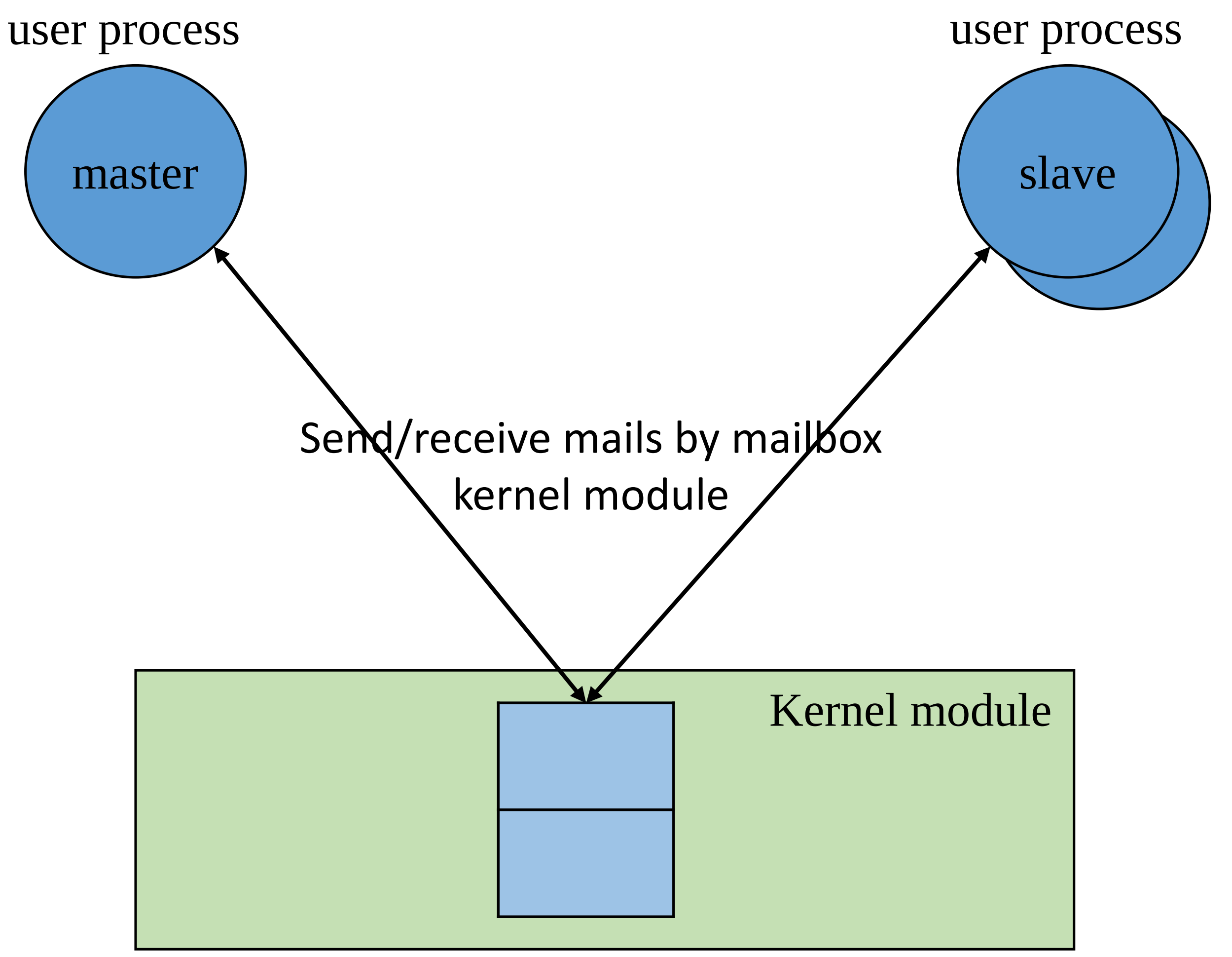
OS 2017

Homework2: mailbox implementation and application

(Due date 12/07 23:59:59)

Lab @NCKU

Architecture



Requirements

1. Write a user application (Master)
   - 2 mandatory arguments: ***QUERY_WORD*** and ***DIRECTORY***
   - 1 optional argument: ***NUM_SLAVE*** (default value = 1)
   - Use fork() and exec() to create ***NUM_SLAVE*** slave(s)
   - Send ***QUERY_WORD*** and ***FILE_PATH*** to slave(s) via the mailbox kernel module
   - Receive result(s) from the slave(s) (also from the mailbox kernel module)
   - Send signals to all slave(s) to kill the slave(s) when receiving all results
2. Write a user application (Slaves)
   - Each time receive from mailbox to obtain a pair of ***QUERY_WORD*** and ***FILE_PATH***
   - Count the number of ***QUERY_WORD*** appearing in ***FILE_PATH***
   - Send the result (***WORD_COUNT*** and ***FILE_PATH)*** back to the master
   - Receive another pair as necessary
3. Write a mailbox (kernel module)
   - Create one sysfs file as module interface
   - Use ***struct list_head*** to implement your mailbox
   - Can receive an optional argument as ***NUM_ENTRY_MAX*** when inserted (default value = 2)
   - Use spin_lock to protect the mailbox from race condition (multi-user read/write)

Argument definition (Master)

$ ./master▽-q▽***QUERY_WORD***▽-d▽***DIRECTORY***▽-s▽***K***

▽: white space(s)

- What word to count

- The target directory
- May be absolute or relative path

- An optional argument
- create ***K*** slaves

The order of the three arguments may change
Ex, it may be "-s ***K*** -q ***QUERY_WORD*** -d ***DIRECTORY***"

User-level mail structures and APIs

mail.h

```c
struct mail_t {
    union {
        char query_word[32];
        unsigned int word_count;
    } data;
    char file_path[4096];
};

int send_to_fd(int sysfs_fd, struct mail_t *mail);
int receive_from_fd(int sysfs_fd, struct mail_t *mail);
```

1. Used by Master and Slave(s)
2. Use the APIs and structures to send/receive mails
   - Please do not modify the definitions of the structures and APIs
3. Implement the send and receive functions (i.e., send_to_fd() and receive_from_fd()) by yourself

Sysfs file (1/2)

module/mailbox.c

```c
static struct kobject *hw2_kobject;
static struct kobj_attribute mailbox_attribute
    = __ATTR(mailbox, 0660, mailbox_read, mailbox_write);

static int num_entry_max = 2;
…
static int __init mailbox_init(void) {
    printk("Insert\n");
    hw2_kobject = kobject_create_and_add("hw2", kernel_kobj);
    sysfs_create_file(hw2_kobject, &mailbox_attribute.attr);
    return 0;
}

static void __exit mailbox_exit(void) {
    printk("Remove\n");
    kobject_put(hw2_kobject);
}

module_init(mailbox_init);
module_exit(mailbox_exit);
```

1. Sysfs file creation has been included in mailbox.c
2. Implement the read and write functions (shown in the next slide)
3. Sysfs file path is /sys/kernel/hw2/mailbox

Sysfs file (2/2)

module/mailbox.h

```
static ssize_t mailbox_read(struct kobject *kobj,
                                                    struct kobj_attribute *attr, char *buf);
static ssize_t mailbox_write(struct kobject *kobj,
                                                       struct kobj_attribute *attr, const char *buf,
                                                        size_t count);
```

In-kernel mail buffer structures

module/mailbox.h

```c
struct mail_buffer_head_t {
    /*
     * some structure members you define
     */
    struct list_head head;
};

struct mail_buffer_entry_t {
    /*
     * some structure members you define
     */
    struct list_head entry;
};
```

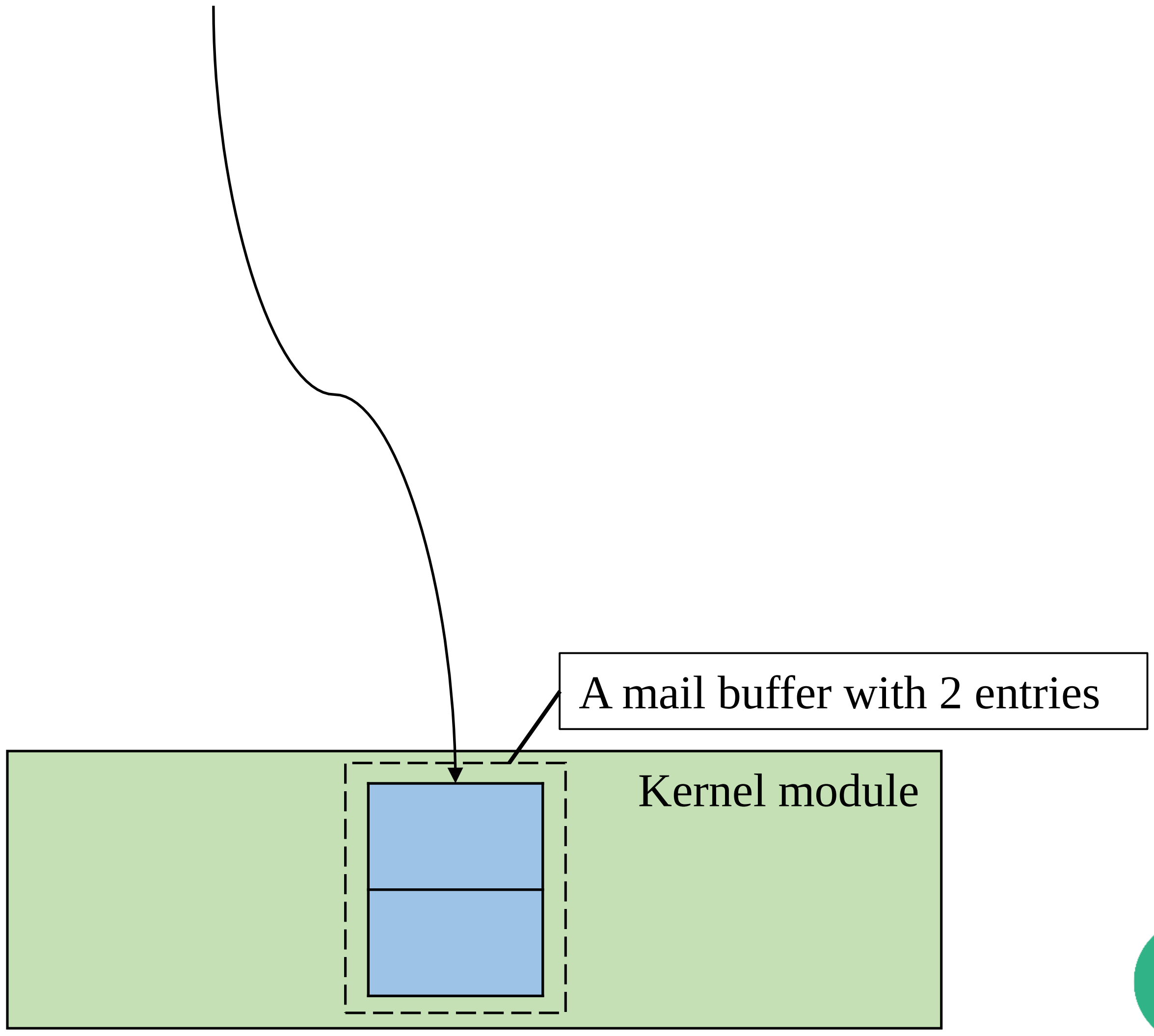
Defined in linux/list.h

```c
struct list_head {
    struct list_head *next, *prev;
};
```

1. Used by the kernel module
2. Use ***mail_buffer_head_t*** and ***mail_entry_t*** to implement your mail buffer
    - You must use ***list_head*** for chaining mail buffers
    - Define other members you need

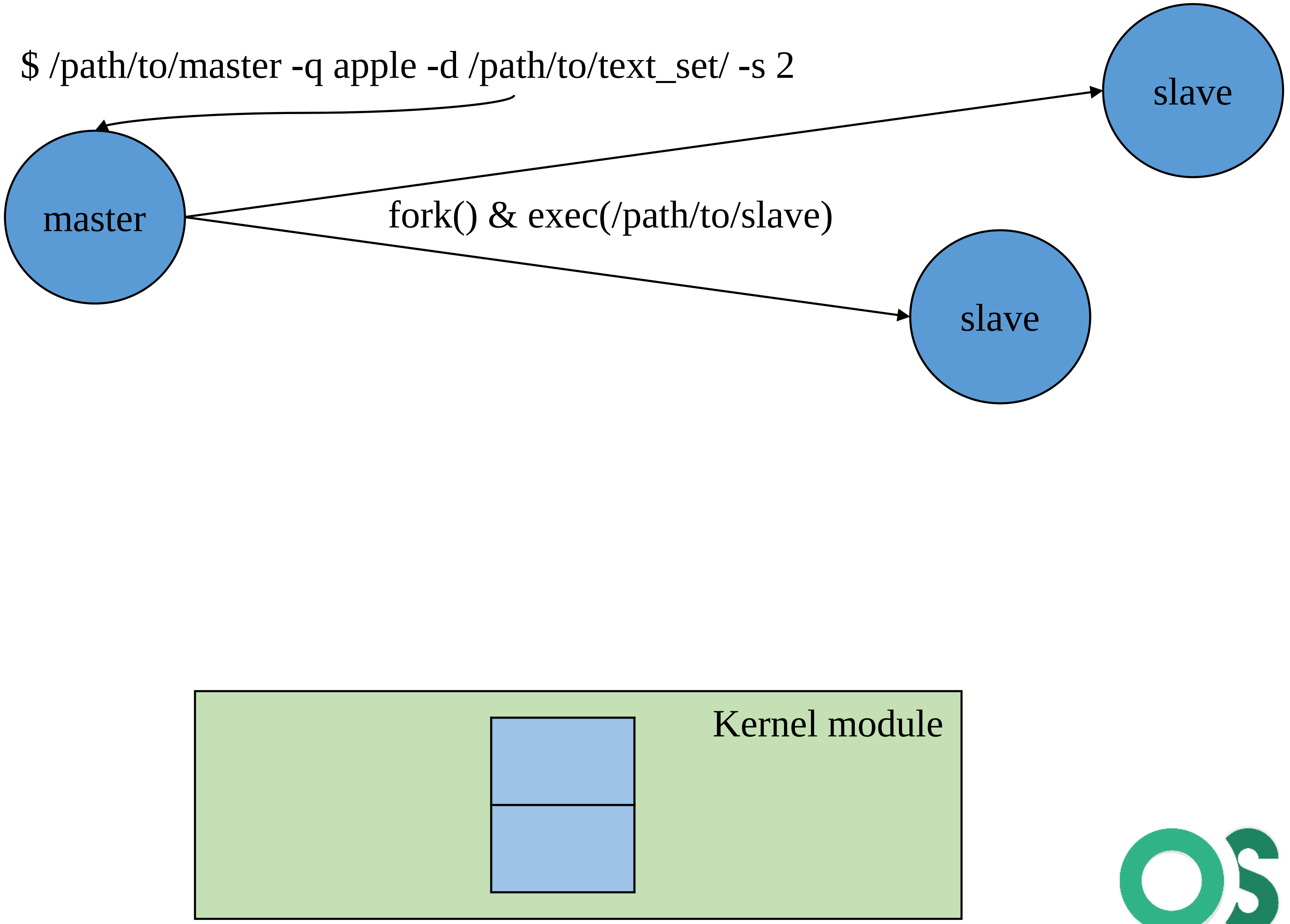
Flow (1/7)

```
$ sudo insmod mail.ko num_entry_max=2
```

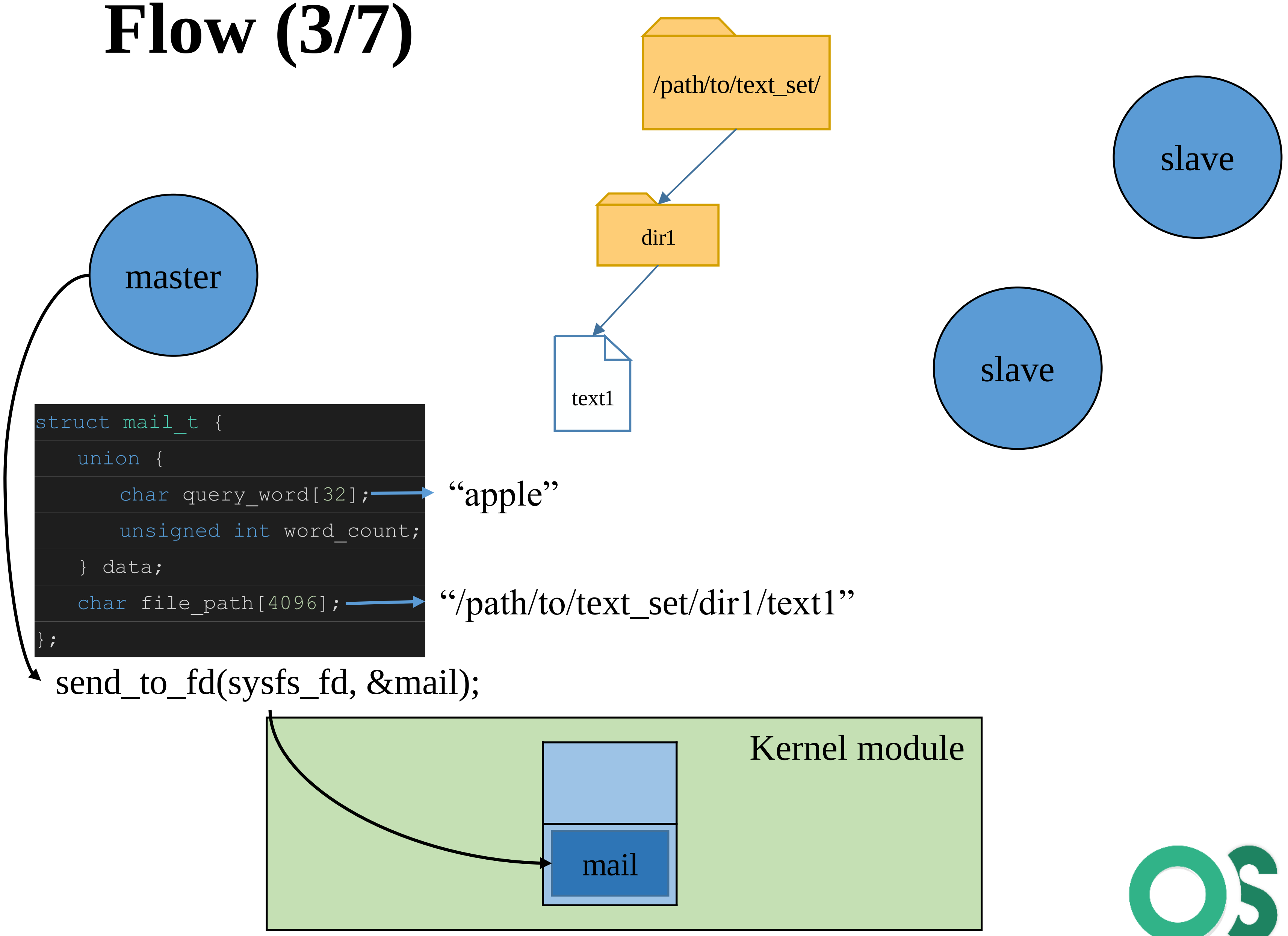


Flow (2/7)

```
$ /path/to/master -q apple -d /path/to/text_set/ -s 2
```



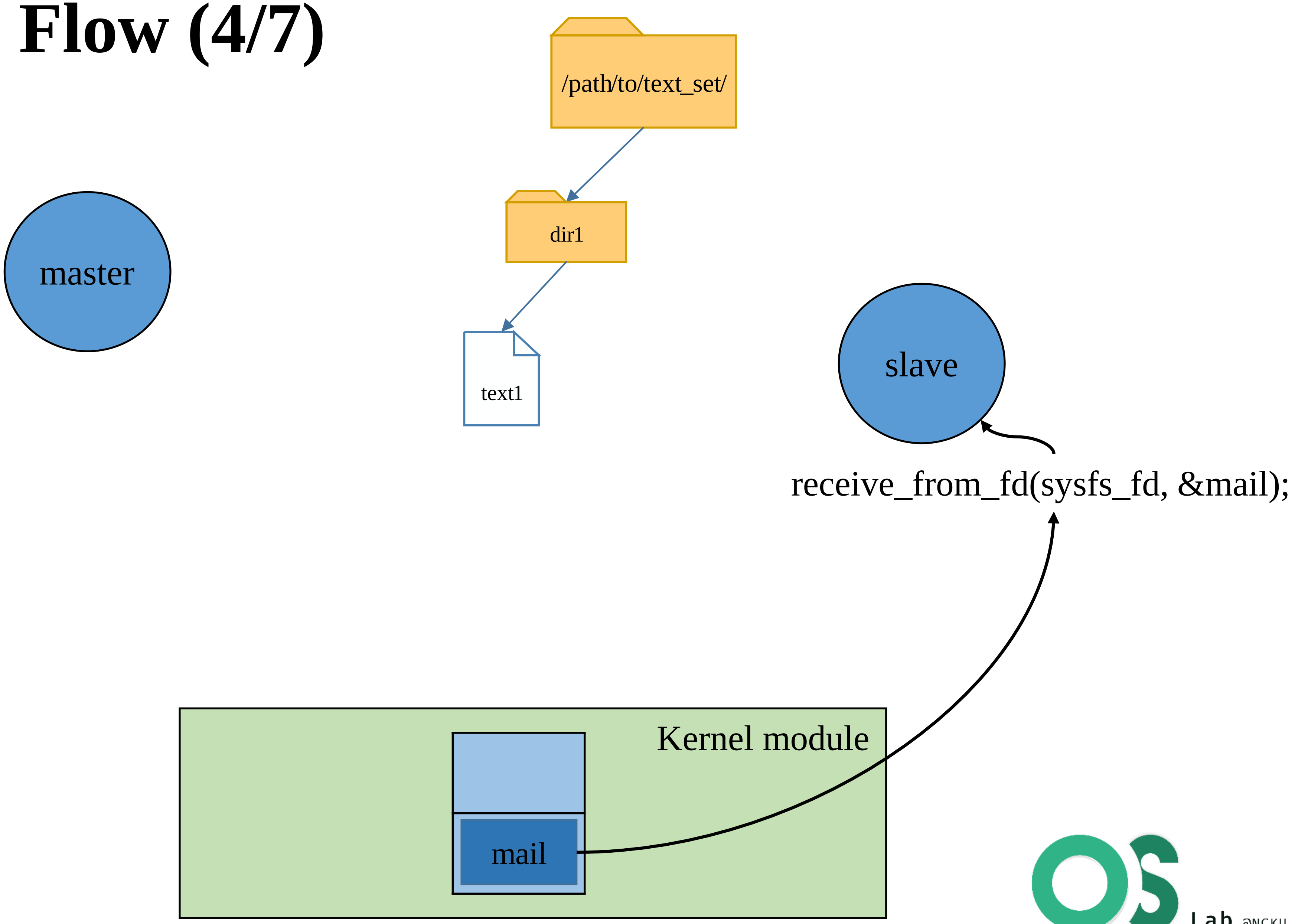
Flow (3/7)



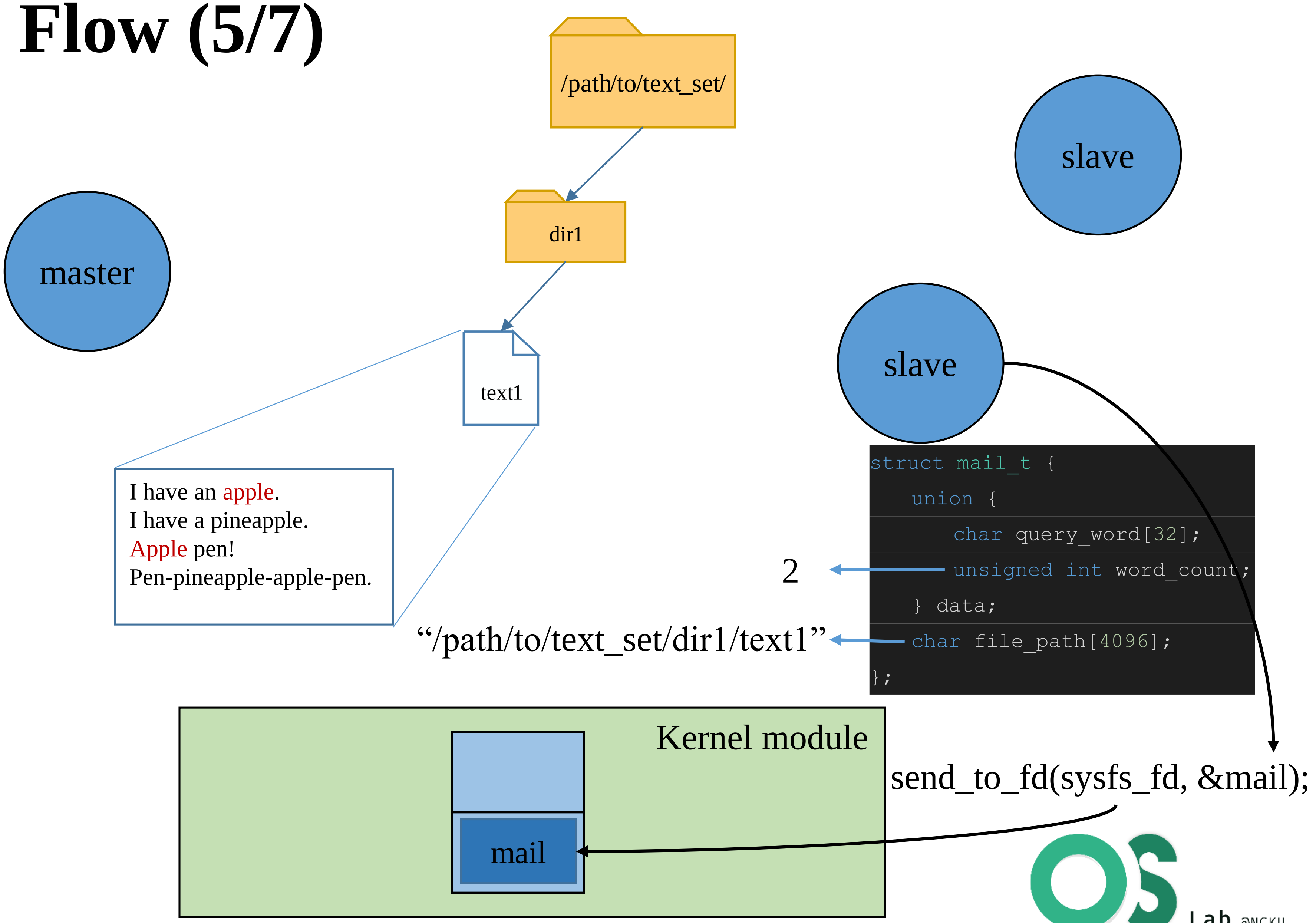
```
struct mail_t {
    union {
        char query_word[32];
        unsigned int word_count;
    } data;
    char file_path[4096];
};
```

send_to_fd(sysfs_fd, &mail);

Flow (4/7)


/path/to/text_set/
dir1
text1
master
slave
receive_from_fd(sysfs_fd, &mail);
Kernel module
mail

Flow (5/7)

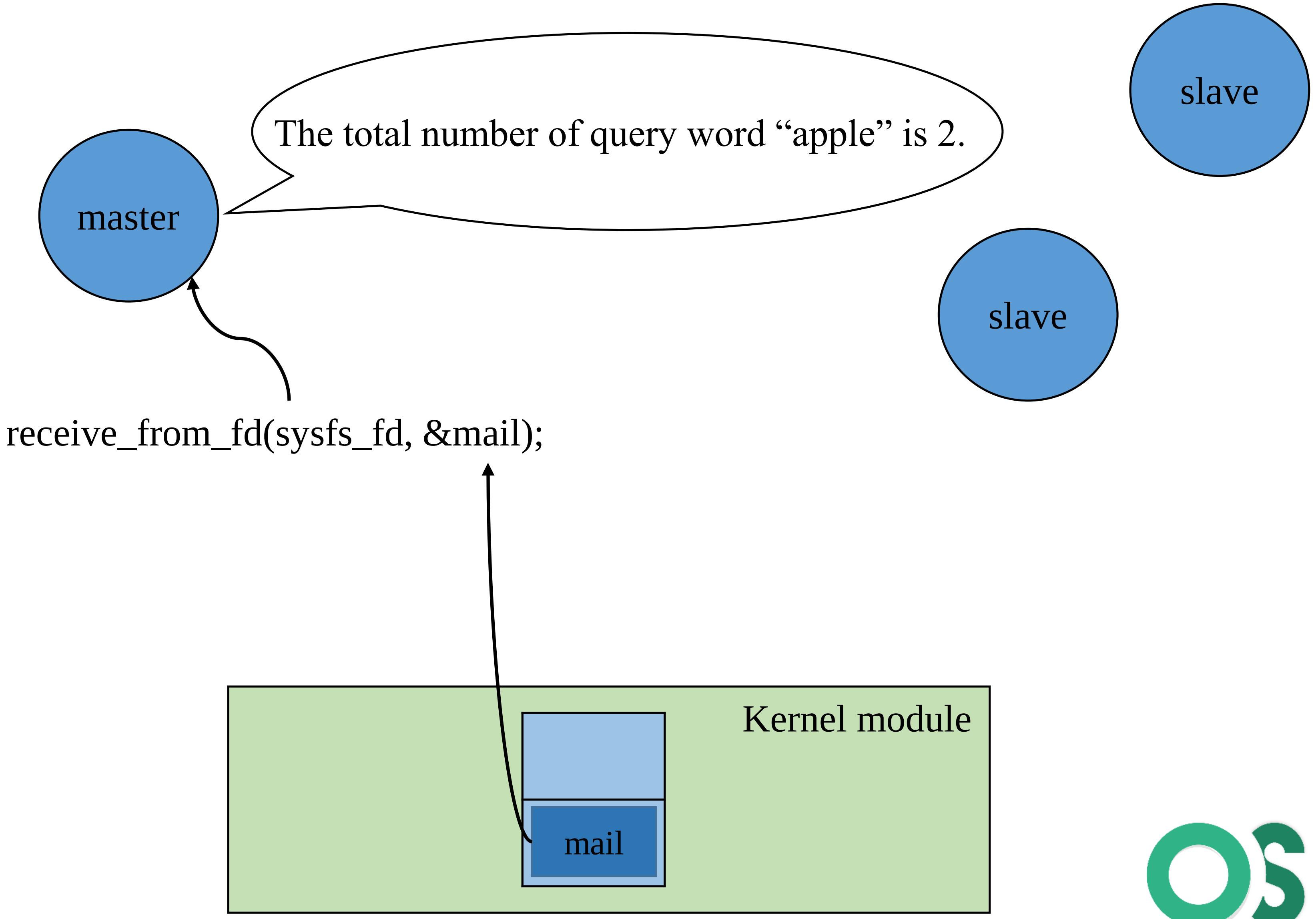


I have an apple.
I have a pineapple.
Apple pen!
Pen-pineapple-apple-pen.

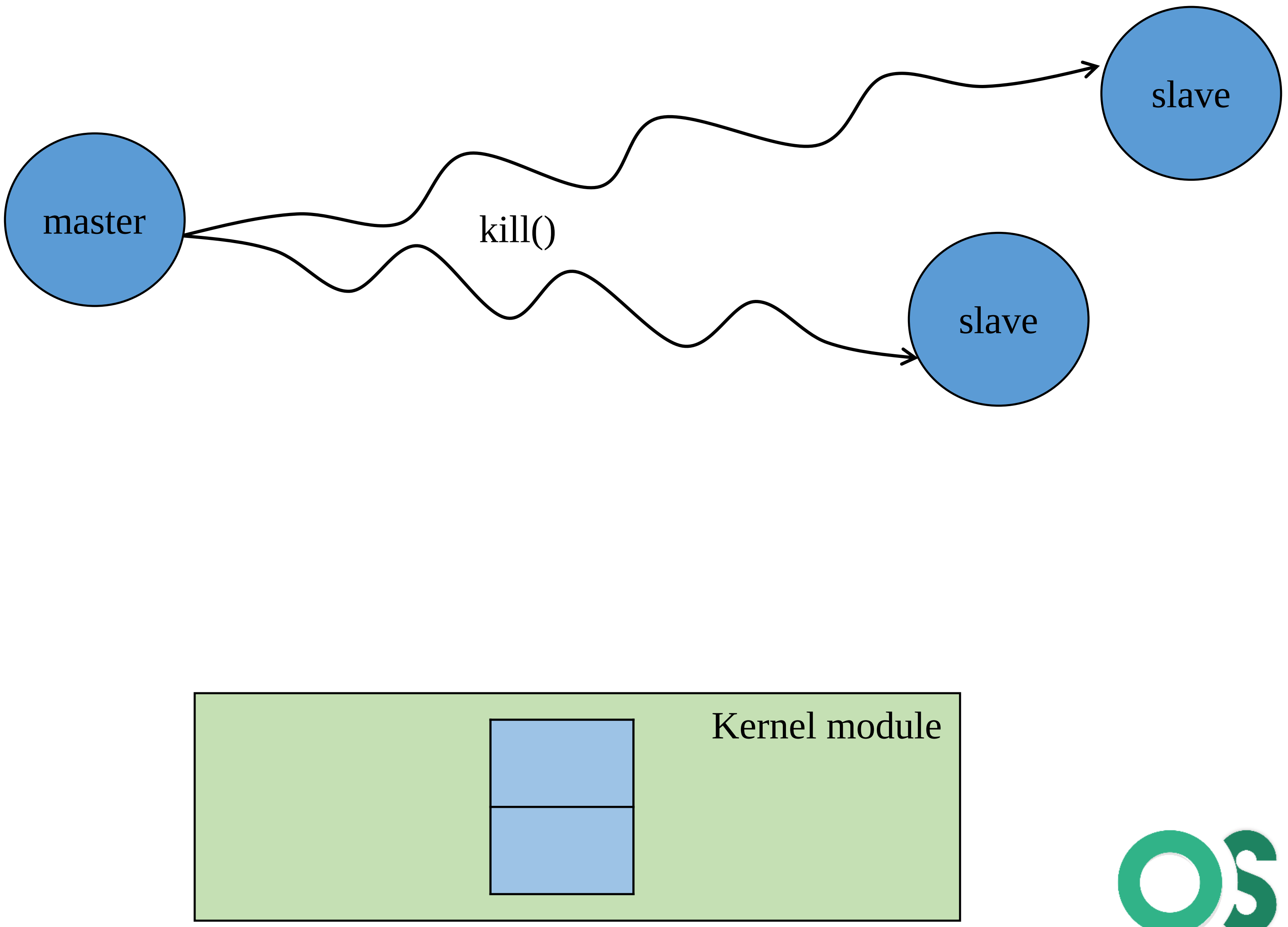
"/path/to/text_set/dir1/text1"

```
struct mail_t {
    union {
        char query_word[32];
        unsigned int word_count;
    } data;
    char file_path[4096];
};
```

Flow (6/7)


slave
The total number of query word “apple” is 2.
master
slave
receive_from_fd(sysfs_fd, &mail);
Kernel module
mail

Flow (7/7)



References (1/2)

- Kernel module
  - The Linux Kernel Module Programming Guide
  - Derekmolloy.ie
  - The Geek Stuff
- Sysfs
  - Man page
  - Penesive
- Linked-List
  - MakeLinux
  - Gitbook
- Spin lock API
  - MakeLinux
  - Gitbook

References (2/2)

- Fork & wait & exec
  - YoLinux Tutorial
- Signal & kill()
  - Man page
- Linux code references
  - Free Electrons
  - The Linux Kernel API