Mutexes & Shared Memory

COMP 310/ECSE 427

Instructor
Rola Harmouche

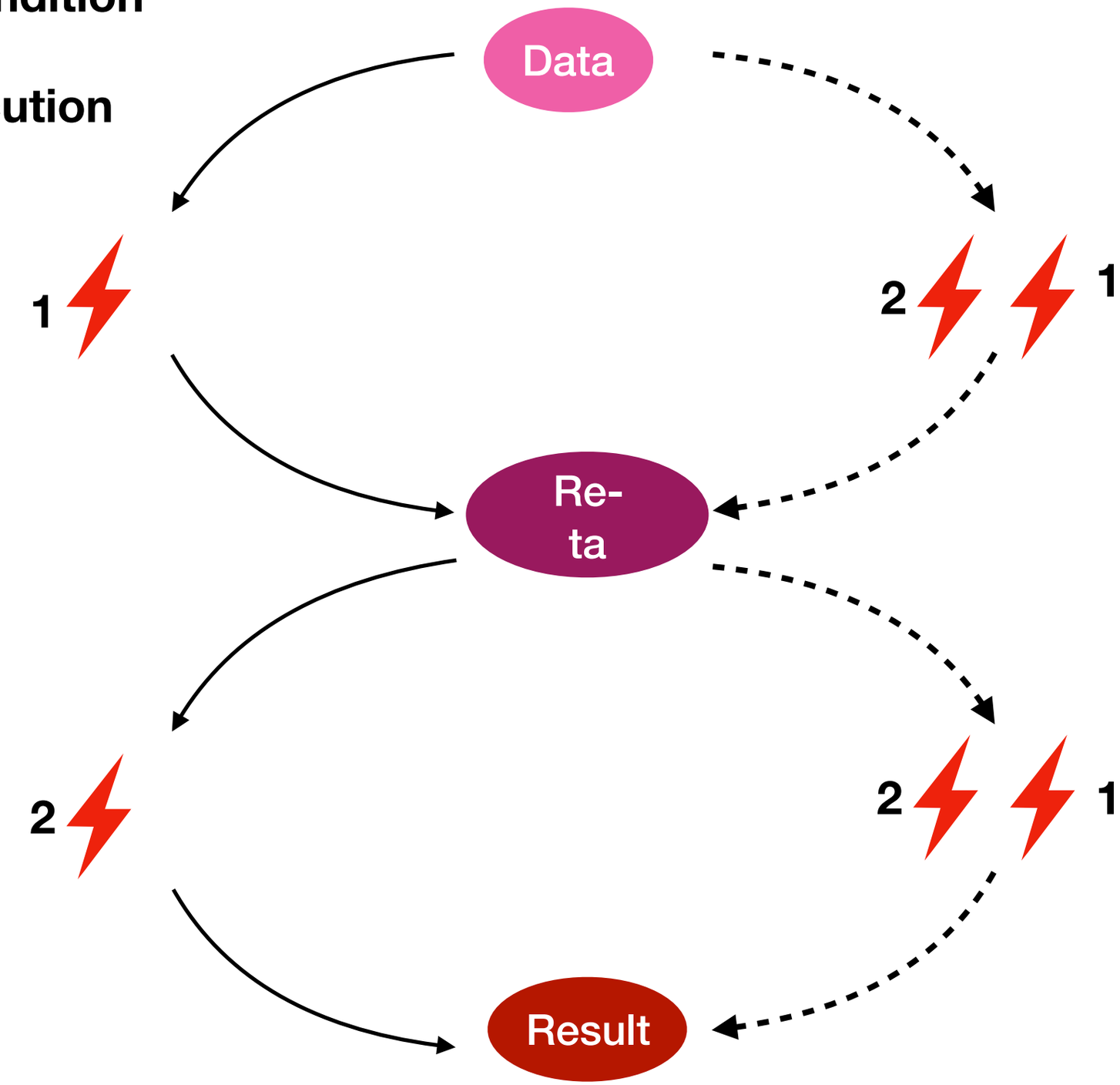
Presented by:-
Aakash Nandi

Problems with threads

Prob 1: Race Condition

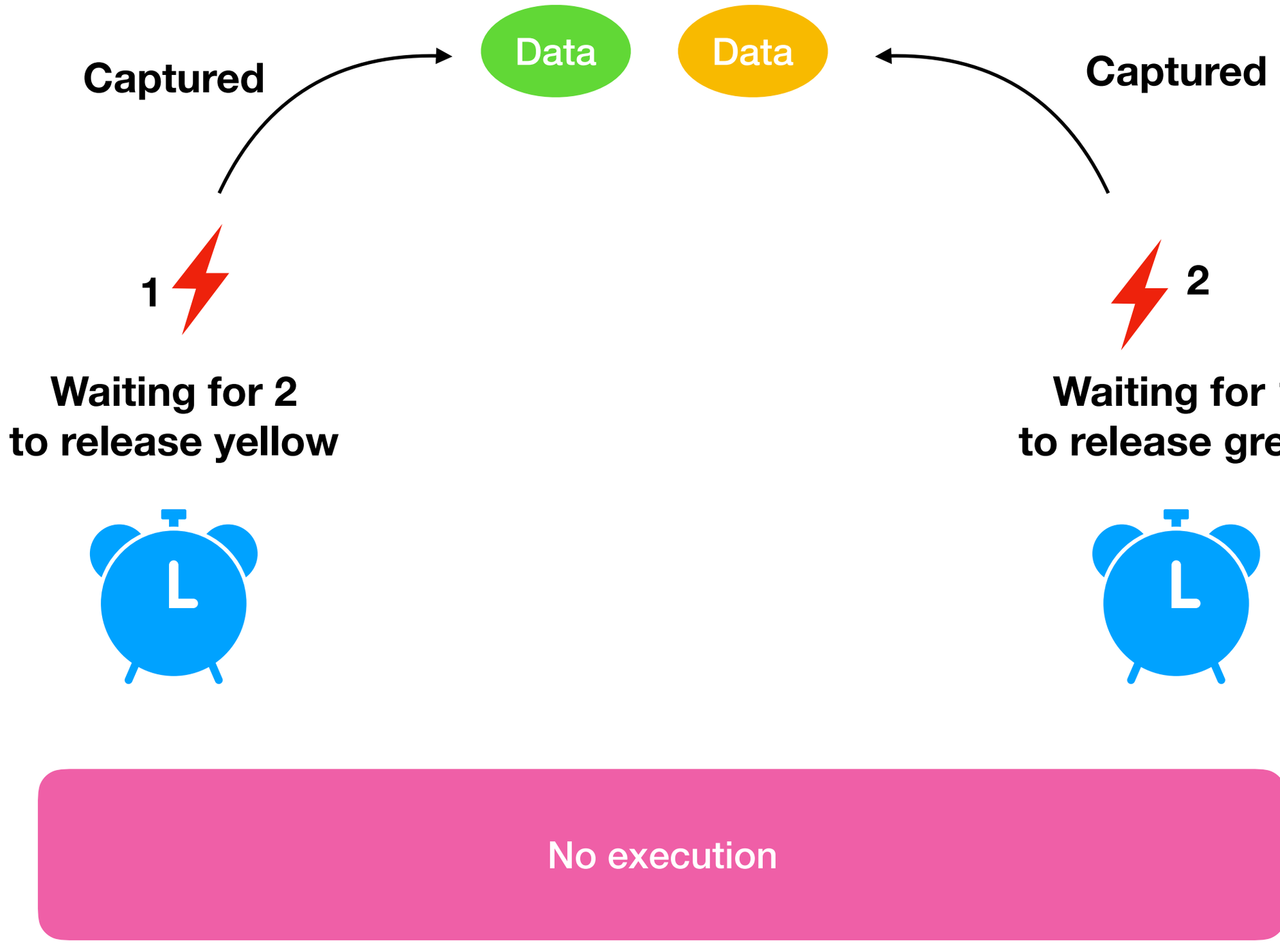
Correct Execution

Wrong Partial Exec



Problems with threads

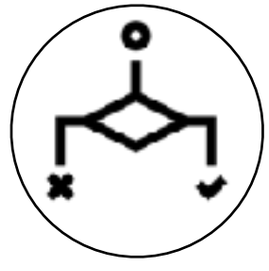
Prob 1: Deadlock



Mutex (Mutual Exclusion)

Can be shared between threads or processes

Mutex



Condition Variable

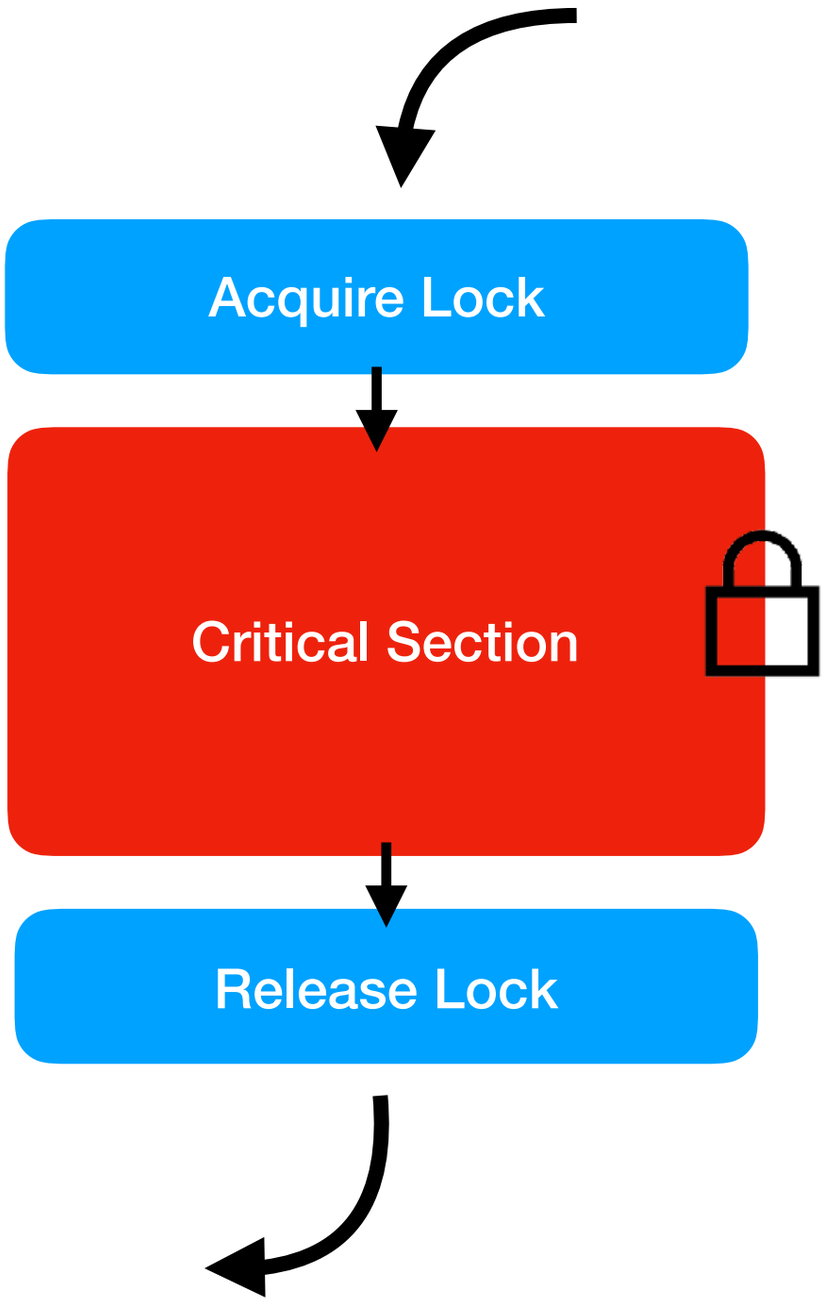
Wait

BroadCast

Signal

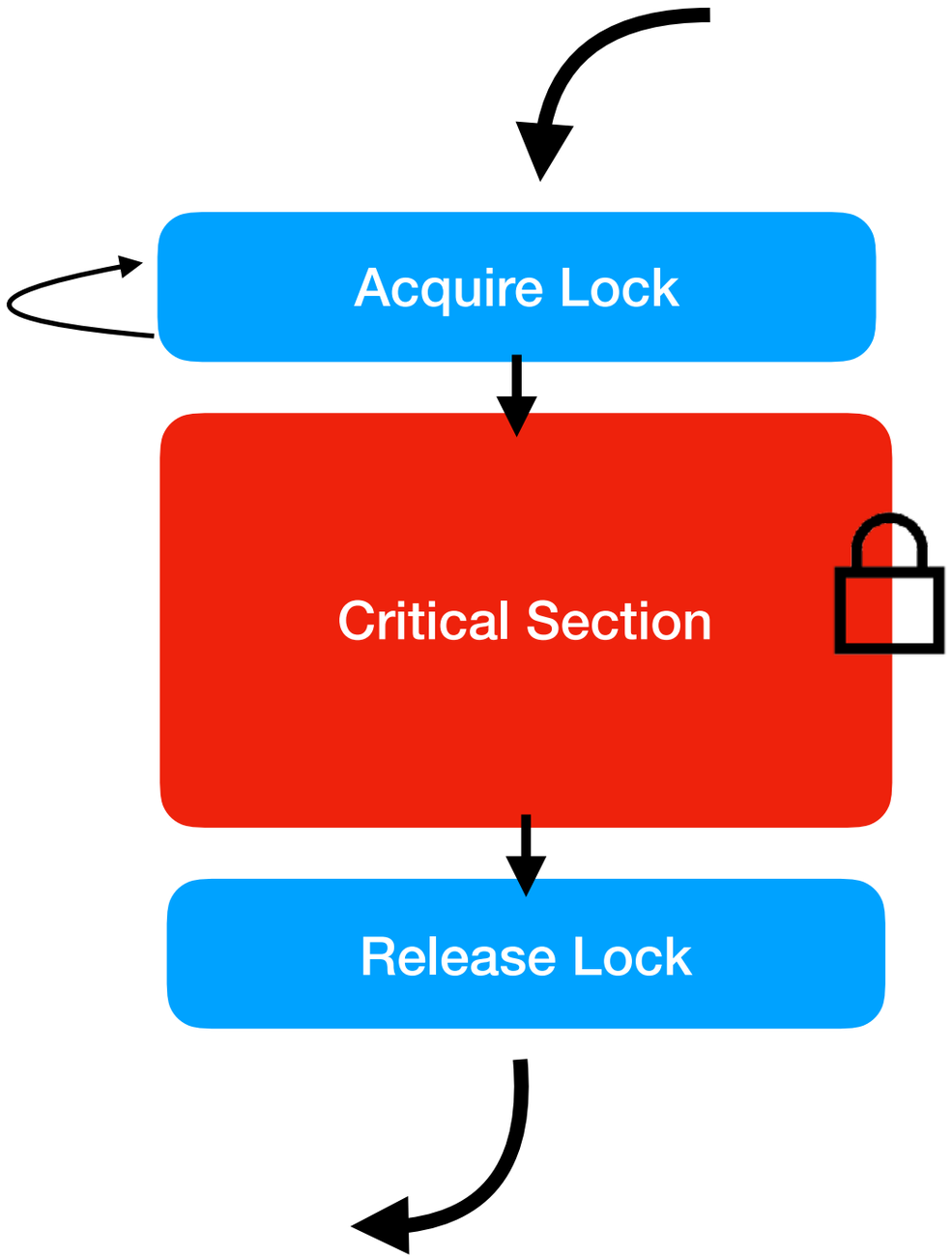
Mutex

Trivial Flow



Mutex

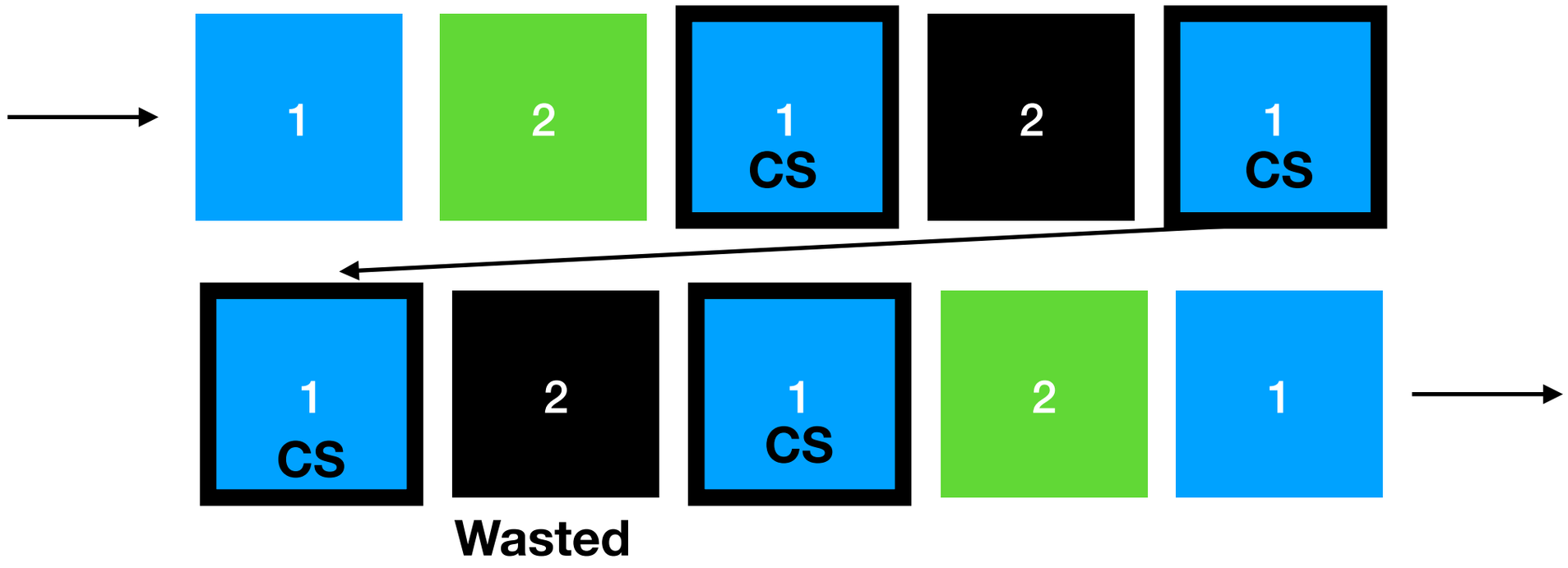
Trivial Flow



Busy Waiting

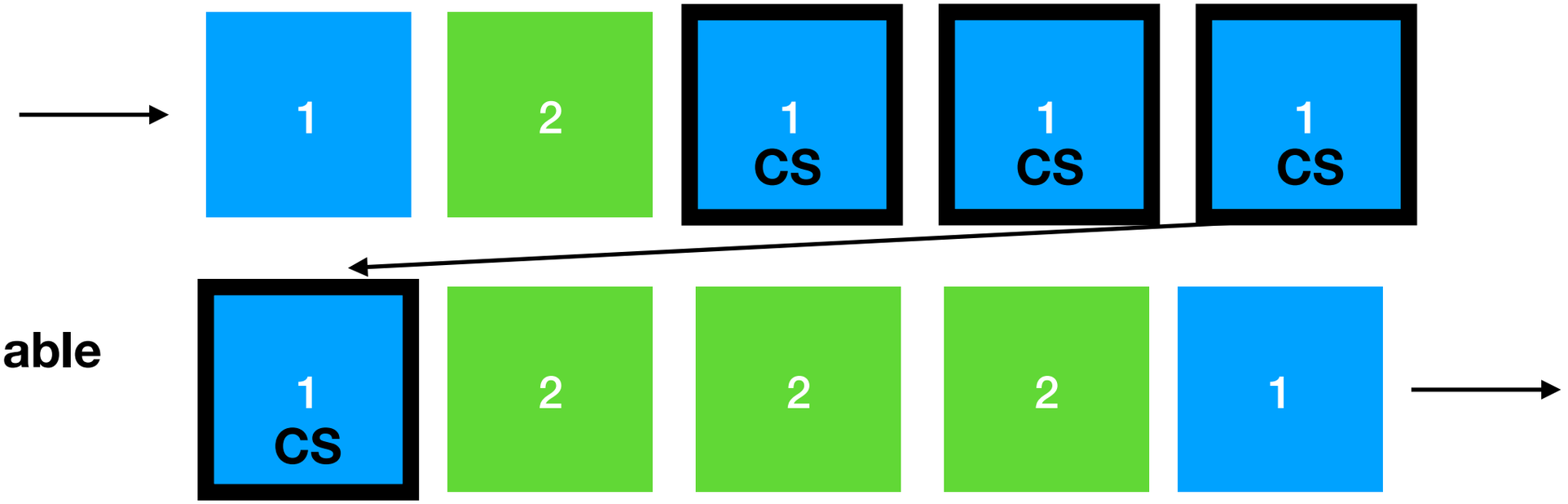
Wasted

Busy Waiting



Wasted

Signal
on
Condition Variable



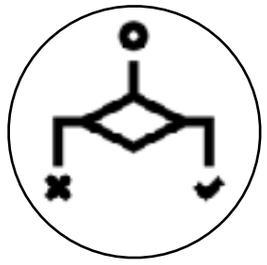
Smarter Flow

Can’t Proceed

Acquire Lock

Wait for Signal

Critical Section



Condition Variable

Release Lock

Signal/BroadCast

Signal other processes waiting for mutex

Mutex

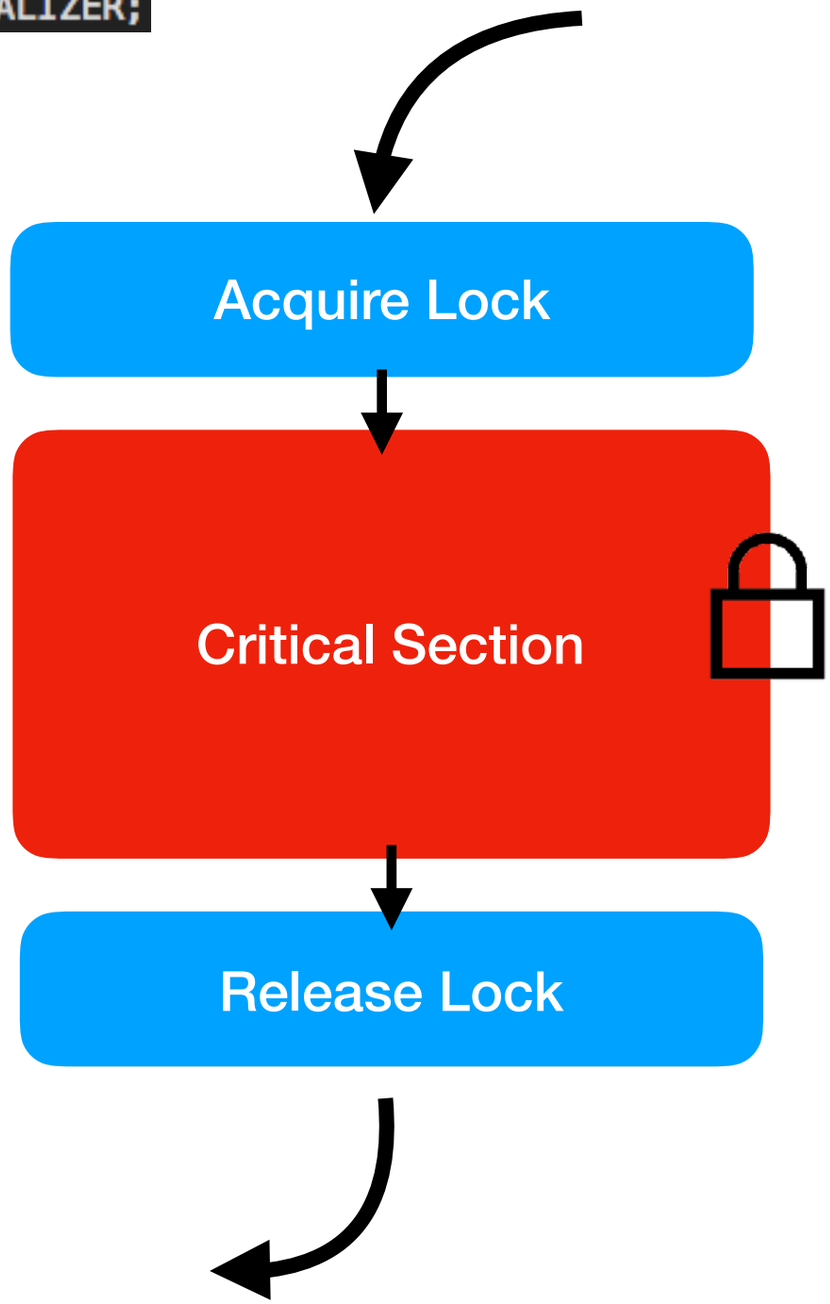
Mutex Usage(Among Threads)

#include<pthread.h>

Global Declaration

```
pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
```

```
pthread_mutex_lock(&m);
```

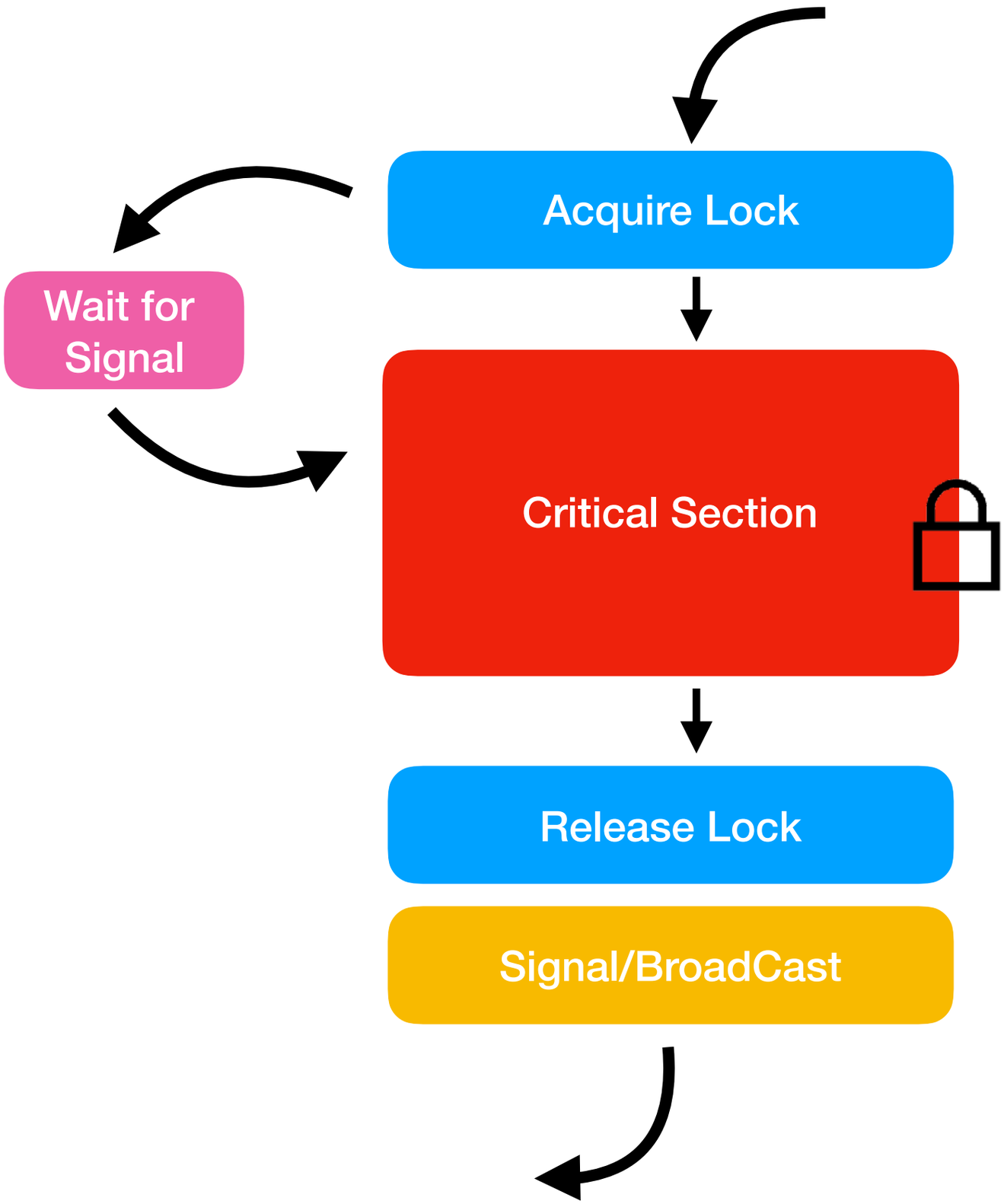



```
pthread_mutex_unlock(&m);
```

Mutex With Condition Variable

```
pthread_cond_t c_cons = PTHREAD_COND_INITIALIZER;
```

Global




```
pthread_mutex_lock(&m);
```

```
pthread_cond_wait(&c_cons, &m);
```

```
pthread_mutex_unlock(&m);
```

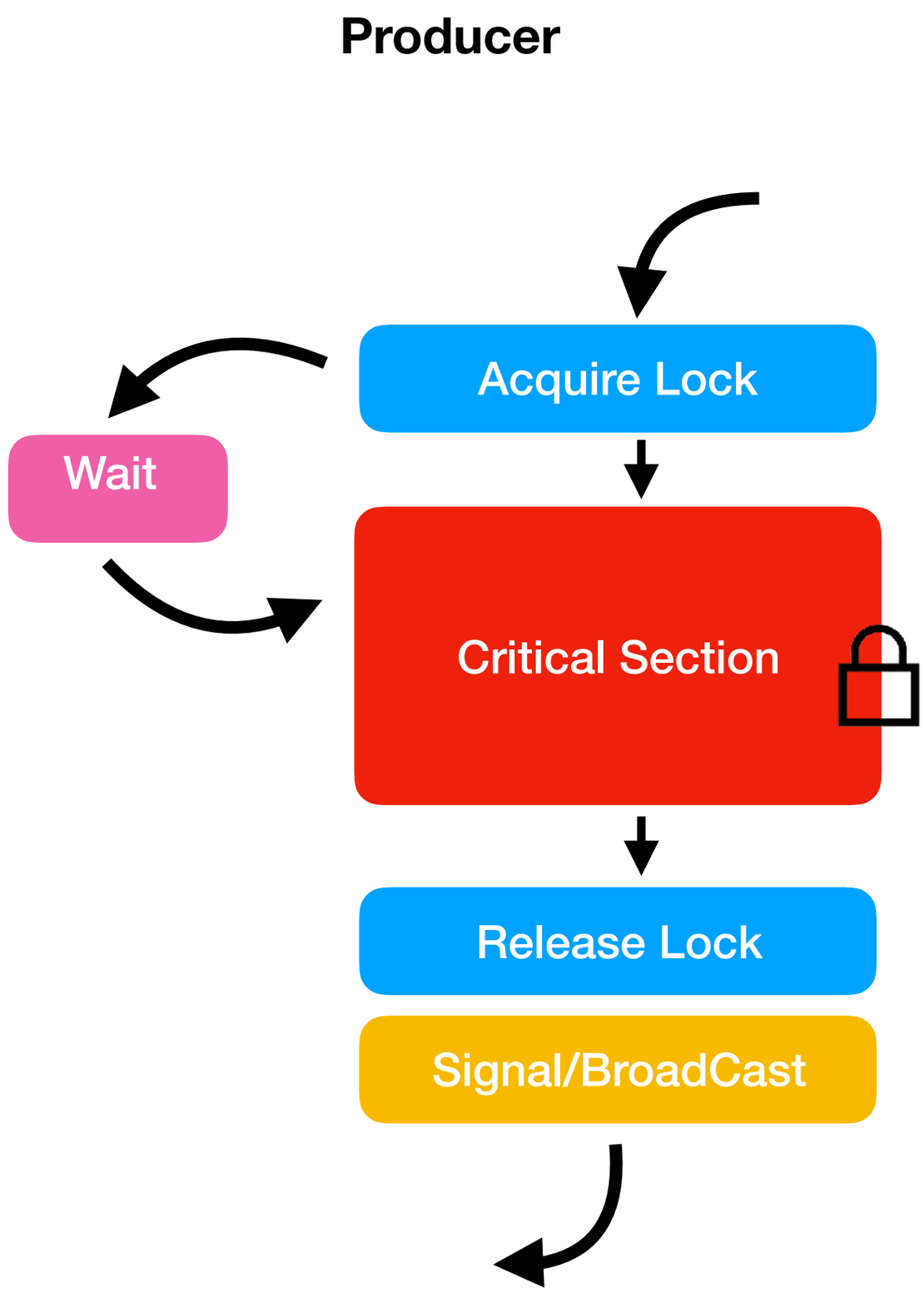
```
pthread_cond_signal(&c_cons);
```

Producer And Consumer

Mutexes ,Threads and condition variables

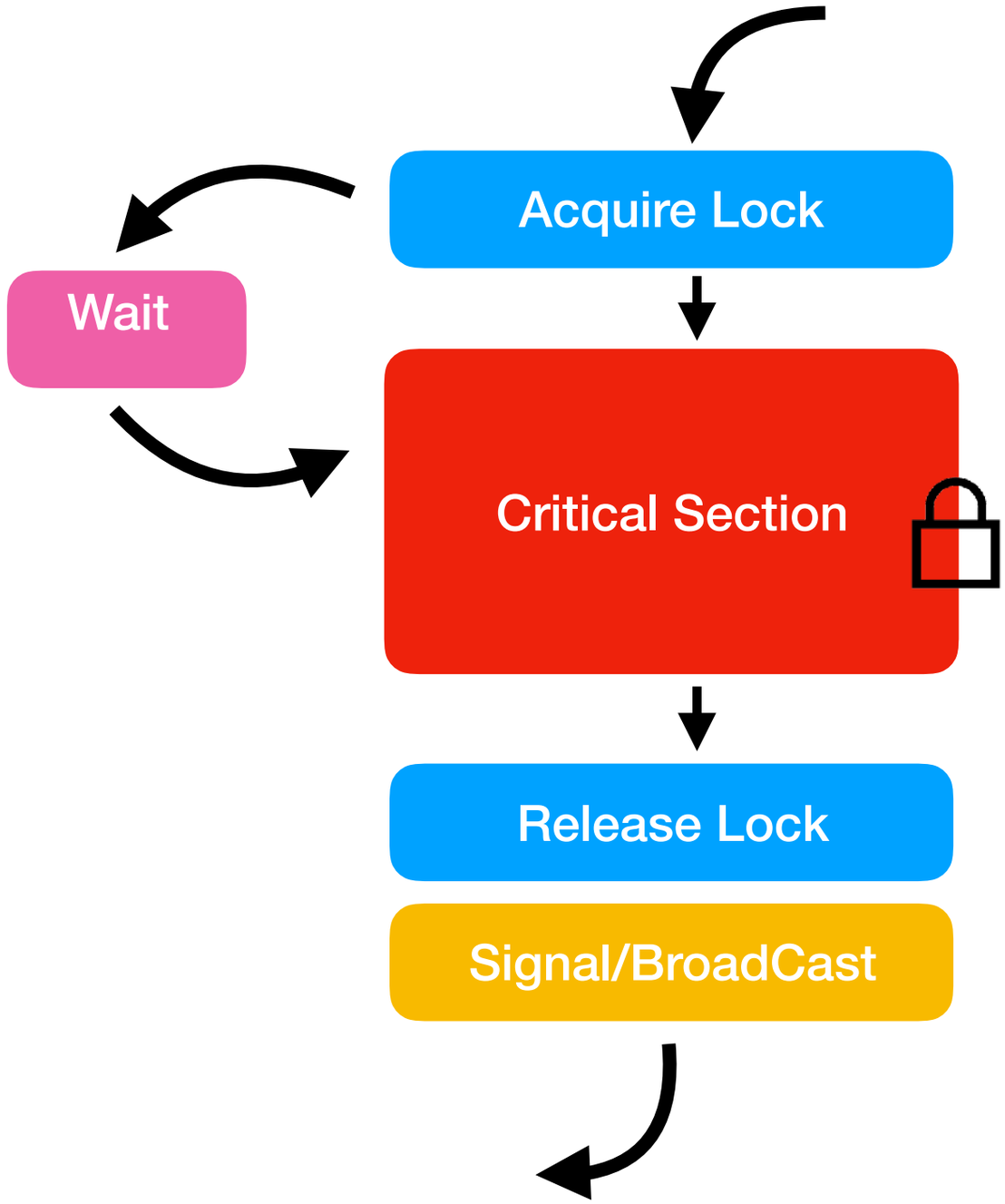
```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>
#include <unistd.h>

#define BUF_SIZE 3 /* Size of shared buffer */

int buffer[BUF_SIZE]; /* shared buffer */
int add = 0;          /* place to add next element */
int rem = 0;          /* place to remove next element */
int num = 0;          /* number elements in buffer */

pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;  /* mutex lock for buffer */
pthread_cond_t c_cons = PTHREAD_COND_INITIALIZER; /* consumer waits on this cond var */
pthread_cond_t c_prod = PTHREAD_COND_INITIALIZER; /* producer waits on this cond var */

void *producer(void *param);
void *consumer(void *param);
```

```c
/* Produce value(s) */
void *producer(void *param)
{
    int w;
    int i = 0;
    while (1)
    {
        w = rand() % 10;
        sleep(w);

        /* Insert into buffer */
        pthread_mutex_lock(&m);
        if (num > BUF_SIZE)
        {
            exit(1); /* overflow */
        }

        while (num == BUF_SIZE)
        { /* block if buffer is full */
            pthread_cond_wait(&c_prod, &m);
        }

        /* if executing here, buffer not full so add element */
        buffer[add] = i;
        add = (add + 1) % BUF_SIZE;
        num++;

        pthread_mutex_unlock(&m);

        pthread_cond_signal(&c_cons);
        printf("producer: inserted %d\n", i);
        i++;
        fflush(stdout);
    }

    printf("producer quiting\n");
    fflush(stdout);
    return 0;
}
```



```c
/* Consume value(s); Note the consumer never terminates */
void *consumer(void *param)
{

    int i;
    int w;
    while (1)
    {
        w = rand() % 10;
        sleep(w);
        pthread_mutex_lock(&m);
        if (num < 0)
        {
            exit(1);
        } /* underflow */

        while (num == 0)
        { /* block if buffer empty */
            pthread_cond_wait(&c_cons, &m);
        }

        /* if executing here, buffer not empty so remove element */
        i = buffer[rem];
        rem = (rem + 1) % BUF_SIZE;
        num--;

        pthread_mutex_unlock(&m);

        pthread_cond_signal(&c_prod);
        printf("Consume value %d\n", i);
        fflush(stdout);
    }
    return 0;
}
```

Consumer



```
Last login: Mon Oct 16 11:14:54 on ttys002
[aakashnandi ~ $ cd Desktop/mutex
[aakashnandi mutex $ ls
a.out           a.out.dSYM      main.c
[aakashnandi mutex $ ./a.out
producer: inserted 0
producer: inserted 1
Consume value 0
producer: inserted 2
Consume value 1
producer: inserted 3
producer: inserted 4
Consume value 2
producer: inserted 5
Consume value 3
producer: inserted 6
Consume value 4
producer: inserted 7
Consume value 5
producer: inserted 8
Consume value 6
producer: inserted 9
Consume value 7
producer: inserted 10
Consume value 8
producer: inserted 11
Consume value 9
Consume value 10
producer: inserted 12
Consume value 11
producer: inserted 13
Consume value 12
producer: inserted 14
Consume value 13
producer: inserted 15
producer: inserted 16
Consume value 14
producer: inserted 17
Consume value 15
Consume value 16
Consume value 17
```

Output

Please use MAN terminal command to find out the parameters of syntax

Shared Memory

Shared between different Processes.

```
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
```

Open SharedMem File

↓

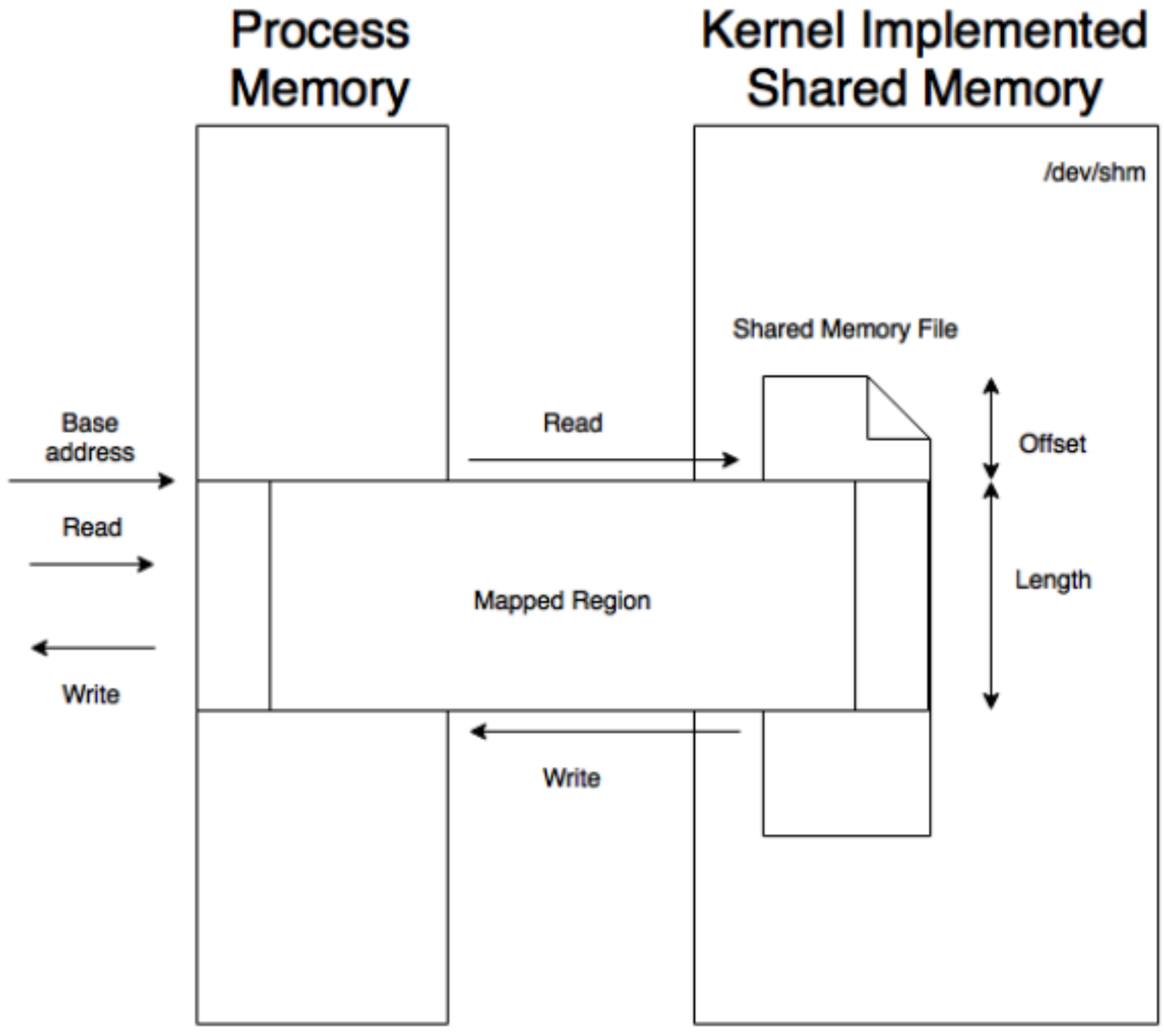
Map it kernel Shared Mem

↓

Read or write

↓

Close the file



Shared Memory

```c
const int SIZE = 4096;
const char *name = "OS";
const char *message0= "Studying ";
const char *message1= "Operating Systems ";
const char *message2= "Is Fun!";

int shm_fd;
void *ptr;

/* create the shared memory segment */
shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);

/* configure the size of the shared memory segment */
ftruncate(shm_fd,SIZE);
```

Open SharedMem File

↓

Map it kernel Shared Mem

↓

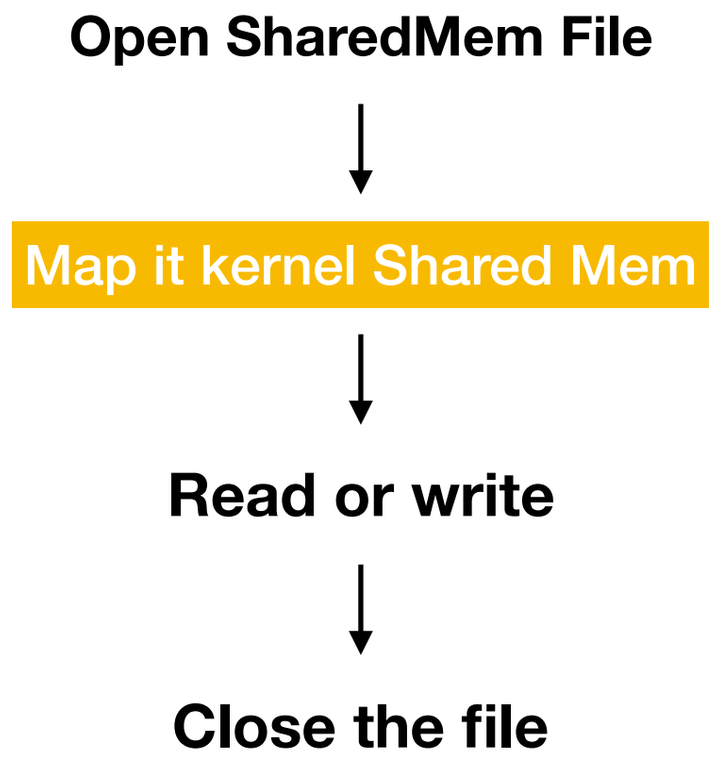
Read or write

↓

Close the file

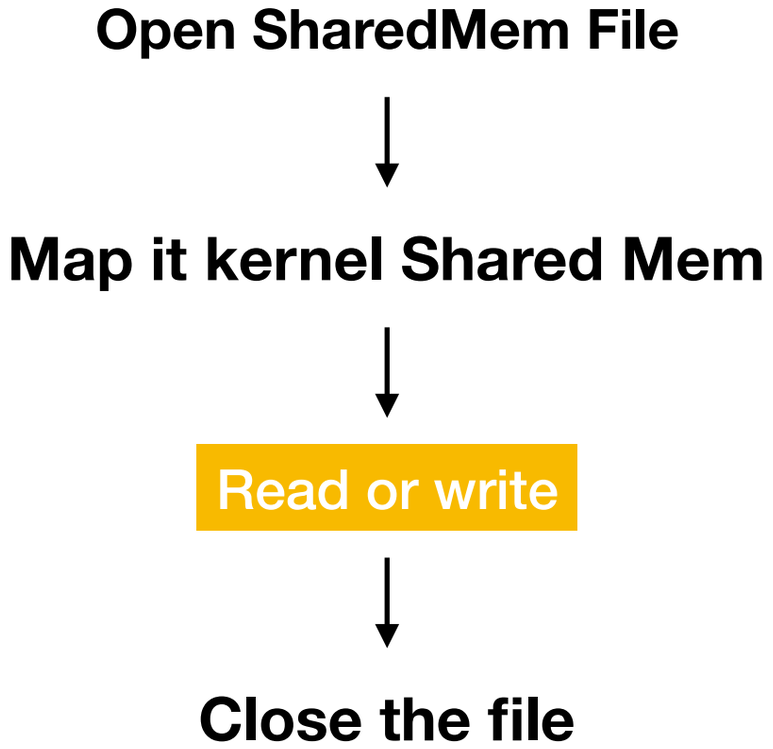
Shared Memory

```c
/* now map the shared memory segment in the address space of the process */
ptr = mmap(0,SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
if (ptr == MAP_FAILED) {
    printf("Map failed\n");
    return -1;
}
```



Shared Memory

```
sprintf(ptr,"%s",message0);
ptr += strlen(message0);
sprintf(ptr,"%s",message1);
ptr += strlen(message1);
sprintf(ptr,"%s",message2);
ptr += strlen(message2);
```



Shared Memory

```c
/* remove the shared memory segment */
if (shm_unlink(name) == -1) {
    printf("Error removing %s\n",name);
    exit(-1);
}
```

Open SharedMem File

↓

Map it kernel Shared Mem

↓

Read or write

↓

Close the file

Server Client

Between Processes

Server

Open SharedMem File

↓

Map it kernel Shared Mem

↓

Read or write

Client

Open SharedMem File

↓

Map it kernel Shared Mem

↓

Read or write

↓

Close the file

Server

```c
int main()
{
    const int SIZE = 4096;
    const char *name = "OS";
    const char *message0= "Studying ";
    const char *message1= "Operating Systems ";
    const char *message2= "Is Fun!";

    int shm_fd;
    void *ptr;

    /* create the shared memory segment */
    shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);

    /* configure the size of the shared memory segment */
    ftruncate(shm_fd,SIZE);

    /* now map the shared memory segment in the address space of the process */
    ptr = mmap(0,SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (ptr == MAP_FAILED) {
        printf("Map failed\n");
        return -1;
    }

    /**
     * Now write to the shared memory region.
     *
     * Note we must increment the value of ptr after each write.
     */
    sprintf(ptr,"%s",message0);
    ptr += strlen(message0);
    sprintf(ptr,"%s",message1);
    ptr += strlen(message1);
    sprintf(ptr,"%s",message2);
    ptr += strlen(message2);

    return 0;
}
```

```c
int main()
{
    const char *name = "OS";
    const int SIZE = 4096;

    int shm_fd;
    void *ptr;
    int i;
    int ret;

    /* open the shared memory segment */
    shm_fd = shm_open(name, O_RDONLY, 0666);
    if (shm_fd == -1) {
        printf("shared memory failed\n");
        exit(-1);
    }

    /* now map the shared memory segment in the address space of the process */
    ptr = mmap(0,SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
    if (ptr == MAP_FAILED) {
        printf("Map failed\n");
        exit(-1);
    }

    /* now read from the shared memory region */
    while(*(char*)ptr!='\0')
    {
    ret=printf("%s",ptr);
    ptr=ptr+ret;
    }
    /* remove the shared memory segment */
    if (shm_unlink(name) == -1) {
        printf("Error removing %s\n",name);
        exit(-1);
    }

    return 0;
}
```

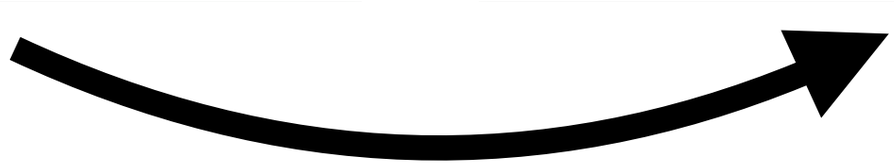
Client

Server

```
Last login: Mon Oct 16 09:34:37 on ttys000
[aakashnandi ~ $ cd Desktop/sharedmem
[aakashnandi sharedmem $ ls
a.out           client.c        consumer.dSYM   server
a.out.dSYM      client.dSYM     producer        server.c
client          consumer        producer.dSYM   server.dSYM
[aakashnandi sharedmem $ ./server
 aakashnandi sharedmem $
```

Client

```
Last login: Mon Oct 16 11:14:25 on ttys001
[aakashnandi ~ $ cd Desktop/sharedmem
[aakashnandi sharedmem $ ls
a.out           client.c        consumer.dSYM   server
a.out.dSYM      client.dSYM     producer        server.c
client          consumer        producer.dSYM   server.dSYM
[aakashnandi sharedmem $ ./client
Studying Operating Systems Is Fun!aakashnandi sharedmem $
```



Thank You