Hi, Futex!
Hello, TowelRoot!

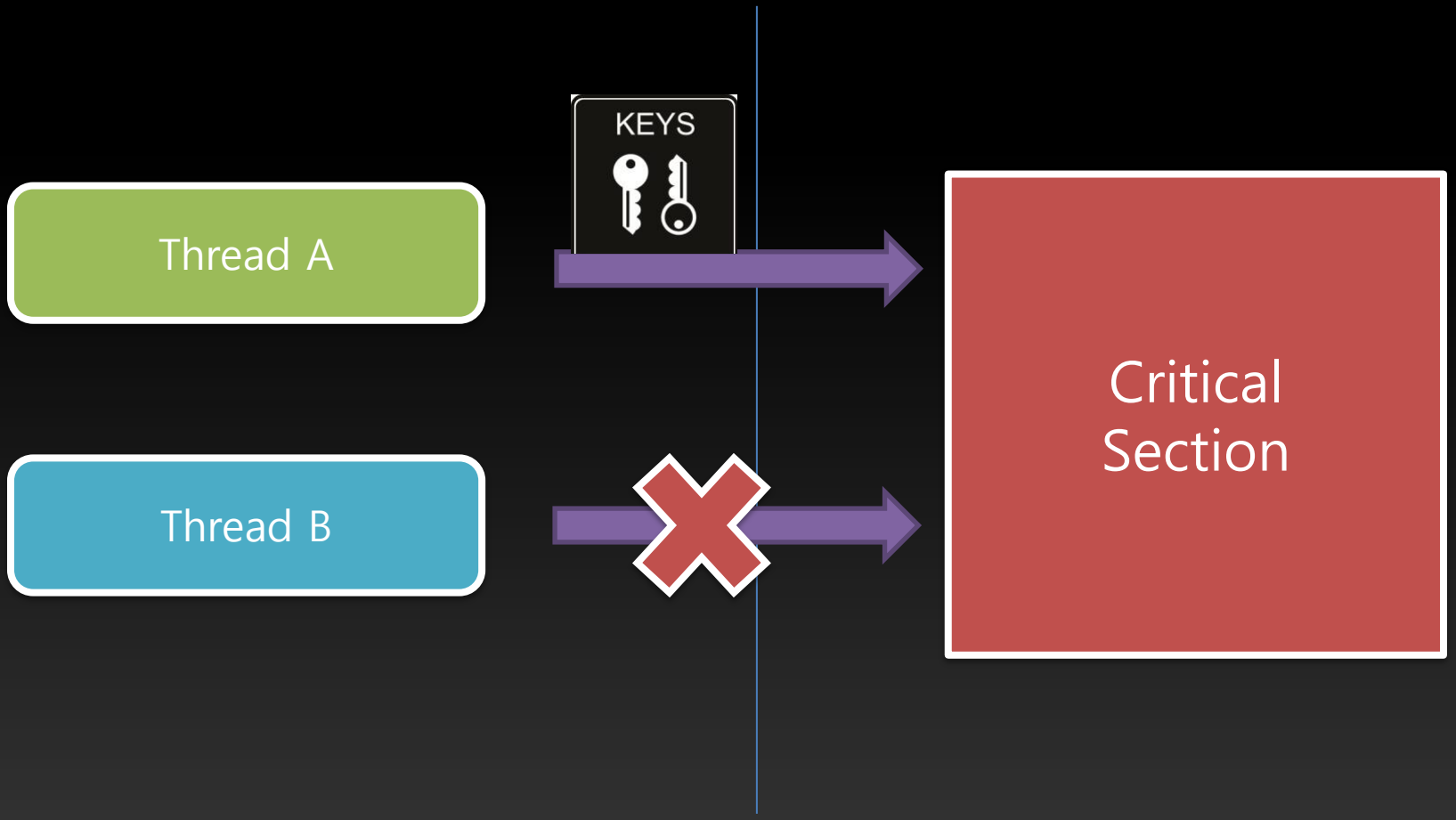
Lee JiHun
wh_pesante@naver.com

Who am I?

- Name : Lee Ji Hun
- NickName : Pesante
- Type: Pwnable, Reversing
- 출몰지역 : 대전, 서울
- Monsterz 잉여 회원
- Argos 잉여 회원

CONTENTS

Mutex



Mutex

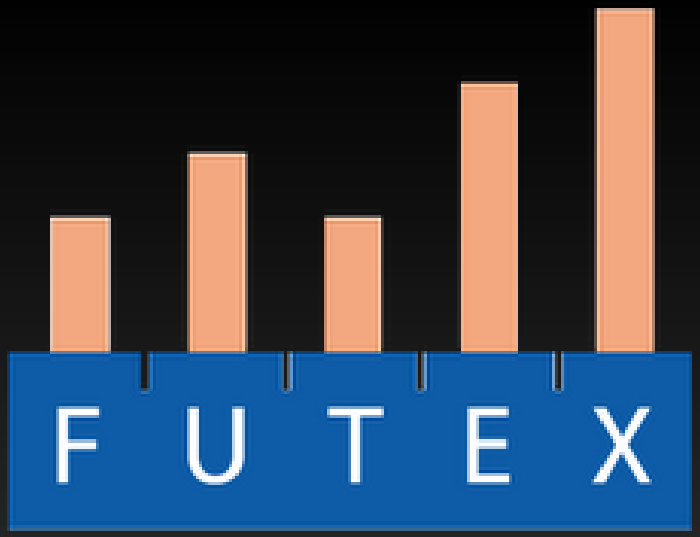
Wait/Release

Condvars

- "시그널 발생"과 "시그널에 대한 기다림" 을 통한 좀더 적극적인 동기화 방법을 제공

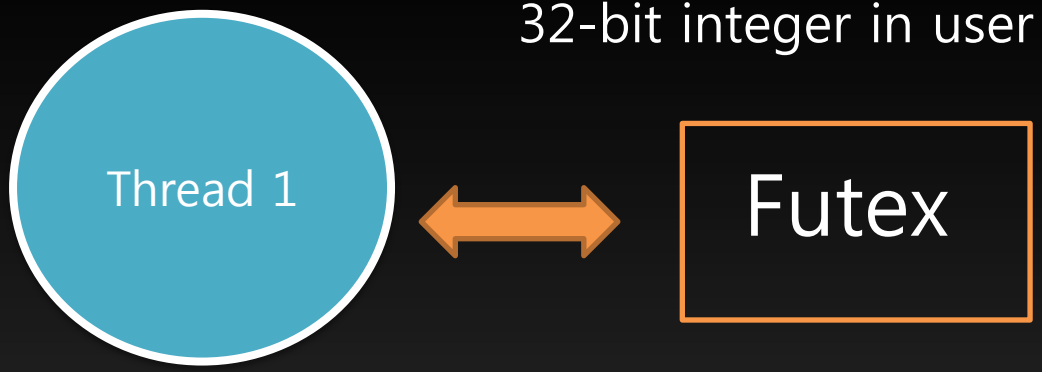
- 이러한 시그널을 기다리기 위해서 pthread_cond_wait() 와 pthread_cond_timedwait() 2개의 함수를 제공

Futex
(Fast Userspace muTEX)



- Linux kernel system call for locking
- Fast user-space mutexes
- 32-bit integer in user memory

Futex
(Fast Userspace muTEX)



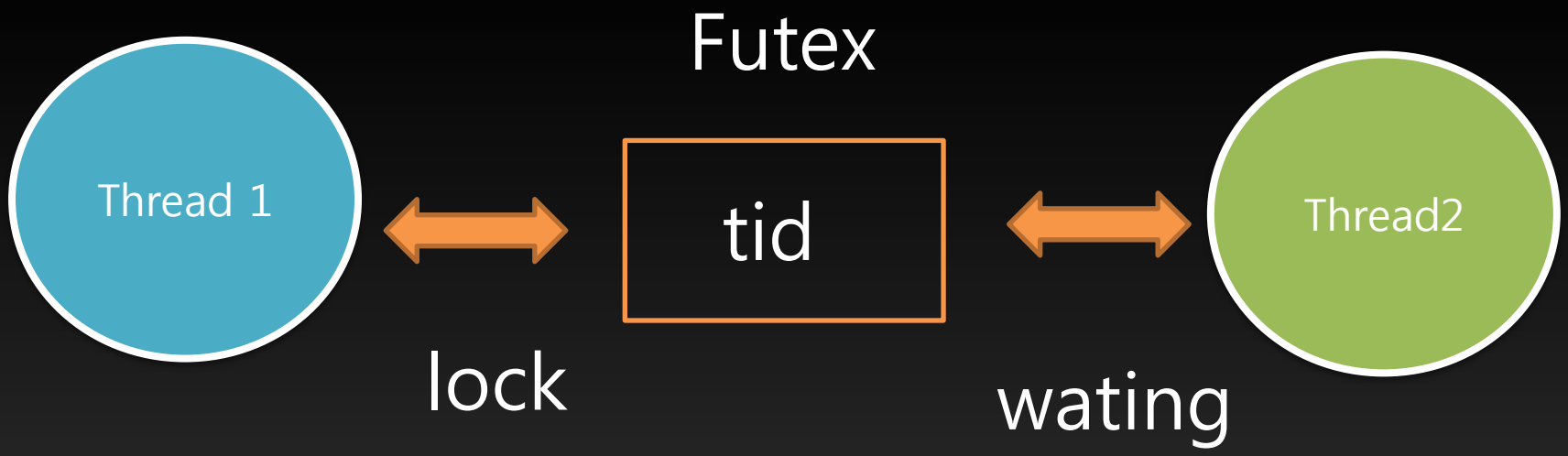
Unlock=0

Lock=tid

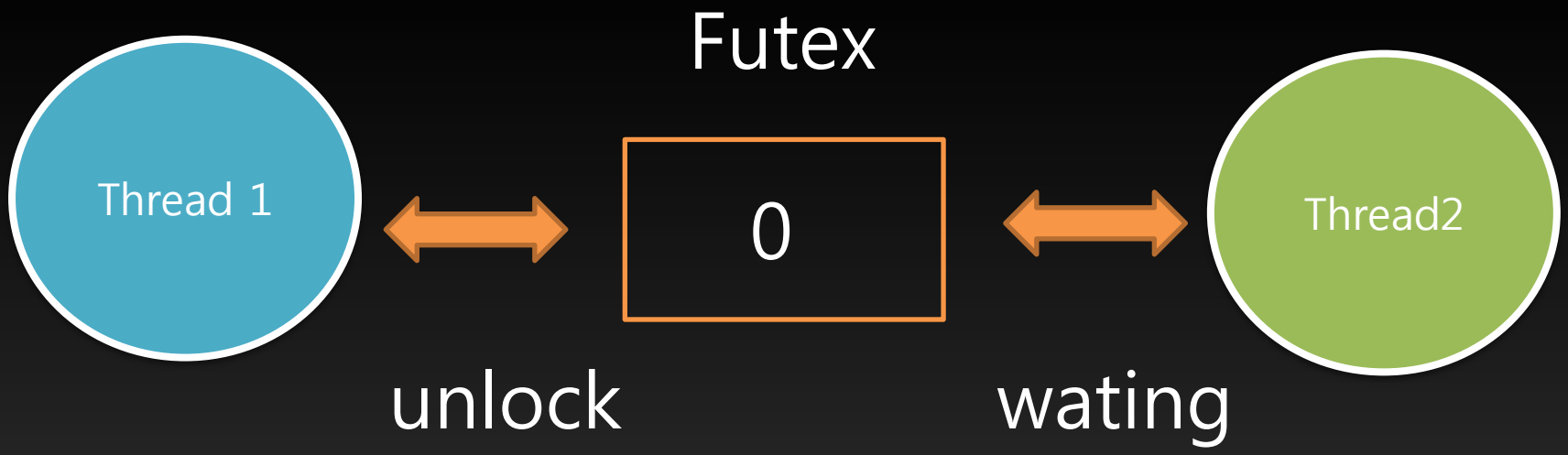
Futex
(Fast Userspace muTEX)

Unlock State

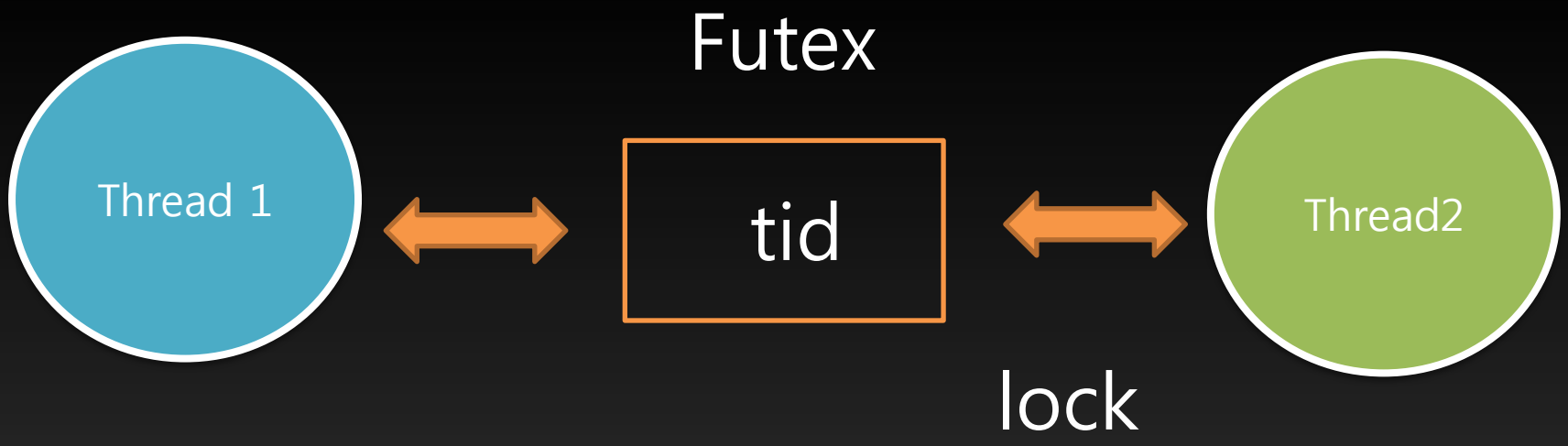
Futex
(Fast Userspace muTEX)



Futex
(Fast Userspace muTEX)



Futex
(Fast Userspace muTEX)



Futex
(Fast Userspace muTEX)

Wait/Release

Futex
(Fast Userspace muTEX)

```
if (*uaddr == 0) {
    *uaddr = tid; /* Lock acquired */
}
```

Futex
(Fast Userspace muTEX)

MONSTERZ

장점

1. 빠르다.
2. User space에 있으므로 공유 메모리에 올려 쓰레드 뿐만 아니라 프로세스끼리도 손쉽게 사용이 가능

단점

1. 멀티코어에서는 그다지 성능을 발휘하지 못함
2. Thread 수가 늘어나면 성능이 급격히 감소
3. 인터럽트를 지원하지 않음

Futex Syscall

int sys_futex(int *uaddr, int op, int val, const struct timespec *timeout, int *uaddr2, int val3);

- Action depends on the op arguments
- The arguments can be unused, or cast to differe nt types

FUTEX_WAIT & FUTEX_WAKE

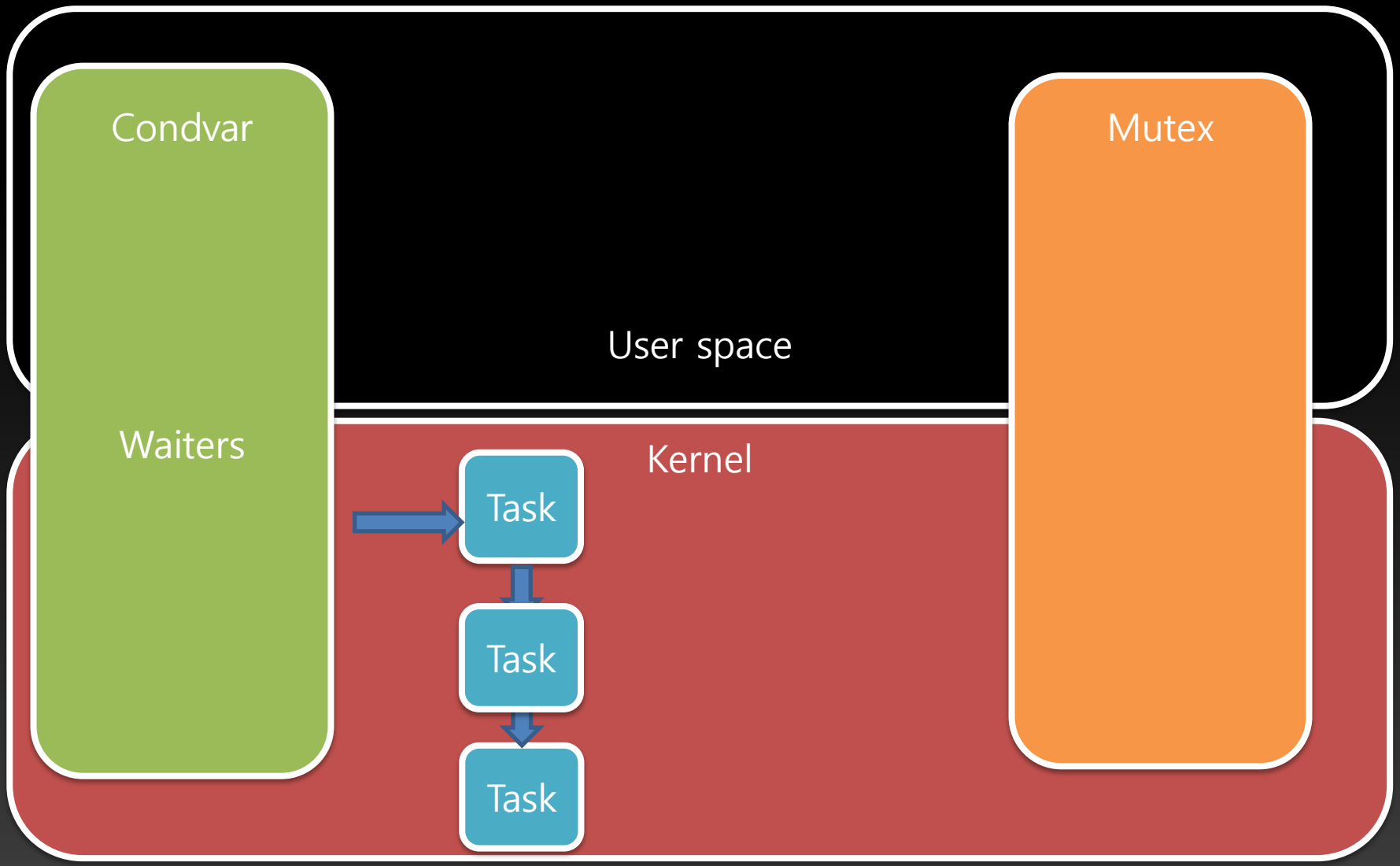
FUTEX_WAIT

This operation causes the thread to be suspended in the kernel until notified

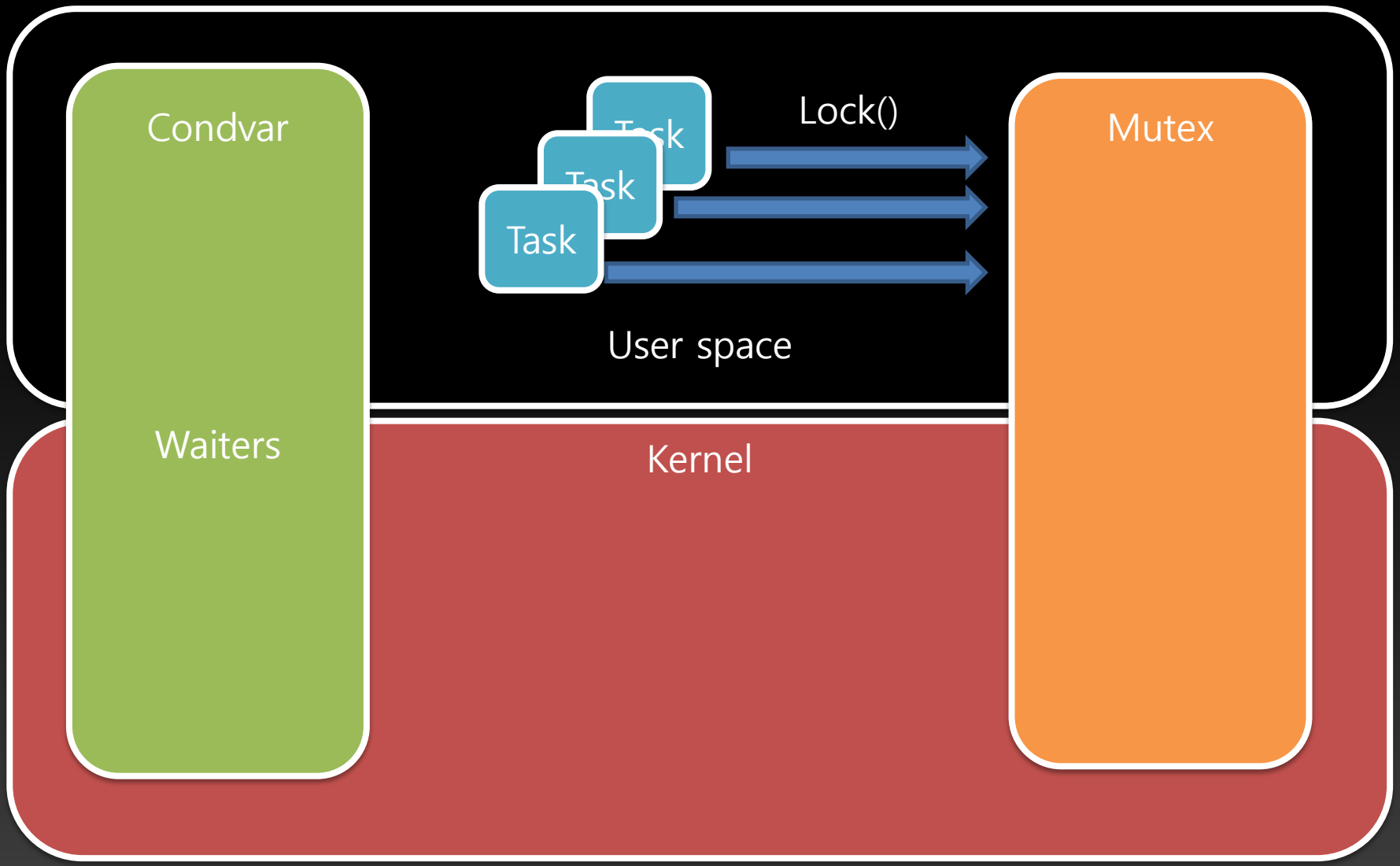
FUTEX_WAKE

To wake up one or more threads waiting on a futex this operation can be used.

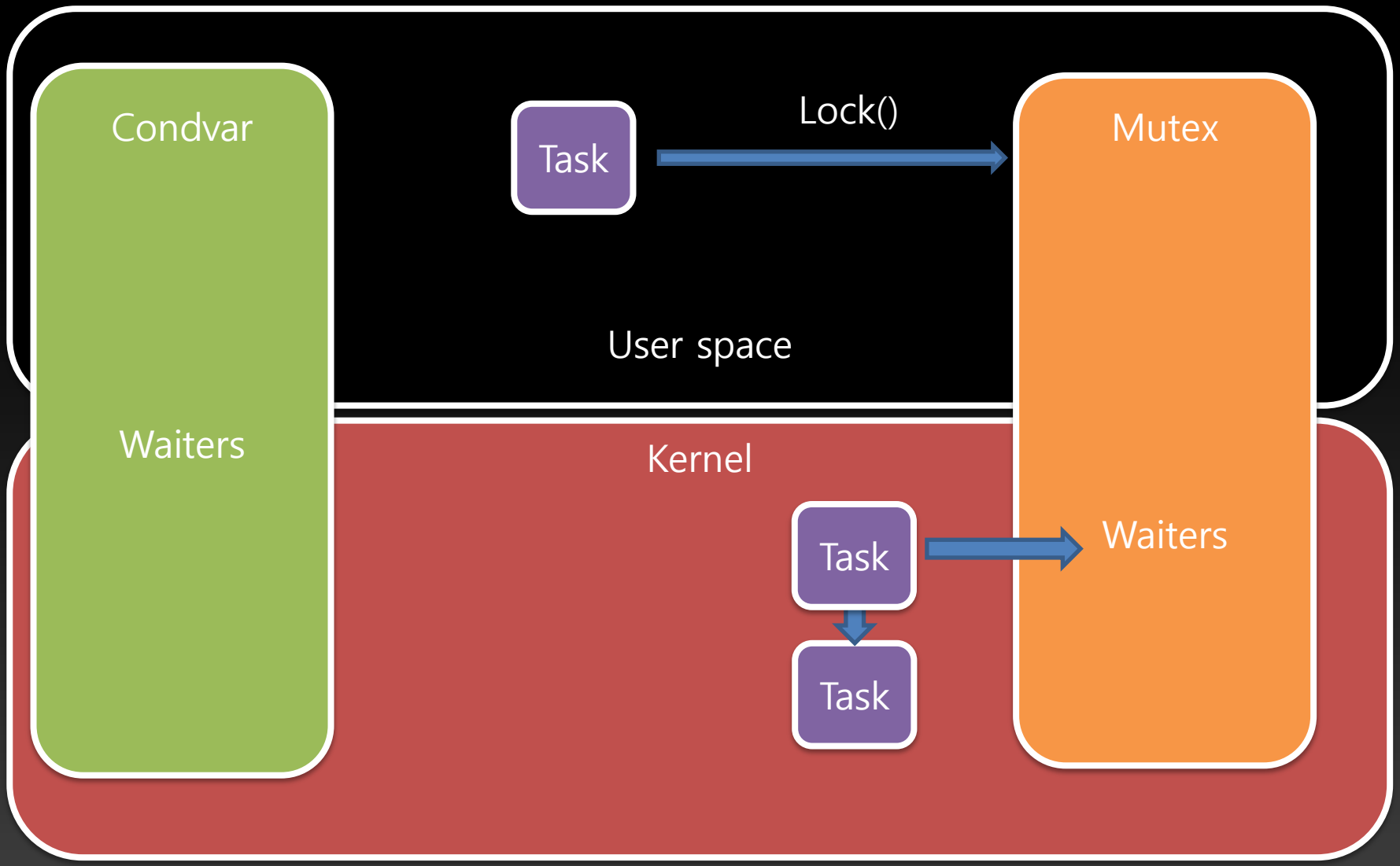
FUTEX_REQUEUE



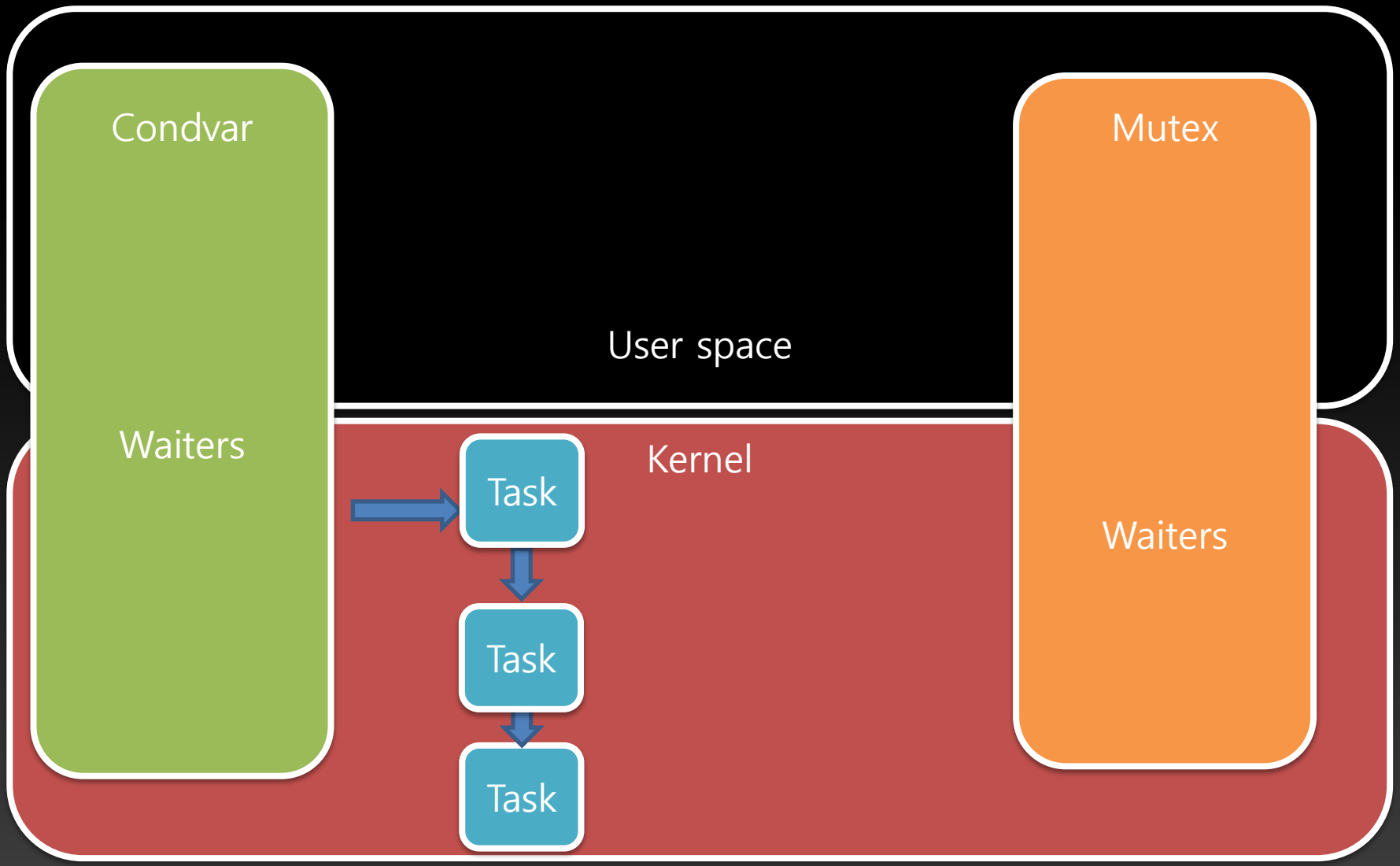
FUTEX_REQUEUE



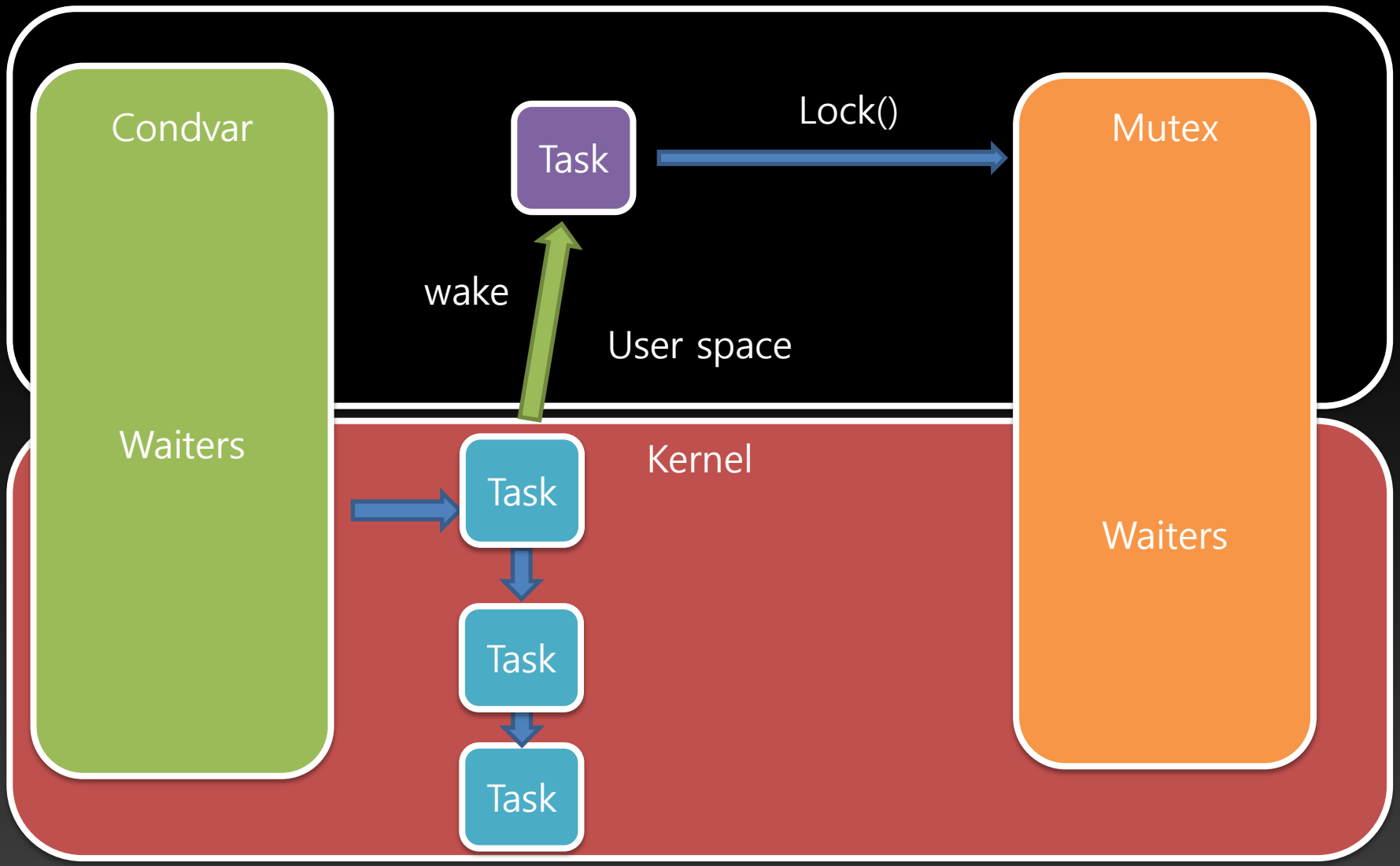
FUTEX_REQUEUE



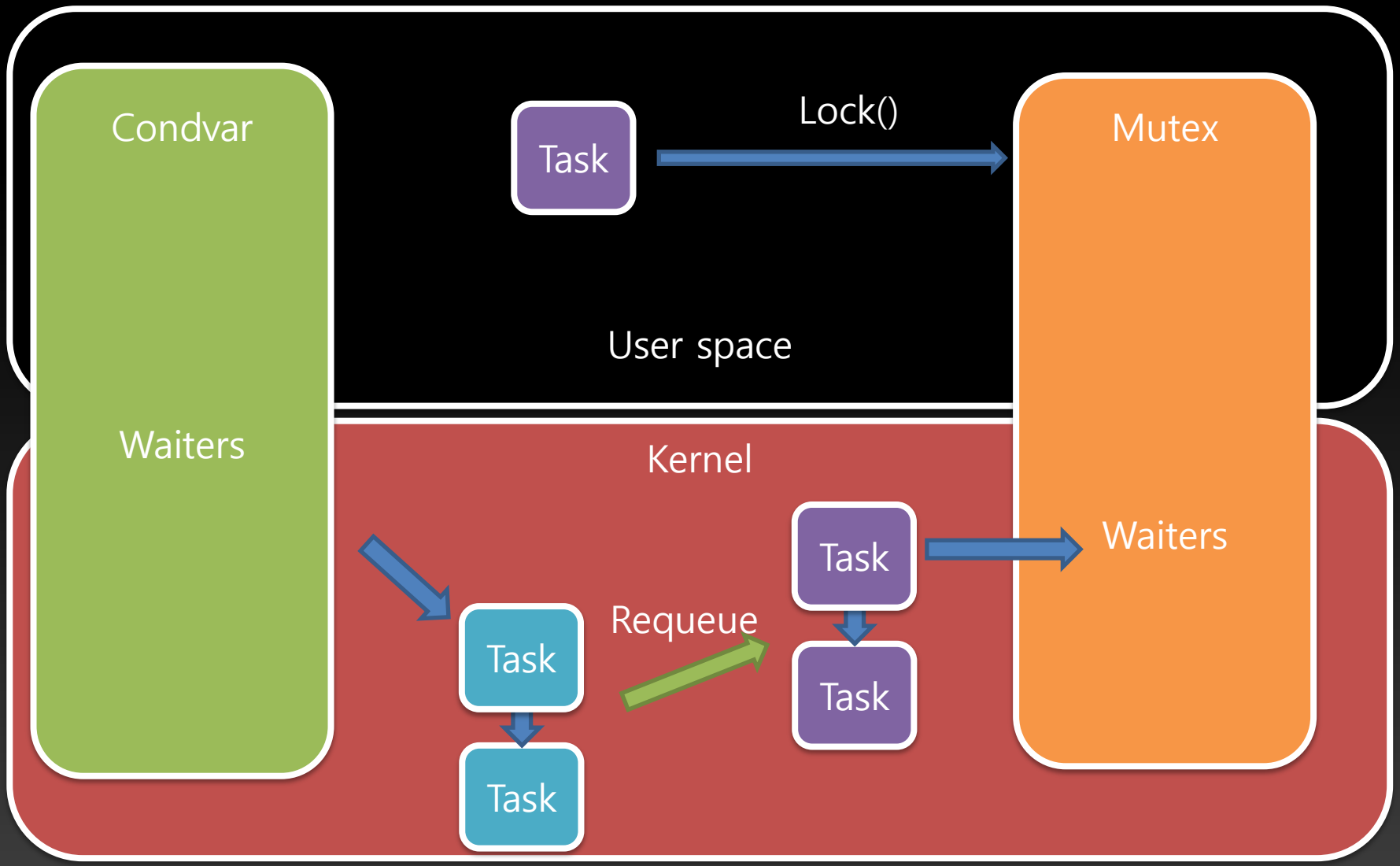
FUTEX_REQUEUE



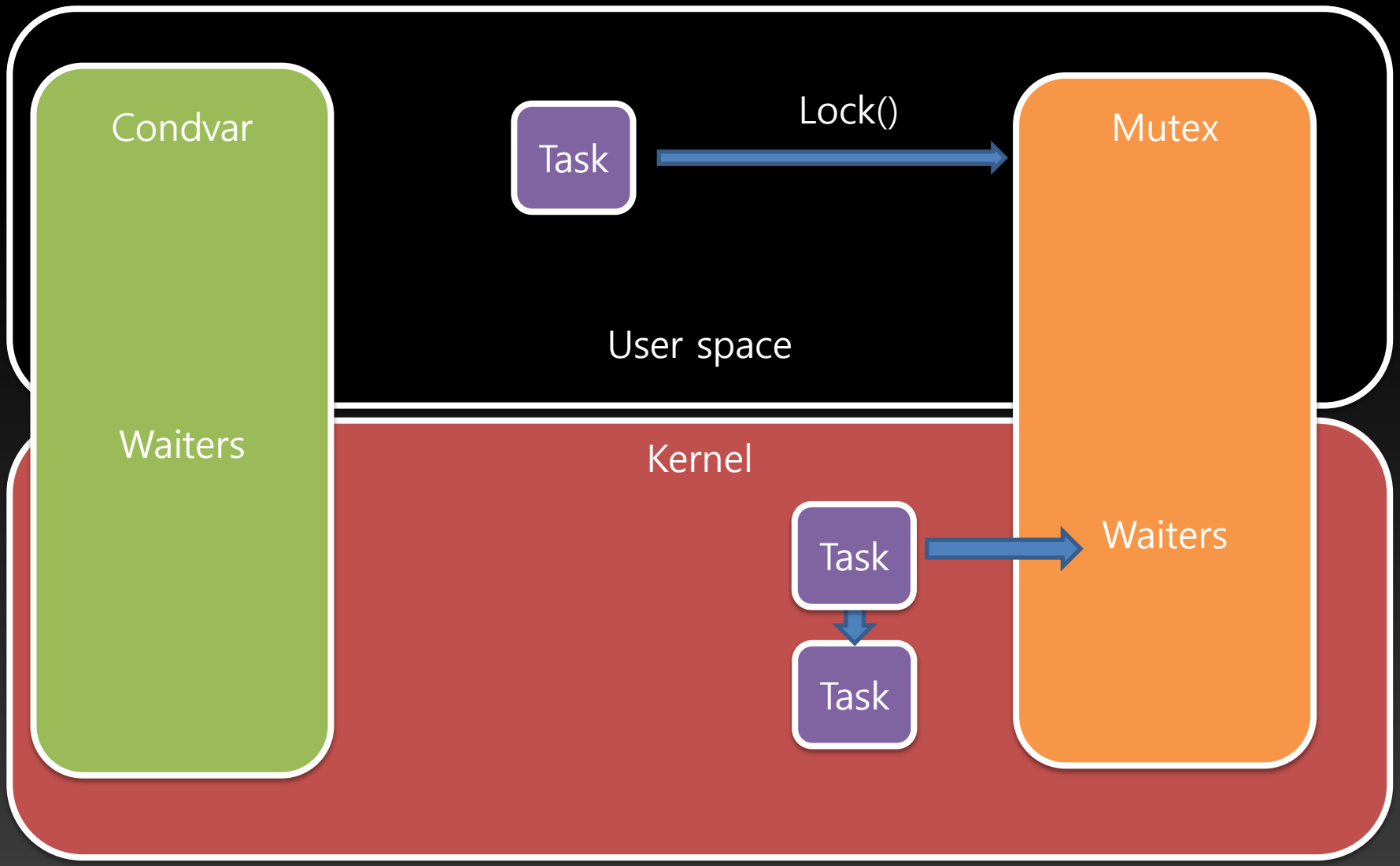
FUTEX_REQUEUE



FUTEX_REQUEUE



FUTEX_REQUEUE



Priority

화장실

아, 화장실

Priority

ㅇㅋ

먼저 쓰세요

Priority

Um...

Wating...

Priority Inversion

1등

2등

3등

Priority Inversion

아, 화장실

Priority Inversion

음...

화장실

Priority Inversion

쓰는 중이에요

Ok.

Priority Inversion

CPU좀 줘봐. 다른 거 할 거 있어.

Priority Inversion

일하는중..

Wating..

Wating..

Priority Inversion

일 끝!

Wating..

Wating..

Priority Inversion

Wating..

일 끝!

Priority Inversion

일 끝

Priority Inversion

1등

2등

3등

우선순위가 가장 높은 사람이 가장 늦게 처리됨

Priority Inheritance

쓰는 중이에요

그러면 나의 우선순위를 받아라!

Priority Inheritance

CPU좀 줘봐. 다른거 할 거 있어.

안돼요.

Priority Inheritance

Priority Inheritance

Priority Inheritance

일 끝

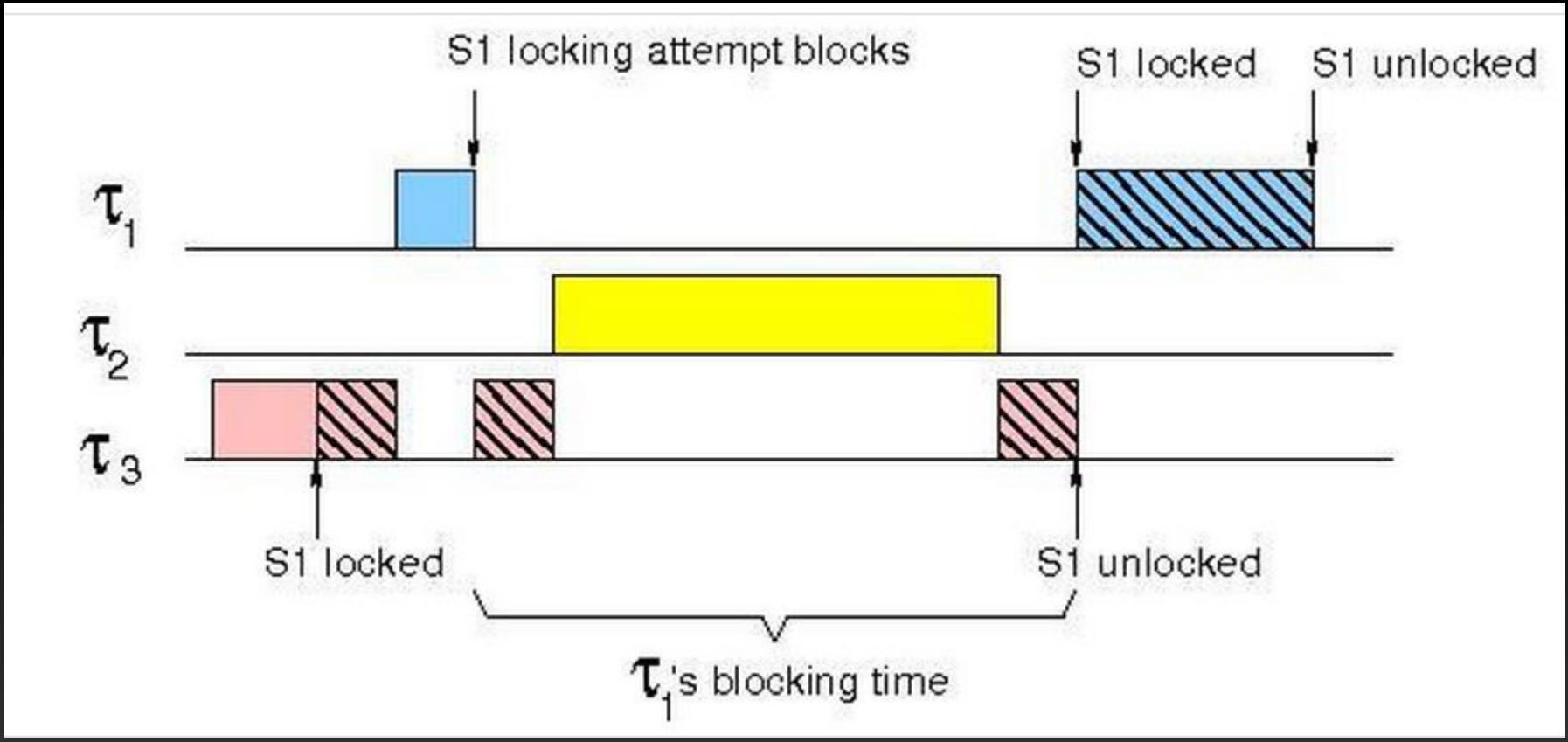
Priority Inheritance

1등

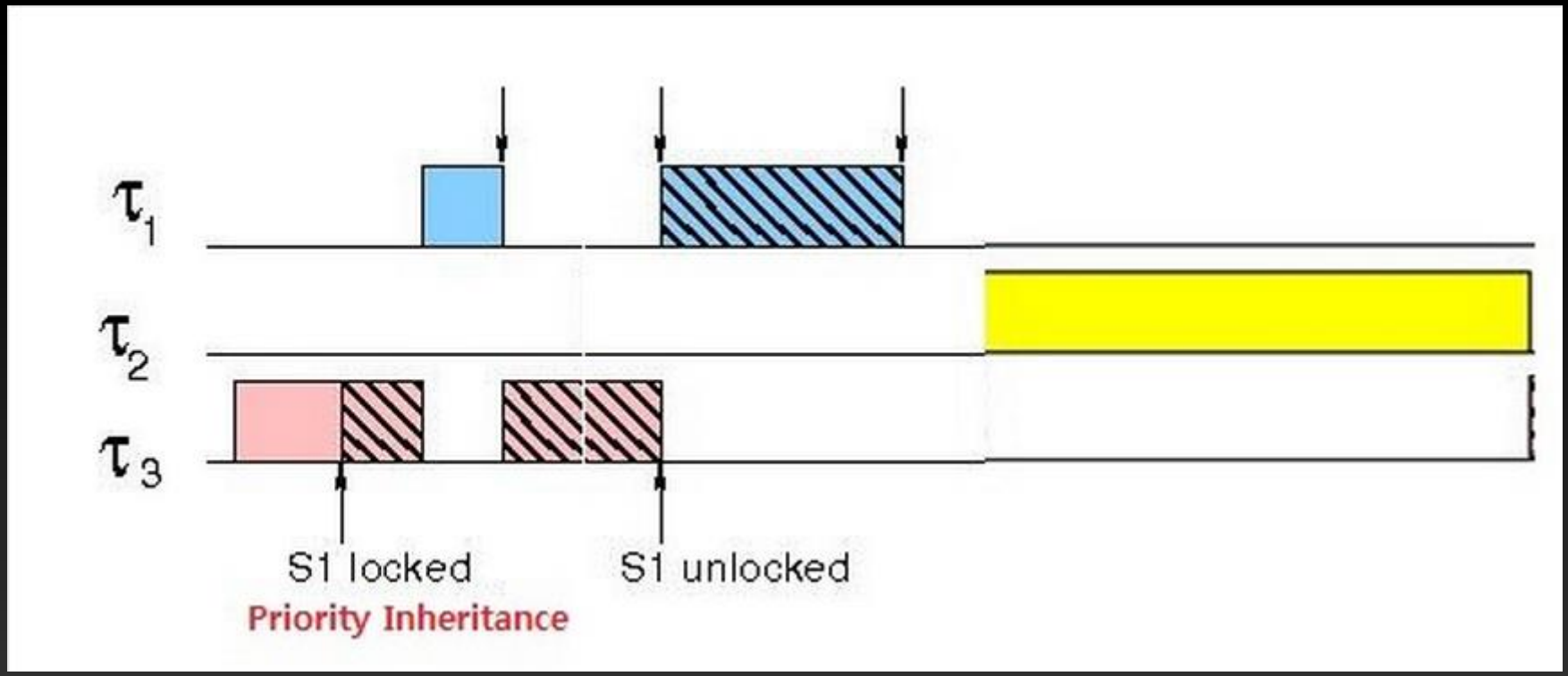
2등

3등

Priority Inheritance



Priority Inheritance



PI futexes

- The user-space futex value is zero for unlocked, or holds the thread ID of the owner

- The FUTEX_LOCK_PI and FUTEX_UNLOCK_PI calls are used instead of wait and wake

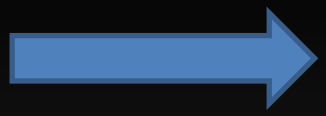
- Unlocking a PI-futex wakes only the highest priority waiter

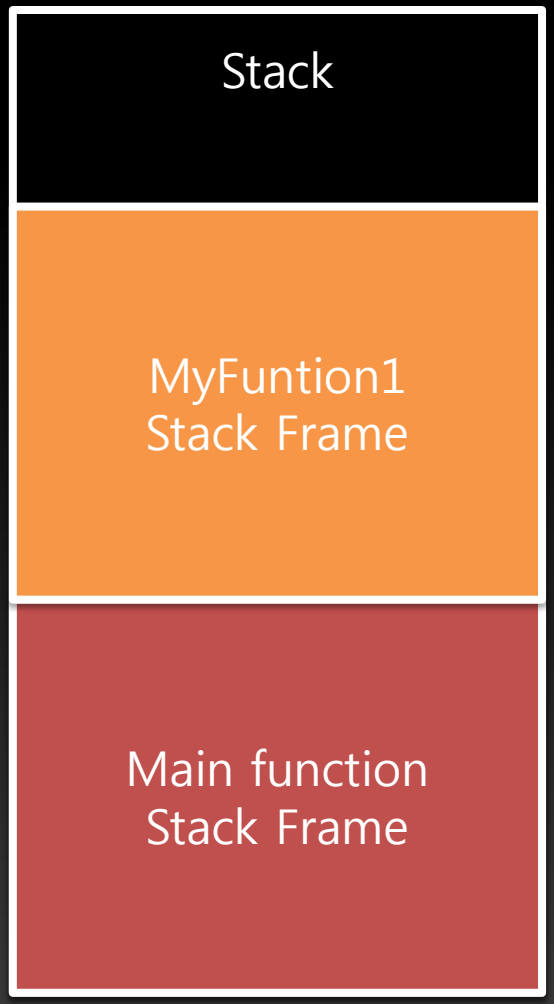
Simpler bug

```
#include <stdio.h>

int main()
{
        ....
        MyFunction()
        MyFunction2()
        ....
}
```

Simpler bug

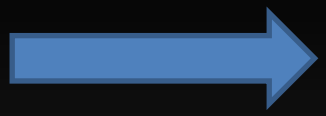
AA라는 지역변수 사용

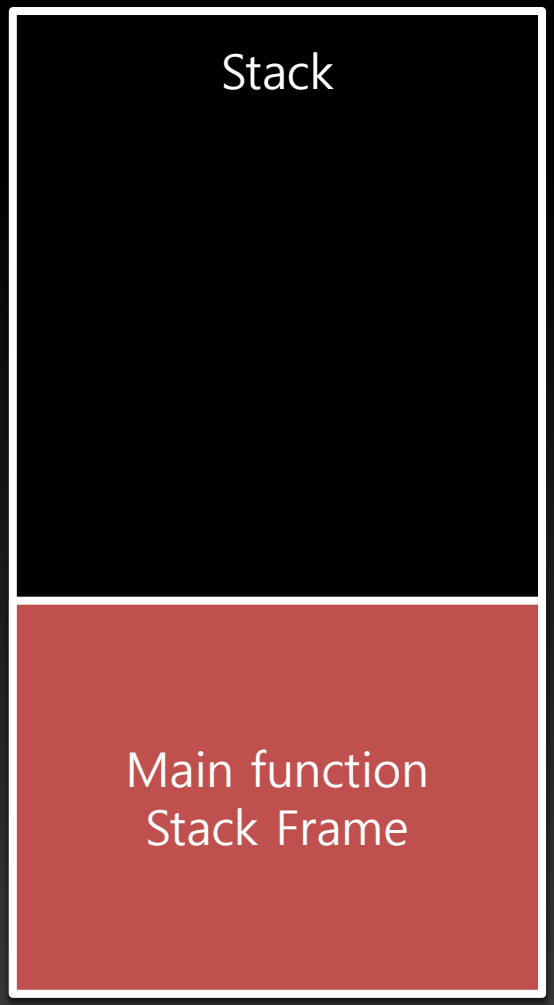




Simpler bug

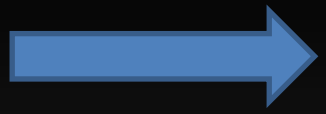
AA의 값은
그대로 남아있음

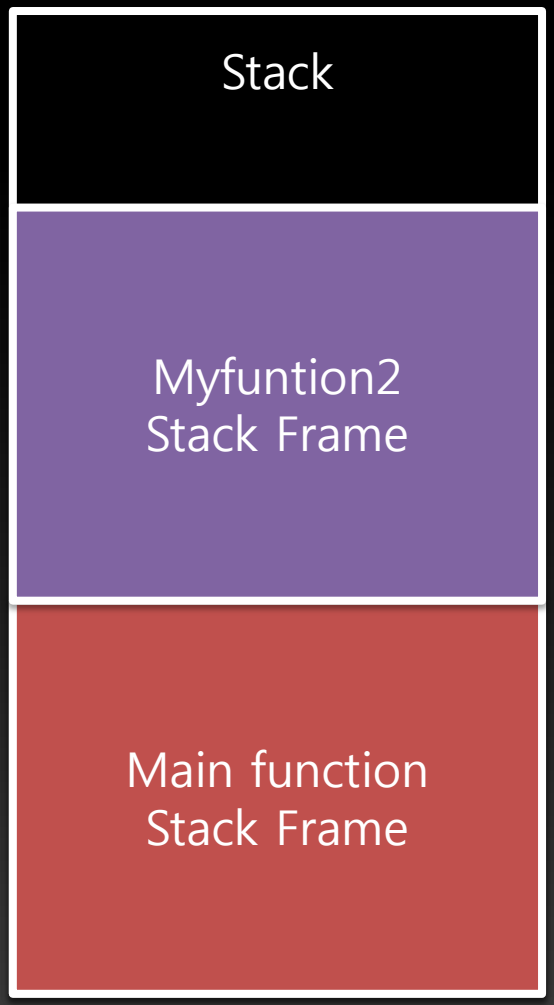




Simpler bug

AA의 값은
그대로 남아있음

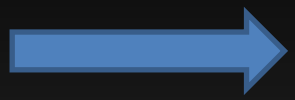




Simpler bug

```c
#include <stdio.h>

int MyFunction1()
{
    int a=3;
    printf("%d\n", a);
}

int MyFunction2()
{
    int b;
    printf("%d\n", b);
}
int main()
{
    MyFunction1();
    MyFunction2();
}
```

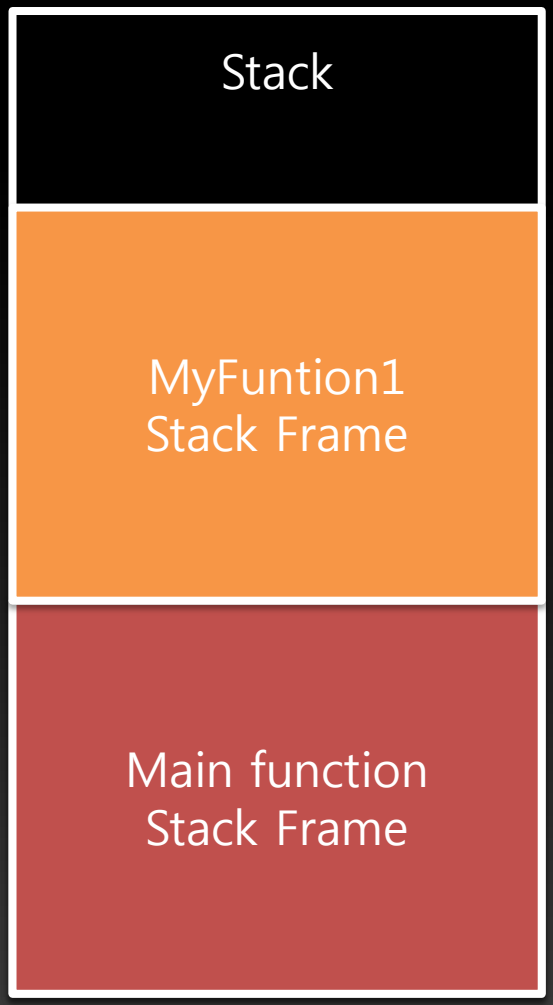
결과



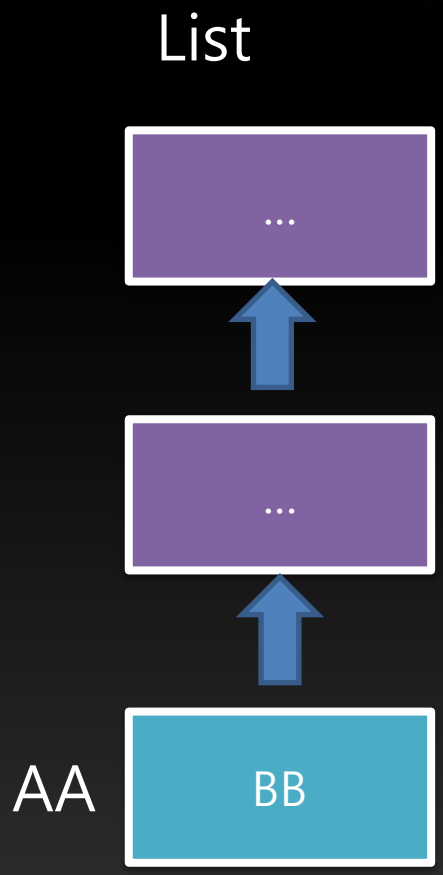
```
pesante@edu:~$ ./monster
3
3
```

Simpler bug

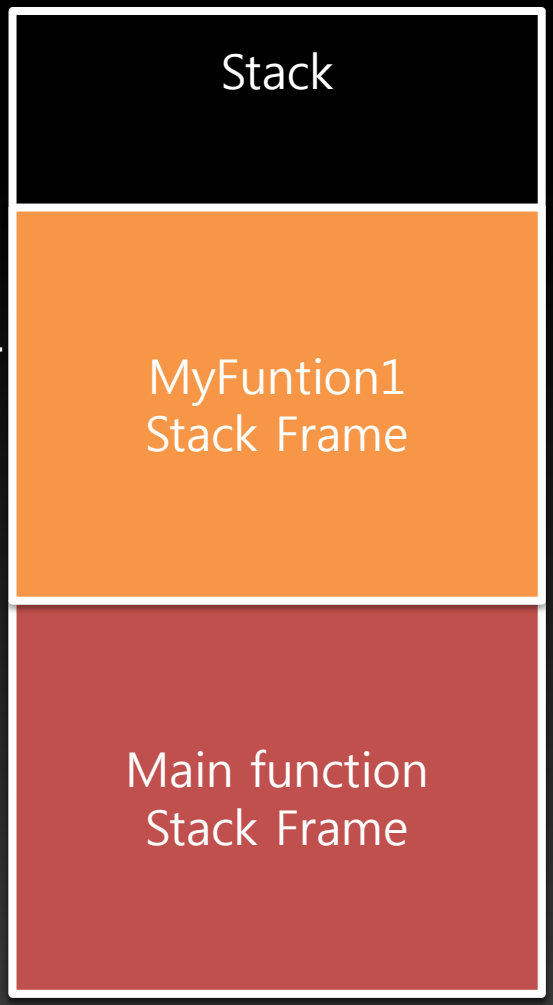
AA라는 지역변수 사용



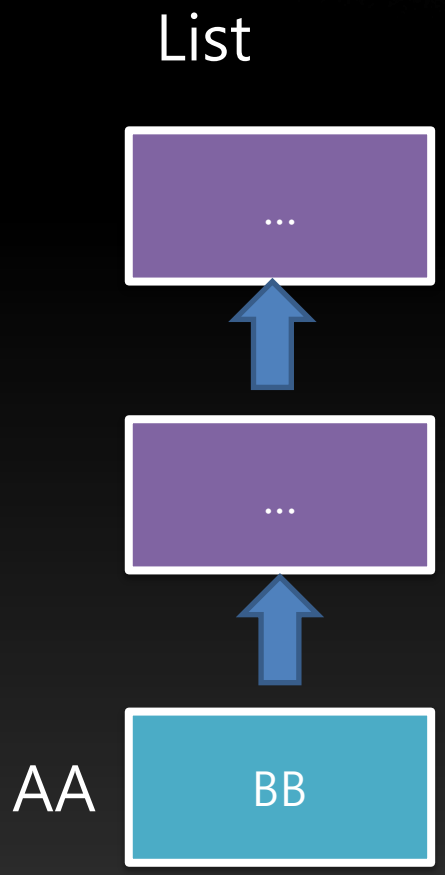
Simpler bug



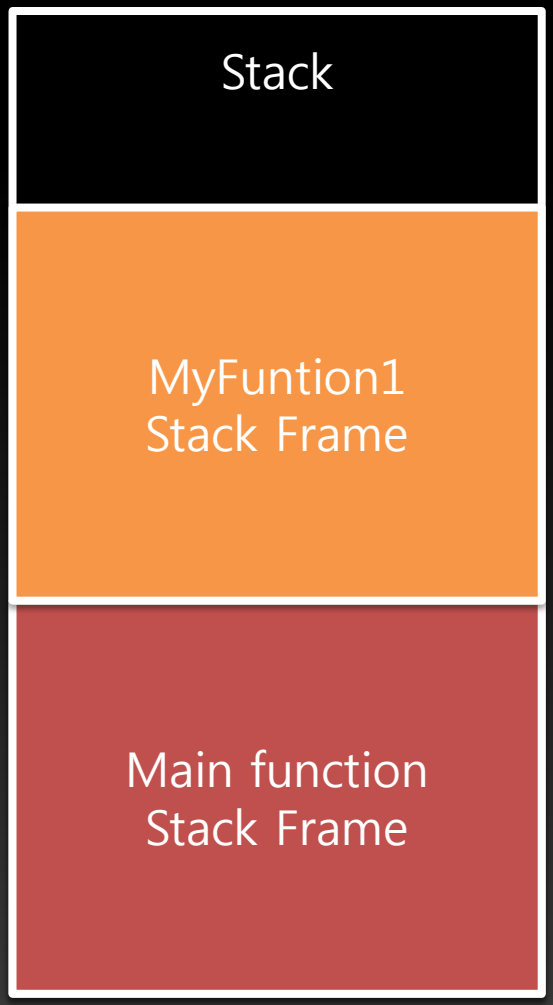
1. AA라는 지역변수 사용
2. 전역변수 List에 등록



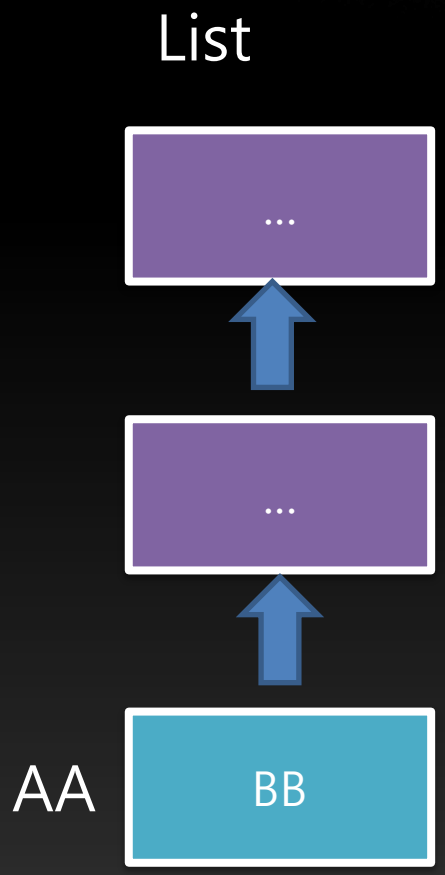
Simpler bug



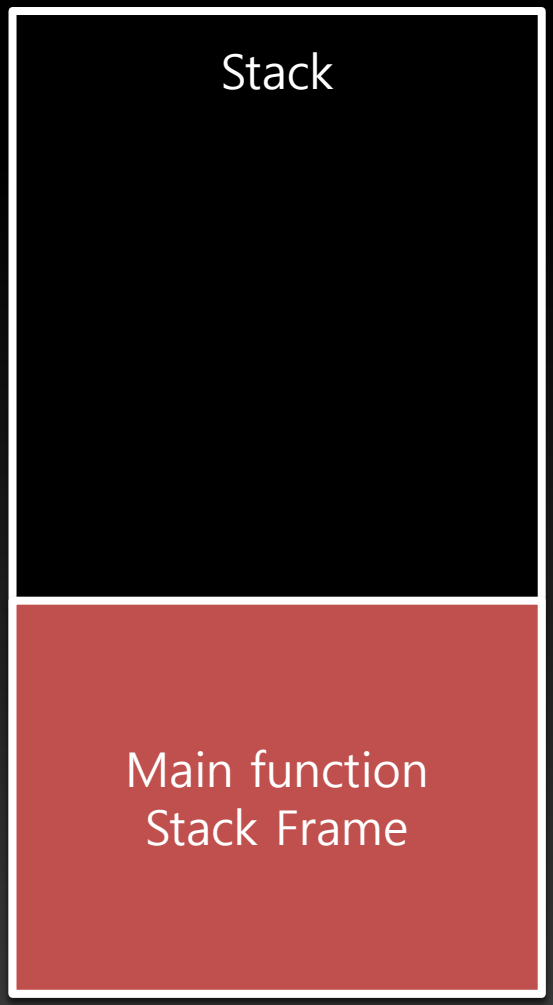
리스트를 해제하지 않고
MyFunction1 종료



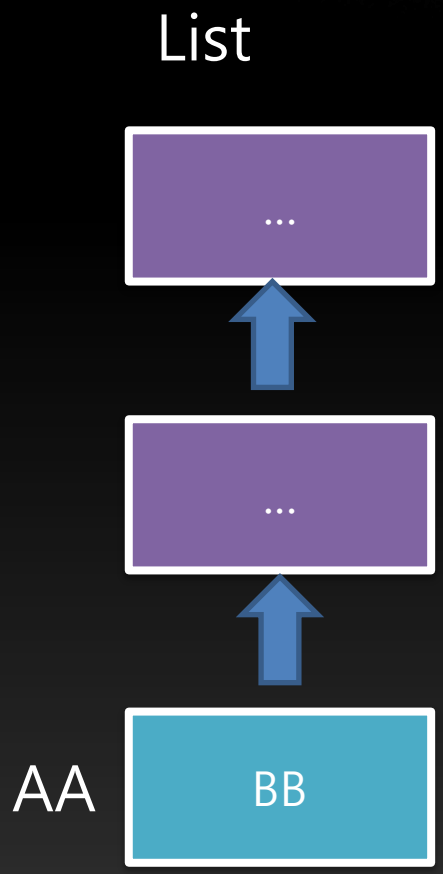
Simpler bug



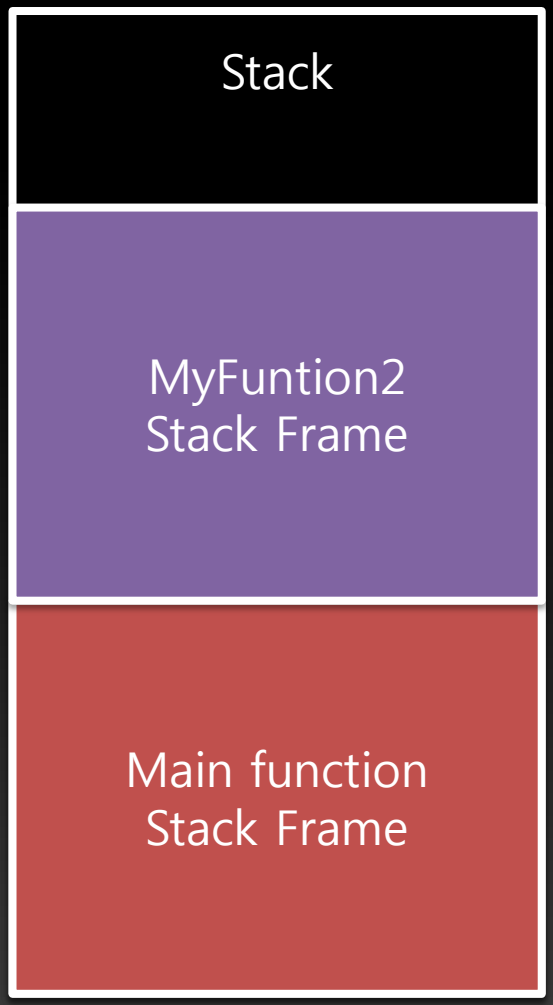
리스트를 해제하지 않고
MyFunction1 종료



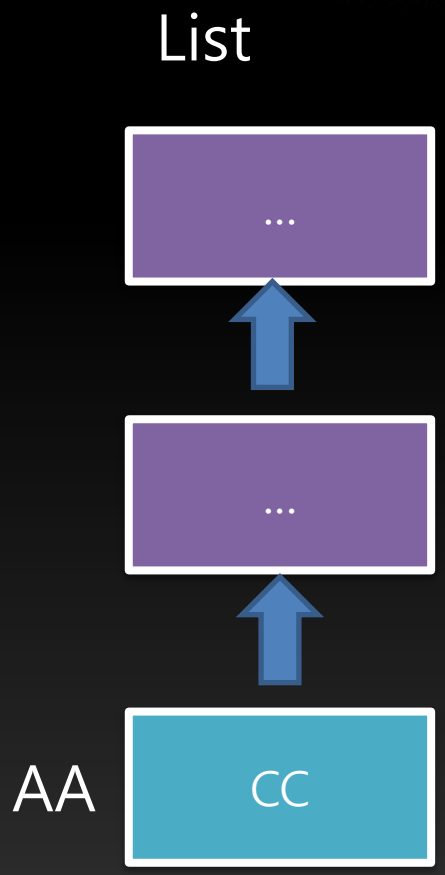
Simpler bug



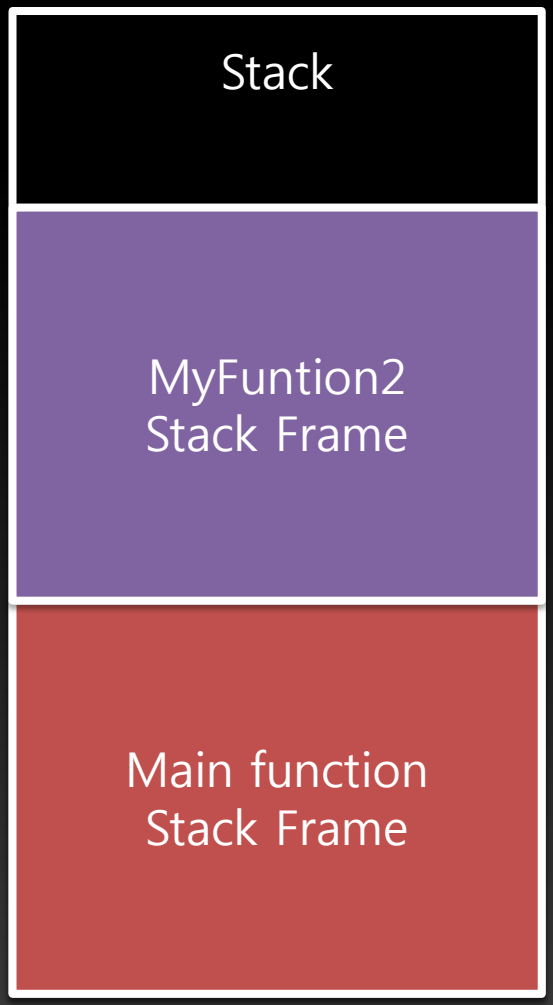
로컬 변수 사용



Simpler bug



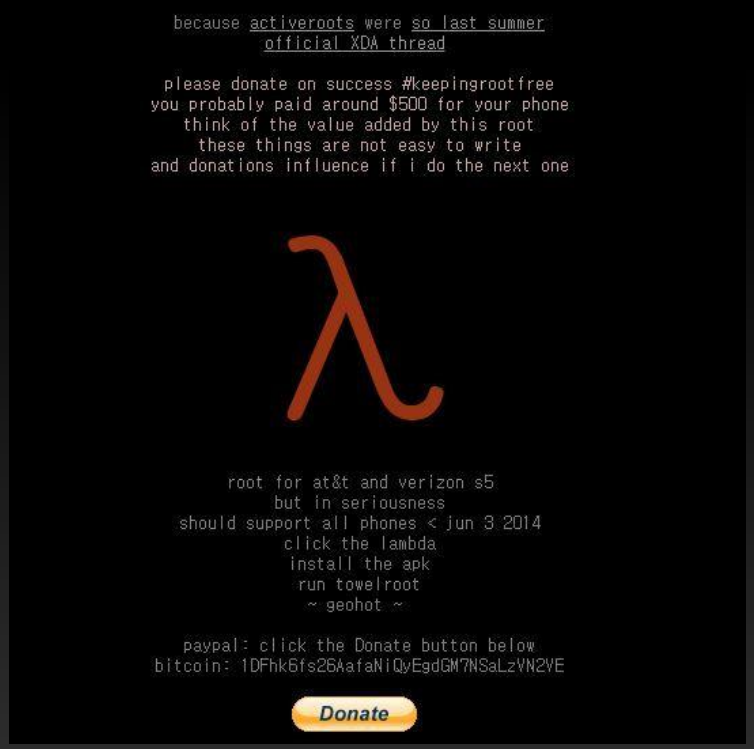
로컬 변수 사용



Simpler bug

1. MyFunction1 내의 지역변수를 리스트에 추가
2. 지역변수(AA)를 리스트에서 해제하지 않고 함수 리턴
3. 다른 함수 내에서 지역변수(BB)를 사용
4. 리스트에 있던 AA의 값이 BB의 값으로 바뀜

Towelroot



- 안드로이드 루팅 어플
- Geohot에 의해 개발
- 커널 취약점을 이용하여 손쉽게 루팅
- CVE-2014-3153

CVE-2014-3153

- The vulnerability is based on two logical bugs in the Linux kernel futex mechanism

- one which allows releasing a pi-lock from userspace

- another which allows requeuing a lock twice.

취약 버전: http://www.securityfocus.com/bid/67906

CVE-2014-3153

```
static int futex_wait_requeue_pi(.....)
{
    struct hrtimer_sleeper timeout, *to = NULL;
      struct rt_mutex_waiter rt_waiter;
    struct rt_mutex *pi_mutex = NULL;
    struct futex_hash_bucket *hb;
    union futex_key key2 = FUTEX_KEY_INIT;
    struct futex_q q = futex_q_init;
    int res, ret;
    //...
    q.rt_waiter = &rt_waiter;
    //...
    //futex_wait_queue_me(hb, &q, to);
    if (!q.rt_waiter) {
    //...
```

rt_mutex_waiter

```
struct rt_mutex_waiter {
        struct rb_node          tree_entry;
        struct rb_node          pi_tree_entry;
        struct task_struct      *task;
        struct rt_mutex          *lock;
        .......
        int prio;
 };
```

CVE-2014-3153

```
static int futex_wait_requeue_pi(.............)
{
        ...................
        if (!q.rt_waiter) {

                if (q.pi_state && (q.pi_state->owner != current)) {
                                .........
                        }
        } else {

                ..........
                ret = rt_mutex_finish_proxy_lock(pi_mutex, to, &rt_waiter, 1);
                debug_rt_mutex_free_waiter(&rt_waiter);

                ........
        }
}
```

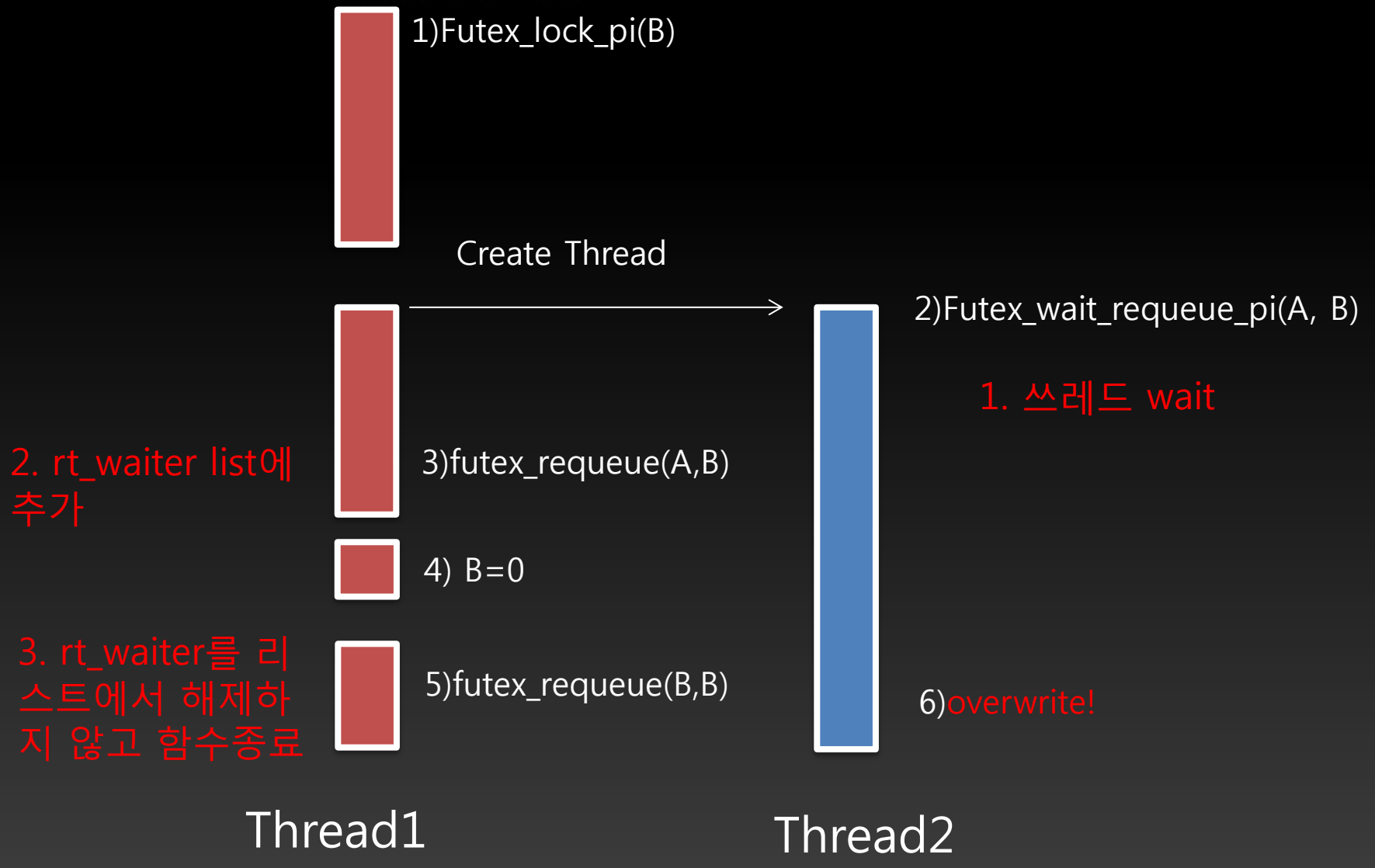
CVE-2014-3153

```
void requeue_pi_wake_futex(struct futex_q *q, union futex_key *key,
                           struct futex_hash_bucket *hb)
{
        get_futex_key_refs(key);
        q->key = *key;

        __unqueue_futex(q);

        WARN_ON(!q->rt_waiter);
        q->rt_waiter = NULL;

        q->lock_ptr = &hb->lock;

        wake_up_state(q->task, TASK_NORMAL);
}
```

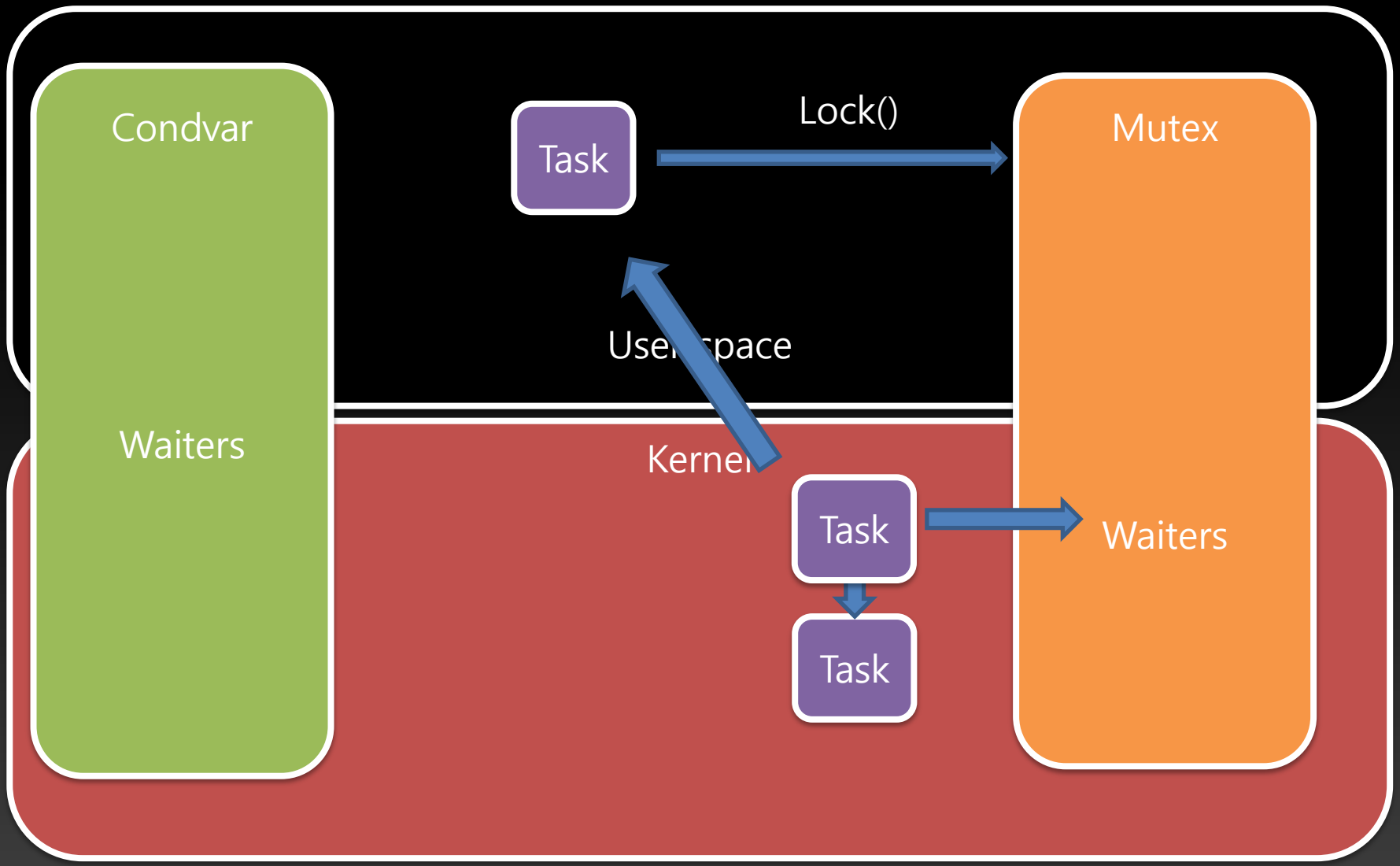
CVE-2014-3153

Thread B: futex_lock_pi(&futex2)
Thread A: futex_wait_requeue_pi(&futex1, &futex2)
Thread B: futex_cmp_requeue_pi(&futex1, &futex2)
Thread B: futex2 = 0
Thread B: futex_cmp_requeue_pi(&futex2, &futex2)

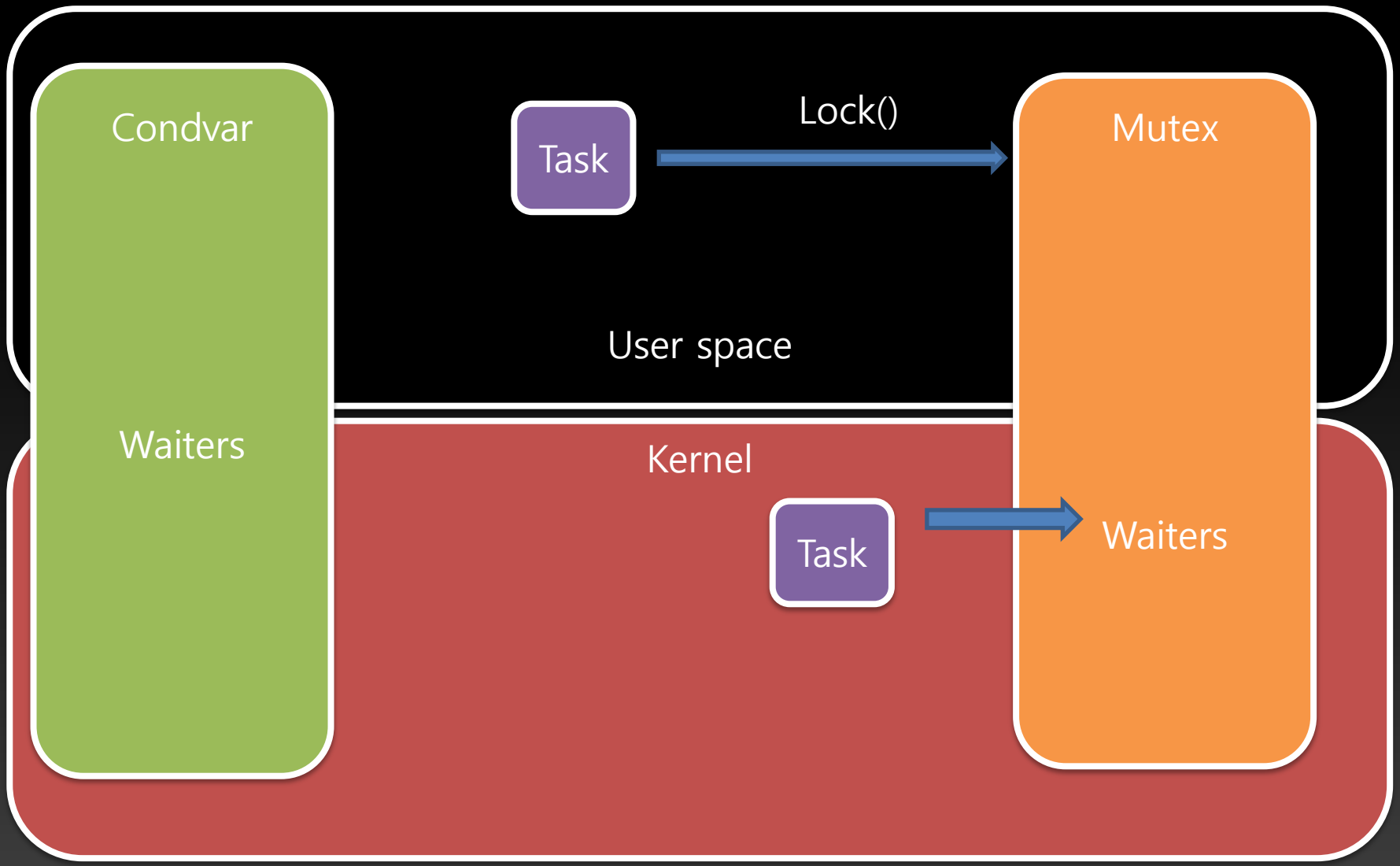
CVE-2014-3153



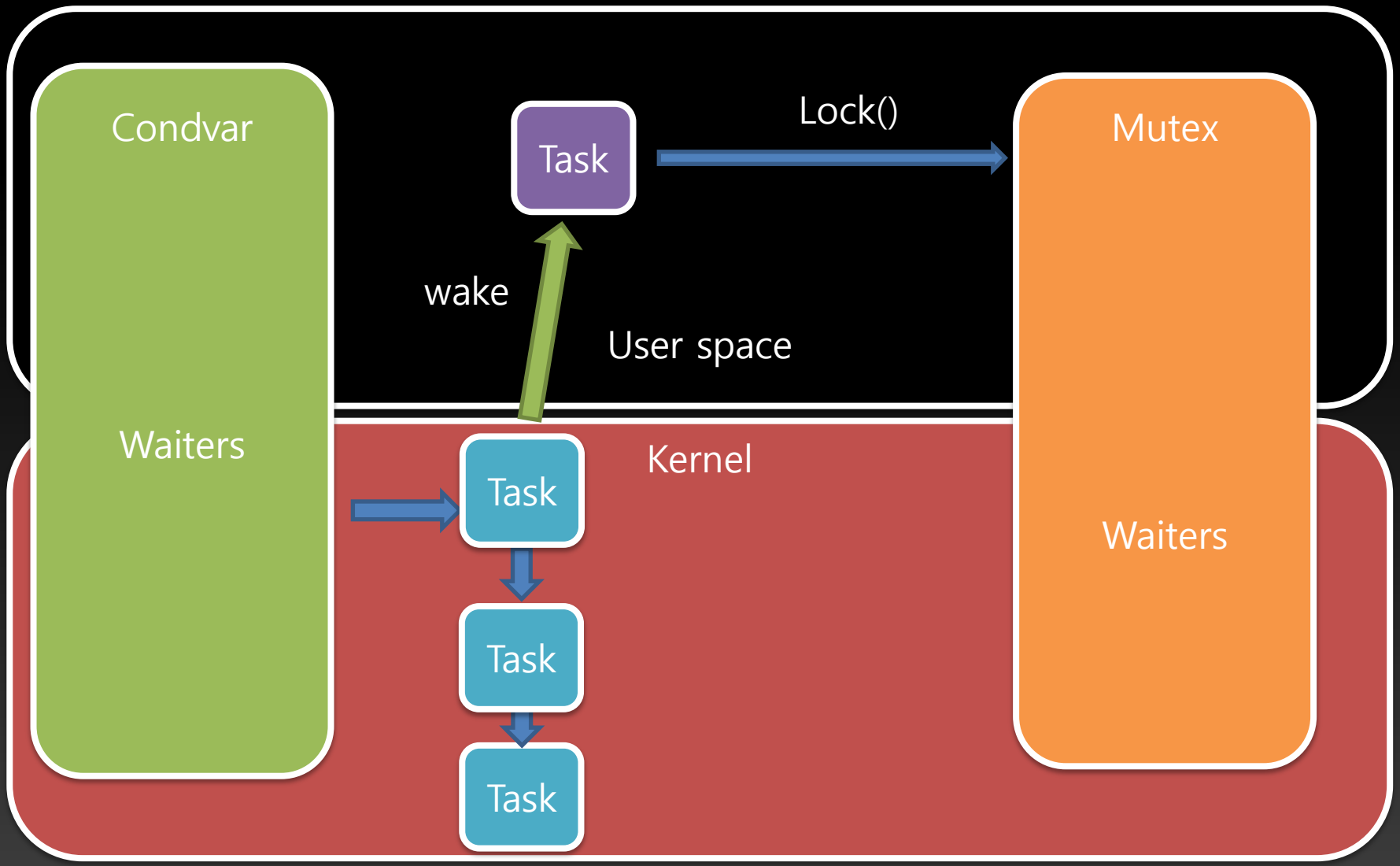
FUTEX_REQUEUE Case 1.



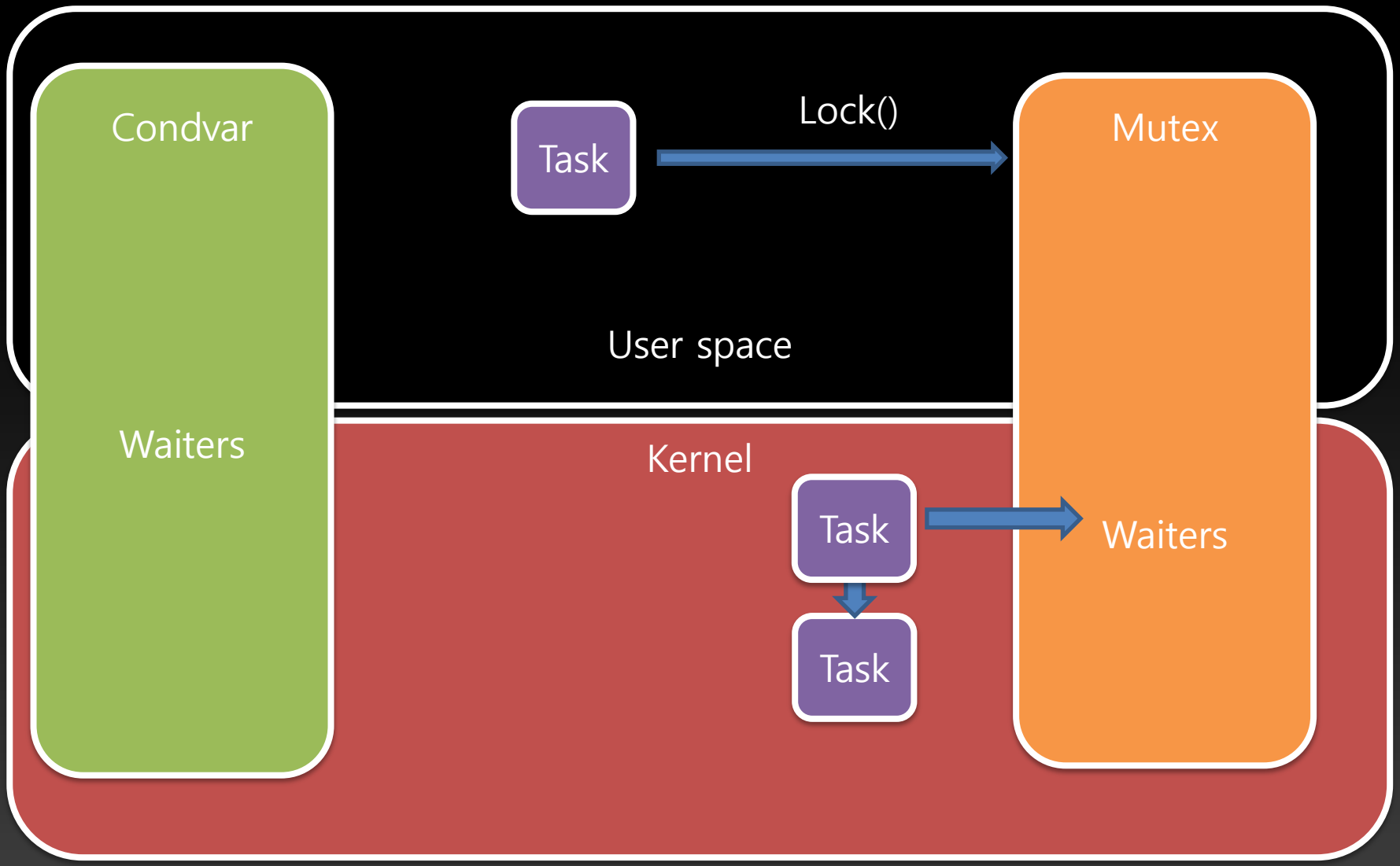
FUTEX_REQUEUE Case 1.



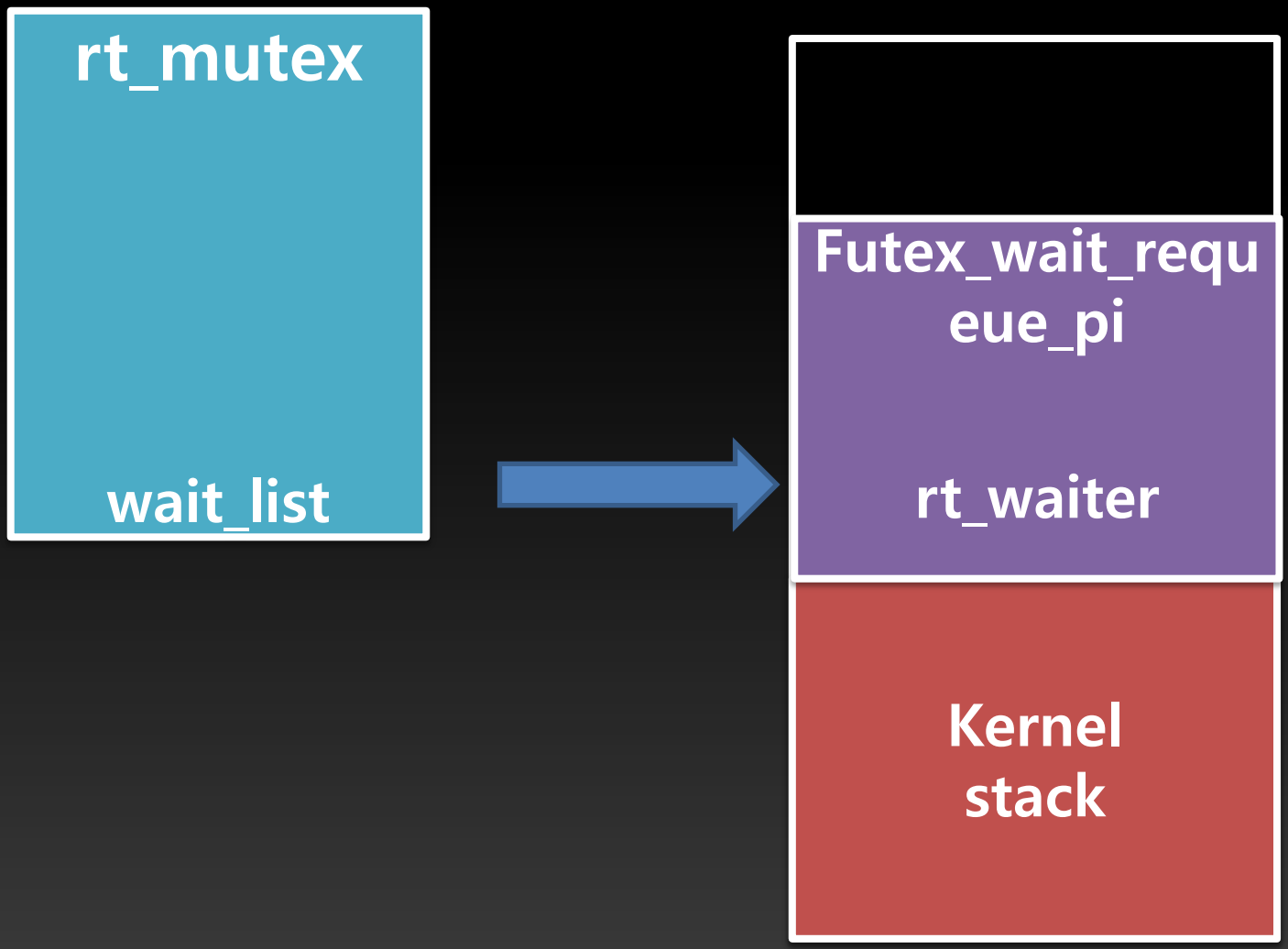
FUTEX_REQUEUE Case 2.



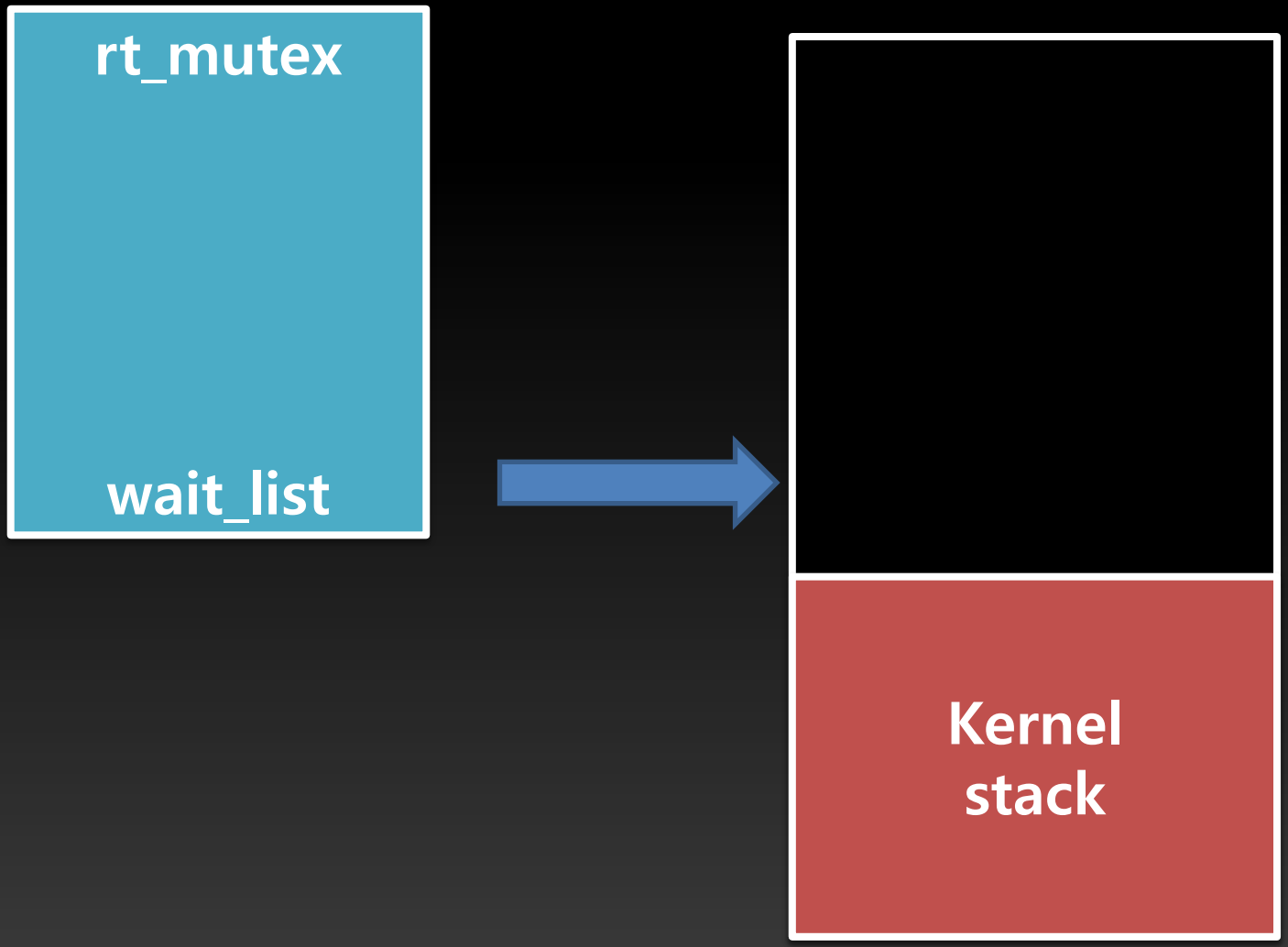
FUTEX_REQUEUE Case 2.



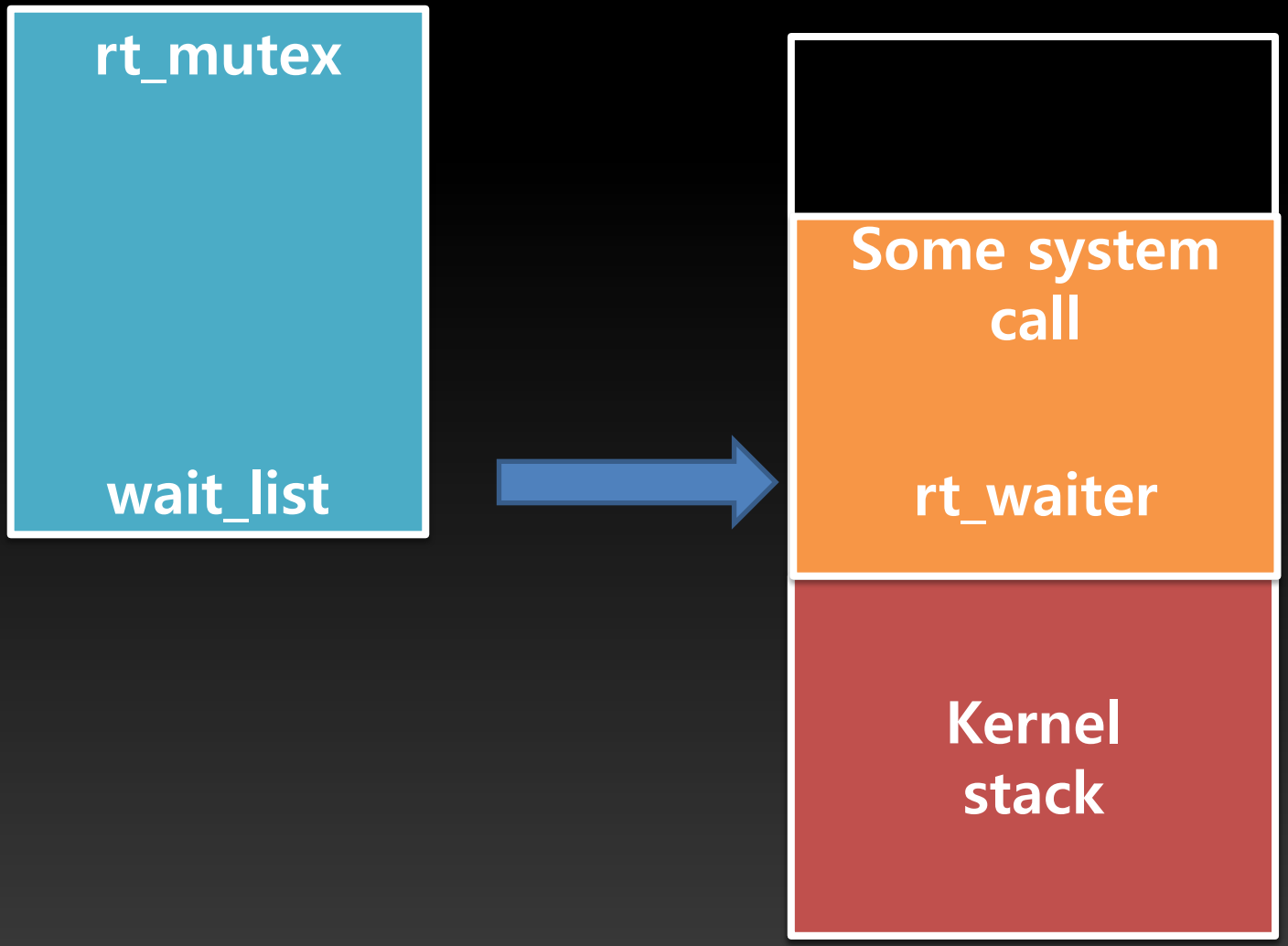
CVE-2014-3153



CVE-2014-3153



CVE-2014-3153



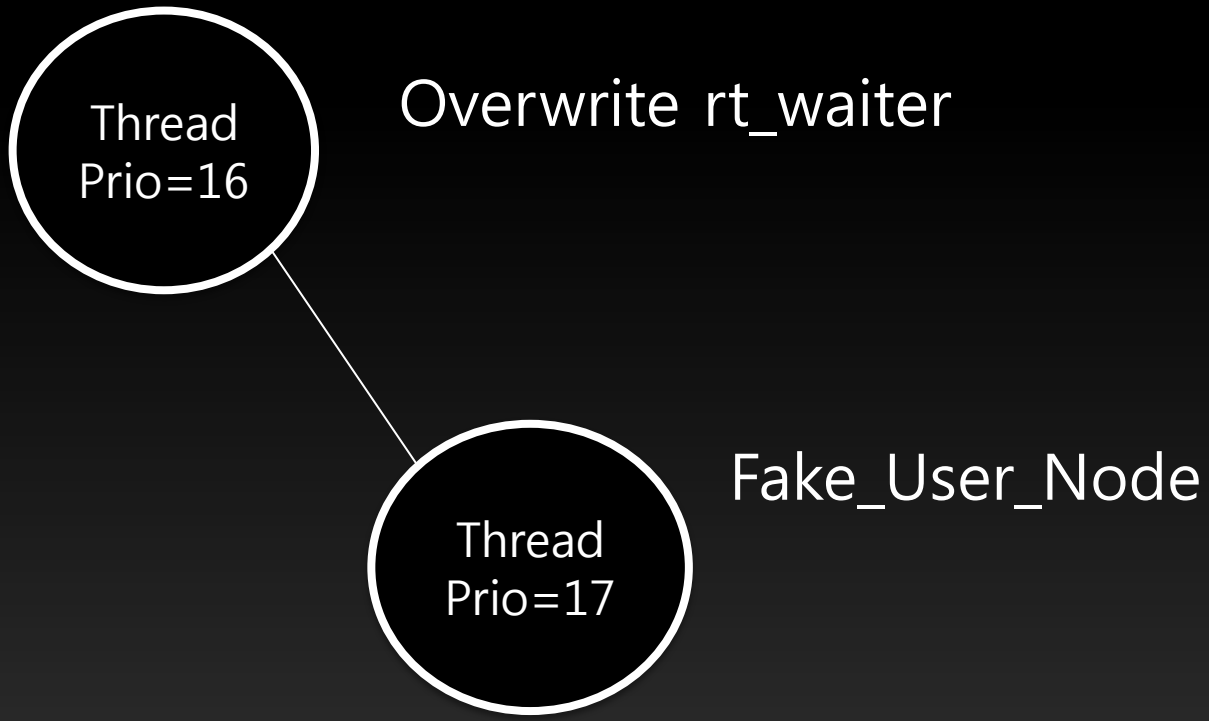
Possible System call

systemcall
sendmmsg
recvmmsg
sendmsg
recvmsg
semctl

Exploitation

1. Kernel memory address leak
2. Arbitary memory corruption
3. Overwrite addr_limit
4. Overwrite cred struct
5. Get shell

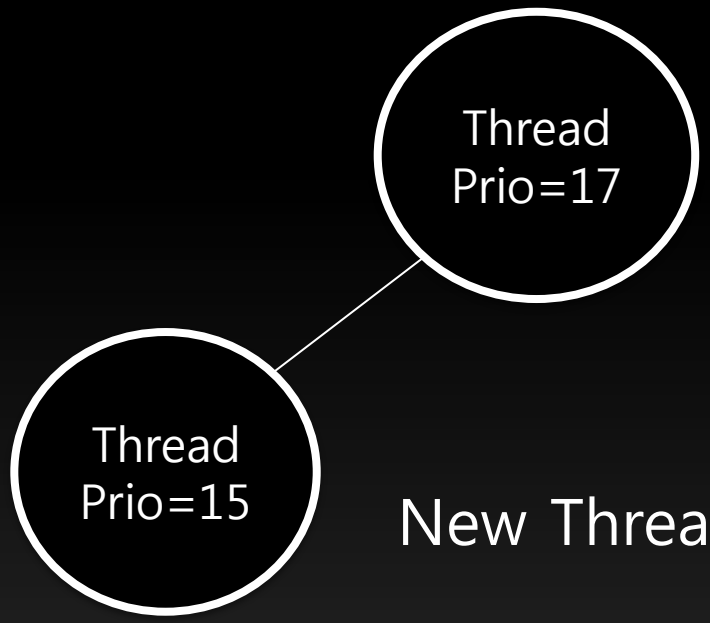
Kernel Memory leak



Kernel Memory leak

Fake_User_Node

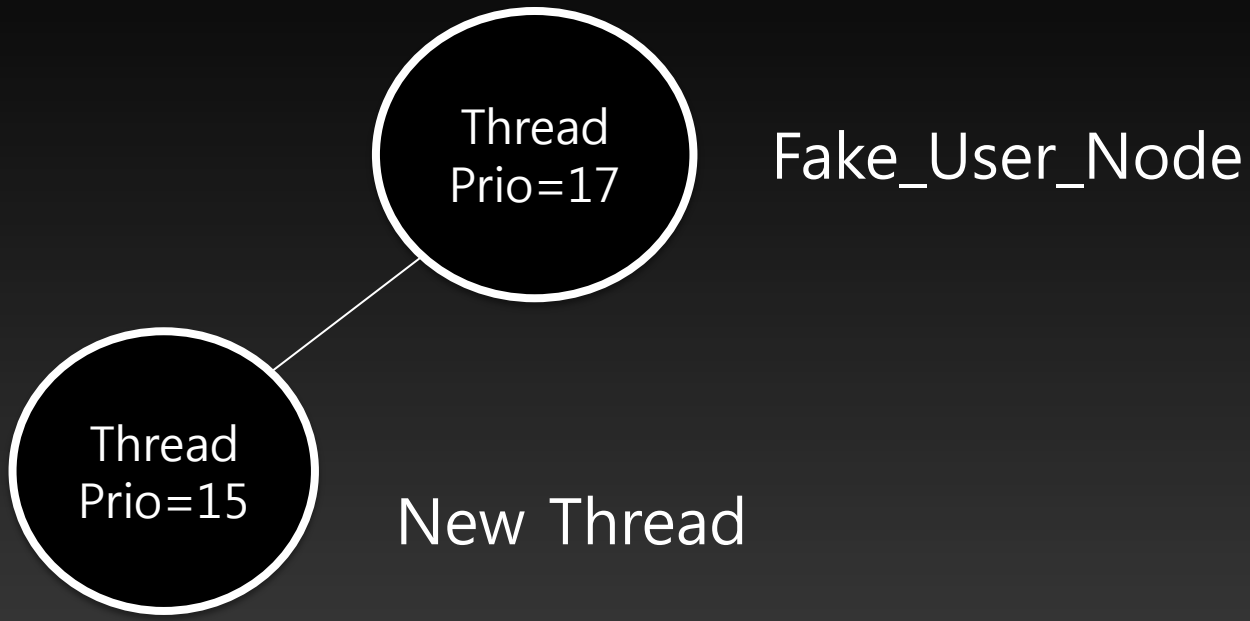
Kernel Memory leak



futex_wait_requeue_pi(&srcfutex,&destfutex) and
futex_cmp_requeue_pi(&srcfutex,&destfutex)
or
Futex_lock_pi

Kernel Memory leak

1. Fake_User_Node는 유저영역
2. 현재 Fake_User_Node의 left가 커널 메모리를 소유



Thread Info

```
struct thread_info {

        .............
        mm_segment_t            addr_limit;     /* address limit */
        struct task_struct      *task;          /* main task structure */
        .............
};
```

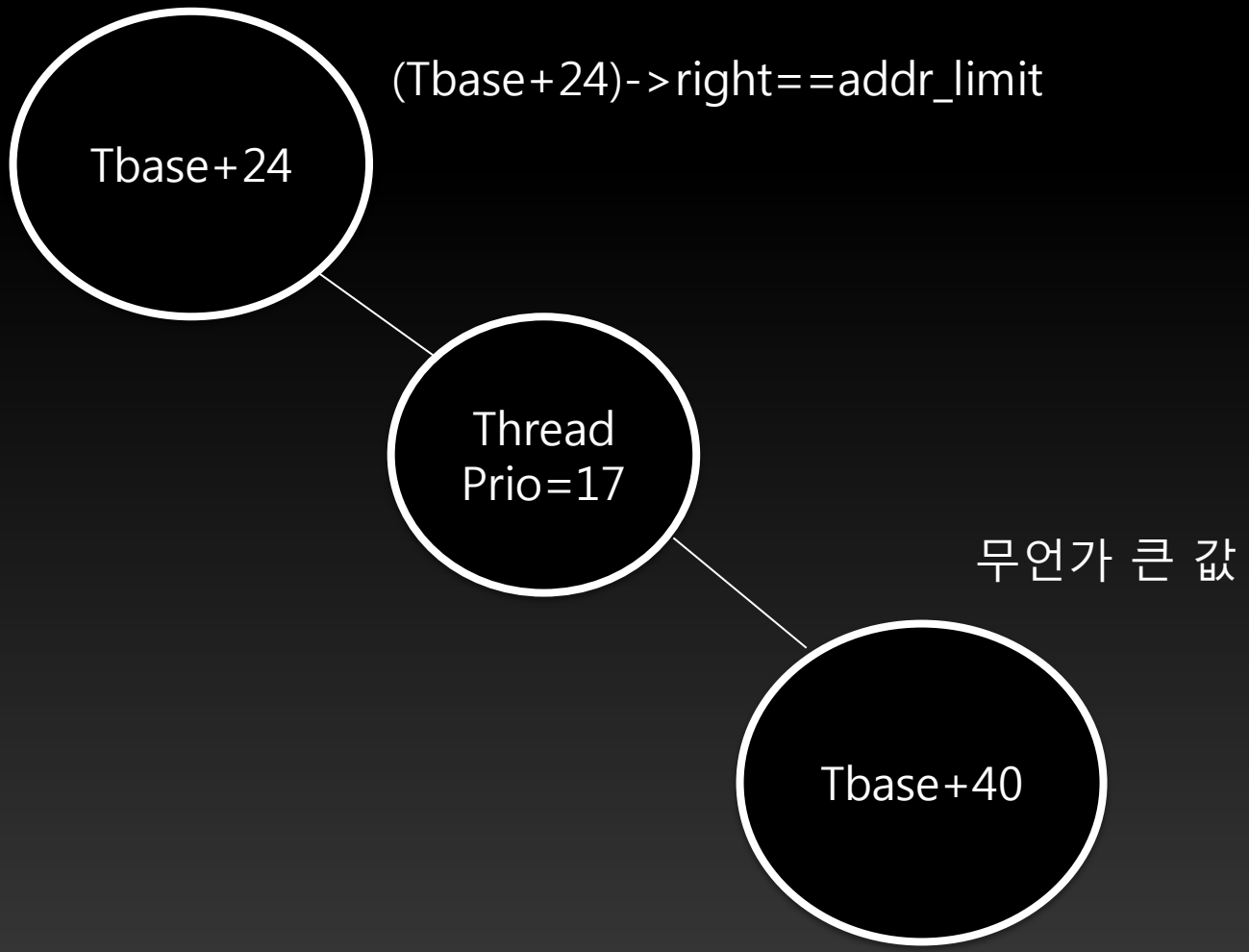
thread_info # 프로세스에 관한 저수준 정보와 커널용 스택을 관리한다.

Thread Info

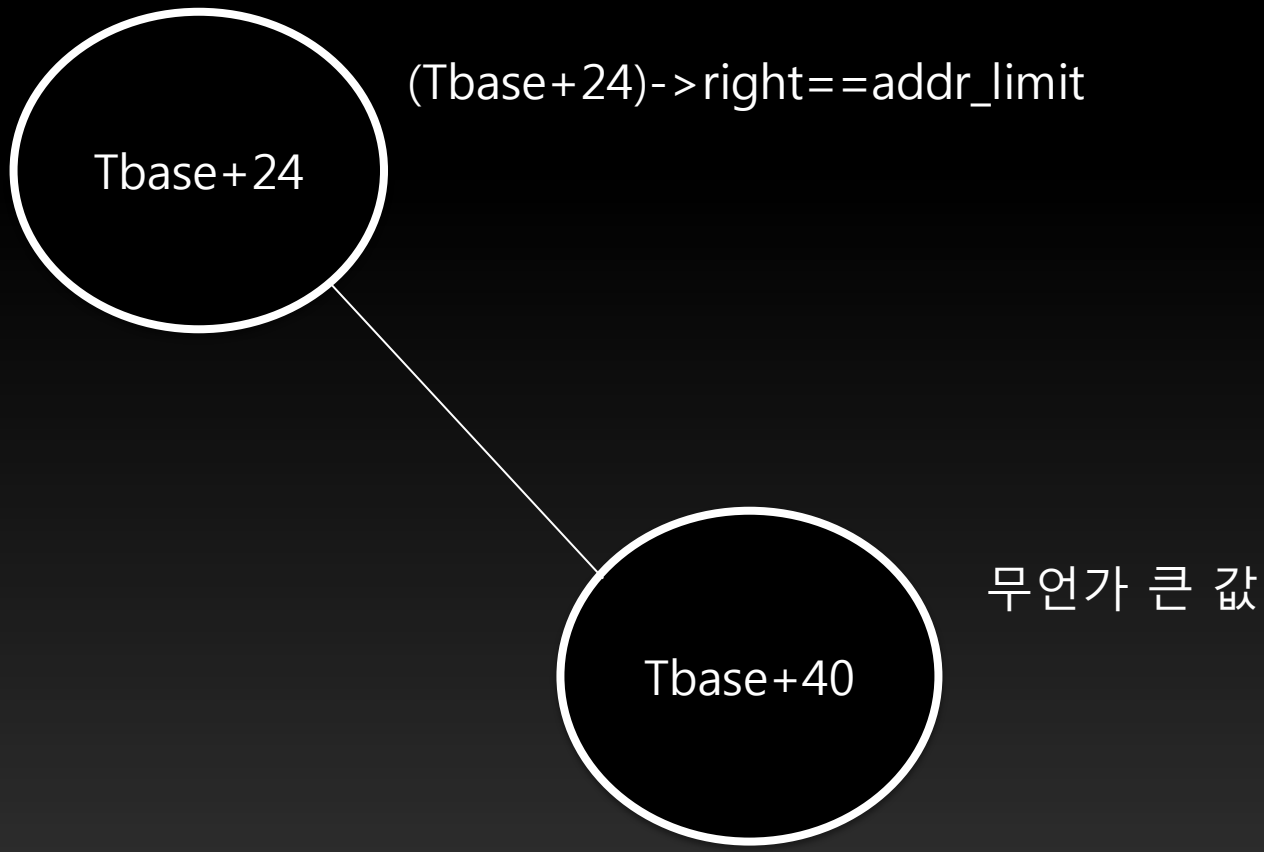
```
static inline struct thread_info *current_thread_info(void)
 {
        register unsigned long sp asm ("sp");
        return (struct thread_info *)(sp & ~(THREAD_SIZE - 1));
 }
```

```
tbase = (unsigned long)kernel_waiter & 0xffffffffffffe000;
```

Arbitary memory corruption



Arbitary memory corruption



Cred Struct

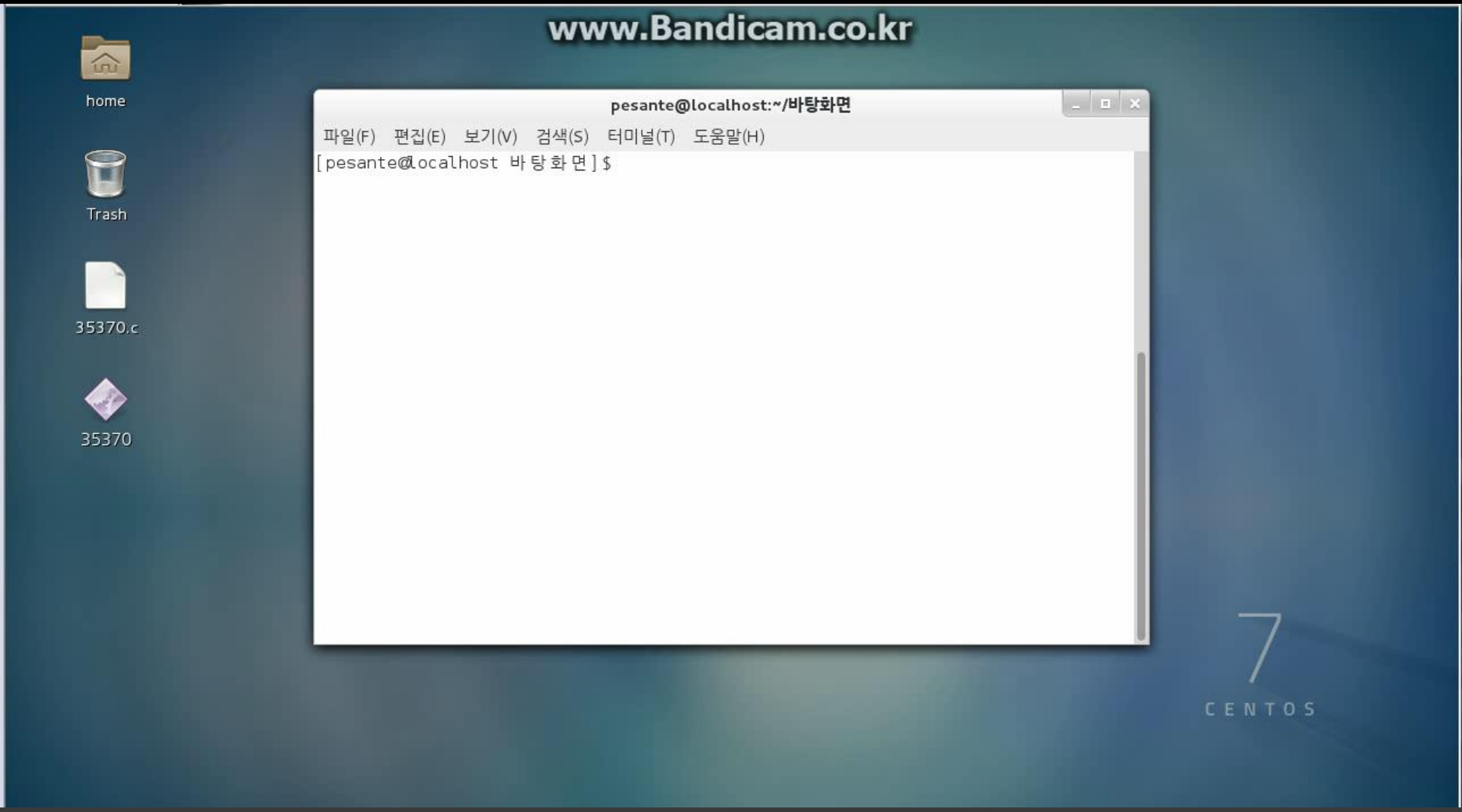
- The security context of a task

Cred Struct

```
struct cred {
        ..........
    uid_t       euid;       /* effective UID of the task */
    gid_t       egid;       /* effective GID of the task */
        ..........
};
```

/bin/sh

시연


www.Bandicam.co.kr
home
Trash
35370.c
35370
pesante@localhost:~/바탕화면
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[pesante@localhost 바탕화면]$
7
CENTOS

QnA

THANK YOU!