**1.Perbedaan monolithic kernel, microkernel, dan layered architecture**

1. **Monolithic Kernel**:
   * **Deskripsi**: Semua layanan kernel seperti driver, scheduler, file system, dan networking,dijalankan dalam satu ruang memori yang sama Kernel bersifat monolitik seperti satu blok besar dan semua modul saling terintegrasi secara ketat.
   * **Kelebihan**: Performa tinggi karena komunikasi antar-komponen cepat (tidak ada overhead switching mode). Mudah dikembangkan untuk fitur kompleks.
   * **Kekurangan**: Kurang modular; jika satu modul crash (misalnya driver buruk), seluruh kernel bisa gagal. Sulit di-debug dan kurang aman karena semua kode punya akses penuh ke hardware.
   * **Cara Kerja**: Semua panggilan sistem (system calls) ditangani langsung di kernel tanpa banyak abstraksi.
2. **Microkernel**:
   * **Deskripsi**: Kernel hanya menangani fungsi dasar minimal, seperti inter-process communication (IPC), basic scheduling, dan manajemen memori sederhana. Layanan lain (driver, file server, dll.) dijalankan sebagai proses terpisah di user space.
   * **Kelebihan**: Sangat modular dan aman; kegagalan satu layanan tidak merusak kernel utama. Mudah dikustomisasi dan portabel ke hardware berbeda. Dukung reliabilitas tinggi untuk sistem real-time.
   * **Kekurangan**: Overhead performa lebih tinggi karena komunikasi antar-layanan melalui message passing (IPC), yang lebih lambat daripada panggilan langsung.
   * **Cara Kerja**: User applications berkomunikasi dengan layanan kernel melalui pesan, mirip client-server model.
3. **Layered Architecture** (atau Hierarchical/Layered Kernel):
   * **Deskripsi**: Kernel dibagi menjadi lapisan-lapisan hierarkis, di mana setiap lapisan bergantung pada lapisan di bawahnya. Lapisan bawah menangani hardware dasar (misalnya CPU dan I/O), sementara lapisan atas menangani abstraksi tinggi (seperti user interface atau aplikasi).
   * **Kelebihan**: Struktur jelas dan mudah dipahami; memudahkan pengembangan dengan memisahkan tanggung jawab (seperti model OSI untuk jaringan). Bagus untuk sistem besar yang kompleks.
   * **Kekurangan**: Performa bisa rendah karena data harus melewati banyak lapisan (overhead vertikal). Kurang fleksibel untuk perubahan, dan kegagalan di lapisan bawah memengaruhi semua lapisan atas.
   * **Cara Kerja**: Mirip stack; misalnya, lapisan 0 (hardware) → lapisan 1 (kernel dasar) → lapisan n (aplikasi). Setiap lapisan hanya berinteraksi dengan tetangganya.

**2.Contoh OS Nyata yang Menggunakan Masing-Masing Model**

1. **Monolithic Kernel**:
   * **Linux**: Kernel utama untuk distribusi seperti Ubuntu, Fedora, dan Android. Semua driver dan modul dimuat ke kernel space, meskipun mendukung modularitas melalui loadable kernel modules (LKM).
   * **FreeBSD/Unix Variants**: Seperti OpenBSD atau NetBSD, yang mengikuti tradisi Unix monolitik.
   * **Windows NT Kernel** (hybrid, tapi dasarnya monolithic): Digunakan di Windows 10/11, di mana sebagian besar layanan di kernel space, meskipun ada elemen modular.
2. **Microkernel**:
   * **Minix**: Dikembangkan oleh Andrew Tanenbaum untuk tujuan pendidikan; versi modern (Minix 3) digunakan di Intel Management Engine untuk keamanan.
   * **QNX**: OS real-time untuk embedded systems, seperti di mobil (BlackBerry QNX untuk infotainment) dan perangkat medis. Sangat andal untuk aplikasi kritis.
   * **L4 Microkernel Family**: Dasar untuk seL4 (verifikasi formal untuk keamanan), digunakan di sistem militer dan mobile (seperti di Tizen RT untuk IoT).
3. **Layered Architecture**:
   * **THE OS** (The Operating System): Dikembangkan oleh Edsger Dijkstra pada 1960-an di Belanda; contoh klasik layered dengan 6 lapisan untuk multiprogramming.
   * **Multics**: Prekursor Unix, dikembangkan oleh MIT/Bell Labs pada 1960-an; menggunakan lapisan untuk keamanan dan sharing resource (pengaruh besar pada Unix).
   * **Contoh Modern**: Beberapa sistem embedded atau legacy seperti IBM OS/360 (mainframe), meskipun jarang murni layered hari ini. Windows juga punya elemen layered di driver stack-nya, tapi bukan murni.

1.**Analisis: Model Mana yang Paling Relevan untuk Sistem Modern?**

Untuk sistem modern (seperti cloud computing, mobile, IoT, dan AI-driven devices), **monolithic kernel** (khususnya varian modular seperti Linux) adalah yang paling relevan dan dominan. Alasannya:

* **Performa dan Skalabilitas**: Sistem modern menuntut kecepatan tinggi untuk big data, virtualization (misalnya Docker/Kubernetes di Linux), dan edge computing. Monolithic menawarkan latency rendah tanpa overhead IPC yang signifikan dari microkernel. Linux menguasai ~90% server cloud (AWS, Google Cloud) dan hampir semua supercomputer.
* **Fleksibilitas dengan Modularitas**: Meskipun monolitik, Linux mendukung hot-swappable modules, yang mengatasi kekurangan modularitas tanpa mengorbankan performa. Ini membuatnya adaptif untuk hardware beragam (dari smartphone hingga HPC).
* **Mengapa Bukan yang Lain?**
  + **Microkernel**: Relevan untuk niche seperti real-time systems (QNX di autonomous vehicles) atau high-security (seperti di perbankan atau militer). Namun, overhead-nya membuatnya kurang cocok untuk desktop/general-purpose OS. Adopsi rendah di consumer market karena performa kalah dari Linux/Windows.
  + **Layered Architecture**: Kurang relevan karena terlalu kaku untuk evolusi cepat hardware/software modern. Lapisan vertikal menyebabkan bottleneck di era multi-core dan distributed systems. Ia lebih historis, memengaruhi desain seperti virtual machine layers (VMware/Hyper-V), tapi bukan kernel utama.