Zhipeng Tian
P436
Assignment 3B
3/31/12

1. List the kernel functions that are called inside the system call sys mmap.

sys_mmap()---->

do_mmap2()---->

do_mmap_pgoff()---->

get_unmapped_area()
vma_merge()
deny_write_access()
shmem_zero_setup()
find_vma_prepare()
vma_link()
make_pages_present()
zap_page_range()

2. List the synchronization primitives and variables that the kernel uses when manipulating the memory-related data structures, mm struct and vm area struct.

mm_struct:
struct vm_area_struct * mmap;
struct rb_root mm_rb;
struct vm_area_struct * mmap_cache;
unsigned long mmap_base;
unsigned long task_size;
unsigned long cached_hole_size;
unsigned long free_area_cache;
pgd_t * pgd;
atomic_t mm_users;
atomic_t mm_count;
int map_count;
spinlock_t page_table_lock;
struct rw_semaphore mmap_sem;
struct list_head mmlist;
unsigned long hiwater_rss;
unsigned long hiwater_vm;
unsigned long total_vm, locked_vm, shared_vm, exec_vm;
unsigned long stack_vm, reserved_vm, def_flags, nr_ptes;
unsigned long start_code, end_code, start_data, end_data;
unsigned long start_brk, brk, start_stack;
unsigned long arg_start, arg_end, env_start, env_end;

```
unsigned long saved_auxv[AT_VECTOR_SIZE];
struct mm_rss_stat rss_stat;
struct linux_binfmt *binfmt;
cpumask_var_t cpu_vm_mask_var;
mm_context_t context;
unsigned int faultstamp;
unsigned int token_priority;
unsigned int last_interval;
atomic_t oom_disable_count;
unsigned long flags;
struct core_state *core_state;
spinlock_t              ioctx_lock;
struct hlist_head       ioctx_list;
struct task_struct __rcu *owner;
struct file *exe_file;
unsigned long num_exe_file_vmas;
struct mmu_notifier_mm *mmu_notifier_mm;
pgtable_t pmd_huge_pte; /* protected by page_table_lock */
struct cpumask cpumask_allocation;
```

vm_area_struct:

```
struct mm_struct * vm_mm;
unsigned long vm_start;
unsigned long vm_end;
struct vm_area_struct *vm_next, *vm_prev;
pgprot_t vm_page_prot;
unsigned long vm_flags;
struct rb_node vm_rb;
union {
        struct {
                struct list_head list;
                void *parent;   /* aligns with prio_tree_node parent */
                struct vm_area_struct *head;
        } vm_set;
        struct raw_prio_tree_node prio_tree_node;
} shared;
struct list_head anon_vma_chain;
struct anon_vma *anon_vma;
const struct vm_operations_struct *vm_ops;
unsigned long vm_pgoff;
struct file * vm_file;
void * vm_private_data;
struct vm_region *vm_region;
struct mempolicy *vm_policy;
```

3. What does the function find vma do and how does it work?

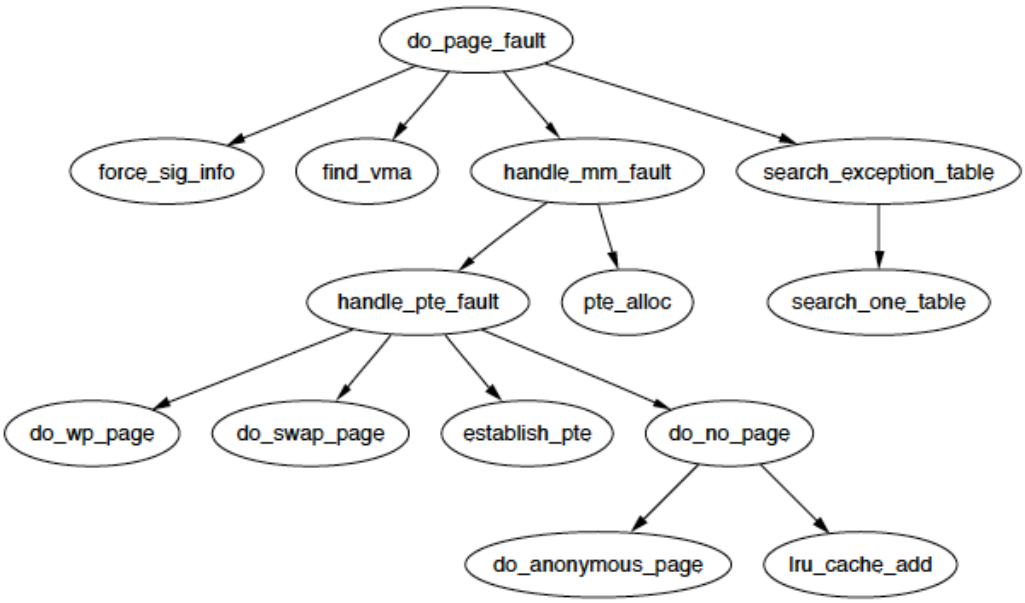
The find_vma() function locates the first memory region which satisfies addr < vm_end and returns the address of its descriptor, returns a NULL pointer if nothing is found.

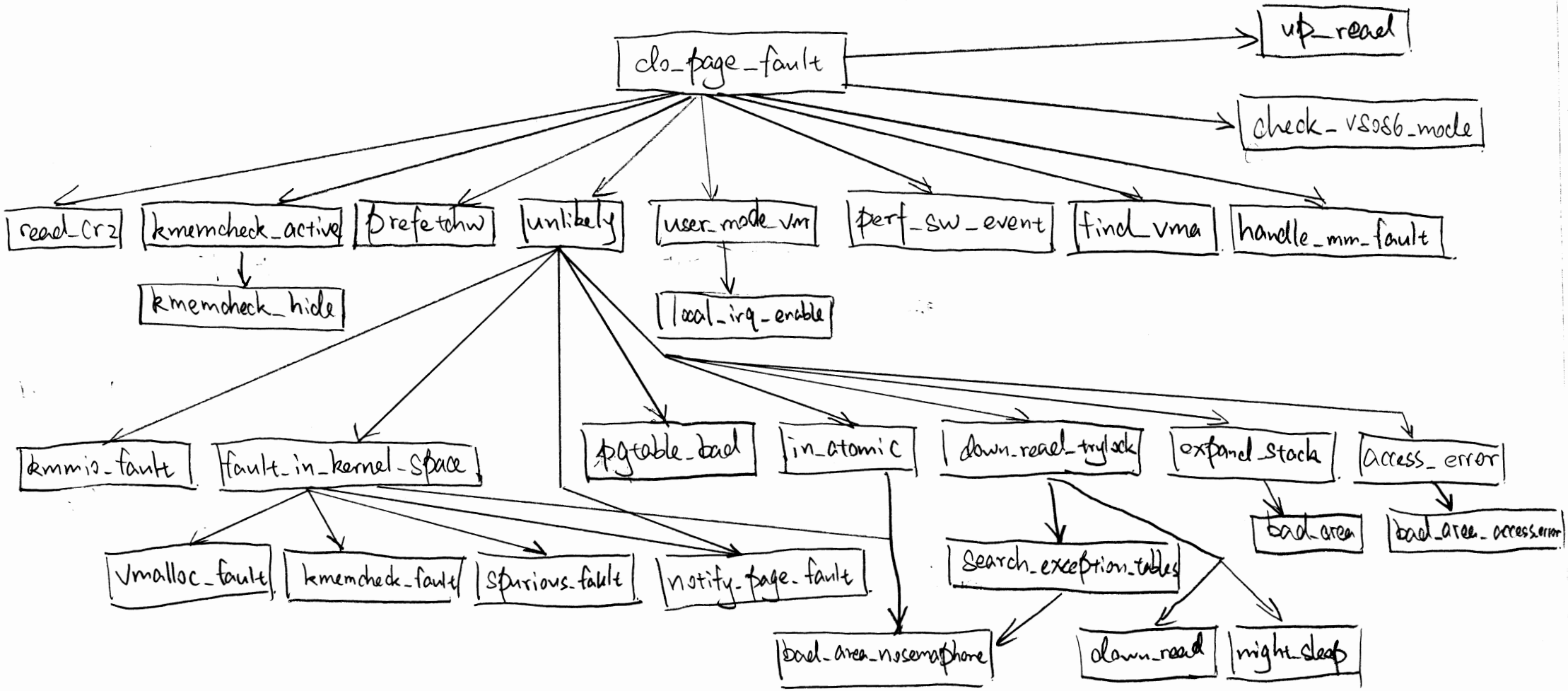
It checks cache first, checks whether the region identified by mmap_cache has addr inside it. If so, it returns the region descriptor pointer. Otherwise, it will scan the memory region of the process by looking up the memory region in the red-black tree.

4. A call graph is a graph in which vertices are function names and there is an edge from a vertex f1 to f2 if function f1 calls function f2. Notice that a call graph may have self edges (due to recursion) and cycles (due to mutual recursion). Draw a call graph for do page fault. This should include all the helper functions called under different circumstances.

The first call graph is the online resource I found, my own version of do_page_fault() call graph is on the next page, and it is based on the file arch/x86/mm/fault.c.

Here is the call graph created by **Mel Gorman** in his "**Code Commentary On The Linux Virtual Memory Manager**". It shows the call graph on a deeper level than me, which helped me understand the do_page_fault() much better.





* arch/x86/mm/fault.c
* All functions calls are inside do_page_fault()

Zhifeng Tian