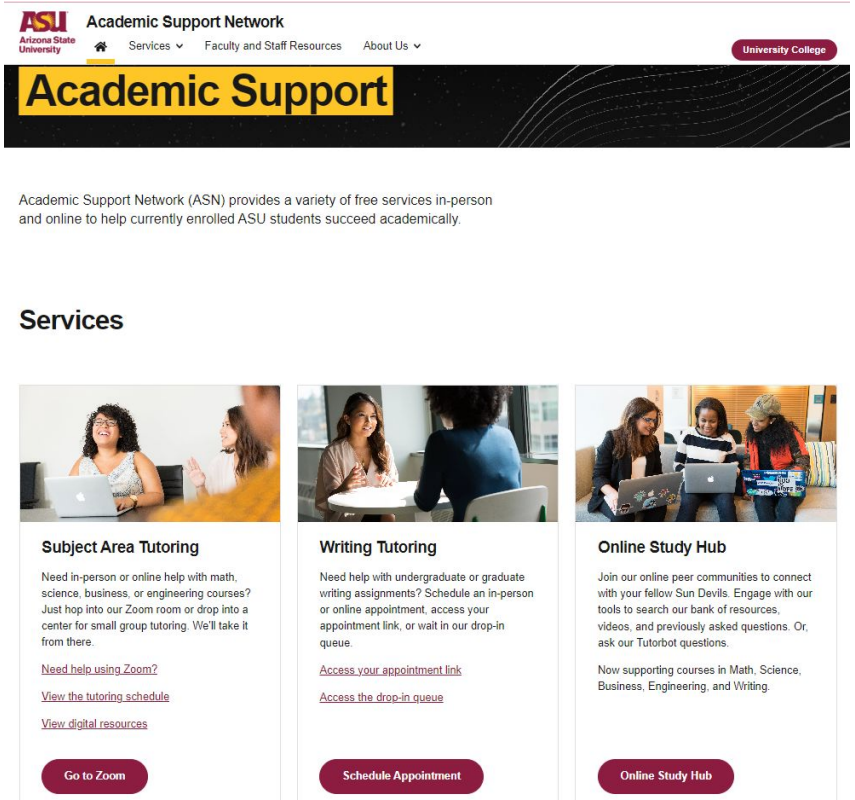
http://bit.ly/ASN2324

22

More Questions?

Check out our other resources!

tutoring.asu.edu

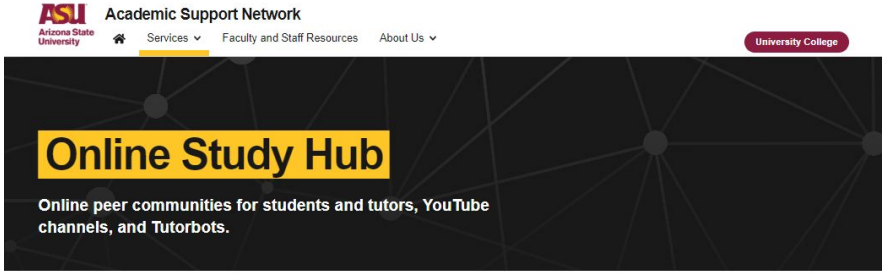


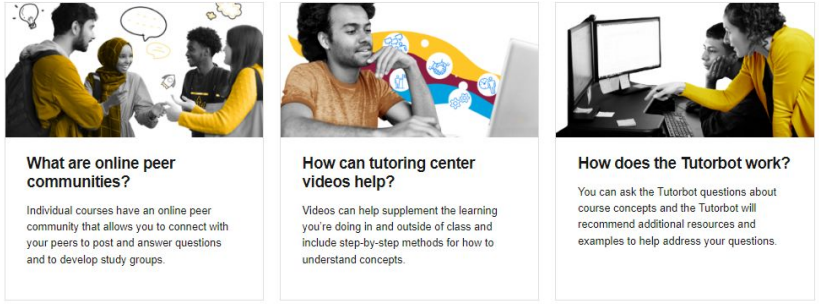
1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

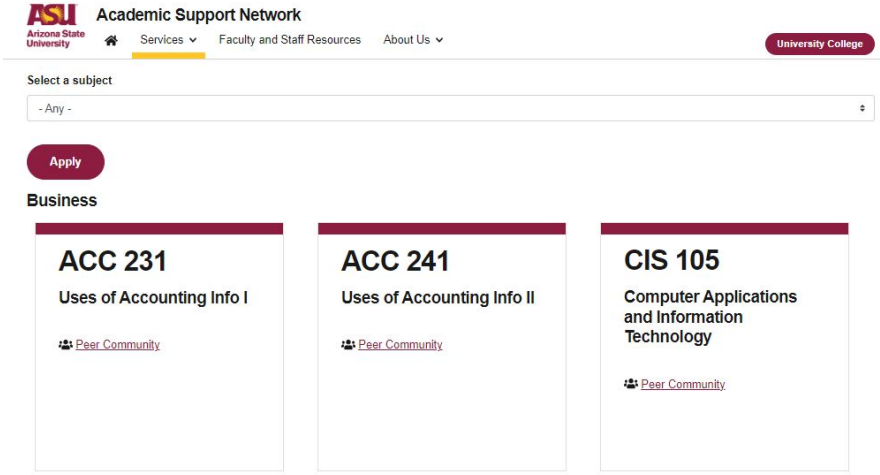
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub







Don't forget to check out the Online Study Hub for additional resources!

Additional Resources

- Course Repo
- Course Discord
- BMP File Format (Wiki)
- Linux Kernel API
- Bootlin - Linux Cross Referencer
- Dining Philosophers Interactive
- Producer/Consumer Visual