Swap Entries

2017-01-10 (http://jake.dothome.co.kr/swap-entry/) Moon Young-il (http://jake.dothome.co.kr/author/admin/) Linux Kernel (http://jake.dothome.co.kr/category/linux/)

<kernel v5.0>

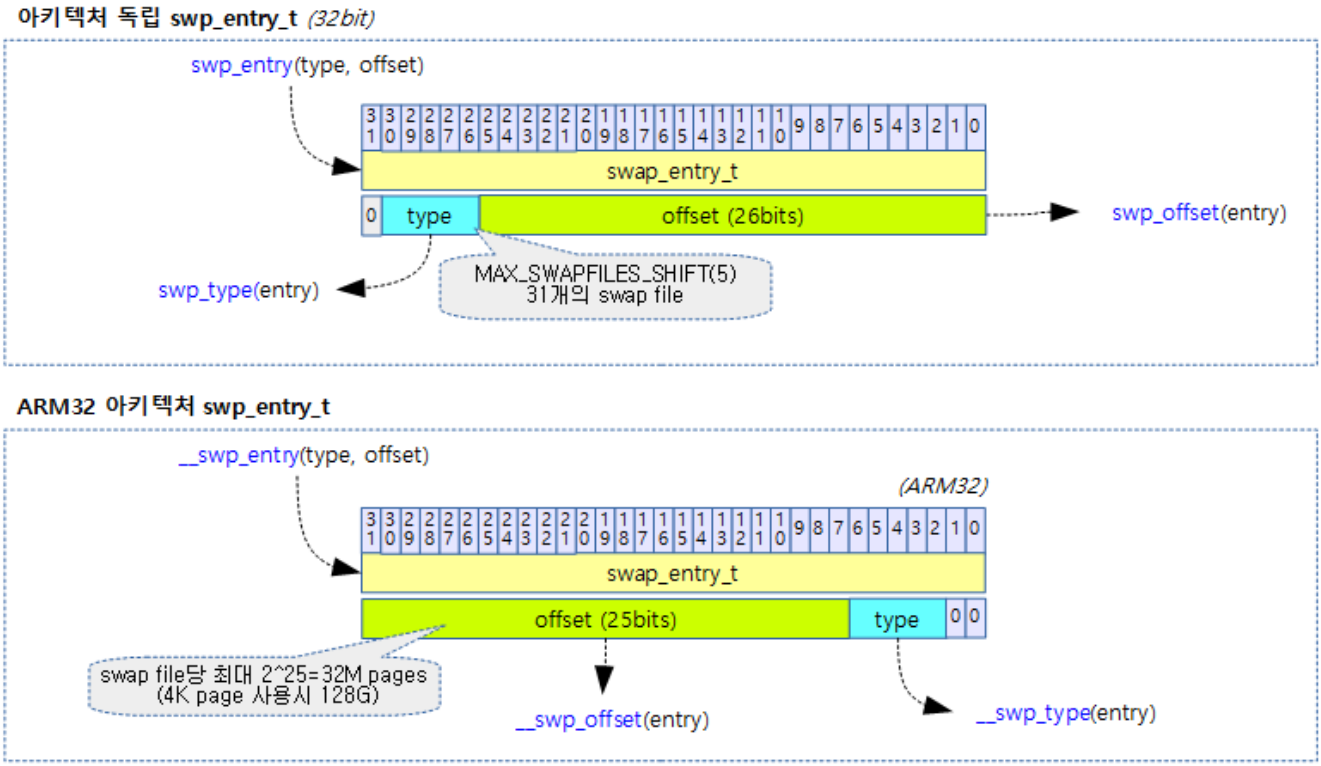
Swap Entries

In kernel v4.20-rc1, the management of swap entries in the Linux kernel has been changed from using Radix Tree Exceptional to using XArray.

- See: xarray: Replace exceptional entries (https://github.com/torvalds/linux/commit/3159f943aafdbacb2f94c38fdaadabf2bbde2a14)

swap_entry_t Structure

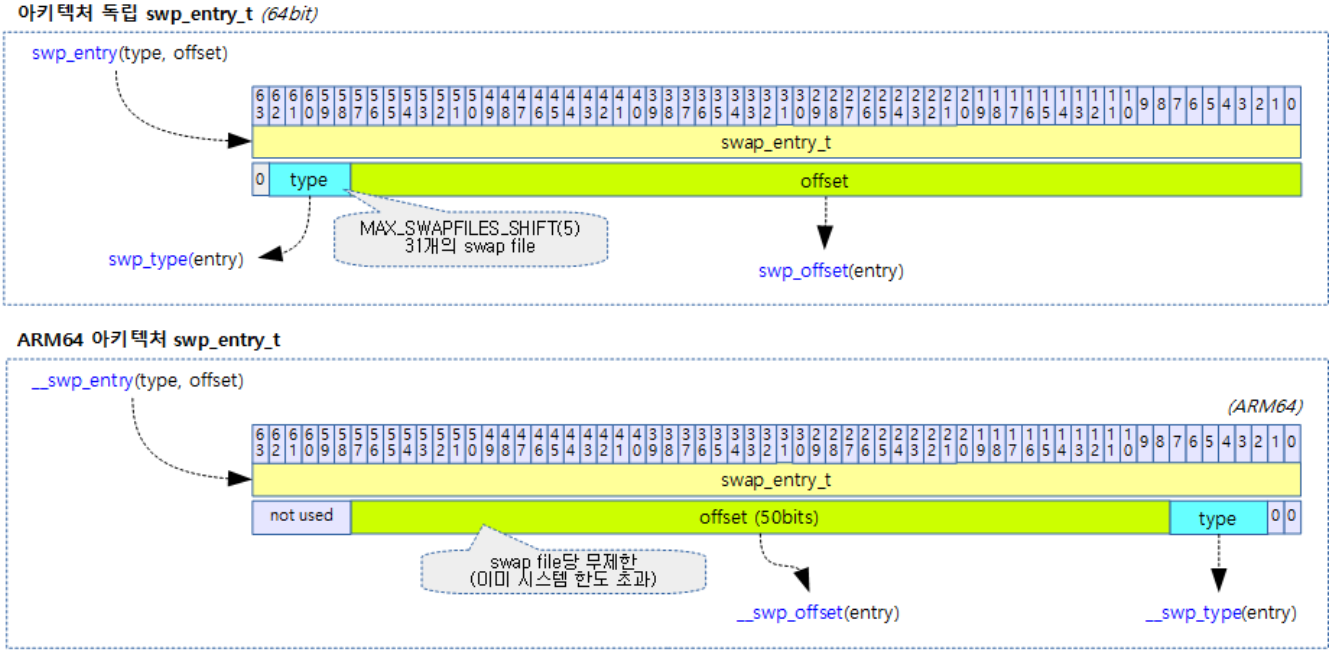
The following figure shows the swap entry structure running on a 32-bit system.



(http://jake.dothome.co.kr/wp-content/uploads/2017/01/swp_entry-1c.png)

- This value is stored in the ARM Linux PTE entry, not the ARM H/W PTE entry, which means that this page is a swap entry.

The following figure shows the swap entry structure running on a 64-bit system.



(http://jake.dothome.co.kr/wp-content/uploads/2017/01/swp_entry-2.png)

swap_entry_t Structure

include/linux/mm_types.h

```
1  /*
2   * A swap entry has to fit into a "unsigned long", as the entry is hidde
   n
3   * in the "index" field of the swapper address space.
4   */
```

```
1  typedef struct {
2          unsigned long val;
3  } swp_entry_t;
```

Architecture-independent swap entry

swp_entry()

include/linux/swapops.h

```
01 /*
02  * Store a type+offset into a swp_entry_t in an arch-independent format
03  */
04 static inline swp_entry_t swp_entry(unsigned long type, pgoff_t offset)
05 {
06         swp_entry_t ret;
07
08         ret.val = (type << SWP_TYPE_SHIFT(ret)) | (offset & SWP_OFFSET_M
   ASK);
09         return ret;
10 }
```

Configure the swap entry with the offset value and the type value.

swp_type()

include/linux/swapops.h

```
/*
 * Extract the `type' field from a swp_entry_t.  The swp_entry_t is in
 * arch-independent format
 */
```

```
static inline unsigned swp_type(swp_entry_t entry)
{
        return (entry.val >> SWP_TYPE_SHIFT);
}
```

Returns a type value from the swap entry.

swp_offset()

include/linux/swapops.h

```
/*
 * Extract the `offset' field from a swp_entry_t.  The swp_entry_t is in
 * arch-independent format
 */
```

```
static inline pgoff_t swp_offset(swp_entry_t entry)
{
        return entry.val & SWP_OFFSET_MASK;
}
```

Returns the value of offset from the swap entry.

SWP_TYPE_SHIFT() & SWP_OFFSET_MASK()

include/linux/swapops.h

```
/*
 * swapcache pages are stored in the swapper_space radix tree.  We want to
 * get good packing density in that tree, so the index should be dense in
 * the low-order bits.
 *
 * We arrange the `type' and `offset' fields so that `type' is at the seven
 * high-order bits of the swp_entry_t and `offset' is right-aligned in the
 * remaining bits.  Although `type' itself needs only five bits, we allow for
 * shmem/tmpfs to shift it all up a further two bits: see swp_to_radix_entry().
 *
 * swp_entry_t's are *never* stored anywhere in their arch-dependent format.
 */
```

```
#define SWP_TYPE_SHIFT        (BITS_PER_XA_VALUE - MAX_SWAPFILES_SHIFT)
#define SWP_OFFSET_MASK       ((1UL << SWP_TYPE_SHIFT) - 1)
```

Macro Constants

include/linux/swap.h

```
/*
 * MAX_SWAPFILES defines the maximum number of swaptypes: things which c
an
 * be swapped to.  The swap type and the offset into that swap type are
 * encoded into pte's and into pgoff_t's in the swapcache.  Using five b
its
 * for the type means that the maximum number of swapcache pages is 27 b
its
 * on 32-bit-pgoff_t architectures.  And that assumes that the architect
ure packs
 * the type/offset into the pte as 5/27 as well.
 */
```

```
#define MAX_SWAPFILES_SHIFT     5
```

include/linux/swap.h

```
/*
 * NUMA node memory migration support
 */
```

```
#ifdef CONFIG_MIGRATION
#define SWP_MIGRATION_NUM 2
#define SWP_MIGRATION_READ      (MAX_SWAPFILES + SWP_HWPOISON_NUM)
#define SWP_MIGRATION_WRITE     (MAX_SWAPFILES + SWP_HWPOISON_NUM + 1)
#else
#define SWP_MIGRATION_NUM 0
#endif
```

ARM32 Swap Entry

__swp_entry()

arch/arm/include/asm/pgtable.h

```
#define __swp_entry(type,offset) ((swp_entry_t) { ((type) << __SWP_TYPE_
SHIFT) | ((offset) << __SWP_OFFSET_SHIFT) })
```

The offset value and the type value are used to construct the arm swap entry.

__swp_type()

arch/arm/include/asm/pgtable.h

```
#define __swp_type(x)           (((x).val >> __SWP_TYPE_SHIFT) & __SWP_T
YPE_MASK)
```

Arm swap entry.

__swp_offset()

arch/arm/include/asm/pgtable.h

```
#define __swp_offset(x)         ((x).val >> __SWP_OFFSET_SHIFT)
```

Arm Swap returns the offset value from the entry.

Macro Constants

arch/arm/include/asm/pgtable.h

```
/*
 * Encode and decode a swap entry.  Swap entries are stored in the Linux
 * page tables as follows:
 *
 *   3 3 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1
 *   1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
 *   <--------------- offset ----------------------> < type -> 0 0
 *
 * This gives us up to 31 swap files and 128GB per swap file.  Note that
 * the offset field is always non-zero.
 */
```

```
#define __SWP_TYPE_SHIFT        2
#define __SWP_TYPE_BITS         5
#define __SWP_TYPE_MASK         ((1 << __SWP_TYPE_BITS) - 1)
#define __SWP_OFFSET_SHIFT      (__SWP_TYPE_BITS + __SWP_TYPE_SHIFT)
```

ARM64 Swap Entry

__swp_entry()

arch/arm64/include/asm/pgtable.h

```
#define __swp_entry(type,offset) ((swp_entry_t) { ((type) << __SWP_TYPE_
SHIFT) | ((offset) << __SWP_OFFSET_SHIFT) })
```

The offset value and the type value make up the arm64 swap entry.

__swp_type()

arch/arm64/include/asm/pgtable.h

```
#define __swp_type(x)           (((x).val >> __SWP_TYPE_SHIFT) & __SWP_T
YPE_MASK)
```

arm64 swap entry.

__swp_offset()

arch/arm64/include/asm/pgtable.h

```
#define __swp_offset(x)         (((x).val >> __SWP_OFFSET_SHIFT) & __SWP
_OFFSET_MASK)
```

arm64 swap entry returns a value of 50 bit offset.

Macro Constants

arch/arm64/include/asm/pgtable.h

```
/*
 * Encode and decode a swap entry:
 *      bits 0-1:       present (must be zero)
 *      bits 2-7:       swap type
 *      bits 8-57:      swap offset
 *      bit  58:        PTE_PROT_NONE (must be zero)
 */
```

```
#define __SWP_TYPE_SHIFT        2
#define __SWP_TYPE_BITS         6
#define __SWP_OFFSET_BITS       50
#define __SWP_TYPE_MASK         ((1 << __SWP_TYPE_BITS) - 1)
#define __SWP_OFFSET_SHIFT      (__SWP_TYPE_BITS + __SWP_TYPE_SHIFT)
#define __SWP_OFFSET_MASK       ((1UL << __SWP_OFFSET_BITS) - 1)
```

```

```

Swap PTE Entry Identification

is_swp_pte()

include/linux/swapops.h

```
/* check whether a pte points to a swap entry */
static inline int is_swap_pte(pte_t pte)
{
        return !pte_none(pte) && !pte_present(pte);
}
```

Returns whether it is a swapped pte entry.

- If the PTE is not NONE and there is no PRESENT setting, it is in a swap state.

Swap Cache

Shared pages with slots reserved for backing storage are considered swap caches. The swap cache differs from the file cache in the following two ways.

- page->mapping uses &swapper_space[]. (address_space)
- Use the add_to_swap_cache() function instead of add_to_page_cache().

name *

email *

Website

WRITE A COMMENT

‹ Exception -4- (ARM32 VFP & FPE) (http://jake.dothome.co.kr/vfp-fpe/)

Exception -6- (MM Fault Handler) › (http://jake.dothome.co.kr/mm-fault/)

Munc Blog (2015 ~ 2024)