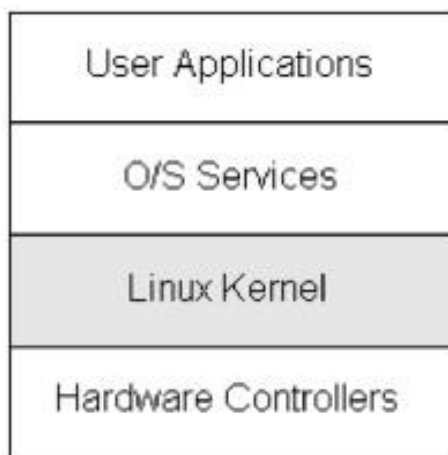




## 第0017讲 2页回收机制 (二)



零声学院讲师: Vico老师



## 一、回收不活动页

## 二、页交换



# 一、回收不活动页

负责回收不活动页shrink\_inactive\_list()执行流程源码分析如下:

```
mm > C vmscan.c
1702 static noinline_for_stack unsigned long
1703 shrink_inactive_list(unsigned long nr_to_scan, struct lruvec *lruvec,
1704                      struct scan_control *sc, enum lru_list lru)
1705 {
1706     LIST_HEAD(page_list);
1707     unsigned long nr_scanned;
1708     unsigned long nr_reclaimed = 0;
1709     unsigned long nr_taken;
1710     struct reclaim_stat stat = {};
1711     isolate_mode_t isolate_mode = 0;
```



回收不活动页的主要工作由函数shrink\_page\_list()实现,  
具体执行流程源码分析如下:

```
mm > C vmscan.c
947  */
948  static unsigned long shrink_page_list(struct list_head *page_list,
949                                       struct pglist_data *pgdat,
950                                       struct scan_control *sc,
951                                       enum ttu_flags ttu_flags,
952                                       struct reclaim_stat *stat,
953                                       bool force_reclaim)
954  {
955      LIST_HEAD(ret_pages);
956      LIST_HEAD(free_pages);
957      int pgactivate = 0;
```



## 不活动页转换成活动页的情况操作:

- 1、页表映射的页
- 2、没有页表映射的文件页

## 不活动页保留在不活动页链表中或者回收的情况操作:

- 1、页表映射的不活动页
- 2、没有页表映射的文件页



## 二、页交换



零声学院

www.0voice.com

一切只为渴望更优秀的你!

页交换（swap）的原理：当内存不足的时候，把最近很少访问的没有存储设备支持的物理页的数据暂时保存到交换区，释放内存空间，当交换区中存储的页被访问的时候，再把数据从交换区读到内存中。其中交换区可以是一个磁盘分区，也可以是存储设备上的一个文件。

目前常用的存储设备：机械硬盘、固态硬盘及NAND闪存。



## 1、技术原理

a.交换区格式（交换区的第一页是交换区首部，内核使用数据结构swap\_header描述

交换区首部，具体源码分析如下：

```
include > linux > C swap.h > ...  
89 union swap_header {  
90     struct {  
91         char reserved[PAGE_SIZE - 10];  
92         char magic[10];          /* SWAP-SPACE or SWAPSPACE2 */  
93     } magic;  
94     struct {  
95         char          bootbits[1024]; /* Space for disklabel etc. */
```






## b.交换区信息

内核定义了交换区信息数组swap\_info，每个数组项存储一个交换区的信息。数组项的数量是在编译时由宏MAX\_SWAPFILES指定的，通常是32，说明最多可以启用32个交换区。

```
mm > C swapfile.c > free_swap_count_continuations(swap_info_struct *)
```

```
86  /*
87  static PLIST_HEAD(swap_avail_head);
88  static DEFINE_SPINLOCK(swap_avail_lock);
89
90  struct swap_info_struct *swap_info[MAX_SWAPFILES];
91
92  static DEFINE_MUTEX(swapon_mutex);
93
94  static DECLARE_WAIT_QUEUE_HEAD(proc_poll_wait);
95  /* Activity counter to indicate that a swapon or swapoff has occurred */
96  static atomic_t proc_poll_event = ATOMIC_INIT(0);
97
```







结构体swap\_info\_struct描述交换区的信息如下:

include > linux > C swap.h > swap\_info\_struct

```
209  /*  
210  struct swap_info_struct {  
211      unsigned long    flags;        /* SWP_USED etc: see above */  
212      signed short     prio;         /* swap priority of this type */  
213      struct plist_node list;        /* entry in swap_active_head */  
214      struct plist_node avail_list; /* entry in swap_avail_head */  
215      signed char type;             /* strange name for an index */  
216      unsigned int     max;          /* extent of the swap_map */  
217      unsigned char *swap_map;      /* vmalloc'ed array of usage counts */  
218      struct swap_cluster_info *cluster_info; /* cluster info. Only for SSD */  
219      struct swap_cluster_list free_clusters; /* free clusters list */
```



### c.交换区间

交换区间 (swap extent) 用来把交换区的连续槽位映射到连续的磁盘块。如果交换区是磁盘分区，因为磁盘分区的块是连续的，所以 只需要一个交换区间。如果交换区是文件，因为文件对应的磁盘块不一定是连续的，所以对于每个连续的磁盘块范围，需要使用一个交换区间来存储交换区的连续槽位和磁盘块范围的映射关系。

```
include > linux > C swap.h > swap_info_struct
```

```
210 struct swap_info_struct {  
211     unsigned long    flags;        /* SWP_USED etc: see above */  
212     signed short     prio;        /* swap priority of this type */  
213     struct plist_node list;       /* entry in swap_active_head */
```

```
227     struct swap_extent *curr_swap_extent;  
228     struct swap_extent first_swap_extent;
```



## d.交换槽位缓存

为了加快为换出页分配交换槽位的速度，每个处理器有一个交换槽位缓存

swap\_slots\_cache数据结构，源码分析如下：

```
include > linux > C swap_slots.h > ...  
11  
12 struct swap_slots_cache {  
13     bool        lock_initialized;  
14     struct mutex alloc_lock; /* protects slots, nr, cur */  
15     swp_entry_t *slots;  
16     int         nr;  
17     int         cur;  
18     spinlock_t  free_lock; /* protects slots_ret, n_ret */  
19     swp_entry_t *slots_ret;  
20     int         n_ret;  
21 };
```



## e.交换项

内核定义数据类型`swp_entry_t`以存储换出在交换区中的位置，我们称为交换项，高7位存储交换区的索引，其他位存储页在交换区中的偏移（单位为页）。

```
include > linux > C mm_types.h > ↩ swp_entry_t

611  */
612  typedef struct {
613      unsigned long val;
614  } swp_entry_t;
615
616  #endif /* !_LINUX_MM_TYPES_H */
617
```



## f.交换缓存

每个交换区有若干个交换缓存,  $2^{14}$ 页对应一个交换缓存, 交换缓存的数量是 (交换区的总页数/ $2^{14}$ )

```
mm > C swap_state.c > ...  
36  
37 struct address_space *swapper_spaces[MAX_SWAPFILES];  
38 static unsigned int nr_swapper_spaces[MAX_SWAPFILES];  
39  
40 #define INC_CACHE_INFO(x)    do { swap_cache_info.x++; } while (0)  
41
```





办学宗旨：一切只为渴望更优秀的你

办学愿景：让技术简单易懂