# 2INC0 Operating Systems
## practical

Assignment 2

# schedule practical

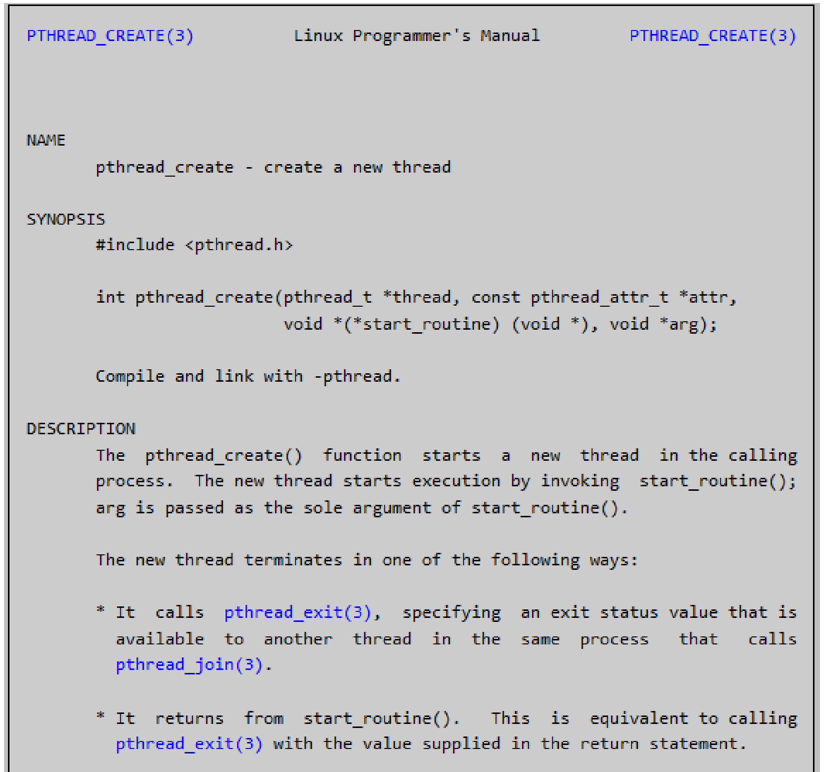
| week | Wed afternoon (7+8) | Fri morning (3+4) |
|---|---|---|
| 1 | Linux, C helloworld | C pointers |
| 2 | explain assignment 1 | C structs + memory alloc |
| 3 | C arrays + strings | |
| 4 | explain assignment 2 | C bit operations |
| 5 | | |
| 6 | explain assignment 3 | |
| 7 | | |

# threaded application

https://www.youtube.com/watch?v=LejoPGtliTs

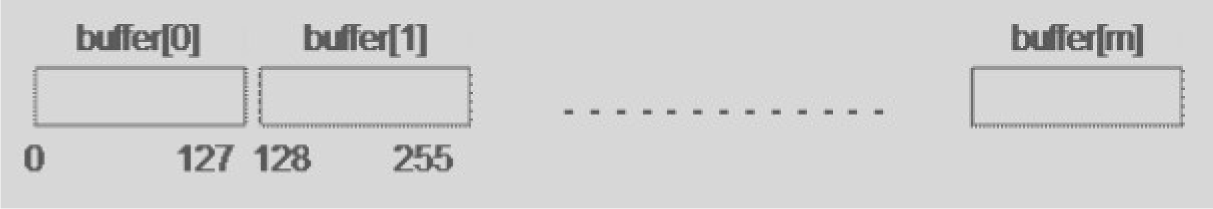
# threaded_basics.c

- source code for:
  - creating a new thread
  - passing parameters to a thread (and back)
  - synchronize threads
  - bit operations on integers
- our advice:
  - study the syntax
  - study the library calls



```
PTHREAD_CREATE(3)          Linux Programmer's Manual          PTHREAD_CREATE(3)



NAME
       pthread_create - create a new thread

SYNOPSIS
       #include <pthread.h>

       int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                          void *(*start_routine) (void *), void *arg);

       Compile and link with -pthread.

DESCRIPTION
       The  pthread_create()  function  starts  a  new  thread  in the calling
       process.  The new thread starts execution by invoking  start_routine();
       arg is passed as the sole argument of start_routine().

       The new thread terminates in one of the following ways:

       * It  calls  pthread_exit(3),  specifying  an exit status value that is
         available  to  another  thread  in  the  same  process   that   calls
         pthread_join(3).

       * It  returns  from  start_routine().   This  is  equivalent to calling
         pthread_exit(3) with the value supplied in the return statement.
```

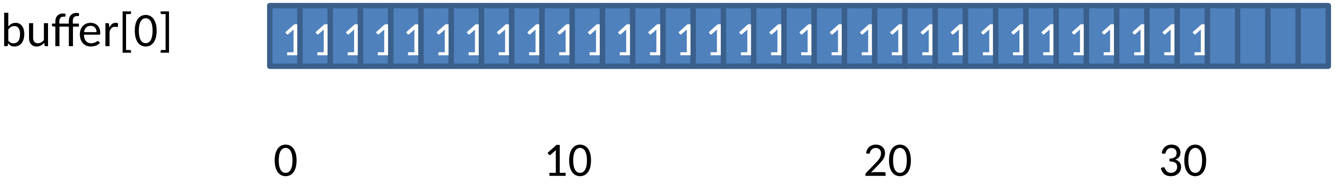
# Central flipping structure:   uint128_t  buffer[N]



## Small example

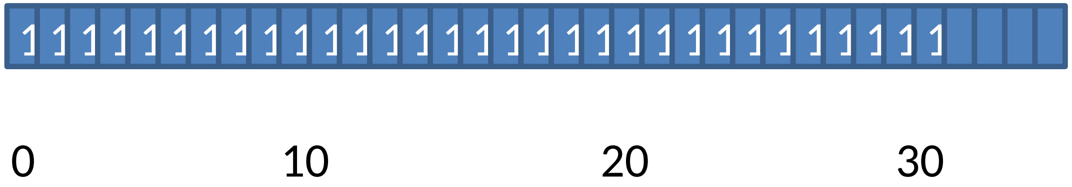
#define  NROF_PIECES  30

#define  NROF_THREADS  2



# All pieces are black

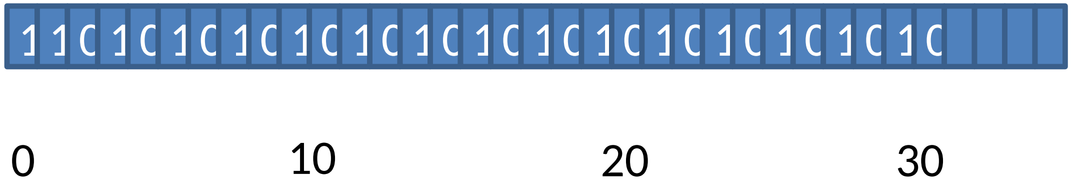
buffer[0]



# Thread (parameter = 2) flips all multiples of 2

```
#define  NROF_PIECES  30
```

buffer[0]



## Thread_1 (parameter = 2) flips all multiples of 2

buffer[0]

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | 1 | C | O | 1 | | | | |

0 10 20 30

## Thread_2 (parameter = 3) flips all multiples of 3

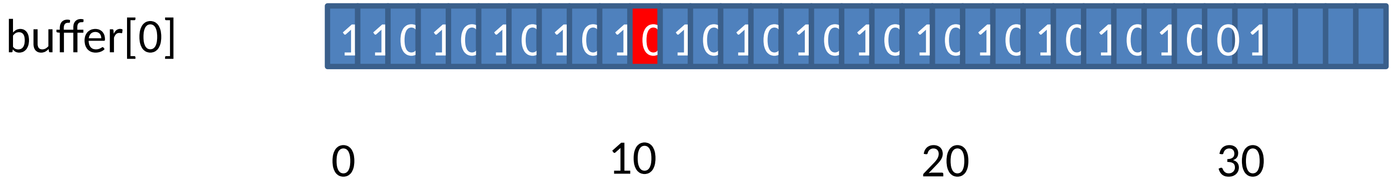
#define NROF_THREADS 2

buffer[0]

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | C | C | C | 1 | 1 | 1 | C | C | C | 1 | 1 | 1 | C | C | C | 1 | 1 | 1 | C | C | C | 1 | 1 | 1 | C | C | C | 1 | 1 | | | | |

0 10 20 30

# Thread (parameter = 2) flips all multiples of 2


buffer[0]
1 1 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C O 1
0
10
20
30

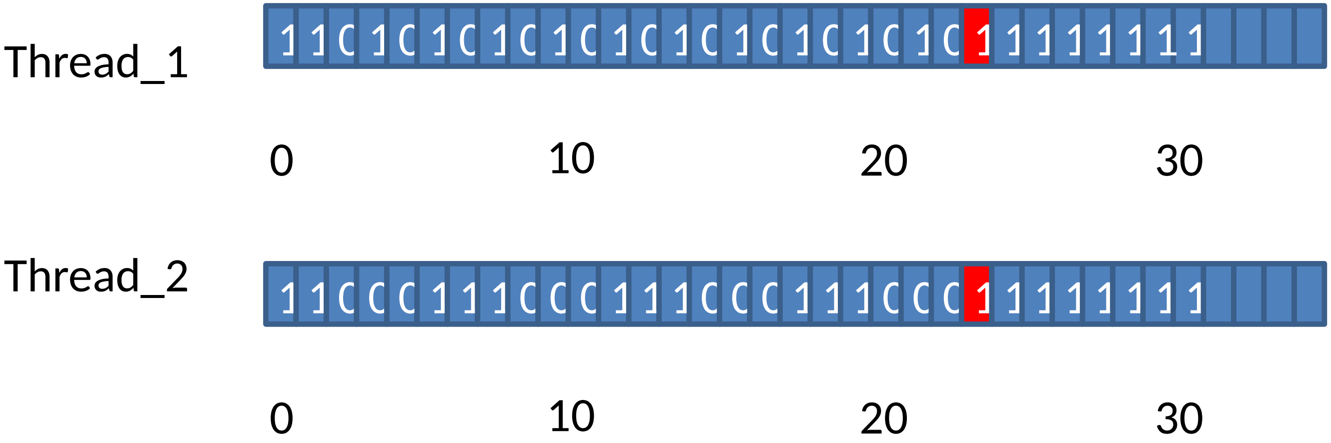
# How does flipping work?

```
if  (BIT_IS_SET( buffer[0], 10))
{
   BIT_CLEAR( buffer[0], 10);
}
else
{
   BIT_SET( buffer[0], 10);
}
```

## Both threads are executed in parellel:

## Thread_1 (parameter = 2) flips all multiples of 2

## Thread_2 (parameter = 3) flips all multiples of 3



Thread_1

1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1

Thread_2

1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1

**Both threads wants to flip number 24 at the same time. What happens?**

**Solution:** use mutex for the critical section

# threaded_basics.c

- source code for:
  - creating a new thread
  - passing parameters to a thread (and back)
  - synchronize threads
  - bit operations on integers