WINDOWS
11 OS

BAHIRDAR UNIVERSITY BiT

FACULTY OF COMPUTING
OPERATING SYSTEM AND SYSTEM PROGRAMMING

COURSE CODE:
OSSP

NAME:
Merdokyos Semeneh

ID NO:1602091

SECTION :
B individual Assignment

SUBMITTED TO:
Lec WENDIMU BAYE

1.Implementation of mlock() system call on windows 11

While mlock() is the Unix-like operating systems' (such as Linux) system call to lock the

memory pages so that they are not swapped out, in Windows 11, one encounters similar

functionality within its native API. The equivalent concept is locking memory through

Windows system calls such as virtualLock() of the Win32 API. These operations are

required by security-conscious or real-time applications for which paging of memory can

introduce latency or a risk of sensitive data being leaked to disk.

The intention behind mlock() and its Windows counterpart is to lock a specified region of

memory into RAM in a manner that causes the operating system not to page it out to the

paging file (i.e., virtual memory on disk). This is of particular importance in the following

circumstances:

Preventing data leaks of sensitive information (e.g., cryptographic keys)

Reducing latency in high-performance or real-time systems

Guaranteeing availability of memory for critical execution

Windows 11 Equivalent: VirtualLock()

Under Windows, the counterpart of mlock() is virtualLock. It locks the specified range of the

process's virtual address space into physical memory.

Usage, like previous Windows NT-based operating systems, offers VirtualLock() for that

developer who needs strict control over memory behavior. Some examples of situations are:

1. Passwords and encryption keys held securely in RAM-only.

2. Optimizing performance in latency-sensitive applications (e.g., trading systems, audio

processing).

3. Scientific or engineering applications requiring predictable memory access times.

Use Case Suppose a developer is writing a secure password manager application for

Windows 11. As the user types in a master password, the application may assign a small

memory buffer, copy the password there, and call VirtualLock() to ensure that:

1. The buffer is not paged out of physical memory.

2. The password is never written to disk, even temporarily.

3. Once used, the memory can be securely cleared and unlocked.

Limitations and Considerations

1.Windows imposes limits on the amount of memory that a process can lock.

2.Excessive or inappropriate use of VirtualLock() can worsen system performance.

3.Programmers must be careful to manage locked memory to avoid leaks and to release and

unlock it appropriately.

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.