

Memory Management

Lap Kan Leung

UNIX History

- UNIX was developed at Bell Labs in the early 1970s. The first version was written by Ken Thompson in assembler for the PDP-7 minicomputer
- This was soon followed by a version for the PDP-11 written in a new language called C that was devised and implemented by Dennis Ritchie
- In 1974, Ritchie and Thompson published a landmark paper about UNIX (Ritchie and Thompson, 1974), which stimulated many universities to ask Bell Labs for a copy of UNIX

UNIX History Con't..

- **The University of California at Berkeley** was one of the many universities that acquired UNIX
- Because the complete source code was available, Berkeley was able to modify the system substantially. The changes were:
 - A port to the VAX minicomputer
 - Addition of paged virtual memory
 - Extension of the file names from 14 characters to 255
 - Inclusion of the TCP/IP networking protocol

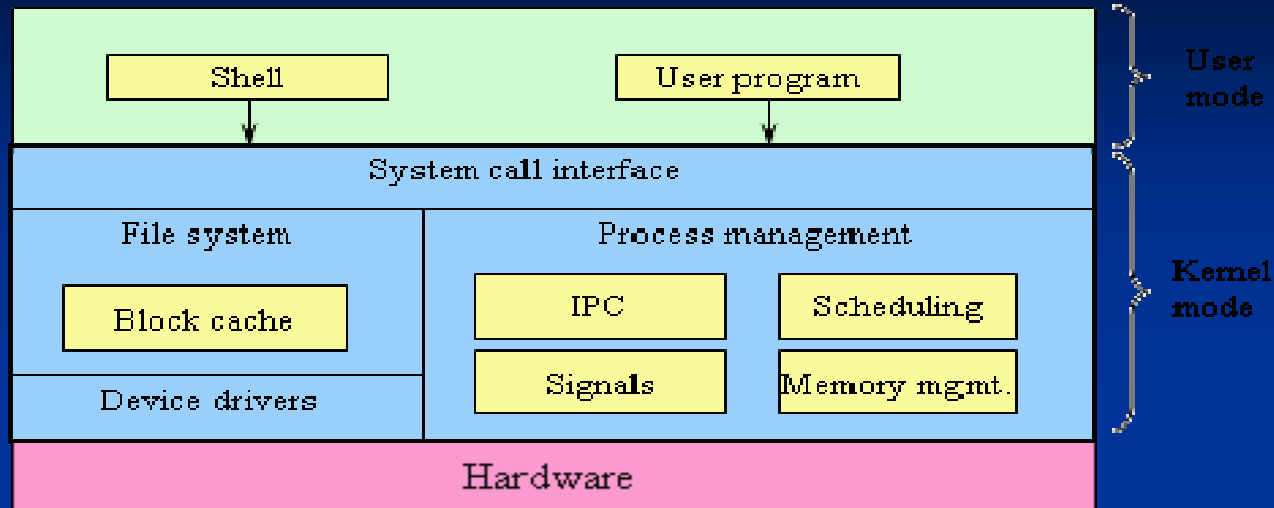
History of Virtual Memory

- Before the development of the virtual memory technique, programmers in the 1940s and 1950s had to manage two-level storage:
 - Main memory or RAM
 - Secondary memory in the form of hard disk or earlier, magnetic drums

History of Virtual Memory Con't..

- It was developed in approximately 1959-1962, at the university of Manchester for the Atlas Computer
- In 1961, Burroughs released the B5000 the first commercial computer with virtual memory
- In 1969, an IBM research team, lead by Davis Sayre, showed the virtual memory overly system worked consistently better than the best manual-controlled system
- In the 1970s, minicomputer models such as VAX models running VMS implemented virtual memory

UNIX Kernel Overview



- The Process management portion, among its various other functions, it handles:
 - IPC (InterProcess Communication), which allows process to communicate with one another and synchronize to avoid race conditions
 - Processing scheduling (based on priorities)
 - Signals, which are a form of (asynchronous) software interrupt
 - Memory management is also done here as well
- Most UNIX systems support demand paged virtual memory, sometimes with a few extra features, such as the ability of multiple process to share common regions of address space

Virtual Memory

- Is a memory management technique, used by multitasking computer operating systems wherein non-contiguous memory is presented to a software application (aka process) as contiguous memory
- The contiguous memory is referred to as the virtual address space
- Virtual memory addressing is typically used in paged memory systems. This in turn is often combined with memory swapping, whereby memory pages stored in primary storage are written to secondary storage, thus freeing faster primary storage for other processes to use
- Virtual memory allows software to run in a memory address space whose size and addressing are not necessarily tied to the computer's physical memory

FreeBSD Memory Management

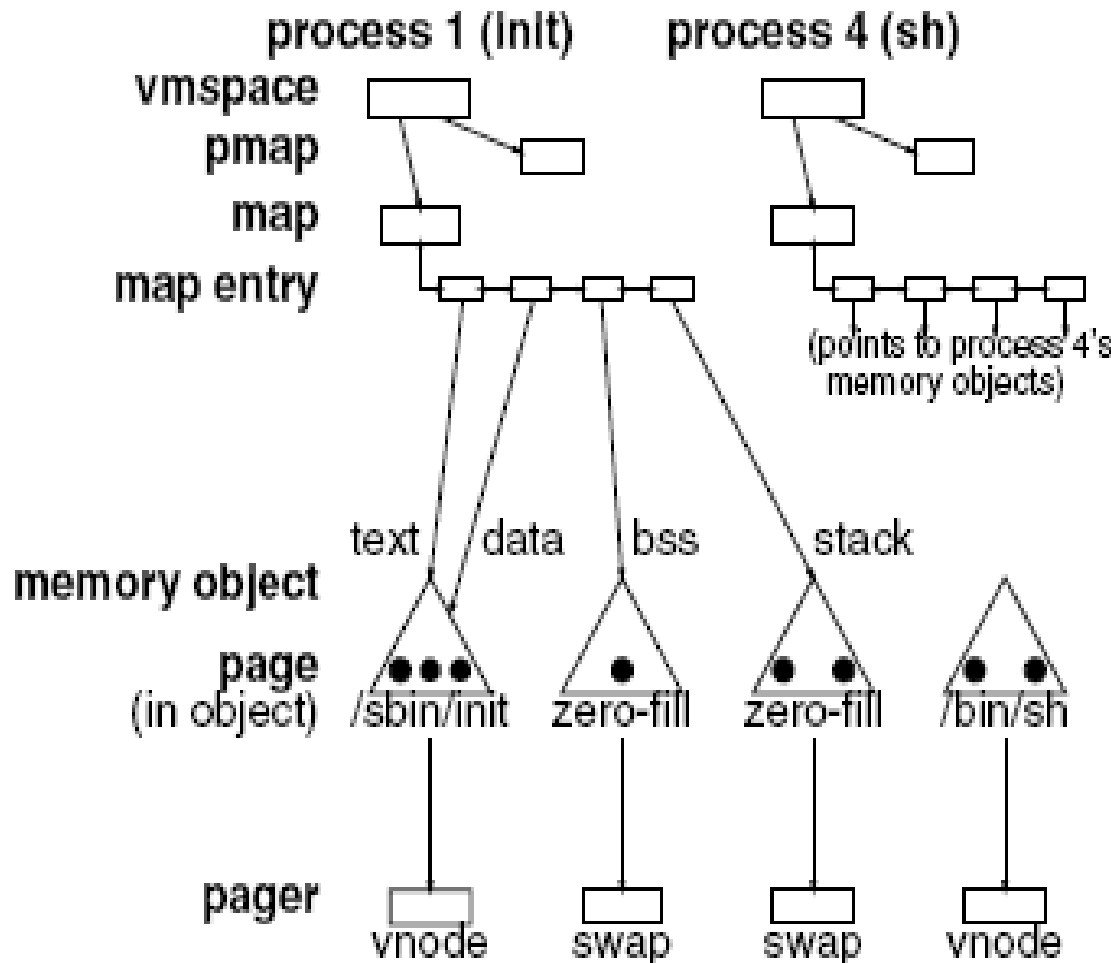
- Versions of BSD prior to 4.4BSD used the old BSD VAX virtual memory system that was tightly bound to the VAX architecture and lacked support for memory mapped files (mmap) and fine-grain data structure locking for multiprocessors
- The 4.4BSD virtual-memory system is based on the Mach 3.0

FreeBSD Memory Mgmt Con't..

- The 4.4BSD virtual-memory system features efficient support for sharing, a clean separation of machine-independent and machine-dependent features
- Process can map files anywhere in their address space. They can share parts of their address by doing a shared mapping of the same file
- Changes made by one process are visible in the address space of the other process, and also are written back to the file itself
- Process can also request private mapping of a file, which prevents any changes that they make from being visible to other processes mapping the file or being written back to the file itself.

FreeBSD Memory Mgmt Con't..

- FreeBSD describes its process address space by a vmpace, divided into logical sections called regions. Hardware-dependent portions are in the (pmap) module and vmap routines handle hardware-independent portions (map) and data structures (VM Objects)



FreeBSD Memory Mgmt Con't..

- FreeBSD makes use of several page queues "active", "inactive", "cached" and "free" to further refine the selection of pages to reuse as well as to determine when dirty pages must be flushed to their backing store
- The system make a distinction between clean pages which can theoretically be freed up at any time, and dirty pages which must first be written to their backing store before being reusable

FreeBSD Memory Mgmt Con't..

- If a page candidate has been found it is moved to the inactive queue if it is dirty, or the cache queue if it is clean. A separate algorithm based on the dirty-to-clean page ratio determines when dirty pages in the inactive queue must be flushed to disk. Once this is accomplished, the flushed pages are moved from the inactive queue to the cache queue
- Pages in the cache queue can still be reactivated by a VM fault at relatively low cost. However, pages in the cache queue are considered to be free able and will be reused in an LRU fashion when the system needs to allocate new memory

FreeBSD Memory Mgmt Con't..

- On FreeBSD, file system caching has been implemented as an integrated part of virtual memory system (unified buffer cache) to speed up some I/O intensive processes. It uses page cache pages as the memory for a buffer's data rather than memory from a separate pool
- A distributed shared memory (DSM) facility permits processes running at separate hosts on a network to share virtual memory in a transparent fashion, as if the processes were actually running on a single processor

LINUX Memory Management

- **Linux uses a memory descriptor to divide the process address space into logical sections called memory areas to describe process address space**
- **It divides machine-dependent layers from machine-independent layers at a much higher level in the software. The code to handle page faults is pretty much machine-dependent from the beginning of the fault handling. It can handle much of the paging code more quickly because there is less data abstraction in the code**
- **Unification of the buffer cache and the page cache. Dirty page cache pages are simply added in both the buffer and the page cache. The system does disk IO directly to and from the page cache page**

LINUX Memory Mgmt Con't..

- Linux also uses different linked lists of pages to facilitate an LRU-style algorithm. Linux divides physical memory into three "zones:" one for DMA (devices I/O) pages, one for normal pages, and one for dynamically allocated memory. Pages move between "hot," "cold," and "free" lists. Frequently accessed pages will be on the "hot" list. Free pages will be on the "cold" or "free" list
- In theory, paging eliminates the need for contiguous memory allocation, but DMA operation accesses the address bus directly while transferring data and may only write into certain addresses. That's why you need a *buddy*

Buddy System

- All page frames are grouped into 10 lists of blocks that contain groups of *1, 2, 4, 8, 16, 32, 64, 128, 256, and 512* contiguous page frames respectively
- If a small area is needed and only a larger area is available, the larger area is split into two halves (buddies), possibly repeatedly
- When a block is released, the kernel attempts to merge together pairs of free buddy

References

- **Memory management unit**
http://en.wikipedia.org/wiki/Memory_management_unit
- **Virtual memory**
http://en.wikipedia.org/wiki/Virtual_memory
- **Design elements of the FreeBSD VM system**
http://www.freebsd.org/doc/en_US.ISO8859-1/articles/vm-design/index.html
- **Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures**
<http://www.cs.nyu.edu/rgrimm/teaching/readings/portable-vm.pdf>
- **Outline of the Linux Memory Management System**
<http://home.earthlink.net/~jknappa/linux-mm/vmoutline.html>
- **Page replacement in Linux 2.4 memory management**
<http://www.surriel.com/lectures/linux24-vm.html>