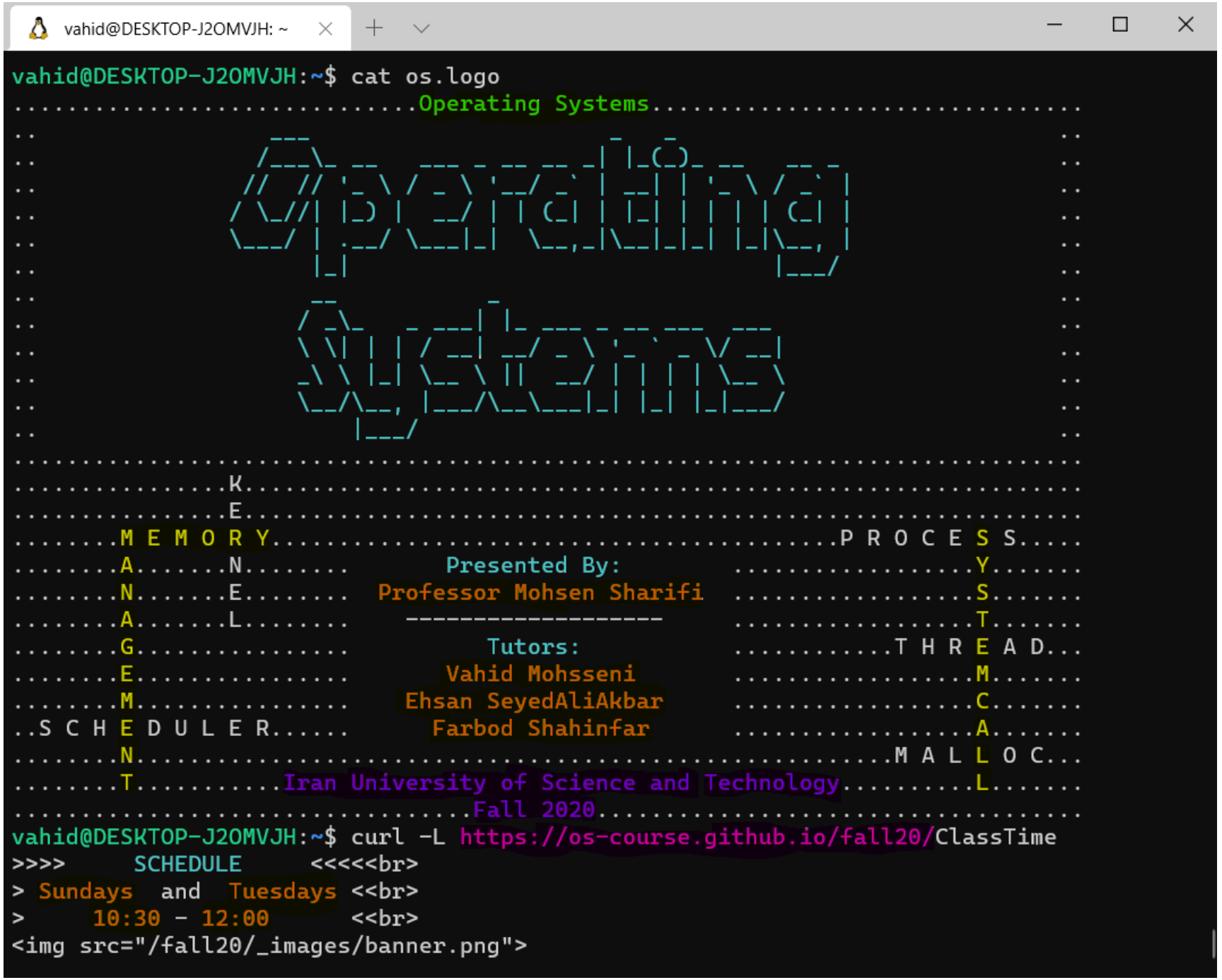
Operating Systems

Introduction to *pthread* and *semaphore* library

Fall 2020


vahid@DESKTOP-J2OMVJH: ~
vahid@DESKTOP-J2OMVJH:~$ cat os.logo
Operating Systems
KERNEL
MEMORY MANAGEMENT
SCHEDULER
PROCESSYS
THREAD
MALLOC
Presented By:
Professor Mohsen Sharifi
Tutors:
Vahid Mohsseni
Ehsan SeyedAliAkbar
Farbod Shahinfar
Iran University of Science and Technology
Fall 2020
vahid@DESKTOP-J2OMVJH:~$ curl -L https://os-course.github.io/fall20/ClassTime
>>>> SCHEDULE <<<<
> Sundays and Tuesdays <
> 10:30 - 12:00 <
<img src="/fall20/_images/banner.png">

Quiz 3

- Quiz is open during 10:35 until 10:57
- Duration: 20 minutes

Agenda

- Working with Threads
- Mutual Exclusion
- Conditional Variables
- Semaphores

Working with Threads

Overview of Section

- Threading
  - pthread_thread_t
  - pthread_create()
  - pthread_join()
  - pthread_exit()

Creating New Threads

```
#include <pthread.h>
#define count_threads 10
typedef struct { int wid; } t_arg;

void *worker_func(void *_arg) { // implementation of the thread … }

int main(int argc, char *argv[]) {
    // defining some variables
    pthread_t threads[count_threads];
    t_arg args[count_threads];
    for (i = 0; i < count_threads; i++) {
        args[i].wid = i;
        pthread_create(&threads[i], NULL,
                       worker_func, (void *)&args[i]);
    }
    for (i = 0; i < count_threads; i++)
        pthread_join(threads[i], NULL);
    return 0;
}
```

Critical Region

```
#include <pthread.h>
#define count_threads 10
typedef struct { int wid; } t_arg;

int tail = 0;
int arr[count_threads];

void *worker_func(void *_arg) {
    t_arg arg = (t_arg *)_arg;
    arr[tail] = arg->wid;
    printf("wid: %d\n", arg->wid);
    tail++;
    pthread_exit(NULL);
}
```

Critical Region

```
#include <pthread.h>
#define count_threads 10
typedef struct { int wid; } t_arg;

int tail = 0;
int arr[count_threads];

void *worker_func(void *_arg) {
    t_arg arg = (t_arg *)_arg;
    arr[tail] = arg->wid;
    printf("wid: %d\n", arg->wid);
    tail++;
    pthread_exit(NULL);
}
```

Shared Resources

Critical Region

Output of Our Example

0: wid: 6	0: wid: 5	0: wid: 7	0: wid: 6
1: wid: 8	1: wid: 0	1: wid: 8	1: wid: 7
2: wid: 9	2: wid: 0	2: wid: 9	2: wid: 8
3: wid: 5	3: wid: 0	3: wid: 0	3: wid: 9
4: wid: 4	4: wid: 0	4: wid: 0	4: wid: 5
5: wid: 0	5: wid: 0	5: wid: 0	5: wid: 4
6: wid: 2	6: wid: 9	6: wid: 0	6: wid: 3
7: wid: 0	7: wid: 6	7: wid: 0	7: wid: 2
8: wid: 0	8: wid: 0	8: wid: 0	8: wid: 1
9: wid: 1	9: wid: 0	9: wid: 0	9: wid: 0

Result of running previous program four times

Mutual Exclusion

Overview of Section

- Lock (Mutex)
  - pthread_mutex_t
  - pthread_mutex_init
  - pthread_mutex_lock
  - pthread_mutex_unlock
  - pthread_mutex_destroy

Creating New Threads

```
#include <pthread.h>
#define count_threads 10
typdef struct { int wid; } t_arg;

pthread_mutex_t lock
void *worker_func(void *_arg) { // implementation of the thread … }

int main(int argc, char *argv[]) {
    // defining some variables
    pthread_t threads[count_threads];
    t_arg args[count_thread];
    pthread_mutex_init(&lock, NULL);
    // create threads like before
    // and wait until they are finished
    pthread_mutex_destroy(&lock);
    return 0;
}
```

Critical Region

```
#include <pthread.h>
#define count_threads 10
typdef struct { int wid; } t_arg;

pthread_mutex_t lock
int tail = 0;
int arr[count_threads];

void *worker_func(void *_arg) {
    t_arg arg = (t_arg *)_arg;
    pthread_mutex_lock(&lock)
    arr[tail] = arg->wid;
    printf("wid: %d\n", arg->wid);
    tail++;
    pthread_mutex_unlock(&lock)
    pthread_exit(NULL);
}
```

Output of Our Example

0: wid: 0	0: wid: 0	0: wid: 0	0: wid: 0
1: wid: 1	1: wid: 1	1: wid: 1	1: wid: 9
2: wid: 2	2: wid: 2	2: wid: 2	2: wid: 1
3: wid: 3	3: wid: 3	3: wid: 3	3: wid: 2
4: wid: 4	4: wid: 4	4: wid: 4	4: wid: 3
5: wid: 5	5: wid: 5	5: wid: 5	5: wid: 4
6: wid: 6	6: wid: 6	6: wid: 6	6: wid: 5
7: wid: 8	7: wid: 8	7: wid: 7	7: wid: 6
8: wid: 9	8: wid: 7	8: wid: 8	8: wid: 7
9: wid: 7	9: wid: 9	9: wid: 9	9: wid: 8

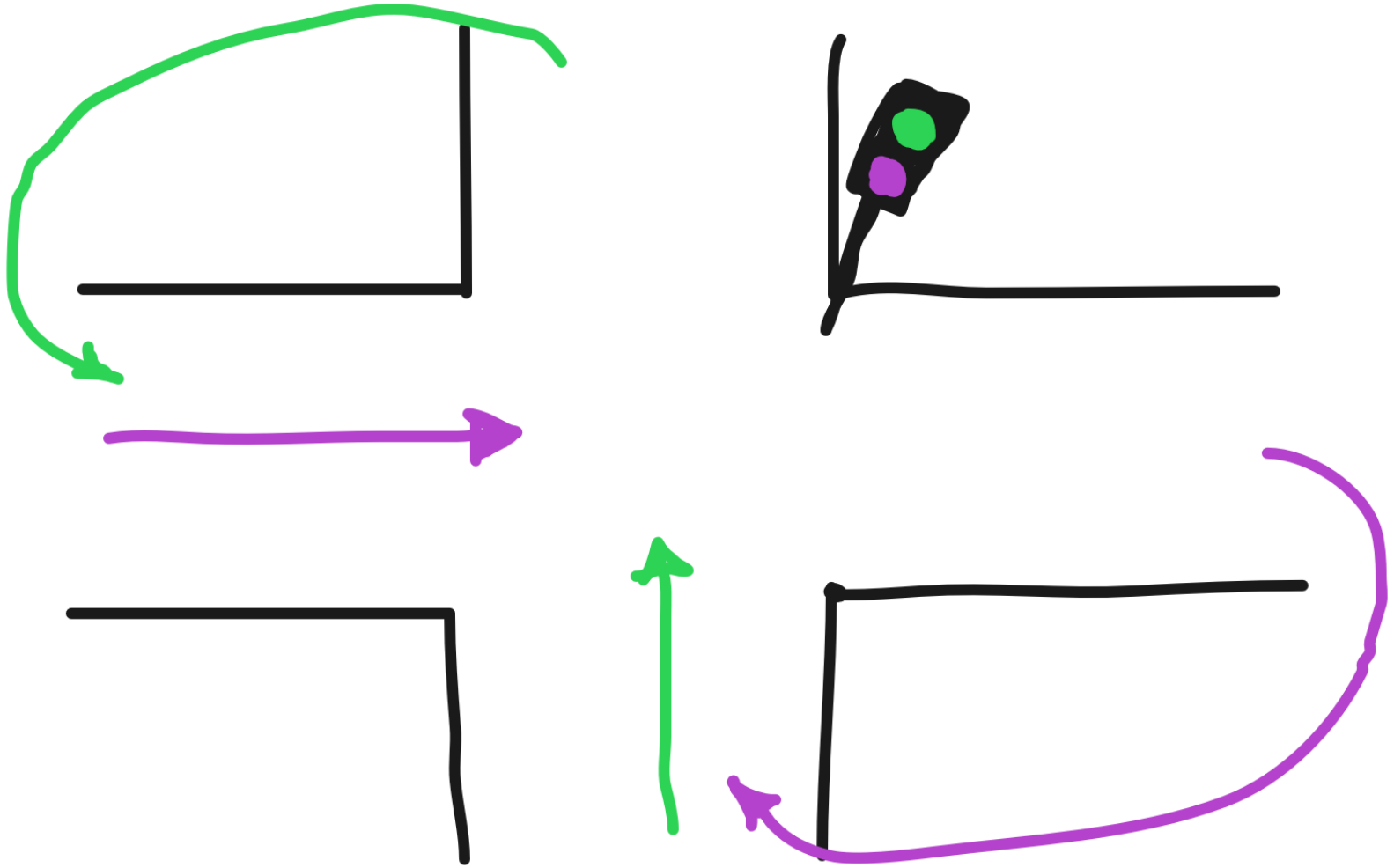
Result of running previous program four times

Conditional Variables

Overview of Section

- Lock (Conditional Variable)
  - pthread_cond_t
  - pthread_cond_init
  - pthread_cond_wait
  - pthread_cond_signal

Intersection Example



Intersection Example

- This section has been explained while sharing desktop
- Consult git repository for the codes

Semaphore

Overview of Section

- Lock (Semaphore)
  - sem_t
  - sem_init
  - sem_wait
  - sem_post
  - sem_destroy

Questions?

?