1.Mengapa system call penting untuk keamanan OS?

karena berfungsi sebagai gerbang kontrol antara program pengguna dan kernel sistem operasi, sehingga mencegah akses tidak sah dan menjaga stabilitas sistem. Kernel memvalidasi setiap permintaan melalui system call, memastikan program hanya mengakses sumber daya yang diizinkan, seperti memori atau perangkat keras

2.Bagaimana OS memastikan transisi user–kernel berjalan aman?

melalui pemisahan mode user dan mode kernel, yang menciptakan dua tingkat hak istimewa terpisah. Transisi ini dikelola oleh mekanisme khusus seperti panggilan sistem (system calls) dan interrupt, yang memvalidasi permintaan dari user mode sebelum kernel mengeksekusinya. Mekanisme Keamanan:

* Operasi Dual-Mode : OS menjlankan dua mode operasi, yaitu user mode (mode pengguna) untuk aplikasi dan kernel mode (mode kernel) untuk kernel.
  + User Mode : Hanya dapat mengakses sumber daya yang diizinkan untuk proses yang sedang berjalan, tidak bisa langsung mengakses hardware atau memori vital.
  + Kernel Mode : Memiliki akses penuh ke seluruh hardware dan memori sistem.
* Kontrol Akses dan Perizinan:

Kernel bertanggung jawab untuk mengelola kontrol akses dan izin pengguna, serta operasi data yang aman untuk mencegah akses tidak sah.

* Panggilan Sistem (System Calls):

Ketika sebuah aplikasi membutuhkan layanan kernel (seperti membaca file atau membuat proses baru), aplikasi harus melakukan panggilan sistem. Panggilan ini adalah cara yang aman untuk meminta izin, dan kernel akan memvalidasi permintaan tersebut sebelum mengeksekusinya.

3. Sebutkan contoh system call yang sering digunakan di Linux.

system call yang paling sering digunakan di Linux adalah

**1. Manajemen Proses**

* fork(): Membuat proses anak yang merupakan salinan persis dari proses induknya.
* execve(): Mengganti citra program dari proses yang sedang berjalan dengan program baru.
* exit(): Mengakhiri eksekusi sebuah proses.
* wait4(): Menunggu hingga proses anak selesai atau statusnya berubah.
* getpid(): Mengambil ID proses dari proses yang sedang berjalan.
* kill(): Mengirim sinyal ke proses tertentu.

**2. Manajemen Berkas (File)**

* open(): Membuka berkas dan mengembalikan *file descriptor* yang digunakan untuk operasi berikutnya.
* read(): Membaca data dari *file descriptor* ke dalam *buffer*.
* write(): Menulis data dari *buffer* ke *file descriptor*.
* close(): Menutup *file descriptor* yang sebelumnya dibuka.
* stat(): Mengambil informasi status berkas, seperti ukuran dan izin akses.
* mkdir(): Membuat direktori baru.
* rmdir(): Menghapus direktori.

**3. Manajemen Perangkat (Device)**

* ioctl(): Melakukan operasi masukan/keluaran pada perangkat keras, seperti mengubah pengaturan *terminal*.
* read() dan write(): Digunakan juga untuk berinteraksi dengan perangkat, tidak hanya berkas.
* select(): Memantau beberapa *file descriptor* untuk aktivitas I/O.

**4. Manajemen Informasi**

* gettimeofday(): Mendapatkan waktu sistem saat ini.
* sysinfo(): Mengambil informasi tentang sistem, seperti penggunaan memori dan beban CPU.
* uname(): Mendapatkan informasi tentang kernel saat ini.
* getuid(): Mendapatkan ID pengguna (UID) dari proses yang sedang berjalan.

**5. Manajemen Memori**

* brk(): Mengubah lokasi batas atas data (*data segment*) suatu proses untuk mengalokasikan atau membebaskan memori.
* mmap(): Memetakan berkas atau perangkat ke memori suatu proses.