Linux Kernel

# File System Tracing

## Hands-on

이호연, 송태웅

## Disclaimer !

본 세션은 **Tracing** 툴을 이용해

**File System**의 전반적 흐름을 살펴봅니다.

# Tracing 실습 도구소개

1. BPF: 커널함수 추적 도구
2. uftrace: 함수호출(실행흐름) 도구
3. btrace: Live block device tracing

...

# BPF 란 ?

1. Berkeley Packet Filter since 1992
2. **Kernel Infrastructure**
   - **Interpreter** in-kernel virtual machine
   - **Hook points** in-kernel callback point
   - **Map**
   - **Helper**

# BPF 란 ?

1. Berkeley Packet Filter since 1992
2. Kernel Infrastructure

# BPF 란 ?

“Safe dynamic programs and tools”

"런타임중 안전하게 커널코드를 삽입하는 기술"

# BPF 란 ?

“Safe dynamic programs and tools”

"런타임중 안전하게 커널코드를 삽입하는 기술"

**Kernel Tracing !!** (+ network(xdp))

# Introduction to uftrace

# uftrace

**Function tracer for C/C++ programs**

- created by Namhyung Kim
  - LG Electronics open-source contribution team
  - Linux kernel developer (since 2010)
    - perf, ftrace, …

- inspired by ftrace framework in the kernel
- **record** and **replay** model

Search or jump to… /

Pull requests Issues Marketplace Explore

namhyung / **uftrace**

Unwatch 79 | Unstar 996 | Fork 153

Code | Issues 62 | Pull requests 19 | Wiki | Insights

Function (graph) tracer for user-space https://uftrace.github.io/slide/

trace function tracer

**2,986** commits | **13** branches | **13** releases | **28** contributors | GPL-2.0

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

**namhyung** Merge pull request #575 from gy741/fix_code ··· Latest commit `63f73ed` 2 days ago

| | | |
|---|---|---|
| arch | mcount: Fix Unchecked Return Value in mcount_setup_trampoline() | a month ago |
| check-deps | build: removed stringop_truncation supression | 5 days ago |
| cmds | record: fix GCC8 string truncation warning | 6 days ago |

README.md




# uftrace

The uftrace tool is to trace and analyze execution of a program written in C/C++. It was heavily inspired by the ftrace framework of the Linux kernel (especially function graph tracer) and supports userspace programs. It supports various kind of commands and filters to help analysis of the program execution and performance.

```
$ sudo uftrace -k a.out
[sudo] password for honggyu:
Hello World!
# DURATION    TID     FUNCTION
   1.116 us [ 6267] | __monstartup();
   0.603 us [ 6267] | __cxa_atexit();
            [ 6267] | main() {
            [ 6267] |   printf() {
            [ 6267] |     sys_newfstat() {
```

```
$ gcc -pg -o fibonacci tests/s-fibonacci.c

$ uftrace -A fib@arg1 -R fib@retval fibonacci 5
# DURATION     TID     FUNCTION
   0.633 us [ 2851] | __monstartup();
   0.480 us [ 2851] | __cxa_atexit();
            [ 2851] | main() {
   0.546 us [ 2851] |   atoi();
            [ 2851] |   fib(5) {
            [ 2851] |     fib(4) {
            [ 2851] |       fib(3) {
   1.146 us [ 2851] |         fib(2) = 1;
   0.077 us [ 2851] |         fib(1) = 1;
   1.823 us [ 2851] |       } = 2; /* fib */
   0.062 us [ 2851] |       fib(2) = 1;
   2.199 us [ 2851] |     } = 3; /* fib */
            [ 2851] |     fib(3) {
   0.061 us [ 2851] |       fib(2) = 1;
   0.067 us [ 2851] |       fib(1) = 1;
   0.474 us [ 2851] |     } = 2; /* fib */
   3.317 us [ 2851] |   } = 5; /* fib */
   4.343 us [ 2851] | } /* main */
```

**uftrace**

**“C/C++ execution flow”**

```
# Default == record + replay

$ uftrace -A fib@arg1 -R fib@retval fibonacci 5
# DURATION     TID     FUNCTION
   0.633 us [ 2851] | __monstartup();
   0.480 us [ 2851] | __cxa_atexit();
            [ 2851] | main() {
   0.546 us [ 2851] |   atoi();
            [ 2851] |   fib(5) {
            [ 2851] |     fib(4) {
            [ 2851] |       fib(3) {
   1.146 us [ 2851] |         fib(2) = 1;
   0.077 us [ 2851] |         fib(1) = 1;
   1.823 us [ 2851] |       } = 2; /* fib */
   0.062 us [ 2851] |       fib(2) = 1;
   2.199 us [ 2851] |     } = 3; /* fib */
            [ 2851] |     fib(3) {
   0.061 us [ 2851] |       fib(2) = 1;
   0.067 us [ 2851] |       fib(1) = 1;
   0.474 us [ 2851] |     } = 2; /* fib */
   3.317 us [ 2851] |   } = 5; /* fib */
   4.343 us [ 2851] | } /* main */
```

uftrace

“C/C++ execution flow”

```
# Default == record + replay

$ uftrace -A fib@arg1 -R fib@retval fibonacci 5
# DURATION     TID     FUNCTION
   0.633 us [ 2851] | __monstartup();
   0.480 us [ 2851] | __cxa_atexit();
            [ 2851] | main() {
   0.546 us [ 2851] |   atoi();
            [ 2851] |   fib(5) {
            [ 2851] |     fib(4) {
            [ 2851] |       fib(3) {
   1.146 us [ 2851] |         fib(2) = 1;
   0.077 us [ 2851] |         fib(1) = 1;
   1.823 us [ 2851] |       } = 2; /* fib */
   0.062 us [ 2851] |       fib(2) = 1;
   2.199 us [ 2851] |     } = 3; /* fib */
            [ 2851] |     fib(3) {
   0.061 us [ 2851] |       fib(2) = 1;
   0.067 us [ 2851] |       fib(1) = 1;
   0.474 us [ 2851] |     } = 2; /* fib */
   3.317 us [ 2851] |   } = 5; /* fib */
   4.343 us [ 2851] | } /* main */
```

uftrace

“C/C++ each function

Execution time”

```
# Default == record + replay

$ uftrace -A fib@arg1 -R fib@retval fibonacci 5
# DURATION     TID     FUNCTION
   0.633 us [ 2851] | __monstartup();
   0.480 us [ 2851] | __cxa_atexit();
            [ 2851] | main() {
   0.546 us [ 2851] |   atoi();
            [ 2851] |   fib(5) {
            [ 2851] |     fib(4) {
            [ 2851] |       fib(3) {
   1.146 us [ 2851] |         fib(2) = 1;
   0.077 us [ 2851] |         fib(1) = 1;
   1.823 us [ 2851] |       } = 2; /* fib */
   0.062 us [ 2851] |       fib(2) = 1;
   2.199 us [ 2851] |     } = 3; /* fib */
            [ 2851] |     fib(3) {
   0.061 us [ 2851] |       fib(2) = 1;
   0.067 us [ 2851] |       fib(1) = 1;
   0.474 us [ 2851] |     } = 2; /* fib */
   3.317 us [ 2851] |   } = 5; /* fib */
   4.343 us [ 2851] | } /* main */
```

uftrace

“Arguments”

based on
Function Call Trace

```
# Default == record + replay

$ uftrace -A fib@arg1 -R fib@retval fibonacci 5
# DURATION     TID     FUNCTION
   0.633 us [ 2851] | __monstartup();
   0.480 us [ 2851] | __cxa_atexit();
            [ 2851] | main() {
   0.546 us [ 2851] |   atoi();
            [ 2851] |   fib(5) {
            [ 2851] |     fib(4) {
            [ 2851] |       fib(3) {
   1.146 us [ 2851] |         fib(2) = 1;
   0.077 us [ 2851] |         fib(1) = 1;
   1.823 us [ 2851] |       } = 2; /* fib */
   0.062 us [ 2851] |       fib(2) = 1;
   2.199 us [ 2851] |     } = 3; /* fib */
            [ 2851] |     fib(3) {
   0.061 us [ 2851] |       fib(2) = 1;
   0.067 us [ 2851] |       fib(1) = 1;
   0.474 us [ 2851] |     } = 2; /* fib */
   3.317 us [ 2851] |   } = 5; /* fib */
   4.343 us [ 2851] | } /* main */
```

uftrace

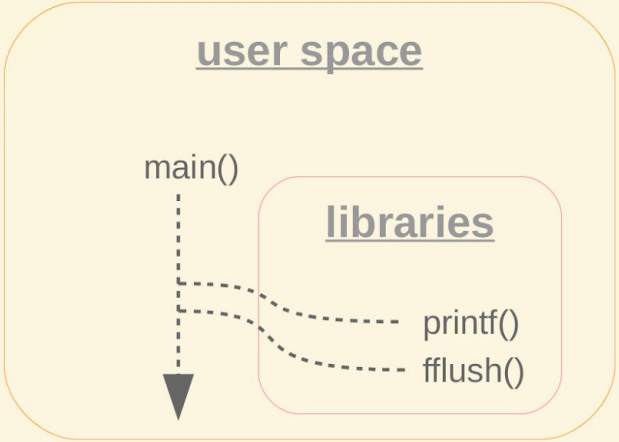
“Return values”

based on
Function Call Trace

uftrace can trace User + Lib + Kernel

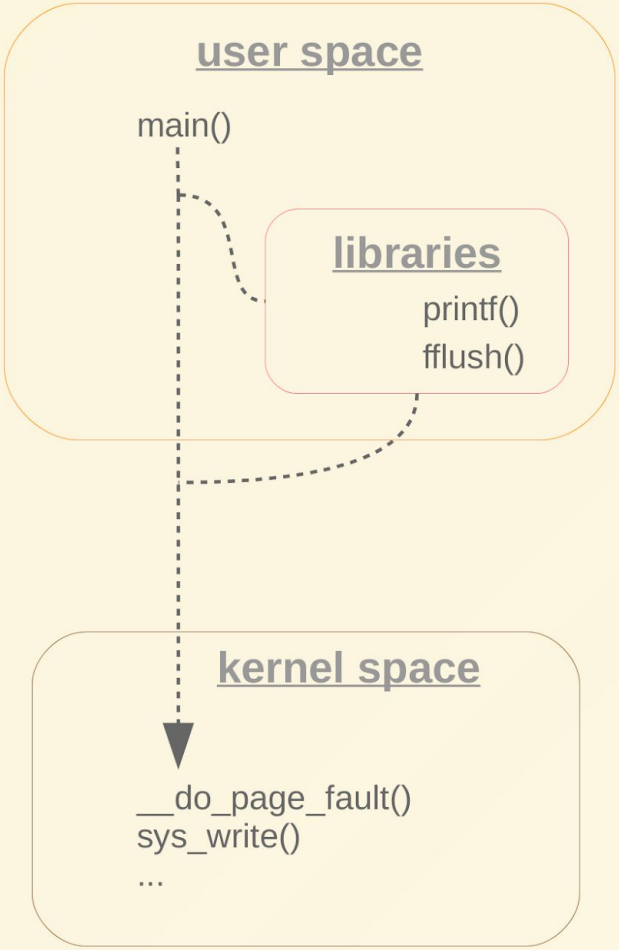
showing the execution flow

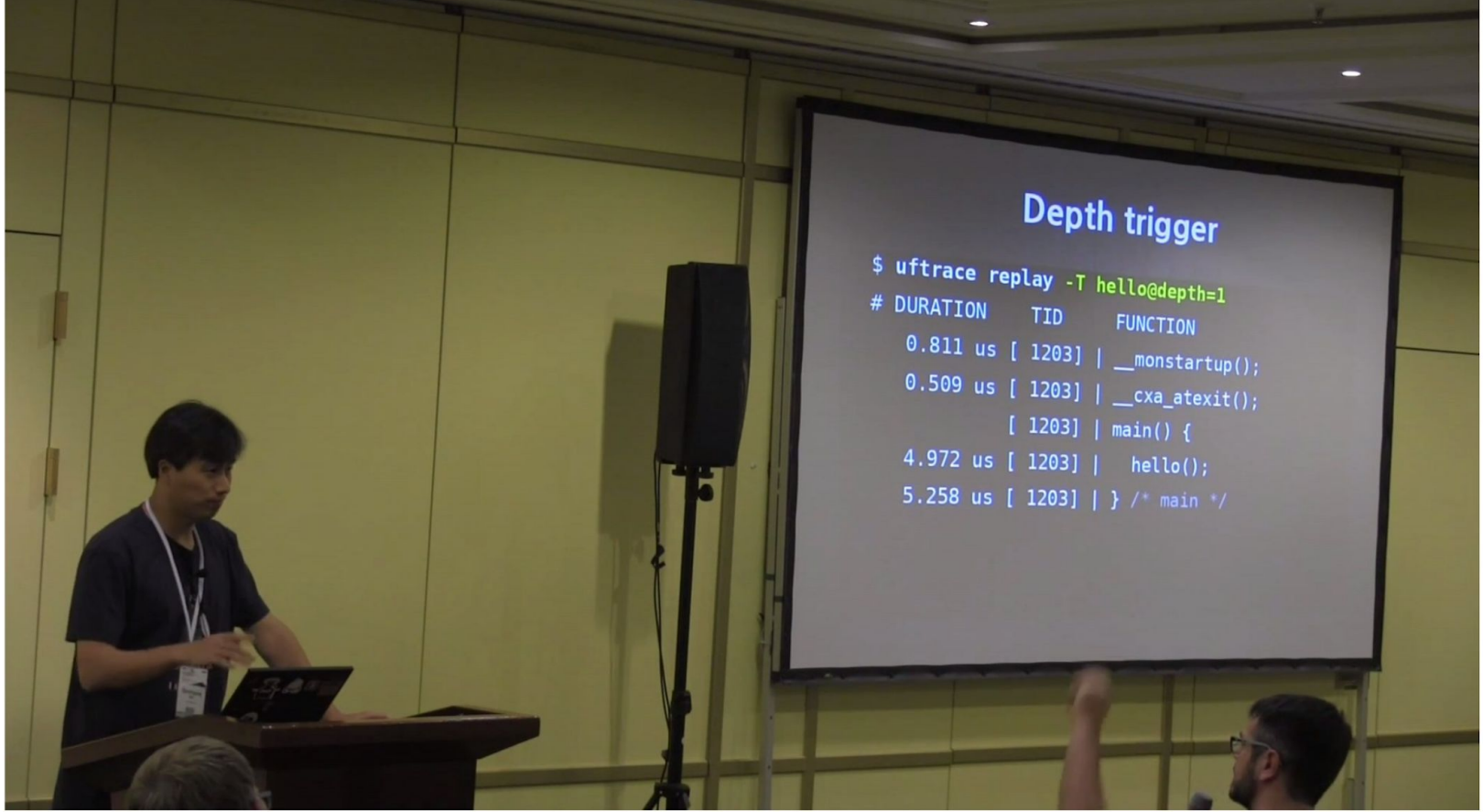
```
$ uftrace record hello
Hello OSSEU17 !!

$ uftrace replay
# DURATION    TID     FUNCTION
   0.710 us [13588] | __monstartup();
   0.713 us [13588] | __cxa_atexit();
            [13588] | main() {
   4.107 us [13588] |   printf();
   4.046 us [13588] |   fflush();
   8.815 us [13588] | } /* main */
```



```
$ uftrace record -k hello
Hello OSSEU17 !!

$ uftrace replay
# DURATION     TID     FUNCTION
   1.060 us [13565] | __monstartup();
   1.113 us [13565] | __cxa_atexit();
            [13565] | main() {
            [13565] |   printf() {
   3.173 us [13565] |     sys_newfstat();
   6.107 us [13565] |     __do_page_fault();
  17.713 us [13565] |   } /* printf */
            [13565] |   fflush() {
   7.198 us [13565] |     sys_write();
  12.270 us [13565] |   } /* fflush */
  30.661 us [13565] | } /* main */
```

**Integrated tracer !**






http://tracingsummit.org/wiki/TracingSummit2016uftrace

https://youtu.be/U3hTR6-pWu0

cppcon | 2016
THE C++ CONFERENCE • BELLEVUE, WASHINGTON

HONGGYU KIM

uftrace: A function graph tracer for C/C++ userspace programs

CppCon.org

```
$ gcc -pg test.c

$ uftrace a.out
# DURATION     TID      FUNCTION
   0.531 us [21315] | __monstartup();
   0.435 us [21315] | __cxa_atexit();
            [21315] | main() {
            [21315] |   foo() {
   0.134 us [21315] |     bar();
   0.564 us [21315] |   } /* foo */
   0.890 us [21315] | } /* main */
```

https://cppcon.org/ https://youtu.be/LNav5qvyK7I

https://github.com/CppCon/CppCon2016/blob/master/Lightning%20Talks%20and%20Lunch%20Sessions/uftrace%20-%20A%20function%20graph%20tracer%20C%20C%2B%2B%20userspace%20programs/uftrace%20-%20A%20function%20graph%20tracer%20C%20C%2B%2B%20userspace%20programs%20-%20Namhyung%20Kim%20and%20Honggyu%20Kim%20-%20CppCon%202016.pdf

# 시나리오 ?

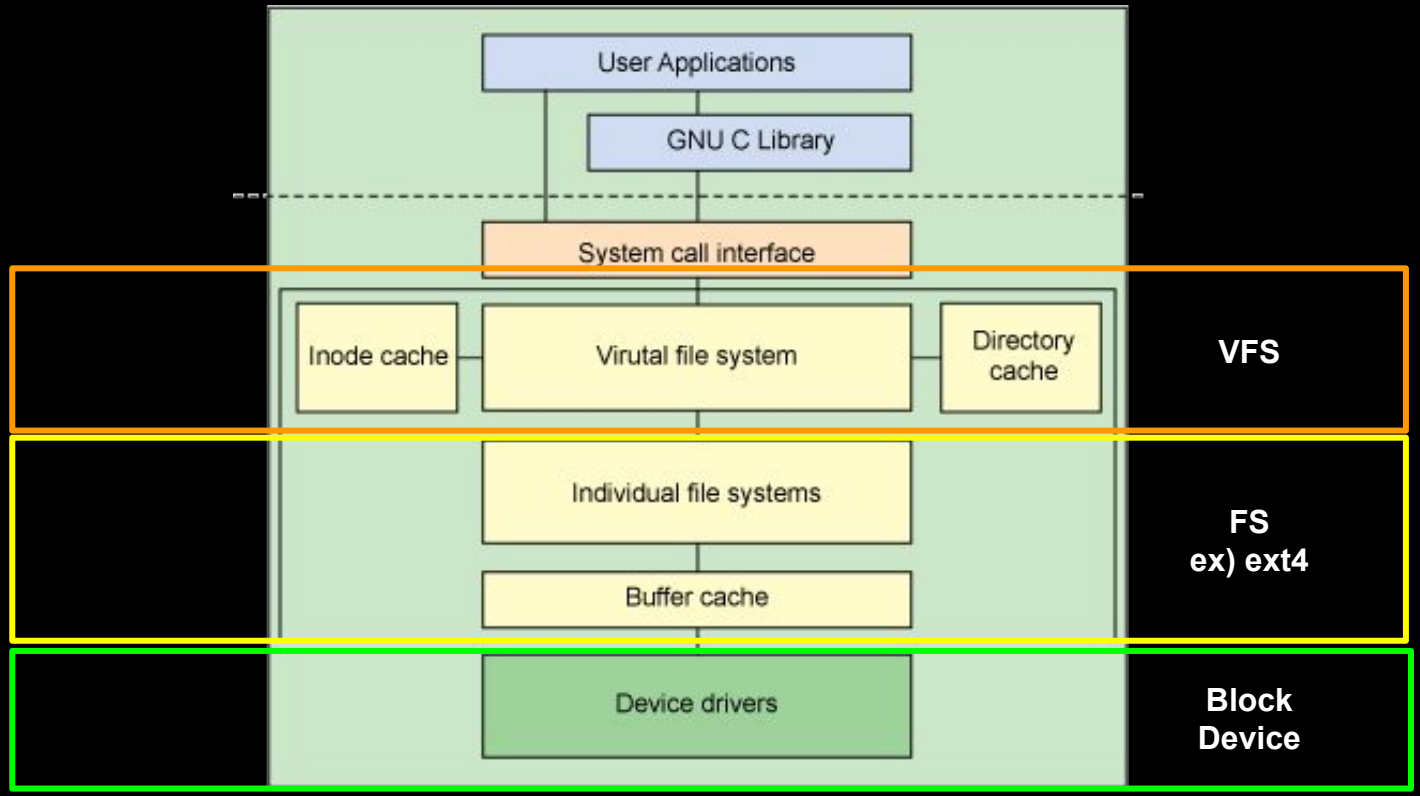
파일은 어떻게 열고 쓰고 읽나요?

# Scenario ?

1 write-sync.c

```c
#include <stdio.h>
#include <unistd.h>

void main()
{
    FILE *f = fopen("kosslab.txt","w");
    fprintf(f, "Hello World\n");
    fclose(f);
    sync();
}
```

# High-level architecture
## (Linux Filesystem components)



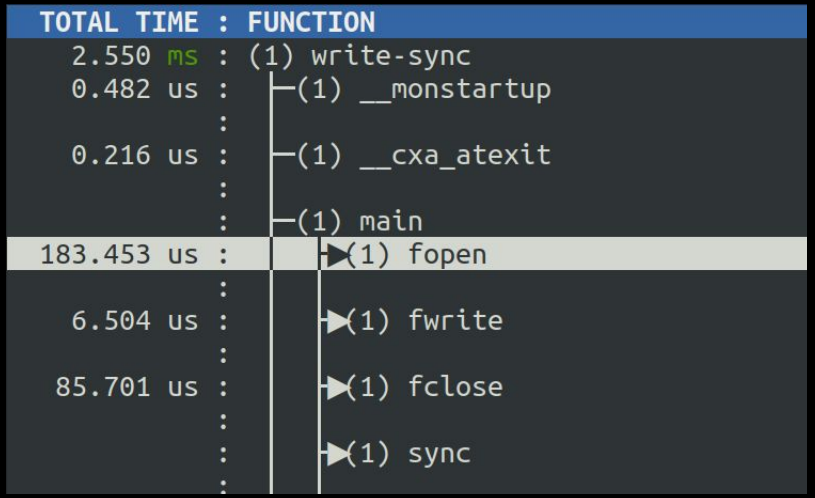

https://www.ibm.com/developerworks/library/l-linux-filesystem/

# Index

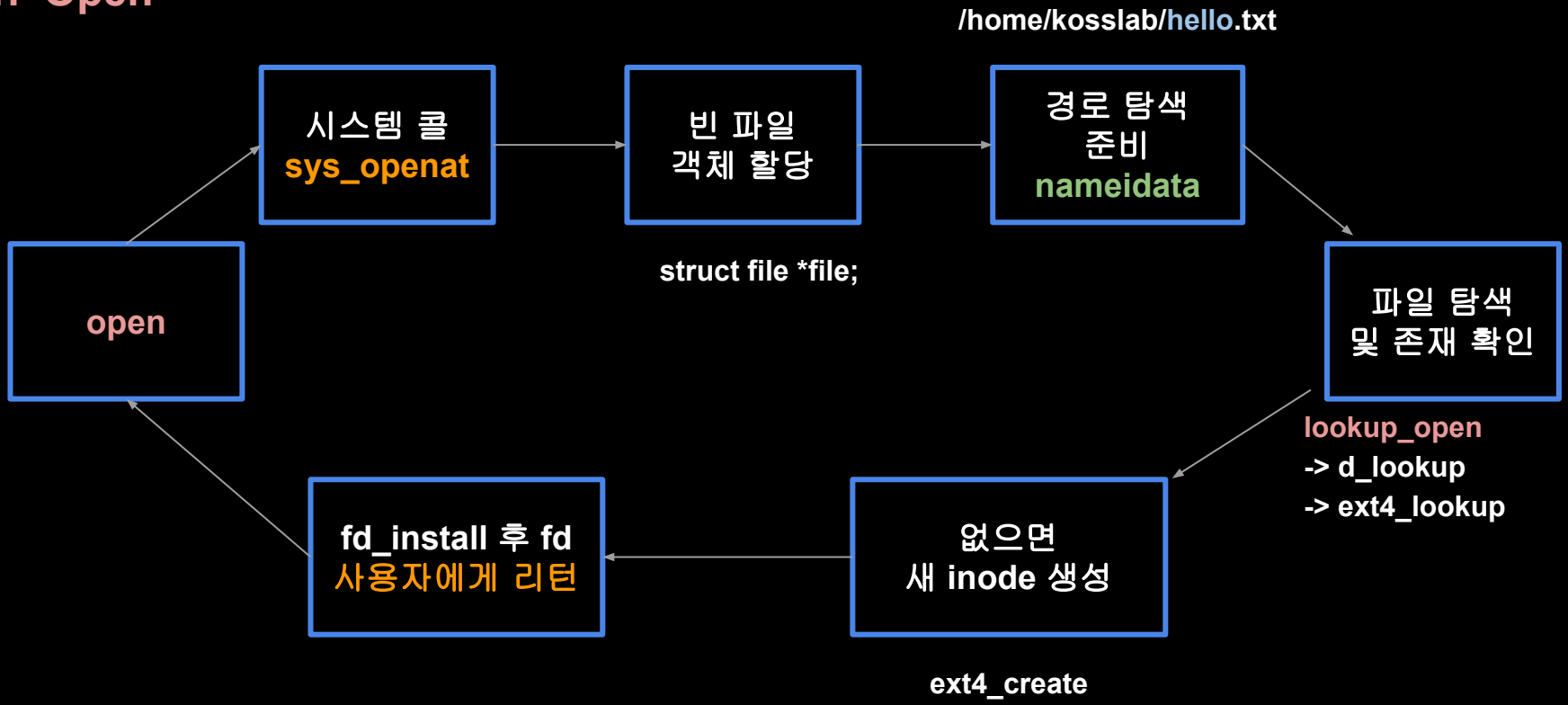
1. OPEN
2. WRITE
3. SYNC
4. READ

# Index

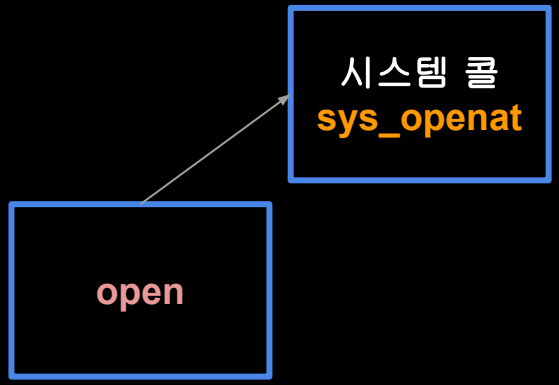
1. OPEN
2. WRITE
3. SYNC
4. READ



# 1. Open


/home/kosslab/hello.txt
시스템 콜
sys_openat
빈 파일
객체 할당
경로 탐색
준비
nameidata
struct file *file;
open
파일 탐색
및 존재 확인
lookup_open
-> d_lookup
-> ext4_lookup
fd_install 후 fd
사용자에게 리턴
없으면
새 inode 생성
ext4_create

# 1. Open




int dfd;
파일 열고자 하는 위치

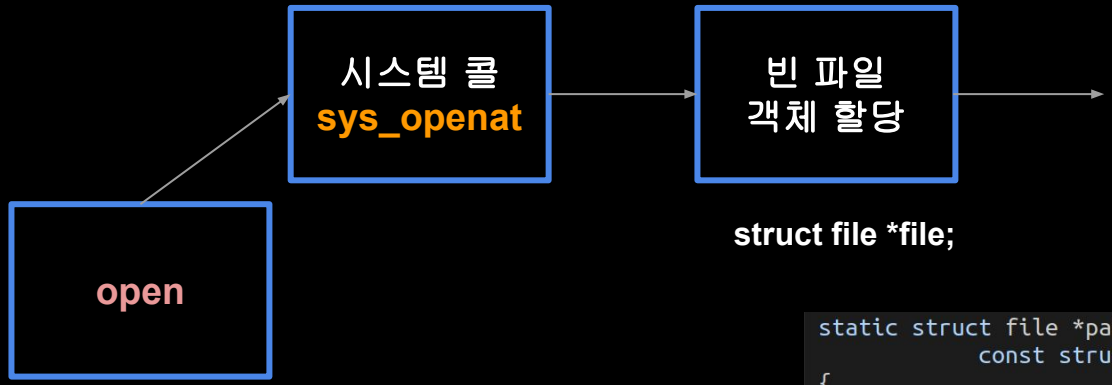
char *__user;
열고자 하는 파일 스트링

```
SYSCALL_DEFINE4(openat, int, dfd, const char __user *, filename, int, flags,
        umode_t, mode)
{
    if (force_o_largefile())
        flags |= O_LARGEFILE;

    return do_sys_open(dfd, filename, flags, mode);
}
```

sys_openat

"~/git/linux/fs/open.c" 1211 lines --89%--

# 1. Open




**struct file *file;**

```
(1) main
  (1) fopen
  (1) __x64_sys_openat
  (1) do_sys_open
    (1) getname
    (1) get_unused_fd_flags
    (1) do_filp_open
      (1) path_openat
        (1) alloc_empty_file
        (1) path_init
        (1) link_path_walk.part.43
        (1) complete_walk
```

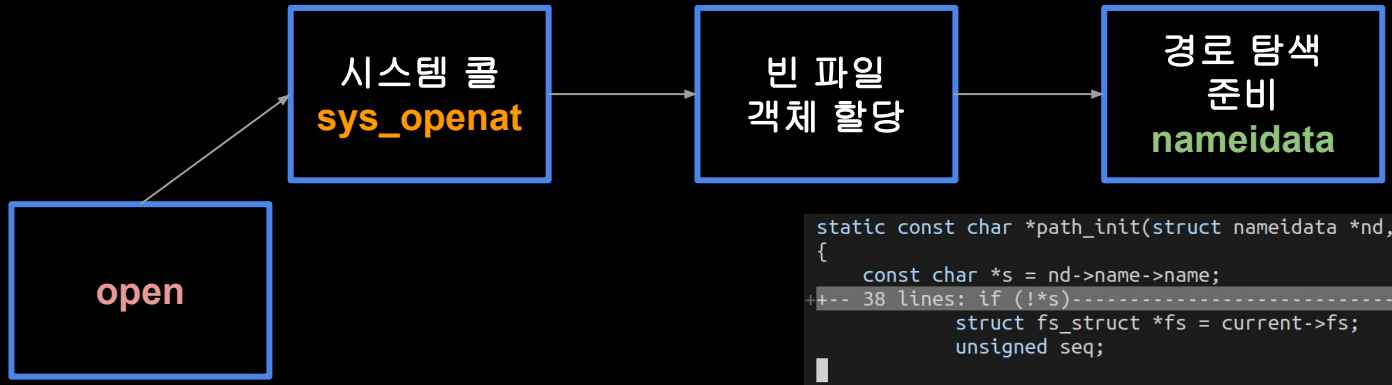
```
static struct file *path_openat(struct nameidata *nd,
            const struct open_flags *op, unsigned flags)
{
    struct file *file;
    int error;

    file = alloc_empty_file(op->open_flag, current_cred());
+----  7 lines: if (IS_ERR(file))-------------------------------
    } else {
        const char *s = path_init(nd, flags);
        while (!(error = link_path_walk(s, nd)) &&
            (error = do_last(nd, file, op)) > 0) {
            nd->flags &= ~(LOOKUP_OPEN|LOOKUP_CREATE|LOOKUP_EXCL);
            s = trailing_symlink(nd);
        }
        terminate_walk(nd);
```

**path_openat**

**"~/git/linux/fs/namei.c" 4863 lines --72%--**

# 1. Open

/home/kosslab/hello.txt




```
(1) main
  (1) fopen
  (1) __x64_sys_openat
  (1) do_sys_open
    (1) getname
    (1) get_unused_fd_flags
    (1) do_filp_open
      (1) path_openat
        (1) alloc_empty_file
        (1) path_init
        (1) link_path_walk.part.43
        (1) complete_walk
```

```
static const char *path_init(struct nameidata *nd, unsigned flags)
{
    const char *s = nd->name->name;
++-- 38 lines: if (!*s)-----------------------------------------------------
            struct fs_struct *fs = current->fs;
            unsigned seq;

            do {
                seq = read_seqcount_begin(&fs->seq);
                nd->path = fs->pwd;
                nd->inode = nd->path.dentry->d_inode;
                nd->seq = __read_seqcount_begin(&nd->path.dentry->d_seq);
```

path_init

"~/git/linux/fs/namei.c" 4863 lines --45%--

```
if (name[0] == '.') switch (hashlen_len(hash_len)) {
    case 2:
        if (name[1] == '.') {
            type = LAST_DOTDOT;
            nd->flags |= LOOKUP_JUMPED;
        }
        break;
    case 1:
        type = LAST_DOT;
```

link_path_walk

# 1. Open

/home/kosslab/hello.txt

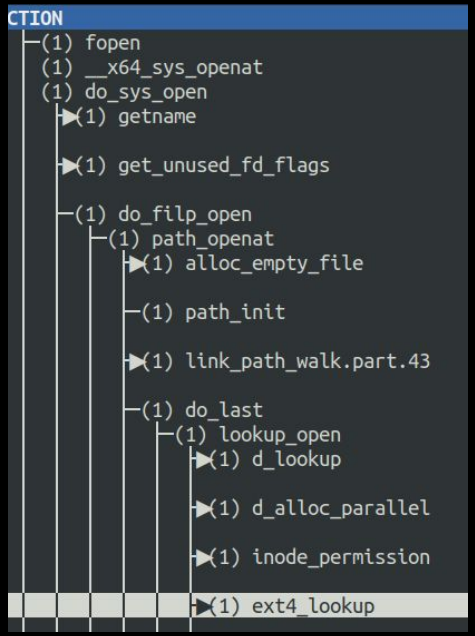
시스템 콜
sys_openat

빈 파일
객체 할당

경로 탐색
준비
nameidata

파일 탐색
및 존재 확인

struct file *file;




lookup_open
-> d_lookup
-> ext4_lookup

d_lookup()

빠른 탐색을 위해 VFS 레벨에서 관리하는
dentry cache 탐색

"~/git/linux/fs/dcache.c" 3119 lines --70%--

ext4_lookup()

실제 대상 디스크 안에 파일 이 존재하는지 확인

"~/git/linux/fs/ext4/namei.c" 3863 lines --39%--

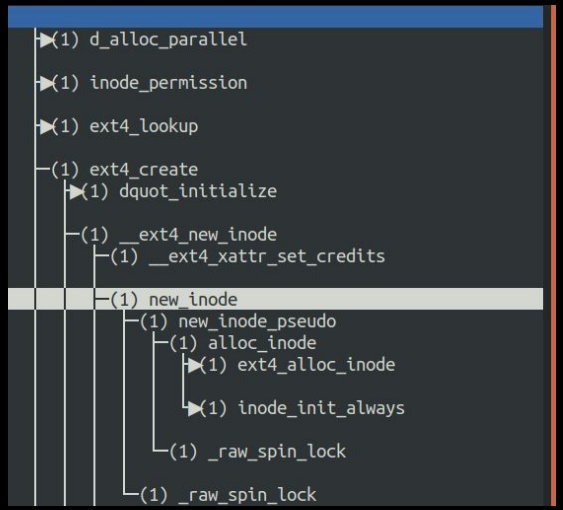
# 1. Open

/home/kosslab/hello.txt

open → 시스템 콜 sys_openat → 빈 파일 객체 할당 → 경로 탐색 준비 nameidata → 파일 탐색 및 존재 확인 → 없으면 새 inode 생성

struct file *file;

ext4_create



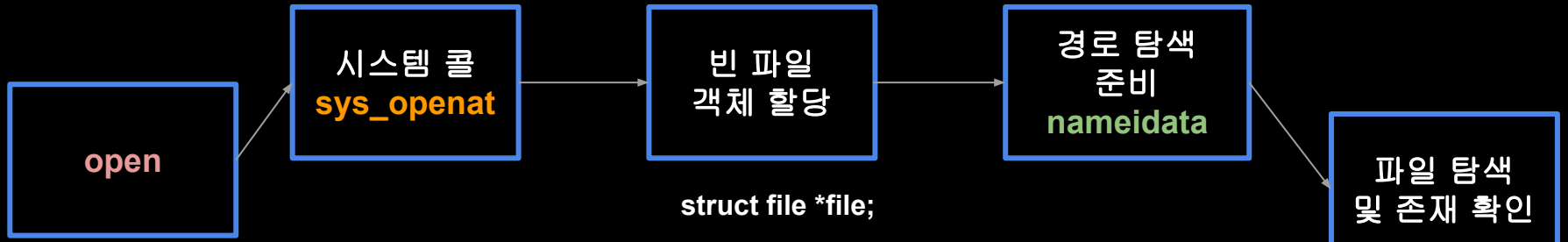

```
1 namei.c
 *
 * If the create succeeds, we fill in the inode information
 * with d_instantiate().
 */
static int ext4_create(struct inode *dir, struct dentry *dentry, umode_t mode,
                       bool excl)
{
++-- 11 lines: handle_t *handle;-----------------------------------------
        inode = ext4_new_inode_start_handle(dir, mode, &dentry->d_name, 0,
                                            NULL, EXT4_HT_DIR, credits);
++--  3 lines: handle = ext4_journal_current_handle();-------------------
        inode->i_op = &ext4_file_inode_operations;
        inode->i_fop = &ext4_file_operations;
        ext4_set_aops(inode);
        err = ext4_add_nondir(handle, dentry, inode);
++--  8 lines: if (!err && IS_DIRSYNC(dir))------------------------------
```

ext4_create

"~/git/linux/fs/ext4/namei.c" 3863 lines --63%--

# 1. Open

/home/kosslab/hello.txt



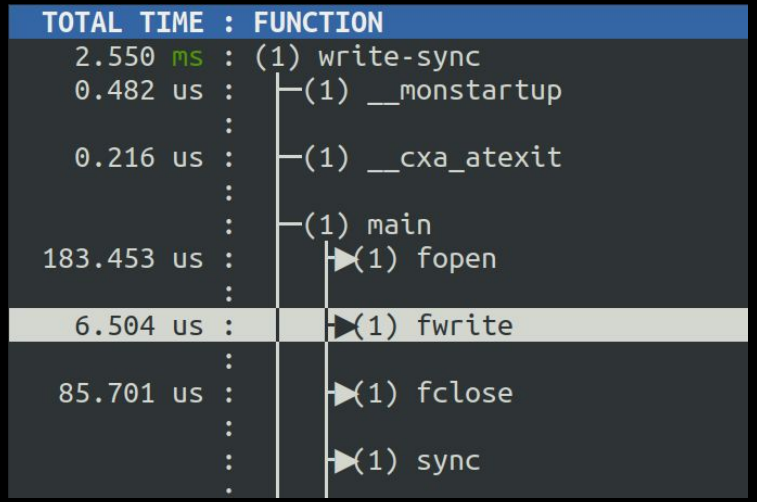

```
long do_sys_open(int dfd, const char __user *filename, int flags, umode_t mode)
{
+-- 11 lines: struct open_flags op;----------------------------------------------
    fd = get_unused_fd_flags(flags);
    if (fd >= 0) {
        struct file *f = do_filp_open(dfd, tmp, &op);
+--  4 lines: if (IS_ERR(f)) {-----------------------------------------------------
            fsnotify_open(f);
            fd_install(fd, f);
        }
    }
    putname(tmp);
    return fd;
}
```

do_sys_open -> fd_install()

"~/git/linux/fs/open.c" 1211 lines --87%--

# Index

1. OPEN
2. WRITE
3. SYNC
4. READ



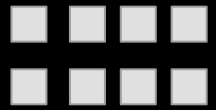
# 2. Write - 파일을 쓰는 과정

inode

Disk Sectors

Backing Device

Blocks



Block Device

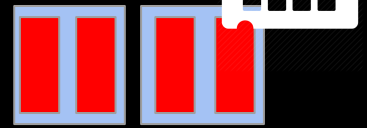
```
struct inode {
    umode_t         i_mode;
    unsigned short      i_opflags;
    kuid_t          i_uid;
    kgid_t          i_gid;
+---  7 lines: unsigned int  i_flags;-----
    const struct inode_operations   *i_op;
    struct super_block  *i_sb;
    struct address_space    *i_mapping;
+---  6 lines: #ifdef CONFIG_SECURITY-----
    unsigned long       i_ino;
+-- 81 lines: Filesystems may only read i_
} __randomize_layout;
```

```
struct address_space {
    struct inode        *host;
    struct xarray       i_pages;
+--  7 lines: gfp_t      gfp_mask;----------
    const struct address_space_operations *a_ops;
+--  5 lines: unsigned long  flags;----------
} __attribute__((aligned(sizeof(long)))) __random
```

[a_ops]
writepages
readpages

Flush Thread
sync
fsync
fdatasync

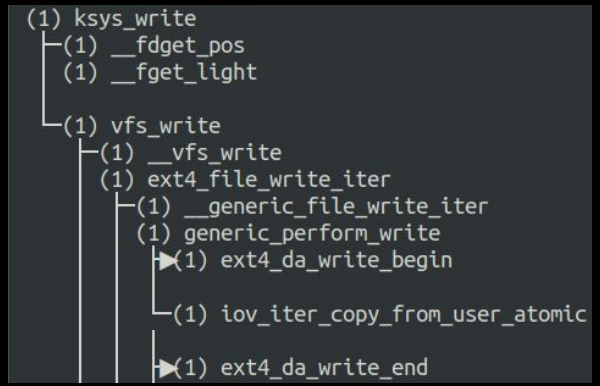
Page Cache

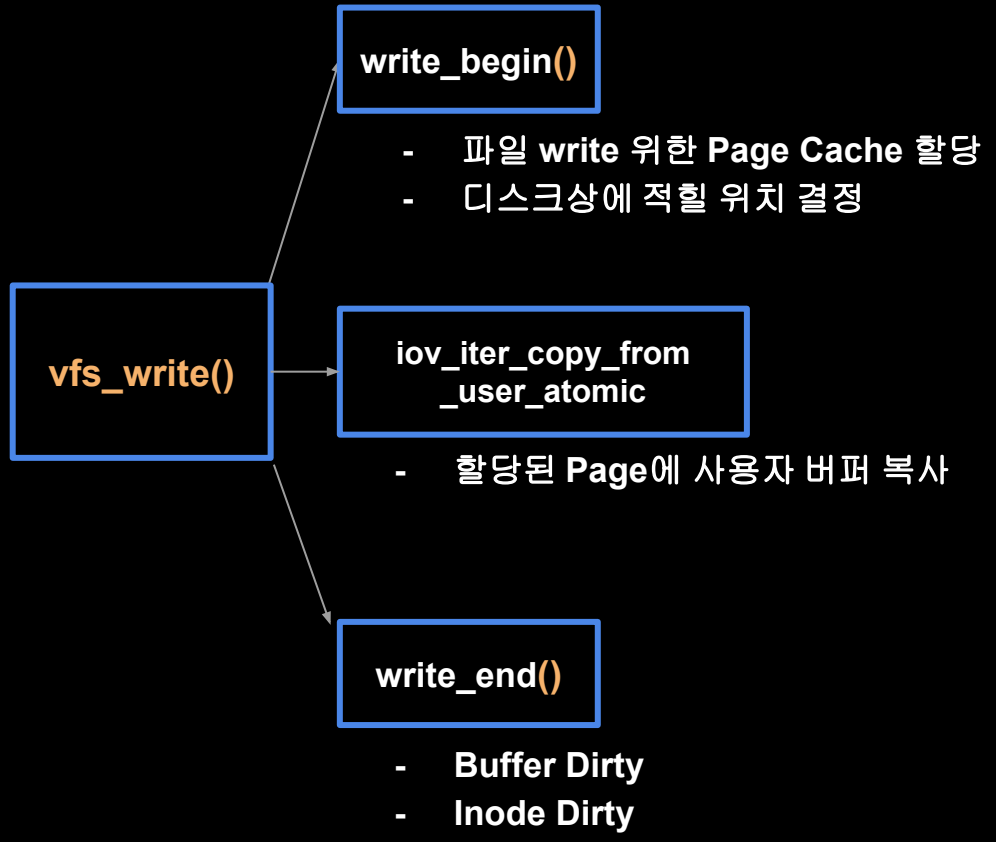


Page Page

write_begin
write_end
[a_ops]

# 2. Write - vfs_write



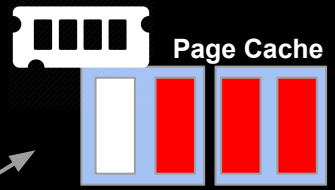



# 2. Write - write_begin

inode

```
struct inode {
    umode_t         i_mode;
    unsigned short      i_opflags;
    kuid_t          i_uid;
    kgid_t          i_gid;
+---  7 lines: unsigned int  i_flags;-----
    const struct inode_operations   *i_op;
    struct super_block  *i_sb;
    struct address_space    *i_mapping;
+---  6 lines: #ifdef CONFIG_SECURITY-----
    unsigned long       i_ino;
+-- 81 lines: Filesystems may only read i_
} __randomize_layout;
```

```
struct address_space {
    struct inode        *host;
    struct xarray       i_pages;
+--  7 lines: gfp_t      gfp_mask;----------
    const struct address_space_operations *a_ops;
+--  5 lines: unsigned long  flags;--------
} __attribute__((aligned(sizeof(long)))) __random
```




Yes

Page Cache 해당
페이지 존재?

No

새 페이지 할당


```
(1) ext4_da_write_begin
  (1) ext4_nonda_switch

  (1) grab_cache_page_write_begin
    (1) pagecache_get_page
      (1) find_get_entry

      (1) __page_cache_alloc
      (1) alloc_pages_current

      (1) add_to_page_cache_lru

    (1) wait_for_stable_page

  (1) __ext4_journal_start_sb
```

```
struct page *grab_cache_page_write_begin(struct address_space *mapping,
                    pgoff_t index, unsigned flags)
{
    struct page *page;
+--  5 lines: int fgp_flags = FGP_LOCK|FGP_WRITE|FGP_CREAT;----------
    page = pagecache_get_page(mapping, index, fgp_flags,
            mapping_gfp_mask(mapping));
    if (page)
        wait_for_stable_page(page);

    return page;
}
```

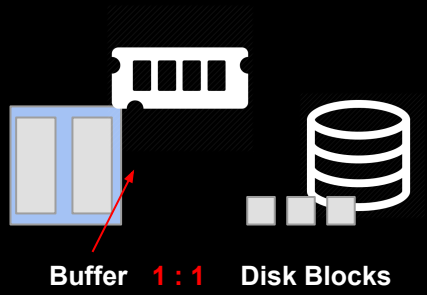
grab_cache_page_write_begin

"~/git/linux/mm/filemap.c" 3332 lines --92%--

# 2. Write - write_begin

```
(1) vfs_write
 ├─(1) __vfs_write
  (1) ext4_file_write_iter
   ├─(1) __generic_file_write_iter
    (1) generic_perform_write
     ├─(1) ext4_da_write_begin
      ├─(1) ext4_nonda_switch
      ├─▶(1) grab_cache_page_write_begin
      ├─▶(1) __ext4_journal_start_sb
      ├─(1) wait_for_stable_page
      └─(1) __block_write_begin
       (1) __block_write_begin_int
        ├─(1) create_page_buffers
         (1) create_empty_buffers
          ├─(1) alloc_page_buffers
           (1) alloc_buffer_head
```

페이지 버퍼? - 디스크에 적을 내용을 블록 단위 구성



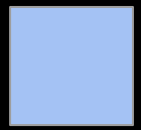

한 페이지 안의 버퍼 수?
= 페이지 사이즈 / 블록 사이즈

ex)
getconf PAGESIZE
blockdev --getbsz /dev/sda2

# 2. Write - write_begin

```
(1) vfs_write
 ├─(1) __vfs_write
  (1) ext4_file_write_iter
   ├─(1) __generic_file_write_iter
    (1) generic_perform_write
     ├─(1) ext4_da_write_begin
      ├─(1) ext4_nonda_switch
      │
      ├─(1) grab_cache_page_write_begin
      │
      ├─(1) __ext4_journal_start_sb
      │
      ├─(1) wait_for_stable_page
      │
      └─(1) __block_write_begin
        (1) __block_write_begin_int
         ├─(1) create_page_buffers
          (1) create_empty_buffers
           ├─(1) alloc_page_buffers
            (1) alloc_buffer_head
```

```
struct buffer_head *alloc_page_buffers(struct page *page,
        unsigned long size, bool retry)
{
    struct buffer_head *bh, *head;
+-- 12 lines: gfp_t gfp = GFP_NOFS | __GFP_ACCOUNT;------
    while ((offset -= size) >= 0) {
        bh = alloc_buffer_head(gfp);
+--  3 lines: if (!bh)----------------------------------
        bh->b_this_page = head;
        bh->b_blocknr = -1;
        head = bh;
        bh->b_size = size;

        /* Link the buffer to its page */
        set_bh_page(bh, page, offset);
    }
```





```
struct buffer_head {
    unsigned long b_state;      /* bu
    struct buffer_head *b_this_page;
    struct page *b_page;        /* th

    sector_t b_blocknr;     /* start
    size_t b_size;          /* size
    char *b_data;           /* pointe
```

Buffer_head

ADDR

**alloc_page_buffers**

할당받은 페이지 위에 버퍼할당

"~/git/linux/fs/buffer.c" 3460 lines --23%--

# 2. Write - write_begin

```
(1) generic_perform_write
  (1) ext4_da_write_begin
    (1) ext4_nonda_switch
    (1) grab_cache_page_write_begin
    (1) __ext4_journal_start_sb
    (1) wait_for_stable_page
    (1) __block_write_begin
    (1) __block_write_begin_int
      (1) create_page_buffers
      (1) ext4_da_get_block_prep
        (1) ext4_es_lookup_extent
        (1) down_read
        (1) ext4_ext_map_blocks
          (1) ext4_find_extent
```

```
int ext4_da_get_block_prep(struct inode *inode, sector_t iblock,
                           struct buffer_head *bh, int create)
{
    struct ext4_map_blocks map;
+-- 13 lines: int ret = 0;-----------------------------------------
    ret = ext4_da_map_blocks(inode, iblock, &map, bh);
+--  3 lines: if (ret <= 0)-------------------------------------------
    map_bh(bh, inode->i_sb, map.m_pblk);
    ext4_update_bh_state(bh, map.m_flags);
+-- 12 lines: if (buffer_unwritten(bh)) {----------------------------
}
```

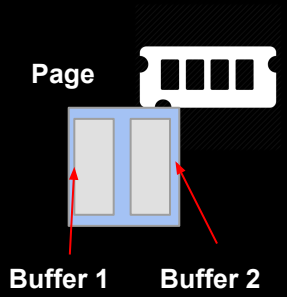
**ext4_da_get_block_prep**

**"~/git/linux/fs/ext4/inode.c" 1959 lines --92%--**

```
static int ext4_da_map_blocks(struct inode *inode, sector_t iblock,
                              struct ext4_map_blocks *map,
                              struct buffer_head *bh)
{
+-- 61 lines: struct extent_status es;-----------------------------
    else if (ext4_test_inode_flag(inode, EXT4_INODE_EXTENTS))
        retval = ext4_ext_map_blocks(NULL, inode, map, 0);
    else
        retval = ext4_ind_map_blocks(NULL, inode, map, 0);
```

**ext4_da_map_blocks**

**"~/git/linux/fs/ext4/inode.c" 1833 lines --92%--**

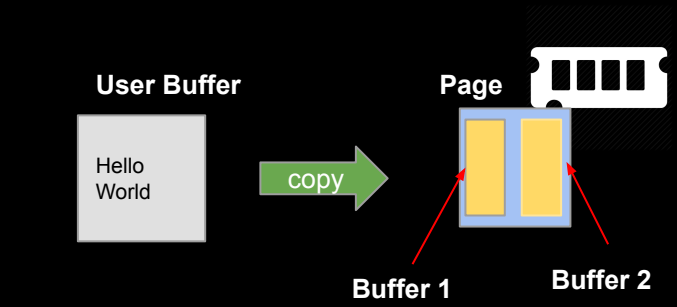



**Buffer_head**

```
struct buffer_head {
    unsigned long b_state;          /* bu
    struct buffer_head *b_this_page;/
    struct page *b_page;            /* th

    sector_t b_blocknr;             /* start
    size_t b_size;                  /* size o
    char *b_data;                   /* pointe
```

# 2. Write - copy buffer

```
(1) ksys_write
  ├─(1) __fdget_pos
  │ (1) __fget_light
  │
  └─(1) vfs_write
    ├─(1) __vfs_write
    │ (1) ext4_file_write_iter
    │   ├─(1) __generic_file_write_iter
    │   │ (1) generic_perform_write
    │   │   ├►(1) ext4_da_write_begin
    │   │   │
    │   │   └─(1) iov_iter_copy_from_user_atomic
    │   │   │
    │   │   ├►(1) ext4_da_write_end
```




```
static ssize_t new_sync_write(struct file *filp,
        const char __user *buf, size_t len, loff_t *ppos)
{
    struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = len };
```

```
size_t iov_iter_copy_from_user_atomic(struct page *page,
        struct iov_iter *i, unsigned long offset, size_t bytes)
{
+-- 10 lines: char *kaddr = kmap_atomic(page), *p = kaddr + offset;---
    iterate_all_kinds(i, bytes, v,
        copyin((p += v.iov_len) - v.iov_len, v.iov_base, v.iov_len),
        memcpy_from_page((p += v.bv_len) - v.bv_len, v.bv_page,
                v.bv_offset, v.bv_len),
        memcpy((p += v.iov_len) - v.iov_len, v.iov_base, v.iov_len)
    )
    kunmap_atomic(kaddr);
    return bytes;
}
EXPORT_SYMBOL(iov_iter_copy_from_user_atomic);
```

**iov_iter_copy_from_user_atomic**

**할당된 Page에 사용자 버퍼 복사**

**"~/git/linux/lib/iov_iter.c" 1643 lines --56%--**

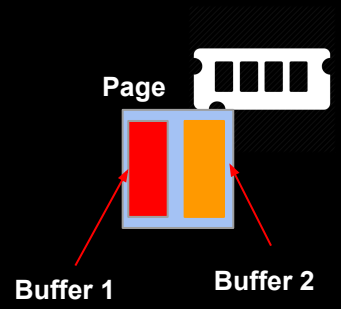
# 2. Write - write_end

```
(1) __fget_light

-(1) vfs_write
  ├─(1) __vfs_write
  │  (1) ext4_file_write_iter
  │  ├─(1) __generic_file_write_iter
  │  │  (1) generic_perform_write
  │  │  ├─▶(1) ext4_da_write_begin
  │  │  │
  │  │  ├─(1) ext4_da_write_end
  │  │  │  ├─(1) generic_write_end
  │  │  │  │  ├─(1) block_write_end
  │  │  │  │  │  (1) __block_commit_write.isra.39
  │  │  │  │  │  (1) mark_buffer_dirty
  │  │  │  │  │  ├─(1) page_mapping
  │  │  │  │  │  │
  │  │  │  │  │  ├─▶(1) __set_page_dirty
  │  │  │  │  │  │
  │  │  │  │  │  └─▶(1) __mark_inode_dirty
  │  │  │  │  │
  │  │  │  │  └─(1) __generic_write_end
  │  │  │  │    ▶(1) __mark_inode_dirty
```

```
void mark_buffer_dirty(struct buffer_head *bh)
{
++-- 10 lines: WARN_ON_ONCE(!buffer_uptodate(bh));----------
    if (buffer_dirty(bh)) {
        smp_mb();
        if (buffer_dirty(bh))
            return;
    }

++---  8 lines: if (!test_set_buffer_dirty(bh)) {----------
                __set_page_dirty(page, mapping, 0);
        }
        unlock_page_memcg(page);
        if (mapping)
            __mark_inode_dirty(mapping->host, I_DIRTY_PAGES);
    }
}
```

**mark_buffer_dirty**

**"~/git/linux/fs/buffer.c" 3459 lines --31%--**




```
struct buffer_head {
    unsigned long b_state;          /* bu
    struct buffer_head *b_this_page;/
    struct page *b_page;            /* th

    sector_t b_blocknr;         /* start
    size_t b_size;              /* size o
    char *b_data;               /* pointe
```

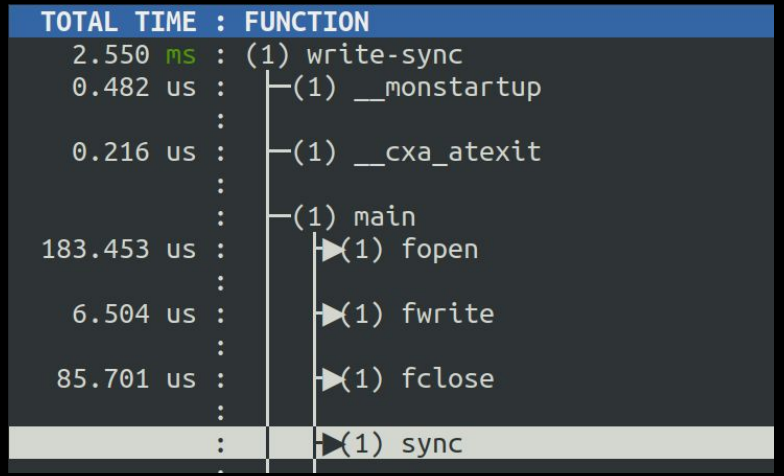
```
static __always_inline int buffer_##name(const struct buffer_head *bh)  \
{                                                                        \
    return test_bit(BH_##bit, &(bh)->b_state);                           \
}
```

**buffer_dirty**

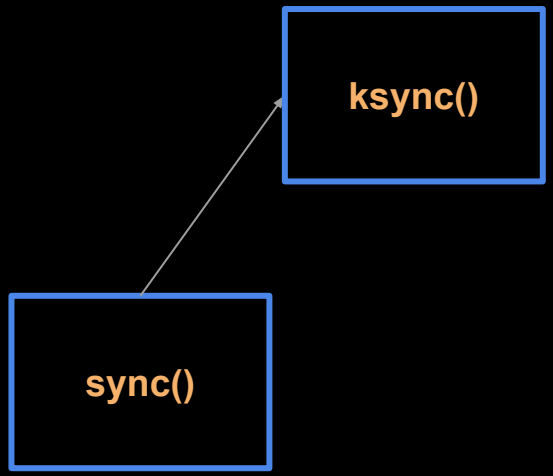
**"~/git/linux/include/linux/buffer_head.h" 97 lines --92%--**

# Index

1. OPEN
2. WRITE
3. SYNC
4. READ



# 3. Sync (1 단계)

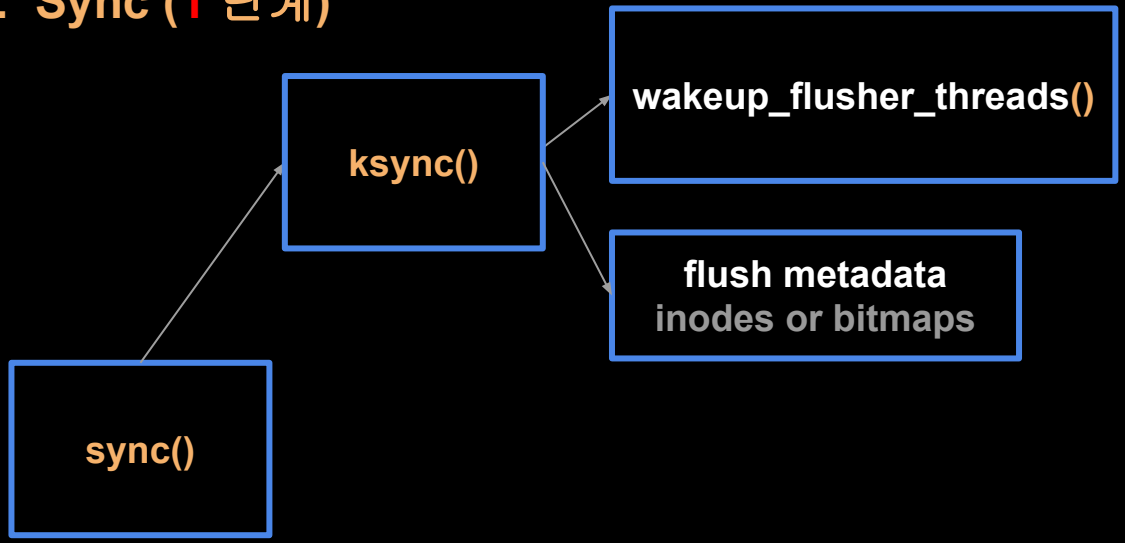


# 3. Sync (1 단계)

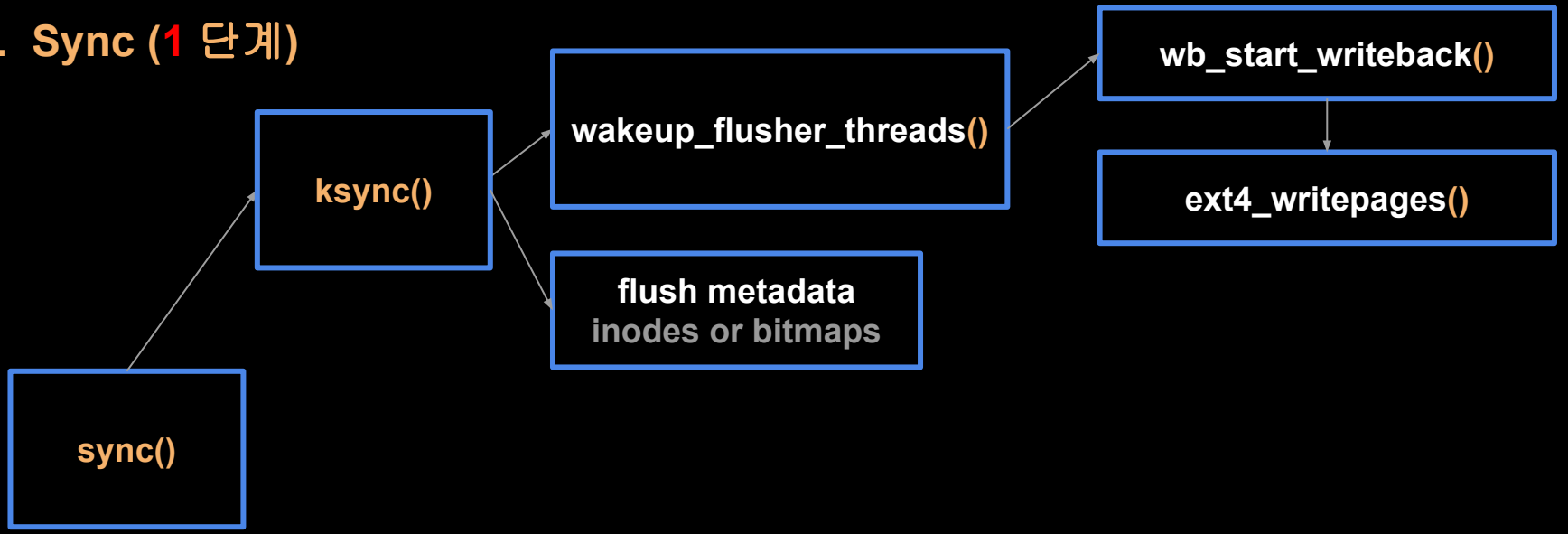
```
 * just write metadata (such as inodes or bitmaps) to block device page cache
 * and do not sync it on their own in ->sync_fs().
 */
void ksys_sync(void)
{
        int nowait = 0, wait = 1;

        wakeup_flusher_threads(WB_REASON_SYNC);
        iterate_supers(sync_inodes_one_sb, NULL);
        iterate_supers(sync_fs_one_sb, &nowait);
        iterate_supers(sync_fs_one_sb, &wait);
        iterate_bdevs(fdatawrite_one_bdev, NULL);
        iterate_bdevs(fdatawait_one_bdev, NULL);
        if (unlikely(laptop_mode))
                laptop_sync_completion();
}
```

**git/linux/fs/sync.c**

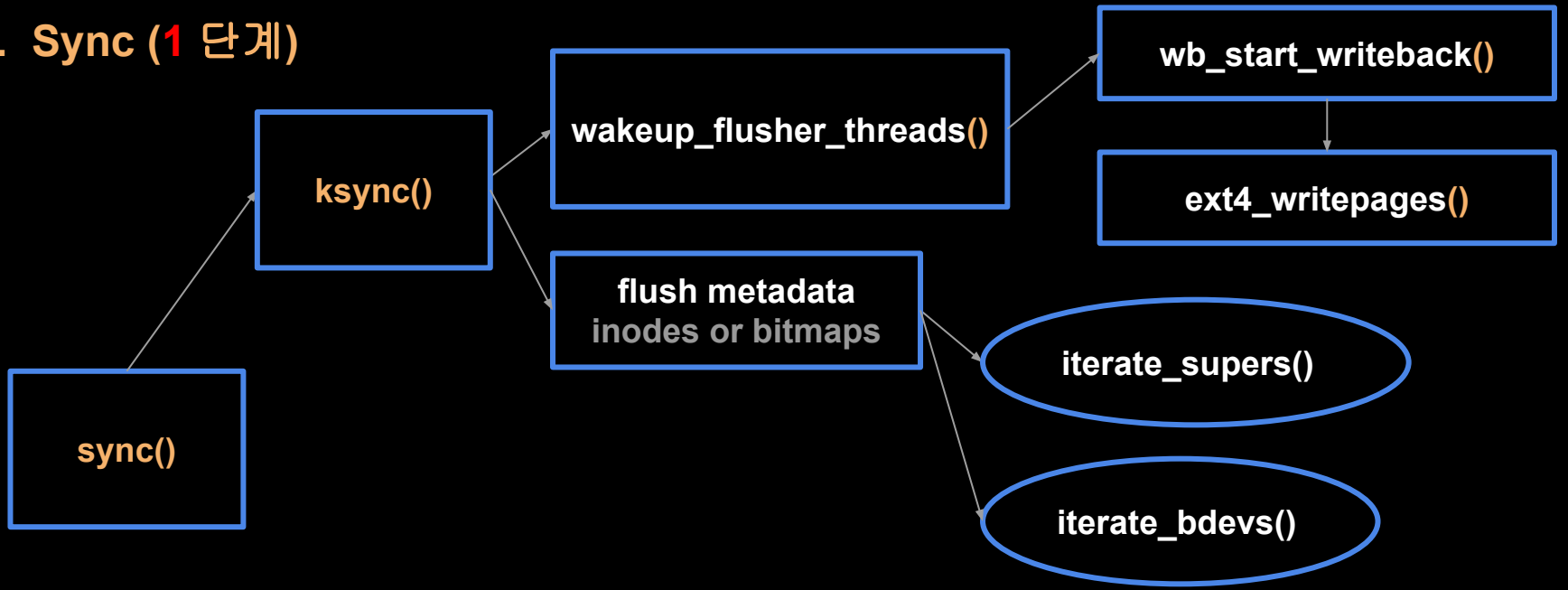
# 3. Sync (1 단계)


ksync()
wakeup_flusher_threads()
flush metadata
inodes or bitmaps
sync()

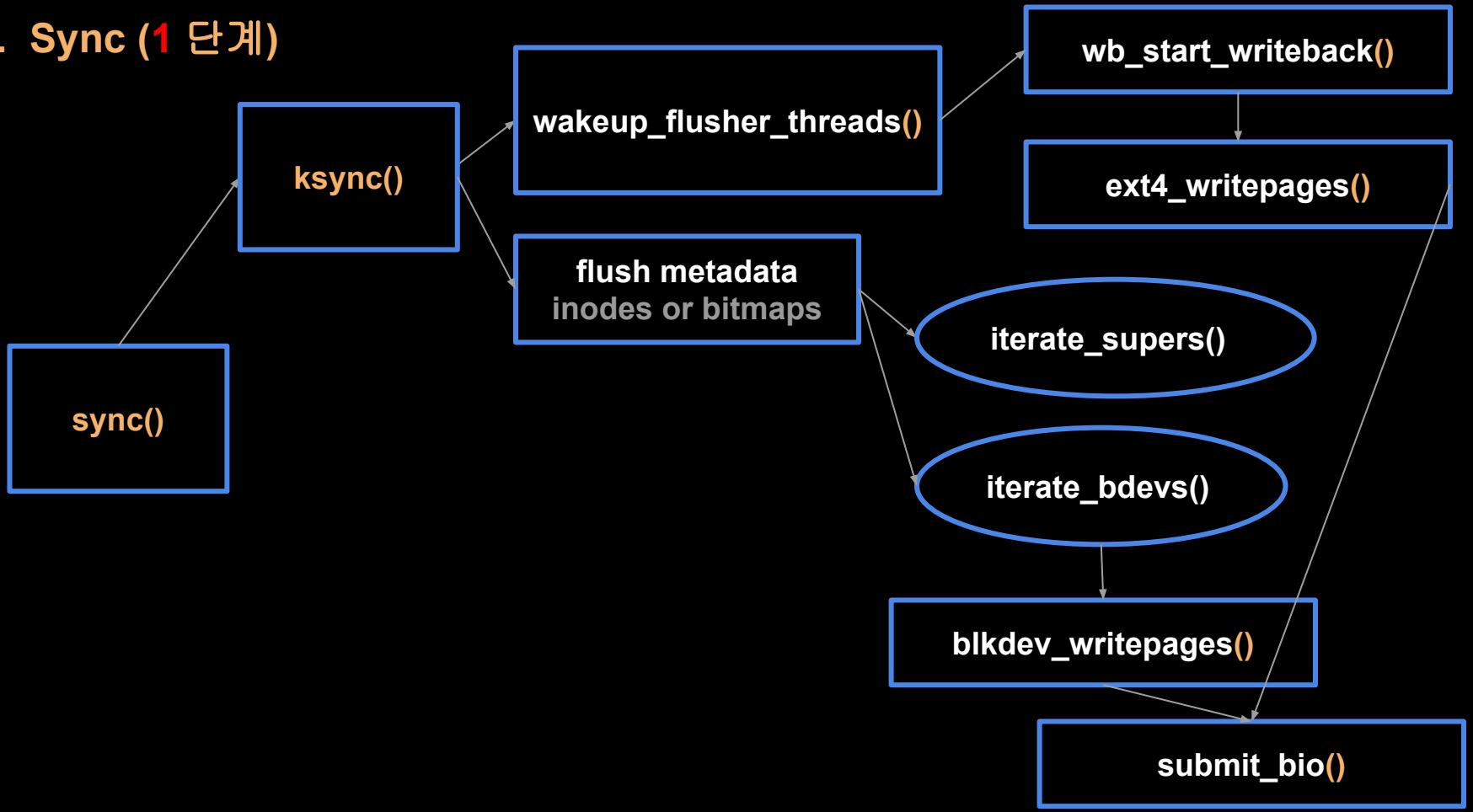
# 3. Sync (1 단계)


sync()
ksync()
wakeup_flusher_threads()
flush metadata
inodes or bitmaps
wb_start_writeback()
ext4_writepages()

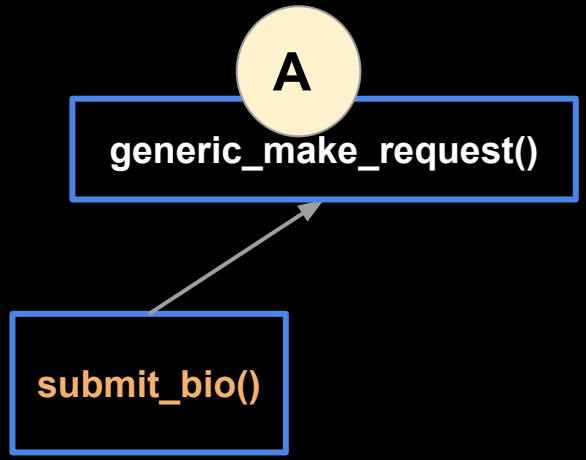
# 3. Sync (1 단계)


wb_start_writeback()
wakeup_flusher_threads()
ksync()
ext4_writepages()
flush metadata
inodes or bitmaps
iterate_supers()
sync()
iterate_bdevs()

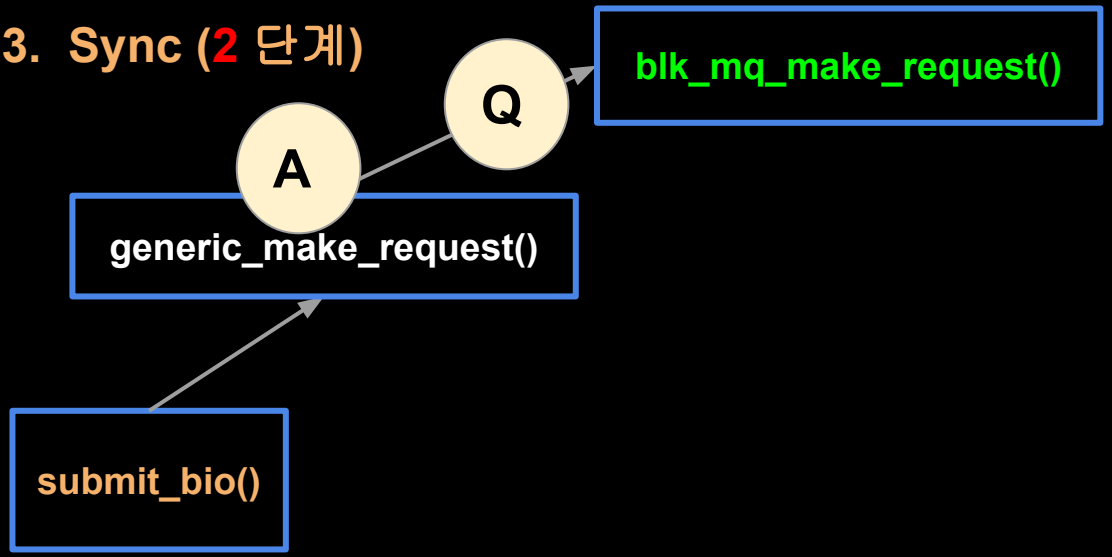
# 3. Sync (1 단계)


wb_start_writeback()
wakeup_flusher_threads()
ksync()
ext4_writepages()
flush metadata
inodes or bitmaps
iterate_supers()
sync()
iterate_bdevs()
blkdev_writepages()
submit_bio()

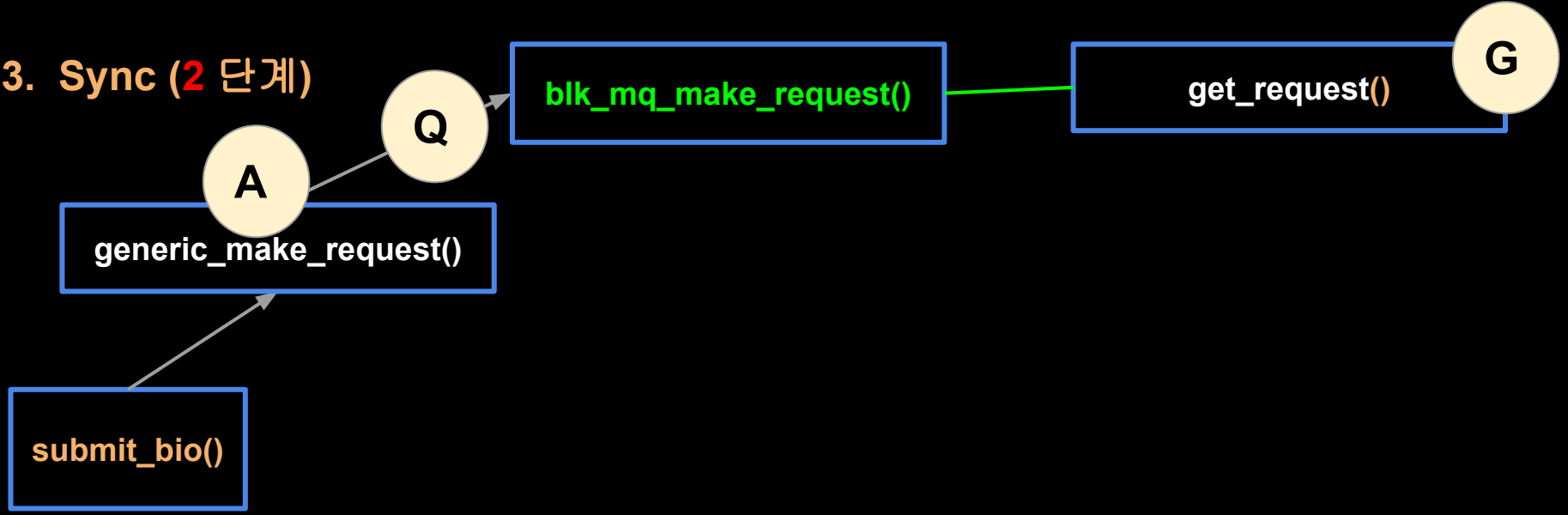
# 3. Sync (2 단계)



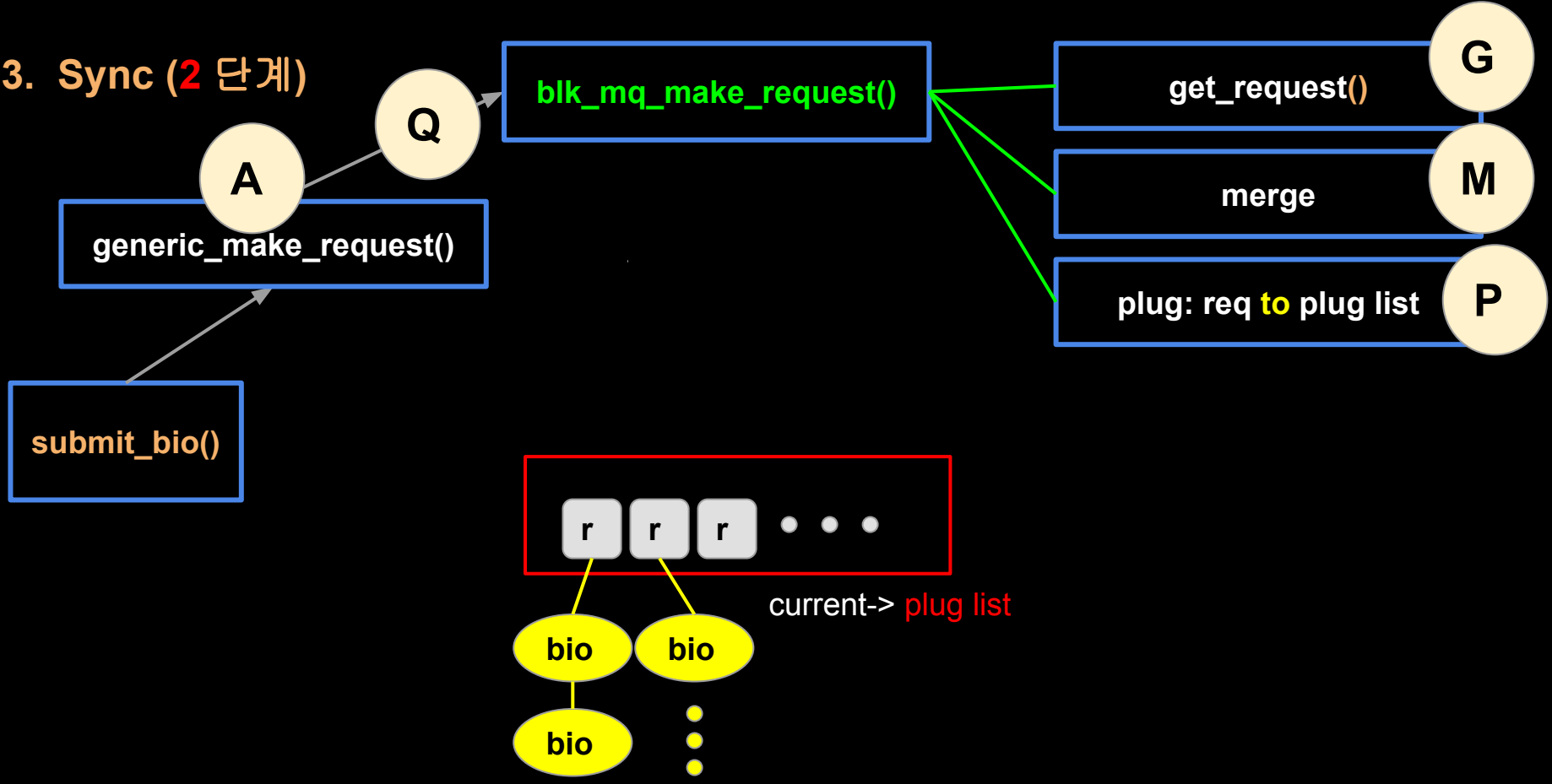
## 3. Sync (2 단계)



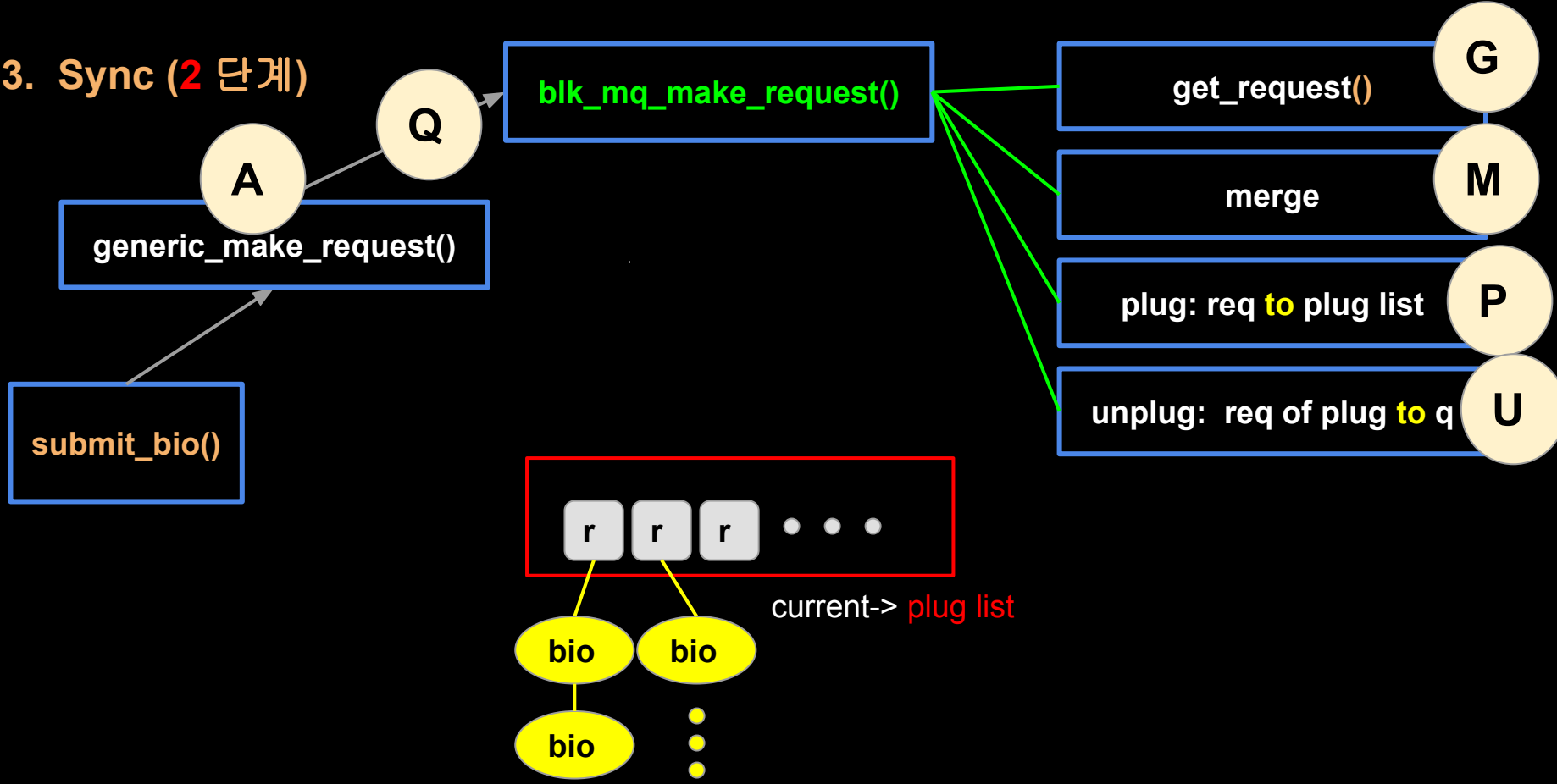
## 3. Sync (2 단계)


A
Q
G
blk_mq_make_request()
get_request()
generic_make_request()
submit_bio()

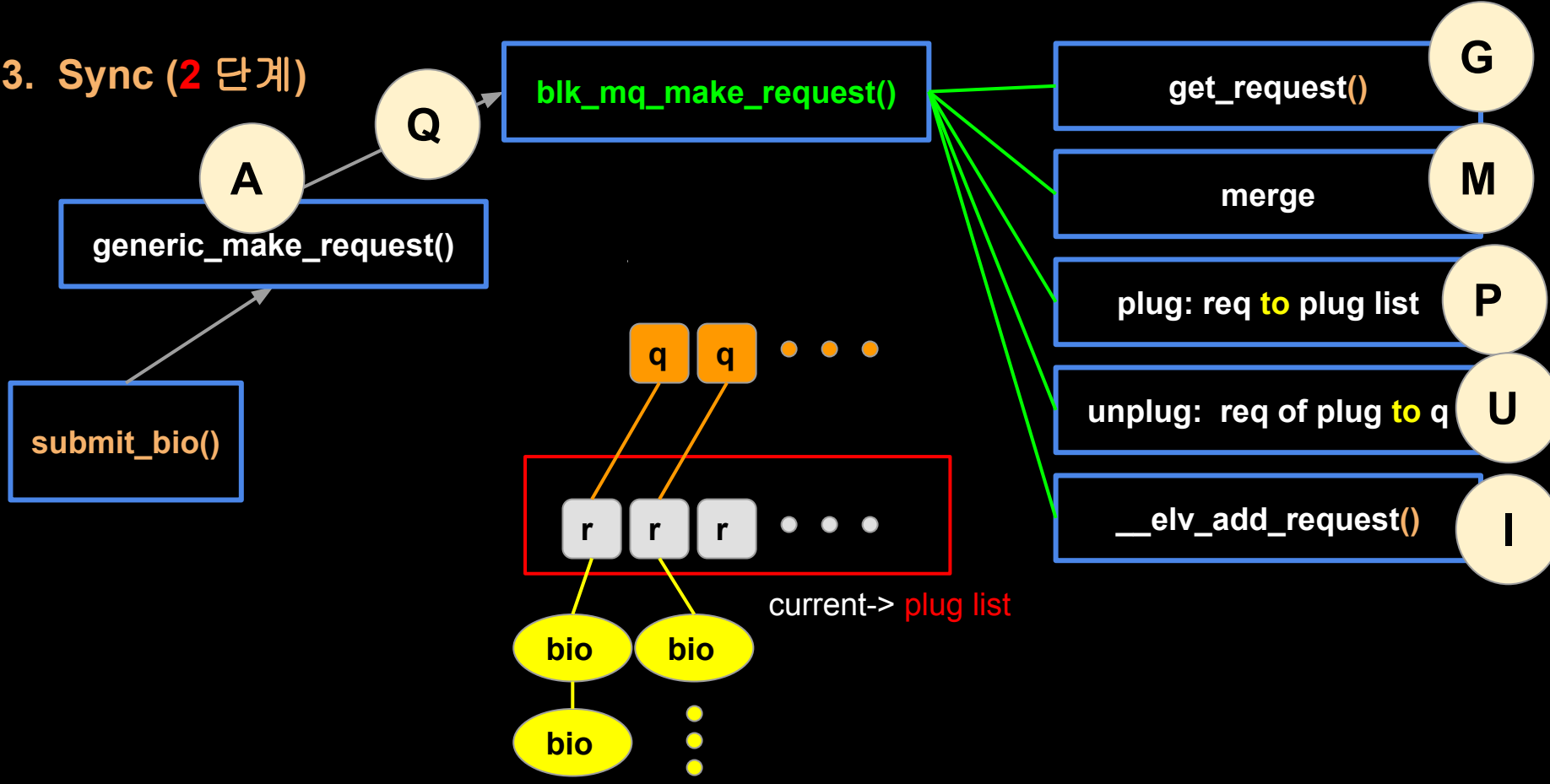
## 3. Sync (2 단계)


Q
A
blk_mq_make_request()
get_request()
G
merge
M
plug: req to plug list
P
generic_make_request()
submit_bio()
r
r
r
current-> plug list
bio
bio
bio

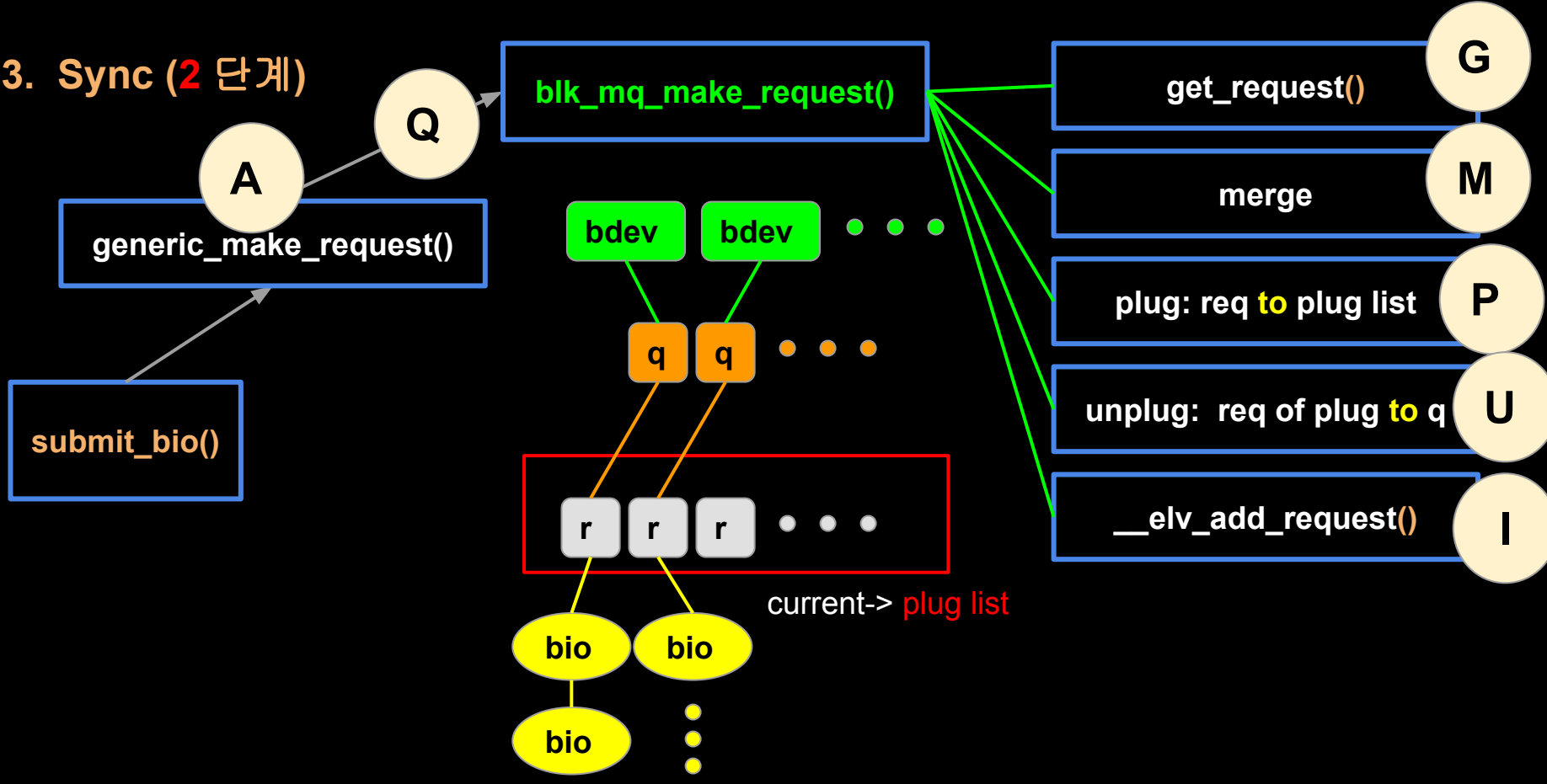
# 3. Sync (2 단계)


Q
A
blk_mq_make_request()
generic_make_request()
submit_bio()
get_request()
G
merge
M
plug: req to plug list
P
unplug: req of plug to q
U
r
r
r
current-> plug list
bio
bio
bio

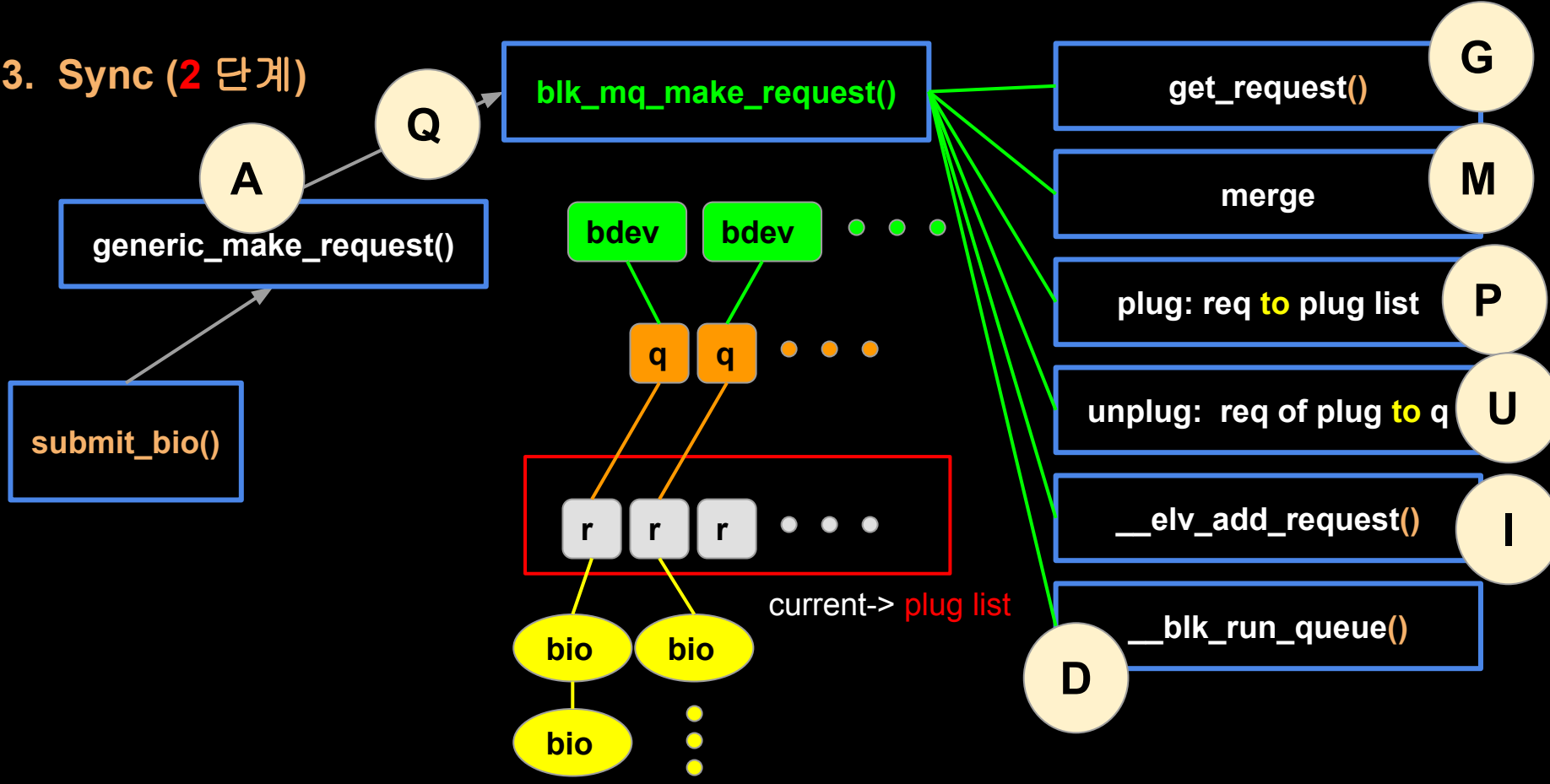
# 3. Sync (2 단계)

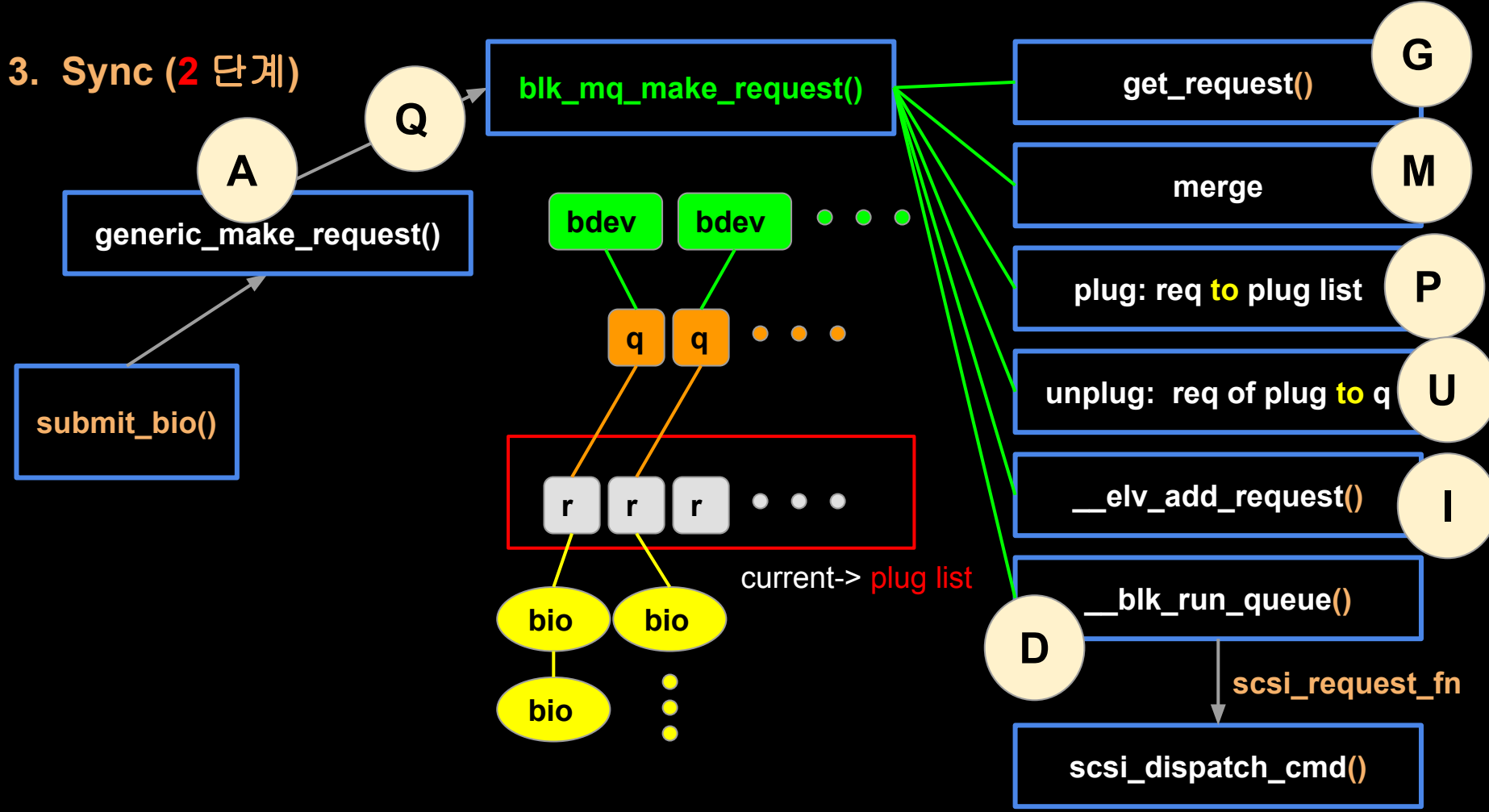

Q
A
blk_mq_make_request()
generic_make_request()
submit_bio()
get_request()
G
merge
M
plug: req to plug list
P
unplug: req of plug to q
U
__elv_add_request()
I
q
q
r
r
r
current-> plug list
bio
bio
bio

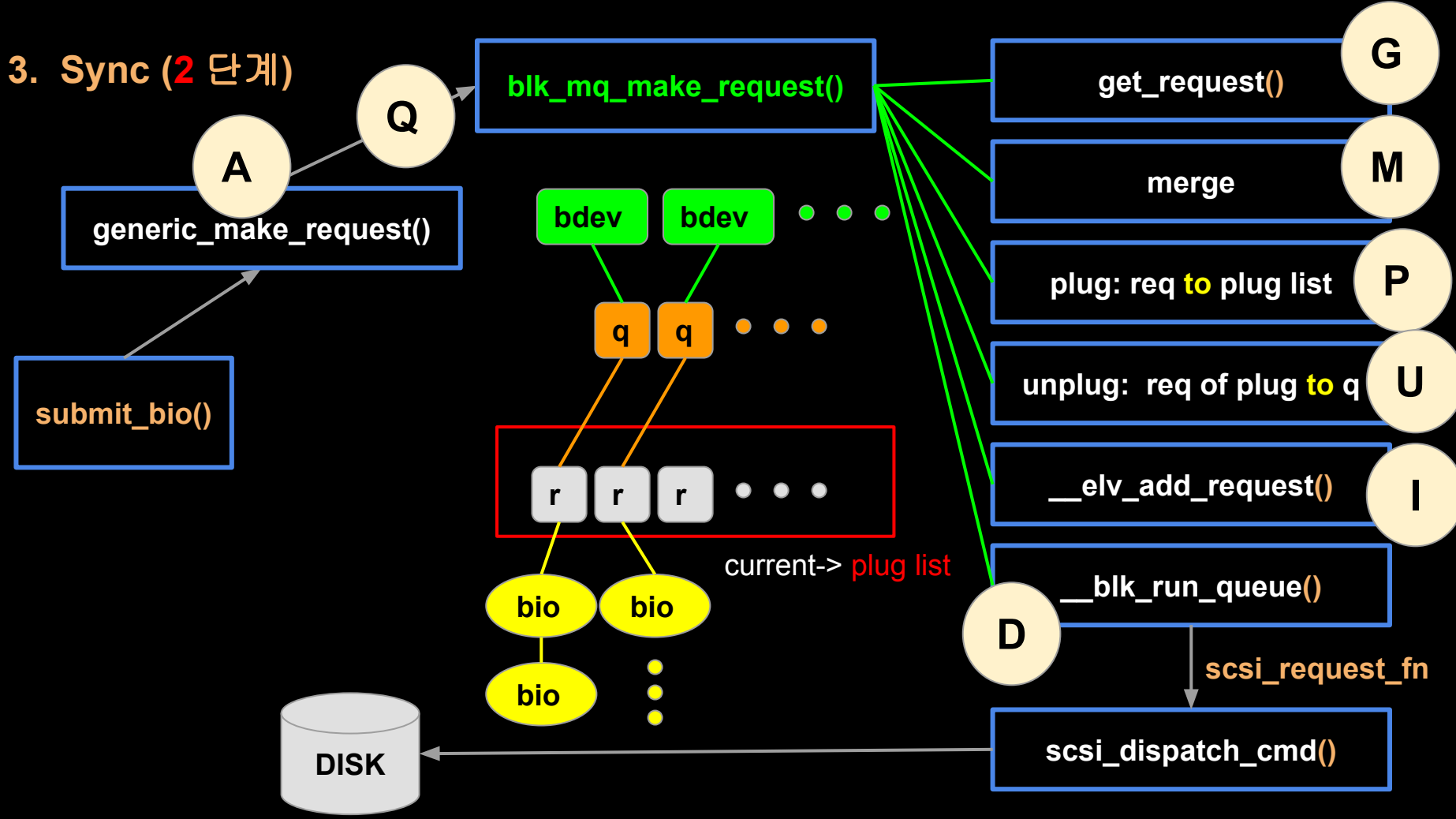
# 3. Sync (2 단계)

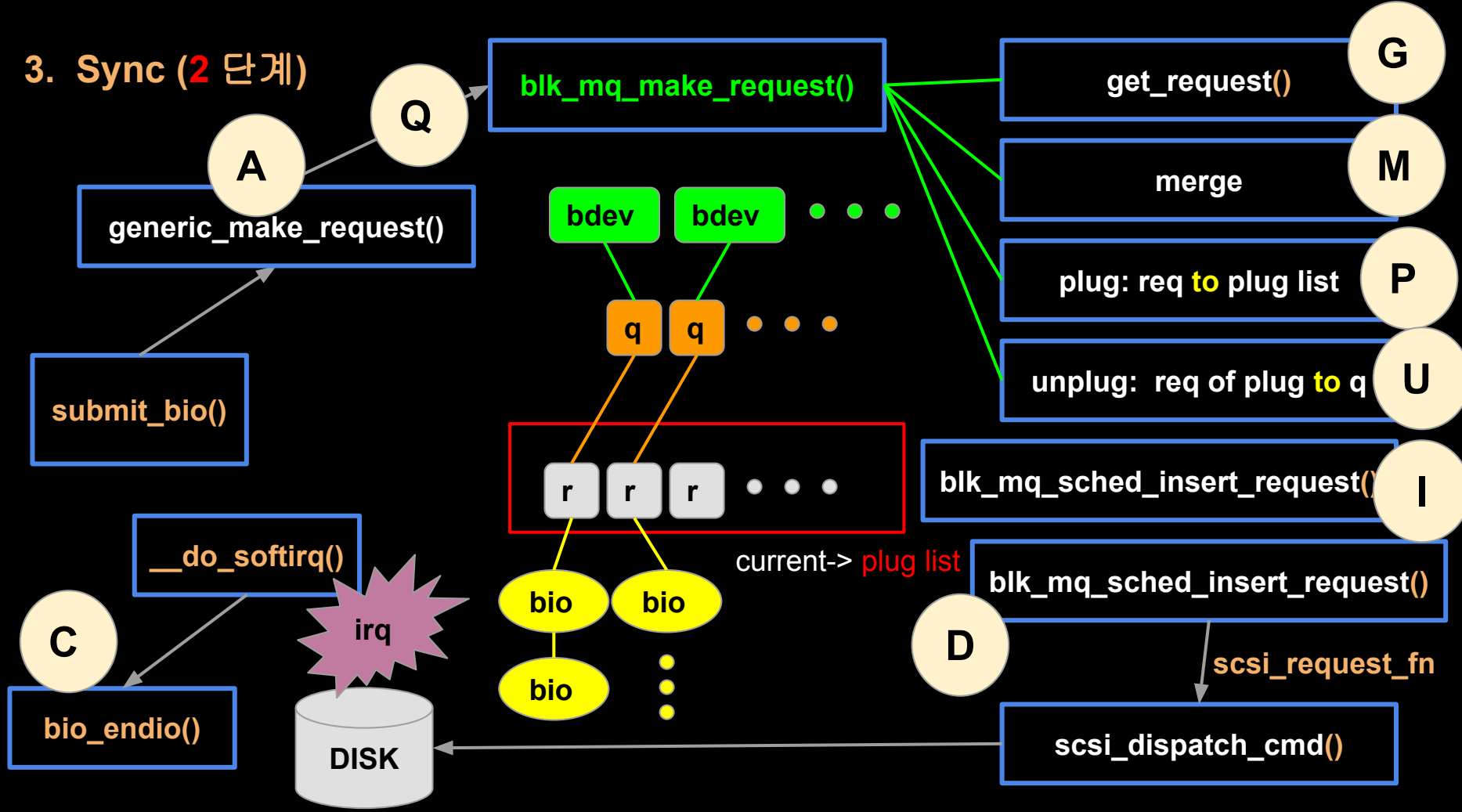

Q
A
blk_mq_make_request()
generic_make_request()
submit_bio()
bdev
bdev
q
q
r
r
r
bio
bio
bio
current-> plug list
get_request()
G
merge
M
plug: req to plug list
P
unplug: req of plug to q
U
__elv_add_request()
I

# 3. Sync (2 단계)

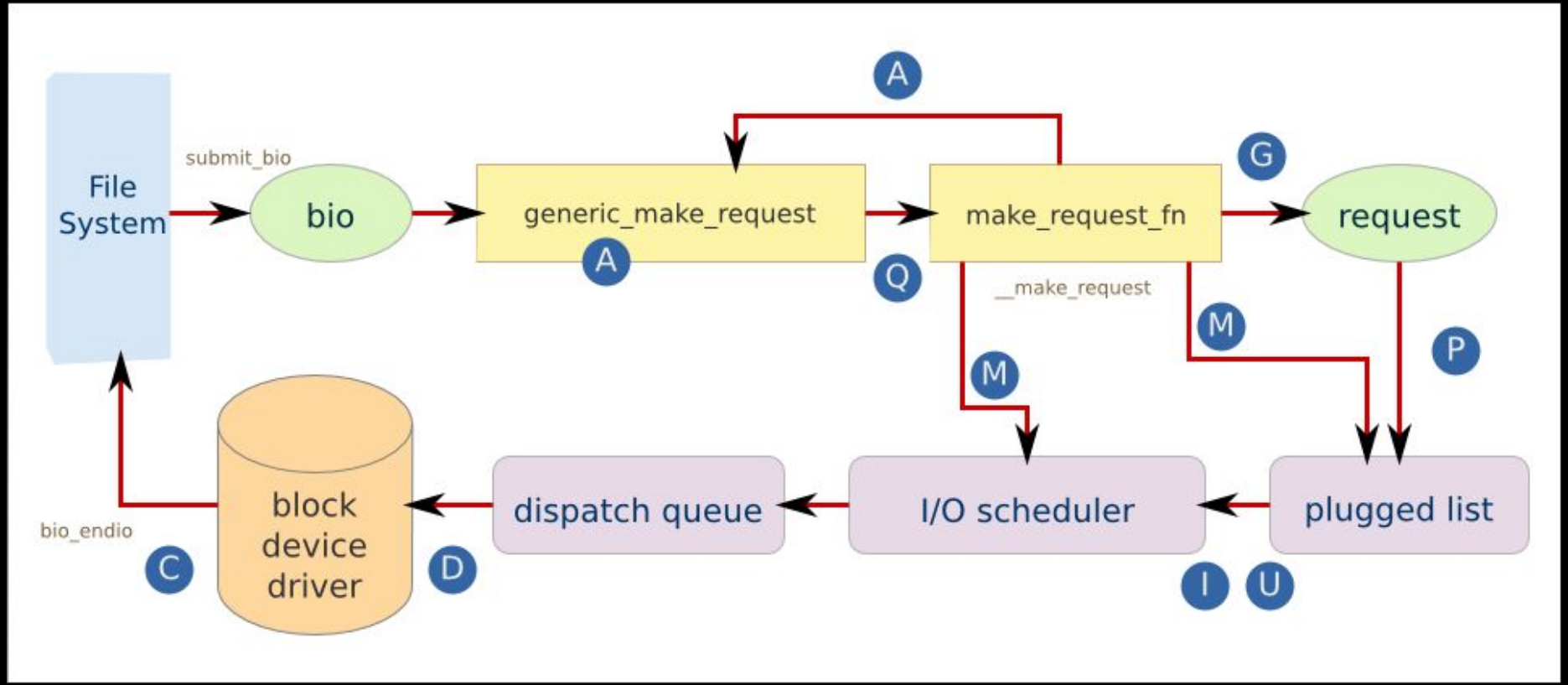

Q
A
blk_mq_make_request()
generic_make_request()
submit_bio()
bdev
bdev
q
q
r
r
r
current-> plug list
bio
bio
bio
get_request()
merge
plug: req to plug list
unplug: req of plug to q
__elv_add_request()
__blk_run_queue()
G
M
P
U
I
D


3. Sync (2 단계)
Q
A
blk_mq_make_request()
generic_make_request()
submit_bio()
bdev
bdev
q
q
r
r
r
current-> plug list
bio
bio
bio
get_request()
G
merge
M
plug: req to plug list
P
unplug: req of plug to q
U
__elv_add_request()
I
__blk_run_queue()
D
scsi_request_fn
scsi_dispatch_cmd()


3. Sync (2 단계)
Q
A
blk_mq_make_request()
generic_make_request()
submit_bio()
bdev
bdev
q
q
r
r
r
current-> plug list
bio
bio
bio
DISK
get_request()
G
merge
M
plug: req to plug list
P
unplug: req of plug to q
U
__elv_add_request()
I
__blk_run_queue()
D
scsi_request_fn
scsi_dispatch_cmd()


3. Sync (2 단계)
Q
A
G
M
P
U
I
D
C
blk_mq_make_request()
get_request()
merge
plug: req to plug list
unplug: req of plug to q
generic_make_request()
submit_bio()
bdev
bdev
q
q
r
r
r
current-> plug list
blk_mq_sched_insert_request()
blk_mq_sched_insert_request()
scsi_request_fn
scsi_dispatch_cmd()
__do_softirq()
irq
bio_endio()
bio
bio
bio
DISK

# Block Device 레벨로 submit_bio 이후 처리과정 (btrace로 tracing)

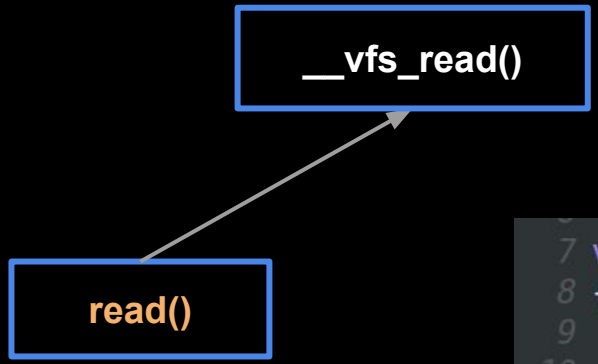



http://studyfoss.egloos.com/5585801

# Index

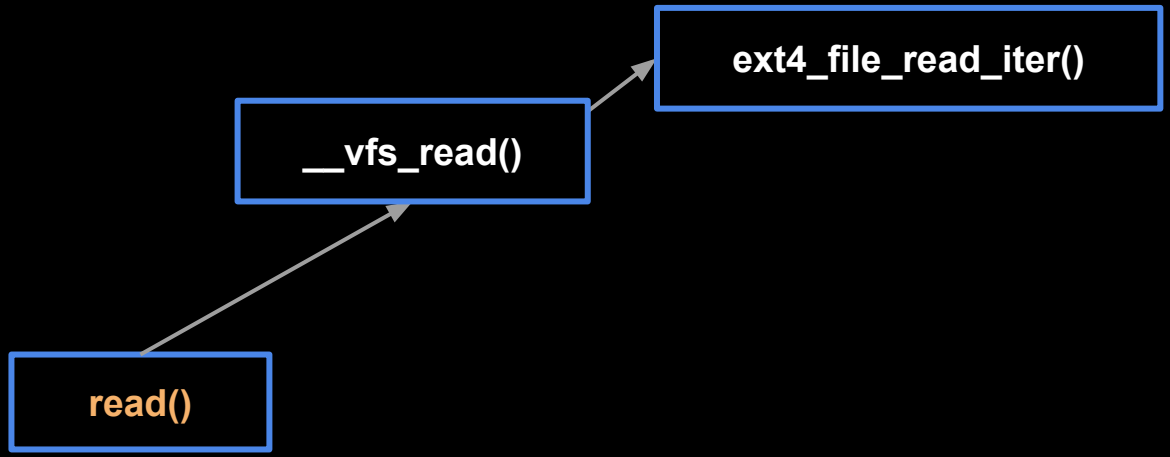
1. OPEN
2. WRITE
3. SYNC
4. READ

# 4. Read

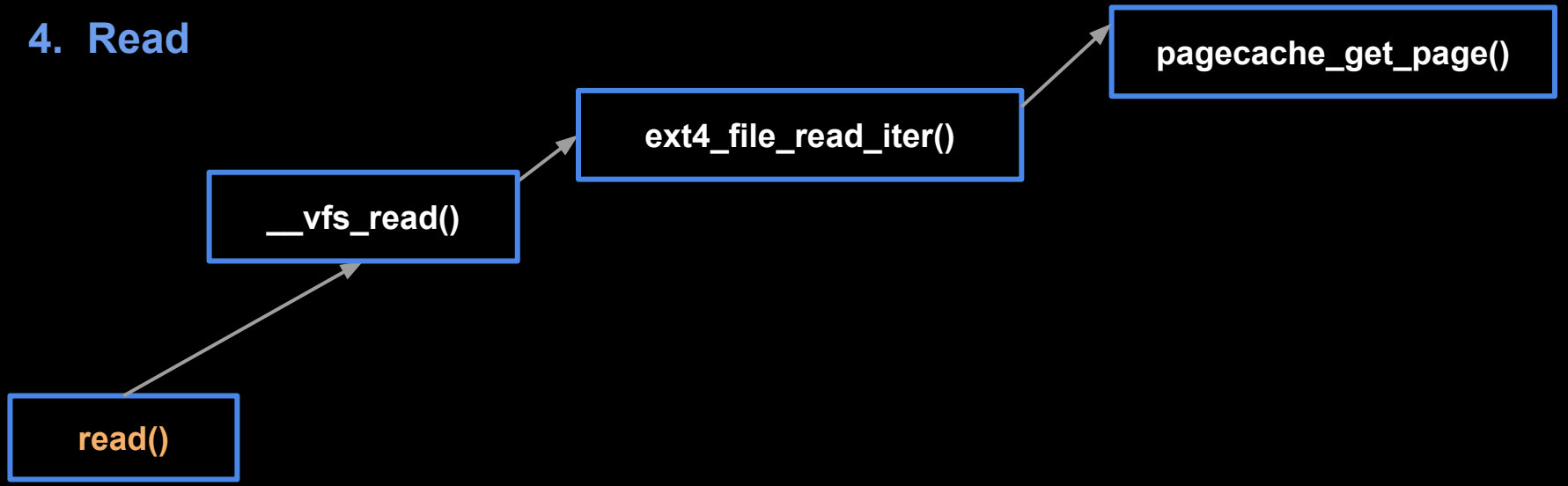



```
7  void main()
8  {
9          FILE *fp = fopen("/home/kosslab/hello.txt","r");
10         char buf[BUFSIZ];
11
12         if (fp) {
13                 fread(buf, SIZE, 1, fp);
14                 printf("%s", buf);
15                 fclose(fp);
16         }
17 }
```

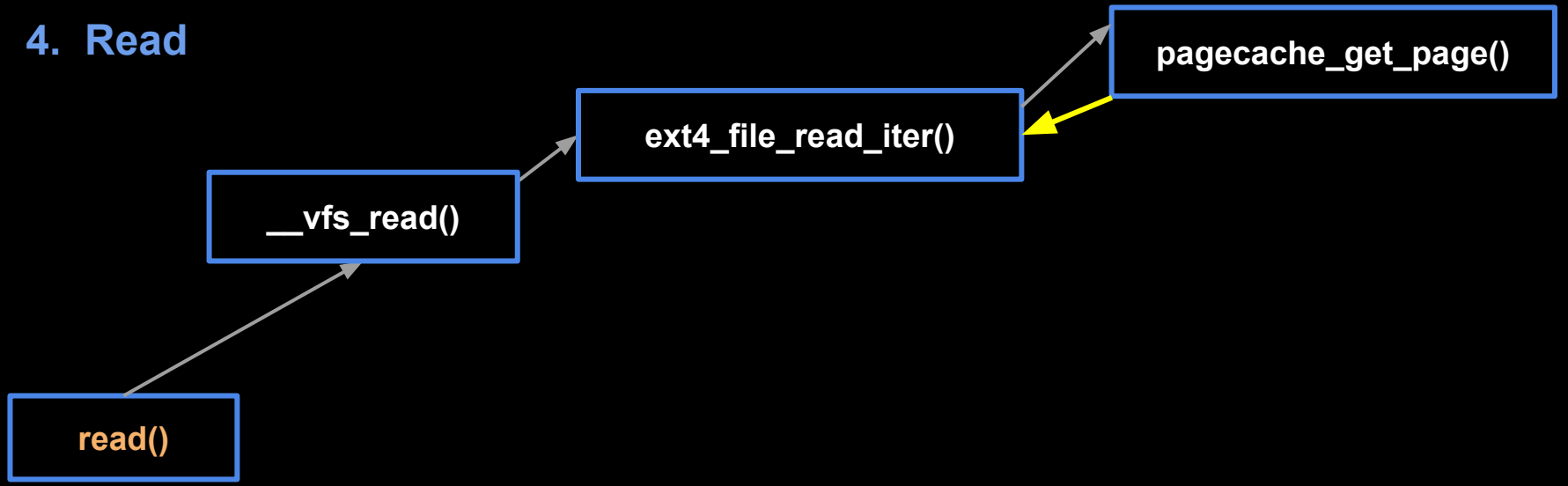
# 4. Read



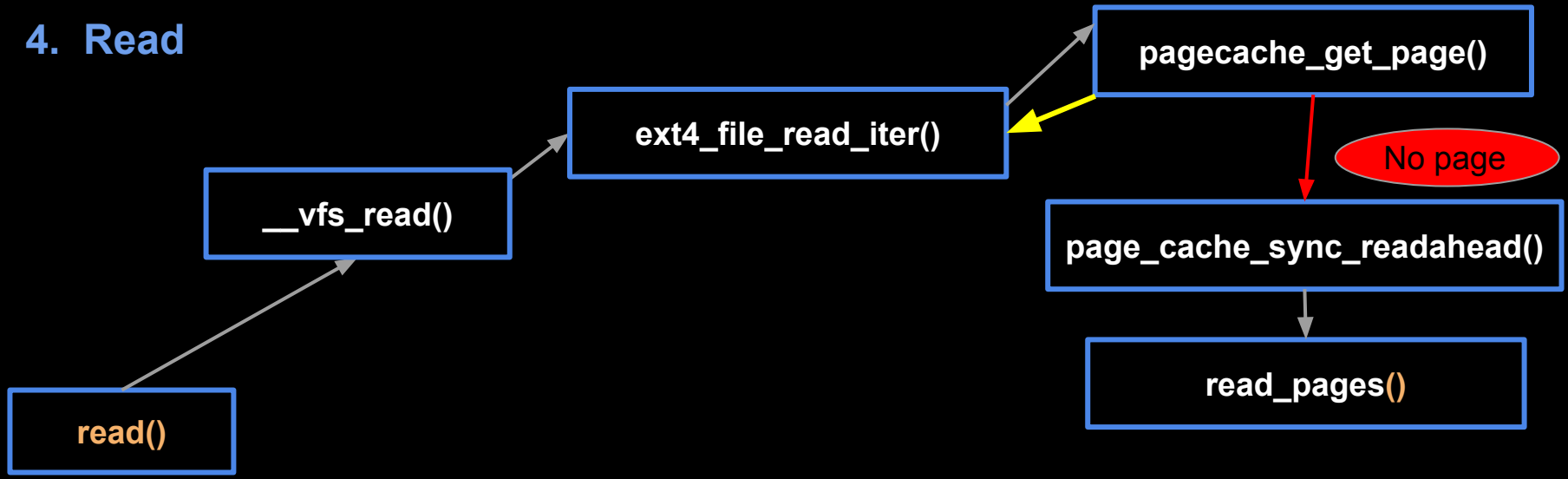
# 4. Read



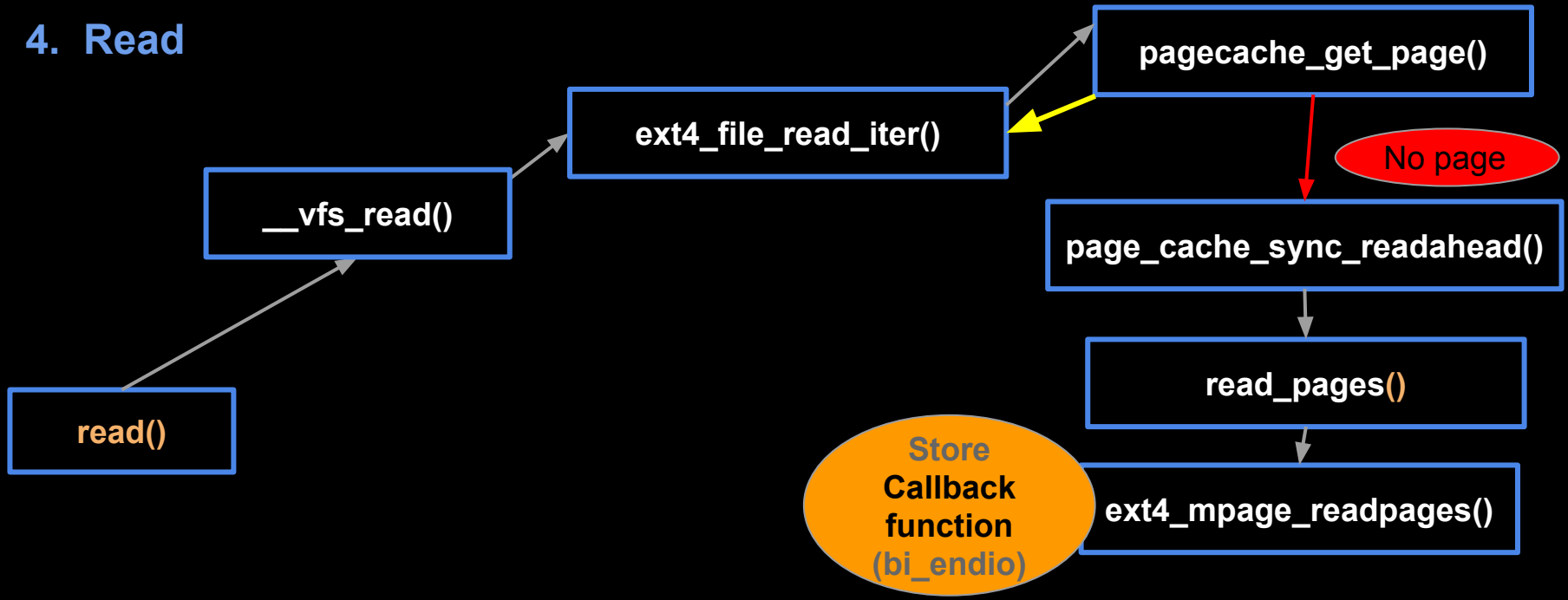
# 4. Read



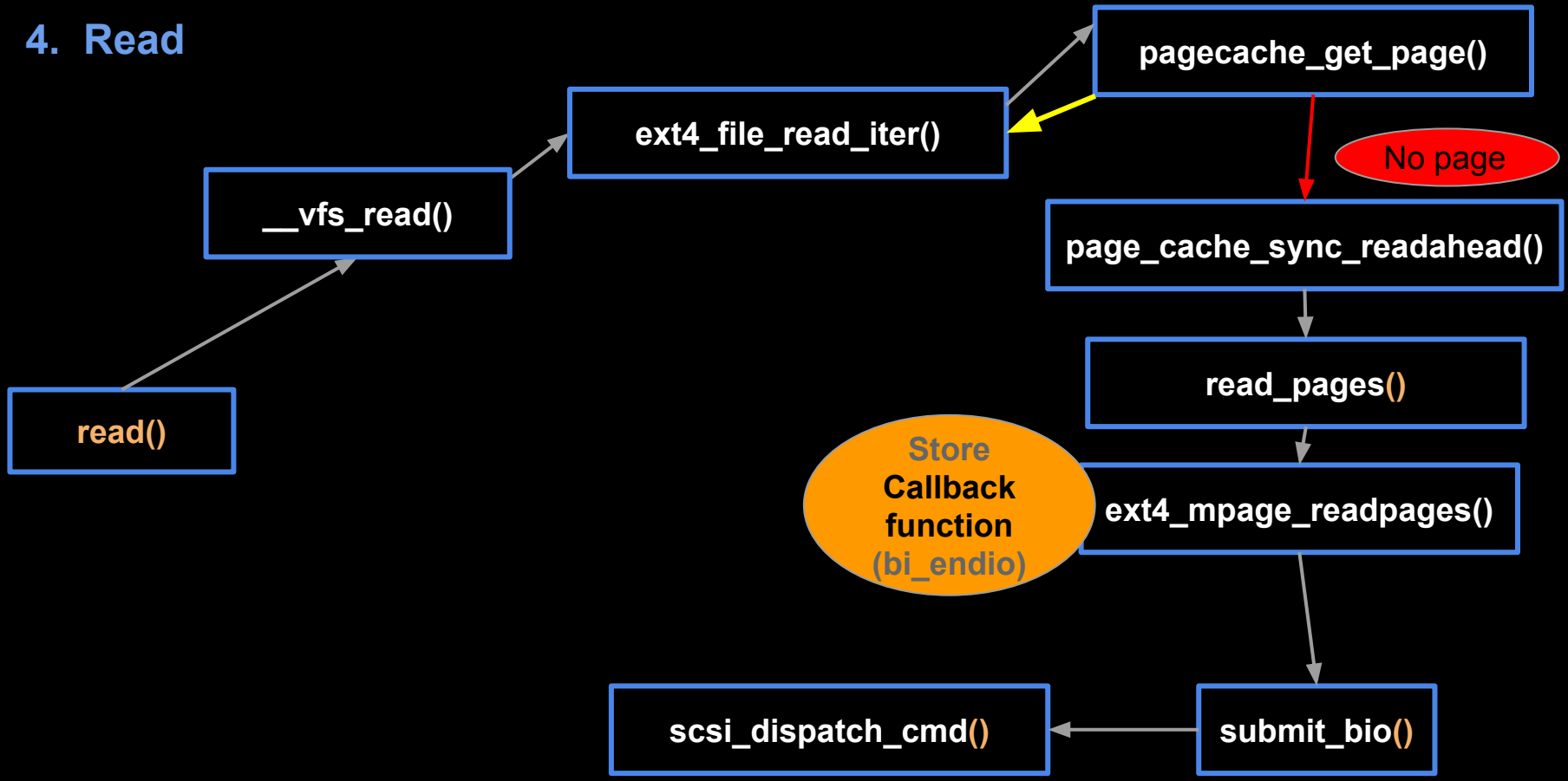
# 4. Read


pagecache_get_page()
ext4_file_read_iter()
No page
__vfs_read()
page_cache_sync_readahead()
read_pages()
read()

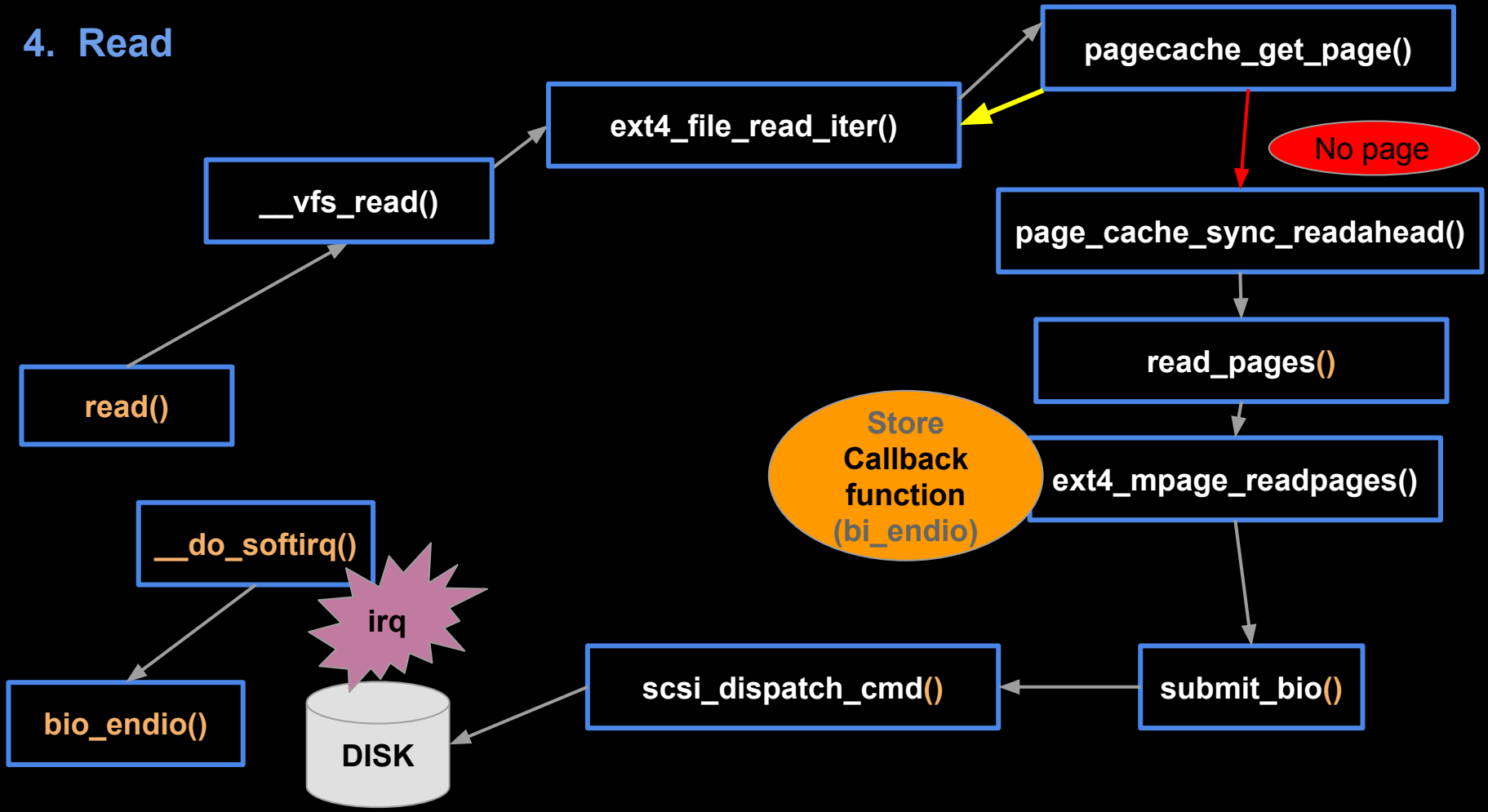
# 4. Read


pagecache_get_page()
ext4_file_read_iter()
No page
__vfs_read()
page_cache_sync_readahead()
read_pages()
read()
Store
Callback
function
(bi_endio)
ext4_mpage_readpages()

# 4. Read


pagecache_get_page()
ext4_file_read_iter()
No page
__vfs_read()
page_cache_sync_readahead()
read_pages()
read()
Store
Callback
function
(bi_endio)
ext4_mpage_readpages()
scsi_dispatch_cmd()
submit_bio()

# 4. Read


pagecache_get_page()
ext4_file_read_iter()
No page
__vfs_read()
page_cache_sync_readahead()
read_pages()
read()
Store
Callback
function
(bi_endio)
ext4_mpage_readpages()
__do_softirq()
irq
bio_endio()
DISK
scsi_dispatch_cmd()
submit_bio()

# Thanks