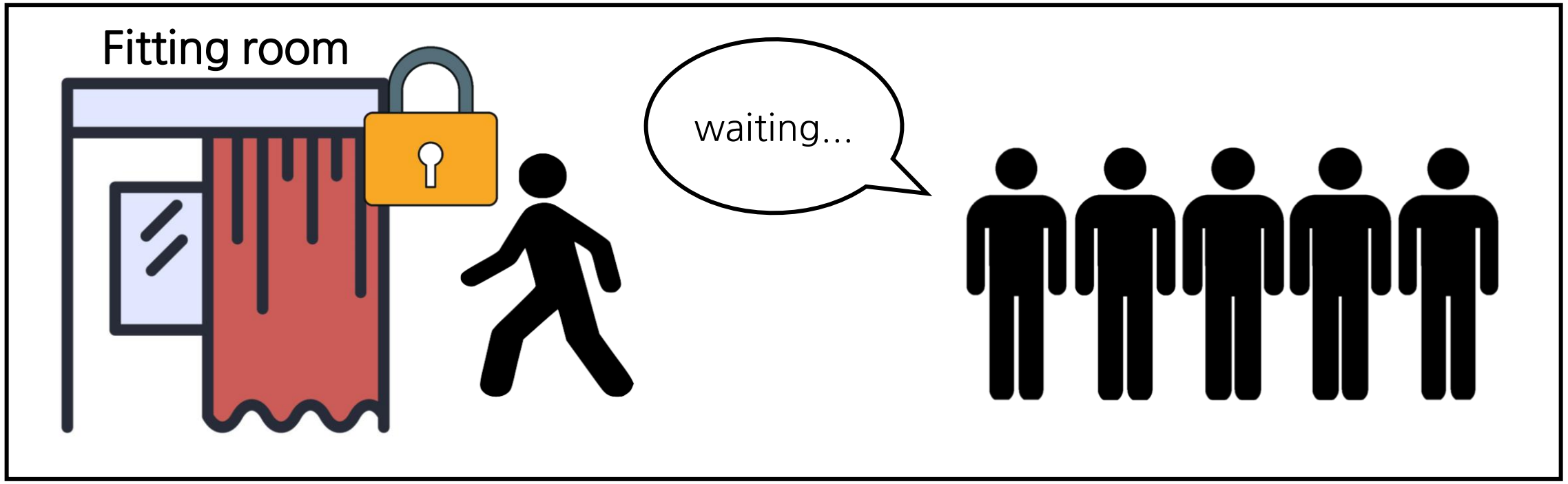
운영체제 및 실습
- Concurrency -

한예진, 김수창, 이승준
Dankook University

목차

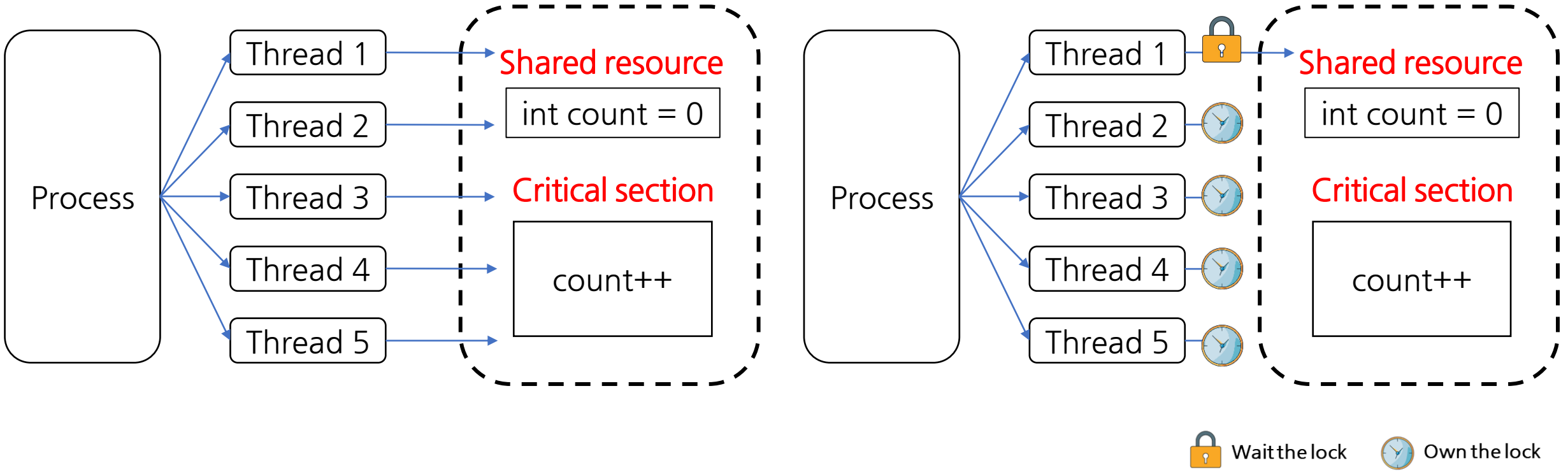
Lock

- Lock programming
  - Lock, Unlock
  - Multi process(thread)
    - Shared resource



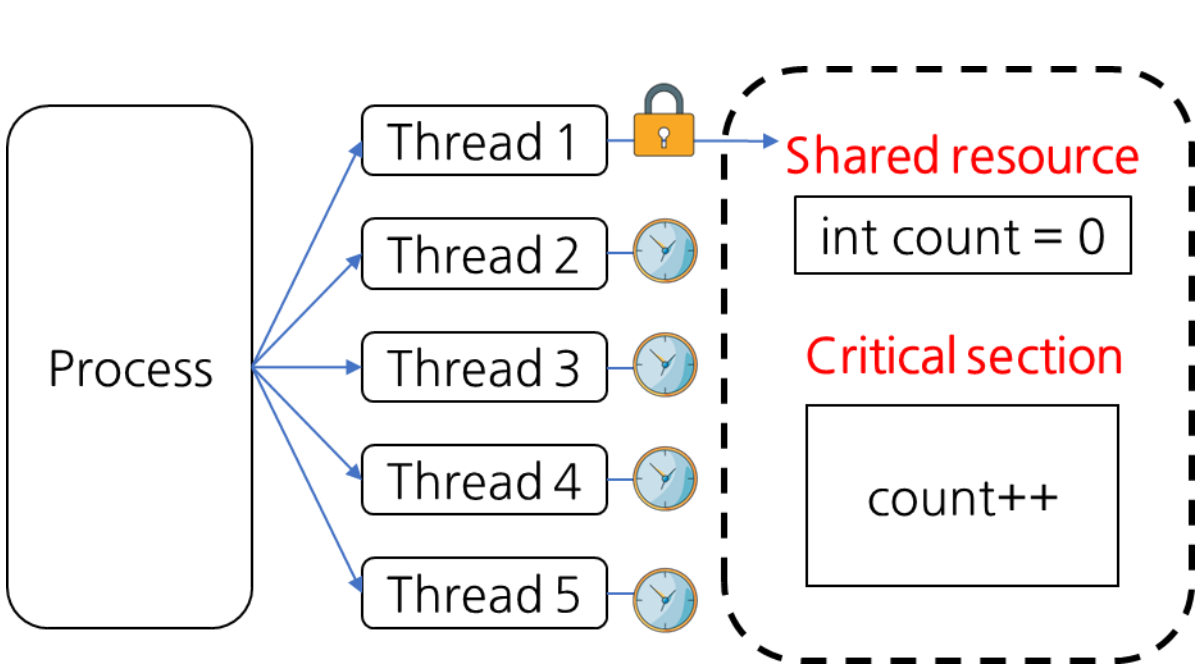
Lock

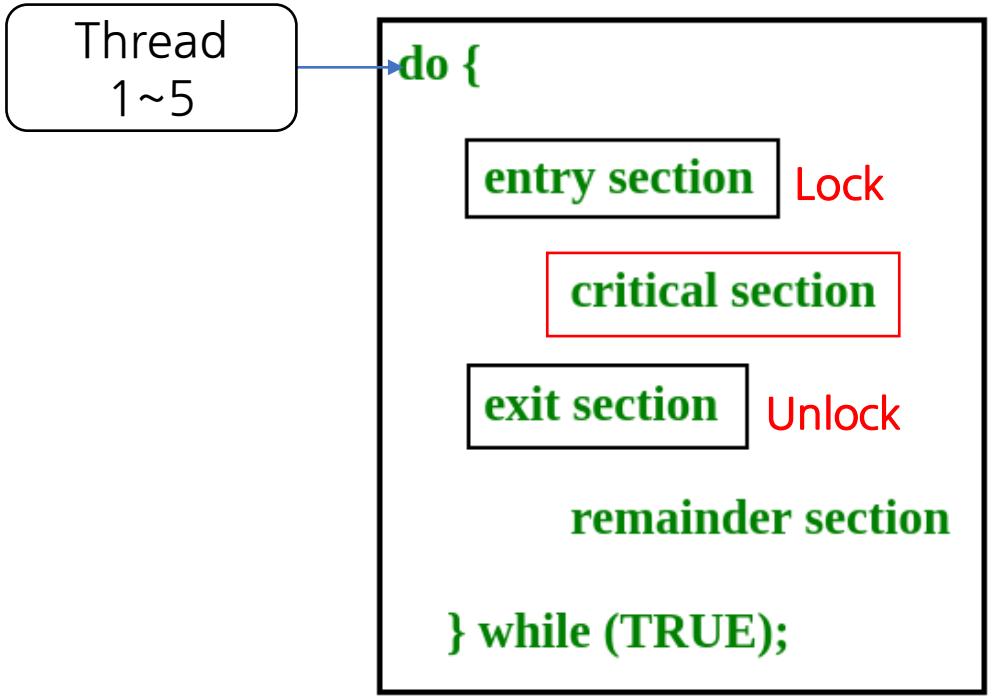
- Lock programming
  - Shared resource
  - Critical Section



Lock

- Lock programming
  - Shared resource
  - Critical Section





실습 1: multi-thread without lock

- thread.c
  - pthread_create(), pthread_join()

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

int count = 0; // shared resource
int nthread=  1;
int worker_loop_cnt = 1;

static void *worker(void *num);
```

```c
int main(int argc, char *argv[]){
    pthread_t *th;
    void *value;
    long i;

    if (argc < 3){
        fprintf(stderr, "%s parameter : nthread, worker_loop_cnt\n", argv[0]);
        exit(-1);
    }

    nthread = atoi(argv[1]);
    worker_loop_cnt = atoi(argv[2]);
    th = malloc(sizeof(pthread_t) * nthread);
    printf("main: begin (count = %d)\n", count);
    for (i=0; i < nthread; i++)
        assert(pthread_create(&th[i], NULL, worker, (void*) i) == 0);

    for (i=0; i < nthread; i++)
        assert(pthread_join(th[i], &value) == 0);
    printf("main: done (count = %d)\n", count);
    return 0;
}
```

```
13:33 [Top]                    /home/yejin/TABA/concurrency/exercise_1/thread.c\
```

실습 1: multi-thread without lock

- thread.c
  - Critical section

```
36
37 static void *worker(void *num)
38 {
39     int number = (int)num;
40     for (int i=0; i < worker_loop_cnt; i++)
41         count++;
42     printf("Thread number %d: %d \n", number, count);
43     return NULL;
<%] [+] /home/yejin/TABA/concurrency/exercise_1/thread.c\
```

실습 1: multi-thread without lock

- 실행파일

```
# gcc thread.c -lpthread -o thread.out
```

- 결과

```
root@yejin:/home/yejin/TABA/concurrency/exercise_1# ./thread.out 5 10000
main: begin (count = 0)
Thread number 0: 11108
Thread number 2: 12843
Thread number 1: 14259
Thread number 4: 18516
Thread number 3: 28312
main: done (count = 28312)
```

Not deterministic!!!

실습 2: multi-thread with lock

- thread_lock.c
  - Lock creation / initilization

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

int count = 0; // shared resource
int nthread=  1;
int worker_loop_cnt = 1;
pthread_mutex_t lock; // mutex variable
static void *worker(void *num);
```

```c
int main(int argc, char *argv[]){
    pthread_t *th;
    void *value;
    long i;

    if (argc < 3){
        fprintf(stderr, "%s parameter : nthread, worker_loop_cnt\n", argv[0]);
        exit(-1);
    }

    nthread = atoi(argv[1]);
    worker_loop_cnt = atoi(argv[2]);
    th = malloc(sizeof(pthread_t) * nthread);
    pthread_mutex_init(&lock, NULL);
    printf("main: begin (count = %d)\n", count);
    for (i=0; i < nthread; i++)
        assert(pthread_create(&th[i], NULL, worker, (void*) i) == 0);

    for (i=0; i < nthread; i++)
        assert(pthread_join(th[i], &value) == 0);
    printf("main: done (count = %d)\n", count);
    return 0;
}
```

```
15:17 [Top]          [+] /home/yejin/TABA/concurrency/exercise_2/thread_lock.c\
```

실습 2: multi-thread with lock

- thread_lock.c
  - pthread_mutex_lock(pthread_mutex_t *mutex)
  - pthread_mutex_unlock(pthread_mutex_t *mutex)

```
38 static void *worker(void *num)
39 {
40     int number = (int)num;
41
42     pthread_mutex_lock(&lock); //lock
43     for (int i=0; i < worker_loop_cnt; i++)
44         count++;
45     printf("Thread number %d: %d \n", number, count);
46
47     pthread_mutex_unlock(&lock); //unlock
48
49     return NULL;
50 }
<] /home/yejin/TABA/concurrency/exercise_2/thread_lock.c\
```

실습 2: multi-thread with lock

- 실행파일

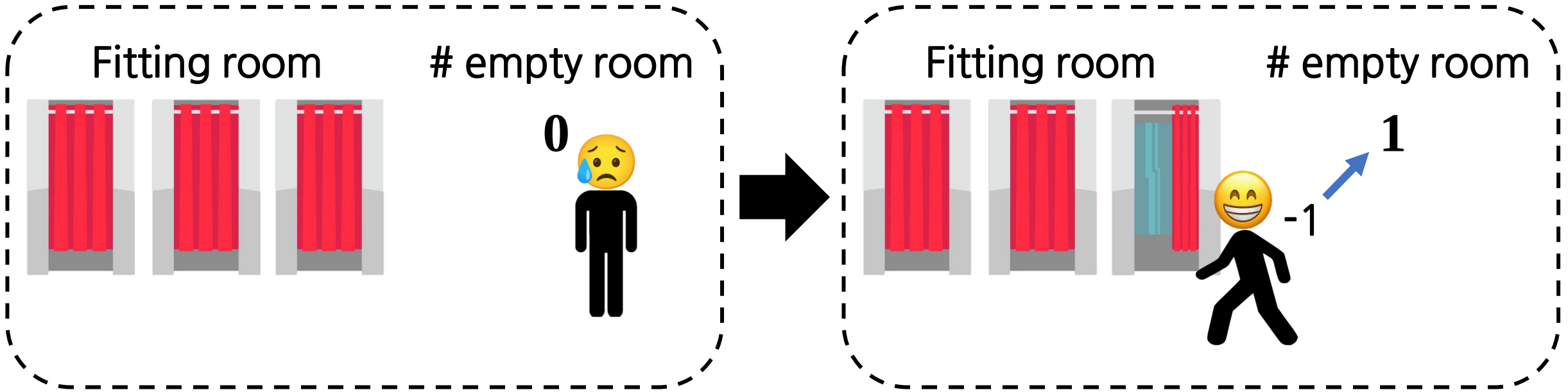
```
# gcc thread_lock.c -lpthread -o thread_lock.out
```

- 결과

```
root@yejin:/home/yejin/TABA/concurrency/exercise_2# ./thread_lock.out 5 10000
main: begin (count = 0)
Thread number 0: 10000
Thread number 1: 20000
Thread number 2: 30000
Thread number 3: 40000
Thread number 4: 50000
main: done (count = 50000)
```

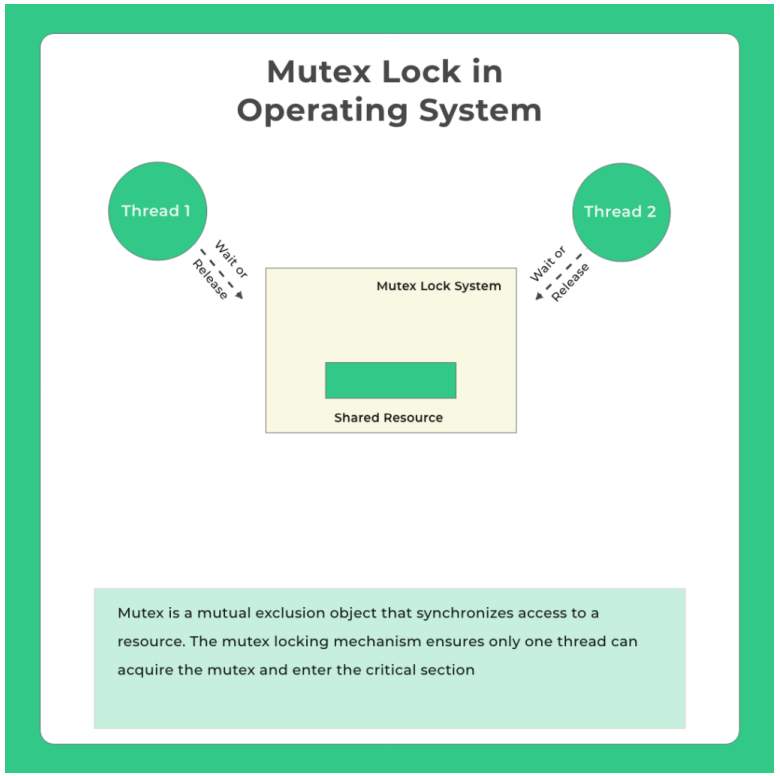
Lock: Mutex, Semaphore

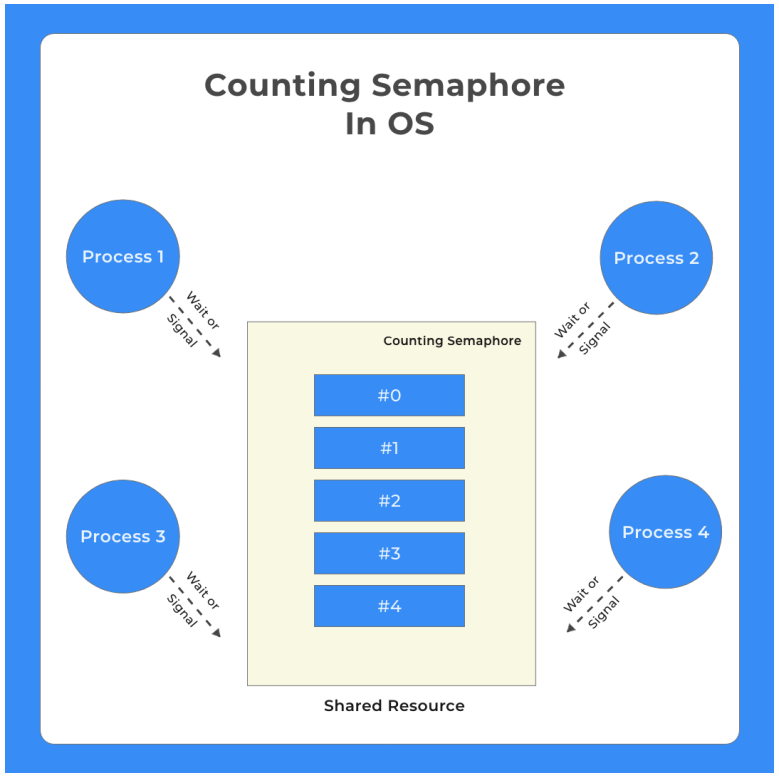
- Lock programming
  - Mutex
  - Semaphore

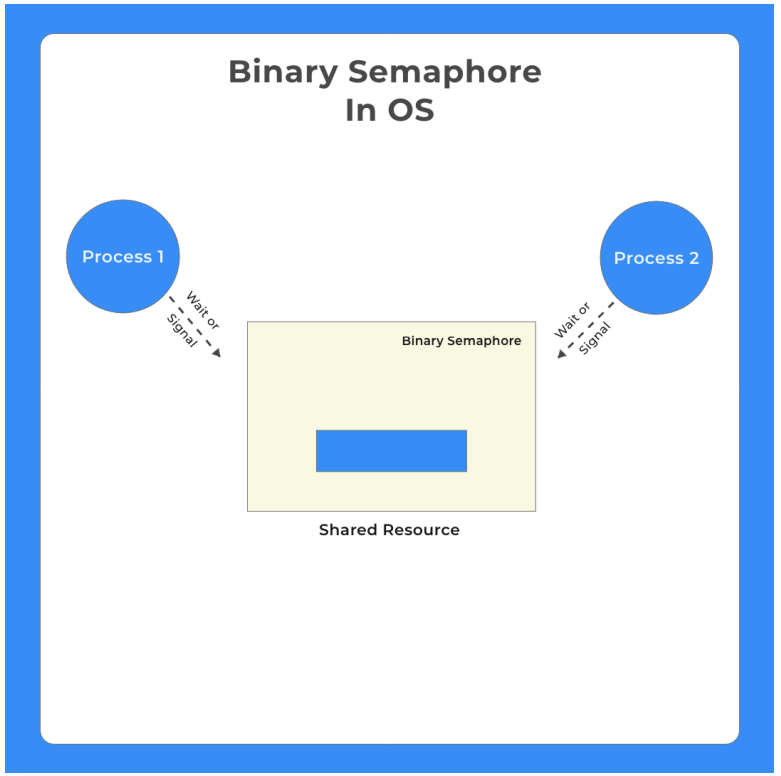


Lock: Mutex, Semaphore

- Lock programming
  - Mutex
  - Semaphore







 (Source: https://prepinsta.com/operating-systems/mutex-vs-semaphore/)

실습 3: binary semaphore

- thread_bin_sem.c: semaphore를 lock처럼 사용하기
  - sem_init(semaphore, p_shared, initial_value), sem_destroy(semaphore)

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

int count = 0;
int nthread = 1;
int worker_loop_cnt = 1;
sem_t semaphore;
static void *work(void *num);

int main(int argc, char *argv[]){
    pthread_t *th;
    void *value;
    long i;

    if(argc < 3){
        fprintf(stderr, "%s parameter : nthread, worker_loop_cnt\n", argv[0]);
        exit(-1);
    }

    nthread = atoi(argv[1]);
    worker_loop_cnt = atoi(argv[2]);
    th = malloc(sizeof(pthread_t)*nthread);

    sem_init(&semaphore, 0, 1);
    printf("main: begin (count = %d)\n",count);
    for(i=0; i<nthread; i++)
        assert(pthread_create(&th[i], NULL, work, (void*) i) == 0);
    for (i=0; i<nthread; i++)
        assert(pthread_join(th[i], &value) == 0);
    sem_destroy(&semaphore);
    printf("main: done (count = %d)\n", count);
}
```

[+] /home/yejin/TABA/concurrency/exercise_3/thread_bin_sem.c\

실습 3: binary semaphore

- thread_bin_sem.c
  - sem_wait(): semaphore 값 감소
  - Sem_post(): semaphore 값 증가

```
38 static void *work(void* num){
39     int number = (int)num;
40
41     sem_wait(&semaphore);
42
43     for(int i = 0; i < worker_loop_cnt; i++)
44         count++;
45     printf("Thread number %d: %d \n", number, count);
46
47     sem_post(&semaphore);
48
49     return NULL;
50 }
~
<] /home/yejin/TABA/concurrency/exercise_3/thread_bin_sem.c\
```

실습 3: binary semaphore

- 실행파일

```
# gcc thread_bin_sem.c -lpthread -o thread_bin_sem.out
```

- 결과

```
root@yejin:/home/yejin/TABA/concurrency/exercise_3# ./thread_bin_sem_out 5 10000
main: begin (count = 0)
Thread number 0: 10000
Thread number 1: 20000
Thread number 3: 30000
Thread number 4: 40000
Thread number 2: 50000
main: done (count = 50000)
```

실습 3: counting semaphore

- thread_counting_sem.c

```
 1 #include <stdlib.h>
 2 #include <unistd.h>
 3 #include <stdio.h>
 4 #include <assert.h>
 5 #include <pthread.h>
 6 #include <semaphore.h>
 7
 8 #define SEM_COUNT 3
 9
10 int count[SEM_COUNT];
11 int working [SEM_COUNT];
12 int nthread = 1;
13 int worker_loop_cnt = 1;
14
15 pthread_mutex_t lock;
16 sem_t semaphore;
<home/yejin/TABA/concurrency/thread_counting_sem.c\
```

실습 3: counting semaphore

- thread_counting_sem.c

```c
int main(int argc, char *argv[])
{
    pthread_t *th;
    void *value;
    long i;

    if (argc < 3) {
        fprintf(stderr, "%s parameter : nthread, worker_loop_cnt\n", argv[0]);
        exit(-1);
    }

    nthread = atoi(argv[1]);
    worker_loop_cnt = atoi(argv[2]);

    th = malloc(sizeof(pthread_t) * nthread);

    pthread_mutex_init(&lock, NULL); // initialize the lock
    sem_init(&semaphore, 0, SEM_COUNT); // init sem

    for(i = 0; i < nthread; i++) {
        assert(pthread_create(&th[i], NULL, work, (void*) i) == 0);
    }

    for(i = 0; i < nthread; i++) {
        assert(pthread_join(th[i], &value) == 0);
    }

    sem_destroy(&semaphore);

    free(th);

    printf("Count array : \n");
    for(int i = 0; i < SEM_COUNT; i++){
        printf("%d ", count[i]);
    }

    printf("\nComplete\n");
```

<home/yejin/TABA/concurrency/thread_counting_sem.c\

실습 3: counting semaphore

- thread_counting_sem.c

```
# gcc thread_counting_sem.c -lpthread -o thread_counting_sem.out
```

```c
static void *work(void* num)
{
    int number = (int)num;
    int count_index = -1;

    sem_wait(&semaphore);// sem count down

    pthread_mutex_lock(&lock); // lock
    for(int i = 0; i < SEM_COUNT; i++){
        if(working[i] == 0){
            working[i] = 1;
            count_index = i;
            break;
        }
    }
    pthread_mutex_unlock(&lock); // unlock

    if(count_index == -1){
        fprintf(stderr, "Thread number %d: count_index < 0", number);
        exit(-1);
    }

    for(int i = 0; i < worker_loop_cnt; i++)
        count[count_index]++;

    //printf("Thread number %d: %d \n", number, count[count_index]);

    pthread_mutex_lock(&lock); // lock
    working[count_index] = 0;
    pthread_mutex_unlock(&lock); // unlock

    sem_post(&semaphore);// sem count up

    return NULL;
}
```

<home/yejin/TABA/concurrency/thread_counting_sem.c\

실습 3: counting semaphore

- 실행파일

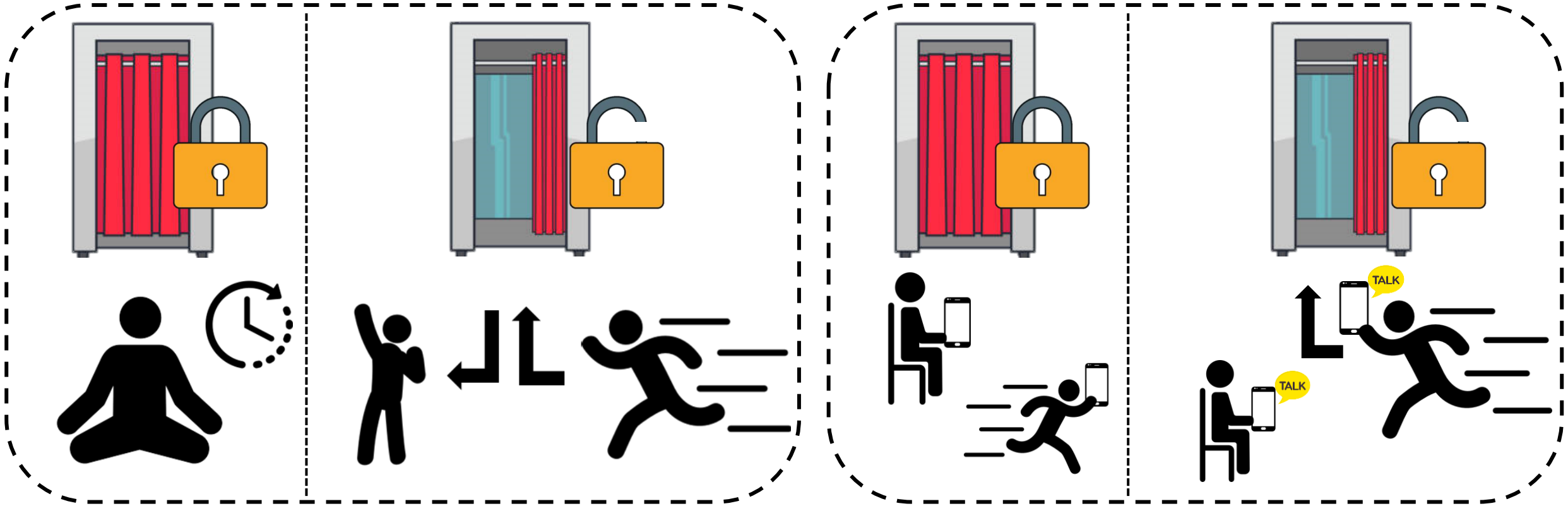
```
# gcc thread_counting_sem.c -lpthread -o thread_counting_sem.out
```

- 결과

```
root@yejin:/home/yejin/TABA/concurrency/exercise_3# ./thread_counting_sem.out 5 100000
Count array :
200000 100000 200000
Complete
```

Lock: Wait

- Lock programming
  - Busy wait
  - Block/wakeup



실습 4: Dead lock

- thread_2lock.c

```
 1 #include <stdlib.h>
 2 #include <unistd.h>
 3 #include <stdio.h>
 4 #include <assert.h>
 5 #include <pthread.h>
 6
 7 int first_count = 0;
 8 int second_count = 0;
 9 int nthread = 1;
10 int nthread_one = 1;
11
12 int main_loop_cnt = 1;
13
14 pthread_mutex_t first_lock;
15 pthread_mutex_t second_lock;
16
17 static void *work_one(void* num)
18 {
19     int number = (int)num;
20     int answer = 0;
<  /home/yejin/TABA/concurrency/exercise_4/thread_2lock.c\
```

실습 4: Dead lock

- thread_2lock.c

```
59 int main(int argc, char *argv[])
60 {
61     pthread_t *th;
62     void *value;
63     long i;
64
65     if (argc < 3) {
66         fprintf(stderr, "%s parameter : nthread, main_loop_cnt\n", argv[0]);
67         exit(-1);
68     }
69
70     nthread = atoi(argv[1]);
71     nthread_one = nthread/2;
72
73     main_loop_cnt = atoi(argv[2]);
74
75     th = malloc(sizeof(pthread_t) * nthread);
76
```

```
77     pthread_mutex_init(&first_lock, NULL); // initialize the lock
78     pthread_mutex_init(&second_lock, NULL); // initialize the lock
79
80     for(int loop = 0; loop < main_loop_cnt; loop++){
81         printf("---- loop %d ----\n", loop);
82
83         for(i = 0; i < nthread_one; i++)
84             assert(pthread_create(&th[i], NULL, work_one, (void*) i) == 0);
85
86         for(i = nthread_one; i < nthread; i++)
87             assert(pthread_create(&th[i], NULL, work_two, (void*) i) == 0);
88
89         for(i = 0; i < nthread; i++)
90             assert(pthread_join(th[i], &value) == 0);
91
92         first_count = 0;
93         second_count = 0;
94     }
86:47 [94%]              /home/yejin/TABA/concurrency/exercise_4/thread_2lock.c\
```

실습 4: Dead lock

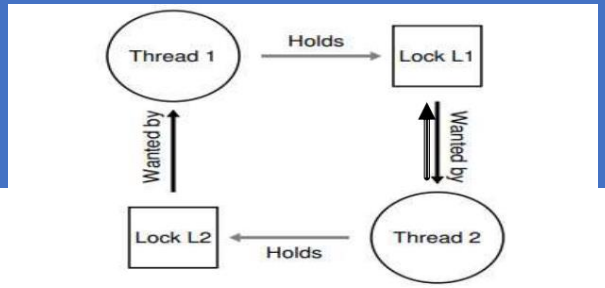


Figure 32.7: **The Deadlock Dependency Graph**

- thread_2lock.c

```c
17 static void *work_one(void* num)
18 {
19     int number = (int)num;
20     int answer = 0;
21
22     pthread_mutex_lock(&first_lock); // lock
23     pthread_mutex_lock(&second_lock); // lock
24
25     answer = first_count + second_count;
26
27     printf("Work_one : %d \n", answer);
28
29     first_count++;
30     second_count++;
31
32     pthread_mutex_unlock(&second_lock); // unlock
33     pthread_mutex_unlock(&first_lock); // unlock
34
35     return NULL;
36 }
```

```c
38 static void *work_two(void* num)
39 {
40     int number = (int)num;
41     int answer = 0;
42
43     pthread_mutex_lock(&second_lock); // lock
44     pthread_mutex_lock(&first_lock); // lock
45
46     answer = first_count + second_count;
47
48     printf("Work_two : %d \n", answer);
49
50     first_count++;
51     second_count++;
52
53     pthread_mutex_unlock(&first_lock); // unlock
54     pthread_mutex_unlock(&second_lock); // unlock
55
56     return NULL;
57 }
58
57:1 [29%]                /home/yejin/TABA/concurrency/exercise_4/thread_2lock.c\
```

```
# gcc thread_2lock.c -lpthread -o thread_2lock.out
```

실습 4: Dead lock

- 결과

```
root@yejin:/home/yejin/TABA/concurrency/exercise_4# ./dl.out 1000 4
---- loop 0 ----
Aquiring mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_one : 0
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Aquiring mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_one : 2
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Aquiring mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_one : 4
```

…

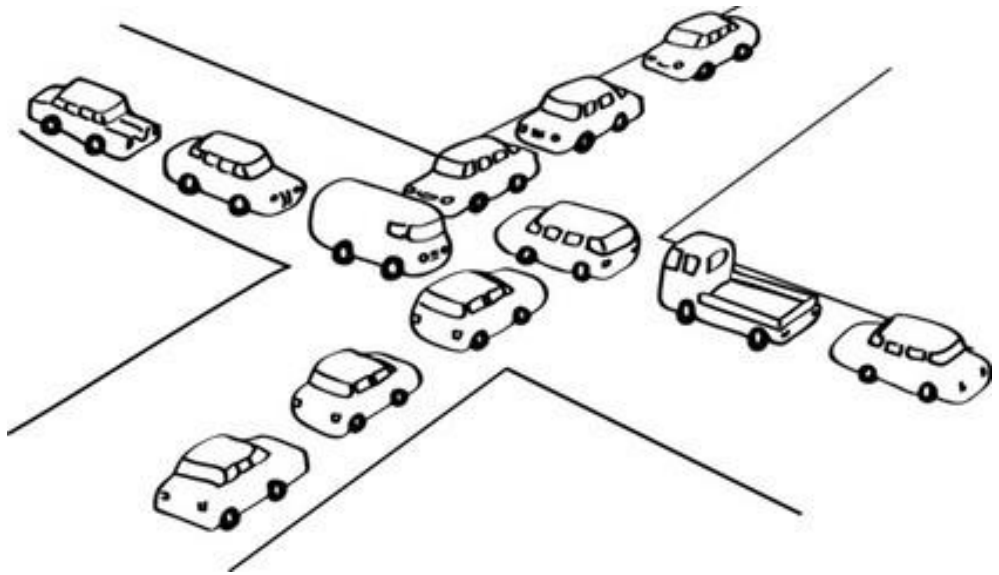
```
Work_one : 220
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Waiting to aquire mutex 2
Releasing mutex 1
Aquiring mutex 1
Aquiring mutex 1
Aquiring mutex 1
Waiting to aquire mutex 2
Aquiring mutex 1
Waiting to aquire mutex 2
Aquiring mutex 1
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Aquired mutex 2
Waiting to aquire mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
```

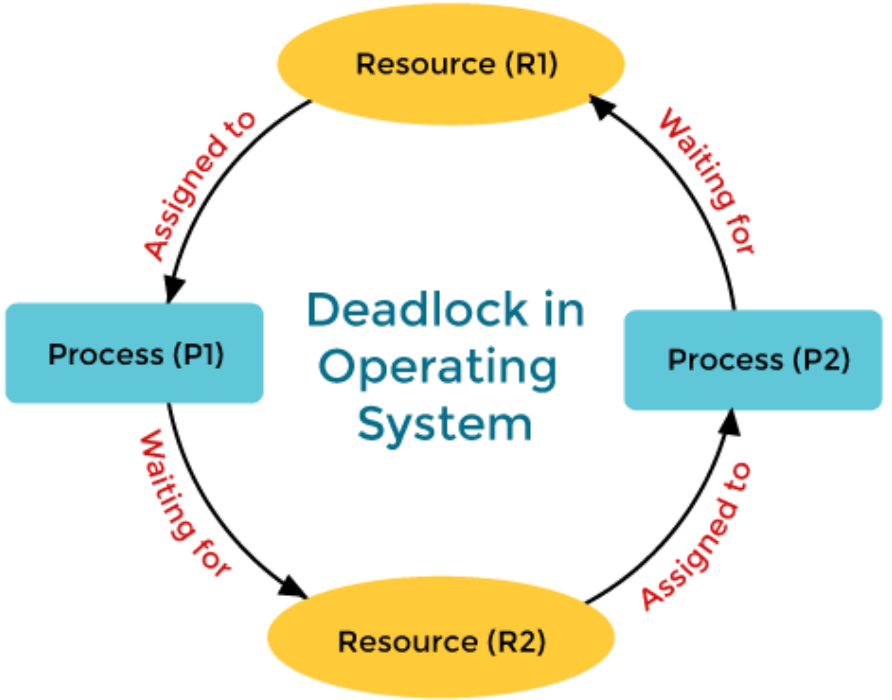
```
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
Waiting to aquire mutex 2
```

…

Lock: Dead lock

- Lock programming
  - Dead lock (교착상태)
    - 두 개 이상의 쓰레드가 절대 발생하지 않을 사건에 대해 기다리는 상태





실습 4: Dead lock

- Dead lock solution

실습 4: Dead lock

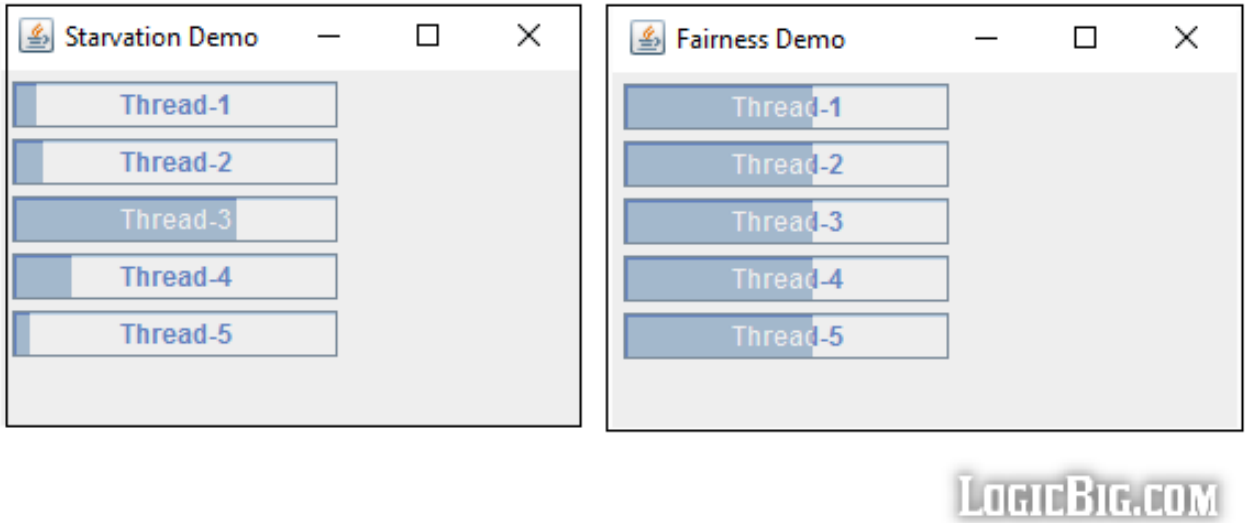
- Dead lock 해결 결과

이 결과 화면이 나오도록 코드 수정

```
root@yejin:/home/yejin/TABA/concurrency/exercise_4# ./prevention.out 1000 4
...
Work_two : 1984
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_two : 1986
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_two : 1988
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_two : 1990
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_two : 1992
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_one : 1994
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_one : 1996
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Aquired mutex 1
Waiting to aquire mutex 2
Aquired mutex 2
Work_one : 1998
Releasing mutex 2
Released mutex 2
Releasing mutex 1
Releasing mutex 1
Complete
```

Lock: Starvation

- Lock programming
  - Starvation



https://www.logicbig.com/tutorials/core-java-tutorial/java-multi-threading/thread-starvation.html

실습 5: Starvation

- thread_starvation.c

```
 1 #include <stdlib.h>
 2 #include <unistd.h>
 3 #include <stdio.h>
 4 #include <assert.h>
 5 #include <pthread.h>
 6
 7 int first_count = 0;
 8 int second_count = 0;
 9 int nthread = 1;
10
11 int max_work_cnt = 1;
12 int work_cnt = 0;
13 int *work_cnt_per_thread;
14
15 pthread_mutex_t first_lock;
16
<n/TABA/concurrency/exercise_5/s/thread_starvation.c\
```

실습 5: Starvation

- thread_starvation.c

```
65 int main(int argc, char *argv[])
66 {
67     pthread_t *th;
68     void *value;
69     long i;
70
71     if (argc < 3) {
72         fprintf(stderr, "%s parameter : nthread, max_work_cnt\n", argv[0]);
73         exit(-1);
74     }
75
76     nthread = atoi(argv[1]);
77
78     max_work_cnt = atoi(argv[2]);
79
80     th = malloc(sizeof(pthread_t) * nthread);
81     work_cnt_per_thread = malloc(sizeof(int) * nthread);
82
83     pthread_mutex_init(&first_lock, NULL); // initialize the lock
84
85     for(i = 0; i < nthread; i++) {
86         assert(pthread_create(&th[i], NULL, work_one, (void*) i) == 0);
87     }
88
89     for(i = 0; i < nthread; i++) {
90         assert(pthread_join(th[i], &value) == 0);
91     }
92
93     for(i = 0; i < nthread; i++) {
94         printf("Thread %ld work_cnt : %d\n", i, work_cnt_per_thread[i]);
95     }
96
97     free(th);
98     free(work_cnt_per_thread);
99
100    printf("Complete\n");
101 }
70:1 [Bot]     /home/yejin/TABA/concurrency/exercise_5/s/thread_starvation.c\
```

실습 5: Starvation

- thread_starvation.c

```
17 static void *work_one(void* num)
18 {
19     int number = (int)num;
20     int answer = 0;
21
22     while(work_cnt < max_work_cnt){
23         pthread_mutex_lock(&first_lock); // lock
24
25         answer = first_count + second_count;
26
27         //printf("Work %d : %d \n", work_cnt, answer);
28
29         first_count++;
30         second_count++;
31
32         work_cnt++;
33         work_cnt_per_thread[number]++;
34
35         pthread_mutex_unlock(&first_lock); // unlock
36         //sleep(1);
37     }
38
39     return NULL;
40 }
41
42 /*
36:18 [ 9%]     /home/yejin/TABA/concurrency/exercise_5/s/thread_starvation.c\
```

실습 5: Starvation

- 실행파일

```
# gcc starvation.c -lpthread -o starvation.out
```

- 결과

```
root@yejin:/home/yejin/TABA/concurrency/exercise_5/s# ./thread_starvation.out 5 1000
Thread 0 work_cnt : 709
Thread 1 work_cnt : 99
Thread 2 work_cnt : 124
Thread 3 work_cnt : 0
Thread 4 work_cnt : 70
Complete
```

실습 5: Starvation

- Starvation solution

실습 5: Starvation

- Starvation 해결 결과

```
root@yejin:/home/yejin/TABA/concurrency/exercise_5/s# ./prevent.out 5 1000
Thread 0 work_cnt : 201
Thread 1 work_cnt : 201
Thread 2 work_cnt : 201
Thread 3 work_cnt : 201
Thread 4 work_cnt : 200
Complete
```

↑위 결과 화면이 나오도록 코드 수정