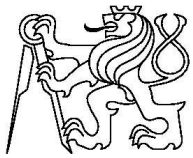
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE



Fakulta stavební

Katedra systémového inženýrství

Diplomová práce

Použití adresářových služeb
v informačních systémech

Autor práce: **Karel Benák**
Vedoucí práce: **doc. RNDr. Jiří Demel, CSc.**
Školní rok: **2003/2004**

květen 2004

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně, pouze za odborného vedení vedoucího diplomové práce doc. RNDr. Jiřího Demela, CSc.

Dále prohlašují, že veškeré podklady, ze kterých jsem čerpal, jsou uvedeny v seznamu literatury.

V Praze dne 7. června 2004 Karel Benák

Poděkování

Touto cestou bych rád poděkoval vedoucímu diplomové práce, panu doc. RNDr. Jiřímu Demelovi, CSc., za konzultace a rady, jak psát diplomovou práci pomocí sázecího systému LaTeX. Dále bych chtěl poděkovat svým kolegům z Klubu Sinkuleho koleje za možnost vyzkoušení a realizace některých mých nápadů a návrhů v ostrém provozu. Velké poděkování patří Ing. Martinu Červenému nejen za zapůjčení cenné pokladnice informací, [UDLDAPDS], ale především za jeho připomínky a rady k realizaci nového informačního systému.

Karel Benák

ABSTRAKT

V diplomové práci se zabývám možností využití adresářových služeb v informačních systémech a jejich možného nasazení jako důležité součásti informačního systému. Práce seznamuje s protokolem LDAP, jeho použitím a návrhem základních datových typů, adresářových struktur a základních metod zabezpečení proti zneužití. Na reálném příkladu je předvedena realizace adresářové služby jako důležité služby počítačové sítě, která dynamicky poskytuje data službám, např. DNS, DHCP a dalším.

Klíčová slova: Adresářové služby, LDAP, objektové třídy, atributy

ABSTRACT

My diploma thesis is engaged in possibilities of usage of directory services in information systems and of their possible use as important part of information system. Thesis acquaints with LDAP protocol, with its use and elementary data types design, directory structures and basic security methods against misuse. Real example demonstrates practical realization of directory service as important service of computer network, which dynamically provides data to other services (e.g. DNS, DHCP, etc.)

Key words: Directory Services, LDAP, ObjectClass, Attributes

Obsah

Seznam tabulek

Seznam obrázků

Kapitola 1

Úvod

Snahou této práce je přiblížit možnosti adresářových služeb, především standardizovaného protokolu LDAP, a jejich použití v informačních systémech, vytipovat vhodné oblasti použití adresářových služeb v informačních systémech a realizovat ukázkové řešení vybraných služeb.

Adresáře a adresářové služby provázejí náš život, aniž bychom si uvědomili jejich existenci a měli tušení o jejich významu. Jejich použití je natolik jednoduché a relativně přirozené, že jejich skutečný význam pro náš život není dostatečně oceněn. Adresářem je klasický telefonní seznam, v adresáři je organizovaný seznam kontaktů, vizitek a celá řada dalších užitečných a praktických informací.

Bohužel, jejich použití je během na dlouhou trať a návrháři informačních systémů musí provést velmi důkladnou analýzu, která je pro zákazníka nákladná. Dále je nutné brát v úvahu cenu licencí software, které se mohou velmi výrazně lišit pro různý počet záznamů, uživatelů, nebo pro různé technické prostředky. Pokud se ještě připočítá zákaznická podpora, možné problémy s migrací stávajících dat, případné výpadky při zavádění systému, náklady na správu dat, nároky kladené na správce informačního systému a přeškolení uživatelů na nový systém, vyšplhá se cena nového informačního systému na poměrně vysokou částku.

Proč se tedy zabývat adresářovými službami, prostředky umožňujícími jejich provoz a spolupráci s dalšími aplikacemi? Důvod je poměrně jednoduchý. Pokud si zákazník svých informací skutečně cení, dokáže v adresářových službách zorganizovat a poskytovat obrovské množství dat podle konkrétních požadavků i s ohledem na další vývoj nároků a požadavků. Adresářové služby jsou velmi silným nástrojem pro aplikace, které vyžadují rychlý, stabilní a bezpečný přístupu k datům.

Pro ukázky v textu práce jsou použity návrhy částí informačního systému společnosti Skanska CZ. Na této společnosti lze názorně vysvětlit a popsat návrh jmenných prostorů a topologie pro rozsáhlejší podnik. Bohužel, popis nebude zcela odpovídat skutečnosti především proto, že se jedná o velmi rozsáhlou společnost a její analýza by zabrala příliš mnoho času. Přesto je Skanska CZ ideálním příkladem většího podniku, který by mohl používat adresářové služby. Závěrečná kapitola je věnována analýze řešení části informačního systému Klubu Sinkuleho koleje, realizovaného v tomto roce.

Kapitola 2

Adresářové služby

2.1 Adresáře a adresářové služby

Adresář slouží k organizování a sdružování dat do skupin, aby se v nich mohl uživatel snáze orientovat. V podstatě jsou to jakési kontejnery pro ukládání dat, specializované databáze, ve kterých lze uchovat a především vyhledávat velké množství informací různého charakteru. Na rozdíl od relačních databází jsou určeny především pro vyhledávání, nikoliv pro klasické transakce, časté ukládání a změny dat anebo pro velmi komplikované dotazy.

V normálním životě se lze většinou setkat s *off-line* adresáři, které jsou nejčastěji k dispozici v tištěné verzi (telefonní seznam apod.). Při práci s počítači se používají *on-line* adresáře, které mají oproti zmíněným off-line adresářím následující výhody:

On-line adresáře jsou dynamické – v případě potřeby lze velmi snadno přidávat nové záznamy, mazat zastaralé záznamy a aktualizovat stávající záznamy. U off-line adresářů je jakákoliv aktualizace obtížná a většinou se provádí novým výtiskem. Příkladem výhody dynamických adresářů může být situace, kdy zaměstnanec z centrály v Praze jede na služební cestu do pobočky v Brně. Během jeho cesty mu lze přidělit příslušná přístupová práva pro objekt v Brněnské pobočce, aniž by muselo dojít k nové registraci zaměstnance. V případě off-line adresáře, kterým může být např. telefonní seznam, katalog zboží ad., je nutné provést jejich nový výtisk.

On-line adresáře jsou flexibilní – v adresáři lze ukládat data různých typů, např. *ASCII* text, text v kódování *UTF-8*, digitální certifikáty, obrázky, audio, odkazy, různé URL ad. V záznamech lze pak jednoduše a velmi rychle vyhledávat podle příslušných pravidel. Mimo to lze data poměrně pružně rozšiřovat a doplňovat o nové vlastnosti, nutné pro nové služby spolupracující s on-line adresářem.

On-line adresáře lze velmi dobře zabezpečit – přenosový protokol je standardizovaný a má i svou zabezpečenou verzi. Uživatelům lze pomocí *ACL* (*access control list*) přesně vymezit, ke kterým záznamům smějí a ke kterým nesmějí přistupovat.

On-line adresáře lze snadno personalizovat – díky adresářovým službám lze velmi snadno uchovat informace o nastavení aplikace nebo prostředí pro konkrétního uživatele systému. Rovněž lze v adresáři snadno uchovat informace o zálibách a uživatelem nejčastěji hledaných informacích a dokumentech a cíleně mu pak předkládat konkrétní reklamu a nabídky. Tento princip je ale postupně nahrazován principem kolaborace.

Adresáře lze také rozdělit podle způsobu, jakým spolupracují s ostatními aplikacemi a systémy:

Aplikační adresáře – adresáře, které si udržují aplikace pro ukládání svých vlastních záznamů. Příkladem může být např. IBM/Lotus Notes, Microsoft Exchange a např. aliasy obsažené v `/etc/aliases`. Ty slouží pro poštovní aliasy *MTA* serverů.

Adresáře síťových operačních systémů (NOS) – adresáře, které využívají různé operační systémy, jakými jsou např. Windows, Linux, Solaris, AIX apod. Tyto adresáře bývají postaveny na některém z uznávaných standardů a mohou kombinovat několik protokolů, jako je tomu např. u Microsoft Active Directory. Ten v sobě obsahuje protokoly *LDAP*, *Kerberos* a *CIFS*. Dalším přikladem je např. Novell eDirectory (dříve známý jako Netware Directory Services), který implementuje protokol *X.500*, nebo *Network Information Service* (*NIS*) od Sun Microsystems užívaný pro distribuci uživatelských účtů mezi UNIXovými počítači.

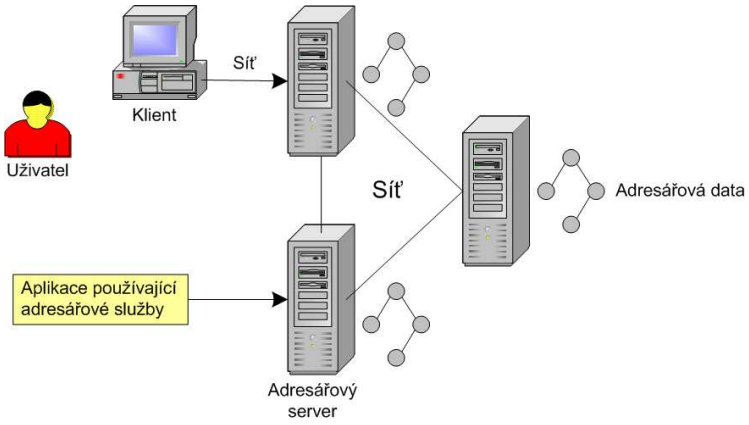
Účelové adresáře – velmi specializovaný typ adresářů určených pro jediný konkrétní účel. Nelze je příliš rozšiřovat o další typy informací, pouze při změně standardu. Typickým příkladem účelového adresáře je *Domain Name System* (*DNS*).

Univerzální standardizované adresáře – standardizované adresářové služby založené na protokolu *X.500* postaveného nad síťovým modelem *OSI* a protokol *LDAP*, který je zjednodušenou variantou protokolu *X.500* a je podstatnou částí této práce.

Velmi často se spojují termíny *adresář* a *adresářová služba* a jsou považovány za synonymum. Adresářové služby jsou souhrnem software, hardware, procesů, politik a administrativních procedur. Skládají se z následujících komponent:

- Informace uložené v adresáři
- Serverový software, který tyto informace poskytuje
- Klientský software, který je schopen uživateli tyto informace zprostředkovat
- Hardware, na kterém klienti a servery pracují
- Podporující software jako jsou operační systémy a ovladače zařízení
- Síťová infrastruktura, která je schopna spojit klienty se servery
- Bezpečnostní politika, která specifikuje oprávnění přistupů k záznamům, aktualizaci dat, apod.
- Procedury, které adresářové služby používají pro udržbu a monitorování
- Software používaný pro údržbu a monitorování adresářových služeb

Obrázek 2.1 zobrazuje typické součásti adresářových služeb, ovšem nezobrazuje kompletní seznam, pouze nejzákladnější součásti.



Obr. 2.1: Součásti adresářových služeb

2.2 Použití adresářových služeb

Adresářové služby mají velmi široké pole využití. Ačkoliv nedosahují takových možností jako relační databáze, přesto se jich ve velké míře užívá pro ukládání informací o uživatelích počítačových sítí různých organizací a společností.

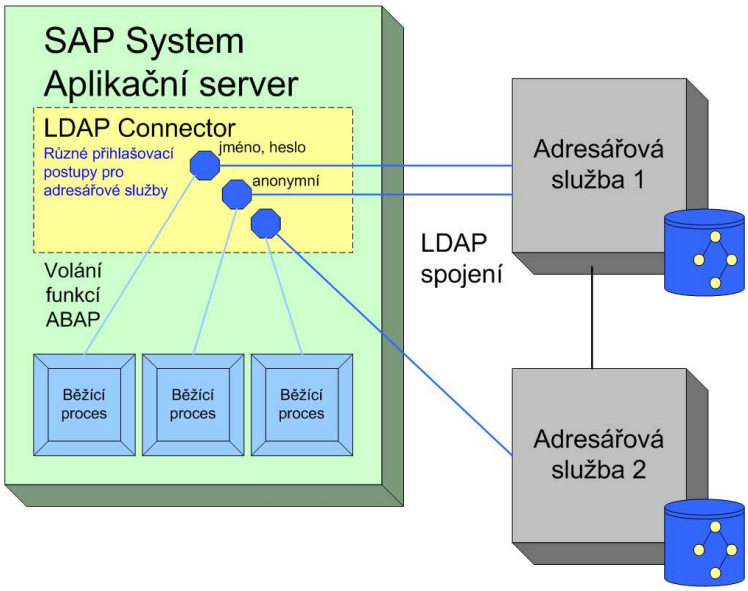
Adresářové servery slouží především jako zdroje pro ověření přístupových práv k daným prostředkům, např. počítačovému účtu a poštovnímu účtu elektronické pošty. Dále může sloužit jako adresář a telefonní seznam společnosti, ve kterém může být uvedeno velké množství informací počínaje jménem a příjmením, přes telefonní linku, mobilní telefon, účet pro instant messengera až po fotografii.

V adresářových službách jsou organizovány i veřejné klíče a digitální certifikáty uživatelů podle norem *PKI*. Model certifikátů používaných pro ověření uživatelské identity je převzatý z protokolu *X.500*, konkrétně *X.509v3* ([X.509] a [Dostalek], kapitola 9.).

Možnosti adresářových služeb jsou opravdu veliké a záleží pouze na návrhářích informačního systému, jak dokážou využít možnosti této technologie. Při správném návrhu je třeba vzít v úvahu nejen aktuální, ale i budoucí potřeby společnosti. Ve většině případů se s adresářovým serverem začíná jako s levným a efektivním zdrojem pro autentizaci uživatelů v počítačové síti společnosti. Po nějaké době přijde návrh na propojení tohoto seznamu uživatelů s vlastním seznamem zaměstnanců a možností jejich správy. Pracovníci zase ocení možnost propojení svého poštovního klienta s adresářovým serverem, ve kterém budou mít uloženy kontakty nejen na své spolupracovníky, ale zároveň i na své zákazníky a dodavatele. V tomto okamžiku se již jedná o provázání s CRM a SRM systémy.

Některé velké podnikové aplikace jsou schopny používat adresářové služby pro ověření uživatele k přístupu ke konkrétním záznamům, např. SAP (obr. 2.2, strana 8.). Produkt mySAP, postavený nad J2EE, je schopen pomocí jednoduchého přemapování svých obvyklých ABAP atributů získávat z adresářové služby daleko více informací, než je pouhá autentizace uživatele[1]. Velký konkurent SAPu, produkt společnosti PeopleSoft[2], je schopen díky svému zaměření především na CRM a HR systémy přímé spolupráce s adresářovými službami a využívá jich jako možného zdroje dat.

[1] `http://www.sap.com`
[2] `http://www.peoplesoft.com`



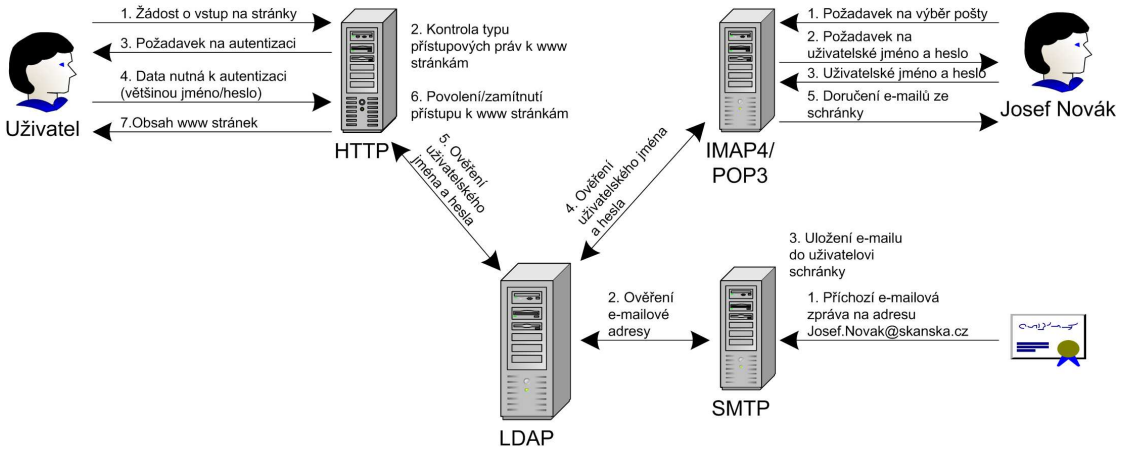
Obr. 2.2: Autentizace v SAPu pomoci adresářových služeb

Dalším příkladem využití adresářových služeb je jejich spojení s poštovním serverem, který dokáže přepisovat poštovní hlavičky s názvem účtu uživatele na jeho poštovní alias a zpět. Poštovní server přijme např. e-mail pro příjemce `josef.novak@skanska.cz` a při dotazu na systémové účty zjistí, že žádný takový účet na daném serveru neexistuje. Podívá se tedy do seznamu poštovních aliasů uloženého v adresářové službě a zjistí, že e-mail má uložit do poštovní schránky `jnovak`. Toto propojení poštovního serveru s adresářovým serverem má např. MS Active Directory s MS Exchange Serverem, který zajišťuje právě zmiňované poštovní služby. MS Exchange Server sice obsahuje vlastní verzi adresáře, ale propojení s již zmiňovaným Active Directory z něj činí často používané řešení pro středně velké a velké podniky. Podobně je na tom [JES], který pracuje i na jiných platformách, než-li jsou Microsoft Windows, a snaží se více dodržovat standardy definované v příslušných *RFC*.

Obrázek 2.3, strana 9., zobrazuje možnost praktického využití adresářového serveru jako možného autentizačního zdroje pro *IMAP/POP3* a *HTTP* server, ovšem jak bude popsáno na straně 70, nemusí se jeho možnosti omezovat pouze na úlohu autentizace. Může sloužit i jako databáze, ze které čerpá potřebné informace *DNS* a *DHCP* server.

Adresářové služby se snaží nahradit některé zastaralé standardy pro sdílení dat pomocí jmenných služeb. Příkladem je noční můra všech administrátorů a odborníků, standardy *NIS* a *NIS+*, vytvořené společností Sun Microsystems. Ve své době se jednalo o pokrokové řešení, bohužel se příliš nedbalo na jejich bezpečnost.

Pokud by z předchozího či následujícího textu mohlo vyplynout, že adresářové služby jsou všemocné, je třeba si uvědomit charakter a povahu dat, která se v nich ukládají a se kterými uživatel pracuje. Adresářové služby nejsou v žádném případě podobné klasickým relačním databázím, jak se je většina lidí snaží pochopit. Lze si je představit jako velmi jednoduchou kartotéku, ve které se ukládají jednotlivé lístky do jednotlivých kontejnerů. Při splnění určitých podmínek lze však v takovéto stromově uspořádané kartotéce velmi rychle vyhledávat.



Obr. 2.3: Centralizace ověření dat

Adresářové služby nejsou určeny k zachycení transakčních dat a neovládají práci s referenční integritou. K tomu se velmi dobře hodí klasické relační databáze, dnes již v drtivé většině využívající jazyka *SQL* (*Structured Query Language*). Např. pravidelné platby je možné v adresářových službách jistým způsobem zachytit, je to ale velmi nepraktické řešení. S těmito daty pak nelze rozumně manipulovat. Jazyk *SQL* má v tomto ohledu velmi široké možnosti (matematické operace, pohledy apod.). Adresářovým službám nelze klást tak sofistikované dotazy, jaké je možné pokládat pomocí *SQL*, a již vůbec nepřipadají v úvahu jakékoliv rozsáhlé transakce spojené např. s vkládáním více záznamů do databáze u kterých je nutné provádět kontrolu správných hodnot dat.

Adresářové služby nejsou určeny k přenosu souborů. K němu je primárně určen protokol *FTP* (*File Transfer Protocol*). Je snahou, aby zaznamenaná data měla co nejmenší velikost, zatímco pomocí FTP lze přenášet data (především soubory) v rozsahu od několika bytů až do několika gigabytů.

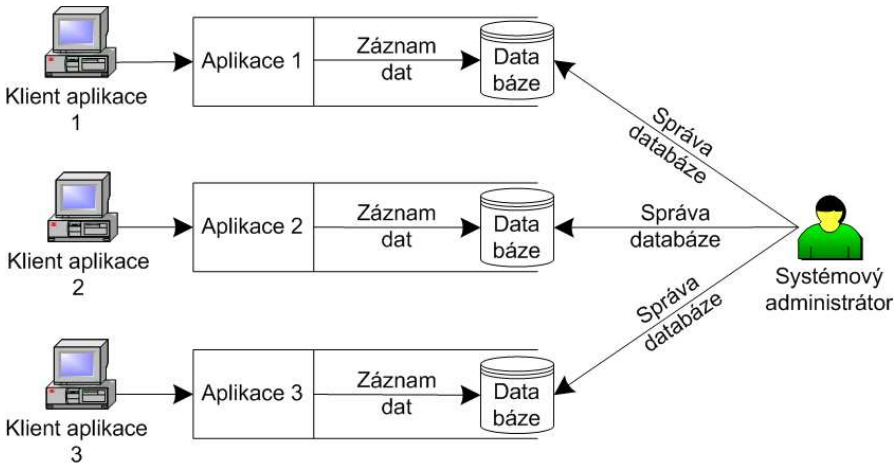
Adresářové služby nejsou určeny k prezentaci dat. K tomuto je určen protokol *HTTP* (*Hyper Text Transfer Protocol*) a služba *WWW* (*World Wide Web*). Podobně jako v případě *FTP* protokolu zde dochází k velkému přenosu dat a především jejich vizualizaci v uživatelově prohlížeči. Adresářová služba však může sloužit jako dobrý základ pro seznam odkazů na konkrétní www stránky.

Je třeba důsledně analyzovat potřeby organizace a využívat všechny služby ke konkrétnímu účelu, ke kterému byly navrženy. Adresářové služby jsou velmi výhodné pro již zmíněnou autentizaci, lze je ovšem využít i pro uložení jiných dat.

2.3 Data a jejich správa

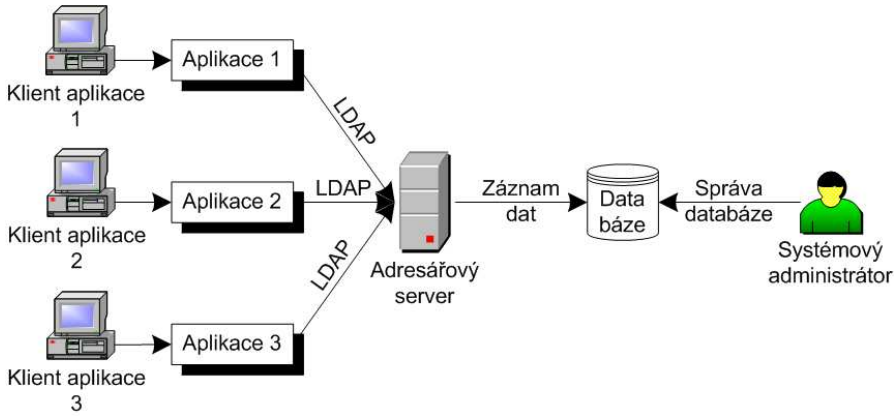
Klasické aplikace používají pro ukládání svých dat různé druhy databází. Ať už se jedná o jejich interní datové formáty souborů, různé embedded databáze založené na principu klič–hodnota, nebo dokonce velké *RDMBS* databáze typu klient–server, vždy je tu problém s duplicitou záznamů. Některé typy dat jsou duplicitně vedené v každé z databází a jejich výměna mezi jednotlivými aplikacemi je možná pouze na základě exportních a importních filtrů, které jsou schopny převést jeden typ dat na druhý. Administrátor je tak nucen dohlížet na několik

databází a musí zajistit jejich konzistenci, správnou údržbu, zálohování a případnou obnovu (obr. 2.4, strana 10.).



Obr. 2.4: Duplicita databází

Data uložená v adresářovém serveru lze organizovat a spravovat z jediného místa a aplikace, které jsou schopny používat adresářové služby jako svůj zdroj dat, je pak mohou velmi snadno sdílet (obr. 2.5, strana 10.). Rovněž je ale možné delegovat administrátorská práva na



Obr. 2.5: Centralizace databází

více pracovníků, takže jednotlivé části adresáře mohou spravovat různí lidé.

Bohužel, adresářové služby nelze použít pro ukládání všech typů dat. Zatímco v relačních databázích lze uložit téměř všechny druhy, v adresářových službách se jedná o data, která nejsou transakčního charakteru a pro jejichž uložení se adresář hodí. Adresářové služby totiž ve většině případů umožňují pouze jednoduché transakce a na rozdíl od relačních databází neobsahují kontrolu referenční integrity[3]. Tu musí zajistit samotná aplikace informačního systému.

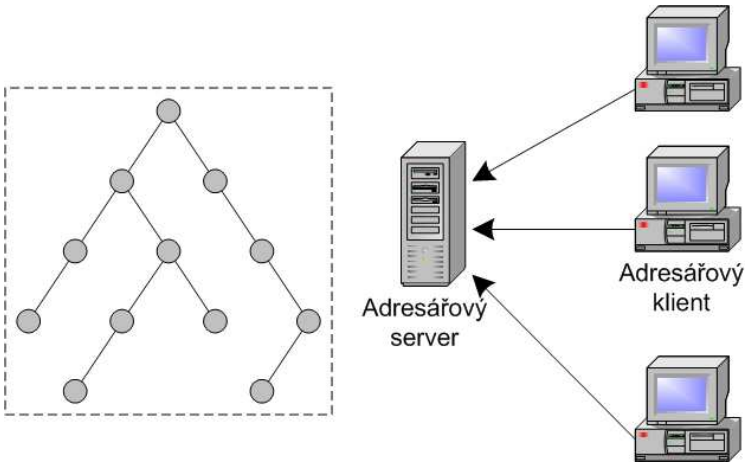
[3]Některé komerční adresářové servery, např. [JES] ji v omezené míře umožňují.

Data jsou většinou uložena v jednoduchých databázích typu Berkeley DB. Ty jsou založeny na principu *klíč/hodnota* (*key/value*), ovšem tento typ databází má v některých implementacích velmi rozsáhlé možnosti, mezi kterými je i možnost transakcí (ty se ovšem netýkají samotné adresářové služby). Mezi dalšími možnostmi ukládání dat jsou klasické textové nebo binární soubory a v některých případech i relační databáze. Např. při použití Oracle Internet Directory je nutné zakoupit i licenci k jejich relační databázi, kterou pak adresář využívá jako úložiště dat. V jiných případech je možné použít pro přístup k relační databázi rozhraní ODBC (*Open Database Connectivity*) nebo *JDBC* (*Java Database Connectivity*).

Každý záznam uložený v databázovém serveru se skládá z atributů a každý z atributů má přiřazenou nějakou hodnotu, jak je vidět z obr. 3.5, strana 17. Blíže o nich pojednává kapitola 3.2.2 na straně 16.

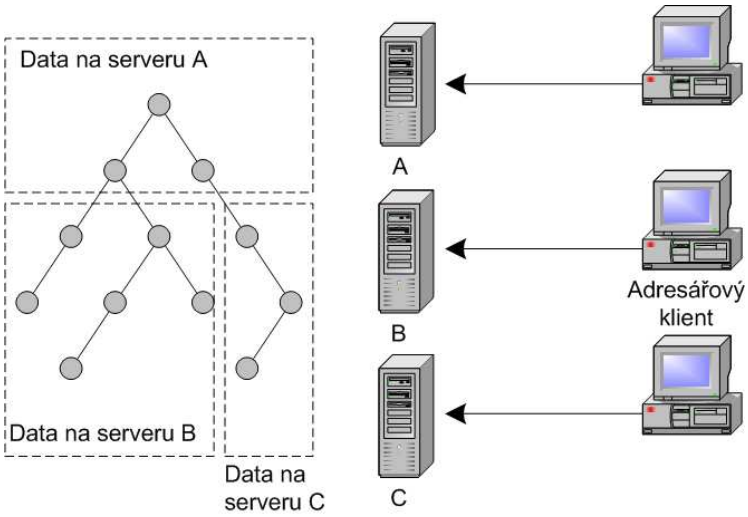
Data adresářových služeb jsou uložena na pevném disku serveru, případně na některém specializovaném hardwarovém a softwarovém řešení typu *RAID*, *LVM*, *SAN* apod. Tato řešení jsou popsána např. v [BfHA].

V případě malého rozsahu bývá adresářový strom uložen na jednom serveru (obr. 2.6, strana 11.), který je navržen na odpovídající výkon a vhodně nakonfigurován.

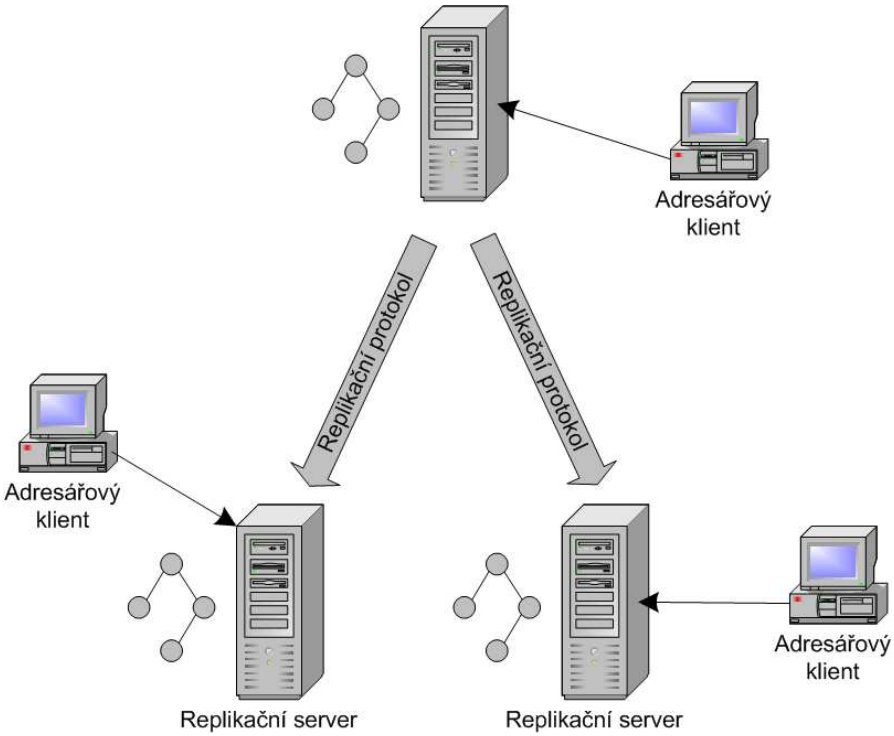


Obr. 2.6: Centralizace adresářových dat na jednom serveru

Někdy je však vhodnější rozdělit adresářový strom na několik jednotlivých větví, které pouze odkazují na data uložená na jiných serverech. V případě rozsáhlých stromů a obrovského množství záznamů, které by zabíraly velký diskový prostor, je výhodné použít rozdělení adresářového stromu na části mezi několik serverů a pak se pomocí referencí odkazovat na jednotlivé větve (obr. 2.7, strana 12.). O tomto problému pojednává kapitola 3.7. Topologie na straně 42. Tuto možnost lze použít při dobrém návrhu adresářového stromu, kdy je třeba vzít v úvahu, jak je to pro konkrétní případ výhodné z hlediska dostupnosti dat a ceny spojení. Pro zvýšení bezpečnosti ukládaných dat je použit princip replikací. *Replikace* je proces, který slouží pro správu a údržbu mnoha kopii dat mezi několika lokalitami. Tyto replikace mohou sloužit pro dotazování klientů a tím odlehčit hlavnímu serveru (obr. 2.8, strana 12.). V některých případech je dokonce vhodné nastavit politiku replikací tak, že uživatelé nemohou z centrálního adresářového serveru číst, ale mohou do nich zapisovat. Naopak z replik je možné pouze číst. Repliky je vhodné umístit na pobočky, které by byly nuceny se neustále dotazovat centrálního serveru a zatěžovaly by tak linky připojení k internetu.



Obr. 2.7: Distribuce adresářových dat mezi více serverů



Obr. 2.8: Replikované adresářové služby

Kapitola 3

LDAP

Lightweight Directory Access Protocol, ve zkratce *LDAP* [RFC 2251], je zjednodušenou verzí adresářového protokolu *X.500* a původně sloužil pro komunikaci s tímto protokolem. Postupem času se vyvinul v plnohodnotný adresářový protokol, který je schopen svému předchůdci zdatně konkurovat a je v podnikové sféře mnohem více použitelnější, než jeho předchůdce. V současné době využívají protokol *X.500* pouze velké telekomunikační společnosti. Podniková sféra a ostatní organizace přecházejí na *LDAP* nebo na Microsoft Active Directory.

Co je to LDAP?

Význam zkratky *LDAP* lze shrnout do několika následujících termínů:

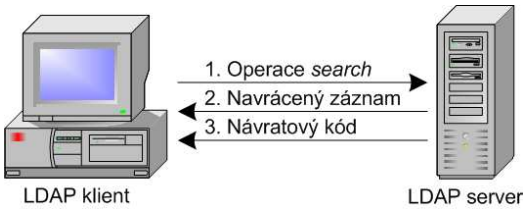
- *Lightweight Directory Access Protocol* – standardizovaný protokol pro přístup k adresářovým službám
- Soubor čtyř modelů – informační, jmenný, funkční a bezpečnostní model
- *LDAP Data Interchange Format* (*LDIF*) – standardizovaný textový formát pro výměnu adresářových dat
- *LDAP* serverový software – soubor software pro provoz LDAP serveru
- *LDAP* klientský software – soubor programů schopných práce s LDAP serverem
- *LDAP API* – aplikační programové rozhraní pro vývoj software

3.1 Protokol LDAP

LDAP je protokol orientovaný na zprávy (*message-oriented protocol*). Klient vytvoří zprávu obsahující nějakou žádost a odešle ji serveru. Ten ji zpracuje a odešle výsledek zpět jako sérii jedné nebo více *LDAP* zpráv.

Pokud např. klient hledá v adresáři konkrétní záznam, zašle *LDAP* serveru žádost o vyhledání (*LDAP search request*). Ta obsahuje pokud možno co nejpřesnější umístění záznamu v adresářovém stromě, vyhledávací filtry, požadované atributy, ověřovací údaje ad.

V případě, že server nalezne více záznamů, zašle je klientu zpět v sérii určitého pořadí (obr. 3.2, strana 14.).



Obr. 3.1: Jeden nalezený záznam

Obr. 3.2: Více nalezených záznamů

3.1.1 Operace protokolu LDAP

Protokol *LDAP* má devět základních operací. Ty lze dále rozdělit do třech kategorií, podrobněji popsaných v kapitole 3.4 Funkční model, strana 30.:

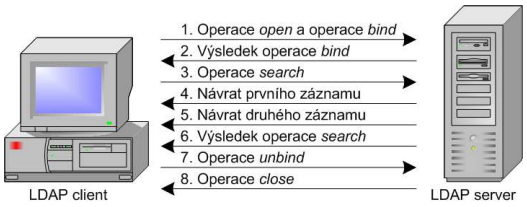
Dotazovací operace: `search`, `compare`. Tyto dvě operace umožňují klást adresářovému serveru dotazy.

Aktualizační operace: `add`, `delete`, `modify`, `modify DN` (`rename`). Tyto operace umožňují úpravy záznamů v adresářovém serveru

Autentizační a řídící operace: `bind`, `unbind`, `abandon`.

Příklad typické komunikace klienta se serverem při hledání záznamu pomocí protokolu *LDAP* je na obr. 3.3, strana 15.

1. Klient otevře *TCP* spojení s *LDAP* serverem a zašle příkaz `bind`. Ten slouží pro autentizaci klienta
2. Server ověří klientovu identitu a autentizuje jej pro další operace. V opačném případě ukončí s klientem spojení
3. Klient vytvoří a odešle žádost o nalezení záznamu
4. Server odešle klientu záznam nebo v sérii za sebou všechny záznamy, které nalezl
5. Server odešle návratový kód
6. Klient odešle serveru žádost o operaci `unbind`, která serveru oznámí, že klient chce ukončit spojení
7. Server uzavře spojení



Obr. 3.3: Typická komunikace protokolu LDAP

3.1.2 Protokol LDAP v provozu

Základní pravidla pro kódování (*Basic Encoding Rules – BER* [Dostalek] kapitola 6.) jsou souborem norem pro zakódování různých typů dat. Těmito typy jsou např. celá čísla, textové řetězce a další. *BER* ukládá data ve zhuštěném a na systému nezávislém tvaru, což je důležité u rozdílných procesorových architektur v nehomogenním síťovém prostředí, jakým je např. Internet. *BER* také definuje způsoby, jakými lze kombinovat datové typy do různých datových struktur jako jsou třídy a sekvence.

Protokol *LDAP* používá zjednodušenou verzi protokolu *BER*, tzv. *Lightweight BER*, zkratka *LBER*. *BER* byl zrevidován a byly z něj vypuštěny některé prvky, které dělaly vlastní protokol *BER* velmi složitým na pochopení a implementaci.

Při vlastním provozu po síti používá protokol *LDAP* kódování dat pomocí *LBER*, takže data neproudí v čistě textové podobě jako například při komunikaci protokolem *HTTP* a nelze je snadno číst např. pomocí telnetu. Pro zpracování přenesených dat je nutné opět využít knihovnu *LBER*. Většina analyzátorů síťového provozu však nemá s protokolem *LDAP* a kódováním *LBER* problémy a není problém např. pomocí oblíbeného programu Ethereal[1] přečíst komunikaci mezi serverem a klientem. Pro její ochranu se proto využívá zabezpečení popsaného v kapitole 3.5 Bezpečnostní model na straně 35.

V případě některého serverového software je možné využít moderního formátu *DSMLv2* (viz. kapitola 3.6 LDIF, strana 39.) založeného na jazyce *XML*.

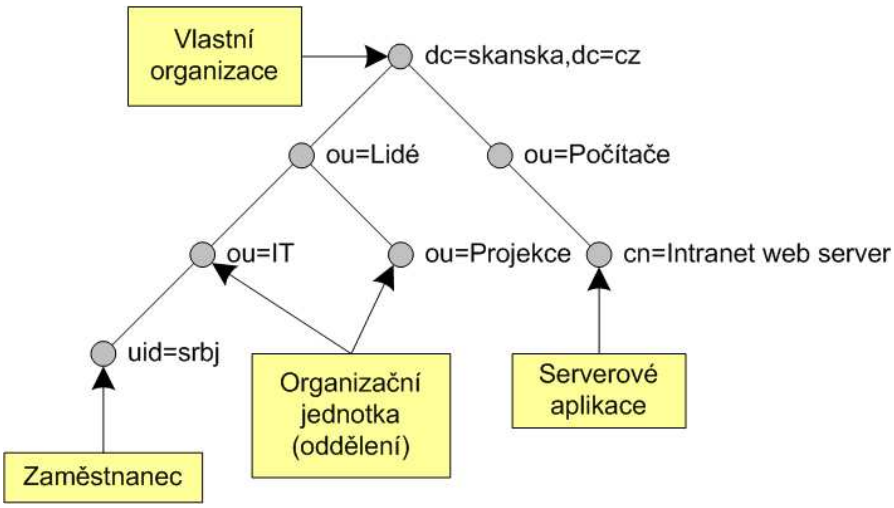
3.2 Informační model

Informační model definuje datové typy a jednotky informací, které lze v adresáři ukládat. V adresáři může být uloženo několik tisíc až milionů podrobných záznamů o lidech, počítačích, tiskárnách, přístrojích, místnostech apod. Záznamy jsou uloženy v přehledné stromové struktuře, která vyjadřuje, jakým stylem jsou záznamy v adresáři organizovány. Jejich organizací se zabývá jmenný model (viz. kapitola 3.3 Jmenný model, strana 22).

3.2.1 Identifikátory objektů

Identifikátory objektů nejsou součástí *LDAP* modelu, je ale důležité se o nich zmínit. Slouží k přesné identifikaci prvku, který je jednoznačně určen svým globálním *OID* (*Object Identifier*). Mají hierarchickou strukturu a jsou unikátní na celém světě, nelze je tedy zvolit podle libosti. Tuto unikátnost zaručuje centrální autorita, která přiděluje čísla (*sufixy*) pro sestavení

[1] Ethereal – `http://www.ethereal.com`



Obr. 3.4: Část typického adresáře

OID. O jejich přidělování soukromým organizacím se stará mezinárodní organizace IANA[2]. Jedné organizaci je možné přidělit pouze jedno OID, které si musí organizace s ohledem na svůj rozvoj vhodným způsobem rozdělit. *OID* mají tvar `1.3.6.1.4.1.XXX`, kde `XXX` je zmíněný přidělený suffix reprezentovaný celým číslem. Seznam sufixů jednotlivých organizací je k dispozici na adrese organizace IANA[3]. Čísla za sufixem jsou spravována společnosti, která si je může rozdělit s ohledem na své interní číselníky. Pokud by např. Skanska CZ získala sufix `332211`, její *OID* by mělo tvar `1.3.6.1.4.1.332211` a jeho další rozdělení na jednotlivé části by mohlo vypadat např. podle tab. 3.1, strana 16.

OID	Význam
1.3.6.1.4.1.332211	OID Skanska CZ
1.3.6.1.4.1.332211.1	SNMP elementy
1.3.6.1.4.1.332211.2	LDAP elementy
1.3.6.1.4.1.332211.2.1	Typy atributů
1.3.6.1.4.1.332211.2.1.1	Vlastní atribut
1.3.6.1.4.1.332211.2.2	Objektové třídy
1.3.6.1.4.1.332211.2.2.1	Vlastní objektová třída

Tab. 3.1: Příklad hierarchie OID

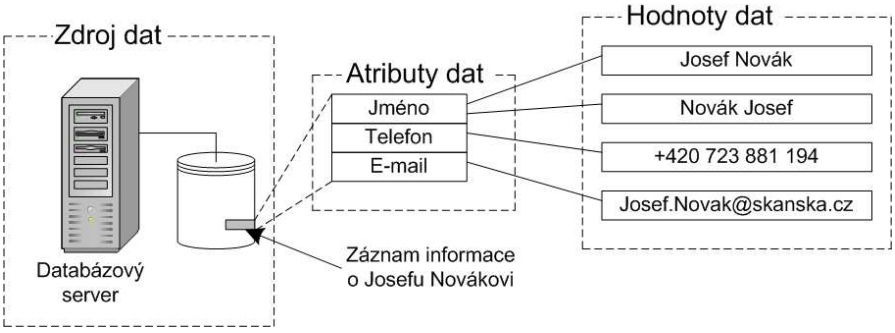
Více informací o *OID* se lze dozvědět např. z [UDLDAPDS], nebo z [Dostalek], kapitola 6.3.

3.2.2 Záznamy

Záznam (*entry*) je základní jednotkou informací v adresářových službách. Jedná se o soubor informací o *objektech*, resp. o *objektových třídách* (*objectclass*). Každý záznam se snaží

[2] `http://www.iana.org`
[3] `http://www.iana.org/assignments/enterprise-numbers`

nějakým způsobem popsat reálný objekt, např. člověka, počítač apod. a je složen z jedné nebo více objektových tříd. Objektová třída je množinou *typů atributů* (*Attribute Types*) nebo zjednodušeně *atributů* (*Attributes*). Ty slouží k popisu vlastností této třídy. Atributy mají jednu nebo více *hodnot* (*value*). Tyto návaznosti zachycuje obr. 3.5, strana 17.



Obr. 3.5: Organizace dat

Příkladem záznamu může být obr. 3.6, strana 17., kde jsou popsány informace, které jsou o dané osobě uchovány.

Typ atributu (*Attribute type*)	**Hodnoty atributu** (*Attribute values*)
`cn:`	Karel Benák Benák Karel
`sn:`	Benák
`telephoneNumber:`	+420 723 881 194
`mail:`	karel.benak@sin.cvut.cz benak.karel@sin.cvut.cz

Obr. 3.6: Adresářový záznam

Atributy mají určitou *syntaxi* (*syntax*) a sadu *pravidel shody* (*matching rules)*. Syntaxe (tab. 3.2, strana 18..) specifikuje, v jaké formě mohou být informace v atributu reprezentovány, například v datovém typu `JPEG Image` lze uložit pouze JPEG obrázek (foto uživatele).

Při návrhu nových atributů, objektových tříd, pravidel shody nebo datových typů je vhodné, aby jejich pojmenování mělo jedinečný prefix. *LDAP* vyžaduje jedinečné názvy pro všechny atributy a objektové třídy. Prefix se nejčastěji volí tak, aby obsahoval zkratku organizace kvůli možnému konfliktu názvů s případnými rozšiřujícími schematy. Např. `myPhoto` je nevhodný název, neboť stejně znějících názvů může být víc. Vhodnější je např. pojmenování příslušného atributu názvem `skanskaPhoto`,

3.2.3 Syntaxe

Syntaxe atributů je definována v [RFC 2252]. Ve své podstatě definuje datový typ, který bude atribut používat. Vybrané typy syntaxe jsou v tab. 3.2, strana 18. Pokud je nutné omezit délku atributu na předem stanovený počet míst, uvede se za *OID* syntaxe sekvence {`počet míst`}.

Název	OID	Popis
binary	1.3.6.1.4.1.1466.115.121.1.5	BER/DER data
boolean	1.3.6.1.4.1.1466.115.121.1.7	boolean hodnota
distinguishedName	1.3.6.1.4.1.1466.115.121.1.12	DN
directoryString	1.3.6.1.4.1.1466.115.121.1.15	UTF-8 řetězec
IA5String	1.3.6.1.4.1.1466.115.121.1.26	ASCII řetězec
Integer	1.3.6.1.4.1.1466.115.121.1.27	celé číslo
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	DN plus UID
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	číselný řetězec
OID	1.3.6.1.4.1.1466.115.121.1.38	object identifier

Tab. 3.2: Syntaxe vybraných typů

3.2.4 Pravidla shody

Pravidla shody jsou rozdělena na pravidla pro:

porovnávání rovnosti hodnot – podle těchto pravidel jsou například hodnoty atributů `sn=Novák` a `sn=novák` při použití pravidla `caseIgnoreMatch` rovny, zatímco při použití jiných pravidel (`caseExactMatch`) tomu tak není.

setřídění hodnot – podle těchto pravidel se například při použití zmíněného pravidla `caseIgnoreMatch` hodnoty setřídí podle lexikografického pořadí. Při použití pravidla `integerMatch` budou hodnoty setříděny podle číselné hodnoty.

Název	OID	Typ	Popis
objectIdentifierMatch	2.5.13.0	shoda	OID
distinguishedNameMatch	2.5.13.1	shoda	DN
caseIgnoreMatch	2.5.13.2	shoda	velikost a mezery nerozlišuje
caseIgnoreOrderingMatch	2.5.13.3	uspořádání	velikost a mezery nerozlišuje
caseIgnoreSubstringsMatch	2.5.13.4	podřetězec	velikost a mezery nerozlišuje
caseExactMatch	2.5.13.5	shoda	velikost rozlišuje, mezery ne
caseExactOrderingMatch	2.5.13.6	uspořádání	velikost rozlišuje, mezery ne
caseExactSubstringsMatch	2.5.13.7	podřetězec	velikost rozlišuje, mezery ne
booleanMatch	2.5.13.13	shoda	boolean

Tab. 3.3: Vybraná pravidla shody

Jak je z tab. 3.3, strana 18. a především z *OID* jednotlivých pravidel vidět, jsou tato pravidla shody většinou převzata z protokolu *X.500*. Podle [RFC 2252], část 4.5 si každý návrhář může definovat vlastní pravidla.

3.2.5 Atributy

Atributy slouží k popisu některé z vlastností objektu. V protokolu *LDAPv3* podle [RFC 2252] mají tvar zobrazený na obr. 3.7, strana 19. Jeho použití je definováno pomocí direktivy `USAGE`, která používá výčtový typ `AttributeUsage` (obr. 3.8, strana 19.).

```
AttributeTypeDescription = "(" whsp
       numericoid whsp              ; Identifikátor Typu atributu
     [ "NAME" qdescrs ]             ; Název používaný typem atributu
     [ "DESC" qdstring ]            ; Popis
     [ "OBSOLETE" whsp ]
     [ "SUP" woid ]                 ; Odvození od jiného typu
                                    ; atributu
     [ "EQUALITY" woid              ; Název pravidla shody pro shodu
     [ "ORDERING" woid              ; Název pravidla shody pro porovnání
     [ "SUBSTR" woid ]              ; Název pravidla shody pro podřetězec
     [ "SYNTAX" whsp noidlen whsp ] ; OID syntaxe
     [ "SINGLE-VALUE" whsp ]        ; Standardně vícehodnotové
     [ "COLLECTIVE" whsp ]          ; Standardně not collective
     [ "NO-USER-MODIFICATION" whsp ]; Standardně modifikovatelné uživatelem
     [ "USAGE" whsp AttributeUsage ]; Standardně userApplications
     whsp ")"
```

Obr. 3.7: Popis typu atributu

```
AttributeUsage =
     "userApplications"     /
     "directoryOperation"   /
     "distributedOperation" / ; DSA-shared
     "dSAOperation"          ; DSA-specific, hodnota závisí na serveru
```

Obr. 3.8: Využití atributu

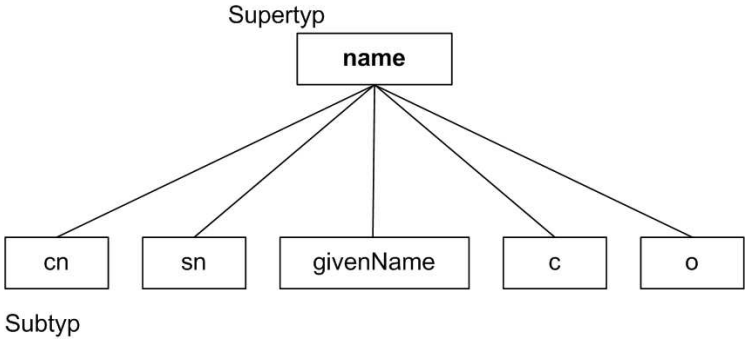
Atributy lze podle direktivy `USAGE` rozdělit do dvou kategorií:

uživatelské – atributy používané pro ukládání uživatelských informací. V takovém atributu může být uloženo například jméno uživatele (`givenName`) apod. Mohou být měněny uživatelem, který k tomu má příslušná oprávnění.

operační – atributy, které využívá adresářový server pro uchování systémových informací, např. data poslední změny `modifyTimeStamp`. Takový atribut má každý záznam a pokud dojde k jeho změně, server automaticky změní i hodnotu tohoto atributu.

Atributy mohou mít jednu (*single-valued*) nebo více (*multivalued*) hodnot, resp. mohou být v objektové třídě pouze jednou (direktiva `SINGLE-VALUE`) nebo vícekrát.

Při vytváření atributů je možné využít principu tzv. *supertypu* a *subtypu*. Supertyp je základní typ, ze kterého se vychází při odvozování nových typů. Schema závislostí zobrazuje obr. 3.9, strana 20..



Obr. 3.9: Vztahy mezi supertypem a subtypy

```
attributeType ( 1.3.6.1.4.1.332211.2.1.12 NAME 'skanskaCard'
     DESC 'ID pristupove karty pro pristup do budov Skanska CZ'
     EQUALITY integerMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27{8}
     SINGLE-VALUE )
```

Obr. 3.10: Příklad nového atributu skanskaCard

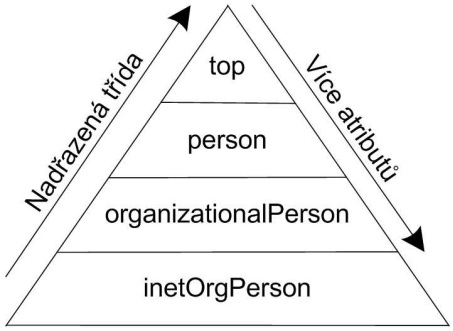
```
attributeType ( 1.3.6.1.4.1.332211.2.1.13 NAME 'skanskaCardCar'
     DESC 'ID pristupove karty do garazi Skanska CZ'
     SUP skanskaCard SINGLE-VALUE )
```

Obr. 3.11: Odvozený atribut skanskaCardCar

3.2.6 Objektové třídy

Objektová třída slouží pro popis konkrétního objektu, např. člověka, zaměstnance, uživatele připojení, místnosti apod. Skládají se z několika atributů a jsou většinou odvozeny od některé

z nadřazených tříd. Hlavní třídou, je abstraktní třída `top`, od které jsou všechny ostatní třídy odvozeny. Princip dědičnosti a jeho význam je na obr. 3.12, strana 21. Při vytváření nové třídy získává po svém předkovi jeho atributy a přidává k němu vlastní.



Obr. 3.12: Dědičnost objektových tříd

Tvar objektové třídy podle *LDAPv3* definované v [RFC 2252] je na obr. 3.12, strana 21..

```
ObjectClassDescription = "(" whsp
     numericoid whsp      ; Identifikace objektové třídy
     [ "NAME" qdescrs ]
     [ "DESC" qdstring ]
     [ "OBSOLETE" whsp ]
     [ "SUP" oids ]       ; Nadřazená třída
     [ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ] ; Standardně structural
     [ "MUST" oids ]      ; Atributy
     [ "MAY" oids ]       ; Atributy
     whsp ")"
```

Obr. 3.13: Definice objektové třídy

Objektová třída musí mít své jedinečné *OID*, jedinečný název, popis, předka, druh a seznam atributů, které mohou být povinné nebo nepovinné. Povinné atributy (`MUST`) musí být vyplněny vždy, jinak adresářový server odmítne záznam uložit nebo modifikovat. Atributy uvedené v direktivě `MAY` nemusejí být vyplněny. V záznamu musí být minimálně jedna třída typu `STRUCTURAL`, nelze jej složit pouze ze tříd typu `AUXILIARY`.

`ABSTRACT` – Abstraktní třída, od které se budou odvozovat nové třídy, nebo třídy sloužící pro správu informačního modelu. Nejčastěji používanou abstraktní třídou je třída `top`, od které jsou odvozeny další třídy a třída `alias`, která slouží pro správu aliasů (kapitola 3.3.3 Aliasy na straně 29).

`AUXILIARY` – Třída, která je doplňkovou třídou k ostatním třídám. Velmi často má jako svého předka třídu `top`. Lze ji přiřadit ke každému typu záznamu, v záznamu je však třeba použít minimálně jednu třídu typu `STRUCTURAL`. Tyto třídy lze v záznamu kombinovat bez ohledu na jejich určení.

`STRUCTURAL` – Podobně jako třída `AUXILIARY` je odvozena od některé třídy uvedené v direktivě `SUP`. Je mnohem více restriktivní a umožňuje kombinaci pouze se třídami nadřazenými, které přesně vystihují typ ukládaného záznamu. Typickým příkladem třídy typu `STRUCTURAL` je třída `inetOrgPerson` (definovaná v [RFC 2798]), odvozená od třídy `organizationalPerson`. K této třídě není možné přiřadit další třídu typu `STRUCTURAL`, např. třídu `account`. Ta je sice potomkem třídy `top`, ovšem není potomkem třídy `person`, tudíž ji není možné sloučit s třídami, které mají třídu `person` jako svého předka a jsou typu `STRUCTURAL`.

```
objectclass ( 1.3.6.1.1.1.2.4 NAME 'ipProtocol' SUP top STRUCTURAL
        DESC 'Abstraction of an IP protocol'
        MUST ( cn $ ipProtocolNumber $ description )
        MAY description )
```

Obr. 3.14: Objektová třída typu STRUCTURAL

```
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount' SUP top AUXILIARY
     DESC 'Abstraction of an account with POSIX attributes'
     MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
     MAY ( userPassword $ loginShell $ gecos $ description ) )
```

Obr. 3.15: Objektová třída typu AUXILIARY

Příklady záznamů jsou uvedeny v kapitole 3.6 LDIF na straně 39.

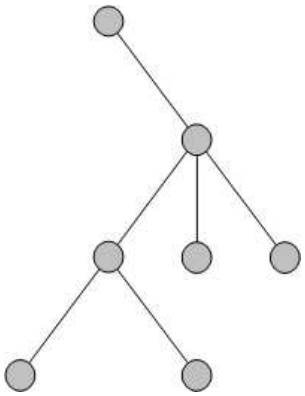
3.2.7 Schémata

Seznam atributů a objektových tříd, se kterými adresářový server pracuje, je uložen v tzv. schematech. Standardizovaná schemata jsou většinou součástí adresářových serverů, stejně jako schemata dodaná výrobcem. Ačkoliv je doporučeno v maximální míře využívat schemat dodávaných spolu s adresářovým serverem, může vzniknout potřeba napsat schema vlastní. Pokud tvůrci informačního systému sáhnou k této variantě, musí počítat s tím, že některé informace budou pro klienty podporující standardizovaná schemata nedostupná, případně se budou muset složitě přemapovat na standardizované atributy.

Podrobnější informace o objektových třídách, atributech, pravidlech syntaxe a pravidel shody jsou v [UDLDAPDS, RFC 2252].

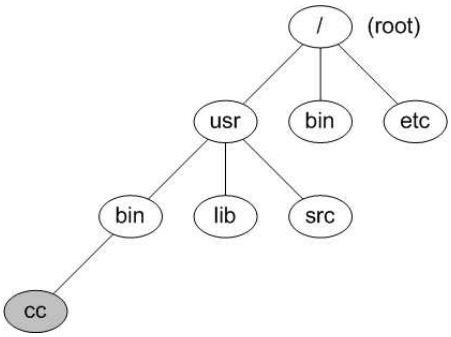
3.3 Jmenný model

Jmenný model LDAP (*LDAP Naming Model*) definuje, jakým způsobem lze organizovat a odkazovat se na data uložená v adresáři. Umožňuje tak flexibilní přístup a jednodušší správu pomocí logické stromové struktury. Typický tvar adresářového stromu, tzv. *DIT* (*Directory Information Tree*), je na obr. 3.16, strana 23. Jedná se o stromový graf, který má své uzly a listy. Listy jsou vlastní záznamy o nějakém reálném objektu, uzly pak definují větve stromu. Uzly jsou nutnou součástí adresářového stromu, do kterých se ukládají jednotlivé listy (záznamy).



Obr. 3.16: Část typického adresářového stromu

Adresářový strom je podobný hierarchickému souborovému systému používaného operačními systémy typu UNIX, ve kterém se systém souborů skládá z adresářů a souborů. Každý adresář má několik souborů anebo podadresářů, jak je vidět z obrázku 3.17.



Obr. 3.17: Část typického souborového systému UNIX

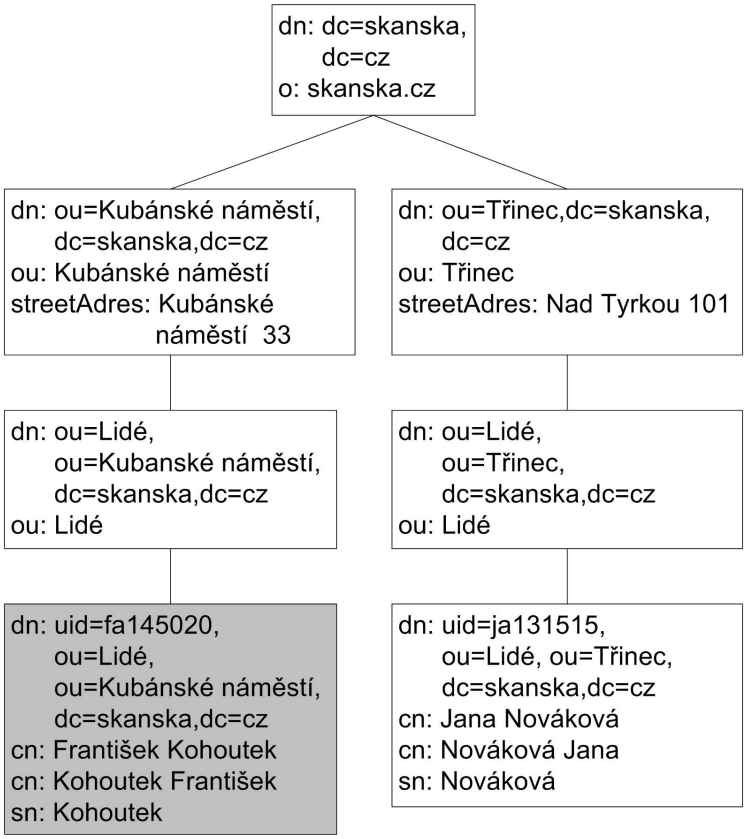
Mezi UNIXovým souborovým systémem a adresářovým stromem *LDAP* jsou však tři podstatné rozdíly. První spočívá v tom, že *LDAP* server ve skutečnosti nemá žádný reálný kořen, zatímco kořen souborového systému je rodičem dalších adresářů a souborů. LDAP server má sice kořenový záznam, ten ale používá pro uložení specifických konfiguračních informací o svých možnostech, tzv. `rootDSE`. Jsou v něm uloženy např. informace o používané verzi protokolu, *FQDN*[4] serveru, podpoře protokolů *TLS* a *SASL* apod., nikoliv však vlastní uživatelská data. Pro ně může mít několik "virtuálních" kořenů, zatímco souborový systém má pouze jediný kořen.

Druhým podstatným rozdílem je způsob ukládání informací. Typický *LDAP* adresář se záznamy a kontejnery ukazuje obr. 3.18, strana 24..

V tomto případě kontejnery

```
dc=skanska, dc=cz
ou=Kubánské~náměstí, dc=skanska, dc=cz
```

[4]FQDN – fully qualified domain name, plný doménový název



Obr. 3.18: Část typického LDAP adresáře

```
ou=Třinec, dc=skanska, dc=cz
ou=Lide, ou=Kubánské náměstí, dc=skanska, dc=cz
ou=Lide, ou=Třinec, dc=skanska, dc=cz
```

vystupují zároveň jako uzly (kontejnery) i listy (záznamy), ve kterých jsou uloženy další kontejnery a záznamy. Naproti tomu v souborovém systému vystupuje adresář pouze jako uzel a soubor pouze jako list. Každý uzel může obsahovat další uzly i listy, nemůže se však stát zároveň listem i uzlem dohromady.

Třetím podstatným rozdílem je způsob, jakým se vyjadřuje cesta k záznamu nebo souboru. Zatímco v případě souborového systému je cesta ke kompilátoru `cc` vyjádřena od kořene systému `/` přes adresáře `usr` a `bin` až k `cc` jako `/usr/bin/cc` z leva do prava, v *LDAP* adresáři je cesta od kořene k záznamu vyjadřována z prava do leva (`uid=fa145020, ou=Lidé, ou=Kubánské náměstí, dc=skanska, dc=cz`). Zatímco vlastní záznam o pracovníkovi s `uid=fa145020` je uveden na prvním místě, kořenový kontejner je uveden až na místě posledním.

3.3.1 Význam jmenného modelu

Správně navržené jmenné prostory jsou velmi důležité a přinášejí velké výhody při správě a údržbě záznamů.

Reference dat – jmenný model umožňuje pohodlné odkazování se z jednoho záznamu na druhý například pomocí referencí, aliasů, nebo pomocí atributů, které mají jako svůj datový typ *DN* zázamu, na který je odkazováno.

Organizace dat – data jsou uložena v jednotlivých kontejnerech podle nějakého specifického znaku. Ten je v daném kontejneru jedinečný a tím je stanovena jedinečnost záznamu.

Rozdělení dat – data lze efektivně rozdělit mezi více poboček a není nutné je držet všechny na jediném serveru v centrále. O topologii pojednává kapitola 3.7 Topologie na straně 42.

Replikace dat – při správně navržených jmenných prostorech je efektivnější zálohovat pouze část příslušného stromu, nikoliv všechny jeho části

Přístupová práva – při správně navržených jmenných prostorech lze přiřadit přístupová práva v konkrétní větvi konkrétnímu člověku, aniž by měl dotyčný práva k datům, která nezbytně nepotřebuje. Příkladem může být situace, kdy správce Třinecké pobočky Skansky spravuje pouze svou větev `ou=Trinec, dc=skanska, dc=cz`, zatímco ke zbylým větvím má přístup pouze pro čtení, nikoliv pro modifikaci záznamů.

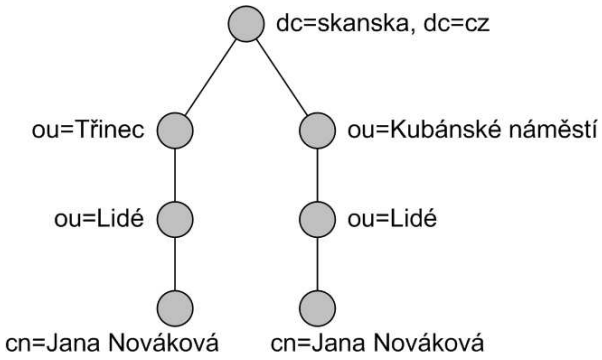
Podpora aplikací – aplikace mohou využívat pouze část stromové struktury a nemusejí zbytečně prohledávat všechny záznamy. Příkladem je seznam pracovníků konkrétní pobočky `ou=Pracovnici, ou=Trinec, dc=skanska, dc=cz`, kdy klient se ptá pouze v této větvi, aniž by musel prohledávat i další větve.

Jmenný model poskytuje unikátní a jedinečné názvy pro každý záznam v adresáři umožňující snadné a přesné odkazování se. Tuto jedinečnou identifikaci záznamu v adresářovém stromu umožňuje tzv. **rozlišovací název**, zkratka **DN**, (*distinguished name*), který je úplnou cestou k danému záznamu. Příkladem typického *DN* je již zmiňovaný záznam o uživateli s `uid=fa145020`. V cestě jsou směrem z prava do leva uvedeny jednotlivé kontejnery oddělené čárkou (mezera

je volitelná). Na konci cesty je uveden vlastní záznam. Název záznamu se skládá z názvu atributu, znaku = a hodnoty atributu. Následující záznamy jsou si rovny:

```
uid=fa145020, ou=Lidé, ou=Kubánské náměstí, dc=skanska, dc=cz
uid=fa145020,ou=Lidé,ou=Kubánské náměstí,dc=skanska,dc=cz
```

S pojmen *rozlišovací název* úzce souvisí pojem **relativní rozlišovací název** (*relative distinguished name* – **RDN**). RDN je název vlastního záznamu který musí být v daném kontejneru jedinečný. Z *DN* `cn=Jana Nováková, ou=Lide, ou=Třinec, dc=skanska, dc=cz` je *RDN* `cn=Jana Nováková`. V této větvi se již žádný další záznam s tímto *RDN* nesmí vyskytovat, ovšem v dalších větvích ano.



Obr. 3.19: Záznamy se shodným RDN

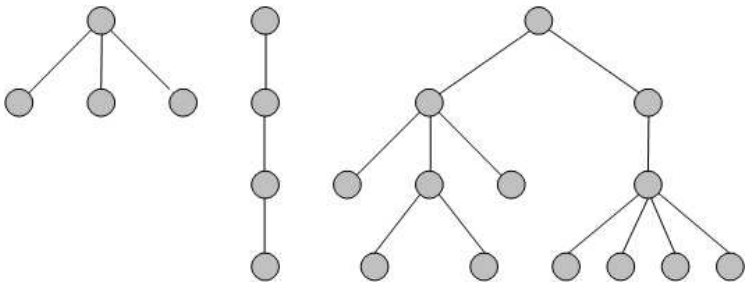
V některých případech je třeba použít stejnou hodnotu atributu v *RDN* pro dva nebo více záznamů, například pokud je třeba v jedné větvi držet záznamy o dvou Janách Novákových. V takovém případě se používá tzv. *vícehodnotové RDN* (*multivalued RDN*). V *RDN* je použito více hodnot atributů oddělených znakem +. Tvar *RDN* pak vypadá např. následovně: `cn=Jana Nováková + mail=jana.novakova@skanska.cz`

Ve vícehodnotovém RDN nezáleží na pořadí, proto si jsou následující DN rovny:

```
cn=Jana Nováková + mail=jana.novakova@skanska.cz, ou=Lidé,
ou=Kubánské náměstí, dc=skanska, dc=cz

mail=jana.novakova@skanska.cz + cn=Jana Nováková, ou=Lidé,
ou=Kubánské náměstí, dc=skanska, dc=cz
```

Bohužel, málo aplikací umí pracovat s vícehodnotovými *RDN*, proto se doporučuje volit název *RDN* tak, aby byla zajištěna jeho unikátnost v rámci daného kontejneru pomocí jediného názvu v *RDN*.

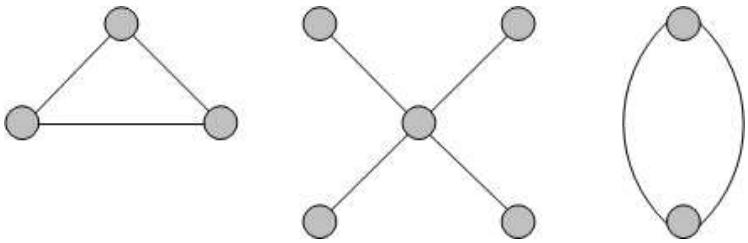
V případě nutnosti použití speciálních znaků v *DN*, jakým je především znak čárky sloužící standardně pro oddělení jednotlivých záznamů, je nutné použít tzv. *únikových* nebo-li *escape sekvencí*, které jsou uvozeny obráceným lomítkem \, tzv. *backslash*. Příkladem by mohl být záznam `o=Skanska CZ, a.s.` V tom případě by *DN* záznamu vypadalo: `o=Skanska CZ\, a.s., c=CZ`



Obr. 3.20: Podporované jmenné prostory

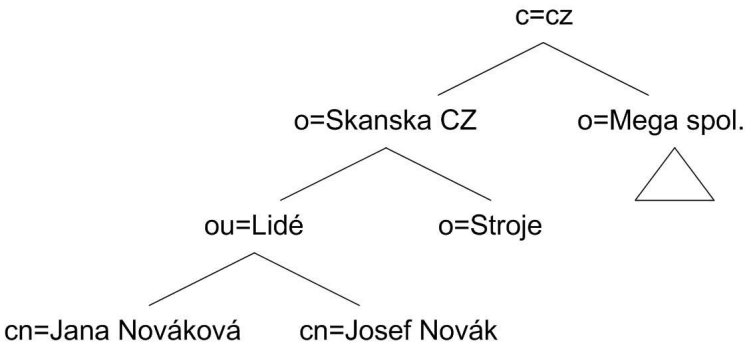
3.3.2 Struktura jmenných prostorů

LDAP podporuje stromové struktury podle obrázku obr. 3.20, strana 27., naopak některé tvary (obr. 3.21, strana 27.) nejsou podporované a odporují stromové struktuře.



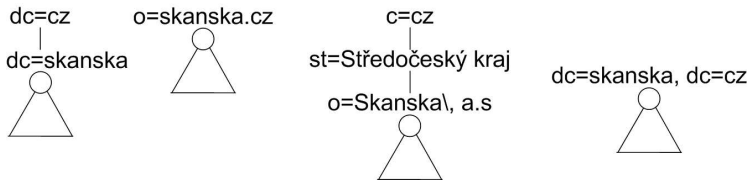
Obr. 3.21: Nepodporované jmenné prostory

Kořen jmenných prostorů se nazývá *sufix* (*suffix*), někdy též *prefix*. Model jmenných prostorů *LDAP* je zděděn z modelu jmenných prostorů standardu *X.500*. Ten začíná vrcholovým prvkem `C` (Country), ve kterém je uvedena *ISO* zkratka země. Pak může následovat prvek `S` pro uvedení názvu státu (standard *X.500* vychází z norem USA) a po něm následuje záznam organizace `O`. Příklad adresářového stromu podle normy *X.500* je na obr. 3.22, strana 27. a *DN* záznamu o Janě Novákové by vypadalo: `cn=Jana Nováková, ou=Lide, o= Skanska CZ, c=cz`



Obr. 3.22: Jmenné prostory podle normy X500

Na rozdíl od *X.500* není model *LDAP* tak restriktivní jako jeho předchůdce a umožňuje flexibilnější návrh sufixu, který více odpovídá potřebám organizace pro kterou je navrhován. Je možné držet se podle normy *X.500*, ovšem doporučuje se spíš použít v sufixu doménové jméno podle [RFC 2247]. Sufix společnosti Skanska CZ pak vypadá `dc=skanska, dc=cz`, více příkladů návrhů sufixů je na obr. 3.23, strana 28. V *DN* záznamů je možné použít i diakritické

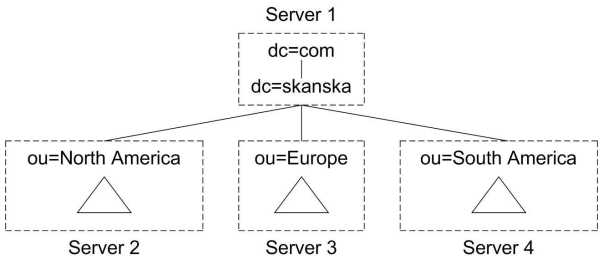


Obr. 3.23: Příklady sufixů v LDAP

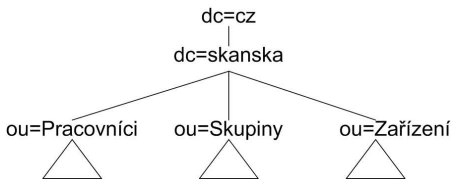
znaky, ovšem pak je nutné uvést *DN* v kódování *UTF-8* podle [RFC 2253].

Další dělení adresářového stromu na jednotlivé větve záleží na podrobnější analýze, kdy je třeba vzít v úvahu potřeby společnosti, její síťovou topologii, potřebu replikací a celou řadu dalších podmínek. V tomto případě se lze setkat s pojmem *kontext* (*context*), což je ve své podstatě cesta od sufixu až k *RDN* záznamu. *DN* se ve své podstatě skládá z *RDN*, kontextu a sufixu. Příkladem je např. *DN* záznamu `uid=ja131515, ou=Lide, ou=Trinec, dc=skanska, dc=cz`, kdy *RDN* záznamu je `uid=ja131515, ou=Lide`, kontext je `ou=Lide, ou=Trinec` a konečně hodnota sufixu `dc=skanska, dc=cz`.

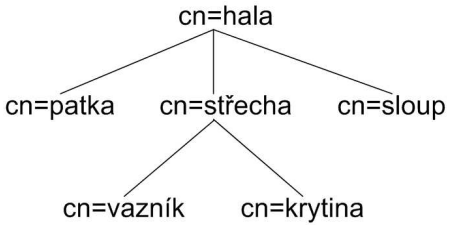
Příklady jmenných prostorů jsou na obr. 3.24, strana 28., obr. 3.25, strana 28. a obr. 3.26, strana 29.



Obr. 3.24: Jmenné prostory podle kontinentů



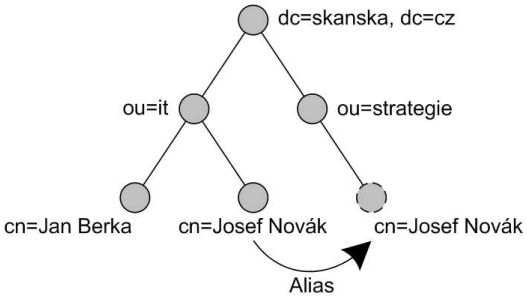
Obr. 3.25: Jmenné prostory podle oddělení



Obr. 3.26: Jmenné prostory podle součástí

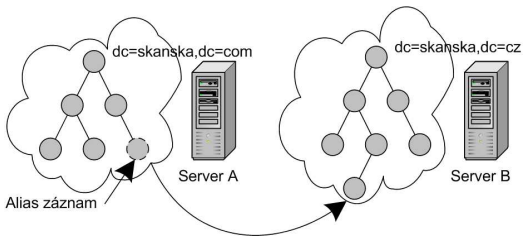
3.3.3 Aliasy

Záznamy *Alias* jsou záznamy v adresářovém stromu, které fungují jako odkazy na uložený záznam. Jsou velmi podobné klasickým symbolickým linkům ze souborového systému typu UNIX nebo zástupcům ve Windows. Pomocí aliasu lze dosáhnout umístění jednoho záznamu do více větví, např. pokud jeden pracovník pracuje pro dvě oddělení. Při použití *SQL* by bylo nutné použít např. speciální tabulku, která by postihovala relaci *m:n* mezi tabulkou pracovníků a tabulkou oddělení. V adresářové struktuře je naproti tomu nutné buď duplikovat záznam, což je nevyhovující kvůli redundanci dat, nebo se zde nabízí možnost použití aliasů, kdy hlavní záznam je uložen v jedné větvi stromu a z druhé větve je na něj vytvořený odkaz.



Obr. 3.27: Aliasy v adresářovém stromu

Aliasy porušují přísně hierarchickou strukturu adresářového stromu, ovšem bez jejich použití nelze postihnout zmíněné vazby *m:n*. Pro vytvoření aliasu je třeba vytvořit záznam z objektové třídy `alias` a jako atribut použít `aliasObjectName`, ve kterém se uvede *DN* záznamu, na který se alias odkazuje.



Obr. 3.28: Aliasy mezi adresářovými servery

Aliasy lze použít i pro odkazy na záznamy uložené na jiných serverech. Bohužel, jejich použití je v tomto ohledu velmi pomalé a neefektivní, proto se doporučuje použít místo nich mechanismu referencí, které jsou popsány v [UDLDAPDS].

3.4 Funkční model

Funkční model LDAP obsahuje tři kategorie operací, které byly ve stručnosti zmíněny v kapitole 3.1.1.

3.4.1 Dotazovací operace LDAP

Dotazovací operace `search` slouží pro vyhledávání záznamů v adresáři a operace `compare` pro testování, zda záznamy obsahují atributy s určitými hodnotami.

Operace `search`

slouží pro prohledávání adresářového stromu a jako svůj výsledek vrací jednotlivé záznamy. Při jejich prohledávání je možné získat úplné záznamy nebo pouze hodnoty některých atributů. Výsledek prohledávání rovněž záleží na přístupových právech, kdy například uživatelské heslo uložené v atributu `userPassword` je přístupné[5] pouze jeho vlastníkovi nebo správci celého *DIT*.

Operace `search` má 8 vstupních parametrů:

Výchozí bod hledání (*base object*) – V doslovném překladu *základní objekt* (v tomto kontextu jsou pojmy objekt a záznam ekvivalentní). Určuje uzel stromu pomocí jeho DN, od kterého se má začít s prohledáváním záznamů. Tento parametr je výhodné použít zejména pro optimalizaci vyhledávání záznamů, předpokladem je ovšem znalost místa, kde je třeba hledat. Příkladem může být obr. 3.29, ve kterém se místo prohledávání celého adresářového stromu prohledává pouze větev od uzlu `ou=Clenove, dc=sin, dc=cvut, dc=cz`.

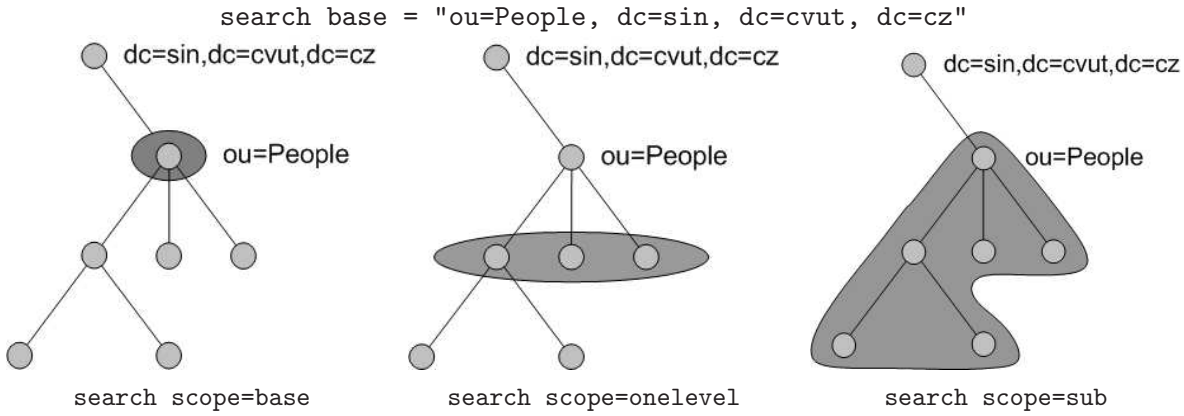
```
ldapsearch -b "ou=Clenove,dc=sin,dc=cvut,dc=cz"
```

Obr. 3.29: Vyhledání v konkrétní větvi

Oblast prohledávání (*search scope*) – Tento parametr určuje, do jaké hloubky se má při prohledávání adresáře jít. K dispozici jsou tři parametry: `base`, `onelevel` a `sub`. Na obr. 3.30 jsou vidět oblasti prohledávání při použití příslušných parametrů.

- `base` – při použití tohoto parametru vrací operace `search` pouze záznam výchozího bodu hledání
- `onelevel` – při použití tohoto parametru vrací operace `search` všechny záznamy první úrovně od výchozího bodu hledání
- `sub` – při použití tohoto parametru vrací operace `search` všechny záznamy od výchozího bodu hledání (včetně)

[5] Čitelné je pouze v případě, že je uloženo ve formátu čistého textu



Obr. 3.30: Prohledávání adresářového stromu

```
ldapsearch -b "ou=Clenove,dc=sin,dc=cvut,dc=cz" -s onelevel
```

Obr. 3.31: Vyhledávání v konkrétní větvi do první úrovně

Dereference zástupců (*alias dereferencing options*) – Tento třetí parametr určuje, jakým způsobem se bude při prohledávání nakládat s aliasy na jiné objekty.

`neverDerefAliases`

`derefInSearching`

`derefFindingBaseObject`

`derefAlways`

Limit počtu navrácených záznamů (*size limit*) – Udává maximální počet záznamů, které je klient ochoten akceptovat. Standardně bývá nastaveno cca 100 záznamů. Při překročení limitu je vrácen návratový kód `LDAP_SIZELIMIT_EXCEEDED`, která informuje o jeho překročení a který musí příslušná aplikace nějakým způsobem zpracovat. Limit nastavený na hodnotu 0 informuje, že není žádný limit nastaven a server může zaslat veškeré informace, které nalezl. Toto lze ovšem nastavit i na straně serveru a neomezený počet nalezených záznamů lze povolit jen privilegovaným uživatelům.

Časový limit (*time limit*) – Nastavuje hodnoty pro časový limit v sekundách, po který je klient ochoten přijímat od LDAP serveru záznamy. Při překročení tohoto limitu je navrácen návratový kód `LDAP_TIMELIMIT_EXCEEDED`.

Pouze atributy (*attributes-only*) – `attrsOnly` je logický parametr (`boolean`), který udává, májí-li se vracet ve výsledku záznamu hodnoty atributů. Při nastavení atributu na hodnotu `true` se vrací pouze seznam atributů, které jsou v záznamu použity, při nastavení na hodnotu `false` se vrátí atributy i jejich hodnoty.

Vyhledávací filtry (*search filter*) – Slouží pro vyjádření hledaných výrazů v adresářovém stromu. Filtry jsou specifikovány v [RFC 2254] a jejich přehled udává tab. 3.4, strana 32.

Typ filtru	**Formát**	**Příklad**	**Odpovídá**
Shoda (*Equality*)	`(atribut=hodnota)`	`(sn=Novák)`	Příjmení přesně odpovídá `Novák`
Podřetězec (*Substring*)	`(atribut=[první]*nějaký*[koncový])`	`(sn=*ová*)`	Příjmení obsahuje "`ová`"
		`(sn=Nová*)`	Příjmení začíná na "`Nová`"
		`(sn=*áček)`	Příjmení končí na "`áček`"
		`(sn=N*v*k)`	Příjmení začíná na "`N`", obsahuje "`v`" a končí na "`k`"
Přibližně (*Approximate*)	`(atribut~=hodnota)`	`cn~=tatar`	Název počítače odpovídá názvu `tatar`, ale může odpovídat i názvu `tartar`
Větší nebo shodné	`(atribut>=hodnota)`	`(floor>=9)`	Lidé, kteří bydlí na devátém nebo vyšším patře
Menší nebo shodné	`(atribut<=hodnota)`	`(floor<9)`	Lidé, kteří bydlí pod devátým patrem
Přítomnost (*Presence*)	`(atribut=*)`	`(sn=*)`	Všechna příjmení
AND	`(&(filtr1)(filtr2)...)`	`(&(sn=Novák)(givenName=Jiří))`	
OR	`(|(filtr1)(filtr2)...)`	`(|(givenName=Jiří)(givenName=Josef))`	
NOT	`(!(filtr))`	`(!(mail=*))`	Všechny záznamy mimo atributu `mail`

Tab. 3.4: Vyhledávací filtry LDAP

Jednotlivé filtry (tab. 3.4, strana 32.) lze mezi sebou vhodně kombinovat, jak je vidět např. na obr. 3.32, strana 32., a tím pokládat fundovanější dotazy.

```
$ ldapsearch (&(|(givenName=Josef)(givenName=Jiří))(sn=Novák))
```

Obr. 3.32: Kombinace vyhledávacích filtrů

Výsledkem tohoto dotazu jsou záznamy všech Josefů nebo Jiří Nováků, které operace `search` v adresáři nalezla. Bohužel není možné klást komplikované dotazy, které by prohledávaly a spojovaly jednotlivé záznamy dohromady, případně prováděly nad těmito daty některé operace (`SUM`, `AVG` apod.), jako je tomu v případě jazyka *SQL*.

Jako atributů pro vyhledávání lze použít nejen atributy obsažené v záznamu, ale i speciální atributy (`rootDSE`), pomocí kterých předává *LDAP* server informace např. o podporovaných autentizačních mechanismech apod.

Seznam atributů pro navrácení – V tomto posledním parametru lze vyjmenovat atributy, které jsou požadovány. Např. při vytváření statických `arp` záznamů jsou důležité pouze údaje o IP adrese (atribut `ipHostNumber`) a HW adrese síťové karty (atribut `macAddress`).

```
ldapsearch -b "ou=Pocitace,dc=sin,dc=cvut,dc=cz" \
> ipHostNumber macAddress
```

Obr. 3.33: Výpis konkrétních atributů

Operace `compare`

slouží pro testování hodnoty atributu s hodnotami uloženými v záznamu. Jako vstupní parametry je použito *DN* záznamu na kterém se hodnoty atributů testují a seznam atributů a jejich hodnot[6]. Funkce pak vrací v případě úspěchu hodnotu `TRUE`, v případě neúspěchu hodnotu `FALSE`. Příkladem použití může být např. porovnání otisku digitálního certifikátu s hodnotou uvedenou v adresářovém serveru, nebo jednoduchý příklad na obr. 3.34, strana 33.

```
ldapcompare "employeeNumber=88592,ou=Clenove,dc=sin,dc=cvut,dc=cz" \
> givenName:Karel
TRUE
```

Obr. 3.34: Operace `compare`

Příkaz `compare` je v protokolu *LDAP* používán většinou z důvodu vazby na protokol *X.500*, ale přesto má své opodstatnění a nelze jej plně nahradit operací `search`. Příkaz `search` vrátí všechny nalezené atributy a v případě hledání jednoho jediného atributu je třeba přenést několik bytů dat, ve kterých bude obsaženo *DN* hledaného záznamu, seznam atributů a jejich

[6] Vyjádřené ve formátu prostého textu nebo zašifrované pomocí `base64`

hodnot. Příkaz `compare` přenese jako svůj výsledek pouze logickou hodnotu `true` nebo `false` návratového kódu výsledku operace.

3.4.2 Aktualizační operace LDAP

LDAP má čtyři aktualizační operace: `add` (přidání), `delete` (smazání), `rename` (změna DN) a `modify` (změna údajů). Tyto operace se doporučuje provádět pomocí *LDIF* souborů (kapitola 3.6, strana 39) případně vhodného editoru. Provádění změn pomoci programů `ldapadd`, `ldapdelete`, `ldapmodify` a `ldapmodrdn` s přímým zápisem atributů, jejich hodnot a požadovaných změn se nedoporučuje. *LDIF* soubor je možné si po sobě několikrát zkontrolovat, než se provede nějaká změna, případně lze vystopovat, jaká změna měla být provedena. Při přímém použití je vysoká pravděpodobnost, že se nepodaří tuto změnu dohledat.

Operace `add`

přidá záznam do adresářového stromu. Má dva parametry: DN záznamu a množinu atributů a jejich hodnot. Operace přidání se povede pouze za předpokladu splnění následujících předpokladů:

- Rodič záznamu musí již v adresářovém stromu existovat
- V této části adresáře nesmí již stejný záznam existovat
- Záznam musí odpovídat a splňovat požadavky schémat, ze kterých se skládá
- Přístupová práva musí tuto operaci umožnit

V případě splnění všech těchto požadavků bude nový záznam do *DIT* přidán.

Operace `delete`

smaže záznam z adresáře. Má jediný parametr, *DN* záznamu který se má smazat. Operace smazání se povede pouze v případě, že jsou splněny následující předpoklady:

- Záznam pro smazání musí existovat
- Záznam nesmí mít žádného potomka, ty musejí být smazány před smazáním vlastního záznamu
- Přístupová práva musí tuto operaci umožnit

V případě splnění všech těchto požadavků bude záznam z DIT smazán.

Operace `rename`

slouží k přejmenování *RDN* a přesunu záznamu mezi větvemi adresářového stromu. Operace přejmenování se povede pouze v případě, že jsou splněny následující předpoklady:

- Záznam pro smazání musí existovat
- Nové *DN* záznamu nesmí být shodné s již existujícím *DN* jiného záznamu
- Přístupová práva musí tuto operaci umožnit

Operace `modify`

slouží ke změně hodnot atributů v záznamech. Operace modifikace se povede pouze v případě, že jsou splněny následující předpoklady:

- Záznam pro modifikaci musí existovat
- Všechny modifikace atributů musí skončit úspěchem
- Modifikace musí splňovat požadavky, které na ně schemata kladou (povinné atributy, správné datové typy apod.)
- Přístupová práva musí tuto operaci umožnit

3.4.3 Autentizační a řídící operace

LDAP má dvě autentizační operace (`bind` a `unbind`) a jednu řídící operaci (`abandon`).

Operace `bind`

Slouží pro ověření identity uživatele (autentizace), který k adresářové službě přistupuje. Po provedení autentizace se provede autorizace a uživateli je dovoleno pracovat s daty v závislosti na jeho přístupových právech. Více o autentizaci v kapitole 3.5.1 Autentizace, strana 36.

Operace `unbind`

oznamuje serveru, že klient s ním hodlá ukončit komunikaci.

Operace `abandon`

adresářovému serveru oznámí, že klient chce ukončit komunikaci. Využije se např. v situaci, kdy server vyhledává velké množství záznamů v rozsáhlém stromu a uživatel chce toto vyhledávání náhle přerušit. Jako vstupní parametr používá číselný identifikátor operace, kterou má *LDAP* server přerušit.

3.5 Bezpečnostní model

Bezpečnostní model zajišťuje zabezpečení přístupu k záznamům uložených v adresářovém serveru podle určitých, předem stanovených pravidel před nežádoucími osobami. Skládá se ze tří částí:

Autentizace

Přístupová práva

TLS/SSL

3.5.1 Autentizace

Autentizace slouží k ověření identity uživatele. Server tak má možnost pomocí některého z autentizačních mechanismů ověřit, zda osoba, která se vydává za konkrétního uživatele, splňuje předepsané autentizační podmínky. Ve většině případů se tak děje na základě znalosti přihlašovacího jména a hesla uživatele, digitálního certifikátu nebo u velmi složitých systémů např. podle biometrických znaků (duhovka oka, otisk prstu apod).

Autentizace může být buď jednocestná, nebo dvojcestná. Při jednocestné autentizaci klient zasílá heslo adresářovému serveru při tzv. `bind` operaci, nebo když adresářový server zasílá svůj certifikát klientu během navazováni chráněného spojení a tím se ověřuje přímo u adresářového serveru pomocí svého *X.509* certifikátu. Příkladem dvojcestné autentizace je situace, kdy si adresářový klient a server vyměňují během vyjednávání *TSL* spojení své certifikáty a teprve poté dojde k ověření uživatele.

Anonymní autentizace při operaci `bind` se serveru nezasílají žádné identifikační údaje o uživateli a jeho přístupových právech. Při používání anonymní autetizace se doporučuje nastavit přístupová práva pouze k vybraným informacím.

Jednoduchá autentizace – po nechráněném spojení protokolem *LDAP* se při operaci `bind` zašle identifikace uživatele pomocí jeho *DN* a hesla. Heslo by mělo být na straně serveru uloženo v zašifrované nebo hashované formě pomocí algoritmů `crypto`[7], `MD5` nebo `SHA1`.

```
ldapsearch -h ldap -b "dc=sin,dc=cvut,dc=cz" \
> -D "uid=uzivatel,ou=lide,dc=sin,dc=cvut,dc=cz" -w mojeheslo
```

Obr. 3.35: Jednoduchá autentizace

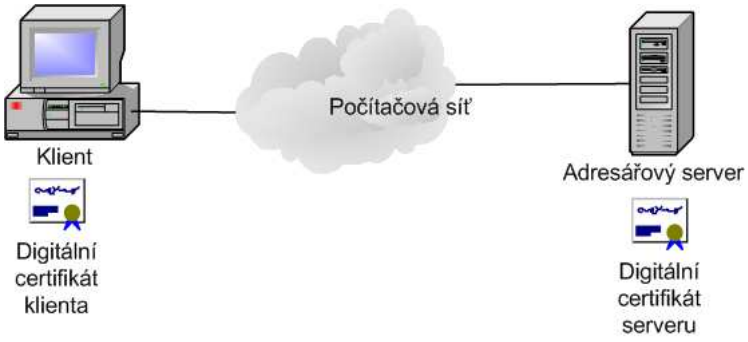
Jednoduchá autentizace přes bezpečný kanál – použit stejný princip jako u jednoduché autentizace uživatele pomocí jména a hesla, ale v tomto případě je při spojení a zasílání dat použit bezpečný kanál *TLS/SSL*. Jedná se o dvojcestnou autentizaci, zmíněnou v úvodu. Před vlastní autentizací uživatele si klient a server vymění informace o svých *X.509* certifikátech a navážou spojení přes bezpečný *TLS/SSL* kanál. Po jeho navázání pak dochází k operaci `bind`, ve které je zasláno *DN* uživatele, který se autentizuje, a jeho heslo.

Proxy autentizace – při tomto druhu autentizace obsaženém v *LDAPv3* je využito toho, že jistý uživatel má možnost nahlížet na hesla ostatních uživatelů, a tak může ověřit, zda uživatel, který se přihlašuje, zná skutečně své uživatelské jméno a heslo a má příslušná práva. K serveru se tedy "`binduje`" proxy uživatel, který provede ověření uživatele např. pomocí operace `compare`.

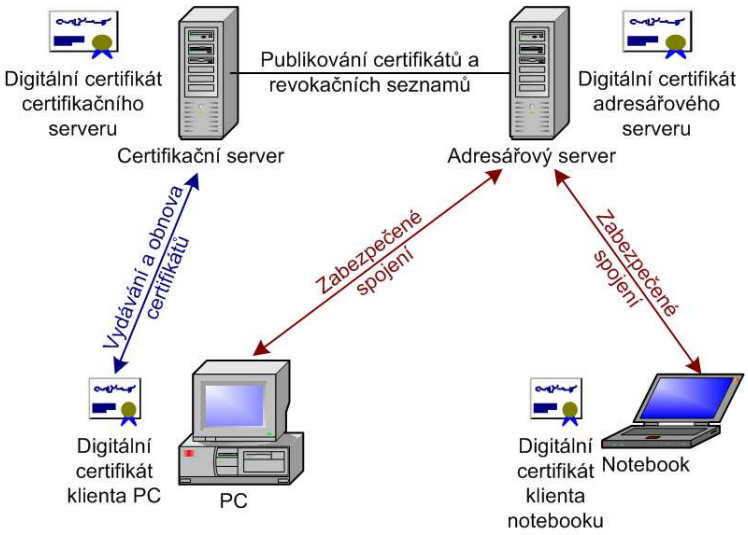
PKI autentizace je založena na principu digitálních *PKI* certifikátů [X.509] (viz. [Dostalek]). Tento druh autentizace ovládají jen některé *LDAP* servery a klienti. Certifikát musí odpovídat struktuře adresářového stromu a je uložen v atributu `userCertificate`. Klient si při spojení vybere konkrétní certifikát ve své klíčence, kterým se chce vůči serveru ověřit a při jeho použití je nucen použít heslo. Server zkontroluje, souhlasí-li uživatelský

[7] Nedoporučuje se používat

certifikát s certifikátem uloženým v serveru. Pokud ano, uživatel je ověřen. Tento způsob je náročný jak pro uživatele, tak pro administrátora, protože musí své digitální certifikáty stále hlídat a v případě potřeby aktualizovat. *LDAP* server může být propojen



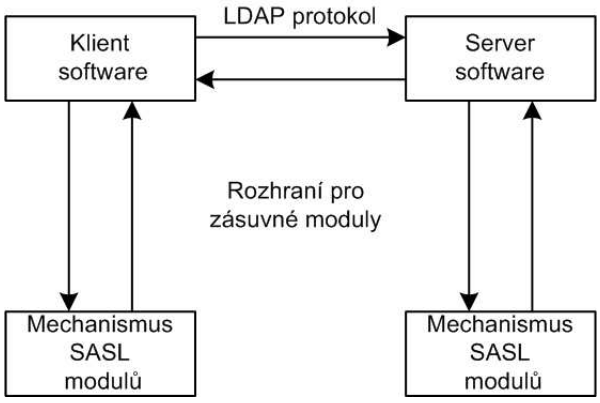
Obr. 3.36: PKI autentizace



Obr. 3.37: Systém PKI

s *OCSP* (*On-Line Certificate Status Protocol*) serverem, zajišťujícím on-line aktualizace informací o digitálních certifikátech.

SASL mechanismus (*Simple Authentication and Security Layer*) [RFC 2222] umožňuje použití svých modulů (`PLAIN`, `LOGIN`, `GSSAPI`, `NTLM`, `OTP`, `CRAM-MD5`, `DIGEST-MD5`, `SRP`, případně další moduly) pro autentizaci uživatele. Autentizační mechanismy `PLAIN` a `LOGIN` neobsahují žádné bezpečnostní mechanismy, proto je při přihlášení k *LDAP* serveru vhodné doplnit přenos jména a hesla vrstvou *TLS*. Pokud je nutné použít nezabezpečenou variantu přenosu `dn` uživatele a jeho hesla, doporučuje se použití mechanismu `DIGEST-MD5`. Praktickým příkladem použití *SASL* a *LDAP* je např. ověření existence uživatele *SMTP* serveru pro přeposílání e-mailu uživatelů zdržujících se mimo vnitřní síť pomocí mechanismu *SMTP-AUTH*.



Obr. 3.38: SASL mechanismus

3.5.2 Přístupová práva

Přístupová práva v adresářových službách mohou být na rozdíl od relačních databází nastavena až do úrovně záznamů. Bohužel zatím neexistuje oficiální standard který by přesně definoval model těchto práv, resp. existují pouze ve stadiu *RFC* draftů[8], proto se jejich implementace bude produkt od produktu lišit.

Ve většině produktů jsou k dispozici tzv. *Access Control Lists*, zkratka *ACL*, které specifikují, jakým způsobem je možné k záznamům a především atributům přistupovat.

Netscape Directory server a jeho nástupce [JES] používají jako doplněk k ACL tzv. *Access Control Information*, zkratka *ACI*, které jejich možnosti dále rozšiřují na omezení typu: odkud daný dotaz vzešel, jakým protokolem vzešel apod. ACI začíná v experimentální podobě používat i oblíbený projekt [OpenLDAP] a nejspíš se v dohledné době stane součástí některého ze standardů *RFC*.

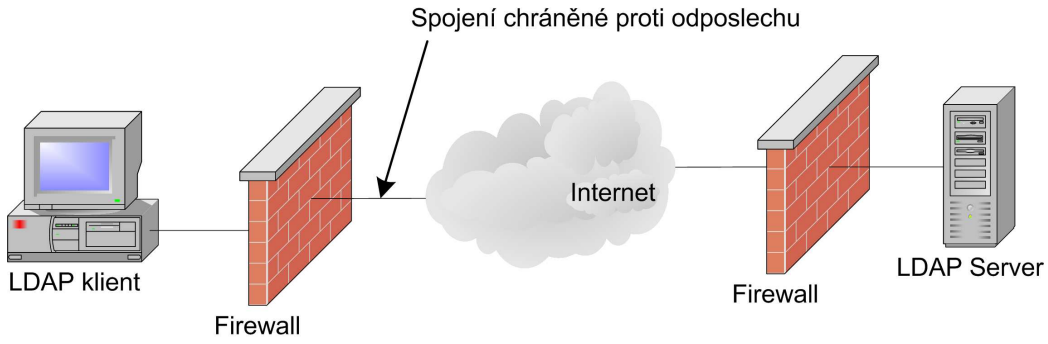
3.5.3 SSL/TLS

V bezpečnostním modelu se pro ochranu komunikace proti nežádoucímu odposlechu využívá možnosti vrstev *TLS* (*Transport Security Layer*) nebo *SSL* (*Socket Security Layer* – starší protokol). Standardní protokol *LDAP* využívá port 389, zatímco protokol *LDAPS* chráněný vrstvou *SSL* využívá port 636. Bezpečnostní model využívající protokol *TLS* podle [RFC 2830] má však ještě lepší možnosti především v tom, že využívá standardní *LDAP* port 389 a server s klientem se při navazování spojení teprve dohodnou, budou-li spojení chránit, či nikoliv. Pokud se dohodnou na chráněném spojení, použijí k tomu volání funkce `StartTLS`.

Zatímco při nechráněném přenosu dat protokolem *LDAP* může případný útočník odposlechnout komunikaci mezi serverem a klientem a získat tak velmi citlivé údaje např. o přihlašovacím *DN* a jeho hesle pro operaci `bind`, při komunikaci přes protokol chráněný vrstvou *TLS* nebo *SSL* je přenos dat zabezpečený proti odposlechu, resp. proti rozkódování této komunikace. K tomu by musel mít útočník privátní klíč klienta nebo serveru. Při této komunikaci je nutné, aby se server, případně i klient, prokazovaly platným [X.509] certifikátem podepsaným některou z certifikačních autorit.

[8] draft-ietf-ldapext-acl-model, draft-legg-ldap-acm-admin, draft-legg-ldap-acm-bac

Chráněné spojení je vhodné především při přenosu dat mezi klientem a serverem, mezi kterými je nedůvěryhodné prostředí, např. Internet. Příklad použití zobrazuje obr. 3.39, strana 39.



Obr. 3.39: Bezpečné spojení přes nedůvěryhodné prostředí

Klienti, kteří se k serveru připojují přes *TLS* vrstvu, mají následující výhody:

- Komunikace je chráněna proti odposlechu
- Komunikace je chráněna proti útoku typu "muž uprostřed"
- Klient se může vůči serveru autentizovat použitím *PKI*
- Může si ověřit platnost certifikátu, který server poskytuje

Bezpečné a autentizované spojení mezi klientem a serverem může vypadat následovně:

- Otevření *TCP/IP* spojení se serverem
- Operace `StartTLS`
- Operace `bind` za použití externího mechanismu SASL, případně jiného *SASL* mechanismu (`DIGEST-MD5`)

3.6 LDIF

LDAP Data Interchange Format je standardní textový formát pro popisování adresářových záznamů, definovaný v [RFC 2849]. *LDIF* umožňuje snadný import a export mezi různými adresářovými servery, které používají rozdílné interní databázové formáty. Textové formáty jsou mezi systémovými administrátory velmi oblíbené především pro svou přehlednost a snadnou editovatelnost.

Za normálních okolností je *LDIF* soubor psán v kódové stránce *ASCII*, pokud je ale třeba použít diakritiku, používá se pro ni standardní kódová stránka *UTF-8*. Data, která nelze vyjádřit jako *ASCII* nebo *UTF-8* jsou kódována pomocí `base64`. Tímto způsobem lze v *LDIF* formátu uložit např. fotografii, digitální certifikát apod.

V poslední době dochází k nástupu různých textových formátů založených na technologii *XML*. Adresářové služby se tomuto trendu nevyhýbají, proto byl vytvořen formát *DSML*

```
dn: employeeNumber=95882,ou=Clenove,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: kskPerson
cn: Karel Benák
sn: Benák
givenName: Karel
employeeNumber: 95882
employeeType: admin
```

Obr. 3.40: Typický LDIF soubor

(*Directory Services Markup Language*), v aktuální verzi *DSMLv2*. Tento protokol je schopen přenosu pomocí protokolu *HTTP* a následného zpracování pomocí protokolů *HTTP/SOAP*. Tato vlastnost může napomoci např. při zabezpečení dat při přenosu v počítačové síti. Ačkoliv i protokol *LDAP* má svou zabezpečenou variantu (protokol *LDAPS* nebo vrstvu *TLS*), je možné použít pro chráněný přenos protokolu `https`.

Z obrázku obr. 3.41, strana 41. je patrné, že v moderních aplikacích (postavených převážně na www technologiích a standardech) programovaných jazyky `Java` nebo `C#` má *DSMLv2* široké uplatnění. Formáty založené na *XML* jsou pro současné aplikace snáze zpracovatelné a čitelnější, narozdíl od klasických textových nebo binárních formátů, jakým je již zmiňovaný *LDIF*. Při programování aplikací založených na *XML* se ve velké míře využívá již hotových řešení a aplikace sama pak nemusí umět pracovat s *LDAP* protokolem a přesto může klást dotazy a získávat odpovědi pomocí protokolu *HTTP/SOAP*.

```
<dsml:entry dn="employeeNumber=95882,ou=Clenove,dc=sin,dc=cvut,dc=cz">
     <dsml:objectclass>
          <dsml:oc-value>top</dsml:oc-value>
          <dsml:oc-value>person</dsml:oc-value>
          <dsml:oc-value>organizationalPerson</dsml:oc-value>
          <dsml:oc-value>inetOrgPerson</dsml:oc-value>
          <dsml:oc-value>kskPerson</dsml:oc-value>
     </dsml:objectclass>
     <dsml:attr name="cn">
          <dsml:value>Karel Benák</dsml:value>
     </dsml:attr>
     <dsml:attr name="sn">
          <dsml:value>Benák</dsml:value>
     </dsml:attr>
     <dsml:attr name="givenName">
          <dsml:value>Karel</dsml:value>
     </dsml:attr>
     <dsml:attr name="employeeNumber">
          <dsml:value>95882</dsml:value>
     </dsml:attr>
     <dsml:attr name="employeeType">
          <dsml:value>admin</dsml:value>
     </dsml:attr>
</dsml:entry>
```

Obr. 3.41: Záznam ve formátu DSMLv2

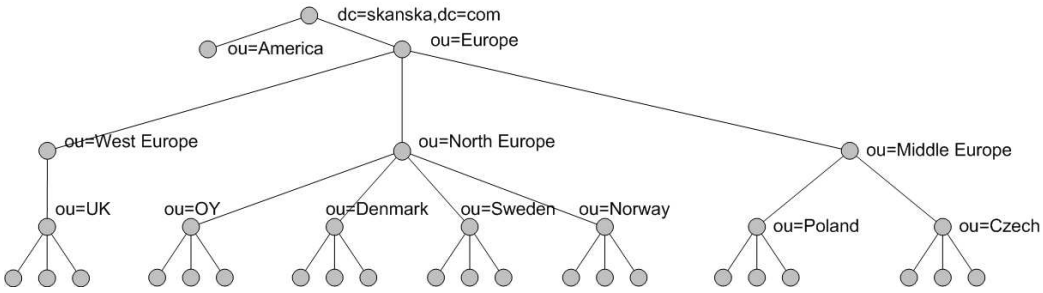
3.7 Topologie

Topologie se zabývá především rozdělením adresářového stromu mezi více serverů. Nejčasteji se navrhuje podle geografického rozmístění poboček, serverů a klientů. Ve své podstatě kopíruje fyzickou topologii sítě a měla by velmi úzce souviset s návrhem jmenných prostorů a replikací.

Jedním z nejčastějších důvodů rozdělení DITu na několik částí je především potřeba spravovat a ukládat konkrétní část stromu na konkrétní pobočce, zatímco ostatní pobočky lokálního administrátora a lokální uživatele nezajímají. Pokud by např. Skanska CZ byla zapojena do adresářového stromu své mateřské společnosti ve Švédsku, nezajímalo by správce české pobočky nastavení norské větve. Zároveň by šetřil místo na disku (discích) pro lokální záznamy, které pobočka potřebuje ke svému běhu, zatímco norské záznamy by v české pobočce zbytečně zabíraly volné místo.

Správný návrh musí respektovat především potřebu dostupnosti dat pro každého klienta. Topologie sítě a adresářových služeb ji musí zajistit bez jakýchkoliv problémů ve stanoveném čase. Je samozřejmě výhodné použít replikací, ty ovšem neřeší problémy spjaté se správou a přenosem dat především po pomalých linkách typu WAN. Pokud se např. klient v pražské pobočce musí autentizovat na stockholmské centrále, může výsledek trvat nepřiměřeně dlouho a ostatní služby (např. Kerberos) mohou klienta označit jako nedůvěryhodného a znemožnit mu práci. Rovněž jakékoliv čekání na odezvu serverů a aplikací je nepříjemné a pro běžného uživatele neakceptovatelné. Navíc tím stoupá provoz po linkách poskytovatele, který si může účtovat cenu za počet přenesených dat nebo za dobu připojení.

Pokud by například Skanska používala adresářové služby ve všech svých pobočkách a měla je zorganizovány v navazujících stromech, mohl by při respektování geografického rozmístění poboček vypadat DIT jako např. na obrázku 3.42. Obrázek samozřejmě nepostihuje všechny mezinárodní pobočky Skansky, pouze se zabývá jejími evropskými pobočkami. Pro ilustraci je ale více než dostačující.



Obr. 3.42: Topologie Skansky podle poboček

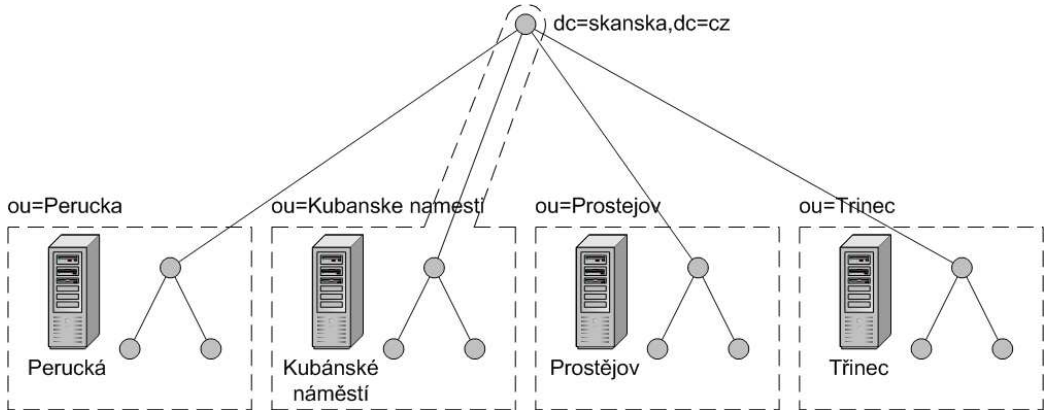
V požadavku by mohlo být řečeno, že kořen DITu začíná v centrále ve Švédsku, kde je rovněž umístěna příslušná větev. Dále se v centrále spravuje celá evropská větev a příslušné větve rozdělující Evropu na regiony West, North a Middle. Větev America a k ní příslušející větve North America a Latin America se spravují v pobočce ve Spojených státech, Větev Asia se spravuje v pobočce v Hong Kongu.

Adresářový strom by měl tedy svůj kořen v `dc=skanska,dc=com`, který je umístěný v centrále ve Švédsku. To znaméná, že prefix `dc=skanska,dc=com` a větev `ou=Sweden,ou=North Europe,ou=Europe,dc=skanska,dc=com` jsou umístěny na centrálním serveru. Na něm jsou rovněž umístěny větve :

```
ou=America,dc=skanska,dc=com

ou=Asia,dc=skanska,dc=com

ou=Europe,dc=skanska,dc=com

    ou=West Europe,ou=Europe,dc=skanska,dc=com
        ou=UK,ou=West Europe,ou=Europe,dc=skanska,dc=com
    ou=North Europe,ou=Europe,dc=skanska,dc=com
        ou=OY,ou=West Europe,ou=Europe,dc=skanska,dc=com
        ou=Denmark,ou=West Europe,ou=Europe,dc=skanska,dc=com
        ou=Norway,ou=West Europe,ou=Europe,dc=skanska,dc=com
    ou=Middle Europe,ou=Europe,dc=skanska,dc=com
        ou=Czech,ou=West Europe,ou=Europe,dc=skanska,dc=com
        ou=Poland,ou=West Europe,ou=Europe,dc=skanska,dc=com
```

Z těchto větví pak vedou odkazy pomocí referencí na příslušné pobočkové servery v jednotlivých zemích.

Jelikož nemá společnost Skanska podobné řešení, mohla by Skanska CZ používat rozdělení svého DIT na adresářové servery podle obrázku 3.43. Skanska CZ však místo tohoto řešení



Obr. 3.43: Topologie Skansky podle místa pracoviště

používá replikací, které jsou pro zajištění provozu dostačující.

Kapitola 4

Software pro adresářové služby

Ve světě otevřeného software je jen málo významných projektů, které jsou schopny s adresářovými službami pracovat. Naopak velké softwarové firmy se snaží tuto technologii ve svých produktech prosazovat, příkladem může být Microsoft Active Directory a jeho výtečná provázanost se systémem, nebo Java Enterprise System, který v sobě integruje.

4.1 Adresářové servery

Adresářových serverů je k dispozici celá řada. V rámci OpenSource software jsou k dispozici servery OpenLDAP (4.1.1) a TinyLDAP (4.1.2).

Pro podnikovou sféru, kde záleží na 100 % přístupnosti k datům nutné podpoře ze strany dodavatele, lze zvolit některý z komerčních produktů. U adresářových serverů by se neměl nakupovat pouze vlastní adresářový server, ale celé řešení, které by mělo obsahovat i další komponenty, např. *PKI*, návaznost na operační systém a jeho služby, identity server ad. Podle mých zkušeností tyto požadavky splňují produkty Java Enterprise System Directory server (strana 45.) od Sun Microsystems, Active Directory (strana 45.) od Microsoft Corporation obsažený např. v MS Windows Server 2003, Tivoli od IBM a Novell E-Directory od Novell.

4.1.1 OpenLDAP

OpenLDAP je vyvíjený pro různé operační systémy, jakými jsou např. Linux, FreeBSD, Solaris apod. Není nijak hardwarově náročný. Ve standardní instalaci podporuje *SSL/TLS*, *Kerberos* a mechanismy *SASL*, obsažené v *LDAPv3*. Podporu *TLS* je možné dodat pomocí knihoven OpenSSL [OpenSSL] nebo GNU TLS [GNU TLS], podporu protokolu *Kerberos* je možné získat z knihoven projektu [MIT Kerberos] nebo [Heimdal] a protokol *SASL* z projektu [Cyrus SASL].

Pokud nejsou na adresářový server kladeny velké nároky na rychlost a dodavatel systému nebo aplikace je schopen sestavit a poskytovat pro OpenLDAP podporu, je [OpenLDAP] vhodným řešením. Sami vývojáři OpenLDAPu připouštějí, že jejich adresářový server není tak rychlý, jako ostatní, především komerční implementace. Paradoxně bývá OpenLDAP poměrně často nasazován. Příkladem může být Fakulta stavební ČVUT v Praze a její systém ELDAP.

4.1.2 TinyLDAP

TinyLDAP je malý ale velmi rychlý adresářový server, který data ukládá do klasických textových souborů. Z protokolu LDAP ovládá pouze nejzákladnější části a např. replikace nemá vůbec vyřešeny.

4.1.3 Sun Java Enterprise Directory Server

Původě se jedná o Netscape Directory Server, posléze převzatý společností iPlanet a nyní je součástí projektu Java Enterprise System, produkovaného společností Sun Microsystems. Implementuje pokročilé možnosti *LDAPv3* a je nasazen například pro autentizaci uživatelů pro přístup k bankovnímu systému `www.mojebanka.cz`. Tento adresářový systém lze za určitých podmínek vyzkoušet zcela zdarma.

4.1.4 Microsoft Active Directory

Active Directory (AD) je typickým příkladem *NOS*. Není to typický *LDAP* server, ačkoliv je schopen pomocí tohoto protokolu komunikovat. V AD je integrován *LDAP*, *Kerberos*, *SMB/CIF* server a další části. AD lze v síti Microsoft Windows integrovat s *DHCP* serverem, *DNS* serverem, tiskovým serverem a dalšími službami. Výbornou vlastností tohoto produktu je úzká vazba na *PKI*, takže se uživatel může velmi snadno prokazovat svými digitálními certifikáty. Provázanost jednotlivých komponent Windows Serveru 2003 s AD je podle mého názoru velkým trumfem proti ostatním serverovým systémům, nehledě na produkty vytvářené v rámci Open Source software.

4.2 Prohlížeče a editory

GQ

GQ je podle mého názoru jedním z nejlepších klientů pro prohlížení a editaci záznamů v *LDAP* adreáři, který je pod operačními systémy typu UNIX k dispozici. Jako grafické prostředí používá knihovnu Gtk a jeho lokalizace do češtiny je poměrně zdařilá. Poskytuje velmi široké možnosti editace a vkládání dat, stejně tak i jejich prohledávání a prohlížení. K zajímavým vlastnostem patří i možnost prohlížení schémat, atributů a datových typů. Klient umožňuje velmi dobře vkládat i binární záznamy (obrázky, zvuk) a digitální certifikáty uživatelů a serverů.


GQ
File Filters Help
Search Browse Schema
ldap
dc=sin,dc=cvut,dc=cz
ou=People
ou=Rpc
ou=Group
ou=Networks
ou=Netgroup
ou=Hosts
ou=Services
ou=Aliases
ou=Protocols
ou=Clenove
employeeNumber=95799
employeeNumber=90222
employeeNumber=59903
employeeNumber=75933
employeeNumber=58013
employeeNumber=89828
employeeNumber=33149
employeeNumber=30888
employeeNumber=48807
employeeNumber=45144
employeeNumber=23727
employeeNumber=61667
employeeNumber=44643
dc=beny
dc=xbenakk,dc=cz
localhost
displayName
employeeNumber
employeeType
givenName
initials
jpegPhoto
mail
mobile
o
roomNumber
Karel Benák
95799
admin
Karel
KB
beny@sin.cvut.cz
karel.benak@sin.cvut.cz
+420 723 168 329
FSv
168
Použít
Add as new
Obnovit

Obr. 4.1: GQ

4.3 Vývojové prostředky

Pro vývoj aplikací spolupracujích s adresářovými službami existují různé vývojové prostředky (Software Developer kit - SDK) pro celou řadu programovacích jazyků. Lze se setkat například s API, které poskytují vývojáři projektu Mozilla, OpenLDAP, Java apod.

4.3.1 C/C++

Při instalaci OpenLDAP je možné zároveň instalovat i statické a dynamické knihovny (`libldap.a`, `libldap_r.a`, `liblber.a`, `libldap.so`, `libldap_r.so`, `liblber.so`) a hlavičkové soubory potřebné pro vývoj aplikací v programovacím jazyce C.

OpenLDAP obsahuje i knihovnu a hlavičkové soubory pro práci s programovacím jazykem C++. Potřebné zdrojové soubory se nacházejí v adresáři `contrib/ldapc++` a je nutné je přeložit zvlášť. Při standardním překladu se nepřekládají a nejsou ani součástí přeložených vývojových balíčků. Pro vlastní překlad je třeba standardní knihovny pro jazyk C a pro generování dokumentace k API je nutné použit program doxygen [Doxygen]. Bez vygenerované dokumentace je obtížné programovat a příslušné API je v ní velmi dobře popsáno.

4.3.2 Java

Programovací jazyk Java poskytuje knihovnu tříd JNDI (Java Naming and Directory Interface) určenou pro práci se jmennými a adresářovými službami. Při vývoji aplikací je třeba mít nainstalovanou SDK verzi Javy, pro běh aplikací postačí JRE verze Javy. K dispozici je rovněž přehledný a čitelný návod [6], jak JNDI knihovnou programovat.

Mezi dalšími je od firmy Novell k dispozici knihovna JLDAP (Java LDAP) [JLDAP] a od firmy OctetString knihovna JDBC-LDAP (JDBC-LDAP Bridge Driver) [JDBC LDAP].

4.3.3 Perl

Pro skriptovací jazyk Perl je možné využít modul perl-ldap [Perl-LDAP], který lze stáhnout z kteréhokoliv CPAN archivu. Spolu s modulem je přiložena i přehledná dokumentace, která je při instalaci instalována jako tzv. POD. perl-ldap vyžaduje před vlastní instalací poměrně mnoho doplňkových balíčků. Je proto dobré se přesvědčit, zda Perl obsahuje správné knihovny (např. pro práci se SSL, SASL, ASN1, Base64 apod.).

Všechny administrační skripty pro IS Sinkuleho koleje spolupracující s LDAP serverem jsou psány právě za použití perl-ldap.

4.3.4 Python

Pro skriptovací jazyk Python je možné použít Python-LDAP [Python-LDAP]. Jedná se o objektově orientované API poskytující přístup aplikacím napsaným v jazyce Python k adresářovým službám, které poskytuje možnosti využití protokolu LDAPv3.

4.3.5 PHP

Skriptovací jazyk PHP standardně neobsahuje podporu LDAP, je ale možné ji do tohoto jazyka přidat. Pro platformu windows je třeba povolit a doplnit v souboru `php.ini` cestu k příslušné sdílené knihovně (`ldap.dll`). Pro operační systémy typu UNIX je nutné při kompilaci použít sdílené knihovny `libldap.so`, `libldap_r.so`, `liblber.so` nebo knihovny statické

(`libldap.a`, `libldap_r.a`, `liblber.a`). Pokud se instaluje PHP a zvlášť rozšíření o LDAP pomocí předkompilovaných balíčků (např. RPM), je nutné mít nainstalovánu odpovídající verzi LDAP knihovny, se kterou bylo PHP původně kompilováno.

4.3.6 ADSI

ADSI je proprietární objektově orientované SDK od firmy Microsoft pro programovací jazyky Visual Basic, C a C++. Primárně je určený pro spolupráci s Microsoft Active Directory, ale je schopen pracovat i s jinými adresářovými servery.

Kapitola 5

Analýza a řešení části IS Klubu Sinkuleho koleje

Klub Sinkuleho koleje je počítačový klub studentů ČVUT spadající pod Studentskou Unii. Členem klubu se může stát každý ubytovaný na koleji V. Sinkule v Zikově ulici 13, Praze 6. Počítačový klub má přibližně 250 členů a jeho náplní je poskytování kvalitních služeb a moderních technologií pro přenos dat členů. V současné době je prováděn přechod na nový informační systém, jehož součástí budou i adresářové služby.

Mezi hlavní požadavky na nový informační systém patří:

- Evidence členů a jejich počítačů
- Evidence plateb členů
- Evidence majetku klubu a jeho zápůjček
- Spolupráce se standardními internetovými službami a protokoly

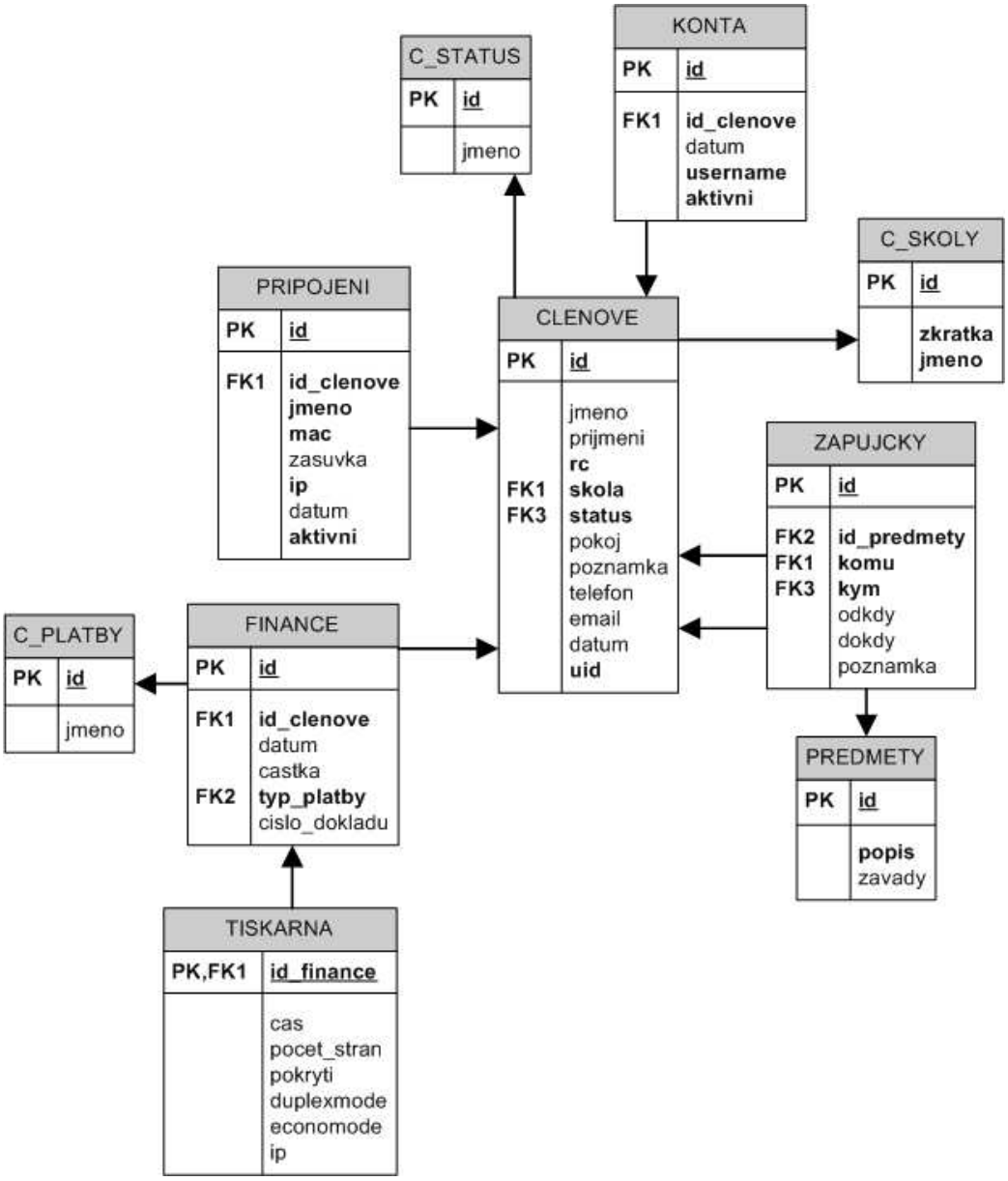
Základním zdrojem dat je relační databáze PostgreSQL, ve které jsou uloženy informace o uživatelích, jejich počítačích, platbách příspěvků a jiných poplatků, evidence majetku klubu a jeho zápůjček členům klubu. Analýza vlastní relační databáze nebude součástí této práce, bude v ní pouze část týkající se adresářových služeb a jejich návaznost na některé další části informačního systému. Adresářová služba bude použita především pro autentizaci uživatelů vůči některým službám, např. `smtp`, `pop3`, `ssh` apod.

5.1 SQL databáze

Návrh relační databáze na obr. 5.1, strana 50. není ještě zcela dokončen a je ve fázi připomínkování. Jeho tvůrci jsou studenty ČVUT Fakulty Elektrotechnické, Petr Macejko a Jiří Zamazal. Ti jej navrhli v rámci semestrální práce z předmětu Jazyk SQL.

5.2 Topologie

Pokud by KSk byl součástí SU ve které by se používalo adresářových služeb, měl by každý klub na svém centrálním serveru uloženu vlastní větev adresářového stromu a odkazy na další větve by se prováděly pomocí referencí. Tyto jednotlivé větve by mohly vypadat následovně:



Obr. 5.1: ER diagram databáze

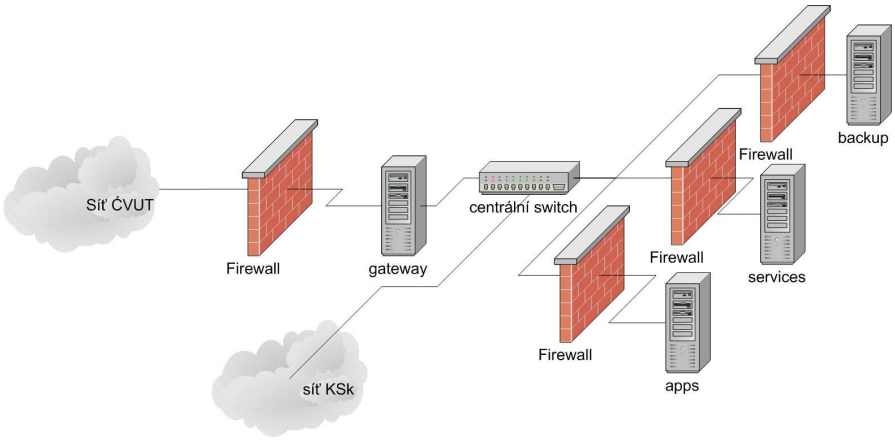
```
dc=su,dc=cvut,dc=cz
    ou=buk,dc=su,dc=cvut,dc=cz
    ou=dik,dc=su,dc=cvut,dc=cz
    ou=dk,dc=su,dc=cvut,dc=cz
    ou=krestani,dc=su,dc=cvut,dc=cz
    ou=mk,dc=su,dc=cvut,dc=cz
    ou=pod,dc=su,dc=cvut,dc=cz
        ou=blok A,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok B,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok C,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok D,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok E,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok F,ou=sh,dc=su,dc=cvut,dc=cz
    ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 1,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 2,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 3,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 4,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 5,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 6,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 7,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 8,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 9,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 10,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 11,ou=sh,dc=su,dc=cvut,dc=cz
        ou=blok 12,ou=sh,dc=su,dc=cvut,dc=cz
    ou=sin,dc=su,dc=cvut,dc=cz
```

Návrh vychází z hierarchického uspořádání Studentské unie a snaží se dodržovat [RFC 2247]. Při použití této struktury jmenných prostorů však není respektován tvar použitelný pro digitální certifikáty [X.509].

Vzhledem ke vztahům uvnitř Unie není snaha o využití adresářových služeb, proto si každý klub řeší informační systém podle svých návrhů, možností a schopností.

V současné situaci je vhodné navrhnout kořen DIT ve tvaru `dc=sin,dc=cvut,dc=cz` vycházející z doménové adresy klubu `sin.cvut.cz` a z již zmiňovaného [RFC 2247]. Vzhledem k jednoduché topologii zobrazené na obr. 5.2, strana 52. nemá cenu dělit DIT mezi více serverů.

Při návrhu topologie sítě bylo vycházeno z požadavků maximální snahy o zabezpečení sítě a jednotlivých serverů. Všechny protokoly, kterými spolu servery komunikují, jsou zabezpečenými variantami standardních protokolů. Jedná se především o prokotoly `ssh`, `https`, `ldaps`, `pop3s`, `imap4s` a `smtps`, rovněž přístup k relačnímu databázovému serveru PostgreSQL je šifrován. Pro tyto zabezpečené protokoly je nutné vygenerovat digitální certifikáty [X.509]



Obr. 5.2: Topologie sítě KSk

podepsané některou z certifikačních autorit, jinak budou klienti odmítat spojení s těmito zabezpečenými servery a zabezpečení tak nebude efektivní. Certifikáty budou nejspíš podepsány lokální certifikační autoritou a digitální certifikát této autority si do své klíčenky importují všichni klienti, kteří s těmito servery budou spolupracovat.

5.2.1 Rozdělení serverů

gateway – centrální směrovač který zajišťuje směrování sítě `147.32.110.0/23` do sítě ČVUT `147.32.252.12/30`. Plní funkci centrálního firewallu zabraňujícího některým útokům a zasílá ohlášení směrovače protokolu IPv6. Dalším zajímavým programem běžícím na tomto počítači je počítání trafficu jednotlivých počítačů a jeho případné omezení. Pro zabezpečení správných vazeb mezi HW adresou a IP adresou je použita statická ARP cache, která je generována z adresářové služby pomocí skriptu napsaného v jazyce Perl a pravidelně aktualizována každých 15 minut.

services – centrální informační a databázový server na kterém jsou provozovány všechny důležité aplikace. Mezi hlavní aplikace patří především databáze členů klubu uložená v relační databázi PostgreSQL. Z této databáze jsou exportovány hodnoty některých údajů o uživatelích a jejich počítačích a jsou importovány do `ldap` serveru. Ten slouží jako zdroj informací pro primární `DNS` server a `DHCP` server běžícím na tomto počítači. Rovněž jsou zde provozovány `http` servery Apache a Apache Tomcat jako aplikační server pro vlastní intranetovou aplikaci, která je napsána pomocí JSP a Java servrletů. Ty využívají technologie JDBC a JNDI pro komunikaci mezi relační databází a adresářovým serverem. Technologie Java byla zvolena pro snadnou portaci i na jiné platformy, než-li je Intel/Linux. Pro tento server musí být vygenerováno několik digitálních certifikátů [X.509] podepsaných lokální certifikační autoritou. Jsou to certifikáty pro domény `services.sin.cvut.cz`, `intranet.sin.cvut.cz`, `ldap.sin.cvut.cz`, `db.sin.cvut.cz` a `ns.sin.cvut.cz`.

apps – aplikační server pro provozování uživatelských aplikací. Přístup je povolen pouze po dohodě s administrátory pomocí protokolu `ssh`, který uživatele autentizuje pomocí

adresářové služby. Standardně je přístup na tento server uživatelům zakázán a povolen pouze na vyžádání. Dále má na tomto serveru každý uživatel svůj poštovní účet dostupný přes protokoly `pop3` nebo `imap4`. Jako serverový software pro tyto protokoly je použit software Cyrus IMAP, který jako autentizační zdroj používá centrální LDAP server dostupný přes protokol `ldaps`. Odesílání pošty zajišťuje server Postfix pomocí protokolu `smtp`. Poštu je možné pomocí tohoto serveru odesílat i mimo síť `147.32.110.0/23` a to díky protokolu `SMTP_AUTH`. Ten využívá protokol `SASL` a `SASL` opět získává autentizační informace pomocí protokolu `ldaps` z centrálního serveru. Pro tento server musí být vygenerováno několik digitálních certifikátů [X.509] podepsaných lokální certifikační autoritou. Jsou to certifikáty pro domény `apps.sin.cvut.cz`, `www.sin.cvut.cz`, `imap.sin.cvut.cz`, `pop3.sin.cvut.cz`, `mail.sin.cvut.cz` a `smtp.sin.cvut.cz`. Na tomto serveru jsou pro potřeby uživatelů klubu k dispozici SQL servery PostgreSQL, MySQL a FireBird a především `http` a `https` server Apache, který zajišťuje veřejnou prezentaci stránek klubu a domovských stránek uživatelů.

backup – zálohovací server na kterém jsou umístěny pravidelné zálohy zbývajících serverů a případně obsahu disků členů klubu, kteří potřebují krátkodobě zálohovat své počítače. Na tomto serveru je záloha aplikace pro správu uživatelů a rovněž replikovaný LDAP server.

Ostatní servery jsou pro další funkčnost sítě nepodstatné a poskytují pouze doplňkové služby, např. TV server nebo news server. U news serveru by mohlo přicházet v úvahu využití adresářové služby pro autentizaci uživatelů k přístupu na privátní diskusní skupiny, ale zatím nebyl tento požadavek vznesen.

Při návrhu a realizaci nelze použít mechanismus Single Sign One pro možnost jediné autentizace pro všechny služby společně. Důvodem je velká heterogenita sítě a neexistence žádného využitelného standardu. Nabízí se sice možnost využití systému Kerberos, jeho konfigurace však není jednoduchou záležitostí a v případě systémů Microsoft Windows došlo k modifikaci tohoto standardu. V podnikovém informačním systému je situace odlišná. Většinou je v nich homogenní prostředí operačních systémů a administrátorská práva k počítačům mají pouze jejich správci. V síti KSk je každý majitel počítače jeho vlastním administrátorem, proto se k těmto počítačům nemůže navrhovaný systém chovat jako k důvěryhodným a autentizace pomocí Kerbera by mohla přinést bezpečnostní rizika. Navíc by bylo nutné přinutit uživatele, aby používali na svých počítačích jména a hesla sjednocená s nově vytvářeným systémem.

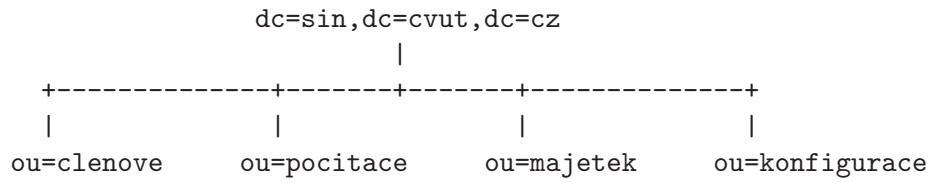
5.3 Jmenné prostory

Adresářový strom má zvolený prefix `dc=sin,dc=cvut,dc=cz` a jeho tvar je navržen podle schématu na obr. 5.3, strana 54..

V DN jednotlivých částí DIT by bylo možno použít názvy dle [RFC 2253] v kódování `UTF-8`, ovšem vzhledem k případnným komplikacím s různými národními prostředími jednotlivých aplikací byly názvy záznamů zvoleny v kódování `ASCII`.

5.3.1 Přehled jednotlivých větví

V následujícím přehledu jednotlivých větví navrženého DIT jsou vysvětleny jejich významy. Některé větve se mohou dále rozšiřovat.



Obr. 5.3: KSk - návrh DIT

ou=clenove

V této větvi jsou uchovány informace o členech klubu jako jsou jejich jméno, příjmení, adresa, ID pod kterým platí své příspěvky, digitální certifikát, foto a celá řada dalších atributů. Proti záznamům v této větvi se mohou uživatelé autentizovat při používání již zmíněných služeb protokolů `imap4`, `pop3` ad. Každý uživatel si bude moci tuto větev připojit do svého e-mailového klienta pro vyhledání e-mailových adres ostatních členů klubu. Při autentizaci by se použil kontext `ou=clenove, dc=sin, dc=cvut, dc=cz`.

ou=pocitace

V této větvi jsou uchovány informace o počítačích členů klubu. Z těchto záznamů dynamicky čerpají data pro svůj běh servery DNS a DHCP. Je však velmi snadné záznamy v této větvi přizpůsobit pro statické generování potřebných souborů. Toto řešení bylo zvoleno především díky snadné distribuci potřebných dat při případném přesunu těchto serverů na jiné stroje. Větev se dále dělí na:

`zoneName=sin.cvut.cz` – kontejner, ve kterém jsou uloženy informace o počítačích členů klubu. Každý z těchto počítačů má v této větvi svůj záznam obsahující jeho název, příslušnou IPv4 adresu, IPv6 adresu vygenerovanou z prefixu získaného od SU, HW adresu pro DHCP server a odkaz na majitele počítače.

`zoneName=110.32.147.in-addr.arpa` – kontejner, ve kterém jsou uloženy záznamy reverzní domény pro síť 147.32.110.0

`zoneName=111.32.147.in-addr.arpa` – kontejner, ve kterém jsou uloženy záznamy reverzní domény pro síť 147.32.111.0

Reverzní doména pro IPv6 zatím nebyla navržena.

ou=majetek

V této větvi jsou uloženy záznamy o majetku klubu. Základní informace jsou uloženy v relační databázi, ve které lze díky vhodně navrženým objektovým třídám a atributům zachytit velmi podrobnou konfiguraci konkrétního zařízení včetně fotografie.

ou=konfigurace

V této větvi jsou uloženy obecně známé informace, které jsou běžně uloženy v souborech `/etc/services`, `/etc/protocols` apod. Pro jejich převod z klasické textové podoby uložené ve

zmíněných souborech lze použít software migration tools od společnosti PADL Software Pty Ltd[1]. Dále tu je uložena konfigurace DHCP serveru.

5.4 Adresářová služba

Vzhledem k požadavkům na používání otevřeného software byl vybrán server [OpenLDAP]. Při jeho instalaci je nutné použít některé doplňkové knihovny, jakým jsou především [OpenSSL] pro podporu protokolu TLS a [Cyrus SASL] pro podporu protokolu SASL. [OpenLDAP] používá pro svůj běh program `slapd`, který by měl být standardně umístěn v adresáři `/usr/sbin` případně v `/usr/libexec`. Používá konfigurační soubor `slapd.conf` umístěný v adresáři `/etc/openldap`. Jeho konfigurace je podrobně popsána v manuálových stránkách a na stránkách projektu [OpenLDAP].

Mezi nejdůležitější parametry v konfiguračním souboru patří následující direktivy:

`include` – Cesty k souborům se schématy

`database` – Databázový backend, standardně nastavena hodnota `bdb` pro Berkley DB, jinak je možno použít backendy `ldbm`, `sql` ad.

`suffix` – Prefix DIT `"dc=sin, dc=cvut, dc=cz"`

`rootdn` – DN správce DIT `"cn=manager, ou=Users, ou=konfigurace, dc=sin, dc=cvut, dc=cz"`

`rootpw` – Heslo správce DIT. Pokud není uvedeno, předpokládá se, že je uloženo v databázi SASLu.

Hesla ve tvaru `SHA1` a `MD5` lze generovat pomocí utility `slappasswd`.

5.5 Schémata

Při návrhu schémat objektových tříd a atributů se vycházelo ze snahy o maximální využití standardů. V první řadě bylo nutné získat jedinečné OID pro potřeby návrhu nových atributů a objektových tříd. Vzhledem k vazbě na SU bylo poskytnuto OID z rozsahu SU a sice `1.3.6.1.4.1.16748.3`. S tímto rozsahem OID bylo možno naložit dle libosti, proto byl rozsah dále rozdělen na části podle tab. 5.1, strana 55.

Vzhledem k možnosti výskytu některých společných názvů v jiných aplikacích by mohlo dojít k duplicitě názvů atributů a objektových tříd, proto bylo zvoleno pojmenování atributů a objektových tříd tak, aby jejich název obsahoval **ksk...**.

5.5.1 Členové

Pro uložení informací o jednotlivých členech klubu jsou z relační databáze importovány základní údaje, kterými jsou:

uid – identifikační číslo člena v rámci klubu

jméno – jméno člena klubu

[1] `www.padl.com`

OID	Význam
1.3.6.1.4.1.16748.3	OID KSk
1.3.6.1.4.1.16748.3.1	SNMP elementy
1.3.6.1.4.1.16748.3.2	LDAP elementy
1.3.6.1.4.1.16748.3.2.1	Atributy
1.3.6.1.4.1.16748.3.2.1.1	Vlastní atribut
1.3.6.1.4.1.16748.3.2.2	Objektové třídy
1.3.6.1.4.1.16748.3.2.2.1	Vlastní objektová třída

Tab. 5.1: Hierarchie OID pro KSk

příjmení – příjmení člena klubu

rodné číslo (číslo pasu) – rodné číslo nebo číslo pasu pro evidenci SU

škola – fakulta ,kterou člen klubu navštěvuje nebo je zaměstnancem

pokoj – číslo pokoje, na kterém člen klubu bydlí

konto – uživatelský účet

e-mail – oficiální e-mailová adresa, na kterou jsou zasílána důležitá upozornění

datum – datum vstupu do klubu

K těmto základním informacím jsou přiřazeny další, které jsou uloženy pouze v adresářovém serveru a nejsou nezbytně nutné pro evidenci členů, Bez nich mu však například nebude správně doručována ani odesílána elektronická pošta. Uvedené atributy SQL databáze jsou přemapovány na LDAP atributy v příslušných objektových třídách.

Základní použité objektové třídy a jejich atributy (povinné jsou vyznačeny silně):

top – základní objektová třída

person – dědí po objektové třídě `top`

- **cn** – Běžný název, v tomto případě jméno a příjmení uživatele
- **sn** – Příjmení uživatele
- `userPassword` – Heslo uživatele. Tato položka je nutná pokud bude chtít uživatel přijímat a odesílat elektronickou poštu z lokálního serveru. Heslo bude uloženo ve formátu `MD5`
- `telephoneNumber` – Telefonní číslo uživatele
- `seeAlso` – DN na související objekty, např. počítače, které uživatel vlastní
- `description` – Popis

organizationalPerson – dědí po objektové třídě `person`

- `title` – Titul, např. Ing., Bc. apod.
- `ou` – Organizační jednotka. V tomto případě studijní obor který uživatel studuje nebo vyučuje

inetOrgPerson – dědí po objektové třídě `organizationalPerson`, definována v [RFC 2798]

`employeeNumber` – UID pod kterým je uživatel v systému zaveden.

`employeeType` – Výčtový typ uživatele (admin, vip, normal)

`jpegPhoto` – Fotografie uživatele ve formátu JPEG

`givenName` – Jméno uživatele

`displayName` – Jméno, které se bude zobrazovat v aplikacích které umějí s adresářovými službami spolupracovat

`initials` – Iniciály uživatele

`preferredLanguage` – Jazyk, který uživatel preferuje při komunikaci

`roomNumber` – Číslo pokoje uživatele

`o` – Organizace, kde uživatel pracuje. V tomto případě je tento atribut použitý pro uložení názvu fakulty, kterou studuje.

`departmentNumber`

`homePostalCode` – Poštovní adresa uživatele

`homePhone` – Telefon na uživatelovu pevnou linku

`mobile` – Mobilní telefon

`userCertificate` – Digitální certifikát uživatele

`userPKCS12` – Digitální certifikát podle normy #PKCS12

`userSMIMECertificate` – Digitální certifikát pro podepisování el. pošty

`mail` – Oficiální uživatelův certifikát

posixAccount – objektová třída pro UNIXové uživatelské účty. Pomocí této třídy bude uživatel přistupovat ke své elektronické poště apod.

uid – Uživatelův účet pro výběr pošty

uidNumber – Číslo uživatelova účtu, standardně to bývá číslo vyšší než-li 1000

gidNumber – Číslo skupiny, do které uživatel patří. Standardně to bývá číslo 100, což odpovídá skupině `users`

homeDirectory – Domovský adresář uživatele

`gecos` – Identifikační údaje pro utility typu `finger`

`loginShell` – Cesta k uživatelovu shellu, standardně nastavenaá na `/bin/false`

shadowAccount – objektová třída pro stínová hesla. Nastavují se zde atributy pro expiraci hesel, jejich minimální a maximální délku apod.

inetLocalMailRecipient – objektová třída pro práci s e-maily podle draftu RFC `draft-lachman-laser-ldap-mail-routing`

Pro záznam dalších informací o uživatelích je nutné navrhnout novou objektovou třídu a příslušné atributy.

kskUser

```
objectclass ( 1.3.6.1.4.1.16748.3.2.2.1
        NAME 'kskUser'
        SUP top AUXILIARY
        DESC 'Schema členů KSk'
        MAY ( kskNativeNumber $ kskPassportNumber $ kskDate $
        kskMD5CertFingerPrint $ kskSHA1CertFingerPrint $
        kskICQ $ kskJabber $ kskYahoo $ kskAIM $ kskMSN $ kskIRC ) )
```

`kskNativeNumber` – Rodné číslo uživatele

```
attributetype (1.3.6.1.4.1.16748.3.2.1.1 NAME 'kskNativeNumber'
DESC 'Rodné číslo uživatele'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

`kskPassportNumber` – Číslo pasu uživatele

```
attributetype (1.3.6.1.4.1.16748.3.2.1.2 NAME 'kskPassportNumber'
DESC 'Číslo pasu zahraničního uživatele'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

`kskDate` – datum vstupu uživatele do klubu

```
attributetype (1.3.6.1.4.1.16748.3.2.1.3 NAME 'kskDate'
DESC 'Datum'
EQUALITY generalizedTimeMatch
ORDERING generalizedTimeOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

`kskMD5CertFingerPrint` – MD5 fingerprint uživatelova certifikátu

```
attributetype (1.3.6.1.4.1.16748.3.2.1.4 NAME 'kskMD5CertFingerPrint'
DESC 'MD5 fingerprint uživatelova certifikátu'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

`kskSHA1CertFingerPrint` – SHA1 fingerprint uživatelova certifikátu

```
attributetype (1.3.6.1.4.1.16748.3.2.1.5 NAME 'kskSHA1CertFingerPrint'
DESC 'SHA1 fingerprint uživatelova certifikátu'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

`kskICQ` – ICQ kontakt na uživatele

```
attributetype (1.3.6.1.4.1.16748.3.2.1.6 NAME 'kskICQ'
DESC 'ICQ kontakt na uživatele
EQUALITY caseIgnoreIA5Match'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

`kskJabber` – Jabber kontakt na uživatele

```
attributetype (1.3.6.1.4.1.16748.3.2.1.7 NAME 'kskJabber'
DESC 'Jabber kontakt na uživatele'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

`kskYahoo` – Yahoo kontakt na uživatele

```
attributetype (1.3.6.1.4.1.16748.3.2.1.8 NAME 'kskYahoo'
DESC 'Yahoo kontakt na uživatele'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

`kskAIM` – AIM kontakt na uživatele

```
attributetype (1.3.6.1.4.1.16748.3.2.1.9 NAME 'kskAIM'
DESC 'AIM kontakt na uživatele'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

`kskMSN` – MSN messenger kontakt na uživatele

```
attributetype (1.3.6.1.4.1.16748.3.2.1.10 NAME 'kskMSN'
DESC 'MSN messenger kontakt na uživatele'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

`kskIRC` – IRC kontakt na uživatele

```
attributetype (1.3.6.1.4.1.16748.3.2.1.11 NAME 'kskIRC'
DESC 'IRC kontakt na uživa
EQUALITY caseIgnoreIA5Match'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

Záznamy o uživatelích jsou uloženy v kontejneru `ou=clenove,dc=sin,dc=cvut,dc=cz` pod RDN `employeeNumber=UID uživatele` a záznam by pak mohl vypadat např. takto:

```
dn: employeeNumber=95799,ou=clenove,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: inetLocalMailRecipient
objectClass: kskUser
sn: Benák
cn: Karel Benák
employeeNumber: 95799
employeeType: admin
givenName: Karel
uid: beny
uidNumber: 1032
gidNumber: 100
homeDirectory: /home/beny
```

```
loginShell: /bin/bash
mail: karel.benak@sin.cvut.cz
mail: beny@sin.cvut.cz
kskNativeNumber: 770319/XXXX
kskICQ: 36646638
```

5.5.2 Počítače

Pro uložení informací o jednotlivých počítačích členů klubu a serverech jsou z relační databáze exportovány základní údaje, kterými jsou:

id_clena – identifikační číslo majitele počítače

jméno – název počítače

mac – HW adresa počítače

ip – IPv4 adresa počítače

zasuvka – Zásuvka, do které je počítač připojen

datum – Datum připojení počítače

stav – Stav počítače. Ten může být buď aktivní, vyřazený nebo dočasně odpojený

Informace o počítačích jsou poněkud komplikovanější a program pro jejich generování musí řešit problematický zápis do reverzních záznamů. Pro uložení normálních záznamů o počítačích slouží větev `zoneName=sin.cvut.cz, ou=pocitace, dc=sin, dc=cvut,dc=cz`. V tomto záznamu jsou použita následující schémata a atributy:

top – základní objektová třída

dnsZone – objektová třída pro spolupráci s DNS serverem

- **zoneName** – jméno zónového souboru
- **relativeDomainName** – název počítače v dané zóně
- `DNSTTL` – atribut pro hodnotu časového údaje time to live
- `DNSClass`
- `ARecord` – IPv4 adresa
- `MDRecord`
- `MXRecord` – MX záznam e-mailového serveru
- `NSRecord` – NS záznam o nameserveru
- `SOARecord`
- `CNAMERecord` – Záznam o aliasu
- `PTRRecord` – PTR záznam používaný v reverzních zónách
- `HINFORecord`
- `MINFORecord`

`TXTRecord`

`SIGRecord`

`KEYRecord`

`AAAARecord` – IPv6 adresa ve formátu AAAA záznamu

`LOCRecord`

`NXTRecord`

`SRVRecord`

`NAPTRRecord`

`KXRecord`

`CERTRecord`

`A6Record` – IPv6 adresa ve formátu A6 záznamu

`DNAMERecord` – Reverzní IPv6 záznam

dhcpSubnet – Objektová třída DHCP pro nastavení podsítě

`dhcpNetMask` – Délka masky podsítě

dhcpOptions – Objektová třída DHCP pro nastavení parametrů

`dhcpOption` – Parametry, které se zasílají klientským stanicím. Atribut může být vícehodnotový.

dhcpComputer – Objektová třída DHCP pro uložení informací o počítači. Tuto objektovou třídu je nutné upravit z typu `STRUCTURAL` na typ `AUXILIARY`, aby ji bylo možno použít společně se třídou `dnsZone`

dhcpHWAddress – HW adresa síťové karty počítače

dhcpStatements – Uložení předdefinované IPv4 adresy kterou bude DHCP server poskytovat stanici poskytovat

dhcpServer – Objektová třída pro nastavení konfigurace serveru. Tuto objektovou třídu je nutné upravit z typu `STRUCTURAL` na typ `AUXILIARY`, aby ji bylo možno použít společně se třídou `dnsZone`

dhcpServiceDN – DN záznamu, kde je uložena konfigurace serveru.

kskComputer

```
objectclass ( 1.3.6.1.4.1.16748.3.2.2.2
        NAME 'kskComputer'
        SUP top AUXILIARY
        DESC 'Schema pro počítače členů KSk'
        MAY ( owner $ roomNumber $ kskDataConnector ) )
```

`owner` – DN vlastníka počítače

`roomNumber` – Číslo pokoje, kde je počítač umístěn

`kskDataConnector` – Číslo zásuvky

```
attributetype (1.3.6.1.4.1.16748.3.2.1.12 NAME 'kskDataConnector'
DESC 'Číslo zásuvky'
EQUALITY integerMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )
```

kskServer

```
objectclass ( 1.3.6.1.4.1.16748.3.2.2.3
        NAME 'kskServer'
        SUP top AUXILIARY
        DESC 'Schema pro počítače členů KSk'
        MAY ( manager $ roomNumber ) )
```

`manager` – DN správce daného serveru

`roomNumber` – Místnost, kde je zařízení umístěno

```
dn: ou=pocitace,dc=sin,dc=cvut,dc=cz
ou: pocitace
objectClass: top
objectClass: organizationalUnit
objectClass: dhcpService
cn: DHCP Config
dhcpStatements: ddns-update-style none
dhcpPrimaryDN: relativeDomainName=services,zoneName=sin.cvut.cz,
ou=pocitace,dc=sin,dc=cvut,dc=cz

dn: zoneName=sin.cvut.cz,ou=pocitace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: dnsZone
objectClass: dhcpSubnet
objectClass: dhcpOptions
zoneName: sin.cvut.cz
relativeDomainName: @
dNSTTL: 86400
aRecord: 147.32.110.2
aAAARecord: 2001:718:2:880:210:5aff:fed7:8380
mXRecord: 10 mail.sin.cvut.cz.
nSRecord: ns.sin.cvut.cz.
nSRecord: apps.sin.cvut.cz.
sOARecord: services.sin.cvut.cz. admin.sin.cvut.cz. 200406011330 10800 3600
604800 86400
cn: 147.32.110.0
dhcpNetMask: 23
dhcpOption: subnet-mask 255.255.254.0
dhcpOption: broadcast-address 147.32.111.255
dhcpOption: routers 147.32.110.1
```

```
dhcpOption: domain-name-servers 147.32.110.2
dhcpOption: domain-name-servers 147.32.110.3
dhcpOption: domain-name "sin.cvut.cz"
```

Záznam větve `zoneName=110.32.147.in-addr.arpa, ou=pocitace, dc=sin, dc=cvut,dc=cz` sloužící pro reverzní doménu sítě 147.32.110 má následující tvar:

```
dn: zoneName=110.32.147.in-addr.arpa,ou=pocitace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: dNSZone
zoneName: 110.32.147.in-addr.arpa
relativeDomainName: @
dNSTTL: 86400
nSRecord: services.sin.cvut.cz.
nSRecord: apps.sin.cvut.cz.
sOARecord: services.sin.cvut.cz. beny.sin.cvut.cz. 2003080301 10800 3600
604800 86400
```

Reverzní doména 147.32.111 má shodný záznam, jen hodnota atributu `zoneName` je nastavena na hodnotu `111.32.147.in-addr.arpa`

Vlastní záznam o počítači člena klubu má následující tvar:

```
dn: relativeDomainName=jsb,zoneName=sin.cvut.cz,ou=Hosts,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: dNSZone
objectClass: dhcpHost
objectClass: kskComputer
zoneName: sin.cvut.cz
relativeDomainName: jsb
aRecord: 147.32.110.45
aAAARecord: 2001:718:2:880:230:4fff:fe13:86ac
aAAARecord: 2001:718:2:880:0000:5efe:9320:6e2d
owner: employeeNumber=58014,ou=clenove,dc=sin,dc=cvut,dc=cz
cn: jsb
dhcpHWAddress: ethernet 00:30:4F:13:86:AC
dhcpStatements: fixed-address 147.32.110.45
```

Následující záznam ukazuje konfiguraci serveru `services.sin.cvut.cz`, u ostatních serverů je situace podobná. Servery jsou totiž konfigurovány pro statickou konfiguraci IP adresy a nepotřebují tedy žádný záznam pro DHCP server.

```
dn: relativeDomainName=services,zoneName=sin.cvut.cz,ou=Hosts,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: dNSZone
objectClass: dhcpServer
objectClass: kskServer
zoneName: sin.cvut.cz
relativeDomainName: services
ARecord: 147.32.110.2
```

```
AAAARecord: 2001:718:2:880:210:5aff:fed7:8380
AAAARecord: 2001:718:2:880:0000:5efe:9320:6e02
CNAMERecord: ns
CNAMERecord: intranet
CNAMERecord: ldap
CNAMERecord: db
owner: dc=sin,dc=cvut,dc=cz
managerer: employeeNumber=95799,ou=clenove,dc=sin,dc=cvut,dc=cz
cn: services
dhcpServiceDN: ou=pocitace,dc=sin,dc=cvut,dc=cz
```

Tyto relativně komplikované záznamy budou využívat DNS server a DHCP server. Pokud se v adresáři změní nějaká hodnota, servery okamžitě zareagují bez nutnosti generování nových konfiguračních souborů a jejich znovunačítání spojeného s případným restartem služby.

5.5.3 Majetek

Tato větev slouží pro rychlé prohlížení evidence zařízení, jakými jsou např. různé switche, huby, scannery apod.

top

device

cn – Obecný název, v tomto případě zastupuje název zařízení, např. switch

`serialNumber` – Sériové číslo zařízení

`seeAlso` – Pokud je toto zařízení součástí jiného, pak je zde uvedeno jeho DN

`description` – Detailní popis zařízení

kskDevice – Objektová třída pro podrobnější popis zařízení a jeho aktuálních zápůjček.

```
objectclass ( 1.3.6.1.4.1.16748.3.2.2.4
        NAME 'kskDevice'
        SUP device AUXILIARY
        DESC 'Majetek KSk'
        MAY ( manager $ jpegPhoto $ roomNumber $ kskBorrower $
        kskDateLoan $ kskDateReturn  ) )
```

`manager` – DN správce zařízení

`jpegPhoto` – Foto zařízení

`roomNumber` – Číslo místnosti, kde je zařízení umístěno

`kskBorrower` – DN člověka, kterému bylo zařízení půjčeno

```
attributetype (1.3.6.1.4.1.16748.3.2.1.13 NAME 'kskBorrower'
DESC 'Osoba, které je zařízení zapůjčeno'
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE )
```

`kskDateLoan` – Datum kdy bylo zařízení půjčeno

```
attributetype (1.3.6.1.4.1.16748.3.2.1.14 NAME 'kskBorrower'
DESC 'Datum zapůjčení zařízení'
EQUALITY generalizedTimeMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE )
```

`kskDateReturn` – Datum kdy bylo zařízení vráceno

```
attributetype (1.3.6.1.4.1.16748.3.2.1.15 NAME 'kskDateReturn'
DESC 'Datum předpokládaného vrácení zařízení'
EQUALITY generalizedTimeMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE )
```

Objektové třídy je možné dále rozvést pro konkrétní potřeby a je také možné např. vytvořit třídu, zabývající se speciálně rozbočovači, přepínači apod.

Příklad záznamu o skeneru:

```
dn: serialNumber=123X321,ou=majetek,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: device
objectClass: kskDevice
cn: Scanner HP
serialNumber: 123X321
manager: employeeNumber=95799,ou=clenove,dc=sin,dc=cvut,dc=cz
roomNumber: x13
```

5.5.4 Konfigurace

V této větvi jsou uloženy obecně sdílené informace jako jsou např. názvy uživatelských skupin, vazby čísel portů a jejich názvů z `/etc/services` apod. V příkladech jsou uvedeny jen některé záznamy, jejich kompletní seznam by byl příliš dlouhý. Jednotlivé části této větve mohou využít i uživatelé, kteří chtějí mít konfiguraci shodnou s konfigurací hlavních serverů pomocí utilit `nss_ldap` a `pam_ldap` od již zmiňované společnosti PADL Software Pty Ltd. Pro migraci textových souborů do adresářového stromu je možné použít `Migration Tools`.

```
dn: ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: organizationalUnit
ou: konfigurace
```

ou=Group – definice systémových skupin, které by měly být na všech serverech společné

```
dn: ou=Group,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: organizationalUnit
ou: Group
```

cn=users – základní skupina pro všechny uživatele

```
dn: cn=users,ou=Group,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: posixGroup
cn: users
gidNumber: 100
```

cn=postfix – skupina uživatele postfix

```
dn: cn=postfix,ou=Group,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: posixGroup
cn: postfix
gidNumber: 20
```

ou=Networks – seznam sítí patřících do SU. Odpovídá souboru `/etc/networks`

```
dn: ou=Networks,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: organizationalUnit
ou: Networks
```

cn=sin.cvut.cz

```
dn: cn=sin.cvut.cz,ou=Networks,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: ipNetwork
cn: sin.cvut.cz
ipNetworkNumber: 147.32.110.0
ipNetmaskNumber: 255.255.254.0
```

ou=Protocols – seznam protokolů. Odpovídá souboru `/etc/protocols`. Aktuální seznam je možné získat z `http://www.iana.org`

```
dn: ou=Protocols,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: organizationalUnit
ou: Protocols
```

cn=icmp

```
dn: cn=icmp,ou=Protocols,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: ipProtocol
cn: icmp
ipProtocolNumber: 1
description: Protocol icmp
```

ou=Rpc – seznam portů pro RPC. Odpovídá souboru `/etc/rpc`. Aktuální seznam je možné získat z `http://www.iana.org`

```
dn: ou=Rpc,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: organizationalUnit
ou: Rpc
```

cn=ypserv

```
dn: cn=ypserv,ou=Rpc,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: oncRpc
cn: ypserv
oncRpcNumber: 100004
description: RPC ypprog
```

ou=Services – seznam služeb a odpovídajících čísel portů. Odpovídá souboru `/etc/services`. Jejich oficiální seznam je možné získat z `http://www.iana.org`

```
dn: ou=Services,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: organizationalUnit
ou: Users
```

cn=ssh

```
dn: cn=ssh,ou=Services,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: ipService
cn: ssh
ipServicePort: 22
ipServiceProtocol: tcp
ipServiceProtocol: udp
```

ou=Users – definice systémových účtů, které by měly být na všech serverech společné.

```
dn: ou=Users,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: organizationalUnit
ou: Users
```

cn=postfix – uživatel *postfix*

```
dn: cn=postfix,ou=Users,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectclass: account
objectClass: posixAccount
cn: postfix
uid: postfix
uidNumber: 100
gidNumber: 20
homeDirectory: /var/spool/postfix
```

V kontejneru ou=Users, ou=konfigurace, dc=sin, dc=cvut, dc=cz budou uloženy i záznamy o uživateli manager, který je hlavním správcem adresářového serveru a uživateli proxyuser, který bude podle bezpečnostního modelu sloužit jako prostředník při autentizaci uživatelů.

```
dn: cn=manager,ou=Users,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: person
cn: manager
password: {MD5}XYZ==

dn: cn=proxyuser,ou=Users,ou=konfigurace,dc=sin,dc=cvut,dc=cz
objectClass: top
objectClass: person
cn: manager
password: {MD5}ZYY==
```

Pro uživatele proxyuser je třeba nastavit příslušná práva ke čtení uživatelských hesel:

```
access to dn=".*,dc=sin,dc=cvut,dc=cz" attr=userPassword
by dn="cn=manager,ou=Users,ou=konfigurace,dc=sin,dc=cvut,dc=cz" write
by dn="cn=proxyuser,ou=Users,ou=konfigurace,dc=sin,dc=cvut,dc=cz" read
by self write
by * auth
```

5.6 Spolupráce se software

5.6.1 Operační systémy

V první řadě je třeba nastavit servery tak, aby spolupracovali s adresářovými službami. Vzhledem k tomu, že použité operační systémy na centrálních serverech jsou typu UNIX, konkrétně Linux, je řešení poměrně snadné. Linux využívá služeb systémové knihovny glibc, která poskytuje systému mnoho standardizovaných funkcí jazyka C, které využívají různé aplikace. Ta má několik knihoven, které mohou používat různé zdroje informací. Např. libnss_files.so umožňuje získávání informací a konfigurací z textových souborů, libnss_dns.so umožňuje použití systému DNS pro převod jmen počítačů na IP adresy a libnss_nis.so umožňuje získávat potřebné informace z NIS serveru.

Již několikrát zmiňovaná knihovna nss_ldap od PADL Software Pty Ltd. umožňuje získávání informací z LDAP serveru. Knihovna nss_ldap implementuje [RFC 2307], a proto umožňuje použití adresářových služeb i na jiných systémech než-li je Linux, např. Solaris, AIX, HP-UX ad. a spolupracuje např. s [OpenLDAP], [JES], NDS, [AD] ad.

Ke správnému provozu knihovny je nutné nainstalovat některou ze systémových knihoven nutných pro práci s LDAP službami, např. knihovny libldap.so, libldap_r.so a liblber.so z projektu [OpenLDAP].

Pro použití adresářových služeb v operačním systému je v prvním kroku nutné nastavit klienta knihovny nss_ldap. Standardně se jeho konfigurační soubor jmenuje /etc/ldap.conf a obsahuje celou řadu parametrů.

host – IP adresy nebo názvy LDAP serverů oddělených mezerou, ze kterých se budou získávány potřebné informace, např. ldap.sin.cvut.cz

`base` – DN kořenového záznamu, např. `dc=sin, dc=cvut, dc=cz`

`ldap_version` – Verze LDAP protokolu, standardně nastavena na hodnotu `3`

`bind`

`bindpwd`

`ssl start_tls` – Direktiva pro použití vrstvy TLS při komunikaci s LDAP serverem

`ssl on`

`tls_cacertdir` – Cesta k certifikátům certifikačních autorit, např. `/etc/ssl/certs`

V tomto souboru je rovněž možné poměrně snadno využít přemapování názvů atributů používaných v adresářovém serveru na názvy atributů používaných jmennými službami podle [RFC 2307]

V dalším kroku je nutné aktualizovat soubor `/etc/nsswitch.conf` pro použití adresářových služeb a u jednotlivých konfiguračních direktiv nastavit správné hodnoty:

```
# Získání informací o˜systémových účtech, napřed ze souborů /etc/passwd,
# /etc/group a /etc/shadowm. Pokud v nich nebude záznam nalezen,
# pokusí se systém hledat v ldap serveru
passwd:     files ldap
group:      files ldap
shadow:     files ldap

# !!!Varování!!! Velmi citlivé na správný překlad jmen,
# bez něj by nemuselo dojít ke kontaktu s LDAP serverem.
# libldap využívá volání funkce gethostbyname().
hosts:      files dns ldap

# LDAP je autoritativní službou ve které se má daný záznam vyhledat
services:   ldap [NOTFOUND=return] files # /etc/services
networks:   ldap [NOTFOUND=return] files # /etc/networks
protocols:  ldap [NOTFOUND=return] files # /etc/protocols
rpc:        ldap [NOTFOUND=return] files # /etc/rpc
```

Doplňkovou knihovnou je knihovna `pam_ldap`. `PAM` moduly slouží pro nastavení různých variant přístupových práv k různým službám.

V definici PAM modulů pro jednotlivé typy přístupů standardně uložených v `/etc/pam.d` je nutné uvést používání modulu `pam_ldap.so`. V přiložených konfiguračních souborech jsou praktické ukázky, jak příslušnou službu nakonfigurovat pro spolupráci s LDAP.

```
# /etc/pam.d/login
auth       required     /lib/security/pam_securetty.so
auth       required     /lib/security/pam_nologin.so
auth       sufficient   /lib/security/pam_ldap.so
auth       required     /lib/security/pam_unix_auth.so try_first_pass
account    sufficient   /lib/security/pam_ldap.so
```

```
account    required     /lib/security/pam_unix_acct.so
password   required     /lib/security/pam_cracklib.so
password   required     /lib/security/pam_ldap.so
password   required     /lib/security/pam_pwdb.so use_first_pass
session    required     /lib/security/pam_unix_session.so
```

Více informací o konfiguraci PAM lze nalézt např. v článku PAM – správa autentizačních mechanismů[2].

5.6.2 DNS

Pro DNS bude použit software BIND od společnosti ISC[3] ve verzi 9.2.X, která má podporu protokolu IPv6 a umožňuje využívat i jiných zdrojů dat pomocí rozhraní `sdb`, než-li jsou klasické textové soubory. Na serveru `services.sin.cvut.cz` bude umístěn primární DNS server, který bude používat data získávána z LDAP serveru. Konkrétně se jedná o větve:

`zoneName=sin.cvut.cz,ou=pocitace,dc=sin,dc=cvut,dc=cz` – zóna `sin.cvut.cz`

`zoneName=110.32.147.in-addr.arpa,ou=pocitace,dc=sin,dc=cvut,dc=cz` – reverzní zóna pro 147.32.110.0

`zoneName=111.32.147.in-addr.arpa,ou=pocitace,dc=sin,dc=cvut,dc=cz` – reverzní zóna pro 147.32.111.0

Server DNS je nutné přeložit s podporou protokolu LDAP. Postup úprav je k nalezení na adrese `http://klobouk.fsv.cvut.cz/~beny/ldap/ldap/ldap-8.html`

Do konfiguračního souboru `bindu` je nutné pro zónu `sin.cvut.cz` uvést:

```
zone "sin.cvut.cz" {
     type master;
     database "ldap ldap://127.0.0.1/zoneName=sin.cvut.cz,
     ou=pocitace,dc=sin,dc=cvut,dc=cz 86400";
};
```

A pro reverzní domény `110.32.147.in-addr.arpa` a `111.32.147.in-addr.arpa`:

```
zone "110.32.147.in-addr.arpa" {
     type master;
     database "ldap ldap://127.0.0.1/zoneName=110.32.147.in-addr.arpa,\
     ou=pocitace,dc=sin,dc=cvut,dc=cz 86400";
};

zone "111.32.147.in-addr.arpa" {
     type master;
     database "ldap ldap://127.0.0.1/zoneName=111.32.147.in-addr.arpa,\
     ou=pocitace,dc=sin,dc=cvut,dc=cz 86400";
};
```

Při jakékoliv změně některého ze záznamů v LDAP se tyto změny projeví okamžitě, což je výhoda kterou použití LDAP serveru přináší.

[2] `http://www.root.cz/clanek/478`

[3] www.isc.org

5.6.3 DHCP

DHCP server zajišťuje přidělování statických IP adres na základě znalosti HW adresy počítače. Je nakonfigurován tak, aby údaje získával dynamicky z LDAP serveru. Tuto funkčnost je nutné zajistit podle návodu na adrese `http://klobouk.fsv.cvut.cz/~beny/ldap/ldap/ldap-9.html`. Dále je nutné naimportovat soubor se schematy do adresáře `/etc/openldap/schema` a zajistit, aby daemon `slapd` používal i toto nové schema použitím direktivy `include` v konfiguračním souboru `/etc/openldap.slapd.conf`. V něm je rovněž nutné serveru oznámit, že má indexovat atributy s názvy `dhcpHWAddress` a `dhcpClassData`

```
index     dhcpHWAddress
index     dhcpClassData
```

Před vlastním spuštěním `slapd` daemona je nutné upravit definici schémat objektových tříd `dhcpHost`, `dhcpServer`, `dhcpSubnet` a `dhcpOptions` a změnit na typ `AUXILIARY`, jinak dochází ke konfliktu se třídou `dnsZone`. Po této úpravě je nutné pomocí `slapdindex` přeindexovat data v databázovém backendu a následně lze spustit daemona `slapd`.

Konfigurace DHCP daemona je poměrně jednoduchá, v konfiguračním souboru `/etc/dhcpd.conf` lze uvést následující hodnoty:

```
ldap-server "ldap.sin.cvut.cz"; # LDAP server
ldap-port 389; # Port
ldap-username "cn=proxyuser,ou=Users,ou=konfigurace,dc=sin,dc=cvut,dc=cz";
ldap-password "XYZ";
ldap-base-dn "dc=sin,dc=cvut,dc=cz";
ldap-method dynamic;
```

Po svém spušťení je `dhcpd` server schopen vyřizovat požadavky na přidělení IP adresy dynamicky pomocí LDAP serveru.

5.7 Závěr

V této kapitole je uvedena jen část informačního systému. K dynamickému získávání dat pro DHCP a DNS servery bylo vzhledem k relativně malému počtu záznamů přistoupeno především z důvodu již hotového a vyzkoušeného řešení. Pokud by se měla data pro jednotlivé servery generovat staticky do jejich konfiguračních souborů, bylo by nutné napsat několik obslužných skriptů nejspíš za použití jazyka Perl a vhodným způsobem vyřešit aktualizaci těchto souborů, např. pomocí systémového programu `cron`. Vlastní záznamy by se tím však zjednodušily. Další součástí je služba `smtp`, která slouží k odesílání a přijímání došlé pošty. LDAP server se tu využívá pro překlad z poštovních aliasů na systémové účty a opačne. Dalším využitím LDAPu v `smtp` je autentizace pomocí protokolu `SMTP_AUTH`, díky kterému budou moci uživatelé odesílat svou poštu i mimo kolejní síť.

Kapitola 6

Závěr

Při psaní této práce jsem mohl zúročit své zkušenosti se správou a návrhem adresářového serveru. Zároveň jsem rád, že se mé návrhy řešení správy členů Klubu Sinkuleho koleje a jejich počítačů podařilo realizovat a tím přispět svým kolegům a nástupcům na pozici administrátorů kolejní sítě ke zjednodušení a zpřehlednění správy uživatelů a aplikací.

Bohužel, téma adresářových služeb je natolik rozsáhlé a komplexní, že je velmi obtížné popsat všechny důležité informace v rozsahu diplomové práce a bylo by nutné počítat spíše s objemem rozsáhlejší publikace.

Využití adresářovýs služeb znamená velký přínos pro správu informačního systému, který by využívalo více než-li 50 uživatelů. Mezi administrátory a návrháři systémů je bohužel malá povědomost o jeho možnostech a výhodách. Podle podle mého názoru s tím souvisí částečná či úplná neznalost standardních protokolů a zaměření pouze na firemní produkty, pro jejichž funkčnost není třeba hlubších znalostí.

Literatura

[UDLDAPDS] *Timothy A. Howes, Ph.D., Mark C. Smith, Gordon S. Good:*
Understanding and Deploying LDAP Directory Services, Second Edition
Vydavatelství Addison Wesley, 2003
ISBN 0672-32316-8

[LDAPSA] *Gerald Carter:*
LDAP System Administration
Vydavatelství O'Reilly, 2003
ISBN 1-56592-491-6

[LDAPDE] *Brian Arkills:*
LDAP Directories Explained: An Introduction and Analysis
Vydavatelství Addison Wesley, 2003
ISBN 0-201-78792-X

[PERL] *Larry Wall, Tom Christiansen, Randal L. Schwartz:*
Programování v jazyce Perl
Vydavatelství ComputerPress
ISBN 80-85896-95-8

[BfHA] *Evan Marcus, Hal Stern:*
Blueprints for High Availibility
Vydavatelství John Wiley & Sons, Inc.
ISBN 0471-35601-8

[Sitera 1] *Jiří Sitera:*
Adresářové služby - úvod do problematiky
`http://www.cesnet.cz/doc/techzpravy/2000-4/`

[Sitera 2] *Jiří Sitera:*
Využití adresářových služeb (LDAP) v projektu MetaCentrum
`http://www.cesnet.cz/doc/techzpravy/2001/02/`

[Sitera 3] *Jiří Sitera:*
LDAP a Kerberos
`http://www.cesnet.cz/doc/techzpravy/2002/ldapkrb/ldapkrb.pdf`

[Dostalek] *Libor Dostálek a kolektiv:*
Velký průvodce protokoly TCP/IP Bezpečnost
2. aktualizované vydání

Vydavatelství Computer Press, Praha 2003
ISBN 80-7226-84-X

X.500

[X.501] *X.501: The Models*

[X.509] *X.509: Authentication Framework*

[X.511] *X.511: Abstract Service Definition*

RFC

[RFC 1274] RFC 1274 – The COSINE and Internet X.500 Schema

[RFC 2222] RFC 2222 – Simple Authentication and Security Layer (SASL)

[RFC 2247] RFC 2247 – Using Domains in LDAP/X.500 Distinguished Names

[RFC 2251] RFC 2251 – Lightweight Directory Access Protocol (v3)

[RFC 2252] RFC 2252 – Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions

[RFC 2253] RFC 2253 – Lightweight Directory Access Protocol (v3): UTF–8 String Representation of Distinguished Names

[RFC 2254] RFC 2254 – The String Representation of LDAP Search Filters

[RFC 2255] RFC 2255 – The LDAP URL Format

[RFC 2256] RFC 2256 – A Summary of the X.500(96) User Schema for use with LDAPv3

[RFC 2307] RFC 2307 – An Approach for Using LDAP as a Network Information Service

[RFC 2444] RFC 2444 – The One-Time-Password SASL Mechanism

[RFC 2798] RFC 2798 – Definition of the inetOrgPerson LDAP Object Class

[RFC 2829] RFC 2829 – Authentication Methods for LDAP

[RFC 2830] RFC 2830 – Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security

[RFC 2849] RFC 2849 – LDIF

[RFC 3377] RFC 3377 – Lightweight Directory Access Protocol (v3): Technical Specification

Dokumentace na WWW

[1] OpenLDAP
`http://www.openldap.org/doc`

[2] SunOne Directory Server 5.2 Deployment Guide
`http://docs-pdf.sun.com/816-6700-10/816-6700-10.pdf`

[3] Sun ONE Directory Server 5.2 Getting Started Guide
`http://docs-pdf.sun.com/816-6696-10/816-6696-10.pdf`

[4] Sun ONE Directory Server 5.2 Installation and Tuning Guide
`http://docs-pdf.sun.com/816-6697-10/816-6697-10.pdf`

[5] Sun ONE Directory Server 5.2 Administration Guide
`http://docs-pdf.sun.com/816-6698-10/816-6698-10.pdf`

[6] The JNDI Tutorial *Building Directory-Enabled Java Applications*
`http://java.sun.com/products/jndi/tutorial/index.html`

Software

Serverový software

[OpenLDAP] OpenLDAP – OpenSource implementace LDAP serveru, klientu a API.
`http://www.openldap.org`

[TinyLDAP] TinyLDAP

[JES] Sun Java Enterprise Directory Server – `http://www.sun.com`

[AD] Microsoft Active Directory – Síťový adresářový server pro operační systémy MS Windows
`http://www.microsoft.com`

[7] IBM Tivoli – Adresářová služba od IBM

Editory a prohlížeče

[GQ] GQ – klient pro prohlížení a editaci LDAP záznamů (strana 45.)
`http://biot.com/gq`

Klienti

Vývojové prostředky

[Doxygen] Doxygen – software pro generování dokumentace

[OpenSSL] OpenSSL – Open source SSL/TLS knihovna
`http://www.openssl.org`

[GNU TLS] GNU TLS – Alternativa k OpenSSL
`http://www.gnu.org/software/gnutls`

[Cyrus SASL] Cyrus SASL
`http://asg.web.cmu.edu/sasl/`

[GNU SASL] GNU SASL
`http://www.gnu.org/software/gsasl`

[MIT Kerberos] MIT Kerberos
`http://web.mit.edu/kerberos`

[Heimdal] Heimdal Kerberos
`http://www.pdc.kth.se/heimdal/`

[GNU Shishi] GNU Shishi
`http://www.gnu.org/software/shishi`

[GNU GSS] GNU Generic Security Service Library
`http://www.gnu.org/software/gss`

[LibNTLM] LibNTLM
`http://josefsson.org/libntlm`

[JLDAP] JLDAP
`http://www.openldap.org/jldap`

[JDBC LDAP] JDBC-LDAP
`http://www.openldap.org/jdbcldap`

[Perl-LDAP] Perl-LDAP
`http://ldap.perl.org`

[Python-LDAP] Python-LDAP
`http://python-ldap.sourceforge.net`