CHAPTER 17

The Microsoft Windows 7 operating system is a 32/64-bit preemptive multitasking operating system for AMD K6/K7, Intel IA-32/IA64 and later microprocessors. The successor to Windows NT/2000, Windows 7, is also intended to replace the MS-DOS operating system. Key goals for the system are security, reliability, ease of use, Windows and POSIX application compatibility, high performance, extensibility, portability and international support. This chapter discusses the key goals for Windows 7, the layered architecture of the system that makes it so easy to use, the file system, networks, and the programming interface. Windows 7 serves as an excellent case study as an example operating system.

Exercises

17.1 Under what circumstances would one use the deferred procedure calls facility in Windows 7?
Answer:
Deferred procedure calls are used to postpone interrupt processing in situations where the processing of device interrupts can be broken into a critical portion that is used to unblock the device and a non-critical portion that can be scheduled later at a lower priority. The non-critical section of code is scheduled for later execution by queuing a deferred procedure call.

17.2 What is a handle, and how does a process obtain a handle?
Answer:
User-mode code can access kernel-mode objects by using a reference value called a handle. An object handle is thus an identifier (unique to a process) that allows access and manipulation of a system resource. When a user-mode process wants to use an object it calls the object manager's open method. A reference to the object is inserted in the process's object table and a handle is returned. Processes can obtain handles by creating an object, opening an existing object, receiving a duplicated handle from another process, or inheriting a handle from a parent process.

17.3 Describe the management scheme of the virtual memory manager. How does the VM manager improve performance?
Answer:
The VM manager uses a page-based management scheme. Pages of data allocated to a process that are not in physical memory are either stored in paging files on disk or mapped to a regular file on a local or remote file system. To improve performance of this scheme, a privileged process is allowed to lock selected pages in physical memory preventing those pages from being paged out. Furthermore, since when a page is used, adjacent pages will likely be used in the near future, adjacent pages are prefetched to reduce the total number of page faults.

17.4 Describe a useful application of the no-access page facility provided in Windows 7.
Answer:
When a process accesses a no-access page, an exception is raised. This feature is used to check whether a faulty program accesses beyond the end of an array. The array needs to be allocated in a manner such that it appears at the end of a page, so that buffer overruns would cause exceptions.

17.5 Describe the three techniques used for communicating data in a local procedure call. What settings are most conducive to the application of the different message-passing techniques?
Answer:
Data is communicated using one of the following three facilities: 1) messages are simply copied from one process to the other, 2) a shared memory segment is created and messages simply contain a pointer into the

shared memory segment, thereby avoiding copies between processes, 3) a process directly writes into the other process's virtual space.

17.6 What manages caching in Windows 7? How is the cache managed?
Answer:
In contrast to other operating systems where caching is done by the file system, Windows 7 provides a centralized cache manager which works closely with the VM manager to provide caching services for all components under control of the I/O manager. The size of the cache changes dynamically depending upon the free memory available in the system. The cache manager maps files into the upper half of the system cache address space. This cache is divided into blocks that can each hold a memory-mapped region of a file.

17.7 How does the NTFS directory structure differ from the directory structure used in UNIX operating systems?
Answer:
The NTFS namespace is organized as a hierarchy of directories where each directory uses a B+ tree data structure to store an index of the filenames in that directory. The index root of a directory contains the top level of the B+ tree. Each entry in the directory contains the name and file reference of the file as well as the update timestamp and file size. The UNIX operating system simply stores a table of entries mapping names to i-node numbers in a directory file. Lookups and updates require a linear scan of the directory structure in UNIX systems.

17.8 What is a process, and how is it managed in Windows 7?
Answer:
A process is an executing instance of an application containing one or more threads. Threads are the units of code that are scheduled by the operating system. A process is started when some other process calls the CreateProcess routine, which loads any dynamic link libraries used by the process, resulting in a primary thread. Additional threads can also be created. Each thread is created with its own stack with a wrapper function providing thread synchronization.

17.9 What is the fiber abstraction provided by Windows 7? How does it differ from the threads abstraction?
Answer:
A fiber is a sequential stream of execution within a process. A process can have multiple fibers in it, but unlike threads, only one fiber at a time is permitted to execute. The fiber mechanism is used to support legacy applications written for a fiber-execution model.

17.10 How does user-mode scheduling (UMS) in Windows 7 differ from fibers? What are some trade-offs between fibers and UMS?
Answer:
Fibers are only UTs, and the kernel has no knowledge of their existence. They do not have their own TEBs and thus cannot reliably run Windows 7 APIs. Because a UT shares the KT of the thread it executes on, it must be careful not to change the state of the KT, such as by using impersonation or canceling I/O. Fibers lose control of the CPU whenever the borrowed KT blocks in the kernel. With UMS, control of the CPU is always returned to the user-mode scheduler.

17.11 UMS considers a thread to have two parts, a UT and a KT. How might it be useful to allow UTs to continue executing in parallel with their KTs?
Answer:
If a UT could continue executing even though its corresponding KT was running in the kernel, it would allow the system services performed by KTs to be asynchronous—as I/O already is in Windows 7. However, the

programming language or run-time would have to provide a way of synchronizing with the results of the system service being performed by each asynchronous KT.

17.12 What is the performance trade-off of allowing KTs and UTs to execute on different processors?
Answer:
If KTs and UTs ran on different processors, the application would have more concurrency and better cache locality—since the KT would not be poisoning the cache used by the UT. However, there would be more latency, as the CPU that ran the KT might be busy with other work when the service request arrived. In addition, there would be higher overheads due to queuing and synchronization costs.

17.13 Why does the self-map occupy large amounts of virtual address space but no additional virtual memory?
Answer:
The virtual memory pages used for the page table have to be allocated whether or not the pages are mapped into the kernel virtual address space. The only cost is the loss of use of one of the PDE entries in the highest-level page directory.

17.14 How does the self-map make it easy for the VM manager to move the page table pages to and from disk? Where are the page-table pages kept on disk?
Answer:
The self-map places the page-table pages into the kernel's virtual address space. The pages are part of kernel virtual memory, so the VM manager can page them in and out of memory just as it does other virtual memory. Because the page-table pages are not backed by a memory-mapped file, they are kept in the page file when not in memory.

17.15 When a Windows 7 system hibernates, the system is powered off. Suppose you changed the CPU or the amount of RAM on a hibernating system. Do you think that would work? Why or why not?
Answer:
It would likely cause the system to crash. Information about the CPU and RAM (as well as many other specifics about the hardware) are captured by the kernel when it boots. When the system resumes from hibernation, the data structures that describe this information will be inconsistent with the changed hardware.

17.16 Give an example showing how the use of a suspend count is helpful in suspending and resuming threads in Windows 7.
Answer:
Suppose an operation suspends a thread, examines some state while the thread is suspended, and then resumes the thread. If two different threads attempt the operation on the same thread, the use of suspend counts will keep the target thread from prematurely resuming.