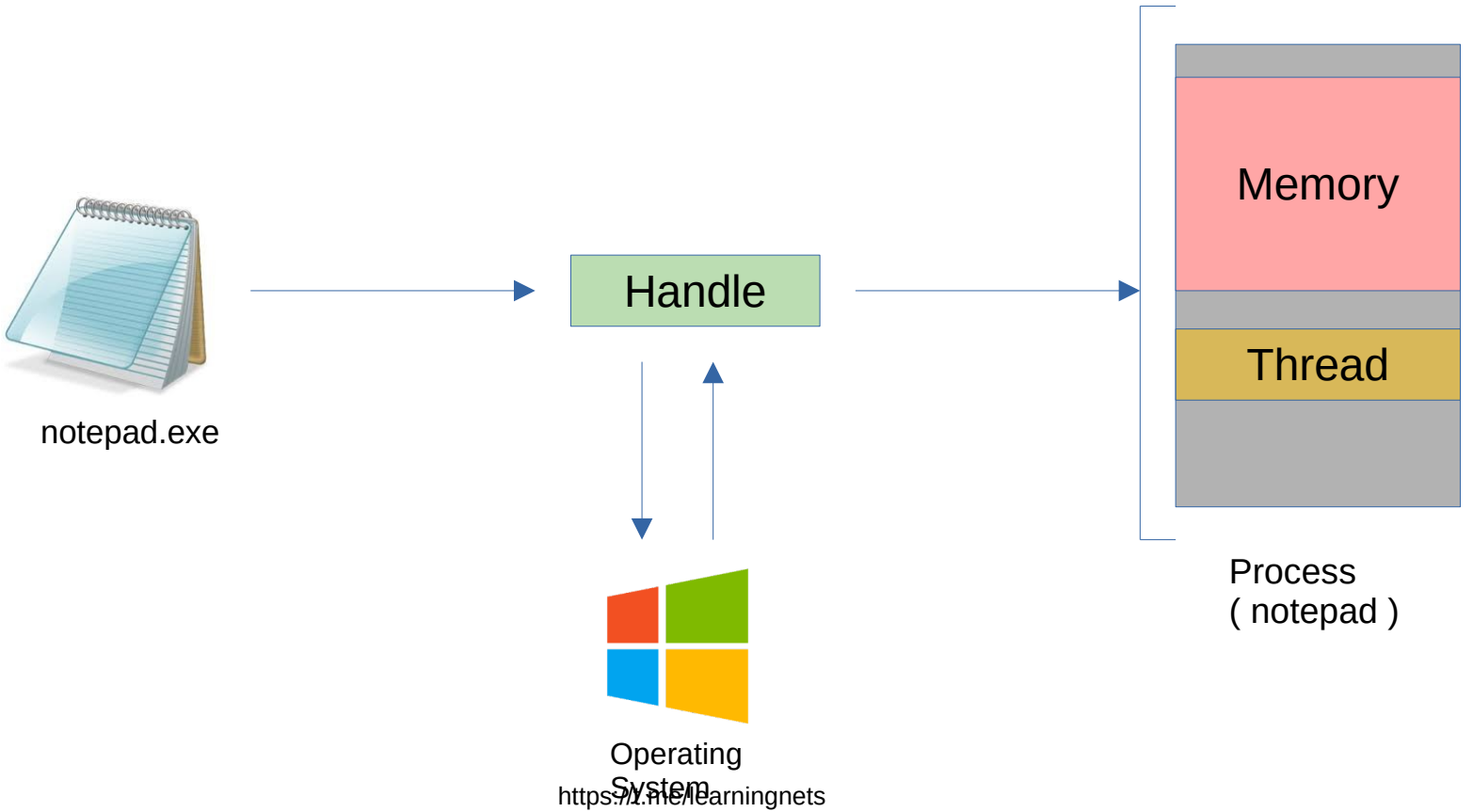
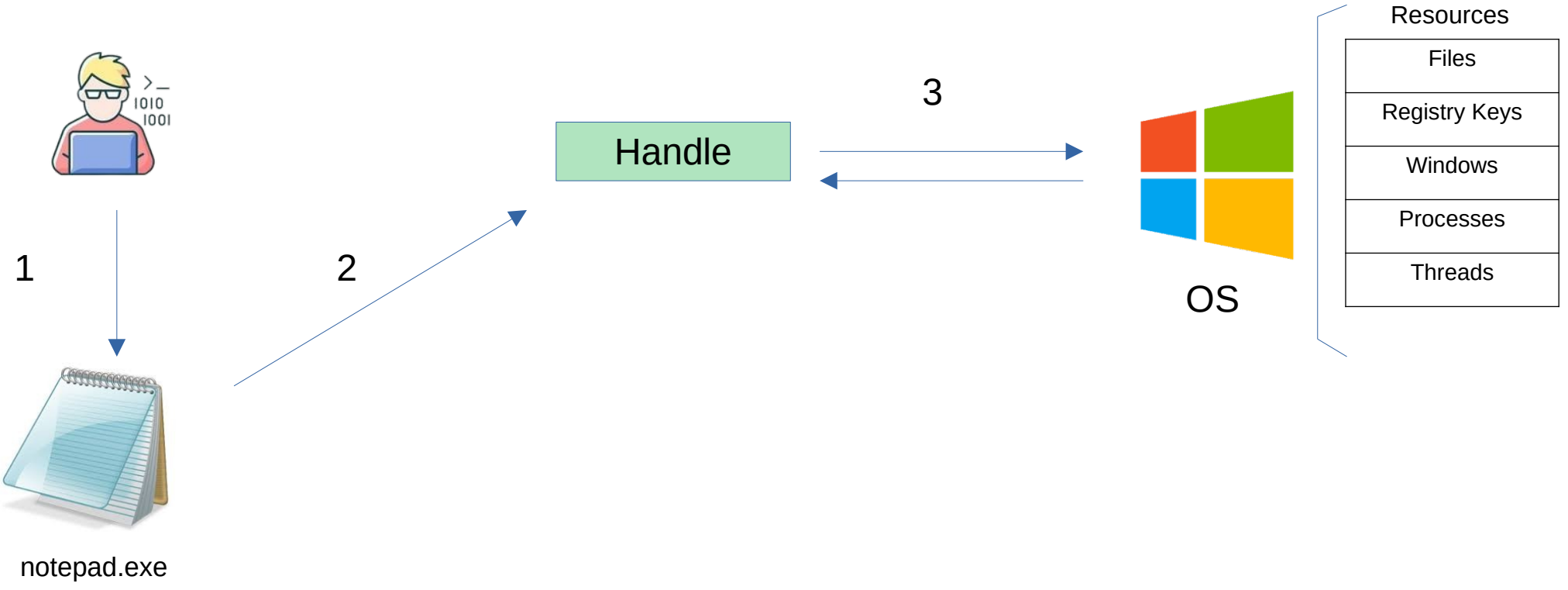
Basic windows API programming

https://t.me/learningnets

Important components in malware programming :



Handles: When you request something from Windows, it gives you a handle so that it can track your request. This handle helps Windows identify and manage different resources like files, windows, and processes.



https://t.me/learningnets

Handles: When you request something from Windows, it gives you a handle so that it can track your request. This handle helps Windows identify and manage different resources like files, windows, and processes.

1

notepad.exe

2

Handle

3

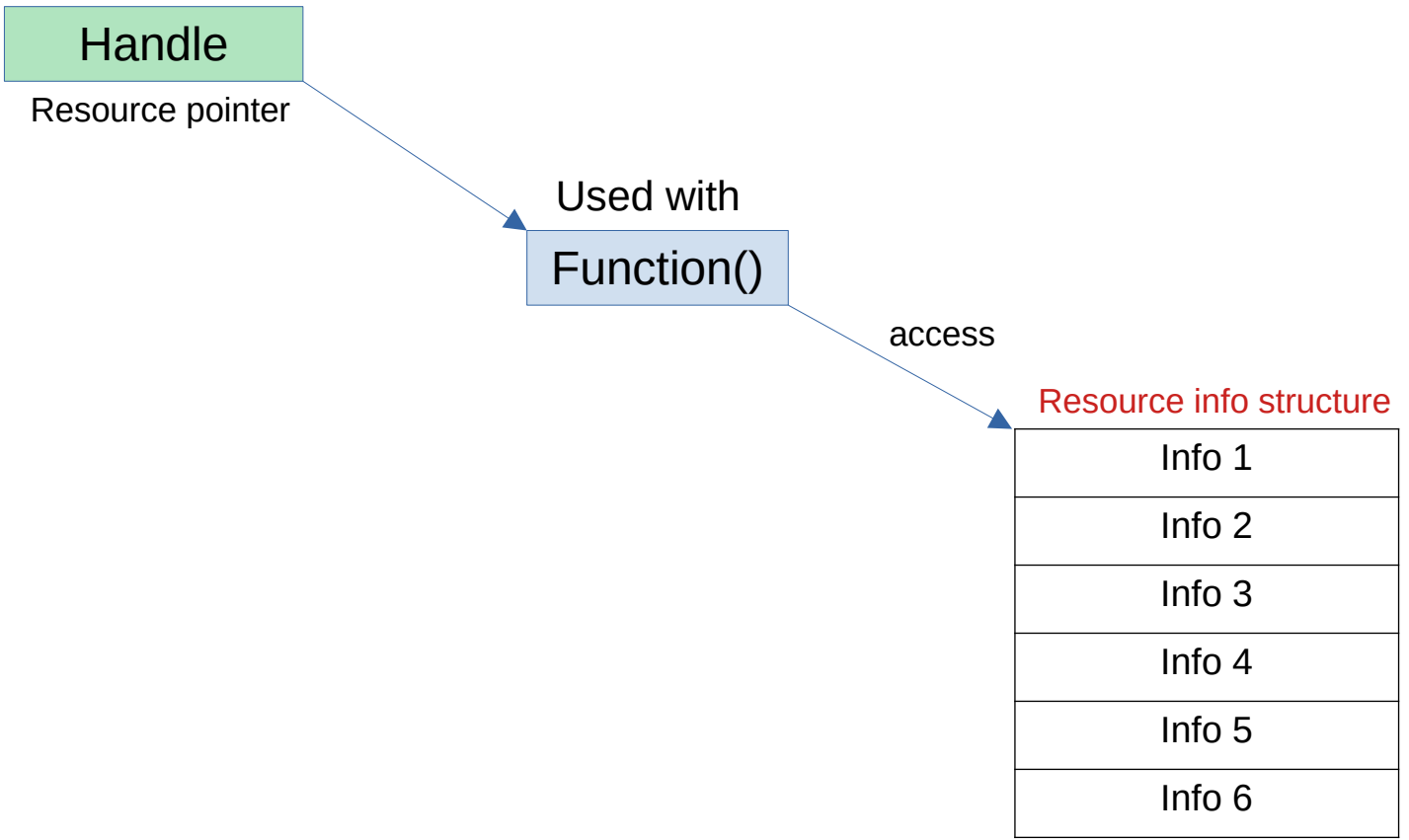
- Files: CreateFile() returns a handle to a file.
- Processes: OpenProcess() returns a handle to a process
- Threads: CreateThread() returns a handle to a thread
- Windows (GUI): FindWindow() returns a handle to a window
- Registry Keys: RegOpenEx() returns a handle to a registry key

OS

Resources

Files
Registry Keys
Windows
Processes
Threads

https://t.me/learningnets



https://t.me/learningnets

📌 Table of Windows Handle Types

Handle Type	Data Type	Represents	Used With	Example Function to Get Handle	Notes
Window Handle	`HWND`	A window created by an application	`FindWindow()`, `GetForegroundWindow()`, `ShowWindow()`	`FindWindow(class, title)`	Used to manipulate windows (resize, move, etc.)
Process Handle	`HANDLE`	A running process	`OpenProcess()`, `GetCurrentProcess()`, `TerminateProcess()`	`OpenProcess(ACCESS_RIGHTS, FALSE, PID)`	Needed for process manipulation (memory reading, injection)
Thread Handle	`HANDLE`	A thread within a process	`OpenThread()`, `CreateThread()`, `SuspendThread()`	`OpenThread(ACCESS_RIGHTS, FALSE, TID)`	Allows control over specific threads in a process
File Handle	`HANDLE`	An open file	`CreateFile()`, `ReadFile()`, `WriteFile()`	`CreateFile(path, ACCESS, SHARE, NULL, OPEN_EXISTING, 0, NULL)`	Required for file I/O operations
Registry Handle	`HKEY`	A registry key	`RegOpenKeyEx()`, `RegQueryValueEx()`, `RegSetValueEx()`	`RegOpenKeyEx(HKEY_LOCAL_MACHINE, path, 0, ACCESS, &hKey)`	Used to read/write Windows registry values
Event Handle	`HANDLE`	A synchronization event	`CreateEvent()`, `SetEvent()`, `ResetEvent()`	`CreateEvent(NULL, TRUE, FALSE, L"EventName")`	Used to signal between threads/processes
Mutex Handle	`HANDLE`	A mutual exclusion object	`CreateMutex()`, `ReleaseMutex()`	`CreateMutex(NULL, FALSE, L"MutexName")`	Ensures only one thread accesses a resource at a time

https://t.me/learningnets

Semaphore Handle	`HANDLE`	A counting semaphore	`CreateSemaphore()`, `ReleaseSemaphore()`	`CreateSemaphore(NULL, INITIAL_COUNT, MAX_COUNT, L"SemName")`	Controls multiple access to a resource
Pipe Handle	`HANDLE`	A named or anonymous pipe	`CreatePipe()`, `CreateNamedPipe()`, `ReadFile()`	`CreateNamedPipe(name, mode, pipeType, ...)`	Used for inter-process communication (IPC)
Token Handle	`HANDLE`	A security token	`OpenProcessToken()`, `DuplicateTokenEx()`	`OpenProcessToken(hProcess, ACCESS, &hToken)`	Stores user security info (privileges, groups)
Job Handle	`HANDLE`	A job object (group of processes)	`CreateJobObject()`, `AssignProcessToJobObject()`	`CreateJobObject(NULL, L"JobName")`	Allows managing multiple processes together
Console Handle	`HANDLE`	A console input/output	`GetStdHandle()`, `WriteConsole()`	`GetStdHandle(STD_OUTPUT_HANDLE)`	Used for reading/writing to the console
Device Handle	`HANDLE`	A device driver interface	`CreateFile()` (for `\\.\DeviceName`)	`CreateFile(L"\\\\.\\PhysicalDrive0", ...)`	Accesses hardware like disks, serial ports
Service Handle	`SC_HANDLE`	A Windows service	`OpenService()`, `StartService()`	`OpenService(hSCManager, L"ServiceName", ACCESS)`	Used to start, stop, configure Windows services
Desktop Handle	`HDESK`	A desktop object	`OpenDesktop()`, `SwitchDesktop()`	`OpenDesktop(L"Winlogon", 0, FALSE, ACCESS)`	Manages GUI desktops (like login screen)
Station Handle	`HWINSTA`	A window station	`OpenWindowStation()`, `SetProcessWindowStation()`	`OpenWindowStation(L"WinSta0", FALSE, ACCESS)`	A collection of desktops, used for GUI isolation

https://t.me/learningnets

Timer Handle	`HANDLE`	A waitable timer object	`CreateWaitableTimer()`, `SetWaitableTimer()`	`CreateWaitableTimer(NULL, FALSE, L"TimerName")`	Used for precise timing operations
Memory Mapping Handle	`HANDLE`	A shared memory object	`CreateFileMapping()`, `MapViewOfFile()`	`CreateFileMapping(INVALID_HANDLE_VALUE, ...)`	Allows different processes to share memory
Clipboard Handle	`HANDLE`	Data stored in the clipboard	`OpenClipboard()`, `GetClipboardData()`	`OpenClipboard(NULL)`	Allows access to clipboard contents
Accelerator Handle	`HACCEL`	Keyboard shortcut table	`CreateAcceleratorTable()`, `TranslateAccelerator()`	`CreateAcceleratorTable(accelArray, numItems)`	Maps keyboard shortcuts to commands
Menu Handle	`HMENU`	A menu of a window	`CreateMenu()`, `GetMenu()`, `SetMenu()`	`GetMenu(hWnd)`	Manages application menus
Cursor Handle	`HCURSOR`	A mouse cursor	`LoadCursor()`, `SetCursor()`	`LoadCursor(NULL, IDC_ARROW)`	Changes the cursor appearance
Icon Handle	`HICON`	An application icon	`LoadIcon()`, `DrawIcon()`	`LoadIcon(NULL, IDI_APPLICATION)`	Represents an icon used in an application

https://t.me/learningnets

Handle Example: Find out running notepad program

Handle Type	Data Type	Represents	Used With	Example Function to Get Handle	Notes
Window Handle	`HWND`	A window created by an application	`FindWindow()`, `GetForegroundWindow()`, `ShowWindow()`	`FindWindow(class, title)`	Used to manipulate windows [resize, move, etc.]

Handle

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

int main()
{
    HWND hWnd = FindWindow(NULL,"Untitled - Notepad");

    if(hWnd) {
        printf("Found Notepad! Handle:%p\n",hWnd);

    }
    else
    {
        printf("Notepad not found!\n");
    }
    return 0;
}
```

https://t.me/learningnets

Handle Example: Find out running notepad program and minimizing it.

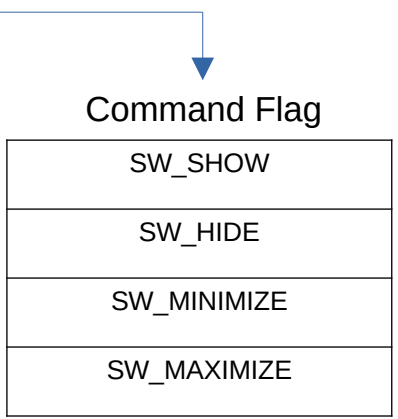
Handle Type	Data Type	Represents	Used With	Example Function to Get Handle	Notes
Window Handle	`HWND`	A window created by an application	`FindWindow()`, `GetForegroundWindow()`, `ShowWindow()`	`FindWindow(class, title)`	Used to manipulate windows (resize, move, etc.)

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

int main()
{
    HWND hWnd = FindWindow(NULL,"Untitled - Notepad");

    if(hWnd) {
        ShowWindow(hWnd, SW_MINIMIZE);

    }
    else
    {
        printf("Notepad not found!\n");
    }
    return 0;
}
```



https://t.me/learningnets

Handle Example: Find out running notepad program and hiding it.

Handle Type	Data Type	Represents	Used With	Example Function to Get Handle	Notes
Window Handle	`HWND`	A window created by an application	`FindWindow()`, `GetForegroundWindow()`, `ShowWindow()`	`FindWindow(class, title)`	Used to manipulate windows (resize, move, etc.)

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

int main()
{
    HWND hWnd = FindWindow(NULL,"Untitled - Notepad");

    if(hWnd) {
        ShowWindow(hWnd,SW_HIDE);

    }
    else
    {
        printf("Notepad not found!\n");
    }
    return 0;
}
```

https://t.me/learningnets

Handle Example: File Handle example (creating a file using file handle).

Handle Type	Data Type	Represents	Used With	Example Function to Get Handle	Notes
File Handle	`HANDLE`	An open file	`CreateFile()`, `ReadFile()`, `WriteFile()`	`CreateFile(path, ACCESS, SHARE, NULL, OPEN_EXISTING, 0, NULL)`	Required for file I/O operations

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


int main()
{
   HANDLE  fileHandle = CreateFile(
                        "example.txt",              //File name
                        GENERIC_WRITE,          // Open for writing
                        0,                      // No sharing
                        NULL,                   //Default security
                        CREATE_ALWAYS,          //Create a new file or overwrite existing
                        FILE_ATTRIBUTE_NORMAL,                  // Normal File
                        NULL                    // No template file
                        );

   if(fileHandle)
   {
      printf("File handle created successfully\n");
       }
   else
{
      printf("Failed to create file. Error: %ld\n",GetLastError());
      return 1;
   }
   return 0;
}
```

https://t.me/learningnets