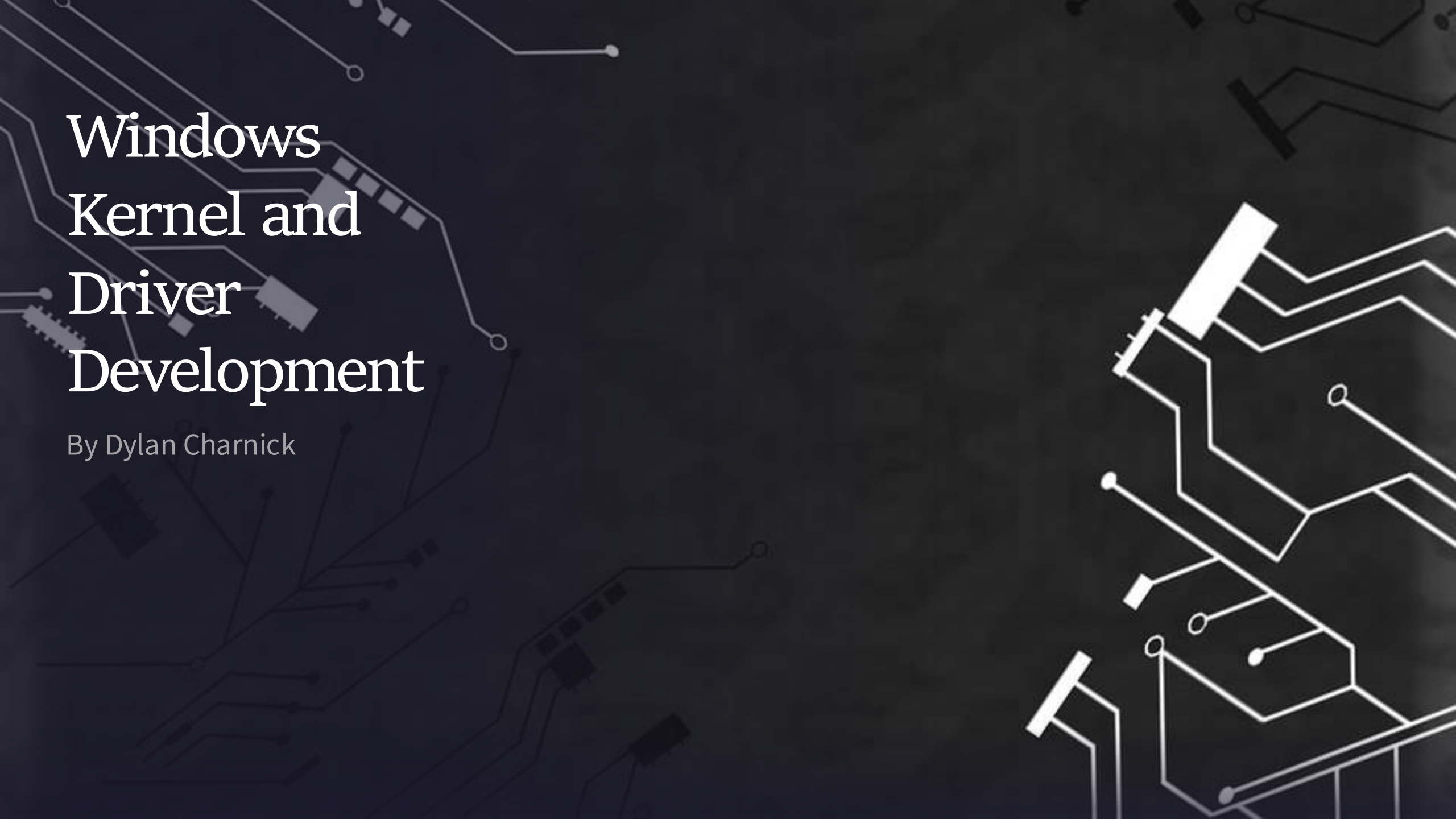
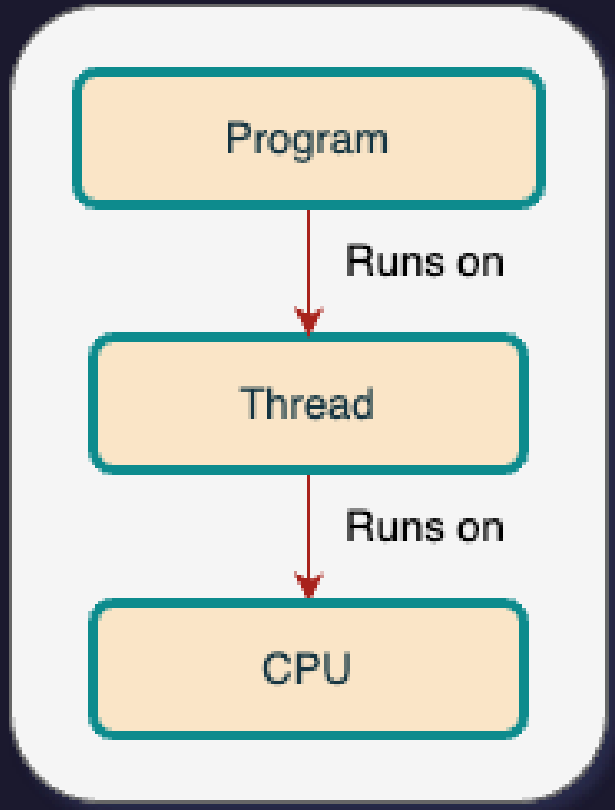
Windows Kernel and Driver Development

By Dylan Charnick



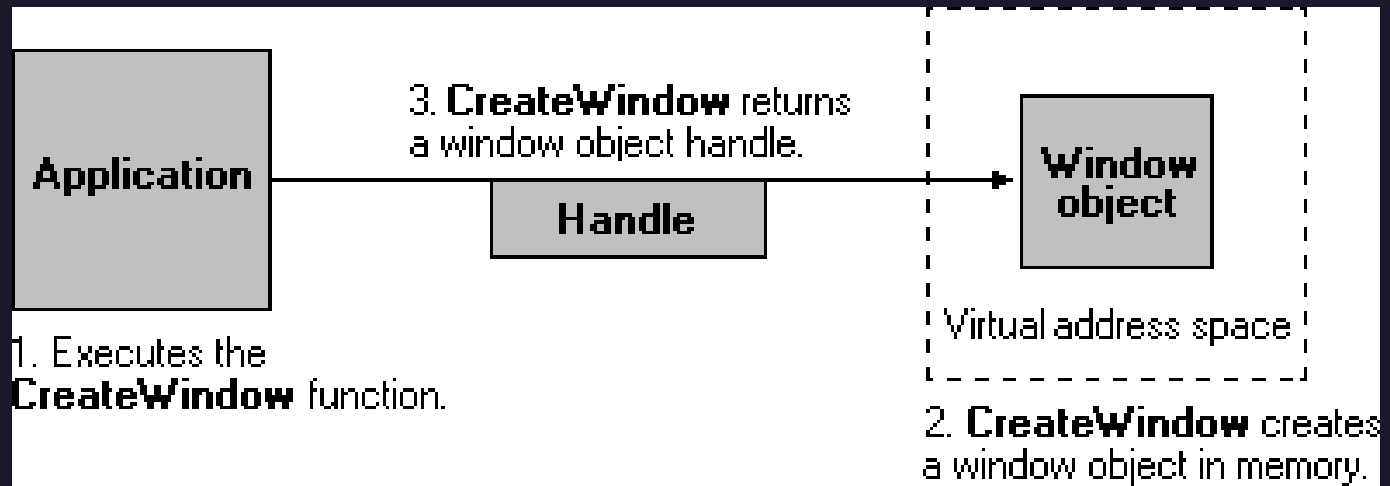
Threads

- A thread is a set of machine code instructions that the processor executes.
- A process can have many threads, each working simultaneously.
- This is when a process implements multiple threads to work simultaneously and report back once they finish their task.



Objects

- An object can be anything, a picture, a string, a section of memory.
- Most objects have handles.
- Kernel objects have handles and user mode processes can use those handles to call kernel mode code



Handles

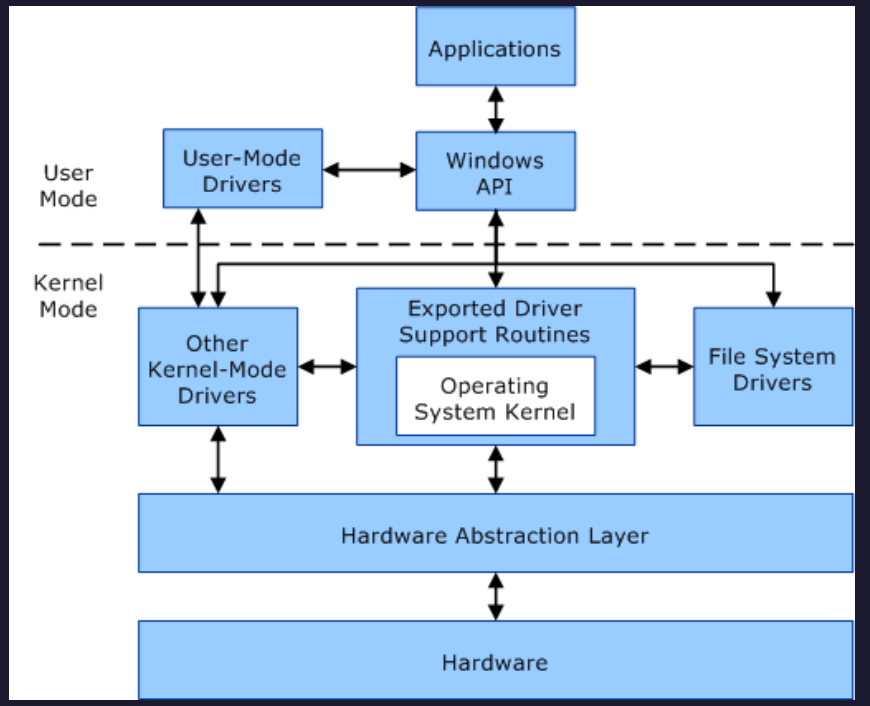
- A handle is a reference to an object which could be a section of memory, a mutex, a pipe, etc.
- If a process wants to get the memory location of the object, they resolve the handle, this gives the process a pointer to the object and freezes the object where it is in memory.
- The object manager is allowed to move the actual object wherever it wants as long as the handle still points to the correct object

Kernel Fundamentals

- What is the kernel?
- Ring levels (0 and 3)
- Paging
- Virtual memory

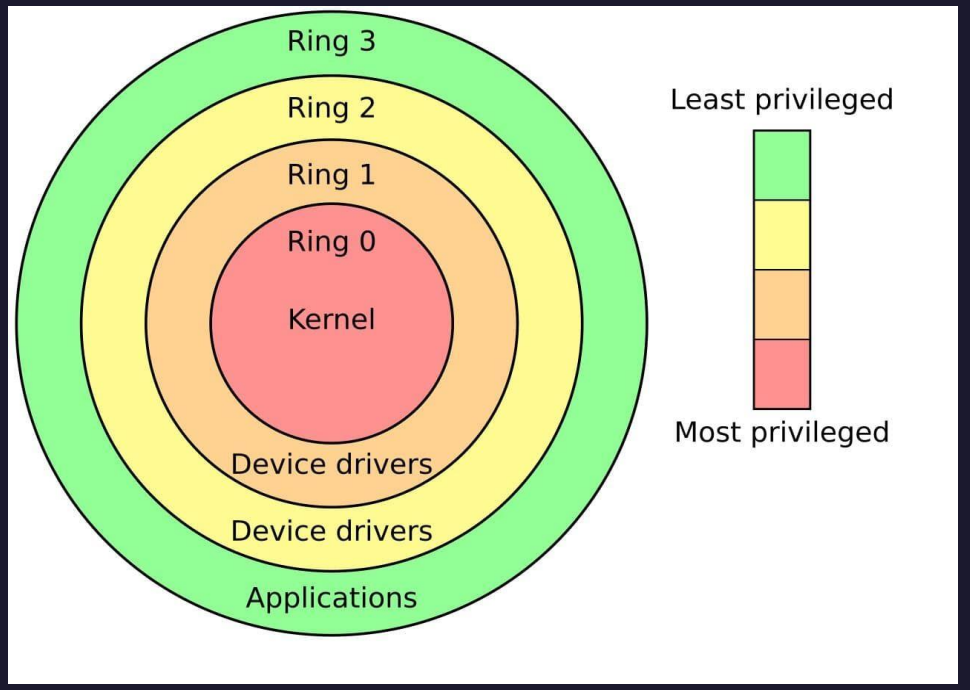
What is the kernel? (Ring O)

- The kernel is the heart of the windows operating system.
- It has access to the entire system and provides the core functionality that the rest of the system uses.
- Anything running at kernel level can see **Everything** happening on your system.



User Mode (often referred to as 'Userland')

- When a process runs in user mode, Windows creates a page table and a new private virtual address space for it.
- This keeps programs separate from each other meaning that if one crashes, the other ones are unaffected.

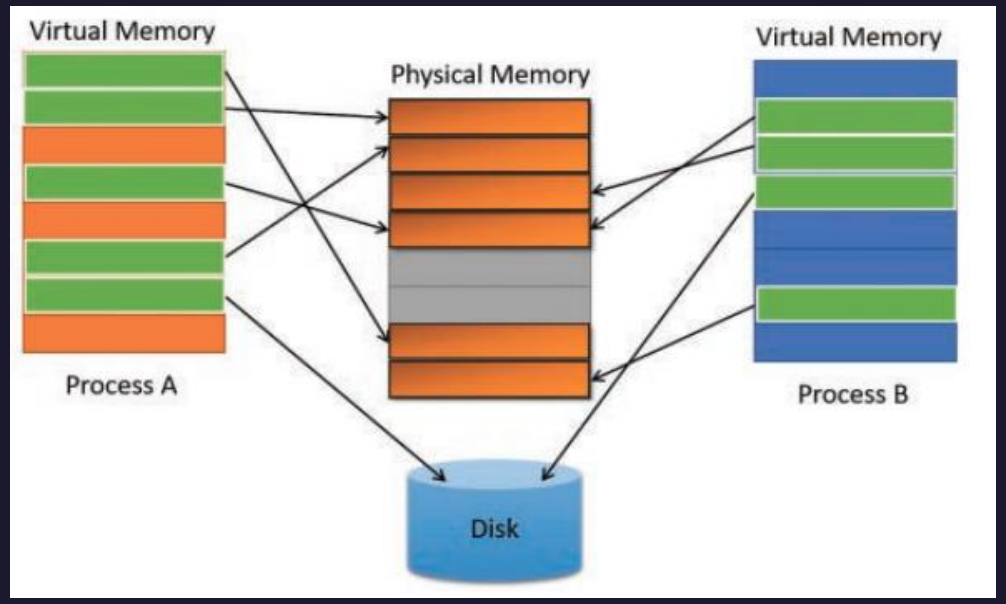


Kernel Mode

- Kernel mode is the execution mode for drivers and kernel-level programs.
- Each program / driver in kernel mode shares the same virtual address space, 0x8-0xFFFFFFFF (A.K.A. system space)
- This means that each driver or program has access to the same memory and if one crashes, they all crash. (A blue screen of death)
- This also means that every application running in kernel mode can access and see the memory of the other applications in kernel mode and user mode.

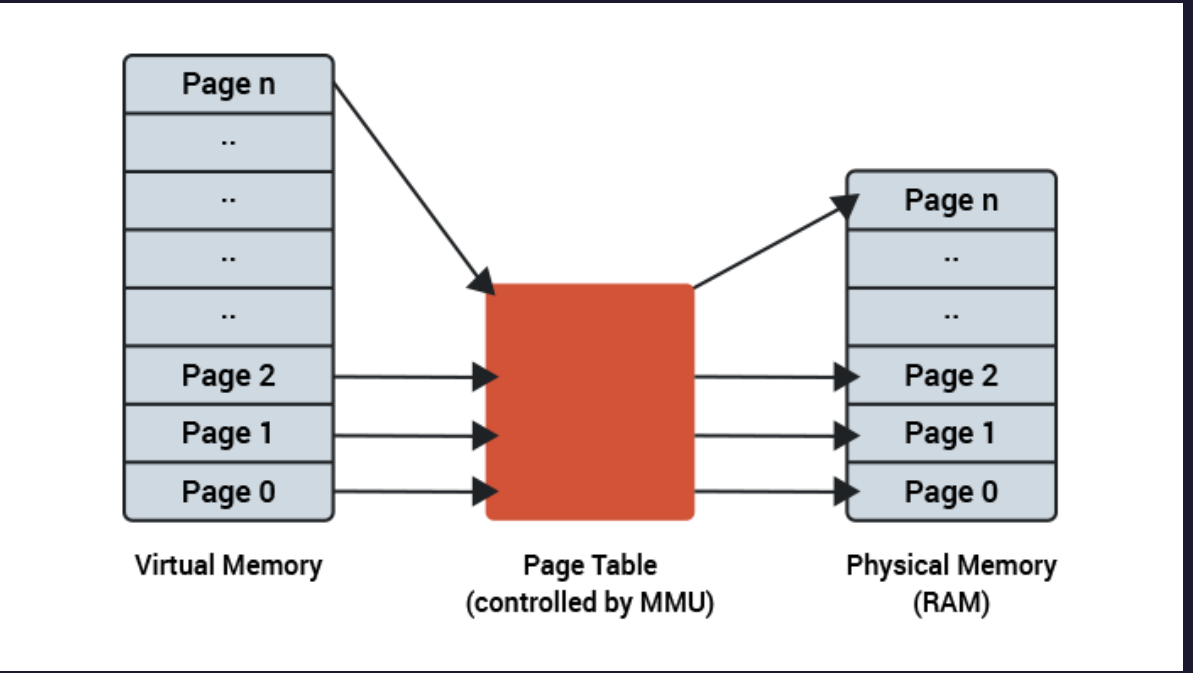
Paging

- A page is a chunk of memory. Each process has a page table
- The operating system automatically loads and unloads pages with the help of page faults and working sets.
- A working set is a list of the pages actively being used by the program
- This set helps the operating system decide which pages of memory to flush



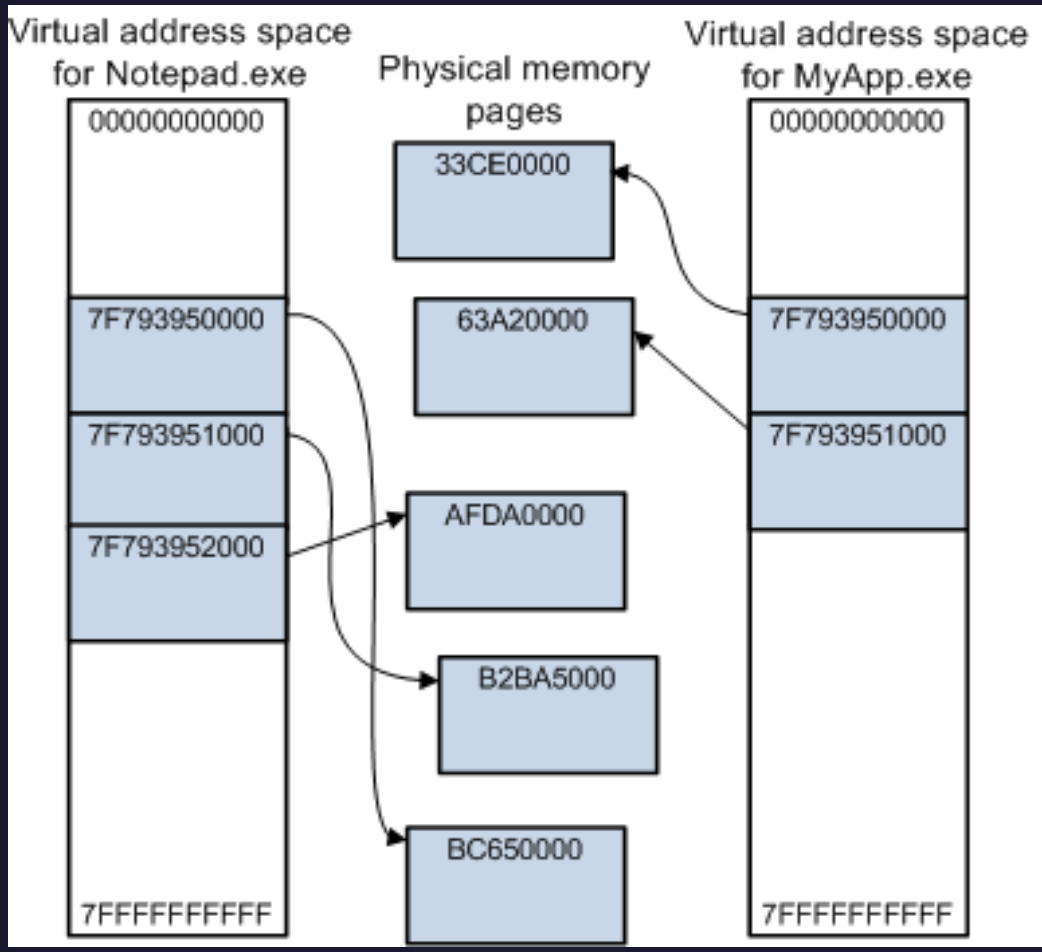
Page Faults

- A page fault is an error that is thrown when the program attempts to access memory at a region where there is no valid page table entry.
- When a page fault occurs, the operating system simply finds the page in the page file and loads it back into memory so the process can use it.



Virtual Memory

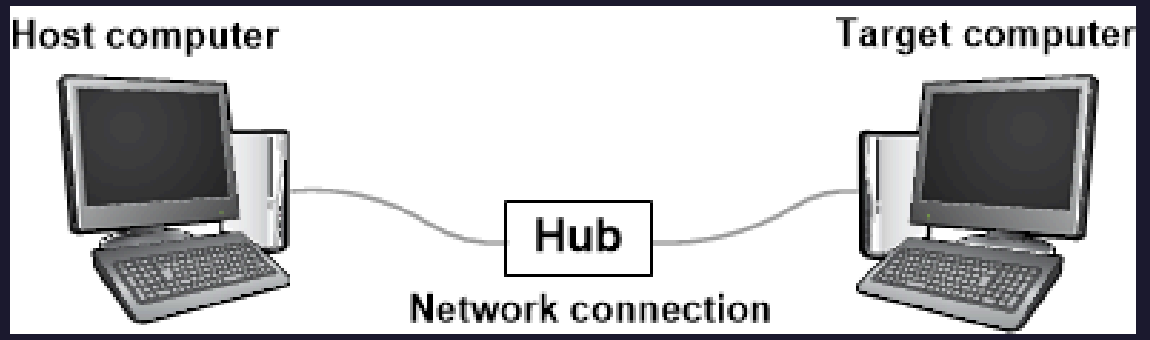
- Virtual memory is a Windows concept designed to isolate and protect the programs and data on your computer.
- Each process in user mode is given a virtual address space and allowed to operate freely within that space. It may not leave that space.
- The kernel has one virtual address space that it shares. (System Space)



Kernel Debugging

- Overview
- Set up the Debugger
- Start Debugging!

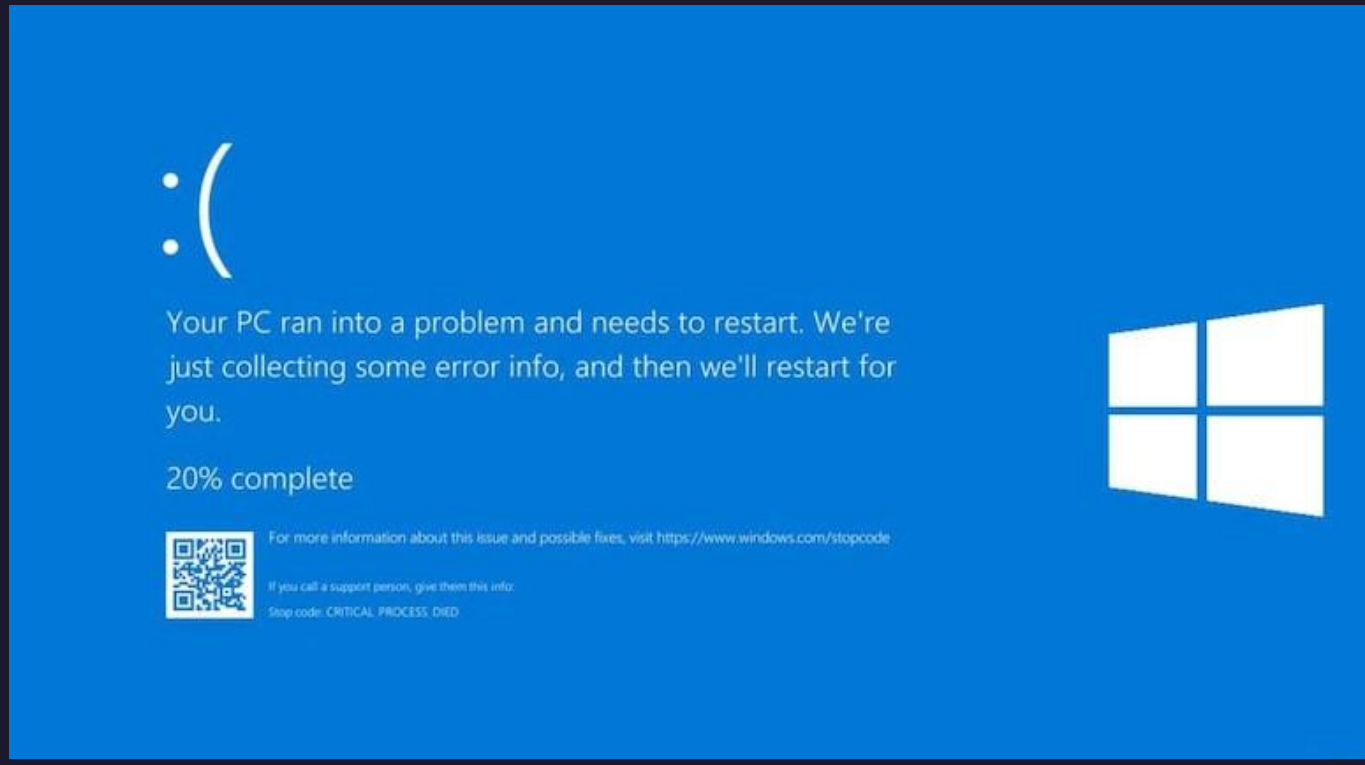
Overview



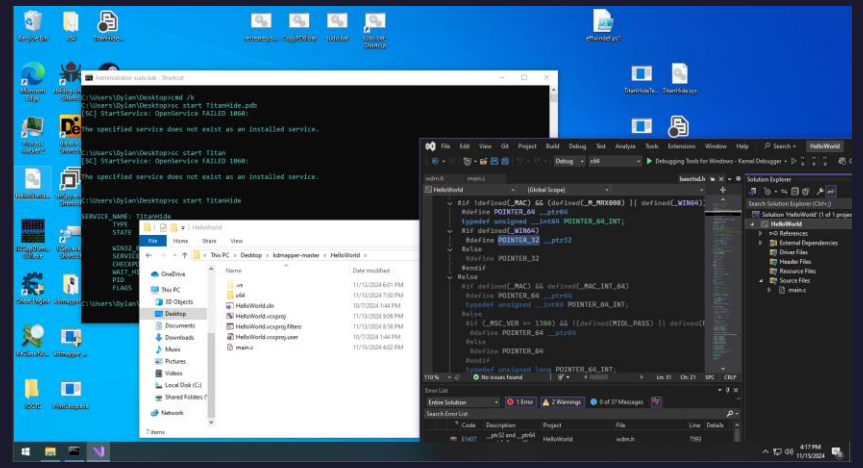
- The concept of kernel debugging is simple
- Instead of attaching a debugger to a program, you attach it to the operating system
- The debuggee is connected to the debugger through a cable or network.

Overview Cont.

- The debugger sends commands over the connection to tell the debuggee what to do.
- The computer you are testing your driver on will not be able to recover if your driver breaks. Having a debugger allows to see what went wrong before it crashes.



Start debugging



- For today I will be using WinDbg to debug my guest.
- I have created a virtual machine with VMware and configured it for kernel debugging.
- Once my guest boots, the debugger will attach and I have kernel debug access

Common Uses

- There are many reasons to write a kernel driver, but the following are common ones
  - To create a way to interface with hardware.
  - Write software that can bypass user mode Anti-Debug checks (TitanHide)
  - Increase performance of certain tasks. (Certain I/O ops are faster at the kernel level)
  - Direct Memory Access (DMA)
  - To write a rootkit (A.K.A. Kernel level Anti-Cheat)

Writing Drivers

- There are two types of Windows drivers, user mode (UMDF) and kernel mode (KMDF)
  - UMD are safer, because if they crash, they don't bring down the system
  - KMD have more access to the computer and don't have as much overhead compared to UMD. But if you crash a KMD, you bluescreen.
- Then you choose what kind of driver, hardware or software.
  - Hardware drivers interface with things connected to the computer. (Keyboards, mice)
  - Software drivers run code at the kernel level for a various number of reasons but typically because it's fast.

My Driver.

- For this demo I will be using the kernel mode driver I uploaded to my GitHub.
- No, it's not pretty, but it gets the job done
- You can modify the var2 variable to change control flow.
- You must disable a lot of security features to get this to work, so just follow what I'm doing for now.
- https://github.com/dcharnic/Windows-Kernel-Driver-Talk

How to use the driver.

- Most drivers are designed to be set up with sc.
- For my driver, because I want to be able to easily load and unload it, I am using KdMapper
- Because this is a memory manipulation driver, we'll want to manipulate the memory of something.
- PrintLoop.exe is our target
- PrintLoop has a value, and as long as that value doesn't change, the program is happy

Final Experiment.

- Start by launching PrintLoop.exe
- Replace the pid and address in the HelloWorld driver with the pid and address output by PrintLoop
- Build the HelloWorld Driver
- Run the driver with kdmapper.exe HelloWorld.sys
- Step past the int3 with windbg
- If everything worked, PrintLoop should be compromised.

Why is any of this important?

- As many of you have probably noticed, running these drivers requires you to disable a lot of security features, and you must already have access to an admin account.
- These drivers with this setup are designed to bypass anti-debug checks in programs running in user mode on your computer.

Why is any of this important? (Cont)

- This is where some companies have started to get smart with their protections.
- The problem with this is when you write a bad kernel driver, you get bluescreens. In the case of a company like CrowdStrike, you get 8.5 million of them.
- The problem with kernel level protections, is that these companies have access to EVERYTHING on your computer to ensure you aren't messing with their software.
- This is a security risk because each company is now responsible for maintaining secure rootkit level access to millions of computers.