Služby

Operačný systém Windows podporuje špeciálny typ aplikácií zvaných služby. Služby sú normálne spustiteľné aplikácie, ktoré obsahujú dodatočnú infraštruktúru. Práve táto infraštruktúra umožňuje služby kontrolovať a monitorovať pomocou *Service Control Manager* (SCM). SCM umožňuje administrátorovi službu spustiť, zastaviť, pozastaviť alebo spustiť jej pokračovanie, a to buď lokálne alebo vzdialene. SCM takisto monitoruje službu a môže ju automaticky reštartovať alebo dokonca reštartovať celý počítač v prípade, že služba bola nečakane ukončená.

Všetky serverové aplikácie dodávané spolu so systémom Windows sú implementované ako služby. *Microsoft Management Console* poskytuje *Services* snap-in, a tým umožňuje administrátorovi kontrolovať všetky nainštalované služby s použitím bežného rozhrania.

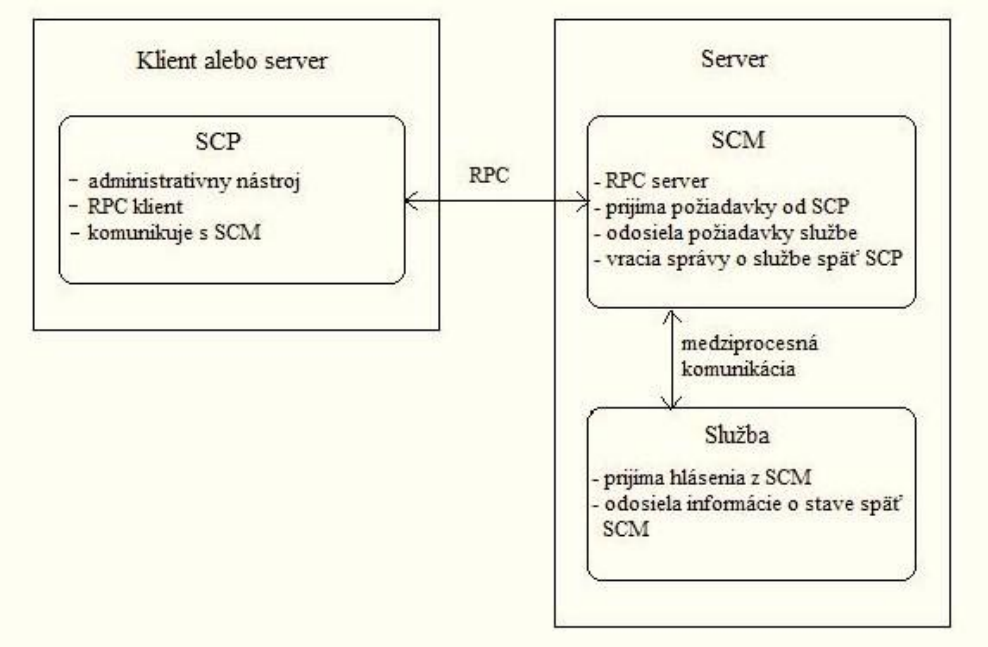
Komunikačná architektúra služieb

Aby služba pracovala, sú potrebné tri typy komponentov:

- *Service Control Manager* – Každý operačný systém Windows je dodávaný spolu s komponentom zvaným *Service Control Manager*. SCM je súčasťou súboru Services.exe a je automaticky spustený pri štarte systému a ukončený pri vypnutí systému. SCM beží so systémovými privilégiami a poskytuje jednotný a bezpečný spôsob kontrolovania služieb. Je zodpovedný aj za komunikáciu s rôznymi službami.

- *Služba* – Služba je aplikácia obsahujúca infraštruktúru nevyhnutnú pre komunikáciu s SCM, ktorý posiela službe hlásenia, či sa má služba zapnúť, zastaviť, pozastaviť, pokračovať alebo vypnúť. Služba tiež volá špeciálne funkcie, ktoré zasielajú správy o jej stave naspäť k SCM.

- *Service Control Program (SCP)* – SCP je aplikácia, ktorá zvyčajne predstavuje užívateľské rozhranie, ktoré umožňuje užívateľovi zapnúť, zastaviť, pozastaviť, spustiť pokračovanie alebo inak ovládať nainštalovanú službu. SCP volá špeciálne funkcie, prostredníctvom ktorých komunikuje s SCM.



Obrázok 7-1: Komunikačná architektúra služieb

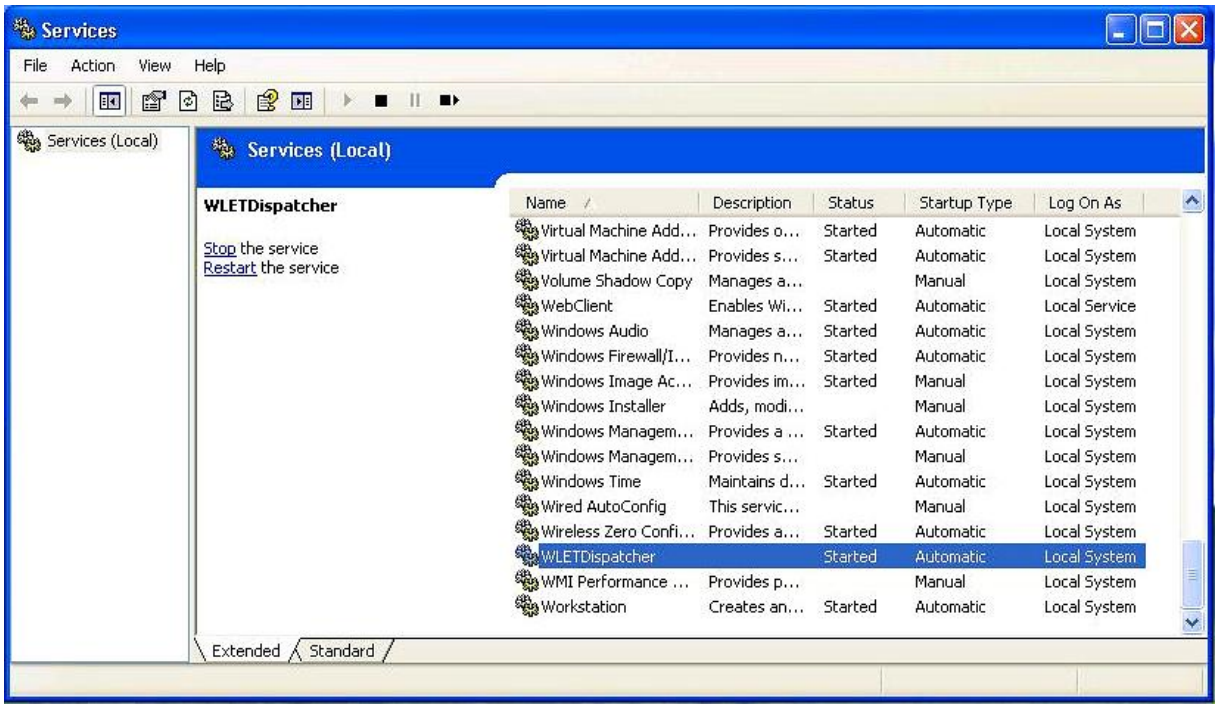
Obrázok 7-1 znázorňuje, ako všetky tieto komponenty navzájom komunikujú. Za povšimnutie stojí najmä to, že SCP aplikácie nekomunikujú so službami priamo, ale vždy cez SCM. Táto architektúra robí vzdialenú administráciu transparentnú vzhľadom k SCP a službám. Je možné naimplementovať aj takú architektúru a taký protokol, že by SCP aplikácia komunikovala priamo so službou, ale komunikačný kód by ste si museli napísať sami. V prípade operačného systému Windows je to však zbytočné, nakoľko už obsahuje SCM.

Services snap-in

Services snap-in (obr. 7-2) je SCP aplikácia, s ktorou sa môžete stretnúť najčastejšie. Tento snap-in zobrazuje zoznam všetkých služieb nainštalovaných na cieľovej stanici. Kolónky *Name* a *Description* obsahujú meno každej služby a informatívny popis jej funkcie. Kolónka *Status* indikuje, či je služba v stave *Started*, *Paused* alebo *Stopped*. Ďalej kolónka *StartupType* špecifikuje, kedy má SCM službu prebudiť a kolónka *LogOnAs* obsahuje bezpečnostný kontext použitý službou v prípade, že služba beží. Táto informácia je uložená v SCM databáze, ktorá sa nachádza v registri pod kľúčom:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
```

K tomuto kľúču by sa nikdy nemalo pristupovať priamo. S databázou uloženou pod týmto kľúčom manipuluje SCP volaním špeciálnych funkcií. Priama zmena obsahu tohto kľúča by totižto viedla k nepredvídateľným výsledkom. Pri inštalácii produktu, ktorý obsahuje aj službu, je inštalačný program tohto produktu vlastne SCP, ktorý pridáva informácie o inštalovanej službe do SCM databázy.



Obrázok 7-2: Services snap-in

Architektúra služieb

Dodatočná infraštruktúra, ktorá zo serverovej aplikácie robí službu, umožňuje, aby táto aplikácia bola spravovateľná vzdialene. Porozumieť architektúre služieb je spočiatku trochu zložité. Zložitosť spočíva v tom, že každý proces obsahuje najmenej dve vlákna a tieto vlákna musia navzájom

komunikovať. Je preto nutné sa vysporiadať so synchronizáciou vlákien a medzivláknovou komunikáciou. Musí sa tiež zobrať do úvahy, že jeden spustiteľný súbor môže obsahovať viacero služieb. To je dané tým, že mnohokrát je implementácia týchto služieb veľmi jednoduchá a bolo by neefektívne, keby každá z nich musela spustiť vlastný proces s jeho vlastným adresovým priestorom a dodatočnou réžiou. Kvôli tejto réžii Microsoft povoľuje, aby jeden spustiteľný súbor obsahoval niekoľko služieb. Príkladom je súbor *Services.exe*, ktorý obsahuje okolo 20 služieb.

Pri vytváraní služby sa musí počítať s tromi typmi funkcií:

- *Vstupný bod procesu* – Vstupným bodom procesu je štandardná funkcia `(w)main` alebo `(w)WinMain`. V prípade služby táto funkcia inicializuje proces ako celok, a potom volá špeciálnu funkciu, ktorá spojí proces s lokálnym systémovým SCM. V tomto bode SCM preberá kontrolu nad primárnym vláknom pre svoje vlastné potreby. Váš kód získa kontrolu naspäť až keď všetky služby v spustiteľnom súbore boli zastavené.

- *Funkcia* `ServiceMain` – Pre každú službu obsiahnutú vo vašom spustiteľnom súbore musíte implementovať funkciu `ServiceMain`. Aby mohla byť služba spustená, SCM vytvorí vlákno vo vašom procese, ktoré vykoná vašu funkciu `ServiceMain`. Pri navrátení vlákna z funkcie `ServiceMain` SCM považuje službu za zastavenú. Funkcia nemusí byť pomenovaná `ServiceMain`, môžete ju pomenovať ako chcete (bližšie vysvetlenie bude uvedené neskôr).

- *Funkcia* `HandlerEx` – Každá služba musí mať nejakú funkciu `HandlerEx`, ktorá je asociovaná s touto službou. SCM komunikuje so službou práve pomocou volania funkcie `HandlerEx`. Kód obsiahnutý vo funkcii `HandlerEx` je vykonávaný primárnym vláknom procesu. Funkcia `HandlerEx` buď vykonáva nevyhnutné akcie, alebo musí sprostredkovávať inštrukcie od SCM vláknu, ktoré vykonáva funkciu `ServiceMain`, použitím nejakej formy medzivláknovej komunikácie. Každá služba môže mať svoju vlastnú `HandlerEx` funkciu alebo viacero služieb (v jednom spustiteľnom súbore) môže zdielať tú istú funkciu `HandlerEx`. Jeden z parametrov, s ktorými je funkcia `HandlerEx` volaná, určuje, s ktorou službou chce SCM komunikovať. Opäť platí, že funkcia nemusí byť pomenovaná `HandlerEx`, ale môže mať ľubovoľný názov (bližšie vysvetlenie bude uvedené neskôr).

Vstupný bod procesu: `(w)main` alebo `(w)WinMain`

Keď administrátor zadá pokyn k spusteniu služby, SCM zistí, či spustiteľný súbor obsahujúci požadovanú službu už beží alebo nie. Ak nie, spustí ho. Za inicializáciu procesu je zodpovedné primárne vlákno procesu. Inicializácia služby by mala byť vykonaná v príslušnej funkcii `ServiceMain`. Po inicializácii procesu musí funkcia kontaktovať SCM, ktorý preberá kontrolu nad procesom. Aby funkcia mohla kontaktovať SCM, musí najskôr alokovať a inicializovať pole štruktúr `SERVICE_TABLE_ENTRY`:

```
typedef struct _SERVICE_TABLE_ENTRY{
    PTSTR serviceName;
    LPSERVICE_MAIN_FUNCTION serviceMainFunc;
    } SERVICE_TABLE_ENTRY, *LPSERVICE_TABLE_ENTRY;
```

Prvý člen označuje interný názov služby a druhý člen je adresa funkcie `ServiceMain`. Pole štruktúr `SERVICE_TABLE_ENTRY` je inicializované v závislosti na počte služieb, ktoré obsahuje spustiteľný súbor, nasledovne:

```
SERVICE_TABLE_ENTRY ServiceTable[] = {
    {TEXT("ServiceName1"), ServiceMain1},
    {TEXT("ServiceName2"), ServiceMain2},
    {NULL, NULL}
    };
```

Uvedené pole je inicializované v prípade, že spustiteľný súbor obsahuje dve služby. V poslednej štruktúre poľa musia byť vždy obidva členy `NULL`, aby bolo jasné, že sa jedná o koniec poľa. Po inicializácii poľa sa proces pripojí k SCM, a to volaním funkcie *`StartServiceCtrlDispatcher`*:

```
BOOL StartServiceCtrlDispatcher(
    CONST SERVICE_TABLE_ENTRY *pServiceTable
    );
```

Volaním tejto funkcie a odovzdaním adresy poľa štruktúr proces určuje, ktoré služby sú v ňom obsiahnuté. V tomto bode SCM vie, ktorú službu sa snaží spustiť a pri prechádzaní poľa sa ju snaží nájsť. V okamihu, keď je služba nájdená, je vytvorené vlákno a začína sa vykonávať funkcia *`ServiceMain`*, ktorej adresa je získaná z poľa štruktúr.

Funkcia *`StartServiceCtrlDispatcher`* sa nevráti pokiaľ všetky služby obsiahnuté v danom spustiteľnom súbore nie sú zastavené. Pokiaľ aspoň jedna zo služieb beží, SCM kontroluje primárne vlákno procesu. Zvyčajne toto vlákno nič nerobí a je v spánkovom režime, aby zbytočne neplytvalo drahým časom procesoru. Ak sa administrátor pokúsi spustiť ďalšiu službu implementovanú v tom istom spustiteľnom súbore, SCM nezačne novú inštanciu spustiteľného súboru, ale namiesto toho komunikuje s primárnym vláknom procesu a opäť prehľadáva zoznam služieb. Keď je služba nájdená, vytvorí sa nové vlákno, ktoré bude vykonávať príslušnú funkciu *`ServiceMain`*.

Interne systém sleduje, ktoré služby vo vnútri procesu sa vykonávajú. Pri každom ukončení služby systém kontroluje, či nejaká služba ešte stále beží. Iba v prípade, že boli ukončené všetky služby, volá funkcia vstupného bodu vrátenie funkcie *`StartServiceCtrlDispatcher`*. Následne by mal kód proces upratať, a potom by sa mala vrátiť funkcia vstupného bodu, a tým ukončiť proces.

Funkcia *`ServiceMain`*

Každá služba v spustiteľnom súbore musí mať svoju vlastnú funkciu *`ServiceMain`*:

```
VOID WINAPI ServiceMain(
    DWORD argc,
    PTSTR *argv
    );
```

SCM spúšťa službu vytvorením nového vlákna, ktoré sa začne vykonávaním funkcie *`ServiceMain`*. V tomto bode sa dostávame k vysvetleniu, prečo názov tejto funkcie nie je podstatný. Názov funkcie *`ServiceMain`* môže byť ľubovoľný vzhľadom na to, že dôležitá je adresa funkcie, ktorá je zaznamenaná v `SERVICE_TABLE_ENTRY` ako jej druhý člen. Každopádne nesmú byť dve funkcie *`ServiceMain`* v jednom spustiteľnom súbore s tým istým názvom. V prípade, že áno, dôjde k chybe a projekt sa nepodarí skompilovať. Funkcia *`ServiceMain`* je volaná s dvomi parametrami. Tieto parametre vytvárajú mechanizmus, ktorý umožňuje administrátorovi spustiť službu s nejakými parametrami príkazového riadku pomocou funkcie *`StartService`*. V praxi sa to však nevyužíva. Lepšou voľbou je nakonfigurovať službu pomocou nastavení obsiahnutých v nasledujúcom kľúči (časť *`ServiceName`* by mala byť nahradená aktuálnym názvom služby):

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ServiceName\Parameters
```

Prvou úlohou funkcie *`ServiceMain`* je oznámiť SCM adresu funkcie *`HandlerEx`*. To sa realizuje volaním funkcie *`RegisterServiceCtrlHandlerEx`*:

```
SERVICE_STATUS_HANDLE RegisterServiceCtrlHandlerEx(
    PCTSTR serviceName,
    LPHANDLER_FUNCTION_EX handlerEx,
    PVOID context
    );
```

Prvý parameter obsahuje názov služby, pre ktorú je nastavovaná funkcia *HandlerEx*. Hodnota tohto parametru sa musí zhodovať s názvom služby použitým pri inicializácii poľa štruktúr SERVICE_TABLE_ENTRY a odovzdaným funkcii *StartServiceCtrlDispatcher*. Druhý parameter je adresa funkcie *HandlerEx*. Posledný parameter je hodnota definovaná užívateľom a odovzdaná funkcii *HandlerEx*.

Návratovou hodnotou funkcie *RegisterServiceCtrlHandlerEx* je SERVICE_STATUS_HANDLE. Táto hodnota umožňuje SCM jednoznačne identifikovať službu. Celá budúca komunikácia služby s SCM bude vyžadovať práve túto *handle* namiesto interného názvu služby. Dôležité je vedieť, že narozdiel od väčšiny *handles* v systéme, *handle* vrátená funkciou *RegisterServiceCtrlHandlerEx* nikdy nie je zatváraná vami.

Okamžite po návrate funkcie *RegisterServiceCtrlHandlerEx* by mala funkcia *ServiceMain* oznámiť SCM, že služba pokračuje v inicializácii. To je realizované volaním funkcie *SetService-Status*:

```
BOOL SetServiceStatus(
    SERVICE_STATUS_HANDLE hService,
    LPSERVICE_STATUS pServiceStatus
    );
```

Táto funkcia má ako prvý parameter *handle* identifikujúcu službu a ako druhý parameter adresu inicializovanej štruktúry SERVICE_STATUS. Definícia štruktúry SERVICE_STATUS je nasledovná:

```
typedef struct _SERVICE_STATUS{
    DWORD dwServiceType;
    DWORD dwCurrentState;
    DWORD dwControlsAccepted;
    DWORD dwWin32ExitCode;
    DWORD dwServiceSpecificExitCode;
    DWORD dwCheckPoint;
    DWORD dwWaitHint;
    }SERVICE_STATUS, *LPSERVICE_STATUS;
```

Všetky prvky štruktúry musia byť správne zadané, kým je štruktúra odovzdaná funkcii *Set-ServiceStatus*. Význam jednotlivých prvkov štruktúry:

- *dwServiceType* – označuje typ spustiteľného súboru obsahujúceho službu. Ak sa jedná o jedinú službu v spustiteľnom súbore, je tento člen nastavený na hodnotu SERVICE_WIN32-_OWN_PROCESS, v prípade viacerých služieb je to SERVICE_WIN32_SHARE_PROCESS. Počas životného cyklu služby by sa táto hodnota nemala nikdy meniť.

- *dwCurrentState* – najdôležitejší prvok štruktúry. Označuje aktuálny stav služby (napr. SERVICE_START_PENDING).

- *dwControlsAccepted* – určuje, aké kontrolné príkazy je služba ochotná dostávať od SCP. Napríklad keď chcete povoliť, aby SCP mohol zastaviť službu, nastavíte hodnotu na SERVICE_ACCEPT_STOP.

- *dwWin32ExitCode* a *dwServiceSpecificExitCode* – tieto dva prvky umožňujú službe nahlásiť chybový kód.

- d*wCheckPoint* a *dwWaitHint* – tieto prvky umožňujú službe nahlásiť jej postup.

Funkcia *ServiceMain* má 80 sekúnd na to, aby zavolala funkciu *SetServiceStatus*, inak si SCM bude myslieť, že zapnutie služby zlyhalo. Ak v danom procese nebežia žiadne ďalšie služby, SCM proces ukončí.

Po ukončení inicializácie služby by funkcia *ServiceMain* mala zavolať *SetServiceStatus* s aktuálnym stavom služby SERVICE_RUNNING. Teraz služba beží. Služba beží tak, že sa ocitne v slučke. Vo vnútri tejto slučky je vlákno služby pozastavené a čaká na požiadavku zo siete alebo hlásenie, ktoré oznámi službe, akú akciu má vykonať. Keď príde požiadavka, služba prebudí vlákno, vykoná požadovanú akciu a opäť sa vráti do slučky. Pokiaľ príde hlásenie, služba ho vykoná. Ak je hlásením príkaz k zastaveniu alebo vypnutiu služby, slučka je ukončená a funkcia *ServiceMain* sa vráti, čím ukončí vlákno. Ak sa jednalo o poslednú bežiacu službu, je ukončený aj proces.

Funkcia *SetServiceStatus* skúma hodnotu prvku *dwCurrentState* štruktúry SERVICE_STATUS. Ak je tento člen nastavený ako SERVICE_STOPPED, funkcia *SetServiceStatus* zatvorí *handle* služby, ktoré je jej prvým parametrom. Toto je dôvod, prečo sa *handle* získané funkciou *RegisterServiceCtrlHandlerEx* nikdy nesmie zatvárať explicitne. Ešte dôležitejšie však je nikdy nevolať funkciu *SetServiceStatus* po tom, čo bola volaná s aktuálnym stavom SERVICE_STOPPED. Pokus o takéto volanie by sa skončil výnimkou.

Funkcia HandlerEx

Každá služba v spustiteľnom súbore musí byť asociovaná s nejakou funkciou *HandlerEx*:

```
DWORD WINAPI HandlerEx(
    DWORD control,
    DWORD eventType,
    PVOID eventData,
    PVOID context
    );
```

Opäť platí, že názov tejto funkcie môže byť ľubovoľný, pretože nie je dôležitý. Podstatná je adresa funkcie, ktorá je odovzdaná ako parameter funkcii *RegisterServiceCtrlHandlerEx*.

Funkcia *HandlerEx* je volaná so štyrmi parametrami. Parameter *control* určuje požadovanú akciu alebo hlásenie. Parametre *eventType* a *eventData* ponúkajú špecifickejšie informácie o akcii alebo hlásení. Parameter *context* je hodnota definovaná užívateľom, pôvodne odovzdaná ako parameter funkcii *RegisterServiceCtrlHandlerEx*. Použitím tejto hodnoty môžete vytvoriť jedinú funkciu *HandlerEx*, ktorá by bola používaná všetkými službami z jedného spustiteľného súboru. Hodnota parametru *context* by mohla byť použitá pri rozhodovaní, s ktorou službou potrebuje funkcia *HandlerEx* komunikovať.

Funkcia *HandlerEx* sa využíva pri komunikácii SCP a služby. V prípade, že SCP chce kontrolovať službu, SCM to oznámi primárnemu vláknu procesu. Vlákno sa prebudí a volá príslušnú funkciu *HandlerEx*. SCM takisto posiela funkcii *HandlerEx* hlásenia o zariadení, hardwarovom profile alebo výkone, čím umožňuje službe, aby sa primerane rekonfigurovala a účastnila sa na prijatí alebo odmietnutí systémových zmien.

Návratová hodnota funkcie *HandlerEx* umožňuje navrátenie nejakých informácií naspäť k SCM. Ak funkcia *HandlerEx* nevie spracovať niektorý kontrolný kód obsiahnutý v prvom parametri, vracia hodnotu ERROR_CALL_NOT_IMPLEMENTED. Pri zamietnutí požiadavku vracia akýkoľvek *Win32* chybový kód. Pre hocijaký iný kontrolný kód by mala vrátiť hodnotu NO_ERROR.

Kontrolné kódy a hlásenie stavu služby

Funkcia *HandlerEx* je zodpovedná za spracovanie všetkých požadovaných akcií služby a všetkých upozornení. Kód určujúci o akú akciu, prípadne upozornenie, sa jedná, je prvým parametrom funkcie *HandlerEx*. Príkladom akcie môže byť SERVICE_CONTOL_STOP, SERVICE_CONTROL_PAUSE, atď. Upozornenia informujú službu o zaujímavých udalostiach v systéme. Napríklad kód SERVICE_CONTROL_PARAMCHANGE značí, že došlo k zmene konfiguračných parametrov služby.

Práca funkcie *HandlerEx* sa mení v závislosti na kontrolnom kóde, ktoré obdrží. Špeciálnu pozornosť si zasluhujú obzvlášť kódy vyžadujúce nejakú akciu. Keď funkcia *HandlerEx* obdrží niektorý z kódov SERVICE_CONTROL_STOP, SERVICE_CONTROL_SHUTDOWN, SERVICE_CONTROL_PAUSE

alebo SERVICE_CONTROL_CONTINUE, musí zavolať funkciu *SetServiceStatus*, aby potvrdila príjem tohto kódu a aby približne určila, koľko bude trvať zmena stavu. Potvrdenie prijatia daného kódu sa prejaví nastavením *dwCurrentState* člena štruktúry SERVICE_STATUS na hodnotu SERVICE-_STOP_PENDING, SERVICE_PAUSE_PENDING alebo SERVICE_CONTINUE_PENDING. Funkcia *HandlerEx* sa musí vrátiť do 30 sekúnd, inak si SCM bude myslieť, že služba prestala odpovedať. Čas potrebný k zmene stavu je určený pomocou členov *dwCheckPoint* a dwWaitHint štruktúry SERVICE_STATUS.

Ak už služba vykonala všetky akcie potrebné k zmene svojho stavu, opäť volá funkciu *SetServiceStatus*. Tentokrát nastaví člen dwCurrentStatus na hodnotu SERVICE_STOPPED, SERVICE_PAUSED alebo SERVICE_RUNNING. Pri hlásení hociktorej z týchto hodnôt by mali byť členy *dwCheckPoint* a *dwWaitHint* 0, pretože služba už ukončila zmenu svojho stavu.

Poznámka:

Po tom, čo služba volá funkciu SetServiceStatus s hodnotou SERVICE_STOPPED, SCM povolí službe bežať ešte 30 sekúnd. Ak služba aj po tomto čase ešte stále beží, SCM ukončí proces, v ktorom beží – pokiaľ v rámci tohto procesu nebeží už žiadna iná služba.

Pri shutdowne obdrží funkcia *HandlerEx* upozornenie SERVICE_CONTROL_SHUTDOWN. Služba by mala spraviť minimálny počet operácií nevyhnutných k uloženiu dát a mala by neodkladne volať funkciu *SetServiceStatus* a hlásiť SERVICE_STOPPED. Implicitne systém poskytne službe 20 sekúnd. Po tomto limite je ukončený SCM proces a počítač pokračuje v shutdowne.

Pri ostatných upozorneniach by funkcia HandlerEx mala spracovať upozornenie a vrátiť sa. Funkciu SetServiceStatus by ste mali volať iba v prípade, že sa mení stav služby.

Všimnite si, že napriek tomu, že kontrolné kódy a upozornenia obdrží funkcia *HandlerEx* v primárnom vlákne procesu, samotnú reakciu prevedie vlákno služby. Je preto vhodné vymyslieť pre vašu službu nejaký vhodný spôsob medzivláknovej komunikácie.

Ladenie služby

Ladenie služby je zložitejšie ako ladenie normálnej aplikácie, a to hneď kvôli niekoľkým dôvodom. Po prvé, ladiaci program nevie spustiť službu, to musí spraviť SCM. Po druhé, mnoho služieb sa spúšťa skôr, ako sa prihlási užívateľ. Preto je dobré pri ladení prepnúť automatickú službu na manuálnu. Po tretie, služba beží na svojej vlastnej stanici a pracovnej ploche, ktorá nie je viditeľná interaktívnemu užívateľovi.

Ako teda ladiť službu? Najlepším spôsobom je ladiť ju ako bežnú aplikáciu namiesto služby. Vo vnútri funkcie *(w)main* alebo *(w)WinMain* sa vytvorí špeciálny prepínač odkazujúci na parameter z príkazového riadku. Ak bude služba spustená s týmto parametrom, zavolá sa priamo funkcia *ServiceMain* namiesto volania funkcie *StartServiceCtrlDispatcher*. Táto technika má, samozrejme, niekoľko nevýhod:

- Ak spustiteľný súbor obsahuje viac služieb, môže sa ladiť iba jedna z nich.
- Spustiteľný súbor beží pod vaším účtom namiesto účtu, ktorý by použil SCM. Toto môže viesť k obmedzeniu prístupu k zdrojom, ktoré by za normálnych okolností boli prístupné.
- Nie je možné zasielať hlásenia o zadaní požiadavku na pozastavenie, pokračovanie, zastavenie alebo vypnutie služby, čo spôsobí nemožnosť testovať ich spracovanie.

Ďalšou možnosťou je pripojiť k službe počas jej behu debugger.

Použitá literatúra:

RICHTER, Jeffrey. CLARK, Jason D. *Programming Server-Side Applications for Microsoft Windows 2000*. 1. vyd. 2000. ISBN 0-7356-0753-2