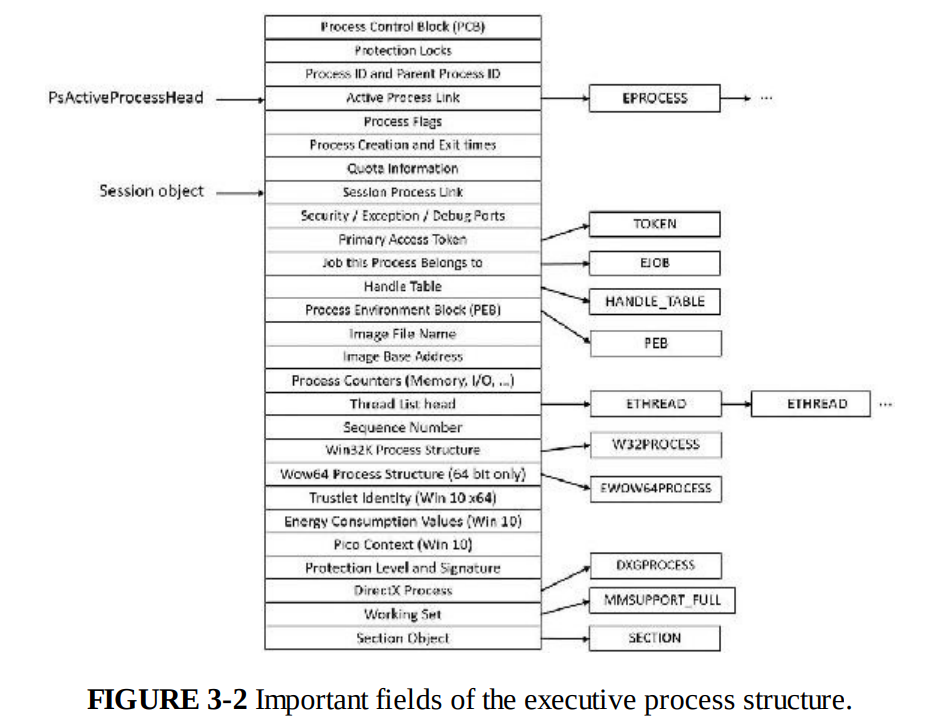
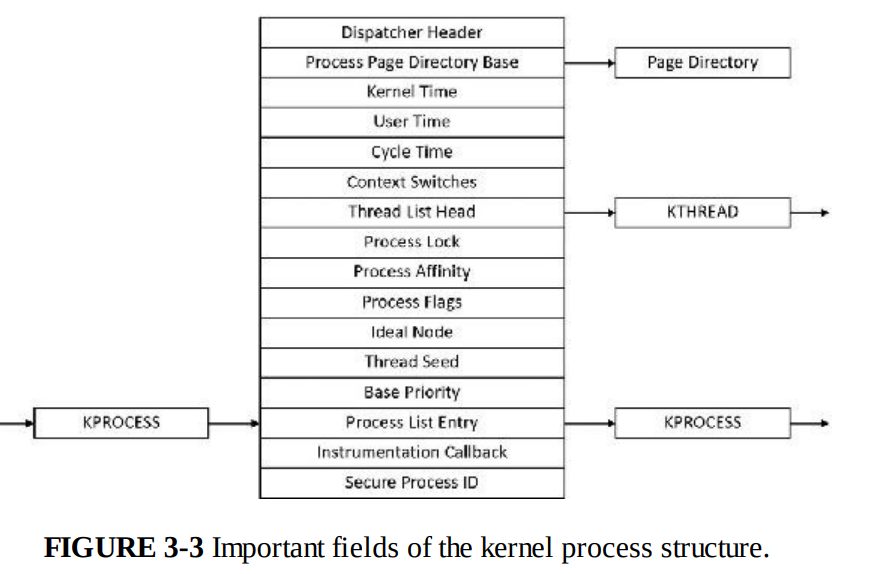
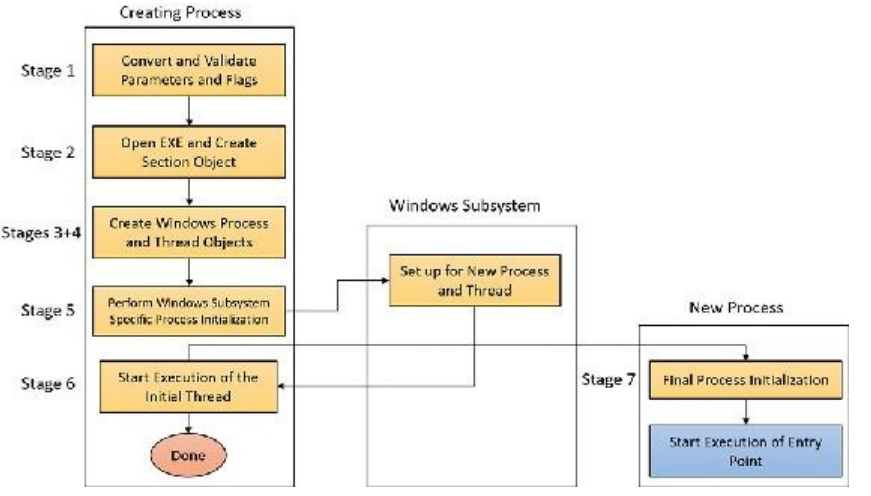
1. This is what happens when CreateProcess is called from kernel32.dll:
   1. (Kernel32.dll) CreateProcess -> CreateProcessInternal
   2. (NtDll.dll) NtCreateUserProcess
   3. (Kernel mode) NtCreateUserProcess
   4. (Kernel mode) PspAllocateProcess -> PspInsertProcess
2. Each Windows Process is represented by an EPROCESS structure
   1. The EPROCESS and most of its related date structures exist in system address space
   2. Exceptions:
      1. Process Environment Block (PEB) which exists in user address space
      2. Working set list (valid only within context of current process)
3. Every EPROCESS structure is encapsulated as a process object by the executive object manager

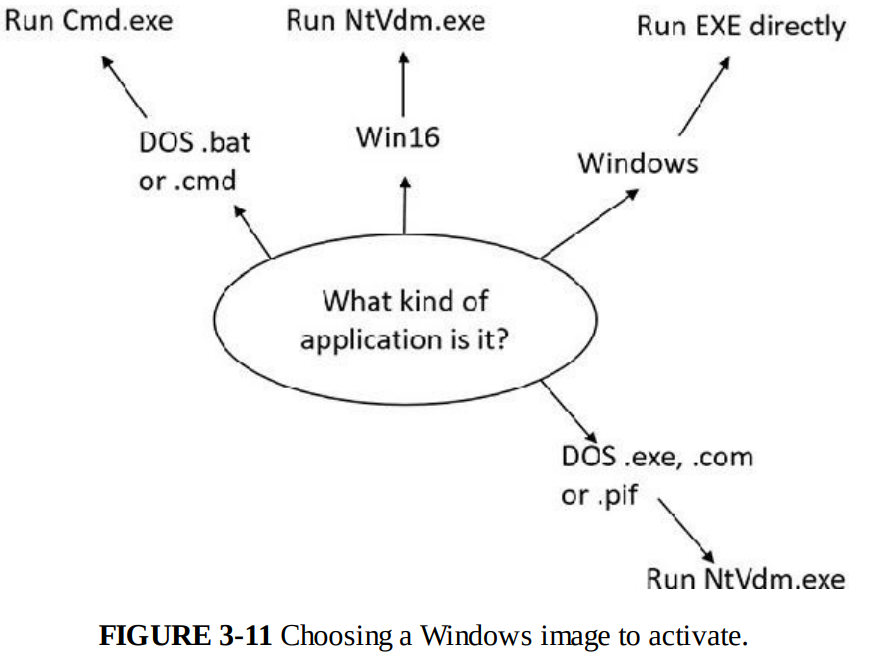


1. A handle to a process provides access to some of the data in the EPROCESS structure
2. The first member of the executive process structure is called Pcb (Process Control Block)
   1. It is a structure of type KPROCESS (Kernel Process)



# Overview of Process Creation





Stage 2:

* Kernel32.dll actually has no idea at this point whether the application image name is a real Windows Application or a batch file , 16-bit or DOS.
* This decision to run/decide is in CreateProcessInternalW

Stage 3:

* At this point, NtCreateUSerProcess has opened a valid Windows Executable file
* And created a section object to map it into the new process address space
* Thern, creates a Windows Executive Process Object to run the image by calling PspAllocateProcess
  + Setting up the EPROCESS object
  + Creating initial process address space
  + Initializing KPROCESS structure
  + Concluding set up
  + Setting up PEB

# Image Loader

* Image loader lives in the usermode system NtDll.dll
* Parsing import table of the application to look for all DLLs that it requires and then follow the parsing of the export table of DLL to make sure functions are actually present
* After process has been created, loader calls NtContinue special native API to continue execution
* Loader maintains a list of all modules (DLLs and primary exe) that are loaded, stored in the PEB structure and as a substructure PEB\_LDR\_DATA

Maintains 3 doubly linked lists, containing structures called loader data table entries

* The export table of an imported DLL can use a forwarder entry, meaning that the actual function is implemented in another DLL. Loader goes back to load the DLL.

Useful Commands:

* dt nt!\_eprocess
* dt nt!\_kprocess
* !process 0 0
* !proces