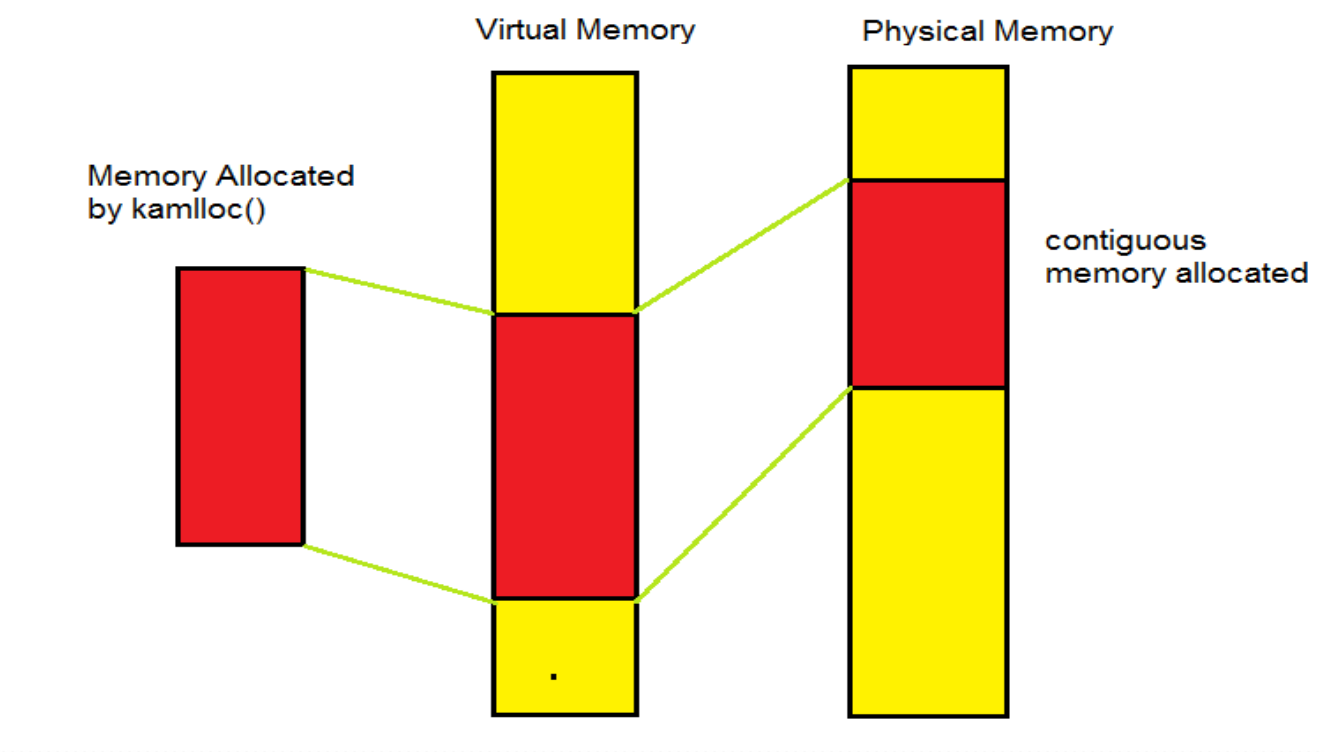
Linux Memory Management- kmalloc and vmalloc

www.embeddedshiksha.com

kmalloc:-

The kmalloc is a kernel function used to obtain kernel memory in byte-sized chunks. If we need whole pages, then we have to use function mentioned in previous section.

void * kmalloc(size_t size, gfp_t flags)



Return Values:-

On success: It returns memory at least size bytes in length.

On failure: Returns NULL.

So we have to check for return value after a call to kmalloc.

kfree

void kfree(const void *ptr)

This method will free block of memory allocated by kmalloc.

gfp_mask Flags

We have seen that the low level page allocators and kmalloc expect gfp flags as a parameter. We will discuss about these flags further and conclude this article.

Here gfp stands for – get free pages and these gfp flags are formed using various other flags. Here our knowledge of zones will come in to picture.

The gfpflags are broken up into three categories: action modifiers, zonemodifiers, and types.

Action Modifiers:-

Action modifiers specify how the kernel is supposed to allocate the requested memory. For example various contexts exist where developer needs to use different action modifiers:

- For Interrupt handlers must instruct the kernel not to sleep (because interrupt handlers cannot reschedule) in the course of allocating memory.
- In Process context – process can sleep in the course of allocating memory.

Low Level GFP Flags

__GFP_WAIT

Indicates that the caller is not high priority and can sleep or reschedule.

__GFP_HIGH

The allocator can access emergency pools and Used by a high priority or kernel process.

__GFP_IO

Indicates that the caller can perform low level IO.

__GFP_FS

The allocator can start filesystem I/O.

__GFP_HIGHIO

Determines that IO can be performed on pages mapped in high memory

Zone Modifiers :-Zone modifiers specify from which memory zone the allocation should originate. Normally,allocations can be fulfilled from any zone.

Zone Modifiers

Flag Description

__GFP_DMA :- Allocates only from ZONE_DMA

__GFP_DMA32 :- All ocates only from ZONE_DMA32

__GFP_HIGHMEM:-Allocates from ZONE_HIGHMEM or ZONE_NORMAL

Type Flags:-The type flags specify the required action and zone modifiers to fulfill a particular typeof transaction.
Low Level GFP Flag Combinations for High Level

Flag	Modifier Flags
GFP_AUTOMATIC	_GFP_HIGH
GFP_NOIO	_GFP_WAIT
GFP_NOFS	(_GFP_WAIT|_GFP_IO)
GFP_KERNEL	(_GFP_WAIT|_GFP_IO|_GFP_FS)
GFP_USER	(_GFP_WAIT|_GFP_IO|_GFP_FS)
GFP_HIGHUSER	(_GFP_WAIT|_GFP_IO|_GFP_FS|_GFP_HIGHMEM)
GFP_DMA	_GFP_DMA

Kernel provides various high level flags used according to the context you are calling the kmalloc function:

GFP_ATOMIC:-

Used to allocate memory in interrupt handlers and other code outside of a process context and it never sleeps.

Interrupts handles are not executing in process context and they cannot call ‘schedule’ function. Also interrupt handles should execute for short period of time and so should not sleep.

GFP_KERNEL

Normal allocation of kernel memory. May sleep. This can be used for example, when executing a system call in kernel on behalf of a process. This can block.

GFP_USER

Used to allocate memory for user-space pages; it may sleep.

GFP_HIGHUSER
Like GFP_USER, but allocates from high memory, if any.
GFP_NOIO
GFP_NOFS

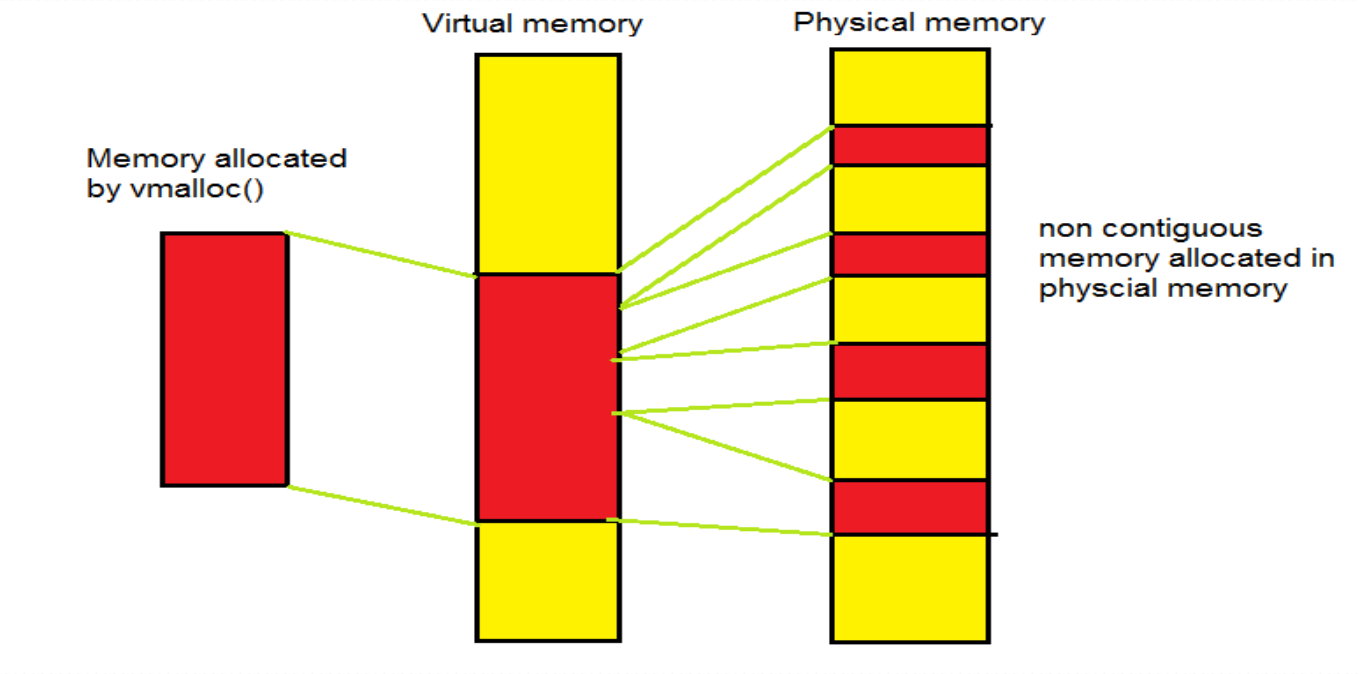
A GFP_NOFS allocation is not allowed to perform any filesystem calls, while GFP_NOIO disallows the initiation of any I/O at all. They are used primarily in the filesystem and virtual memory code where an allocation may be allowed to sleep.

Vmalloc

It allocates memory which is virtually contiguous and not necessarily physically contiguous. But do remember ‘kmalloc’ will return physically contiguous and virtual contiguous memory.

So if memory is virtually contiguous and physically not, then how virtual memory is mapped to physical memory?

Kernel does this by using page tables. This we discuss later articles.



You may also get a doubt, which is faster in allocating memory? The answer lies in the above discussion.
Since vmalloc use page tables to map the virtual address to physical address, so whenever you access memory allocated using vmalloc, you have to traverse through page table. This is additional overhead compared to kmalloc. So vmalloc should be used where it is really necessary.
void * vmalloc(unsigned long size)

On success:- Returns a pointer to at least size bytes of virtually contiguous memory.
On failure:- the function returns NULL.
To free an allocation obtained via vmalloc(), use
void vfree(const void *addr)

Thank You