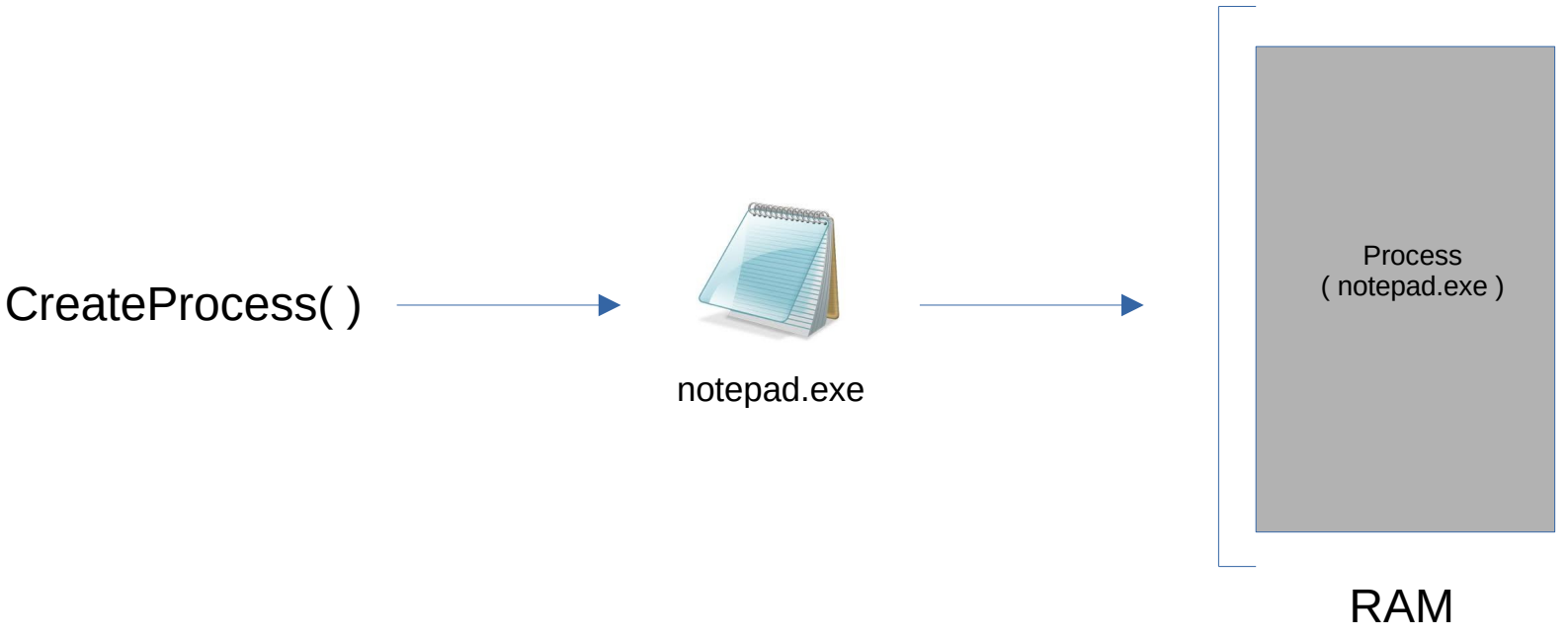
Creating Process

Creating Process



Function Prototype

```
BOOL CreateProcess(
 lpApplicationName,        // Path to the executable
lpCommandLine,             // Command-line arguments
lpProcessAttributes,     // Security attributes for the process
lpThreadAttributes,      // Security attributes for the primary thread
bInheritHandles,           // Inherit handles from parent process
dwCreationFlags,           // Process creation flags
lpEnvironment,             // Pointer to environment block
lpCurrentDirectory,        // Working directory of the new process
lpStartupInfo,         // Pointer to STARTUPINFO structure
lpProcessInformation     // Pointer to PROCESS_INFORMATION structure
);
```

Function Prototype

```
BOOL CreateProcess(
 lpApplicationName,        // Path to the executable
lpCommandLine,             // Command-line arguments
lpProcessAttributes,     // Security attributes for the process
lpThreadAttributes,      // Security attributes for the primary thread
bInheritHandles,           // Inherit handles from parent process
dwCreationFlags,           // Process creation flags
lpEnvironment,             // Pointer to environment block
lpCurrentDirectory,        // Working directory of the new process
lpStartupInfo,         // Pointer to STARTUPINFO structure
lpProcessInformation     // Pointer to PROCESS_INFORMATION structure
);
```

Flag	Description
CREATE_NEW_CONSOLE	Creates a new console window for the child process.
CREATE_SUSPENDED	Creates the process in a suspended state. (Used in process injection.)
CREATE_NO_WINDOW	Runs the process without showing any window (used for stealth malware).
CREATE_DEFAULT_ERROR_MODE	Uses the parent's error mode.
CREATE_BREAKAWAY_FROM_JOB	Allows the process to run outside of a job object.
CREATE_SEPARATE_WOW_VDM	Runs the process in a separate virtual DOS machine (for 16-bit apps).
CREATE_SHARED_WOW_VDM	Runs in a shared DOS environment (legacy stuff).
DEBUG_PROCESS	Allows the parent process to debug the child process.
DEBUG_ONLY_THIS_PROCESS	Only allows the parent to debug this process.
DETACHED_PROCESS	Runs the process without associating with the parent console.
EXTENDED_STARTUPINFO_PRESENT	Allows passing extended startup information.
INHERIT_PARENT_AFFIINITY	Inherits the CPU affinity of the parent process.
CREATE_PROTECTED_PROCESS	Creates a protected process (used in anti-debugging).

Function Prototype

```
BOOL CreateProcess(
 lpApplicationName,        // Path to the executable
lpCommandLine,             // Command-line arguments
lpProcessAttributes,     // Security attributes for the process
lpThreadAttributes,      // Security attributes for the primary thread
bInheritHandles,           // Inherit handles from parent process
dwCreationFlags,           // Process creation flags
lpEnvironment,             // Pointer to environment block
lpCurrentDirectory,        // Working directory of the new process
lpStartupInfo,         // Pointer to STARTUPINFO structure
lpProcessInformation     // Pointer to PROCESS_INFORMATION structure
);
```

Function Prototype

```
BOOL CreateProcess(
 lpApplicationName,        // Path to the executable
lpCommandLine,             // Command-line arguments
lpProcessAttributes,     // Security attributes for the process
lpThreadAttributes,      // Security attributes for the primary thread
bInheritHandles,           // Inherit handles from parent process
dwCreationFlags,           // Process creation flags
lpEnvironment,             // Pointer to environment block
lpCurrentDirectory,        // Working directory of the new process
lpStartupInfo,         // Pointer to STARTUPINFO structure
lpProcessInformation     // Pointer to PROCESS_INFORMATION structure
);
```

It controls how the new process starts (window size, position, appearance, etc.).

```
typedef struct _STARTUPINFO {
  DWORD cb;               // Size of the structure
  LPSTR lpReserved;       // Reserved (always NULL)
  LPSTR lpDesktop;        // Desktop name (NULL = default)
  LPSTR lpTitle;          // Console title (NULL = default)
  DWORD dwX, dwY;         // Window position (ignored if not set)
  DWORD dwXSize, dwYSize; // Window size
  DWORD dwFlags;          // specify which flag is used (e.g., SW_HIDE)
  WORD wShowWindow;       // Controls the window state (SW_SHOW, SW_HIDE, etc.)
  ... (Other fields not commonly used)
} STARTUPINFO;
```

Function Prototype

```
BOOL CreateProcess(
 lpApplicationName,        // Path to the executable
lpCommandLine,             // Command-line arguments
lpProcessAttributes,     // Security attributes for the process
lpThreadAttributes,      // Security attributes for the primary thread
bInheritHandles,           // Inherit handles from parent process
dwCreationFlags,           // Process creation flags
lpEnvironment,             // Pointer to environment block
lpCurrentDirectory,        // Working directory of the new process
 lpStartupInfo,          // Pointer to STARTUPINFO structure
lpProcessInformation     // Pointer to PROCESS_INFORMATION structure
);
```

It controls how the new process starts (window size, position, appearance, etc.).

```
typedef struct _STARTUPINFO {
  DWORD cb;               // Size of the structure
  LPSTR lpReserved;       // Reserved (always NULL)
  LPSTR lpDesktop;        // Desktop name (NULL = default)
  LPSTR lpTitle;          // Console title (NULL = default)
  DWORD dwX, dwY;         // Window position (ignored if not set)
  DWORD dwXSize, dwYSize; // Window size
  DWORD dwFlags;          // specify which flag is used (e.g., SW_HIDE)
  WORD wShowWindow;       // Controls the window state (SW_SHOW, SW_HIDE, etc.)
  ... (Other fields not commonly used)
} STARTUPINFO;
```

FLAGS

1.	**STARTF_USESHOWWINDOW**	(0X00000001)
2.	STARTF_USESIZE	(0X00000002)
3.	STARTF_USEPOSITION	(0X00000004)
4.	STARTF_USECOUNTCHARS	(0X00000008)
5.	STARTF_USEFILLATTRIBUTE	(0X00000010)
6.	STARTF_RUNFULLSCREEN	(0X00000020)
7.	STARTF_FORCEONFEEDBACK	(0X00000040)
8.	STARTF_FORCEOFFFEEDBACK	(0X00000080)
9.	**STARTF_USESTDHANDLES**	(0X00000100)
10.	STARTF_USEHOTKEY	(0X00000200)

Function Prototype

```
BOOL CreateProcess(
 lpApplicationName,        // Path to the executable
lpCommandLine,             // Command-line arguments
lpProcessAttributes,     // Security attributes for the process
lpThreadAttributes,      // Security attributes for the primary thread
bInheritHandles,           // Inherit handles from parent process
dwCreationFlags,           // Process creation flags
lpEnvironment,             // Pointer to environment block
lpCurrentDirectory,        // Working directory of the new process
lpStartupInfo,         // Pointer to STARTUPINFO structure
lpProcessInformation    // Pointer to PROCESS_INFORMATION structure
);
```

Holds important information about the newly created process.

```
typedef struct _PROCESS_INFORMATION {
  HANDLE hProcess;   // Handle to the process
  HANDLE hThread;    // Handle to the main thread
  DWORD dwProcessId; // Process ID
  DWORD dwThreadId;  // Thread ID
} PROCESS_INFORMATION;
```