CIS657 Principles of Operating Systems

Zhen Ren

1 Homework1

Ex.1.1 — What is the role of the magic number for binary executable files? Where is it stored?

The magic number is usually to help determine the program/text sized and data area, like a signature. And it usually stored at the beginning of the file.

Ex. 1.2 — List all segments that consist of an address space and explain each segment.

The address space more of less minimally consists of four different segments:
The text segment: contains the binary executable code for the program.
The static data segment: contains the static data structures that were created during the compilation time of the program
The dynamic data segment
The stack segment
The dynamic data segment and the stack segment grows and shrinks as the program executes the command like malloc() function are deleted. For example, MATRIX does not have any malloc() command therefore the dynamic segment will stay at size zero. The stack segment is needed for pushing and popping the parameters and the return address of caller when a subroutine is called

Ex. 1.3 — What are von-Neumann machines, and how are they related to the computers we use today?

The fundamental computing paradigm has not changed at all; all computers are von-Neumann machines, which means programs are stored in memory and the CPU executes programs by fetching instructions from memory.
Almost all the computers we use today are the von-Neumann machines.

Ex. 1.4 — Anyone who programmed in Unix environment using C or C++ (or other languages as well) surely experienced the famous error message segmentation fault (core dumped).” Explain what this error message mean and list possible reason for this error. What does segmentation” mean

in this context? What does ” mean in this context?

Segmentation faults are caused by a program trying to read or write an illegal memory location.
The first reason is the types of array-related segfaults error
The second reason in C is oversight in the use of pointers
The third will occur when a program attempts to operate on a memory location in a way that is not allowed.
The fourth is to occur when your program runs out of stack space.
Segmentation means different segments: a text segment for program instructions, a data segment for variables and arrays defined at compile time, a stack segment for temporary (or automatic) variables defined in subroutines and functions, and a heap segment for variables allocated during runtime by functions, such as malloc in C.
The core is a file name, which is called core. We usually use dump the content of memory at the time of the crash into a core file.

Ex. 1.5 — When a compiler generates the binary code for a source program written in a high-level programming language, it does not know where and how the binary code will be loaded by the operating system. Why not? In order to generate a binary code, the compiler, however, must make certain assumptions on where and how the binary code will be loaded in the main program by the operating system. What are the reasonable assumptions that are made by most of compilers in terms of where and how?

Actually, the compiler only knows its' virtual address rather than physical address.
Every time the compile starts compiling the binary code, the OS will allocate a memory space which starts 0x0, then map the program's start address of the memory space in physical memory. In the compiler, it just need the code address offset from the virtual address at 0x0. The OS will get the offset of the virtual address, then plus the program's start address to get the physical address

Ex. 1.6— What happens when we double-click a program icon? Or type a command at the prompt? Describe all the steps happening in OS until we see the application on the screen.

Following steps:
1. User double-clicks the program icon.
2. OS creates a Process Control Block (PCB) data structure for the program.
3. OS creates the address space for the program and load the program's address space according to the memory management scheme being used.
4. OS then put the PCB to the ready queue of the OS. The ready queue contains all processes in the computer system at the time that are ready to run by the CPU.

5. OS then performs a context-switching to let the program to run.

Ex. 1.7 — Finding the physical address from a virtual address in a contiguous memory allocation scheme.

In contiguous memory allocation scheme, we always load he program to main memory without any discontinuity. For example, if there is a program with size of 64 bytes total,7 the compiler will assume the program will be loaded from address 0 of the physical memory and loaded until the address 63, without any discontinuity in the physical memory address. Notice however, the example program must be loaded from the address 0x80; it cannot be loaded from physical address 0 because doing so will overwrite the OS. This means that the virtual address 0 of the program is now mapped onto the physical address of 0x80. Assume the program is 8 bytes. Then we will load the program from 0x80 to 0x87. The virtual address 7 is then mapped on to 0x87.

Ex. 1.8 — What is the round-robin scheduler?

Round-robin is a scheduling algorithm. Each process is given a certain amount of CPU time (about some milliseconds) and then inserted back in the ready queue to give a chance to the next process in the ready queue to run on the CPU. In this way, no one process can run forever on the CPU.