**Memory management**

Android – the system focuses on keeping the memory as full as possible, so that it's easy to switch back to closed applications. This is possible thanks to the Zygote process, which instead of creating new processes from scratch uses already established essential system libraries and resources. Android also uses two things for low memory management: kernel swap daemon and low-memory killer. Kernel swap daemon deletes empty pages, which are backed by storage and have not been modified. If a process tries to access the page it gets copied from the storage to RAM. Low-memory killer on the other hand starts when kernel swap daemon can't free enough space, and it starts shutting down processes starting with the background ones up to the ones actually doing work for the OS.

https://developer.android.com/topic/performance/memory-management

IOS – iOS maintains a secure sandboxing mechanism that isolates application processes, ensuring stability and security. It prevents applications from accessing one another's memory space. IOS also uses Automatic Reference Counting (ARC), which allocates a chunk of memory to store information about each new instance of a class of a program. This memory holds information about the type of the instance, together with the values of any stored properties associated with that instance. When an instance is no longer needed, ARC frees up the memory used by that instance so that the memory can be used for other purposes instead. This ensures that class instances don't take up space in memory when they're no longer needed.

https://docs.swift.org/swift-book/documentation/the-swift-programming-language/automaticreferencecounting/

MacOs - when MacOs starts it divides RAM into two partitions – one reserved for the System, and the rest for running applications (each new application has its own partition within it). It also has a virtual memory system, which provides up to 4 gigabytes of addressable space per 32-bit process. The space is kept in form of logical addresses, which can be accessed by the processor and its memory management unit in order to translate it into a hardware memory access. The pages are generally always available, however there is also a special handler which stops the currently executing code, locates a free page of physical memory, loads the page containing the needed data from disk, updates the page table, and then returns control to the program's code, which can then access the memory address normally.

https://developer.apple.com/library/archive/documentation/Performance/Conceptual/ManagingMemory/Articles/AboutMemory.html

https://developer.apple.com/library/archive/documentation/mac/pdf/Memory/Intro_to_Mem_Mgmt.pdf

Windows – on top using virtual memory similar to the one of MacOS, Windows also uses a designated Kernel-Mode memory manager. It manages the allocation and deallocation of memory virtually and dynamically and supports the concepts of memory-mapped files, shared memory, and copy-on-write. It also uses memory compression, which is a feature the user can disable and enable. It dynamically reduces the size of data before writing it to RAM, which allows the computer to store more files in physical memory than it normally would, reducing the need for paging which slows down the computer.

https://www.makeuseof.com/windows-memory-compression-guide/

https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/_kernel/#memory-allocation-and-buffer-management

**Process management**

Android – process management in Android uses scheduling, memory allocation, and resource management. If it comes to task scheduling Android uses Linux-based kernel, which utilizes Completely Fair Scheduler algorithm to allocate CPU time to different processes, which helps in a balanced distribution of resources to the running processes. To optimize power consumption Android also introduced a feature called Background Task Manager, which categorizes processes based on their importance and ensures the ones with higher priority will receive better treatment. Lastly when it comes to resource management the processes and applications are allocated into the CPU cores, which optimizes the hardware resource utilization.

IOS – State preservation and restoration allow applications to save their current state and later restore that state upon reopening. optimizes background execution by managing processes that run in the background. This mechanism allows certain types of tasks to continue running even when the app is not actively in use.

MacOs – macOS supports so called Quality of Service classes. These are: user-interactive (work is virtually instantaneous), user initiated (takes a few seconds or less), utility (takes a few seconds to few minutes) and background (minutes to hours). These classes can be chosen for processes by app developers. The more OS-oriented stuff is rather similar to Windows.

https://developer.apple.com/library/archive/documentation/Performance/Conceptual/power_efficiency_guidelines_osx/PrioritizeWorkAtTheTaskLevel.html

Windows –Windows divides processes into three categories – running (executing instructions of the CPU), ready (awaiting execution from the CPU) and blocked (waiting for an operation to occur – until then they are removed from the CPU scheduler). The scheduler itself also has a couple of priority levels. If processes have the same priority levels, the way of determining which ones goes ways is done in a round-robin fashion. Windows also supports interprocess communications, which facilitates the division of labor among several specialized processes.

https://learn.microsoft.com/en-us/windows/win32/ipc/interprocess-communications

https://learn.microsoft.com/en-us/windows/win32/procthread/scheduling-priorities

**File management**

Android – there are several options if it comes to saving data on an Android device. There is shared storage, which can be accessed by all applications, there are preferences which are private data in key-value pairs, databases, which store structured data in a private database using the Room persistence library, and app-specific storages, the contents of which are visible solely for the apps themselves.

IOS – When installing an app on an IOS system, it generally cannot access or create files outside of its container directories, which are a part of a so-called sandbox. There is a bundle container and a data container. The bundle container holds the app's bundle. The data container is divided into further folders which hold the data of the app and the user. Apps might also request access into additional containers, such as the one for iCloud.

MacOs – in MacOS the file system is divided into domains – local domain, system domain, network domain and user domain. The user domain contains resources specific to the users who log in to the system and each user has access to and control over the files in their own home directory. The local domain contains resources such as apps that are local to the current computer and shared among all users of that computer. The local domain does not correspond to a single physical directory, but instead consists of several directories on the local boot (and root) volume. This domain is typically managed by the system, but users with administrative privileges may add, remove, or modify items in this domain. The network domain contains resources such as apps and documents that are shared among all users of a local area network. Items in this domain are typically located on network file servers and are under the control of a network administrator. The system domain contains the system software installed by Apple. The resources in the system domain are required by the system to run. Users cannot add, remove, or alter items in this domain.

Windows – Files in Windows are placed in a hierarchical structure. At the bottom there are regular files, a bit higher are directories, which are a hierarchical collection of directories and files, and at the top there are volumes which are also collections of directories and files. This allows for organization and management of the files.

https://developer.apple.com/library/archive/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html