OS 书面作业一

张洋彬 191220169

- What are the two main functions of an operating system?

1、对计算机硬件功能的管理，包括处理器管理、存储管理和设备管理。处理器的管理和调度是对进程和线程的管理和调度，存储管理的主要任务是管理内存资源，设备管理的主要任务是管理各种外部设备，完成用户提出的 I/O 请求；提高设备的利用率等等。对信息资源的管理，现代计算机系统通常将程序和数据以文件形式储存在外存上，供用户使用，在外存上存在着很多文件。文件管理就是对用户文件和系统文件进行有效管理，实现文件共享和安全性控制，实现文件存储空间管理等等。

2、向用户程序提供一个友好的编程接口，即系统调用。

- On early computers, every byte of data read or written was handled by the CPU (i.e., there was no DMA). What implications does this have for multiprogramming?

I/O 操作较慢而 CPU 运算速度较快，I/O 操作时并不需要 CPU，如果每个字节的数据读写都由 CPU 来操控的话，多道程序处理系统就会将读写操作变为一个进程，会大大增加读写操作的时间，并会降低 CPU 的利用率。

- What is the difference between kernel and user mode? Explain how having two distinct modes aids in designing an operating system

内核态：cpu 可以访问内存的所有数据（硬件设备和所有内存空间），包括外围设备，例如硬盘，网卡，cpu 也可以将自己从一个程序切换到另一个程序。

用户态：只能受限的访问内存，且不允许访问外围设备，占用 cpu 的能力被剥夺，cpu 资源可以被其他程序获取

所有用户程序都是运行在用户态的，但是有时候程序确实需要做一些内核态的事情，例如从硬盘读取数据，或者从键盘获取输入等. 而唯一可以做这些事情的就是操作系统，所以此时程序就需要先操作系统请求以程序的名义来执行这些操作。

这时需要一个这样的机制：用户态程序切换到内核态，但是不能控制在内核态中执行的指令。这种机制叫系统调用，在 CPU 中的实现称之为陷阱指令(Trap Instruction)。

他们的工作流程如下：

用户态程序将一些数据值放在寄存器中，或者使用参数创建一个堆栈(stack frame)，以此表明需要操作系统提供的服务。系统调用完成后，操作系统会重置 CPU 为用户态并返回系统调用的结果。

当一个任务（进程）执行系统调用而陷入内核代码中执行时，我们就称进程处于内核运行态（或简称为内核态）。此时处理器处于特权级最高的（0 级）内核代码中执行。当进程处于内核态时，执行的内核代码会使用当前进程的内核栈。每个进程都有自己的内核栈。当进程在执行用户自己的代码时，则称其处于用户运行态（用户态）。即此时处理器在特权级最低的（3 级）用户代码中运行。当正在执行用户程序而突然被中断程序中断时，此时用户程序也可以象征性地称为处于进程的内核态。因为中断处理程序将使用当前进程的内核栈。这与处于内核态的进程的状态有些类似。

- What is a trap instruction? Explain its use in operating systems.

陷阱指令（int $0x80）： 通过陷阱处理机制使系统调用进入内核的系统调用处理程序。这个指令用于用户程序呼唤操作系统，陷入内核态，从而执行系统内核程序。

- What type of multiplexing (time, space, or both) can be used for sharing the following resources: CPU, memory, disk, network card, printer, keyboard, and display?

1、时间多路复用：CPU，网卡，打印机，键盘。

2、空间多路复用：内存，磁盘。

3、时空多路复用：显示器。

- To a programmer, a system call looks like any other call to a library procedure. Is it important that a programmer know which library procedures result in system calls? Under what circumstances and why?

通常操作系统如何管理计算资源对于程序员来说是不可见的，应用程序想要使用系统资源必须通过操作系统，从这个角度讲操作系统更像是 server，我们的应用程序是 client，client 只需要向 server 发出 request 然后得到 response，至于 server 如何处理请求并不要 client 关心。同样的道理，应用程序只需要进行系统调用，而操作系统通过系统调用来屏蔽了处理细节。

作为程序员应该意识到，我们的程序在运行时，CPU 不仅仅在执行我们的程序，当涉及到文件、网络、进程控制、线程控制、I/O 等时，单单依靠我们的代码是没有办法来完成这些操作的，这些只能依靠操作系统，对于程序员来说就是调用系统调用。当系统调用开始执行时，CPU 从用户模式切换到内存模式并开始运行操作系统的代码，即操作系统开始运行来完成上述用户程序请求。

实际上作为程序员我们使用的基本上是使用 C 标准库或 Win32 API 进行编程，而这些 API 又封装了系统调用(所谓封装，也就是这些 API 最终会调用到系统调用)，这也是为什么很多程序员根本没有意识到自己的程序在进行系统调用的原因。虽然系统调用在程序员眼里仅仅是一个普通的函数调用，但是深刻理解系统调用对于理解操作系统的运行方式来说是非常重要的。

- Explain how separation of policy and mechanism aids in building microkernel-based operating systems.

机制与策略分离原则有助于使内核保持短小和良好结构。通过将机制植入操作系统而策略留给上层软件，即使存在改变策略的需要，系统也可以保持不变。在这种机制下，应用问题的解决均可分成两部分，即“提供并实现确定的功能（机制），将机制作为系统的可信软件来实现”和“如何使用这些功能（策略），可在不可信的环境中定义策略。”，这种软件不但适应性好且易于开发。