Exercise 00 ~(20min)##### *Really good read about threads.*

Lets Talk Threads

- Why use ***threads***? ~ (5min)
- What are threads? ~ (5min)
- What are the differences between processes and threads ~ 5(min)

Ex 01 ~(20min)

Description:

Create a thread, see what it does, how it works.

Goal to achieve:

Create a new thread that outputs this message!

```
$ ./ex01.out
Hi From thread. You can call me philosopher 0
```

Allowed functions:

`printf pthread_create pthread_join`

---

what is pthread_t ~(5min)

1. What is the prototype?

pthread_create ~ (5min)

1. What is the prototype?
2. What is `void \*(\*start_routine)(void \*)`
3. How would you pass data to the start_routine function?
4. What is the attr argument?

pthread_join ~ (5min)

1. What is the prototype?
2. What is void **value_ptr used for?

Ex02 ~(5min - 10min)

Goal:

Create 20 threads that will print the following

```
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
Hi From thread. You can call me philosopher 0
```

Ex03 ~35min

Race conditions

Whatch a video about data races

1. What are race conditions?
2. What is a critical section?

How to spot race conditions?

Do you see a data race in this code?

```
// example code
#include <stdio.h>
#include <pthread.h>
```

```
void    *rutine(void *ptr)
{
    while (*(int *)ptr < 1000)
    {
        *(int *)ptr += 1;
    }
    printf("Done\n");
    return (NULL);
}

int    main()
{
    pthread_t   thread;
    int         index;

    index = 0;
    pthread_create(&thread, NULL, rutine, &index);
    while (index < 10000)
    {
        index++;
    }
    pthread_join(thread, NULL);
    return (0);
}
```

What are mutexes?

What is the pthread_mutex_t data type?

pthread_mutex_init ~ 5(min)

What does this function do?

pthread_mutex_destroy ~ 5(min)

```
What does this function do?
Do you have to free the mutex?
```

pthread_mutex_lock ~ 5 (min)

```
What does this function do?
Can you lock a mutex that is not inited?
```

pthread_mutex_unlock ~ 25 (min)

```
What does this function do?
What happens when u unlock a mutex 2times?
```

Goal:
Make a program that inits, locks, unlock and destroys a mutex!

Goal:
Prevent the data race in the example code.

ex04

Deadlocks

What is a deadlock?
When does a deadlock occur?

Goal:

Goal:
Produce a program that has a deadlock.

Break

ex05

creat 20 threads that will print the following
the order does not matter

```
Hi From thread. You can call me philosopher 1
Hi From thread. You can call me philosopher 2
Hi From thread. You can call me philosopher 3
Hi From thread. You can call me philosopher 4
Hi From thread. You can call me philosopher 5
Hi From thread. You can call me philosopher 6
Hi From thread. You can call me philosopher 7
Hi From thread. You can call me philosopher 8
Hi From thread. You can call me philosopher 9
Hi From thread. You can call me philosopher 10
Hi From thread. You can call me philosopher 11
Hi From thread. You can call me philosopher 12
Hi From thread. You can call me philosopher 13
Hi From thread. You can call me philosopher 14
Hi From thread. You can call me philosopher 15
Hi From thread. You can call me philosopher 16
Hi From thread. You can call me philosopher 17
```

```
Hi From thread. You can call me philosopher 18
Hi From thread. You can call me philosopher 19
Hi From thread. You can call me philosopher 20
Hi From thread. You can call me philosopher 21
```

Bonus

Make a program that will use 3 created threads to add up an int to 42. Threads can increment the int every .5sec
once the value is 42 the program has to print "Got it!\n" and exit.

catch:
The threads do not know when the value is 42

What is a monitoring thread?

A monitoring thread is a concept used in the philosopher's project. a monitoring thread will keep track of the int variable. If the value is 42 let the threads know to finish and exit.