between you and your classroom neighbors, how many different languages do y'all speak?

clicker.cs.illinois.edu

Q1

~Code~
340

CS 340

Threading

Updates

1. MP5 Allocator Steps 1-6 due next tuesday

Hw 4-5

2. HW 5 out today

Agenda

1. Processes Review
2. Threading
3. Synchronization in programming

Review

- A process is ... running instance of a program
- A single core CPU can run ... 1 process at a time
  ~ a million inst. a second
- On a single core CPU, processes run ...
  Concurrently
- The OS ... schedules and loads processes in and out of the CPU

What prints?

main.c

```c
#include <stdio.h>
#include <unistd.h>

unsigned int calc(int n){
    unsigned int ret = 1;
    for(int i = n; i > 0; i--){
        ret = i * ret;
    }
    return ret;
}

int main(int argc, char *argv[]) {
    sleep(3); // Pauses for 3 seconds
    int intValue = argv[1][0] - '0';
    printf("%i", calc(intValue));
}
```

4 · 3 · 2 = 24

```
gcc main.c
./a.out 4
```

→ a.out

"4" "0"

argv[1]

clicker.cs.illinois.edu

Q2

~Code~
340

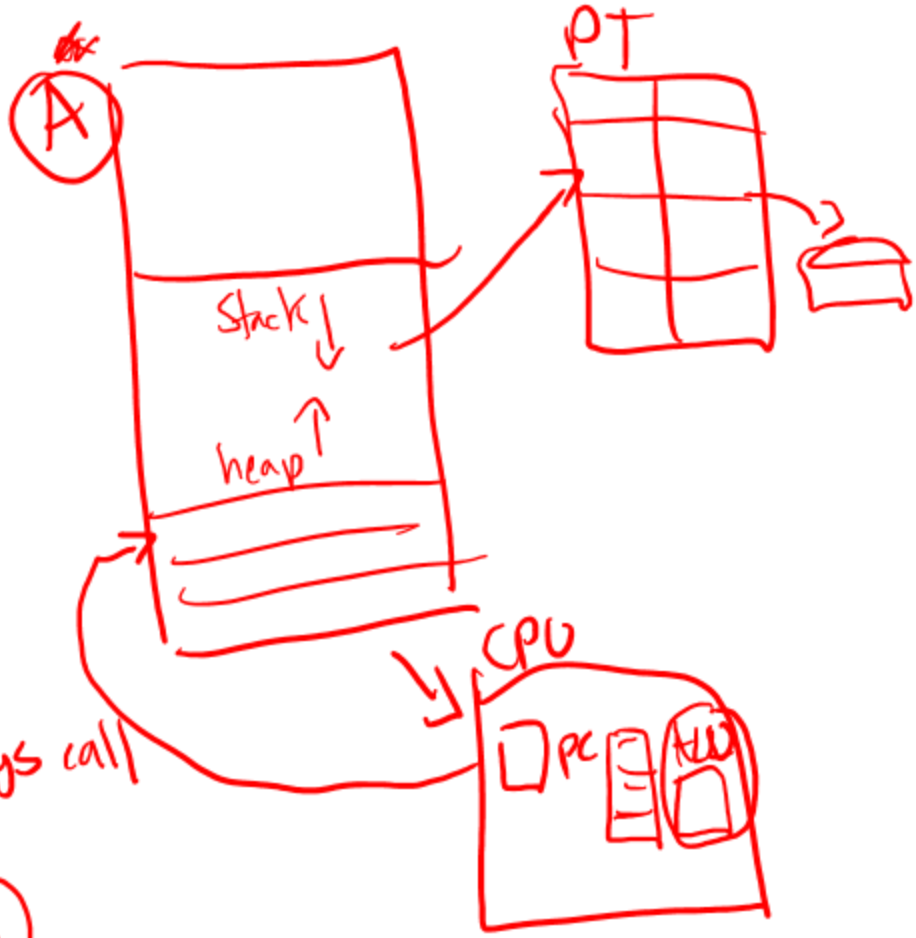
main.c

```c
#include <stdio.h>
#include <unistd.h>

unsigned int calc(int n){
    unsigned int ret = 1;
    for(int i = n; i > 0; i--){
        ret = i * ret;
    }
    return ret;
}

int main(int argc, char *argv[]) {
    sleep(3); // Pauses for 3 seconds
    int intValue = argv[1][0] - '0';
    printf("%i", calc(intValue));
}
```

```
gcc main.c
./a.out 4
```

machine instr.

start process A



main.c

```c
#include <stdio.h>
#include <unistd.h>

unsigned int calc(int n){
    unsigned int ret = 1;
    for(int i = n; i > 0; i--){
        ret = i * ret;
    }
    return ret;
}

int main(int argc, char *argv[]) {
    sleep(3); // Pauses for 3 seconds
    int intValue = argv[1][0] - '0';
    printf("%i", calc(intValue));
}
```

gcc main.c

Ter 1

./a.out 3

Ter 2

./a.out 4

Process Vs. Threads

Process

- own VM
- own PC
- own reg values

Thread

within a process ← 1 thread or more

Thread
- Own PC
- Own reg values

Shared VM

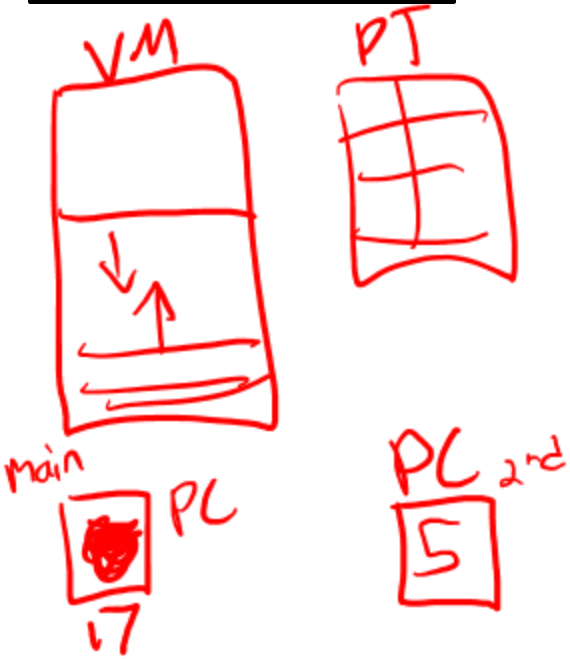
- UI responsiveness
- handle many user requests
- Shared computation

Thread Example

main.c

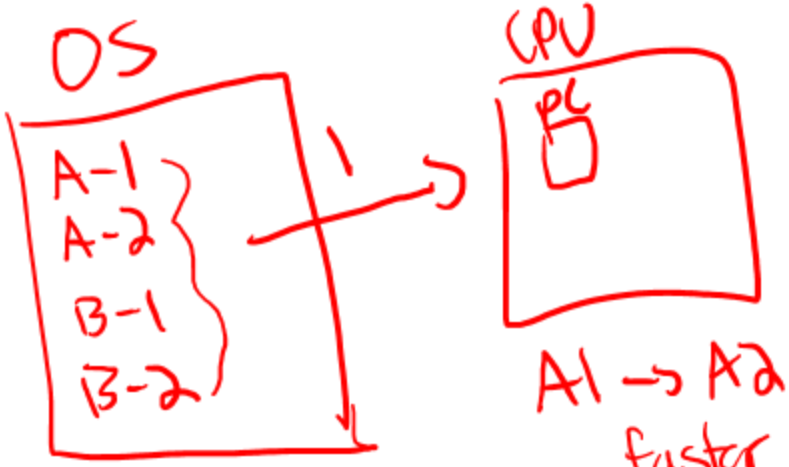
```c
#include <stdio.h>
#include <pthread.h>

void* thread_function(void* arg) {
    printf("Hello from the new thread!\n");
    return NULL;
}

int main() {
    pthread_t thread;

    // Create a new thread
    if (pthread_create(&thread, NULL, thread_function, NULL) != 0) {
        perror("Failed to create thread");
        return 1;
    }

    printf("Hello from the main thread!\n");

    // Wait for the new thread to finish
    pthread_join(thread, NULL);

    printf("Thread has finished.\n");
}
```

```
gcc main.c
./a.out
```



Scheduling

- Works on a thread level



A1 -> A2
faster

A1 -> B2

B2 -> A2 slow

Threads
VM
own stack frame

processes

A ./a.out
B ./a.out

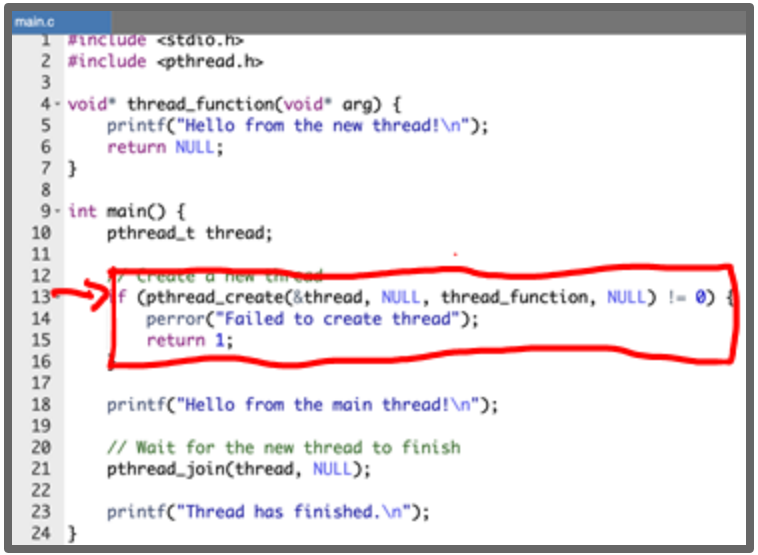
```c
#include <stdio.h>
#include <pthread.h>

void* thread_function(void* arg) {
    printf("Hello from the new thread!\n");
    return NULL;
}

int main() {
    pthread_t thread;

    // Create a new thread
    if (pthread_create(&thread, NULL, thread_function, NULL) != 0) {
        perror("Failed to create thread");
        return 1;
    }

    printf("Hello from the main thread!\n");

    // Wait for the new thread to finish
    pthread_join(thread, NULL);

    printf("Thread has finished.\n");
}
```

What are the max number of threads that could be running concurrently if I run this program from 5 terminals at the same time?

clicker.cs.illinois.edu

Q3

~Code~
340



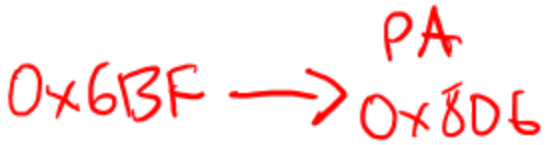
10 Threads

How many different page tables would the computer need to store to run this program from 5 terminals concurrently?

clicker.cs.illinois.edu

Q4

~Code~
340



main.c

```c
#include <stdio.h>
#include <pthread.h>

void* thread_function(void* arg) {
    printf("Hello from the new thread!\n");
    return NULL;
}

int main() {
    pthread_t thread;

    // Create a new thread
    if (pthread_create(&thread, NULL, thread_function, NULL) != 0) {
        perror("Failed to create thread");
        return 1;
    }

    printf("Hello from the main thread!\n");

    // Wait for the new thread to finish
    pthread_join(thread, NULL);

    printf("Thread has finished.\n");
}
```

With a 1 core CPU how many processes could I run in parallel?

clicker.cs.illinois.edu

Q5

~Code~
340

main.c

```c
#include <stdio.h>
#include <pthread.h>

void* thread_function(void* arg) {
    printf("Hello from the new thread!\n");
    return NULL;
}

int main() {
    pthread_t thread;

    // Create a new thread
    if (pthread_create(&thread, NULL, thread_function, NULL) != 0) {
        perror("Failed to create thread");
        return 1;
    }

    printf("Hello from the main thread!\n");

    // Wait for the new thread to finish
    pthread_join(thread, NULL);

    printf("Thread has finished.\n");
}
```

With a 1 core CPU how many threads could I run in parallel?

clicker.cs.illinois.edu

Q6

~Code~
340

main.c

```c
#include <stdio.h>
#include <pthread.h>

void* thread_function(void* arg) {
    printf("Hello from the new thread!\n");
    return NULL;
}

int main() {
    pthread_t thread;

    // Create a new thread
    if (pthread_create(&thread, NULL, thread_function, NULL) != 0) {
        perror("Failed to create thread");
        return 1;
    }

    printf("Hello from the main thread!\n");

    // Wait for the new thread to finish
    pthread_join(thread, NULL);

    printf("Thread has finished.\n");
}
```

Threading Example

main.c

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

void *compute(void* arg) {
    int* ptr = (int*)arg;
    printf("my value is %i\n", *ptr);
    return NULL;
}

int main() {
    pthread_t threads[2];
    int args[2] = {0, 1};

    for (int i = 0; i < 2; i++) {
        if (pthread_create(&threads[i], NULL, compute, &(args[i])) != 0) {
            perror("Failed to create thread");
            return 1;
        }
    }

    //wait for both threads to finish
    for (int i = 0; i < 2; i++) {
        pthread_join(threads[i], NULL);
    }
}
```

here

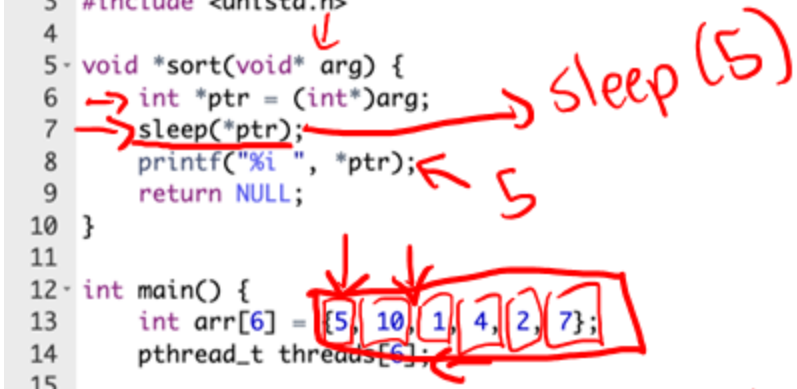
0 1

3 threads

What will this code print?

main.c

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

void *sort(void* arg) {
    int *ptr = (int*)arg;
    sleep(*ptr);
    printf("%i ", *ptr);
    return NULL;
}

int main() {
    int arr[6] = {5, 10, 1, 4, 2, 7};
    pthread_t threads[6];

    for (int i = 0; i < 6; i++) {
        if (pthread_create(&threads[i], NULL, sort, &(arr[i])) != 0) {
            perror("Failed to create thread");
            return 1;
        }
    }

    //wait for both threads to finish
    for (int i = 0; i < 6; i++) {
        pthread_join(threads[i], NULL);
    }
}
```



clicker.cs.illinois.edu

Q7

~Code~
340

main.c

```c
#include <pthread.h>
#include <unistd.h>

int arr[2] = {0, 0};
int counter = 0;

void *compute(void* arg) {
    for(int i = 0; i < 2; i++){
        if(arr[i] == 0){
            printf("changing index %i\n", i);
            arr[i] = counter;
            counter++;
        }
    }
    return NULL;
}

int main() {
    pthread_t threads[2];

    for (int i = 0; i < 2; i++) {
        if (pthread_create(&threads[i], NULL, compute, NULL) != 0) {
            perror("Failed to create thread");
            return 1;
        }
    }

    //wait for both threads to finish
    for (int i = 0; i < 2; i++) {
        pthread_join(threads[i], NULL);
    }

    for (int i = 0; i < 2; i++) {
        printf("%i ", arr[i]);
    }
}
```

A

B

race condition

clicker.cs.illinois.edu

Q8

~Code~
340

Could this code print "3 2"

3 2 ?

yes

Synchronization

Idea - take turns accessing critical sections (share resource)

Mutex - lock - A gets and releases

Mutex

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;  // static init

int arr[2] = {0, 0};
int counter = 1;

void *compute(void* arg) {
    pthread_mutex_lock(&m);
    for(int i = 0; i < 2; i++){
        if(arr[i] == 0){
            printf("changing index %i\n", i);
            arr[i] = counter;
            counter++;
        }
    }
    pthread_mutex_unlock(&m);

    return NULL;
}

int main() {
    pthread_t threads[2];

    for (int i = 0; i < 2; i++) {
        if (pthread_create(&threads[i], NULL, compute, NULL) != 0) {
            perror("Failed to create thread");
            return 1;
        }
    }
    //wait for both threads to finish
    for (int i = 0; i < 2; i++) {
        pthread_join(threads[i], NULL);
    }

    for (int i = 0; i < 2; i++) {
        printf("%i ", arr[i]);
    }
}
```

get the lock function

unlock

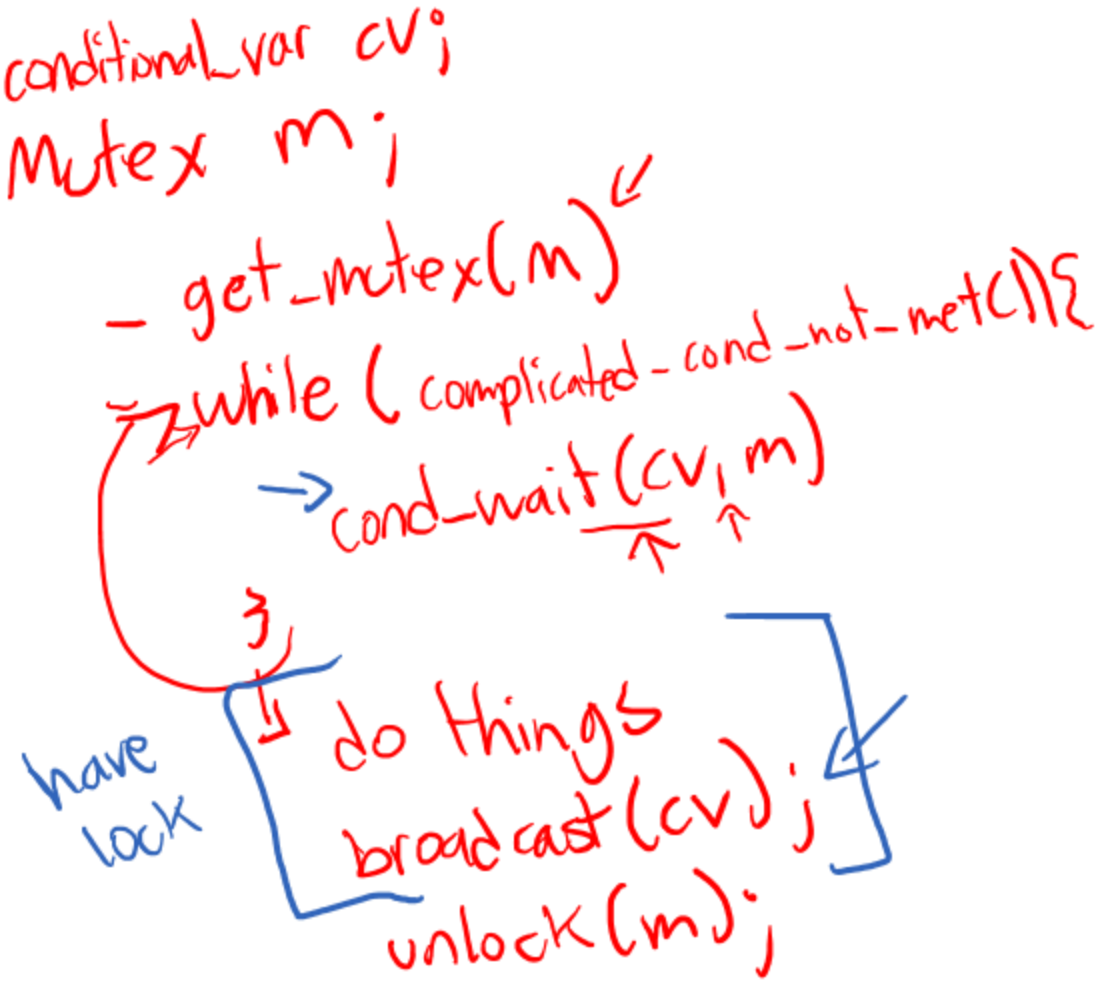
sharing resources

indiv. resources

Synchronization

Idea - take turns

condition variable



Conditional Variable

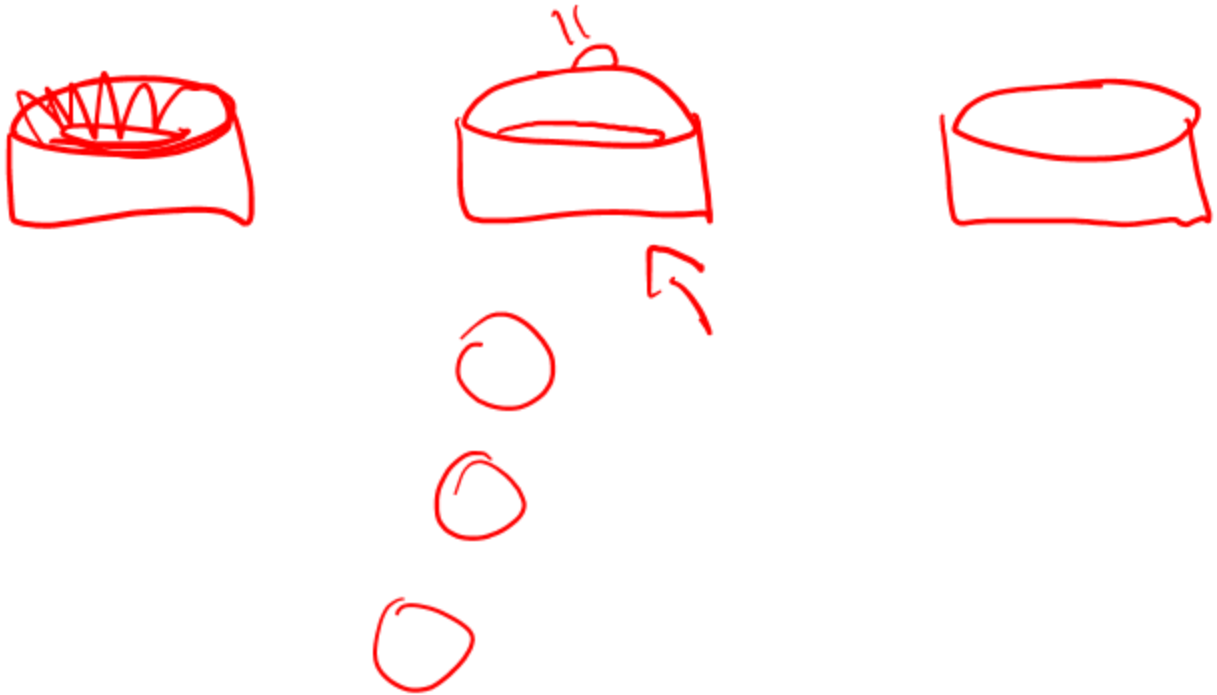
```c
pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t  cv = PTHREAD_COND_INITIALIZER;

int ready = 0;
int data  = 0;

void* worker(void* arg) {
    pthread_mutex_lock(&m);
    while (!ready) {
        pthread_cond_wait(&cv, &m);
    }
    int value = data;
    pthread_mutex_unlock(&m);

    printf("worker got data = %d\n", value);
    return NULL;
}

int main(void) {
    pthread_t t;
    pthread_create(&t, NULL, worker, NULL);

    // doing work to produce the data
    sleep(1);

    pthread_mutex_lock(&m);
    data = 42;
    ready = 1;
    pthread_cond_signal(&cv);
    pthread_mutex_unlock(&m);

    pthread_join(t, NULL);
    return 0;
}
```

give up m wait

unlock

unlock m

Buffet Example

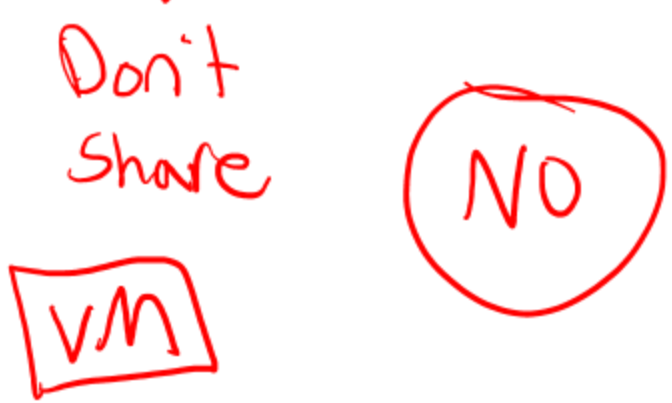


Do I potentially need a mutex for handling multiple processes running on my computer? Why or why not?

clicker.cs.illinois.edu

Q9

~Code~
340



Deadlock

Alice – gets A → hold on B

Bob – gets B → Hold on A

Deadlock - Dining Philosophers

What does deadlock look like in code?

infinite loop

clicker.cs.illinois.edu

Q10

~Code~
340