Universitatea Politehnica București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

 **SISTEME DE OPERARE**

Top of Form

**Subsistemul POSIX și interfața de proces Win32**

**Profesor coordonator:** Studenți:

prof. univ. dr. ing. Ştefan Stăncescu Bardă Alexandru Mihai

Radu Viorel Flaviu

Grupa 433A

CUPRINS:

**1 Subsistemul POSIX (BARDA MIHAI ALEXANDRU)**

 **1.1 Thread-uri POSIX ( Pthreads)**

 **1.2 Subsisteme integrale**

 **1.3 Subsistemele mediilor de lucru**

**1.4.POSIX pentru Windows**

**2. Win32 API ( RADU VIOREL-FLAVIU)**

**2.1** **Tipuri de date noi folosite de funcţiile Win32 API**

**2.2. Scurt Istoric**

**2.3. Versiuni API**

**2.4.Windows API**

**2.5. Servicii de bază**

**2.6. Servicii avansate**

**2.7. Interfata dispozitive grafice**

**2.8. Interfaţa utilizator**

**2.9. Biblioteca de ferestre de dialog uzuală**

**2.10. Biblioteca de control uzuală**

**2.11. Windows shell**

**2.12. Servicii reţea**

 **3. Bibliografie**

1. **Subsistemul POSIX**

POSIX – este un acronim pentru Portable Operating System Interface şi reprezintă o familie de standarde implementate de IEEE (Institute of Electrical and Electronics Engineers) pentru a realiza compatibilitatea între sistemele de operare. POSIX defineşte API (Aplication Programming Interface) împreună cu linii de comandă shell şi interfeţe de utilitate, astfel încât programatorii să-şi poată muta cu uşurinţă aplicaţiile de pe un sistem pe altul.

 **1.1 Thread-uri POSIX ( Pthreads)**

Standardul, POSIX 1.c, defineşte un API pentru a creea şi manipula thread-uri. Implementările API sunt compatibile cu multe sisteme de operare bazate pe Unix precum: FreeBSD, NetBSD, OpenBSD, Linux, Mac OS X şi Solaris. Desigur, astfel de implementări există şi pentru Windows: subsistemul SFU/SUA, are la baza o implementare primară a unui anumit număr de API şi de asemenea în pachete terţiare precum pthreads-w32 care implementează pthread-uri în plus pe lângă cele de Windows.

 **1.2 Subsisteme integrale**

Efectuează funcţii necesare de sistem. Ele acoperă un cadru larg de lucru. Iată câteva din responsabilităţile lor:

- împreună cu Security Accounts Manager ( managerul de securitate ) şi procesul Logon, Local Security Authority defineşte politica de securitate pentru sistem.

- Service Control Manager ( managerul de servicii ) încarcă, supraveghează şi scoate din funcţiune componente de sistem cum ar fi serviciile sau driverele.

- procesele RPC Locator şi RPC Service oferă suport aplicaţiilor distribuite ce folosesc apeluri de procedură.

**1.3 Subsistemele mediilor de lucru**

Au rolul de a da o interfaţă şi un mediu de lucru pentru aplicaţii proprii anumitor sisteme de operare. În prezent Windows NT conţine următoarele subsisteme:

- Win32 care implementează interfaţa de lucru pentru Windows NT - Virtual DOS Machine (VDM) permite aplicaţiilor pe 16 biţi să ruleze sub Windows NT.
- POSIX ce are suport API pentru programele ce îndeplinesc standardul POSIX 1003.1. Deoarece POSIX 1003.1 nu este un standard binar, aplicaţiile trebuie să fie compilate şi legate înordine în acest subsistem.

‐ interfaţă grafică) pentru întreg sistemul. Asta include intrările şi ieşirile pentru alte subsisteme .

- implementează GUI‐ul vizualizat de programatori sau utilizatori.

- funcţiile USER ( utilizator ) sunt în legătură cu obiectele GUI (graphic user inteface ) cum ar fi meniurile şi butoanele.

- funcţiile GDI ce efectuează operaţii de desenare pe dispozitive precum monitoarele şi imprimantele.

- funcţii KERNEL ce monitorizează procese, threaduri, obiecte de sincronizare, memoria partajată şi O anumită aplicaţie este mereu în legătură cu un anume subsistem şi poate folosi doar facilităţile acelui subsistem. Spre exemplu, o apicatie POSIX nu poate apela funcţii Win32. Subsistemele diferite de Win32 sunt integrate pentru Windows NT în mare parte pentru compatibilitate.



Figura 1. Implementarea subsistemului POSIX

* 1. **POSIX pentru Windows**

Subsistemul Microsoft POSIX este unul din cele 3 subsisteme ale mai multor sisteme de operare din familia Windows NT (împreună cu OS / 2 şi subsistemele Windows).
Microsoft Windows implementează doar prima versiune a standardelor POSIX, şi anume POSIX.1. Codul oficial al POSIX.1 este ISO / IEC 9945-1: 1990 sau IEEE standard de 1003.1-1990.
Mediul de execuţie a subsistemului este asigurat de două fişiere: *psxss.exe* şi *psxdll.dll*. O aplicaţie POSIX foloseşte *psxdll.dll* pentru a comunica cu subsistemul, în timp ce comunicăcu posix.exe pentru a oferi funcţionalitate de afişare pe desktop-ul Windows.
Subsistemul POSIX a fost eliminat cu Windows XP / Windows Server 2003. Acesta a fost înlocuit cu "Windows Services for UNIX", care utilizează subsistemul Interix.



Figura 2 – Interacțiunea fișierelor in subsistemul POSIX

Standardul POSIX 1003

 1003.1   functii de biblioteca, apeluri de sistem

1003.2   shell‐ul și utilitarele

1003.3   metode de testare

1003.4   timp real

1003.5   limbajul Ada pentru Unix

1003.6   securitatea

1003.7   administratia sistemului

1003.8   accesul la fisiere

1003.9   Limbajul FORTRAN pentru Unix

1003.10 super‐calculatoare

1003.12 interfetele independente de protocol

1003.13 real‐time profiles

1003.15 interfetele pentru supercalculatoare ‐ batch processing

1003.16 limbajul C

 1003.17 servicii de directoare

 1003.18 POSIX standardized profile

1003.19 FORTRAN 90

**2. Win32 API**

Windows API este o interfaţă destinată programării aplicaţiilor pentru sistemul de operare Microsoft Windows (Application Programming Interface). Windows API este cunoscută, în general, cu numele de Win32 API, însă denumirea Windows API reflectă mai precis capacităţile şi utilitatea să, suportul atât pentru Windows 32-biţi, cât şi pentru Windows 64-biţi.

Microsoft Windows SDK ( Software Development Kit) conţine documentaţia şi unelte necesare programatorilor pentru a realiza aplicaţii folosind Windows API.

Prin Win32 API, programatorul are acces direct la o mare parte a funcţiilor de nivel jos (en. low-level) ale sistemului de operare, putând crea într-un mod foarte flexibil aplicaţii.Windows API conţine o ofertă de servicii pentru toate aplicaţiile bazate pe ferestre.Această interfaţă de programare permite utilizatorilor să realizeze o interfaţă grafică propriilor aplicaţii, să acceseze sistemul computerului, memoria acestuia, dispozitivele (fie de intrare, fie de ieşire), să implementeze sunete, imagini, video sau chestiuni de reţea (respectiv Internet) în acestea (în aplicaţii). Programarea cu Windows API înseamnă primirea, interpretarea, trimiterea mesajelor, către ferestre, sau controale (obiecte controlabile - ex. ToolBox, EditBox, Button, Text, CheckBox).

Funcţia principala a unei aplicaţii Win32 conţine o buclă) (ex. while() - limbajul C++) în care sunt apelate funcţiile de preluare şi traducere a mesajelor trimise de către utilizator (prin intermediul dispozitivelor de intrare, apoi al sistemului de operare, în cazul nostru Windows). Mesajele sunt interpretate şi trimise mai departe ferestrei active. Mesajele pot fi trimise atât de sistem, asemenea unor mesaje din "subconştient", dacă ar fi să facem o comparaţie cu creierul uman, cât şi explicit, "conştient", prin funcţia SendMessage(). Fiecărei ferestre i se asociază o procedură responsabilă cu interpretarea mesajelor primite. Spre exemplu, dacă unei ferestre i se trimite un mesaj de distrugere, aceasta&nbsp;va&nbsp;dispărea. Dacă unui control de tip CheckBox i se va trimite mesajul de validare, în dreptul său va apărea binecunoscutul marcaj de validare.

**2.1** **Tipuri de date noi folosite de funcţiile Win32 API**

Datorită diferenţelor dintre programele DOS şi Windows, Win32 API va utiliza o serie de tipuri de date noi. Câteva din acestea sunt:

 • HANDLE – este un tip generic (identificator) , utilizat pentru manipularea obiectelor folosite în program (fişiere, ferestre, etc.). Pentru a manipula un astfel de obiect, este necesară întâi obţinerea unui asemenea HANDLE, în urmă apelării unei funcţii care îl returnează;

 • HWND - este definit că typedef HANDLE HWND; şi este folosit pentru manipularea ferestrelor;

• DWORD – (Double Word) este un întreg fără semn pe 32 de biţi. Un DWORD este compus din două mărimi WORD. Uzual, marea majoritate a compilatoarelor permit extragerea celor două componente WORD prin funcţii de tipul high() şi low();

 • LPVOID – (Long Pointer Void) este definit că typedef void\* LPVOID, fiind deci un pointer void reprezentat pe 32 de biţi;

• LPCSTR – (Long Pointer Constant String) – reprezintă un pointer pe 32 de biţi spre un şir de caractere constant. Este utilizat de obicei atunci când şirul este utilizat ca parametru al unei funcţii şi funcţia nu îl modifică;

• LPCTSTR – (Long Pointer Constant To String) – reprezintă un pointer pe 32 de biţi spre un şir de caractere constant Unicode; Unicode este un cod de caractere pe 16 biţi, capabil să reprezinte caracterele tuturor limbilor. Este utilizat de platformele Windows NT (2000, XP).Windows 95, 98 şi Milenium nu îl utilizează. Pentru a defini un pointer similar, dar spre un şir ASCII, vom declara tipul LPCSTR;

• LPTSTR – (Long Pointer To String) – reprezintă un pointer pe 32 de biţi spre un şir de caractere în format Unicode;

• LPSTR – (Long Pointer String) – reprezintă un pointer pe 32 de biţi spre un şir de caractere în format ASCII;

• WPARAM şi LPARAM – cuvinte cu lungimea de 32 de biţi, utilizate în general pentru a transmite parametri asociaţi unui mesaj Windows;

 • LRESULT – o valoare pe 32 de biţi, returnată de o funcţie;

**2.2. Scurt Istoric**

 Windows API a fost mereu expusă la o mare parte din substructure sistemelor windows. Acest lucru a avut avantajul de a da programatorilor pe Windows o mare flexibilitate şi putere asupra cererilor lor . Cu toate acestea, de asemenea, a dat aplicaţiei Windows o mare responsabilitate în gestionarea diferitelor nivele de prioritate scăzut , operaţiuni care sunt asociate cu o interfaţă grafică de utilizator .

 Charles Petzold , scriitor a mai multor cărţi despre programare împotriva Windows API , a declarat : " Programul mondial original HELLO în Windows 1.0 SDK a fost un pic de scandal. Hello.c a fost de aproximativ 150 de rânduri , iar scriptul HELLO.RC . avut alte aproximativ 20 de linii . ( ... ) Programatorii Veteran adesea se ghemuies în groază sau ras când întâmpina programul Windows HELLO-WORLD"

 De-a lungul anilor , diverse modificări şi completări au fost efectuate la sistemul de operare Windows şi Windows.. Windows API pentru Windows 1.0 a sprijinit mai mult de 450 de apeluri de funcţii ,în timp ce astăzi în versiuni moderne ale Windows API există mii . Cu toate acestea , în general , interfaţă a rămas destul de constantă , iar o aplicaţie veche pentru Windows 1.0 cerere va arată în continuare familiară pentru un programator care foloseşte aplicaţii moderne Windows API

 Un compromis a fost făcută de Microsoft în menţinerea compatibilităţii software. Pentru a realiza acest lucru , atunci când elaborează o nouă versiune de Windows , Microsoft a trebuit uneori să pună în aplicare soluţii pentru a permite compatibilitatea cu software-ul terţ care utilizează o versiune anterioaraAPI într- un mod nedocumentat sau chiarilegal ( programatic ) . Raymond Chen , un dezvoltator Microsoft care lucrează pe Windows API , a declarat : " Aş putea scrie luni de zile numai despre lucrurile rele făcute de aplicaţii şi ceea ce am avut de făcut pentru a le face să lucreze din nou. Dacă orice cerere nu a rulat pe Windows 95 , am luat-o ca un eşec personal . "

 Una dintre cele mai mari schimbări API suferit pentru Windows a fost trecerea de la Win16 ( livrat în Windows 3.1 şi mai mari ) a Win32 ( Windows NT şi Windows 95 ) . În timp ce Win32 a fost introdus iniţial cu Windows NT 3.1 , Win32s permis utilizarea unui subset Win32 înaintea Windows 95 . Abia cu apariţia Windows 95 , a început de portarea pe scară largă a aplicaţiilor la Win32..

 **2.3. Versiuni API**

 Aproape fiecare nouă versiune a Microsoft Windows a introdus propriile completări şi modificări la Windows API .Numele API , cu toate acestea , a rămas constant între versiuni diferite de Windows , iar modificările de nume au fost păstrate limitat la schimbările arhitecturale şi de platforme majore pentru Windows. Microsoft a schimbat în cele din urmă numele current al familiei Win32 API în Windows API , şi a făcut-o într-un termen catch-all pentru ambele versiuni anterioare şi viitoare ale API .

 • Win16 este API pentru primele versiuni de 16 de biţi de Microsoft Windows . Acesta a fost iniţial numit simplu"Windows API " , dar a fost mai târziu redenumit Win16 într-un efort de al distinge de , versiunea mai nouă pe 32 de biţi a Windows API . .

 • Win32este API pe 32 de biţi pentru versiunile moderne de Windows. API este format din funcţii implementate , ca şi Win16 , în DLL-uri de sistem . DLL-uri de baza ale Win32 sunt Kernel32.dll , user32.dll , şi gdi32.dll . Win32 a fost introdus cu Windows NT . Versiunea de Win32 livrată cu Windows 95 a fost iniţial numită Win32c , cu " c " în picioare pentru " compatibilitate " , dar acest termen a fost ulterior abandonat de Microsoft în favoarea Win32 .

 • Win32s este o extensie pentru familia Windows 3.1x de Microsoft Windows , care a implementat un subset al API Win32 pentru aceste sisteme . " S " vine de la " subset " .

 • Win64 este varianta API implementate pe platforme pe 64 de biţi ale arhitecturii pentru Windows ( începând din 2011 AMD64 şi IA - 64 ) . Ambele versiuni pe 32 de biţi şi 64 de biţi ale unei cereri pot fi compilate dintr-un singur cod de baza , deşi unele API-uri mai vechi au fost depreciate , iar unele dintre API-urile care au fost deja depreciate în Win32 au fost în întregime eliminate . Toţi indicii de memorie sunt pe 64 de biţi în mod implicit ( modelul LLP64 ) , astfel încât codul sursă trebuie să fie verificat pentru compatibilitate cu pointer aritmetic pe 64 de biţi şi rescris după cum este necesar .

 • WinCE este punerea în aplicare a API pentru Windows pentru sistemul de operare Windows CE .

**2.4.Windows API**

Windows API in cea mai mare parte se referă cu interacţiunea dintre sistemul de operare şi o cerere. Pentru o comunicare între diferitele aplicaţii Windows, Microsoft a dezvoltat o serie de tehnologii principale alături de Windows API. Acestea au început cu Dynamic Data Exchange (DDE), care a fost înlocuit de către Object Linking and Embedding (OLE), iar ulterior de către Component Object Model (COM), Obiecte de automatizare, controale ActiveX, şi NET Framework.. Nu este întotdeauna o distincţie clară între aceste tehnologii, şi nu există destul de multe de suprapuneri

Varietăţile de termeni sunt în esenţă rezultatul de grupare mecanisme de software care se referă la un aspect special de dezvoltarea de software. Automatizarea specifica se referă la funcţionalitatea exportarii unei cereri sau a unei componente (ca un API), astfel încât să poată fi controlată de o altă aplicaţie în loc ci nu de către un utilizator uman..

Funcţionalitatea oferita de Windows API poate fi grupata în opt categorii:

**2.5. Servicii de bază**

Asigură accesul la resursele fundamentale disponibile pentru un sistem Windows. Sunt incluse lucruri cum ar fi sisteme de fişiere, dispozitive, procese si fire, şi tratarea erorilor. Aceste funcţii au reşedinţa în kernel.exe, krnl286.exe sau fişiere krnl386.exe pe 16-bit Ferestre, şi Kernel32.dll pe 32-bit.

**2.6. Servicii avansate**

Ofera acces la funcţionalitatea care este o completare a kernel-ului. Sunt incluse lucruri ca Windows registry, shutdown / pauză sistem, pornire / oprire / .
Dispozitiv interfaţă grafică.
 Oferă funcţionalitate pentru afişarea de conţinut grafic la monitoare, imprimante şi alte dispozitive de ieşire .O regasim în gdi.exe pe 16-bit Ferestre, gdi32.dll şi pe Windows 32-biţi în user-mode. Kernel-mode GDI sprijin oferit de Win32k.sys care comunică direct cu driverul grafic .

**2.7. Interfata dispozitive grafice**
 Oferă funcţionalitate pentru afişarea de conţinut grafic la monitoare, imprimante şi alte dispozitive de ieşire. Ea se afla în gdi.exe pe Windows 16-biti sigdi32.dll pe Windows 32-biţi în modul utilizator. Suportul GDI pentru modul Kernel este oferit de Win32k.sys care comunică direct cu driverul grafic.

**2.8. Interfaţa utilizator**

Oferă funcţionalitatea pentru crearea şi administrarea ferestrelor şi a majorităţii controalelor de bază cum ar fi butoanele şi barele de scroll, primeşte informaţia de la mouse şi tastatură şi are alte funcţionalităţi asociate cu partea de interfaţă grafică de utilizator (GUI) a Windows. Această unitate funcţională se regăseşte în user.exe pe Windows pe 16 biţi şi în user32.dll pe Windows pe 32 biţi. Începând cu versiunile Windows XP controalele de bază se găsesc în comctl32.dll alături de controalele uzuale.

**2.9. Biblioteca de ferestre de dialog uzuală**

Oferă aplicaţiilor ferestrele de dialog pentru deschiderea şi salvarea de fişiere, alegerea culorii şi fontului etc. Biblioteca se regăseşte într-un fişier numit commdlg.dll pe Windows pe 16 biţi şi în comdlg32.dll pe Windows pe 32 biţi.

**2.10. Biblioteca de control uzuală**

Oferă aplicaţiilor acces la unele controale avansate oferite de sistemul de operare. Acestea includ bare de stare, bare de progres, toolbar-uri şi tab-uri. Biblioteca se regăseşte într-un fişier dll denumit commctrl.dll pe Windows pe 16 biţi şi în comctl32.dll pe Windows pe 32 biţi.

**2.11. Windows shell**

Componentă a Windows API ce permite aplicaţiilor acces la funcţionalităţile oferite de shell-ul sistemului de operare ca şi posibilitatea de a-l schimba şi îmbunătăţii. Componenta se regăseşte în shell.dll pe Windows pe 16 biţi şi în shell32.dll pe Windows pe 32 biţi .

**2.12. Servicii reţea**

Oferă acces la diversele elemente de reţea ala sistemului de operare. Subcomponentele sale includ NetBIOS, Winsock, NetDDE, RPC şi multe altele. Win32 API nu a fost destinat să fie interfaţă de programare originală pentru Microsoft Windows NT. Deoarece proiectul Windows NT a început ca un înlocuitor pentru OS/2 versiunea 2, interfata de programare principală a fost APi-ul pe 32 de biţi OS/2 Presentation Manager. După un an de lucru la proiect s-a lansat Microsoft Windows 3.0 şi a avut un mare succes. Ca urmare Microsoft a modificat direcţia de dezvoltare şi a făcut Windows NT înlocuitorul familiei de produse Windows şi nu a lui OS/2. La acest punct de turnură a apărut necesitatea specificării Windows API înainte de asta Windows API există numai ca o intefaţă pe 16 biţi.

Chiar dacă Windows API va introduce multe funcţionalităţi noi care nu erau disponibile pe Windows 3.1 Microsoft a decis să facă noul API compatibil cu numele pe 16 biţi ale funcţiilor Windows API şi să folosească tipurile de date oricând era posibil pentru a uşura portabilitatea aplicaţiilor Windows pe 16 biţi pe Windows NT

**3. Bibliografie**:

http://social.technet.microsoft.com/wiki/contents/articles/10224.posix-and-unix-support-in-windows.aspx

<http://en.wikipedia.org/wiki/POSIX#Parts_before_1997>

<https://www.cs.cmu.edu/~mihaib/articole/unix-2/unix-2-html.html>

 <http://linux.about.com/cs/linux101/g/POSIX__Portable.htm>

 <http://en.wikipedia.org/wiki/POSIX_Threads>

 <http://en.wikipedia.org/wiki/Windows_API>

 Linux Shell Scripting Tutorial - A Beginner's handbook Michael Horowitz - A comparison of Linux and Windows

Figura 1 - <http://social.technet.microsoft.com/wiki/cfs-filesystemfile.ashx/__key/communityserver-wikis-components-files/00-00-00-00-05/6740.POSIX_5F00_Subsystem.gif>

Figura 2 - <http://social.technet.microsoft.com/wiki/cfs-filesystemfile.ashx/__key/communityserver-wikis-components-files/00-00-00-00-05/2313.POSIX_5F00_Files_5F00_Interact.gif>