Memory Mapping Implementation (mmap) in Linux Kernel

Adrian Huang | May, 2022

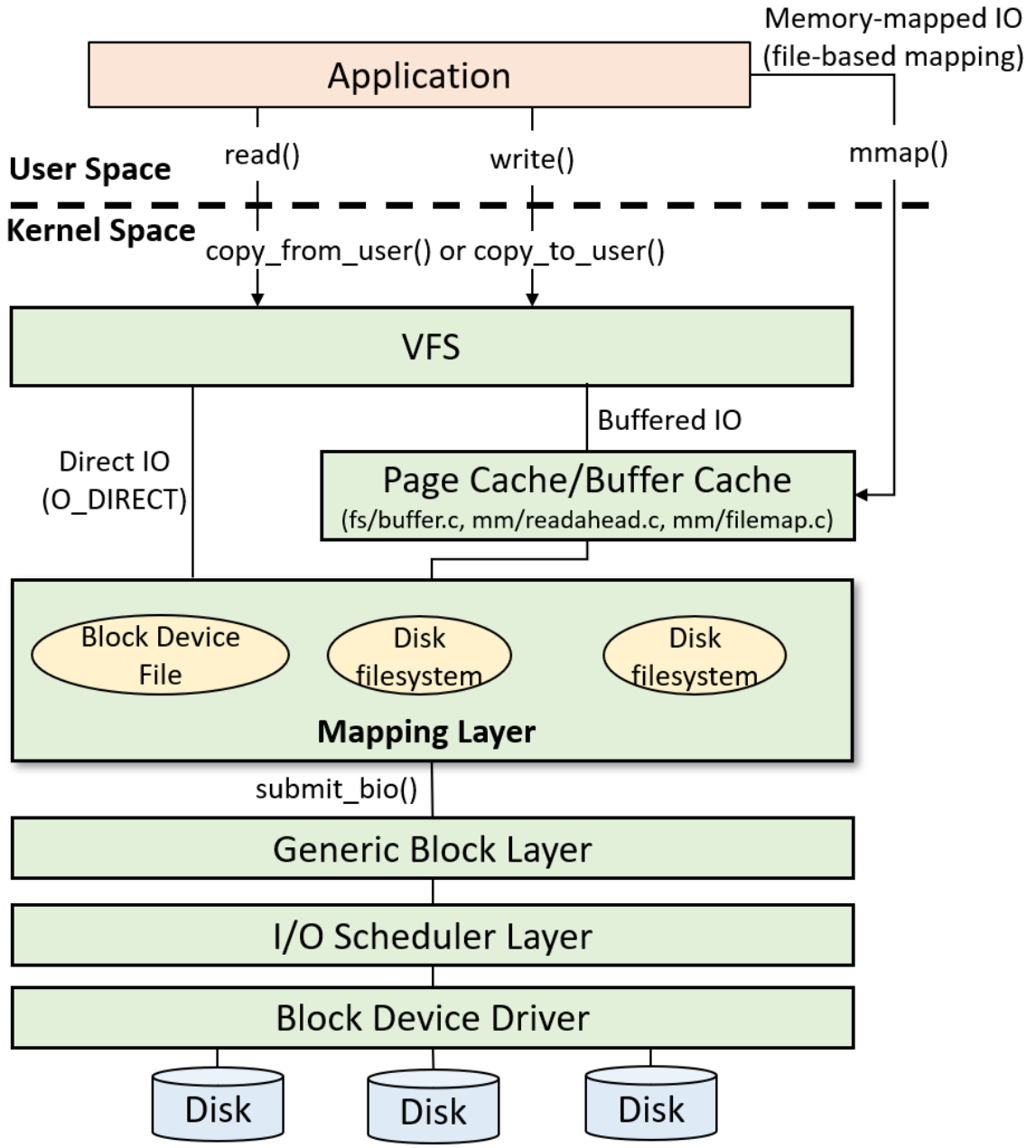
*** Based on kernel 5.11 (x86_64) – QEMU**
*** SMP (4 CPUs) and 8GB memory**
*** Kernel parameter: nokaslr norandmaps**
*** Userspace: ASLR is disabled**
*** Legacy BIOS**

Agenda

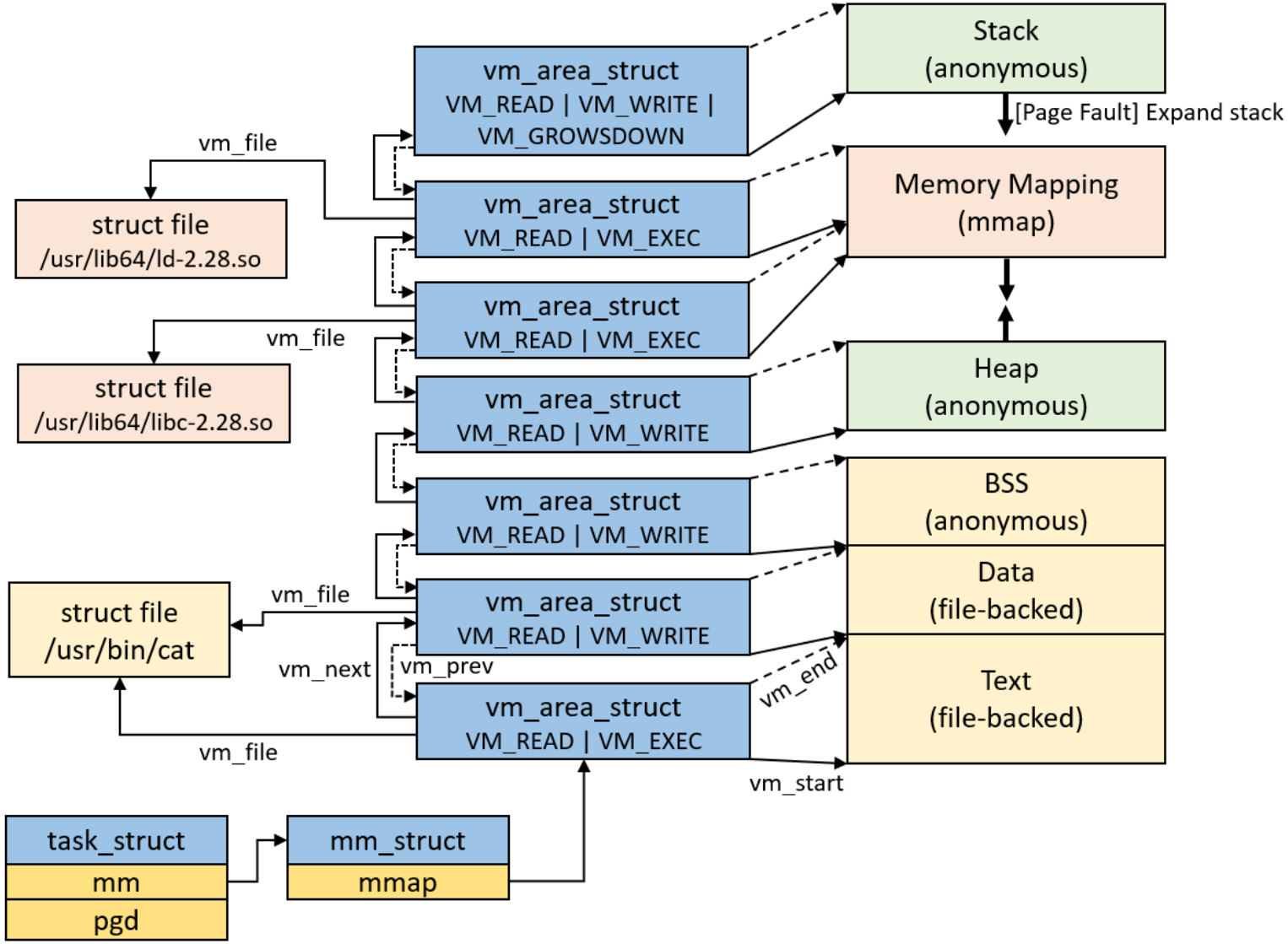
- Three different IO types
- Process Address Space – mm_struct & VMA
- Four types of memory mappings
- mmap system call implementation
- Demand page: page fault handling for four types of memory mappings
- fork()
  - COW mapping configuration: set ‘write protect’ when calling fork()
  - COW fault call path

Three different IO types

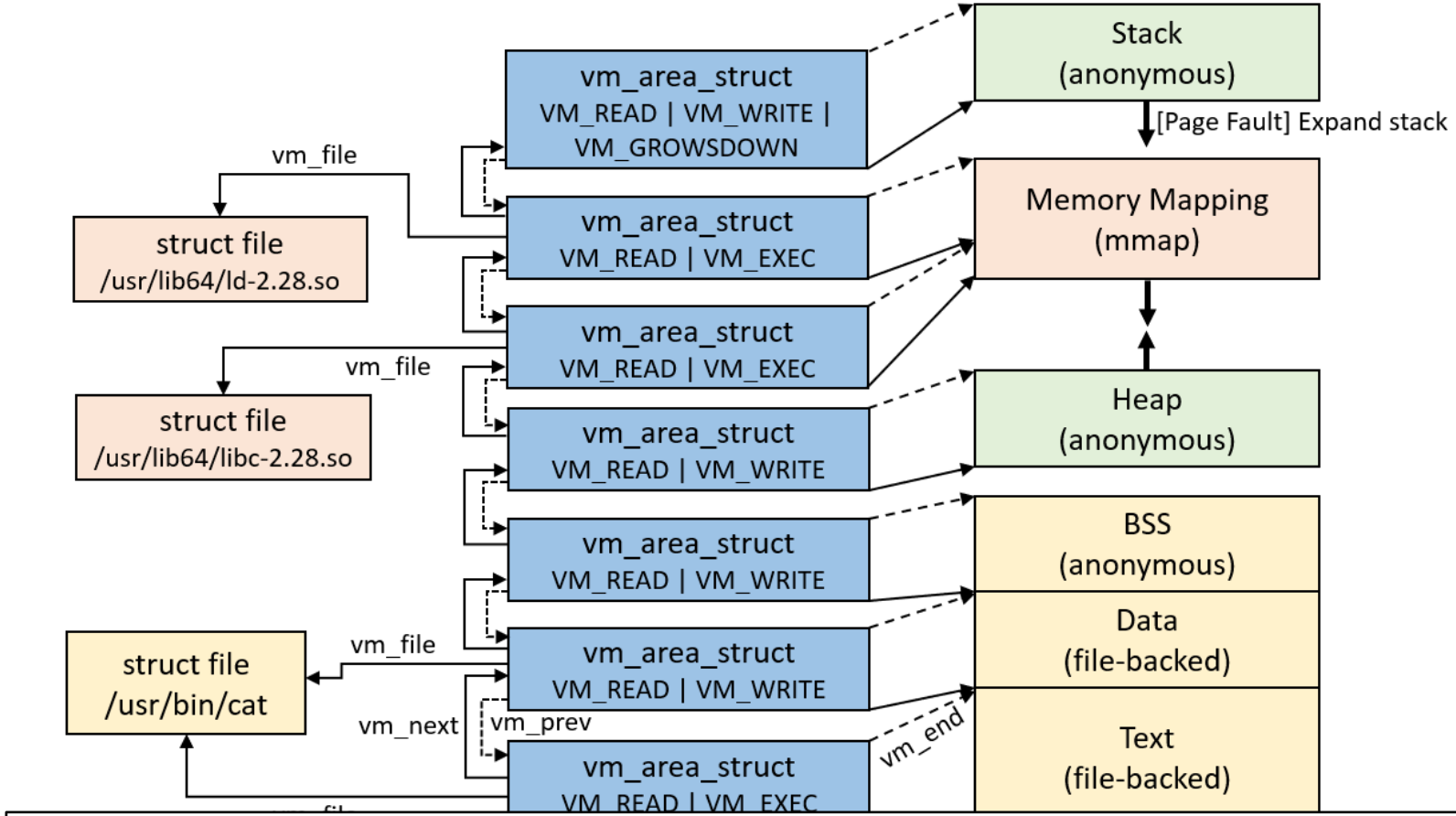
- **Buffered IO**
  - Leverage page cache
- **Direct IO**
  - Bypass page cache
- **Memory-mapped IO (file-based mapping, memory-mapped file)**
  - File content can be accessed by operations on the bytes in the corresponding memory region.



Process Address Space – mm_struct & VMA (1/2)



Process Address Space – mm_struct & VMA (2/2)



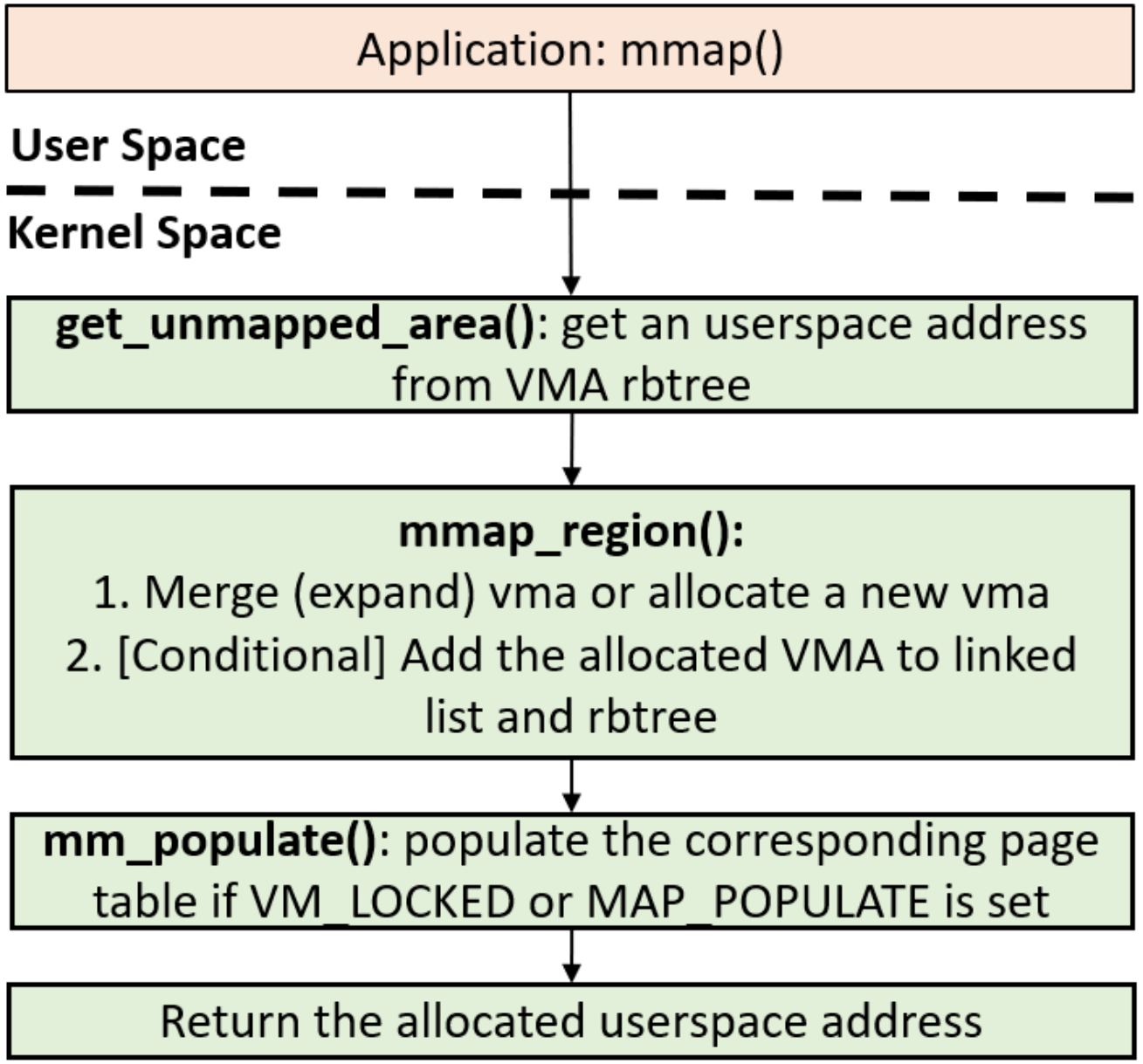

```
/ # cat /proc/43/maps
00400000-00401000 r--p 00000000 00:02 114                                /anon_mmap
00401000-00496000 r-xp 00001000 00:02 114                                /anon_mmap
00496000-004bd000 r--p 00096000 00:02 114                                /anon_mmap
004be000-004c1000 r--p 000bd000 00:02 114                                /anon_mmap
004c1000-004c4000 rw-p 000c0000 00:02 114                                /anon_mmap
004c4000-004e8000 rw-p 00000000 00:00 0                                  [heap]
7ffff7ffa000-7ffff7ffe000 r--p 00000000 00:00 0                          [vvar]
7ffff7ffe000-7ffff7fff000 r-xp 00000000 00:00 0                          [vdso]
7ffffffde000-7fffffffd000 rw-p 00000000 00:00 0                          [stack]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

Four types of memory mappings

Visibility of Modification	Mapping Type	
	File	Anonymous
Private (Modification is not visible to other processes)	Initializing memory from contents of file Example: Process's .text and .data segements (Changes are not carried through to the underlying file)	Memory Allocation
Shared (Modification is visible to other processes)	1. Memory-mapped IO: Changes are carried through to the underlying file 2. Sharing memory between processes (IPC)	Sharing memory between processes (IPC)

Reference from: Chapter 49, The Linux Programming Interface

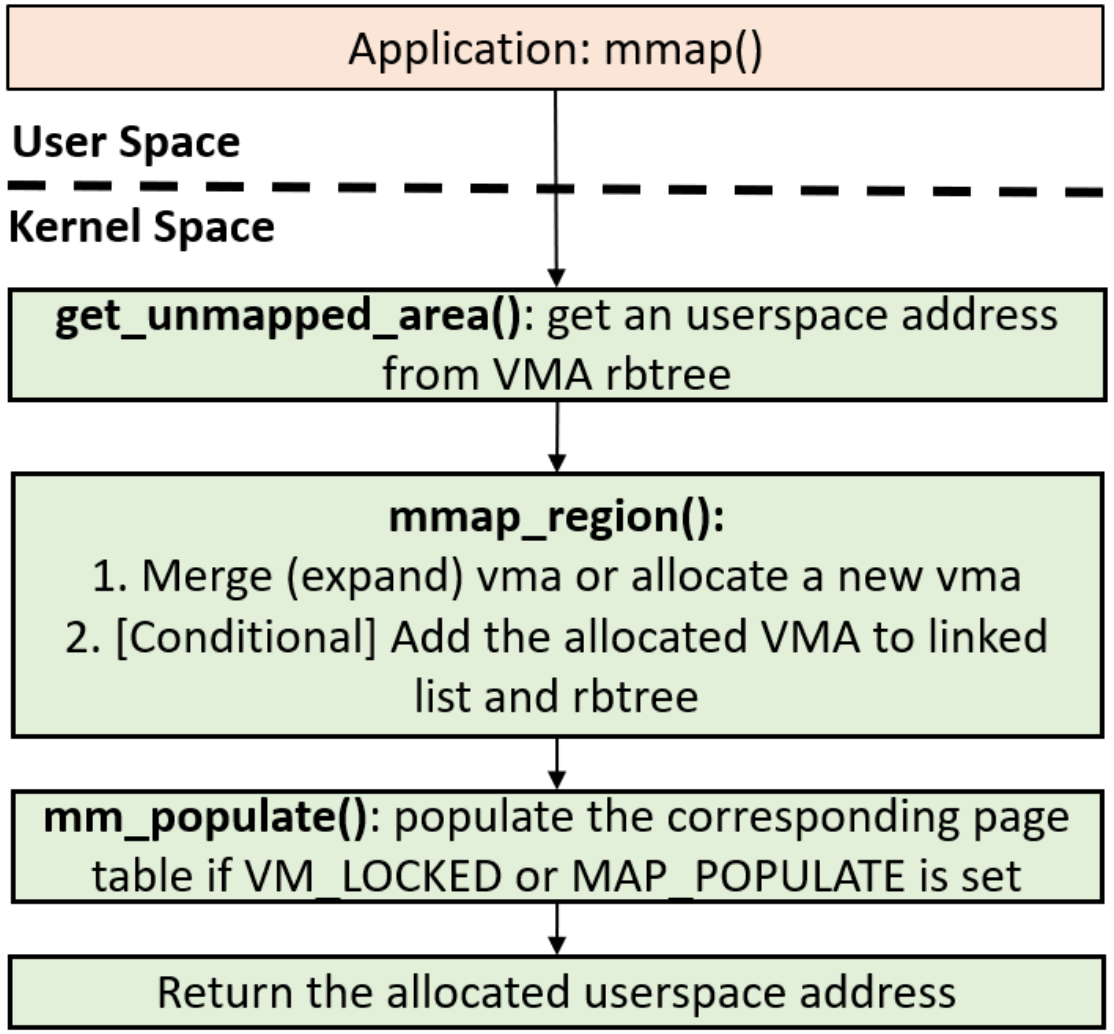
mmap system call implementation (1/2)



mmap system call implementation (2/2)

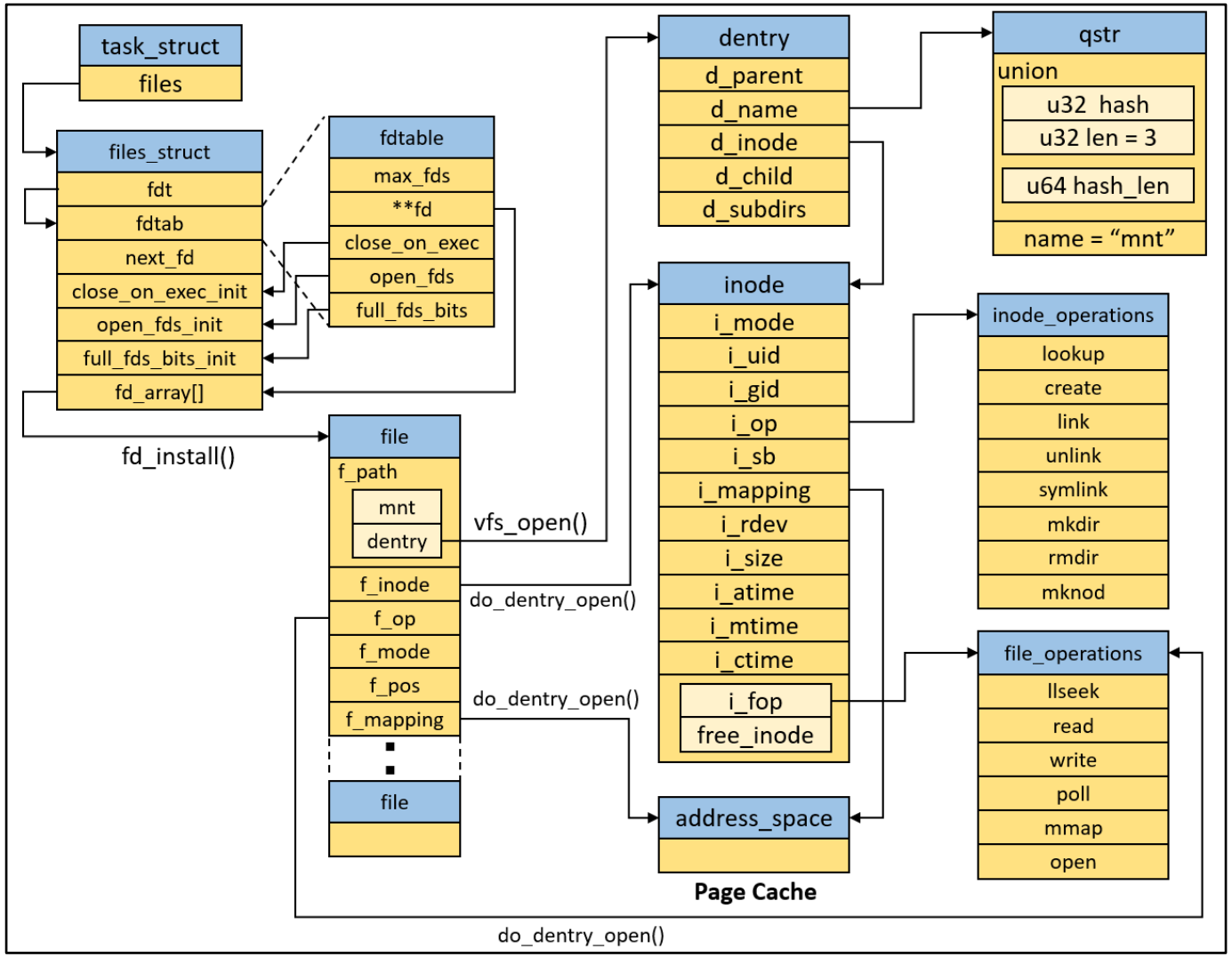
```
mmap or sys_mmap ["arch/x86/kernel/sys_x86_64.c"]
  |- ksys_mmap_pgoff [mm/mmap.c]
    |- vm_mmap_pgoff
      |- do_mmap
        |- get_unmapped_area: return an userspace address
          if (file memory mapping)
            |- file->f_op->get_unmapped_area
              |- thp_get_unmapped_area
          else if (flags & MAP_SHARED)
            |- shmem_get_unmapped_area
          else [anonymous page]
            |- arch_get_unmapped_area_topdown
        |- mmap_region
          /* Merge (expand) vma or allocate a new vma */
      |- mm_populate if VM_LOCKED or MAP_POPULATE is set
```

```
thp_get_unmapped_area ["mm/huge_memory.c"]
  if (is dax)
    return __thp_get_unmapped_area
  else
    return current->mm->get_unmapped_area

arch_get_unmapped_area_topdown ["arch/x86/kernel/sys_x86_64.c"]
  Prepare a vm_unmapped_area_info struct
  |- addr = vm_unmapped_area(&info)
    |- unmapped_area_topdown
  return addr
```



1. thp_get_unmapped_area() eventually calls arch_get_unmapped_area_topdown()
2. unmapped_area_topdown() is the key point for allocating an userspace address

inode_operations & file_operations: file




```
|- do_dentry_open
  if (f->f_op->open)
    |- f->f_op->open(...)
```

```
const struct inode_operations ext4_file_inode_operations = {
        .setattr        = ext4_setattr,
        .getattr        = ext4_file_getattr,
        .listxattr      = ext4_listxattr,
        .get_acl        = ext4_get_acl,
        .set_acl        = ext4_set_acl,
        .fiemap         = ext4_fiemap,
};
```

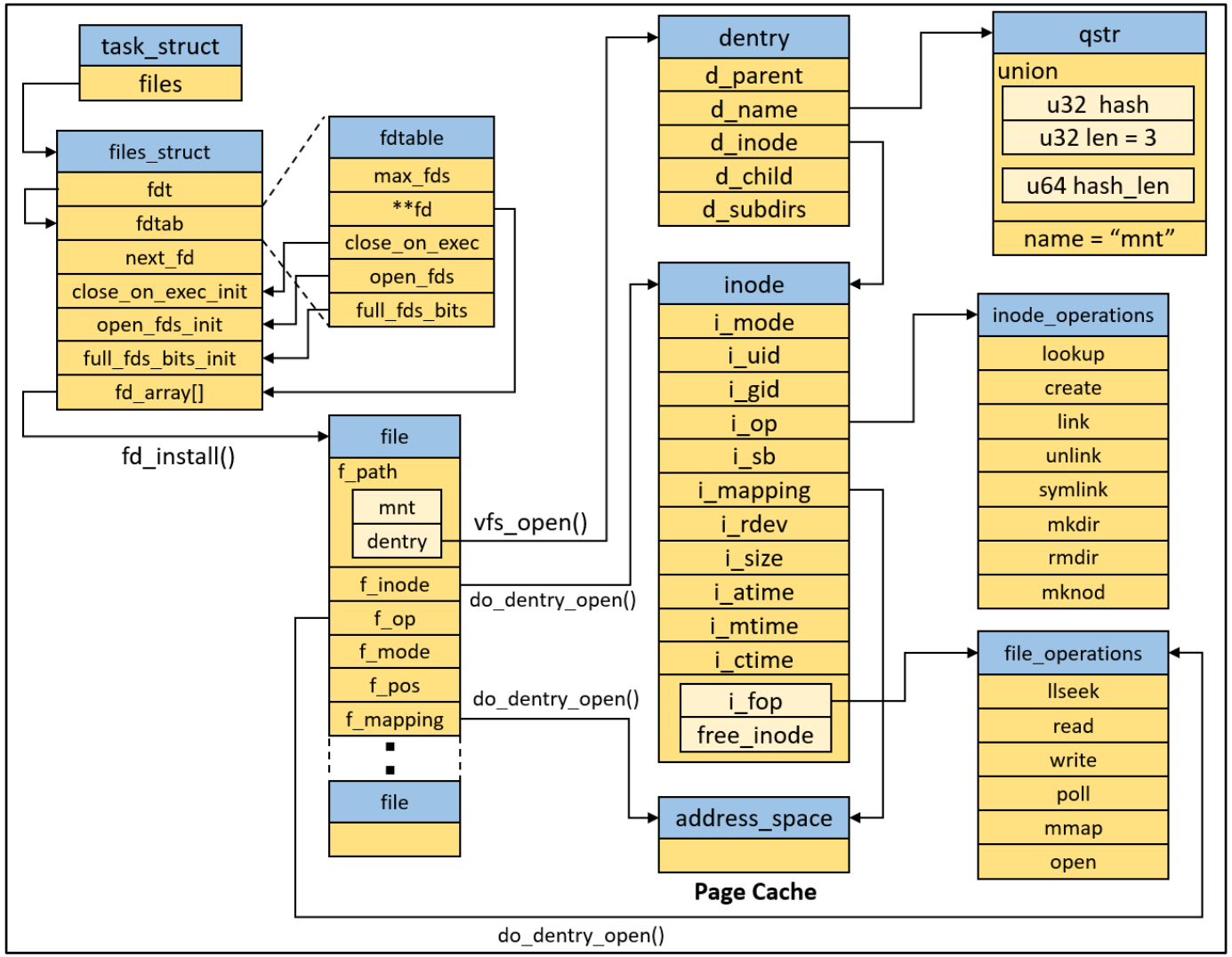
fs/ext4/file.c

```
const struct file_operations ext4_file_operations = {
        .llseek         = ext4_llseek,
        .read_iter      = ext4_file_read_iter,
        .write_iter     = ext4_file_write_iter,
        .iopoll         = iomap_dio_iopoll,
        .unlocked_ioctl = ext4_ioctl,
#ifdef CONFIG_COMPAT
        .compat_ioctl   = ext4_compat_ioctl,
#endif
        .mmap           = ext4_file_mmap,
        .mmap_supported_flags = MAP_SYNC,
        .open           = ext4_file_open,
        .release        = ext4_release_file,
        .fsync          = ext4_sync_file,
        .get_unmapped_area = thp_get_unmapped_area,
        .splice_read    = generic_file_splice_read,
        .splice_write   = iter_file_splice_write,
        .fallocate      = ext4_fallocate,
};
```

fs/ext4/file.c

thp_get_unmapped_area(): DAX supported for huge page

inode_operations & file_operations: directory




```
|- do_dentry_open
  if (f->f_op->open)
    |- f->f_op->open(...)
```

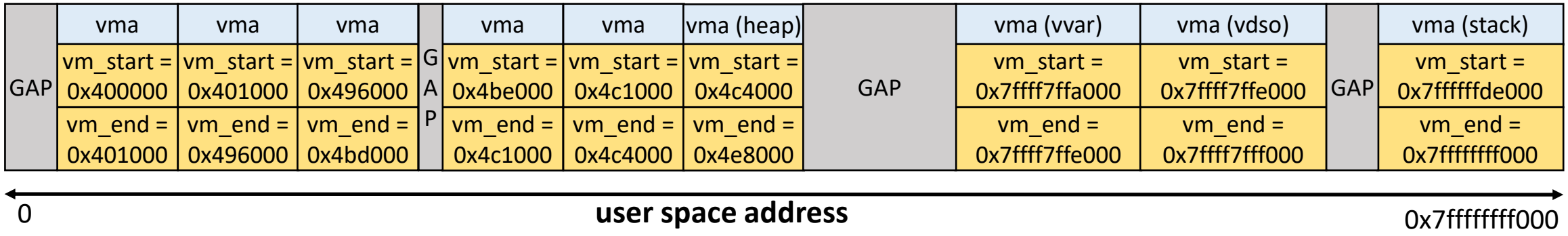
```c
const struct inode_operations ext4_dir_inode_operations = {
        .create         = ext4_create,
        .lookup         = ext4_lookup,
        .link           = ext4_link,
        .unlink         = ext4_unlink,
        .symlink        = ext4_symlink,
        .mkdir          = ext4_mkdir,
        .rmdir          = ext4_rmdir,
        .mknod          = ext4_mknod,
        .tmpfile        = ext4_tmpfile,
        .rename         = ext4_rename2,
        .setattr        = ext4_setattr,
        .getattr        = ext4_getattr,
        .listxattr      = ext4_listxattr,
        .get_acl        = ext4_get_acl,
        .set_acl        = ext4_set_acl,
        .fiemap         = ext4_fiemap,
};
```

fs/ext4/namei.c

```c
const struct file_operations ext4_dir_operations = {
        .llseek         = ext4_dir_llseek,
        .read           = generic_read_dir,
        .iterate_shared = ext4_readdir,
        .unlocked_ioctl = ext4_ioctl,
#ifdef CONFIG_COMPAT
        .compat_ioctl   = ext4_compat_ioctl,
#endif
        .fsync          = ext4_sync_file,
        .release        = ext4_release_dir,
};
```

fs/ext4/dir.c

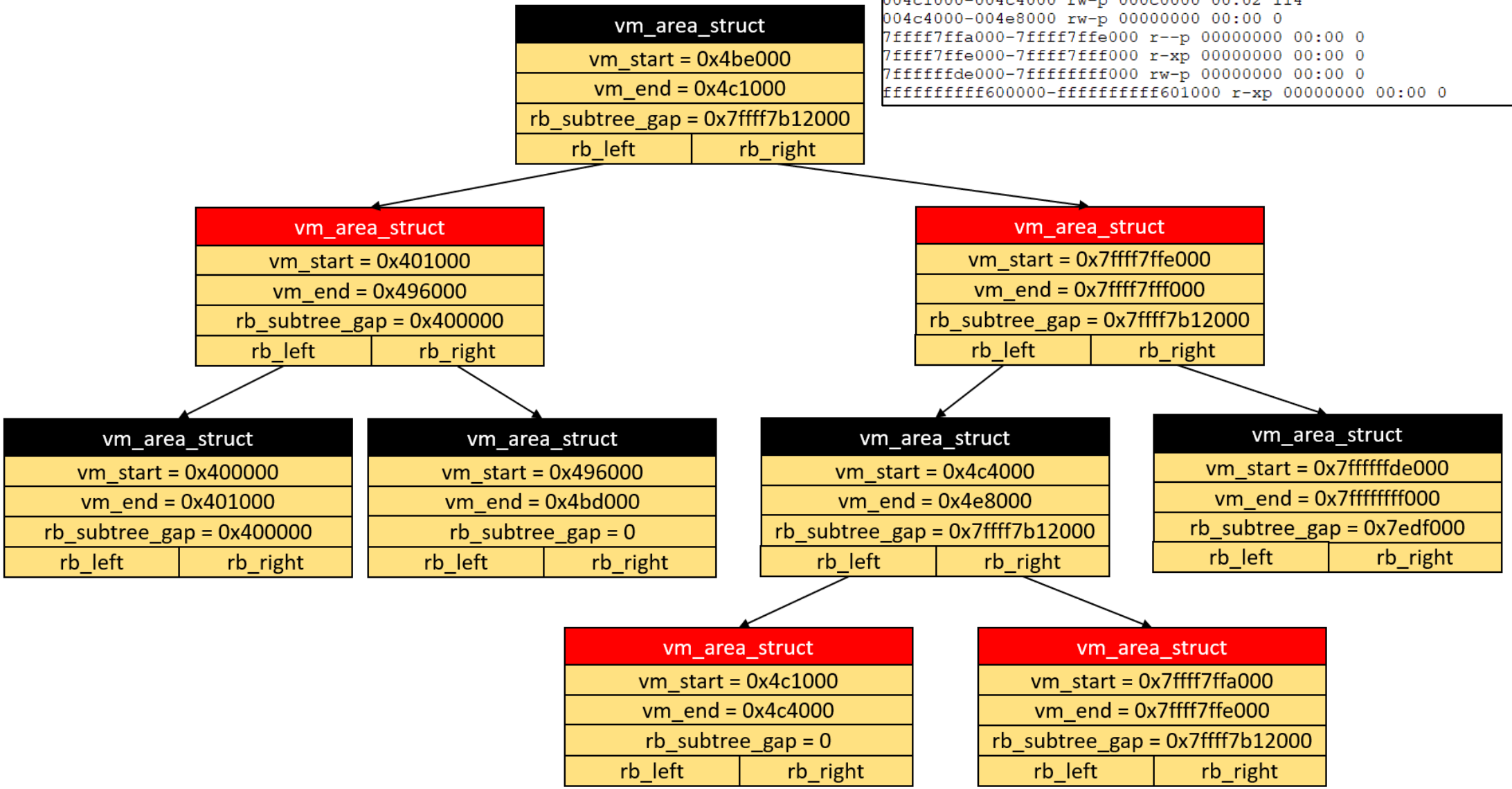
Process address space: Doubly linked-list




```
/ # cat /proc/43/maps
00400000-00401000 r--p 00000000 00:02 114                                /anon_mmap
00401000-00496000 r-xp 00001000 00:02 114                                /anon_mmap
00496000-004bd000 r--p 00096000 00:02 114                                /anon_mmap
004be000-004c1000 r--p 000bd000 00:02 114                                /anon_mmap
004c1000-004c4000 rw-p 000c0000 00:02 114                                /anon_mmap
004c4000-004e8000 rw-p 00000000 00:00 0                                  [heap]
7ffff7ffa000-7ffff7ffe000 r--p 00000000 00:00 0                          [vvar]
7ffff7ffe000-7ffff7fff000 r-xp 00000000 00:00 0                          [vdso]
7fffffffde000-7fffffffff000 rw-p 00000000 00:00 0                        [stack]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

Process address space: VMA rbtree

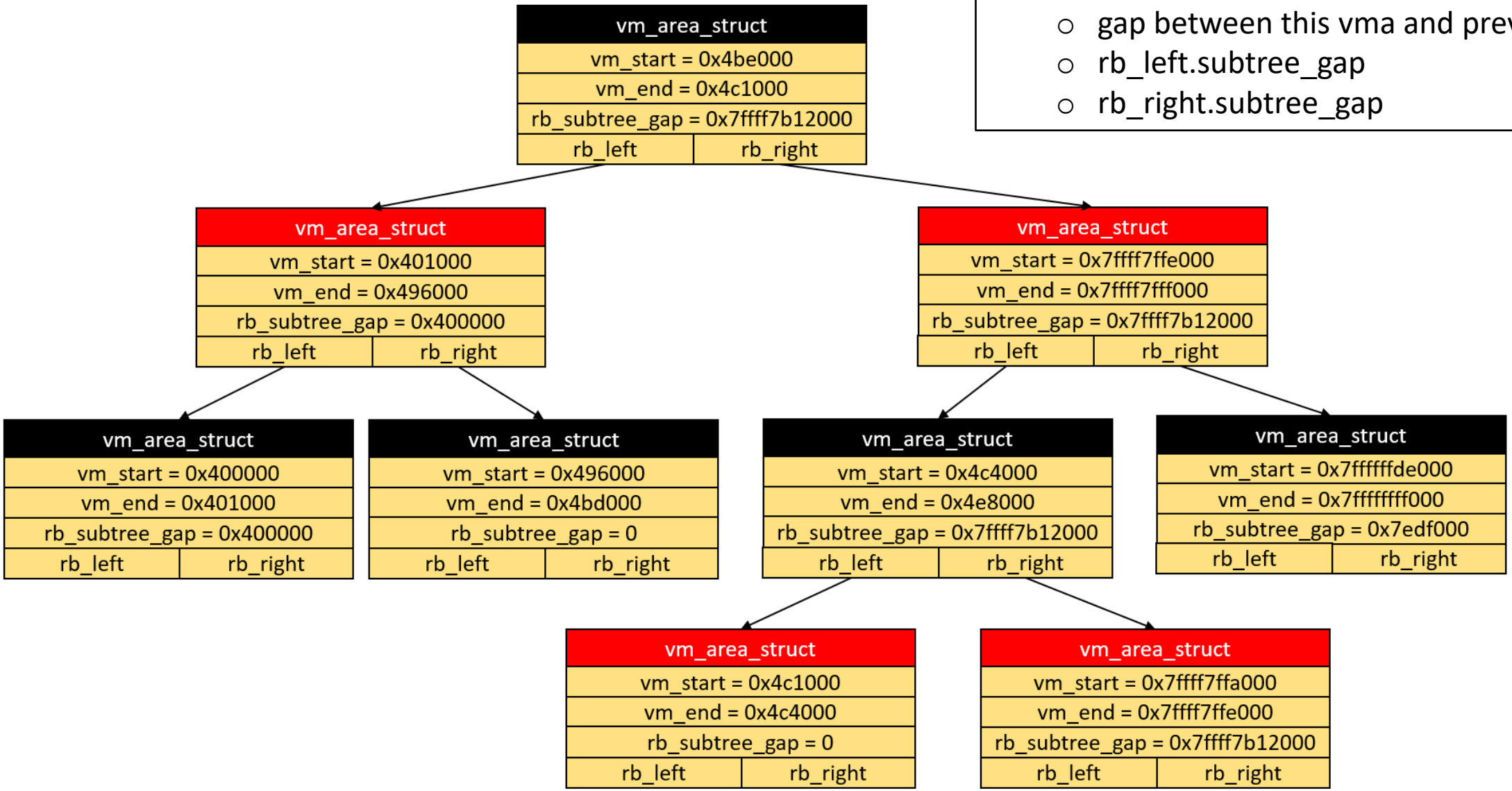
```
/ # cat /proc/43/maps
00400000-00401000 r--p 00000000 00:02 114                                /anon_mmap
00401000-00496000 r-xp 00001000 00:02 114                                /anon_mmap
00496000-004bd000 r--p 00096000 00:02 114                                /anon_mmap
004be000-004c1000 r--p 000bd000 00:02 114                                /anon_mmap
004c1000-004c4000 rw-p 000c0000 00:02 114                                /anon_mmap
004c4000-004e8000 rw-p 00000000 00:00 0                                  [heap]
7ffff7ffa000-7ffff7ffe000 r--p 00000000 00:00 0                          [vvar]
7ffff7ffe000-7ffff7fff000 r-xp 00000000 00:00 0                          [vdso]
7fffffffde000-7ffffffff000 rw-p 00000000 00:00 0                         [stack]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```



Process address space: VMA rbtree -> rb_subtree_gap

rb_subtree_gap calculation

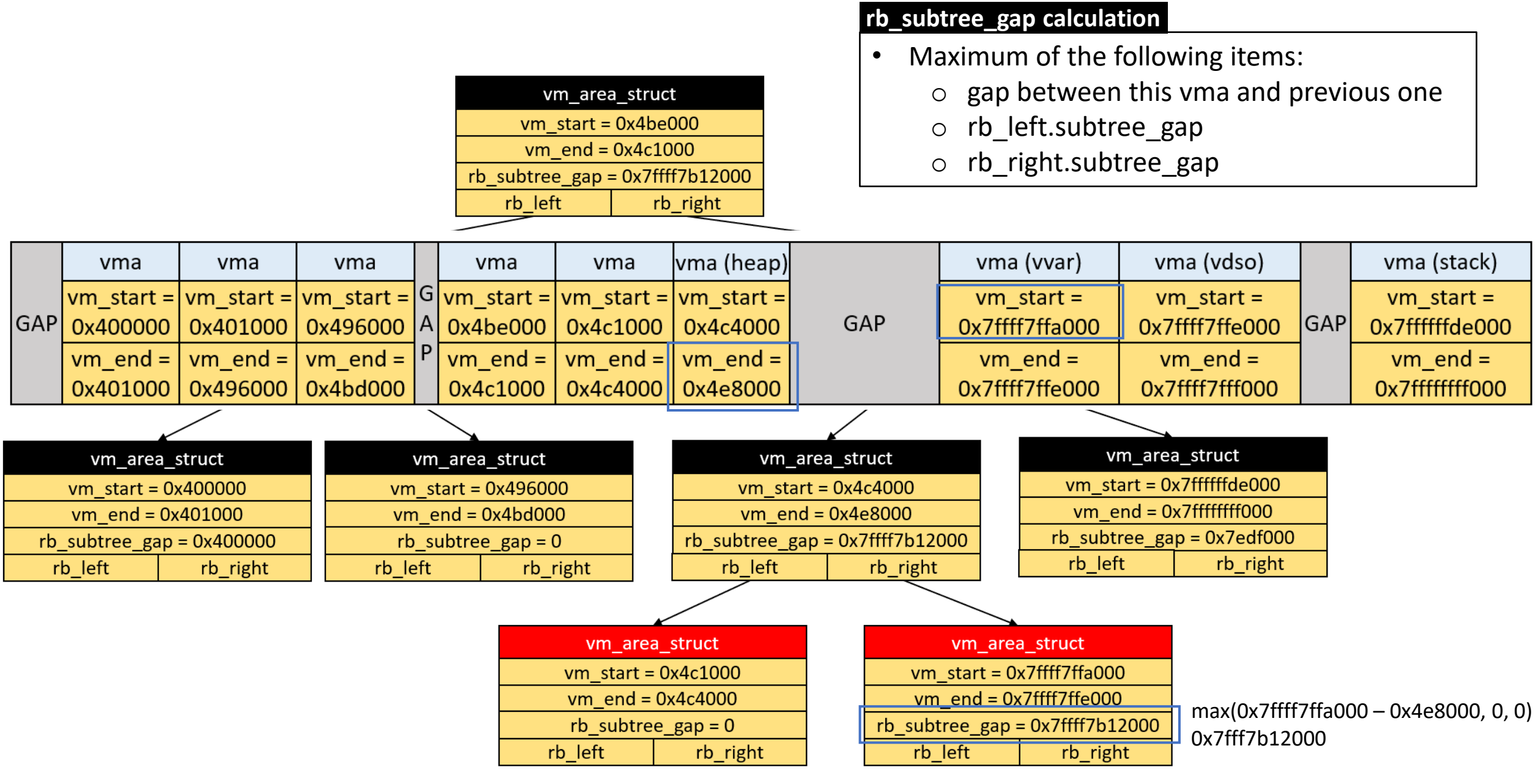
- Maximum of the following items:
  - gap between this vma and previous one
  - rb_left.subtree_gap
  - rb_right.subtree_gap



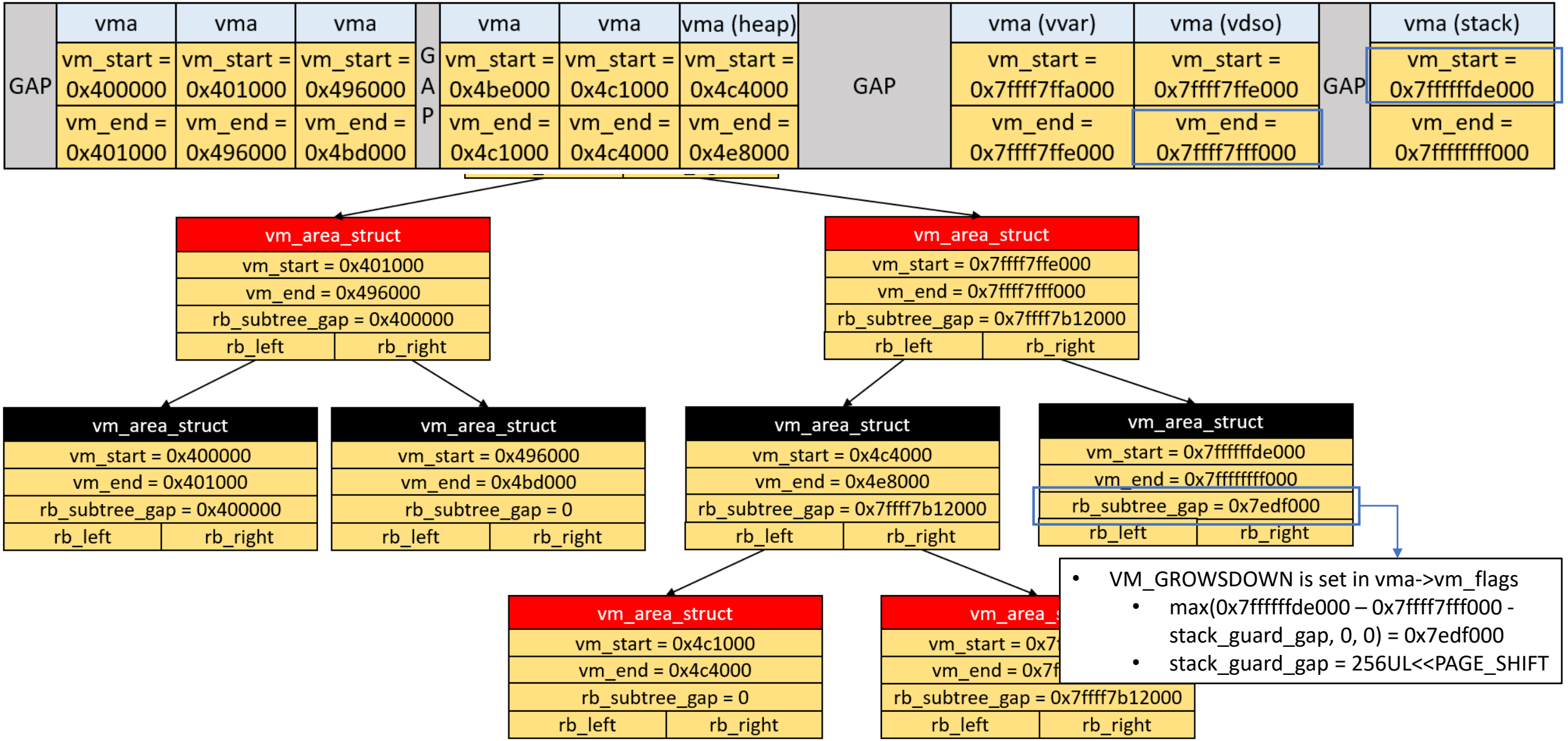
Process address space: VMA rbtree -> rb_subtree_gap

rb_subtree_gap calculation

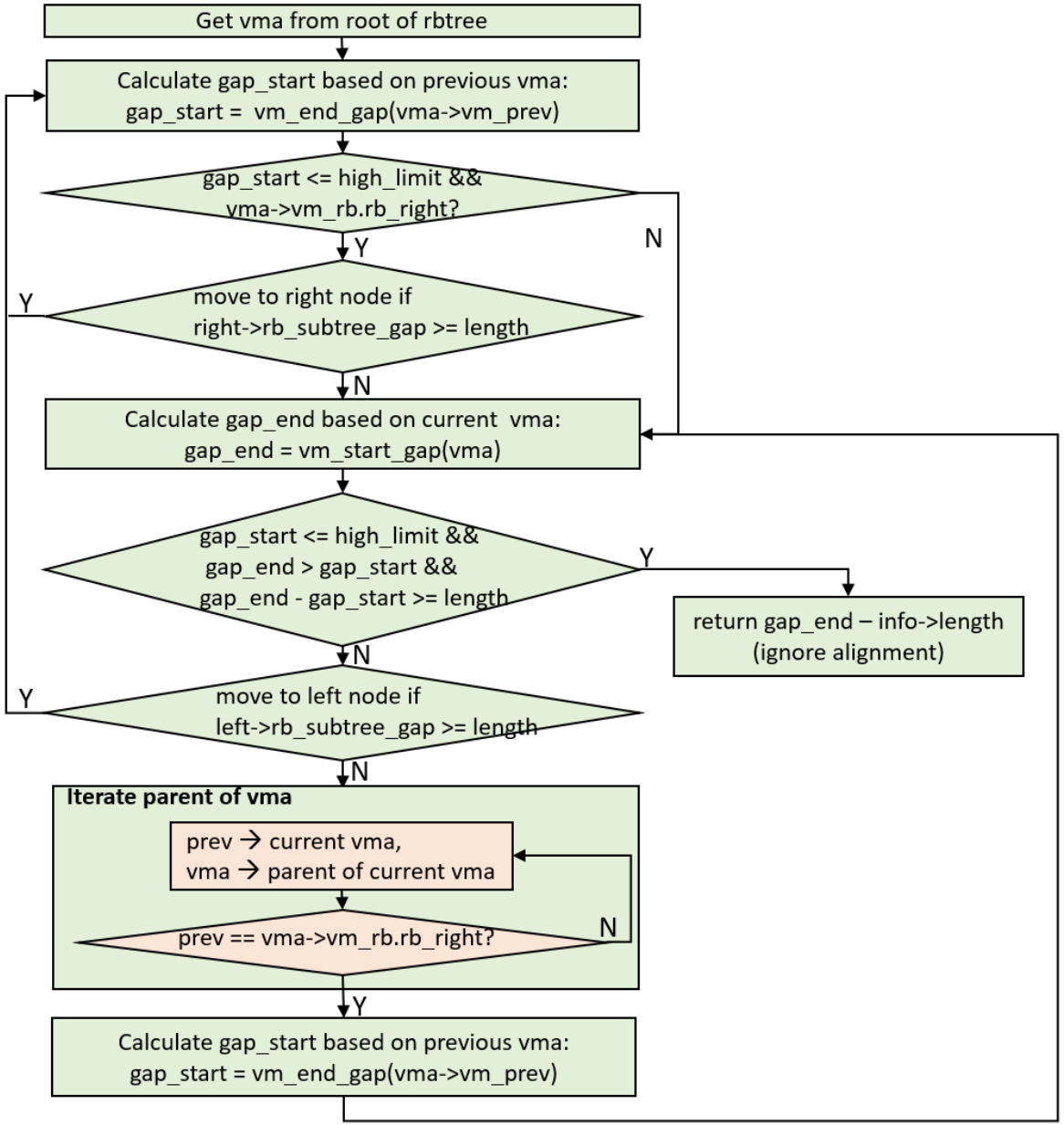
- Maximum of the following items:
  - gap between this vma and previous one
  - rb_left.subtree_gap
  - rb_right.subtree_gap



Process address space: VMA rbtree -> rb_subtree_gap



unmapped_area_topdown()



unmapped_area_topdown() - Principle

```
thp_get_unmapped_area ["mm/huge_memory.c"]
  if (is dax)
    return __thp_get_unmapped_area
  else
    return current->mm->get_unmapped_area

arch_get_unmapped_area_topdown ["arch/x86/kernel/sys_x86_64.c"]
  Prepare a vm_unmapped_area_info struct
  |- addr = vm_unmapped_area(&info)
    |- unmapped_area_topdown
  return addr
```

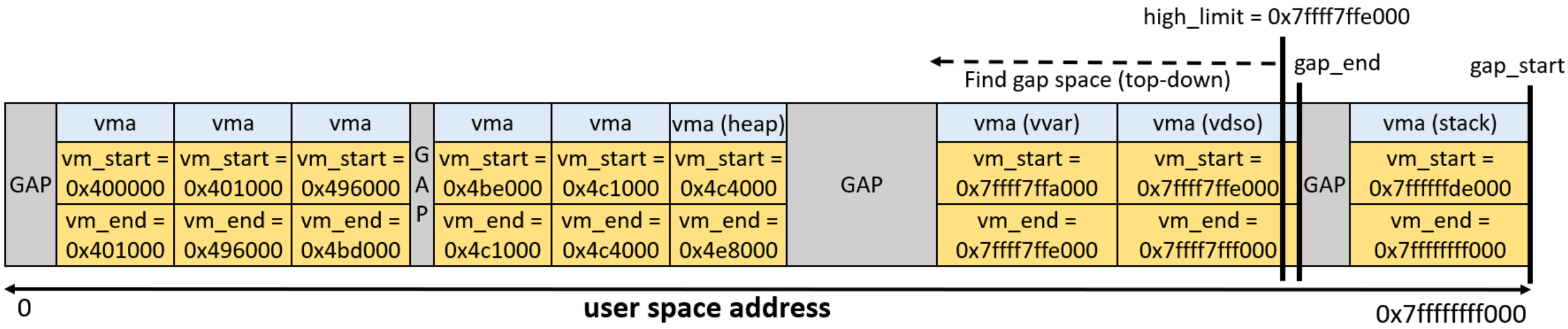
unmapped_area_topdown(): Preparation

vm_unmapped_area_info
length = 0x1000
low_limit = PAGE_SIZE
high_limit = mm->mmap_base = 0x7ffff7fff000

gap_start = mm->highest_vm_end = 0x7fffffff000

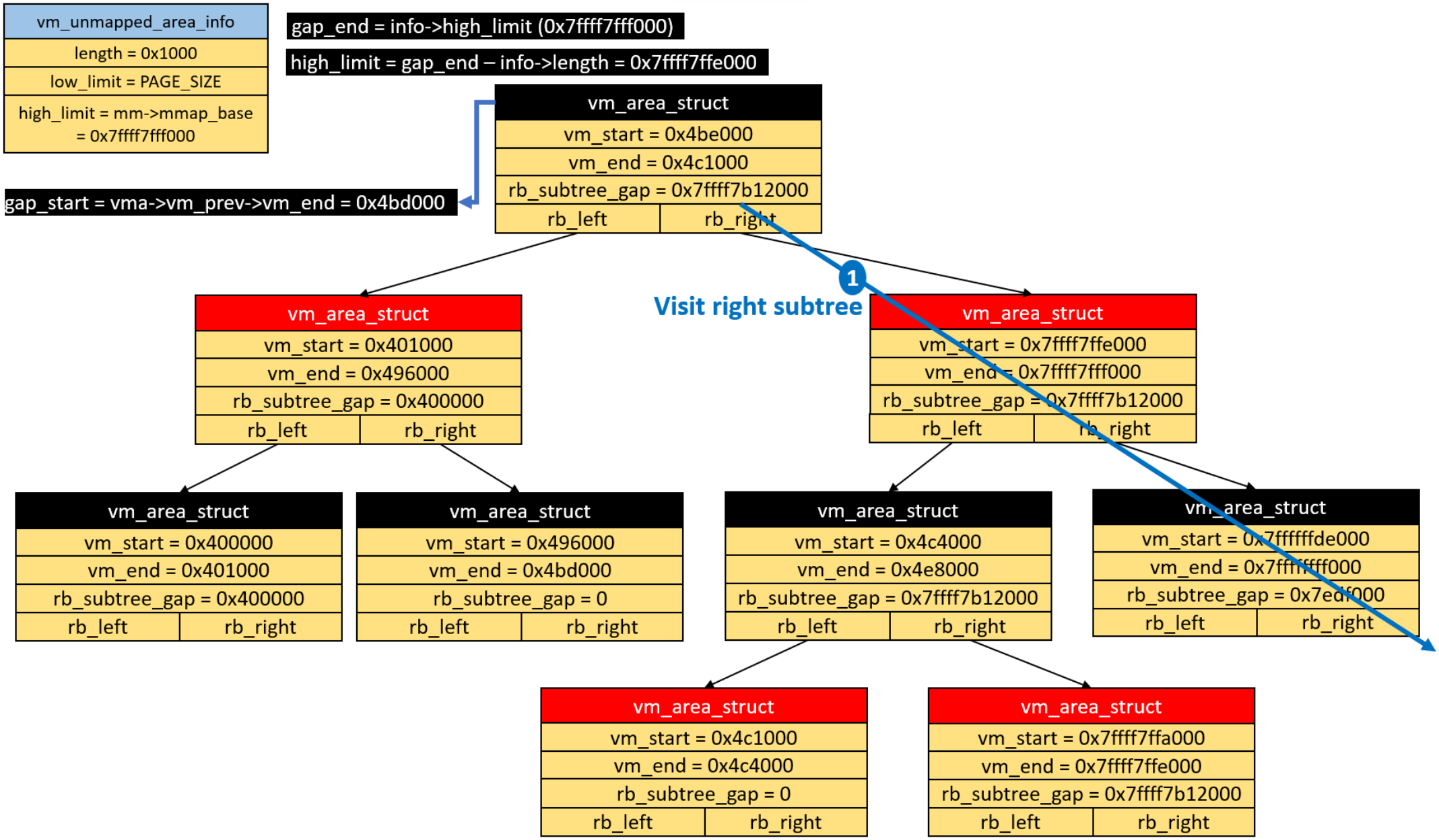
gap_end = info->high_limit (0x7ffff7fff000)

high_limit = gap_end – info->length = 0x7ffff7ffe000

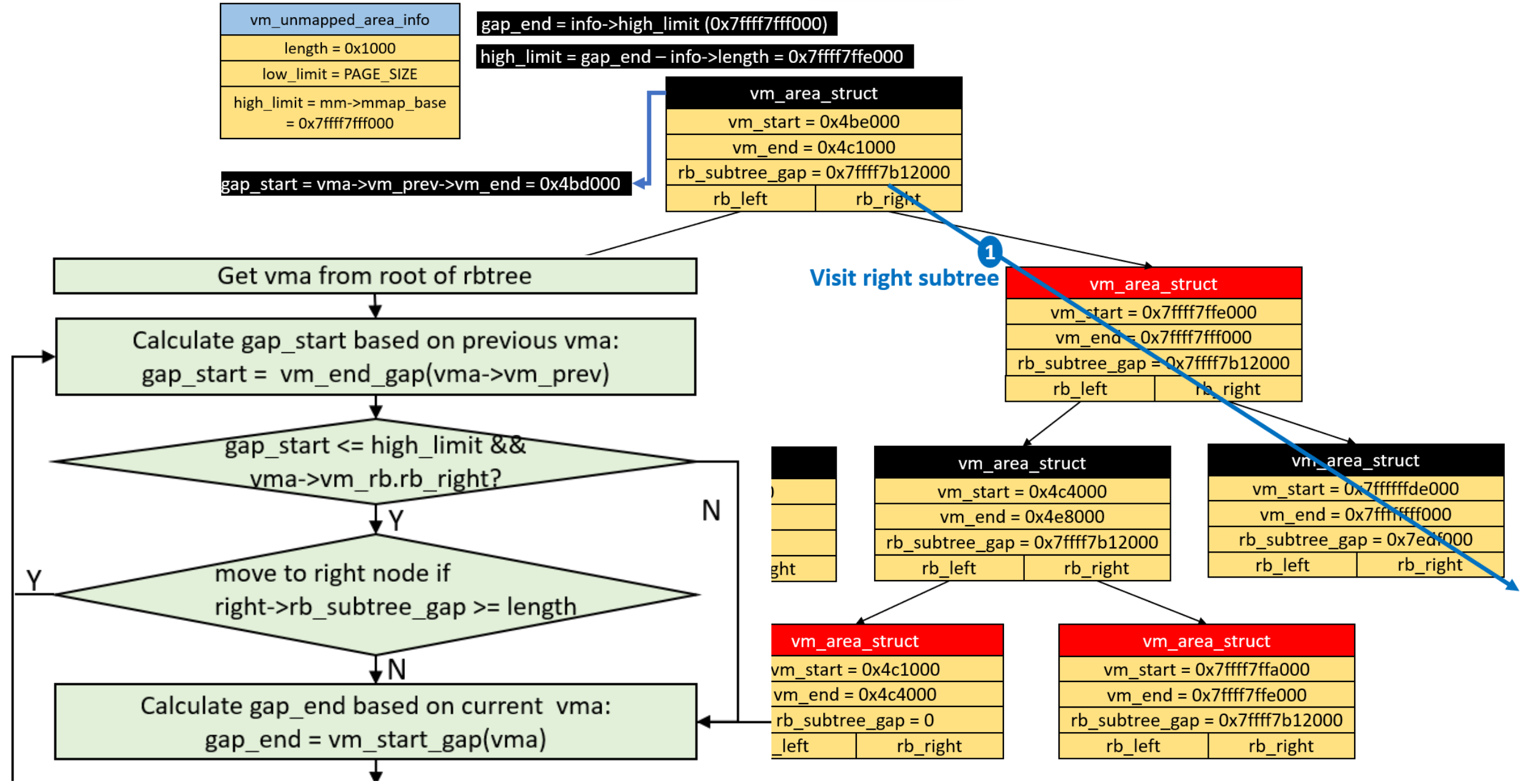


[Efficiency] Traverse vma rbtree to find gap space instead of traversing vma linked list

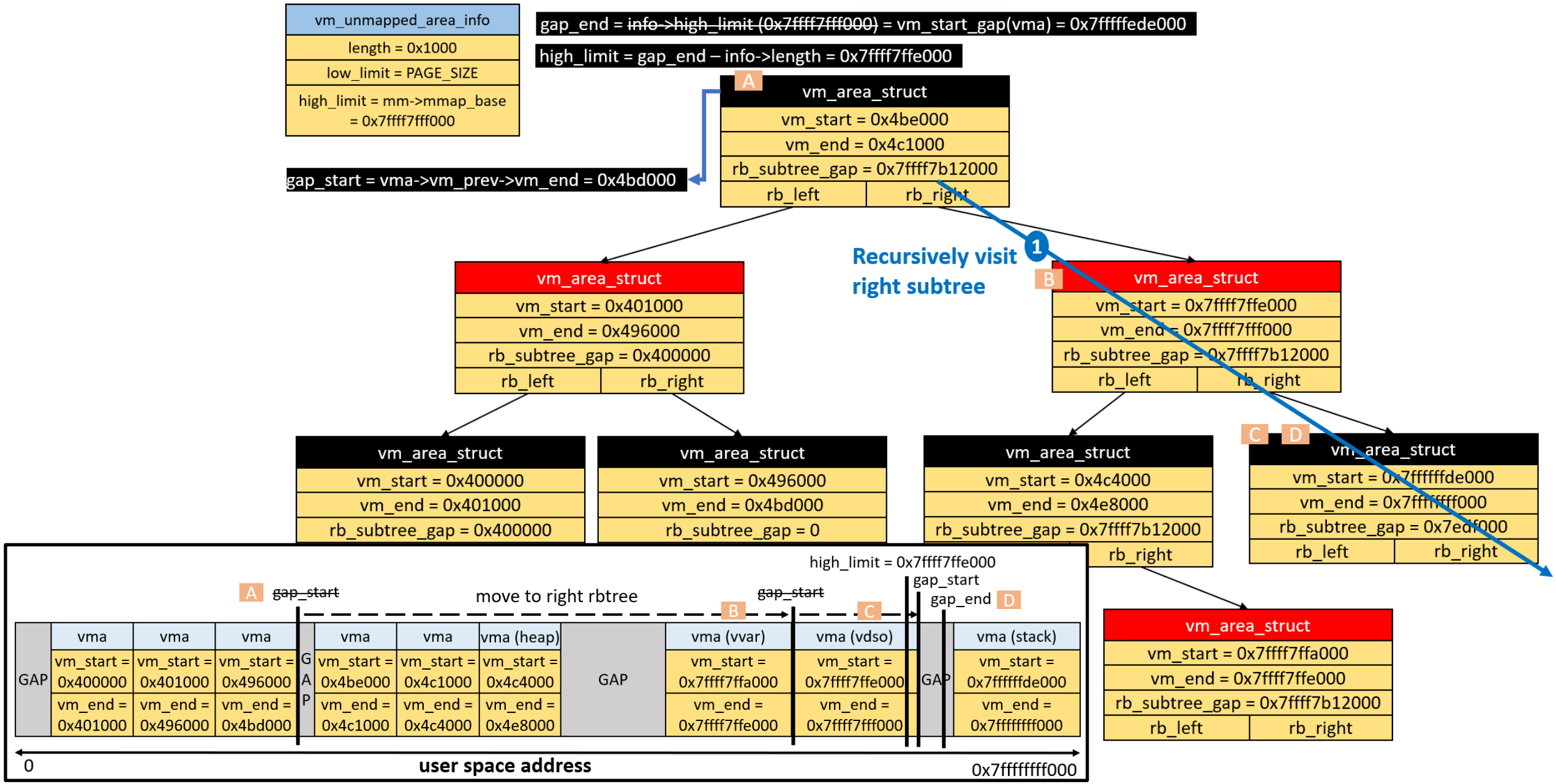
unmapped_area_topdown()



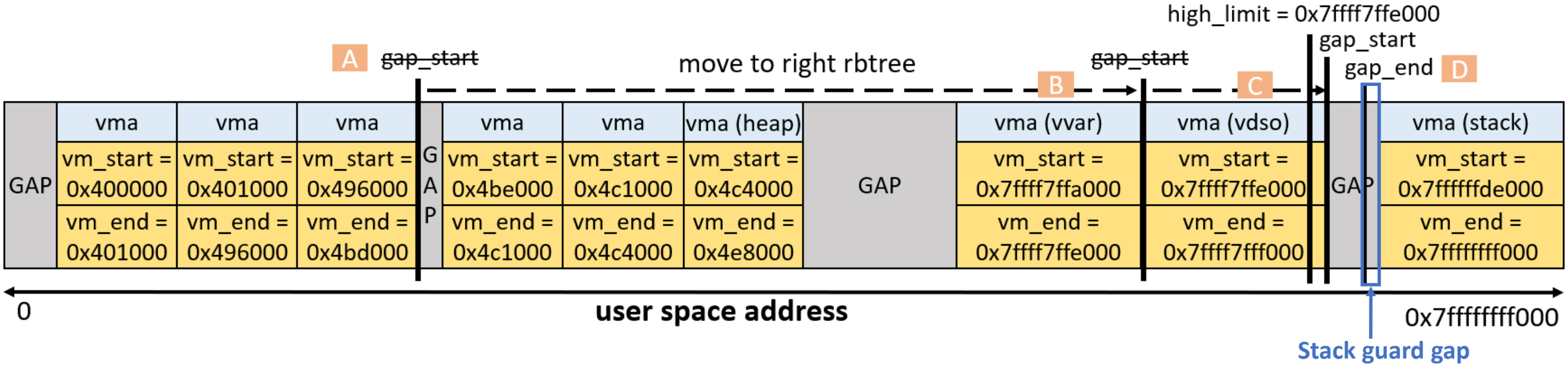
unmapped_area_topdown()



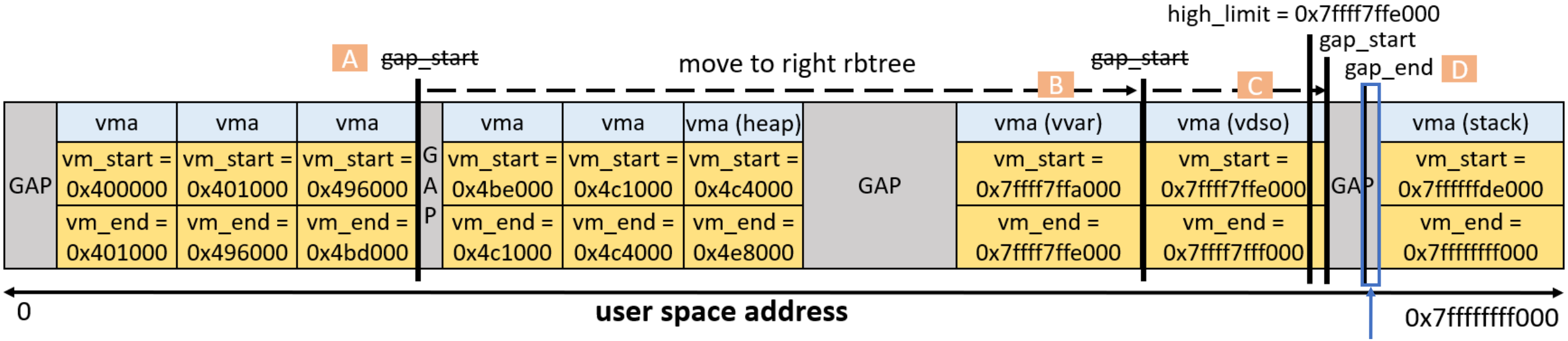
unmapped_area_topdown()

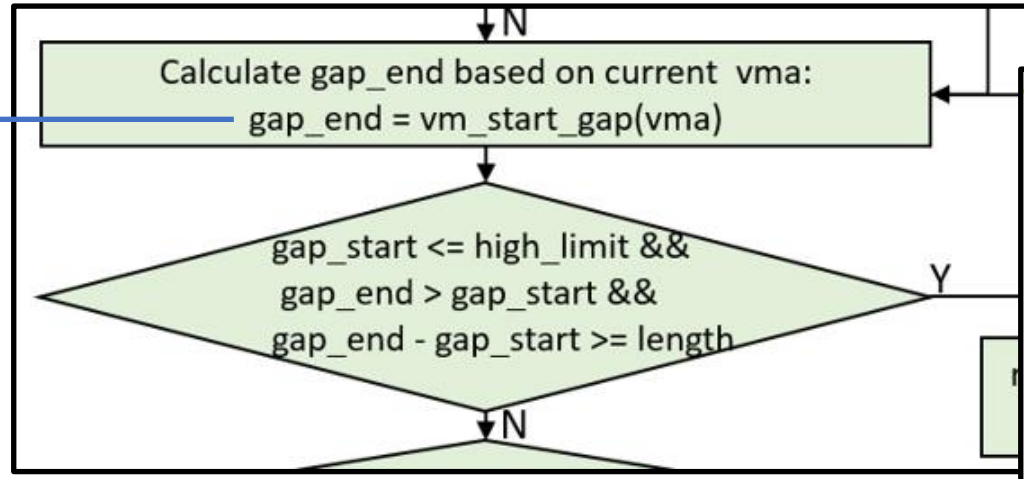


unmapped_area_topdown()



unmapped_area_topdown()



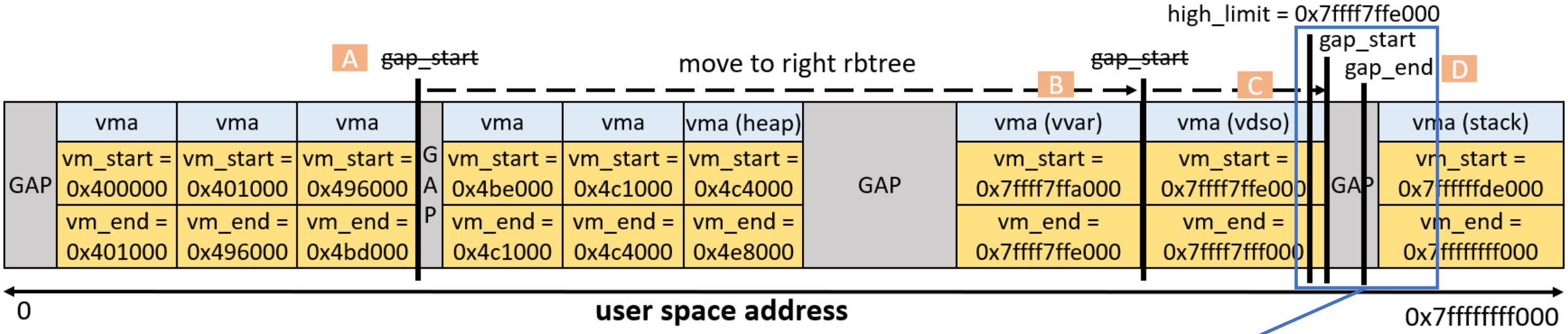



```
static inline unsigned long vm_end_gap(struct vm_area_struct *vma)
{
        unsigned long vm_end = vma->vm_end;

        if (vma->vm_flags & VM_GROWSUP) {
                vm_end += stack_guard_gap;
                if (vm_end < vma->vm_end)
                        vm_end = -PAGE_SIZE;
        }
        return vm_end;
}
include/linux/mm.h                                    2672,1
```

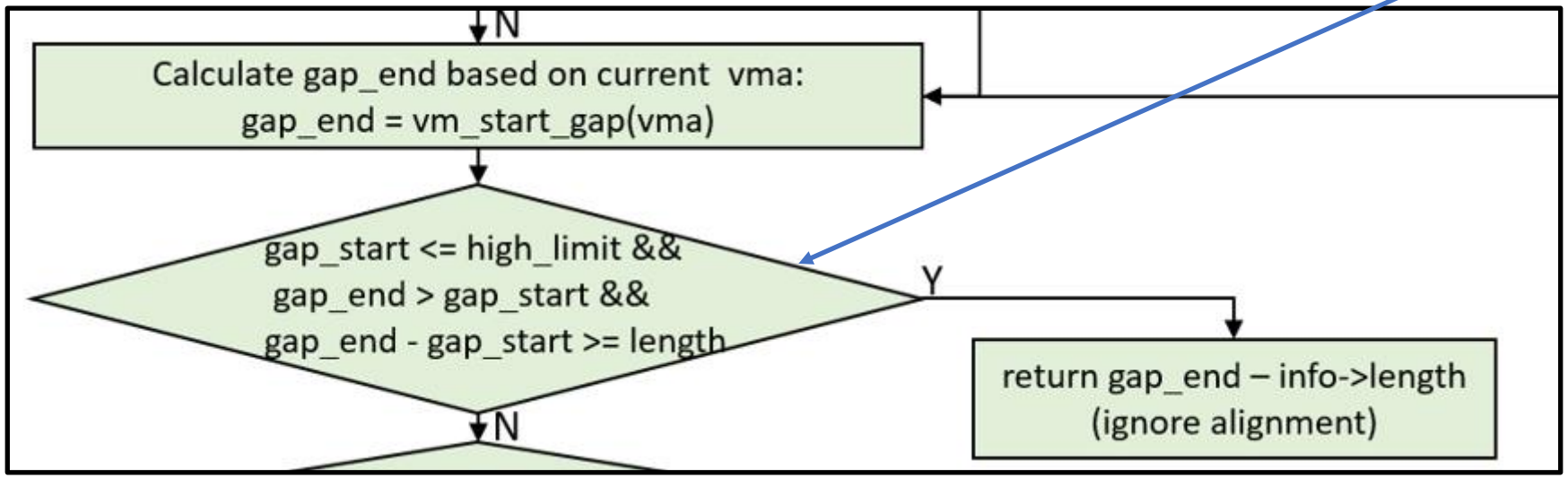
gap_end = 0x7fffffde000 – 0x100000 = 0x7fffffede000

* stack_guard_gap = 256UL<<PAGE_SHIFT = 0x100000

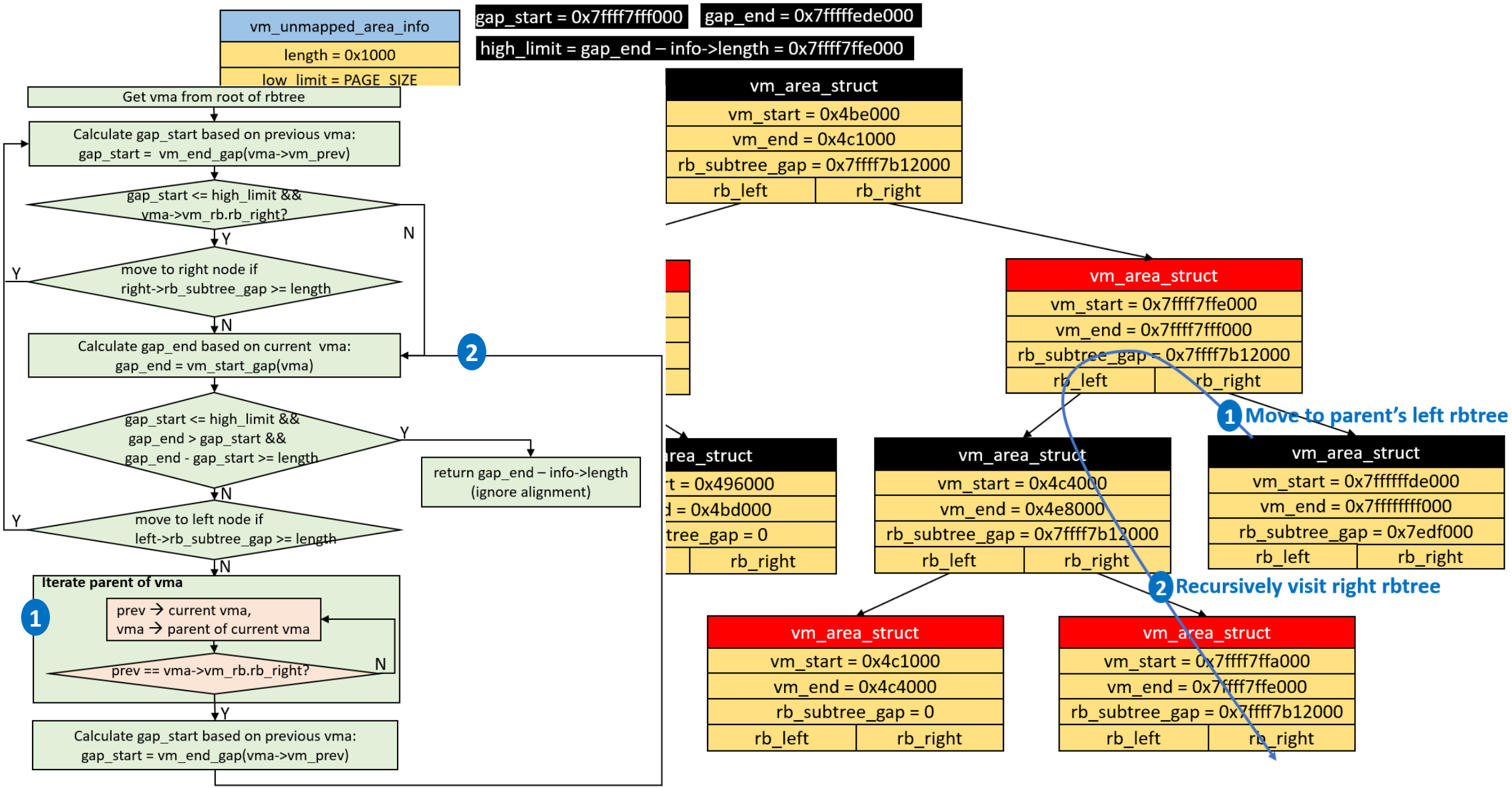
unmapped_area_topdown()



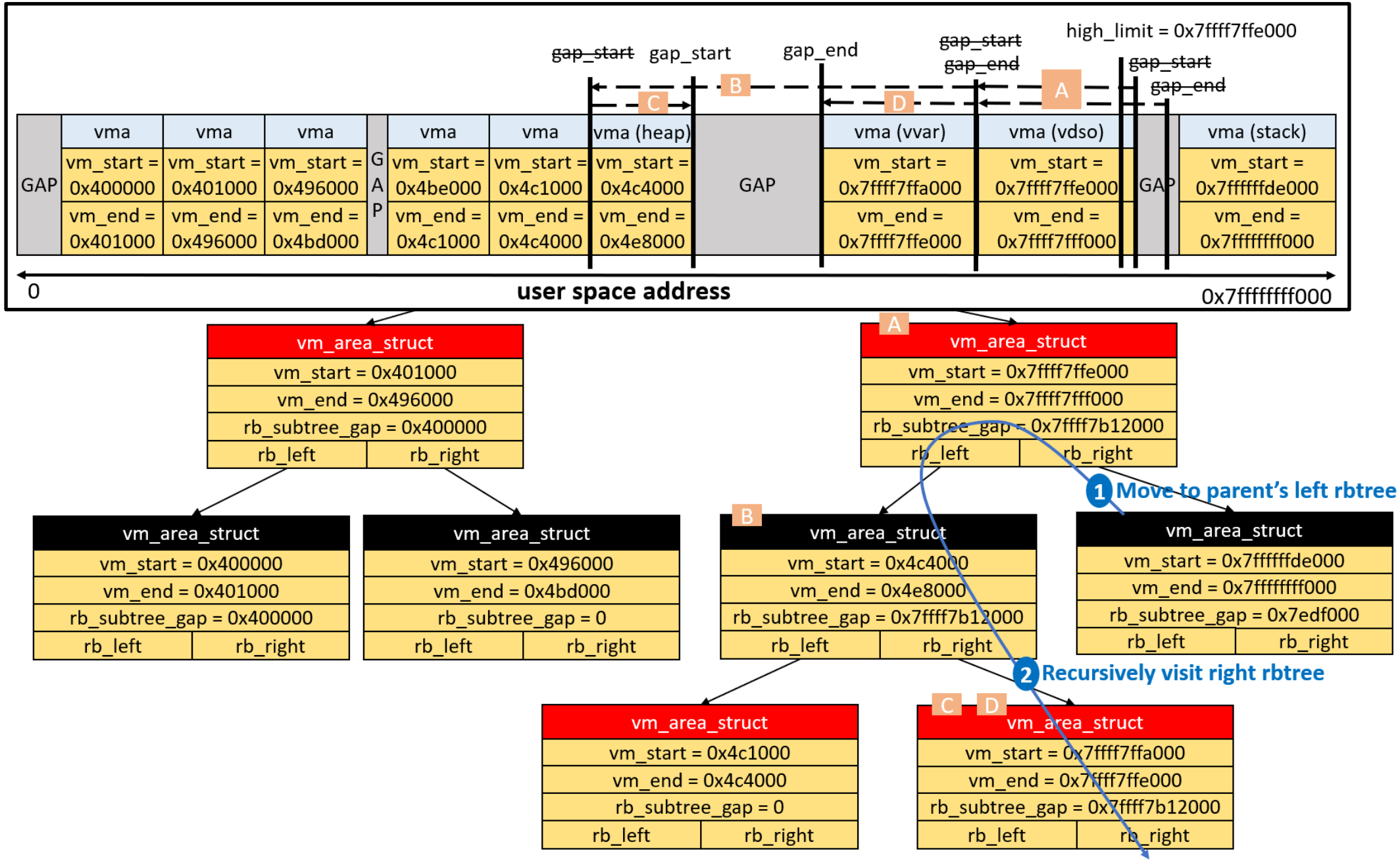
Not found yet



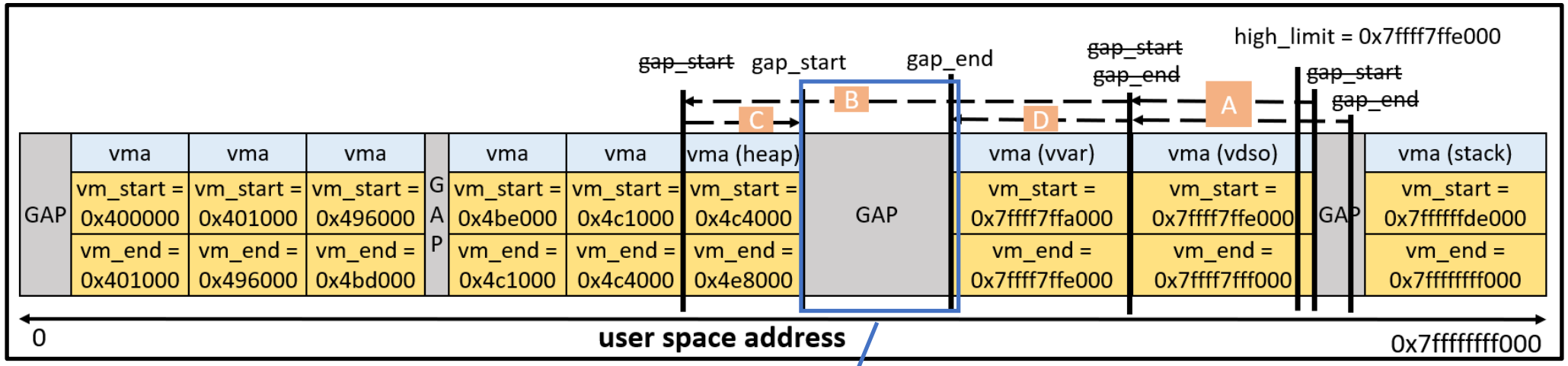
unmapped_area_topdown()



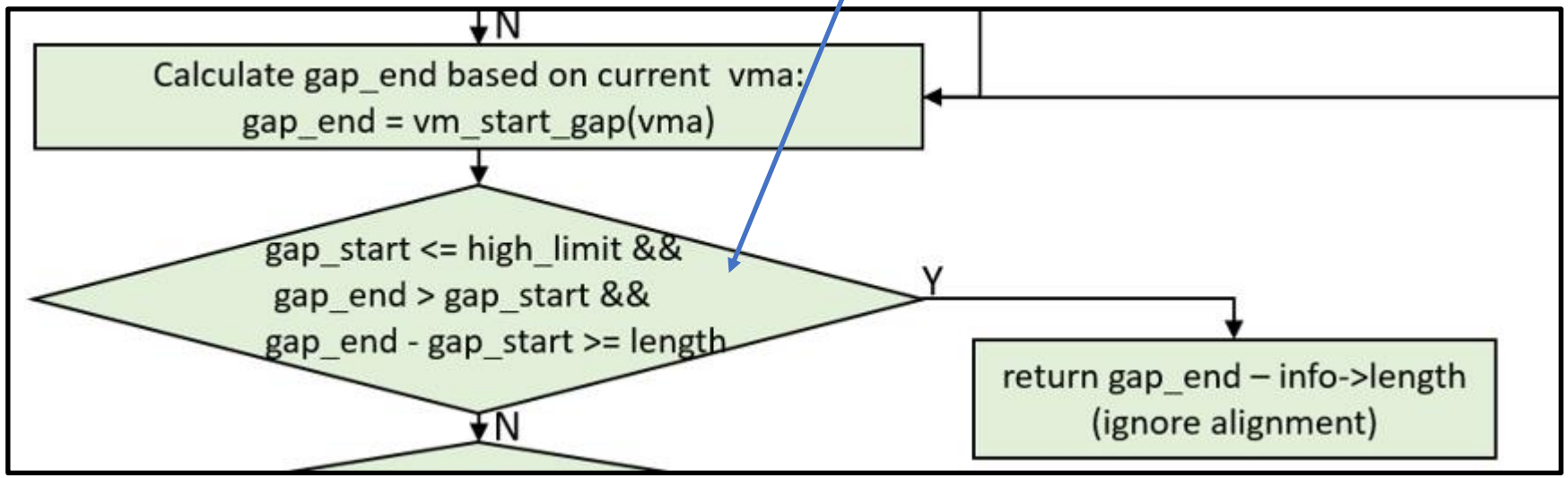
unmapped_area_topdown()



unmapped_area_topdown()

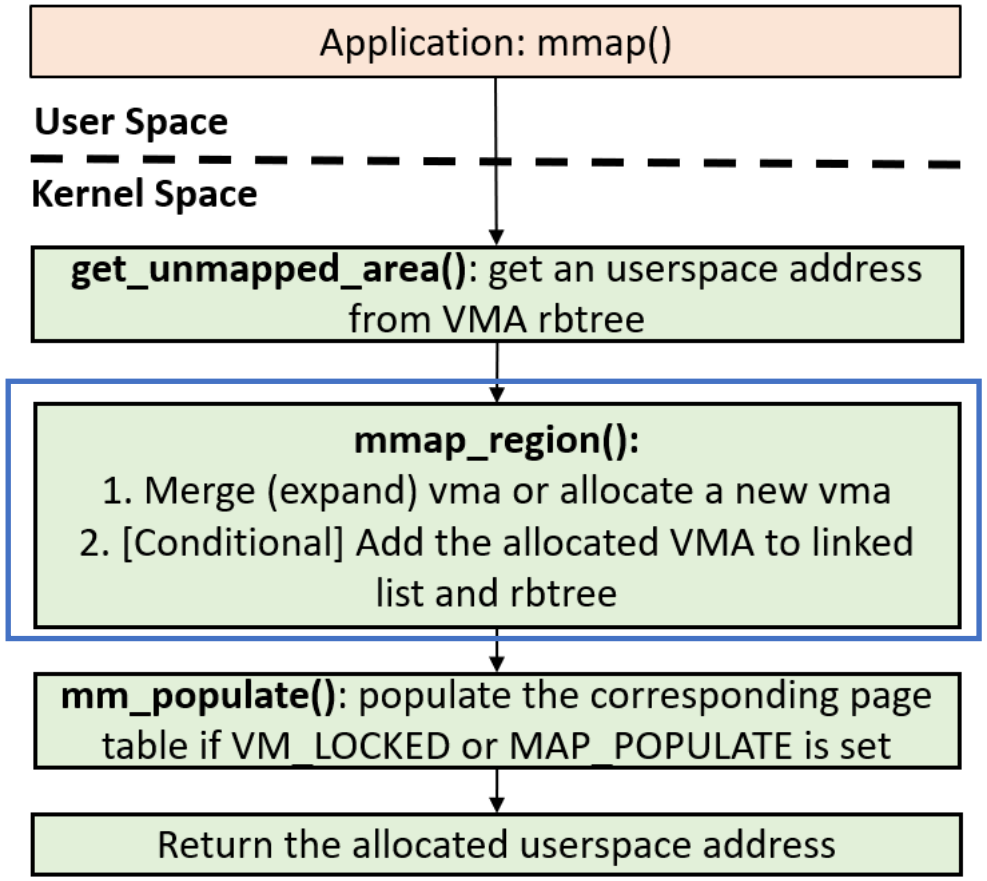


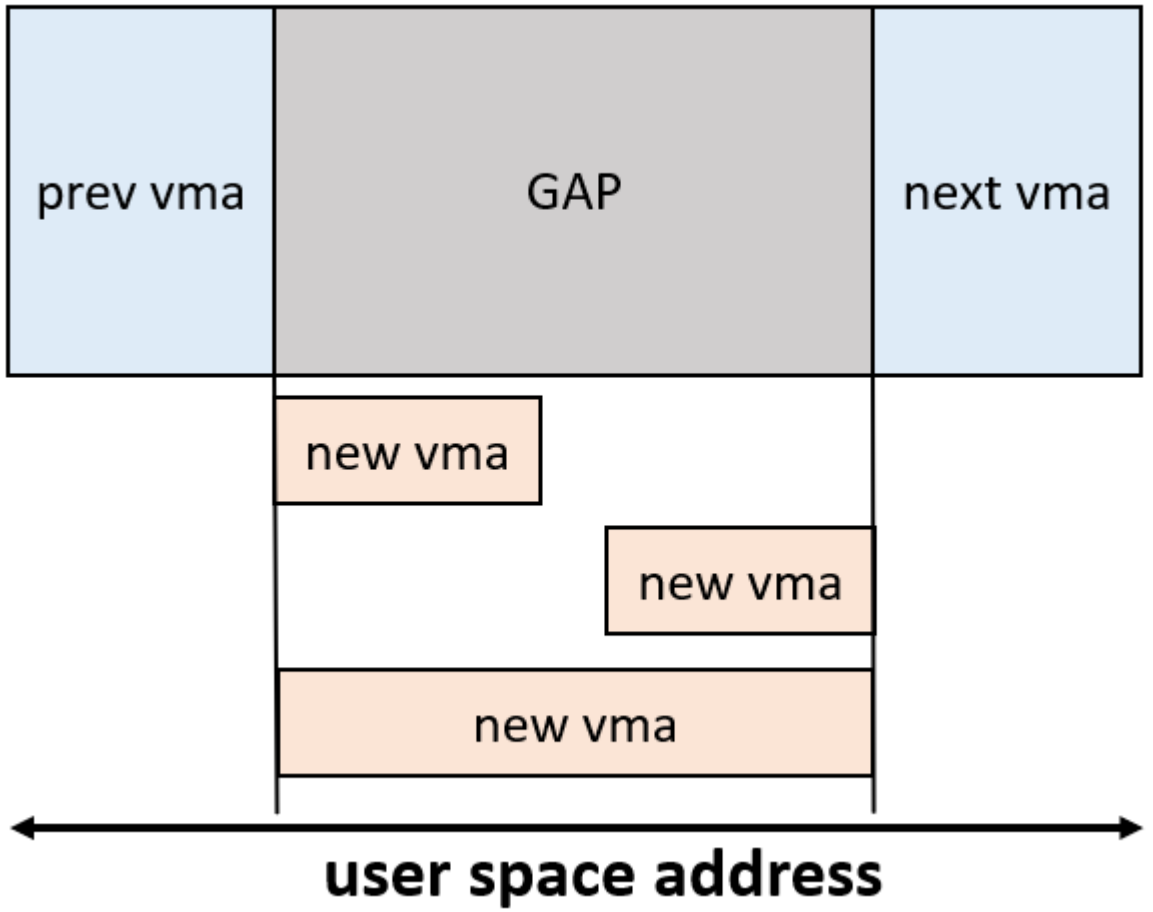
Found



return 0x7ffff7ffa000 – 0x1000 = 0x7ffff7ff9000

vma_merge() – Possible ways to merge





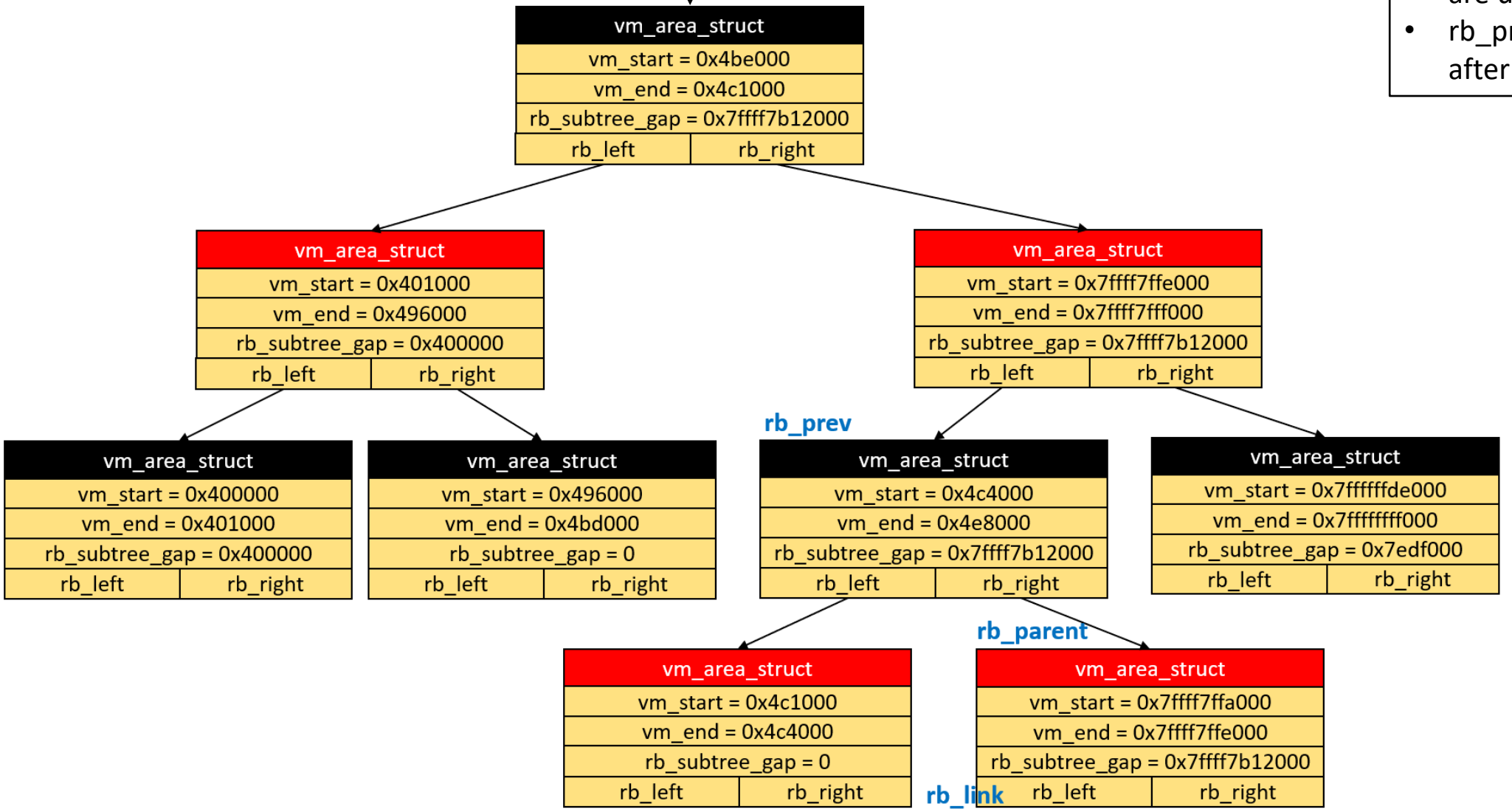
Note: VMAs must have the same attributes

mmap_region()->munmap_vma_range()->find_vma_links

find_vma_links(..., 0x7ffff7ff9000, ...)

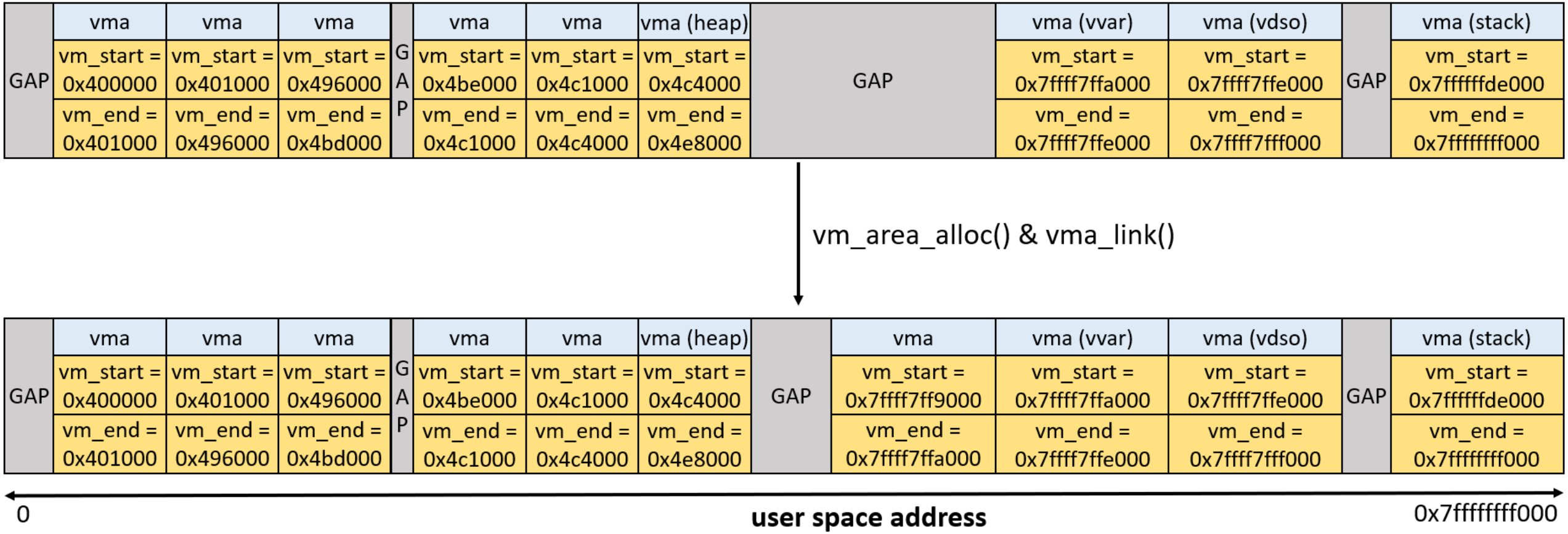
Iterate rbtree to find an empty VMA link (rb_link)
- rb_parent: The parent of rb_link
- rb_prev: Previous vma of rb_parent

Note
- rb_prev is used in vma_merge()
- rb_link, rb_parent and rb_prev are used in vma_link().
- rb_prev is the previous VMA after the new VMA is inserted.

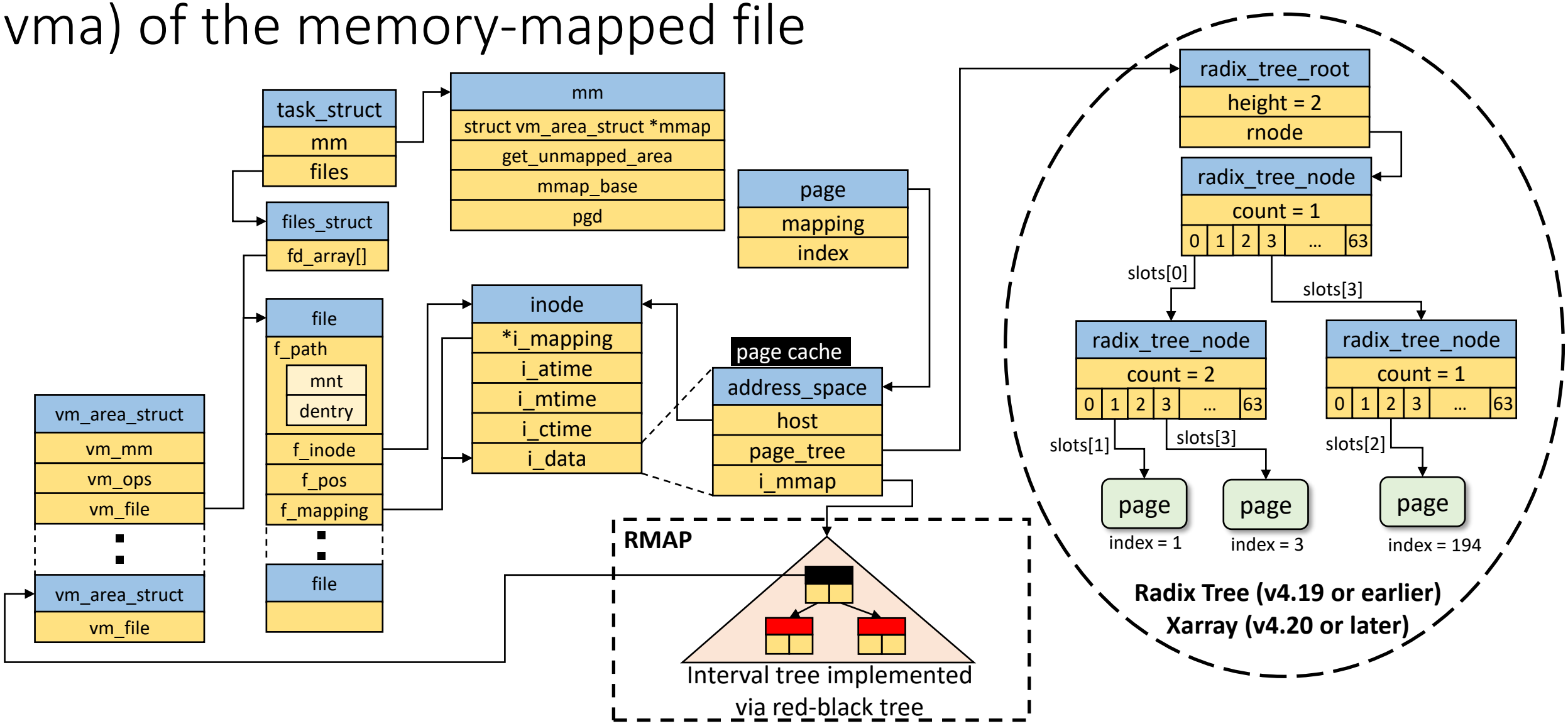


vm_area_alloc() & vma_link()

```
mmap_region ["mm/mmap.c"]
  |- vma_link
    |- __vma_link
      |- __vma_link_list
      |- __vma_link_rb
    |- __vma_link_file
      |- vma_interval_tree_insert
```

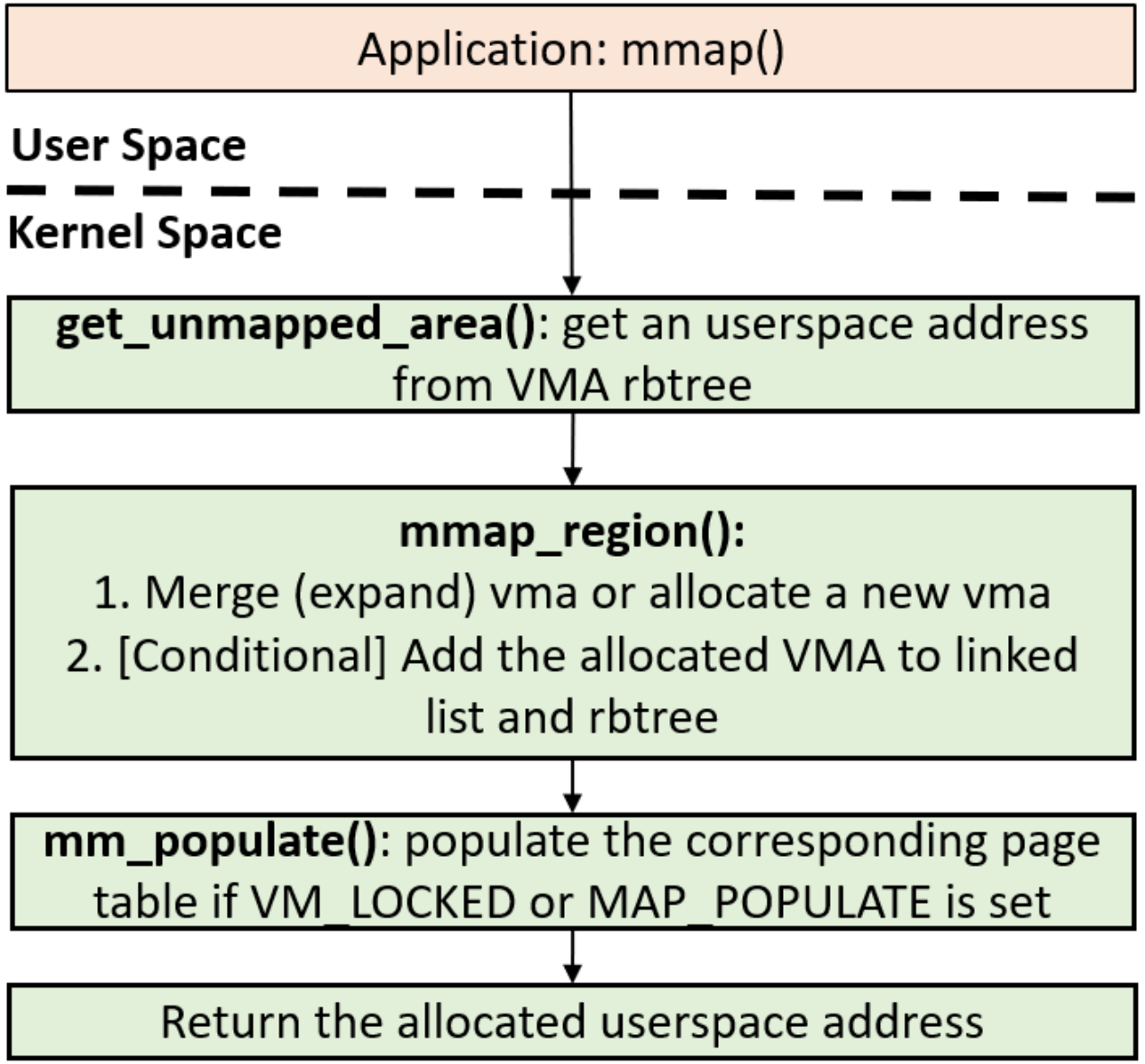


__vma_link_file(): i_mmap for recording all memory mappings (vma) of the memory-mapped file



1. **i_mmap: Reverse mapping (RMAP) for the memory-mapped file (page cache) → check __vma_link_file ()**
2. **anon_vma: Reverse mapping for anonymous pages -> anon_vma_prepare() invoked during page fault**

mm_populate()


Application: mmap()
User Space
Kernel Space
get_unmapped_area(): get an userspace address from VMA rbtree
mmap_region():
1. Merge (expand) vma or allocate a new vma
2. [Conditional] Add the allocated VMA to linked list and rbtree
mm_populate(): populate the corresponding page table if VM_LOCKED or MAP_POPULATE is set
Return the allocated userspace address

Anonymous Page: Page Fault – Discussion List

- Private (MAP_PRIATE)
  - Write
  - Read before a write
- Share (MAP_SHARED)

Demand Paging: page fault – Anonymous page (MAP_PRIVATE)

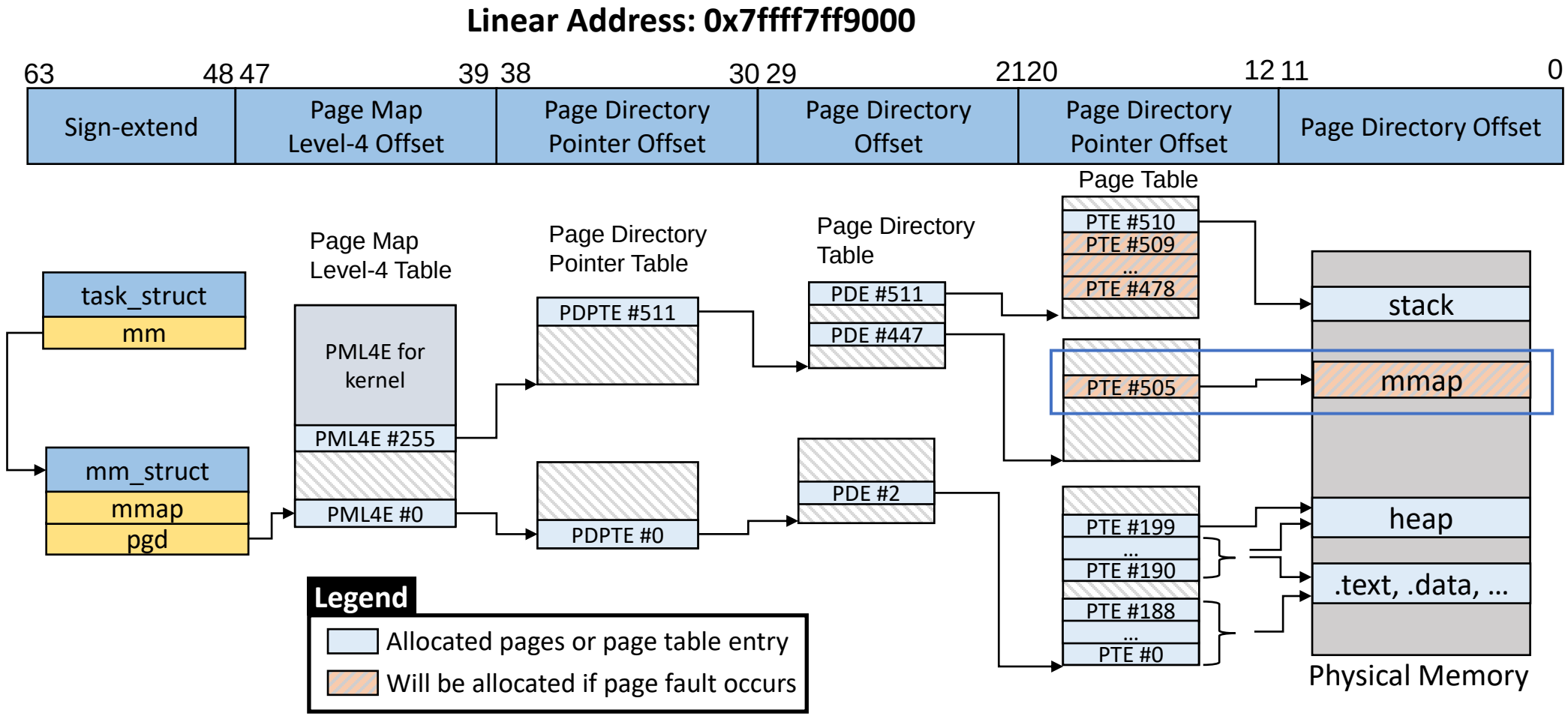
```
(gdb) bt
#0  do_user_addr_fault (regs=regs@entry=0xffffc90000077f58,
    hw_error_code=hw_error_code@entry=6, address=address@entry=140737354108928)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include
/asm/current.h:15
#1  0xffffffff81190a4b in handle_page_fault (address=140737354108928,
    error_code=6, regs=0xffffc90000077f58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1450
#2  exc_page_fault (regs=0xffffc90000077f58, error_code=6)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1506
#3  0xffffffff81200a6b in asm_exc_page_fault ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include
/asm/idtentry.h:580
#4  0x0000000000000000 in ?? ()
(gdb) p /x address
$8 = 0x7ffff7ff9000
```

```
int *addr;

addr = mmap(NULL, len, PROT_READ | PROT_WRITE,
            MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
*addr = 10;
```

```
/*
 * Page fault error code bits:
 *
 *   bit 0 ==     0: no page found       1: protection fault
 *   bit 1 ==     0: read access         1: write access
 *   bit 2 ==     0: kernel-mode access  1: user-mode access
 *   bit 3 ==                            1: use of reserved bit detected
 *   bit 4 ==                            1: fault was an instruction fetch
 *   bit 5 ==                            1: protection keys block access
 *   bit 15 ==                           1: SGX MMU page-fault
 */
enum x86_pf_error_code {
        X86_PF_PROT     =               1 << 0,
        X86_PF_WRITE    =               1 << 1,
        X86_PF_USER     =               1 << 2,
        X86_PF_RSVD     =               1 << 3,
        X86_PF_INSTR    =               1 << 4,
        X86_PF_PK       =               1 << 5,
        X86_PF_SGX      =               1 << 15,
};
```

```
do_user_addr_fault [arch/x86/mm/fault.c]
  |- find_vma
  |- handle_mm_fault
    /* Allocate corresponding page tables */
    if (huge page)
      |- hugetlb_fault
    else
      |- __handle_mm_fault
        /*
         * 1. Allocate pgd, PDPTE, PDE if needed
         * 2. handle_pte_fault():
         *     anonumous page, file mmaped page,
         *     cow fault, shared fault, swap page,
         *     or numa page?
         */
```

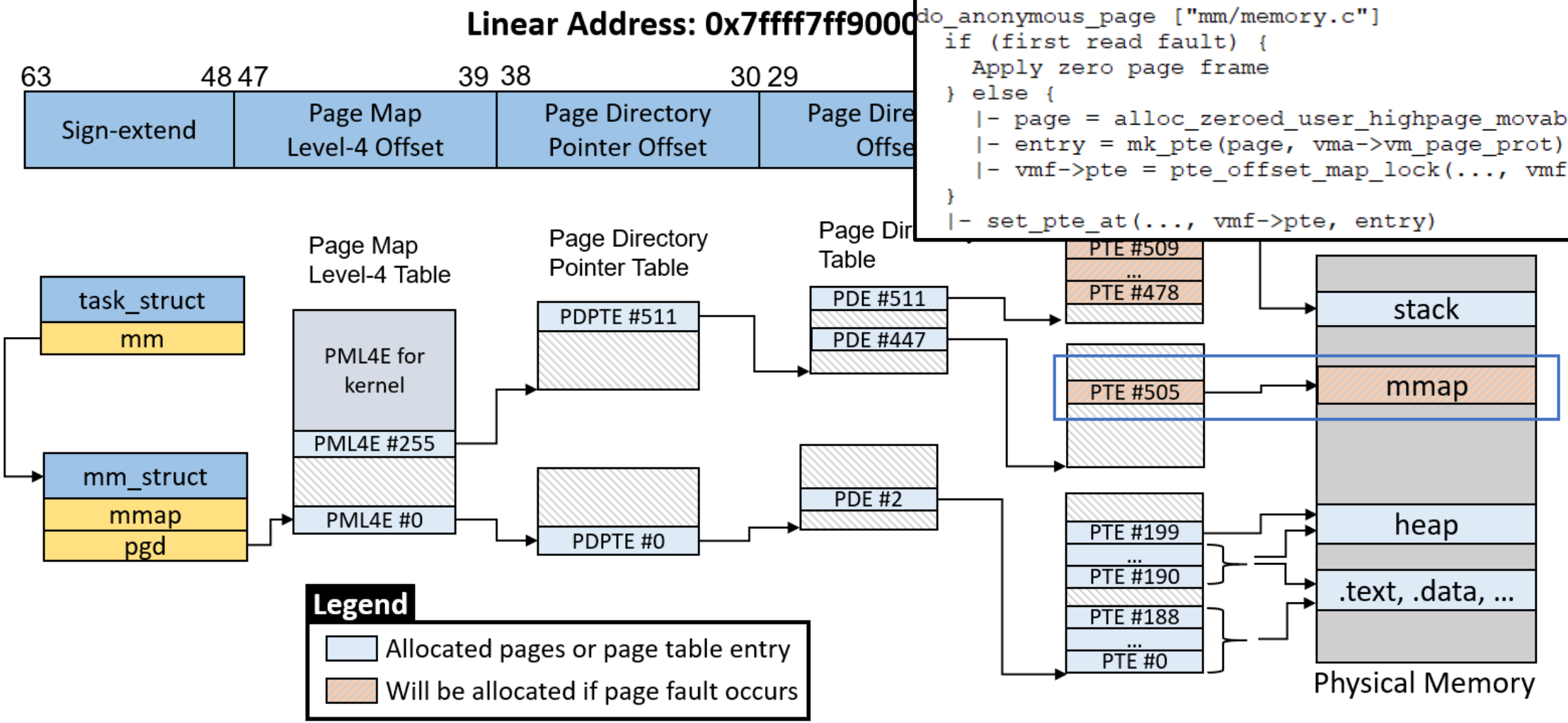
Demand Paging: page fault → Page table configuration - Anonymous page (MAP_PRIVATE)



Demand Paging: page fault → Page table configuration

[anonymous page: MAP_PRIVATE] page fault flow

```
/*
 * 1. Allocate a page (struct page *) or apply
 *    zero page frame
 * 2. Find the pte entry address
 * 3. Configure pte entry
 */
do_anonymous_page ["mm/memory.c"]
  if (first read fault) {
    Apply zero page frame
  } else {
    |- page = alloc_zeroed_user_highpage_movable()
    |- entry = mk_pte(page, vma->vm_page_prot)
    |- vmf->pte = pte_offset_map_lock(..., vmf->address, ...)
  }
  |- set_pte_at(..., vmf->pte, entry)
```



Demand Paging: page fault → Page table configuration - Anonymous page (MAP_PRIVATE)

man mmap

```
MAP_ANONYMOUS
       The mapping is not backed by any file; its contents are initial-
       ized  to zero.  The fd argument is ignored; however, some imple-
       mentations require fd to be -1 if MAP_ANONYMOUS (or MAP_ANON) is
       specified,  and  portable  applications should ensure this.  The
       offset argument should be zero.  The  use  of  MAP_ANONYMOUS  in
       conjunction  with  MAP_SHARED  is  supported on Linux only since
       kernel 2.4.
```

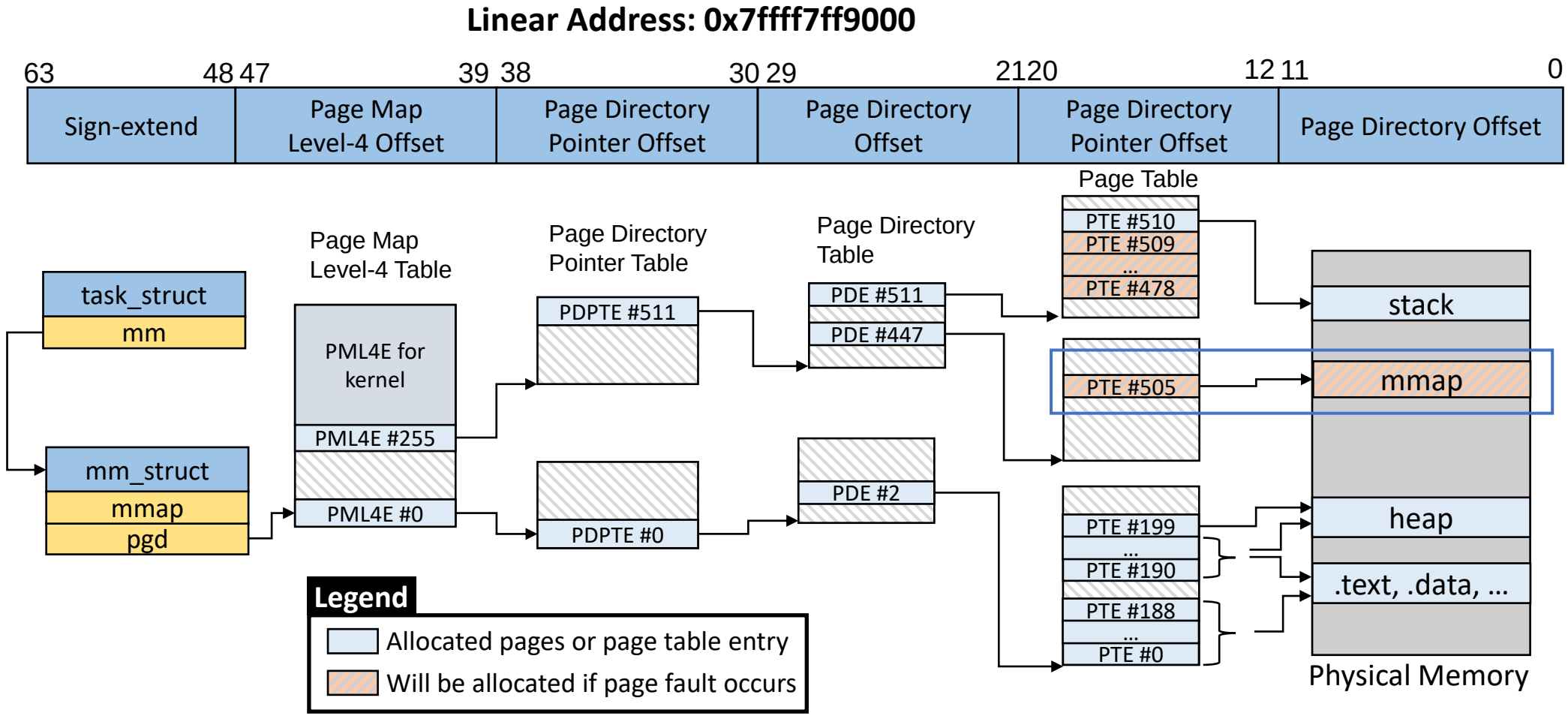
[anonymous page: MAP_PRIVATE] page fault flow

```
/*
 * 1. Allocate a page (struct page *) or apply
 *     zero page frame
 * 2. Find the pte entry address
 * 3. Configure pte entry
 */
do_anonymous_page ["mm/memory.c"]
 if (first read fault) {
   Apply zero page frame
 } else {
   |- page = alloc_zeroed_user_highpage_movable()
   |- entry = mk_pte(page, vma->vm_page_prot)
   |- vmf->pte = pte_offset_map_lock(..., vmf->address, ...)
 }
 |- set_pte_at(..., vmf->pte, entry)
```

[gdb] backtrace

```
(gdb) bt
#0  0xffffffff810a736c in clear_page (page=<optimized out>) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include/asm/page_64.h:49
#1  clear_highpage (page=0xffffea000900cb00) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/highmem.h:203
#2  kernel_init_free_pages (numpages=<optimized out>, page=<optimized out>) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/page_alloc.c:1212
#3  post_alloc_hook (gfp_flags=1052106, order=0, page=0xffffea000900cb00) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/page_alloc.c:2300
#4  prep_new_page (alloc_flags=2305, gfp_flags=1052106, order=0, page=0xffffea000900cb00) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/page_alloc.c:2306
#5  get_page_from_freelist (gfp_mask=gfp_mask@entry=1052106, order=order@entry=0, alloc_flags=2305, ac=ac@entry=0xffffc9000007fde0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/page_alloc.c:3945
#6  0xffffffff810a81db in __alloc_pages_nodemask (gfp_mask=gfp_mask@entry=1052106, order=order@entry=0, preferred_nid=preferred_nid@entry=0, nodemask=nodemask@entry=0x0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/page_alloc.c:4995
#7  0xffffffff81092f85 in __alloc_pages (preferred_nid=0, order=0, gfp_mask=1052106) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/gfp.h:538
#8  __alloc_pages_node (order=0, gfp_mask=1052106, nid=0) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/gfp.h:524
#9  alloc_pages_node (order=0, gfp_mask=1052106, nid=0) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/gfp.h:538
#10 alloc_pages (order=0, gfp_mask=1052106) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/gfp.h:557
#11 alloc_zeroed_user_highpage_movable (vma=0xffff8881044a92c0, vaddr=<optimized out>) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/include/linux/highmem.h:197
#12 do_anonymous_page (vmf=0xffffc9000007fe38) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3535
#13 handle_pte_fault (vmf=0xffffc9000007fe38) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4385
#14 __handle_mm_fault (flags=597, address=140737354108928, vma=<optimized out>) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4522
#15 handle_mm_fault (vma=<optimized out>, address=address@entry=140737354108928, flags=flags@entry=597, regs=regs@entry=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4620
#16 0xffffffff81024d37 in do_user_addr_fault (regs=regs@entry=0xffffc9000007ff58, hw_error_code=hw_error_code@entry=6, address=address@entry=140737354108928)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/fault.c:1393
#17 0xffffffff81190a4b in handle_page_fault (address=140737354108928, error_code=6, regs=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/fault.c:1450
#18 exc_page_fault (regs=0xffffc9000007ff58, error_code=6) at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/fault.c:1506
#19 0xffffffff81200a6b in asm_exc_page_fault () at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include/asm/idtentry.h:580
#20 0x0000000000000000 in ?? ()
(gdb) p /x 140737354108928
$14 = 0x7ffff7ff9000
```

Demand Paging: page fault → Page table configuration - Anonymous page (MAP_PRIVATE)




```
(gdb) x /gx ((pte_t *) 0xffff88810448b000 + 505)
0xffff88810448bfc8:     0x0000000000000000
```

```
(gdb) x /gx ((pte_t *) 0xffff88810448b000 + 505)
0xffff88810448bfc8:     0x8000000240309067
```

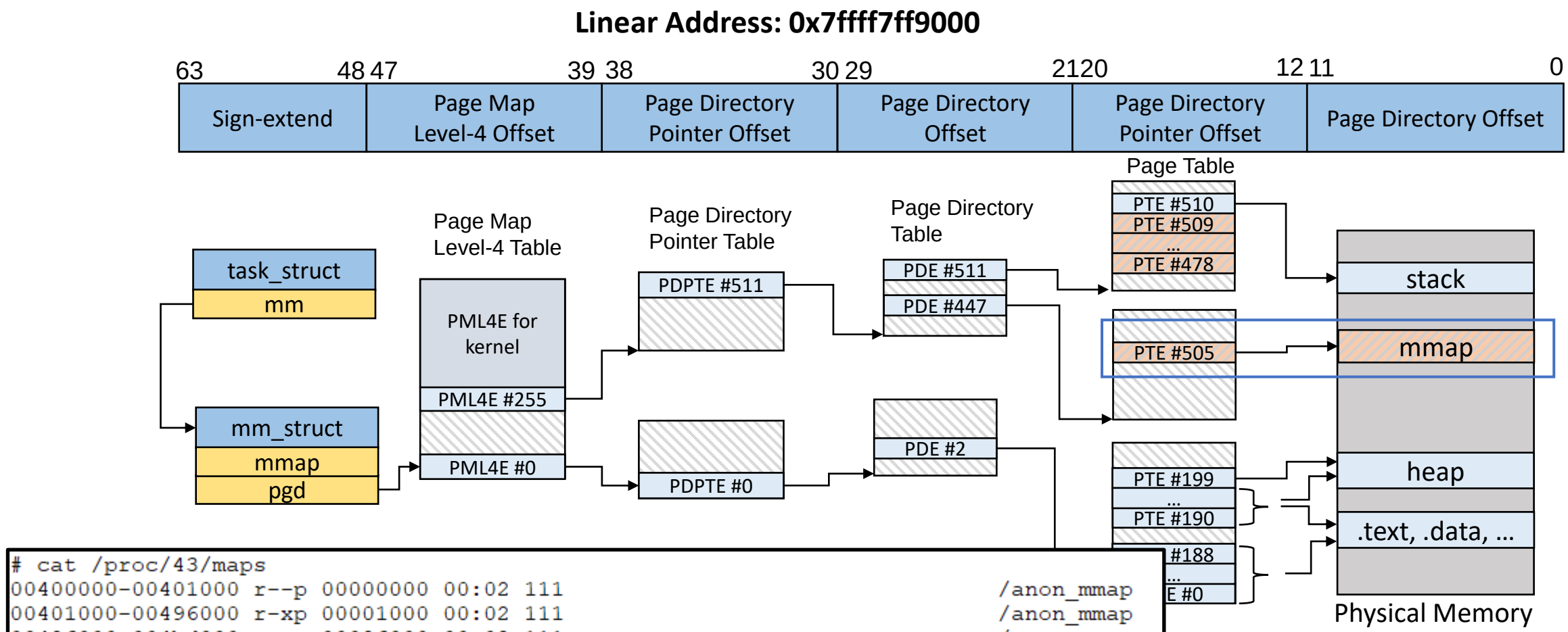
Demand Paging: page fault → call trace - Anonymous page (MAP_PRIVATE)

```
(gdb) bt
#0  set_pte_at (mm=0xffff8881000e4700, addr=140737354108928, pte=...,
    ptep=0xffff88810448bfc8)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include
/asm/pgtable.h:1038
#1  do_anonymous_page (vmf=0xffffc90000077e38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3577
#2  handle_pte_fault (vmf=0xffffc90000077e38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4385
#3  __handle_mm_fault (flags=597, address=140737354108928, vma=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4522
#4  handle_mm_fault (vma=<optimized out>,
    address=address@entry=140737354108928, flags=flags@entry=597,
    regs=regs@entry=0xffffc90000077f58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4620
#5  0xffffffff81024d37 in do_user_addr_fault (
    regs=regs@entry=0xffffc90000077f58, hw_error_code=hw_error_code@entry=6,
    address=address@entry=140737354108928)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1393
#6  0xffffffff81190a4b in handle_page_fault (address=140737354108928,
    error_code=6, regs=0xffffc90000077f58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1450
#7  exc_page_fault (regs=0xffffc90000077f58, error_code=6)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1506
#8  0xffffffff81200a6b in asm_exc_page_fault ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include
/asm/idtentry.h:580
#9  0x0000000000000000 in ?? ()
(gdb) p /x addr
$20 = 0x7ffff7ff9000
```

```
(gdb) x /gx ((pte_t *) 0xffff88810448b000 + 505)
0xffff88810448bfc8:     0x0000000000000000
```

```
(gdb) x /gx ((pte_t *) 0xffff88810448b000 + 505)
0xffff88810448bfc8:     0x8000000240309067
```

Demand Paging: page fault → VMAs – Anonymous page (MAP_PRIVATE)

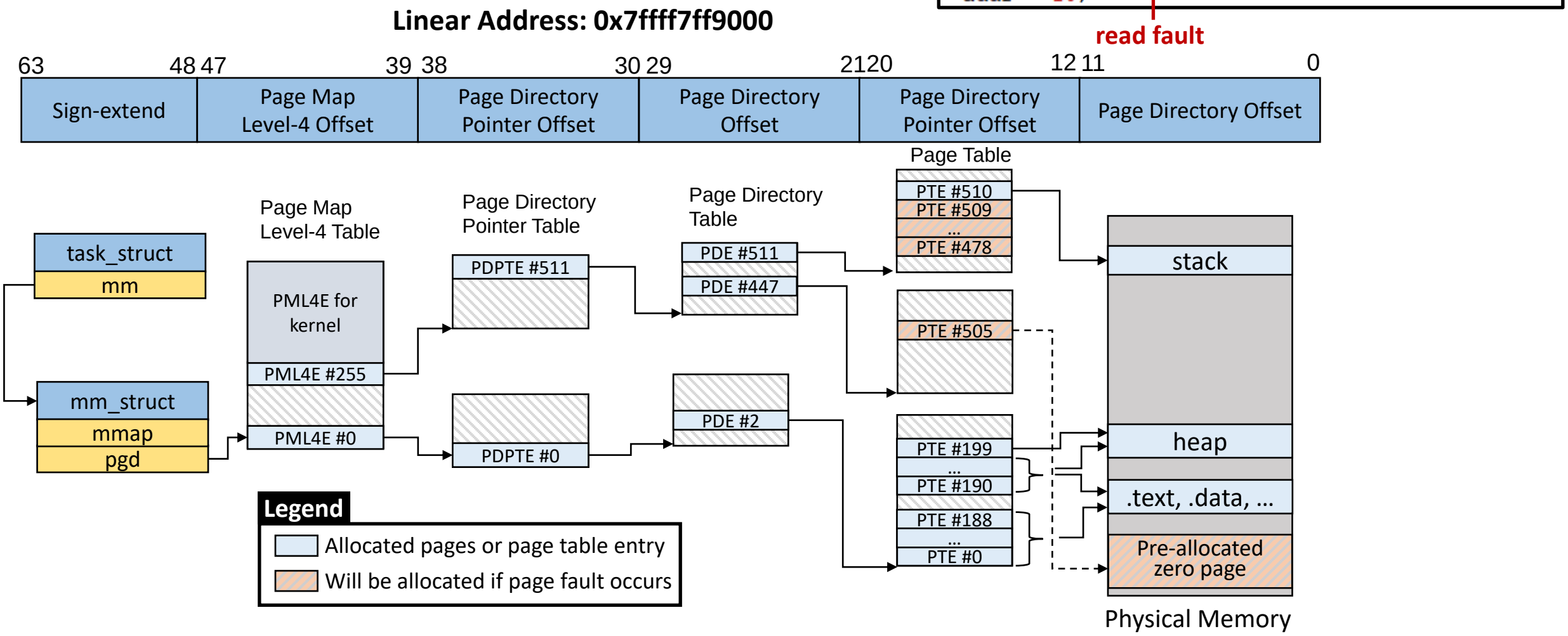



```
# cat /proc/43/maps
00400000-00401000 r--p 00000000 00:02 111                                /anon_mmap
00401000-00496000 r-xp 00001000 00:02 111                                /anon_mmap
00496000-004bd000 r--p 00096000 00:02 111                                /anon_mmap
004be000-004c1000 r--p 000bd000 00:02 111                                /anon_mmap
004c1000-004c4000 rw-p 000c0000 00:02 111                                /anon_mmap
004c4000-004e8000 rw-p 00000000 00:00 0                                  [heap]
7ffff7ff9000-7ffff7ffa000 rw-p 00000000 00:00 0
7ffff7ffa000-7ffff7ffe000 r--p 00000000 00:00 0                          [vvar]
7ffff7ffe000-7ffff7fff000 r-xp 00000000 00:00 0                          [vdso]
7ffffffde000-7ffffffff000 rw-p 00000000 00:00 0                          [stack]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
```

Demand Paging: page fault → VMAs – Anonymous page (MAP_PRIVATE) – Read before writing data (read fault)

```
addr = mmap(NULL, 4, PROT_READ | PROT_WRITE,
            MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
printf("data: %d\n", *addr);
*addr = 10;
```

read fault



1. **A pre-allocated zero page is initialized during system init -> Check init_zero_pfn()**
2. **Anonymous private page + read fault -> Link to the pre-allocated zero page**

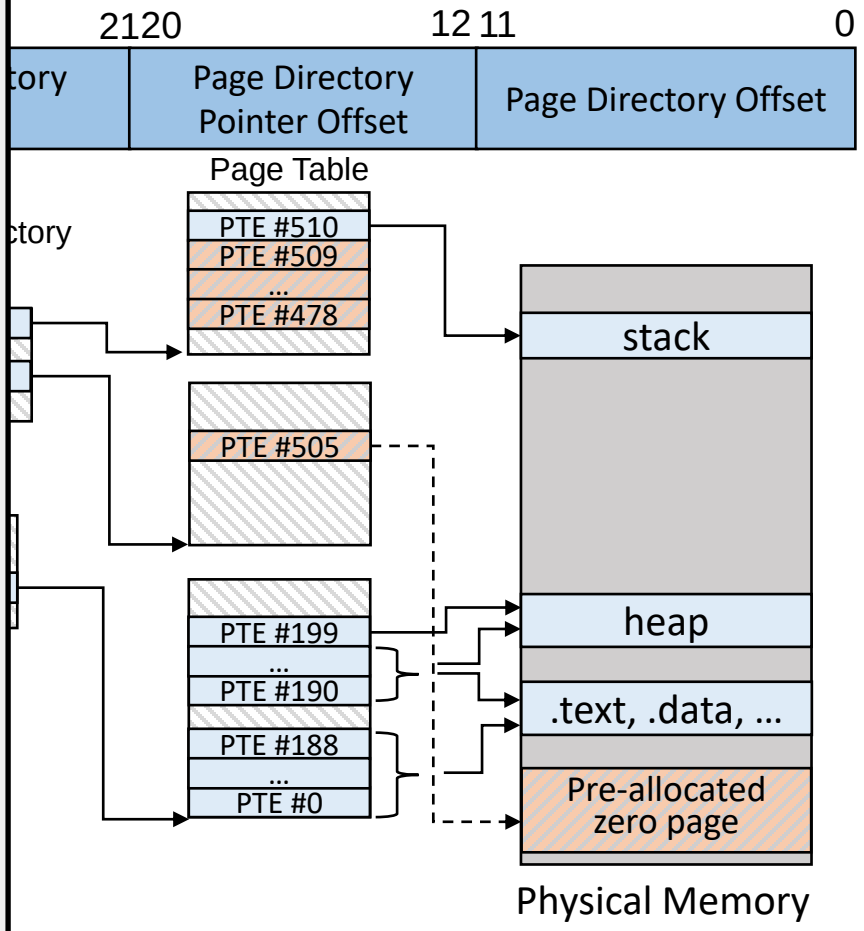
Demand Paging: page fault → VMAs – Anonymous page (MAP_PRIVATE) – Read before writing data (read fault)

```
(gdb) bt
#0  set_pte_at (mm=0xffff8881000e4700, addr=140737354108928, pte=...,
    ptep=0xffff8881049a9fc8)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include
/asm/pgtable.h:1038
#1  do_anonymous_page (vmf=0xffffc900000d7e38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3577
#2  handle_pte_fault (vmf=0xffffc900000d7e38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4385
#3  __handle_mm_fault (flags=596, address=140737354108928, vma=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4522
#4  handle_mm_fault (vma=<optimized out>,
    address=address@entry=140737354108928, flags=flags@entry=596,
    regs=regs@entry=0xffffc900000d7f58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4620
#5  0xffffffff81025107 in do_user_addr_fault (
    regs=regs@entry=0xffffc900000d7f58, hw_error_code=hw_error_code@entry=4,
    address=address@entry=140737354108928)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1393
#6  0xffffffff8122232b in handle_page_fault (address=140737354108928,
    error_code=4, regs=0xffffc900000d7f58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1450
#7  exc_page_fault (regs=0xffffc900000d7f58, error_code=4)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1506
#8  0xffffffff81400a6b in asm_exc_page_fault ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include
/asm/idtentry.h:580
#9  0x0000000000000000 in ?? ()
(gdb) p /x entry
$62 = {
  pte = 0x8000000001b44225
}
(gdb) p /x zero_pfn
$63 = 0x1b44
(gdb) f 1
#1  do_anonymous_page (vmf=0xffffc900000d7e38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3577
3577            set_pte_at(vma->vm_mm, vmf->address, vmf->pte, entry);
(gdb) p /x vmf->address
$64 = 0x7ffff7ff9000
```

Link to the pre-allocated zero page

```
addr = mmap(NULL, 4, PROT_READ | PROT_WRITE,
            MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
printf("data: %d\n", *addr);
*addr = 10;
```

read fault



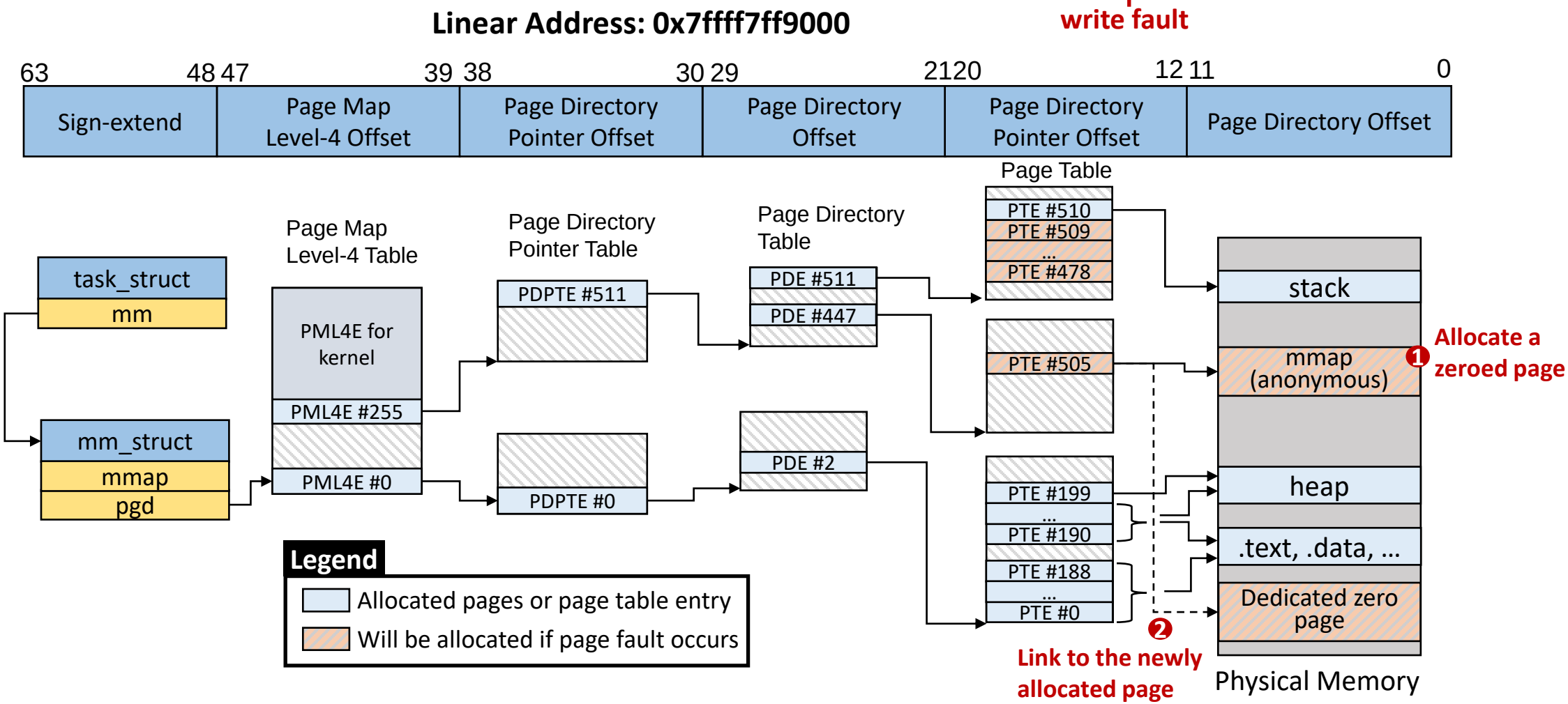
during system init -> Check init_zero_pfn()

2. Anonymous private page + read fault -> Link to the pre-allocated zero page

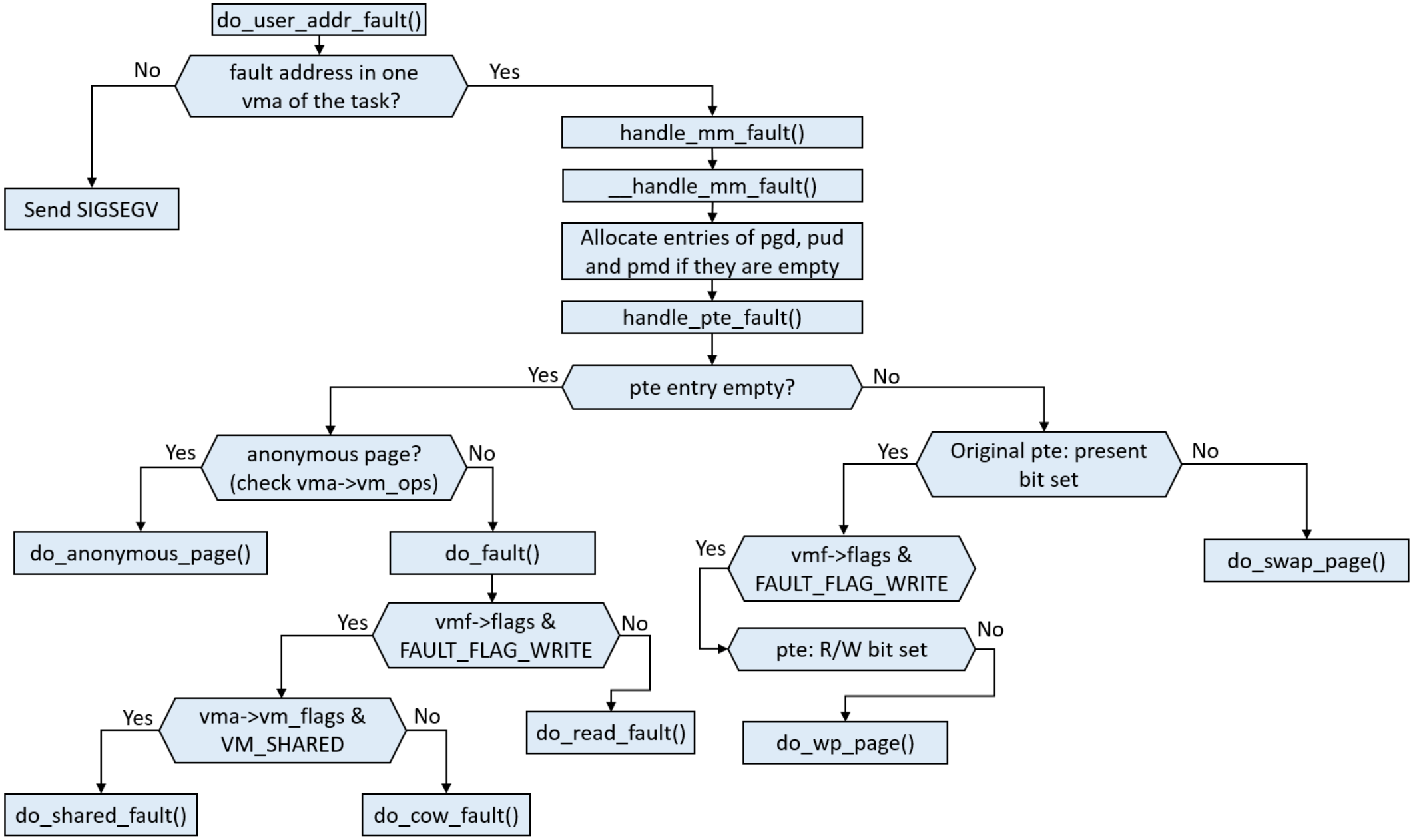
Demand Paging: page fault → VMAs – Anonymous page (MAP_PRIVATE) – Read before writing data (read fault) -> Write fault

```
addr = mmap(NULL, 4, PROT_READ | PROT_WRITE,
            MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
printf("data: %d\n", *addr);
*addr = 10;
```

write fault



Page fault handler for userspace address

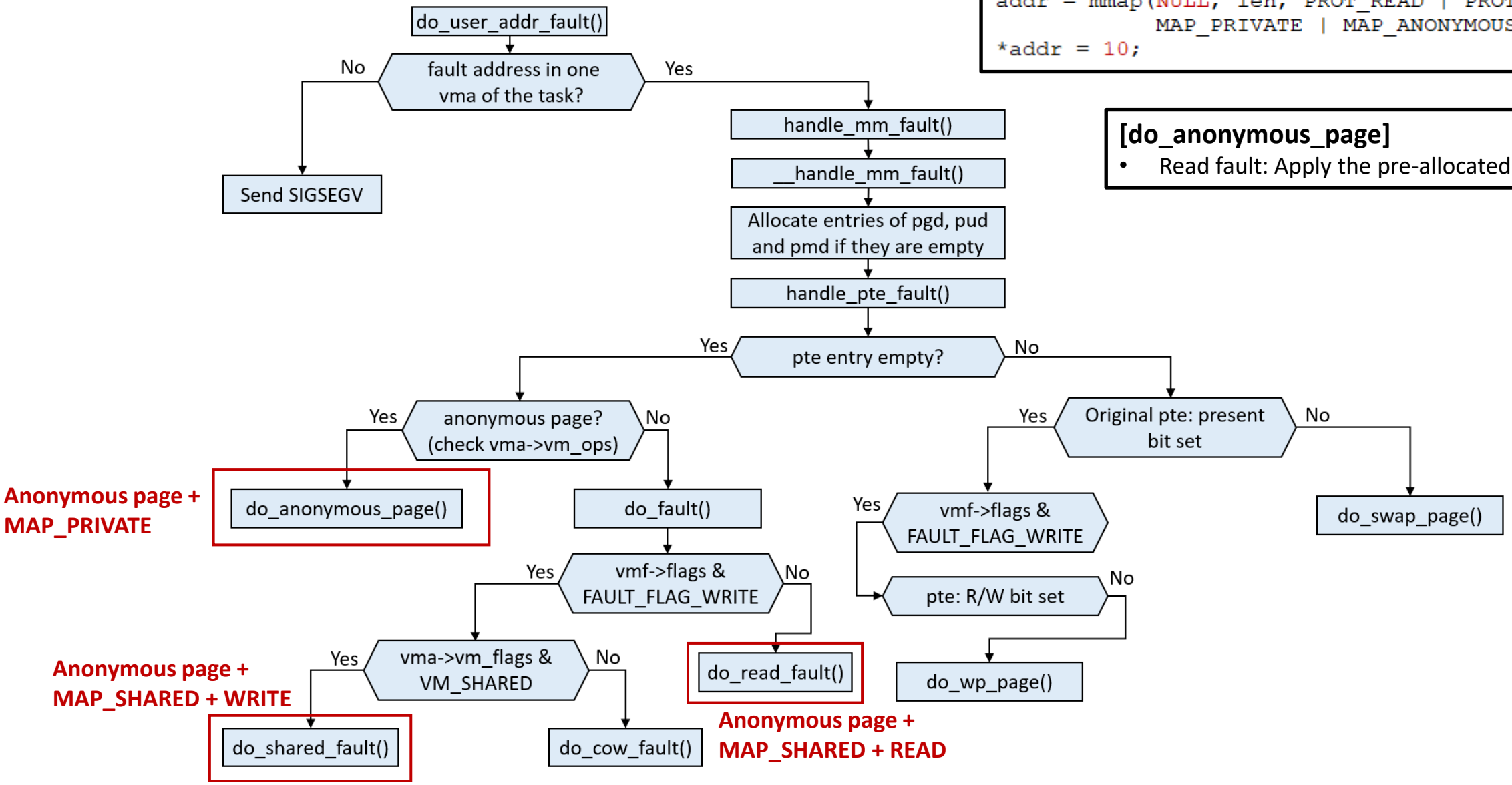


Demand Paging: page fault → VMAs – Anonymous page (MAP_PRIVATE & MAP_SHARED): first-time access

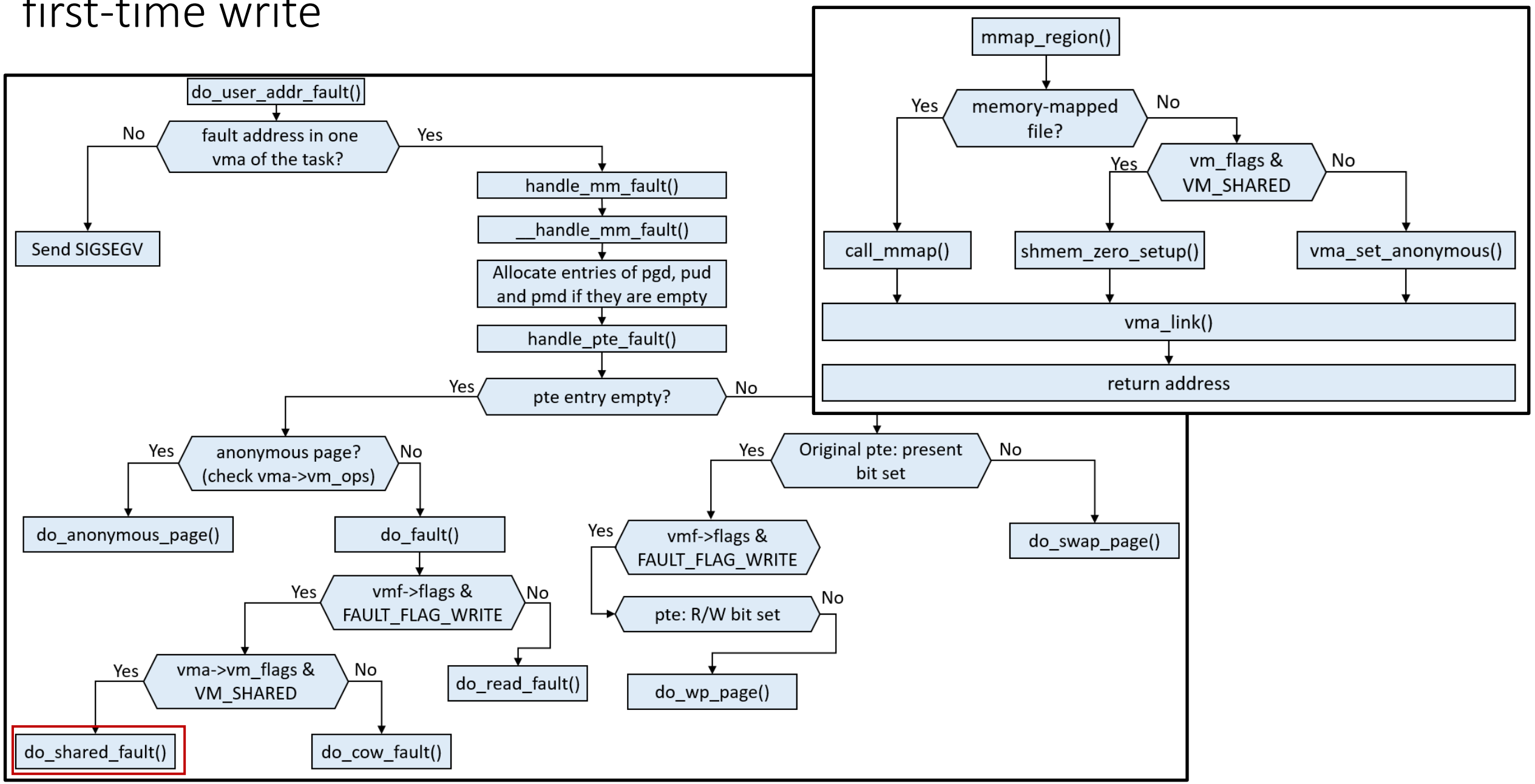
```
int *addr;

addr = mmap(NULL, len, PROT_READ | PROT_WRITE,
            MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
*addr = 10;
```

[do_anonymous_page]

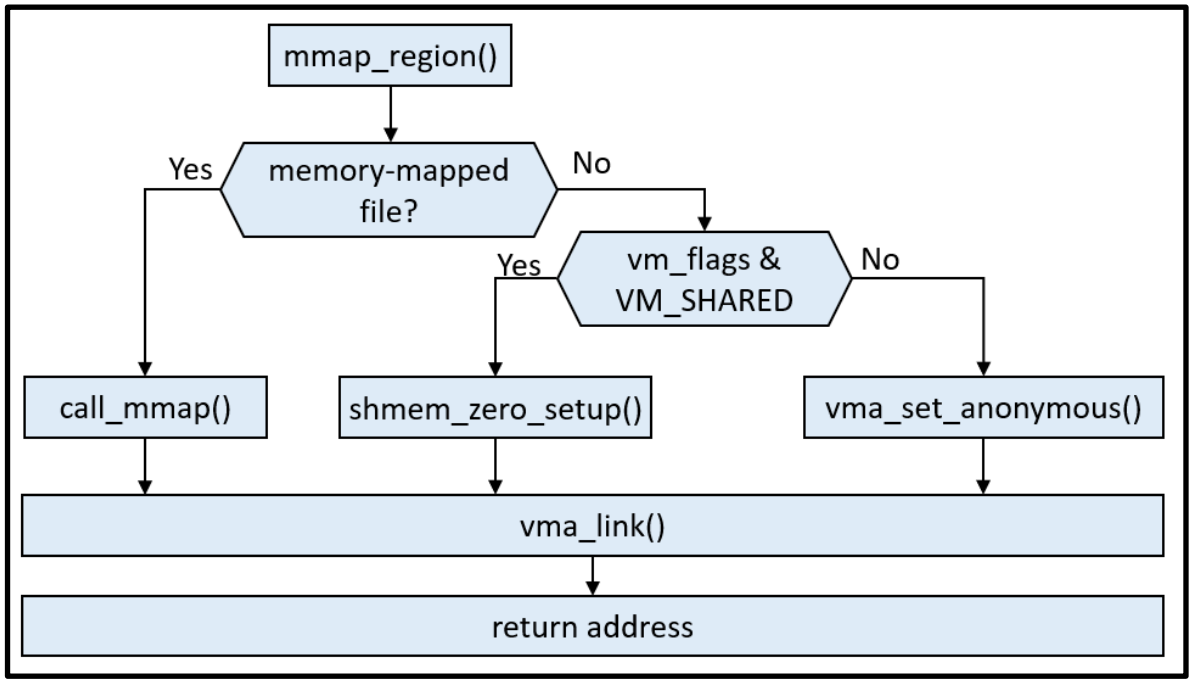
- Read fault: Apply the pre-allocated zero page



Demand Paging: page fault → VMAs – Anonymous page (MAP_SHARED): first-time write



Demand Paging: page fault → VMAs – Anonymous page (MAP_SHARED): first-time write




```
/**
 * shmem_zero_setup - setup a shared anonymous mapping
 * @vma: the vma to be mmapped is prepared by do_mmap
 */
int shmem_zero_setup(struct vm_area_struct *vma)
{
        struct file *file;
        loff_t size = vma->vm_end - vma->vm_start;

+--  6 lines: Cloning a new file under mmap_lock leads to a lock orderir
        file = shmem_kernel_file_setup("dev/zero", size, vma->vm_flags);
+---  2 lines: if (IS_ERR(file))-------------------------------------

        if (vma->vm_file)
                fput(vma->vm_file);
        vma->vm_file = file;
        vma->vm_ops = &shmem_vm_ops;

+--  6 lines: if (IS_ENABLED(CONFIG_TRANSPARENT_HUGEPAGE) &&----------
        return 0;
}
mm/shmem.c                                                    4255,1
```

Anonymous page + MAP_SHARED

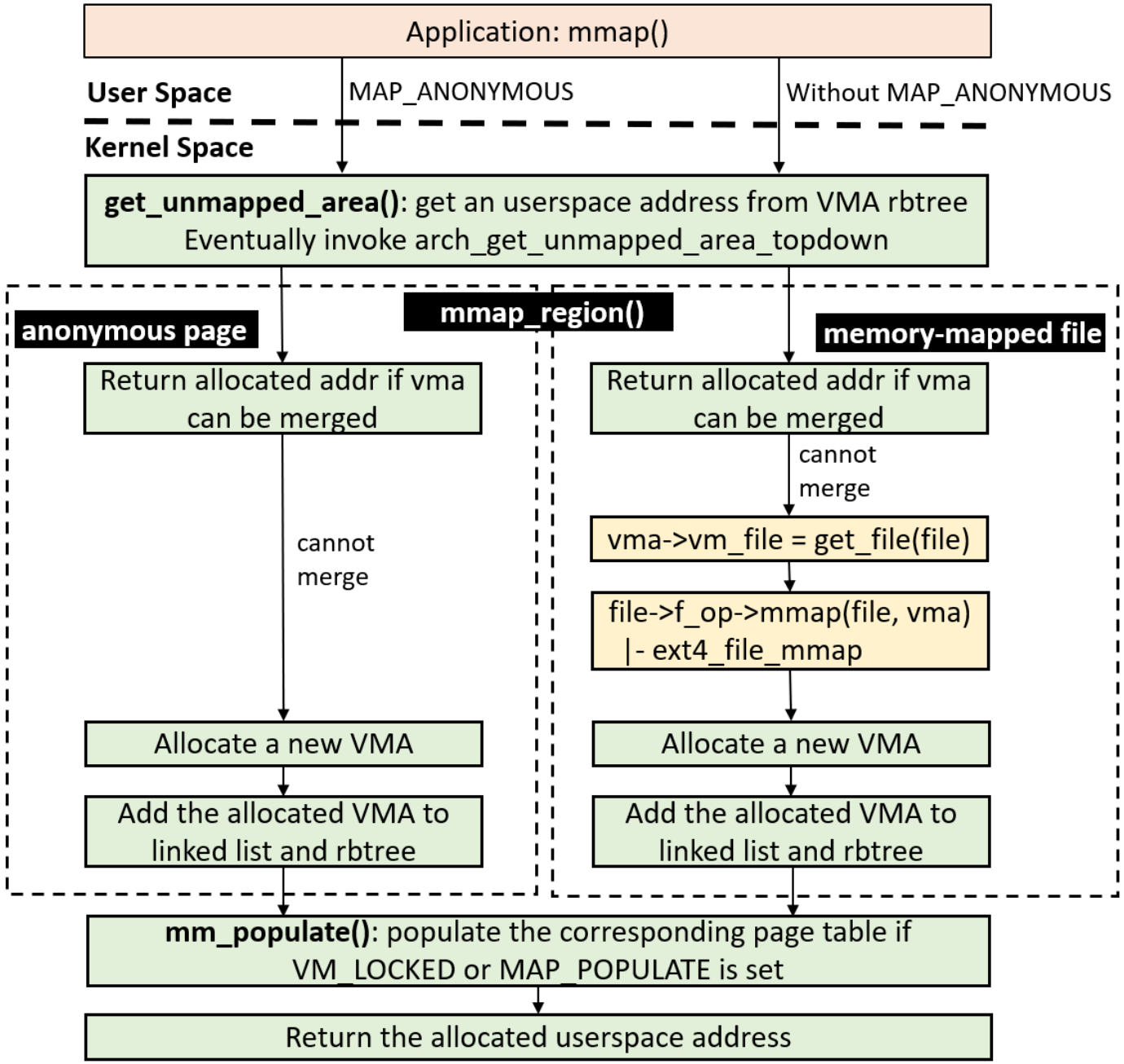
```
/ # cat /proc/43/maps
00400000-00401000 r--p 00000000 00:02 112                  /anon_mmap
00401000-00496000 r-xp 00001000 00:02 112                  /anon_mmap
00496000-004bd000 r--p 00096000 00:02 112                  /anon_mmap
004be000-004c1000 r--p 000bd000 00:02 112                  /anon_mmap
004c1000-004c4000 rw-p 000c0000 00:02 112                  /anon_mmap
004c4000-004e8000 rw-p 00000000 00:00 0                    [heap]
7ffff7ff9000-7ffff7ffa000 rw-s 00000000 00:01 909          /dev/zero (deleted)
7ffff7ffa000-7ffff7ffe000 r--p 00000000 00:00 0            [vvar]
7ffff7ffe000-7ffff7fff000 r-xp 00000000 00:00 0            [vdso]
7ffffffde000-7ffffffff000 rw-p 00000000 00:00 0            [stack]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0 [vsyscall]
```

Types of memory mappings: update

Visibility of Modification	Mapping Type		
	File	Anonymous	
		Description	Backing file
Private (Modification is not visible to other processes)	Initializing memory from contents of file Example: Process's .text and .data segements (Changes are not carried through to the underlying file)	Memory Allocation	No
Shared (Modification is visible to other processes)	1. Memory-mapped IO: Changes are carried through to the underlying file 2. Sharing memory between processes (IPC)	Sharing memory between processes (IPC)	/dev/zero

Reference from: Chapter 49, The Linux Programming Interface

Memory-mapped file

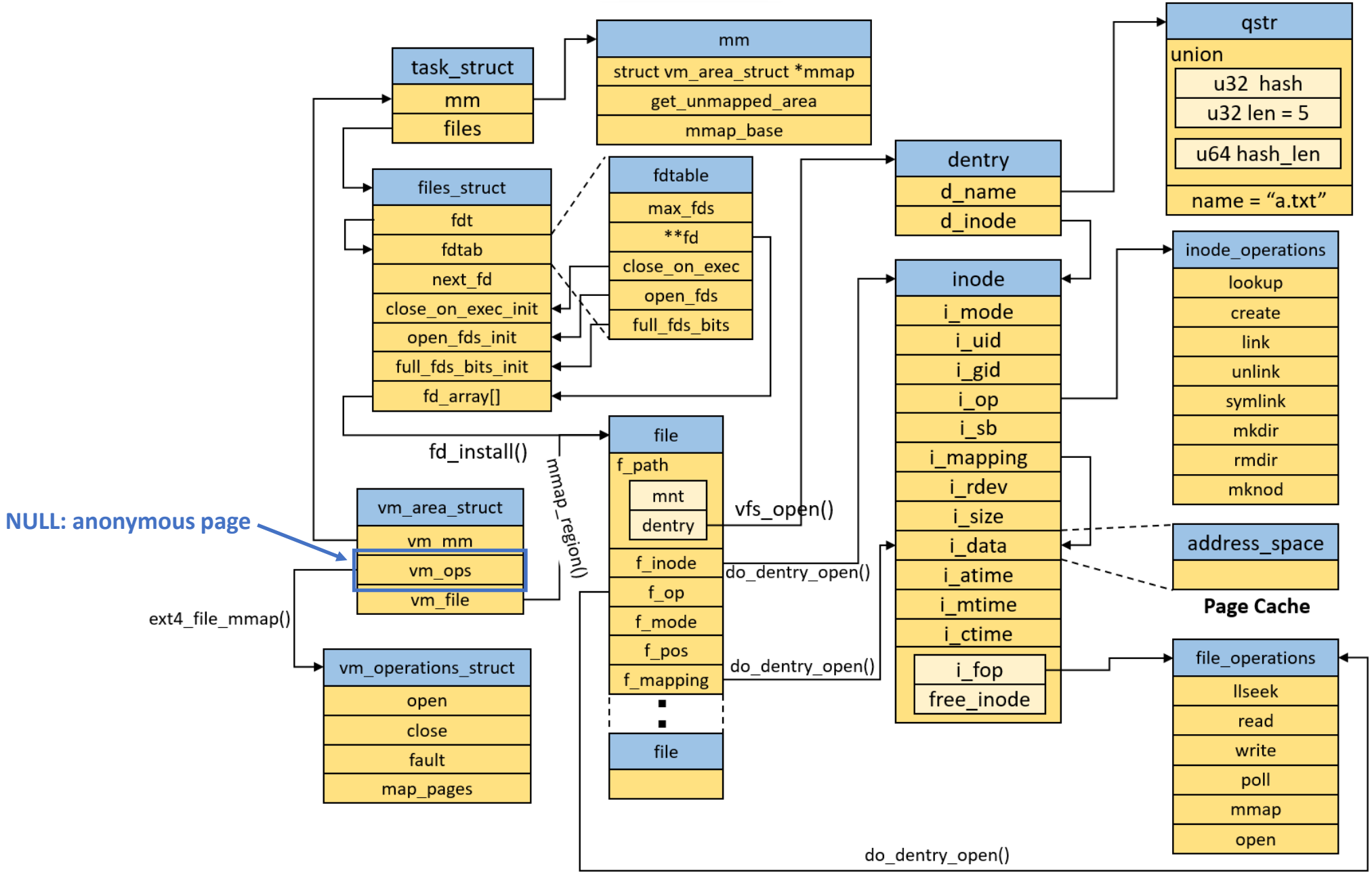



```
static const struct vm_operations_struct ext4_file_vm_ops = {
        .fault          = ext4_filemap_fault,
        .map_pages      = filemap_map_pages,
        .page_mkwrite   = ext4_page_mkwrite,
};

static int ext4_file_mmap(struct file *file, struct vm_area_struct *vma)
{
+-- 15 lines: struct inode *inode = file->f_mapping->host;----------
        if (IS_DAX(file_inode(file))) {
                vma->vm_ops = &ext4_dax_vm_ops;
                vma->vm_flags |= VM_HUGEPAGE;
        } else {
                vma->vm_ops = &ext4_file_vm_ops;
        }
        return 0;
}

static int ext4_sample_last_mounted(struct super_block *sb,
fs/ext4/file.c                                              738,1
```

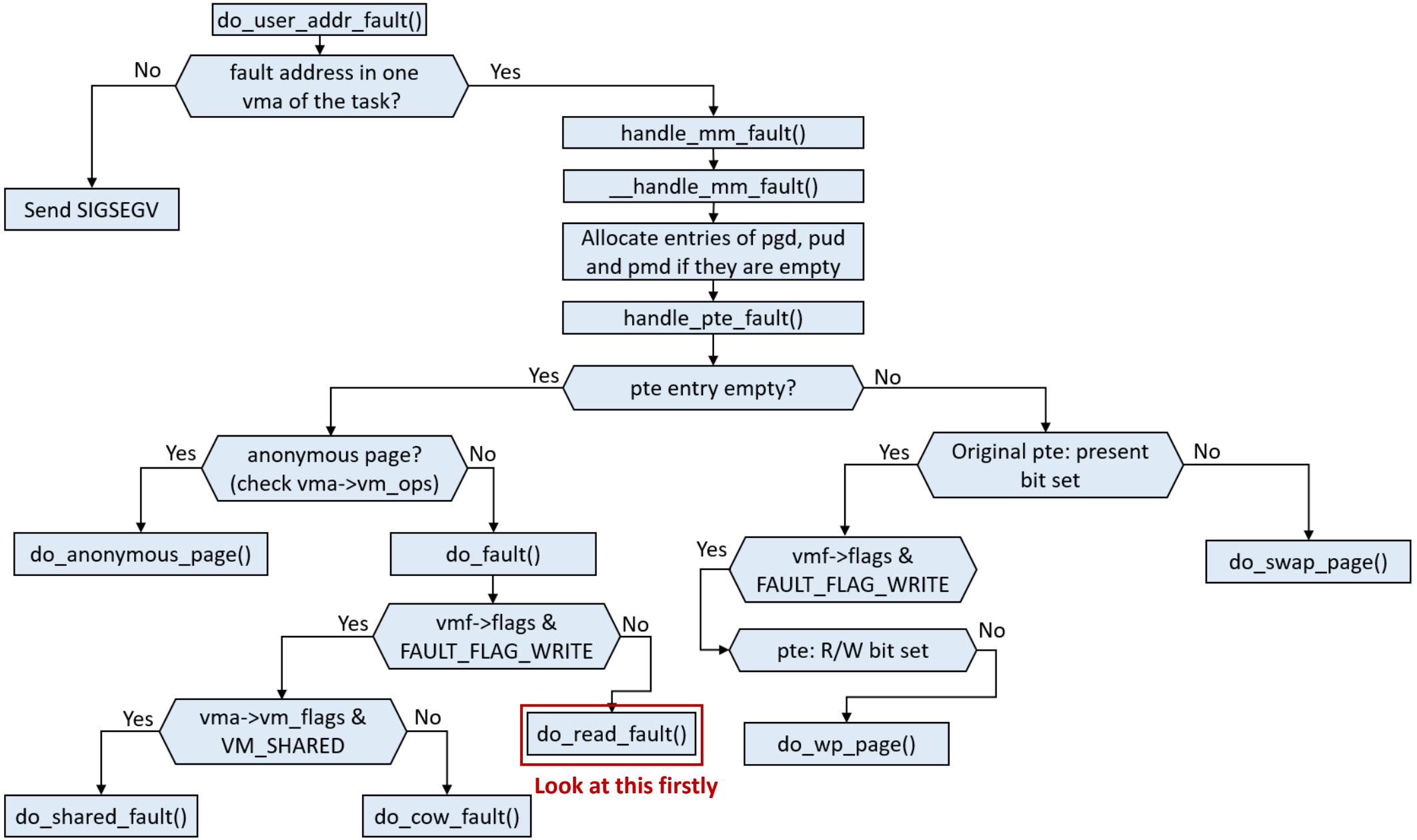
vma->vm_ops: invoke vm_ops callbacks in page fault handler

- struct vm_operations_struct
  - fault : Invoked by page fault handler to read the corresponding data into a physical page.
  - map_pages : Map the page if it is in page cache (check function ‘do_fault_around’: warm/cold page cache).
  - page_mkwrite: Notification that a previously read-only page is about to become writable.

Memory-mapped file: related data structures



Page fault handler for userspace address: Memory-mapped file



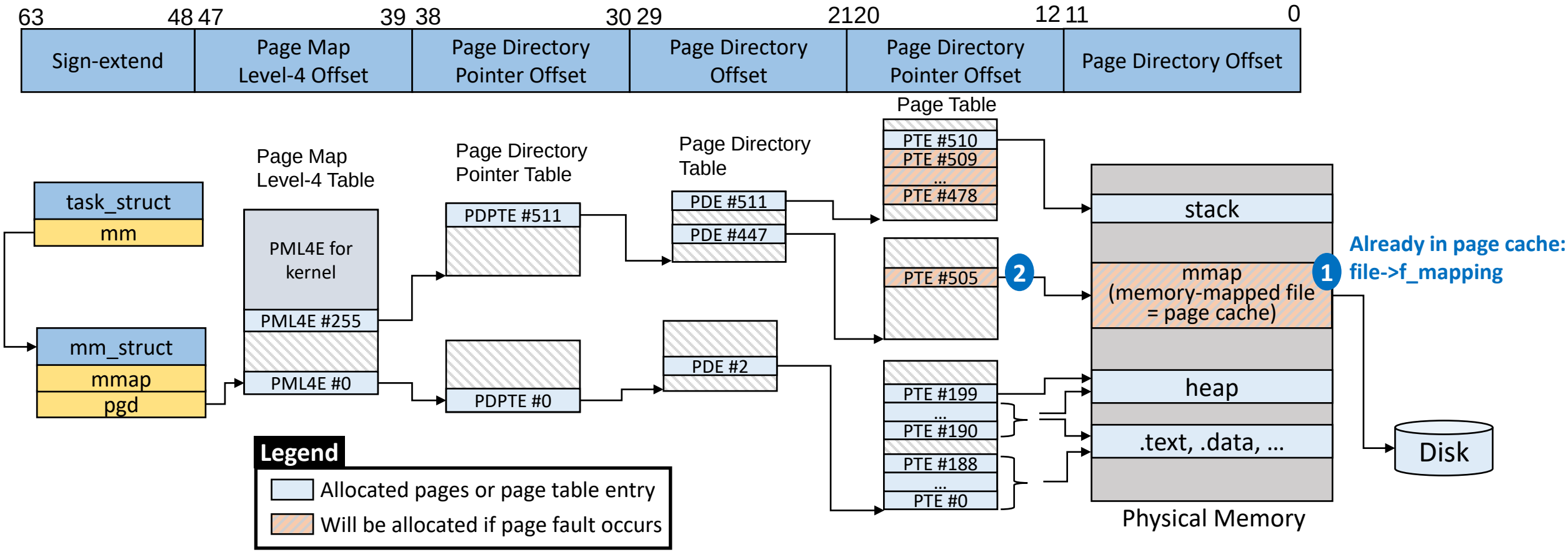
Demand Paging: page fault → do_read_fault(): warm page cache

Memory-mapped file

```
fd = open(argv[1], O_RDWR);
addr = mmap(NULL, 16, PROT_READ | PROT_WRITE,
            MAP_SHARED, fd, 0);

printf("Current string=%.*s\n", 16, addr);
strncpy(addr, "Lenovo", 6);
```

← read page fault

Linear Address: 0x7ffff7ff9000



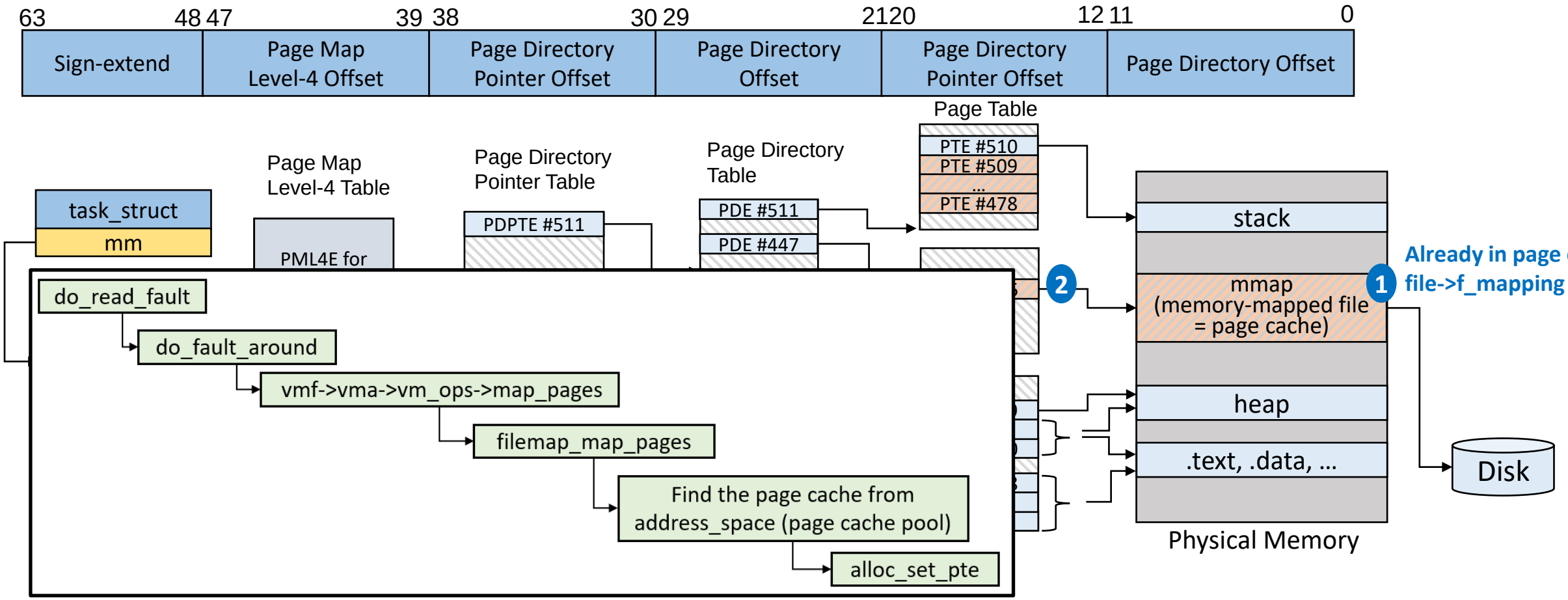
Demand Paging: page fault → do_read_fault(): warm page cache

Memory-mapped file

```
fd = open(argv[1], O_RDWR);
addr = mmap(NULL, 16, PROT_READ | PROT_WRITE,
            MAP_SHARED, fd, 0);

printf("Current string=%.*s\n", 16, addr);
strncpy(addr, "Lenovo", 6);
```

read page fault

Linear Address: 0x7ffff7ff9000

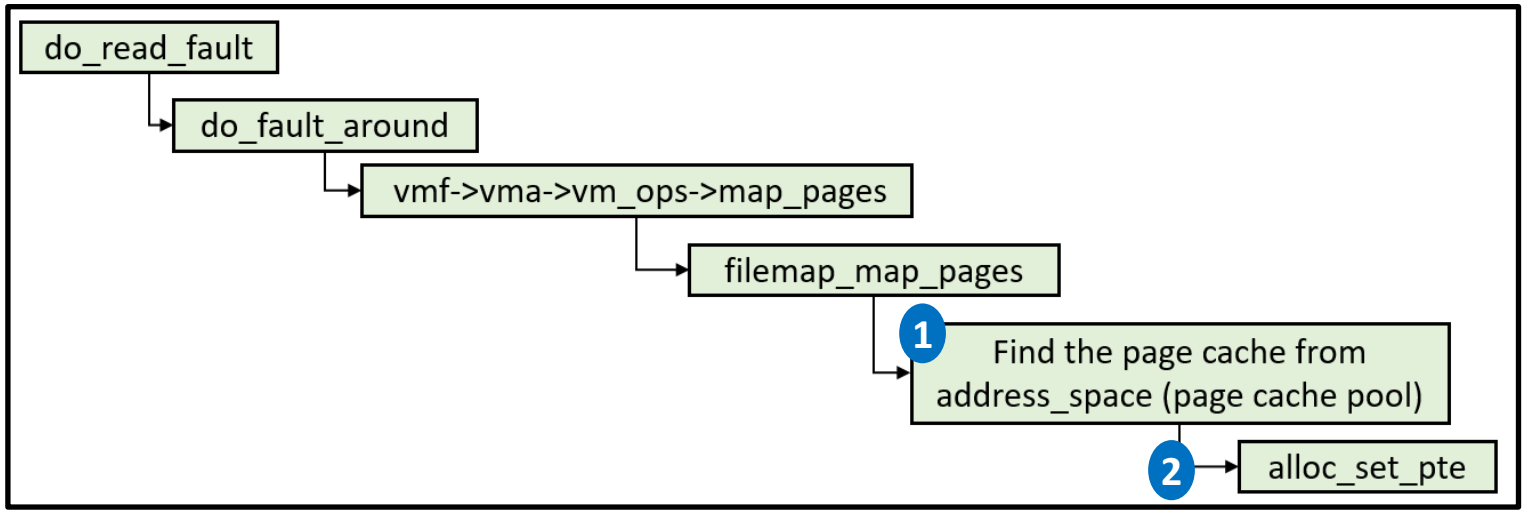


Demand Paging: page fault → filemap_map_pages(): warm page cache

Memory-mapped file

```
fd = open(argv[1], O_RDWR);
addr = mmap(NULL, 16, PROT_READ | PROT_WRITE,
            MAP_SHARED, fd, 0);

printf("Current string=%.*s\n", 16, addr);
strncpy(addr, "Lenovo", 6);
```

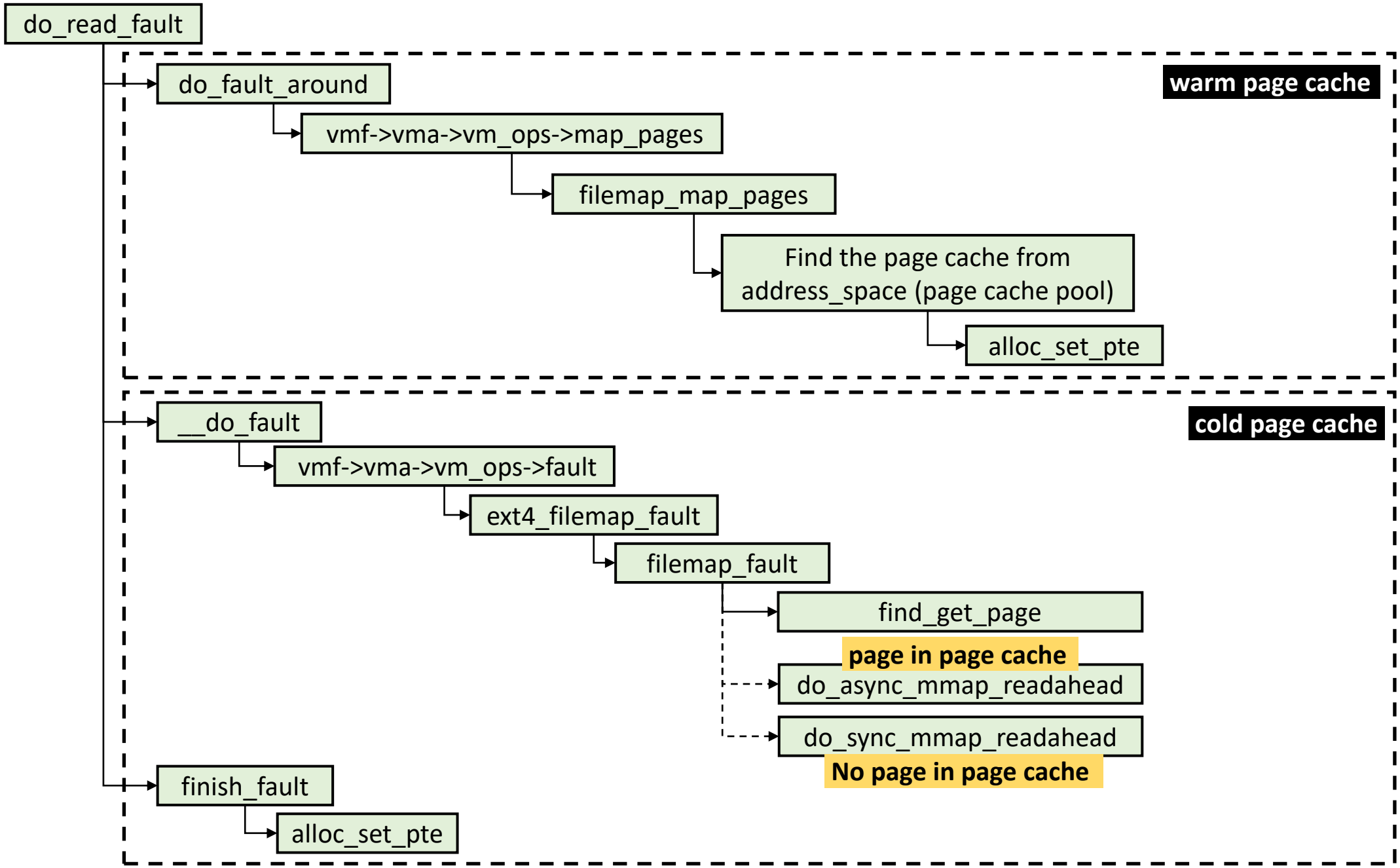
← read page fault



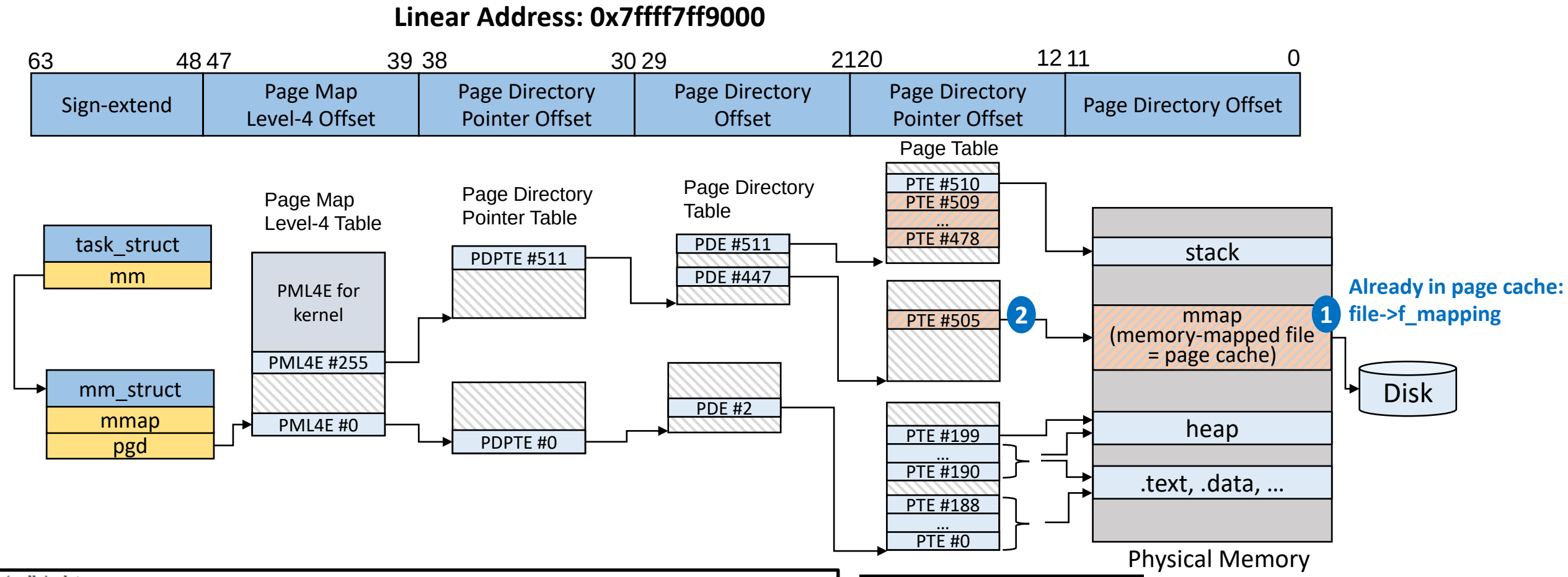

```
static const struct vm_operations_struct ext4_file_vm_ops = {
        .fault          = ext4_filemap_fault,
        .map_pages      = filemap_map_pages,
        .page_mkwrite   = ext4_page_mkwrite,
};

fs/ext4/file.c                                          723,0-1
```

map_pages callback: map a page cache (already available) to process address space (process page table) → warm page cache

Demand Paging: page fault → call path for warm/cold page cache



warm page cache: make sure “page cache = mmap physical page” (1/3)




```
(gdb) bt
#0  alloc_set_pte (vmf=vmf@entry=0xffffc9000007fe38,
    page=page@entry=0xffffea000900c400)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3802
#1  0xffffffff81079432 in filemap_map_pages (vmf=0xffffc9000007fe38,
    start_pgoff=<optimized out>, end_pgoff=0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/filemap.c:297
1
#2  0xffffffff81093d80 in do_fault_around (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3980
#3  do_read_fault (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4014
```

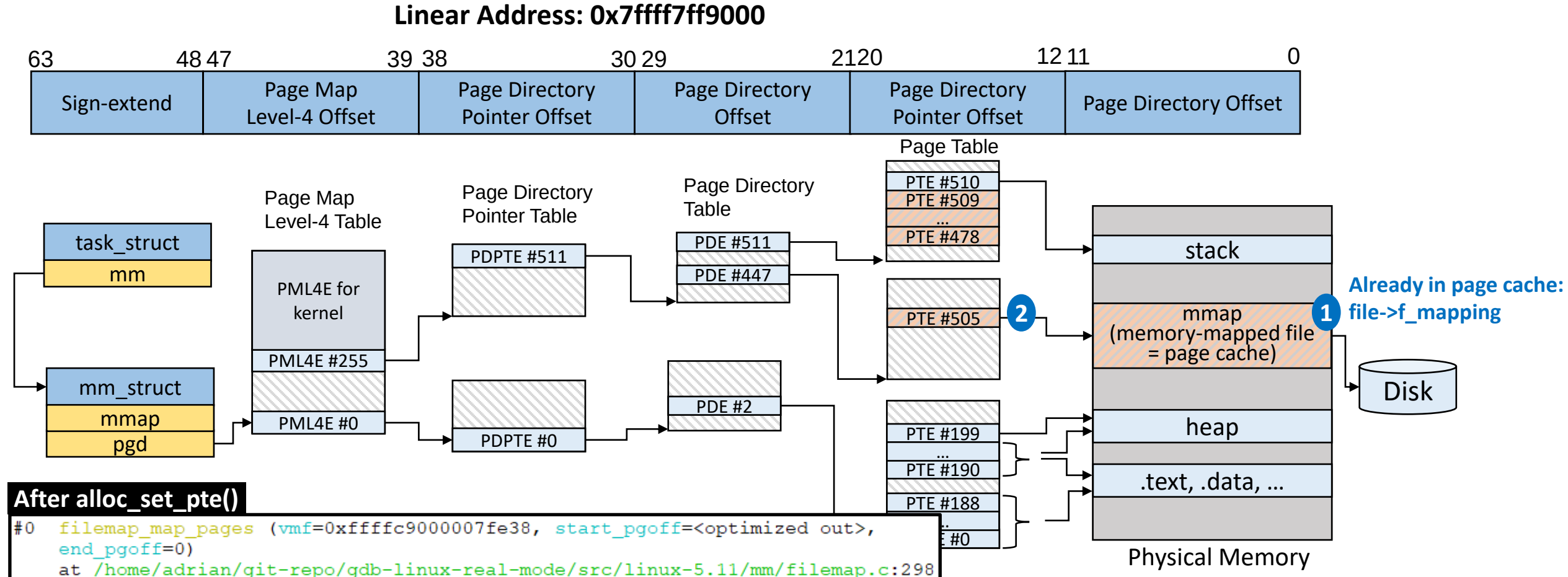
Page Cache - Verification

```
(gdb) p page->mapping
$9 = (struct address_space *) 0xffff8881004f0648
(gdb) print (int) PG_mappedtodisk
$10 = 17
(gdb) p /x page->flags
$11 = 0x4000000000002203d
```

page->mapping

- **[bit 0] = 1: anonymous page → mapping field = anon_vma descriptor**
- **[bit 0] = 0: page cache → mapping field = address_space descriptor**

warm page cache: make sure “page cache = mmap physical page” (2/3)



After alloc_set_pte()

```
#0  filemap_map_pages (vmf=0xffffc9000007fe38, start_pgoff=<optimized out>,
    end_pgoff=0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/filemap.c:298
5
#1  0xffffffff81093d80 in do_fault_around (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3980
#2  do_read_fault (vmf=0
    at /home/adrian/git-                                    l1/mm/memory.c:4014
#3  do_fault (vmf=vmf@en
    at /home/adrian/git-                                    l1/mm/memory.c:4147
```

After alloc_set_pte()

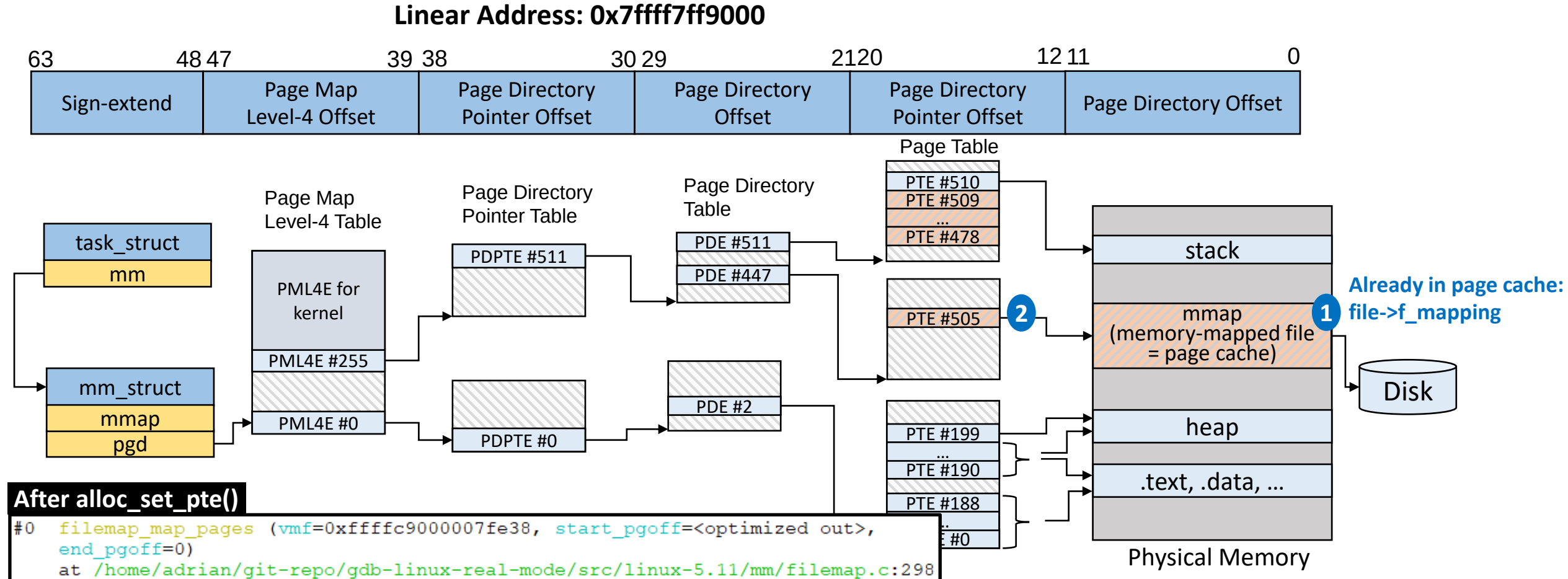
```
(gdb) p vmf->pte
$46 = (pte_t *) 0xffff888104706fc8
(gdb) p /x *vmf->pte
$47 = {
  pte = 0x8000000240310025
}
```

<< PAGE_SHIFT

```
(gdb) info macro __page_to_pfn
...
#define __page_to_pfn(page) (unsigned long)((page) - vmemmap)
```

```
(gdb) p /x page
$38 = 0xffffea000900c400
(gdb) p vmemmap
$39 = (struct page *) 0xffffea0000000000
(gdb) p /x page - vmemmap
$40 = 0x240310
```

warm page cache: make sure “page cache = mmap physical page” (3/3)



After alloc_set_pte()

```
#0  filemap_map_pages (vmf=0xffffc9000007fe38, start_pgoff=<optimized out>,
    end_pgoff=0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/filemap.c:298
5
#1  0xffffffff81093d80 in do_fault_around (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/
#  =0xffffc9000007fe38)
#  t-repo/gdb-linux-real-mode/src/linux-5.11/mm/
#  0007fe38)
```

After alloc_set_pte()

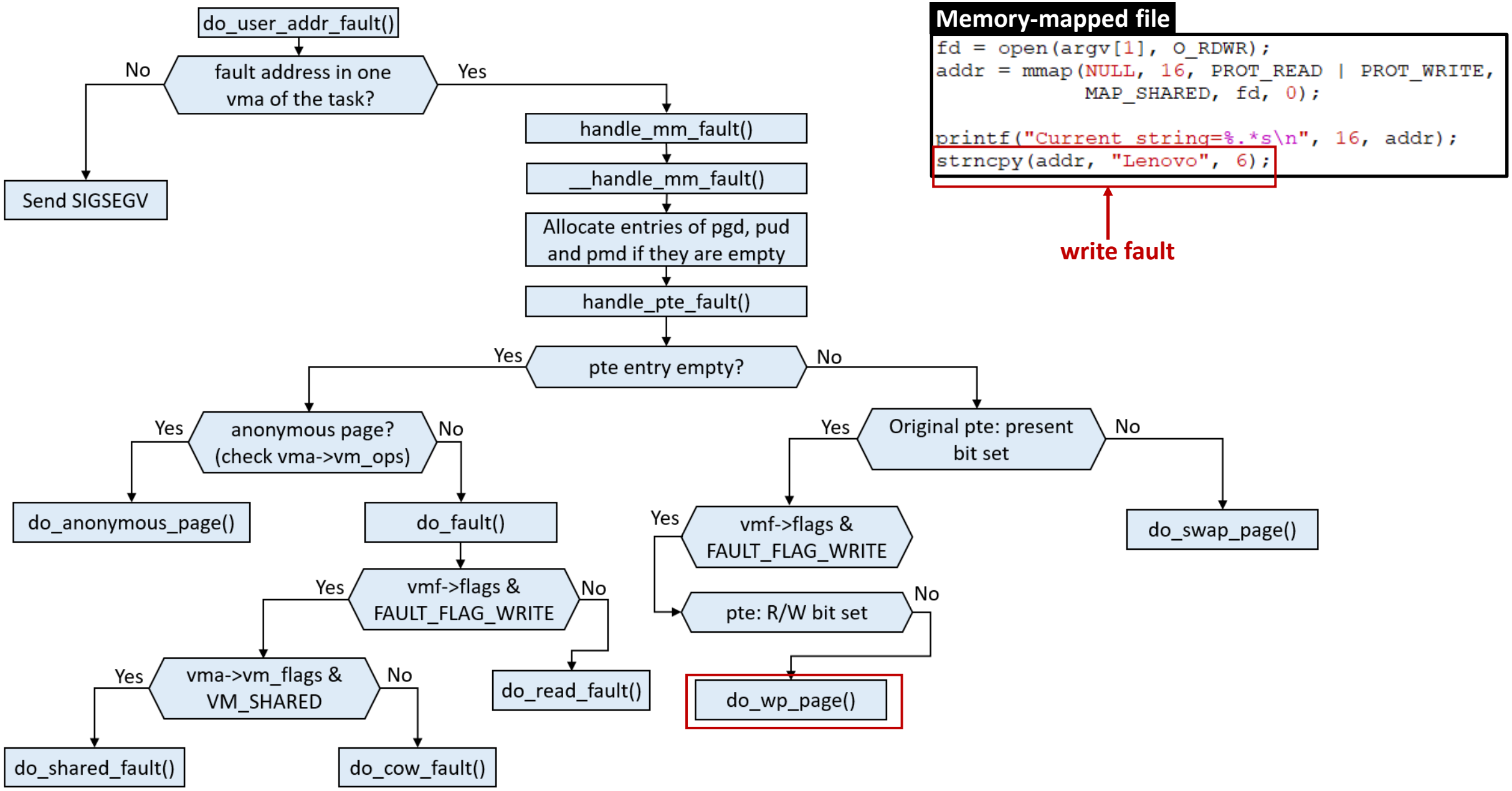
```
(gdb) p vmf->pte
$46 = (pte_t *) 0xffff888104706fc8
(gdb) p /x *vmf->pte
$47 = {
  pte = 0x8000000240310025
}
```

will generate a write fault for next write

Table 4-20. Format of a Page-Table Entry that Maps a 4-KByte Page

Bit Position(s)	Contents
0 (P)	Present; must be 1 to map a 4-KByte page
1 (R/W)	Read/write; if 0, writes may not be allowed to the 4-KByte page referenced by this entry (see Section 4.6)
2 (U/S)	User/supervisor; if 0, user-mode accesses are not allowed to the 4-KByte page referenced by this entry (see Section 4.6)
3 (PWT)	Page-level write-through; indirectly determines the memory type used to access the 4-KByte page referenced by this entry (see Section 4.9.2)

write fault (write-protected fault)



Memory-mapped file

```
fd = open(argv[1], O_RDWR);
addr = mmap(NULL, 16, PROT_READ | PROT_WRITE,
            MAP_SHARED, fd, 0);

printf("Current string=%.*s\n", 16, addr);
strncpy(addr, "Lenovo", 6);
```

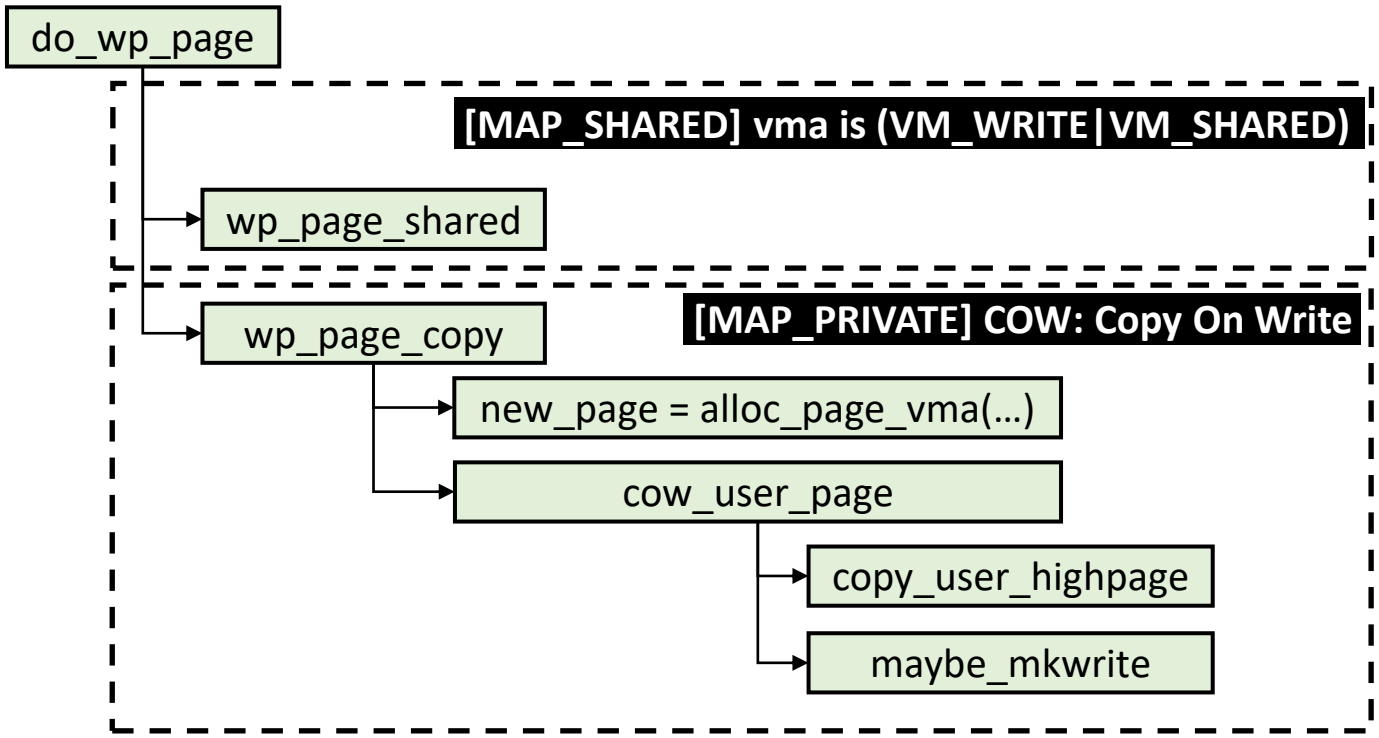
write fault

write fault (write-protected fault): do_wp_page

Memory-mapped file

```
fd = open(argv[1], O_RDWR);
addr = mmap(NULL, 16, PROT_READ | PROT_WRITE,
            MAP_SHARED, fd, 0);

printf("Current string=%.*s\n", 16, addr);
strncpy(addr, "Lenovo", 6);
```

write fault



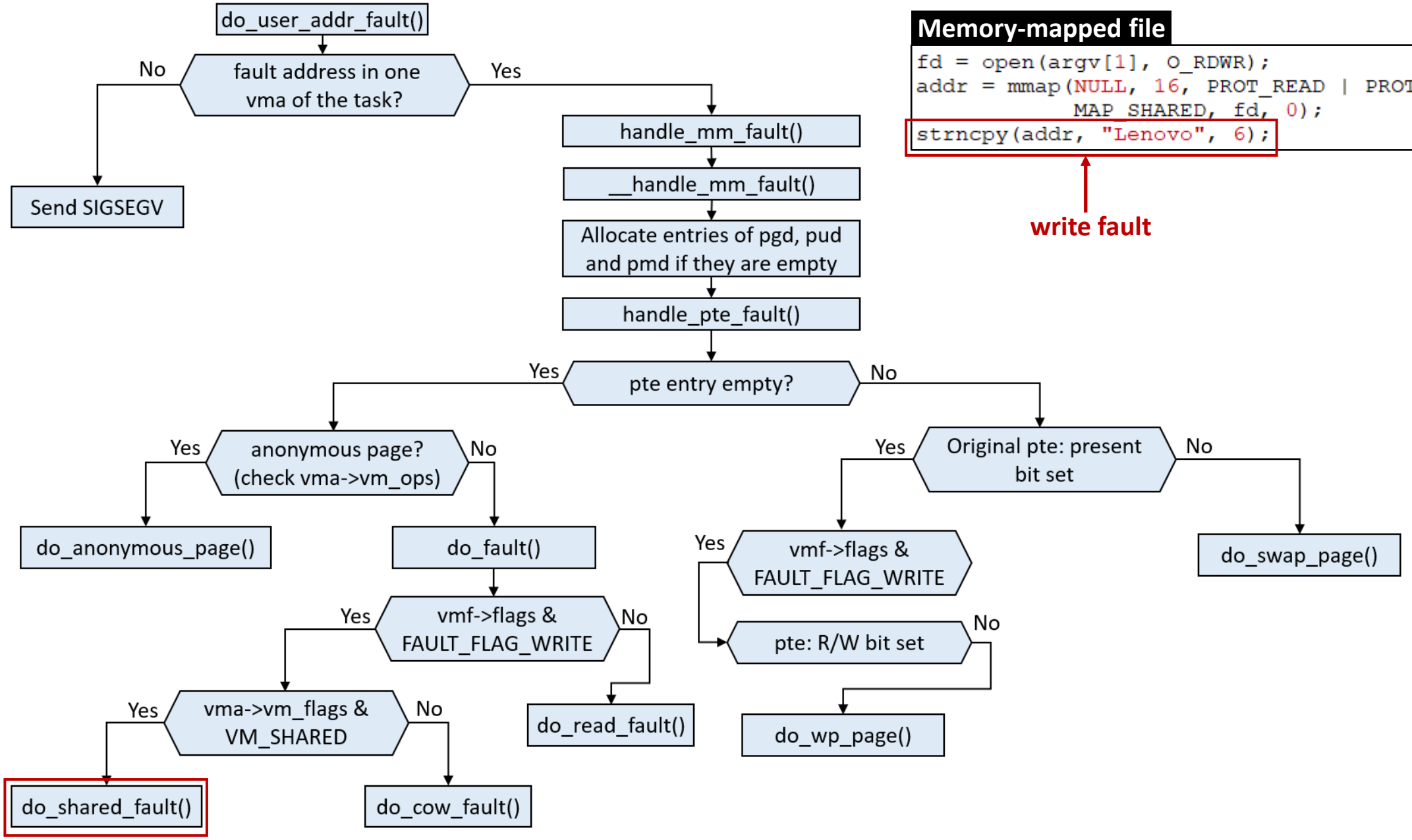
[MAP_PRIVATE] COW: Quote from `man mmap`

```
MAP_PRIVATE
       Create a private copy-on-write mapping.  Updates to the  mapping
       are  not  visible  to other processes mapping the same file, and
       are not carried through to the underlying file.  It is  unspeci-
       fied  whether changes made to the file after the mmap() call are
       visible in the mapped region.
```

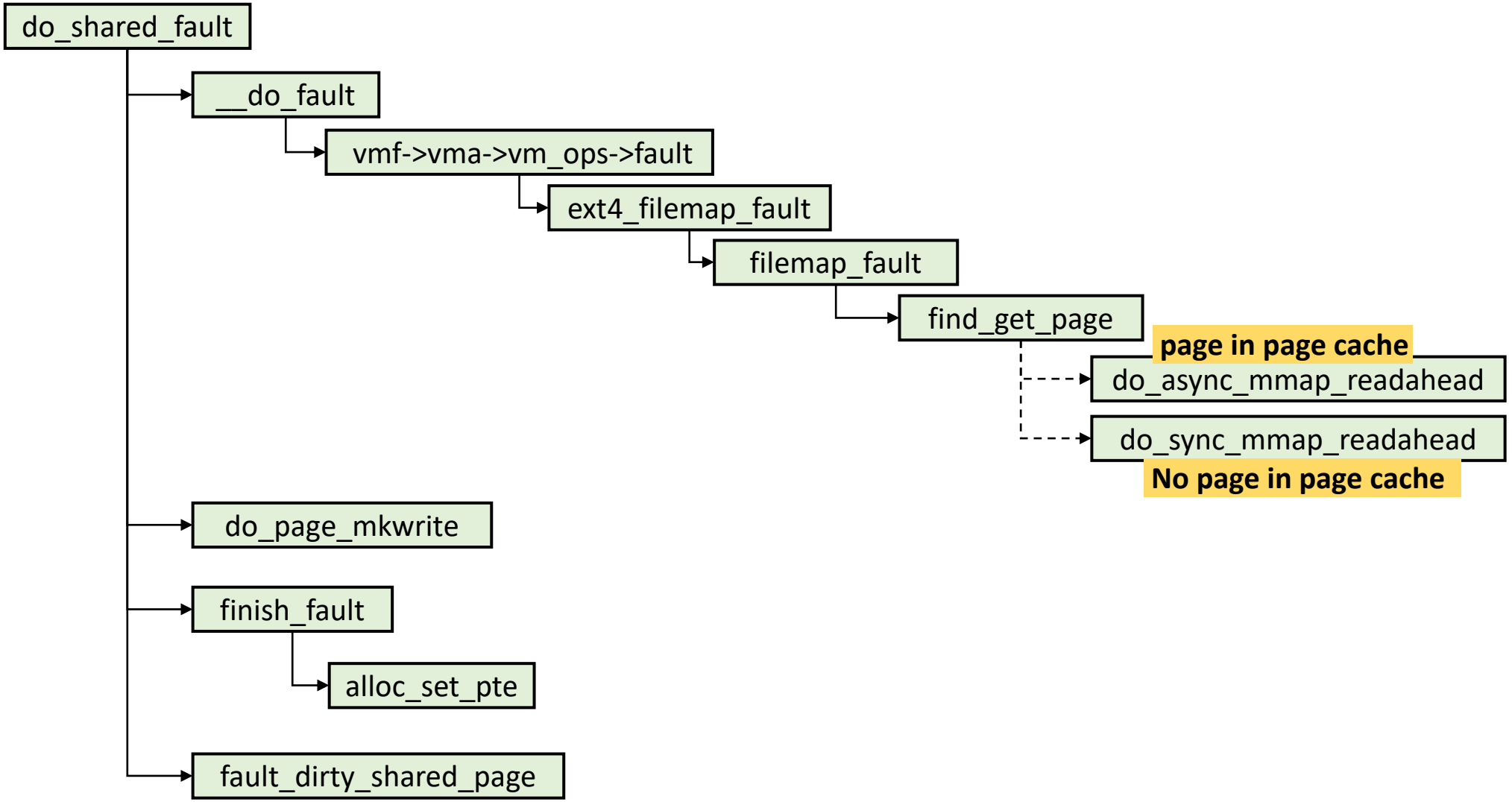
write fault (write-protected fault) – call path

```
(gdb) bt
#0  ext4_page_mkwrite (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/fs/ext4/inode.c:
6058
#1  0xffffffff81090e90 in do_page_mkwrite (vmf=vmf@entry=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:2714
#2  0xffffffff810931db in wp_page_shared (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3045
#3  do_wp_page (vmf=vmf@entry=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3138
#4  0xffffffff81094783 in handle_pte_fault (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4405
#5  __handle_mm_fault (flags=597, address=140737354108928, vma=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4522
#6  handle_mm_fault (vma=<optimized out>,
    address=address@entry=140737354108928, flags=flags@entry=597,
    regs=regs@entry=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4620
#7  0xffffffff81025107 in do_user_addr_fault (
    regs=regs@entry=0xffffc9000007ff58, hw_error_code=hw_error_code@entry=7,
    address=address@entry=140737354108928)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1393
#8  0xffffffff8122232b in handle_page_fault (address=140737354108928,
    error_code=7, regs=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1450
#9  exc_page_fault (regs=0xffffc9000007ff58, error_code=7)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1506
#10 0xffffffff81400a6b in asm_exc_page_fault ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include
/asm/idtentry.h:580
#11 0x0000000000000000 in ?? ()
(gdb) p /x 140737354108928
$3 = 0x7ffff7ff9000
```

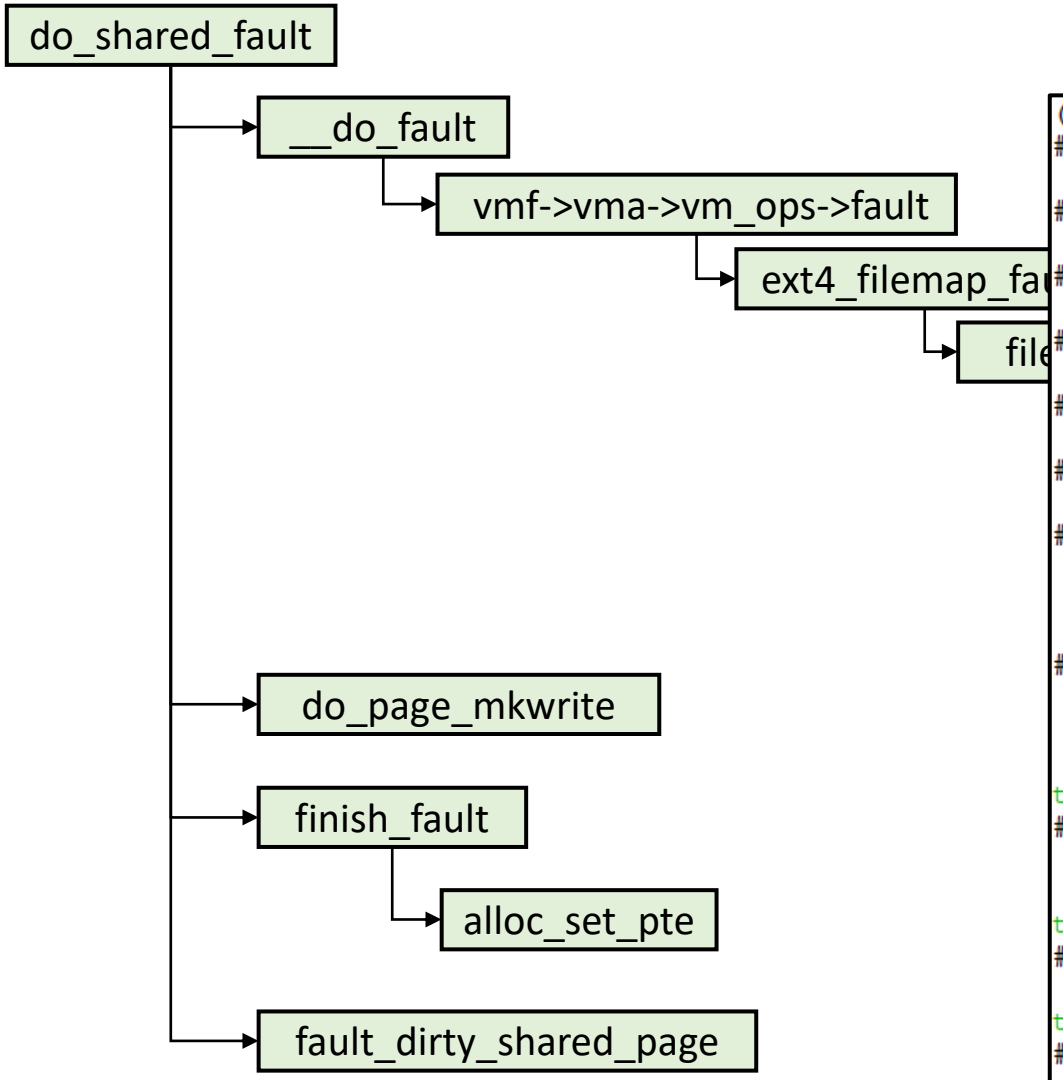
write fault without previously reading (MAP_SHARED): do_shared_fault



write fault without previously reading (MAP_SHARED): do_shared_fault

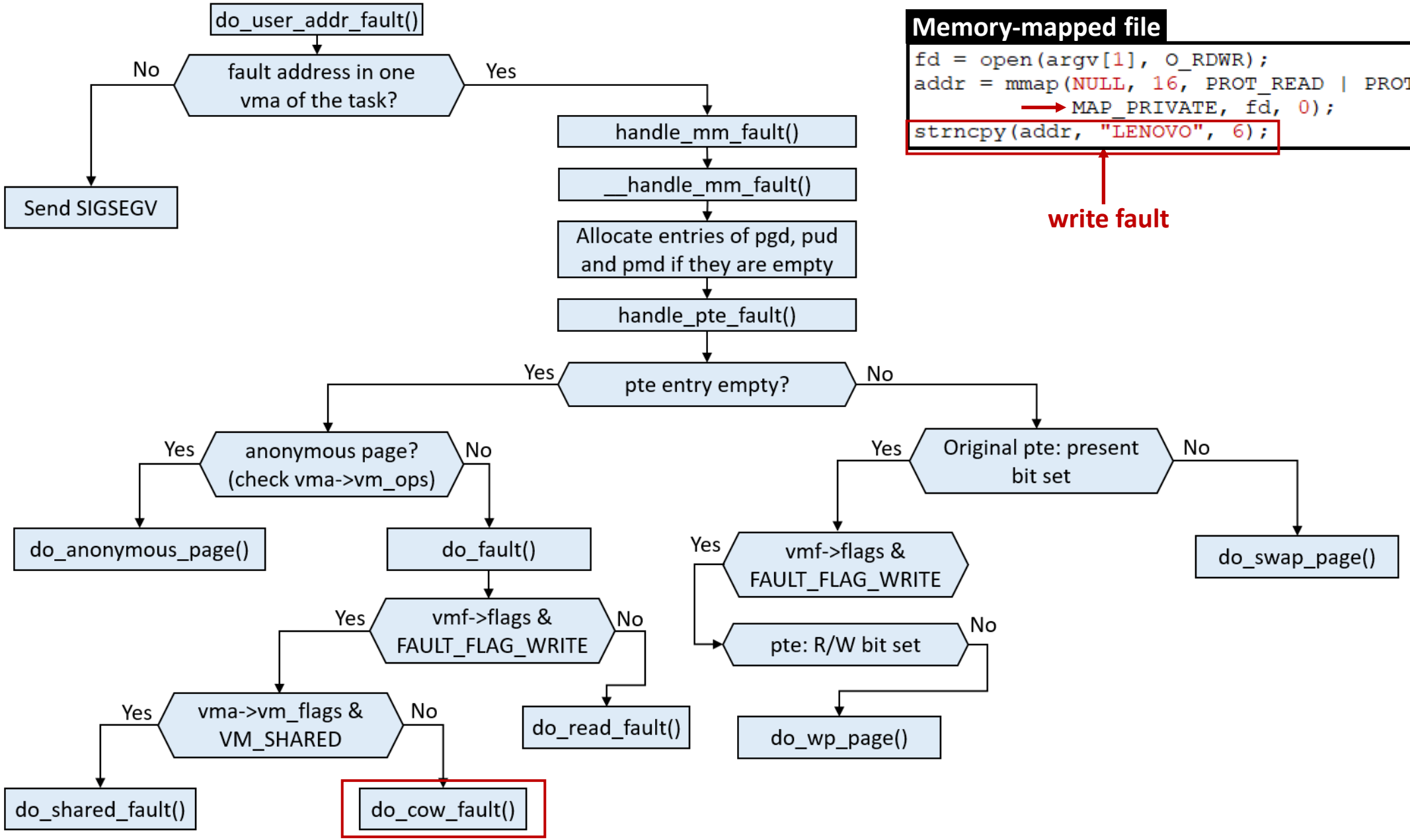


write fault without previously reading (MAP_SHARED): do_shared_fault

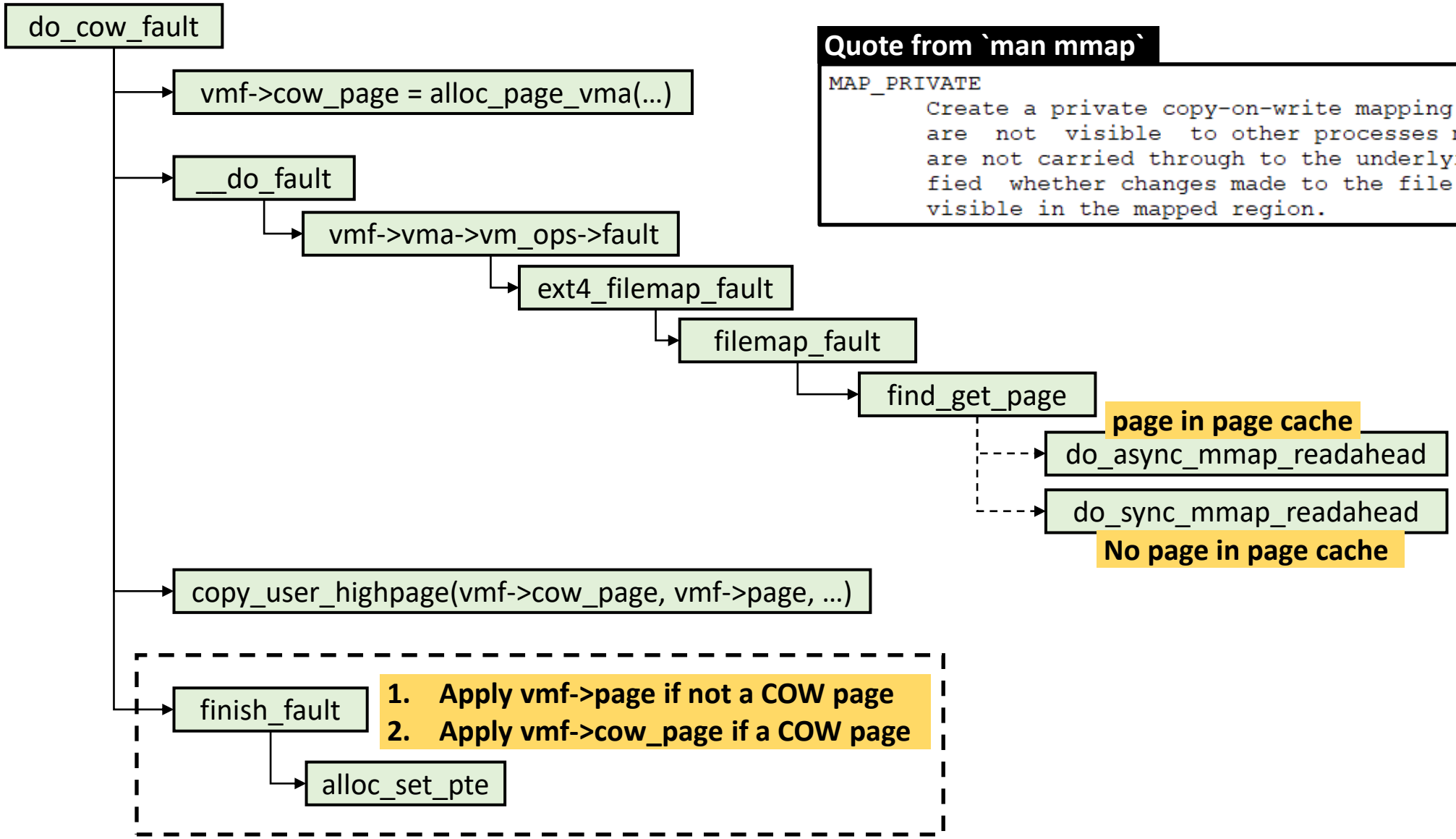



```
(gdb) bt
#0  alloc_set_pte (vmf=vmf@entry=0xffffc90000077e38, page=0xffffea000900c4c0)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3802
#1  0xffffffff81093b2b in finish_fault (vmf=vmf@entry=0xffffc90000077e38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3882
#2  0xffffffff81093efb in do_shared_fault (vmf=0xffffc90000077e38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4091
#3  do_fault (vmf=vmf@entry=0xffffc90000077e38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4151
#4  0xffffffff81094458 in handle_pte_fault (vmf=0xffffc90000077e38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4387
#5  __handle_mm_fault (flags=629, address=140737354108928, vma=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4522
#6  handle_mm_fault (vma=<optimized out>,
    address=address@entry=140737354108928, flags=flags@entry=629,
    regs=regs@entry=0xffffc90000077f58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4620
#7  0xffffffff81025107 in do_user_addr_fault (
    regs=regs@entry=0xffffc90000077f58, hw_error_code=hw_error_code@entry=6,
    address=address@entry=140737354108928)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1393
#8  0xffffffff8122232b in handle_page_fault (address=140737354108928,
    error_code=6, regs=0xffffc90000077f58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1450
#9  exc_page_fault (regs=0xffffc90000077f58, error_code=6)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1506
#10 0xffffffff81400a6b in asm_exc_page_fault ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include
/asm/idtentry.h:580
#11 0x0000000000000000 in ?? ()
(gdb) p /x vmf->address
$2 = 0x7ffff7ff9000
(gdb) p /x 140737354108928
$3 = 0x7ffff7ff9000
```

write fault without previously reading (MAP_PRIVATE): do_cow_fault



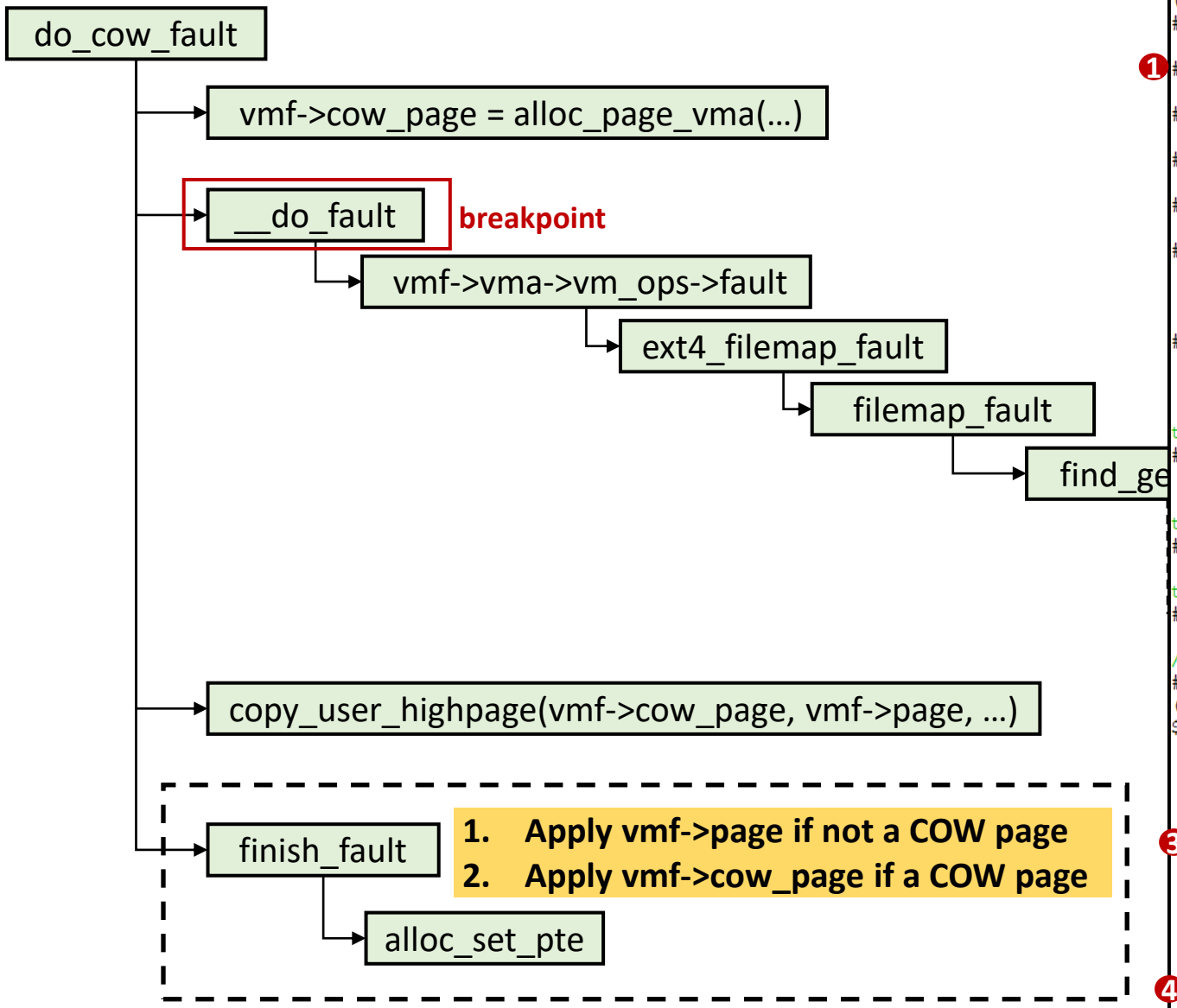
write fault without previously reading (MAP_PRIVATE): do_cow_fault



Quote from `man mmap`

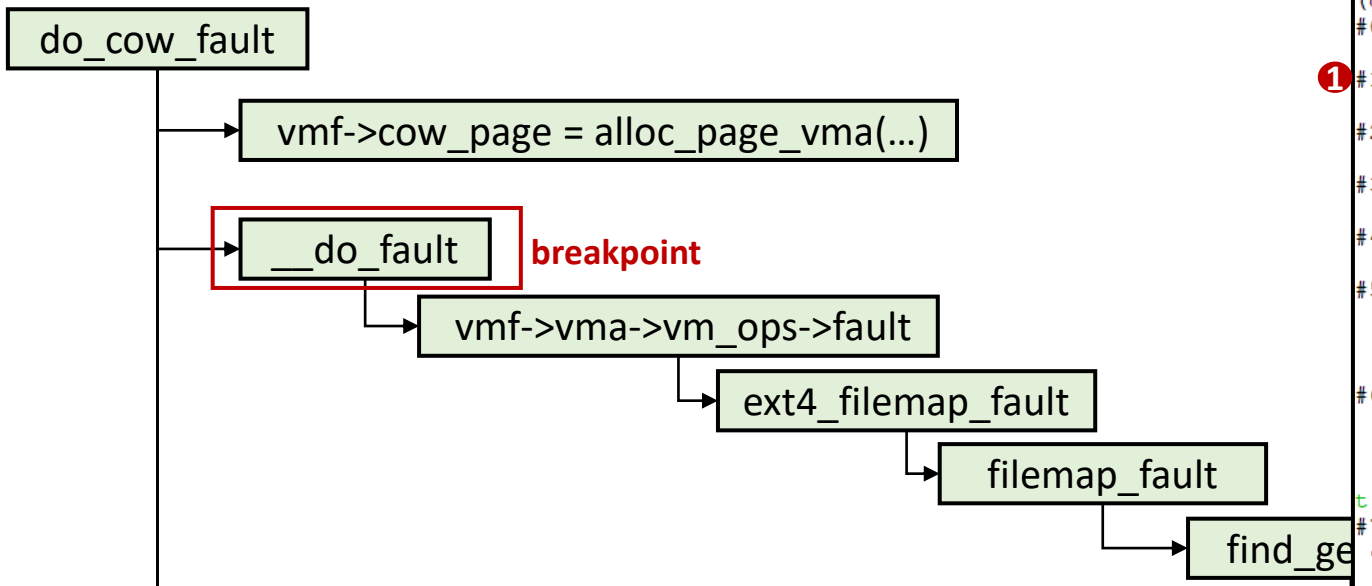
```
MAP_PRIVATE
       Create a private copy-on-write mapping.  Updates to the  mapping
       are  not  visible  to other processes mapping the same file, and
       are not carried through to the underlying file.  It is  unspeci-
       fied  whether changes made to the file after the mmap() call are
       visible in the mapped region.
```

write fault without previously reading (MAP_PRIVATE): do_cow_fault




```
(gdb) bt
#0  __do_fault (vmf=vmf@entry=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3600
#1  0xffffffff81093bc1 in do_cow_fault (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4048
#2  do_fault (vmf=vmf@entry=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4149
#3  0xffffffff81094458 in handle_pte_fault (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4387
#4  __handle_mm_fault (flags=597, address=140737354108928, vma=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4522
#5  handle_mm_fault (vma=<optimized out>,
    address=address@entry=140737354108928, flags=flags@entry=597,
    regs=regs@entry=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4620
#6  0xffffffff81025107 in do_user_addr_fault (
    regs=regs@entry=0xffffc9000007ff58, hw_error_code=hw_error_code@entry=6,
    address=address@entry=140737354108928)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/fault.c:1393
#7  0xffffffff8122232b in handle_page_fault (address=140737354108928,
    error_code=6, regs=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/fault.c:1450
#8  exc_page_fault (regs=0xffffc9000007ff58, error_code=6)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/fault.c:1506
#9  0xffffffff81400a6b in asm_exc_page_fault ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include/asm/idtentry.h:580
#10 0x0000000000000000 in ?? ()
(gdb) p /x *vmf
$5 = {
  vma = 0xffff8881047740b8,
  flags = 0x255,
  gfp_mask = 0x100cca,
  pgoff = 0x0,
  address = 0x7ffff7ff9000,
  pmd = 0xffff888104785df8,
  pud = 0xffff88810484fff8,
  orig_pte = {
    pte = 0x0
  },
  cow_page = 0xffffea000900c640,
  page = 0x0,
  pte = 0x0,
  ptl = 0x0,
  prealloc_pte = 0x0
}
```

write fault without previously reading (MAP_PRIVATE): do_cow_fault



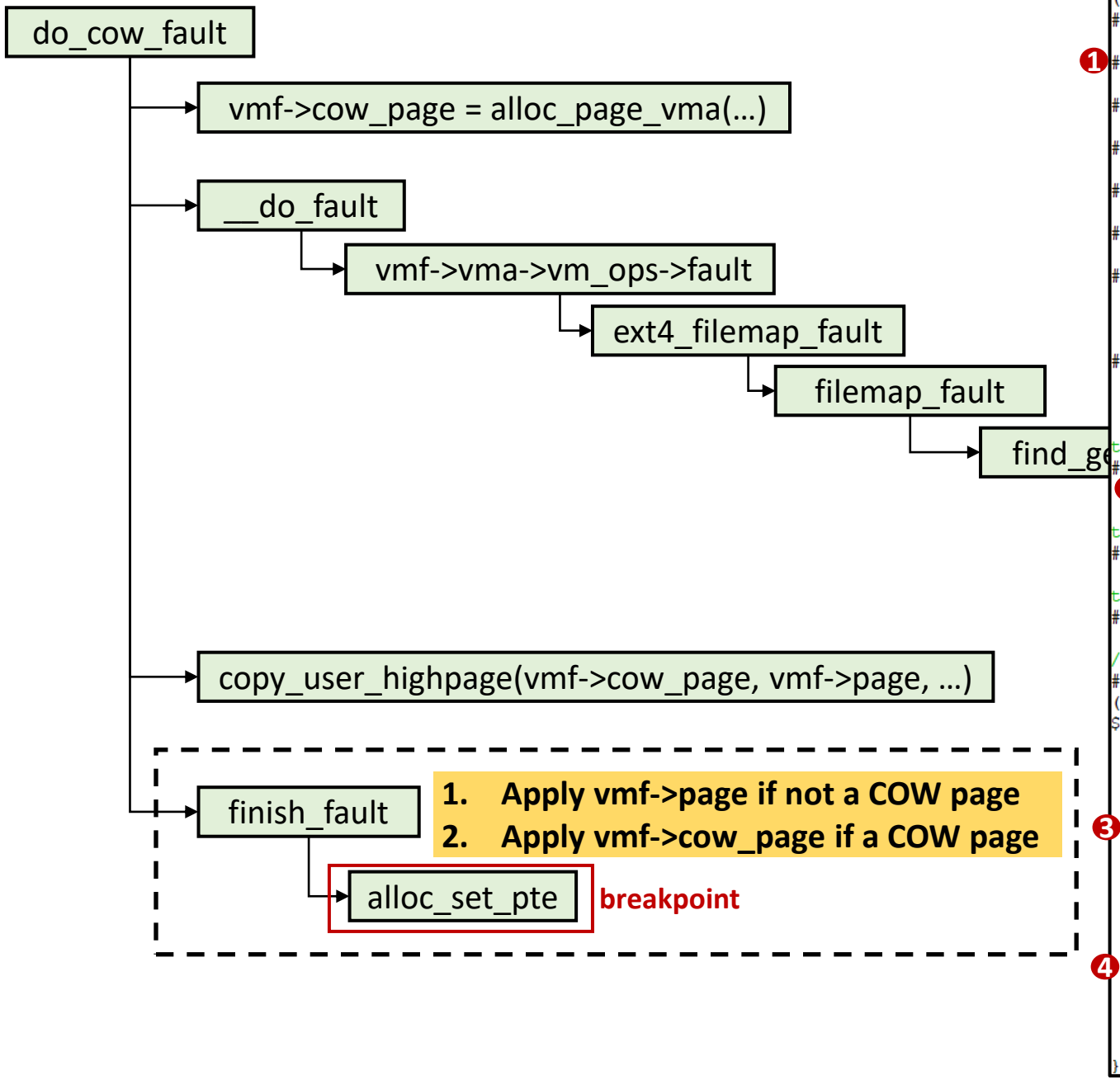
❷

```
/*
 * Page fault error code bits:
 *
 *   bit 0 ==     0: no page found       1: protection fault
 *   bit 1 ==     0: read access         1: write access
 *   bit 2 ==     0: kernel-mode access  1: user-mode access
 *   bit 3 ==                           1: use of reserved bit detected
 *   bit 4 ==                           1: fault was an instruction fetch
 *   bit 5 ==                           1: protection keys block access
 *   bit 15 ==                          1: SGX MMU page-fault
 */
enum x86_pf_error_code {
        X86_PF_PROT     =               1 << 0,
        X86_PF_WRITE    =               1 << 1,
        X86_PF_USER     =               1 << 2,
        X86_PF_RSVD     =               1 << 3,
        X86_PF_INSTR    =               1 << 4,
        X86_PF_PK       =               1 << 5,
        X86_PF_SGX      =               1 << 15,
};
```

❶

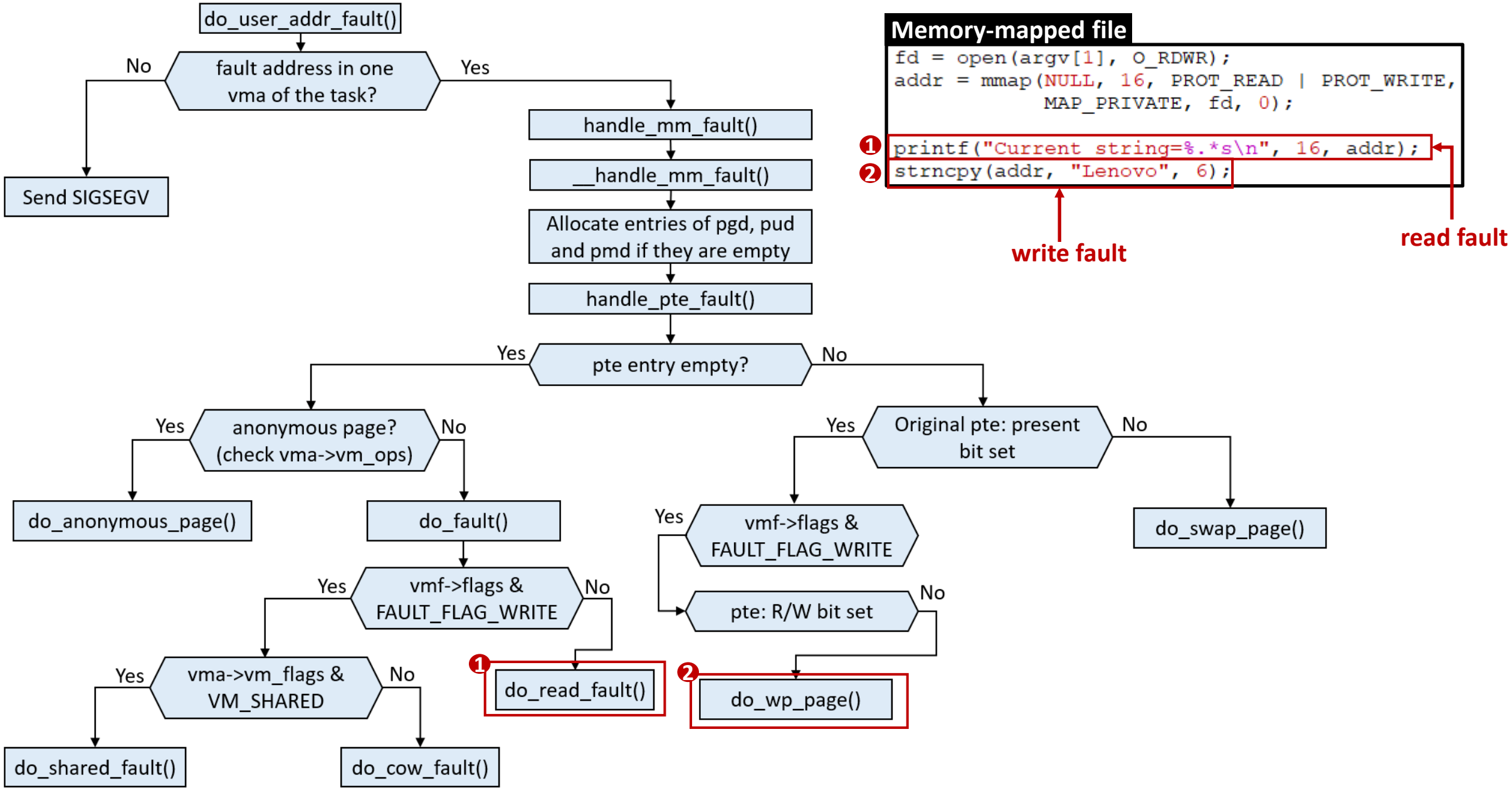
```
(gdb) bt
#0  __do_fault (vmf=vmf@entry=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3600
#1  0xffffffff81093bc1 in do_cow_fault (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4048
#2  do_fault (vmf=vmf@entry=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4149
#3  0xffffffff81094458 in handle_pte_fault (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4387
#4  __handle_mm_fault (flags=597, address=140737354108928, vma=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4522
#5  handle_mm_fault (vma=<optimized out>,
    address=address@entry=140737354108928, flags=flags@entry=597,
    regs=regs@entry=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4620
#6  0xffffffff81025107 in do_user_addr_fault (
    regs=regs@entry=0xffffc9000007ff58, hw_error_code=hw_error_code@entry=6,
    address=address@entry=140737354108928)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/fault.c:1393
#7  0xffffffff8122232b in handle_page_fault (address=140737354108928,
    error_code=6, regs=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/fault.c:1450
#8  exc_page_fault (regs=0xffffc9000007ff58, error_code=6)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/fault.c:1506
#9  0xffffffff81400a6b in asm_exc_page_fault ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include/asm/idtentry.h:580
#10 0x0000000000000000 in ?? ()
(gdb) p /x *vmf
$5 = {
  vma = 0xffff8881047740b8,
  flags = 0x255,
  gfp_mask = 0x100cca,
  pgoff = 0x0,
  address = 0x7ffff7ff9000,
  pmd = 0xffff888104785df8,
  pud = 0xffff88810484fff8,
  orig_pte = {
    pte = 0x0
  },
  cow_page = 0xffffea000900c640,
  page = 0x0,
  pte = 0x0,
  ptl = 0x0,
  prealloc_pte = 0x0
}
```

write fault without previously reading (MAP_PRIVATE): do_cow_fault

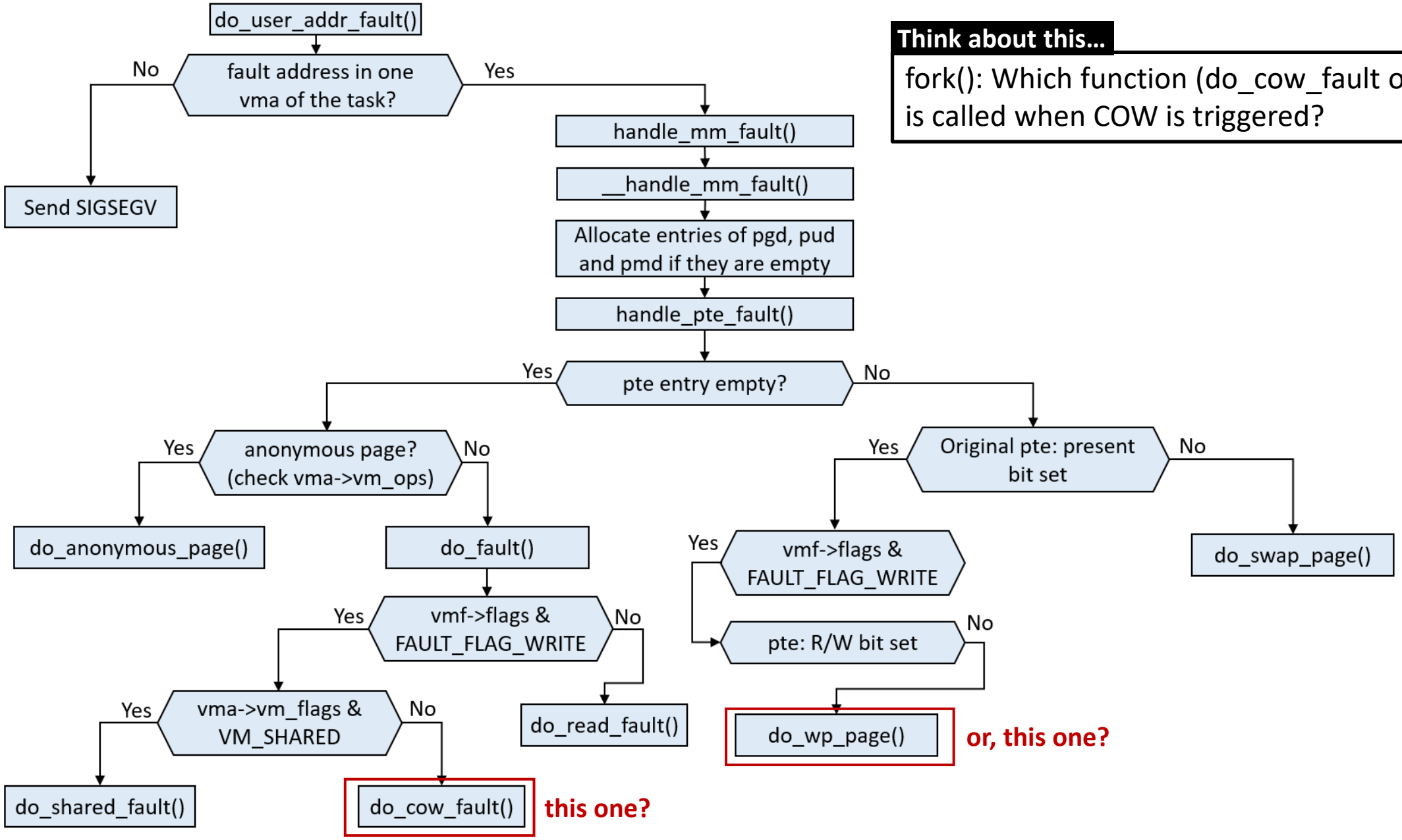



```
(gdb) bt
#0  alloc_set_pte (vmf=vmf@entry=0xffffc9000007fe38, page=0xffffea000900c640)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3802
❶#1  0xffffffff81093b2b in finish_fault (vmf=vmf@entry=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:3882
#2  0xffffffff81093c4a in do_cow_fault (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4057
#3  do_fault (vmf=vmf@entry=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4149
#4  0xffffffff81094458 in handle_pte_fault (vmf=0xffffc9000007fe38)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4387
#5  __handle_mm_fault (flags=597, address=140737354108928, vma=<optimized out>)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4522
#6  handle_mm_fault (vma=<optimized out>,
    address=address@entry=140737354108928, flags=flags@entry=597,
    regs=regs@entry=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/mm/memory.c:4620
#7  0xffffffff81025107 in do_user_addr_fault (
    regs=regs@entry=0xffffc9000007ff58, hw_error_code=hw_error_code@entry=6,
    address=address@entry=140737354108928)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1393
#8  0xffffffff8122232b in handle_page_fault (address=140737354108928,
❷   error_code=6, regs=0xffffc9000007ff58)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1450
#9  exc_page_fault (regs=0xffffc9000007ff58, error_code=6)
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/mm/faul
t.c:1506
#10 0xffffffff81400a6b in asm_exc_page_fault ()
    at /home/adrian/git-repo/gdb-linux-real-mode/src/linux-5.11/arch/x86/include
/asm/idtentry.h:580
#11 0x0000000000000000 in ?? ()
(gdb) p /x *vmf
$7 = {
  vma = 0xffff8881047740b8,
  flags = 0x255,
  gfp_mask = 0x100cca,
  pgoff = 0x0,
❸ address = 0x7ffff7ff9000,
  pmd = 0xffff888104785df8,
  pud = 0xffff88810484fff8,
  orig_pte = {
    pte = 0x0
  },
❹ cow_page = 0xffffea000900c640,
  page = 0xffffea000900c4c0,
  pte = 0x0,
  ptl = 0x0,
  prealloc_pte = 0x0
}
```

[Recap] Think about this: #1



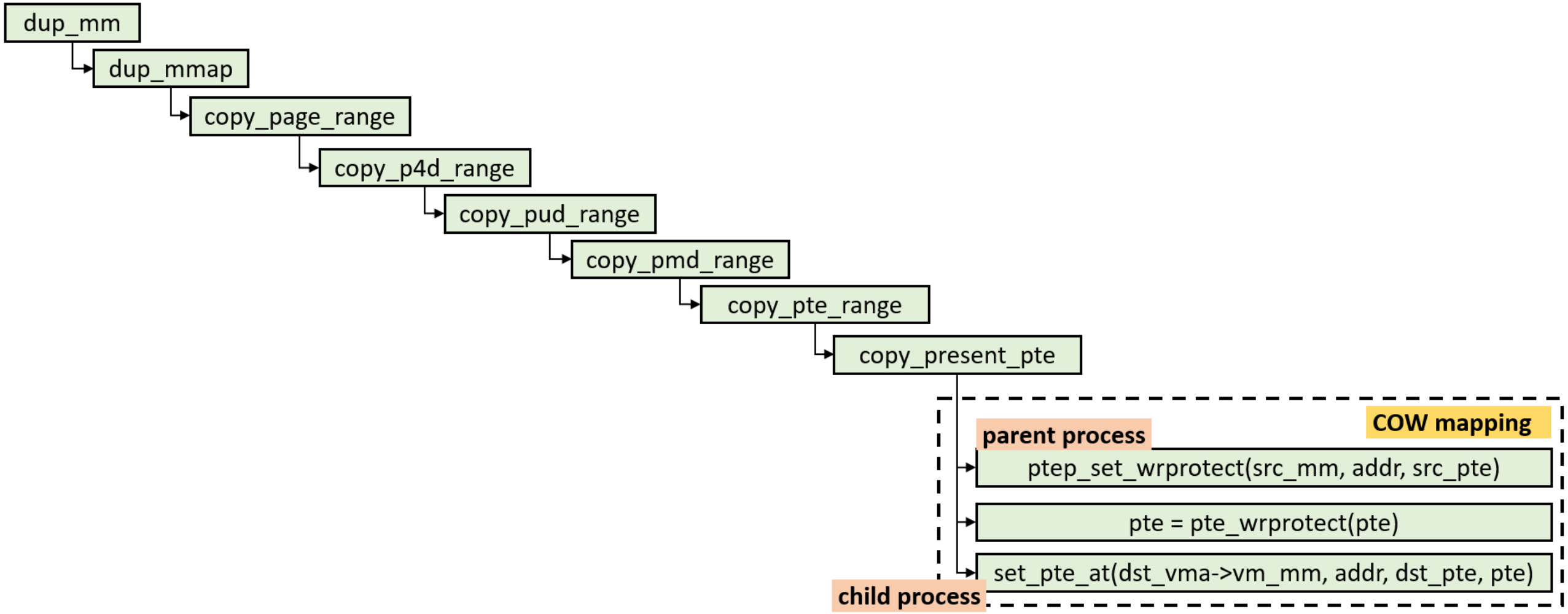
Think about this: #2



Think about this…

fork(): Which function (do_cow_fault or do_wp_page) is called when COW is triggered?

fork(): COW mapping – set ‘write protect’ for src/dst PTEs



fork(): COW mapping – set ‘write protect’ for src/dst PTEs

```
static inline bool is_cow_mapping(vm_flags_t flags)
{
        return (flags & (VM_SHARED | VM_MAYWRITE)) == VM_MAYWRITE;
}
```

mm/internal.h 287,1

dup_mm
→ dup_mmap
→ copy_page_range
→ copy_p4d_range
→ copy_pud_range
→ copy_pmd_range
→ copy_pte_range
→ copy_present_pte

```
static inline int
copy_present_pte(struct vm_area_struct *dst_vma, struct vm_area_struct *src_vma,
                 pte_t *dst_pte, pte_t *src_pte, unsigned long addr, int *rss,
                 struct page **prealloc)
{
+-- 23 lines: struct mm_struct *src_mm = src_vma->vm_mm;----------------------
        if (is_cow_mapping(vm_flags) && pte_write(pte)) {
                ptep_set_wrprotect(src_mm, addr, src_pte);
                pte = pte_wrprotect(pte);
        }

+-- 16 lines: If it's a shared mapping, mark it clean in----------------------
        set_pte_at(dst_vma->vm_mm, addr, dst_pte, pte);
        return 0;
}
```

mm/memory.c 841,1-8 16%

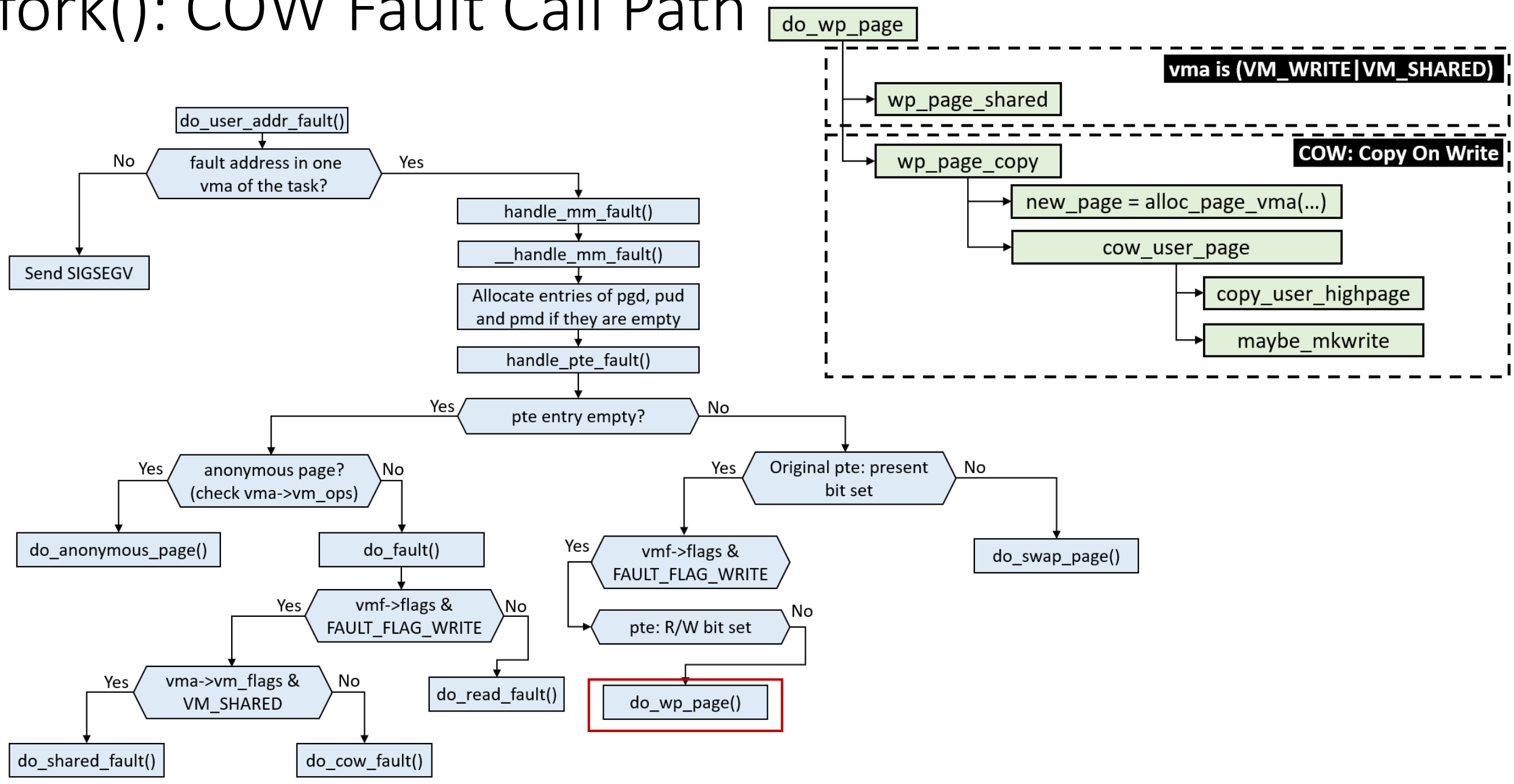
COW mapping

parent process

- ptep_set_wrprotect(src_mm, addr, src_pte)
- pte = pte_wrprotect(pte)
- set_pte_at(dst_vma->vm_mm, addr, dst_pte, pte)

ild process

fork(): COW Fault Call Path



Backup

vdso & vvar

```
/ # cat /proc/43/maps
00400000-00401000 r--p 00000000 00:02 114                                /anon_mmap
00401000-00496000 r-xp 00001000 00:02 114                                /anon_mmap
00496000-004bd000 r--p 00096000 00:02 114                                /anon_mmap
004be000-004c1000 r--p 000bd000 00:02 114                                /anon_mmap
004c1000-004c4000 rw-p 000c0000 00:02 114                                /anon_mmap
004c4000-004e8000 rw-p 00000000 00:00 0                                  [heap]
7ffff7ffa000-7ffff7ffe000 r--p 00000000 00:00 0                          [vvar]
7ffff7ffe000-7ffff7fff000 r-xp 00000000 00:00 0                          [vdso]
7ffffffde000-7ffffffff000 rw-p 00000000 00:00 0                          [stack]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

- vsyscall (Virtual System Call)
  - The context switch overhead (user <-> kernel) of some system calls (gettimeofday, time, getcpu) is greater than execution time of those functions: Built on top of the fixed-mapped address
  - Machine code format
    - Core dump: debugger cannot provide the debugging info because symbols of this area are unavailable
- vDSO (Virtual Dynamic Shared Object)
  - ELF format
  - A small shared library that the kernel automatically maps into the address space of all userspace applications
- VVAR (vDSO Variable)

```
$ readelf -S out/obj/linux/vmlinux
Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  ...
  [16] .vvar             PROGBITS         ffffffff81ac4000  00cc4000
       0000000000001000  0000000000000000  WA       0     0     16
  ...
```