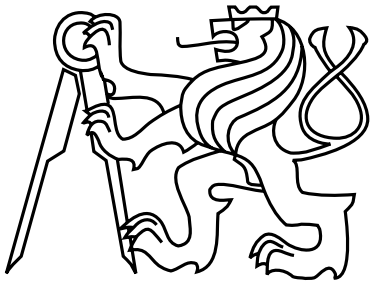
Na tomto místě bude oficiální zadání vaší práce

- Toto zadání je podepsané děkanem a vedoucím katedry,
- musíte si ho vyzvednout na studiijním oddělení Katedry počítačů na Karlově náměstí,
- v jedné odevzdané práci bude originál tohoto zadání (originál zůstává po obhajobě na katedře),
- ve druhé bude na stejném místě neověřená kopie tohoto dokumentu (tato se vám vrátí po obhajobě).

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Simulátor virtuální počítačové sítě Linux

Tomáš Pitřinec

Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

19. května 2010

Poděkování

Zde můžete napsat své poděkování, pokud chcete a máte komu děkovat.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Červeném Kostelci dne 25. 5. 2010 .....

Abstract

Translation of Czech abstract into English.

Abstrakt

Abstrakt práce by měl velmi stručně vystihovat její podstatu. Tedy čím se práce zabývá a co je jejím výsledkem/přínosem.
Očekávají se cca 1 – 2 odstavce, maximálně půl stránky.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Jednou z laboratorních úloh předmětu Počítačové sítě (Y36PSI) na FELu je postavení sítě mezi několika linuxovými a ciscovými počítači. Studenti, kteří tento úkol plní, nemají často s takovou činností žádnou osobní zkušenost, a tak během úlohy řeší různé banální problémy, kterým by se mohli vyhnout, kdyby měli možnost zkusit si nastavit podobou síť již před samotnou laboratorní úlohou. Mohl by se jim hodit simulátor, který by jednoduše spustili na svém počítači a na kterém by si mohli nastavování síťových parametrů na linuxu a ciscu zkusit. Právě návrhem a implementací takového síťového simulátoru počítačů s OS Linux se zabývá tato bakalářská práce.

1.1 Cíle práce

Cílem práce je v programovacím jazyce Java SE navrhnout a implementovat aplikaci, která umožní vytvoření virtuální počítačové sítě, pro potřeby předmětu Y36PSI. Z pohledu uživatele by aplikace měla vypadat stejně jako reálná síť. Uživatel spustí aplikaci v konsoli a pak se pomocí telnetu připojí k jejím jednotlivým virtuálním počítačům, podobně jako protokolem ssh k počítačům s OS Linux. Aplikace bude podporovat příkazy potřebné ke konfiguraci síťových rozhraní (ifconfig, ip address), směrování (route, ip route) a překladu adres (iptables -t nat). Pro ověření správnosti konfigurace sítě budou implementovány příkazy ping a traceroute.

Nastavenou konfiguraci sítě bude možné uložit do souboru a zase ji ze souboru načíst. Uživatel bude mít možnost vytvářet libovolné sítě s libovolným počtem počítačů typu linux nebo cisco tak, že infrastrukturu sítě napíše do konfiguračního souboru a pak ji z něho načte.

1.2 Rozdělení práce

Protože kompletní síťový simulátor pro počítače s Cisco IOS i OS Linux by přesahoval rozsah jedné bakalářské práce, byla práce na aplikaci rozdělena na tři části:

- **Jádro aplikace**
  Jedná se především o datové struktury virtuálního počítače, komunikaci s uživatelem

po síti a načítání a ukladání souborů. Na této části jsem spolupracoval se Stanislavem Řehákem.

- **Linuxová část**
  V této části je potřeba napsat parsery linuxových příkazů a zjistit, jak se chovají počítače s linuxem v síťové komunikaci a toto chování implementovat.

- **Ciscová část**
  Touto částí se tato práce nezabývá, zabývá se jí bakalářská práce Simulátor virtuální počítačové sítě Cisco Stanislava Řeháka.

1.3 Struktura práce

Tady bude popis struktry týhle práce, jako toho textu.

Kapitola 2

Existující řešení

Tady bude rešerše, až ji udělám.

Kapitola 3

Analysa a návrh aplikace

V této kapitole se zabývá analysou a návrhem aplikace jako celku. Shrnuji a analysuji požadavky, diskutuji zvolený jazyk a uživatelské rozhraní, popisuji architekturu aplikace, spolupráci s kolegou, který dělal druhou část aplikace, a postup implementace.

3.1 Požadavky na aplikaci

Nejprve shrnu všechny požadavky na mojí aplikaci.

3.1.1 Funkční požadavky

1. Vytvoření počítačové sítě založené na počítačích OS Linux.
2. Aplikace umožňuje konfiguraci rozhraní pomocí příkazů ifconfig a ip addr.
3. Aplikace obsahuje funkční směrování a umožňuje jeho nastavování pomocí příkazů route a ip route.
4. Aplikace implementuje překlad adres
5. Aplikace podporuje ukládání a načítání do/ze souboru.
6. Pro ověření správnosti jsou implementovány příkazy ping a traceroute.
7. K jednotlivým počítačům aplikace je možné se připojit pomocí telnetu.
8. Pomocí telnetu bude možno se připojit zároveň k více virtuálním počítačům.
9. Pomocí telnetu bude možno připojit se k jednomu počítači vícekrát najednou.

3.1.2 Nefunkční požadavky

1. Aplikace bude multiplatformní - alespoň pro operační systémy Windows a Linux
2. Aplikace musí být spustitelná na běžném[1] studentském počítači.
3. Aplikace by měla být co nejvěrnější kopií reálného počítače s Linuxem.

3.2 Analysa požadavků

3.2.1 Připojení pomocí telnetu

Jedním z funkčních požadavků mé aplikace je možnost připojit se k jednotlivým virtuálním počítačům pomocí protokolu telnet. Tento požadavek vypadá jednoduše, pokud pod pojmem Telnet chápeme jednoduchý protokol na přenos textových dat. Takový protokol ovšem neumožňuje doplňování příkazů a jejich historii, což je pro práci s počítačem, byť virtuálním, obrovské omezení. Oproti tomu, implementovat telnet protokol, jako NVT[2], kde se posílá a potvrzuje každý napsaný znak, by překračovalo rozsah této bakalářské práce. Proto jsem se rozhodl implementovat jen první možnost a na straně klienta řešit doplňování příkazů a jejich historii pomocí programu rlwrap, který funguje pod linuxem nebo přes cygwin i pod windows. Spouštění programu přes cygwin ve windows je sice velkou nevýhodou, ale neměl jsem jinou možnost. I tak ovšem základní požadavek, že s aplikací bude možno komunikovat pomocí telnetu, zůstává zachován, uživatel ovšem přijde o komfort, který mu nabízí možnost doplňování, editace a historie příkazů.

3.2.2 Podobnost simulátoru se skutečným linuxem

Aby byl simulátor využitelný pro výukové účely, musí být dostatečně podobný skutečnému linuxu, aby uživatel mohl věřit, že to, co funguje v simulátoru, bude fungovat i na skutečném linuxu a naopak. K tomu je ale potřeba implementovat jen ty příkazy, kterými se nastavují síťové parametry, a jen v takovém rozsahu, jaký je pro tyto výukové účely potřeba. Implementoval jsem tedy příkazy ifconfig, route, ping a traceroute, z příkazu ip jsem implementoval jeho podpříkazy addr a route. Pro potřeby nastavení překladu adres jsem implementoval malou část příkazu iptables. Aby uživatel mohl nastavovat některé hodnoty souborů v adresáři /proc, implementoval jsem ve velmi omezené míře i příkazy cat a echo, ovšem jen pro tyto soubory. Pro ukončení spojení je implementován příkaz exit. Pro potřeby simulátoru ale nebylo potřeba implementovat kompletní příkazy ifconfig nebo ip, ale jen tu jejich část, kterou se nastavují parametry rozhraní, jako IP, maska a další. O ostatních parametrech pak vetšinou simulátor vypíše, že ve skutečnosti sice existují, ale simulátorem zatím nejsou podporované.

[1] Slovem „běžné“ se myslí v podstatě jakýkoliv počítač, na kterém je možné nainstalovat prostředí Javy - Java Runtime Environment

[2] NVT – Network Virtual Terminal, česky: Síťový virtuální terminál; poskytuje standardní rozhraní příkazové řádky

3.3 Programovací jazyk a uživatelské rozhraní

3.3.1 Programovací jazyk

Aplikaci jsem se rozhodl programovat v programovacím jazyku Java z několika důvodů. Java je programovací jazyk, který nabízí velký programátorský komfort, stabilitu a zároveň možnost vytvořené aplikace používat pod různými operačními systémy, což je další z nefunkčních požadavků. Tento jazyk navíc disponuje hotovými knihovnami pro práci se sítí v balíčku java.net. Dalším důvodem je také to, že s programováním aplikací v Javě mám zatím asi největší zkušenosti.

3.3.2 Uživatelské rozhraní

Jak plyne ze zadání, uživatel se přihlašuje k jednotlivým virtuálním počítačům pomocí programu telnet, nemusím tedy vytvářet žádného speciálního klienta. S aplikací samotnou nebude uživatel nijak pracovat, jenom ji spustí se správným konfiguračním souborem a případně číslem výchozího portu, dále již bude nastavovat pouze jednotlivé virtuální počítače pomocí telnetu. Pro takovou aplikaci je nejlepším uživatelským rozhraním příkazová řádka, vytváření grafického uživatelského rozhraní by nemělo smysl.

3.4 Struktura aplikace

Aplikace se skládá ze dvou vrstev. Komunikační vrstva zajišťuje síťovou komunikaci s klientem, tedy odesílání a přijímání textových dat. Z velké části byla převzata z jiné práce, kterou jsme kdysi dělali jako domácí úkol na předmět Y36PSI. Aplikační vrstva je tvořena samotnou virtuální sítí. Avšak tyto vrstvy od sebe nejsou striktně odděleny. Nejprve si rozebereme druhou vrstvu.

3.4.1 Virtuální síť

Virtuální počítačová síť poskytuje mojí aplikaci především tyto funkcionality:

- Možnost konfigurace jednotlivých síťových prvků.
- Posílání paketů mezi síťovými prvky.

Skutečná počítačová síť se skládá ze síťových prvků různých druhů. Stejně tak i virtuální síť se bude skládat ze síťových prvků, které jsou interně reprezentovány objekty.

3.4.1.1 Síťové prvky

V laboratořích předmětu Y36PSI studenti nastavují pouze PC nebo směrovače na 3. (síťové) vrstvě ISO/OSI modelu[3]. Síťové prvky pracující na 2. vrstvě ISO/OSI modelu[4], switche a bridge se v laboratořích vůbec neuvažují. Proto i ve své práci uvažuji jediný druh síťových prvků - počítače s OS Linux.

[3] 3. vrstva ISO modelu, tzv. síťová vrstva, zajišťuje spojení mezi jakýmikoliv 2 uzly sítě.

[4] 2. vrstva ISO/OSI modelu, tzv. spojová nebo linková vrstva, zajišťuje spojení mezi dvěma sousedními systémy.

Virtuální počítač

Jedinými síťovými prvky mojí aplikace jsou počítače s operačním systémem Linux, které fungují jako směrovače na síťové vrstvě ISO/OSI modelu. Síťovou komunikaci skutečného počítače zajišťuje modul v jádře operačního systému. Počítač má několik síťových rozhraní, které se skládají z fysické části, kterou je síťový adaptér, a z části softarové, kterou je jeho konfigurace. Ke každému síťovému adaptéru, tzn. ke každému rozhraní, může být připojen maximálně jeden síťový kabel. Virtuální počítač i síťové rozhraní mají svou vlastní třídu.

Routovací a natovací tabulka

Jádro operačního systému směruje pakety podle tzv. routovací tabulky, což je datová struktura, která obsahuje cílové adresy a akce, které se mají vykonat s paketem poslaným na danou cílovou adresu. Tuto datovou strukturu musí obsahovat i moje virtuální síť, proto jsem pro ni udělal vlastní třídu. Její analýzou a implementací se budu zabývat v implementační části.
Aby virtuální počítače mohly překládat adresy přes ně procházejících paketů, potřebují, stejně jako skutečné počítače, další datovou strukturu, natovací tabulku. Touto datovou strukturou se ale tato práce nezabývá, vytvořil ji můj kolega Stanislav Řehák.

Příkazy

Uživatel musí mít možnost virtuální počítač nakonfigurovat, kvůli tomu přece aplikaci programuji. Ke konfiguraci mu mají sloužit standartní příkazy, které by použil při konfiguraci skutečného počítače. K virtuálnímu počítači se uživatel připojuje telnetem, o toto připojení se stará komunikační vrstva. Na vrstvě virtuální sítě ale probíha parsování a vykonávání těchto příkazů. O tom budu psát v další kapitole.

3.4.1.2 Infrastruktura sítě

Jak bylo napsáno dříve, jediným síťovým prvkem mojí aplikace je počítač s OS Linux. Ke každému rozhraní, může být připojen maximálně jeden síťový kabel, který, protože neuvažuji switche, je zapojen do jiného rozhraní nějakého počítače, nebo není zapojen nikam. Infrastruktura sítě je proto jednoznačně určena dvojicemi síťových rozhraní, která jsou propojena síťovým kabelem. Tuto infrastrukturu sítě je v konfiguračním souboru dobré oddělit od ostatní konfigurace, aby ji mohl uživatel lehce přečíst a pochopit, popř. změnit. Síťové rozhraní je jednoznačně identifikováno jménem počítače a jménem rozhaní.

3.4.1.3 Posílání paketů

Virtuální síť musí umět posílat virtuální pakety, aby uživatel pomocí příkazů ping nebo traceroute zjistil, jestli virtuální síť správně nakonfiguroval. Po virtuální síti, representované objekty, se samozřejmě posílají virtuální pakety, representované objekty třídy Paket. Virtuálni paket ponese potřebné informace stejně jako skutečný paket, akorát těch informací pro potřeby mojí aplikace není tolik. Posílání paketů v mé virtuální síti není prakticky rozděleno do OSI-ISO vrstev, IP paket se nebude balit do rámců linkové vrstvy. Virtuální pakety si

mezi sebou budou posílat, přijímat a přeposílat virtuální počítače pomocí speciálních metod k tomu určených. Je nutné,aby při stejné konfiguraci skutečné a virtuální počítačové sítě tato síť i stejně fungovala. To jest, aby ve virtuální počítačové síti do cíle došly právě ty pakety, které při stejné konfiguraci dojdou na síti skutečné. Více se anylýzou a implementací posílání paketů budu zabývat v samostatné kapitole.

3.4.2 Komunikační vrstva

Komunikační vrstva naší aplikace zajišťuje spojení aplikace s klientem. Z tohoto pohledu je aplikace klasickým síťovým serverem, který poslouchá na několika portech, přijímá spojení a zpracovává je. Uživatel po síti konfiguruje jednotlivé virtuální počítače, proto každý virtuální počítač musí poslouchat na jednom portu. O tuto komunikaci se nestará přímo virtuální počítač, má k tomu speciální třídu, i tak ale třída virtuálního počítače zasahuje do obou vrstev programu. Aby aplikace mohla poslouchat na více portech najednou, je nutné vytvořit více vláken, každý virtuální počítač tedy poběží v samostatném vláknu. Jak plyne z posledního funkčního požadavku, musí jeden virtuální počítač umět zpracovat i více spojení najednou, jako i na reálný linuxový počítač je možné se připojit k několika jeho terminálům pomocí protokolu ssh nebo telnet. Proto je nutné, aby vlákno, které poslouchá na portu, pro příchozí spojení vytvořilo jiné vlákno, které spojení obslouží, a samo dále poslouchalo na určeném portu.

3.5 Spolupráce na aplikaci

Na aplikaci jsem spolupracoval se svým kolegou Stanislavem Řehákem, který implementuje její druhou část - simulátor Cisca. Spolupráce však přesahuje jen tuto oblast a zasahuje také do společného jádra aplikace.
V této práci chci popisovat především moji část výsledného simulátoru, rád bych však upozornil, že když zde popisuji implementaci nějaké části programu, neznamená, že jsem ji celou implementoval sám. U každé třídy v kódu je napsáno, kdo je jejím autorem. Pokud jsme na třídě pracovali oba, je autorství uvedeno u jejích metod.
Architektura aplikace je založena na oboustranné dohodě. Komunikační vrstvu jsme z velké části přejali z domácího úkolu na předmět Y36PSI, server Karel, který jsme programovali na podzim roku 2008. Pro potřeby naší aplikace jsme potom tuto vrstvu společně upravili. Je těžké, připsat autorství této vrstvy jednomu nebo druhému z nás, ale vzhledem k tomu, že se jedná a sice nutnou, ale málo rozsáhlou část aplikace, to dle mého názoru není nutné. Síťové rozhraní je stejné pro oba počítače, jeho třídu jsme dělali společně. Oba typy počítačů mají mnoho společného, ale v něčem se liší. Proto jsme vytvořili třídu abstraktní počítač, jejíž autorem jsem já, a od ní jsme pak dědili každý svoji vlastní třídu pro virtuální počítač. Já jsem autorem všeho, co souvisí s posíláním paketů a routovací tabulky, kterou však kolega nemohl přímo využít. Natovací tabulku programoval kolega, stejně tak i veškeré ukládání do souboru a načítání z něj, a třídu Main. Důležitou třídu IpAdresa jsme programovali společně, autorství je uvedeno u jejích jednotlivých metod, stejně tak i abstraktní příkaz a abstraktní parser příkazů, kde jsou vyčleněny společné metody pro parsování a vykonávání příkazů.

3.6 Postup implementace

Samotnou implementaci jsem zahájili komunikační vrstvou, následně jsme naprogramovali základní datové struktury, jako třídy pro virtuální počítač, síťové rozhraní, IP adresu ap. Pokračoval jsem implementací linuxových příkazů. Nejdříve jsem zpracoval příkaz ifconfig, abych mohl nastavovat virtuální síťová rozhraní. Po něm jsem zpracoval příkaz route a zároveň s ním také routovací tabulku. Po těchto dvou základních příkazech jsem pracoval na posílání paketů mezi počítači. Když jsem to měl hotové, implementoval jsem ostatní příkazy jako ping, traceroute, ip, iptables, echo a cat.
V implementační části této práce se nepopisuji jednotlivé části programu ve stejném pořadí, jako jsem je implementoval.

Kapitola 4

Implementace virtuální sítě

V této kapitole se zabývám analysou a implementací jednotlivých částí aplikace. Rozebírám zde nejprve analýzu a implementaci třídy `IpAdresa`, dále implementaci virtuálního počítače, analýzu a implementaci routovací tabulky a analýsu a implementaci posílání paketů. Nezabývám se zde analysou a implementací jednotlivých příkazů, vzhledem k rozsáhlosti tohoto tématu jsem ho vyčlenil do zvláštní kapitoly, která následuje za touto kapitolou.

4.1 IP adresa

Třída `IpAdresa` je sice jen jednou z mnoha tříd, vzhledem k jejímu významu ji ale v následujících odstavcích popíšu podrobněji.

4.1.1 Analýza

Protože simulátor se zabývá především simulací síťové vrstvy ISO/OSI modelu, je síťová adresa počítače, tzv. IP adresa velmi často používanou datovou strukturou, pro kterou se jistě vyplatí mít speciální třídu. Ta se v v aplikaci jmenuje `IpAdresa` a patří do balíčku `datoveStruktury`. Je používána jako parametr síťového rozhraní, jako prefix v routovací tabulce, jako zdrojová a cílová adresa v paketech. Při bližším pohledu je zřejmé, že kontext jejího použití se v těchto případech částečně liší. Například pro posílání paketů je nutné, aby paket obsahoval zdrojovou a cílovou adresu i s portem. Port by samozřejmě nemusel být součástí adresy, ukázalo se to však jako jednoduší a pro posílání paketů přehlednější možnost. Pro IP adresu jako parametr rozhraní je naopak port zcela nesmyslný parametr, nutně ale potřebuje parametr pro síťovou masku, která je naproti tomu nesmyslná pro posílání paketů. Paket posílám na IP adresu, ne na adresu s maskou.

4.1.2 Vnitřní reprezentace

Přes tyto rozdíly jsem se rozhodl vytvořit pro IP adresu jednu třídu, která má parametry `adresa`, `maska` a `port`, přičemž pro danou situaci nepotřebné parametry prostě ignoruji. Protože Java neobsahuje žádný 32-bitový bezeznaménkový datový typ, jsou parametry adresa a maska vnitřně reprezentovány 32-bitovým integerem, který ale obsahuje bity skutečné adresy,

jeho číselná hodnota není důležitá. Operace s nimi se provádí především pomocí bitových operátorů. Parametr port je normální integer.

4.1.3 Veřejné metody

`IpAdresa` má konstruktory, aby ji bylo možné vytvořit ze `Stringu`, s maskou zadanou jako `String`, `Integer` nebo v jednou řetězci s adresou. Adresu je možné převést na String nebo porovnat s jinou adresou mnoha různými způsoby, například jen podle adresy, adresy s portem, adresy s maskou nebo čísla sítě. `IpAdresa` umí vrátit své číslo sítě nebo broadcast jako jinou `IpAdresu`. O těchto metodách se zde nerozepisuji podrobně, v kódu jsou dobře okomentované.

4.2 Virtuální počítač

Virtuální počítač je základním stavebním prvkem naší aplikace. Pracuje na obou jejích vrstvách. Na vrstvě komunikační přijímá a zpracovává příchozí spojení, na aplikační vrstvě, tj. na vrstvě virtuální sítě přijímá, posílá a přeposílá pakety. Posíláním paketů se zabývám až v posledním odstavci této kapitoly, zde proberu komunikační vrstvu počítače a jeho rozhraní.

Protože linuxový a ciscový počítač, který dělal kolega, mají mnoho společného, vytvořil jsem abstraktní třídu `AbstraktniPocitac`, který je předkem počítačů obou typů. Třída `LinuxPocitac` má ale jen jednu metodu, která se týká posílání paketů, proto ji zatím ignoruji.

Všechny virtuální počítače jsou vytvářeny v rámci inicialisace aplikace dle konfiguračního souboru na začátku jejícho běhu, za chodu aplikace není již možné další počítač přidat nebo nějaký odebrat. O komunikaci s uživateli se stará třída Komunikace. Počítač si drží seznam svých síťových rozhraní, svoji routovací tabulku a natovací tabulku. Má jediný konstruktor, kde je mu zadáno jméno (pro přehlednost) a port, na kterém má být dostupný pro uživatele.

4.2.1 Komunikace s počítačem

V konstruktoru abstraktního počítače je volán konstruktor jeho parametru třídy `Komunikace`. Tato třída se stará o veškerou komunikaci s uživatelem. Je potomkem třídy Thread. Běží ve vlastním vlákně, které se startuje v jejím konstruktoru, a poslouchá na portu, který jí byl zadán. Pro každé nové příchozí spojení vytvoří instanci třídy konsole, která spojení obslouží, aby komunikce mohla dále poslouchat na portu a zpracovávat další spojení. Třída Konsole je také potomkem třídy Thread. Obsluhuje jedno telnetové připojení. Drží si instanci třídy ParserPrikazu z balíčku Prikazy (o něm v následující kapitole). Přijímá textová data od uživatele až po enter (sekvence `\r\n`), tedy vlastně načítá data po řádcích. Každý řádek, který ji uživatel pošle, předá parseru na zpracování a pak sama pošle uživateli prompt. Parseru poskytuje metody pro posílání textových dat uživateli. Pro uživatele tak komunikace s touto Konsoli vypada stejně jako práce s příkazovou řádkou na skoutečném počítači.

4.2.2 Síťové rozhraní

Z hlediska infrastruktury sítě jsou základními prvky počítače jeho síťová rozhraní. Ty si počítač drží v seznamu. Jsou vytvořeny při parsovaní konfiguračního souboru a během běhu

aplikace je nelze nijak měnit, přidávat nebo mazat. Třída `SitoveRozhrani` má svoje jméno a fysickou (mac) adresu. Protože v naší aplikaci není implementován arp[1] protokol, mac adresa nemá jiný význam, než že se vypisuje příkazy jako např. ifconfig.

Skutečné síťové rozhraní může mít více adres. Tato možnost však není v předmětu PSI využívána, proto jsem ji neimplementoval. Znamenalo by to totiž poměrně velké problémy v posílání paketů. Musel bych složitě zjišťovat, kdy se paket odešle s jakou síťovou adresou, pokud je jich na daném rozhraní více. Pro potřeby statického natování především na ciscovém routeru je ale nutné mít na rozhraní více adres.[2] Proto má třída `SitoveRozhrani` seznam IP adres, ale jeho první adresa je privilegovaná. Každý paket, který je přes dané rozhraní posílán, má jako odchozí adresu adresu právě tohoto rozhraní. Tuto jedinou adresu lze nastavovat a vypisovat. Ostatní adresy jsou přidávány jen pro potřeby statického natování. Pokud rozhraní nemá nastavenou žádnou adresu, je první (privilegovaná) adresa `null`.

4.3 Routovací tabulka

Počítače směrují pakety podle tzv. routovací, neboli směrovací, tabulky. „Routovací tabulka je datový soubor uložený v RAM paměti, který je používán k uchovávání informací ohledně přímo připojených i vzdáleně připojených sítích. Její obsah napovídá routeru, kterým rozhraním je možno nejoptimálněji dosáhnout cílové sítě.“[12]. V této části se zabývám nejprve analýzou routovací tabulky na skutečném linuxu a potom popisuji její implementaci v simulátoru.

Třída `RoutovaciTabulka` měla být původně stejně použitelná pro linux i pro cisco. Až po tom, co jsem jí implementoval, kolega zjistil, že pro potřeby Cisca není tato třída bez úprav použitelná. Proto implementoval třídu `CiscoWrapper`, která obaluje třídu `RoutovaciTabulka` a dodává jí funkce potřebné pro cisco. To však není obsahem mojí práce.

4.3.1 Analýza routovací tabulky na skutečném počítači

4.3.1.1 Struktura tabulky

V řádcích routovací tabulky jsou záznamy pro jednotlivé sítě. Každý záznam má tyto parametry:

- adresát - IP adresa s maskou, pro kterou je tento záznam platný
- brána - IP adresa počítače, na který se má paket poslat. Tento sloupec nemusí být vždy vyplněn.
- příznaky - O těch více píšu v samostatné části.
- metrika - Jedno z kriterií priority.

[1] Address Resolution Protocol se v počítačových sítích s IP protokolem používá k získání ethernetové MAC adresy sousedního stroje z jeho IP adresy. Používá se v situaci, kdy je třeba odeslat IP datagram na adresu ležící ve stejné podsíti jako odesilatel. Data se tedy mají poslat přímo adresátovi, u něhož však odesilatel zná pouze IP adresu. Pro odeslání prostřednictvím např. Ethernetu ale potřebuje znát cílovou ethernetovou adresu.[13]

[2] Více o natování v bakalářské práci mého kolegy Stanislava Řeháka

- rozhraní - Rozhraní, přes které se paket posílá.

Parametr metrika není pro výukové účely potřeba, proto se jím již dále nezabývám.

Pro lepší představu zde vkládám routovací tabulku tak, jak je vypsána příkazem `route -n`:

```
Adresát         Brána           Maska           Přízn Metrik Odkaz  Užt Rozhraní
147.32.125.128  0.0.0.0         255.255.255.128 U     1      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth0
0.0.0.0         147.32.125.129  0.0.0.0         UG    0      0        0 eth0
```

4.3.1.2 Adresát

V hořejším výpisu tabulky pomocí příkazu `route` se adresáta týkají 2 sloupce, sloupec Adresát a sloupec Maska, ve kterém jsou vypsány masky k IP adresám uvedeným ve sloupci Adresát. Tyto IP adresy s maskami jsou vždy číslem sítě a reprezentují všechny adresy, které do této sítě patří. Tak například adresát 0.0.0.0/0 reprezentuje úplně všechny IP adresy, adresát 147.32.125.128/25 reprezentuje adresy v rozmezí 147.32.125.128 až 147.32.125.255, adresát 1.1.1.1/32 reprezentuje jedinou adresu 1.1.1.1 a adresát 192.168.1.0/24 reprezentuje všechny adresy, které začínají byty 192.168.1.x. Adresáta 147.32.125.128/24 nelze zadat, protože číslo této sítě je 147.32.125.0/24.

4.3.1.3 Příznaky

Záznam routovací tabulky má několik příznaků. Má vždy minimálně jeden příznak, může mít ale všechny 3 příznaky najednou. Zde je jejich popis:

- Příznak `U` znamená, že záznam obsahuje rozhraní. Protože záznam bez vyplněného rozhraní není možné zadat, má tento příznak každý záznam.
- Záznam má příznak `G`, jestliže je vyplněn sloupec brána.
- Příznak H znamená, že adresátem daného záznamu je jeden počítač, tzn. adresát má masku 255.255.255.255.

Příznak `H` jen informuje, že adresát není sítí ale jediným počítačem, není tedy nijak důležitý. Podle příznaků existují 2 typy záznamů, záznamy s příznakem `U` a záznamy s příznakem `UG`. Tyto typy se liší jak při přidávání nových záznamů do routovací tabulky, tak při posílání paketu podle tohoto záznamu. Posílá-li počítač paket podle záznamu U, není z tohoto záznamu zřejmé, jakému sousednímu počítači (na linkové vrstvě) se má paket poslat. Počítač se tedy pokusí poslat paket přímo na cílovou IP adresu uvedenou v paketu. Posílá-li se paket podle záznamu UG, posílá se na adresu brány uvedenou v záznamu. Více se této problematice věnuji v kapitole o posílání paketů.

4.3.1.4 Přidávání záznamů a jejich řazení

Routovací tabulka nesmí obsahovat 2 stejné záznamy. Za stejné záznamy se považují záznamy, které mají stejného adresáta včetně masky, stejné rozhraní a stejnou bránu.

Záznam typu `U` lze přidat vždycky. Záznam typu `UG` lze přidat jen pod podmínkou, že jeho brána je v okomžiku přidání dosažitelná záznamem typu `U`. Tím je možné dosáhnout zajímaveho chování: Když do routovací tabulky přidám defaultní routu [3] záznamu typu U, můžu pak přidat routu na jakoukoliv síť v internetu se záznamem UG. Když potom smažu původní defaultní routu, můžu posílat pakety pouze na tu síť s příznakem `UG` a na počítač v mé síti paket neodešlu. Zde uvádím příklad:

```
root: /home/neiss# route add default eth0
root: /home/neiss# route add -net 89.190.94.0/24 gw 89.190.94.1
root: /home/neiss# route del default
root: /home/neiss# route
Směrovací tabulka v jádru pro IP
Adresát         Brána           Maska           Přízn Metrik Odkaz  Užt Rozhraní
89.190.94.0     89.190.94.1     255.255.255.0   UG    0      0        0 eth0
root: /home/neiss# ping -c1 89.190.94.58
PING 89.190.94.58 (89.190.94.58) 56(84) bytes of data.
64 bytes from 89.190.94.58: icmp_seq=1 ttl=53 time=14.1 ms

--- 89.190.94.58 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 14.141/14.141/14.141/0.000 ms
root: /home/neiss# ping -c1 147.32.125.129 # toto je adresa brány, přes kterou šel minul
connect: Network is unreachable
```

Tento pokus funguje ale jen tehdy, pokud moje brána, v tomto případě 147.32.125.129 je cisco (viz část o posílání paketů).

Záznamy se v tabulce řadí podle masky adresáta. Nahoře jsou záznamy s nejdelší maskou, tzn. záznamy nejkonkrétnější. Pokud vkládám více záznamů se stejnou maskou, chová se routovací tabulka naprosto nepředvídatelně, což je vidět na následujícím příkladě:

```
node-4:/home/dsn# route add -net 1.1.5.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.6.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.7.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.8.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.9.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.10.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.11.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.12.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.13.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.14.0/25 dev eth0
node-4:/home/dsn# route add -net 1.1.15.0/25 dev eth0
```

[3] Defaultní routa je záznam platný pro ccelý internet, jeho adresátem je 0.0.0.0/0

```
node-4:/home/dsn# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
1.1.10.0        *               255.255.255.128 U     0      0        0 eth0
1.1.11.0        *               255.255.255.128 U     0      0        0 eth0
1.1.8.0         *               255.255.255.128 U     0      0        0 eth0
1.1.9.0         *               255.255.255.128 U     0      0        0 eth0
1.1.14.0        *               255.255.255.128 U     0      0        0 eth0
1.1.6.0         *               255.255.255.128 U     0      0        0 eth0
1.1.15.0        *               255.255.255.128 U     0      0        0 eth0
1.1.7.0         *               255.255.255.128 U     0      0        0 eth0
1.1.12.0        *               255.255.255.128 U     0      0        0 eth0
1.1.13.0        *               255.255.255.128 U     0      0        0 eth0
1.1.5.0         *               255.255.255.128 U     0      0        0 eth0
```

Podle jakého algoritmu jsou nové záznamy zařazovány je mi opravdu záhadou.

4.3.1.5 Mazání záznamů

Smazat je možno jakýkoliv záznam tabulky. Pro mazání záznamů je potřeba zadat správně minimálně adresáta záznamu. Pokud pak existuje více záznamů se zadanými parametry, smaže se první z nich. Smazání jakéhokoliv záznamu nijak neovlivní ostatní záznamy.

4.3.1.6 Použití pro směrování

Ke směrování paketu se použije první záznam odpovídající cílové adrese. To znamená, že bude-li v routovací tabulce dané adrese odpovídat více záznamů, použije se ten nejvíce nahoře. Protože záznamy jsou řazeny podle délky síťové masky, je vrácený záznam ten nejkonkrétnější.

4.3.2 Implementace routovací tabulky v simulátoru

Routovací tabulka je imlementována třídou `RoutovaciTabulka`, jejíž odkaz si drží `AbstraktniPocitac` a podle ní směruje pakety.

4.3.2.1 Vnitřní reprezentace

Tabulka je vnitřně reprezentována seznamem objektů typu `Zaznam`, který reprezentuje jeden záznam, tj. řádek tabulky. Záznam routovací tabulky má v simulátoru jen tyto parametry: adresát, brána a rozhraní, které fungují tak, jak bylo popsáno v odstavci o analyse. Parametry adresát a brána jsou typu `IpAdresa`, parametr rozhraní je typu `SitoveRozhrani`. Parametr záznamy není vůbec potřeba. Příznak `U` musí mít záznam vždy, příznak `H` má právě tehdy, když adresát má masku 255.255.255.255, a příznak `U` má záznam právě tehdy, když má vyplněnou položku brána, proto ani ten není potřeba.

4.3.2.2 Přidávání, mazání a řazení záznamů

Záznamy jsou přidávány pomocí 2 metod se stejným názvem `pridejZaznam` ale jinými parametry. Jedna přidává záznam typu `U`, druhá, která má navíc paramatr brána, záznam typu `UG`. U obou se kontroluje, jestli tabulka již stejný záznam neobsahuje, u té druhé se navíc kontroluje dosažitelnost brány, jak bylo popsáno v analýze.

Záznamy se samozřejmě řadí podle masky jako v reálné tabulce, záznamy se stejnou maskou se ale vloží vždy nad původní záznam. Tak jsou novější záznamy vždy nahoře. Zmatečné řazení reálné tabulky jsem samozřejmě neimplementoval.

Pro mazání má `RoutovaciTabulka` metodu `SmazZaznam`, funguje stejně jako na reálném počítači.

Navíc obsahuje `RoutovaciTabulka` ještě metodu `pridejZaznamBezKontrol`, která je využívána při vytváření počítače z konfiguračního souboru, jinde se nepoužívá.

4.3.2.3 Použití při směrování

K samotnému směrování slouží metoda `najdiSpravnejZaznam`, která vrací celý řádek routovací tabulky. Funguje stejně jako na reálném počítači.

4.4 Posílání paketů

Posílání paketů v naší aplikaci slouží k tomu, aby uživatel pomocí příkazů `ping` a `traceroute` mohl ověřit, zda síť správně nakonfiguroval. Bez této části by naší aplikaci nebylo možné nazvat síťovým simulátorem.

4.4.1 Referenční model ISO/OSI

„Referenční model ISO/OSI vypracovala organizace ISO jako hlavní část snahy o standardizaci počítačových sítí.“[17]. Dle tohoto modelu probíhá síťová komunikace v sedmi vrstvách, z níž každá poskytuje přesně definované funkce a komunikuje jen s vrstvou sousední. Každá vrstva má svůj formát přenášených dat, obyčejně dělených do bloků. Pro moji aplikaci jsou důležité vrstvy 2 - 4, tzn. spojová, síťová a transportní vrstva.

4.4.1.1 Spojová vrstva

Spojová nebo linková vrstva[4] „poskytuje spojení mezi dvěma sousedními systémy.“[17]. Tuto vrstvu zajišťuje na skutečné síti v laboratoři technologie Ethernet[14], ke zjištění fysické adresy sousedního systému se používá protokol ARP[13]. Sousednímy systémy se zde rozumí počítače zapojené do stejné sítě. Blok dat na linkové vrstvě se nazývá rámec. Vzhledem k tomu, že simulátor neobsahuje žádné switche, zajištuje tato vrstva v naší aplikaci spojení mezi 2 počítači propojenými kabelem.

[4] Více v [16]

4.4.1.2 Síťová vrstva

Síťová vrstva „se stará o směrování v síti a síťové adresování. Poskytuje spojení mezi systémy, které spolu přímo nesousedí.“[17] Zajišťuje spojení meze jakýmikoliv dvěma uzly sítě. Obyčejně je realizována protokolem IP[5]. Blok dat na síťové vrstvě se nazává paket.

4.4.1.3 Transportní vrstva

Transportní vrstva „zajišťuje přenos dat mezi koncovými uzly“[17]. Pro potřeby příkazů ping a traceroute je tato vrstva realisována protokolem ICMP[6]. Blok dat se v transportní vrstvě nazývá datagram.

4.4.1.4 Datové bloky

Datový blok jakékoliv vrstvy se skládá z hlavičky, která obsahuje režijní informace té vrstvy, a z datové části, která obsahuje samotná data, ale i hlavičky vyšších vrstev. Tak například datagram protokolu ICMP je obalen hlavičkou IP na síťové vrstvě a hlavičkou protokolu Ethernet na spojové vrstvě.

4.4.2 Implementace třídy Paket

Pro můj síťový simulátor by bylo nesmyslné implementovat posílání paketů včetně jejich zabalování do datových bloku ruzných vrstev, tak jak je to popsáno v předchozím odstavci 4.4.1.4. Proto jsem vytvořil třídu `Paket`, která obsahuje všechny potřebné informace z datových bloků všech tří vrstev. `Paket` má parametry síťové vrstvy, jako zdrojovou a cílovou adresu a `ttl`, a parametry transportní vrstvy, jako typ a kód protokolu ICMP. Pro snadnější implementaci příkazu ping má parametr `cas`, kam se ukládá náhodně generovaný čas běhu paketu, který vypisuje příkaz ping. Protože na jednom virtuálním počítači může běžet více příkazů `ping` najednou, nese paket i odkaz na příkaz, který ho poslal, aby ho tento příkaz mohl také po jeho návratu zpracovat.

Typy a kódy ICMP paketů jsou označeny stejně jako ve skutečnosti:

- typ 0 - ozvěna (icmp reply) - odpověď na požadavek icmp request
- typ 3 - vyslaný paket nemohl být doručen
- typ 8 - žádost o ozvěnu (icmp request)
  - kód 0 - network unreachable (nedosažitelná síť)
  - kód 1 - host unreachable (nedosažitelná adresa)
- typ 11 - `ttl` vypršelo

[5] Internet Protocol
[6] Internet Control Message Protocol, více v [15]

4.4.3 Chování reálného počítače při posílání paketů

4.4.3.1 Transportní vrstva - protokol ICMP

V souboru `/proc/sys/net/ipv4/icmp_echo_ignore_all` je nastaveno, jestli počítač odpovídá na dotazy icmp reply. Pokud je v tomto souboru `0`, počítač na dotazy odpovídá. Toto je defaultní nastavení, proto jsem v simulátoru tento problém vůbec neřešil a počítač odpovídá na icmp request vždy.

4.4.3.2 Síťová vrstva - protokol IP

Přeposílání paketů

Počítač preposílá pakety jenom tehdy, pokud je v souboru `/proc/sys/net/ipv4/ip_forward` jednička, jinak ne. Toto nastavení ale již není defaultní, proto ho musím v simulátoru implementovat. Proměnná `ip_forward` je parametrem virtuálního počítače a je nastavována pomocí příkazu `echo`.

Směrování paketů

Pakety jsou směrovány podle routovací tabulky, kde se vybere první záznam odpovídající cílové adrese paketu, jak je popsáno v 4.3.1.6. Pokud v routovací tabulce nebyl nalezen žádný záznam pro cílovou adresu paketu, pošle se na jeho zdrojovou adresu paket `icmp_net_unreachable`[7].

ttl

Každý prvek, který pracuje i na síťové vrstvě sníží procházejícím paketům hodnotu `ttl`. Pokud po tomto snížení dosáhne hodnota `ttl` nuly, pošle počítač na zdrojovou adresu paketu zprávu `icmp_ttl_exceeded`[8].

Next hop

Předtím než síťová vrstva předá odeslání paketu linkové vrstvě, musí pro tento paket zjistit tzv. next hop, neboli sousední adresu. „Next hop je sousední směrovač, na který je paket poslán nebo přeposlán z daného směrovače na své cestě k cíli.“[?] Tato adresa je velmi důležitá pro linkovou vrstvu, která paket posílá právě na tuto adresu. Pokud je paket směrován podle záznamu typu `UG` (viz 4.3.1.3), je adresa next hop uvedena ve sloupci brána routovací tabulky. Pokud je paket směrován podle záznamu typu `U`, je adresa next hop cílová adresa paketu. Podle routovací tabulky

Adresát	Brána	Maska	Přízn	Metrika	Rozhraní
147.32.125.128	0.0.0.0	255.255.255.128	U	0	eth0
0.0.0.0	147.32.125.129	0.0.0.0	UG	0	eth0

je pro pakety směrované podle prvního záznamu next hop rovná jejich cílové adrese. Pro pakety směrované podle druhého záznamu je next hop `147.32.125.128`. Záznamy typu `U` jsou tak používány pro počítače v mé síti, které jsou dosažitelné přímo bez jakéhokoliv mezilehlého směrovače. Záznamy `UG` jsou používány pro počítače, na které je paket posílán přes jeden nebo více směrovačů.

[7] Tj ICMP paket typu 3 kódu 0
[8] Tj. icmp paket typu 11

4.4.3.3 Linková vrstva

Na linkové vrstvě se rámce přeposílají jen mezi sousedními počítači. Běžně ji zajišťuje protokol ethernet. IP adresu sousedního počítače, na kterou má rámec poslat, dostane od síťové vrstvy, musí ji přeložit na fysickou (MAC[9]) adresu, což dělá protokolem ARP. Protokol ARP vyšle ethernetový rámec na všechny okolní počítače žádost, která obsahuje zadanou IP adresu. Pokud některý z počítačů má tokovou IP adresu, původnímu počítači pošle zpátky svoji fysickou adresu a ten pak může odeslat paket. Aby počítač nemusel fysickou adresu zjišťovat při odesílání každého rámce, ukládá si v datové struktuře nazvané ARP tabulka záznamy s IP adresami, které již dříve překládal.

Pokud počítač nemůže linkovou vrstvou odeslat rámec (paket), například proto, že počítač s takovou adresou na síti neexistuje, pošle původnímu odesílateli ICMP paket `net unreachable`[10].

Počítač s operačním systémem linux odpovídá na ARP dotazy jen tehdy, když má požadovanou IP adresu nastavenou na svém rozhraní. V tom se liší od Cisca, které na ARP dotaz odpoví i v případě, že adresu na svém rozhraní nemá, ale ví, kam má paket dále směřoav, tzn. má pro požadovanou adresu záznam v routovací tabulce, a naopak na ARP dotaz neodpoví v případě, že nemá v routovací tabulce záznam pro IP adresu, odkud dotaz přišel. Pro lepší pochopení přepisuji podmínku ještě v jazyce booleanovských výrazů v javě: Cisco odpoví na ARP dotaz právě tehdy, když:
Má záznam v routovací tabulce pro počítač, který ARP dotaz posílá `&&` (Má nastavenou požadovanou IP adresu `nebo` Má záznam v routovací tabulce pro cílovou adresu posílaného paketu)

4.4.4 Implementace v simulátoru

Pakety jsou mezi počítači posílány vzájemným voláním metod, které si mezi sebou předávají objekt typu `Paket`. Všechny tyto metody jsou ve třídě `AbstraktniPocitac`, protože to jsou právě virtuální počítače, které si mezi sebou pakety posílají. Jen metoda `prijmiEthernetove` je sice v `AbstraktniPocitac` zadeklarována a implementována v jeho potomcích.

Metody virtuálního počítače pro posílání paketů jsem bylo rozumné implementovat dle vrstev. Je to přehlednější, než kdybych měl jednu metodu pro přijmutí paketu a je to dobré i vzhledem k tomu, že linux a cisco se v některých případech liší a to jen na některých vrstvách. Metody jedné vrstvy tak volají jen metody vrstvy sousední, jako na reálné síti jedna vrstva komunikuje jen s vrstvami sousedními.

4.4.4.1 Linková vrstva

ARP protokol na zjišťování fysických adres nemusel být implementován, protože v síti nejsou žádné switche a v linkové vrstvě tak není potřeba žádné směrování. Aby ale byly splněny podmínky doručení nebo nedoručení paketu uvedené v 4.4.3.3, byly vytvořeny metody `odesliEthernetove` a `prijmiEthernetove`, které si mezi sebou předávají pakety tak, aby byly tyto podmínky splněny. Metoda `prijmiEthernetove` na linuxu nepřijme paket, pokud nesouhlasí očekávaná IP adresa, jako kdyby neodpověděla na ARP dotaz.

[9] Media Access Control
[10] Tj ICMP paket typu 3 kódu 1

4.4.4.2 Síťová vrstva

Na síťové vrstvě si pakety předávají metody `odesliNovejPaket`, `preposliPaket` a `prijmiPaket`. Tyto metody směrují pakety a provádí překlad adres.

4.4.4.3 Transportní vrstva

Do transportní vrstvy patří více metod, které slouží především k posílání různých typů ICMP paketů.

Kapitola 5

Popis problému, specifikace cíle

- Popis řešeného problému, vymezení cílů DP/BP a požadavků na implementovaný systém.
- Popis struktury DP/BP ve vztahu k vytyčeným cílům.
- Rešeršní zpracování existujících implementací, pokud jsou známy.

Kapitola 6

Analýza a návrh řešení

Analýza a návrh implementace (včetně diskuse různých alternativ a volby implementačního prostředí).

Kapitola 7

Realizace

Popis implementace/realizace se zaměřením na nestandardní části řešení.

Kapitola 8

Testování

- Způsob, průběh a výsledky testování.
- Srovnání s existujícími řešeními, pokud jsou známy.

Kapitola 9

Závěr

- Zhodnocení splnění cílů DP/BP a vlastního přínosu práce (při formulaci je třeba vzít v potaz zadání práce).
- Diskuse dalšího možného pokračování práce.

Literatura

[1]

[2]

[3]

[4]

[5]

[6] .

[7] .

[8]

[9] [online].

[10] [online].

[11]

[12] Martin Mikulec. *Směrování v síti* [online]. 2010. [cit. 18. 5. 2010]. Dostupné z: <http://owebu.bloger.cz/PC-site/Smerovani-v-siti-routovaci-tabulka-1-dil>.

[13] Přispěvatelé Wikipedie. *Address Resolution Protocol* [online]. 2010. [cit. 18. 5. 2010]. Dostupné z: <http://cs.wikipedia.org/wiki/Address_Resolution_Protocol>.

[14] Přispěvatelé Wikipedie. *Ethernet* [online]. 2010. [cit. 19. 5. 2010]. Dostupné z: <http://cs.wikipedia.org/wiki/Ethernet>.

[15] Přispěvatelé Wikipedie. *Internet Control Message Protocol* [online]. 2010. [cit. 19. 5. 2010]. Dostupné z: <http://cs.wikipedia.org/wiki/ICMP>.

[16] Přispěvatelé Wikipedie. *Linková vrstva* [online]. 2010. [cit. 19. 5. 2010]. Dostupné z: <http://cs.wikipedia.org/wiki/Spojová_vrstva>.

[17] Přispěvatelé Wikipedie. *Referenční model ISO/OSI* [online]. 2010. [cit. 19. 5. 2010]. Dostupné z: <http://cs.wikipedia.org/wiki/Referenční_model_ISO/OSI>.

Dodatek A

Testování zaplnění stránky a odsazení odstavců

Tato příloha nebude součástí vaší práce. Slouží pouze jako příklad formátování textu.

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která

se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká

pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Dodatek B

Pokyny a návody k formátování textu práce

Tato příloha samozřejmě nebude součástí vaší práce. Slouží pouze jako příklad formátování textu.

Používat se dají všechny příkazy systému LaTeX. Existuje velké množství volně přístupné dokumentace, tutoriálů, příruček a dalších materiálů v elektronické podobě. Výchozím bodem, kromě Googlu, může být stránka CSTUG (Czech Tech Users Group) [1]. Tam najdete odkazy na další materiály. Vetšinou dostačující a přehledně organizovanou elektronikou dokumentaci najdete například na [6] nebo [7].

Existují i různé nadstavby nad systémy TeX a LaTeX, které výrazně usnadní psaní textu zejména začátečníkům. Velmi rozšířený v Linuxovém prostředí je systém Kile.

B.1 Vkládání obrázků

Obrázky se umísťují do plovoucího prostředí `figure`. Každý obrázek by měl obsahovat **název** (`\caption`) a **návěští** (`\label`). Použití příkazu pro vložení obrázku `\includegraphics` je podmíněno aktivací (načtením) balíku graphicx příkazem `\usepackage{graphicx}`.

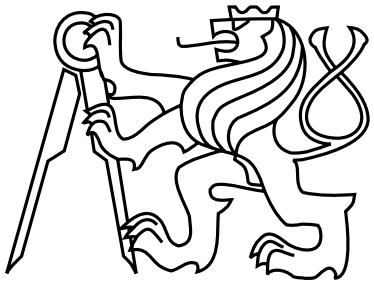
Budete-li zdrojový text zpracovávat pomocí programu `pdflatex`, očekávají se obrázky s příponou `*.pdf`[1], použijete-li k formátování `latex`, očekávají se obrázky s příponou `*.eps`.[2]

Příklad vložení obrázku:

```
\begin{figure}[h]
\begin{center}
\includegraphics[width=5cm]{figures/LogoCVUT}
\caption{Popiska obrazku}
\label{fig:logo}
```

[1] pdflatex umí také formáty PNG a JPG.

[2] Vzájemnou konverzi mezi snad všemi typy obrazku včetně změn vekostí a dalších vymožeností vám může zajistit balík ImageMagic (http://www.imagemagick.org/script/index.php). Je dostupný pod Linuxem, Mac OS i MS Windows. Důležité jsou zejména příkazy convert a identify.



Obrázek B.1: Popiska obrázku

DTD	construction	elimination
|	`in1|A|B a:sum A B` `in1|A|B b:sum A B`	`case([_:A]a)([_:B]a)ab:A` `case([_:A]b)([_:B]b)ba:B`
+	`do_reg:A -> reg A`	`undo_reg:reg A -> A`
$*, ?$	the same like | and + with `emtpy_el:empty`	the same like | and + with `emtpy_el:empty`
R(a,b)	`make_R:A->B->R`	`a: R -> A` `b: R -> B`

Tabulka B.1: Ukázka tabulky

```
\end{center}
\end{figure}
```

B.2 Kreslení obrázků

Zřejmě každý z vás má nějaký oblíbený nástroj pro tvorbu obrázků. Jde jen o to, abyste dokázali obrázek uložit v požadovaném formátu nebo jej do něj konvertovat (viz předchozí kapitola). Je zřejmě vhodné kreslit obrázky vektorově. Celkem oblíbený, na ovládání celkem jednoduchý a přitom dostatečně mocný je například program Inkscape.

Zde stojí za to upozornit na kreslící programe Ipe [4], který dokáže do obrázku vkládat komentáře přímo v latexovském formátu (vzroce, stejné fonty atd.). Podobné věci umí na Linuxové platformě nástroj Xfig.

Za pozornost ještě stojí schopnost editoru Ipe importovat obrázek (jpg nebo bitmap) a krelit do něj latexovské popisky a komentáře. Výsledek pak umí exportovat přímo do pdf.

B.3 Tabulky

Existuje více způsobů, jak sázet tabulky. Například je možno použít prostředí `table`, které je velmi podobné prostředí `figure`.

Zdrojový text tabulky B.1 vypadá takto:

```
\begin{table}
\begin{center}
\begin{tabular}{|c|l|l|}
\hline
\textbf{DTD} & \textbf{construction} & \textbf{elimination} \\
\hline
$\mid$ & \verb+in1|A|B a:sum A B+ & \verb+case([_:A]a)([_:B]a)ab:A+\\
&\verb+in1|A|B b:sum A B+ & \verb+case([_:A]b)([_:B]b)ba:B+\\
\hline
$+$&\verb+do_reg:A -> reg A+&\verb+undo_reg:reg A -> A+\\
\hline
$*,?$& the same like $\mid$ and $+$ & the same like $\mid$ and $+$\\
& with \verb+emtpy_el:empty+ & with \verb+emtpy_el:empty+\\
\hline
R(a,b) & \verb+make_R:A->B->R+ & \verb+a: R -> A+\\
 & & \verb+b: R -> B+\\
\hline
\end{tabular}
\end{center}
\caption{Ukázka tabulky}
\label{tab:tab1}
\end{table}
\begin{table}
```

B.4 Odkazy v textu

B.4.1 Odkazy na literaturu

Jsou realizovány příkazem `\cite{odkaz}`.

Seznam literatury je dobré zapsat do samostatného souboru a ten pak zpracovat programem bibtex (viz soubor `reference.bib`). Zdrojový soubor pro `bibtex` vypadá například takto:

```
@Article{Chen01,
  author  = "Yong-Sheng Chen and Yi-Ping Hung and Chiou-Shann Fuh",
  title   = "Fast Block Matching Algorithm Based on
             the Winner-Update Strategy",
  journal = "IEEE Transactions On Image Processing",
  pages   = "1212--1222",
  volume  =  10,
  number  =   8,
  year    = 2001,
}

@Misc{latexdocweb,
```

```
  author  = "",
  title   = "{\LaTeX} --- online manuál",
  note    = "\verb|http://www.cstug.cz/latex/lm/frames.html|",
  year    = "",
}
...
```

Pozor: Sazba názvů odkazů je dána BibTEX stylem (`\bibliographystyle{abbrv}`). BibTEX tedy obvykle vysází velké pouze počáteční písmeno z názvu zdroje, ostatní písmena zůstanou malá bez ohledu na to, jak je napíšete. Přesněji řečeno, styl může zvolit pro každý typ publikace jiné konverze. Pro časopisecké články třeba výše uvedené, jiné pro monografie (u nich často bývá naopak velikost písmen zachována).

Pokud chcete BibTEXu napovědět, která písmena nechat bez konverzí (viz `title = "{\LaTeX} --- online manuál"` v předchozím příkladu), je nutné příslušné písmeno (zde celé makro) uzavřít do složených závorek. Pro přehlednost je proto vhodné celé parametry uzavírat do uvozovek (`author = "..."`), nikoliv do složených závorek.

Odkazy na literaturu ve zdrojovém textu se pak zapisují:

```
Podívejte se na \cite{Chen01},
další detaily najdete na \cite{latexdocweb}
```

Vazbu mezi soubory `*.tex` a `*.bib` zajistíte příkazem `\bibliography{}` v souboru `*.tex`. V našem případě tedy zdrojový dokument `thesis.tex` obsahuje příkaz `\bibliography{reference}`.

Zpracování zdrojového textu s odkazy se provede postupným voláním programů `pdflatex <soubor>` (případně `latex <soubor>`), `bibtex <soubor>` a opět `pdflatex <soubor>`.[3]

Níže uvedený příklad je převzat z dříve existujících pokynů studentům, kteří dělají svou diplomovou nebo bakalářskou práci v Grafické skupině.[4] Zde se praví:

```
...
j) Seznam literatury a dalších použitých pramenů, odkazy na WWW stránky, ...
 Pozor na to, že na veškeré uvedené prameny se musíte v textu práce
 odkazovat -- [1].
Pramen, na který neodkazujete, vypadá, že jste ho vlastně nepotřebovali
a je uveden jen do počtu. Příklad citace knihy [1], článku v časopise [2],
stati ve sborníku [3] a html odkazu [4]:
[1] J. Žára, B. Beneš;, and P. Felkel.
     Moderní počítačová grafika. Computer Press s.r.o, Brno, 1 edition, 1998.
     (in Czech).
```

[3] První volání `pdflatex` vytvoří soubor s koncovkou `*.aux`, který je vstupem pro program `bibtex`, pak je potřeba znovu zavolat program `pdflatex` (`latex`), který tentokrát zpracuje soubory s příponami `.aux` a `.tex`. Informaci o případných nevyřešených odkazech (cross-reference) vidíte přímo při zpracovávání zdrojového souboru příkazem `pdflatex`. Program `pdflatex` (`latex`) lze volat vícekrát, pokud stále vidíte nevyřešené závislosti.

[4] Několikrát jsem byl upozorněn, že web s těmito pokyny byl zrušen, proto jej zde přímo necituji. Nicméně příklad sám o sobě dokumentuje obecně přijímaný konsensus ohledně citací v bakalářských a diplomových pracích na KP.

[2] P. Slavík. Grammars and Rewriting Systems as Models for Graphical User Interfaces. Cognitive Systems, 4(4--3):381--399, 1997.
[3] M. Haindl, Š. Kment, and P. Slavík. Virtual Information Systems. In WSCG'2000 -- Short communication papers, pages 22--27, Pilsen, 2000. University of West Bohemia.
[4] Knihovna grafické skupiny katedry počítačů: http://www.cgg.cvut.cz/Bib/library/

... abychom výše citované odkazy skutečně našli v (automaticky generovaném) seznamu literatury tohoto textu, musíme je nyní alespoň jednou citovat: Kniha [5], článek v časopisu [2], příspěvek na konferenci [8], www odkaz [11].

Ještě přidáme další ukázku citací online zdrojů podle české normy. Odkaz na wiki o frameworcich [9] a ORM [10]. Použití viz soubor `reference.bib`. V seznamu literatury by nyní měly být živé odkazy na zdroje. V `reference.bib` je zcela nový typ publikace. Detaily dohledal a dodal Petr Dlouhý v dubnu 2010. Podrobnosti najdete ve zdrojovém souboru tohoto textu v komentáři u příkazu `\thebibliography`.

B.4.2 Odkazy na obrázky, tabulky a kapitoly

- Označení místa v textu, na které chcete později čtenáře práce odkázat, se provede příkazem `\label{navesti}`. Lze použít v prostředích `figure` a `table`, ale též za názvem kapitoly nebo podkapitoly.
- Na návěští se odkážeme příkazem `\ref{navesti}` nebo `\pageref{navesti}`.

B.5 Rovnice, centrovaná, číslovaná matematika

Jednoduchý matematický výraz zapsaný přímo do textu se vysází pomocí prostředí `math`, resp. zkrácený zápis pomocí uzavření textu rovnice mezi znaky `$`.

Kód `$ S = \pi * r^2 $` bude vysázen takto: $S = \pi * r^2$.

Pokud chcete nečíslované rovnice, ale umístěné centrovaně na samostatné řádky, pak lze použít prostředí `displaymath`, resp. zkrácený zápis pomocí uzavření textu rovnice mezi znaky `$$`. Zdrojový kód: `|$$ S = \pi * r^2 $$|` bude pak vysázen takto:

$$S = \pi * r^2$$

Chcete-li mít rovnice číslované, je třeba použít prostředí `eqation`. Kód:

```
\begin{equation}
  S = \pi * r^2
\end{equation}

\begin{equation}
  V = \pi * r^3
\end{equation}
```

je potom vysázen takto:

$$S = \pi * r^2 \tag{B.1}$$

$$V = \pi * r^3 \tag{B.2}$$

B.6 Kódy programu

Chceme-li vysázet například část zdrojového kódu programu (bez formátování), hodí se prostředí `verbatim`:

```
         (* nickname2 *)
Lego> Refine in1
             (do_reg (nickname1 h));
Refine by  in1 (do_reg (nickname1 h))
   ?4 : pcdata
   ?5 : pcdata
          (* surname2 *)
Lego> Refine surname1 h;
Refine by  surname1 h
   ?5 : pcdata
          (* email2 *)
Lego> Refine undo_reg (email1 h);
Refine by  undo_reg (email1 h)
*** QED ***
```

B.7 Další poznámky

B.7.1 České uvozovky

V souboru `k336_thesis_macros.tex` je příkaz `\uv{}` pro sázení českých uvozovek. „Text uzavřený do českých uvozovek.“

Dodatek C

Seznam použitých zkratek

2D Two-Dimensional

ABN Abstract Boolean Networks

ASIC Application-Specific Integrated Circuit

⋮

Dodatek D

UML diagramy

Tato příloha není povinná a zřejmě se neobjeví v každé práci. Máte-li ale větší množství podobných diagramů popisujících systém, není nutné všechny umísťovat do hlavního textu, zvláště pokud by to snižovalo jeho čitelnost.

Dodatek E

Instalační a uživatelská příručka

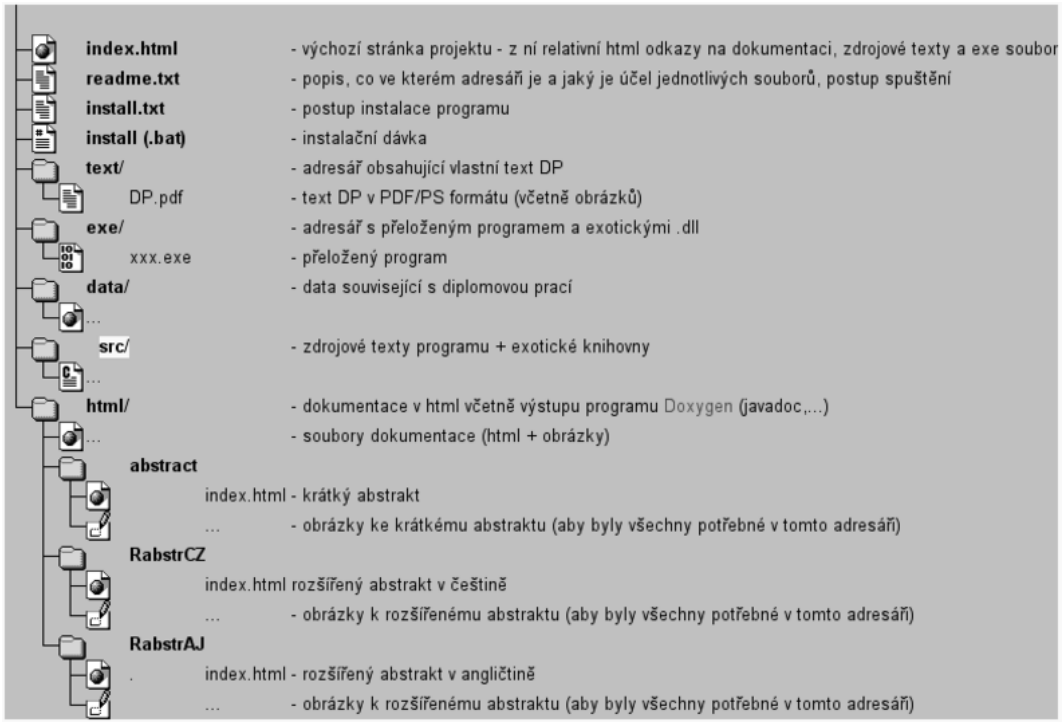
Tato příloha velmi žádoucí zejména u softwarových implementačních prací.

Dodatek F

Obsah přiloženého CD

Tato příloha je povinná pro každou práci. Každá práce musí totiž obsahovat přiložené CD. Viz dále.

Může vypadat například takto. Váš seznam samozřejmě bude odpovídat typu vaší práce. (viz [3]):



Obrázek F.1: Seznam přiloženého CD — příklad

Na GNU/Linuxu si strukturu přiloženého CD můžete snadno vyrobit příkazem:

```
$ tree . >tree.txt
```

Ve vzniklém souboru pak stačí pouze doplnit komentáře.

Z **README.TXT** (případne index.html apod.) musí být rovněž zřejmé, jak programy instalovat, spouštět a jaké požadavky mají tyto programy na hardware.

Adresář **text** musí obsahovat soubor s vlastním textem práce v PDF nebo PS formátu, který bude později použit pro prezentaci diplomové práce na WWW.