Operační systémy 1

8. Cvičení

Radek Janoštík

KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Úkoly z minula – jak to šlo?

- Kolik se vám toho podařilo udělat?
  1. Program `rect`
  2. Program `mypwd`
  3. Program `mypwd2`
  4. Úpravy ukázkového příkladu
- Dalo něco zabrat?
- Nějaké dotazy?
- Jako prezenčka poslouží písemky

WinAPI

- = Funkcionalita OS Windows, kterou mohou programy využívat pro svůj běh
- Manipulace s procesy
- Práce se souborovým systémem
- GUI, komunikace po síti, ...
- Poskytována jako sada funkcí v jazyce C

WinAPI – typový systém

- Vlastní sada datových typů s jednoznačnou definicí velikostí a rozsahů hodnot.
- Nejpoužívanější datové typy:

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` *)

WinAPI – typový systém

- Typů je násobně více
- Více na `https://learn.microsoft.com/en-us/windows/win32/winprog/windows-data-types`
- Nemusí být kompatibilní s typovým systémem jazyků C/C++
- Např. ve VisualStudiu je potřeba upravit nastavení chování překladače:
  - Project→Properties→C/C++→Language→Conformance Mode→Default

WinAPI – řetězce

- WinAPI má dva typy pro řetězce:
  - *ANSI* – jednobajtové znaky (`char`)
  - *Unicode* – dvoubajtové znajky (`wchar_t`) – „široké znaky“
- Většina WinAPI funkcí ve dvou variantách:
  - Přípona „A“ pro *ANSI* znaky. Např.: `CreateFileA`
  - Přípona „W“ pro *Unicode* znaky. Např.: `CreateFileW`
- Programátor si může jednu variantu vybrat a tu natavit projektu
  - VStudio: Project→Properties→Advanced→Character Set
- Nebo využít sadu maker z hlavičkového souboru `tchar.h`

WinAPI – řetezce – `tchar.h`

- `tchar.h` má sadu maker, které expandují podle nastavení projektu
- Např. makro `_TCHAR` expanduje buď na `char` nebo `wchar_t`

	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`

WinAPI – práce se soubory

- Dokumentace:
  `https://docs.microsoft.com/en-us/windows/win32/api/fileapi/`
- Většina funkcí má velké množství argumentů (spolupracujte s dokumentací!)
- Funkce `CreateFile`
  - Cesta k souboru
  - Režim práce se souborem (různé flagy)
  - Režim přístupu
  - Bezpečnostní atributy
  - Jak naložit s (ne)existujícími soubory
  - Příznaky souboru
  - Práce se šablonou
- Ukázka zdrojového kódu

WinAPI – práce se soubory

- Návratová hodnota: `HANDLE`
  - Unikátní identifikátor(v daném procesu) objektů poskytnutých jádrem OS
  - Např. soubor, proces, vlákno, synchronizační objekt, ...
  - Technicky typ `void *`, ale nespoléhat na to
- Funkce `WriteFile`
  - `HANDLE` – kam se má zapsat
  - `LPCVOID` – co se má zapsat
  - `DWORD` – kolik se toho má zapsat
  - `LPDWORD` – kam se uloží počet zapsaných bajtů
  - `LPOVERLAPPED` – struktura pro asynchronní zápis (nebudeme používat)
- Po dokončení práce je potřeba soubor uzavřít:
  - Funkce `CloseHandle`

WinAPI – vytvoření procesu

- Funkce `CreateProcess`, důležité argumenty:
  - `LPCSTR` – plná cesta ke spustitelnému souboru
  - `LPSTR` – úplný seznam argumentů
  - ...
  - Struktura `STARTUPINFO` – doplňující informace pro spouštěný proces (pozice okna, ...)
    - + dodat velikost struktury v bajtech do atributu `cb`
  - Struktura `PROCESS_INFORMATION` – pro (výstupní) informace o vytvořeném procesu
- Ukázka

Úkol (1/2)

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. Změnte 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`

Úkol (2/2)

4. Rozšiřte ukázkový příklad o volání funkce `GetExitCodeProcess`, která vrací návratový kód ukončeného procesu, a tento kód vypište. Tyto úpravy provádějte postupně.
   - Spusťte vhodnou aplikaci (notepad, kalkulačku), v TaskManageru zjistěte jeho pid.
   - Pomocí funkce `GetProcessImageFileName` zjistěte cestu ke spuštěnému souboru.
   - Pomocí funkce `TerminateProcess` jej ukončete.