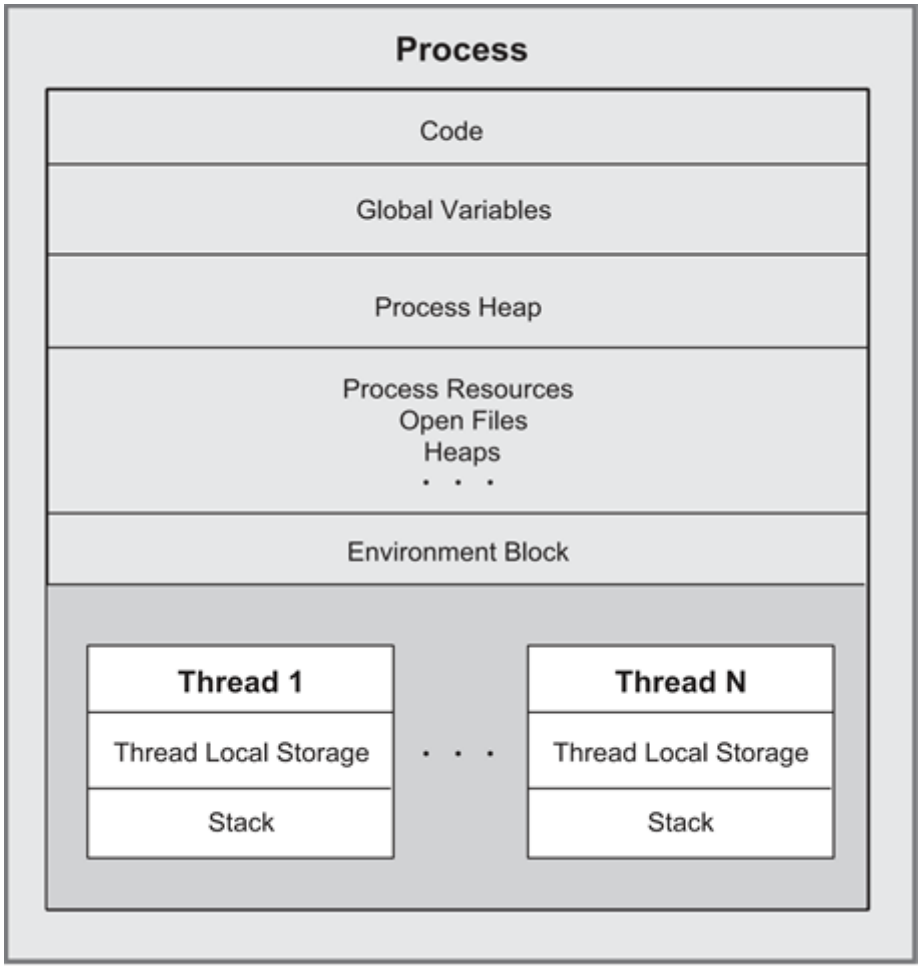
## Proces ve Windows

Proces obsahuje:

- jedno nebo více vláken
- vlastní virtuální adresní prostor
- kód
- haldu procesu (zajišťuje přidělování paměti)
- různé zdroje (například další haldy)
- informace o prostředí (aktuální cesta atd.)




Vytvoření procesu zajišťuje funkce *CreateProcess*:

```
BOOL WINAPI CreateProcess(
  _In_opt_    LPCTSTR lpApplicationName,
  _Inout_opt_ LPTSTR lpCommandLine,
  _In_opt_    LPSECURITY_ATTRIBUTES lpProcessAttributes,
  _In_opt_    LPSECURITY_ATTRIBUTES lpThreadAttributes,
  _In_        BOOL bInheritHandles,
  _In_        DWORD dwCreationFlags,
  _In_opt_    LPVOID lpEnvironment,
  _In_opt_    LPCTSTR lpCurrentDirectory,
```

```
    _In_        LPSTARTUPINFO lpStartupInfo,
    _Out_       LPPROCESS_INFORMATION lpProcessInformation
);
```

`CreateProcessA` – verze funkce používající kódování ASCII

`CreateProcessW` – verze funkce používající kódování Unicode

Parametry funkce:

| | |
|---|---|
| `lpApplicationName` | Jméno programu včetně přípony nebo cesta k programu.<br>Parametr může být *NULL*, pak jméno programu nebo cesta k programu musí být uvedeny v parametru *lpCommandLine*. |
| `lpCommandLine` | Příkazový řádek.<br>Je-li uveden parametr *lpApplicationName*, uvedeme zde jen parametry pro volaný program nebo *NULL*, nemá-li parametry. |
| `lpProcessAttributes` | `NULL` (implicitní hodnota *security* parametru) |
| `lpThreadAttributes` | `NULL` (implicitní hodnota *security* parametru) |
| `bInheritHandles` | `FALSE` (*handles* nebudou děděny) |
| `dwCreationFlags` | **`CREATE_SUSPENDED`** – primární vlákno nového procesu je vytvořeno v pozastaveném stavu a nebude spuštěno, dokud nebude zavolána funkce `ResumeThread`.<br>**`CREATE_NEW_CONSOLE`** – nový proces bude mít novou konzoli místo toho, aby (implicitně) zdědil konzoli volajícího procesu.<br>**`DETACHED_PROCESS`** – je-li nový konzolový proces volán z konzolového procesu, (implicitně) nezdědí jeho konzoli. Tu si může později vytvořit funkcí `AllocConsole`. Tuto hodnotu nelze použít současně s `CREATE_NEW_CONSOLE`.<br>**`ABOVE_NORMAL_PRIORITY_CLASS`** – proces má prioritu nad `NORMAL_PRIORITY_CLASS` a pod `HIGH_PRIORITY_CLASS`.<br>**`BELOW_NORMAL_PRIORITY_CLASS`** – proces má prioritu nad `IDLE_PRIORITY_CLASS` a pod `NORMAL_PRIORITY_CLASS`.<br>**`HIGH_PRIORITY_CLASS`** – vysoká priorita pro časově kritické procesy. Použít jen v nutných |

| | |
|---|---|
| | případech, neboť aplikace s charakterem zpracování jen procesorem může spotřebovat téměř všechen strojový čas.<br>**IDLE_PRIORITY_CLASS** – proces, který běží, jen když systém není využit jiným procesem.<br>**NORMAL_PRIORITY_CLASS** – proces s běžnou prioritou.<br>**REALTIME_PRIORITY_CLASS** – nejvyšší možná priorita určená pro důležité zpracování. Pokud proces s touto prioritu běží déle než kratší interval, některé části systém (například myš) mohou stagnovat. |
| `lpEnvironment` | `NULL` (blok prostředí volaného procesu bude stejný, jako má volající proces) |
| `lpCurrentDirectory` | `NULL` (aktuální adresář volaného procesu bude stejný, jako má volající proces) |
| `lpStartupInfo` | Ukazatel na strukturu `STARTUPINFO` nebo `STARTUPINFOEX`.<br>Při použití struktury `STARTUPINFOEX` uveďte hodnotu `EXTENDED_STARTUPINFO_PRESENT` v parametru *dwCreationFlags*.<br>Uvedené *handles* v `STARTUPINFO` nebo `STARTUPINFOEX` zavřeme funkcí `CloseHandle`, nejsou-li již zapotřebí. |
| `lpProcessInformation` | Ukazatel na strukturu `PROCESS_INFORMATION`, ve které volající proces dostane informace o novém procesu.<br>*Handles* v `PROCESS_INFORMATION` zavřeme funkcí `CloseHandle`, nejsou-li již zapotřebí. |

```
BOOL WINAPI CloseHandle(_In_ HANDLE hObject);
```

Návratová hodnota funkce:
Funkce vrací `TRUE`, když vytvoření procesu bylo úspěšné. Pokud vrátí `FALSE`, lze kód chyby zjistit funkcí `GetLastError`.

Struktura, ve které volaný proces vrací informace:

```
typedef struct _PROCESS_INFORMATION {
    HANDLE hProcess;
    HANDLE hThread;
    DWORD  dwProcessId;
```

```
    DWORD  dwThreadId;
} PROCESS_INFORMATION, *LPPROCESS_INFORMATION;
```

---

Vyčkání na ukončení volaného procesu:

```
DWORD WINAPI WaitForSingleObject(
  _In_ HANDLE hHandle,
  _In_ DWORD dwMilliseconds
);
```

Za parametr `dwMilliseconds` lze rovněž dosadit argument `INFINITE`, pak funkce čeká neomezenou dobu.

Návratová hodnota funkce:

| | |
|---|---|
| `WAIT_OBJECT_0` | Přišel signál z daného objektu. |
| `WAIT_TIMEOUT` | Časový interval uplynul, aniž přišel signál z daného objektu. |
| `WAIT_FAILED` | Funkce selhala. Bližší informace o chybě lze získat funkcí `GetLastError`. |

---

V případě výskytu chyby lze dle čísla chyby její popis najít na stránkách
https://msdn.microsoft.com/en-us/library/windows/desktop/ms681381%28v=vs.85%29.aspx

nebo lze následující funkcí zjistit slovní popis chyby

```
const char *LastErrorMessage()
{ static char m[256];
  if (FormatMessageA(FORMAT_MESSAGE_FROM_SYSTEM,NULL,
      GetLastError(),
      MAKELANGID(LANG_NEUTRAL,SUBLANG_DEFAULT),
      m,256,NULL)==0)
  sprintf(m,"Kód chyby: %u.",GetLastError());
  return m;
}
```

V případě, kdy používáme kódování Unicode, je zapotřebí funkci upravit na Unicode (*FormatMessageA* nahradit funkcí *FormatMessageW* atd.).

---

Některé funkce API:

```
LPTSTR WINAPI GetCommandLine();
```

```
DWORD WINAPI GetTickCount();
```
– vrací počet milisekund uplynulých od spuštění systému

`BOOL WINAPI SwitchToThread();` – přepne na jiné čekající vlákno (je-li nějaké)

`VOID WINAPI Sleep(_In_ DWORD dwMilliseconds);` – zastaví na daný počet milisekund vykonávání vlákna

---

Definice funkcí API jsou v hlavičkovém souboru:

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

---

***Nastavení ASCII kódování v projektu vývojového systému VS***

Volbu

`Project` → *`jméno_projektu`* `Properties` → `Advanced` → `Character Set`

nastavíme na `Use Multi-Byte Character Set`

***Deaktivace Security Check***

Volbu

`Project` → *`jméno_projektu`* `Properties` → `C/C++ Code Generation` → `Security Check`

nastavíme na `Disable Security Check`

---

***Přidání dalšího projektu ke stávajícímu projektu***

K projektu přidáme další projekt volbou

`File` → `Add` → `New Project` → `C++` → `Console App`

`Project Name`: … *vložíme jméno dalšího projektu*

stiskneme `Create`