1. mlock() Equivalent Implementation Example in Windows

1.1 User Space: VirtualLock() in Windows API

When you want to lock memory in Windows to prevent it from being paged to disk (like mlock() in Linux), you use the **Windows API function VirtualLock()**. Here's what look like:

#

This function uses the Windows API call VirtualLock() to request that the specified memory region remains in RAM.

C

```
#include <windows.h>
#include <stdio.h>

int lock_memory(void *addr, SIZE_T size) {
    if (VirtualLock(addr, size)) {
        return 0; // Success
    } else {
        return GetLastError(); // Failure
    }
}
```

1.2 Kernel Layer: Underlying Behavior

While you don't directly interact with the Windows kernel for this, VirtualLock() internally:

- Maps the pages into the process working set.
- Prevents the OS from paging them to disk.
- May require special privileges (e.g., SeLockMemoryPrivilege).

Step	Component	Action
1	VirtualLock()	User-space WinAPI call
2	ntdll.dll	Translates to NtLockVirtualMemory syscall
3	ntoskrnl.exe	Kernel handles syscall; Memory Manager processes the request
4	Memory Manager (Mm)	Pins memory pages, sets flags to prevent paging

1.3 Privilege Considerations

To use VirtualLock() effectively, the process may need the **“Lock pages in memory”** privilege, configurable through Local Security Policy:

- Run secpol.msc > Local Policies > User Rights Assignment > "Lock pages in memory"
- Add your user account.

Without this privilege, the function may fail or only allow a small amount of memory to be locked.