

## 1. Slide27



---

## 2. Slide1

### **Memory Management in Linux**

- **Physical memory-management system;**  
deals with allocating and freeing pages, groups of pages, and small blocks of memory.
  - **Virtual memory;**  
It is memory mapped into the address space of running processes.
-

## 3. Slide19

### Physical Memory

The primary physical memory manager in the Linux kernel is the page allocator.

- The page allocator allocates and frees all physical pages; it can allocate ranges of physically contiguous pages on request.

Memory allocations in the Linux kernel occur

- statically (drivers reserve a contiguous area of memory during system boot time)
- dynamically (via the page allocator)

---

## 4. Slide5

### What is virtual memory?

Virtual memory means that the system's memory is 'virtualized' among all processes on the system.

Each process gets its own independent address space and its own view of the memory in the machine.

Virtual memory creates pages of virtual memory on demand, and manages the loading of those pages from disk or their swapping back out to disk as required.

---

## 5. Slide4

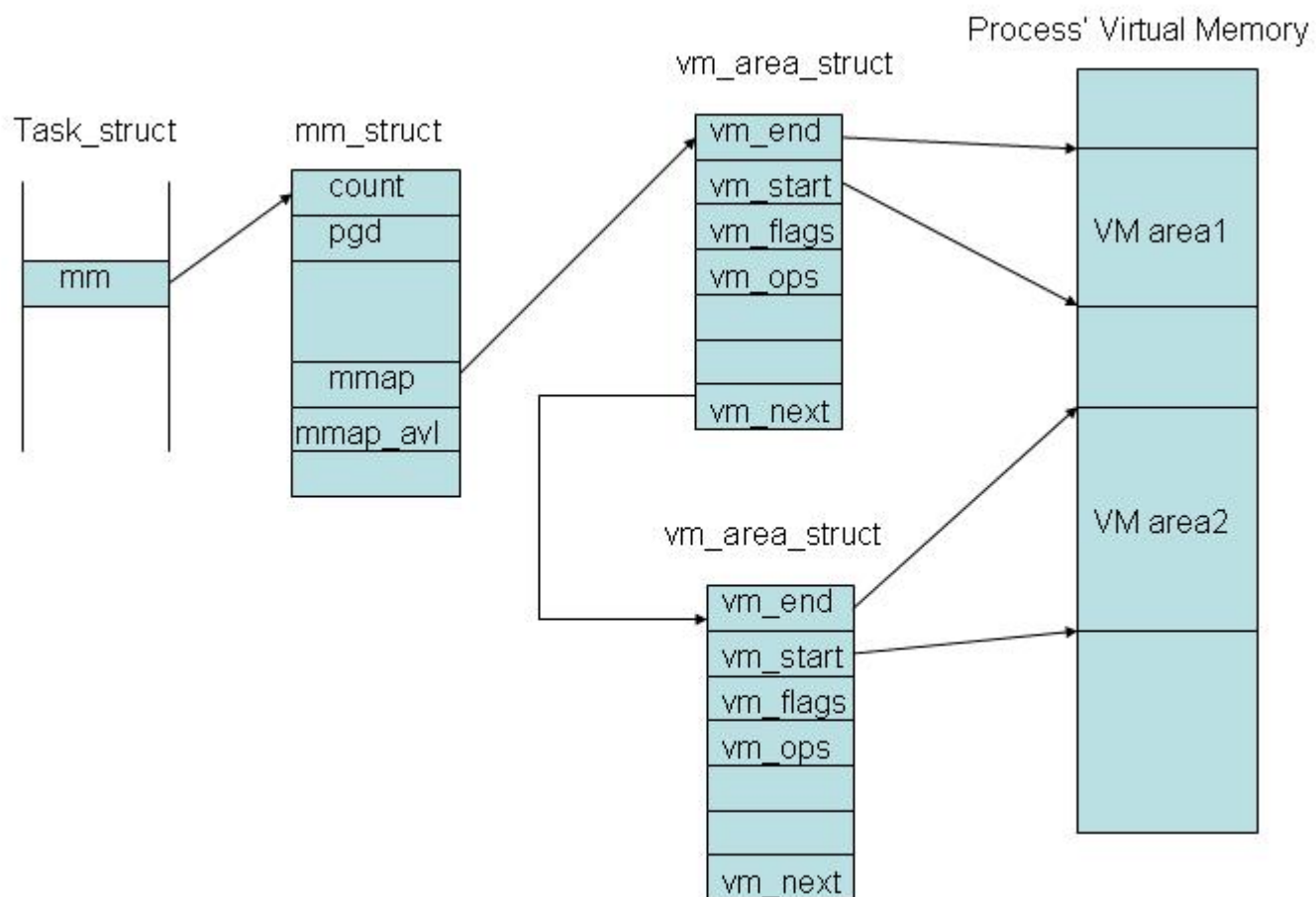
### Virtual Memory

The VM manager maintains two separate views of a process's address space:

- A logical view; describing instructions concerning the layout of the address space
  - The address space consists of a set of nonoverlapping regions, each representing a continuous, page-aligned subset of the address space.
  - Each region is described internally by a single `vm_area_struct`.
- A physical view; storing in the hardware page tables for the process.
  - The page-table entries determine the exact current location of each page of virtual memory, whether it is on disk or in physical memory.

---

## 6. Slide26



**Figure 2: A Process's Virtual Memory**

---

## 7. Slide20

### Virtual Memory Areas



- ❖ A process' address space is made up of VMA's (Virtual memory area's, or just memory areas).
- ❖ Each memory area is represented by a `vm_area_struct`.
- ❖ A process descriptor contains a pointer to a `vm_area_struct` which is a single node in a NULL terminated linked list of `vm_area_struct` objects.

```
struct vm_area_struct
{
    struct vm_area_struct*    vm_next;
    struct mm_struct*        vm_mm;
    unsigned long            vm_start; /* in bytes */
    unsigned long            vm_end;
    unsigned long            vm_flags;
};
```

---

## 8. Slide21

### Virtual Memory Areas

`vm_start`  the memory range that makes up this particular memory area.  
`vm_end` 

`vm_flags`  a value that represents many flags that can be used to describe the properties of a specific memory area.

`VM_READ` → Pages in this memory area can be read from.

`VM_WRITE` → Pages in this memory area can be written to.

`VM_EXEC` → Pages in this memory area can be executed.

`VM_SHARED` → Pages are shared.

`VM_SEQ_READ` → Pages seem to be accessed sequentially.

`VM_RAND_READ` → Pages seem to NOT be accessed sequentially.

---

## 9. Slide24

### Page Tables

- ❖ Processes operate on virtual memory addresses that are mapped to physical memory addresses.
- ❖ Before a request to access virtual memory reaches the processor it must be converted from virtual to physical. This is done with page tables.

A **page table** is a simple mapping of virtual to physical addresses.

There are three distinct types of page tables.

Each one maps a virtual address to the next page table type, except the last which maps it directly to a physical address.

- PGD (Page Global Directory) – Maps page requests to correct PMD.
- PMD (Page Middle Directory) – Maps page requests to correct PTE.
- PTE (Page Table Entry) – Maps to the corresponding page struct that contains the physical address.

---

## 10. Slide25

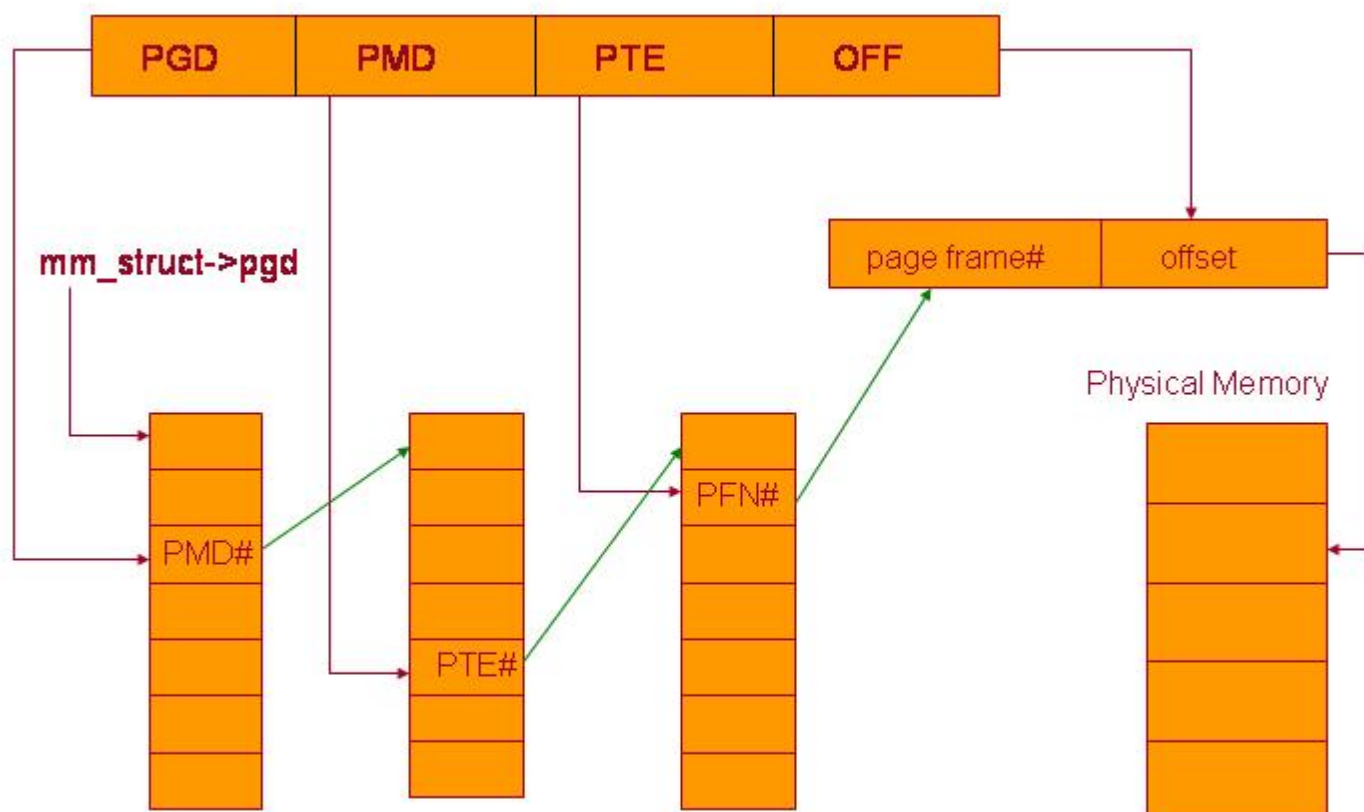


Figure3: Linux Page Tables

---

## 11. Slide9

Virtual memory regions are characterized by:

- The backing store, which describes from where the pages for a region come; regions are usually backed by a file or by nothing (*demand-zero* memory)
  
- The region's reaction to writes  
The mapping of a region into the process' address space can be either private or shared.  
private → copy on write(cow)  
shared → page sharing

---

## 12. Slide22

### Lifetime of a Virtual Address Space

The kernel creates a new virtual address space

- 1) When a process runs a new program with the `exec` system call.

On executing a new program, the process is given a new, completely empty virtual-address space; the program-loading routines populate the address space with virtual-memory regions.

- 2) Upon creation of a new process by the `fork` system call

Creating a new process with `fork` involves creating a complete copy of the existing process's virtual address space.

The kernel copies the parent process's VMA descriptors, then creates a new set of page tables for the child.

The parent's page tables are copied directly into the child's, with the reference count of each page covered being incremented

After the `fork`, the parent and child share the same physical pages of memory in their address spaces.

---

## 13. Slide23

### Paging and Segmentation

Virtual memory can be implemented via:

- Demand paging
- Demand segmentation

**Segmentation** splits the physical memory exactly as needed by each program

Arbitrary start of a block  
Arbitrary length

**Paging** splits the memory in equal small blocks (e.g., 4-64K) and assigns as many as needed to each program.

---

## 14. Slide11

Like most of the virtual memory systems, Linux use a technique called **paging**.

- ❖ The VM paging system relocates pages of memory from physical memory out to disk when the memory is needed for something else
  - ❖ The VM paging system can be divided into two sections:
    - The pageout-policy algorithm decides which pages to write out to disk, and when.
    - The paging mechanism actually carries out the transfer, and pages data back into physical memory as needed.
-

## 15. Slide12

### Kernel Virtual Memory

- ❖ The Linux kernel reserves a constant, architecture-dependent region of the virtual address space of every process for its own internal use
- ❖ The page table entries that map to these kernel pages are marked as protected, so that the pages are not visible or modifiable when the processor is running in user mode.

**Note:** The detailed information can be found in the book *Understanding the Linux Virtual Memory*, Mel Gorman, Prentice Hall, 2004

---

## 16. Slide15

### Kernel Virtual Memory

This kernel virtual-memory area contains two regions:

- A static area that contains page table references to every available physical page of memory in the system, so that there is a simple translation from physical to virtual addresses when running kernel code.
  - The core of the kernel, plus all pages allocated by the normal page allocator, reside in this region.
- The remainder of the reserved section is not reserved for any specific purpose; its page-table entries can be modified to point to any other areas of memory
  - The kernel provides allowing processes to use the this virtual memory.
  - The `vmalloc` function allocates an arbitrary number of physical pages of memory, and maps them into a single region of kernel virtual memory .



## 17. Slide28

### References

1. ***Operating System Concepts with Java***, 6th Edition by Avi Silberschatz, Peter Baer Galvin, and Greg Gagne, John Wiley and Sons.
2. <http://people.clarkson.edu/~jets> , **Linux Virtual Memory Manager**



ABC Amber PowerPoint Converter Trial version

Please register to remove this banner.

<http://www.thebeatlesforever.com/processtext/abcpower.html>