KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Windows API a základní práce s procesy

KMI/OS1 Operační systémy I

Mgr. Markéta Trnečková, Ph.D.
www.marketa-trneckova.cz

Windows API a základní práce s procesy

Učební text ke cvičení:
https://phoenix.inf.upol.cz/~krajcap/courses/2025LS/OS1/tutorial08.pdf

Windows API a základní práce s procesy

- OS Microsoft Windows poskytuje širokou škálu funkcí, které mohou programy pro svůj běh využívat
  - práce s procesy
  - práce se soubory
  - komunikace po síti
  - tvorba grafických aplikací
  - ...
- Souhrnně tuto funkcionalitu nazýváme **Windows API** (WinAPI)
- Je poskytována jako sada funkcí jazyka C

Typový systém WinAPI

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 *`)

Další typy: `https://docs.microsoft.com/en-us/windows/win32/winprog/windows-data-types`

Typový systém WinAPI
Poznámky

- Může se stát, že datové typy WinAPI nejsou kompatibilní s typovým systémem jazyka C/C++
- Zejména se to týká novějších překladačů (jsou striktnější)
- Je nutno upravit nastavení projektu a povolit benevolentnější práci typy:
  Project → *C/C++* → *Language* → *Conformance Mode* → *Default*
- Další úskalí – řetězce

Typový systém WinAPI
Řetězce

- Typy řetězců:
  - ANSI řetězce – 1 bytové hodnoty (typ `char`)
  - Unicode řetězce – „široké znaky“, 2 byty (typ `wchar_t`)
- Funkce z WinAPI nabízené ve dvou variantách
  - s příponou `A`, pracující s ANSI řetězci
    `CreateFileA`
  - s příponou `W`, pracující s Unicode
    `CreateFileW`
- Reálně se používá makro `CreateFile`, které se expanduje na `CreateFileA` nebo `CreateFileW` dle nastavení projektu

Typový systém WinAPI
Řetězce

- Programátor má 2 možnosti:
  - Zvolí jeden typ řetězců, nastaví jej v projektu a v celém projektu používá jen tento typ
  - Použije pomocná makra z `tchar.h`, která se expandují na daný typ pro práci s řetězci, podle nastavení projektu
- Druhé řešení je univerzálnější, ale zápis kódu je trochu jiný, než známe z dřívějších kurzů

Typový systém WinAPI

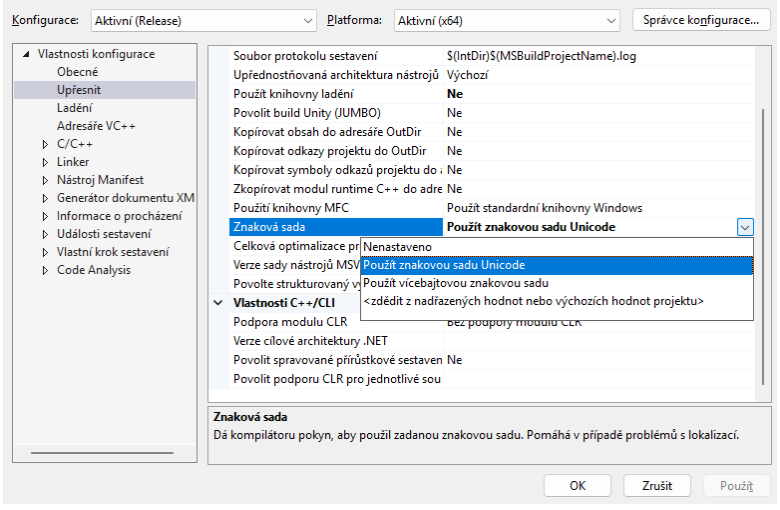
Řetězce

	ANSI	Unicode	`tchar.h`
typ znaku	`char`	`wchar_t`	`_TCHAR`
řetězcový literál	`"abc"`	`L"abc"`	`_T("abc")`
hlavní funkce	`main`	`wmain`	`_tmain`
délka řetězce	`strlen`	`wcslen`	`_tcslen`
kopie řetězce	`strcpy`	`wcscpy`	`_tcscpy`
spojení řetězců	`strcat`	`wcscat`	`_tcscat`
porovnání řetězců	`strcmp`	`wcscmp`	`_tcscmp`
formátovaný výstup	`printf`	`wprintf`	`_tprintf`

Project → *Properties* → *Advanced* → *Character set*

Typový systém WinAPI

Řetězce



Project → *Properties* → *Advanced* → *Character set*

Rozhraní pro práci se soubory

- Pro práci se soubory má Windows vlastní sadu funkcí
  `https://docs.microsoft.com/en-us/windows/win32/api/fileapi/`
- **otevření nebo vytvoření souboru**: `CreateFile`
- argumenty: cesta k souboru a příznaky, jak se souborem pracovat
- příznaky:
  - režim přístupu (čtení/zápis)
  - režim souběžného přístupu (zda může být otevřený soubor znovu otevřen)
  - jak se zachovat, pokud soubor existuje nebo neexistuje
  - bezpečností atributy
  - atributy souboru
  - šablona

Příklad

Příklad

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

int _tmain(int argc, TCHAR* argv[]) {
    HANDLE hFile = CreateFile(
        _T("foo.txt"), // cesta k souboru
        GENERIC_READ | GENERIC_WRITE, // rezim prace se souborem
        FILE_SHARE_READ, // rezim sdileneho pristupu k souboru (pro cteni)
        NULL, // bezpecnostni atributy
        OPEN_ALWAYS, // jak nalozit s (ne)existujicimi soubory
        0, // priznaky souboru
        NULL);  // odkaz na sablonu

    if (hFile == INVALID_HANDLE_VALUE) {
        _tprintf(_T("error opening file\n"));
        return -1;
    }

    _TCHAR* str = _T("Hello world!");

    if (WriteFile(hFile, str, sizeof(_TCHAR) * _tcslen(str), NULL, NULL))
        _tprintf(_T("ok"));

    CloseHandle(hFile);
    return 0;
}
```

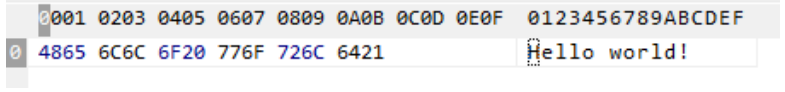
Příklad

- návratová hodnota funkce `CreateFile` je hodnota typu `HANDLE`
- unikátní identifikátor (v rámci spuštěného procesu) pro jednotlivé objekty poskytované jádrem operačního systému
- používá se i pro práci s dalšími objekty, např. procesy, vlákny, synchronizačními objekty atd.
- technicky se jedná o typ `void *`
- v případě, že `CreateFile` z nějakého důvodu selže, je jako návratová hodnota vrácena konstanta `INVALID_HANDLE_VALUE`
- Podrobnosti o chybě můžeme zjistit pomocí funkce `GetLastError()`
- zápis do souboru pomocí `WriteFile` (podobná `fwrite` ze standardní knihovny jazyka C)
- má navíc 2 argumenty – jedním získáme počet zapsaných bytů (předáváme ukazatel, kam se má uložit), druhý nebudeme používat, ale slouží pro asynchronní práci se souborem
- ukončení práce se souborem pomocí `CloseHandle`

Příklad

Příklad

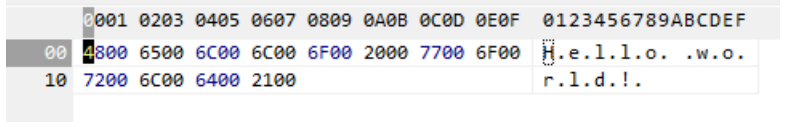
Spusťte program a podívejte se do vytvořeného souboru, ideálně s textovým editorem nebo prohlížečem, který umí soubor zobrazit v hexadecimální podobě (např. Total Commander).



Příklad

Příklad

Změňte nastavení znaků v projektu. Spusťte program a podívejte se do vytvořeného souboru, ideálně s textovým editorem nebo prohlížečem, který umí soubor zobrazit v hexadecimální podobě (např. Total Commander).



Příklad

Příklad

Upravte program tak, aby přečetl obsah vámi zvoleného souboru a vypsal jej co konzole. Použijte funkci `ReadFile`.

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

Pozor na nastavení projektu a na kódování souboru, který otevíráte!

Vytvoření nového procesu

- Přímočaré pomocí `CreateProcess`
  `https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa`
- argumenty:
  - název (cesta) ke spouštěnému programu
  - argumenty a další příznaky či odkazy na struktury, kterými se má vytvoření souboru řídit
- důležité struktury:
  - `STARTUPINFO` – doplňující informace pro spuštěný proces (např. pozice okna)
  - `PROCESS_INFORMATION` – informace o vytvořeném procesu (např. handle na vytvořený proces)
- do `STARTUPINFO` je do atributu `cb` nutno zadat velikost struktury v bytech

Příklad

Příklad

```
#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));
    _TCHAR argumenty[] = _T("C:\\WINDOWS\\system32\\notepad.exe foo.txt");
    // vytvori novy proces
    if (!CreateProcess(_T("C:\\WINDOWS\\system32\\notepad.exe"), // cesta k programu
        argumenty, // uplny seznam argumentu
        NULL, // nastaveni bezpecnostnich atributu
        NULL, // nastaveni bezpecnostnich atributu
        FALSE, // zakaz dedeni handlu mezi procesy
        0, // priznaky pro vytvoreni procesu
        NULL,    // definice prostredi (pouzije se prostredi aktualniho procesu)
        NULL,   // pracovni adresar (pouzije se adresar aktualniho procesu)
        &si,  // ukazatel na strukturu s informacemi ke spoustenumu procesu
        &pi)) {   // ukazatel na strukturu, kterou jsou predany informace o vzniklem procesu
        _tprintf(_T("CreateProcess failed (%d).\n"), GetLastError());
        return 1;
    }
    // ceka na ukonceni procesu
    WaitForSingleObject(pi.hProcess, INFINITE);
    // uzavre proces
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
    return 0;
}
```

Příklad

Příklad

Rozšiřte ukázkový příklad o volání funkce `GetExitCodeProcess`, která vrací návratový kód ukončeného procesu, a tento kód vypište.

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

Procesy

- každý proces má ve Windows přiřazeno `id` (`pid`)
- funkce `OpenProcess` – vrací handle na existující proces na základě identifikátoru `https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocess`
- `HANDLE hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pid);`
- Pokud máme dostatečná oprávnění, můžeme o něm zjistit informace nebo ho ukončit

Úkoly k procvičení

1. Spusťte vhodnou aplikaci (notepad, kalkulačku), v TaskManageru zjistěte jeho `pid`.
2. Pomocí funkce `GetProcessImageFileName` zjistěte cestu ke spuštěnému souboru.
   `https://docs.microsoft.com/en-us/windows/win32/api/psapi/nf-psapi-getprocessimagefilenamea`
3. Pomocí funkce `TerminateProcess` jej ukončete.
   `https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-terminateprocess`