BinaryMutex: Technical Documentation

Amlal El Mahrouss

November 30, 2025

1 Abstract

The BinaryMutex is a core component of NeKernel (NeKernel/Legacy VMKernel) based systems. The pattern excludes other acquirers to own the USER_PROCESS that is currently being hold. Thus the acquiree is the USER_PROCESS itself

2 Overview

The BinaryMutex comes from the need to make sure that no race conditions occurs in kernel code. Most of those race conditions happens in process handling code. Thus the design of the BinaryMutex (which holds a process handle to it)

```
BinaryMutex mux;
mux.Lock(process);

// Say we want to interact with the process itself on this thread,
we can then make sure that no race condition happens by using:
constexpr auto kSecondsMax = 5;
while (mux.WaitForProcess(kSecondsMax));

// finally do our task now.

process.DoFoo();
```

3 Implementation

The source implementation consists of this simple C++ class:

```
class BinaryMutex final {
 public:
  explicit BinaryMutex() = default;
  ~BinaryMutex()         = default;

 public:
```

```
  bool IsLocked() const;
  bool Unlock() ;

 public:
  BOOL WaitForProcess(const UInt32& sec) ;

 public:
  bool Lock(USER_PROCESS* process);
  bool LockAndWait(USER_PROCESS* process, TimerInterface* timer);

 public:
  NE_COPY_DEFAULT(BinaryMutex)

 private:
  USER_PROCESS* fLockingProcess;
};
```

4 Conclusion

This design pattern is useful for systems that need to make sure that a process isn't tampered by concurrent usages. Thus its existence in Legacy VMKernel and NeKernel.

5 References

NeKernel: NeKernel
    Legacy VMKernel: Legacy VMKernel
    BinaryMutex: BinaryMutex