PB153 Operační systémy a jejich rozhraní

Struktura a rozhraní OS

Komponenty OS

- Správa procesů
- Správa operační paměti
- Správa souborů
- Správa I/O zařízení
- Správa sekundární paměti
- Správa síťových služeb
- Ochranný systém
- Interpret příkazů (shell)

Správa procesů

- Proces (aktivní entita) je spuštěný program (pasivní entita)
  - jeden program může být zároveň spuštěn řadou uživatelů
  - proces je jednotkou práce systému
    - procesy jádra OS
    - uživatelské procesy
- Proces pro svou činnost potřebuje zdroje
  - procesor, paměť, soubory
  - zdroje se alokují při spuštění nebo při běhu

Správa procesů (2)

- Typické aktivity OS v oblasti správy procesů
  - vytváření a ukončování uživatelských a systémových procesů
  - potlačování (suspending) a obnovování (resuming) procesů
  - mechanismy pro synchronizaci procesů
  - mechanismy pro komunikaci mezi procesy
  - mechanismy pro detekci a řešení uváznutí (deadlock)

Správa operační paměti

- Při spuštění procesu musíme program nahrát do paměti
- Později může proces vyžadovat dodatečnou paměť pro data a tuto paměť OS vracet
- Jakmile proces končí, musí OS veškerou paměť užívanou procesem opět uvolnit

Správa operační paměti

- Typické aktivity OS v oblasti správy paměti
  - znalost, která část paměti je využívána a kým (kterým procesem)
  - přidělování (alokace) a uvolňování (dealokace) paměti podle požadavků procesů
  - rozhodování o tom, který proces kdy zavést do paměti

Správa souborů

- Soubor
  - kolekce souvisejících informací definovaná svým tvůrcem
  - může mít strukturu, ale nemusí
  - z hlediska OS typicky jen posloupnost bajtů
- I/O zařízení na kterých jsou soubory uloženy mohou být nejrůznějšího typu
  - magnetické disky, optické disky, magnetické pásky
  - OS zavádí abstraktní koncept souboru

Správa souborů (2)

- Typické aktivity OS v oblasti správy souborů
  - vytváření a rušení souborů
  - vytváření a rušení adresářů (složek)
  - základní operace pro manipulaci se soubory a adresáři (např. čtení ze souboru, zápis do souboru, seznam souborů v adresáři)
  - zálohování na nevolatilní (energeticky nezávislá) média

Správa I/O zařízení

- Snaha o skrytí specifik jednotlivých I/O zařízení
  - co nejjednotnější přístup k jednotlivých I/O zařízením
  - mnoho OS zpřístupňuje I/O zařízení přes speciální soubory
    - /dev/sda, /dev/lp0 v UNIXu
    - \\.\PHYSICALDRIVE2, CONIN$ ve Win32
- Ovladače jednotlivých HW komponent
- Řízení bufferů, kešování, spooling

Správa sekundární paměti

- Typické sekundární paměti jsou disky
- Správa sekundárních pamětí obvykle formou souborů – souborového systému
- Typické aktivity OS
  - správa volného místa
  - přidělování místa
  - plánování činnosti disku (které požadavky kdy vyřídit)

Správa síťových služeb

- Komunikace je řízena protokolem
  - protokol je několikastranný algoritmus pro dosažení určitého cíle
- Snaha o transparentnost
  - síťové souborové systémy (SMB, NFS, …)
  - API podobné jako přístup k souborům

Ochranný systém

- V multitaskingovém a multiuživatelském OS musíme jednotlivé procesy navzájem chránit
- **Ochrana** je mechanismus, který řídí přístup programů, procesů a uživatelů ke zdrojům počítačového systému
  - HW za pomocí OS zajišťuje, že proces může používat pouze adresy svého adresového prostoru
  - časovač brání jednomu procesu v získání plného přístupu k CPU
  - režim CPU brání uživatelským procesům spouštět privilegované instrukce
- Ochranný mechanismus rozlišuje autorizované a neautorizované použití prostředků

Interpret příkazů

- Interpret příkazů, neboli command-line interpreter, neboli shell
  - úkolem je získávat příkazy od uživatele a provádět je
- Za tento interpret příkazů můžeme považovat i moderní GUI
- Někdy je interpret příkazů přímo součást jádra operačního systému, někdy je to jen speciální program

Služby OS

- Spouštění programů
  - zavedení programu do paměti, běh procesu
- I/O operace
  - uživatelské procesy nemají přístup k I/O instrukcím, vše musí poskytovat OS
- Souborový systém
  - manipulace se soubory a adresáři
- Komunikace
  - mezi procesy nebo počítači
- Detekce chyb
  - musí počítat s chybami HW (např. parita paměti)

Interní služby OS

- Zabezpečují efektivní provoz samotného OS
- Alokace zdrojů
  - plánovací algoritmy pro přidělování CPU
  - přidělování přístupu k tiskárnám apod.
- Účtování
  - máme přehled o tom, kteří uživatelé kdy využily které zdroje
- Ochrana
  - autentizace, řízení přístupu

Systémová volání

- Systémová volání tvoří rozhraní mezi uživatelským procesem a OS
  - typicky jsou popsána jako instrukce assembleru a jsou uvedena v programátorském manuálu k OS
  - vyšší programovací jazyky obsahují některé funkce, které odpovídají systémovým voláním (např. open, write) a dále knihovní funkce, které poskytují vyšší funkčnost a v rámci této spouští (třeba hned několik) systémových volání (např. fopen, fwrite).

Systémová volání (2)

- Různé OS a různé HW platformy mívají různé způsoby jak volat služby OS a různou strukturu těchto služeb
- Nicméně existují určité standardy, které usnadňují přenositelnost programového kódu
  - v oblasti UNIXu: POSIX
  - V oblasti Windows: Win32
- Teoreticky kód který bude psán podle standardu bude přeložitelný na kompatibilních platformách, v praxi však existuje celá řada verzí standardu a mnoho výjimek co je a není implementováno

Kategorie systémových volání

- Správa procesů
  - spusť, ukonči, čekej, nastav atributy
- Správa souborů
  - vytvoř, smaž, otevři, čti, zapiš, zavři
- Správa I/O
  - získej přístup, připoj, čti, zapiš, odpoj
- Správa informací
  - datum a čas, atributy procesů a souborů
- Komunikace
  - vytvoř spojení, odešli, přijmi

Příklad systémového volání

- Funkce Win32 „**CreateProcessAsUser**“

```
BOOL CreateProcessAsUser(
  HANDLE hToken,         // handle to a token representing the logged-on user
  LPCTSTR lpApplicationName,  // pointer to name of executable module
  LPTSTR lpCommandLine,       // pointer to command line string
  LPSECURITY_ATTRIBUTES lpProcessAttributes, // process security attributes
  LPSECURITY_ATTRIBUTES lpThreadAttributes,  // thread security attributes
  BOOL bInheritHandles,         // whether new process inherits handles
  DWORD dwCreationFlags,        // creation flags
  LPVOID lpEnvironment,         // pointer to new environment block
  LPCTSTR lpCurrentDirectory,   // pointer to current directory name
  LPSTARTUPINFO lpStartupInfo,  // pointer to STARTUPINFO
  LPPROCESS_INFORMATION lpProcessInformation  // pointer to PROCESS_INFORMATION
);
```

- **Windows NT:** Requires version 3.51 or later.
  Windows: Unsupported.
  Windows CE: Unsupported.
  Header: Declared in winbase.h.
  Import Library: Use advapi32.lib.
  Unicode: Implemented as Unicode and ANSI versions on Windows NT.

Způsob předání parametrů

- Registry
  - `mov ah,01h`
  - `mov cx,2000h`
  - `int 10h`
- Zásobník
  - `mov ax,0001h`
  - `push ax`
  - `int 10h`
- Blok (struktura, tabulka) v paměti & pointer
  - `mov ax,0001h`
  - `mov [tabulka1],ax`
  - `mov bx,tabulka1`
  - `int 10h`

Příklad 1: MS-DOS

- DOS interrupts
  - INT 20H Terminate a program
  - INT 21H DOS Services
  - INT 25H/26H Absolute Disk Read/Write
  - INT 27H Terminate but Stay Resident
  - INT 28H DOS Safe Interrupt / DOS Timeslice (UNDOCUMENTED)
  - INT 2eH Perform DOS Command (UNDOCUMENTED)
  - INT 2fH Multiplex Interrupt (print spooler, TSR control)
  - INT 33H Mouse Support
  - INT 67H Expanded Memory Manager (LIM-EMS)
- Address Pointers (not used as software interrupts)
  - INT 22H Terminate address
  - INT 23H Control-Break address
  - INT 24H Critical Error Handler address
- Volání služeb DOSu:
  - AH = číslo služby
  - ostatní registry (AL, DL, DX, CX, DS:DX, DS:SI, ES:DI, ES:BX) = parametry
  - INT 21H
  - návratová hodnota bývá v AL

Příklad 1: MS-DOS (2)

- Systémové volání pro otevření existujícího souboru
- DOS Fn 3dH: Open a File Handle
  - Expects:
    - AH=3dH
    - DS:DX=address of an ASCIIZ string of a filespec
    - AL=Open Mode
      - AL = 0 to open for reading
      - AL = 1 to open for writing
      - AL = 2 to open for reading and writing
  - Returns: AX=error code if CF is set to CY

    =file handle if no error

Příklad 1: MS-DOS (3)

- Programový kód

```
filename: DATA “c:\autoexec.bat”,0

mov ax,3d00h
mov dx,[filename]
int 21h
```

Příklad 2: Linux

- Systémová volání
  - standardně přes `int 0x80`
  - nově i přes instrukci `syscall (sysenter)`
- Číslo systémového volání
  - v registru `eax`
  - v jádře 2.2 přibližně 200 volání, v jádře 2.6 přes 300 volání
- Parametry systémového volání
  - v registrech `ebx, ecx, edx, esi, edi, ebp`
  - kromě fce 117 (parametrem je odkaz na strukturu)
- Výsledek
  - uložen v `eax`

Příklad 2: Linux (2)

- Hello world v assembleru pod Linuxem

```
section .text
global _start
msg db 'Hello, world!',0xa
len equ $ - msg
_start:
    mov edx,len ;message length
    mov ecx,msg ;message to write
    mov ebx,1 ;file descriptor (stdout)
    mov eax,4 ;system call number (sys_write)
    int 0x80 ;call kernel
    mov eax,1 ;system call number (sys_exit)
    int 0x80 ;call kernel
```

Příklad 3: FreeBSD

- Systémová volání
  - přes VOLÁNÍ FUNKCE, která obsahuje `int 0x80`
  - nebo přes volání brány `call 7:0`
- Číslo systémového volání
  - v registru `eax`
- Parametry systémového volání
  - na zásobníku (první parametr je ukládán poslední)
- Výsledek
  - v registru `eax`
  - volající proces musí vyčistit zásobník (parametry)

Příklad 3: FreeBSD (2)

- Hello World v assembleru ve FreeBSD

```
 section .text
     global _start
 msg db "Hello, world!",0xa
 len equ $ - msg
  _syscall: int 0x80 ;system call
     ret
 _start: ;tell linker entry point
     push dword len ;message length
     push dword msg ;message to write
     push dword 1 ;file descriptor (stdout)
     mov eax,0x4 ;system call number (sys_write)
     call _syscall ;call kernel
     add esp,12 ;clean stack (3 arguments * 4)
     push dword 0 ;exit code
     mov eax,0x1 ;system call number (sys_exit)
     call _syscall ;call kernel
```

Komplexita OS

- Počet systémových volání OS / API volání

OS	rok	počet systémových volání
UNIX	1971	33
UNIX	1979	47
SunOS	1989	171
4.3 BSD	1991	136
SunOS	1992	219
SunOS	1997	190
Linux	1998	229
NT 4.0	1999	3443

Komplexita OS (2)

- Rozsah zdrojového kódu Windows

Verze	**rok**	**miliony řádků**
NT 3.1	1993	4-5
NT 3.5	1994	7-8
95	1995	10
NT 4.0	1996	11-12
98	1998	18
2000	2000	29
XP	2001	40
2003 Svr	2003	50