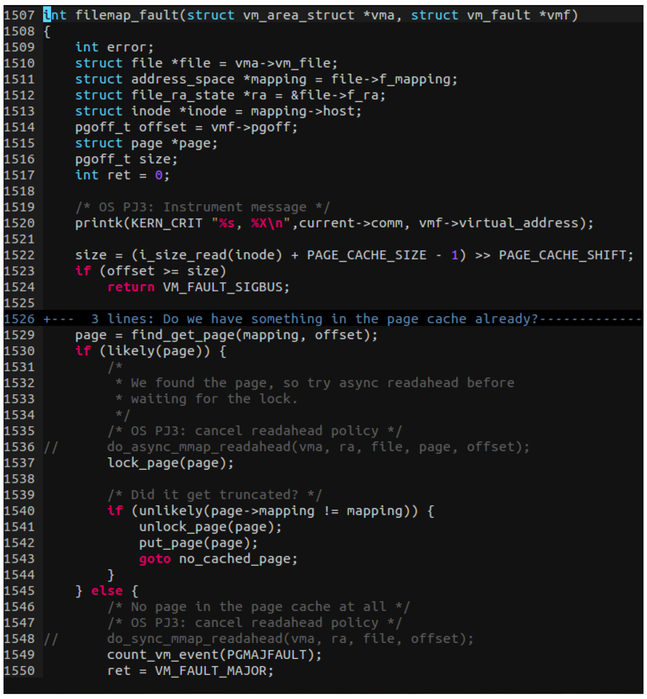
# Operating System Project 3

## 1. Revise Source Codes

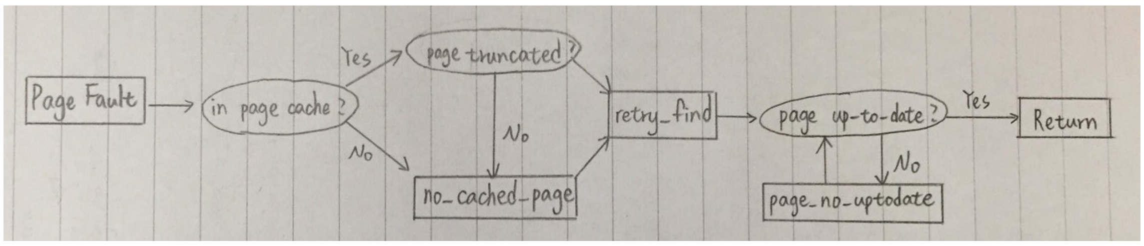


```
int filemap_fault(struct vm_area_struct *vma, struct vm_fault *vmf)
{
    int error;
    struct file *file = vma->vm_file;
    struct address_space *mapping = file->f_mapping;
    struct file_ra_state *ra = &file->f_ra;
    struct inode *inode = mapping->host;
    pgoff_t offset = vmf->pgoff;
    struct page *page;
    pgoff_t size;
    int ret = 0;

    /* OS PJ3: Instrument message */
    printk(KERN_CRIT "%s, %X\n",current->comm, vmf->virtual_address);

    size = (i_size_read(inode) + PAGE_CACHE_SIZE - 1) >> PAGE_CACHE_SHIFT;
    if (offset >= size)
        return VM_FAULT_SIGBUS;

+---  3 lines: Do we have something in the page cache already?------------
    page = find_get_page(mapping, offset);
    if (likely(page)) {
        /*
         * We found the page, so try async readahead before
         * waiting for the lock.
         */
        /* OS PJ3: cancel readahead policy */
//      do_async_mmap_readahead(vma, ra, file, page, offset);
        lock_page(page);

        /* Did it get truncated? */
        if (unlikely(page->mapping != mapping)) {
            unlock_page(page);
            put_page(page);
            goto no_cached_page;
        }
    } else {
        /* No page in the page cache at all */
        /* OS PJ3: cancel readahead policy */
//      do_sync_mmap_readahead(vma, ra, file, offset);
        count_vm_event(PGMAJFAULT);
        ret = VM_FAULT_MAJOR;
```

## 2. Trace Codes

### 2.1. filemap_fault( )

The function will be called when page fault occurs. First, it checks whether the target page is in the page cache, if not, then it use goto method to jump to no_cached_page code session to read pages from storage. if the page is in the page cache, it checks if the page is truncated, if not, it will samely jump to no_cached_page session. After going through the control flow mentioned before, it then consequently goes to retry_find session. The last step is checking whether the page is up-to-date, if yes, the function directly returns, otherwise, it jumps to page_no_uptodate session to update the page before returning.



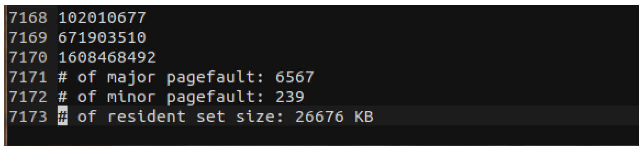
### 2.2. mmap( )

The mmap() function shall establish a mapping between a process' address space and a file, shared memory object, or typed memory object.

## 3. Compare Pure Demand Paging & Read Ahead Algorithm

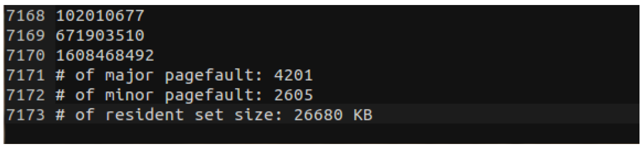
### 3.1. Test Outputs

pure demand paging : (pd_output)



```
7168 102010677
7169 671903510
7170 1608468492
7171 # of major pagefault: 6567
7172 # of minor pagefault: 239
7173 # of resident set size: 26676 KB
```

read ahead algorithm : (ra_output)



```
7168 102010677
7169 671903510
7170 1608468492
7171 # of major pagefault: 4201
7172 # of minor pagefault: 2605
7173 # of resident set size: 26680 KB
```

### 3.2. Comparison Table

| | major pagefaults | minor pagefaults | resident set size |
|---|---|---|---|
| pure demand paging | 6567 | 239 | 26676 KB |
| read ahead algoritm | 4201 | 2605 | 26680 KB |