第 0019 讲 4 缺页异常分析

内存管理专题--4 缺页异常分析

一、写时复制缺页异常

写时复制（COW，Copy-on-write）是一种可推迟甚至避免复制数据信息的技术，在 Linux 内核当中主要用于在 fork()系统调用里面。

应用场景：fork，父进程利用 fork()函数创建子进程，父子进程都共享父进程的匿名页面，当其中一方需要修改数据内核时，COW 就是会发生。

do_wp_page(.....)

二、文件映射缺页中断

Linux 内核关联具体文件的内存映射称为文件映射，产生的物理内存叫页高速缓存。当页面为文件映射时，会定义 VMA 的 fault 方法函数，fault 方法表示当要访问的物理页面不在内存时，此方法就称为缺页中断处理函数调用。

include > linux > C mm.h > vm_operations_struct

```
415  /*
416   * These are the virtual MM functions - opening of an area, closing and
417   * unmapping it (needed to keep files on disk up-to-date etc), pointer
418   * to the functions called when a no-page or a wp-page exception occurs.
419   */
420  struct vm_operations_struct {
421      void (*open)(struct vm_area_struct * area);
422      void (*close)(struct vm_area_struct * area);
423      int (*split)(struct vm_area_struct * area, unsigned long addr);
424      int (*mremap)(struct vm_area_struct * area);
425      vm_fault_t (*fault)(struct vm_fault *vmf);
426      vm_fault_t (*huge_fault)(struct vm_fault *vmf,
427              enum page_entry_size pe_size);
428      void (*map_pages)(struct vm_fault *vmf,
429              pgoff_t start_pgoff, pgoff_t end_pgoff);
430      unsigned long (*pagesize)(struct vm_area_struct * area);
431
```

应用场景：动态库映射、mmap 读文件数据信息。

三、匿名页面缺页异常

Linux 内核中没有关联到文件映射的页面称为匿名页面，比如采用 malloc 函数分配内存或者采用 mmap 分配匿名映射的内存等等，在缺页异常处理中匿名页面处理的核心函数为：do_anonymous_page(.....)。

mm > C memory.c > ...

```
2878  /*
2879   * We enter with non-exclusive mmap_sem (to exclude vma changes,
2880   * but allow concurrent faults), and pte mapped but not yet locked.
2881   * We return with mmap_sem still held, but pte unmapped and unlocked.
2882   */
2883  static vm_fault_t do_anonymous_page(struct vm_fault *vmf)
2884  {
2885      struct vm_area_struct *vma = vmf->vma;
2886      struct mem_cgroup *memcg;
2887      struct page *page;
2888      vm_fault_t ret = 0;
2889      pte_t entry;
```

应用场景：malloc()分配内存。

判断条件：pte 页表项中 PRESENT 没有置位、pte 内容为空并且没有指定 vma->vm_ops->fault()函数指针。

swap 缺页异常 do_swap_page()。