Chapter 8

A Short Introduction to Virtual Memory

Virtual memory is an abstraction of physical memory. The purpose of virtual memory is generally to simplify application development and to let processes address more memory than what is actually physically present in the machine. We also don't want applications messing with the kernel or other applications' memory due to security.

In the x86 architecture, virtual memory can be accomplished in two ways: *segmentation* and *paging*. Paging is by far the most common and versatile technique, and we'll implement it the next chapter. Some use of segmentation is still necessary to allow for code to execute under different privilege levels.

Managing memory is a big part of what an operating system does. Paging and page frame allocation deals with that.

Segmentation and paging is described in the [33], chapter 3 and 4.

Virtual Memory Through Segmentation?

You could skip paging entirely and just use segmentation for virtual memory. Each user mode process would get its own segment, with base address and limit properly set up. This way no process can see the memory of another process. A problem with this is that the physical memory for a process needs to be contiguous (or at least it is very convenient if it is). Either we need to know in advance how much memory the program will require (unlikely), or we can move the memory segments to places where they can grow when the limit is reached (expensive, causes fragmentation - can result in "out of memory" even though enough memory is available). Paging solves both these problems.

It is interesting to note that in x86_64 (the 64-bit version of the x86 architecture), segmentation is almost completely removed.

Further Reading

- LWN.net has an article on virtual memory: http://lwn.net/Articles/253361/
- Gustavo Duarte has also written an article about virtual memory: http://duartes.org/gustavo/blog/post/memory-translation-and-segmentation