CHAPTER 02

1. LIST AND EXPLAIN SYSTEM PROCESSES PRESENT IN THE WINDOWS OPERATING SYSTEM

On the Windows subsystem, there are several processes that play different roles and functions. Here are the main processes and their functions:

1. **Idle process:** This process is responsible for ***accounting idle CPU time.*** It contains ***one thread per CPU.***
2. **System process:** This process includes the majority of the ***kernel-mode system threads.***
3. **Session manager (Smss.exe):** The session manager is ***responsible for starting and managing sessions*** on the system.
4. **Local session manager (Lsm.exe):** The local session manager ***handles the creation and deletion of sessions.***
5. **Windows subsystem (Csrss.exe):** The Windows subsystem is a required component for any Windows system. It ***handles windowing and display I/O functions for other subsystems.***
6. **Session 0 initialization (Wininit.exe):** This process initializes Session 0, which is a ***non-interactive session used for system services.***
7. **Logon process (Winlogon.exe):** The logon process ***handles user logon and logoff operations.***
8. **Service control manager (Services.exe):** The service control manager is ***responsible for managing system services.*** It creates child service processes like ***Svchost.exe.***
9. **Local security authentication server (Lsass.exe):** The local security authentication server ***handles security-related operations,*** such as ***authentication*** and ***security policy enforcement.***

These ***processes interact with each other*** and with the ***kernel-mode components*** to ***provide necessary functionality and services*** to the Windows operating system. In addition to these system processes, there are also many other system processes that are responsible for specific tasks, such as ***managing device drivers, networking, and power management.***

2. EXPLAIN WINDOWS SUB SYSTEM.

Windows Subsystem is a critical ***architectural component*** in the Windows operating system, designed to support ***diverse application environments seamlessly.*** The decision to centralize ***windowing and display I/O functions*** within the Windows subsystem was driven by the need to ***prevent redundancy and optimize performance.*** This design choice ensures that critical ***processes related to user interface and display*** are efficiently managed within a ***unified subsystem.***

Key aspects of the Windows Subsystem include:

1. **Design Decision:**
    - Windows is intentionally crafted to ***accommodate multiple independent environment subsystems.***

- To enhance system efficiency and eliminate redundancy, the decision was made to combine windowing and display I/O functions within the Windows subsystem.

2. **Component and Critical Process:**
   - The ***Windows subsystem is not just an optional feature but a mandatory component for all Windows systems,*** even on servers without active user sessions.
   - Marked as a critical process, ***any unexpected exit of the Windows subsystem*** can lead to system crashes, underscoring its pivotal role in ***system stability.***
3. **Major Components:**
   - **Environment Subsystem Process:**
     - Loads three essential DLLs supporting functions like ***process and thread management, 16-bit VDM processes,*** and ***side-by-side support.***
   - **Kernel-Mode Device Driver (Win32k.sys):**
     - Encompasses the ***window manager, GDI for graphics output devices,*** and ***DirectX support wrappers.***
   - **Console Host Process (Conhost.exe):**
     - ***Facilitates console applications.***
   - **Subsystem DLLs:**
     - ***Translate documented Windows API functions into kernel-mode system calls.***
   - **Graphics Device Drivers:**
     - Support hardware-dependent ***graphics display drivers, printer drivers,*** and ***video miniport drivers.***
4. **Application Interaction:**
   - Applications interact with the Windows Subsystem through ***standard USER functions,*** creating ***user interface controls like windows and buttons.***
   - The ***window manager communicates these requests to the Graphics Device Interface (GDI),*** which, in turn, forwards them to graphics device drivers for proper formatting on the display device.
5. **Graphics Device Interface (GDI):**
   - GDI provides standard two-dimensional functions ***enabling applications to communicate with graphics devices.***

- It acts as a mediator between applications and graphics devices.

In essence, the ***Windows Subsystem is a foundational and indispensable feature of Windows,*** offering ***users and developers flexibility, cost-effectiveness,*** and ***enhanced productivity*** by providing a unified and optimized environment for diverse applications.

3. WHAT IS AN EXECUTIVE? EXPLAIN IT / EXPLAIN IN DETAIL WINDOWS EXECUTIVE SYSTEM.

The Windows Executive System is a ***pivotal layer within the Windows operating system, situated above the kernel (Ntoskrnl.exe),*** encompassing a diverse ***set of functions and components*** that are integral to the ***management and operation of the system.***

1. Types of Functions in the Executive:

- **System Services (Exported Functions):**
  - Callable from ***user mode via the Windows API*** or ***other environment subsystem APIs.***
- **Device Driver Functions:**
  - Invoked through the ***DeviceIoControl*** function, ***providing a generalized interface from user mode to kernel mode.***

2. Kernel Mode Functions:

- **Callable Only from Kernel Mode:**
  - ***Documented and undocumented functions in the Windows Driver Kit (WDK).***

3. Major Components of the Executive:

- **Configuration Manager:** Manages the ***system registry.***
- **Process Manager:** Controls processes and threads.
- **Security Reference Monitor:** Enforces security policies.
- **I/O Manager:** Handles I/O operations.
- **Plug and Play Manager:** Manages Plug and Play events.
- **Power Manager:** Coordinates power events.
- **Cache Manager:** ***Optimizes file-based I/O performance.***
- **Memory Manager:** Manages virtual memory.
- **Logical Prefetcher and SuperFetch:** Optimize system start-up.

4. Support Functions:

- **Object Manager:** ***Creates and manages system resources.***

- **Advanced LPC Facility (ALPC):** Facilitates ***interprocess communication.***
- **Common Run-Time Library Functions:** Provides common functions.
- **Executive Support Routines:** Handles ***memory allocation and synchronization objects.***

Here is an example of how the Windows Executive System works:

1. A user launches an application.
2. The Process Manager creates a new process for the application.
3. The Memory Manager allocates memory to the process.
4. The Security Reference Monitor checks to ensure that the user is authorized to launch the application.
5. The Object Manager creates an object for the application.
6. The ***I/O Manager loads the application's executable file into memory.***
7. The ***Process Manager schedules the process*** for execution.
8. The CPU begins executing the application's code.
9. The application uses the services provided by the Executive to access resources and perform its tasks.
10. When the application finishes executing, the ***Process Manager terminates the process and deallocates the resources*** that were allocated to it.

In essence, the Windows Executive System orchestrates ***a seamless interplay of functions***, ***efficient system management*** in both user and kernel modes. It forms the ***backbone of the Windows OS,*** providing the necessary infrastructure for diverse operations.

4. WHAT IS NTDLL.DLL? EXPLAIN IN BRIEF.

NTDLL.DLL is a vital ***system support library*** primarily used by subsystem DLLs. It serves two main functions:

1. **Interface to Windows Executive System Services:**
    - ***Provides more than 400 functions,*** such as NtCreateFile and NtSetEvent, which ***act as a bridge between user mode and the Windows Executive system services.***
    - Contains ***entry points with the same name as these functions.*** These ***entry points trigger a transition into kernel mode*** to invoke the ***system service dispatcher,*** which ***then calls the actual kernel-mode system service in Ntoskrnl.exe.***
2. **Internal Support Functions:**

- Houses various support functions for ***subsystems and native images,*** including the ***image loader, heap manager, communication functions for Windows subsystem processes, run-time library routines, support for user-mode debugging, event tracing,*** and ***user-mode asynchronous procedure call dispatcher*** and ***exception dispatcher.***
- Also includes a ***limited subset of C Run-Time routines*** related to ***string and standard libraries.***

Ntdll.dll acts as a ***crucial intermediary for subsystems,*** housing functions that facilitate communication with the Windows Executive system services and providing internal support for various system operations.

Ntdll.dll is a special type of DLL that is required for Windows to run. It ***contains the Windows Native API,*** which is ***used by user-mode components*** of the operating system. Other DLLs are optional and may contain APIs for specific purposes.

Here are some of the key functions that are exported by Ntdll.dll:

- **NtCreateProcess:** Creates a new process.
- **NtTerminateProcess:** Terminates a process.
- **NtOpenFile:** Opens a file.
- **NtReadFile:** Reads data from a file.
- **NtWriteFile:** Writes data to a file.
- **NtMapViewOfSection:** ***Maps a view of a memory section*** into the address space of the current process.
- **NtUnmapViewOfSection:** ***Unmaps a view of a memory section*** from the address space of the current process.

5. WITH NEAT DIAGRAM EXPLAIN WINDOWS SYSTEM ARCHITECTURE/ EXPLAIN IN DETAILS VARIOUS COMPONENTS PRESENT IN WINDOWS.

The Windows system architecture is a ***layered design*** that consists of two main components: ***user mode and kernel mode.***

User mode is the environment in which applications run. Applications in user mode have ***limited access to system resources*** and ***cannot directly access the hardware.***

Kernel mode is the environment in which the operating system kernel runs. The kernel has ***full access to system resources*** and can ***directly access the hardware.***

The kernel is ***responsible for managing the system's hardware and resources,*** such as the ***CPU, memory, and disk storage.*** The kernel also provides services to applications, such as the ability to read and write to files, and to communicate with other computers.

Applications in user mode ***communicate with the kernel through system calls. System calls are functions*** that allow applications to ***request services from the kernel.***

The following diagram shows a simplified overview of the Windows system architecture:

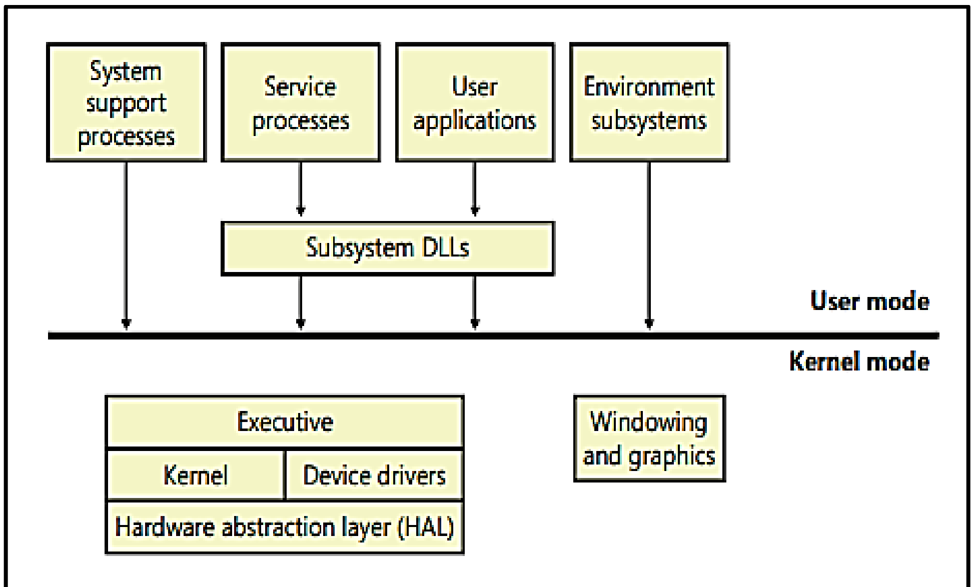

FIGURE 2-1 Simplified Windows architecture

The diagram shows the two main components of the Windows system architecture: user mode and kernel mode. It also shows the different layers of the operating system, and the ***flow of control*** between the layers.

Here is a brief explanation of each layer:

Kernel Mode:

- **Hardware Abstraction Layer (HAL):**

  The HAL provides an abstraction layer between the kernel and the hardware. This ***allows the kernel to be ported to different hardware platforms*** without having to be modified.

- **Kernel:**

  The kernel is the core of the operating system. It is responsible for managing the system's hardware and resources, and providing low level services to applications, such as managing CPU, memory, disk storage etc.

- **Device Drivers:**
  Includes ***hardware and nonhardware device drivers,*** responsible for ***translating user I/O function calls to specific hardware device I/O requests.***

- **Windowing and Graphics System:**
  Implements GUI functions, such as ***window management, user interface controls, and graphics rendering.***

- **Executive:**

  The Executive represents the upper layer in the Windows operating system kernel architecture (Ntoskrnl.exe). It includes exported and callable functions, device driver functions, kernel mode functions, and internal functions. Essential components of the Executive include the Configuration Manager, Process Manager, Security Reference Monitor, I/O Manager, PnP Manager, Power Manager, Cache Manager, Memory Manager, and Logical Prefetcher and SuperFetch.

User Mode:

- **User mode subsystems:**

  User mode subsystems provide a ***layer of abstraction between applications and the kernel.*** This allows applications to be developed without having to know the details of the underlying hardware and kernel.

- **Environment Subsystems and Subsystem DLLs:**

  Environment subsystems provide the environment within which user-mode applications execute. This includes the Win32 console environment, POSIX, the OS/2 environment, and the Windows API.

- **Applications:**

  Applications are programs that run on the operating system. They can be used for a variety of purposes, such as productivity, gaming, and entertainment.

The Windows system architecture is a complex design, but it is essential for the operation of the operating system. The ***layered design allows for flexibility and scalability,*** and it makes it ***easier to develop and maintain applications.***

6. EXPLAIN IN BRIEF SCALABILITY AND PORTABILITY OF WINDOWS OPERATING SYSTEM.

Scalability and portability are two important characteristics of the Windows operating system.

Scalability

Scalability is the ***ability of an operating system to handle increasing workloads without sacrificing performance.*** The Windows operating system is scalable because it can be ***configured to run on a wide range of hardware platforms,*** from small laptops to large enterprise servers. It can also be scaled up by adding more hardware resources, such as ***CPU cores and memory.***

There are two types of scalabilities: vertical scalability and horizontal scalability.

- **Vertical Scalability:** Vertical scalability, also known as scaling up, refers to the ability of an operating system or a system to handle increased workload or resource demands by improving the capabilities of a single system. This can involve upgrading hardware components, such as adding more powerful processors, increasing memory capacity, or enhancing storage capabilities. With vertical scalability, the focus is on maximizing the performance of a single system to meet growing demands. It is often associated with improving the performance and capacity of a single server or machine. For example, a ***Windows server*** can be vertically scaled by adding more ***CPU cores, memory,*** and ***disk storage.***

- **Horizontal Scalability**: Horizontal scalability, also known as scaling out, involves expanding the overall capacity and performance of a system by adding more machines or nodes to the system. Instead of enhancing the capabilities of a single system, horizontal scalability distributes the workload across multiple systems, allowing them to work together in parallel. This can involve adding more servers, machines, or nodes to a network or system architecture. Horizontal scalability is often associated with distributed systems, where tasks are divided among multiple machines to handle increasing workload or accommodate more users. For example, a Windows web application can be horizontally scaled by adding more web servers to the cluster.

The ***Windows operating system is both vertically and horizontally scalable.*** This makes it a good choice for businesses of all sizes, from small businesses to large enterprises.

Portability

Portability is the ability of an operating system to run on different hardware platforms. The Windows operating system is portable because it can be run on a wide range of hardware platforms, ***from x86*** and ***x64 processors*** to ***ARM processors.***

The ***portability of the Windows operating system is made possible by the use of a hardware abstraction layer (HAL).*** The HAL provides a layer of abstraction between the kernel and the hardware. This allows the kernel to be ported to different hardware platforms without having to be modified.

The portability of the Windows operating system makes it a good choice for businesses that need to deploy applications on a variety of hardware platforms. For example, a Windows application can be deployed on both desktop computers and laptops, as well as on servers and mobile devices.