Systémové programovanie Windows

Memory management

Andrea Číková
Martin Osovský

O čom to dnes bude?

- Virtuálna pamäť
  - Adresový priestor procesu
  - Rezervovanie, komitnutie a uvoľnenie pamäte
  - Zásobník vlákna
- Memmory-mapped files
  - Čo sú a ako s nimi pracovať
- Halda
  - Čo to je a kedy vytvoriť dodatocnú haldu

Mechanizmy pre prácu s pamäťou

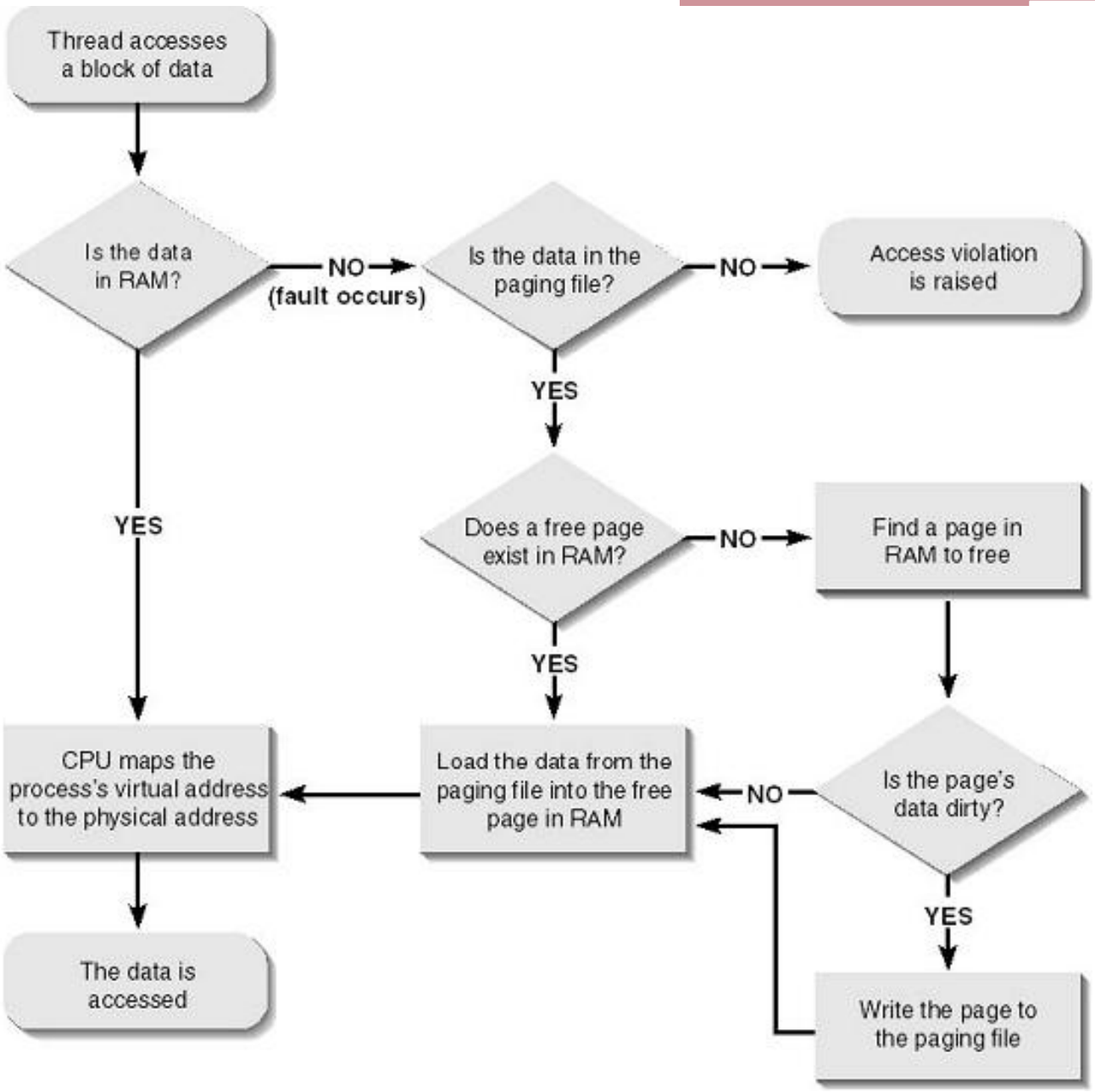
- **Virtuálna pamäť** – vhodná pre správu veľkých polí objektov a štruktúr
- **Memory-mapped files** – vhodné pre správu obsiahlych prúdov dát
- **Haldy (heaps)** – vhodné pre správu veľkého množstva malých objektov

Virtuálna pamäť – adresový priestor procesu

- Každý proces má svoj vlastný adresový priestor – vlákno môže pristupovať iba k adresovému priestoru svojho procesu
- 32-bitový proces = 4GB adr. priestor
- 64-bitový proces = 16EB (exabytes) adr. priestor
- Jedná sa o **virtuálny** adresový priestor, nie fyzickú pamäť!!

Free, reserved, committed,...

- Časť adresového priestoru po vytvorení je voľná (***free***)
- Pre použitie časti tohto priestoru je nutné alokovať oblasť (*region*) volaním funkcie `VirtuallAlloc` – daná časť je potom rezervovaná (***reserved***); veľkosť oblasti je násobkom veľkosti stránky
- Pre použitie rezervovanej oblasti je nutné alokovať fyzickú pamäť volaním `VirtualAlloc` – ***committing***
- Uvoľnenie fyzickej pamäte – `VirtualFree` – ***decommitting***
- Uvoľnenie rezervovanej oblasti – `VirtualFree` - ***releasing***


Thread accesses a block of data
Is the data in RAM?
NO (fault occurs)
Is the data in the paging file?
NO
Access violation is raised
YES
YES
Does a free page exist in RAM?
NO
Find a page in RAM to free
YES
CPU maps the process's virtual address to the physical address
Load the data from the paging file into the free page in RAM
NO
Is the page's data dirty?
YES
The data is accessed
Write the page to the paging file

Funkcia VirtualAlloc

```
PVOID VirtualAlloc(
      PVOID address,
      SIZE_T regionSize,
      DWORD allocationType,
      DWORD protection);
```

allocationType:

- MEM_RESERVE – rezervuje oblasť v adr. priestore
- MEM_COMMIT – pridelí oblasti fyzickú pamäť
- MEM_RESERVE | MEM_COMMIT – rezervuje oblasť a pridelí jej fyzickú pamäť v jednom kroku

Zásobník vlákna

Memory Address	State of Page
0x080FF000	Top of stack: committed page
0x080FE000	Committed page with guard protection attribute flag
0x080FD000	Reserved page
0x08003000	Reserved page
0x08002000	Reserved page
0x08001000	Reserved page
0x08000000	Bottom of stack: reserved page

Zásobník vlákna

Memory Address	State of Page
0x080FF000	Top of stack: committed page
0x080FE000	Committed page
0x080FD000	Committed page
0x08003000	Committed page
0x08002000	Committed page with guard protection attribute flag
0x08001000	Reserved page
0x08000000	Bottom of stack: reserved page

Zásobník vlákna

Memory Address	State of Page
0x080FF000	Top of stack: committed page
0x080FE000	Committed page
0x080FD000	Committed page
0x08003000	Committed page
0x08002000	Committed page
0x08001000	Committed page
0x08000000	Bottom of stack: reserved page

Memory-mapped files

- EXE a DLL súbory namapované do pamäte
- Dátové súbory namapované do pamäte
- Zdieľanie dát medzi procesmi pomocou *memory-mapped* súborov

Namapovanie súborov

- `CreateFile` – vytvorí objekt typu súbor
- `CreateFileMapping` – vytvorí *file-mapping* objekt
- `MapViewOfFile` – namapuje *file-mapping* objekt

... práca s namapovaným súborom ...

- `UnmapViewOfFile` – odstránenie namapovania
- `CloseHandle` – zatvorenie *handle* k súboru a k *file-mapping* objektu

Halda

- Halda je oblasť rezervovaného adresového priestoru
- Vhodná pre správu veľkého množstva malých objektov
- Sériový prístup
- Implicitná halda procesu – `GetProcessHeap()`
- Dodatočné haldy procesu – `CreateHeap()`

Dôvody pre vytvorenie dodatočnej haldy

- Ochrana komponentov
- Efektívnejšia správa pamäte
- Lokálny prístup
- Rýchle uvoľnenie pamäte

Otázky?

Ďakujeme za pozornosť☺