COSC1112/1114: Operating Systems Principles

Tutorial 01 (week 02)

1. **List five services provided by an operating system. Explain how each provides convenience to the users. Explain also in which cases it would be impossible for user-level programs to provide these services.**

Answer:

i. **Program execution** - The operating system loads the contents (or sections) of a file into memory and begins its execution. A user-level program could not be trusted to properly allocate CPU time.

ii. **I/O operations** - Disks, tapes, serial lines, and other devices must be communicated with at a very low level. The user need only specify the device and the operation to perform on it, while the system converts that request into device or controller specific commands. User-level programs cannot be trusted to access only devices they should have access to and to access them only when they are otherwise unused.

iii. **File-system manipulation** - There are many details in file creation, deletion, allocation, and naming that users should not have to perform. Blocks of disk space are used by files and must be tracked. Deleting a file requires removing the name file information and freeing the allocated blocks. Protections must also be checked to assure proper file access. User programs could neither ensure adherence to protection methods nor be trusted to allocate only free blocks and deallocate blocks on file deletion.

iv. **Communications** - Message passing between systems requires messages to be turned into packets of information, sent to the network controller, transmitted across a communications medium, and reassembled by the destination system. Packet ordering and data correction must take place. Again, user programs might not coordinate access to the network device, or they might receive packets destined for other processes.

v. **Error detection** - Error detection occurs at both the hardware and software levels. At the hardware level, all data transfers must be inspected to ensure that data have not been corrupted in transit. All data on media must be checked to be sure they have not changed since they were written to the media. At the software level, media must be checked for data consistency; for instance, whether the number of allocated and unallocated blocks of storage match the total number on the device. There, errors are frequently process-independent (for instance, the corruption of data on a disk), so there must be a global program (the operating system) that handles all types of errors. Also, by having errors processed by the operating system, processes need not contain code to catch and correct all the errors possible on a system.

2. **What are the advantages and disadvantages of using the same system-call interface for manipulating both files and devices?**

Answer:

Each device can be accessed as though it was a file in the file system. Since most of the kernel deals with devices through this file interface, it is relatively easy to add a new device driver by implementing the hardware-specific code to support this abstract file interface. Therefore, this benefits the development of both user program code, which can be written to access devices and files in the same manner, and device-driver code, which can be written to support a well-defined API. The disadvantage with using the same interface is that it might be difficult to capture the functionality of certain devices within the context of the file access API, thereby resulting in either a loss of functionality or a loss of performance. Some of this could be overcome by the use of the ioctl operation that provides a general-purpose interface for processes to invoke operations on devices.

3. **What is the purpose of interrupts? What are the differences between a trap and an interrupt? Can traps be generated intentionally by a user program? If so, for what purpose?**

Answer:

An interrupt is a hardware-generated change of flow within the system. An interrupt handler is summoned to deal with the cause of the interrupt; control is then returned to the interrupted context and instruction. A trap is a software-generated interrupt. An interrupt can be used to signal the completion of an I/O to obviate the need for device polling. A trap can be used to call operating system routines or to catch arithmetic errors.

4. **When multiple interrupts from different devices appear at about the same time, a priority scheme could be used to determine the order in which the interrupts would be serviced. Discuss what issues need to be considered in assigning priorities to different interrupts.**

Answer:

A number of issues need to be considered in order to determine the priority scheme to be used to determine the order in which the interrupts need to be serviced. First, interrupts raised by devices should be given higher priority than traps generated by the user program; a device interrupt can therefore interrupt code used for handling system calls. Second, interrupts that control devices might be given higher priority than interrupts that simply perform tasks such as copying data served up a device to user/kernel buffers, since such tasks can always be delayed. Third, devices that have real-time constraints on when their data is handled should be given higher priority than other devices. Also, devices that do not have any form of buffering for its data would have to be assigned higher priority since the data could be available only for a short period of time.

5. **In a multiprogramming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems.**
   a) **What are two such problems?**
   b) **Can we ensure the same degree of security in a time-shared machine as in a dedicated machine? Explain your answer.**

Answer:

a) Stealing or copying one's programs or data; using system resources (CPU, memory, disk space, peripherals)without proper accounting.
b) Probably not, since any protection scheme devised by humans can inevitably be broken by a human, and the more complex the scheme, the more difficult it is to feel confident of its correct implementation.