Linux Device Driver
(Kernel Memory Allocation)

Amir Hossein Payberah
payberah@yahoo.com
Contents

- kmalloc
- get_free_page
- vmalloc
The kmalloc allocation engine is a powerful tool,
- It is similar to malloc.
- The function is fast and it doesn’t clear the memory it obtains.
- The allocated region is also contiguous in physical memory.
kmalloc

- `void *kmalloc(unsigned int size, int priority);`
- It is defined in `<linux/malloc.h>`
Kmalloc priority

- **GFP_KERNEL**
  - Normal allocation of kernel memory. May sleep.

- **GFP_ATOMIC**
  - Used to allocate memory from interrupt handlers and other code outside of a process context. Never sleeps.

- **_ _GFP_DMA**
  - This flag requests memory usable in DMA data transfers to/from devices.
Kmalloc size

- Linux handles memory allocation by creating a set of pools of memory objects of fixed sizes.
- Allocation requests are handled by going to a pool that holds sufficiently large objects, and handing an entire memory chunk back to the requester.
- The data sizes available are generally powers of two.
kfree

- void **kfree**(void *obj);
- It is used to free memory.
Contents

- kmalloc
- get_free_page
- vmalloc
get_free_page and friends

- If a module needs to allocate **big chunks** of memory, it is usually better to use a **page-oriented** technique.
get_free_page and friends

- unsigned long **get_zeroed_page**(int flags);
  - Returns a pointer to a new page and fills the page with zeros.

- unsigned long **_get_free_page**(int flags);
  - Similar to **get_zeroed_page**, but doesn’t clear the page.

- unsigned long **_get_free_pages**(int flags, unsigned long order);
  - Allocates and returns a pointer to the first byte of a memory area that is several (physically contiguous) pages long, but doesn’t zero the area.

- unsigned long **_get_dma_pages**(int flags, unsigned long order);
  - Similar to **get_free_pages**, but guarantees that the allocated memory is DMA capable.
get_free_page and friends

- The **flags** argument works in the same way as with kmalloc.
- **order** is the base-two logarithm of the number of pages you are requesting or freeing ($\log_2 N$).
  - For example, order is 0 if you want one page and 3 if you request eight pages.
free_page

- When a program is done with the pages, it can free them with one of the following functions.
  - void free_page(unsigned long addr);
  - void free_pages(unsigned long addr, unsigned long order);
Contents

- kmalloc
- get_free_page
- vmalloc
vmalloc

- It allocates a contiguous memory region in the virtual address space.
  - Although the pages are not necessarily consecutive in physical memory.
vmalloc

- void *vmalloc(unsigned long size);
- void vfree(void * addr);
- void *ioremap(unsigned long offset, unsigned long size);
- void iounmap(void * addr);
- They are defined in <linux/vmalloc.h>
vmalloc and ioremap

- **Like vmalloc**, ioremap builds new page tables.
- **Unlike vmalloc**, however, it doesn’t actually allocate any memory.
- ioremap is most useful for mapping the (physical) address of a PCI buffer to (virtual) kernel space.
Question?