BIBLIOTEKI ŁĄCZONE DYNAMICZNIE (DLL)

Biblioteka łączona dynamicznie jest to funkcja systemu operacyjnego umożliwiająca oddzielne przechowywanie procedur wykonywalnych (zazwyczaj służących jako określone funkcje lub zestawy funkcji) w postaci plików z rozszerzeniem nazwy dll. Procedury te są ładowane tylko wówczas, gdy są potrzebne programowi, który je wywołuje.

Zalety:

- Jedna kopia dla wielu procesów – oszczędność pamięci
- Modyfikacja funkcji w DLL'u nie powoduje potrzeby przekompilowania całej aplikacji
- Możliwość zmiany na DLL'a na nowszy
- Niezależność od języka programowania
- Skraca się czas kompilacji i linkowania dużego programu
- Raz napisany kod może być używany w wielu programach
- dll-e mogą być ładowane na żądanie, z dowolnego katalogu, a po użyciu usuwane z pamięci, co usprawnia gospodarowanie nią
- Biblioteki dll są znakomitym narzędziem do pisania filtrów (np. obsługa wielu formatów plików), a także znanych z przeglądarek plug-ins (fragmentów kodu umożliwiających rozszerzenie aplikacji o nowe możliwości

Wady:

- Jeśli zastosujesz funkcję LoadLibrary, będziesz musiał wywoływać GetProcAddress dla każdej funkcji z biblioteki, którą zechcesz wywołać. Funkcja GetProcAddress pobiera adres wejścia do funkcji w danej bibliotece DLL. Zatem twój kod mógłby być nieco większy i nieco wolniejszy, lecz nie za wiele.
- tzw. *„dll hell"* - nadpisywanie bibliotek systemowych, sterowników, co może spowodować problemy w działaniu innych programów lub nawet załamanie systemu.

SYSTEM WINDOWS 2000 - SPOSOBY ŁADOWANIA ORAZ PRZYDATNE FUNKCJE SYSTEMOWE:

Load-time:

Podczas uruchamiania program używa informacji linkera umieszczonych w pliku do zlokalizowania nazw DLLi, które mają być używane przez proces. Następnie system próbuje zlokalizować te DLLe. Lista przeszukiwania zależy od wartości klucza „HKLM\System\CurrentControlSet\Control\Session Manager\SafeDllSearchMode", znajdującego się w rejestrze systemowym.

Jeśli wartość SafeDllSearchMode wynosi 1 (domyślnie), lista przeszukiwania jest następująca:

- Folder, z którego została uruchomiona aplikacja.
- Folder systemowy. Używa funkcji GetSystemDirectory do uzyskania pełnej ścieżki dostępu do katalogu.
- Folder systemowy 16-bit..

W Windows Me/98/95: folder ten nie istnieje

- Folder windows . Używa funkcji GetWindowsDirectory do uzyskania pełnej ścieżki dostępu do katalogu.
- Dotychczasowy folder.
- Ścieżki dostępu do katalogów znajdujące się w zmiennej środowiskowej PATH

Jeśli wartość SafeDllSearchMode wynosi 1, lista przeszukiwania jest następująca:

Folder, z którego została uruchomiona aplikacja.

- Dotychczasowy folder.
- Folder systemowy. Używa funkcji GetSystemDirectory do uzyskania pełnej ścieżki dostępu do katalogu.
- Folder systemowy 16-bit..

W Windows Me/98/95: folder ten nie istnieje

- Folder windows . Używa funkcji GetWindowsDirectory do uzyskania pełnej ścieżki dostępu do katalogu.
- Ścieżki dostępu do katalogów znajdujące się w zmiennej środowiskowej PATH .

W Windows XP: domyślna wartość klucza HKLM\System\CurrentControlSet\Control\Session Manager\SafeDllSearchMode wynosi 0

Jeśli system nie może zlokalizować pliku DLL, to wyświetla okienko dialogowe z błędem. W przeciwnym wypadku system mapuje DLL do wirtualnej przestrzeni adresowej procesu.

Następnie system wywołuje funkcje punktu wejścia. Funkcja otrzymuje kod wskazujący, że proces ładuje plik DLL. Jeśli funkcja punktu wejścia nie zwróci wartości „TRUE", system kończy proces i wyświetla komunikat o błędzie.

Run-Time:
Kiedy aplikacja wywołuje funkcję LoadLibrary lub LoadLibraryEx, system próbuje zlokalizować plik DLL (szczegóły tego procesu zostały już opisane wyżej). Jeśli plik zostanie znaleziony, system mapuje moduł DLL do wirtualnej przestrzeni adresowej procesu i zwiększa licznik referencji. Jeśli wywołana funkcja LoadLibrary lub LoadLibraryEx określa DLL, którego kod jest już zmapowany w wirtualnej przestrzeni adresowej procesu, to funkcja po prostu zwróci uchwyt do DLL i zwiększy licznik referencji.
Następnie system wywołuje funkcje punktu wejścia jako kontekst wątku o nazwie LoadLibrary lub LoadLibraryEx. Funkcja punktu wejścia nie jest wywoływana, jeśli DLL był już załadowany przez wywołanie funkcji LoadLibrary lub LoadLibraryEx bez wcześniejszego użycia funkcji FreeLibrary.
Jeśli system nie może znaleźć pliku DLL lub funkcja punktu wejścia zwróciła wartość „FALSE", to funkcja LoadLibrary lub LoadLibraryEx zwraca wartość „NULL". W przeciwnym wypadku funkcja LoadLibrary lub LoadLibraryEx zwraca uchwyt do modułu DLL. Proces może użyć tego uchwytu do zidentyfikowania pliku DLL podczas wywoływania go przy pomocy funkcji GetProcAddress, FreeLibrary, lub FreeLibraryAndExitThread.
Funkcja GetModuleHandle zwraca uchwyt do pliku DLL , który możemy użyć w funkcjach GetProcAddress, FreeLibrary lub FreeLibraryAndExitThread, jednakże funkcji tej możemy użyć tylko wtedy, gdy moduł DLL, został już zmapowany do przestrzeni adresowej procesu przez ładowanie sposobem load-time, lub przez wcześniejsze wezwanie funkcji LoadLibrary lub LoadLibraryEx. Funkcja GetModuleFileName odbiera ścieżkę dostępu do modułu powiązaną z uchwytem zwróconym przez funkcje GetModuleHandle, LoadLibrary lub LoadLibraryEx.
Proces może użyć funkcji GetProcAdress do uzyskania adresu wyeksportowanej funkcji z pliku DLL, używając uchwytu do modułu tego DLL'a zwróconego przez funkcję LoadLibrary lub LoadLibraryEx, GetModuleHandle.
Jeśli DLL nie jest już potrzebny możemy użyć funkcji FreeLibrary lub FreeLibraryAndExitThread, które pozwalają nam na zwolnienie zbędnego pliku z pamięci. Różnica między FreeLibrary i FreeLibraryAndExitThread polega na tym, że ta druga funkcja oprócz zwolnienia pliku z pamięci kończy wątek procesu.

DEFINIOWANIE PUNKTU WEJŚCIA
System Windows wywołuje funkcję punktu wejścia zawsze, gdy:
- biblioteka DLL została załadowana po raz pierwszy.
- biblioteka DLL jest usuwana z pamięci
- w tym samym procesie zostaje utworzony nowy wątek
- wątek jest usuwany w tym samym procesie

Funkcja ta przyjmuje trzy parametry, z których tylko dwa pierwsze są istotne:
- **hInstDLL** jest uchwytem modułu biblioteki DLL.
- **reason** może przyjmować jedną z czterech wartości:
  - **DLL_PROCESS_ATTACH -** Biblioteka DLL otrzymuje tą wartość, gdy po raz pierwszy zostaje umieszczona w obszarze adresowym procesu.
  - **DLL_PROCESS_DETACH -** Biblioteka DLL otrzymuje tą wartość przy usuwaniu jej z obszaru adresowego procesu.
  - **DLL_THREAD_ATTACH -** Biblioteka DLL otrzymuje tą wartość, gdy dany proces tworzy nowy wątek.
  - **DLL_THREAD_DETACH -** Biblioteka DLL otrzymuje tą wartość, gdy wątek jest usuwany z procesu.

SYSTEM LINUX
Program korzystający z dynamicznie ładowanej biblioteki w pierwszej kolejności powinien wywołać funkcję **dlopen**. Prototyp tej funkcji wygląda następująco:

void *dlopen (const char *filename, int flag);
Funkcja ta ładuje z pliku wskazywanego przez wskaźnik filename, bibliotekę dynamiczną i zwraca uchwyt do załadowanej biblioteki dynamicznej. Jeśli filename nie jest ścieżką absolutną (np. nie rozpoczyna się od "/"), to plik jest poszukiwany w następujących miejscach:
- Rozdzielonej dwukropkami liście katalogów, zdefiniowanej w zmiennej środowiskowej LD_LIBRARY_PATH.
- Liście bibliotek podanej w /etc/ld.so.cache.
- /lib, a potem w /usr/lib.

Argument flag może mieć jedną z następujących wartości :

- **RTLD_LAZY -** co, oznacza, że niezdefiniowane symbole są rozszyfrowywane podczas wywołania biblioteki dynamicznej,
- **RTLD_NOW -** co oznacza, że rozszyfrowywane są wszystkie niezdefiniowane symbole zanim zakończy się funkcja dlopen, zakończy się niepowodzeniem jeżeli wszystkie symbole są zdefiniowane,
- **RTLD_GLOBAL -** co oznacza, że symbole zewnętrzne zdefiniowane w bibliotece będą udostępniane kolejno ładowanym bibliotekom.

Jeśli dlopen z jakiejś przyczyny zawiedzie, to zwraca NULL. Czytelny dla człowieka napis, zawierający opis ostatniego błędu, który pojawił się w którejś z funkcji dl (dlopen, dlsym lub dlclose), można wyciągnąć przy użyciu dlerror().
dlerror zwraca NULL, jeśli od czasu inicjalizacji lub poprzedniego wywołania dlerror nie wystąpił błąd. (Wywołanie dlerror() dwa razy pod rząd zawsze spowoduje, że drugie wywołanie zwróci NULL.)

Funkcja **dlsym** pobiera uchwyt biblioteki dynamicznej zwróconej przez dlopen i zakończoną zerem nazwę symbolu, zwracając adres pod którym załadowany jest ten symbol. Jeśli symbol nie zostanie znaleziony dlsym zwraca NULL. Prototyp tej funkcji :
void *dlsym(void *handle,char *symbol);

Funkcja dlclose wyładowuje bibliotekę dzieloną. Ilość wywołań dlclose powinna być równa wywołaniom funkcji dlopen.

REJESTR SYSTEMU WINDOWS

Rejestr jest centralną bazą danych przeznaczoną do przechowywania w ujednolicony sposób wszystkich informacji konfiguracyjnych systemu operacyjnego i aplikacji. Zawiera on kompletny zestaw wpisów dotyczących ustawień takich elementów, jak programy obsługi (sterowniki) urządzeń, pamięć czy programy obsługi sieci. Narzędzia systemu operacyjnego pozwalają na dostęp do niego zarówno z komputera, którego dotyczy, jak również poprzez sieć.

Architektura Rejestru jest połączeniem idei znanych z Windows 3.x plików INI oraz występującego także w tym środowisku Rejestru (przechowującego znacznie mniej informacji). Rejestr Windows 2000 ujednolica i łączy te mechanizmy. Sekcjom plików INI odpowiadają tzw. klucze Rejestru, a poszczególnym wpisom -- wartości Rejestru. Podstawowe różnice to wprowadzenie struktury hierarchicznej (drzewiastej) i umożliwienie korzystania z wartości binarnych. Podstawową zaletą wprowadzenia
w systemie Windows 2000 struktury drzewiastej rejestru jest przechowywanie wszelkich informacji w ujednolicony sposób, co pozwala nam dowolne modyfikowanie wszelkich danych w rejestrze.
Niestety, wprowadzenie koncepcji jednolitego Rejestru jako bazy danych konfiguracyjnych nie pozwala na natychmiastowe "uwolnienie się" od takich plików, jak WIN.INI, SYSTEM.INI, ATM.INI, CONFIG.SYS czy AUTOEXEC.BAT. Pomimo tego, że ich rola w Windows 2000 sprowadzona została do pozostałości po poprzednim środowisku pracy, ich usunięcie z dysku twardego może wręcz uniemożliwić uruchomienie komputera.

BUDOWA REJESTRU

ELEMENT	OPIS	DODANIE ELEMENTU	ZMIANA ELEMENTU
Kategoria	Jeden z sześciu kluczy głównych,	Brak możliwości	Brak możliwości
Klucz	Jeden z folderów Rejestru. Pomijając hierarchiczną strukturę, klucze porównać można do sekcji plików INI	Podobnie jak przy tworzeniu folderu w Eksploratorze, wybranie klucza nadrzędnego, a	Wybranie z menu kontekstowego polecenia Zmień nazwę (F2)
Wartość	Widoczna tylko w prawym oknie Edytora Rejestru, w kolumnie Nazwa. Każdy klucz może zawierać jedną lub więcej wartości. Można je porównać do tej	Wybranie klucza, do którego należeć ma wartość, a następnie z kontekstowego menu klucza **Nowy | Wartość** ciągu. Można również	Jeżeli chodzi o zmianę nazwy wartości, wybieramy z jej menu kontekstowego Zmień nazwę (F2).
Dane	Dane porównać można do wpisu w pliku INI po prawej stronie znaku **=**. Dane mogą być tekstowe, binarne lub typu DWORD. Dwa ostatnie rodzaje	Dane są ściśle przyporządkowane wartościom.	Podwójne kliknięcie na nazwie wartości (ENTER).

Rejestr ma strukturę hierarchiczna. Jest on baza danych o konfiguracji systemu operacyjnego. Rejestr posiada pięć głównych kluczy:

- ***HKEY_CLASSES_ROOT (HKCR)*** – klucz służy do ustalania powiązań pomiędzy rozszerzeniami (formatami) plików i obsługującymi je aplikacjami.
- ***HKEY_CURRENT_USER (HKCU)*** – za pomocą tego klucza konfigurowany jest system dla aktualnie zalogowanego użytkownika (profil użytkownika). Klucz jest generowany za każdym razem, gdy użytkownik loguje się do systemu. Do generowania profilu wykorzystywane są informacje zawarte w podkluczu HKEY_USERS oraz plikach ntuser.dat oraz ntuser.man znajdujących się w profilu użytkownika.
- ***HKEY_LOCAL_MACHINE (HKLM)*** – za pomocą tego klucza można otrzymać informacje o sprzęcie, systemie operacyjnym, aplikacjach zainstalowanych w komputerze. Wartość klucza jest niezależna od użytkownika, aplikacji czy procesu.
- ***HKEY_USERS (HKU)*** – klucz ten służy do zapisywania kopii profilu każdego użytkownika, który kiedykolwiek zalogować się do systemu.
- ***HKEY_CURRENT_CONFIG (HKCC)*** – klucz służy do ustawiania bieżącej konfiguracji systemu.

Każdy klucz grupuje podklucze i wartości dotyczące jednej dziedziny działania systemu operacyjnego. Swoja struktura rejestr przypomina strukturę drzewiasta. Poniżej jest przedstawione przeznaczenie głównych podkluczy:

- HKLM\HARDWARE – wszystkie konfiguracje sprzętowe systemu. Podklucz jest generowany podczas pierwszego uruchamiania systemu.
- HKLM\SAM – bazy danych użytkowników. Nazwa klucza wynika z zachowania zgodności z wcześniejszymi wersjami Windows.
- HKLM\SECURITY – zabezpieczenia systemu takie jak uprawnienia aktualnie zalogowanych użytkowników, zasady zabezpieczeń, itp. Podklucz nie podlega modyfikacji.
- HKLM\SOFTWARE – aplikacje zainstalowane w systemie. Windows 2000 także jest aplikacja, wiec można znaleźć informacje o konfiguracji systemu.
- HKLM\SYSTEM – bieżąca sesja, ostatnia znana dobra konfiguracja.

TYPY DANYCH REJESTRU

Nazwa	**Typ danych**	**Opis**
Wartość binarna	REG_BINARY	Dane binarne. Większość informacji o składnikach sprzętowych jest przechowywana jako dane binarne i wyświetlana w Edytorze rejestru w formacie szesnastkowym.
Wartość DWORD	REG_DWORD	Dane reprezentowane przez liczbę o długości 4 bajtów (32-bitową całkowitą). Wiele parametrów sterowników urządzeń i usług jest tego typu i jest wyświetlanych w Edytorze rejestru w formacie binarnym, szesnastkowym lub dziesiętnym. Wartości pokrewne to DWORD_LITTLE_ENDIAN (najmniej znaczący bajt znajduje się pod najniższym adresem) i REG_DWORD_BIG_ENDIAN (najmniej znaczący bajt znajduje się pod najwyższym adresem).
Wartość ciągu rozwijalnego	REG_EXPAND_SZ	Ciąg danych o zmiennej długości. Ten typ danych obejmuje zmienne, których wartości są obliczane, jeśli program lub usługa korzysta z danych.
Wartość wielociągu	REG_MULTI_SZ	Ciąg wielokrotny. Wartości zawierające zestawienia lub wartości wielokrotne, zapisane zwykle w formie możliwej do odczytania przez ludzi. Wpisy są oddzielane spacjami, przecinkami lub innymi znacznikami.
Wartość ciągu	REG_SZ	Ciąg tekstowy o stałej długości.
Wartość binarna	REG_RESOURCE_LIST	Seria zagnieżdżonych tablic służących do przechowywania listy zasobów używanych przez sterownik urządzenia sprzętowego lub jedno z urządzeń fizycznych, którymi on steruje. Te dane są wykrywane i zapisywane w drzewie

		\ResourceMap przez system oraz wyświetlane w Edytorze rejestru w formacie szesnastkowym jako wartość binarna.
Wartość binarna	REG_RESOURCE_REQUIREMENTS_LIST	Seria zagnieżdżonych tablic służących do przechowywania listy zasobów sprzętowych, z których może korzystać dany sterownik urządzenia lub jedno ze sterowanych przez niego urządzeń fizycznych. Podzestaw tej listy system zapisuje w drzewie \ResourceMap. Te dane są wykrywane przez system i wyświetlane w Edytorze rejestru w formacie szesnastkowym jako wartość binarna.
Wartość binarna	REG_FULL_RESOURCE_DESCRIPTOR	Seria zagnieżdżonych tablic służących do przechowywania listy zasobów używanych przez fizyczne urządzenie sprzętowe. Te dane są wykrywane i zapisywane w drzewie \HardwareDescription przez system i są wyświetlane w Edytorze rejestru w formacie szesnastkowym jako wartość binarna.
Brak	REG_NONE	Dane bez określonego typu. Te dane są zapisywane w rejestrze przez system lub aplikacje i wyświetlane w Edytorze rejestru w formacie szesnastkowym jako wartość binarna.
Łącze	REG_LINK	Ciąg Unicode określający łącze symboliczne.
Wartość QWORD	REG_QWORD	Dane reprezentowane przez 64-bitową liczbę całkowitą. Te dane są wyświetlane w Edytorze rejestru jako wartość binarna i zostały wprowadzone po raz pierwszy w systemie Windows 2000.

Narzędzia do edycji rejestru.
System Windows 2000 dostarcza dwóch aplikacji do edycji rejestru:

- **regedit.exe -** Program ten występował w edycji Windows 95. Pozwala na wyszukiwanie kluczy i wartości danych oraz na zmianę i kopiowanie nazw kluczy. Posiada mechanizmy eksportowania i importowania danych. Edytor Rejestru jest zaawansowanym narzędziem do zmiany ustawień w rejestrze systemu, który zawiera informacje dotyczące działania komputera. Program wyświetla okno
  z hierarchiczna struktura w postaci drzewa.
- **regedt32.exe -** Program opracowano na potrzeby Windows NT 3.1. Posiada możliwości modyfikowania uprawnień dostępu do rejestru. Program umożliwia wyszukiwanie tylko kluczy. Po uruchomieniu programu edytor otwiera pięć okien. Każde z nich zawiera jeden główny klucz rejestru. Program posiada dodatkowo możliwości nadawania uprawnień do kluczy rejestru systemowego.

Kopiowanie parametrów wprowadzonych w rejestrze na inne komputery
Aby skopiować na inne komputery wprowadzone do Rejestru parametry powinniśmy skorzystać z dostępnej w Rejestrze funkcji eksportu. Edytor Rejestru pozwala bowiem na skopiowanie wybranych gałęzi bazy do pliku o rozszerzeniu REG, (pliki te są
o rozszerzeniu tekstowym co pozwala na łatwiejszy eksport oraz import). Aby następnie zaimportować tak zapisane parametry systemu, wystarczy dwukrotnie kliknąć utworzony plik. Najlepszym rozwiązaniem jest wyeksportowanie parametrów na dyskietkę, dzięki czemu mogą one być później wczytane na innych komputerach.
W celu zapisania odpowiednich parametrów wybieramy odpowiednią gałąź
i uaktywniamy funkcję Rejestr | Eksportuj plik Rejestru. Trzeba przy tym zwrócić uwagę, aby nie eksportować całego Rejestru, lecz tylko wybraną gałąź.

Kopia zapasowa rejestru
Podczas edycji rejestru należy zachować duża ostrożność. Niepoprawna edycja rejestru może spowodować poważne uszkodzenia systemu.

Przed wprowadzeniem zmian w rejestrze powinno się wykonać kopie zapasowa wszelkich cennych danych. Po uszkodzeniu systemu może być możliwa naprawa rejestru lub przywrócenie go do tej samej wersji, co podczas ostatniego udanego uruchomienia komputera. W przeciwnym razie można doprowadzić do niestabilności systemu jego powtórna instalacje. Podczas ponownej instalacji systemu można utracić wszystkie dokonane zmiany, jak np. uaktualnienia dodatków Service Pack, które trzeba zainstalować ponownie osobno.

Najpierw oczywiście należy uruchomić edytora rejestru. W tym celu z menu Start należy wybrać opcję "Uruchom", wpisać regedit i nacisnąć ENTER. Zobaczymy okno e.r. Z górnego menu wybieramy opcję Rejestr, a następnie "eksportuj". Pojawi się standardowe okno dialogowe z tą różnicą, że na dole zobaczymy ramkę; upewnijmy się, że zaznaczona jest opcja WSZYSTKO. Wprowadzamy nazwę
i klikamy OK. Aby wprowadzić kopię rejestru trzeba na nią podwójnie kliknąć. Pamiętajmy, że po zainstalowaniu jakiegoś programu musimy zrobić nową kopię. Inaczej mogą wystąpić problemy w jego działaniu.

Prawa dostępu do rejestru
System Windows 2000 daje możliwość nadawania praw dostępu do kluczy rejestru poszczególnym użytkownikom. Istnieją dwa prawa, które można przydzielić lub odmówić:

- Odczyt,
- Pełna kontrola.

Oprócz dwóch głównych uprawnień istnieje grupa uprawnień specjalnych:

- Badanie wartości
- Ustawianie wartości
- Utwórz podklucz
- Wylicz podklucze
- Powiadom
- Tworzenie łącza
- Usuń
- Zapisz DAC
- Zapisz właściciela.

Możliwości przeprowadzania inspekcji.

- *Pytaj o wartość:* Wszystkie próby odczytu wpisu wartości z klucza rejestru
- *Ustaw wartość:* Wszystkie próby ustawienia wartości wpisów w kluczu rejestru
- *Utwórz podklucz:* Wszystkie próby utworzenia kluczy podrzędnych dla wybranego klucza rejestru
- *Wylicz podklucze:* Wszystkie próby zidentyfikowania kluczy podrzędnych klucza rejestru
- *Powiadom Zdarzenia:* powiadamiające, pochodzące z klucza w rejestrze
- *Utwórz łącze:* Wszystkie próby utworzenia łącza symbolicznego dla określonego klucza
- *Usuń:* Wszystkie próby usunięcia obiektu rejestru
- *Zapisz DAC:* Wszystkie próby zapisania poufnych danych listy kontroli dostępu dla klucza
- *Zapisz właściciela:* Wszystkie próby zmiany właściciela wybranego klucza
- *Kontrola odczytu:* Wszystkie próby otwarcia poufnej listy kontroli dostępu dla klucza

Gdzie jest przechowywany rejestr??
W systemach Windows 2000 i Windows XP Rejestr jest przechowywany w kilku plikach, w folderach \windows\system32\config i \Documents and Settings\nazwa użytkownika.

Zalety użytej w Windows 95-2000 koncepcji Rejestru są następujące:

1. Jedno miejsce przechowywania danych wykorzystywanych przy enumeracji i konfigurowaniu urządzeń, ich sterowników, aplikacji oraz samego systemu operacyjnego.
2. System operacyjny automatycznie tworzy kopię ostatniej poprawnej konfiguracji wykorzystywanej przy uruchamianiu komputera.
3. Jeżeli zastosowana zostanie opcja wykorzystywania indywidualnych profili konfiguracyjnych użytkowników, odpowiednie dane mogą być przechowywane na serwerze sieci. Pozwala to na korzystanie z własnej konfiguracji niezależnie od tego, której końcówki sieci używamy.
4. Administratorzy sieci mogą korzystać ze specjalnych narzędzi umożliwiających wprowadzenie wszelkich zmian konfiguracyjnych z dowolnej końcówki sieci, niezależnie od jej rodzaju.

Interfejs programistyczny, przydatne funkcje systemowe Win32 API.

Poniższa tabel przedstawia kilka najważniejszych funkcji systemowych Win32 API.

RegCloseKey	Zamyka otwarty klucz rejestru
RegConnectRegistry	Łączy się z rejestrem z innego komputera
RegCreateKeyEx	Tworzy nowy podklucz
RehDeleteKey	Usuwa klucz z rejestru
RegDeleteValue	Usuwa wartość z klucza
RegDisablePredefinedCache	Wyłącza preyfinowaną tablicę wskaźników HKEY_CURRENT_USER dla procesów
RegFluschKey	Zapisuje zmiany rejestru
RegNotifyChangeKeyValue	Sygnalizuje zmiany klucza lub wartości

RegOpenKeyEx	Otwiera istniejący klucz rejestru w rozszerzeniu Win32
RegQueryInfoKey	Zwraca informacje o kluczu
RegQueryValueEx	Zwraca wartość (w typie danych dla Win32)
RegSetValueEx	Przypisanie wartości do klucza

ROZRUCH SYSTEMU LINUX

Przyjmujemy: Linux znajduje się na dysku HDD-0 (/dev/hda)
BIOS odczytuje sektor rozruchowy MBR (Master Boot Record) na dysku HDD-0. Przestrzeń sektora MBR zawiera program rozruchowy np. Lilo, Grub lub inny. Program rozruchowy proponuje użytkownikowi wybór parametrów startu systemu, czyli wyboru kernela, uruchomienia framebuffer-a itd. Parametry rozruchu zawarte są np. w pliku konfiguracyjnym Lilo - **/etc/lilo.conf**
W zależności od wyboru (humoru) , użytkownik wywołuje (wybiera) określoną sekwencje startową. Bliższe informację na temat konfiguracji lilo w man lilo, docs lilo. Następnym krokiem jest załadowanie jądra systemu. Jądro każdego systemu operacyjnego jest rdzeniem całego oprogramowania systemowego. Jedynie bardziej fundamentalne zadanie spoczywa na sprzęcie. Ktoś mądry kiedyś powiedział : "Linux to Kernel" i na pewno się nie pomylił! Jądro ma bardzo wiele zadań. Podstawowym jest uniezależnienie oprogramowania od sprzętu oraz zapewnienie środowiska pracy oprogramowania. Środowisko musi obsługiwać takie elementy jak dostęp do dysku, pamięć wirtualną, sieć, wielozadaniowość, itd.
Lista zadań jest ogromna, bliżej dowiesz się o tym czytając dokumentację załączoną np. w źródłach kernela lub na stronie http://kernel.org/ Jądro Linux-a jest niewielką częścią systemu, mimo tego jest najważniejsze. Również jest bardzo krytycznym elementem.Zawieszenie się jądra jest bardzo niebezpieczną sytuacją, gdyż zazwyczaj prowadzi do uszkodzenia całego systemu.
Na szczęście Linux może pochwalić się niesamowicie dużą stabilnością jądra. Okres pomiędzy poszczególnymi restartami systemu dla Linuksa mierzy się w latach. Oczywiście taki stan rzeczy dotyczy serwerów. Maszyny domowe, zazwyczaj są włączane na kilka godzin. Mimo tego jądro przetrzymuje ten test całkiem nieźle. Po załadowaniu jądra jest podłączamy główny system plików.
Gdy system podłączy główny system plików zostaje uruchomiony pierwszy proces - **Init**.

Proces **Init** jest patronem wszystkich procesów (tzw "Ojciec procesów"), jego identyfikator zawsze wynosi 1. Jeśli wystąpi błąd procesu **Init**, to podąży za nim cały system, resetując system.
Proces Init w systemie spełnia dwa zadania:
1. Init zawsze jest ostatnim procesem macierzystym, gdyż proces init nigdy nie umiera. System zawsze jest pewny, że Init istnieje i może w razie potrzeby odwołać się do niego. Odwołanie się do Init występuje zazwyczaj wtedy, gdy ginie jakiś proces, zanim wszystkie jego potomne procesy zakończą swoje działanie. Wtedy wszystkie osierocone procesy potomne dziedziczą Init jako swój proces macierzysty.

2. Init obsługuje uruchamianie/przełączanie poszczególnych poziomów pracy systemu (rulevels), oraz wywoływanie poszczególnych programów po wejściu w określony poziom.
Takie zachowanie definiowane jest w pliku **/etc/inittab**
Format każdej lini konfiguracyjnej wygląda następująco
id:poziom_uruchomieniowy:akcja:proces

id - Niepowtarzalna sekwencja od 1 do 4 znaków, która identyfikuje wpis w pliku **/etc/inittab** np. 2:bbbbbb:ccccccc:ddddddd

poziom_uruchomieniowy - Poziom na którym proces powinien zostać wywołany. Wpis 245 mówi o tym że proces ma zostać wykonany na poziomie 2, 4 i 5

akcja - Opisuje typ akcji
Wartość pola akcja może przyjąć następujące atrybuty:
- respawn - Proces będzie uruchamiany ponownie jeśli zakończy swoje działanie
- wait - Proces będzie uruchamiany, gdy zostanie osiągnięty dany poziom. Init czeka na jego zakończenie.
- once - Proces będzie uruchamiany, gdy zostanie osiągnięty dany poziom. Init nie będzie czekał na jego zakończenie, przejdzie do następnego poziomu.
- boot - Proces wywołany podczas startu systemu. Pole z poziomem uruchomienia jest ignorowane.
- bootwait - Proces będzie uruchamiany przy starcie systemu. Init będzie czekał na jego zakończenie.

- ondemand - Proces będzie uruchomiony, gdy wystąpi określone żądanie poziomu uruchomieniowego (a, b, c). Nie następuje zmiana poziomu.
- initdefault - Określony domyślny poziom dla Init podczas startu systemu.
- sysinit - Proces bedzie uruchomiony przed jakimkolwiek wpisem boot lub bootwait.
- powerwait - Jeśli Init otrzyma sygnał o problemach z zasilaniem, to zacznie uruchamiać dany proces. Init przed kontynuacją poczeka na zakończenie tego procesu.
- powerfail - Jeśli Init otrzyma sygnał o problemach z zasilaniem, to zacznie uruchamiać dany proces. Init nie będzie czekac na zakończenie procesu.
- powerokwait - Proces zostanie uruchomiony dokładnie jak przy akcji powerwait, oraz zostanie wczytany string z /etc/powerstatus.
- ctrlaltdel - Proces zostanie uruchomiony, gdy użytkownik wciśnie z klawiatury kombinację przycisków CTRL-ALT-DELETE

Na początku system uruchamia poziom 1. uruchomi wszystkie skrypty przewidziane z poziomu 1 z parametrem start.
Następnie Init ustawi system na poziom 3 (l3:3:wait:/etc/rc.d/rc 3) lub 5 (l5:5:wait:/etc/rc.d/rc 5) i zostaną uruchomione wszystkie skrypty z poziomu trzeciego /etc/rc.d/rc3.d lub z poziomu piątego z parametrem start.
Init wie kiedy ma uruchomić poszczególny proces, po łączu symbolicznym do katalogu init.d .
gdyż:
w katalogu **/etc/rc.d/** znajduje się katalog **/init.d/** oraz katalogi(dowiązanie symboliczne do katalogu init.d) o nazwach: od rc0.d do rc6.d, gdzie cyfra odpowiada poszczególnemu poziomowi startu. Gdy Init ustawi np. poziom 3, wówczas odczytuje wszystkie skrypty z katalogu symbolicznego o nazwie rc3.d .Skrypty znajdujące się w katalogach **rc*.d** są również dowiązaniem symbolicznym do skryptów w katalogu **init.d**
Zamiast używać rzeczywistych nazw skryptów istniejących w **/etc/rc.d/init.d/**, symboliczne linki poprzedzone są prefiksami:

- S - jeśli skrypt uruchamia serwis
- K - jeśli skrypt wyłącza lub zatrzymuje serwis

Sposób użycia tych dwóch liter jest ściśle określony, aby Init potrafił określić użycie skryptu.
Jeśli wykonamy skrypt S90crond to będzie oznaczać żę wykonujemy skrypt **/etc/rc.d/init.d/crond start**
Wielokrotnie kolejność wykonania skryptów jest bardzo ważna np. demon httpd nie może być uruchomiony gdy nie jest skonfigurowany interfejs sieci!
Dlatego, aby wymusić prawidłową kolejność startu poszczególnych skryptów, stosuje się pomiędzy prefiksem (S lub K) a dalszą nazwą skryptu wstawkę, w postaci dwucyfrowej liczby. Niższy numer jest wykonywany przed wyższym.
Kiedy, skrypty z **/etc/rc.d/init.d/** są uruchamiane po przez określony katalog poziomu pracy systemu, wywoływane są w kolejności alfabetycznej.
Najpierw są uruchamiane skrypty zaczynające się na **K**, czyli system włącza określone serwisy/demony.
Następnie skrypty zaczynające się na **S**, czyli system startuje kolejno serwisy przewidziane dla danego poziomu.
Wykonanie wszystkich skryptów z poziomu 3 lub 5, oznacza uruchomienie systemu. Teoretycznie dysponujemy pełnosprawnym systemem. Ale tylko teoretycznie. Jesteśmy pozbawieni możliwości zalogowania się do systemu.
Dalszym krokiem jest uruchomienie przez Init wirtualnych terminali (mingetty) :
1:2345:respawn:/sbin/mingetty tty1
Teraz możemy się już zalogować do systemu.

ROZRUCH SYSTEMU WINDOWS

W pierwszym sektorze dysku twardego jest umieszczony Główny Rekord Startowy MBR (Master Boot Record). MBR zawiera kod wykonywalny zwany głównym kodem startowym, podpis dysku oraz tablicę partycji. Posiada zatem wszystkie informacje potrzebne do uruchomienia systemu operacyjnego. BIOS ładuje MBR do pamięci, uruchamia go i przekazuje sterowanie programowi NTLDR.
Partycja startowa – to z niej ładuje się jadro, przechowywane w katalogu System32 w postaci pliku NTOSKRNL.EXE. Pozostałe katalogi znajdujace się na partycji startowej : WINNT, Program Files, Documents and Settings, Temp. Katalogi te, z wyjątkiem ukrytego pliku PAGEFILE.SYS(plik stronicowania), znajdującego się w katalogu macierzystym partycji startowej, składają się na partycję startową.

Proces uruchamiania systemu Windows 2000
Pliki startowe Windows 2000 w kolejności ich ładowania podczas uruchamiania systemu :

- • Ntldr
- • Boot.ini

- • Bootsect.dos - domyślnie nie występuje we wszystkich systemach Windows 2000 podczas uruchamiania
- • Ntdetect.com - skanuje cały sprzęt podstawowy w komputerze
- • Ntbootdd.sys - domyślnie nie występuje we wszystkich systemach Windows 2000 podczas uruchamiania
- • Ntoskrnl.exe - znajduje się w winnt\System32
- • Hal.dll - znajduje się w winnt\System32

Etapy uruchamiania Windows 2000
- POST (Power-On Self Test)
- Początkowy proces uruchamiania
- Ładowanie plików startowych
- Wybór systemu operacyjnego
- Wykrywanie sprzętu
- Wybór profilu sprzętowego
- Ładowanie jądra
- Proces logowania do systemu operacyjnego

POST
Faza ta odpowiedzialna jest za to co dzieje się z komputerem przed uruchomieniem systemu operacyjnego. Gdy włączamy komputer najpierw następuje inicjalizacja karty graficznej, następnie system sprawdza pamięć zainstalowana w komputerze, wykrywa dyski twarde oraz niektóre z zainstalowanych urządzeń.

Początkowy proces uruchamiania
BIOS szuka pierwszego sektora dysku startowego, w którym umieszczony jest główny rekord startowy. MBR w Windows 2000 szuka i ładuje plik NTLDR.

Ładowanie plików startowych
NTLDR ładuje pliki startowe systemu operacyjnego, wyświetla menu wyboru systemu operacyjnego oraz menu wyboru profilu sprzętowego, a także wykrycie urządzeń zainstalowanych w komputerze. NTLDR nie zadziała poprawnie, jeśli w systemie nie będą występować pliki boot.ini oraz ntdetect.com.

Wybór systemu operacyjnego
NTLDR szuka pliku boot.ini , który jest zwykłym plikiem tekstowym informującym NTLDR, jakie systemy operacyjne są zainstalowane w komputerze, gdzie się znajdują, który z nich powinien zostać domyślnie uruchomiony oraz jaki ma być czas oczekiwania na dokonanie wyboru. W pliku boot.ini można użyć kilku przełączników np.

• /basevideo uruchamia system operacyjny ze standardowymi sterownikami VGA. Tę metodę uruchamiania obsługują wszystkie karty graficzne VGA
• /maxmem:x – ogranicza ilość pamięci, jaka może zostać użyta przez Windows 2000. Przełącznik ten używany jest gdy podejrzewa się używanie uszkodzonych kości pamięci
• /debug – ładuje debugger, dzięki czemu można go użyć po nawiązaniu połączenia z serwerm
• /sos – wyświetla na ekranie wszystkie ładowane sterowniki urządzeń. Używany do wyszukiwania urządzenia mogącego powodować problemy z siecią.
• /baudrate=xx - Używane do debuggowania. Ustawia prędkość transmisji podczas debuggowania serwera. Automatycznie zostanie użyta także opcja /debug.
• /crashdebug – Ładuje debugger, ale nie uruchamia go aż do momentu wystąpienia błędu jądra Windows 2000
• debugport=comx – Wybiera port komunikacyjny używany do debuggowania. Automatycznie zostanie użyta także opcja /debug.
• /fastdetect=comx – program ntdetect.com nie próbuje wykryć myszy w podanym porcie komunikacyjnym. Ten przełącznik jest używany wtedy, gdy do portu szeregowego jest podłączony kabel sterujący UPS-em
• /notebug – wyłącza debuggowanie
• /numpoc=x – ogranicza ilość procesów używanych w wieloprocesorowym systemie
• /pae – włącza rozszerzenie adresu fizycznego

Przykładowy plik boot.ini:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Microsoft  Windows  2000  Professional"
/fastdetect
```

Posiada on dwie sekcje pierwsza *boot loader* definuje czas wyświetlania menu startowego oraz domyślny system operacyjny, który należy załadować.
Sekcja *operating systems* zawiera wyszczególnione systemy operacyjne zainstalowane na komputerze.
Multi lub *SCSI* definuje którą z magistral (IDE, SCSI) wykorzystuje system przy dostępie do parycji systemowej Windows 2000. Pierwszy kontroler IDE oznaczmy *multi(0)*, a drugi *multi(1)*.
Disk – identyfikator SCSI urządzenia końcowego. *Disk* dla urządzeń *multi* jest zawsze równy zero.
Rdisk definiuje urządzenie dyskowe podłączone do kontrolera *multi*. Pierwszy napęd magistrali kontrolera jest urządzeniem 0, drugi urządzeniem 1.
Partition definiuje partycję dysku z której startować będzie system operacyjny.

Wykrywanie sprzętu
NTLDR uruchamia aplikację zwana ntdetect.com, której zadaniem jest wykrycie urządzeń zainstalowanych w komputerze i pobranie informacji dotyczących ich konfoguracji. Dokonuje następujących sprawdzeń:
- czas i data zachowane w systemowym CMOS,
- rodzaje magistral w systemie (np. ISA, PCI, EISA, MCA) i identyfikatory urządzeń podłączonych do tych magistral,
- liczba, rozmiar i typ napędów dyskowych w systemie,
- rodzaje myszy połączonych z systemem,
- liczba i rodzaj portów (np. równoległe, USB, komunikacyjne) skonfigurowanych w systemie.

Ładowanie jądra
Następnie Ntldr zaczyna ładować pliki z bootowalnej partycji, które są potrzebne do inicjalizacji jądra
- Ładuje odpowiednie obrazy kernel i HAL (domyślnie Ntoskrnl.exe i Hal.dll).
- Wczytuje informacje zgromadzone w systemowym pliku w katalogu \Winnt\System32\Config\System i dzięki temu wie, które sterowniki urządzeń muszą być załadowane aby ukończyć proces bootowania.
- Skanuje pamięć wewnętrzną i plik wspomniany powyżej i lokalizuje wszystkie bootujące sterowniki urządzeń. Sterowniki te są wskazane w rejestrze poprzez startową wartość SERVICE_BOOT_START.
- Dodaje plik systemowego sterownika, odpowiedzialnego za implementację kodu dla każdego rodzaju partycji (FAT, FAT32 lub NTFS), na której znajduje się katalog instalacyjny, aby stworzyć listę bootujących sterowników do załadowania.
- Ładuje bootujące sterowniki. Aby pokazać postęp ładowania Ntldr uaktualnia pasek postępu pokazywany poniżek tekstu "Starting Windows". Jeżeli jest włączony przełącznik /SOS w pliku Boot.ini, Ntldr nie pokazuje pasku postępu, ale pokazuje nazwę aktualnie ładowanego sterownika. Sterowniki te są w tm momęcie ładowane, ale nie inicjowane.
- Przygotowuje rejestry CPU to wykonania Ntoskrnl.exe.

Ostatnią czynnością wykonywaną przez NTLDR jest ładowanie jądra Windows 2000(ntoskrnl.exe) i przekazanie mu sterowania. Sygnalizuje to pojawienie się graficznego ekranu startowego Windows 2000. Ntoskrnl ładuje plik warstwy uniezależnienia, hal.dll (HAL) oraz konfigurację systemu zapisaną w Rejestrze. Potem zaczyna ładować usługi systemowe oraz niskopoziomowe sterowniki urządzeń.
Wszystkie te czynności podzielone są na dwa etapy: faze 0 i faze 1.
Główną cechą fazy 0 jest nieobsługiwanie przerwań. Celem jej jest zbudowanie elementarnych struktur potrzebnych do przywołania usług w fazie 1. Wywołuje funkce HAL, która jest odpowiedzialna za przygotowanie kontrolera przerwań systemowych każdego CPU w systemie do obsługi i konfiguracji przerwań zegara, który jest urzywany dla obliczenia zegara CPU. Następnie inicjowane są:
- menadżer pamięci,
- menadżer obiektów,
- monitor ochrony odniesień,
- menadżer procesów,
- menadżer Plug and Play.

Menadżer pamięci tworzy strone tablic i wewnętrzne struktury danych które są niezbędne do dostarczania prostych usług pamięci. Tworzy i rezerwuje obszar dla systemu plików pamięci podręcznej oraz tworzy obszary pamięci dla stronicowanych i niestronicowanych pól. Pola te urzywane są m.in. przez podsystemy wykonawcze, jądro i sterowniki urządzeń. Menadżer obiektów tworzy przestrzen nazw oraz tabele uchwytów. Monitor ochrony odniesień przygotowywuje pierwsze znamiona dla zadań procesów wstępnych. Menadżer procesów określa procesy i wątek typów objektów, a także wstawia listy do ścieżek aktywnych procesów i wątków. Innymi jego zadaniami są stworzenie obiektów procesowych, procesu Systemowego oraz wątku systemu do wykonywania programu standardowego fazy 1.

Faza 1 wykonuje wiele czynności, m.in. wywołuje funkce umożliwiwiającą obsługe przerwań, uruchamia sterownik wideo (Bootvid.dll), menadżera mocy, pozostałe procesory, dyspozytor i menadżera wyjścia/wejścia. Jego wykonywanie jest najważniejszą i najdłużej trwającą czynnością tej fazy. Najpierw inicjuje różnorodne struktury wewnętrzne i tworzy typy objektów sterowników i urządzeń. Następnie sterowniki urządzeń systemu startowego są ładowane i uruchamiane. Ostatnik krokiem fazy 1 jest stworzenie procesu podsystemu menadżera sesji (Smss). Smss jest odpowiedzialny za stworzenie środowiska trybu urzytkownika, które dostarcza interfejsu dla Windows 2000.

Podczas trwania tej fazy na ekranie monitora wyświetlany jest pasek postępu ekranu startu Windows 2000.

Jednymi z procesów biorącyni udział w starcie systemu są Smss (menadżer konfiguracji) i Csrss.Głównymi zadaniami Smss są:

- tworzenie objektu portu LPC i dwoch wątków do czekania na żądania klienta,
  - definiowanie symbolicznych odnośników dla nazw urządzeń MS-DOS'a (takie jak COM1 i LPT1),
  - uruchomianie programu Autochk służącego do sprawdzania i naprawy danych na dysku,
  - otwieranie znnych bibliotek DLL,
  - tworzenie dodatkowych plików stronicowania,
  - inicjowanie rejestru,
  - tworzenie parametrów środowiska systemu,
  - ładowanie część trybu jądra podsystemu Win32,
  - uruchamianie procesów podsystemu zawierających program Csrss,
  - rozpoczęcie procesu logowania (Winlogon),
  - tworzenie portów LPC dla wiadomośći zdarzeń debugera i wątków do nasłuchiwania na wybranych portach.

Proces logowania do systemu operacyjnego

Ostatnim etapem uruchamiania systemu jest włączenie podsystemu logowania Windows 2000 (winlogon.exe) sygnalizowane poprzez pojawienie się powitalnego ekranu logowania.

Rozpoczynając prace z systemem na komputerze z systemem Windows 2000 musimy podać w oknie logowania identyfikator użytkownika oraz prawidłowe hasło dostępu. Windows sprawdza czy podane informacje są zgodne z danymi o autoryzowanych użytkownikach, zapisanymi w wewnętrznej bazie danych. Jeśli wszystko jest prawidłowe użytkownik zostanie wpuszczony. Jeżeli poda błędne dane na ekranie pojawi się komunikat o błędzie i należy zalogować się jeszcze raz.

Konta użytkowników służą do identyfikacji poszczególnych użytkowników w systemie i uzyskania dostępu do zasobów. Kiedy dany użytkownik loguje się po raz pierwszy do systemu tworzony jest profil użytkownika zawierający zestaw plików i folderów przeznaczonych do wyłącznej dyspozycji tego użytkownika. Czyli jest to miejsce gdzie system zapisuje wszystkie osobiste dane użytkownika i informacje o różnych ustawieniach (np. zawartość folderu Moje dokumenty) w folderze Documents and Settings.

Za każdym razem kiedy użytkownik loguje się do systemu, jego profil jest lokalnie buforowany tzn. że gdy następnym razem użytkownik zaloguje się do tego samego systemu, poprzez sieć zostaną pobrane jedynie te elementy profilu, które uległy zmianie.

W tym czasie Winlogon tworzy stacje inicjacji okna i objektów desktop, ładuje GINA DLL oraz tworzy proces menadżera kontroli usług (SCM) (\Winnt\System32\Services.exe), który ostatecznie załadowuje wszystkie usługi i sterowniki urządzeń ważne dla Autostartu, i lokalnego podsystemu ochrony autoryzacji (Lsass).

Start powłoki systemowej

Po zweryfikowaniu nazwy i hasła użytkownika zostaje załadowany interfejs graficzny Windows 2000. System uruchamia programy zdefiniowane w Autostarcie oraz w następujących kluczach rejestru:

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices