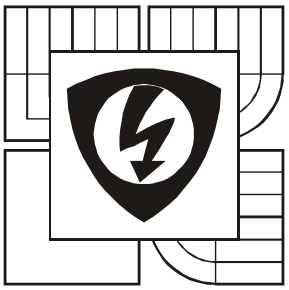
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



Počítačová cvičení předmětu
Síťové operační systémy

Garant předmětu:
prof. Dan Komosný

Autoři textu:
Dan Komosný a kolektiv

BRNO 2023

Obsah

1 ÚVOD

V rámci počítačových cvičení je prakticky probírána látka uvedená v teorii předmětu Síťové operační systémy. Náročnost cvičení je řešena tak, že v úvodních cvičeních je uváděn postup ve stylu krok za krokem. V dalších cvičeních, které již staví na předešle získaných dovednostech, jsou pouze uváděny pokyny k provedení. Návody k použití obecných nástrojů jsou uvedeny v příloze. Jedná se například o základní příkazy pro práci se soubory/adresáři nebo použití textového editoru *vi*.

Při úvodním vstupu do problematiky operačních systémů nemusí být smysl některých cvičení zřejmý. Typickým příkladem je odesílání emailu pomocí příkazové řádky. Kdo by mohl odesílat email pomocí příkazové řádky, když existující propracovaní klienti pracující v grafickém prostředí!? Odpovědí je, že odeslání emailu z příkazové řádky je často využíváno při dohledu nad činností operačního systému nebo konkrétních aplikací. Správce, který má na starosti větší počet stanic či serverů, má možnost si zhotovit skript, v rámci kterého bude periodicky prováděna kontrola určité činnosti. Při zjištění závady pak skript automaticky odešle email z příkazového řádku na adresu správce. Správce se tak nemusí neustále vzdáleně přihlašovat na dozorované stanice a kontrolovat jejich činnost – stačí mu sledovat příchozí emaily. Dalším příkladem využití může být potřeba odeslat email ze stanice, která nedisponuje grafickým prostředím. Tato situace je běžná u serverů a u dedikovaných zařízení, jako jsou například směrovače.

Skripta jsou každým rokem aktualizována podle poznatků z průběhu cvičení. Proto pracujte vždy s **aktuální verzí**. Na těchto skriptech se postupně podílelo více osob v podobně dílčích námětů, textů a vylepšení, zejména Michal Soumar, Jiří Balej a Ladislav Pecho. Dále se na skriptech podílela celá řada studentů s připomínkami a náměty na vylepšení.

2 POZNÁMKY K POUŽÍVÁNÍ SKRIPT

Tato skripta jsou psána jako návod pro realizaci počítačových cvičení. Úroveň detailnosti skript postupně klesá s vyšším pořadovým číslem cvičení. Studenti nabývají postupně znalosti z předchozích cvičení a tyto znalosti pak uplatňují ve cvičeních následujících. Z tohoto důvodu se **nedoporučuje pořadí cvičení měnit**.

Každé cvičení je opatřeno stručným teoretickým popisem látky, které se cvičení týká. Pak následuje postup vypracování. Cvičení je ukončeno samostatným úkolem.

Pro lepší orientaci je text skript rozlišen do několika druhů.

Text bez odlišení slouží k popisu vypracování cvičení. Pokud v tomto textu je pokyn k provedení příkazu, je nutné jej provést. Nemusí zde však být přesně určeno, jak má příkaz vypadat. Příkladem může být požadavek „rozbalte archiv do domovského adresáře“.

Text odlišený *kurzívou* značí názvy programů, příkazů, funkcí atd. Zde je kurzíva použita pro lepší odlišení názvů od běžného textu. `Tento druh sazby` označuje text tak, jak má být zadán do terminálu.

Výpisy z terminálu jsou ohraničeny rámečkem. Text ve výpisu začínající znaky `[]$` rozlišuje příkazy, které mají být zadány do terminálu.

```
[]$ cp -a /usr/src/redhat/ ~/rpmbuild
```

Pozornost věnujte tomu, zda byl příkaz rozdělen na **více řádků** z důvodu formátování textu (rozsah delší než šířka stránky). Následující řádek značí pouze jeden příkaz, ovšem zapsaný na dvou řádcích.

```
[]$ rsync -a --delete --progress --rsh='ssh -p 2222' ~/dokumenty/
   <ucet>@<pocitac>:~/zaloha_<lokalni_pocitac>
```

Dále jsou uvedeny dva příkazy na dvou řádcích.

```
[]$ cp -a /usr/src/redhat/ ~/rpmbuild
[]$ echo '%_topdir %(echo $HOME)/rpmbuild' > ~/.rpmmacros
```

Jak je vidět z obou výše uvedených výpisů, rozdíl je v tom, že u příkazu rozděleného na více řádků není na následujícím řádku znak `[]$`.

Text v rámečku bez uvozujících znaků `[]$` zobrazuje výstupy příkazů. Například se může jednat o výstup příkazu ke kontrole, že bylo dosaženo správného řešení.

```
[]$ make -C /lib/modules/2.6.18-92.1.22.el5/build M=/home/student/modul
   modules

make[1]: Entering directory `/usr/src/kernels/2.6.18-92.1.22.el5-i686'
CC [M]  /home/student/modul/hello.o
Building modules, stage 2.
MODPOST
CC      /home/student/modul/hello.mod.o
```

```
LD [M]  /home/student/modul/hello.ko
make[1]: Leaving directory '/usr/src/kernels/2.6.18-92.1.22.el5-i686'
```

Dále znakem `[]$` nejsou uvozeny obsahy souborů, které se mají vytvořit. Níže je uveden příklad obsahu souboru k vytvoření.

```
uname -a
cat /proc/cpuinfo
```

> Text zvýrazněný čarou vlevo přináší vysvětlení uvedeného postupu a popisuje další možnosti. Někdy jsou uvedena i možná vylepšení. Pokud je v tomto textu uveden postup s uvedením příkazu, **tento příkaz není součástí cvičení a do terminálu se nezadává**.

Další formy zápisu textu jsou:

- Text v příkazu mezi znaky `<>` (bez mezer) značí, že má být **nahrazen**. Například v příkazu `<ucet>@<pocitac>:<soubor>` je nutno „ucet“ nahradit názvem konkrétního účtu a „pocitac“ nahradit IP adresou nebo doménovým jménem.
- Text v příkazu označený `< >` (s mezerami) se nenahrazuje. Například se jedná o konstrukci `cat < /etc/passwd > ~/hesla.txt; more hesla.txt`.

V přílohách skript naleznete různé pomůcky ulehčující realizaci cvičení. V první příloze je uveden seznam nejčastěji používaných příkazů. Ze začátku tak odpadne časté „lovení“ v paměti, který příkaz a jaké jeho parametry je nutno použít pro triviální operace. Především se jedná o příkazy editoru *vi*, které budou stále potřeba. Druhá příloha popisuje nejdůležitější požadavky na operační systém pro realizaci jednotlivých cvičení. To by mělo ulehčit práci těm, kteří by si chtěli cvičení provést mimo laboratoř.

Snahou bylo cvičení nevázat na grafické prostředí a vytvořit tak **univerzální návody nezávislé na konkrétní distribuci**. Jednotlivé distribuce se však v některých věcech liší (např. systém pro automatickou instalaci programů). Proto je doporučeno cvičení provádět na distribuci CentOS, za pomocí které byla tato cvičení sestavena.

3 POČÍTAČOVÉ CVIČENÍ Č. 1 – PŘÍKAZOVÝ INTERPRET

3.1 Účel cvičení

Seznámit studenty s operačním systémem Linux a příkazovým řádkem. Demonstrovat příklad vytvoření jednoduchého skriptu. Cvičení slouží jako úvodní pro realizaci dalších cvičení a projektu.

3.2 Teoretický úvod

3.2.1 Příkazový interpret – shell

Hlavním úkolem shellu je zprostředkování interakce mezi uživatelem a operačním systémem. Příkazový interpret není součástí jádra systému, ale jedná se o samostatný program. Protože je OS Linux modulární, je možno vybírat z mnoha různých shellů. Přehled je uveden v tabulce 3.1

Tabulka 3.1: Přehled příkazových interpretů.

Název	**Popis**
bash	Bourne Again Shell z projektu GNU, patří k nejrozšířenějším
sh	Původní Bourne shell (přelom 60. a 70. let)
csh, tcsh	Tzv. C shell, syntaxe podobná jazyku C (počátek 80. let), a jeho zdokonalený nástupce Tenex C shell
ksh, pdksh	Korn shell a jeho volně dostupný klon (pd = public domain)
rc	Ještě úžeji spjatý s jazykem C, taktéž GNU projekt
zsh	Z-shell, obsahuje velké množství funkcí

Výčet v tabulce není kompletní, existují desítky různých příkazových interpretů. Shell neslouží jen ke spouštění příkazů, je to kompletní programovací jazyk. Soubory obsahující příkazy pro shell se nazývají **skripty**. Pro zajištění jejich přenositelnosti mezi různými interprety byly definovány standardy **POSIX**. Pracujeme-li s příkazovým interpretem splňujícím normu POSIX, bude skript či sekvence příkazů pracovat i v ostatních shellech bez jakýchkoliv úprav. Programování v shellu je rychlé a jednoduché, navíc některý z uvedených interpretů je dostupný ve všech verzích Linuxu. Často se používají malé nástroje, které provádí některé relativně jednoduché úkoly, při nichž je více než efektivita důležitá jednoduchost konfigurace, snadná údržba a přenositelnost.

3.2.2 Programy v příkazovém interpretu

Po přihlášení uživatele se příkazový interpret na terminálu ohlásí výpisem znaku `$` pro obyčejného uživatele a znakem `#` pro privilegovaného (root). Uvedené znaky jsou výzvou pro zadání příkazového řádku. Ten má formát *arg0 arg1 arg2* ..., kde arg0 až argN jsou

textové řetězce oddělené navzájem mezerou nebo tabulátorem. Řetězec *arg0* je vždy jméno příkazu, který může být buď vnitřní (built-in) nebo vnější (external).

Vnitřní příkazy jsou akce, kterými uživatel požaduje určitou změnu v chování příkazového interpretu. Například se jedná o změnu pracovního adresáře, příkaz `cd`, nebo příkaz pro odhlášení. Provedení vnitřních příkazů je rychlé, protože tyto příkazy nejsou obsluhovány spuštěním samostatného programu, jako je tomu u externích příkazů. Neodpovídá-li zadaný příkaz žádnému z vnitřních příkazů, je považován za vnější. Po jeho zadání je pod tímto jménem hledán soubor, který příkaz realizuje. Prohledávány jsou typicky adresáře `/bin`,`/usr/bin` atd., podle obsahu proměnné PATH.

Další argumenty příkazového řádku (*arg1* až *argN*) mohou mít různý význam daný funkcí příkazu. Jedná se tedy o parametry. Existují dva způsoby, jak psát programy v příkazovém interpretu. První možností je práce s příkazovým řádkem, tzn. interaktivní zadávání posloupnosti příkazů, které se mají provádět. Druhou možností je uložení příkazů do souboru jakožto skript a jeho spuštění ve formě programu.

3.2.3 Proměnné příkazového interpretu

Proměnnou uživatel vytváří a současně definuje přiřazením jména a jeho obsahu, kde jméno je jméno proměnné a obsah je textový řetězec, kterým bude proměnná naplněna. Dereference je uvozena metaznakem `$` – bezprostředně následovaným názvem proměnné. Příkazový interpret definuje své vlastní proměnné, které mají zvláštní význam nebo dokonce ovlivňují chování spuštěných programů. Výběr některých je uveden v následující tabulce 3.2.

Tabulka 3.2: Významné statické proměnné příkazového interpretu.

Proměnná	**Popis**
HOME	Cesta k domovskému adresáři uživatele
PATH	Cesty k adresářům, ve kterých se vyhledávají spustitelné soubory
USER	Uživatelské jméno

Výše uvedené jsou proměnné, které jsou definovány bez iniciativy uživatele. V době běhu příkazového interpretu existují ještě další proměnné, jejichž obsah se ale dynamicky mění podle současného stavu spuštěných procesů. Některé z nich jsou zachyceny v Tabulce 3.3.

Tabulka 3.3: Významné proměnné při běhu příkazového interpretu.

Proměnná	**Popis**
`?`	Návratový kód posledního provedeného příkazu
`#`	Počet parametrů skriptu
`0`	První řetězec příkazového řádku skriptu (= jméno skriptu)
`1 - 9`	První (až devátý) parametr skriptu. GNU Bash povoluje i více parametrů než 9. Odkazuje se na ně dereferencí ve tvaru $N

3.2.4 Skripty

Skript lze spustit těmito způsoby: i) pomocí příkazu (například *sh*), kterému se jako parametr předloží název skriptu, nebo ii) přímým spuštěním pomocí zadání názvu skriptu (soubor má předešle nastaven příznak spustitelnosti).

Skript může začínat záhlavím, které je nepovinné. Pomocí záhlaví lze deklarovat **program** (příkazový interpret), **jenž bude skript provádět** (tj. příkazy v těle skriptu) při jeho spuštění pomocí zadání názvu souboru. Není-li záhlaví uvedeno, příkazy skriptu jsou prováděny výchozím interpretem. Pokud je v záhlaví uveden jiný interpret příkazů, než pro který jsou uvedeny následující příkazy, tak skript nebude pracovat. Příklady záhlaví mohou být `#!/bin/bash` (příkazový interpret Bash) nebo `#!/usr/bin/env python3` (příkazový interpret Python).

Úplné základy práce s příkazovým řádkem lze nalézt v publikaci [1]. Z anglicky psané literatury lze doporučit pro začátky práce publikaci [2]. Další informace lze nalézt například v knize, která se zabývá přechodem na OS Linux [3].

3.3 Pokyny pro vypracování

3.3.1 Práce v příkazovém řádku

- Význam a funkci příkazů naleznete v manuálových stránkách. Vyvolání manuálových stránek se děje příkazem `man <prikaz/program>`. Pokud je vám nějaký příkaz nejasný (nebo jeho parametr), **zjistěte si jeho význam pomocí jeho manuálové stránky**. Manuálové stránky opustíte stiskem klávesy [q].

```
[]$ man uname
```

Zjištění, zda se jedná o vnitřní a vnější příkaz shellu lze zjistit příkazem *type*.

```
[]$ type cd; type logout
cd je soucast shellu
logout je soucast shellu

[]$ type find;type cat
find je /usr/bin/find
cat je /usr/bin/cat
```

> Na jeden řádek je možné uvést více příkazů oddělených metaznakem `;` (středník), například `uname -a; whereis bash; pwd`.

- Umístění souboru s programem, který realizuje daný příkaz, lze také zjistit pomocí příkazů *which* nebo *whereis*. Oba příkazy prohledávání adresáře specifikované v proměnné PATH. Druhý příkaz zobrazuje podrobnější informace, například adresář s konfigurací a umístění knihoven.

```
[]$ which more
```

```
/usr/bin/more

[]$ whereis ssh python
ssh: /usr/bin/ssh /etc/ssh ..
python: /usr/bin/python /usr/lib/python /usr/include/python ...
```

Pokud není soubor s programem v prohledávaných adresářích (proměnná PATH), tak je nutno zadat **celou cestu k příkazu** pro jeho spuštění, například `/usr/sbin/ifconfig`.

- Nalezení libovolného souboru (adresáře) lze provést příkazem *locate*. Tento příkaz vyhledá všechny soubory podle zadaného jména, tj. neprohledává jenom adresáře specifikované v proměnné PATH. Tento příkaz je rychle provedený protože ve skutečnosti neprohledává všechny adresáře, ale pracuje s předešle vytvořenou databází umístění souborů. Tato databáze je periodicky obnovována, například každý den. Může se tedy stát, že nové soubory nebudou nalezeny, nebo budou nalezeny soubory nedávno smazané. Příkaz vyhledá všechny soubory a adresáře, které v názvu obsahují zadaný hledaný text.

```
[]$ locate python3
/usr/bin/python3
/usr/lib/python3.9
/usr/lib/python3.9/site-packages
...
```

- Programy při práci přetvářejí vstupní data na výstup. Pomocí operátorů přesměrování (`<`, `>`, `<<`, `>>`) je možno přesměrovat standardní vstup/výstup. Standardní vstup je typicky klávesnice, výstup pak terminál. Zobrazte výstup příkazu *time*, proveďte přesměrování výstupu příkazu do souboru `obsah.txt` a poté si obsah souboru vypište.

```
[]$ time ls -al
[]$ time ls -al > obsah.txt
[]$ more obsah.txt
```

> V OS Linux se používají termíny „standardní vstup“ (stdin), „standardní výstup“ (stdout) a „standardní chybový výstup“ (stderr) . První typicky představuje klávesnici, oba následující pak terminál. Standardní výstup obsahuje výstupní data vytvořená programem, chybový výstup informace o chybách během zpracování vstupních dat. Příslušná čísla deskriptorů souborů jsou 0, 1 a 2 (v tomto pořadí – tedy standardní vstup má deskriptor 0, výstup 1 a chybový výstup 2).

- Vyzkoušejte následující použití příkazu *cat*, který má s využitím operátorů přesměrování funkci podobnou příkazu *cp* (copy). Vypište si obsah souborů `passwd` a `hesla.txt`. Soubory porovnejte.

```
[]$ cat < /etc/passwd > ~/hesla.txt
[]$ more /etc/passwd
```

```
[]$ more hesla.txt
```

Konkrétně u příkazu *cat* není přesměrování vstupu (operátor `<`) nezbytný, protože příkaz *cat* očekává parametr se jménem vstupního souboru. Metaznak `~` (tilda) značí **domovský adresář** aktuálního uživatele. Různé speciální znaky, které příkazový interpret interpretuje, například (`<`, `>`, `|`, `&`), nejsou argumenty programů spouštěných příkazovým interpretem. Například v následujícím případě příkaz *cat* o řetězci `< /etc/passwd > ~/hesla.txt` vůbec neví. Ve skutečnosti by pořadí mohlo být libovolné: `< /etc/passwd > ~/hesla.txt cat`.

- Jiným typem přesměrování vstupu a výstupu je tzv. roura, která se zapisuje metaznakem `|`. Roura spojuje výstup jednoho programu se vstupem druhého. Nejprve proveďte výpis obsahu adresáře.

```
[]$ ls -l
```

- Nyní použijte rouru pro přesměrování výstupu programu *ls* na vstup programu *wc*. Program *wc* slouží pro získání statistik o textovém dokumentu.

```
[]$ ls -l | wc -l
```

Zřetězení může být několikanásobné, což znamená, že se tento metaznak může vyskytovat na příkazové řádce několikrát. Příkaz `wc -l` vypíše počet řádků výpisu programu *ls*.

- Pomocí příkazu *set* si vypište obsah všech nadefinovaných proměnných.
- Zavedené proměnné, které nebudete dále potřebovat, lze zrušit příkazem *unset* nebo prázdným přiřazením:

```
[]$ test=cest; echo $test
[]$ unset test; echo $test
#nebo
[]$ test=; echo $test
```

3.3.2 Vytvoření skriptu

- Vytvořte skript (např. v editoru *vi*, jak pracovat s tímto editorem lze nalézt v příloze). Do skriptu vložte příkazy pro výpis informací o systému a o procesoru. Skript uložte jako `skript.sh`.

```
#!/bin/bash
uname -a
cat /proc/cpuinfo
```

- Vyzkoušejte dva způsoby jeho spuštění. Spustitelnost skriptu nastavte příkazem `chmod +x skript.sh`.

```
[]$ sh skript.sh
[]$ ./skript.sh
```

> Druhý způsob začíná tečkou – jedná se o znak **aktuálního adresáře**. V proměnné PATH, která specifikuje adresáře se soubory programů k vyhledání, není aktuální adresář **z bezpečnostních důvodů uveden**. Otevřela by se tak cesta ke spouštění příkazů z pracovního adresáře „omylem“. Tyto příkazy by mohly nahrazovat standardní systémové příkazy a například místo vypsání obsahu souboru příkazem *more* by příkaz *more* (soubor s programem s názvem *more* v aktuálním adresáři) soubor smazal!

- Na konec skriptu dále vložte ještě příkaz *exit* a číslo návratového kódu (exit status).

```
exit 0
```

> Je-li návratový kód nulový, program ukončil svou činnost v **pořádku**. Je-li nenulový, program skončil s chybou. Číslo návratového kódu pak označuje **kód chyby programu**, který lze typicky dohledat v manuálových stránkách. Příkaz *exit* akceptuje jako parametr návratový kód. Pozor – je-li parametr vynechán, je návratová hodnota nastavena dle **předešlého vykonaného příkazu**.

- Skript spusťte pomocí obou způsobů a ověřte návratový kód skriptu výpisem proměnné `$?`.

```
[]$ echo Navratova hodnota je: $?
```

> Příkaz *echo* vytiskne zadaný text do terminálu.

3.3.3 Samostatný úkol

- Vytvořte vlastní libovolný skript pro příkazový interpret Bash. Soubor s tělem skriptu uložte do domovského adresáře. Skript vytvořte jednoduchý (nezáleží na náročnosti jeho obsahu). Skript vytvořte tak, aby obsahoval tyto části:
  1. Správné záhlaví skriptu pro provedení příkazů interpretem Bash `#!/bin/bash`.
  2. Definice vlastních proměnných a práce s nimi (například sloučení textu, sčítání atd.). Matematické operace lze provádět pomocí příkazu *let*, například `let x=x+1`.
  3. Použití přesměrování výstupu do souboru (pomocí znaků `>` nebo `>>`).
  4. Použití řetězení příkazů pomocí roury (znak `|`).

5. Vytvoření podmínky `if ... then ... else`. Způsob konstrukce podmínky vyhledejte na Internetu. Vracení různého návratového kódu skriptu podle stavu podmínky – číslo 2 (podmínka splněna) a číslo 3 (podmínka neplněna); hodnoty jsou voleny záměrně pro demonstraci činnosti skriptu.

V editoru *vi* lze pro lepší orientaci zobrazit čísla řádků. Tyto čísla nejsou součástí skriptu.

```
1 :set number
```

- Se skriptem proveďte tyto operace a předveďte je vyučujícímu:
  1. Skript spusťte dvěma způsoby – i) příkazem *sh* a název souboru a ii) přímým zadáním názvu souboru (nastavit soubor jako spustitelný a použít konstrukci `./<soubor>`).
  2. Zobrazte výstupy činnosti skriptu (proměnná, soubor).
  3. Na základě změny podmínky zobrazte různé návratové kódy skriptu (hodnoty 2 a 3).
  4. Vyzkoušejte, že po odstranění příkazu *exit* je vrácena návratová hodnota **posledního příkazu** ve skriptu.

> Číslování řádků v editoru vi zrušíte příkazem `:set nonumber`.

4 POČÍTAČOVÉ CVIČENÍ Č. 2 – SOUBOROVÝ SYSTÉM

4.1 Účel cvičení

Práce se soubory, adresáři a uživateli. Použití přístupových práv. Možnost přepínání mezi uživateli. Cvičení slouží jako úvodní pro realizaci dalších cvičení a projektu.

4.2 Teoretický úvod

4.2.1 Účel souborových systémů a jejich přehled

Operační systém Linux podporuje práci s různými **souborovými systémy**. Kromě souborových systémů vyvinutých pro jádro Linux, jako je například Ext4 (Fourth Extended Filesystem), podporuje i souborové systémy náležící k jiným operačním systémům, např. NTFS (New Technology File System), FAT (File Allocation Table) a další. Pro podporu více souborových systémů a pro současnou práci s nimi vznikl tzv. virtuální souborový systém, který poskytuje pro každý reálný souborový systém jednotné rozhraní.

4.2.2 Soubory a adresáře

Adresářová struktura v OS Linux začíná v tzv. kořenu (`/`), který je jedinečný pro celý systém. Výpis položek souborového systému ve zvoleném adresáři lze provést příkazem *ls*. Zda se jedná o soubor či adresář lze poznat podle prvního písmenka u výpisu pomocí příkazu `ls -l`. Písmeno `d` značí adresář a znak `-` značí soubor.

```
drwx------. 17 student student 4096  3. rij 15.52 ..
-rw-rw-r--.  1 student student    0  3. rij 15.52 soubor
```

Soubory, které zavádí nové (další) jméno pro již existující soubor, se nazývají **odkazy**. Při vytváření odkazu (na rozdíl od kopírování klasického souboru) nedojde k fyzickému přesouvání dat, ale pouze se vytvoří nové jméno, které je spojeno s původním souborem. Odkazy se používají tehdy, pokud uživatel potřebuje přímou dostupnost jednoho souboru z více adresářů. Existují dva typy odkazů:

- **Pevný odkaz** (hard link) – Při vytvoření pevného odkazu se počet cest k souboru zvýší o jedna. Při smazání odkazu zůstane obsah zachován, jen se odebere jedna z cest k souboru. Při nulovém počtu cest k souboru je soubor smazán. Typicky nelze vytvořit pevný odkaz na adresář.
- **Symbolický odkaz** (symbolic link) – jedná se o soubor, který jako data obsahuje cestu k cílovému souboru. Tento odkaz lze zavést i na neexistující soubor nebo adresář.

Pevný odkaz je v souborném systému stejný jako vlastní soubor. Pevný nebo symbolický odkaz lze rozeznat pomocí prvního písmenka u výpisu pomocí příkazu `ls -l`.

Symbolický odkaz začíná písmenem `l`. Dále uvedená plná práva jsou ignorována – platí práva souboru, na které symbolický odkaz odkazuje.

```
lrwxrwxrwx. 1 komosny komosny   6  4. rij 09.02 odkazNaSoubor -> soubor
-rw-rw-r--. 1 komosny komosny 128  4. rij 09.02 soubor
```

4.2.3 Práva souborů

Práva pro přístup k souborům jsou dělena do skupin. Jedná se o tři skupiny: **pro vlastníka, pro skupinu vlastníka a pro všechny ostatní** (mimo vlastníka a skupinu vlastníka). Tyto skupiny jsou označovány pomocí písmen `u` (user), `g` (group) a `o` (other). Celá množina (`ugo`) je souhrnně označována písmenem `a` (all). Práva v každé z těchto skupin mohou být následující: `r` – čtení, `w` – zápis, `x` – spouštění.

Při spuštění souboru, který obsahuje spustitelný kód, má vzniklý proces práva, jako uživatel, který soubor spustil. Existují však výjimky při těchto nastavených právech:

- `s` (setuid, platí pro vlastníka) – pokud je soubor spuštěn (tj. soubor obsahuje spustitelný kód), vzniklý proces má přístupová práva jako vlastník souboru, nikoliv jako ten, kdo kód ze souboru spustil,
- `s` (setgid, platí pro skupinu vlastníka) – po spuštění kódu z tohoto souboru má vzniklý proces přístupová práva jako skupina, která vlastní tento soubor, nikoliv jako skupina uživatele, který kód ze souboru spustil.

4.2.4 Práva adresářů

Práva pro přístup k adresářům jsou dělena do skupin tak, jako je tomu u souborů. Práva v rámci skupin jsou následující:

- `r` – Přístup k souborům v adresáři (zobrazení obsahu adresáře).
- `w` – Vytváření a mazání souborů v adresáři. Nemá vliv na možnost zapisovat do existujících souborů. Pokud chce uživatel smazat nějaký soubor v adresáři, nepotřebuje právo `w` u souboru, pokud je u adresáře. Naopak, soubor nelze smazat, i když je právo `w` u souboru, ale chybí u adresáře, ve kterém soubor je.
- `x` – Možnost vstoupit do adresáře.
- `s` (setuid, platí pro vlastníka) – Ignorováno.

Vytvářené soubory v adresáři získávají vlastnictví a práva podle uživatele i skupiny uživatele, který je vytvořil. To je možné změnit pomocí práva:

- `s` (setgid, platí pro skupinu vlastníka) – Vytvářené soubory v takto nastaveném adresáři získají skupinové vlastnictví podle adresáře, nikoliv podle uživatele, který soubor vytváří – hodí se při sdílení adresáře více uživateli společné skupiny.

V případě práva ke vstupu a zápisu do adresáře může uživatel mazat všechny soubory v tomto adresáři, i když není jejich vlastníkem. Toto lze ovlivnit pomocí práva:

- `t` (sticky) – V adresáři s tímto oprávněním mohou uživatelé sice vytvářet soubory (pokud existuje také oprávnění `w`), ale mohou mazat pouze své soubory; uživatel může cizí soubor smazat pouze tehdy, pokud je vlastníkem společného adresáře.

U práv `r` a `x` mohou u adresářů nastat zvláštní případy:

- `r` zapnuto, `x` vypnuto – Do adresáře nelze vstoupit, ale lze vypsat seznam souborů.
- `r` vypnuto, `x` zapnuto – Do adresáře lze vstoupit, ale nelze vypsat seznam souborů. Pokud známe název souboru uvnitř tohoto adresáře, můžeme získat jeho obsah.

4.2.5 Nastavení oprávnění

Oprávnění se mění pomocí příkazu *chmod*. Tento příkaz akceptuje parametry `[ugoa] [-+=] [rwxst]` a název souboru/adresáře. Nejprve je uvedeno komu práva měníme, pak zda právo odebíráme (`-`), přidáváme (`+`), nebo nastavujeme podle následujícího výčtu (`=`).

Práva lze vyjádřit i v číselné podobě a pracovat s nimi jako parametrem příkazu *chmod*. Právo `x` má hodnotu 1, `w` má 2 a `r` má hodnotu 4. V číselném vyjádření se práva sečtou a levá číslice udává právo vlastníka souboru, prostřední právo skupiny a pravá číslice práva ostatních uživatelů. Například právo 640 u souboru znamená, že vlastník může číst a zapisovat (4+2), skupina může číst (4) a ostatní nemají žádná práva (0). Pro další práva platí následující číselné vyjádření: sticky (`t`) – 1000, setgid (`s` u skupiny) – 2000 a setuid (`s` u uživatele) – 4000. Tedy pokud k oprávnění 640 budeme chtít přidat bit setgid, musíme použít oprávnění 2640.

Například oprávnění 640 můžeme zapsat jako `u=rw,g=r,o=` (bez mezer) a oprávnění 2640 jako `u=rw,g=rs,o=` (bez mezer). Pokud se například rozhodneme zakázat ostatním a skupině právo zapisovat, použijeme `go-w` a právo zápisu vlastníka souboru zůstane nedotčené (nezáleží, jak bylo předtím nastaveno). Pokud například chceme přidat jenom bit setgid, použijeme zápis `g+s`.

4.2.6 Správa uživatelů v systému

Nový účet je možné založit třemi základními způsoby. Základní je použít příkaz *vipw*, který edituje přímo soubor `/etc/passwd`, ve kterém jsou uloženy informace o jednotlivých uživatelích. V souboru `/etc/group` jsou pak informace o uživatelských skupinách. Pokročilejší (snazší) nastavení umožňují příkazy *adduser* a *useradd*. Jedná se sice o dva příkazy, ale ve skutečnosti dělají to samé. Rozdíl oproti přímé editaci souborů s uživateli je v tom, že zadávání nových uživatelů je interaktivní na základě kladených otázek. Uživatelsky nejméně náročné jsou grafické nadstavby. Tyto však mohou být rozdílné pro různé distribuce. Heslo danému uživateli lze nastavit pomocí příkazu *passwd*. Dalšími nástroji pro správu účtů například jsou *userdel* – odstranění uživatele, *usermod* – změna vlastností uživatele, například přidání do další skupiny.

Další informace ohledně základní správy uživatelů a práv lze nalézt v tomto kapesním průvodci administrátora [4]

4.3 Pokyny pro vypracování

4.3.1 Práce s právy souborů a adresářů

- V domovském adresáři vytvořte soubor (například pomocí příkazu *touch* nebo editoru *vi*). Nastavte různá práva pro vámi vytvořený soubor pomocí příkazu *chmod*. Změnu práv si ověřte příkazem `ls -l`. Vyzkoušejte si i nastavení práv pomocí zadání číselné reprezentace. Můžete například použít tyto příkazy:

```
[]$ touch soubor1
[]$ chmod u=rw,g=r,o= soubor1
[]$ ls -l
[]$ chmod 664 soubor1
[]$ ls -l
```

- Následně si také zkuste nastavovat práva pro adresář, který si vytvořte (použijte příkaz *mkdir*).

4.3.2 Práce s uživateli

Nyní založte dva uživatele, vytvořte pro ně společnou skupinu, nastavte potřebná práva a vytvořte sdílený adresář pro tyto uživatele.

- Založení nového uživatele a skupiny proveďte následovně:

```
[]$ sudo /usr/sbin/useradd uzivatel
[]$ sudo /usr/sbin/groupadd skupina_uzivatelu
```

> Příkazem *sudo* vykonáte následující příkaz s právy administrátora. Systém, ve kterém pracujete, je nastaven tak, že máte povoleny vykonat všechny příkazy s právem administrátora, pokud před ně uvedete *sudo*.

- Že *uzivatel* v systému existuje si ověřte výpisem existence jeho domovského adresáře.

```
[]$ ls /home
```

- Výpisem souboru `/etc/group` si ověřte, že skupina *skupina_uzivatelu* existuje.

```
[]$ more /etc/group
gdm:x:42:
student:x:500:
apache:x:48:
vboxsf:x:501:
uzivatel:x:502:
skupina_uzivatelu:x:503:
```

- Podobně ověřte vytvoření uživatele *uzivatel* v souboru `/etc/passwd`.
- Následně přidejte uživatele *uzivatel* do vytvořené skupiny *skupina_uzivatelu*. To zajistíte parametrem `-G` u příkazu *usermod*.

```
[]$ sudo /usr/sbin/usermod -G skupina_uzivatelu uzivatel
```

- Že uživatel *uzivatel* patří do skupiny *skupina_uzivatelu* (mimo své vlastní skupiny) si ověřte pomocí příkazu *groups*, který vám vypíše, do jakých skupin zadaný uživatel náleží.

```
[]$ groups uzivatel
uzivatel : uzivatel skupina_uzivatelu
```

- Stejnou věc ověřte výpisem souboru `/etc/group`.
- Vytvořte dalšího uživatele *uzivatel2*. Pokud vytváříme nového uživatele, lze další skupinu(y), do které uživatel patří, zadat rovnou parametrem `-G`. Skupina musí už existovat.

```
[]$ sudo /usr/sbin/useradd uzivatel2 -G skupina_uzivatelu
```

- Opět ověřte, že *uzivatel2* patří již do skupiny *skupina_uzivatelu*.

```
[]$ groups uzivatel2
uzivatel2 : uzivatel2 skupina_uzivatelu
```

- **Přepněte se do domovského adresáře.**

```
[]$ cd ~
```

> Znak ~ značí zkratku pro cestu k domovskému adresáři uživatele.

- Nyní vytvořte v domovském adresáři adresář s názvem `sdileny_adresar`.
- Změňte skupinové vlastnictví vytvořeného adresáře na skupinu *skupina_uzivatelu*.

```
[]$ sudo chgrp skupina_uzivatelu sdileny_adresar
```

- Samostatně nastavte práva příkazem *chmod*. Práva vlastníka souboru ponechejte stejná. Členům skupiny umožněte **do adresáře vstoupit, načítat soubory, mazat a vytvářet nové soubory**. Ostatním uživatelům nepovolte **žádná práva**. Práva k adresáři si ověřte pomocí příkazu *ls*.
- Přepněte se do adresáře `sdileny_adresar`.

```
[]$ cd sdileny_adresar
```

- Přepněte se do uživatele *uzivatel*.

```
[]$ sudo su uzivatel
```

Poznámka: **mezi uživateli přecházíme v adresáři** `sdileny_adresar` z důvodu nastavených potřebných práv u tohoto adresáře.

> Příkazem *su* lze provést změnu uživatele bez odhlášení a přihlášení nového uživatele. Parametrem příkazu je jméno uživatele, na kterého chceme provést změnu. Pro příkaz je potřeba právo administrátora, proto je před příkazem uvedeno *sudo*.

- Jako *uzivatel* vytvořte v adresáři `sdileny_adresar` jeden soubor, například pomocí příkazu *touch*.

```
[]$ touch nazev_souboru
```

> Příkazem *touch* vytvoříte prázdný soubor.

- Pomocí příkazu `ls -l` si ověřte, jaká skupina (a s jakými právy) vlastní nově vytvořený soubor. Soubor je vlastněný skupinou *uzivatel*.

Soubor dále upravíte jako *uzivatel2*. Nejprve je však potřeba se na tohoto uživatele přepnout. Změnu uživatele dosáhnete opět pomocí příkazu `sudo su`. Do účtu *uzivatel2* se **lze přepnout pouze** z uživatele *student*, který na to má právo (příkaz *sudo*).

- Předchozího uživatele proto nejprve opusťte příkazem *exit*.
- Nyní budete v systému jako uživatel ***student*** a můžete se přepnout do uživatele *uzivatel2*.

```
[]$ sudo su uzivatel2
```

- Proveďte zápis do souboru, například pomocí příkazu *echo*.

```
[]$ echo "text k zapsani do souboru" > nazev_souboru
bash: nazev_souboru: Operace zamitnuta
```

Do souboru nepůjde zapisovat, protože soubor je vlastněn skupinou *uzivatel*. Jako *uzivatel2*, který není v této skupině, ho tedy nemůžete měnit. Tuto skutečnost si lze ověřit výpisem pomocí příkazu `ls -l`.

> V této situaci ovšem lze soubor smazat, protože adresář `sdileny_adresar` má oprávnění `w` pro skupinu *skupina_uzivatelu*, do které *uzivatel2* patří. Ze stejného důvodu může *uzivatel2* soubor přejmenovat příkazem *mv* (přesun dat). Nepřímé změny souboru lze tedy dosáhnout jeho smazáním a opětovným vytvořením. Některé editory tuto možnost samy nabídnou, například *vi*, příkazem `:w!`

Pro dosažení sdíleného adresáře musíme tedy přidat adresáři `sdileny_adresar` oprávnění, které zajistí, aby **skupinové vlastnictví nových souborů** patřilo vždy skupině *skupina_uzivatelu*.

- Ukončete práci jako *uzivatel2* příkazem *exit*. V systému byste **měli vystupovat jako uživatel** *student*. Tuto skutečnost zjistíte pomocí příkazu *id*.

> Příkaz *id* vypíše informace o aktuálním uživateli, včetně jeho *uid* a *gid*.

- Proveďte změnu práv adresáře `sdileny_adresar` (musíte být o jeden adresář výše). Pozor: **změnu provádíte jako uživatel *root***.

```
[]$ sudo chmod g+s sdileny_adresar
```

> Nastavili jsme právo *setgid*, takže vytvářené soubory v takto nastaveném adresáři získají skupinové vlastnictví podle adresáře, nikoliv podle uživatele, který soubor vytváří.

- Přepněte se do sdíleného adresáře. Změňte uživatele na *uzivatel* a vytvořte další soubor ve sdíleném adresáři. Zkontrolujte skupinu nově vytvořeného souboru pomocí příkazu `ls -l` (skupina by měla být *skupina_uzivatelu*). Příklad výstupu je uveden níže.

Poznámka: pokud vám nelze s adresářem pracovat, ověřte, že u práv adresáře máte s (malé s) – `rws` u skupiny. Pokud zde máte S (velké S) – `rwS`, tak jste po přidání bitu (g+s) provedli ještě změnu skupiny, což je neplatný stav. Adresář smažte a postup zopakujte ve správném pořadí, tj. nastavení skupiny a pak až bitu.

```
-rw-rw-r--. 1 uzivatel skupina_uzivatelu 5 30. zar 17.47 druhy_soubor
-rw-rw-r--. 1 uzivatel uzivatel  0 30. zar 17.28 soubor
```

- Jako *uzivatel2* ověřte, zda může zapisovat do předešle vytvořeného souboru uživatelem *uzivatel*.
- Dále vytvořte jako *uzivatel2* třetí soubor a pomocí výpisu si ověřte, že patří skupině *skupina_uzivatelu*.
- Ověřte, že *uzivatel* může zapisovat do třetího souboru
- Přepněte se do uživatele *student*.

4.3.3 Samostatný úkol

- V adresáři `/home/student` vytvořte adresář `sdileny_adresar2`. Zařiďte, že do této složky budou mít plný přístup **uživatelé ze skupiny *skupina_uzivatelu*** (*uzivatel* a *uzivatel2*). Jiní uživatelé (mimo vlastníka) *nebudou mít žádný přístup k adresáři*. Práva vlastníka adresáře ponechejte na původní hodnotě. Nastavte také u adresáře právo *setgid*.
- Vytvořte dalšího uživatele *uzivatel3*, abyste mohli vyzkoušet, že do adresáře nemá přístup. Tento uživatel **nebude** členem skupiny *skupina_uzivatelu*.
- Vstupte do vytvořeného adresáře.

- Přepněte se do uživatele *uzivatel3* a zkuste vytvořit v adresáři soubor.

```
bash: /home/sdileny_adresar2/soubor: Operace zamitnuta
```

- Přepněte se do uživatele *uzivatel* (je potřeba se přepnout z uživatele *student*). Jako uživatel *uzivatel* vytvořte v adresáři soubor. Dokažte, že *uzivatel2* může do souboru zapisovat. Dokažte, že *uzivatel3* nemůže do souboru zapisovat.

Poslední bod řešení samostatného úkolu zobrazte vyučujícímu.

5 POČÍTAČOVÉ CVIČENÍ Č. 3 – MODUL JÁDRA

5.1 Účel cvičení

Podpořit probíranou látku základní stavby operačních systémů. Demonstrovat funkci monolitického jádra a možnosti jeho rozšiřování pomocí modulů. Seznámit s kompilací zdrojového kódu.

5.2 Teoretický úvod

Funkce monolitického jádra lze rozšiřovat pomocí modulů. Tyto moduly mohou například řešit podporu souborových systémů nebo to mohou být ovladače zařízení, které umožňují jádru použít daný hardware. Bez těchto modulů bychom museli pro práci s novým souborovým systémem či hardware provést kompilaci jádra a novou funkčnost přidat přímo do binárního souboru s jádrem. Moduly mohou být vloženy a odebrány z jádra na dynamicky za běhu systému (tj. není potřeba jádro a tedy celý operační systém restartovat). Bližší popis monolitického jádra Linux lze například nalézt v publikaci [5].

5.3 Pokyny pro vypracování

5.3.1 Vytvoření jednoduchého modulu

Modul jádra musí obsahovat tyto části:

- odkaz na hlavičkový soubor *module.h*,
- inicializační funkci *int init_module()*, která vrací hodnotu 0, pokud nedošlo k chybě,
- ukončovací funkci *void cleanup_module()*.

Aby vytvářený modul vůbec něco dělal, použijeme systémové volání *printk()*, které funguje obdobně jako standardní funkce *printf()* z jazyka C. Rozdíl spočívá v tom, že text se nevypisuje do terminálu, ale do **systémového logu**, který zobrazíme pomocí příkazu *dmesg*. Pro zobrazenou hlášku lze uvést i prioritu, která je definována v hlavičkovém souboru `/lib/modules/.../source/include/linux/kern_levels.h` takto:

```
KERN_EMERG KERN_SOH "0" /* system is unusable */
KERN_ALERT KERN_SOH "1" /* action must be taken immediately */
KERN_CRIT KERN_SOH "2" /* critical conditions */
KERN_ERR KERN_SOH "3" /* error conditions */
KERN_WARNING KERN_SOH "4" /* warning conditions */
KERN_NOTICE KERN_SOH "5" /* normal but significant condition */
KERN_INFO KERN_SOH "6" /* informational */
KERN_DEBUG KERN_SOH "7" /* debug-level messages */
```

Nejzávažnější hlášení mají prioritu 0, méně závažné mají vyšší prioritu.

> Funkce *printk()* není určena pro komunikaci s uživatelem, ale pro předávání informací o činnosti jádra. Pokud například zapíšeme zprávu s prioritou KERN_DEBUG, uživatel se ji nedozví, ale může se k ní dostat vývojář, který bude hledat v kódu chyby. Je vhodnější použít konstanty KERN_*, než psát prioritu konkrétními čísly.

Do modulu také přidáte licenci GPL (General Public License). Údaj není povinný. Nemusíme se příliš zabývat rozdíly mezi různými licencemi. Důležité je, jestli je licence svobodná, nebo proprietární. Proprietární nebo žádná licence modulu „poskvrní“ jádro a v systémovém logu uvidíte hlášku `module licence <nazev> taints kernel`. Tento mechanizmus umožňuje uživateli/vývojáři zkontrolovat, jestli má výhradně svobodný software v jádře a komunitě umožňuje nezabývat se řešením problémů jádra se součástmi (moduly), které nejsou svobodné. Pokud licence není uvedena v modulu vůbec, je modul považován za **nesvobodný**! Pokud dojde k poskvrnění jádra, nestačí modul odebrat – je nutné jádro spustit znovu (to znamená restartovat).

- Otevřete terminál. Pracujte ve svém domovském adresáři. Pomocí editoru *vi* vytvořte soubor `hello.c` s následujícím obsahem.

```
#include <linux/module.h>  /* musi obsahovat kazdy modul */
#include <linux/kernel.h>  /* je v~nem definovan KERN_ALERT */
MODULE_LICENSE("GPL");
int init_module(void)
{
printk(KERN_SOH "1" "Hello world.\n"); /* priorita uvedena rucne */

return 0; /* nenulova hodnota znamena chybu */
}
void cleanup_module(void)
{
printk(KERN_ALERT "Goodbye world.\n");
}
```

Poznámka: při vytváření tohoto souboru (i dalších) dbejte na správné použití mezer.

5.3.2 Kompilace, vložení modulu do jádra, odstranění

Vytvoření souboru `Makefile`

- Pomocí editoru *vi* vytvořte soubor `Makefile` (velké písmeno M, jedno slovo). Tento soubor bude sloužit pro kompilaci modulu. Do souboru vložte následující text:

```
obj-m := hello.o
KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)
default:
	$(MAKE) -C $(KDIR) M=$(PWD) modules
clean:
	$(MAKE) -C $(KDIR) M=$(PWD) clean
```

Pozor: Přípona souboru `hello` je písmeno „o“. Dále na místě `__` musí být jeden znak TAB.

Blíže si vysvětlíme, k čemu jednotlivé příkazy slouží. Nejprve jsme definovali příkazem `obj-m`, které objekty chceme vytvořit jako modul. Další přípustné varianty jsou `obj-y` a `obj-n`. Možnosti `y, n, m` udávají volby konfigurace jednotlivých součástí jádra následovně:

- budou **pevnou součástí jádra** (`y`),
- **nebudou v jádře** vůbec (`n`),
- budou jako samostatný **modul** (`m`).

Objekty `obj-n` bude program *make* ignorovat. Objekty `obj-y` budou kompilovány jako součást binárního souboru jádra. Objekty `obj-m` program *make* kompiluje jako samostatné moduly s příponou `.ko`.

V souboru `Makefile` dále definujeme dvě proměnné – `KDIR`, která odkazuje na adresář s hlavičkovými soubory aktuálně spuštěného jádra (výraz v závorce bude nahrazen číslem verze spuštěného jádra), dále proměnnou `PWD`, což je aktuální adresář. Následuje definice výchozího (default) cíle (příkaz, který se má vykonat, pokud voláme *make* bez parametrů).

Program *make* tedy spustí další instanci programu *make* s parametrem `-C` a cestou k hlavičkovým souborům jádra (KDIR), s parametrem `-M` a aktuálním adresářem (PWD) a nakonec je uveden způsob kompilace – `modules`.

Další účel kompilace, který jsme definovali, je `make clean`. Tento účel kompilace funguje obdobně. Opět zavolá další instanci programu *make* a přidá parametry s cestou k hlavičkovým souborům a aktuálnímu adresáři. Dále přidá akci `clean` pro vymazání souborů, které vznikly při překladu.

Kompilace zdrojového kódu

- Zadejte příkaz `make` (bez parametrů). Měl by následovat tento výpis

```
make -C /lib/modules/<verze jadra>/build M=/home/student/modul modules
make[1]: Entering directory '/usr/src/kernels/<verze jadra>'
  CC [M]  /home/student/modul/hello.o
  Building modules, stage 2.
  MODPOST
  CC      /home/student/modul/hello.mod.o
  LD [M]  /home/student/modul/hello.ko
make[1]: Leaving directory '/usr/src/kernels/<verze jadra>'
```

Na prvním řádku vidíme, že byl spuštěn příkaz *make* s doplněnými parametry, řádky začínající textem `CC` znamenají spuštění překladače (v našem případě *gcc*), `LD` spouští linker, zde *ld*.

- Po úspěšné kompilaci si prohlédněte vytvořené soubory. Soubor `hello.ko` obsahuje nově vytvořený samostatný modul jádra.
- Smažte soubory vytvořené při kompilaci příkazem `make clean`.
- Zkontrolujte, že soubory byly vymazány.
- Proveďte znovu kompilaci modulu.
- Nově vytvořený modul prozkoumejte příkazem

```
[]$ /sbin/modinfo hello.ko
filename:       hello.ko
license:        GPL
srcversion:     DF8DE965DEFED469EB1E13B
depends:
vermagic:       <verze jadra> SMP mod_unload 686 ...
```

Ve výpisu vidíme licenci (GPL), se kterou byl modul zkompilován.

- Nyní vytvořený modul vložte do jádra.

```
[]$ sudo /sbin/insmod hello.ko
```

- Nahlédněte do systémového logu pomocí příkazu `dmesg | tail`. Na konci výpisu uvidíte text „Hello world“, což je účelem práce modulu, viz jeho zdrojový kód.
- Dále se podívejte, že je modul přítomen v jádře pomocí příkazu *lsmod*.

```
[]$ /sbin/lsmod | head
Module      Size  Used by
...
hello       908   0
...
```

- Nyní modul odstraňte z jádra pomocí příkazu

```
[]$ sudo /sbin/rmmod hello
```

- Přesvědčte se pomocí příkazu *lsmod*, že modul byl odstraněn.
- Náhledem do systémového logu systému příkazem `dmesg | tail` ověřte, že uvidíte text „Goodbye world“. Jedná se opět o účel práce modulu, viz jeho zdrojový kód.

> Poznámka: Proč jednou píšeme příponu `.ko` a podruhé ne? Příkaz *insmod* vyžaduje jako parametr vždy název souboru, a ne název modulu. Příkaz *rmmod* vždy prohledává seznam modulů, které jsou aktuálně nahrané přímo v jádře, akceptuje jak název bez přípony, tak s příponou `.ko`. My ji nepíšeme jednoduše z toho důvodu, že je to kratší. Alternativou k *insmod* je příkaz *modprobe*, který příponu `.ko` neakceptuje, dokáže automaticky načíst i moduly, na kterých je zaváděný modul závislý a nemohl by jinak

fungovat. Příkaz *modprobe* hledá pouze mezi moduly, které se nachází v cestě `/lib/modules/...`. Pokud bychom chtěli přidat vlastní modul, nakopírovali bychom ho kamkoliv do této cesty, nejlépe do adresáře `misc`, a následně spustili příkaz `depmod -a`, který projde celý adresář `/lib/modules/...` a zapíše si seznam všech modulů, které jsou k dispozici, a jejich vzájemné závislosti. Od tohoto okamžiku bude *modprobe* fungovat s novým modulem.

5.3.3 Samostatný úkol

Jako samostatný úkol proveďte stažení zdrojového kódu modulu jádra `procfs.c` a proveďte jeho kompilaci. Tento modul realizuje v jádře funkci počítadla pomocí čtení ze souboru ve virtuálním souborovém systému `procfs`, který je dostupný v adresáři `/proc`. Soubory v tomto virtuálním souborovém systému nejsou reálně uloženy na paměťovém médiu, ale jejich obsah je **generován dynamicky** na základě požadavku přístupu k datům souboru.

Ve staženém zdrojovém kódu modulu `procfs.c` zařiďte následující:

- Modul je nesvobodný (viz popis licence GPL).
- Autorem modulu jste vy (MODULE_AUTHOR).
- Hlášení do systémového logu o navýšení počítadla má prioritu KERN_NOTICE .

Zkompilujte soubor `procfs.c` (přidejte `obj-m` do `Makefile`). Vložte nový modul do jádra (pokud vkládáte modul opětovně, je potřeba předchozí modulu z jádra vyjmout). V adresáři `/proc/bsos` vznikne soubor `pocitadlo`. Pokud budete prohlížet obsah souboru `pocitadlo`, bude se v něm zvyšovat číslo (na základě každého požadavku čtení).

Dále proveďte následucjící:

- Několikrát zobrazte obsah souboru `pocitadlo`.
- Dokažte, že autorem zkompilovaného modulu jste vy.
- Dokažte, že hlášení o navýšeni počítadla má požadovanou prioritu – systémový log zobrazte s parametrem pro zobrazení závažnosti hlášení, tedy `dmesg -x`.
- Dokažte, že vámi vytvořený modul je nesvobodný a poskvrní tak jádro.

6 POČÍTAČOVÉ CVIČENÍ Č. 4 – VIRTUALIZACE

6.1 Účel cvičení

Podpořit probíranou látku architektury operačních systémů – virtualizace. Využití virtualizaci pro bezpečné hrátky s bezpečností.

6.2 Teoretický úvod

6.2.1 Virtuální prostředí

Virtualizovat lze na několika úrovních. V případě virtualizace na úrovni operačního systému je pro virtualizaci využito jádro hostitelského operačního systému. Toto jádro umožňuje provoz nezávislých virtuálních počítačů, které se označují jako **virtuální prostředí** nebo **kontejnery**. Tyto virtuální prostředí/kontejnery se typicky jeví pro jejich uživatele jako reálný operační systém. Primární výhodou je oddělený běh aplikací v jednotlivých virtuálních prostředích a také oddělení uživatelů (z důvodu bezpečnosti), kteří v nich pracují.

Jednou z možných realizací virtuálních prostředí je pomocí programu *chroot* (change root), který lze provozovat na většině operačních systémů s jádrem Linux. Tento program mění kořenový adresář pro běžící procesy. Procesy běžící v tomto novém prostředí nemohou přistupovat k souborům (a tudíž i k prostředkům) mimo svůj vymezený adresářový prostor. Jsou tedy „uvězněny“ (anglicky označováno jako „chroot jail“) v nové adresářové struktuře.

6.2.2 Uživatelské účty

Každý uživatel v systému má svůj účet, který je přístupný pomocí uživatelského jména a hesla. Všechny soubory v systému jsou přiděleny k nějakému účtu. Účty dělíme na uživatelské a systémové. Tyto účty mohou být:

- Osobní – Patří konkrétní osobě.
- Administrátorské (root) – Účet správce.
- Systémové – Nevlastní je konkrétní osoba a nepoužívají se tak pro přihlášení do systému. Jsou používány pro provoz operačního systému.

Vytvoření účtů provádí správce systému. Každý účet má svůj záznam v souboru `/etc/passwd`. Tento soubor obsahuje záznamy (jeden řádek pro každý účet), které specifikují pro každý účet jeho atributy, jako je uživatelské jméno, skutečné jméno uživatele atd. Každý záznam v tomto souboru má tvar `jmeno:heslo:uid:gid:popis:adresar:shell`. Pro správce systému může záznam vypadat takto:

```
1 root:x:0:0:Frantisek Dobrota:/root:/bin/bash
```

Jedná se o účet se jménem *root*, který používá chráněná hesla. Chráněná hesla jsou typicky uložena v souboru `/etc/shadow`. Použití chráněných hesel je označeno znakem `x`. Číselná identifikace uživatele – UID (User Identification) – je rovna 0 stejně jako identifikace skupiny, do které uživatel *root* patří – GID (Group Identification). Další položka je komentář (celé jméno, telefon, číslo kanceláře apod.). Následuje domovský adresář `/root` a implicitní interpret pro zadávané příkazy – `/bin/bash`.

Nový účet je možné založit několika způsoby. Nejelementárnější cesta vede přes příkaz *vipw*, který edituje přímo soubor `/etc/passwd`. Snazší nastavení umožňuje dvojice nástrojů *adduser* a *useradd*. Uživatelsky nejméně náročné jsou grafické nadstavby, které se však liší pro různé distribuce a nebudeme se jimi zabývat (jejich ovládání je intuitivní, nehodí se však pro dávkové zpracování ve skriptech).

6.2.3 Uživatelské skupiny

Používání uživatelských skupin usnadňuje organizaci více účtů, které sdílí stejná práva. Každý soubor v systému má jak vlastníka, tak i vlastnickou skupinu uživatelů. Záznamy o skupinách jsou uloženy v souboru `/etc/group`, kde jsou uvedeny záznamy ve tvaru `jmeno:heslo:gid:clenove`.

```
1 root:x:0:root
```

Jméno skupiny může být stejné jako jméno účtu. Členů skupiny může být více a oddělují se čárkami.

6.2.4 Autentizace a autorizace uživatele

Autentizace je proces pro ověření **identity** uživatele, například pomocí znalosti názvu uživatelského účtu a hesla. Po úspěšné autentizaci následuje autorizace, která povoluje/-zakazuje provedení určité operace, například se souborem podle přidělených práv.

Pokročilejší informace o bezpečnosti v OS Linux lze například nalézt v publikacích [6] a [7].

6.3 Pokyny pro vypracování

6.3.1 Příprava pracovního prostředí

V této úloze bude zprovozněno virtuální prostředí (kontejner) pomocí programu *chroot*. V tomto virtuálním prostředí bude spuštěn **další operační systém**, který bude nainstalován do adresáře `~/chroot`.

> Samotný příkaz *chroot* změní kořenový adresář a spustí implicitní shell `/bin/bash`. V původní adresářové hierarchii to tedy je `~/chroot/bin/bash`.

- Stáhněte si soubor `chroot_linux.tgz` (80 MiB) a skript *chroot.sh*. Archiv `chroot_linux.tgz` obsahuje zkomprimovanou kompletní distribuci OS Linux (CentOS

Docker Container Image), která bude základem pro další práci s uživatelskými účty. Změny provedené v prostředí *chroot* neovlivní činnost hostitelského systému. Skript `chroot.sh` se využije k přepnutí do nové distribuce.

- Soubor `chroot.sh` zkopírujte do adresáře `/var`.
- Archiv `chroot_linux.tgz` nakopírujte do adresáře `chroot` ve vašem domovském adresáři. Ten již musí existovat, neboť archiv jej neobsahuje. Adresář musí být před rozbalením prázdný (**případné předchozí pokusy s virtuálním operačním systémem odstraňte**). Archiv rozbalte následujícím příkazem.

```
[]$ sudo tar xvpf chroot_linux.tgz
```

Rozbalení neprovádějte pomocí nástroje v grafickém prostředí. Nástroj grafického prostředí nerozbalí archiv správně a virtuální systém pak nebude pracovat.

- Příkazem *chroot* změňte kořenový adresář systému.

```
[]$ sudo sh /var/chroot.sh
```

- Úspěšnou změnu kořenového adresáře (a tedy vstup do virtuálního prostředí s novým operačním systémem) zjistíte změnou promptu. Změnu uživatele ověříte příkazem *id*. Nově budete vystupovat jako **uživatel *root***.

```
uid=0(root) gid=0(root) groups=0(root)
```

> Zpět do původního systému se dostanete příkazem *exit* (tím se ukončí shell spuštěný příkazem *chroot*). Zpět do prostředí *chroot* se v případě nutnosti dostanete opět přes `sudo sh /var/chroot.sh`.

6.3.2 Vytvoření nového uživatele – *vipw*

- Přímou editací souboru `/etc/passwd` přes *vipw* (bez parametrů) přidejte nového uživatele tak, že na konec seznamu uživatelů přidáte následující záznam:

```
student:!!:1000:1000:Josef Slicny:/home/student:/bin/bash
```

> Po spuštění příkazu *vipw* se zkontroluje, zda v téže chvíli needituje soubor `passwd` někdo jiný. Pokud ne, spustí se implicitně editor *vi*. Účty, u kterých je v druhém poli (oddělovačem je dvojtečka, viz úvod) znak „*“ (hvězdička) nebo „!!“ (dva vykřičníky), jsou zablokovány a nelze se jejich prostřednictvím do systému přihlásit.

- Přímou editací souboru `/etc/group` příkazem *vigr* (bez parametrů) vytvořte také novou skupinu. Je to ekvivalentní postup k výše použitému *vipw*, pouze formát záznamu je odlišný:

```
student:x:1000:
```

> Položka heslo se v záznamech souboru `group` používá jen zřídka. V této úloze s hesly ke skupinám nebudeme pracovat.

- Uživatel *student* má mít domovský adresář `/home/student`. Vytvořte jej. Při přihlášení se automaticky nastaví obsah proměnné HOME na tento adresář.
- Upravte práva adresáře tak, aby jej vlastnil uživatel *student* včetně skupiny.

```
[]# chown -R student:student /home/student
```

- Uživatel student dosud nemá přiřazeno heslo pro přístup ke svému účtu (autentizace heslem). Zvolte z výukových důvodů „slovníkové“ heslo „Smetana“. **Heslo je nutné zadat přesně tak, jak je uvedeno, tj. s velkým počátečním písmenem.**. Toto slovo je příkladově použito pro zdůraznění **faktu, že smetana s malým „s“ ve slovníku není**, a tudíž by se útok nezdařil.

```
[]# passwd student
```

Program *passwd* automaticky kontroluje, zda heslo není příliš krátké nebo příliš jednoduché – viz varovné hlášení „bad password“, které **v tomto cvičení ignorujte**.

- Taktéž nastavte heslo uživateli *root*. Jako výstražný příklad nastavte obdobně „složité“ slovníkové heslo – „Ochrana“.

```
[]# passwd
```

Poznámka: Pokud vám nepůjde nastavit heslo, tak chyba je pravděpodobně v nesprávném rozbalení archívu `chroot_linux.tgz`, viz výše.

> Takto jednoduchá hesla **nikdy** (mimo tuto cvičnou situaci) nevolte.

> Není-li u příkazu *passwd* uvedeno jméno účtu, automaticky se mění nastavení hesla aktuálního uživatele. Jelikož aktuálně pracujete jako *root*, tak měníte heslo tomuto uživateli.

- Zkontrolujte změnu hesel v souboru `/etc/shadow`. Do souboru jsou ukládána chráněná (stíněná) hesla, tudíž se v druhém poli záznamu objeví posloupnost znaků – zašifrované heslo.

```
[]# grep ^root /etc/shadow
[]# grep ^student /etc/shadow
```

> Příkaz *grep* se používá k hledání hodnot v textu podle zadaného výrazu. Znak `^` značí začátek textu; tedy `^root` označuje text začínající s *root*. Znak `\` uvozuje metaznak `|` (jinak by byl brán jako součást výrazu). Znak `|` slouží pro vyhledání textu vyhovujícímu prvnímu nebo druhému výrazu.

- Ověřte, že soubor `/etc/shadow` není čitelný pro ostatní uživatele, (např. `stat /etc/shadow`).
- Pomocí příkazu *vipw* vytvořte nového uživatele. Heslo bude tentokrát uloženo do souboru `/etc/shadow` – viz `x` v druhém poli záznamu.

```
ucitel:x:502:502:Viktor Cabadaj:/home/ucitel:/bin/bash
```

6.3.3 Vytvoření nového uživatele – *useradd*

Vytvoření uživatelského účtu vyžaduje provedení několika kroků – je to přidání záznamu do souboru `/etc/passwd`, vytvoření domovského adresáře uživatele a případně nastavení jeho konfiguračních souborů (v domovském adresáři). S nástrojem *useradd* se tyto kroky zjednoduší.

- Vytvořte uživatelský účet, který bude ve stejné skupině jako uživatel *student*. Ověřte existenci příslušného záznamu v `/etc/passwd`.

```
[]# useradd -c "Frantisek Koudelka" -g student -m -n studentB
```

> Poté, co se vytvoří uživatelský účet, jsou z adresáře `/etc/skel` zkopírovány soubory do domovského adresáře nového uživatele. V pracovním systému je tento adresář prázdný, obvykle však obsahuje základní konfigurační soubory pro uživatele.

- Nastavení hesla uživatele *studentB* proveďte stejně jako u předchozích uživatelů. Heslo zvolte **stejné jako je název účtu**, tedy „studentB“.

6.3.4 Slovníkový útok

Jednou z používaných strategií pro prolomení hesla je výběr slov ze slovníku popřípadě jejich kombinace podle často užívaných tvarů. Jiným přístupem je útok hrubou silou. Ten spočívá ve zkoušení všech kombinací znaků.

Součástí nového virtuálního operačního systému je program *John the Ripper*, pomocí kterého lze provést slovníkový útok na hesla. Nejprve je potřeba získat přístup k informacím o účtech v systému. Nepoužívají-li se chráněná hesla, stačí k útoku jediný soubor – `/etc/passwd`.

- Jako uživatel *root* proto převeďte příkazem `pwunconv` účty tak, aby hesla byla dostupná v tomto souboru.

> Příkaz *pwunconv* „přesune“ hesla ze souboru `/etc/shadow` do `/etc/passwd`. Jaká je výhoda chráněných hesel? Soubor `/etc/passwd` je **čitelný pro všechny uživatele**. I když se spoléhá na nereverzibilní proces převodu otevřeného hesla do zašifrované podoby jednosměrnou funkcí MD5, z důvodu možnosti útoků hrubou silou je mnohem bezpečnější oddělit názvy účtů (v `/etc/passwd`) od vlastních hesel (v `/etc/shadow`), která může číst pouze administrátor.

- Přidejte právo zápisu a vstupu pro všechny do adresáře `/etc`.
- Přepněte se na uživatele *student*.

Poznámka: pokud vám nepůjde se přepnout do uživatele student, tak se vám nepodařila změna práv k adresáři `/etc`.

- Zobrazte si slovník slov v souboru `/usr/share/dict/words`, která budou použita pro lámání hesla.
- Spusťte program *john* – jako parametr mu zadejte slovník s českými slovy `/usr/share/dict/words` a soubor s hesly `/etc/passwd`.

```
[]$ john --wordlist=/usr/share/dict/words /etc/passwd
```

> Pokud mají uživatelské účty hesla dle zadání, budou prolomena za nepříliš dlouhou dobu (řádově desítky sekund). Postupně se prochází slovník `words`, jenž je standardní součástí distribuce. Slovníky pro různé jazyky jsou dostupné na Internetu. Prolomená hesla po skončení běhu je možné získat následujícím způsobem: `john -show /etc/passwd`.

- Vypište zjištěné heslo uživatele *root*. Vypište také zjištěné heslo uživatele *student*.

```
[]$ john --show --users:root /etc/passwd
```

6.3.5 Samostatný úkol

Předchozí zjištění hesel selhalo pro uživatele *studentB*. Jako samostatný úkol proveďte zjištění tohoto hesla. K tomu využijte následující postup:

- *John the Ripper* má vlastní nástroj, který kombinuje údaje ze souborů `/etc/shadow` a `/etc/passwd` (provádí stejnou operaci jako příkaz *pwunconv*). Nejprve proto vytvořte znovu stínová hesla příkazem *pwconv*. K tomu potřebujete práva administrátora, proto opusťte účet *student*.

```
[]$ exit
[]# pwconv
```

- Následujícím příkazem se provede převod zpět. K tomuto již běžně dojít nemůže, protože běžný uživatel nemá práva ke čtení `/etc/shadow`. Výsledek si uložte do souboru `/tmp/hesla.txt`.

```
[]# unshadow /etc/passwd /etc/shadow > /tmp/hesla.txt
```

- Přepněte se do uživatele *student*. Dále se přepněte do adresáře `/etc`. Spusťte program *john* v režimu luštění, který jako možná hesla použije údaje z komentáře účtu a **samotný název účtu**. Tento režim spustíte pomocí parametru `-single`. Jako další parametr uveďte soubor `/tmp/hesla.txt`, který jste si předešle vytvořili.
- Vyučujícímu zobrazte prolomená hesla všech uživatelů, která jsou uložená v souboru `~/.john/john.pot`.

7 POČÍTAČOVÉ CVIČENÍ Č. 5 – PROCESY

7.1 Účel cvičení

Demonstrovat základní práci s procesy. Cvičení je zaměřeno na praktické využití. Je provedena synchronizace dat mezi počítači, což zahrnuje i použití síťové části operačních systémů.

7.2 Teoretický úvod

7.2.1 Automatické spuštění procesů

Pro automatické spuštění procesů v zadaném čase slouží program *cron*. Tento program v zadaných časových intervalech spustí určený program. Časové intervaly a programy ke spuštění jsou definovány v konfiguračním souboru `/etc/crontab`. Pro práci s tímto souborem slouží stejnojmenný příkaz *crontab*.

7.2.2 Telnet

Telnet je klasickou službou vzdáleného přihlášení. Klientský program *telnet* slouží k navázání spojení se vzdáleným systémem – serverem, na němž běží proces/démon *telnetd*. Pro připojení na vzdálený server je třeba se autentizovat (přihlásit) uživatelským jménem a heslem. Nevýhoda programu *telnet* spočívá v tom, že komunikuje nešifrovaně. V prostředí veřejného Internetu tento způsob připojení není bezpečný a proto se nepoužívá (by se neměl používat). Telnet se pro svou jednoduchost používá v privátních sítích bez napojení na vnější síť.

7.2.3 Secure Shell

Služba SSH (Secure Shell) má oproti službě Telnet výhodu v tom, že je **zabezpečená** (šifrovaná). SSH komunikuje opět v režimu klient-server. Je to komplexní služba, která umožňuje vzdálené přihlášení a také přenos dat přes šifrovaný kanál. Ke vzdálenému přihlášení se používá klient *ssh*. Ten nemá na rozdíl od programu *telnet* příkazový režim. Autentizovat se lze pomocí uživatelského jména a hesla. Také lze autentizaci provést na základě vygenerovaných klíčů (tajný a veřejný). Služba SSH je vhodná pro použití ve veřejném Internetu. K přenosu dat mezi vzdálenými počítači slouží příkaz *scp*. Tento příkaz funguje podobně jako příkaz pro lokální kopírování dat *cp*.

7.2.4 File Transfer Protocol

FTP (File Transfer Protocol) je služba pro přenos dat mezi počítači. Pracuje opět v režimu klient-server. Před vlastním přenosem je nutno provést přihlášení pomocí uživatelského jména a hesla. Stejně jako program *telnet* má i program *ftp* svůj interaktivní příkazový režim. FTP komunikuje nešifrovaně, což je bezpečnostní nedostatek.

Další informace o možnostech automatizace úkonů pomocí skriptů v příkazovém řádku, včetně praktických příkladů, lze například nalézt zde [8].

7.3 Pokyny pro vypracování

7.3.1 Autentizace a přenos dat pomocí SSH

Vzdálené přihlášení prostřednictvím uživatelského jména a hesla

- Přihlaste se ke vzdálenému počítači prostřednictvím služby SSH. Jako vzdálený počítač zvolte jeden z počítačů v učebně. Jako `<ucet>` použijte studentský účet. Vzdálený počítač definujete **vnější IP adresou** daného stroje (tedy adresou hlavního OS), kterou nahradíte text `<pocitac>`.

```
[]$ ssh -p 2222 <ucet>@<pocitac>
```

V laboratoři je pro připojení do virtualizovaného OS použit překlad portů na hostitelském OS. Číslo překládaného portu je 2222. Použití jiného portu než standardního pro službu SSH (port 22) je vyvoláno parametrem `-p`. Bližší informace o konfiguraci přesměrování portů pro vzdálené připojení pomocí SSH do virtualizovaného OS je uvedeno v příloze C.

Po úspěšném přihlášení je mezi místním a vzdáleným počítačem vytvořeno šifrované spojení. Pomocí něj je pak možné na vzdáleném systému pracovat podobně, jako kdybychom byli na počítači přihlášeni lokálně.

- Po přihlášení vyzkoušejte práci na vzdáleném počítači (pohyb v adresářové struktuře apod.).

Odhlaste se ze vzdáleného počítače pomocí příkazu *exit*.

7.3.2 Autentizace pomocí tajného klíče a její využití

- Vygenerujte si pomocí programu *ssh-keygen* dvojici klíčů, které budou později sloužit k autentizaci. Všechny výzvy programu ponechte prázdné a potvrďte je klávesou [Enter] (tedy nezadávejte další heslo pro generované klíče).

```
[]$ ssh-keygen -t ssh-rsa
```

Program *ssh-keygen* vytvoří automaticky dvojici klíčů, jeden veřejný a druhý tajný. Za parametrem `-t` je uveden typ klíče, zde tedy RSA. Pokud při vytváření klíče nezadá uživatel jinak, tajný klíč se uloží do souboru `id_rsa` (na jiných linuxových distribucích může být tento soubor nazván např. `identity`). Program také umožňuje zadat heslo k tajnému klíči, které ale není povinné. Uživatel se může tedy vzdáleně

přihlašovat pouze na základě vlastnictví tajného klíče. Veřejný identifikační klíč se uloží do souboru `id_rsa.pub` (jak lze odhadnout, označení *pub* znamená „public“). Tento veřejný klíč je potřeba přenést na stanici, ke které se bude v budoucnu přihlašovat pomocí tajného klíče.

- Vypište si obsah adresáře `~/.ssh`, který dvojici vygenerovaných klíčů obsahuje, a zobrazte si také obsah souboru s tajným a veřejným identifikačním klíčem.

- Vygenerovaný veřejný identifikační klíč v `id_rsa.pub` přeneste na vzdálený počítač. Jako vzdálený počítač zvolte jeden z počítačů v učebně. Soubor přejmenujte na **číslo vašeho počítače nebo vaše jméno**, např. `26.pub`. Tuto akci provádíte z důvodu možné kolize souborů od několika uživatelů na stejném počítači, které by se navzájem přepisovaly.

```
[]$ scp -P 2222 ~/.ssh/id_rsa.pub <ucet>@<pocitac>:~/.ssh/<cislo
   pocitace nebo jmeno>.pub
```

U příkazu *scp* se překlad portu definuje velkým „P“. Proč je u příkazu *ssh* pro překlad portu použito malé „p“ a zde velké „P“ je dobrá otázka a jednoduchá odpověď může být „pro zmatení nepřítele“ (potažmo studenta). Pouze informativně a pro zajímavost – nahlédnutím do manuálových stránek příkazu *scp* se můžeme dočíst pravý důvod: „*Pro tuto volbu je použito velké písmeno ‘P’, protože význam volby -p (zachování časů a práv souboru) byl převzat z programu rcp(1)*“.

Poznámka: Pokud vám kopírování souboru nepůjde provést z důvodu, že na vzdáleném počítači není vytvořen adresář `.ssh` v domovském adresáři uživatele, tak se nejprve připojte na vzdálený počítač a tento adresář vytvořte. Je potřeba také adresáři `.ssh` změnit jeho standardní práva po vytvoření: `chmod go-rw ~/.ssh`.

- Na **vzdáleném** počítači přidejte obsah přeneseného souboru na konec souboru `authorized_keys` (v adresáři `.ssh`). Pro tuto akci použijte výpis obsahu souboru pomocí příkazu `more` a **přidání** provedeného výpisu na konec souboru `authorized_keys` pomocí `>>` (například `more 26.pub >> authorized_keys`). Přidáním dat soubor `authorized_keys` rovnou vytvoříte (pokud předešle neexistoval). V případě, že váš klíč do souboru nepřidáte, ale soubor **přepíšete**, tak můžete **smazat** veřejné klíče jiných studentů, již předešle uložených v tomto souboru.

- Dále změňte práva tohoto souboru tak, abyste měli k němu **přístup pouze vy**.

```
[]$ chmod go-rw authorized_keys
```

- Nyní by mělo být možné přihlašovat se na vzdálený počítač bez zadávání hesla. Vyzkoušejte, že se můžete přihlásit pouze pomocí klíče.

7.3.3 Ruční synchronizace dat mezi počítači

Synchronizaci dat mezi počítači lze provádět pomocí programu *rsync*, který data aktualizuje na základě **detekce jejich změn**. Nepřenáší tedy při každé synchronizaci všechna data. Přenos dat je šifrovaný pomocí SSH.

- V domovském adresáři vytvořte adresář pro synchronizaci s názvem `data`. Do adresáře vyrobte několik souborů (například příkazem `touch <nazev souboru>`, kterým vytvoříte prázdný soubor).
- V domovském adresáři zhotovte skript s názvem *sync.sh* s **obsahem ve výpisu níže**. Opět jako vzdálený počítač zvolte vybraný počítač v učebně. Část názvu adresáře pro zálohu na vzdáleném počítači **pojmenujte označením vašeho počítače**, například `zaloha_26`.

Pozor: z důvodu následného automatického provádění synchronizace je nutno zvolit vzdálený počítač, na který jste **předešle přenesli** vygenerovaný veřejný klíč.

```
rsync -a --delete --progress --rsh='ssh -p 2222' ~/data
   <ucet>@<pocitac>:~/zaloha_<cislo vaseho pocitace nebo jmeno>
```

Poznámka: Příkaz je uveden na jednom řádku (pouze z důvodu šířky stránky je text rozdělen na dva řádky). Znak ' je apostrof (ne obrácená jednoduchá uvozovka).

> Parametry příkazu *rsync* značí následující: `-a` značí mód archivace (tj. vytvoření „kompletního zrcadla“), `--delete` smaže soubor u cíle pokud je soubor smazán u zdroje, `--progress` zobrazuje průběh přenosu.

- Vytvořený skript spusťte. Tímto se provede prvotní synchronizace všech souborů.
- Připojte se pomocí SSH na zvolený vzdálený počítač a ověřte funkčnost synchronizace tj. zda na vzdáleném počítači jsou vaše data ze zálohovaného adresáře `data`. Následně se odpojte.
- U vybraných souborů pro synchronizaci v adresáři `data` změňte jejich obsah.
- Proveďte opětovnou synchronizaci. Všimněte si, že bude synchronizován **jen změněný soubor(y)**. Obdobně zkuste v adresáři `data` nějaký soubor smazat a provést synchronizaci. Tímto ověřte, že synchronizovány jsou pouze změny (přidání, smazání, úprava souboru).

7.3.4 Automatické spuštění programu pro synchronizaci dat

Vytvořenou jednorázovou synchronizaci lze automatizovat. K tomuto účelu bude využit program *cron*, který umožňuje automatické spuštění zvoleného programu v určený čas. V našem případě budeme spouštět již vytvořený skript `sync.sh`.

K použití programu *cron* je potřeba připravit soubor, v němž se určí program a čas jeho spuštění. Pomocí textového editoru vytvořte konfigurační soubor `sync.cron`. Do souboru vložte následující text. Tímto bude provedeno spouštění skriptu každou minutu. Pro jinou frekvenci spouštění je nutno vhodně upravit počátek řádku v tomto souboru.

```
1 * * * * * ~/sync.sh >> ~/zaloha.log
```

Struktura souboru je následující: [minuty] [hodiny] [den v měsíci] [měsíc] [den v týdnu] [program ke spuštění]. Znak „*“ zastupuje všechny možnosti. Pokud jsou použity znaky „*“ na všech pozicích, tj. `* * * * * ~/sync.sh >> ~/zaloha.log`, tak bude k synchronizaci docházet každou minutu, každou hodinu, každý den atd.

Dále je výstup skriptu přesměrován do souboru `zaloha.log`. Tímto lze získat přehled o provedených zálohách (co a kdy bylo zálohováno).

- Vytvořenému skriptu `sync.sh` přiřaďte **příznak pro spuštění** (příkaz `chmod`).
- Nyní je potřeba definovaný soubor oznámit programu *cron*.

```
1 []$ crontab sync.cron
```

- Ověřte zda nová úloha byla k programu *cron* přiřazena. K tomu použijte příkaz `crontab -l`.
- Otestujte funkci automatické synchronizace vytvořením/editací/smazáním některého ze souborů v adresáři `data`. Následně se přihlaste na vzdálený server pomocí SSH a zkontrolujte stav souborů. **Nezapomeňte však počkat na zvolený interval synchronizace.**
- Prohlédněte si soubor `~/zaloha.log` s historií provedených záloh.

Poznámka: Pokud vám automatická záloha **nebude fungovat**, chybu naleznete v emailových zprávách, které vám bude zasílat systém. Informaci o přijetí zprávy uvidíte pomocí zobrazené hlášky v terminálu. Tyto zprávy si lze přečíst po zadání příkazu `mail`. Po zadání příkazu napište číslo posledního emailu pro jeho přečtení. Program *mail* opustíte stiskem znaku [q].

- Ukončete automatické provádění skriptu pomocí příkazu `crontab -r`.

7.3.5 Samostatný úkol

Vytvořte automatizovanou úlohu, která **každou minutu nakopíruje** obsah adresáře `data1` na vzdálenou stanici. Adresář `data1` vytvořte ve vašem domovském adresáři a do adresáře vyrobte několik souborů, které budete kopírovat.

Pro realizaci úlohy je potřeba vytvořit skript, který bude provádět kopírování souborů pomocí příkazu *scp*. Před příkazem *scp* použijte ve skriptu příkaz *date*. Tím bude zřejmé,

kdy ke kopírovaní došlo. Na vzdáleném počítači je také potřeba vytvořit adresář, kam se budou soubory kopírovat (adresář není při kopírování vytvořen automaticky).

```
1 date; scp -P 2222 ~/data1/* <ucet>@<pocitac>:~/kopie_<cislo vaseho
    pocitace nebo jmeno>/
```

Poznámka: Příkaz je uveden na jednom řádku. Znaky `/*` za adresářem `data1` značí, že budete kopírovat všechny soubory z tohoto adresáře.

U příkazu *scp* nezapomeňte na přesměrování portu. Dále nezapomeňte přiřadit vytvořenému skriptu příznak pro spuštění. Činnost skriptu si ověřte pomocí ručního spuštění (soubory z adresáře by se měly nakopírovat na vzdálenou stanici).

Pro spouštění skriptu každou minutu je potřeba vytvořit konfigurační soubor pro program *cron*, ve kterém budete definovat časové intervaly a co se má spustit (tedy váš skript pro kopírovaní).

Záznam o kopírování ukládejte **do logu**, tedy použijte obdobnou konstrukci z postupu vypracování v souboru `sync.cron`. Log s několika záznamy o kopírovaní zobrazte vyučujícímu.

8 POČÍTAČOVÉ CVIČENÍ Č. 6 – KONFIGURACE SLUŽEB

8.1 Účel cvičení

Podpořit probíranou látku souborových systémů a organizace adresářů. Demonstrovat možnosti práce s konfiguračními soubory síťových služeb – DNS a webový server.

8.2 Teoretický úvod

8.2.1 Standard pro organizaci souborů a adresářů

V operačních systémech s jádrem Linux definuje adresářovou strukturu standard **FHS** (Filesystem Hierarchy Standard). Jeho varianty se používají v řadě dalších systémů. Pro jednotlivé adresáře definované tímto standardem je určeno, které soubory mají obsahovat. Adresáře jsou specifikovány jako povinné a volitelné.

V adresáři `/var` se nacházejí soubory s měnícími se daty. Typicky jsou to soubory se záznamem o běhu operačního systému a jeho služeb (tzv. logy), emailové schránky uživatelů a data webových serverů.

V adresáři `/etc` se nachází konfigurační soubory operačního systému a jeho služeb. Je zde také uložena konfigurace síťových služeb, jako například systému DNS a webových serverů.

8.2.2 Systém doménových jmen

Systém doménových jmen **DNS** (Domain Name System) je služba pro přidělení textových názvů síťovým rozhraním stanic v síti Internet, které používají IP adresy. Doménové jméno lze využít i pro specifikaci konkrétní služby na stanici, jako například webový server. Služba DNS pracuje v režimu klient-server, kdy klient žádá server o přeložení doménového jména na IP adresu.

Seznam doménových serverů pro dotazování je uložen v souboru `/etc/resolv.conf`. Typicky je zde hlavní DNS server a záložní DNS server.

```
1 nameserver 147.229.71.10
2 nameserver 147.229.71.13
```

Pro přeložení doménového jména na IP adresu slouží příkaz *nslookup*. Program umožňuje i **reverzní** překlady, tedy získat doménové jméno pro IP adresu.

Kromě dotazů na DNS servery lze zadat překlad doménového jména na IP adresu lokálně. K tomuto účelu slouží záznamy v konfiguračním souboru `/etc/hosts`. Zde může být například tento záznam.

```
1 147.229.141.111 pc118.utko.feec.vutbr.cz
```

U tohoto příkladu bude doménové jméno *pc118.utko.feec.vutbr.cz* lokálně přeloženo na IP adresu 147.229.141.111.

8.2.3 Webový server

Jedním z nejrozšířenějších webových serverů je Apache. Apache je dostupný pro různé operační systémy a poskytuje mnoho funkcí, jako například podpora **virtuálních serverů**. Funkce webového serveru Apache lze rozšiřovat pomocí modulů.

8.3 Pokyny pro vypracování

8.3.1 IP adresy a doménová jména

- Proveďte zjištění IP adresy pro různá doménová jména pomocí programu *nslookup* (zkuste např. seznam.cz a další).

```
[]$ nslookup seznam.cz
Server: 147.229.71.10
Address: 147.229.71.10#53
Non-authoritative answer:
Name: seznam.cz
Address: 77.75.77.53
```

> Přeložená IP adresa k doménovému jménu je uvedena až **za adresou doménového serveru** (2×), který se pro překlad použil. Je možné, že pro některá doménová jména bude vráceno více adres, tj. jsou dostupné pomocí více serverů.

- Proveďte překlad adresy i v opačném směru, tedy zadejte jako parametr IP adresu a zjistěte doménové jméno počítače. Začněte třeba vlastní IP adresou a pokračujte adresami, které jste zjistili v předchozím případě.

```
Server: 147.229.71.10
Address: 147.229.71.10#53
Non-authoritative answer:
3.76.75.77.in-addr.arpa name = www.seznam.cz.
```

> Může se stát, že jméno, které obdržíte po zpětném překladu IP adresy, se neshoduje s původním doménovým jménem. Je to proto, že počítač může mít k jedné IP adrese definováno více jmen. Např. počítače, které poskytují současně služby WWW a FTP, mají obvykle samostatné jméno pro každou z těchto služeb.

- Spusťte webový prohlížeč. Dále spusťte na pozadí příkaz *tshark* s takovým nastavením filtrů, aby zachytával pouze data služby DNS, které putují na **vaše síťové rozhraní** (tedy příchozí i odchozí). Výstup programu přesměrujte do souboru `dns_log.txt`.

```
[]$ sudo /usr/bin/tshark -i <sitove rozhrani> -Y dns -x > dns_log.txt
```

> Parametr `-Y` uvozuje filtr pro zachycení dat („dns“). Parametr `-x` udává formát dat pro zápis do souboru.

Poznámka: `<sitove rozhrani>` nahraďte názvem síťového rozhraní, který získáte například výpisem seznamu rozhraní pomocí příkazu `/sbin/ip addr`.

- Nyní zadejte do prohlížeče libovolnou webovou adresu. Po úspěšném dosažení požadované webové stránky prohlížeč zavřete, ukončete monitorování dat ([Ctrl] + [C]) a podívejte se do souboru `dns_log` na zachycenou komunikaci DNS. Pro přehledné zobrazení dat v souboru (mimo přímé prohlížení obsahu) lze využít i program Wireshark (File -> Import from Hex dump).

> V zachycené komunikaci by měl být dobře patrný dotaz na DNS server a také odpověď s hledanou IP adresou webového serveru.

- Ověřte, zda dotazovaný server pro překlad doménových jmen odpovídá záznamu v `/etc/resolv.conf`, kde jsou uloženy záznamy o používaných serverech.

8.3.2 Konfigurace webového serveru *Apache*

- Spusťte server příkazem

```
[]$ sudo systemctl start httpd
```

- Ověřte činnost serveru v prohlížeči zadáním adresy `http://localhost:80`. Následně se zobrazí testovací stránka.
- Základní nastavení serveru pro kořenový adresář je v `/var/www/html`. Abychom mohli pracovat se soubory vytvořenými uživatelem *root*, tak z výukových důvodů převezměte vlastnictví adresáře `/var/www/`.

```
[]$ sudo chown -R student /var/www/
```

> Parametr `-R` mění rekurzivně vlastníka i u podadresářů/souborů.

- Pro otestování funkce PHP vytvořte v adresáři `/var/www/html` vlastní dokument `phpinfo.php` a do něj vložte následující text.

```
<?php phpinfo(); ?>
```

- V prohlížeči zobrazte adresu `http://localhost/phpinfo.php`. Výpis informací vám potvrdí funkčnost PHP.
- Konfigurační parametry webového serveru jsou uloženy v `/etc/httpd/conf/httpd.conf`. Pro možnost editovat soubor `httpd.conf` jako uživatel *student* upravte jeho práva.

Nastavte jméno serveru, abyste se nemuseli stále odkazovat jménem *localhost*. K tomu slouží direktiva *ServerName*. Použijte direktivu, která je uvedena jako první v konfiguračním souboru – nepoužívejte direktivy pro virtuální servery, které jsou uvedeny až ke konci konfiguračního souboru. Pro použití direktivy je potřeba ji přepnout do stavu platný tím, že odstraníte znak komentáře (#) na začátku příslušného řádku. Zvolte si například jméno serveru `www.mujserver.cz`.

```
1 ServerName www.mujserver.cz
```

Poznámka: *ServerName* je uvedeno ke konci rozsáhlého souboru `httpd.conf`. Abyste řádek dlouho nehledali, lze použít v editoru *vi* vyhledávání textu pomocí konstrukce `:/<hledany_text>`. Na následující výskyt hledaného textu přejdete klávesou [n] (na předchozí pak klávesou [N]).

- Správnost provedených změn otestujte. **Nepokračujte dále**, pokud test konfigurace neproběhne v pořádku (v případě chyb se dále nespustí webový server).

```
1 []$ sudo apachectl configtest
```

- Změny provedené v konfiguračním souboru se projeví až po jeho opětovném načtení serverem. K tomu dochází vždy při startu serveru. **Proto server po každé změně restartujte.** Restartování proveďte obdobně jako start serveru, pouze s parametrem `restart`.
- Pouhá editace souboru `httpd.conf` ovšem k požadované funkčnosti nevede. Aby stránky byly dostupné zadáním vámi zvoleného jména do adresního řádku prohlížeče, musí ke jménu existovat i **odpovídající překlad doménového jména na IP adresu**. K tomuto účelu editujte soubor `/etc/hosts` a přidejte nový záznam zajišťující odpovídající překlad k lokální adrese.

> V souboru `hosts` jsou vypsána jména známých domén. Této lokální databáze se využívá nejdříve. Teprve až když záznam o hledaném počítači v databázi není, je o nalezení adresy požádán server DNS.

- Vytvořte soubor `index.html`, který představuje hlavní stránku, která se zobrazí při žádosti na server. Název dokumentu `index.html` je **povinný** a nelze použít jiný název. Důvodem je, že webový server jako první stránku zobrazuje obsah souboru `index.html`. Do souboru zapište nějaký text, který se zobrazí na stránce. Soubor vytvořte v adresáři `/var/www/html/`.

- Otestujte funkčnost nové adresy a vytvořené stránky.

8.3.3 Samostatný úkol

Webový server Apache podporuje provoz virtuálních serverů, tedy více serverů na jednom umístění. Pro každý virtuální server je potřeba vytvořit v konfiguračním souboru blok *<VirtualHost>*. V každém takovém bloku musí být specifikovány minimálně direktivy *ServerName* a *DocumentRoot* k nastavení jména a **kořenového adresáře** serveru, kde se nachází soubor `index.html`. Je také možné přistupovat k jednomu serveru přes více jmen díky direktivě *ServerAlias*, ovšem za předpokladu zajištění DNS překladu.

> Pokud příchozí požadavek neodpovídá, žádnému jménu nebo aliasu serveru, je použit první blok *<VirtualHost>* jako základní.

V rámci samostatného úkolu vytvořte dva virtuální servery. K tomu využijte následujících pokynů:

- Vytvořte v adresáři `/var/www/html/` dvě složky, kde budou uloženy soubory pro vaše virtuální servery.
- Na první virtuální webový server se odkazujte jako na `www.seznam.cz` a `seznam.cz`. Na druhý virtuální webový server se odkazujte jako `www.google.com` a `google.com`. První webový server bude zobrazovat vaše jméno a příjmení. Druhý webový server bude zobrazovat vaše rodné město.
- Editujte konfigurační soubor serveru Apache `/etc/httpd/conf/httpd.conf` tak, že na konec přidáte virtuální servery podle předlohy níže. Je potřeba konfiguraci zadat dvakrát pro každý virtuální server.

```
<VirtualHost *:80>
DocumentRoot ...
ServerName ...
ServerAlias ...
</VirtualHost>
```

Vytvořené webové stránky zobrazte v prohlížeči.

Několik poznámek, které vám mohou pomoci:

- Zobrazuje se originální stránka? Nezapomněli jste upravit lokální překlad doménových jmen?
- Jestliže vám bude prohlížeč stále zobrazovat předchozí obsah stránek, tak má tyto stránky uloženy ve své vyrovnávací paměti. Tu smažete pomocí [CTRL]+[F5] pro platný obsah stránky. Při hlášení nenalezené stránky je potřeba smazat celou vyrovnávací paměť prohlížeče: Preferences → Privacy & Security → Cookies and Site Data → Clear Data (zvolte vše).
- Správný syntax konfigurace si můžete ověřit pomocí příkazu `sudo apachectl configtest`.

- Pokud vám spuštění serveru bude vypisovat varovné hlášení k direktivě *DocumentRoot*, tak v této direktivě se uvádí **adresář**.

9 POČÍTAČOVÉ CVIČENÍ Č. 7 – START PO SÍTI

9.1 Účel cvičení

Práce s jednoduchými jádry v podobě zavaděčů. Demonstrovat použití kombinace znalostí pro vytvoření komplexního síťového systému. Zahrát si počítačovou hru.

9.2 Teoretický úvod

Toto cvičení se věnuje konfiguraci komplexního síťového systému, který umožní spustit po síti aplikaci (nebo celý operační systém) na stanici bez vlastního operačního systému. Pro realizaci cvičení je využita bezdisková stanice.

9.2.1 Servery DHCP a TFTP

Pro síťový start aplikací nebo operačního systému na bezdiskové stanici je na serverové stanici nutné zprovoznit server **DHCP** (Dynamic Host Configuration Protocol) a server **TFTP** (Trivial File Transfer Protocol). DHCP server sdělí klientské stanici síťovou konfiguraci. Mimo základní informace, jako je IP adresa a maska, je předána bezdiskové stanici také IP adresa serveru TFTP a název souboru, který má být z tohoto serveru stažen. Tento soubor obsahuje **zavaděč**, pomocí kterého lze spouštět aplikace na bezdiskové stanici. Servery DHCP a TFTP lze realizovat jedním programem, například *dnsmasq*.

9.2.2 Síťový zavaděč

Zavaděč obvykle stáhne na bezdiskovou stanici další soubory, případně zobrazí nabídku uživateli a následně spustí aplikaci nebo operační systém. Aplikace/operační systém pak běží v paměti RAM. Jako zavaděč bude použit program *pxelinux*, který je součástí projektu *syslinux*.

9.3 Pokyny pro vypracování

9.3.1 Příprava serveru

Protože by bylo nepraktické mít v učebně dalších 26 počítačů jako bezdiskové stanice, bude využit program *VirtualBox* pro virtualizaci další stanice. V tomto programu je připravena virtuální stanice bez disku. Tato bezdisková stanice je propojena s virtuálním serverem pomocí vnitřní sítě *intnet*. Virtualizovaný server je do této sítě připojen svým druhým rozhraním *eth1*, bezdisková stanice je připojena svým jediným rozhraním *eth0*.

- Nastavte statickou konfiguraci síťového rozhraní *eth1*. Pro tuto akci je potřeba editovat soubor `/etc/sysconfig/network-scripts/ifcfg-eth1`. Tento soubor editujte editorem *vi*. První dvě hodnoty přepište v současné konfiguraci, druhé dva řádky připište.

```
1 BOOTPROTO=static
```

```
ONBOOT=yes
```

```
IPADDR=192.168.57.1
NETMASK=255.255.255.0
```

- Restartujte síťové rozhraní *eth1* pro provedení změn.

```
[]$ sudo systemctl restart NetworkManager
```

- Pomocí příkazu `/sbin/ip addr` ověřte, že rozhraní *eth1* má přidělenou IP adresu 192.168.57.1 a masku 255.255.255.0. Nepokračujte dále, pokud rozhraní adresu přidělenou nemá!

Poznámka: pokud bude úprava konfigurace *eth1* správná a stále rozhraní nebude mít přidělenou adresu, zkontrolujte zda v programu *Virtualbox* je nastavena správně MAC adresa rozhraní *eth1* (Network → Adapter 2 → Advanced) na 0800277DBD12.

- Na serveru spusťte zachytávání komunikace pomocí programu *tcpdump*.

```
[]$ sudo /usr/sbin/tcpdump -i eth1
```

- Následně spusťte bezdiskovou stanici. Měli byste zachytit DHCP dotaz klientské stanice na rozhraní serveru *eth1*.

```
IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from
    08:00:27:dc:fa:9b (oui Unknown), length: 548
```

Poznámka: při kliknutí na obrazovku bezdiskové stanice zmizí kurzor myši, jelikož se nacházíte v textovém rozhraní. Pro jeho obnovení stiskněte pravé [CTRL].

Poznámka: pokud se nebudou zobrazovat zachycené žádosti o síťovou konfiguraci z bezdiskové stanice, tak ověřte následující: i) rozhraní *eth1* je **aktivováno**, ii) v programu *VirtualBox* je bezdisková stanice propojena vnitřní sítí *intnet* k serveru (Network → Adapter 2 → Attached to Internal Network, name Intnet).

9.3.2 Instalace projektu *syslinux* a soubory serveru TFTP

- V domovském adresáři vytvoříte adresář `work`. Následně se do něj přesuňte.
- Nejprve vytvořte základ pro TFTP server, ze kterého si bezdisková stanice bude stahovat potřebné soubory. Za tímto účelem uložte do adresáře `work` archív `syslinux-3.83.zip`.
- Následujícím postupem z archívu získáte potřebné binární soubory. Archiv rozbalíte do adresáře `temp`. Následně provedete instalaci binárních souborů do adresáře `~/work/syslinux`. Postup vychází z aktuálního adresáře `work`.

```
[]$ mkdir temp
[]$ cd temp
```

```
[]$ unzip ../syslinux-3.83.zip
[]$ make INSTALLROOT=~/work/syslinux local-install
```

Poslední příkaz spouští program *make* a makro *local-install*, které je definované v souboru `Makefile` uvnitř archívu. Toto makro slouží k linkování již předkompilovaných objektů a instalaci hotových binárních souborů do systému. Je to obdoba příkazu `make install`, který pouze instaluje binární soubory. Proměnnou INSTALLROOT je určeno, kam se má instalace provést.

Nyní je v pracovním adresáři `work` stažený archív, adresář temp a adresář `syslinux` , do kterého se nakopírovaly všechny součásti zavaděče *syslinux* bez zdrojových kódů. Binární soubory jsou v adresáři `work/syslinux/usr/share/syslinux`.

- V pracovním adresáři `work` vytvořte adresář `tftp`, který později bude kořenový adresář pro server TFTP.
- Z adresáře `work/syslinux/usr/share/syslinux` nakopírujte do adresáře `work/tftp` tyto soubory: `pxelinux.0`, `menu.c32`, `vesamenu.c32`, `mboot.c32`, `poweroff.com`, `reboot.c32`. Tyto soubory obsahují různá **velmi jednoduchá jádra**, které budeme dále označovat pouze jako programy či zavaděče. Pro kopírování více souborů najednou můžete použít konstrukci `cp <soubor1> <soubor2> <souborN> <cilovy adresar/>`.

`pxelinux.0` je hlavní zavaděč, `menu.c32` a `vesamenu.c32` jsou pomocné programy, které načtou konfigurační soubor (bude vytvořen dále) a zobrazí nabídku, `mboot.c32` je zavaděč pracující se standardem MultiBoot, `poweroff.c32` a `reboot.c32` jsou programy, které vypnou, respektive restartují počítač.

Nyní je v adresáři pro server TFTP nahraný zavaděč *pxelinux*.

- V adresáři `~/work/tftp/` vytvořte adresář `pxelinux.cfg` (název adresáře obsahuje připonu `.cfg`). V tomto adresáři vytvořte soubor `default` a vložte do něj text `prompt 1`.

V souboru `default` definujete konfiguraci zavaděče. Tímto jste zavaděč *pxelinux* nastavili tak, aby ihned po spuštění zobrazil příkazovou řádku a čekal na reakci uživatele.

9.3.3 Konfigurace a spuštění serveru DHCP a TFTP

- V pracovním adresáři `work` vytvořte konfigurační soubor `diskless.conf` s následujícím obsahem.

```
interface=eth1
bind-interfaces
dhcp-leasefile=/tmp/dnsmasq.leases
dhcp-range=192.168.57.50,192.168.57.250,600
dhcp-boot=pxelinux.0
enable-tftp
```

```
7 tftp-root=/home/student/work/tftp
```

První dva řádky určují, na kterém rozhraní server bude odpovídat, volba `bind-interfaces` vyžaduje, aby rozhraní *eth1* opravdu existovalo. `dhcp-leasefile` určuje soubor, ve kterém jsou vedeny záznamy o přidělených adresách, následuje rozsah adres, které server bude přidělovat a dobu, po které je stanice povinna požádat o prodloužení v sekundách. Konečně `dhcp-boot` je jméno zavaděče, který si má stanice stáhnout ze serveru TFTP. Příkaz `enable-tftp` určuje, že program *dnsmasq* bude poskytovat službu TFTP. Poslední je kořenový adresář pro server TFTP. Síťové parametry, které nebyly uvedeny, si program *dnsmasq* zjistí automaticky – masku sítě zjistí z nastavení síťového rozhraní eth1. Výchozí brána, server DNS a TFTP odpovídá IP adrese na rozhraní *eth1*.

- Ve výchozím nastavení firewallu není povolena komunikace s kombinovaným serverem DHCP/TFTP. Aby server, který vzápětí spustíte, mohl dostávat zprávy z bezdiskové stanice, povolte ve firewallu příslušné porty. Nakonec firewall restartujte pro aplikování změn.

```
1 []$ sudo firewall-cmd --zone=public --permanent --add-port=67/udp
2 []$ sudo firewall-cmd --zone=public --permanent --add-port=69/udp
3 []$ sudo firewall-cmd --reload
```

Server DHCP naslouchá na portu č. 67. Server TFTP naslouchá na portu č. 69.

- **Otevřete nový terminál**. Přesuňte se do adresáře, ve kterém jste vytvořili konfigurační soubor `diskless.conf`. Spusťte server *dnsmasq*, kterému jako parametr předáte vámi vytvořenou konfiguraci.

```
1 []$ sudo /usr/sbin/dnsmasq -C diskless.conf -d
```

Poznámka: pokud zapomenete uvést parametr `-d`, tak se server spustí na pozadí. Další pokus o spuštění serveru selže, protože dvě instance serveru využívající stejný port nemůžou být spuštěny. Je tedy potřeba zabít předchozí instanci spuštěnou na pozadí příkazem `sudo kill` s parametrem čísla procesu (PID), který realizuje server. PID procesu získáte například pomocí konstrukce `ps ax | grep dnsmasq`.

Parametr `-C` mění výchozí konfigurační soubor z `/etc/dnsmasq.conf` na zadaný. Parametr `-d` zabrání programu spuštění na **pozadí**. Takto činnost serverů TFTP a DHCP z výukového důvodu uvidíte přímo v terminálu. Při ostrém provozu se tento parametr samozřejmě nepoužívá.

- Pokud se neobjevila žádná chybová hláška, můžete spustit virtuální počítač bez disku se zapnutým spouštěním po síti. Po úspěšném spuštění počítač čeká na reakci uživatele s hláškou `boot:`. Níže je uveden výpis, který byste měli obdržet na bezdiskové stanici.

```
CLIENT MAC ADDR: 08 00 27 DC FA 9B GUID: 5C67807B-...
CLIENT IP: 192.168.57.182 MASK: 255.255.255.0 DHCP IP: 192.168.57.1
GATEWAY IP: 192.168.57.1

PXELINUX 3.83 2009-10-05 Copyright (C) 1994-2009 H. Peter Anvin et al
...
My IP address seems to be C0A839B6 192.168.57.182
ip=192.168.57.182:192.168.57.1:192.168.57.1:255.255.255.0
TFTP prefix:
Trying to load: pxelinux.cfg/5c67807b-b0fb-47eb-8119-1acc3914d2a2
Trying to load: pxelinux.cfg/01-08-00-27-dc-fa-9b
Trying to load: pxelinux.cfg/C0A839B6
Trying to load: pxelinux.cfg/C0A839B
...
#hledani konfigurace
...
Trying to load: pxelinux.cfg/default
boot:
```

Nejprve bezdisková stanice získá síťovou konfiguraci z DHCP serveru (IP adresa, maska a výchozí brána). Dále se bezdiskové stanici předá adresa serveru TFTP a soubor, který si má stáhnout. Následně se stáhne soubor `pxelinux.0` a spustí se. Zavaděč se pak snaží najít vhodnou konfiguraci a kontaktuje stejný TFTP server. Konfiguraci hledá v adresáři `pxelinux.cfg`. Po různých kombinacích pro hledání konfiguračního souboru (nejdříve podle GUID – Globally Unique Identifier, následně podle MAC adresy a pak podle postupného zkracování IP adresy v hexadecimálním formátu) je použita výchozí konfigurace v souboru `default`. V tomto souboru je zatím uloženo, aby se zobrazila příkazová řádka (`boot:`).

Poznámka: **pokud bude DHCP server psát chybovou hlášku, že není nastaven rozsah pro IP adresy**, chyba je pravděpodobně v statické konfiguraci síťového rozhraní *eth1*. Například nebyl proveden restart pro aplikaci statické adresy nebo je chyba v souboru `/etc/sysconfig/network-scripts/ifcfg-eth1`. Ověřte výpisem síťové konfigurace, že rozhraní *eth1* má přidělenu zadanou statickou IP adresu.

DHCP server po úspěšném spuštění bezdiskové stanice by měl vypsat:

```
dnsmasq: DHCPDISCOVER(eth1) 08:00:27:dc:fa:9b
dnsmasq: DHCPOFFER(eth1) 192.168.57.182 08:00:27:dc:fa:9b
dnsmasq: DHCPREQUEST(eth1) 192.168.57.182 08:00:27:dc:fa:9b
dnsmasq: DHCPACK(eth1) 192.168.57.182 08:00:27:dc:fa:9b
dnsmasq: TFTP sent /home/student/work/tftp/pxelinux.0 to 192.168.57.182
dnsmasq: TFTP sent /home/student/work/tftp/pxelinux.cfg/default to
    192.168.57.182
```

Bezdisková stanice hledá DHCP server – DISCOVER, server odpovídá a nabízí volnou adresu – OFFER, bezdisková stanice o ni žádá – REQUEST, server potvrzuje – ACK. Dále je ve výpisu vidět soubor `pxelinux.0`, který je předán bezdiskové stanici pro

spuštění. Na posledním řádku je zobrazeno předání konfigurace v podobě souboru `default`.

- Ukončete spuštěný DHCP server.

9.3.4 Konfigurace zavaděče *syslinux*

- Zprovozněte grafické menu zavaděče tím, že do souboru `~/work/tftp/pxelinux.cfg/default` vložíte následující obsah. Vyberte si nabídku, která se vám víc líbí. Komentář následně *odstraňte*. Dále vytvořte položky menu pro restart a vypnutí počítače. Ukázka je uvedena níže.

```
ui menu.c32 # nebo ui vesamenu.c32

menu title BSOS startovaci server

menu separator
label
menu label Konec:
menu disable

label poweroff
menu label Vypnout PC
menu indent 1
kernel poweroff.com

label reboot
menu label Restartovat PC
menu indent 1
kernel reboot.c32
```

Programy *menu.c32* nebo *vesamenu.c32* spustí další program *(vesa)menu*, který projde konfiguraci a zobrazí nabízené položky.

Strukturu zobrazovaného menu definujete následovně:

- Popisek v záhlaví celého menu zavedete příkazem *menu title <text>*.
- Položku pojmenujete pomocí *menu label <text>*.
- Prázdný řádek přidáte pomocí *menu separator*.
- Neaktivní řádek, který lze použít jako popisek, vyrobí příkaz *menu disable*.
- Odsazení položky od okraje se realizuje příkazem *menu indent <cislo>*.

Spustitelnou položku v nabídce přidají následující texty:

- *label* (popisek),
- *kernel* (jádro),

– *append* (další parametry pro start jádra (nepovinné))

Vytvořené menu vyzkoušejte včetně vytvořených položek (vypnutí a restart bezdiskové stanice).

- Znovu spusťte kombinovaný DHCP/TFTP server.
- Restartujte bezdiskovou stanici.

9.3.5 Samostatný úkol

Na bezdiskové stanici si zahrajte hru *Invaders*. Hru spusťte pomocí položky ve startovacím menu.

- Stáhněte si soubor `grub-invaders_1.0.0-10_i386.deb`.
- Rozbalte archiv `grub-invaders_1.0.0-10_i386.deb`.

```
[]$ ar xv grub-invaders_1.0.0-10_i386.deb
```

- Rozbalte soubor `data.tar.gz`.

```
[]$ tar -zxvf data.tar.gz
```

- Bezdiskové stanici je třeba předat soubor `invaders.exec`. Tento soubor je po rozbalení obou archívů v podadresáři `boot`. Soubor přemístěte na správné místo tak, aby byl k dispozici pro přenos ze serveru TFTP na bezdiskovou stanici.
- Dále je potřeba zavést jádro projektu SysLinux, které podporuje práci s formátem spustitelného souboru se hrou. Tímto formátem je *MultiBoot* a je podporován **jádrem/zavaděčem** *mboot.c32*. Samotný soubor ke spuštění pak připojíte pomocí parametru *append*. Nápověda pro konfiguraci položky menu je uvedena níže.

```
kernel mboot.c32
append invaders.exec
```

Jako výstup samostatného úkolu ukažte vyučujícímu, že hrajete hru.

10 POČÍTAČOVÉ CVIČENÍ Č. 8 – SÍŤOVÝ SUBSYSTÉM

10.1 Účel cvičení

Podpořit probíranou látku síťového subsystému. Seznámit se s konfigurací firewallu a proxy serveru. Demonstrovat možnost podvrhu webové stránky a modifikace jejich obsahu při přístupu na nezabezpečený sever.

10.2 Teoretický úvod

10.2.1 Datové jednotky

Na transportní vrstvě protokolového modelu TCP/IP jsou přenášeny segmenty TCP a datagramy UDP. Tyto přenášené jednotky obsahují data z vyšší vrstvy, kterou je vrstva aplikační. V záhlaví je uveden zdrojový a cílový **port**. Jedná se o adresy procesů v operačním systému, které mezi sebou komunikují.

Na síťové vrstvě jsou přenášeny pakety. Tělo paketu obsahuje data předchozí vyšší vrstvy, tedy transportní. V záhlaví je uvedena zdrojová a cílová IP adresa, která určuje komunikující stanice.

10.2.2 Firewall

Firewall pracuje jako **filtr**, který na základě informací v záhlaví datových jednotek (pokud se jedná o aplikační firewall, tak i v jejich obsahu) rozhoduje o tom, zda mají být data propuštěna, přesměrována, pozměněna nebo zahozena. Firewall postupuje sériově podle definovaného řetězce **pravidel**. Při první shodě s pravidlem je aplikována akce, která je uvedená u tohoto pravidla. Záleží tedy na pořadí definovaných pravidel. Pokud není nalezena shoda se žádným pravidlem, je aplikováno **výchozí pravidlo**.

10.2.3 Program *iptables*

Program *iptables* implementuje **stavový** firewall. Program obsahuje tři typy pracovních režimů. Tyto režimy se liší podle počtu řetězců pravidel, se kterými lze pracovat.

1. Filtrační režim

Jedná se o základní režim, který obsahuje řetězce INPUT, OUTPUT a FORWARD.

- Řetězec INPUT – pravidla se aplikují, pokud paket má být doručen místnímu systému.
- Řetězec OUTPUT – pravidla se aplikují, pokud paket odchází z místního systému.
- Řetězec FORWARD – systém pracuje jako směrovač. Pravidla jsou aplikována při přeposílání mezi rozhraními směrovače. Neaplikují se pravidla pro místní systém, tj. OUTPUT/INPUT.

2. Režim úpravy zdroje a cíle

Tento režim je označován jako NAT (Network Address Translation) a obsahuje řetězce PREROUTING, POSTROUTING a OUTPUT.

- Řetězec OUTPUT – pravidla se aplikují, pokud pakety odchází z místního systému, lze aplikovat úpravu **cílové adresy** – DNAT (Destination NAT).
- Řetězec PREROUTING – systém pracuje jako směrovač. Pravidla jsou aplikována na pakety před směrováním, tj. nalezením záznamu ve směrovací tabulce. Opět lze použít pro úpravu cílové adresy.
- Řetězec POSTROUTING – systém pracuje jako směrovač. Pravidla jsou aplikována na pakety po směrování, tj. po nalezení záznamu ve směrovací tabulce. Pravidla se používají pro úpravu **zdrojové adresy** – SNAT (Source NAT).

3. Režim komplexní úpravy

Tento režim je označován jako MANGLE a obsahuje řetězce PREROUTING, POSTROUTING, OUTPUT, INPUT a FORWARD. Používá se pro komplexní úpravu dat. V tomto režimu jsou zpřístupněny i nestandardní pravidla, například pro změnu hodnoty TTL (Time to Live).

10.2.4 Proxy server

Proxy server je firewall pracující na aplikační vrstvě, který umožňuje **monitorování** a případnou **změnu** přenášeného obsahu. Proxy server pracuje s daným protokolem na aplikační úrovni, jelikož musí „znát“ formát tohoto protokolu. Proxy servery často pracují s webovým obsahem (protokolem HTTP), mohou podporovat i více protokolů. Webové proxy servery mají různé použití. Jedná se například o ukládání předešle navštívených stránek do **vyrovnávací paměti**. Při opakovaném požadavku na tyto stránky dochází k jejich rychlejšímu poskytnutí z vyrovnávací paměti bez zátěže originálního serveru a prostředků v Internetu. Dalšími příklady jsou blokování závadného obsahu stránek nebo poskytnutí anonymity.

Transparentní proxy nevyžaduje žádnou konfiguraci u klienta a klient si tudíž přítomnosti proxy serveru nemusí být vědom. U **netransparentní** webové proxy je vyžadována konfigurace u klienta v konkrétním webovém prohlížeči. Lze tak určit, který provoz a pomocí kterého prohlížeče jde přes proxy. U transparentní webové proxy jde veškerá webová komunikace přes proxy server.

10.2.5 Proxy server *squid*

Jedna z rozšířených implementací proxy serveru je program *squid*. Tento proxy server pracuje s protokolem HTTP, ale podporuje i jiné, například FTP. Program *squid* umožňuje ukládání webového obsahu do vyrovnávací paměti pro obsluhu opakovaných požadavků. Umožňuje také filtrování provozu a změnu jeho obsahu. Proxy server *squid* může pracovat v transparentním i netransparentním režimu. Lze ho také provozovat na velké škále operačních systémů.

Kompletní přehled pokročilejších síťových administrátorských úkonů, včetně nastavení firewallu, je uveden v publikaci [9].

10.3 Pokyny pro vypracování

10.3.1 Vytváření řetězců a pravidel

- Proveďte nastavení firewallu *iptables* do výchozí stavu.

```
[]$ sudo /sbin/iptables -F
[]$ sudo /sbin/iptables -X
```

- Následně si vypište tabulku pravidel firewallu.

```
[]$ sudo /sbin/iptables -L
```

Firewall by neměl obsahovat žádná pravidla v uvedených řetězcích INPUT, FORWARD a OUTPUT.

- Vytvořte nová **výchozí pravidla**, jimiž zahodíte všechny odchozí a přesměrované pakety, které nevyhoví žádným dalším pravidlům. Poté se podívejte na výpis pravidel, co vše se změnilo.

```
[]$ sudo /sbin/iptables -P INPUT ACCEPT
[]$ sudo /sbin/iptables -P OUTPUT DROP
[]$ sudo /sbin/iptables -P FORWARD DROP
```

Takto vytvořená výchozí pravidla určují, jak se má naložit s pakety, které nevyhoví některému z předešlých pravidel, které si dále nadefinujete pro jednotlivé řetězce. Příkazy definují změnu nakládání s pakety v řetězcích vstup (INPUT), výstup (OUTPUT) a přesměrování (FORWARD). Parametr `-P` (z anglického Policy) označuje změnu výchozích pravidel pro tyto tři základní řetězce. Parametr ACCEPT znamená přijetí paketů, parametr DROP značí zahození paketů. Je to jeden z přístupů k zabezpečení – „zahodit vše, co nevyhoví“.

Pokud bychom nechali jen tyto pokyny, žádná komunikace by nefungovala, neboť všechny pakety by byly zahozeny. Běžným postupem je omezovat komunikaci na vstupu. Zde však z výukových důvodů (viz dále) budeme převážně pracovat s výstupem.

- Spusťte webový prohlížeč a pokuste se načíst libovolnou stránku. Všechny odchozí pakety jsou zahazovány a komunikace nefunguje.
- Vytvořte nový řetězec pro filtrování paketů, který se bude jmenovat `tcp_pakety` a bude propouštět odchozí TCP pakety na porty HTTP (port 80) a HTTPS (port 443). Protokol HTTPS je zabezpečená varianta protokolu HTTP.

```
[]$ sudo /sbin/iptables -N tcp_pakety
[]$ sudo /sbin/iptables -A OUTPUT -p TCP -j tcp_pakety
[]$ sudo /sbin/iptables -A tcp_pakety -p TCP --dport 443 -j ACCEPT
[]$ sudo /sbin/iptables -I tcp_pakety -p TCP --dport 80 -j ACCEPT
```

První řádek z předchozí sekvence příkazů definuje nový řetězec pravidel s názvem `tcp_pakety` (`-N` označuje nový řetězec a následuje jméno). Druhý řádek znamená, že nově definovaný řetězec bude patřit k výstupnímu řetězci (OUTPUT). Parametr `-p` (protocol) specifikuje komunikační protokol.

Na třetím řádku parametr `-A` (add) značí **přidání pravidla** do řetězce (stejně jako přidání řetězce v druhém řádku). Na čtvrtém řádku parametr `-I <cislo pozice>` (insert) značí **vložení pravidla na zadanou pozici** v řetězci (nebo před zadanou pozici, pokud již v řetězci existuje). Pokud pozice není zadána, tak je pravidlo vloženo **na začátek řetězce**.

Pravidla v řetězcích postihnou segmenty TCP s cílovými porty 80 (protokol HTTP) a 443 (protokol HTTPS), tedy službu webových stránek. To je zajištěno parametrem `--dport`. Za parametrem `-j` je uveden pokyn pro naložení se segmentem. V našem případě jej vpustíme (ACCEPT).

- Podívejte se, jak se předchozí pravidla projevila na výpisu nastavení firewallu. **Sledujte pořadí pravidel v řetězci `tcp_pakety`.**

```
Chain tcp_pakety (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:https
```

- Spusťte webový prohlížeč a zadejte libovolnou stránku. Stránka by se vám neměla načíst, protože není povolen protokol DNS pro překlad jmen.
- Povolte odchozí datagramy protokolu UDP pro službu DNS (port 53).

```
[]$ sudo /sbin/iptables -N udp_pakety
[]$ sudo /sbin/iptables -A OUTPUT -p UDP -j udp_pakety
[]$ sudo /sbin/iptables -A udp_pakety -p UDP --dport 53 -j ACCEPT
```

- Vypište si opět seznam pravidel, která jste nadefinovali. Ve webovém prohlížeči zadejte jméno livolného serveru. Komunikace by měla fungovat.

10.3.2 Práce firewallu v režimu DNAT

- Smažte výchozí nastavení firewallu.

```
[]$ sudo /sbin/iptables -F
[]$ sudo /sbin/iptables -X
[]$ sudo /sbin/iptables -P OUTPUT ACCEPT
```

```
[]$ sudo /sbin/iptables -P FORWARD ACCEPT
[]$ sudo /sbin/iptables -L
```

- Zobrazte si nastavení firewallu v režimu NAT.

```
[]$ sudo /sbin/iptables -t nat -L
```

Povšimněte si změny zobrazených řetězců: PREROUTING, POSTROUTING a OUTPUT.

Režim NAT umožňuje změnu cílové a zdrojové adresy (DNAT, SNAT). Jako demonstraci této funkce využijeme možnost **změny cílové adresy**. Provedete nastavení firewallu tak, aby veškerý provoz na webové stránky byl přesměrován na jednu webovou stránku. Provoz přesměrujeme na spřátelené ČVUT, www.cvut.cz. Veškerý webový provoz můžete také přesměrovat na vaši zvolenou stránku, nebo váš vlastní webový server.

- Do firewallu přidejte tato pravidla. IP adresu nahraďte zjištěnou adresou webového serveru www.cvut.cz nebo vámi zvoleného serveru.

```
[]$ sudo /sbin/iptables -t nat -A OUTPUT -p tcp --dport 80 -j DNAT
   --to-destination <IP adresa>
[]$ sudo /sbin/iptables -t nat -A OUTPUT -p tcp --dport 443 -j DNAT
   --to-destination <IP adresa>
```

> Parametrem `-t nat` specifikujeme režim firewallu. Další nastavení DNAT je dostupné pouze v tomto režimu. Proč je použit řetězec OUTPUT? Důvod je jednoduchý, pokud si zopakujete teoretický úvod na začátku úlohy. V našem případě pakety odcházejí z místní stanice. Stanice nepracuje jako směrovač, tj. pakety neprochází mezi rozhraními. Další položky definují protokol (tcp) a cílový port, pro který budou pravidla aplikována. Příkaz je uveden dvakrát, abychom pokryli možné varianty přístupu na web – nezabezpečený (port 80, protokol HTTP) a zabezpečený (port 443, protokol HTTPS). Poslední položka určuje, jak bude upravena cílová adresa. Dojde pouze ke změně cílové IP adresy.

- Otevřete webový prohlížeč a napište libovolnou adresu. Pokud například zvolíte www.vutbr.cz (výchozí stránka), zobrazí se vám stránka s varovným hlášením. V sekci rozšířeném popisu se dočtete, že stránka www.vutbr.cz používá chybný certifikát, který patří www.cvut.cz. To je způsobeno tím, že původně jste chtěli přistoupit na www.vutbr.cz, ale firewall vás přesměroval na www.cvut.cz. Udělte souhlas s pokračováním a zobrazí se vám stránka ČVUT. Vyzkoušejte libovolné jiné stránky, vždy přejdete na ČVUT.

10.3.3 Transparentní proxy server

- Opět smažte pravidla firewallu, tentokráte v režimu NAT.

```
[]$ sudo /sbin/iptables -t nat -F
[]$ sudo /sbin/iptables -t nat -L
```

- Stáhněte soubor s programem v Perlu *obracene.pl*. Tento soubor nyní nepotřebujete, ale využijete ho později. Soubor si stahujete nyní, protože později vám bude záměrně zablokován přístup na zabezpečené stránky (včetně e-larningu). Důvod je ten, že bude provádět útok typu „man-in-the-middle attack“.
- Spusťte webový server.

```
[]$ sudo systemctl start httpd
```

- Jako proxy server bude využít program *squid*, **který také nastartujte**.
- Proveďte ověření, že proxy server je spuštěný tak, že přistoupíte na webovou stránku 127.0.0.1:3128. Port 3128 je výchozí port, na kterém běží proxy server *squid*. Měli byste vidět logo proxy serveru (krakatice) a chybovou hlášku, že stránku nelze zobrazit. To je v pořádku, protože zatím jste neprovedli konfiguraci proxy serveru.

```
ERROR
The requested URL could not be retrieved
The following error was encountered while trying to retrieve the URL: /
Invalid URL
Some aspect of the requested URL is incorrect.
...
```

- Do firewallu přidejte pravidlo, které všechen odchozí nezabezpečený provoz (HTTP) na port 80 přesměruje na proxy server. Další pravidlo přesměruje na proxy server i zabezpečený provoz (HTTPS, port 443).

```
[]$ sudo  /sbin/iptables -t nat -A OUTPUT -p tcp --dport 80 -j DNAT
   --to-destination 127.0.0.1:3128
[]$ sudo  /sbin/iptables -t nat -A OUTPUT -p tcp --dport 443 -j DNAT
   --to-destination 127.0.0.1:3128
```

- Nyní ve webovém prohlížeči přistupte na libovolnou **nezabezpečenou** stránku (v záhlaví adresy je **http://**) a zkontrolujte, že se vždy objeví stránka **s logem proxy serveru** – krakatice (dotazovaná stránka by se stále neměla zobrazit). To značí, že veškerý webový provoz je přesměrován na váš proxy server.
- Zobrazte si libovolnou **zabezpečenou** stránku (v záhlaví adresy je https://). Zobrazí se vám chybová hláška, že zabezpečené spojení nemohlo být sestaveno. To je správně, protože pomocí přesměrování komunikace přes transparentní proxy jste právě provedli z pohledu bezpečnosti komunikace útok typu „man-in-the-middle attack“ (zatím nedokončený, útok dokončíte později).

```
Secure Connection Failed
An error occurred during a connection to www.vutbr.cz.
...
```

Výše uvedené pravidlo přidané do firewallu se může zdát správné, nicméně je v něm skrytá vada. Proxy server, ač správně nakonfigurovaný, nebude pracovat. Důvod je v tom, že **odchozí provoz z proxy serveru** bude také směrován na proxy server a vytvoří se tak **nekonečná smyčka**. Proto je potřeba odchozí provoz z proxy serveru z tohoto přesměrování vynechat.

- Povolte provoz z proxy serveru tak, aby se vyhnul přesměrování. Vytvořte pravidlo firewallu, které povolí provoz procesů spuštěných s GID skupiny *squid*.

```
[]$ sudo /sbin/iptables -t nat -A OUTPUT -p tcp --dport 80 -m owner
   --gid-owner squid -j ACCEPT
[]$ sudo /sbin/iptables -t nat -A OUTPUT -p tcp --dport 443 -m owner
   --gid-owner squid -j ACCEPT
```

- **V tomto postupu je chyba (poslední dva pokyny pro přidání čtyř pravidel do firewallu), najděte ji a opravte ji.** Nápověda: chybu zjistíte poté, co si zobrazíte pravidla ve firewallu. Pomůže vám také se vrátit k teoretickému úvodu, konkrétně princip práce firewallu. Také vám napoví úvodní postup popisující základní vytváření pravidel.

- Po opravě chyby proveďte konfiguraci proxy serveru. Konfigurace je uložena v souboru `/etc/squid/squid.conf`, který otevřete jako *root*. Zde převeďte proxy do transparentního režimu přidáním parametru `intercept` na řádku `http_port 3128`. Dále přidejte ještě řádek `http_port 8080`, na kterém bude spuštěn proxy server v běžném, tj. netransparentním režimu.

```
http_port 3128 intercept
http_port 8080
```

Poznámka: Pokud je konfigurace prázdná, tak jste soubor neotevřeli jako *root*. Pokusem otevřít soubor jako běžný uživatel, ke kterému nemá přístup, byl otevřen nový prázdný soubor. Konfiguraci otevřete jako *root*.

- Proxy server restartujte.

```
[]$ sudo systemctl restart squid
```

- Ověřte, že vám proxy server pracuje. Přistupte na libovolnou **nezabezpečenou** stránku (protokol HTTP) a ta by se měla za pomocí proxy serveru zobrazit (pokud ne, neopravili jste chybu a čtěte dále). Pokuste se přistoupit také na libovolný **zabezpečený** web (protokol HTTPS). Zde by se měla zobrazit chybová hláška, tak jako v předchozím případě. Je to opět správně, protože pomocí transparentní proxy jste vložili do komunikace další prvek – „man-in-the-middle“.

- Zobrazte si soubor se záznamem běhu proxy serveru (log soubor) `/var/log/squid/access.log`, kde jsou zaznamenány stránky, které prochází proxy serverem. **Pokud zde nemáte záznamy o stránkách, které jste zadali ve webovém prohlížeči, tak komunikace neprobíhá přes proxy server. Případnou chybu opravte, viz přesměrování na proxy server.**

Jestliže vám webový prohlížeč zobrazuje hlášku

```
ERROR The requested URL could not be retrieved
The following error was encountered while trying to retrieve the URL:
    XXX
Access Denied.
```

tak komunikace probíhá přes proxy server. **Neopravili jste však předešlou chybu v postupu při konfiguraci firewallu.** Pro další nápovědu si zobrazte záznam logu běhu proxy serveru v souboru `/var/log/squid/cache.log`. Hledejte varovná hlášení o smyčce.

```
WARNING: Forwarding loop detected for:
GET /Artwork/SN.png HTTP/1.1
Host: www.squid-cache.org
```

Správná konfigurace (po opravě chyby) firewallu je níže. Sledujte **pořadí** zadaných pravidel.

```
Chain OUTPUT (policy ACCEPT)
target     prot opt source      destination
ACCEPT     tcp  --  anywhere    anywhere      tcp dpt:http OWNER GID
    match squid
ACCEPT     tcp  --  anywhere    anywhere      tcp dpt:https OWNER GID
    match squid
DNAT       tcp  --  anywhere    anywhere      tcp dpt:http
    to:127.0.0.1:3128
DNAT       tcp  --  anywhere    anywhere      tcp dpt:https
    to:127.0.0.1:3128
```

10.3.4 Samostatný úkol

Upravte proxy server tak, aby **zasahoval** do nezabezpečených stránek tím, že bude **modifikovat jejich obsah** – provedete útok „man-in-the-middle“. Jelikož jste nakonfigurovali transparentní proxy server, uživatel si nemusí být vědom toho, že proxy server zobrazované stránky upravuje.

Úpravu proveďte tak, že obrázky na webových stránkách budou překlopeny horizontálně. Toho si jistě každý uživatel povšimne. Námětem modifikace obrázků na webových stránkách jsou různé aprílové žertíky, jako například zde www.ex-parrot.com/pete/upside-down-ternet.html. K tomu účelu použijte již stažený program v Perlu `obracene.pl`. Soubor je také dostupný z gist.github.com/j9s/5075302, pro účely cvičení však byl upraven.

Skript `obracene.pl` na svém vstupu přijímá URL adresy a v případě, že se jedná o obrázek, provede otočení pomocí nástroje *ImageMagick*. Upravený obrázek následně uloží do adresáře webového serveru ve vyrovnávací paměti.

Soubor `obracene.pl` uložte do adresáře `/etc/squid/`. Souboru nastavte práva, aby jej **mohl číst a spouštět každý uživatel**. Dále upravte konfiguraci proxy serveru tak, aby byl tento soubor spouštěn při přístupu na webové stránky. Toho dosáhnete tím, že na konec konfiguračního souboru proxy serveru `/etc/squid/squid.conf` přidáte tento řádek.

```
1 url_rewrite_program /etc/squid/obracene.pl
```

Proxy ve výchozím stavu povoluje práci pěti procesům pro úpravu webu. Proto se některé obrázky **nemusí rotovat**. Abyste zvýšili počet rotovaných obrázků, zvyšte možný počet procesů realizujících program proxy serveru. Pro výukové účely můžete zadat i velké číslo. Následující řádek opět připište na konec souboru `/etc/squid/squid.conf` .

```
1 redirect_children 500
```

Po úpravách konfigurace nezapomeňte proxy server **restartovat**.

Nakonec vytvořte složku `/var/www/html/proxy/`, kde se budou rotované obrázky ukládat. Tomu adresáři nastavte **plná práva pro všechny uživatele**. Proxy server restartujte.

Přistupte na libovolný **nezabezpečený** web a sledujte dění na stránkách. Zobrazte si také adresář `/var/www/html/proxy/` v grafickém prohlížeči souborů. Zde uvidíte upravené obrázky tak, jak je upravuje proxy server.

Vyučujícímu zobrazte nezabezpečenou stránku s rotovanými obrázky a také obsah adresáře s rotovanými obrázky. Zobrazte také libovolnou zabezpečenou stránku (respektive chybové hlášení, jako důsledek toho, že proxy server narušuje komunikaci – útok „man-in-the-middle“).

Několik poznámek pokud vám samostatný úkol nefunguje:

1) Pokud se vám obsah webu nemění a proxy server je nakonfigurován v pořádku, tak jste nezadali potřebná práva adresáři `/var/www/html/proxy/`. To poznáte tím, že v tomto adresáři nebudou žádné obrázky z navštívených webových stránek. Také nahlédněte do logu `/var/log/squid/cache.log`, zobrazte si **konec logu** (například příkaz `tail`). Vyhledejte, že proxy server nyní spouští soubor `obracene.pl`. Pokud v logu u souboru `obracene.pl` uvidíte hlášku

```
1 /etc/squid/obracene.pl: (13) Permission denied
```

tak to značí, že jste nenastavili požadovaná práva. Po úpravě práv je potřeba proxy server znovu **restartovat**.

2) Obrázky v adresáři `/var/www/html/proxy/` jsou uloženy obráceně, ale zvolená nezabezpečená webová stránka nezobrazuje žádné obrázky. V tomto případě ověřte, že máte spuštěný webový server *Apache*.

3) Proxy server ukládá předešle navštívené stránky do vyrovnávací paměti. Pokud jste opravovali chybu, tak je potřeba navštívit jiné stránky (ty které nejsou předešle navštívené a mají tak uloženy obsah do vyrovnávací paměti). Mazání vyrovnávací paměti pro aktuálně otevřenou stránku lze také provést klávesami [CTRL]+[F5] ve Firefoxu.

4) Pokud jste si ponechávali změny z předchozích cvičení (virtuální systém nemáte uzamčený), tak je potřeba odstranit z `httpd.conf` konfiguraci virtuálních serverů. Například při začlenění doplňkové konfigurace pomocí `IncludeOptional sites/* conf` je potřeba tento řádek smazat a *Apache* restartovat. Ponechaná konfigurace se projeví tak, že obrázky v adresáři budou, ale na stránce se nebudou zobrazovat (neplaný odkaz).

BIBLIOGRAFIE

1. -. *Linux Dokumentační projekt*. Computer Press, 2008. ISBN 978-80-251-1525-1.
2. GARRELS, M. *Bash Guide for Beginners*. Fultus Publishing, 2004. ISBN 0-9744339-4-2.
3. KYSELA, M. et al. *Přecházíme na Linux*. Computer Press, 2003. ISBN 80-7226-844-9.
4. KYSELA, M. *Linux – Kapesní průvodce administrátora*. Grada, 2004. ISBN 80-247-0733-0.
5. JELÍNEK, L. *Jádro systému Linux – Kompletní průvodce programátora*. Computer Press, 2008. ISBN 9788025120842.
6. DOBŠÍČEK, M. *Linux bezpečnost a exploity*. KOPP, 2008. ISBN 978-80-7232-243-5.
7. KRCMAR, P. *Linux – Tipy a triky pro bezpečnost*. Grada, 2004. ISBN 80-247-0812-4.
8. KAMENÍK, P. *Příkazový řádek v Linuxu – Praktická řešení*. Computer Press, 2011. ISBN 978-80-251-2819-0.
9. SCHRODER, C. *Linux – Kuchařka administrátora sítě*. Computer Press, 2009. ISBN 978-80-251-2407-9.

A PŘEHLED ČASTO POUŽÍVANÝCH PŘÍKAZŮ

Tabulka A.1: Práce se stromovou strukturou.

Příkaz	**Popis**
`cd <adresar>`	změna pracovního adresáře
`cd ..`	skok o jednu úroveň směrem ke kořenovému adresáři
`cd /`	skok do kořenového adresáře
`cd`	skok do domovského adresáře (stejné jako `cd ~`)
`ls`	výpis obsahu aktuálního adresáře
`ls -a`	výpis skrytých souborů v adresáři
`pwd`	zjištění aktuálního adresáře

Tabulka A.2: Práce s právy.

Příkaz	**Popis**
`ls -l`	rozšířený výpis obsahu aktuálního adresáře (práva, typy souborů)
`chmod <prava> <soubor>`	změna práv u souboru/adresáře
`chown <uzivatel> <soubor>`	změna vlastníka
`chgrp <skupina> <soubor>`	změna skupiny

Tabulka A.3: Vytváření, mazání a obsah souboru.

Příkaz	**Popis**
`more <soubor>`	vypsání obsahu souboru
`cat <soubor>`	vypsání obsahu souboru
`head <soubor>`	vypsání prvních deseti řádků souboru
`tail <soubor>`	vypsání posledních deseti řádků souboru
`mkdir <adresar>`	vytvoření nového adresáře
`rmdir <adresar>`	smazání adresáře
`touch <soubor>`	vytvoří prázdný soubor
`rm <soubor>`	smazání souboru
`rm -rf <adresar>`	smazání adresáře, včetně obsahu bez dotazu

Tabulka A.4: Kopírování a přesun souborů.

`cp <soubor> <cil>`	kopírování souboru
`mv <soubor> <cil>`	přesun souboru
`scp <ucet@server:soubor> <cil>`	kopírování souboru ze vzdáleného počítače
`scp <soubor> <ucet@server:cil>`	kopírování souborů na vzdálený počítač
`wget <adresa>`	stažení souboru ze serveru

Tabulka A.5: Další užitečné příkazy.

Příkaz	**Popis**
`man <prikaz>`	zobrazení manuálových stránek programu (příkazu)
`clear`	vymazání obsahu obrazovky
`id`	zjištění identity
`whereis <program>`	zjištění umístění programu
`sudo <prikaz>`	vykonání příkazu s právy administrátora
`tar -xzvf <archiv.tar.gz>`	rozbalení (`x`) .gz (`z`) archívu, vypisuje průběh (`v`)
`find <adresar> -name <soubor>`	nalezení souboru

Tabulka A.6: Editace textových souborů.

Příkaz	**Popis**
`vi <soubor>`	editace souboru
klávesa [Insert],[i]	přepnutí do režimu psaní textu
klávesa [Esc]	zpět do příkazového režimu
klávesa [:]	přepnutí do režimu příkazového řádku
klávesa [/] <retezec>	vyhledání řetězce napsaného za lomítko, (klávesou [n] se přesouvá k dalším nalezeným pozicím)

Tabulka A.7: Vybrané příkazy editoru *vi*.

Příkaz	**Popis**
w	uložení souboru
w <soubor>	uložení souboru pod jiným jménem
w!	nucené uložení souboru
q	ukončení editoru
q!	nucené ukončení editoru
wq	uložení souboru a ukončení editoru
set number	zapnutí číslování řádků

Příkazy se zadávají v režimu příkazového řádku. Po vložení textu (režim psaní textu) se tento režim vyvolá stiskem klávesy [Esc] (ukončení předchozího režimu psaní) a pak znakem [:]. Příkazy se píšou za dvojtečku na posledním řádku.

B KONFIGURACE OPERAČNÍHO SYSTÉMU

Všechna cvičení

- Vytvořené účty *student* a *student2*.
- Pro tyto účty povoleno spouštění příkazů s právy administrátora (editace souboru `/etc/sudoers`).
- Grafické prostředí GNOME.
- Odstraněny všechny repositáře pro instalaci aktualizací a aplikací (kontrola pomocí `yum repolist`).
- Deaktivováno rozhraní *eth1*.

Cvičení – Modul jádra

- Instalované balíčky: *gcc*, *make*, *kernel-devel*, *kernel-headers*.

Cvičení – Virtualizace

- Vypnutý (disabled) *selinux* (setenforce 0).

Cvičení – Procesy

- Povolený démon *crond*.
- Nastavení přesměrování portů, viz Příloha C.

Cvičení – Start po síti

- V programu VirtualBox vytvořený systém s názvem *DiskLess*. Tento systém nemá připojeno žádné startovací médium (odznačit položku Boot Hard Disk). Jeho první (a jediné) síťové rozhraní je zapojeno do vnitřní sítě s názvem *intnet*. Dále v konfiguraci, záložka systém, v okně pořadí startování je zatržena položka síť a je umístěna na první místo v pořadí. Vnitřní síť *intnet* je prostředek, přes který jsou oba stroje propojeny.
- V programu VirtualBox je u CentOS další síťová karta *eth1*. Přidání se provede v nastavení → síť → karta 2 → povolit síťovou kartu. Tato síťová karta je zapojena do vnitřní sítě (internal network) s názvem *intnet*. Karta musí mít MAC adresu 0800277DBD12. Tato adresa je uvedena v připraveném síťovém profilu eth1. Pokud zde bude jiná MAC adresa, tak nepůjde nastavit IP adresa na eth1.
- Instalované balíčky: *dnsmasq*.
- Ve firewallu povoleny porty 67 (bootps) a 69 (tftp) pro protokol UDP (součástí návodu).

Cvičení – Síťový subsystém

- Instalované balíčky: *httpd*, *squid*, *perl-DBI* a *ImageMagick*.
- Vypnutý (disabled) *selinux* (setenforce 0).
- Smazané natrvalo přijaté certifikáty navštívených stránek ve Firefoxu (Preferences → Advanced → View Certificates → Servers).

C KONFIGURACE PŘESMĚROVÁNÍ PORTŮ PRO SSH

Přidělování síťové konfigurace systému CentOS v programu VirtualBox je řešeno pomocí NAT. Pro přístup pomocí SSH do vnitřního systému je nutno provést přesměrování portů pro protokol TCP – vnější port 2222 hostitelského systému na port 22 do hostovaného systému. K tomuto účelu lze využít tyto příkazy:

```
[]$ VBoxManage setextradata <guestname>
   "VBoxInternal/Devices/pcnet/0/LUN#0/Config/ssh/HostPort" 2222
[]$ VBoxManage setextradata <guestname>
   "VBoxInternal/Devices/pcnet/0/LUN#0/Config/ssh/GuestPort" 22
[]$ VBoxManage setextradata <guestname>
   "VBoxInternal/Devices/pcnet/0/LUN#0/Config/ssh/Protocol" TCP
```

kde `<guestname>` je název operačního systému. Řetězec `<ssh>` může být libovolný, ale musí být stejný ve všech příkazech. Při zadávání příkazů nesmí být VirtualBox spuštěný.

Zadané nastavení lze ověřit pomocí příkazu

```
[]$ VBoxManage getextradata <guestname> enumerate
```

Dále je ve firewallu hostitelského OS potřeba povolit port 2222.