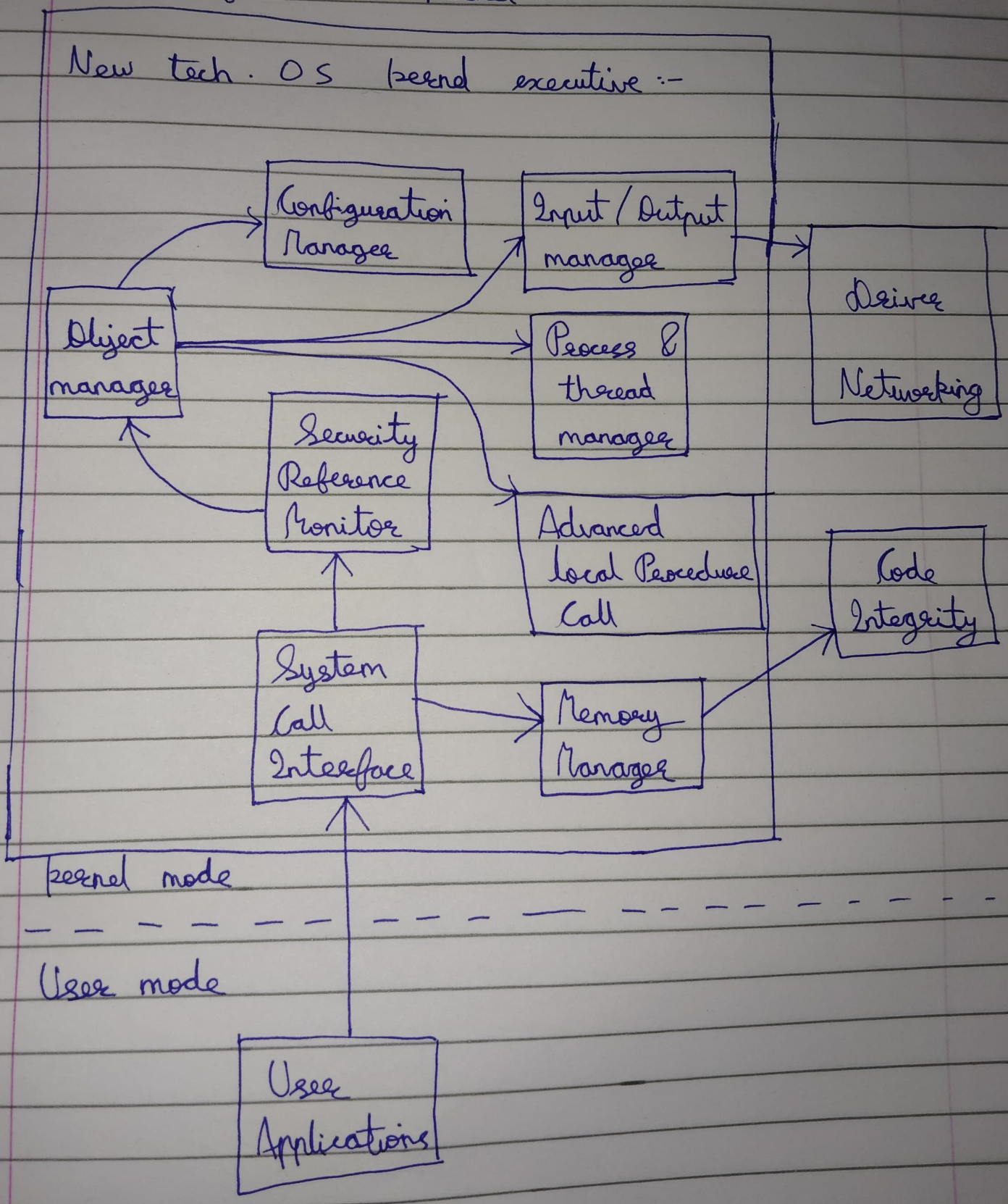
Windows Security Internals

Chapter 2 : Kernel

Heart of OS → kernel

New tech. OS kernel executive :-



Date
Page

✡ Overview

1) User application :

- eg. chrome, ms word
- cannot directly access kernel functions

2) System call Interface

- Gateway which allows user mode applications to access kernel mode functions

3) Memory manager :

- manages & controls different types of memory

4) Code Integrity :

- prevent un-authorized code execution

5) Security Reference Monitor

- Central management of security access

6) Object Manager :

- Resource management.

7) Configuration manager

- manage system database configuration

8) Input / Output Manager

- Coordination of all input / output operation

9) Process and thread manager

- Manage Program execution units

10) Advanced Local Procedure Call (ALPC)

- Inter process communication

11) Drivers

- Interface for hardware and software devices

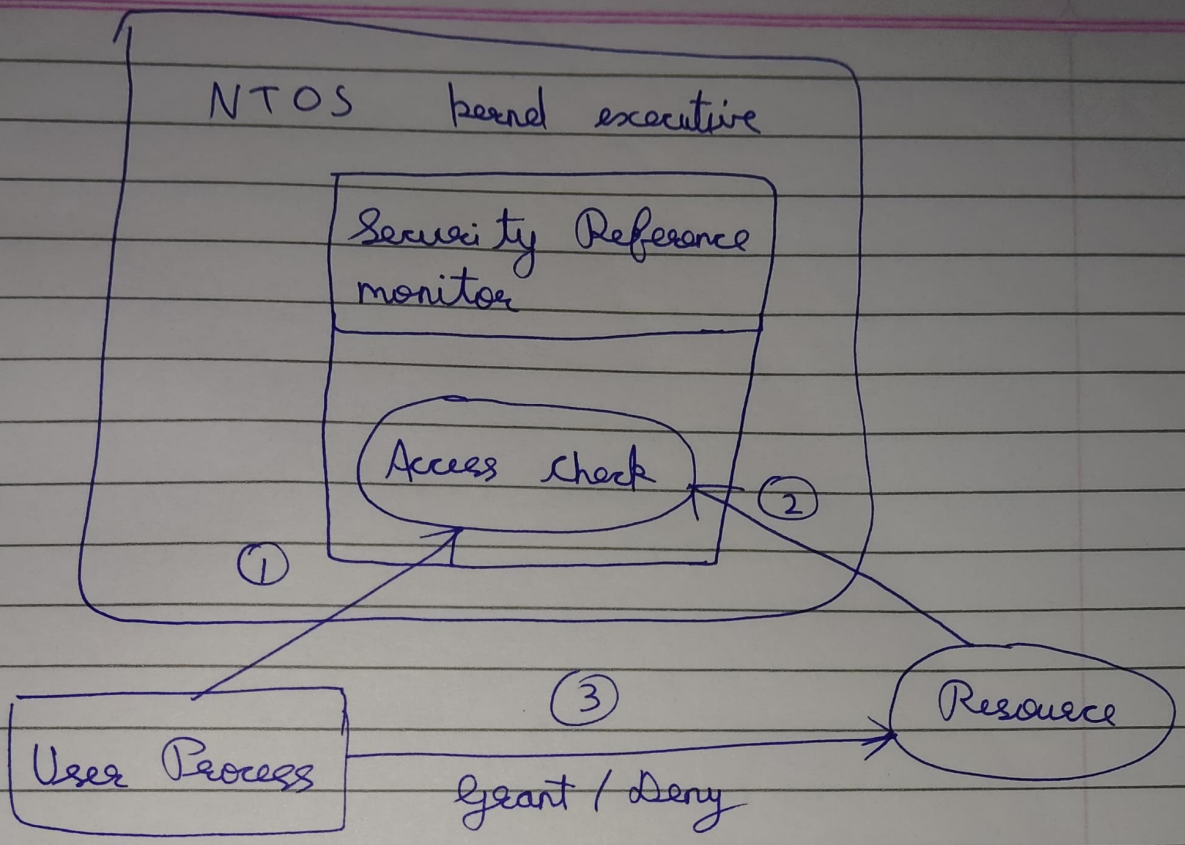
- All these system components communicate with each other using APIs

- All API names have specific kind of prefix according to their sub system

Prefix	Subsystem
Nt / Zw	System Call Interface
Se	Security Reference Monitor
Ob	Object Manager
Ps	Process & thread manager
Cm	Configuration manager
Mm	Memory manager
Io	Input / output manager
Ci	Code Integrity

The Security Reference Monitor:

- Security mechanism - restrict - which user has access to what resource



System → every process → access token

Access token → managed by SRM

↳ defines identity of user associated with that process

SRM → access token ⇒ access check

Access check → query ⇒ resource's security descriptor

↓

calculate level of access to ⇐ compare with access token grant

SRM → audit event → user access resource

SRM → users and groups → binary structures
↓
Security Identifiers (SIDs)

Users → name ✓
→ SID ✗

SRM → SID ✓
→ name ✗

Conversion → name - SID → Local Security Authority Subsystem (LSASS)
↓
User independant privileged process

(SDDL)
name → Security Descriptor Definition Language
↓
SID ← String

SDDL → entire Security Descriptor → resource

Powershell :-

- Get-NtSid -Name "Users"

- Get → Retrieve

Nt → API system call
Sid → security identifier
- Name → parameter
"Users" → value of name parameter
Retrieve Sid of Group Users

Output:

BuiltIn is domain
S → SDDL SID
1 → version
5 → Security Authority
32/544 → Nt Authority Group

User 1
↓
Process
↓
Access token (User 1)
↓ → SDDL → name → SDI
SRM
↓
Access check - access token
↓
level of access grant

The Object Manager :-

Windows → kernel → everything (file / process / thread)
↓
object structure

Object → Security Descriptor → which user access, type of access (read, write, exec)

kernel → object manager → object → allocation, resource, lifetime

kernel → several types of Object → each unique functionality

Powershell → Get-NtType

- - - - - - -

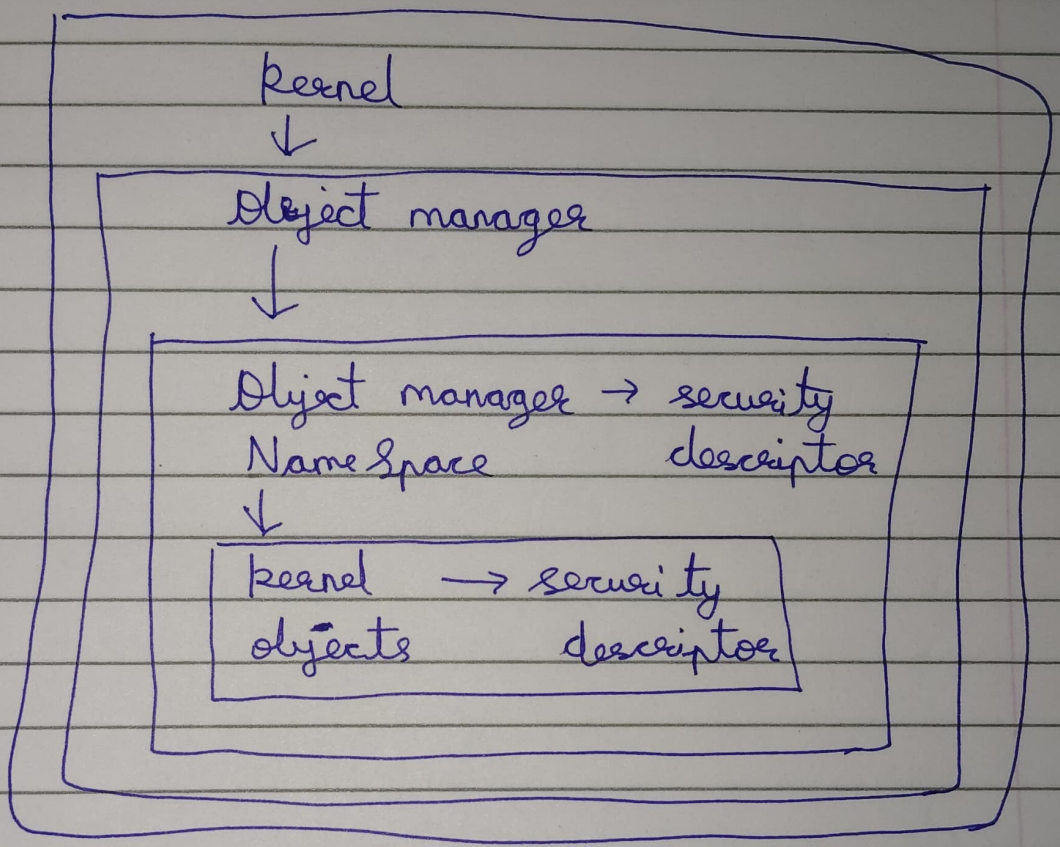
User → file system → files

kernel → file system → kernel objects → Object Manager Namespace (OMNS)

OMNS → directory objects → kernel objects
↓
security descriptor

OMNS ~~em~~ enumeration :
Powershell : ls Nt Object:\ | Sort-ObjectName

Symbolic links → redirect one OMNS path → another



System Calls

User mode app → process → system call → kernel

kernel object ← OMNS ← Object manager

System call → name → starts with Nt or Zw

System call → verbs → operate on kernel object :-

- Create new object
- Open existing object
- Query object information & properties
- Set object information & properties

User → process → thread → [System call → Mutant Object]

kernel object ← OMNS ← Object manager ← kernel ←

Process → thread 1 → Mutant Object → thread x → kernel
Process → thread 2 → Mutant Object → thread x → kernel

x = 1 → prevent deadlock

Creation of mutant object → NtCreateMutant sys call

```
NTSTATUS    NtCreateMutant (
    HANDLE*  FileHandle,
    ACCESS_MASK  DesiredAccess,
    OBJECT_ATTRIBUTES*  ObjectAttributes,
    BOOLEAN  InitialOwner );
```

NTSTATUS → function's return type

HANDLE* → pointer to handle → reference to kernel object → retrieve

ACCESS_MASK* → level of access

OBJECT_ATTRIBUTES* → attributes of object

BOOLEAN → true / false → grant / deny ownership

Object Attributes :

- Inherit
- Permanent
- Exclusive
- Case Insensitive
- Open if → handle to existing object if avail.
- Open link → open object if linked to
- kernel handle → when kernel mode → kernel's handle
- Force Access check
- Ignore Impersonated device map
- Don't traverse → do not follow symbolic link

NTSTATUS Code :-

- All system calls return NTSTATUS code

code → 32 bit

Severity	Customer code	Reserved Bit	Facility	status code
30-31	29	28	16-27 bits	0-15 bits

Severity → success → 0
→ informational → 1
→ warning → 2
→ error → 3

Customer code (cc) → microsoft defined → 0
→ third part defined → 1

Reserved Bit (R) → 0

Facility → component / subsystem → default → 0
→ debugger → 1
→ Win32 API → 7

Status code → unique for facility → decided by implementor

Status codes → powershell → Get-NtStatus

Object handles :-

Object manager → pointer → kernel memory

System call → handle → process → User
↓ < handle >
kernel ← Object Manager
object

Process → handle table → handle's numeric identifier
→ granted access level to handle
→ pointer to object in kernel mem

User → process → handle value → system call → kernel API → check
→ type of access
→ object type
→ approved → handle table → handle to kernel pointer → pointer returned
→ denied

Access Masks:-

handle table → granted access level value → Access Mask

Access mask → 4 components :

Generic Access	~~Special~~ ~~Specific~~ access	Standard Access	type specific access	
28-31	21-27	16-20	0-15	Bits

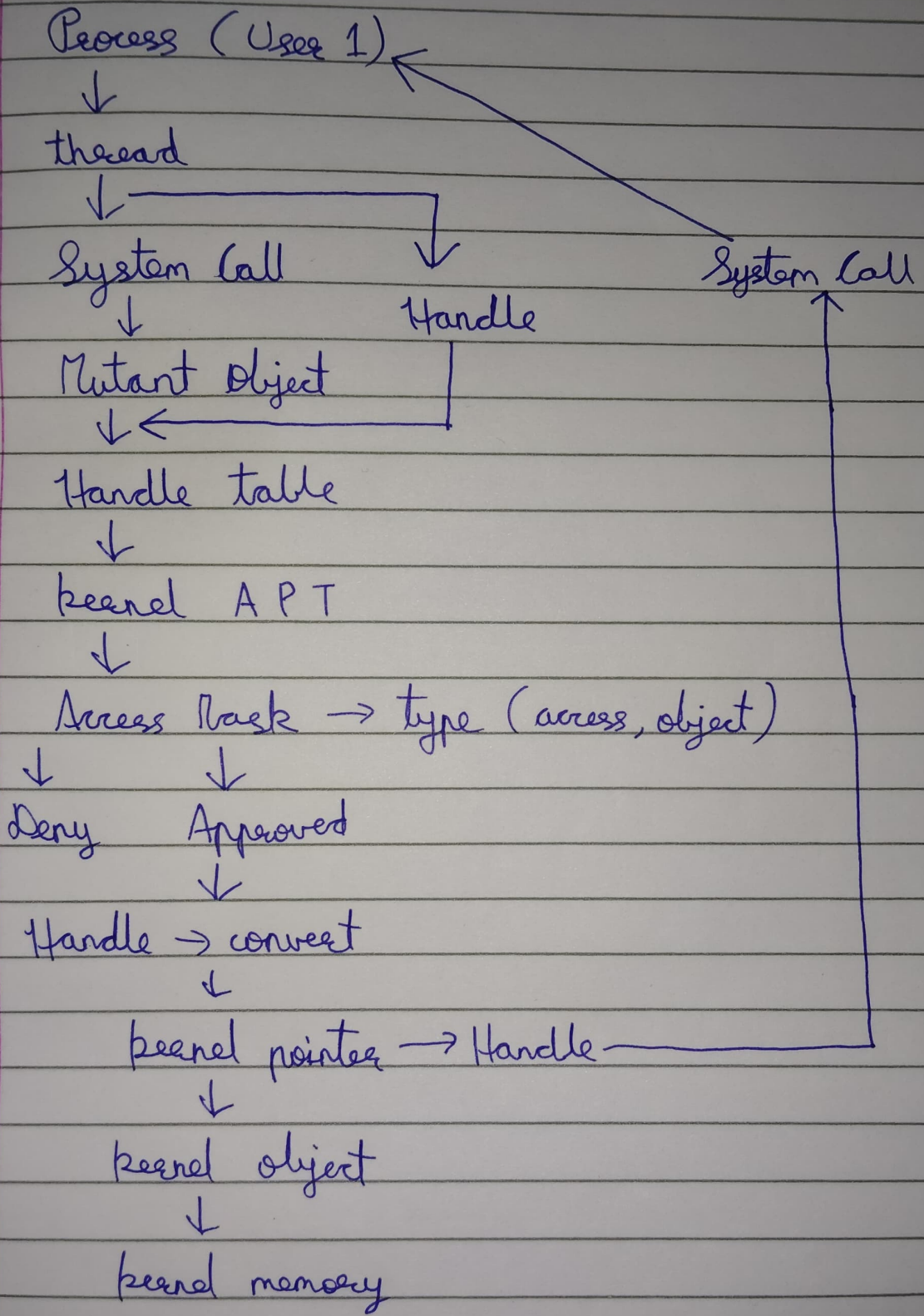
type specific → operations allowed → kernel object type

standard access → operations allowed → any kernel object type
- → Delete
- → Read SD
- → Write SD
- → Write owner
- → Synchronize
- → SD = Security Desc.

Special Access → Access system security → read / write audit info
- → max allowed access

generic access → request access to kernel object

Handle → convert → kernel pointer
- → generic mapping table → read, R
  write, W
  execute, E
  All A


Process (User 1)
thread
System Call
Handle
Mutant Object
Handle table
kernel APT
Access Mask → type (access, object)
Deny
Approved
Handle → convert
kernel pointer → Handle
kernel object
kernel memory
System Call

Date
Page

Handle ~~Rutation~~ Duplication :-

Process → Handle 1 → Privilege X → kernel object

Process → Handle 1 → Destroy → kernel object
Destruct

Process → Handle 1 → Privilege X → kernel object
create ↓ Y > X ↑ kernel object
→ Handle 2 → Privilege Y
→ Anti Destruct Policy

Process → Handle 1 → destruct — X
→ Handle 2 → System can't destruct → Privilege + → kernel object

- - - - - - - - - -

Query and Set Information System Calls :-

kernel object → info → state

first param → object handle
second param → type of process info
third param → enumeration of names of info classes
fourth param → buffer length to recieve info
fifth param → if success, how much buffer was used.

✡ The Input / Output Manager :-

- Access to Input / output Devices via device drivers

I/O Manager (Directory) → Physical Device Drivers
→ file system drivers

Load → new driver → Nt Load Driver System Call

Directory → list content → admin ✓
→ normal user ✗

normal user → interact → driver → Device Object

Directory → Drivers → Io Create Devic API
↓ ↓ create
Possible 0 ← Device Objects
(if no interact")

User → dedicated system calls → I/O manager ✗

✓ → User → file object → Device
system calls

Physical System drive → System Root → symbolic link
↓ OMNS
↓ Windows Directory
↓ PE files etc.

User → search → System Root PE file → control
↓
Object Manager → find full path → hand over to I/O manager → new protocols

---

✱ The Process and thread manager :-

User-mode code → Process → x threads

User → unknown Process A → modify SD
↓
Privilege
↓
therefore not enabled

User → access process → name ✗
→ access process ID → ✓

Idle process → PID 0 → OS is idle

System process → PID 4 → kernel mode
↓
background thread exec

☆ The Memory Manager :-

Process → space → Virtual Memory Address

32 bit process → access → 2 GB on 32 bit sys
↳ 4 GB on 64 bit sys

64 bit process → access → 128 TB

kernel → memory manager → allocation → Virtual memory address space

Virtual memory space → shared → process running state
↓ ↳ executable code
memory allocations → protection states → read/ write/ exec

Process → Memory Region → Address
→ Size
→ Protection
→ State

State → Commit → allocated + avail to use
→ Reserve → allocated, not avail to use
→ free → not allocated, not avail to use

Virtual memory allocation → minimum default allocation size
↓
System dependent

Section Objects

Another method → virtual memory allocation → Section objects
↓
type of kernel object
↓
memory mapped

Section object → read/write file into memory
→ sharing memory bet^n processes
↳ modify one → reflect other

Process → Create → Section → size, protection, handle
Object
↓
map into virtual memory (same protection type)
↓
supply OMNS path
↓
shared with other process

✱ Code Integrity :-

- Verify and restrict → what files/code → execute in kernel
↓
optionally in user mode

- every executable on Win is signed using mechanism → authenticode → cryptographic sign embedded in file

status → file validity, sign verified

✡ Advanced local Procedure Call :-

ALPC → local, cross platform communication → local remote procedure call APIs

ALPC server port → put into OMNS → client

✡ The Configuration Manager :-

CM / Registry → config info → overall system
↓
key : value
folder : file
↓
accessed through OMNS