pagemap, from the userspace perspective

---

pagemap is a new (as of 2.6.25) set of interfaces in the kernel that allow userspace programs to examine the page tables and related information by reading files in /proc.

There are three components to pagemap:

* /proc/pid/pagemap. This file lets a userspace process find out which physical frame each virtual page is mapped to. It contains one 64-bit value for each virtual page, containing the following data (from fs/proc/task_mmu.c, above pagemap_read):

  ```
  * Bits 0-54  page frame number (PFN) if present
  * Bits 0-4   swap type if swapped
  * Bits 5-54  swap offset if swapped
  * Bits 55-60 page shift (page size = 1<<page shift)
  * Bit  61    reserved for future use
  * Bit  62    page swapped
  * Bit  63    page present
  ```

  If the page is not present but in swap, then the PFN contains an encoding of the swap file number and the page's offset into the swap. Unmapped pages return a null PFN. This allows determining precisely which pages are mapped (or in swap) and comparing mapped pages between processes.

  Efficient users of this interface will use /proc/pid/maps to determine which areas of memory are actually mapped and llseek to skip over unmapped regions.

* /proc/kpagecount. This file contains a 64-bit count of the number of times each page is mapped, indexed by PFN.

* /proc/kpageflags. This file contains a 64-bit set of flags for each page, indexed by PFN.

  The flags are (from fs/proc/page.c, above kpageflags_read):

  ```
   0. LOCKED
   1. ERROR
   2. REFERENCED
   3. UPTODATE
   4. DIRTY
   5. LRU
   6. ACTIVE
   7. SLAB
   8. WRITEBACK
   9. RECLAIM
  10. BUDDY
  11. MMAP
  12. ANON
  13. SWAPCACHE
  14. SWAPBACKED
  15. COMPOUND_HEAD
  ```

```
    16. COMPOUND_TAIL
    16. HUGE
    18. UNEVICTABLE
    19. HWPOISON
    20. NOPAGE
    21. KSM

Short descriptions to the page flags:

 0. LOCKED
    page is being locked for exclusive access, eg. by undergoing read/write IO

 7. SLAB
    page is managed by the SLAB/SLOB/SLUB/SLQB kernel memory allocator
    When compound page is used, SLUB/SLQB will only set this flag on the head
    page; SLOB will not flag it at all.

10. BUDDY
    a free memory block managed by the buddy system allocator
    The buddy system organizes free memory in blocks of various orders.
    An order N block has 2^N physically contiguous pages, with the BUDDY flag
    set for and _only_ for the first page.

15. COMPOUND_HEAD
16. COMPOUND_TAIL
    A compound page with order N consists of 2^N physically contiguous pages.
    A compound page with order 2 takes the form of "HTTT", where H donates its
    head page and T donates its tail page(s).  The major consumers of compound
    pages are hugeTLB pages (Documentation/vm/hugetlbpage.txt), the SLUB etc.
    memory allocators and various device drivers. However in this interface,
    only huge/giga pages are made visible to end users.
17. HUGE
    this is an integral part of a HugeTLB page

19. HWPOISON
    hardware detected memory corruption on this page: don't touch the data!

20. NOPAGE
    no page frame exists at the requested address

21. KSM
    identical memory pages dynamically shared between one or more processes

    [IO related page flags]
 1. ERROR     IO error occurred
 3. UPTODATE  page has up-to-date data
              ie. for file backed page: (in-memory data revision >= on-disk one)
 4. DIRTY     page has been written to, hence contains new data
              ie. for file backed page: (in-memory data revision >  on-disk one)
 8. WRITEBACK page is being synced to disk

    [LRU related page flags]
 5. LRU         page is in one of the LRU lists
 6. ACTIVE      page is in the active LRU list
18. UNEVICTABLE page is in the unevictable (non-)LRU list
                It is somehow pinned and not a candidate for LRU page reclaims,
```

```
                eg. ramfs pages, shmctl(SHM_LOCK) and mlock() memory segments
 2. REFERENCED  page has been referenced since last LRU list enqueue/requeue
 9. RECLAIM     page will be reclaimed soon after its pageout IO completed
11. MMAP        a memory mapped page
12. ANON        a memory mapped page that is not part of a file
13. SWAPCACHE   page is mapped to swap space, ie. has an associated swap entry
14. SWAPBACKED  page is backed by swap/RAM
```

The page-types tool in this directory can be used to query the above flags.

Using pagemap to do something useful:

The general procedure for using pagemap to find out about a process' memory usage goes like this:

1. Read /proc/pid/maps to determine which parts of the memory space are mapped to what.
2. Select the maps you are interested in -- all of them, or a particular library, or the stack or the heap, etc.
3. Open /proc/pid/pagemap and seek to the pages you would like to examine.
4. Read a u64 for each page from pagemap.
5. Open /proc/kpagecount and/or /proc/kpageflags. For each PFN you just read, seek to that entry in the file, and read the data you want.

For example, to find the "unique set size" (USS), which is the amount of memory that a process is using that is not shared with any other process, you can go through every map in the process, find the PFNs, look those up in kpagecount, and tally up the number of pages that are only referenced once.

Other notes:

Reading from any of the files will return -EINVAL if you are not starting the read on an 8-byte boundary (e.g., if you seeked an odd number of bytes into the file), or if the size of the read is not a multiple of 8 bytes.