ČVUT PRAHA

Fakulta elektrotechnická

Diplomová práce

2004 **Oto Válek**

Zadání diplomové práce

Název tématu: Implementace SMB protokolu pro Palm OS

Zásady pro vypracování: Implementujte SMB klienta ve formě VFS knihovny. Výsledkem práce bude funkční klient umožňující práci se vzdáleným diskem, podobně jako se pracuje s paměťovou kartou v PalmOS.

Vedoucí diplomové práce: Ing. Lukáš Mikšíček

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne ……. …….
podpis

Anotace

Účelem tohoto projektu je umožnit uživateli operačního systému PalmOS, používanému v mobilních zařízeních, pracovat s daty na síťovém disku, která server zpřístupňuje protokolem SMB. Aplikace pod PalmOS přistupují k souborovém systémům přes rozhraní VFS. Typickým využitím VFS rozhraní je práce s daty uloženými na paměťových kartách, lze ho však rozšířit o nové souborové systémy. Řešením je tedy implementovat klientskou stranu protokolu SMB ve formě VFS knihovny. Na takovou knihovnu je v systému PalmOS kladena řada požadavků a omezení, zejména paměťových. Na druhou stranu je očekávána korektní podpora protokolu SMB, včetně zabezpečené autentifikace, a efektivní řešení sémantických rozdílů mezi rozhraním VFS a protokolem SMB. Výsledkem práce je funkční knihovna a uživatelský ovládací panel, splňující tyto požadavky na ně kladené.

Annotation

The goal of this project is to provide a mobile PalmOS user with access to data stored on a SMB server. PalmOS applications use the VFS interface to work with file systems. Although a typical use of the VFS interface is a support for flash memory cards, it can be extended to support more file systems. Therefore, a solution of the project is to implement a SMB protocol client in the form of the VFS library. Such a library has to obey limitations of PDA's hardware, where low memory requirements are a priority. On the other hand, a correct support, including security features, of SMB protocol is expected. The semantic differences, between VFS interface and SMB protocol, are taken into account. The work resulted in the fully functional VFS library, as well as the user control panel.

Obsah

1 Úvod

Mobilní kapesní počítače nazývané PDA (Personal Digital Assistant) zažívají rychlý rozvoj. Jednou z výhod, které uživatelům přináší, je možnost mít svá data stále k dispozici. Zároveň se prudce rozvíjí i odvětví mobilních komunikací. Není problém být se svým PDA trvale připojen k síti. Nabízí se tak možnost pracovat kdekoliv nejenom s daty uloženými v paměti kapesního počítače, ale i s daty ve vzdáleném osobním počítači nebo na podnikovém serveru.

Cílem diplomové práce je toto nabídnout uživatelům PDA s operačním systémem PalmOS, který je dnes patří k nejrozšířenějším. Umožní jim se připojit k serveru protokolem SMB, běžným v sítích založených na MS Windows, ale zároveň podporovaným i alternativními operačními systémy.

Pro PalmOS je typické, že data jsou uložena v databázích. Aplikace, u nichž to má smysl, podporují navíc i tradiční souborové systémy na paměťových kartách, a to přes rozhraní VFS. Tento projekt tedy musí vytvořit vrstvu, předávající data mezi aplikací podporující VFS a vzdáleným SMB serverem. Tato vrstva musí být co nejúspornější co do využívání omezených zdrojů PDA, zejména paměti. Zároveň se však musí vypořádat s rozdíly mezi oběma stranami. Projekt jsem nazval SmbFS.

2 Rozbor problematiky

2.1 SMB/CIFS protokol

Historie a rozšíření, verze protokolu

Protokol SMB (zkratka pro Server Message Block) je protokol pro sdílení souborů, tiskáren a dalších zařízení (roury, API) především v lokálních sítích. Původně byl vytvořen firmou IBM v roce 1985, ale rozšíření přineslo až jeho použití společností Microsoft v sítích MSNET postavených na MS-DOSu v roce 1988 [3]. Jedná se o protokol typu client-server, request-response, to znamená, že server odpovídá na dotazy (zprávy) vyvolané klientem. Definováno je několik desítek běžných operací se soubory a adresáři, jejichž sémantika odpovídá stylu práce s nimi v moderním OS. Je to stavový protokol, po připojení je navázána relace se serverem, a na otevřené soubory se odkazuje přidělenou handle (rukojetí).

Během vývoje vzniklo několik verzí protokolu, které přidávají nové typy zpráv i chybových kódů, nicméně zachovávají do značné míry zpětnou kompatibilitu. Protože volbu protokolu i typů zpráv provádí klient, server by měl být univerzálnější a být připraven na všechny verze protokolu i zpráv. Proto je implementace klienta poněkud jednodušší než serveru. Protokol sám je poměrně složitý a byl nepříliš dobře dokumentovaný, zejména co se týkalo proprietárních rozšíření. Objevila se snaha Microsoftu a dalších společností [4], o lepší dokumentaci a úpravu pro použití v prostředí Internetu. Výsledkem je CIFS (Common Internet File System) [1].

V současné době všechny významné implementace (MS Windows 9x až XP, Samba) podporují tuto nejnovější verzi protokolu, identifikovanou řetězcem „NT LM 0.12“. Tato práce také používá tuto verzi, jak je popsána v dokumentu „Common Internet File System (CIFS) Technical Reference“ [1].

Architektura

SMB je aplikační protokol, který od nižších vrstev modelu OSI vyžaduje zaručené doručení zpráv ve správném pořadí. V původním nasazení pod MS-DOS to byl nejčastěji NetBIOS+IPX nebo NetBEUI, pod Windows 95/98/ME NetBIOS+TCP. Ze služeb NetBIOSu byla využita zejména jmenná služba pro překlad názvu serveru na jeho IPX/IP adresu. Standard CIFS však již definuje možnost použít k tomuto účelu internetový jmenný systém DNS. Tím potřebu NetBIOSu eliminuje na pouhé další hlavičky ke zprávám SMB a také umožňuje protokol nasadit v prostředí čisté TCP/IP sítě. Pro názornost uvádím upravené schéma přejaté z dokumentace projektu Samba [1]. Zvýrazněna je architektura v našem případě :

<table>
<tr><td>OSI</td><td colspan="4"></td><td>TCP/IP</td></tr>
<tr><td>Application</td><td colspan="4" rowspan="2">SMB</td><td rowspan="2">Application</td></tr>
<tr><td>Presentation</td></tr>
<tr><td>Session</td><td>NetBIOS</td><td rowspan="3">NetBEUI</td><td>NetBIOS</td><td>NetBIOS</td><td></td></tr>
<tr><td>Transport</td><td rowspan="2">IPX</td><td rowspan="2">DECnet</td><td>TCP & UDP</td><td>TCP/UDP</td></tr>
<tr><td>Network</td><td>IP</td><td>IP</td></tr>
<tr><td>Link</td><td>802.X</td><td>802.X</td><td>Ethernet V2</td><td>Ethernet/PPP</td><td>Ethernet</td></tr>
<tr><td>Physical</td><td colspan="4"></td><td></td></tr>
</table>

Protokol SMB poskytuje určitou úroveň zabezpečení. Při přihlašování k serveru klient prokazuje své oprávnění pomocí challenge/response principu. Na výzvu serveru reaguje klient odpovědí odvozenou z výzvy a platného hesla schématem využívajícím algoritmy MD4 [12] a DES [11]. Server autentifikovaný není. Během komunikace může autenticita jednotlivé zprávy být dále potvrzována kontrolním MD5 [13] součtem obsahu zprávy a tajné sdílené informace, vzniklé při přihlašování.

To znamená, že je protokol zabezpečený proti odposlechu hesla a změně obsahu předávaných dat. Proti falešné identitě serveru a odposlechu přenášených dat už zabezpečený není. Odpovídá to jeho vzniku jako protokolu pro lokální, tedy důvěryhodnou síť. Pro sdílení citlivých dat ve veřejné síti, jakou je Internet, je nutné řešit zabezpečení na nižší vrstvě - některou z technologií virtuální privátní sítě (VPN).

Existující implementace

SMB jako původně v podstatě proprietární protokol implementovaný v sítích typu Microsoft Networks se, přes nedostatek dokumentace, dočkal dalších implementací. Nejznámější je open-source projekt Samba [5], realizující jak klientskou, tak serverovou stranu. Vzniká od roku 1997 a nyní je k dispozici pro téměř všechny unixové platformy. Další implementací je jCIFS [6], knihovna pro klientskou stranu protokolu vytvořená kompletně v Javě.

Pro záměr této práce - implementaci protokolu v operačním systému PalmOS, se ani jedna ze zmíněných knihoven nejevila vhodná. Samba kvůli své značné rozsáhlosti a komplexnosti. jCIFS kvůli nepoužitelnosti jazyka Java pod PalmOS pro systémové aplikace.

2.2 Systém PalmOS

Historie a specifika

Systém PalmOS je jedním z nejrozšířenějších systémů pro zařízení PDA (Personal Digital Assistant) a SmartPhone (inteligentní mobilní telefon). Kromě tvůrce, Palm Computing, ho ve svých výrobcích používají i firmy Sony, Samsung, Garmin, a další.

Se svými 51% trhu ve druhém (čtvrtletí 2003) [7] konkuruje především systému PocketPC společnosti Microsoft, nasazeným v PDA firem Hewlett-Packard, Dell nebo Toshiba.

Hlavní výhodou PalmOS je hardwarová nenáročnost. Proto mohou PDA s PalmOS konkurovat nižší cenou. Výhodou je i snadný vývoj a tím i velký výběr aplikací. Výhodou PocketPC je naopak dobrá kompatibilita s desktopovým prostředím MS Windows a formáty MS Office.

V současné době (začátek roku 2004) je uváděna na trh verze PalmOS 6. Jako platforma pro vývoj této práce byl použit PalmOS 4 a úspěšně testována byla i pod PalmOS 5.Systém PalmOS totiž do značné míry zachovává zpětnou kompatibilitu, a to i přes změnu hardwarové platformy mezi verzemi 4 a 5 - přechod z procesoru M68 na ARM.

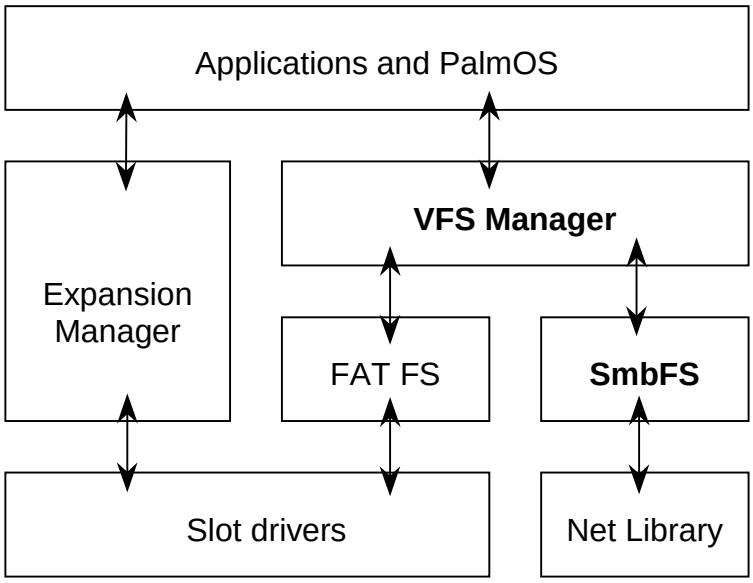
VFS rozhraní

Práce s daty v PalmOS je poměrně neobvyklá. Souborový systém je tvořen databázemi, které nejsou uspořádány v hiearchii adresářů. Je uložen v RAM. Existují dva druhy databází - programové (aplikace a knihovny) a datové. Databáze je dále členěna na záznamy volitelné a proměnlivé délky. Teprve obsah těchto záznamů je přístupný aplikačním rozhraním. To se děje zdánlivě složitým postupem, na blok paměti není odkazováno adresou, ale speciální rukojetí (handle) a během přístupu k němu musí být uzamčen a tak zjištěn ukazatel s dočasnou platností. To umožňuje systému právě nepoužívané bloky paměti přesouvat. Zároveň je třeba pro zápis používat speciální funkci (DmWrite), která kontroluje korektnost rozsahů adres. To umožňuje systému (který jinak ochranu paměti nepodporuje), udržet integritu dat. Výhodou tohoto databázového paradigmatu je snazší vývoj typické aplikace, možnost efektivní defragmentace paměti a rychlejší synchronizace a zálohování - provádí se po záznamech. Nevýhodou je nekompatibilita formátů datových souborů s ostatními platformami a nutnost existence konverzních programů. V souvislosti s tím je vhodné zmínit existenci souborového formátu, který se používá, je-li třeba databázi z PalmOS uložit na jiném OS. Je definován v [14] a používá přípony .pdb a .prc. Kromě distribuce aplikací a dat se hodí právě k ukládání databází na souborové systémy paměťových karet.

Kromě paměti pro data (Storage Heap), jejíž velikost je obvykle nad 2 MB, existuje samozřejmě i klasická dynamická paměť RAM (Dynamic Heap), kterou využívají běžící aplikace pro svá data, s nimiž pracují. Její velikost je typicky 256 kB, což je poměrně málo. Na druhou stranu, ale tato paměť neobsahuje žádny kód programu - ten je spouštěn přímo ze Storage Heap. Také s daty, uloženými v databázích lze snadno pracovat přímo na místě, bez jejich kopírování do dynamické RAM. To činí vývoj aplikací netradiční, ale výsledkem je, že aplikace pod PalmOS bývají i na pomalém HW relativně výkonné.

Na další problém se narazilo, když byly modely PDA vybaveny čtečkami paměťových karet (CompactFlash, SmartMedia, a další). Na těchto kartách je použit klasický souborový systém FAT, na který ostatní zařízení, např. digitální fotoaparáty ukládají data v běžných souborových formátech (JPEG). Tím by se stala data pro aplikace nedostupná, a proto bylo definováno rozhraní VFS (Virtual File System) a knihovna VFS Manager, který služby rozhraní poskytuje aplikacím.

Následující schéma převzaté z dokumentace PalmOS [15] ilustruje místo VFS Manageru a naší knihovny SmbFS v architektuře rozšiřujících modulů PalmOS (Palm OS expansion architecture) :



VFS Manager využívá další knihovny jako ovladače jednotlivých souborových systémů (FS library). Některé z nich (ty co pracují s paměťovými kartami), využívají služby Slot Manageru, který zpřístupňuje hardware čteček (viz schéma „PalmOS Expansion architecture“). Příkladem je knihovna FATFS. Mohou ale existovat i souborové systémy, které Slot Manager nepoužijí. To je případ knihovny HostFS, která v PalmOS emulátoru (PC aplikace), poskytuje přístup k disku hostitelského počítače. Ze zdrojového textu [8] této knihovny také vycházela tato práce, protože z něj byly patrné vstupní body FS knihovny, které jinak nejsou dokumentované.

Služeb VFS Manageru využívá operační systém i aplikace. Operační systém po vložení paměťové karty a zavolá funkci VFS Manageru pro připojení disku (funkce VFSVolumeMount). Aplikace si vyžádá seznam připojených disků a může k němu k souborům přistupovat (funkce VFSFileOpen, apod.). VFS Manager tato volání předává ovladači filesystému, tedy předmětu této práce. Hlavním úkolem tak bylo překonat určité rozdíly v sémantice i efektivitě souborových a adresářových služeb VFS a SMB. O tom se podrobněji zmíním v následujících kapitolách.

Aplikace využívající VFS

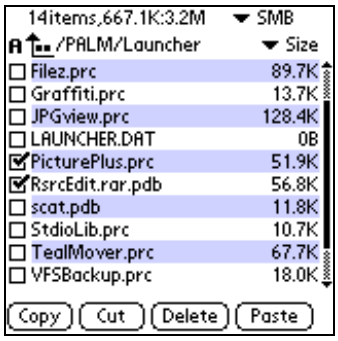
Zde uvedu stručný přehled některých aplikací, které využívají rozhraní VFS a které jsem použil pro testovaní funkčnosti vyvíjeného ovladače.

Launcher

Launcher je integrální součásti systému PalmOS, sloužící ke spouštění instalovaných aplikací. Zobrazuje jejich ikony a umožňuje je třídit do kategorií. Připojené VFS disky se zde objeví jako další kategorie. Na disku Launcher využívá jen adresář /PALM/Launcher, a programové databáze v něm pak tvoří obsah kategorie. Umožňuje také kopírování mezi diskem a interní pamětí, ale omezené jen na programové databáze aplikací.

FileProg v1.02

Aplikace k obecné správě souborů na VFS disku. Umožňuje běžné diskové operace (kopírování, přesun, přejmenování a mazání souborů a adresářů). Práce s více disky je řešena pomocí virtuální schránky, shodně jako u obdobných aplikací na PC.

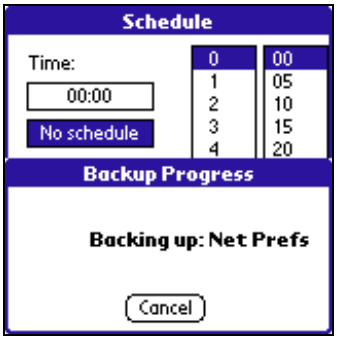


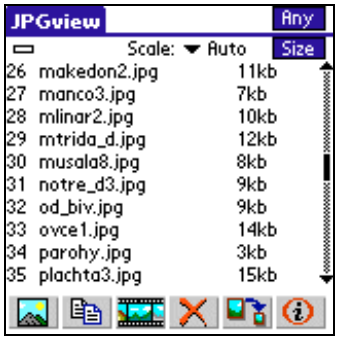
VFSBackup v1.0

Slouží k záloze a obnovení databází z interní pamětí na VFS disk. Lze zvolit množinu zálohovaných databází i naplánovat zálohu na určitý čas.

JPGView

Zobrazuje grafické soubory formátu JPEG z kteréhokoliv adresáře na disku, jednotlivě, nebo jako prezentaci. Vhodná pro práci s fotografiemi zapsanými na paměťovou kartu digitálním fotoaparátem.





VFSMark

Aplikace je určena k měření výkonu VFS disku. Provádí v opakovaných sériích různé diskové operace. Z naměřených časů pak sestaví index výkonnosti vztažený k referenčnímu modelu Palm m500 s originální 16 MB paměťovou kartou.

3 Řešení a implementace

3.1 Požadavky na knihovnu

Jak bylo řečeno dříve, existující implementace klientské strany protokolu SMB byly pro účely této práce nevhodné. PalmOS je systém s limitovanou dynamickou pamětí, knihovna nesmí být příliš rozsáhlá. Tuto paměť navíc využívá právě běžící aplikace, zatímco ovladač filesystému musí být v paměti také. Vzhledem k tomu, že systém PalmOS podporuje v podstatě jen jednu právě běžící aplikaci, jejíž tvůrci tak předpokládají, že se kapacita volné paměti nebude měnit, bylo žádoucí i co možná nejvíce omezit alokaci dynamické paměti.

Dalším aspektem je existence globálních statických proměnných, které se v projektu Samba (jako téměř v každém C/C++ programu) vyskytují. Ovladač VFS filesystému v PalmOS je programová knihovna, ve které se globální proměnné nemohou vyskytovat, respektive mohou, ale jen ve speciální struktuře, kterou udržuje systém v tabulce otevřených knihoven a jejíž adresu je nutné při každém volání funkcí knihovny zjišťovat (SysLibTblEntry(fsLibRefNum)->globalsP). To by si vyžádalo do stávajících implementací protokolu rozsáhlé zásahy.

Proto jsem se rozhodl implementovat pro protokol SMB zcela novou knihovnu, která bude tyto nedostatky řešit. Knihovna, v zájmu zjednodušení, narozdíl od projektu Samba, implementuje pouze nejnovější verzi protokolu „NT LM 0.12“. Vzhledem k tomu, že o použité verzi protokolu rozhoduje klient během připojování k serveru, a současné serverové implementaci tuto verzi podporují, nejedná se o výrazný nedostatek.

V zájmu ušetření nepříliš rozsáhlé dynamické paměti jsem se snažil co nejvíce využít paměť pro databáze (Storage Heap), a to alokované prostřednictvím volání FtrPtrNew. Takto alokovaní paměť má proti standardnímu postupu tu výhodu, že je v případě restartu systému automaticky uvolněna. Takto je to implementováno u vyrovnávací paměti pro čtení obsahu adresářů, u vyrovnávací paměti pro čtení souborů a u paměti pro komunikaci mezi uživatelským rozhraním (UI Panel) a vlastní knihovnou. Naopak ho nebylo možné použít u kombinovaného bufferu pro příjem a odesílání dat po TCP spojení. Síťový subsystém PalmOS (Net Library) vyžaduje jeho umístění v dynamické pamětí, neboť do něj sama zapisuje přímo, bez volání systémové funkce DmWrite.

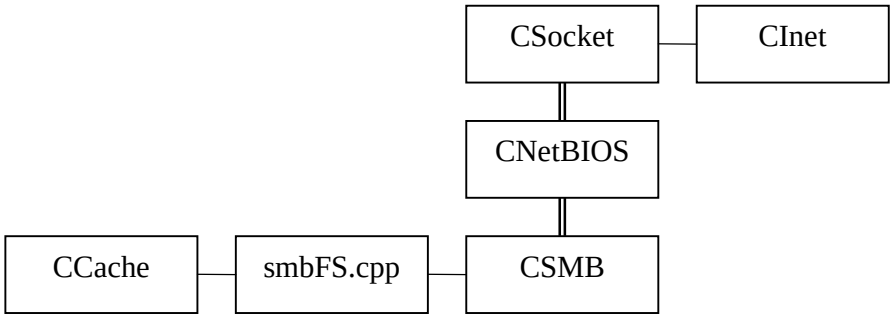
3.2 Architektura

Z hlediska PalmOS je aplikace tvořena třemi programovými databázemi. Vlastní knihovnou „SmbFS Library“, která je ovladačem filesystému pro VFS Manager. Dále Ovládacím panelem „SmbFS Panel“, který je z hlediska uživatale integrován do aplikace Preferences, určené k nastavování ostatních parametrů systému. Dále je součástí knihovna „DES Library“, do které jsem soustředil funkce pro šifrování DES a kódy MD4 a MD5, použité v autentifikačních schématech protokolu.

Ovládací panel komunikuje s knihovnou voláním funkce FSCustomControl, která je k účelu rozšíření rozhraní VFS přímo určena. Touto cestou panel především realizuje operace, které VFS Manager nedefinuje - zjišťování a nastavování parametrů jednotlivých předdefinovaných spojení a zjišťování informací o stavu spojení. Během testování na Simulátoru PalmOS 5 se však vyskytl problém s předáváním parametrů

odkazem. Proto byla pro tyto parametry použita sdílená paměť alokovaná systémovým voláním FtrPtrNew (tzv. Feature Memory). Pro připojování a odpojování jednotek využívá standardní volání VFS. Knihovnu „DES Library“ volá ovladač obvyklým způsobem, přes 8 exportovaných funkcí.

Z hlediska programovacího jazyka C++ se skládá z několika samostatných celků. Jádrem je implementace protokolu SMB ve formě C++ třídy, a to platformově nezávislá, lze ji tedy provozovat i na jiných OS. Je potomkem tříd CNetBIOS (implementující nejmenší nutný základ NetBIOSu) a CSocket (implementující rozhraní TCP), což odpovídá hiearchii Vrstev těchto protokolu v modelu OSI. Dále je zde samostatná třída CInet obalující funkce související s otevíráním a uzavíráním síťové knihovny a CCache, implementující souborovou read-ahead cache. Kód těchto tříd je v adresářovém stromu projektu v adresáři /libs . Vlastní knihovna, protože exportuje funkce, nemůže být ve formě třídy. Je to tedy klasický programový modul (smbFS.cpp). Dále uvedu jednoduché schéma, ilustrující hiearchii klíčových tříd a tohoto hlavního zdrojového modulu v projektu. Dvojitá čára značí dědičnost, jednoduchá prosté vkládání objektů nebo použití třídy v modulu.



3.3 Sémantické rozdíly

Ovladač, který je výsledkem této práce, funguje vlastně jako překlad funkcí rozhraní VFS na funkce protokolu SMB. Tato rozhraní se ale v mnohém liší, funkce mají odlišné parametry a způsob použití. Proto je nutné tyto sémantické rozdíly překonat, a naopak, sémantické podobnosti tvorbu ovladače ulehčují.

Otevírání souboru nebo adresáře (FSFileOpen)

Otevírání souboru je v obou případech podobné. Otevřený soubor je identifikován číslem, takzvanou rukojetí (handle). Toto číslo však nelze předat beze změny, protože s otevřených souborem musí ovladač udržovat další informace (např. offset v souboru) a tak nám musí handle, kterou vracíme VFS Manageru tuto strukturu (OpenFileType) identifikovat. Abych zabránil nutnosti udržovat speciální tabulku s relací handle -

adresa, jako handle pro VFS vrací FSFileOpen přímo adresu v RAM. Platnost této adresy je při dalších voláních, které ji mají jako parametr, ověřována přes pevně zvolenou 32-bitovou hodnotu (magic number) ukládanou na začátek struktury. Tato metoda je obvyklá hlavně v souborových formátech, například GZIP, nebo PNG. Protože se při odpojování disku předpokládá uzavření všech souborů, a na důslednost

aplikačních programů třetích stran nelze spoléhat, jsou navíc struktury OpenFileType zřetězeny v obousměrném spojovém seznamu.

Při otevíráni souboru lze ze strany VFS specifikovat bitovou maskou několik režimů (vfsModeExclusive, vfsModeWrite, vfsModeCreate, ...). U SMB je stejná logika vyjádřena komplexněji, třemi parametry. Překlad hodnot je zajištěn ošetřením sedmi různých smysluplných kombinací bitové masky.

Tato funkce je využívána i k otevírání adresářů, které však nemá jiný účel, než umožnit pozdější čtení jeho obsahu, k čemuž ale v SMB už nemusí být otevřen. Proto, pokud se jedná o adresář, je ihned po otevření zase uzavřen. Tím je ověřena jeho existence a zároveň je uloženo jeho jméno, které je pro SMB parametrem při čtení jeho obsahu.

Čtení ze souboru (FSFileRead)

Zde je rozdíl v tom, že VFS předpokládá existenci implicitního ukazatele aktuální polohy v souboru, který se automaticky posouvá s provedeným čtením. Offset, od kterého má čtení probíhat, tedy není parametrem, avšak SMB ho vyžaduje. Proto je jeho hodnota ukládána a aktualizována ve struktuře OpenFileType. Navíc čtení probíhá skrz souborovou vyrovnávací paměť (cache), viz příslušná kapitola.

Zápis do souboru (FSFileWrite)

Situace je obdobná jako u čtení. Jen cache se nevyužije, naopak, vzhledem k tomu, že nyní její obsah může být neplatný, pro tento soubor se její obsah vymaže. Byla by sice možná její komplikovaná aktualizace, ale přístupy se střídavým zápisem a čtením zřejmě nejsou příliš časté.

Výmaz souboru nebo adresáře (FSFileDelete)

Sémantika se liší v tom, že protokol SMB rozlišuje operaci vymazání souboru od vymazání adresáře. VFS ovšem ne. Vzhledem k tomu, že operace je volána na neotevřený (a tedy neznámý) soubor, je třeba ho nejprve otevřít, zjistit jeho typ, a pak zavolat příslušnou funkci. To bohužel zvýší počet odeslaných SMB zpráv při této operaci na tři.

Přejmenování souboru nebo adresáře (FSFileRename)

Zde je jen drobná odchylka. SMB vyžaduje plnou cestu u obou parametrů (starého i nového jména), VFS jen u starého. Doplnění nepředstavuje problém.

Práce s ukazatelem a koncem souboru (FSFileSeek, FSFileEOF, FSFileTell, FSFileSize)

Vzhledem k tomu, že ukazatel i informace o velikosti souboru je udržována na straně klienta, tyto operace vůbec interakci se serverem nepotřebují. Implementace se skládá jen ze čtení či změny struktury náležící otevřenému souboru.

Čtení atributů souborů (FSFileGetAttributes, FSFileGetDate)

Ani tyto funkce nepotřebují interakci se serverem. Protokol SMB při otevírání souboru vrátí mimo jiné i informace o něm, včetně atributů a data. Ovladač je uloží ve struktuře OpenFileType a nyní je jen vrátí. Co se týká různých druhů atributů souborů,

jak SMB, tak VFS vychází ze souborového systému FAT. Proto jsou i atributy shodné (vfsFileAttrDirectory, vfsFileAttrReadOnly, vfsFileAttrHidden, vfsFileAttrSystem).

Ve formátu pro uložení data souboru jsou větší rozdíly. PalmOS ukládá datum jako počet sekund od 1.1.1904. Protokol SMB jako počet 100ns jednotek uplynulých od 1.1.1601, což je velmi vysoké číslo, uložené na 64-bitech. Přepočet navíc nelze provést jen bitovými posuny, je třeba plná 64-bitová aritmetika, kterou použitý překladač pro PalmOS nepodporoval. Upravil jsem proto k tomuto účelu pro PalmOS knihovnu Math64 [9]. SMB také definuje u souboru o jeden více časových údajů než VFS. Proto jsem dvojici LastWriteTime a ChangeTime u SMB pokládal za odpovídající jednomu údaji vfsFileDateModified u VFS.

Při nastavování hodnot času a atributů se navíc projeví rozdíl, že protokol SMB má pro tyto účely funkci jen jednu společnou. Obsluha funkce VFS tak má málo informací k naplnění bloku parametrů a je třeba přečíst část stávajících údajů. Na výkonu však tento fakt neubere, jelikož soubor je třeba otevřít v každém případě, u SMB operace pracuje nad otevřeným souborem, u VFS nad zavřeným.

Připojení a odpojení jednotky (FSVolumeMount, FSVolumeUnmount)

Tyto funkce, jako jediné, nevolají aplikace, ale vlastní ovládací panel.

Při připojování se provede kompletní navázání síťového spojení se serverem, jak je popsáno v kapitolách „Negotiate Protocol“ a „Session Setup“ dokumentace [1] (více v kapitole o autentifikaci). Údaje potřebné k přihlášení přečte ovladač z databáze uložených předdefinovaných spojení. Index do této databáze se předává jako parametr poseMountParam->poseSlotNum.

Používá se tak zde pro způsob připojení typ jednotky definovaný konstantou vfsMountClass_POSE. Je v systému definována právě kvůli ovladači HostFS, který je určen pro softwarový emulátor PalmOS (Palm OS Emulator) a v běžném HW zařízení nefunguje. Tuto hodnota parametru mount však lze použít i v běžném fyzickém zařízení, jak bylo otestováno na Palm IIIx. Jedná se tak patrně o jedinou možnost, jak se v PalmOS obejít bez nutnosti použít k implementaci vlastní FS knihovny Slot Manager a implementovat tak Slot Driver.

Co se týká odpojení jednotky, zde se kromě odhlášení uživatele od serveru operací LOGOFF_ANDX provede uvolnění paměti všech souborů, eventuálně ponechaných otevřených. Na serveru se však explicitně nezavírají, neboť to zajistí samotný server.

Informace o jednotce (FSVolumeInfo)

Tato funkce vrací konstantní údaje o souborovém systému, a to mediaType=expMediaType_Any a fsCreator='SMBf'. Údaj slotRefNum je nulový, protože tento souborový systém Slot Manager nevyužívá.

Velikost volného a obsazeného prostoru na jednotce (FSVolumeSize)

Zde se odpovídající funkce SMB liší výrazně. Vrací několik hodnot (totalUnits, availUnits, sectorsPerUnit, bytesPerSector), které je potřeba mezi sebou vynásobit. Dále vrací velikost dostupného prostoru místo velikost volného. Údaje ovšem nelze jednoduše odečíst, neboť jsou 64-bitové a vzniklé součinem. Toto je tedy druhé místo, kde je uplatněna již zmíněná knihovna Math64 [9]. Konečně je třeba, aby číslo na

výstupu funkce bylo 32-bitové, což je vyřešeno tak, že velikost nad 4 GB je udávána jako 4 GB bez závažnějších dopadů.

Zjištění názvu jednotky (FSVolumeGetLabel)

Přestože SMB umožňuje tuto informaci zjistit, rozhodl jsem se umožnit tuto hodnotu nastavit uživatelsky v ovládacím panelu. Název disku (label) se totiž zobrazuje ve většině aplikací, které s ním pracují. Uživatel má tak možnost tento název ovlivnit, což přispěje k jeho lepší orientaci mezi jednotkami, zvlášť má-li jich připojeno více.

Čtení obsahu adresáře (FSDirEntryEnumerate)

Při této operaci jsou odlišnosti velmi podstatné a přinášejí mnoho komplikací. VFS rozhraní vyžaduje čtení po jednotlivých položkách adresáře. Ačkoliv je toto v protokolu SMB také možné (zde lze počet čtených položek zvolit), bylo by to vzhledem ke zpoždění v síti velmi neefektivní.

Pro každý otevřený adresář, ze kterého aplikace právě čtou, je tak třeba udržovat v paměti vyrovnávací paměť (buffer) s ještě nepřečtenými položkami. Počet potřebných bufferů je navíc předem neodhadnutelný. Některé aplikace (PicturePlus), prohledávají adresářový strom rekurzivně a špičkový počet otevřených adresářů odpovídá hloubce adresářového stromu. Proto byla pro buffery zvolena Feature Memory (viz dříve), která neplýtvá dynamickou pamětí.

Dále bylo nutné zvolit vhodný počet položek, který se bude po síti přenášet protokolem SMB najednou. Položky adresáře jsou v odpovědi serveru vráceny jako záznamy s proměnlivou délkou. Maximální délka názvu souboru (podstatná část adresářové položky) je ale 256 znaků, ačkoliv ve většině případů této délky zdaleka nedosahuje. To souvisí s vysoké požadavky na buffer adresáře (velký 8192 B). Počet položek čtených najednou byl proto zvolen 29. Při typické délce jména souboru kolem 12 znaků (a adresářové položky 36) tak obsah adresáře přenáší v cca 1 kB blocích, což je typická velikost paketu v lokální síti.

Sémantický rozdíl je v indikaci konce obsahu adresáře. Rozhraní VFS po nás vyžaduje vrátit informaci o konci adresáře už při přečtení poslední položky (navrácením hodnoty vfsIteratorStop). Tuto informaci však pomocí SMB nelze získat, je-li počet položek v adresáři násobkem počtu položek čtených najednou (tedy 29). V takovém případě vrátí další volání nulový počet položek bez předchozího varování. Počet položek adresáře přitom nelze dopředu zjistit. Kromě toho, že ovladač uchovává blok položek adresáře v paměti, musí tak být (kvůli této eventualitě) i minimálně o jednu položku v hledání napřed, kterou musí rovněž v paměti udržovat.

Při čtení obsahu adresáře opakovaným voláním funkce FSDirEntryEnumerate je jedním parametrem enumerátor, index právě čtené položky. Ačkoliv to lze očekávat, dokumentace k VFS [10] se explicitně nezmiňuje o tom, že by se během čtení toto číslo muselo inkrementovat o jedničku a začínat na nule. Nicméně komentovaný příklad v dokumentaci doporučuje toto použití a všechny mnou zkoumané aplikace se tím řídí. Proto tento postup předpokládá i ovladač a při nečekané hodnotě předaného enumerátoru vrací systémový chybový kód sysErrParamErr. Jiné řešení by implementaci funkce silně komplikovalo, neboť sekvenční čtení obsahu adresáře protokol SMB vyžaduje.

Nepodporované funkce (FSVolumeFormat, FSFileResize, FSVolumeSetLabel)

Funkce FSVolumeFormat, FSFileResize nebyly implementovány. Pro formátování ani nastavování názvu disku neexistuje ekvivalentní operace SMB. Formátování ostatně u virtuálního síťového disku postrádá smysl. Pro změnu velikosti souboru (kromě zkrácení na nulovou) také ne. Navíc jsem se u testovaných aplikací s tímto voláním vůbec nesetkal. Obě vrací chybový kód expErrUnimplemented.

Chybové kódy

Při chybových situacích je třeba přeložit i hodnoty indikující druh nastalé chyby. U protokolu SMB existují dva druhy chybových kódů, nazývané ERRDOS a NTERR. Servery v současné době používající verzi protokolu „NT LM 0.12“ vrací chybové kódy NTERR. Problém však je, že tyto nejsou v dokumentaci [1] vůbec popsány. Zdrojem k jejich identifikaci byl hlavičkový soubor „nterr.h“ ze zdrojového textu projektu Samba [5], vzniklý zřejmě postupným reverse engineeringem.

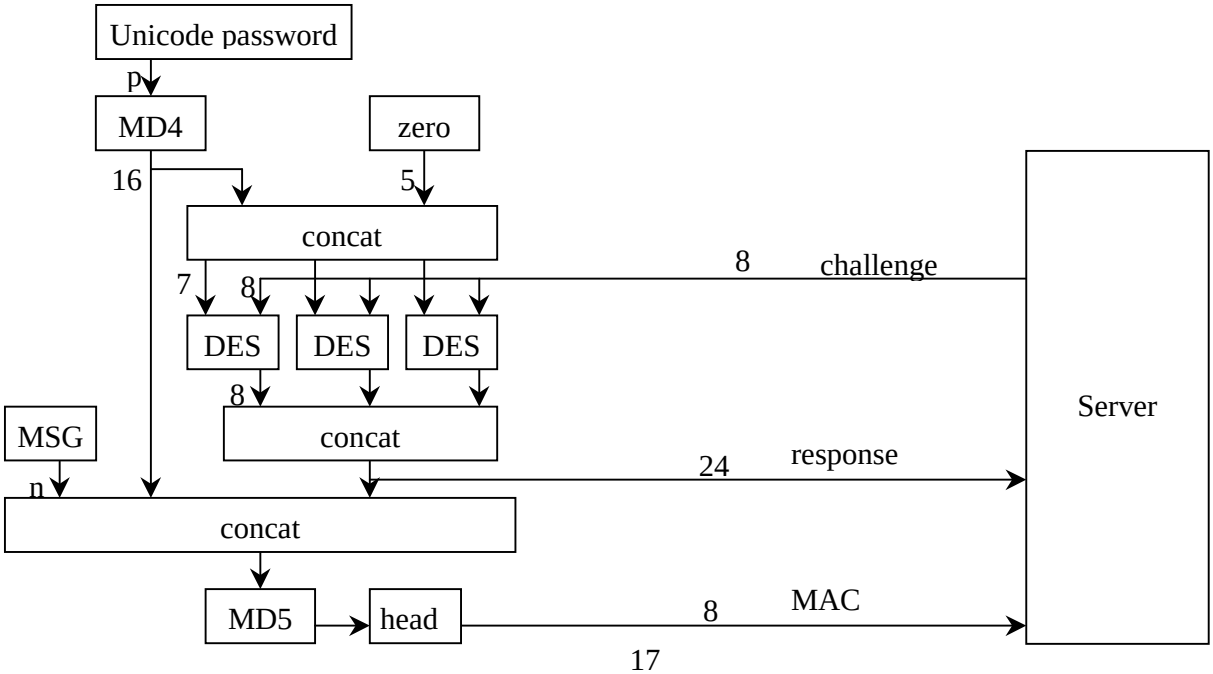
Na straně VFS je dokumentace podrobnější, různých chybových kódů je však poměrně málo (20 proti 500 u SMB), a tak pro řadu neexistuje ekvivalent. Program to řeší tak, že překládá (viz funkce HandleError) 10 nejběžnějších, u ostatních vrací kód vfsErrFileGeneric, k tomu určený.

3.4 Autentifikace a zabezpečení

Zabezpečení v protokolu SMB lze rozdělit na autentifikaci klienta a autentifikaci přenášených dat. Co tento protokol naopak nepodporuje, je autentifikace serveru a šifrování přenášených dat.

V SMB protokolu existují dvě metody zabezpečení. Tzv. „Share Level Security“, nezná uživatelská jména, heslo je přiřazeno každému sdílenému disku. Není příliš flexibilní, je považována za zastaralou a tento ovladač ji nepodporuje. Místo toho podporuje „User Level Security“, kde se klient prokazuje dvojicí jméno + heslo. O oprávnění ke konkrétnímu disku rozhoduje server sám.

Dále existují dva způsoby autentifikace. „Plain text“ spočívá v jednoduchém zaslání hesla jako otevřeného textu po síti, což je velký bezpečnostní risk. Implementace ovladače ho ve výchozím nastavení použije, jen pokud server nepodporuje lepší zabezpečenou autentifikaci, a to tzv. „challenge/response“ protokol (viz schéma).



- MSG – zpráva k podepsání
- MD4, MD5 – funkce kontrolních kódů [12], [13]
- DES – blok DES [11] se 7bit klíčem a 8bit daty
- concat – zřetězení a případné rozdělení řetězce
- head – začátek řetězce
- zero – generátor nul
- čísla/písmena u datových toků – bitové délky

Toto dokonalejší autentifikační schéma používá algoritmy DES, MD4 a MD5. Jejich implementaci z volně dostupných zdrojů [11] [12] [13], jsem mírně upravil, aby z nich mohla být vytvořena PalmOS knihovna „DES Library“.

Klient při tomto schématu prokáže serveru znalost sdílené tajné informace bez jejího zasílání po síti, tím, že odpoví na výzvu serveru (challenge) správnou odpovědí (response). Výstupem tohoto schématu je i klíč, který se během další komunikace použije k podepisování zpráv kontrolním kódem vygenerovaným algoritmem MD5. Protokol tak dokáže rozpoznat podvrženou zprávu, nezajistí ji ale proti odposlechu.

Pokud se týká podpory zabezpečené autentifikace u serverů, během přihlašování ji podporuje v současné verzi Samba i současné verze Windows (NT 4.0,2000,XP). Podepisování zpráv Samba (v2.2.3a) nepodporuje, Windows NT 4.0 ne, Windows (2000,XP) už ano. O tom, zda se autentifikace použije rozhodují jednak schopnosti serveru, které se lze zjistit během přihlašování, a dále uživatelské nastavení. S výchozím nastavením ovladač zabezpečenou autentifikaci použije, pokud je dostupná. Uživatel ji má navíc v ovládacím panelu možnost potlačit úplně, nebo naopak vyžadovat vždy.

3.5 Datové struktury

Diskové jednotky a seznam otevřených souborů

Základní datovou strukturou je blok globálních proměnných. Knihovna pod PalmOS globální data ve smyslu terminologie jazyka C++ mít nemůže. Místo toho si musí zaregistrovat v systémové tabulce knihoven blok paměti funkcí SysLibTblEntry. Ten si pak stejnou funkcí nechá vrátit při každém volání některé své vlastní funkce. Je to tak jediná možnost, jak pracovat s persistentními daty během činnosti ovladače.

Zde vypadá tento blok paměti (GlobalsType) takto:

```
struct GlobalsType {
  Int32 openCount;        // TIMES IS THE LIBRARY IS OPEN
  CInet inet;             // SHARED INET LIBRARY
  UInt16 DESLibRefNum;    // REFERENCE TO DES LIBRARY
  CSMB::CSMBBuffer buffer;// BUFFER SHARED AMONG SMB LIBS
  MountedVolumeInfoType volumes[MAX_VOLUMES];
  UInt16 lastFeatureNum;  // LAST FTR NUMBER ALLOCATED
  Cache cache;            // FILE CACHE OBJECT
};
```

Konstanta MAX_VOLUMES byla zvolena 4. Ovladač tedy může mít najednou připojené 4 síťové jednotky. Větší počet nemá smysl, protože standardní implementace TCP pod PalmOS podporuje maximálně 4 otevřené sockety. Každá jednotka je reprezentována následující strukturou:

```
struct MountedVolumeInfoType {
  OpenCommonPtr openFiles; // LINKED LIST OF OPEN FILES
  int openFilesCount;      // COUNT OF FILES OPEN
  UInt16 volRefNum; // OF VOLUME MOUNTED IN THIS SLOT
  int share;        // INDEX OF SHARE IN DataStore
  char label[16];   // VOLUME LABEL TO BE RETURNED
  CSMB smb;         // SMB LIBRARY SERVING THIS VOLUME
};
```

Podstatný je především vložený objekt knihovny pro SMB, volRefNum, což je číslo, přidělený operačním systémem při připojování jednotky. Z aplikačního hlediska je povinným parametrem každé volané funkce rozhraní VFS a odkaz na seznam otevřených souborů. Protože se v informacích, nutných uchovávat v paměti, liší otevřený soubor od adresáře, je tvořený variantními záznamy, obousměrně zřetězenými.

```
struct OpenCommonType {
  unsigned long magic;         // MAGIC NUMBER
  OpenCommonType *next, *prev; // LINKED LIST
  UInt16 volRefNum;            // A VOLUME
  CSMB::SMBFileInfo info;      // SIZE, TIMES, ATTRIB.
};

struct CSMB::SMBFileInfo {
  unsigned long size;
  unsigned long created, modified, accessed;
  unsigned char attributes;
};
```

Společné souboru i adresáři jsou informace o velikosti, atributech a datu, které je výhodné uschovat pro rychlejší vyřízení eventuálních dotazů aplikace. Dále identifikace jednotky, ke které soubor patří, při operacích se soubory je totiž parametrem předávaným ovladači jen identifikace souboru a jednotku je nutné v efektivně zjistit. Společná je rovněž speciální hodnota (magic number) indikující platnost celého záznamu. Po uzavření souboru je hodnota vynulována, je tak vyloučeno, aby aplikace využívající náš ovladač způsobila pád systému odkazováním na uzavřený soubor.

```
struct OpenFileType : public OpenCommonType {
  unsigned short fid;            // SMB FILE ID
  unsigned long offset;          // ACTUAL FILE POSITION
  Cache::HandleType cacheHandle; // FOR CCache OBJECT
};
```

U souboru uchováváme rukojeť (handle) pro protokol SMB a aktuální pozici v souboru, je to nutné kvůli neexistenci souborové pozice v SMB. Dále je tu odkaz do souborové vyrovnávací paměti.

```
struct OpenDirType : public OpenCommonType {
  unsigned long pos;       // DATA FOR ENUMERATING
  CSMB::SMBDirEntry entry;
  CSMB::SMBDirHandle handle;
  UInt16 featureNum;
  char name[];           //FILE NAME FOR DIR ENUMERATION
};
```

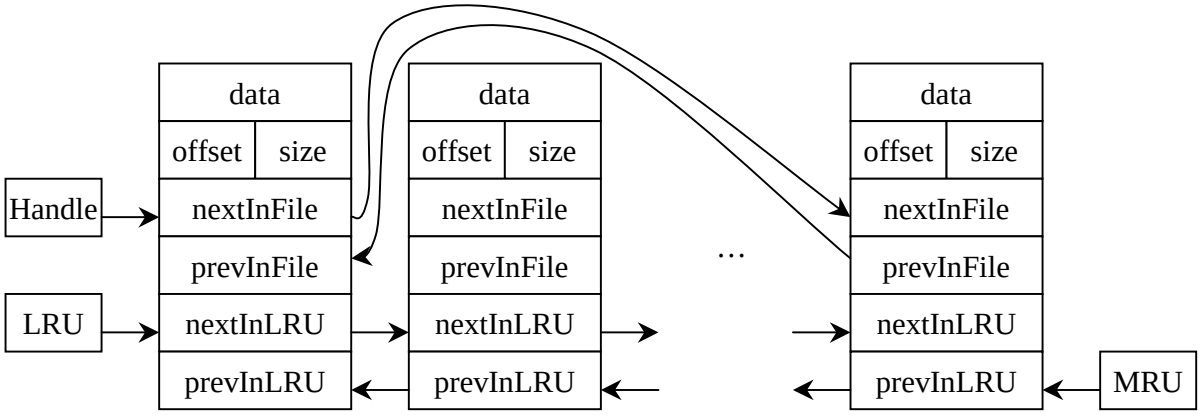
První položka je aktuální pozice enumerátoru při čtení obsahu adresáře. Je to hodnota, kterou očekáváme od aplikací jako parametr při příštím volání funkce FSDirEntryEnumerate. Další tři hodnoty složí pro ukládání adresářových položek, přečtených s předstihem, do Feature Memory. Nutné je také uchovávat jméno adresáře, aplikace se může rozhodnout vyvolat enumeraci adresáře znovu, a pak je jméno nutné jako parametr příslušné služby SMB.

Read-ahead cache

Pro efektivnější čtení souborů jsem navrhl jednoduchou vyrovnávací paměť (cache) s čtením v předstihu (read-ahead). Typický přístup k souboru je totiž čtení dat po malých blocích. Na diskové jednotce instalované aplikace jsou na ní sice uloženy jako soubory, ale svojí strukturu databáze si zachovávají. Jsou uloženy ve speciálním formátu [14], kdy se na začátku souboru vyskytuje kromě 32B hlavičky index záznamů databáze. Pak teprve následují vlastní data uložená v blocích relativně velkých. Zmíněný index se ale skládá z krátkých, 8-10B položek, které operační systém čte jednotlivě. Především načítání ikon programů aplikací Launcher a dále spouštění aplikací tímto jevem trpí. Projeví se vysoká doba odezvy sítě, se kterou tvůrci PalmOS u paměťové karty nepočítali.

Vyrovnávací paměť se to snaží řešit následovně. Je definována určitá velikost bloku cache (konkrétně 1024B). Pokud jsou čtená data větší než velikost bloku, do cache se vůbec neukládají, protože čtení těchto dat je efektivní a zbytečně bychom si cache zaplňovali daty, která se s velkou pravděpodobností už znovu číst nebudou (jak bylo řečeno, šetrnost k paměťovým zdrojům je priorita ovladače). Pokud jsou data menší než velikost bloku, přečte se s předstihem celý blok a uloží. V praxi se tak do paměti uloží prvních 1024B souboru najednou, což je většinou celý index. Pokud výskyt krátkých záznamů v databázi trvá i na jiném místě (příkladem jsou prvky uživatelské rozhraní, jež jsou rovněž v programové databázi uložené) dojde k tomuto efektu i později. Další požadavky na záznamy jsou tak obsluhovány ve značně kratším čase z obsahu cache.

Schematická zjednodušená struktura cache následuje:



Cache je organizována jako 32 záznamů. Vlastní data v nich uložená nejsou, na ty je zde jen odkaz. Jsou uložena v paměti pro databáze Storage Heap v zájmu šetření dynamickou pamětí. Každý záznam také nese informaci od délce dat v něm uložených a offsetu těchto dat od začátku souboru. Záznamy jsou zřetězené do dvou obousměrných spojových seznamů.

První seznam je několikanásobný, rozděluje bloky do řetězců, podle souborů, k nimž patří. Jeho začátek je v tzv. CacheHandle, struktuře, uložené ve struktuře příslušné k otevřenému souboru. Je použitý při čtení ze souboru, k hledání, zda není žádaná část souboru již v cache uložena.

Druhý seznam propojuje bloky v pořadí, ve kterém do nich bylo naposledy přistupováno. Na jeden konec ukazuje ukazatel LRU (Least Recently Used). Záznam na tomto konci je kandidátem na odstranění z cache a přepsání jinými daty. Na druhý konec ukazuje MRU (Most Recently Used), sem jsou vkládány záznamy, ke kterým bylo právě přistoupeno.

Ve struktuře tvořící cache jsou i další datové položky, které slouží ke statistice úspěšnosti (cache hit rate).

3.6 Rozšíření protokolu

Pokusil jsem se i o implementaci dvou funkcí, které jsou nad rámec klasického protokolu SMB, protože jde o funkce pro uživatele zajímavé. Nejde o funkce dokumentované v dokumentu, ze kterého jsem vycházel [1]. Jde o posílání zpráv (známých ze sítí Microsoft Network, jde o příkaz net send) a zjištění seznamu síťových disků nabízených serverem ke sdílení (shares). Jedná se o obalení protokolu DCE RPC do protokolu SMB.

V obou případech není volně k dispozici použitelná dokumentace. Vycházel jsem tedy ze zdrojových textů projektu Samba a z analýzy zachycené komunikace programů smbclient a SMBMate se serverem.

Posílání zpráv je z hlediska protokolu SMB další nedokumentovaný typ zprávy, avšak servery ho plně podporují. MS Windows zobrazí uživateli okno s textem zprávy. Projekt Samba dovoluje správci nastavit vlastní příkaz, který se při příjmu zprávy provede.

Žádost o vrácení seznamu dostupných disků je složitější. Jedná se o připojení virtuálnímu zařízení disku \IPC$ a otevření souboru \srvsvc, jako by šlo o běžný soubor a dále do operací podle Named Pipe Transaction Protokolu [1] zabalené nedokumentované funkce RPC. Výsledná implementace se zdá být na testovaných serverech funkční. Uživatel může seznam disků vyvolat v dialogu s parametry spojení zvolením tlačítka „Share“ a následně zvolit vybraný disk v zobrazeném dialogu.

3.7 Zpracování chyb

K přenosu zprávy o chybovém stavu uvnitř ovladače jsou použity výjimky. Protože byla část projektu (třída pro vlastní protokol SMB) koncipována jako platformově nezávislá, a systém PalmOS se v práci s výjimkami liší, definoval jsem v souboru except.h sadu abstraktních maker TRY, CATCH, ENDCATCH a THROW.

V systému PalmOS totiž výjimky jazyka C++ realizované kompilátorem nelze použít. Místo toho je třeba použít makra ze SDK a volání jádra systému. S tím souvisí fakt, že při průchodu výjimky hierarchií vnořených funkcí nedochází k uvolňování dynamické paměti voláním destruktorů. To však v našem případě není problém, protože užití dynamické paměti je omezeno na několik málo případů, ošetřených zvlášť.

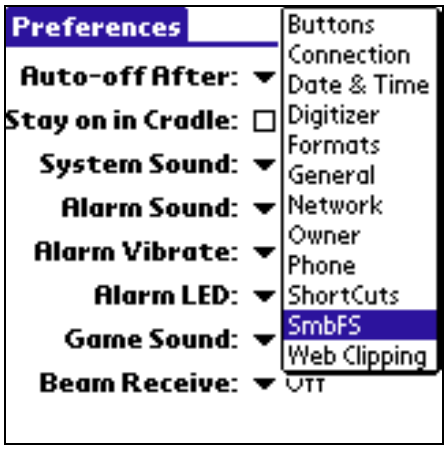
Kvůli tomuto omezení také nemůže být hodnotou výjimky nic jiného než 32-bitové číslo. Proto každé třídě přiděluji 16-bitový prostor chybových kódů. Ty statická členská funkce třídy umí přeložit na textové řetězce, zobrazitelné uživateli. Svůj prostor mají přiděleny i chybové kódy vrácené SMB serverem.

Výjimka je zachycena na nejvyšší úrovni, kterou jsou vstupní body FS knihovny v soubor u smbFS.cpp. Dále do aplikací cizích ji pochopitelně šířit nelze, je zavolána funkce HandleError, která známé SMB chybové kódy přeloží na VFS. U ostatních chyby popsaným způsobem získá textový popis, který zobrazí uživateli systémovým hlášením FrmCustomAlert.

3.8 Uživatelské rozhraní

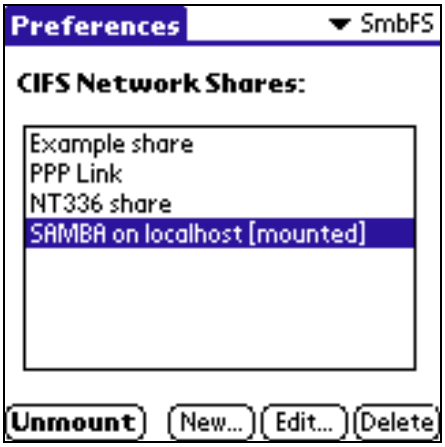
Instalace ovladače se provádí klasickým způsobem. Pomocí SW Palm Desktop pro MS Windows nebo unixového projektu pilot-link je třeba do zařízení nahrát tři databáze uložené v souborech smbFS.prc, smbFSui.prc a deslib.prc. Po instalaci je nutný restart operačního systému, aby systém knihovnu zaregistroval.

Ovládání a nastavení parametrů ovladače je soustředěno do ovládacího panelu, který je samostatnou aplikací. Je ale integrována do operačního systému tak, že se uživateli jeví jako další kategorie v aplikaci Preferences.



Základní dialog panelu umožňuje uživateli definovat si profily s předdefinovanými parametry pro připojení k serveru. Návrh dialogu vychází z obdobného panelu PalmOS, Connection. Heslo je zadáváno způsobem v systému obvyklým - speciálním dialogem, který umožňuje heslo přiřadit, ne však zobrazit nebo editovat.

Vzhledem k tomu, že aplikace nepoužívá jmennou službu NetBIOSu ke zjišťování názvů okolních počítačů, musí uživatel vyplnit všechny údaje nutné k navázání spojení - DNS jméno počítače nebo ip adresu, port, název domény, plný název disku včetně NetBIOSového jména serveru, své uživatelské jméno a heslo. Dále si může zvolit úroveň zabezpečení.



Přes roletové menu je přístup k informacím a statistikám připojení a k možnosti poslat zprávu na PC protokolem SMB. Je zde také možnost ovladač odinstalovat. To je velmi důležitá funkce, protože databázi s knihovnou nelze ze systému smazat standardními prostředky, protože je stále otevřena. Jedinou možností, jak za běhu systému otevřenou FS knihovnu odstranit, je funkce VFSRemoveFSLib VFS Manageru. Poté je možné smazat databázi samotnou. Aplikaci ovládacího panelu však uživatel musí odstranit ručně, na což je upozorněn.

3.9 SW nástroje pro vývoj

Nyní popíšu SW prostředky, které jsem k vývoji použil. Vývoj probíhal pod OS Linux Debian Woody 3.0. Použitý překladač byl 2.95.4 pro procesor M68 z balíku prc-tools 2.0.92 s PalmOS SDK 4.0. K editaci zdrojových textu pak editor CoolEdit 3.11.5 se zvýrazňováním integrovaný v souborovém manažeru Midnight Commander. Ke kompilaci prvků uživatelského rozhraní nástroj PilRC v2.8. Testování probíhalo kromě Linuxu i pod Windows 2000 a to v emulátoru PalmOS Emulator (pose) 3.5-1 (Linux) a Palm OS Emulatoru 3.5 (Windows).

Program lze přeložit s definovaným makrem DEBUG=1, které zapíná výpis ladících informací makrem LOG(). Implementace makra zapisuje tyto informace do textového souboru na disku hostitelského počítače, kde běží emulátor (funkce HostFOpen, ...). Dále je ve zdrojovém textu částí závislých na PalmOS použito standardních ladících maker ErrDisplay() a ErrDisplayFatalIf().

Tato lze při kompilaci distribuční verze odstranit nastavením makra ERROR_CHECK_LEVEL = ERROR_CHECK_FULL.

4 Výsledky

4.1 Rozsah práce

Celkový rozsah C++ kódu vytvořeného v rámci práce je cca 5170 řádků (bez přejatých knihoven pro DES, MD4 a MD5). Z toho tvoří 2220 platformově nezávislá knihovna pro klientskou stranu SMB protokolu a 920 řádků uživatelské rozhraní - panel. Následují velikosti přeložených programů (programových databází), vhodných k distribuci.

Verze s ladícími výpisy:

24714B deslib.prc
33718B smbFS.prc
44927B smbFSui.prc

Verze bez ladících výpisů:

24714B deslib.prc
23970B smbFS.prc
42603B smbFSui.prc

4.2 Testování

Výkon

Již zmíněným programem VFSMark jsem nechal změřit „indexy výkonnosti“ proti Palm m100 a 16 MB SD kartě (= 100 bodů), a to při různých typech síťových spojení k serveru. Vše na konfiguraci AMD Athlon XP 1800+, 300 Mhz DDR RAM, OS Linux 2.4.23, s emulátorem Palm m515 a PalmOS 4.1.

- spojení z emulátoru na stejný počítač (Local)
- spojení z emulátoru na jiný počítač v 10mb/s lokální síti (Ethernet)
- spojení z emulátoru přes tři bezdrátové spoje a Internet na server nt336.felk.cvut.cz (Wifi)
- spojení ze skutečného zařízení (Palm IIIx) po PPP sériovém spojení 57600 b/s (Serial)

Předmětem dalšího měřením byla záloha databází z RAM a její obnovení programem VFSBackup. Jednalo se o 25 databází o celkové velikosti 520kB. Z naměřeného času jsem vypočítal průměrnou přenosovou rychlost. Pro srovnání je v tabulce i maximální rychlost připojení. Tu jsem změřil programem wget resp. LGet pod PalmOS. Mělo by se tedy zhruba jednat o maximum dosažitelné protokolem TCP.

Následuje tabulka naměřených hodnot.

	Local	Ethernet	Wifi	Serial
Ping RRT [ms]	0.0	0.5	75.9	49.8
Maximální rychlost [kb/s]		5200	475	42
Test VFSMark				
File Create	392	392	390	12
File Delete	150	150	149	32
File Write	490	490	491	5
File Read	143	143	91	0
File Seek	231	236	231	30
DB Export	139	138	138	1
DB Import	112	111	98	3
Record Access	210	211	166	2
Resource Access	213	214	170	2
VFSMark	231	231	213	9
Test VFSBackup				
Rychlost zálohy [kb/s]	832.0	594.3	208.0	28.7
Využití linky		11.4 %	43.8 %	68.3 %
Rychlost obnovení [kb/s]	462.2	416.0	160.0	27.7
Využití linky		8.0 %	33.7 %	66.0 %
Test SMBClient				
Rychlost zálohy [kb/s]	1386.7	832.0	231.1	
Využití linky		16.0 %	48.7 %	
Rychlost obnovení [kb/s]	1254.2	594.3	189.1	
Využití linky		11.43	39.81	

Na výsledcích naměřených programem VFSMark je vidět, že více než na délce odezvy síťového spojení záleží na jeho rychlosti. To lze očekávat, jelikož během testů přenese zhruba 3,5 MB dat v každém směru. Naopak zvláštní vypadá rozdíl mezi operacemi čtení a zápisu. Čtení vypadá pomalejší, přestože ho VFSMark provádí záměrně v části testu po malých blocích, a měl by se projevit vliv cache. Index je však porovnáváním s paměťovou flash kartou s pomalým zápisem, což výsledek vysvětluje.

Ve druhém testu šlo především o zjištění, zda dokáže implementace protokolu využít dobře kapacitu spoje. Pro srovnání jsem na stejných linkách (pokud to bylo možné) provedl co nejpodobnější test programem smbclient z projektu Samba. V obou případech, jak je vidět, s rostoucí kapacitou klesá. Z výsledků je rovněž vidět, že projekt Samba vykazuje o 10% až 40% lepší výsledek. Je však možné, že je tento výsledek, způsoben faktem, že systém PalmOS souborový formát představující databázi ukládá po jednotlivých záznamech. Oproti tomu smbclient i při přenosu stejných dat data přenáší po větších blocích.

Kompatibilita

Ovladač byl během vývoje testován na těchto SMB serverech:

- Integrovaný server ve Windows NT 4.0, build 1381, Service Pack 6
- Integrovaný server ve Windows 2000, build 2195, Service Pack 3
- Samba 2.2.3a na Linuxu Debian 3.0 Woody, jádro verze 2.4.23

Mohu konstatovat, že se z našeho hlediska protokol příliš neliší. Především proto, že verzi i typy zpráv kterými se bude komunikovat, vybírá klient. Snažil jsem se používat vždy ty zprávy, které byly v dokumentaci [1] označeny jako doporučení pro poslední verzi protokolu „NT LM 0.12“. Drobné odchylky mezi servery jsem zjistil jen v navracených chybových kódech.

Uspokojivou kompatibilitu s VFS využívajícími aplikacemi se podařilo zajistit. Ovladač byl testován na aplikacích FileProg, Filez, JPGview, PicturePlus, TealMover, VFSBackup, VFSdos, VFSMark. U některých PalmOS Emulator hlásil nekritické ladící chybová hlášení za běhu a neuvolnění bloků paměti (memory leaks), ale jak jsem se přesvědčil na ovladači HostFS, chyba je pravděpodobně na straně autorů těchto aplikací.

Efektivita cache

Efektivitu vyrovnávací paměti jsem orientačně určil následujícím sledem operací.

- načtení ikon 10 aplikací Launcherem.
- spuštění 78 kB aplikace.
- prohlédnutí pěti cca 10 kB fotografií programem JPEGView
- zálohování a obnovení obsahu RAM (výchozích databází po resetu)

Výsledkem testu byla míra úspěšnosti (Cache Hit Ratio) 58%. To znamená, že více než polovina čtecích operací nemusela být provedena. Oproti stavu s vypnutou vyrovnávací pamětí se však přeneslo o 66 kB dat (z celkových 262 kB) více. To je dáno principem činnosti read-ahead cache. To však přesto operace nezpomalí, protože přenos 50B a 1024B paketu, započítáme-li dobu potřebnou na reakci serveru, trvá řádově stejně dlouhou dobu. Může to však vadit při mobilním síťovém spojení, kde je zpoplatněna přenesená data (GPRS). V takovém případě by bylo vhodné zmenšit velikost bloku cache.

4.3 Možnosti rozšíření

Téma nabízí řadu námětů k dalšímu obohacení funkčnosti.

Nyní je pro připojení síťového disku třeba vyplnit explicitně údaje potřebné k navázání spojení - jméno serveru, uživatelské jméno. Protokol NetBIOS ale podporuje zjišťování jmen počítačů dostupných v LAN. Lze si tak představit uživatelský interface podobný zobrazení známému z MS Windows jako „Network Neighborhood“ nebo „Computers Near Me“.

Instalaci aplikace by bylo možné také rozšířit o tzv. Palm Desktop Conduit, programový modul, který by uživateli umožnil importovat do PDA profily odpovídající přímo síťovým diskům, které má připojené na PC.

Zdokonalit by bylo možné i vyrovnávací paměť. Rozšířit ji i o zápis (write-back cache) a uchování dat v paměti i po uzavření souboru, pro případ jeho znovuotevření.

Protokol SMB nedostatečně zabezpečený proti odposlechu dat. Pro by jistě stálo za pokus prozkoumat možnosti, jak na platformě PalmOS zajistit dodatečnými SW prostředky vyšší bezpečnost - pro spojení se severem přes nedůvěryhodnou síť, např. Internet.

Rezervy má také uživatelské rozhraní. Ovladač má řadu parametrů, které by zkušenější uživatel mohl pokládat za užitečné nastavovat - velikost cache, velikost přijímacích a odesílacích bufferů, délku timeoutů při síťovém spojení. Ovládací panel by bylo vhodné o tuto možnost rozšířit.

5 Závěr

Úkol zadaný jako diplomová práce se podařilo vyřešit. Seznámil jsem se podrobně s protokolem SMB a architekturou VFS rozhraní v PalmOS. Snažil jsem se o co nejefektivnější překonání rozdílů mezi oběma systémy.

Vzniklá knihovna byla úspěšně otestována jak na emulátoru, tak na skutečném zařízení s PalmOS. Je schopná se připojit minimálně ke třem současným serverovým implementacím.

Ukázalo se rovněž, že dané téma je široké, a že možnosti obohaceni projektu existují, ať už směrem k vyšší uživatelské přívětivosti, nebo ke zvýšení výkonu.

Ačkoliv se při testování na skutečném zařízení potvrdilo, že přístup k datům po síti nebude rychlejší než k datům paměťové kartě, mám za to, že pro uživatele znamená přínos a novou možnost k využití PDA.

6 Reference

[1] Common Internet File System (CIFS) Technical Reference rev. 1.0
http://www.snia.org/tech_activities/CIFS/CIFS-TR-1p00_FINAL.pdf

[2] Christopher R. Hertel:Implementing CIFS
http://ubiqx.org/cifs/

[3] Microsoft Networks/OpenNET File Sharing Protocol, Intel, Microsoft, 1988
http://s2.samba.org/samba/ftp/specs/smb-core.ps

[4] More Than 40 Companies to Evaluate Proposed Standard For Internet Networking
http://www.microsoft.com/presspass/press/1996/aug96/Cfwkspr.asp

[5] Projekt Samba
http://www.samba.org/

[6] JCIFS
http://jcifs.samba.org/

[7] Gartner Dataquest (August 2003)
http://www3.gartner.com/5_about/press_releases/pr15aug2003b.jsp

[8] Zdrojový kód HostFS.c
http://www.palmsource.com/developers/

[9] Knihovna MATH64
http://www.funet.fi/pub/crypt/mirrors/rpub.cl.msu.edu/bignum/math64.shar.gz

[10] Palm OS Programmer's API Reference, Palm Inc., 2001
http://www.palmos.com/dev/support/docs/palmos/

[11] DES Library, Eric Young, 1997
ftp://ftp.psy.uq.oz.au/pub/Crypto/DES/

[12] RFC 1320 - The MD4 Message-Digest Algorithm
http://www.faqs.org/rfcs/rfc1320.html

[13] RFC 1321 - The MD5 Message-Digest Algorithm
http://www.faqs.org/rfcs/rfc1321.html

[14] Palm File Format Specification, Palm, Inc., 2001
http://www.palmos.com/dev/support/docs/palmos/

[15] Palm OS Programmer’s Companion, Palm, Inc., 2001
http://www.palmos.com/dev/support/docs/palmos/

7 Přílohy

Přílohou diplomové práce je 1 CD s textem diplomové práce, zdrojovými texty, zkompilovanou knihovnou, HTML prezentací a vybranými dokumenty z referencí.

Adresářový strom kompletního projektu:

```
├──smbfsui
│       makefile
│       smbFSui.rcp
│       smbFSui.cpp
│       smbFSui.def
│       resource.h
│       secr0001.h
│       secr0001.cpp
├──deslib
│   │   makefile
│   │   deslib.h
│   │   deslib.def
│   │   deslib.c
│   ├──md4
│   │       global.h
│   │       md4.c
│   │       md4.h
│   ├──des
│   │       des.h
│   │       des_enc.c
│   │       des_locl.h
│   │       ecb_enc.c
│   │       podd.h
│   │       set_key.c
│   │       sk.h
│   │       spr.h
│   └──md5
│           md5.h
│           global.h
│           md5c.c
└──smbfs
    │   makefile
    │   smbFS.h
    │   smbFS.cpp
    │   smbFS.def
    ├──deslib
    │       global.h
    │       des.h
    │       deslib.h
    │       md4.h
    └──libs
            cache.h
            cache.cpp
            except.h
            inet.h
            inet.cpp
            replace.cpp
            smb.h
            smb.cpp
            uint64.h
            uint64.c
```