파이썬 – HW7

임베디드스쿨1기

Lv1과정

2020. 09. 16

강경수

1. Python class

▣ Python Class

2020.09.15

1. What is class?

- object(객체)를 만드는 틀 → Class
- Class에 의하여 만들어진 것 → object(객체)
- 클래스로 만든 객체는 객체마다 고유의 특성을 갖는다.

2. class 생성 예시

```
class schoolman:
    pass

a = schoolman()
b = schoolman()
```

- a 와 b는 각각 객체이며 서로에게 영향을 주지 않는다

3. 객체와 인스턴트의 차이점

- a는 객체이다
- a는 schoolman의 인스턴트이다.

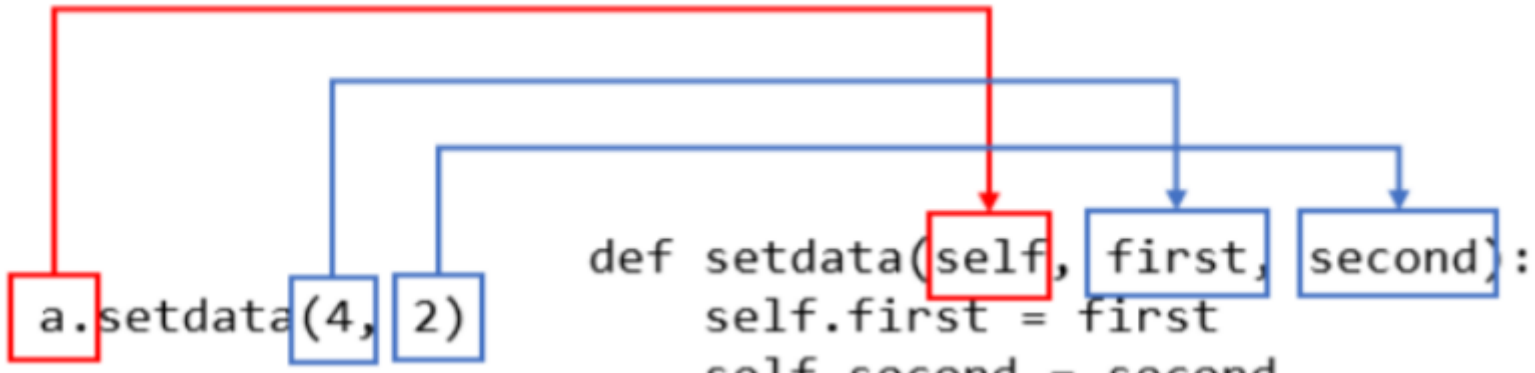
4. Method

- 클래스 안의 함수를 Method라 한다.

```
class calculate:
    def data(self,first,second):
        self.first = first
        self.second = second
```

1. Python class

5. Method 호출 방법



메서드를 사용할때 일반 함수 사용과다르게 (객체명).함수명(인자) 형태를 취한다.

```
class calculate:
    def data(self,first,second):
        self.first = first
        self.second = second
```

메서드를 호출하는 두 가지 방법

아래 방법은 잘 사용되지 않는다.

```
a = calculate()

a.data(3,5)
calculate.data(a,5,10)
```

1. Python class

6. 생성자(Constructor)

- Method의 이름으로 __init__를 사용하면 된다.
- __init__ 즉 생성자는 객체가 생성될때 자동으로 호출되는 Method 이다.
- 생성자를 사용시 위와 같은 별도의 객체변수값 지정 Method 호출 없이 객체 생성과 동시에 객체변수값 할당

```
class calculate:
    def data(self,first,second):
        self.first = first
        self.second = second
    def sum(self):
        self.sum_val = self.first + self.second
        return self.sum_val


a = calculate(3,5)
print(a.sum())
```

1. Python class

7.Class의 상속

- 상속은 기존 class에 기능을 추가하거나 기존 기능을 변경하기 위하여 사용
- Class가 라이브러리로 제공되거나 수정이 불가능할 경우 상속하여 사용한다.

```
class calculate:
    def data(self,first,second):
        self.first = first
        self.second = second
    def sum(self):
        self.sum_val = self.first + self.second
        return self.sum_val

class calculate_edit(calculate):
    def mul(self):
        self.mul_val= self.first * self.second
        return self.mul_val

a = calculate_edit()
a.data(3,5)
print(a.sum())
print(a.mul())
```

← 메서드를 상속하는 방법

1. Python class

8. Class Method , Static Method

```
class mystatic_method:
    @staticmethod
    def add(a,b):
        add_val = a + b
        return add_val

print(mystatic_method.add(10,30))

class my_classmethod:
    count = 0

    def __init__(self):
        my_classmethod.coun

    @classmethod
    def print_count(cls):
        print("{0}명 생성되었습니다.".format(cls.count))

kks = my_classmethod()
kkp = my_classmethod()

my_classmethod.print_count()
```

← staticmethod 를 활용하여 class에서 method 직접호출

static method의 경우 메서드의 실행이 외부에 영향을 끼치지 않는 경우에 사용한다

← static method와 같이 인스턴스 없이 호출 가능

하지만 클래스 메서드는 메서드 안에서 클래스 속성

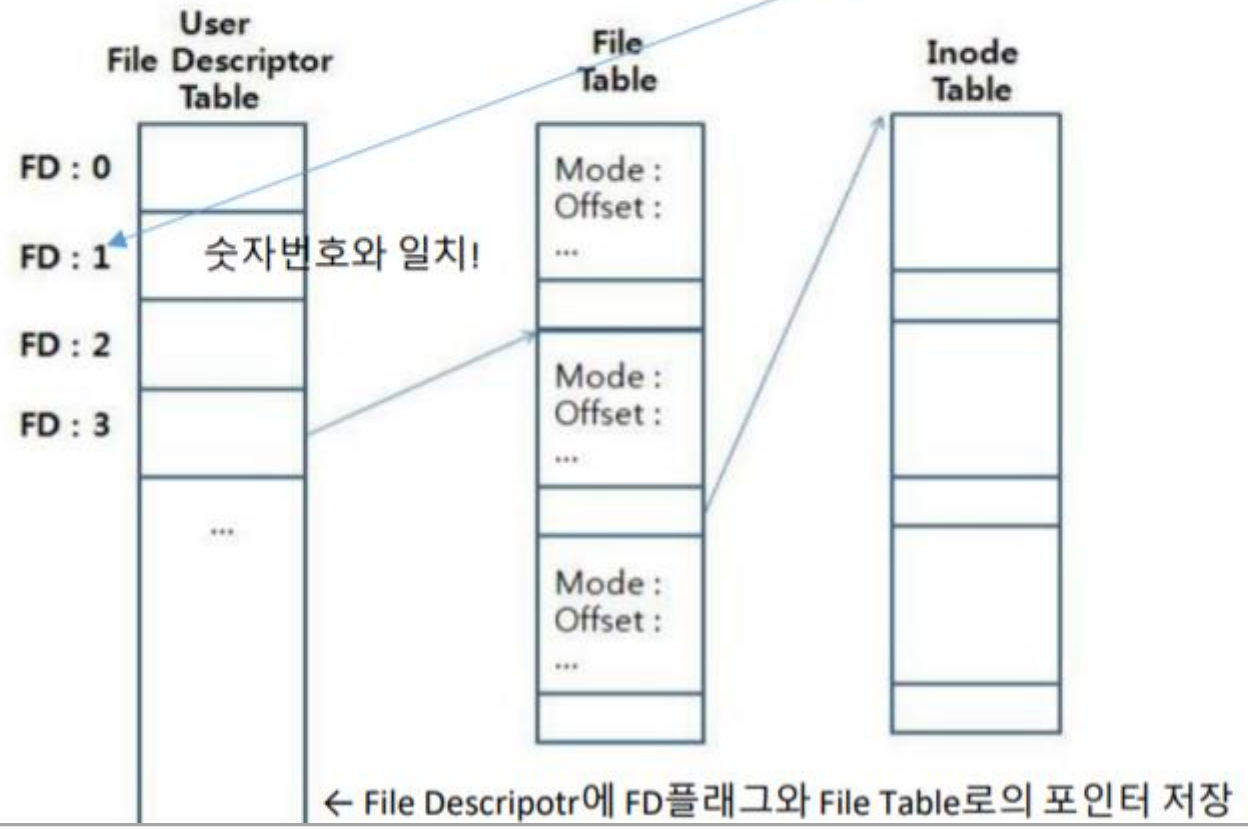
클래스 메서드에 접근해야 할 때 사용.

▣ LINUX KERNEL

2020.09.15

1. FD(File Descriptor)란 무엇인가?

- 유닉스 시스템의 철학 '모든것은 파일이다' (디렉토리,네트워크,외부하드웨어,소켓 등)
- 유닉스 시스템에서 프로세서가 이 파일들에 접근할 때 파일 디스크립터라는 개념을 이용
- 프로세서가 실행중에 파일을 open 하면 커널은 해당 프로세스 파일에 fd 값(0이아닌 정수) 중 가장 작은값 할당
- 시스템콜을 이용하여 프로세서가 파일에 접근할때 fd값을 이용하여 파일 지칭



2. task_struct

```
struct task_struct {
#ifdef CONFIG_THREAD_INFO_IN_TASK
        /*
         * For reasons of header soup (see current_thread_info()), this
         * must be the first element of task_struct.
         */
        struct thread_info              thread_info;
#endif
        /* -1 unrunnable, 0 runnable, >0 stopped: */
        volatile long                   state;
```

- 커널 내부에서 task에 접근할때 task_struct 사용

```
923             /* Open file information: */
924             struct files_struct             *files;
925
926             /* Namespaces: */
```

- tast_struct 내부의 구조체 files_struct 확인

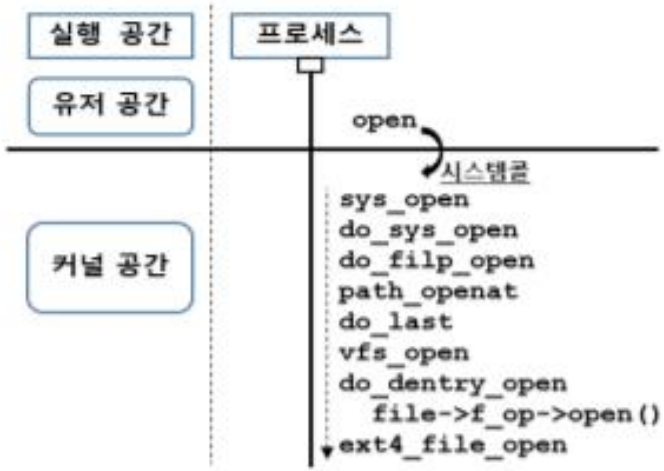
```
struct files_struct {
  /*
   * read mostly part
   */
        atomic_t count;
        bool resize_in_progress;
        wait_queue_head_t resize_wait;

        struct fdtable __rcu *fdt;
        struct fdtable fdtab;
  /*
   * written part on a separate cache line in SMP
   */
        spinlock_t file_lock ____cacheline_aligned_in_smp;
        unsigned int next_fd;
        unsigned long close_on_exec_init[1];
        unsigned long open_fds_init[1];
        unsigned long full_fds_bits_init[1];
        struct file __rcu * fd_array[NR_OPEN_DEFAULT];
};
```

2. LINUX KERNEL

```
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
        ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
        int (*iopoll)(struct kiocb *kiocb, bool spin);
        int (*iterate) (struct file *, struct dir_context *);
        int (*iterate_shared) (struct file *, struct dir_context *);
        __poll_t (*poll) (struct file *, struct poll_table_struct *);
        long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
        long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        unsigned long mmap_supported_flags;
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *, fl_owner_t id);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, loff_t, loff_t, int datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
        ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int);
        unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned long, unsigned long);
        int (*check_flags)(int);
        int (*flock) (struct file *, int, struct file_lock *);
        ssize_t (*splice_write)(struct pipe_inode_info *, struct file *, loff_t *, size_t, unsigned int);
        ssize_t (*splice_read)(struct file *, loff_t *, struct pipe_inode_info *, size_t, unsigned int);
        int (*setlease)(struct file *, long, struct file_lock **, void **);
        long (*fallocate)(struct file *file, int mode, loff_t offset,
                          loff_t len);
        void (*show_fdinfo)(struct seq_file *m, struct file *f);
#ifndef CONFIG_MMU
        unsigned (*mmap_capabilities)(struct file *);
#endif
        ssize_t (*copy_file_range)(struct file *, loff_t, struct file *,
                        loff_t, size_t, unsigned int);
        loff_t (*remap_file_range)(struct file *file_in, loff_t pos_in,
                                   struct file *file_out, loff_t pos_out,
                                   loff_t len, unsigned int remap_flags);
        int (*fadvise)(struct file *, loff_t, loff_t, int);
} __randomize_layout;
```

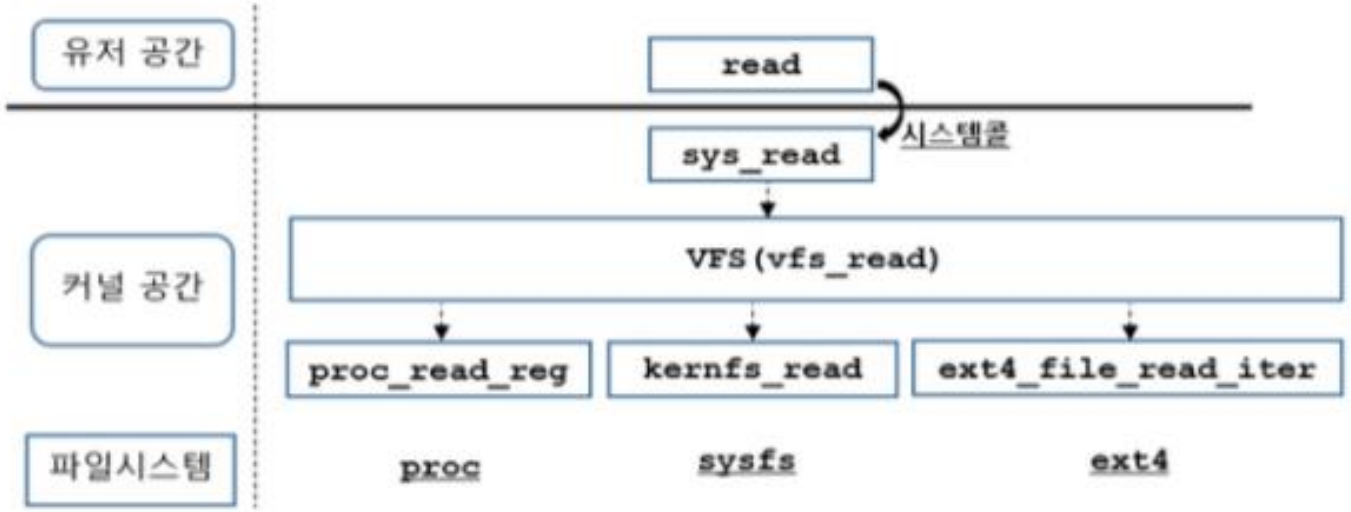
- files_operation 확인

3.open()



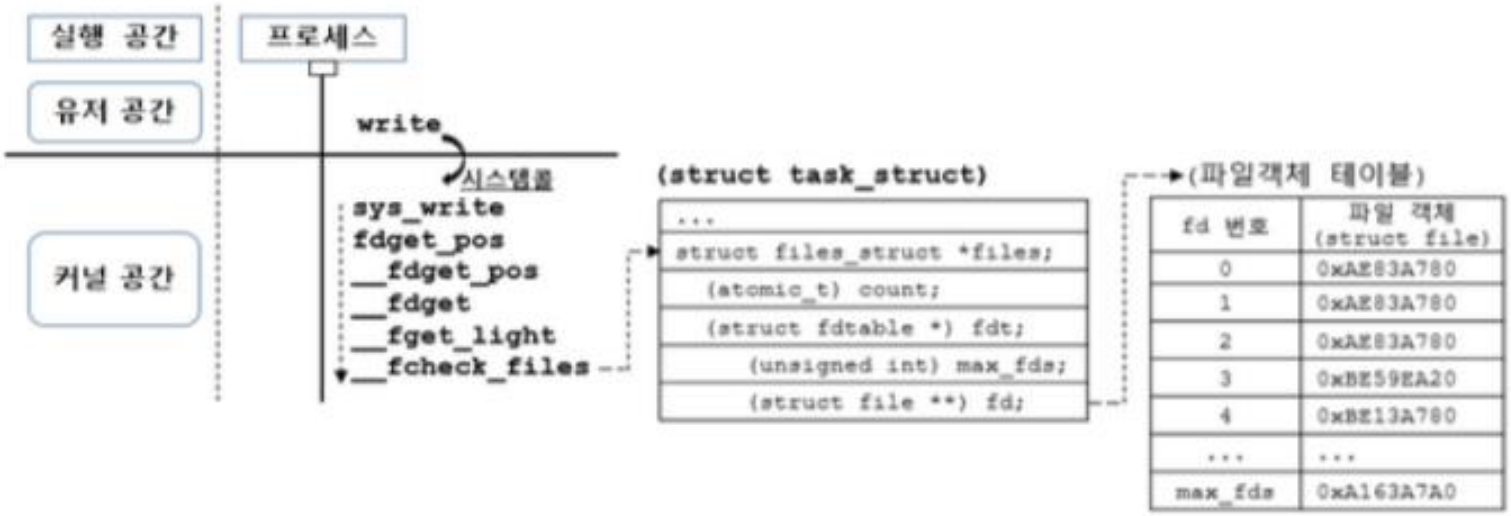
- open() 실행시 커널공간 sys_oepn 실행

4. read()



- 유저 공간에서 read()호출시 시스템 콜 발생 → 커널공간 sys_read 실행

5. write()



- 유저 공간에서 write()호출시 시스템 콜 발생 → 커널공간 sys_write 실행

- open(),read(),write()모두 files_operation 함수 포인터로서 동작