

Windows API a základní práce s procesy

8

Operační systém Microsoft Windows poskytuje velmi širokou škálu funkcí, které mohou programy pro svůj běh využívat, počínaje základní prací se systémem, jako je práce s procesy a soubory, přes komunikaci po síti, a tvorbou grafických aplikací konče. Tato funkcionalita se souhrnně označuje jako Windows API (nebo zkráceně WinAPI) a je poskytována jako sada funkcí jazyka C. Práce s tímto rozhraním má svá specifika, která si v tomto cvičení ukážeme.

1 Typový systém WinAPI

Rozhraní WinAPI využívá vlastní sadu datových typů, které narozdíl od jazyka C mají jednoznačně definované velikosti a rozsahy hodnot.

Výčet nejčastěji používaných typů ukazuje Tabulka 1.

typ	význam
DWORD	32bitové neznaménkové celé číslo
WORD	16bitové neznaménkové celé číslo
BYTE	8bitové neznaménkové celé číslo
BOOL	pravdivostní hodnota
VOID	neplatná hodnota
LPDWORD	ukazatel na hodnotu typu DWORD
LPWORD	ukazatel na hodnotu typu WORD
LPBYTE	ukazatel na hodnotu typu BYTE
LPBOOL	ukazatel na hodnotu typu BOOL
LPVOID	ukazatel na hodnotu typu VOID
LPSTR	ukazatel na hodnotu typu char, řetězec obsahující jednobytové znaky (ANSI)
LPWSTR	ukazatel na hodnotu typu wchar_t, řetězec obsahující jednobytové znaky (UNICODE)
HANDLE	obecný identifikátor objektu (technicky void *)

Tabulka 1: Přehled nejčastěji používaných typů ve WinAPI.

Tento výčet není ani v nejmenším úplný, další typy lze nalézt v dokumentaci.¹ Může se stát, že datové typy, které používá WinAPI, nebudou zcela kompatibilní s typovým systémem jazyka C/C++. Zejména se to týká novějších verzí překladačů, které jsou při práci s datovými typy striktnější. V takovém případě je nutné upravit nastavení projektu a povolit benevolentnější práci s typy a to následovně: *Project* → *Properties* → *C/C++* → *Language* → *Conformance Mode* → *Default*.

¹<https://docs.microsoft.com/en-us/windows/win32/winprog/windows-data-types>

2 Práce s řetězci

Další úskalí, které WinAPI přináší, spočívá v práci s řetězci. WinAPI umožňuje pracovat s dvěma typy řetězců, které jsou označovány jako *ANSI* a *Unicode*. První typ řetězců používá pro reprezentaci znaků jednobytové hodnoty (typ `char`) a druhý typ používá „široké znaky“ o velikost dva byty (typ `wchar_t`).

Pracuje-li některá z funkcí WinAPI s řetězci, je obvykle nabízena ve dvou variantách. Buď s příponou *A*, pokud pracuje s ANSI řetězci, nebo s příponou *W* pracuje-li s „Unicode“ řetězci, např. `CreateFileA` (ANSI) nebo `CreateFileW` (Unicode). Při programování se reálně používá makro `CreateFile`, které se podle nastavení projektu expanduje na `CreateFileA` nebo `CreateFileW`.

Standardně jsou k dispozici funkce pro práci s oběma typy řetězců a programátor má na výběr ze dvou možností: (i) Buď si zvolit jeden typ řetězců, nastavit jej v projektu, a ten konzistentně v rámci jedné aplikace používat nebo (ii) použít pomocné sady maker v hlavičkovém souboru `tchar.h`, které se expandují na daný typ funkcí pro práci s řetězci podle nastavení projektu. Toto řešení je univerzálnější, ale zápis kódu se bude lišit od tradičního kódu v jazyce C.

Rozdíly při práci s jednotlivými typy znaků nastiňuje Tabulka 2.

	ANSI	Unicode	tchar.h
typ znaku	<code>char</code>	<code>wchar_t</code>	<code>_TCHAR</code>
řetězcový literál	<code>"abc"</code>	<code>L"abc"</code>	<code>_T("abc")</code>
hlavní funkce	<code>main</code>	<code>wmain</code>	<code>_tmain</code>
délka řetězce	<code>strlen</code>	<code>wcslen</code>	<code>_tcslen</code>
kopie řetězce	<code>strcpy</code>	<code>wscpy</code>	<code>_tcscopy</code>
spojení řetězců	<code>strcat</code>	<code>wscat</code>	<code>_tcscat</code>
porovnání řetězců	<code>strcmp</code>	<code>wscmp</code>	<code>_tcscmp</code>
formátovaný výstup	<code>printf</code>	<code>wprintf</code>	<code>_tprintf</code>

Tabulka 2: Práce s řetězci různých typů

Jaký typ znaků bude použit lze nastavit ve vlastnostech projektu: *Project* → *Properties* → *Advanced* → *Character Set*.

3 Rozhraní pro práci se soubory

Základní principy práce s WinAPI a to, jak pracovat s řetězci univerzálním způsobem, si ukážeme na práci se soubory. Pro práci se soubory má Windows vlastní sadu funkcí.² Klíčovou funkcí je `CreateFile`, která slouží k vytvoření nebo otevření souboru.³

Funkce `CreateFile`⁴ jako svůj argument bere cestu k souboru a další příznaky, jak se souborem pracovat. Jedná se jednak o režim přístupu k souboru (zda jej chceme číst nebo do něj zapisovat), režim souběžného

²<https://docs.microsoft.com/en-us/windows/win32/api/fileapi/>

³Název `CreateFile` přirozeně asociuje vytvoření souboru, proto může být matoucí, že funkce slouží i k otevření existujícího souboru. Ve skutečnosti se slovo *Create* vztahuje k vytvoření objektu, který reprezentuje soubor, a v tomto smyslu je slovo *Create* používáno konzistentně i v dalších částech WinAPI.

⁴<https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfile>

přístupu, zda může být otevřený soubor otevřen ještě jednou (např. pro čtení), jak se zachovat, pokud soubor existuje nebo neexistuje. Další příznaky, jako bezpečnostní atributy, atributy souboru nebo šablonu souboru nebudeme využívat. Otevření nebo vytvoření souboru pro zápis i čtení ilustruje následující kód.

```
1 #include <windows.h>
2 #include <tchar.h>
3 #include <stdio.h>
4 int _tmain(int argc, TCHAR* argv[])
5 {
6     HANDLE hFile = CreateFile(
7         _T("foo.txt"),           // cesta k souboru
8         GENERIC_READ | GENERIC_WRITE, // režim práce se souborem
9         FILE_SHARE_READ,        // režim sdíleného přístupu k souboru (pro čtení)
10        NULL,                    // bezpečnostní atributy
11        OPEN_ALWAYS,            // jak naložit s (ne)existujícími soubory
12        0,                      // příznaky souboru
13        NULL);                  // odkaz na šablonu
14
15    if (hFile == INVALID_HANDLE_VALUE) {
16        _tprintf(_T("error opening file\n"));
17        return -1;
18    }
19
20    _TCHAR* str = _T("Hello world!");
21    if (WriteFile(hFile, str, sizeof(_TCHAR) * _tcslen(str), NULL, NULL))
22        _tprintf(_T("ok"));
23
24    CloseHandle(hFile);
25    return 0;
26 }
```

Všimněme si, že návratovou hodnotou funkce `CreateFile` je hodnota typu `HANDLE`. Jedná se o unikátní identifikátor (v rámci spuštěného procesu) pro jednotlivé objekty poskytované jádrem operačního systému. Datový typ `HANDLE` se používá i pro práci s dalšími objekty, např. procesy, vlákny, synchronizačními objekty atd. Technicky se jedná o typ `void *`, ale správně bychom tuto vlastnost neměli předjímat.

Pokud operace `CreateFile` z nějakého důvodu selže, je jako návratová hodnota vrácena konstanta `INVALID_HANDLE_VALUE`. Podrobnosti o chybě můžeme zjistit pomocí funkce `GetLastError()`.

V ukázkovém kódu je na řádce 21 ukázán zápis do souboru pomocí funkce `WriteFile`.⁵ Svým chováním se funkce příliš neliší od funkce `fwrite` ze standardní knihovny jazyka C, má navíc dva argumenty. Jedním je možné získat počet zapsaných bytů (předáváme ukazatel na hodnotu, kam se výsledek zapíše), druhým je struktura umožňující asynchronní práci se souborem, což nebudeme využívat.

⁵<https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-writefile>

Práci se souborem je nutné ukončit jeho uzavřením, k tomu slouží funkce `CloseHandle`. Tato funkce je obecná a slouží k práci i s dalšími objekty.

Úkoly

1. Spusťte program a podívejte se do vytvořeného souboru, ideálně s textovým editorem nebo prohlížečem, který umí soubor zobrazit v hexadecimální podobě (např. Total Commander).
2. Změňte nastavení znaků v projektu a opakujte krok 1.
3. Upravte program tak, aby přečetl obsah vámi zvoleného souboru a vypsal jej na terminál. Použijte funkci `ReadFile`.⁶

4 Vytvoření nového procesu

Vytvoření procesu je ve WinAPI přímočaré. Funkci `CreateProcess`⁷ předáme název (resp. cestu) ke spouštěnému programu, argumenty, a další příznaky či odkazy na struktury, kterými se vytvoření souboru má řídit. Důležité jsou dvě struktury `STARTUPINFO` a `PROCESS_INFORMATION`. První struktura slouží k předání doplňujících informací pro spouštěný proces (např. pozice okna). Druhá struktura slouží k předání informací o vytvořeném procesu, např. handle na vytvořený proces. Při inicializaci struktury `STARTUPINFO` je nutné do atributu `cb` zadat velikost struktury v bytech.

```
#include <windows.h>
#include <tchar.h>
#include <stdio.h>
int _tmain(int argc, TCHAR* argv[])
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;

    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));

    // vytvoři nový proces
    if (!CreateProcess(
        _T("C:\\WINDOWS\\system32\\notepad.exe"), // cesta k programu
        _T("C:\\WINDOWS\\system32\\notepad.exe foo.txt"), // úplný seznam argumentů
        NULL, // nastavení bezpečnostních atributů
        NULL, // nastavení bezpečnostních atributů
        FALSE, // zakázání dělení handle mezi procesy
        0, // příznaky pro vytvoření procesu
        NULL, // definice prostředí (použije se prostředí aktuálního procesu)
```

⁶<https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-readfile>

⁷<https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa>

```

    NULL,          // pracovní adresar (použije se adresar aktuálního procesu)
    &si,           // ukazatel na strukturu s informacemi ke spuštěnému procesu
    &pi)          // ukazatel na strukturu, kterou jsou předány informace o vzniklém procesu
) {
    _tprintf(_T("CreateProcess failed (%d).\n"), GetLastError());
    return 1;
}

// čeka na ukončení procesu
WaitForSingleObject(pi.hProcess, INFINITE);

// uzavře proces
CloseHandle(pi.hProcess);
CloseHandle(pi.hThread);
return 0;
}

```

Úkol:

4. Rozšířte ukázkový příklad o volání funkce `GetExitCodeProcess`,⁸ která vrací návratový kód ukončeného procesu, a tento kód vypišete.

V rámci operačního systému má každý proces přiřazené číslo (*process id* nebo zkráceně *pid*). Pomocí funkce `OpenProcess`⁹ můžeme získat handle na existující proces na základě tohoto identifikátoru a to například následovně:

```
HANDLE hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pid);
```

Pokud máme dostatečná oprávnění, můžeme s procesem dále pracovat, například zjistit o něm informace nebo jej ukončit.

Úkol:

5. Spusťte vhodnou aplikaci (notepad, kalkulačku), v TaskManageru zjistěte jeho pid.
6. Pomocí funkce `GetProcessImageFileName`¹⁰ zjistěte cestu ke spuštěnému souboru.
7. Pomocí funkce `TerminateProcess`¹¹ jej ukončete.

⁸<https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-getexitcodeprocess>

⁹<https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocess>

¹⁰<https://docs.microsoft.com/en-us/windows/win32/api/psapi/nf-psapi-getprocessimagefilename>

¹¹<https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-terminateprocess>