Unit 3
Cs 23 Operating system and shell programming

ADDRESS BINDING

A program resides on a disk as a binary executable file . The program must be brought into memory and placed within a process for it to be executed.Depending on the memory management in use,the process may be moved during its execution.The collection of processes on the disk that is waiting to be brought into memory for execution forms the input queue.
The binding instructions and data to memory addresses can be done at any step along the way:

1. Complie time: If we know at complie time where the process will reside in memory, then absolute code can be generated. For example , if we know a priori that a user process resides starting the location R, then the generated compiler code will start at the location and extend up from there. If at some time later, the starting location changes , then it will be necessary to recompile this code. This MS-DOS.COM format programs are absolute code bound at compile time.
2. Load Time: If it is not known at complie time wher the process will reside in memory, then the compiler must generate relocatable code. In this case, final binding is delayed until load time. If starting address changes, we need only to reload the user code to incorporate the changed value.
3. Execution time: If the process can be moved during its execution from one memory segment to another, the binding must be delayed until run time.

DYNAMIC LINKING AND LOADING

Dynamic Loading

The size of a process is limited to the size of physical memory.To obtain better memory space utilisation, we can use dynamic loading. With dynamic loading, a routine is not loaded until it is called. All routines are kept on disk in a relocatable load format. The main program is loaded into memory and executed. When a routine needs to call another routine, the calling routine first checks to see whether the other routine is loaded. If not, the relocatable linking loader is called to load the desired routine into memory and to update the program's address tables to reflect the change.Then control is passed to the newly loaded routine.
The advantage of dynamic loading is that an unused routine is never loaded. This method is particularly useful when large amounts of code are needed to handle infrequently occurring cases, such as error routines. In this case,

although the total program size may be ,arge, the portion that's used may be smaller.
Dynamic loading does not require special support from the operating system. It is the responsibility of the users to design their programs to take advantage of suc a method. Operating systems may help the programmer however, by providing library routines to implement dynamic loading.

Dynamic Linking

Some operating systems support only static linking in which system language libraries are treated like any other object module and are combined by the loader into the binary program image. The concept of dynamic linking is similar to,that of dynamic loading. Rather than loading being postponed until execution time, linking is postponed. This feature is usually used with system libraries such as language subroutine libraries. Without this facility, all programs on a system need to have a copy of their language library included in the executable image. This requirement wastes both disk space and main memory. With dynamic linking, a stub is included in the image for each library routine reference. This stub is a small piece of code that indicates how to locate the appropriate memory-resident library routine, or how to load the library if the routine is not already present.
Unlike dynamic loading, dynamic linking generally requires help from the operating system. If the processes in memory are protected from one another, the the operating system is the only entity that can check to see whether the needed routine is in another process' memory addresses.

LOGICAL AND PHYSICAL ADDRESS

As address generated by the CPU is commonly referred to as a logical address, whereas an address seen by the memory unit i.e. the one loaded into the memory address register of the memory- is commonly referred to as a physical address.
The compile time and load time address binding methods generate identical logical and physical addresses. However, the execution time address binding scheme results in differing logical and physical addresses. In this case, we usually refer to the logical address as a virtual address. We use logical address and virtual address interchangeably in this text. The set of all logical addresses generated by a program is a logical addr we space; the set of all physical address corespondimg to these logical addresses is a physical address space. Thus, in the execution time address bund mg schemes the logical and physical address spaces differ.
The run time mapping from virtual to physical addresses is done by a hardware device called the memory management unit. We can choose from among many different methods to accomplish such a mapping.

The base register is now called a relocation register. The value in the relocation register is added to every address generated by a user process at the time it is sent to memory. For example, if the base is at 14000, then an attempt by the user to address location 0 is dynamically relocated to location 14000; an access to location 346 is mapped to location 1436. The MS-DOS operating system running on the Intel 80x86 family of processors uses four relocation registers when loading and running processes.
The user program never sees the real physical addresses. The program can create a pointer to location 346, store it in memory, manipulate it, compare it, to other addresses- all as the number 346. Only when it is used as memory register is it relocated relative to the base register. The user program deals with logical addresses. The memory mapping hardware converts physical addresses into physical addresses. The final location of a referenced memory address is not determined until the referendum is made.

By Mamta.
Mob. No. 9430964676