**Nice coverage on memory mgmt..**

<http://landley.net/writing/memory-faq.txt>

**CPU Memory**

<http://www.akkadia.org/drepper/cpumemory.pdf>

The paper points out some of the security aspects in programming which I came across today.. I think its one the best papers...

**Caching** - <http://arstechnica.com/gadgets/2002/07/caching/1/>

**Bandwidth and Latency** - <http://arstechnica.com/features/2002/11/bandwidth-latency/>

**RAM Memory - Part 1, 2, 3**

<http://archive.arstechnica.com/paedia/r/ram_guide/ram_guide.part3-1.html>

**Chap 8-9 covers FS in depth..**https://ci6.googleusercontent.com/proxy/RnNZfQn2o2xpggJQqefCOervMbPIci5mujDPJnvl43kv6Rtxjyh5gHN_JKVzeU-aaGz3pePFgxfoAAtZJZNx8mveVTc-11j98EfuAJVcumUenA=s0-d-e1-ft#https://ssl.gstatic.com/ui/v1/icons/mail/images/cleardot.gif

**Professional Linux Architecture – Wolfgang Mauerer – Wiley Publication**

**VFS is well explained in Robert Love book**

**SRAM , DRAM**

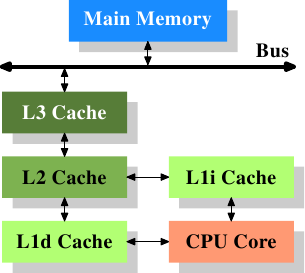
**SRAM – faster, expensive, 6 transistor**

**DRAM – slower, cheap , 1 transistor**

**DRAM Access**

**A program selects a memory location using a virtual address. The processor translates this into a physical address and finally the memory controller selects the RAM chip corresponding to that address. To select the individual memory cell on the RAM chip, parts of the physical address are passed on in the form of a number of address lines.**

**Cache**



**L1d is the level 1 data cache, L1i the level 1 instruction cache,**

The memory regions needed for code and data are pretty much independent of each other, which is why independent caches work better

 A cache line which has been written to and which has not been written back to main memory is said to be “dirty”. Once it is written the dirty flag is cleared.

To be able to load new data in a cache it is almost always first necessary to make room in the cache. An eviction from L1d pushes the cache line down into L2 (which uses the same cache line size). This of course means room has to be made in L2. This in turn might push the content into L3 and ultimately into main memory. Each eviction is progressively more expensive.

**Thread and Process:**

Process -

include a set of resources such as open files and pending signals, internal kernel data, processor state, a memory address space with one or more memory mappings, one or more threads of execution, and a data section containing global variables

Thread includes a unique program counter, process stack, and set of proces- sor registers

A process is an executing instance of an application, for example when we double click on MSWord, a process is launched. A thread on the other hand is only a path of execution within a process.   
A process can contain multiple threads. When you start Word, the operating system creates a process and begins executing the primary thread of that process.   
  
A thread can do anything a process can do. But since a process can consist of multiple threads, a thread could be considered a ‘lightweight’ process. Thus, the essential difference between a thread and a process is the work that each one is used to accomplish. Threads are used for small tasks, whereas processes are used for more ‘heavyweight’ tasks – basically the execution of applications.   
  
Another difference between a thread and a process is that threads within the same process share the same address space, whereas different processes do not. This allows threads to read from and write to the same data structures and variables, and also facilitates communication between threads.

**Mutex and Semaphore**

Both are locking mechanisms that ensure that only 1 thread executes thread at 1 time.

**Difference between Mutex and Semaphore -**Mutex has concept of ownership, only the thread which locked can unlock it. Semaphore have no such thing, One thread can decrement it and another can increment it.

**Semaphore Atomic increase and decrease –**

**P:**

when (semaphore > 0 ) [

Semaphore = Semaphore – 1;

]

**V:**

[ Semaphore = Semaphore + 1 ]

**Problems with Mutex –**

1. Reader Writer
2. Producer Consumer

**Typical causes of a segmentation fault:**

* Dereferencing NULL pointers – this is special-cased by memory management hardware
* Attempting to access a nonexistent memory address (outside process's address space)
* Attempting to access memory the program does not have rights to (such as kernel structures in process context)
* Attempting to write read-only memory (such as code segment)

**DMA vs Programmed I/O**

Direct memory access (DMA) is a feature of computerized systems that allows certain hardware subsystems to access main system memory independently of the central processing unit (CPU).

Without DMA, when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU initiates the transfer, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller when the operation is done.

**virtual memory provides three important capabilities.**

**Virtual memory is a feature of an operating system (OS) that allows a computer to compensate for shortages of physical memory by temporarily transferring pages of data from random access memory (RAM) to disk storage.**

(1) It uses main memory efficiently by treating it as a cache for an address space stored on disk, keeping only the active areas in main memory, and transferring data back and forth between disk and memory as needed.

(2) It simplifies memory management by providing each process with a uniform address space.

(3) It protects the address space of each process from corruption by other processes.

**Questions**

**1)**

result of this:

main()

{

fork();

fork();

fork();

printf("hello\n");

}

For Such Cases.We can Calculate the Output by one Formula   
  
(2^(n) -1) + 1

**Struct and Union**

With a union, you're only supposed to use one of the elements, because they're all stored at the same spot. This makes it useful when you want to store something that could be one of several types. A struct, on the other hand, has a separate memory location for each of its elements and they all can be used at once.

To give a concrete example of their use, I was working on a Scheme interpreter a little while ago and I was essentially overlaying the Scheme data types onto the C data types. This involved storing in a struct an enum indicating the type of value and a union to store that value.

union foo {

int a; // can't use both a and b at once

char b;

} foo;

struct bar {

int a; // can use both a and b simultaneously

char b;

} bar;

union foo x;

x.a = 3; // OK

x.b = 'c'; // NO! this affects the value of x.a!

struct bar y;

y.a = 3; // OK

y.b = 'c'; // OK

|  |  |
| --- | --- |
| Structure | Union |
| 1.The keyword  **struct** is used to define a structure | 1. The keyword union is used to define a union. |
| 2. When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of  sizes of its members. The smaller members may end with unused slack bytes. | 2. When a variable is associated with a union, the  compiler allocates the  memory by considering the size of the largest memory. So, size of union is equal to the size of largest member. |
| 3. Each member within a structure is assigned unique storage area of location. | 3. Memory allocated is shared by individual members of union. |
| 4. The address of each member will be in ascending order This indicates that memory for each member will start at different offset values. | 4. The address is same for all the members of a union. This indicates that every member begins at the same offset value. |
| 5 Altering the value of a member will not affect other members of the structure. | 5. Altering the value of any of the member will alter other member values. |
| 6. Individual member can be accessed at a time | 6. Only one member can be accessed at a time. |
| 7. Several members of a structure can initialize at once. | 7. Only the first member of a union can be initialized. |