Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Rozdíly v systémech zavádění Linuxu

Bakalářská práce

Michal Kuchta

Vedoucí práce: Mgr. Jiří Pech, Ph.D.

České Budějovice 2015

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Michal Kuchta
(jméno, příjmení, tituly)

Obor – zaměření studia: 1801R001 / Aplikovaná informatika
.....

Školitel: Mgr. Jiří Pech
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Rozdíly v systémech zavádění Linuxu.

Cíle práce :

Student popíše jednotlivé systémy zavádění Linuxu (System V init, systemd, upstart atd.) a ukáže jak se liší psaní zaváděcích skriptů pro tyto systémy. Práce by měla zejména obsahovat následující body:

1. Student popíše historický vývoj systémů zavádění a správy démonů u Linuxu.
2. Pro každý ze systémů popíše způsob vytváření skriptů pro zavádění (ukončování a správu) služeb a skriptů pro automatické činnosti prováděné při startu nebo vypnutí systému.
3. Na několika příkladech popíše rozdíly v jednotlivých metodách.
4. Pro každou metodu zvolí vhodnou distribuce Linuxu, která ji používá a příklady ověří v praxi.
5. Jednotlivé metody na závěr porovná a zhodnotí.

Základní doporučená literatura :

KRČMÁŘ, Petr. Upstart zpřehlední a zrychlí start systému. *Root.cz* [online]. [cit. 2013-10-11]. Dostupné z: http://www.root.cz/clanky/upstart-zprehledni-a-zrychli-start-systemu/

KRAUSE, Michal. Linux - druhý pohled do nitra. *Root.cz* [online]. [cit. 2013-10-11]. Dostupné z: http://www.root.cz/clanky/linux-druhy-pohled-do-nitra/

Financování práce :.....

Vedoucí práce: Mgr. Jiří Pech, Ph.D. podpis : .....

U externích vedoucích fakultní garant práce..... podpis : .....

Garant oboru bakalářského studia

..... podpis : .....

Vedoucí katedry: RNDr. Libor Dostálek podpis .....

Případný souhlas vedoucího ústavu AV ..... podpis : .....

V Českých Budějovicích dne ..... 4.11.2013.....

Převzal/a dne..... 4.11.2013..... podpis : .....

Bibliografické údaje

Kuchta M. 2015: Rozdíly v systémech zavádění Linuxu. [Differences in the init systems of Linux. Bc.. Thesis, in Czech.] – 80 p. , Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Bakalářská práce se zabývá rozdíly ve psaní skriptů pro jednotlivé *init* systémy, které jsou využívané v různých Linuxových distribucích. V práci jsou popsány základní užívané *init* systémy pro tyto distribuce. Hlavní část je zaměřena na metodiku psaní skriptů v *init* systémech *System V init*, *SystemD* a *Upstart*, spolu s příklady několika skriptů, které ukáží rozdíl mezi těmito *init* systémy. Pomocí zjištěných výsledků jsou porovnány vybrané *init* systémy a vybrán nejvhodnější *init* systém.

Annotation

The bachelor thesis deals with the differences in writing scripts for particular *init* systems used in different Linux distributions. The thesis also describes basic *init* systems used for these distributions. The main part of this thesis is focused on the methodology of writing scripts in following *init* systems: *System V init*, *SystemD* and *Upstart* including several scripts which show differences between these systems. Collected results are used for the comparison of chosen *init* systems and the best *init* system is selected.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích, dne 24.4.2015

.....

podpis

Poděkování

Chtěl bych poděkovat vedoucímu mé práce, Mgr. Jiřímu Pechovi, Ph.D., za ochotu, vstřícný přístup, odborné rady a konstruktivní připomínky při vypracování této bakalářské práce.

Také bych chtěl poděkovat svým známým za nekonečnou trpělivost a morální podporu při psaní této práce.

Obsah

1 Úvod

Tato bakalářská práce se zabývá rozdíly mezi jednotlivými systémy zavádění služeb a správy démonů (*init* systémy) v Linuxu, mezi které patří *Sysvinit, Systemd* a *Upstart*. Samotný *init* má v distribucích, založených na operačním systému Linux, velký vliv. Obsahuje *PID* (*process ID*) s číslem 1 a stará se o spuštění nebo zastavení jednotlivých scriptů, pomocí kterých jsou následně spouštěny služby, jako jsou například síť nebo sběrnice *D-bus*, díky kterým je systém schopen plně fungovat hned při naběhnutí systému.

Práce se věnuje jmenovaným druhům *init* systémů, jejich hlavním výhodám a nevýhodám, a také využití těchto *init* systémů v těch linuxových distribucích, pro které jsou využity jako výchozí *init* systémy.

Dále se práce zabývá rozdíly ve tvoření skriptů v jednotlivých *init* systémech. Pro každý systém zavádění je rozebráno několik skriptů, některé již připravené v těchto systémech a některé nově vytvořené, které jsou porovnány v jednotlivých distribucích.

Pomocí jejich porovnání, a také na základě zjištěných poznatků, je navrženo nejvhodnější zavádění pro každou distribuci *Linuxu*. K tomuto porovnání a vyhodnocení slouží také názory uživatelů těchto systémů.

2 Cíle práce

Hlavním cílem této bakalářské práce je zjištění rozdílů pří psaní skriptů v jednotlivých systémech zavádění *Linuxu*, *System V init*, *Systemd* a *Upstart*. Práce má zá cíl několik bodů:

1) Popsat historický vývoj systémů zavádění a správy démonů[1] v Linuxu.

[1]Démon neboli Daemon(**D**isk **a**nd **E**xecution **Mon**itor) je dlouhotrvající proces, běžící na pozadí, který odpovídá na požadavky pro jednotlivé služby.

2) Popsat způsoby vytváření skriptů pro zavádění (ukončování a správu) služeb a skriptů pro automatické činnosti prováděné při startu nebo vypnutí systému pro každý ze systémů.

3) Rozdíly v jednotlivých systémech popsat na několika příkladech.

4) Pro každý systém zvolit vhodnou distribuci Linuxu. Příklady ověřit v praxi.

5) Jednotlivé metody porovnat a zhodnotit.

Teoretická část se zabývá vývojem těchto systémů zavádění a *démonů* v Linuxu, spolu s jejich hlavními výhodami a nevýhodami.

Praktická část obsahuje metodiku psaní skriptů pro zavádění *System V init*, *Systemd* a *Upstart*. Dále obsahuje popis několika skriptů pro každé zavádění, díky kterým jsou zobrazeny jednotlivé rozdíly mezi nimi. Celkově jsou popsány 4 skripty, z toho dva již obsažené v samotných systémech. Zbylé dva skripty jsou, pro každé zavádění systému, nově vytvořeny pomocí zmíněných metodik.

Závěr se věnuje výběru vhodné distribuce Linuxu pro každé zavádění a také vyhodnocení jednotlivých *init* systémů.

3 Použité nástroje

Pro psaní této bakalářské práce byly využity linuxové distribuce *Ubuntu*, *Debian* a *Fedora,* na kterých byly následně všechny zkoumané skripty testovány.

Všechny tyto zmíněné distribuce byly nainstalovány jako virtuální počítače pomocí programu *Oracle VM VirtualBox,* ve verzi 4.3.8. Na těchto virtuálních počítačích byly následně testovány jednotlivé *init* systémy.

Linuxová distribuce *Ubuntu* byla testována ve verzi 13.10 spolu s *init* systémem *Upstart* 1.10. Operační systém *Debian* byl ve verzi 7.4.0 a

jeho *init* systém byl *Sysvinit* verze 2.88. Poslední distribuce Linuxu*, Fedora,* byla nainstalována a testována ve verzi 20 s *init* systémem *Systemd.*

Pro všechny distribuce byly použity 32bitové verze těchto systémů.

4 Teoretická Část

4.1 Init

Init je prvním běžícím procesem *UNIX* systémů, spuštěným po načtení jádra, který se stará nejen o správný start systému, ale také o jeho ukončení.

Jelikož se *init* proces stává po načtení jádra kořenem ostatních procesů (má *PID 1*), není možné ho jakkoliv ukončit příkazy při běhu systému, a k jeho ukončení tak dochází až při ukončení celého systému.

Samotný *init* je program, který se nachází v adresáři */sbin*. Jeho konfigurace se však provádí v adresáři */etc/inittab*. V konfiguraci je několik řádků obsahující příkazy, které se skládají ze čtyř částí - identifikátor příkazu, výpis všech úrovní běhu, pro které bude daný příkaz platit, dále akce, která se provede pro daný příkaz a nakonec příkaz, který se vykoná. Všechny části jsou odděleny *":"*. Každý řádek je tedy ve formátu *id:runlevels:action:process*[2].

V textovém souboru *inittab* se nastavuje například výchozí úroveň běhu systému nebo který *rc* adresář bude obsluhovat jednotlivé úrovně běhu. V případě přepnutí do jiné úrovně běhu dojde ke spuštění skriptů z daného *rc* adresáře a spuštění daných služeb. Případně může dojít k jejich ukončení. Všechny tyto skripty se nacházejí pouze v adresáři */etc/init.d*. Do všech *rc* adresářů jsou však zavedeny symbolické odkazy typu *S02ssh -> ../init.d/ssh*, které odkazují na skutečné umístění skriptů. V tomto souboru je také

[2] MOLIČ, Jan. Inicializace aneb od Initu k Runitu. Dostupné z: http://www.root.cz/clanky/inicializace-aneb-od-initu-k-runitu/

nastaveno, co dělat při současném stisknutí kláves *ctrl, alt* a *delete*, nebo také nastavení startu 6 konzolí (*tty*) [3].

4.1.1 System V init (Sysvinit)

Jedním z prvních *init* systémů byl *Sysvinit init* systém, který byl prvně použit v roce 1983[4].

Oproti novějším *init* systémům, jako je například *Upstart* nebo *Systemd*, zde záleží na pořadí jednotlivých služeb. Tyto služby jsou zapsány v */etc/rc$úroveň.d/*, v případě systému *Ubuntu*, nebo v */etc/init.d/rc$úroveň.d. $úroveň* určuje *runlevel*, ze kterého budou skripty spuštěny. Například */etc/rc6.d* (*runlevel* 6 označuje restart). V těchto adresářích jsou obsaženy symbolické odkazy těchto skriptů, které se skládají ze tří částí. První částí je buď znak *S* (*start*), který označuje start služby, nebo znak *K* (*kill*), který slouží k ukončení procesu. Druhá část obsahuje pouze číslo, které udává, kdy se daný skript spustí. Čím menší číslo, tím dříve se daný skript vykoná. Poslední částí je název služby, která se má spustit. Příkladem může být *S55sshd*, kdy bude symbolický odkaz ve tvaru *S55sshd -> ../init.d/sshd*. Služby jsou navíc spouštěny jednotlivě, což má za následek jejich pomalejší start než u ostatních *init* systémů, které dokáží spustit služby paralelně.

V případě, že mají dvě služby stejné pořadové číslo, dojde ke spuštění služeb v abecedním pořadí. Pokud je tedy služba *S55x*, a služba *S55k*, jako první bude v tomto případě spuštěna služba *S55k*.

Sysvinit také dokáže sledovat procesy, ale pouze ty, které se nacházejí v adresáři */etc/inittab*[5].

[3] Introduction to Linux: Processes. In: [online]. [cit. 2014-10-27]. Dostupné z: http://www.tldp.org/LDP/intro-linux/html/sect_04_02.html
[4] System V Definition. [online]. [cit. 2014-10-21]. Dostupné z: http://www.linfo.org/system_v.html
[5] Systemd – úvod a představení System V init: Na počátku byl (System V) init. [online]. [cit. 2014-10-26]. Dostupné z: http://www.abclinuxu.cz/clanky/systemd-uvod-a-predstaveni-system-v-init#na-pocatku-byl-system-v-init

4.1.2 BSD init

BSD Init je dalším systémem zavádění. Oproti *Sysvinit*, který je používán zejména na Linuxových distribucích, má však *BSD Init* své uplatnění hlavně na unixových systémech typu *BSD (Berkeley Software Distribution)*. I přes to je ho možné najít i v některých používaných distribucích *Linuxu*. Takovým příkladem může být například distribuce *Slackware*.

Mezi *UNIX* operační systémy používající *BSD init* patří *FreeBSD*, *OpenBSD* nebo *NetBSD*[6].

Výhodou *BSD init* je, že se zde, oproti *Sysvinit* init systému, nachází v adresáři */etc/rc shell* skript *rc.conf*, ve kterém je nastaveno, které služby se mají spustit, a díky tomu je start systému rychlejší. Je však potřeba dbát na správnost kódu tohoto skriptu, kdy v případě chyby nemusí dojít k naběhnutí systému[7].

Při zavádění systému jsou přečteny soubory */etc/rc.conf* a *etc/defaults/rc.conf*, ve kterých je určeno, který služby mají být spuštěny. Tyto služby jsou následně spuštěny pomocí jejich skriptů, které jsou uloženy v adresáři */etc/rc.d*[8].

4.1.3 Upstart

Zavádění *Upstart* bylo vytvořeno v roce 2006 *Scottem James Remnantem* pro distribuci *Ubuntu*. Zatím poslední vydanou, a také stabilní verzí, je verze 14.10, která byla vydána 23. Října 2014.

[6] HAGARA, Ladislav. Source Mage GNU/Linux: Není init jako init: BSD init. [online]. [cit. 2014-11-11]. Dostupné z: http://www.root.cz/clanky/source-mage-gnu-linux-init/
[7] Jak startuje systém: Proces init. In: [online]. [cit. 2015-03-31]. Dostupné z: http://www.linuxexpres.cz/praxe/jak-startuje-system
[8] System Startup. [online]. [cit. 2014-11-11]. Dostupné z: https://www.freebsd.org/doc/en/articles/linux-users/startup.html

Upstart je náhrada pro */sbin/init* démona, který dohlíží na zastavení a spuštění služeb při zavádění systému (spuštění služeb) a také při jeho vypnutí (zastavení služeb)[9].

Jednou z mnoha předností *Upstart* je zachování zpětné kompatibility s klasickým *Sysvinit*, kdy není potřeba přepsání startovacích skriptů jako je tomu potřeba u některých ostatních zavádění. Oproti klasickému *Sysvinit* spravuje *Upstart* služby zcela jiným způsobem, protože dokáže spouštět určité služby bez potřeby znalosti pořadí zaváděných služeb, tedy zcela asynchronně a dynamicky. Výhodu je i to, že služby mohou být spouštěny najednou, nezávisle na sobě. Díky tomu má *Upstart* také rychlejší start než *Sysvinit*. Toho je dosaženo zavedením systému akcí (*jobs*), kdy jsou tyto akce volány na základě událostí. Tato událost může spustit více akcí[10].

Upstart obsahuje dvě hlavní části. Událost „*event*“ a akci „*job*“. Jako *job* je buď označován úkol "*Task*" nebo služba "*Service*"[11].

Pokud určitá služba splňuje určité požadavky, je spuštěna událost a daná služba je automaticky zastavena nebo může být naopak spuštěna. Více akcí může být spuštěno najednou „paralelně“, pokud mají stejné podmínky pro spuštění této akce[12].

4.1.3.1 Job

Pojem akce „*job"* není nic jiného než označení pro skript, který je využitý v Upstart. Tyto skripty se dělí do tří druhů. Jedním z nich je *Task Job*, který spouští krátce běžící procesy, které mají konečný stav a je jasné,

[9] Upstart - event-based init deamon. In: [online]. [cit. 2014-03-14]. Dostupné z: http://upstart.ubuntu.com/index.html
[10] KRČMÁŘ, Petr. Upstart zpřehlední a zrychlí start systému. Root.cz [online]. [cit. 2013-10-11]. Dostupné z: http://www.root.cz/clanky/upstart-zprehledni-a-zrychli-start-systemu/
[11] HUNT, James a Clint BYRUM. Upstart Intro, Cookbook and Best Practises: Concepts and Terminology. In: [online]. [cit. 2014-11-15]. Dostupné z: http://upstart.ubuntu.com/cookbook/#concepts-and-terminology
[12] HUNT, James a Clint BYRUM. Upstart Intro, Cookbook and Best Practises: Upstart's Design: Why It Is Revolutionary. In: [online]. [cit. 2014-11-15]. Dostupné z: http://upstart.ubuntu.com/cookbook/#upstart-s-design-why-it-is-revolutionary

že budou někdy ukončeny. Příkladem může být mazání souboru, kdy je *Task Job* spuštěn při akci mazání a ukončen je ve chvíli, kdy je soubor smazán.

Dalším druhem je *Service Job*, který je oproti *Task Job* dlouhotrvající proces. *Service Job* nemusí být sám ukončen. Příkladem může být webový server nebo databáze.

Posledním typem je *Abstract Job*, který je pouze v *Upstartu* samotném. Tento typ neobsahuje žádnou *skript* sekci a také nemůže mít žádný podřízený proces (*PID*). Pokud se spustí určitá úloha (*job*), běží tato úloha, dokud není ukončena administrátorem.

Všechny skripty jsou definovány v Job konfiguračním souboru *<name>.conf,* který obsahuje *Stanzy*. *Stanza* je směrnice v konfiguračním souboru, pomocí které je tvořen *Upstart* skript (*job*). Jako stanzy jsou využívány zejména *exec*, *script*, *start on* nebo *respawn*.

Konfigurační soubor se může nacházet buď v */etc/init/*, a to pokud se jedná o *System Job*, nebo v *$HOME/.init/* pokud jde o *User Job*. *User Job* byl přidán v *Upstart* verzi 1.3 a k jeho aktivaci je potřeba modifikovat soubor *upstart.conf*, který se nachází v */etc/dbus-1/system.d/Upstart.conf*, a povolit uživateli všechny *D-Bus* vlastnosti a metody[13].

[13] HUNT, James a Clint BYRUM. Upstart Intro, Cookbook and Best Practises: Concepts and Terminology. In: [online]. [cit. 2014-11-15]. Dostupné z: http://upstart.ubuntu.com/cookbook/#concepts-and-terminology

Současný stav	***Cílový stav***	
	Spuštění	***Zastavení***
waiting	*starting*	*n/a*
starting	*pre-start*	*stopping*
pre-start	*spawned*	*stopping*
spawned	*post-start*	*stopping*
pre-start	*running*	*stopping*
running	*stopping*	*pre-stop* **nebo** *stopping*
pre-stop	*running*	*stopping*
stopping	*killed*	*killed*
killed	*post-stop*	*post-stop*
post-stop	*starting*	*waiting*

Tabulka 1 Všechny možné stavy a jejich přechody v *Upstart init* systému

Skripty jsou tvořeny stavy „*states*“ a událostmi „*events*“. V *Upstart* může být nastaveno 10 různých stavů v daných službách – *waiting*, *starting*, *pre-start*, *spawned*, *post-start*, *running*, *pre-stop*, *stopping*, *killed* a *post-stop*.

Přechod mezi jednotlivými stavy je velice jednoduchý. Pokud je nyní „*job*“ ve stavu *pre-start*, a má mít cílový *start*, bude změněn do stavu *spawned*. V opačném případě, pokud má mít cílový stav *stop*, bude přesunut do stavu *stopping*.

4.1.3.2 Spuštění a zastavení Upstart skriptu

Předtím než je *job* spuštěn, je zobrazen jako *stop/waiting*. *Stop* značí cíl tohoto jobu, zatímco druhým uvedeným, *waiting*, je označen jeho stav. Pokud je job spuštěn, je změněn jeho cíl na *start* a jeho stav na *starting*. Poté je emitován *starting* event. V případě, že je tento job použit u jiných jobů ve *start on starting* nebo *stop on starting* stanzách, dojde k jejich spuštění. Jakmile je *starting* event dokončen, dojde ke změně stavu daného jobu na *pre-start*. Pokud by existovala *pre-start* sekce, byl by vytvořen její

proces a došlo by k jejímu provedení. Kdyby během *pre-start* procesu došlo k určité chybě, celý skript by byl ukončen. Pokud by však k žádné chybě nedošlo, byl by následně vytvořen hlavní proces. Následně by byl job převeden do stavu *spawned* a bylo by určeno konečné *PID* tohoto jobu. Job je poté převeden do stavu *post-start*, kdy dojde k provedení *post-start* stanzy, pokud se v daném skriptu také nachází. V případě, že by v daném jobu nebyla tato stanza obsažena, byl by job změněn do dalšího stavu pro spuštění, kterým je *running*. Poté je emitován *started* event a všechny joby, ve kterých je tento job obsažen ve *start on started* nebo *stop on started* stanzách, budou provedeny[14].

Když je job úspěšně proveden, nachází se ve stavu *start/running*. Pokud má být zastaven, je jeho cíl změněn ze *start* na *stop* a jeho stav je změněn z *running* na *pre-stop*. V případě, že je v daném skriptu obsažena *pre-stop* stanza, je vytvořen její proces, a stav jobu je následně upraven na *stopping*. Když je job ve stavu *stopping*, je emitován *stopping* event. Poté budou provedeny joby, obsahující ve *stop on* nebo *start on* stanzách *stopping* event s hodnotou tohoto jobu. Job je následně přesunut do stavu *killed* a dochází k zastavení hlavního procesu. Po zastavení hlavního procesu dojde v případě existence *post-stop* stanzy k jejímu provedení a následně na to k převedení stavu do *waiting*. Nakonec je emitován event *stopped* a všechny joby, ve kterých je tento job obsažen ve *start on stopped* nebo *stop on stopped* stanzách, budou provedeny. Po dokončení *stopped* eventu se job bude opět nacházet ve stavu *stop/waiting*[15].

4.1.3.3 Event

Pod pojmem událost *„event“* si je možné představit buď signály, metody *„methods“* nebo takzvané *„hooks“* události, které oznamují, že

[14] HUNT, James a Clint BYRUM. Job Lifecycle: Starting a job. [online]. [cit. 2015-01-19]. Dostupné z: http://upstart.ubuntu.com/cookbook/#starting-a-job

[15] HUNT, James a Clint BYRUM. Job Lifecycle: Stopping a Job. [online]. [cit. 2015-01-19]. Dostupné z: http://upstart.ubuntu.com/cookbook/#stopping-a-job

došlo ke změně v systému. Mezi *„hooks“ eventy* patří *starting* nebo *stopping*[16]. Všechny tyto možnosti jsou prováděny pomocí celého systému. V případě, že je událost nebo akce emitována, neboli vytvářena a vysílána, není ji možné v tu chvíli ukončit.

Event je vytvořen příkazem *# initctl emit <event>.*

Akce „*job*“ je ve většině případů spuštěna nebo ukončena v případě spuštění a ukončení dalších akcí. V *Upstart* je obsažena sada speciálních událostí, které obsahují přechody mezi těmito akcemi. Ačkoliv názvy některých událostí mají stejný název s přechody akcí, je důležité si uvědomit, že nepopisují stejnou věc. Události jsou pouze 4 a patří mezi ně *starting*, *started*, *stopped a stopping*.

Event starting je proveden, jakmile má dojít ke spuštění příslušné akce. *Started event* je emitován v případě, že daná akce již probíhá. *stopping* je vytvořen v momentě, kdy má dojít k ukončení akce. Poslední událost, *stopped*, je provedena jakmile je dokončena daná akce a nezáleží, jestli úspěšně nebo neúspěšně[17].

4.1.4 Systemd

Systemd je další z mnoha *Init* systémů pro Linux, vyvinutý Lennartem Poetteringem a vydaný v dubnu roku 2010. Zatím poslední stabilní verzí je 219. *Systemd* je použit jako výchozí *init* v distribuci Fedora od verze 15.

[16] HUNT, James a Clint BYRUM. Event Types: Hooks. In: [online]. [cit. 2015-03-05]. Dostupné z: http://upstart.ubuntu.com/cookbook/#hooks
[17] HUNT, James a Clint BYRUM. *Upstart Intro, Cookbook and Best Practises: Event* [online]. [cit. 2014-11-16]. Dostupné z: http://upstart.ubuntu.com/cookbook/#event

Mezi jeho výhody patří částečná kompatibilita s klasickým *Sysvinit*. Podporuje například *init* skripty, *LSB* hlavičky. Oproti klasickému *Sysvinit* může start služeb probíhat buď *přes D-bus* nebo přes použití *socketů*[18].

Ke správě služeb zde najdeme program *systemctl*, pomocí kterého mohou být službu jak blokovány, tak i povolovány. Může také docházet k jejich restartování nebo načtení konfigurace dané služby.

Základní částí *Systemd* jsou jednotky „*Units*“ a cíle „*Targets*“.

4.1.4.1 Systemd UNITS

Základem *Systemd* jsou jednotky „*units*“, kterých je dohromady 12. Ve všech jsou obsaženy důležité objekty potřebné například ke správnému spuštění systému. Ve většině případů probíhá konfigurace těchto jednotek v konfiguračních souborech. Některé jednotky jsou již vytvořeny pomocí existujících jednotek, a to například při změně stavu systému. Každá jednotka je složena ze dvou částí, jména a třídy (například *default.target*). Jméno jednotky se shoduje s názvem konfiguračního souboru. Může mít 3 stavy: *aktivní* (spuštěno), *nečinný* (zastaveno) a *neúspěšný* (chyba v kódu). Pokud je nastaven stav na neúspěšný, je zaznamenána daná chyba[19].

Tyto jednotky jsou uloženy v adresářích */etc/systemd/system* a */usr/lib/systemd/system*. Nastaveni je poté konfigurováno v *[Unit]* nebo *[Install]* sekci jednotek souborů[20].

[18] VYSKOČIL, Michal. Systemd – principy: Co je systemd. [online]. [cit. 2014-11-16]. Dostupné z: http://www.abclinuxu.cz/clanky/systemd-principy
[19] *Concepts* [online]. [cit. 2014-11-15]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.html
[20] *Description* [online]. [cit. 2014-11-15]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.unit.html

4.1.4.2 Systemd třídy

Služba (.service) - Používá se ke konfiguraci *.service* souborů. Slouží k nastavení démonů. Nastavuje spuštění, restartování nebo např. zastavení služby. Pokud neexistuje konfigurační soubor (například *netcfg.service*) jednotky, je použit *SysV init* skript, který má stejné jméno, ale už neobsahuje *suffix .service* (pouze *netcfg*). Z tohoto skriptu je poté vytvořena jednotka služby[21].

Zařízení (.device) - Slouží ke konfiguraci zařízení, které lze najít v */dev* nebo také v */sysfs*. Název souboru musí odpovídat danému zařízení, například pro */dev/sda5* je konfigurační soubor *dev-sda5.device*[22].

Socket (.socket) - Kóduje informaci o *IPC* nebo o síťovém socketu. Mezi sockety může patřit například *AF_UNIX*, *AF_INET* nebo *AF_NETLINK*. Pro socket soubor existuje *service* soubor, který je spuštěn v okamžiku připojení na socket. Například *nscd.socket* spouští službu *nscd.service*. Jméno služby a socketu je ve většině případů shodné, ale je možné změnit *service* jednotku. Toho lze dosáhnout změnou v konfiguraci, kdy se změní hodnota *Service=* na název *service* jednotky, která se spustí v okamžiku připojení na socket. To platí pouze pro sockety s proměnnou *Accept=no*. K tomuto nastavení dochází jen v ojedinělých případech[23].

Připojení (.mount) - Představuje body k připojení v souborovém systému. Tato jednotka je pojmenována podle přípojného bodu. Přípojný bod */home/majkl* je tedy konfigurován v souboru *home-majkl.mount*. V případě, že je přípojný bod pod jiným přípojným bodem, je závislost mezi nimi vytvořena automaticky. Konfigurace je prováděna přes */etc/fstab*. Je

[21] Systemd.service: Descripion. In: [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.service.html
[22] Systemd.device: Descripion. In: [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.device.html
[23] Systemd.socket: Descripion. In: [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.socket.html

možné konfigurovat také přes *unit* soubory. Podobně pracuje také automatické připojení (*.automount*)[24].

Automatické připojení (.automount) - Představuje bod k připojení v souborovém systému, který je připojen při prvním přístupu. Podobně jako *mount* jednotka musí být pojmenováno podle přípojného bodu, ale jako suffix je zde *.automount* namísto *.mount* . Například */home/majkl* je tedy konfigurován v souboru *home-majkl.automount* . Konfigurace je, stejně jako u jednotky *mount*, prováděna přes */etc/fstab*[25].

Snímek (.snapshot) - Pomocí této jednotky lze uložit aktuální stav dalších jednotek. To může nastat například při nouzové opravě. Jednotky jsou po této akci znovu obnoveny. Služby jsou ve většině případů vypnuty a poté při spuštění znovu zapnuty. Snímky jsou vytvořeny pomocí *systemctl*. Uložený stav je zavolán přes *systemctl isolate*[26].

Časovač (.timer) - Spouští další jednotku jakmile uplyne určitý čas. Pokud není zadáno jinak, každá jednotka *timer* se shoduje se *service* jednotkou, která je spuštěna po uplynutí času. Například *script.timer* aktivuje službu *script.service*. Jednotku, která se má spustit, je možné změnit v nastavení *timer* souboru pomocí příkazu *unit=*[27].

Odkládací paměť (.swap) - Nese informaci o úložištích pro tuto paměť. Pokud je *swap* jednotka spuštěna z nějakého zařízení, je mezi nimi automaticky nastavena závislost. Konfigurace je, jako u většiny jednotek, prováděna přes */ets/fstab*[28].

---

[24] Systemd.mount: Descripion. In: [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.mount.html
[25] Systemd.automount: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.automount.html
[26] Systemd.snapshot: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.snapshot.html
[27] Systemd.timer: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.timer.html
[28] Systemd.swap: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.swap.html

Proměnná (.path) - Jednotka může aktivovat další služby, v případě, že je změněn nebo modifikován určitý soubor nebo adresář. K tomu slouží inotify API, který monitoruje systém souborů. V nastavení *path* sekce může být například parametr *PathModified=*, který aktivuje jednotku, která se má spustit, kdykoli je do pozorovaného souboru něco zapsáno. Dalším nastavením může být *PathChanged=*, kdy je jednotka aktivována až po ukončení modifikovaného souboru. Pomocí *Unit=* je zvolena jednotka, která bude aktivována[29].

Slice (.slice) – Zdroj pro skupinu procesu, které jsou řízeny hierarchicky. Toho je dosahováno vytvořením uzlu v *cgroup tree*. Pro určitý *slice* může být přiřazena další jednotka, například jednotka *scope* nebo *service*. Cesta od *root slice* do *slice* se uvádí pomocí názvů, které jsou odděleny pomlčkou. V *root .slice* se může nacházet *foo.slice*, ve kterém se může nacházet například *foo-bar.slice.* Tyto jednotky se běžně nacházejí v *systém.slice*[30].

Scope (.scope) – *Scope* jednotky jsou vytvářeny programově pomocí bus rozhraní *Systemd*. Nejsou tedy konfigurovány přes *unit* konfigurační soubory. Tato jednotka nastavuje množství systémových procesů. Účelem této jednotky je seskupení pracovních procesů systémových služeb, které slouží k řízení zdrojů[31].

Cíl (.target) - Nepoužívá speciální nastavení konfigurace, jako je tomu například u jednotky *service* nebo *device*, ale řídí se podle nastavení pro všechny jednotky. Když je například nastaveno *ctrl-alt-del.target* a do konzole je zadáno *control+alt+delete* je tento cíl spuštěn. Cíle mohou být

[29] Systemd.path: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.path.html
[30] Systemd.slice: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.slice.html
[31] Systemd.scope: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.scope.html

také spuštěny, když se zařízení objeví v systému. Například při *printer.target*, je cíl spuštěn poté, co je tiskárna zapojena do systému[32].

4.1.4.3 systemd Targets

Mechanizmus, pomocí kterého je *Systemd* schopen spouštět více procesů ve stejný čas, jako je tomu například při zavádění systému[33].

Targets využívají *target* jednotky, kde je jejich cílem vytvořit závislosti mezi ostatními *Units*. Například pomocí *multi-user.target* může být spuštěn *NetworkManager.service* a také je aktivována unit *basic.target*[34].

Úroveň běhu	*Target jednotka*	*Popis úrovně běhu*
0	runlevel0.target, poweroff.target	Vypnutí systému
1	runlevel1.target, rescue.target	Jednouživatelský režim
2	runlevel2.target, multi-user.target	Standartní uživatelská režim
3	runlevel3.target, multi-user.target	Víceuživatelský režim
4	runlevel4.target, multi-user.target	Uživatelský definovaný režim
5	runlevel5.target, graphical.target	Grafický víceuživatelský režim
6	runlevel6.target, reboot.target	Reset systému

Tabulka 2 Porovnání úrovní běhu v *SysV* se *Systemd Targets* v Red Hat Enterprise

[32] Systemd.target: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.target.html
[33] Getting Started with systemd: Terminology. In: [online]. [cit. 2014-11-16]. Dostupné z: https://coreos.com/docs/launching-containers/launching/getting-started-with-systemd/
[34] Working with systemd Targets. In: [online]. [cit. 2014-11-16]. Dostupné z: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/sect-Managing_Services_with_systemd-Targets.html

5 Praktická část

Praktická část se věnuje metodice psaní skriptů pro zavádění *System V init*, *Systemd* a *Upstart*. Dále obsahuje popis několika skriptů pro každé zavádění, díky kterým jsou zobrazeny jednotlivé rozdíly mezi nimi. Celkově jsou popsány 4 skripty, z toho dva již připravené v samotných systémech. Zbylé dva skripty jsou, pro každé zavádění systému, nově vytvořeny pomocí zmíněných metodik.

5.1 Sysvinit

Prvními skripty, které budou ukázány, jsou *Sysvinit* skripty. K jejich provedení může dojít buď při zavádění systému, nebo při změně úrovně běhu. Pro každou z úrovní běhu jsou nastaveny určité skripty, které budou spuštěny nebo ukončeny. Všechny skripty jsou uloženy v adresáři */etc/init.d/*[35]. Skripty, které mají být spuštěny v určitých úrovních běhu, jsou skrze symbolické odkazy vytvořeny v adresářích */etc/init.d/rcY.d*, kde jsou znakem *Y* reprezentovány úrovně běhu, ve kterých má být tento skript buď spuštěn, nebo ukončen.

Pokud má být při spuštění nebo ukončení systému spuštěna určitá služba, je potřeba vytvořit nový skript, který bude uložen do adresáře */etc/init.d*, kde se nachází všechny skripty. Tomuto souboru poté musí být přidělena daná práva. To je dosaženo příkazem *chmod 755 /etc/init.d/z*, kde je znakem *z* označen název skriptu, kterému mají být přiřazena příslušná práva. Nakonec musí být vytvořen symbolický odkaz v těch úrovních běhu, ve kterých má být daný skript spuštěn nebo ukončen. K jeho vytvoření dojde pomocí příkazu *ln –s /etc/init.d/z název_adresáře/./XYz*, kde znak *X* obsahuje buď znak *S* (start) nebo *K* (kill), podle toho, zda má být skript spuštěn nebo ukončen. Znak *Y* obsahuje pouze dvoumístné číslo, kterým je určeno, kdy má dojít k provedení tohoto skriptu.

[35] Creating your own init skripts. In: [online]. [cit. 2014-11-16]. Dostupné z: http://www.softpanorama.org/Commercial_linuxes/Startup_and_shutdown/creating_your_own_init_skripts.shtml

Kdyby měl být například skript *dbus* ukončen při úrovni běhu 6, bude symbolický odkaz vytvořen v adresáři */etc/init.d/rc6.d* a bude vytvořen pomocí příkazu *ln –s /etc/init.d/dbus /etc/init.d/rc6.d/./K30dbus*. Nejprve jsou spuštěny *Kill* skripty v daném pořadí. Poté dojde ke spuštění *Start* skriptů.

Skript je ovládán pomocí příkazu *service název_skriptu příkaz.* Příkazů pro Sysvinit skript je dohromady 6 – *start, stop, restart, reload, force-reload* a *status*. První dva příkazy, *start* a *stop*, spouští a zastavují službu, kterou skript spouští. *Restart* zastaví a znovu spustí určitou službu, pokud je již spuštěna. V případě, že by služba byla zastavená, má stejný význam jako příkaz *start.* Pomocí příkazu *reload* je znovu načtena konfigurace služby bez toho, aby došlo k zastavení a restartu služby. Příkaz *force-reload* má podobné vlastnosti jako *reload*, ale funguje pouze v případě, že to daná služba podporuje. Pokud služba tento příkaz nepodporuje, dojde k restartu služby. Příkazy *start, stop, restart* a *force-reload* by měly být podporovány u všech init skriptů. Přes příkaz *status* je vypsán stav služby.

```
etc/init.d# service bluetooth status
[FAIL] bluetooth is not running … failed!

etc//init.d# service bluetooth restart
[ ok ] Stopping bluetooth: /sbin/bluetoothd.
[ ok ] Starting bluetooth: bluetoothd rfcomm.
```

Všechny příkazy, které je možné u daného skriptu použít, jsou dostupné pomocí příkazu *service název_skriptu*.

```
/etc/init.d# service bluetooth
Usage: /etc/init.d/bluetooth
{start|stop|restart|force-reload|status}
```

K vytvoření vlastního skriptu je možné použít šablonu, která se nachází v adresáři */etc/init.d/skeleton*.

Na začátku každého *Sysvinit* skriptu by měl být obsažen tzv. *INIT INFO* blok. Blok je ve tvaru

```
### BEGIN INIT INFO
# klíčové slovo: argument_1 argument_2 …
### END INIT INFO
```

Mezi tyto klíčová slova patří *provides, required-start, required-stop, should-start, should-stop, default-start, default-stop, short-description* a *description*. Skript by měl obsahovat alespoň tyto řádky zmíněně níže.

```
### BEGIN INIT INFO
# Provides:          FOO
# Required-Start:    $syslog $remote_fs
# Required-Stop:     $syslog $remote_fs
# Default-Start:     3 5
# Default-Stop:      0 1 2 6
# Description:       Start FOO to allow XY
### END INIT INFO
```

První řádka tohoto bloku, *Provides,* určuje jméno služby, která bude spuštěna tímto skriptem. Řádky *Required-Start* a *Required-stop* informují, které služby nebo zaváděcí zařízení musí být dostupné před tím, než dojde ke startu nebo ukončení tohoto skriptu. Pomocí *Default-Start* a *Default-stop* je určeno, v jaké úrovni běhu by mělo dojít ke spuštění nebo naopak k ukončení tohoto skriptu. Řádka *Description*, stejně jako u *Upstart jobu*, pouze popisuje, k čemu je daná služba určena. V tomto bloku se může nacházet také řádka *# Short-Description.* Její účel je ale stejný jako u *# Description*[36]. *Should-start* a *should-stop* obsahují seznam služeb, které by měli být dostupné pří spuštění nebo ukončení skriptu.

[36] Creating your own init skripts. In: [online]. [cit. 2014-11-16]. Dostupné z: http://www.softpanorama.org/Commercial_linuxes/Startup_and_shutdown/creating_your_own_init_skripts.shtml

Všechny skripty jsou prováděny přes příkaz *case*, pomocí kterého je vytvořena konstrukce, ve které budou následně vykonány příkazy pro dané stavy skriptu, jako jsou například *start* nebo *stop*. Příkaz *case* je složen z několika prvků a jeho syntaxe vypadá takto.

```
Case proměnná in
Případ_1)příkaz;;
Případ_2)příkaz;;
esac
```

Příkaz *esac* je využit k uzavření celé *case* konstrukce. Příkazy u jednotlivých případů jsou ukončeny středníkem. Pokud se však jedná o jediný nebo poslední příkaz pro daný případ, dojde k jeho uzavření dvěma středníky.

```
root@debian:/etc/init.d# cat mujscript
case "$1" in
start)
     echo "Script je spusten";;
stop)
     echo "Script je zastaven";;
esac
```

V přikladu, který je zde uveden, je jako proměnná uveden znak "*$1*", pomocí kterého je přečten a použit první parametr příkazu. Pokud dojde manuálně ke spuštění skriptu pomocí příkazu *service mujscript start*, bude jako proměnná použit parametr *start* a bude vypsána příslušná *echo* hláška.

```
root@debian:/etc/init.d# service mujscript start
Script je spusten
```

Skript zde uvedený dokáže reagovat pouze na parametr *start* a *stop*. Pokud by byl zadán příkaz *restart* nebo *status*, neproběhla by žádná akce. Aby systém reagoval na další proměnné, je potřeba, aby tyto možnosti byly přidány do *case* podmínky v *Sysvinit* skriptu.

```
root@debian:/etc/init.d# cat mujscript
case "$1" in
restart)
     spusteni;
     ukonceni;;
*)
echo "použijte platny parametr {start|stop}";;
esac
```

V tomto příkladu byl navíc přídán do *case* případů znak ***, který je využit pro všechny ostatní zadané parametry. Když bude zadán příkaz *service mujscript reload*, bude zobrazena *echo* hláška.

```
/etc/init.d# service mujscript reload
pouzijte platny parameter {start|stop|restart}
```

Důležitou věcí pro *Sysvinit* skripty jsou také funkce, pomoci kterých je program rozdělen do částí, ve kterých jsou obsaženy příkazy, které budou pří volání této funkce provedeny. Díky tomu se sada příkazů, která má být provedena během *case* podmínky, nenachází v *case* bloku, a tak může být použita stejná funkce u více *case* možností. Samotná funkce je ve tvaru *název_funkce() { příkaz }* s tím, že se ukončující znak, pravá složená závorka, musí nacházet na nové řádce. Každý příkaz ve funkci je psán na novou řádku. Více příkazů může být zapsáno i do stejné řádky, je však potřeba, aby příkazy byly odděleny středníkem. Tyto funkce jsou poté použity v *case* podmínkách.

```
root@debian:/etc/init.d# cat mujscript
start(){
     echo "Script je spusten"
}
case "$1" in
 start)
     start;;
esac
```

Třetí, a také poslední částí, jsou proměnné, které usnadňují práci v unixových příkazech. Jednotlivé proměnné jsou deklarovány přes *název_proměnné=hodnota_proměnné*. Hodnota proměnné je ve skriptu

volána prostřednictvím *$název_proměnné*. Proměnná *NAME* může být využita například u *mkdir* příkazu.

```
root@debian:/etc/init.d# cat mujscript
NAME=mujscript
case "$1" in
 start)
     mkdir /etc/init.d/$NAME;;
esac
```

V *avahi-deamon* skriptu je využito celkem 5 proměnných.

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin
DESC="Avahi mDNS/DNS-SD Daemon"
NAME ="avahi-daemon"
DAEMON="/usr/sbin/$NAME"
SCRIPTNAME=/etc/init.d/$NAME
```

V *PATH* jsou obsaženy adresáře, ve kterých budou hledány spustitelné programy. Proměnná *DESC* je využita k popisu skriptu. *NAME* obsahuje přesné jméno skriptu a je také volána u některých proměnných. K oddělení těchto adresářů je využita dvojtečka. Proměnná *SCRIPTNAME* má hodnotu */etc/init.d/avahi-daemon*.

V *UNIX* skriptu může být použit také příkaz *set –e*, který ukončí okamžitě *shell* v případě, že příkaz nebyl proveden, nebo pokud je u tohoto příkazu vrácena nenulová výstupní hodnota. Příkazy *if, elif, while* a *until* jsou, v případě nastavení *set –e*, ignorovány. Použití tohoto příkazu by však mělo být pořádně zváženo. Ve vlastně psaných skriptech je doporučeno tento příkaz nepoužívat.

5.1.1 Skript 1 - dbus

Prvním Sysvinit skriptem, který bude rozebrán, je *dbus*, pomocí kterého spolu komunikují určité aplikace. Tento skript bude rozdělen postupně na několik částí, které budou popsány.

Hned na první řádce celého skriptu je obsažen příkaz *#!/bin/sh*. Znakem *#* je v unixových systémech označen komentář, avšak na první řádce spolu se znakem *!* neoznačuje komentář, ale takzvaný *shebang*, který dává systému informaci, jaký interpret má být pro daný skript použit. *Bin/sh* stanovuje, že jako interpret bude použit program *sh*, neboli *Bourne shell*, uložený v adresáři */bin*.

Po této řádce následuje *BEGIN INIT INFO blok*, který je vždy obsažen na začátku skriptu.

```
### BEGIN INIT INFO
# Provides:          dbus
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: D-Bus systemwide message bus
# Description:       D-Bus is a simple interprocess
# messaging system, used for sending messages
# between applications.
### END INIT INFO
```

Tato celá část se nachází za znakem *#*, pomocí kterého jsou zakomentovány jednotlivé řádky.

Provides informuje uživatele, která služba bude spuštěna tímto skriptem, v tomto případě tedy *dbus*. Než je spuštěn nebo zastaven skript, musí být připojeny všechny lokální filesystémy a také musí být v provozu logovací systém. Skript je spuštěn v případě úrovně běhu 2, 3, 4 a 5. Řádka *default-stop* neobsahuje žádné úrovně běhu a tak k jejímu zastavení může dojít pouze manuálně pomocí příkazu *service dbus stop*. *Short-description* a *description* popisují význam D-Bus.

```
# Copyright © 2003 Colin Walters
# Copyright © 2005 Sjoerd Simons
```

Za *INIT INFO* blokem následuje seznam autorů tohoto skriptu, který je také obsažen v komentované řádce.

set -e zajistí, že pokud některý příkaz v *dbus* skriptu bude obsahovat chybu nebo bude mít návratovou hodnotu jinou než 0, dojde k ukončení shellu a skript tak nebude proveden.

Předtím, než jsou vytvořeny funkce pro tento skript, jsou definovány proměnné. Seznam všech proměnných užitých v tomto skriptu je vypsaný zde. Tyto proměnné jsou poté volány v určitých příkazech tohoto skriptu.

```
DAEMON=/usr/bin/dbus-daemon
UUIDGEN=/usr/bin/dbus-uuidgen
UUIDGEN_OPTS=--ensure
NAME=dbus
DAEMONUSER=messagebus
PIDDIR=/var/run/dbus
PIDFILE=$PIDDIR/pid
DESC="system message bus"
```

Proměnné *DAEMON* a *UUIDGEN* reprezentují soubory *dbus-daemon* a *dbus-uuidgen*, které jsou dostupné v adresáři */usr/bin*. *UUIDGEN_OPTS* obsahuje nastavení *--ensure* pro soubor *dbus-uuidgen*. *NAME* obsahuje pouze přesné jméno skriptu. *PIDDIR* obsahuje adresu adresáře ve kterém je uložen *PID* soubor pro *dbus*. V proměnné *PIDFILE* je obsažena hodnota *PIDDIR* a spolu odkazují na soubor *pid*, který je obsažen v adresáři */var/run/dbus*.

```
test -x $DAEMON || exit 0
```

Příkaz *test* slouží k provedení různých testů. Může kontrolovat, zda existuje určitý soubor, zda je spustitelný, ale také může porovnávat různé hodnoty. Nastavení *–x* testuje, zda soubor, reprezentovaný proměnnou *DAEMON*, existuje a jestli je spustitelný. Zde je testován soubor *dbus-daemon* z adresáře */usr/bin*. Tento soubor v distribuci Debian existuje a tak nebude provedena žádná akce. Kdyby však tento soubor neexistoval,

a nebyla by tedy splněna tato podmínka, došlo by k provedení příkazu, který je obsažen za znaky "||", tedy *exit 0*. Pokud by neexistoval soubor *dbus-daemon* nedošlo by k provedení dbus skriptu.

```
. /lib/lsb/init-functions
```

Příkaz *"."* má význam jako *exec*, tedy čte a provádí příkazy z určitého souboru. Oproti *exec* se však nestará o to, kdo má u tohoto souboru která práva, a tak může být proveden i uživatelem, který nemá žádná práva. Tento soubor by měl být volán v každém *Sysvinit* skriptu, protože obsahuje některé funkce a příkazy, které bez tohoto souboru nefungují, například *log_succes_msg* nebo *log_failure_msg*.

```
if [ -e /etc/default/dbus ]; then
  . /etc/default/dbus
fi
```

Dojde k provedení příkazu *./etc/default/dbus*, v případě, že soubor *dbus*, nacházející se v adresáři */etc/default*, existuje.

Ve skriptu jsou dále obsaženy celkem 4 funkce – *create_machineid, start_it_up, shut_it_down* a *reload_it*.

```
create_machineid() {
  if [ -x $UUIDGEN ]; then
    $UUIDGEN $UUIDGEN_OPTS
  fi
}
```

Jak název funkce vypovídá, v případě existence tohoto souboru dojde k provedení příkazu *usr/bin/dbus-uuidgen--ensure*, reprezentovaného proměnnými *UUIDGEN* a *UUIDGEN_OPTS***,** který vytvoří *UUID*. Tento příkaz je již podrobněji popsán v praktické části *Upstart* u *dbus* skriptu.

```
start_it_up()
{
  if [ ! -d $PIDDIR ]; then
    mkdir -p $PIDDIR
```

```
    chown $DAEMONUSER $PIDDIR
    chgrp $DAEMONUSER $PIDDIR
  fi
```

Testuje, zda existuje adresář *var/run/dbus*, reprezentovaný proměnnou *PIDDIR*. V případě neexistence tohoto adresáře dojde k jeho vytvoření. Následně je, pomocí příkazů *chown* a *chgrp*, změněn vlastník a také vlastnická skupina tohoto souboru na hodnotu proměnné *DAEMONUSER,* kterou je *messagebus***.**

```
if ! mountpoint -q /proc/ ; then
     log_failure_msg "Can't start $DESC - /proc
     is not mounted"
     return
fi
```

Příkaz *mountpoint* zkoumá, zda adresář slouží jako přípojný bod, neboli „*mountpoint*“. Pokud by byl proveden pouze příkaz *mountpoint /proc/,* došlo by buď k vypsání */proc/ is a mountpoint nebo /proc/ is not a mountpoint*. Nastavení *–q* toto vypsání blokuje. Pokud je splněna *if* podmínka, tedy že */proc* není připojný bod, dojde k vypsání *fail* hlášky.

```
[FAIL] Can´t start dbus - /proc is not mounted …
failed!

if [ -e $PIDFILE ]; then
  if $0 status > /dev/null ; then
    log_success_msg "$DESC already started; not
starting."
    return
  else
    log_success_msg "Removing stale PID file
$PIDFILE."
    rm -f $PIDFILE
  fi
fi
```

Pokud existuje soubor *pid*, dojde k provedení jedné z podmínek, obsažených ve vnitřní *if* podmínce. Pokud nebude splněna podmínka *$0 status > /dev/null*, dojde k vypsání dané hlášky a následnému smazání

souboru *pid*. *$0* vypisuje aktuální soubor. */dev/null* je speciální soubor, který přijímá různá zapsaná data, například chybové výpisy, bez toho aby je ukládal. Pokud by tento soubor byl přečten, byl by prázdný[37].

```
create_machineid

  log_daemon_msg "Starting $DESC" "$NAME"
start-stop-daemon --start --quiet --pidfile
$PIDFILE \ --exec $DAEMON -- --system $PARAMS
log_end_msg $?
}
```

Je zavolána funkce *create_machineid* a následně je spuštěn *start-stop-daemon* příkaz, který je určen k vytváření a ukončení procesů. Pomocí nastavení *--start* je spuštěn program spolu s argumentem *--quiet*, který omezuje výstupní hlášku, a *--pidfile*, obsahujícího adresu *pid* souboru, který má být zkontrolován. Druhá část *start-stop-daemon* příkazu, *--exec*, kontroluje, zda je proces *dbus-daemon* spustitelný. Navíc je pro něj použit standartní konfigurační soubor. *Log_eng_msg* funkce vypisuje finální hlášku po akci, která se má provést. Ta má buď stav *„failed!“* nebo *„ok“*, podle toho zda soubor *pid* existuje nebo neexistuje.

```
shut_it_down()
{
  log_daemon_msg "Stopping $DESC" "$NAME"
  start-stop-daemon --stop --retry 5 --quiet
 --oknodo --pidfile $PIDFILE \
    --user $DAEMONUSER
  log_end_msg $?
  rm -f $PIDFILE
}
```

Tato část kódu slouží k zastavení skriptu. Pomocí nastavení *--retry* je kontrolováno, zda dané procesy byly zabity. *--user* kontroluje, zda je vlastník souboru *messagebus.* Nakonec je vymazán soubor *var/run/dbus/pid*.

[37] *Soubor /dev/null* [online]. [cit. 2015-03-06]. Dostupné z: http://www.abclinuxu.cz/slovnik/soubor-dev-null

```
reload_it()
{
 create_machineid
 log_action_begin_msg "Reloading $DESC config"
 dbus-send --print-reply --system --type=method_call \
           --dest=org.freedesktop.DBus \
           / org.freedesktop.DBus.ReloadConfig >
/dev/null
 log_action_end_msg $?
}
```

Je zavolána *funkce create_machineid* a poté dojde k provedení *dbus-send* příkazu, který odesílá zprávu pro *d-bus message bus*. Zde jsou zprávy odesílány systémové sběrnici. Přes *--print-reply* je využit blok pro odpověď odeslané zprávě. Tato odpověď je následně zobrazena v čitelné podobě. Zpráva bude obdržena na adresách, určených v parametru *--desc*.

```
case "$1" in
  start)
    start_it_up
  ;;
  stop)
    shut_it_down
  ;;
  reload|force-reload)
    reload_it
  ;;
  restart)
    shut_it_down
    start_it_up
  ;;
  status)
    status_of_proc -p $PIDFILE $DAEMON $NAME && exit 0
|| exit $?
    ;;
  *)
    echo "Usage: /etc/init.d/$NAME
{start|stop|reload|restart|force-reload|status}" >&2
    exit 2
  ;;
esac
```

Case podmínka pro jednotlivé argumenty dbus skriptu jako je *start* nebo *restart*. Příkaz *status_of_proc* zjišťuje aktuální stav dbus procesu.

Tento příkaz navíc funguje pouze v případě, že je zavolán soubor */lib/lsb/init-functions*. V případě zadání příkazu *service dbus* dojde k vypsání všech možnosti, které mohou být použity.

5.1.2 Skript 2 - rsyslog

Pomocí *rsyslog* skriptu jsou logovány zprávy v *UNIX* systému. Autorem samotného skriptu pro *Sysvinit* je Michael Biebl.

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          rsyslog
# Required-Start:    $remote_fs $time
# Required-Stop:     umountnfs $time
# X-Stop-After:      sendsigs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: enhanced syslogd
# Description:       Rsyslog is an enhanced multi-
threaded syslogd.
### END INIT INFO
```

V *BEGIN INIT INFO* bloku jsou obsaženy důležité informace. Oproti *dbus* skriptu zde příkaz *#! /bin/sh* obsahuje mezeru**.** Jeho význam a vlastnosti jsou však totožné s příkazem *#!/bin/sh* (bez mezery) který je využit u *dbus* skriptu. Před samotným spuštěním skriptu by měli být načteny všechny souborové systémy a měl by být nastaven systémový čas. Při zastavení by měly být, stejně jako při spuštění, dostupné všechny souborové systémy a spolu s nimi také *umountfs*. Při úrovních běhu 2,3,4 a 5 by měl být skript spuštěn. Pokud by byl operační systém v úrovni běhu 0,1 a nebo 6, měl by skript být zastaven. *X-Stop-after* poskytuje zpětné závislosti, v případě, že mají daná zařízení také parametry *should-start* a *should-stop*.

```
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="enhanced syslogd"
NAME=rsyslog
RSYSLOGD=rsyslogd
RSYSLOGD_BIN=/usr/sbin/rsyslogd
RSYSLOGD_OPTIONS="-c5"
RSYSLOGD_PIDFILE=/var/run/rsyslogd.pid
```

```
SCRIPTNAME=/etc/init.d/$NAME
```

Proměnná *PATH* obsahuje seznam všech adresářů, ve kterých se budou hledat spustitelné programy. *DESC* a *RSYSLOGD* jsou využity pouze k výpisu stanoveného textu. Zbylé proměnné reprezentují adresu spustitelného balíku pro ***rsyslog,*** který je obsažen v adresáři */usr/sbin*, dále nastavení pro tento soubor a také adresu k souboru s *rsyslogd PID*.

```
[ -x "$RSYSLOGD_BIN" ] || exit 0
```

Příkaz *test* byl již zmíněn u *sysvinit dbus* skriptu. Příkaz *–x* ověří, zda v adresáři */usr/sbin* existuje a může být proveden soubor *rsyslogd*, který je zde nahrazen volanou proměnnou. Pokud tento soubor nebude nalezen, nebo nebude spustitelný, dojde k zastavení provádění příkazu ve skriptu.

```
[ -r /etc/default/$NAME ] && . /etc/default/$NAME
```

Druhý test příkaz obsahuje argument *–r*, který zkoumá, zda může být soubor přečten a jestli vůbec tento soubor existuje. Pokud je splněna podmínka, je proveden příkaz, nacházející se za znaky „*&&*”, tedy *. /etc/default/$NAME*. V případě, že bude existovat soubor *rsyslog*, budou provedeny příkazy uvnitř tohoto souboru.

```
. /lib/lsb/init-functions
```

Tento příkaz by měl existovat v každém psaném skriptu. V *init-functions* souboru se nachází mnoho příkazu a funkcí, které jsou volány právě v *Sysvinit* skriptech. Z *init-functions* souboru jsou v *rsyslog* skriptu využity *log_daemon_msg*, *log_end_msg* a *status_of_proc* funkce, které by, bez načtení tohoto souboru, nefungovali.

```
do_start()
{
     DAEMON="$RSYSLOGD_BIN"
     DAEMON_ARGS="$RSYSLOGD_OPTIONS"
     PIDFILE="$RSYSLOGD_PIDFILE"

     start-stop-daemon --start --quiet --pidfile
$PIDFILE --exec $DAEMON -- $DAEMON_ARGS
}
```

Funkce je volána pří spuštění tohoto skriptu. *Start-stop-daemon* zkoumá, zda je daný soubor spuštěn. Pokud není spuštěn, jsou provedeny příkazy ze souboru *rsyslogd* spolu s argumentem *–c5,* který reprezentuje mód kompatibility. Pro tento příkaz byly vytvořeny nové proměnné, které však pouze obsahují hodnotu dříve vytvořených proměnných a tak by zde nemuseli být ani využity.

```
do_stop()
{
     DAEMON="$RSYSLOGD_BIN"
     PIDFILE="$RSYSLOGD_PIDFILE"

     start-stop-daemon --stop --quiet --
retry=TERM/30/KILL/5 --pidfile $PIDFILE --exec
$DAEMON
}
```

Start-stop-daemon kontroluje, zda existuje systémový proces *rsyslogd*, který je zde reprezentovaný proměnnou *DAEMON*. Pokud by neexistoval, došlo by k ukončení s chybovým statusem. V případě, že bude proces, reprezentovaný proměnou *PIDFILE*, obsažen v tabulce procesů, dojde k odeslání *TERM* signálu, který je specifikovaný v nastavení *--retry*, a zastavení *rsyslogd* procesu. Stejně jako u *do_start* funkce jsou zde volány proměnné *DAEMON* a *PIDFILE*, které však pouze odkazují na již vytvořené proměnné.

```
do_rotate() {
     DAEMON="$RSYSLOGD_BIN"
     PIDFILE="$RSYSLOGD_PIDFILE"

     start-stop-daemon --stop --signal HUP --quiet --
pidfile $PIDFILE --exec $DAEMON
}
```

Význam *start-stop-daemon* příkazu je stejný jako u funkce *do_stop*, popsané výše. Nastavení *--signal* říká systému, který signál má být použit k zastavení procesu, v tomto případě *HUP*. V případě, že soubor, reprezentovaný proměnnou *DAEMON*, je ve stavu *running*, dojde k zabití definovaného procesu. *--quiet* omezuje výpis informačních zpráv na terminál. Na terminál jsou vypsány pouze chybové zprávy.

```
create_xconsole() {
     XCONSOLE=/dev/xconsole
     if [ "$(uname -s)" != "Linux" ]; then
          XCONSOLE=/run/xconsole
          ln -sf $XCONSOLE /dev/xconsole
     fi
     if [ ! -e $XCONSOLE ]; then
          mknod -m 640 $XCONSOLE p
          chown root:adm $XCONSOLE
          [ -x /sbin/restorecon ] && /sbin/restorecon
$XCONSOLE
     fi
}
```

Tato funkce slouží k vytvoření *xconsole* pomocí několika *if* podmínek. Pokud název jádra není roven hodnotě *"Linux"* dojde k nakopírování textu ze souboru */run/xconsole* do souboru *xconsole*, uloženého v adresáři */dev* a k vytvoření symbolického odkazu. Druhá *if* podmínka kontroluje existenci souboru */dev/xconsole*. Pokud tento soubor neexistuje, je pomocí příkazu *mknod* vytvořen, jako speciální soubor typu fronta (FIFO), soubor *xconsole* v adresáři */dev*, spolu s právy, která jsou určena argumentem *–m*. Také dojde ke změně vlastníka a skupiny na *root* a *adm* pro tento vytvořený soubor. Nakonec je pomocí příkazu *–x*

kontrolováno, zda existuje soubor *restorecon*. Na základě toho dojde k provedení příkazu.

```
sendsigs_omit() {
     OMITDIR=/run/sendsigs.omit.d
     mkdir -p $OMITDIR
     ln -sf $RSYSLOGD_PIDFILE $OMITDIR/rsyslog
}
```

Je vytvořen adresář */run/sendsigs.omit.d*, který je definovaný v proměnné *OMITDIR*. Pokud již existuje, je pomocí argumentu *-p* blokována chybová hláška. V adresáři */run/sensigs.omit.d* je vytvořen symbolický odkaz pro soubor *rsyslog*, který obsahuje *PID* hodnotu souboru *rsyslogd.pid*, který se nachází v adresáři */var/run*.

```
case "$1" in
  start)
     log_daemon_msg "Starting $DESC" "$RSYSLOGD"
     create_xconsole
     do_start
     case "$?" in
          0) sendsigs_omit
             log_end_msg 0 ;;
          1) log_progress_msg "already started"
             log_end_msg 0 ;;
          *) log_end_msg 1 ;;
     esac
     ;;
```

Na terminál je vypsána *log_daemon_msg* zpráva, následně jsou zavolány funkce *create_xconsole* a *do_start*. Podle návratové hodnoty jsou, pomocí vnitřní *case* podmínky, provedeny jednotlivé akce. Znak *$?* zkoumá návratovou hodnotu *do_start* funkce. Pokud došlo ke spuštění démona, je návratová hodnota 0, a je zavolána funkce *sendigs_omit* a na terminál vypsána *log_end_msg* tvaru *[ok] Starting enhanced syslogd: rsyslogd*. Pokud bude návratová hodnota jiná než 0 nebo 1, dojde k chybě při spuštění démona, je vypsána *failed* hláška. Jestli je démon již spuštěn, je provedena podmínka definovaná pro *case* hodnotu 1.

```
stop)
   log_daemon_msg "Stopping $DESC" "$RSYSLOGD"
   do_stop
   case "$?" in
        0) log_end_msg 0 ;;
        1) log_progress_msg "already stopped"
           log_end_msg 0 ;;
        *) log_end_msg 1 ;;
   esac
   ;;
```

Stejně jako u možnosti *start* je zde obsažena vnitřní *case* podmínka, která zkoumá návratovou hodnotu *do_stop* funkce. Pokud bude *rsyslog* démon úspěšně zastaven, dojde k provedení příkazů pro možnost 0. Možnost 1 provede příkazy za předpokladu, že je *rsyslog* démon již zastaven. Když dojde během průběhu *do_stop* funkce k chybě, bude návratová hodnota větší než 1 a bude proveden příkaz *log_end_msg 1*, označující chybový výpis.

```
rotate)
   log_daemon_msg "Closing open files" "$RSYSLOGD"
   do_rotate
   log_end_msg $?
   ;;
```

Je provedena *do_rotate* funkce a pomocí *log_end_msg $?* je zkoumána návratová hodnota, která bude poté vypsána spolu s *log_daemon_msg* zprávou na terminál.

```
restart|force-reload)
   $0 stop
   $0 start
   ;;
```

Při možnostech *restart* nebo *force-reload* dojde k provedení příkazů z *case* podmínek *stop a start*.

```
status)
     status_of_proc -p $RSYSLOGD_PIDFILE $RSYSLOGD_BIN
$RSYSLOGD && exit 0 || exit $?
     ;;
  *)
     echo "Usage: $SCRIPTNAME
{start|stop|rotate|restart|force-reload|status}" >&2
     exit 3
     ;;
esac
```

Příkaz *status_of_proc* zjišťuje aktuální stav *rsyslog* démona. Podle toho zda je démon spuštěn nebo zastaven je následně vypsán jeho stav na terminál. V případě zadání jakékoliv jiné hodnoty pro příkaz *service rsyslog* než jsou tyto zmíněné, dojde k vypsání *echo* hlášky se všemi možnostmi, které mohou být pro tento skript použity.

5.1.3 Skript 3 - Odeslání IP adresy na e-mail

Tento skript zjistí IP adresu daného počítače a odešle ji na předem určenou e-mailovou adresu.

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          mail
# Required-Start:    $network
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: Vypis IP adresy na email
# Description:       zjisteni IP adresy PC a nasledne
#                    odeslani na mail.
### END INIT INFO

# Michal Kuchta <kuchta.m@seznam.cz>
```

BEGIN INIT INFO blok obsahuje popis toho skriptu. Skript by měl být spuštěn při úrovních běhu 2, 3, 4 a 5. K jeho zastavení by nemělo dojít v žádné úrovni běhu. Navíc by mělo být dostupná síťová karta počítače předtím, než dojde ke spuštění tohoto skriptu. To je určeno pomocí

Requires-start: $network. Pod tímto blokem je obsažena komentovaná řádka s informací o autorovi tohoto skriptu.

```
odeslani()
{
soubor=/etc/init.d/ipadresa.txt

echo "IP adresa PC je:" > /etc/init.d/ipadresa.txt
ifconfig | grep 'Bcast' | grep –o –P 'addr.{12}' >>
/etc/init.d/ipadresa.txt
mail –s " Pocitac se prihlasil a ma tuto adresu "
kuchta.m@gmail.com < $soubor
}
```

Funkce *odeslani()* slouží ke zjištění IP adresy počítače a následně jejího odeslaní na stanovenou e-mailovou adresu. Na začátku této funkce je obsažena proměnná *soubor*, která reprezentuje soubor *ipadresa.txt*. *echo* řádka slouží k vytvoření souboru *ipadresa.txt* a k vložení daného *echo* textu do tohoto souboru. Znak *">"* vždy přepíše aktuální obsah v tomto souboru, takže soubor nebude obsahovat nadbytečná data.

Hlavní částí je příkaz *ifconfig*, obsahující informace o síťovém rozhraní. Spolu s dvěma příkazy *grep*, které slouží k vyhledání textu, vypíše pouze potřebnou IP adresu. Přes znak *">>"* dojde k uložení této adresy do *ipadresa.txt* souboru. Obsah souboru *ipadresa.txt* bude vypadat následovně:

```
IP adresa PC je :
addr:10.0.2.15
```

K odeslání informace o IP adrese počítače je využit příkaz *mail*. Přes argument *–s* je nastaven předmět zprávy. Zpráva bude odeslána na e-mail *kuchta.m@gmail.com*. Přes znak *"<"* je ke zprávě připojen obsah souboru *ipadresa.txt*, který je zde zavolán přes proměnnou *soubor*.

```
case "$1" in
     start)
          odeslani
          ;;
```

```
     *)
          echo "Usage: /etc/init.d/mail {start}"
          ;;
esac
```

Case podmínka přijímá pouze hodnotu *service mail start*. Pokud je tato podmínka splněna, dojde k zavolání funkce *odeslani*, tedy k odeslání IP adresy počítače na stanovenou e-mailovou adresu. V případě zadání jiného parametru dojde k vypsání *echo* hlášky, která je obsažena v **)* možnosti.

5.1.4 Skript 4 – Výpis skriptů

Přes tento skript je v adresáři */zaloha* vytvořen soubor *soubor.txt*, který obsahuje seznam všech skriptů z adresáře */etc/init.d*. Tento soubor je poté zabalen pomocí příkazu *gzip*. Tento soubor je přístupný pouze uživateli *root*.

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          soubor
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: vypis scriptu
# Description:       script vypise seznam všech
scriptu, ktery nasledne zabali a nastavi mu prava
### END INIT INFO

. /lib/lsb/init-functions

     SOUBOR=/zaloha/soubor.txt
     SCRIPT=/etc/init.d/vypis

vytvoreni_adresare(){
if [ -e /zaloha ]; then
log_failure_msg "adresar /zaloha jiz existuje"
rm –f /zaloha/soubor.gz
else
log_daemon_msg "vytvarim adresar /zaloha"
mkdir /zaloha
log_end_msg $?
fi
}
```

```
vypis_scriptu(){
log_daemon_msg "vypisuji seznam scriptu do $SOUBOR"
ls -l /etc/init.d/ > $SOUBOR
log_end_msg $?
}

zabaleni_souboru(){
log_daemon_msg "zabaluji $SOUBOR"
gzip -c $SOUBOR > /zaloha/soubor.gz
chmod 700 /zaloha/soubor.gz
log_end_msg $?
}

smazani_souboru(){
     log_deamon_msg "smazani $SOUBOR"
     rm -f $SOUBOR
     log_end_msg $?
}

case "$1" in
start)
     vytvoreni_adresare
     vypis_scriptu
     zabaleni_souboru
     smazani_souboru
     ;;
*)
     echo "$SCRIPT:usage{start}"
```

BEGIN INIT INFO sekce obsahuje popis tohoto skriptu spolu s tím, kdy má dojít k automatickému spuštění tohoto skriptu.

Základem tohoto skriptu je příkaz *. /lib/lsb/init-functions*, díky kterému budou fungovat funkce *log_failure_msg*, *log_daemon_msg* a *log_end_msg*.

Celý skript je složen ze 4 funkcí – *vytvorení_adresare*, *vypis_scriptu*, *zabaleni_souboru* a *smazani_souboru*. První funkcí je *vytvoreni_adresare*. Nejprve je pomocí *[-e /zaloha]* kontrolováno, zda existuje adresář */zaloha*. Pokud tento adresář již existuje, dojde k vypsání *fail* hlášky, která je obsažena ve funkci *log_failure_msg*, a k provedení příkazu *rm –f /zaloha/soubor.gz*, který v případě existence tohoto souboru smaže tento

soubor. Pokud adresář */zaloha* neexistuje, dojde k jeho vytvoření. Přes příkaz *log_end_msg $?* je vypsán koncový stav této funkce. *$?* vypisuje návratovou hodnotu funkce.

Funkce *vypis_scriptu* vytvoří soubor, který bude obsahovat seznam skriptů, které jsou obsaženy v adresáři */etc/init.d/*. Tento soubor je vytvořen pomocí příkazu *ls –l*. Proměnná *$SOUBOR* má hodnotu */zaloha/soubor.txt*. Stejně jako u předchozí funkce je zde obsažen příkaz *log_end_msg*, který kontroluje finální stav této funkce.

Jakmile je tento soubor vytvořen, je zavolána funkce *zabaleni_souboru*, pomocí které je přes příkaz *gzip* zabalen tento vytvořený soubor. Tomuto souboru jsou poté přiřazena přístupová práva.

Pokud existuje soubor */zaloha/soubor.txt*, je pomocí funkce *smazani_souboru* následně smazán.

V *case* podmínce je určen seznam funkcí, které budou v případě spuštění *service vypis start* postupně provedeny.

5.2 Upstart

Oproti *Sysvinit* a *Systemd* zavádění jsou v *Upstart Init* skripty známy pod pojmem „*Job*“ (úloha) a jsou rozděleny do dvou částí. „*System Job*“, které je uložen v adresáři */etc/init/*. Druhou částí je „*User Job*“, nacházející se v adresáři *$HOME/.init/*, které je vytvářen běžným uživatelem. Všechny skripty jsou označeny koncovkou *.conf* .[38]

Základním kamenem pro psaní všech *Upstart* úloh jsou tzv. „*stanzy*“, neboli směrnice v konfiguračním souboru, pomocí kterých je psán celý skript. V každém skriptu musí být obsažena minimálně jedna stanza.

[38] HUNT, James a Clint BYRUM. Upstart Intro, Cookbook and Best Practises: Concepts and Terminology. In: [online]. [cit. 2014-11-15]. Dostupné z: http://upstart.ubuntu.com/cookbook/#concepts-and-terminology

Ve většině případů se jedná o stanzu *exec* nebo *script* spolu s *description* stanzou. Seznam všech stanz, které mohou být použity, je dostupný zde[39].

Na začátku každého *Upstart* skriptu by měla být stanza *description,* sloužící pouze jako informace pro uživatele, k čemu daný skript slouží. Spolu s ní může být na začátku skriptu také *author* stanza, obsahující informace o autorovi tohoto skriptu. Tyto dvě stanzy však slouží pouze jako informační zdroj pro daného uživatele a nemají žádné speciální vlastnosti jako ostatní stanzy. Hodnota stanz *description* a *author* je vložena do uvozovek. Job, který neobsahuje *exec*, *script*, *pre-script* a další *script* sekce, je označen jako abstraktní job.

```
Description "script odesilajici email"
author "Michal Kuchta <kuchta.m@seznam.cz>"
```

Mezi dvě hlavní, a také nejpoužívanější, stanzy patří *exec* a *script*, které slouží k definici procesů. Každý psaný script musí vždy obsahovat minimálně jednu z těchto dvou variant. První z nich, *exec*, provede příkaz nacházející se pouze na dané řádce. Za *exec* poté následují klasické příkazy používané v operačních systémech Linux, jako jsou například *chown*, *cp*, *mkdir* nebo *rm*. Příkladem může být příkaz

```
# script pro smazání souboru
exec rm etc/scripty/priklad
```

kdy pomocí příkazu *rm* dojde k vymazání souboru *priklad* z adresáře *etc/scripty*.

Zatímco u první možnosti, *exec*, je vždy proveden příkaz nacházející se jen na dané řádce, u druhé možnosti, *script*, je tomu jinak. Jeden nebo více příkazů, které mají být provedeny, jsou uloženy v bloku, ze kterého jsou postupně provedeny. V tomto bloku mohou být použity všechny možné příkazy systému Linux. Navíc zde může být obsažena i stanza *exec*. Konec

[39] Seznam všech stanz je dostupný na adrese http://upstart.ubuntu.com/cookbook/#stanzas-by-category

tohoto bloku je označen příkazem *end script,* který je obsažený na nové řádce. Stanza *script* může být obsažena v *Upstart* skriptu pouze jednou. Jako příklad může být použit skript

```
# script pro prenos souboru
script
     mkdir /etc/priklad
     cp etc/init/soubor.conf etc/priklad
end script
```

pomocí kterého dojde k vytvoření adresáře *priklad* v adresáři */etc* a následně dojde k nakopírování souboru *soubor.conf* do této složky.

Ve velké míře používané stanzy z této kategorie patří *pre-start script* a *post-stop script*. Strukturou se nijak neliší od stanzy *script*, ale jsou provedeny v jiný okamžik než hlavní část daného jobu (*exec* nebo *script*). První z výše uvedených, *pre-start*, je proveden před hlavním skriptem a může být používán například pro smazání, vytvoření a nastavení přístupových práv daných souborů a adresářů předtím, než dojde k provedení hlavní části scriptu. *Post-stop script* je naopak proveden až po hlavní části daného skriptu a ve většině případů slouží k „úklidu“ po hlavní části skriptu. Například k vymazání určitého souboru nebo odeslání *kill* signálu určitému *PID*. Příkladem je

```
#vypis vsech scriptu z adresare init
pre-start script
     ls -l *.conf>>soubor.txt
end script
script
     cp soubor.txt /etc/scripty/
end script
post-stop script
     rm /etc/init/soubor.txt
end script
```

V *pre-start* sekci budou pomocí příkazu *ls* zapsány všechny skripty do souboru *soubor.txt*. Hlavní část skriptu poté překopíruje tento soubor do adresáře */etc/scripty*. Pomocí *post-stop* sekce bude vymazán soubor

soubor.txt z adresáře */etc/init*. *Post-stop* stanza se v tomto případě stará o vymazání souboru, aby při dalším spuštění tohoto skriptu nedošlo pomocí příkazu *ls* k zapsání do již existujícího souboru. To by způsobilo, že by byl v souboru obsažen vícekrát stejný obsah.

Jako další jsou používány zejména stanzy *start on* a *stop on*. Jednotlivé skripty mohou být spuštěny manuálně přes příkaz *service název_služby start* nebo *start název_služby*. Pomocí těchto dvou stanz dochází ke spuštění nebo zastavení skriptů automaticky na základě různých eventů.

Jak už bylo zmíněno v teoretické části, tyto eventy mají 4 stavy: *starting, started, stopping* a *stopped*. K těmto eventům je poté přiřazena služba, která určuje, kdy má být daný skript spuštěn.

Ve většině případů jsou eventy *started* a *starting* použity spolu se *start on* stanzou. Není to však podmínkou a tak možné je použít i se *stop on* stanzou.

```
Start on started apache
Start on stoping apache
Stop on stopped apache
```

apache v těchto případech označuje název jobu, na kterém je daný skript závislý. To je dáno tím, že syntaxe *start on started apache* je ve skutečnosti zkratka *start on started job=apache*.

Ke spuštění a zastavení skriptů může také docházet i bez použití těchto 4 eventů. Mezi používané možnosti patří spuštění skriptu na základě úrovní běhů, zastavení v případě souběžného stisknutí kláves *control*, *alt* a *delete* nebo použití *mounted*, který je použít v *mounted* scriptech, jako jsou *mounted-var.conf*, *mounted-tmp.conf* nebo *mounted-run.conf*.

```
Start on runlevel [2345]
Start on mounted MOUNTPOINT=/var
Stop on control-alt-delete
```

Pokud by byla v jobu obsažena stanza *start on* nebo *stop on* vícekrát, byl by job automaticky spuštěn až při podmínce poslední stanzy. Kdyby se v daném jobu nacházely dvě *start on* stanzy, *start on job_A* a o řádku níže *start on job_B*, byl by skript automaticky spuštěn až při druhé podmínce. V tomto případě tedy při *job_B*. Tato podmínka může být ověřena v systému pomocí příkazu *initctl show-config název_jobu*.

```
/etc/init# initctl show-config priklad
priklad
  start on job_B
  stop on job_C
```

Používanou stanzou je také *respawn*. Ta zaručuje, že v případě, kdy je *exec* příkaz nebo hlavní *script* příkaz ukončen bez nastavení cílové stavu jobu na *stop*, bude tento job vždy znovu spuštěn. Kromě opětovného spuštění hlavní *script* sekce dojde i ke znovu spuštění *pre-start*, *post-start* a *post-stop* bloků.

K nastavení znovu spuštění může být spolu s *respawn* stanzou použita i *respawn limit* stanza, u které je nastaveno, kolikrát se má stanza *respawn* pokusit o znovuspuštění daného jobu a v jakých časových odstupech. Pokud job do té doby nebude znovu spuštěn, bude poté označen jako neúspěšný a bude ukončen. *Respawn limit* je ve tvaru *respawn limit počet_opakování čekací_čas*. Například

```
respawn
respawn limit 15 10
```

V tomto případě se systém 15x pokusí obnovit daný job, a to vždy po 10 sekundových intervalech.

Expect stanza slouží ke zjištění *PID* (*process ID*) daného jobu. V případě, že by tato stanza nebyla použita v daném skriptu, byl by sledován první *PID* přiřazený *exec* nebo *script* stanzou. Většina unixových

služeb však „démonizuje“. Ve zkratce je pomocí funkce *fork()* vytvořen nový proces, který je přesnou kopií volaného procesu. Původní proces je poté označen jako rodič (*parent process*) zatímco nový proces je známý jako tzv. potomek původní procesu (*child process*). Aby byl zjištěn *PID* daného jobu, potřebuje *Upstart* vědět, kolikrát daný proces zavolá funkci *fork()*. Ve většině případů „*forkuje“* dvakrát. *Upstart* sám o sobě není schopen na tuto otázku odpovědět, protože nedokáže přesně určit, který *PID* je přiřazen rodiči. K určení *PID* tohoto procesu a také ke zjištění, kolikrát daná služba *forkuje*, je použita právě *expect* stanza. Tato stanza se navíc stahuje pouze k *script* a *exec* stanze. Na *pre-start* a *post-start* stanzy nemá žádný vliv.

Expect stanza je rozdělena na dvě možnosti, které mohou být v jobu použity, *expect fork* a *expect daemon*. Ty se liší pouze v tom, kolikrát bude *Upstart* očekávat, že bude zavolána funkce *fork()* pro určený proces. V případě *fork* bude Upstart očekávat zavolání funkce *fork()* právě jednou. Pokud je v jobu obsažena stanza *expect daemon*, bude *Upstart* předpokládat volání funkce *fork()* přesně dvakrát.[40]

Poslední stanza, která bude zmíněna je *task*. Ta spolu s *pre-start*, *respawn* nebo *expect fork* stanzou patří mezi užívanější stanzy v základních *Upstart* skriptech.

Task může být označen jako krátce žijící job. Pokud by v daném skriptu nebyla obsažena tato stanza, eventy, které slouží ke spuštění jobu, by byly spuštěny ve stejném momentě, kdy bude job ve stavu *started*. V případě existence této stanzy bude event, vedoucí ke spuštění zvoleného jobu blokován, dokud se daný job nedostane do stavu *stopped*. Tato stanza je tedy určena pro skript který má být spuštěn, a dokončen, když se stane určitý event.

[40] HUNT, James a Clint BYRUM. Upstart Intro, Cookbook and Best Practises: Expect. [online]. [cit. 2015-01-14]. Dostupné z: http://upstart.ubuntu.com/cookbook/#expect-daemon

```
# priklad
start on started memcached
stop on stopping memcached
respawn
exec /usr/local/bin/priklad
# priprava-pameti
start on starting priklad
task
exec /path/to/priprava-pameti
```

Kdyby zde nebyla použita stanza *task*, job *priklad* by mohl být spuštěn, jakmile provedeme */path/to/priprava-pameti*, což by mohlo být spuštěno předtím, než bude vyrovnávací paměť připravena.

Úplný seznam eventů, které je možné použít, je dostupný přímo v daném systému pomocí příkazu *man upstart-events*. Seznam je dostupný i online[41]. V online verzi však nejsou obsaženy všechny eventy.

Přes příkaz *initctl list* je možné zjistit stav všech jobů.

Znak "*#*" slouží k zakomentování určité řádky.

Skript je spuštěn buď pomocí příkazu *service nazev_jobu start* nebo přes *start nazev_jobu*. Všechny tyto job soubory obsahují příponu *.conf*. Při spuštění je však použit název bez koncovky *.conf*. Pokud chceme spustit *mujscript.conf*, použijeme příkaz *service mujscript start*.

5.2.1 Script 1 - dbus.conf

Prvním Upstart skriptem, který bude popsán, je job *dbus.conf*, pomocí kterého mohou uživatelské aplikace komunikovat s „démony".

[41] http://manpages.ubuntu.com/manpages/utopic/en/man7/upstart-events.7.html

```
description description  "D-Bus system message bus"

start on local-filesystems
stop on deconfiguring-networking

expect fork
respawn

pre-start script
    mkdir -p /var/run/dbus
    chown messagebus:messagebus /var/run/dbus

    exec dbus-uuidgen --ensure
end script

exec dbus-daemon --system --fork

post-start exec kill -USR1 1

post-stop exec rm -f /var/run/dbus/pid
```

V tomto jobu je obsaženo několik „stanz“ – *description*, *start on, stop on, expect fork, respawn, Spre-start, exec, post-start* a *post-stop*.

```
start on local-filesystems
stop on deconfiguring-networking
```

Na začátku skriptu jsou použity stanzy *start on* a *stop on*, spadající do kategorie definice událostí. Řádka *start on* označuje, že se má daný job spustit v případě, že jsou spuštěny všechny lokální souborové systémy. V případě dekonfigurace síťového připojení naopak dojde k zastavení skriptu pomocí *stop on*.

```
expect fork
```

Jak už bylo zmíněno výše, tento řádek slouží k zajištění správného *PID*. Jelikož se jedná o stanzu *expect fork*, bude *Upstart* očekávat volání *fork()* funkce právě jednou.

```
respawn
```

Stanza *respawn* v tomto případě zaručí, že kdykoliv je příkaz *exec* ukončen a nemá nastavený cílový stav na *stop*, dojde ke znovuspuštění jobu.

```
pre-start script
     mkdir –p /var/run/dbus
     chown messagebus:messagebus /var/run/dbus
     exec dbus-uuidgen --ensure
end script
```

Předtím než dojde k provedení hlavního scriptu, bude provedena *pre-start script* sekce. V tomto případě je pomocí příkazu *mkdir,* spolu s přepínačem *–p,* vytvořen adresář */var* a jeho podadresáře */run/dbus*.

Druhou částí *pre-start* sekce je příkaz *chown*. Tento příkaz pochází z anglického slova *change owner*. Jak už název napovídá, pomocí tohoto příkazu je možné změnit vlastníka daného souboru. Příkaz *chown* je zde ve formátu *chown vlastník:skupina* a *soubor,* kterému budou pozměněna práva. U kořenového adresáře */var* a jeho podadresářů */run/dbus* dojde ke změně vlastníka a skupiny na *messagebus*.

Poslední stanzou v *pre-start* sekci je příkaz *exec* neboli *execute command*, která slouží k provedení daného příkazu. *dbus-uuidgen* je služba, díky které jsou vytvářeny nebo čteny *UUID*[42] (*Universal Unique Identifier*). Samotný příkaz *dbus-uuidgen --ensure* zajistí, že existuje adresář */var/lib/dbus/machine-id*, obsahující dané *UUID*. Navíc nedojde, minimálně do restartu systému, k přepsání stávajícího *UUID* pro daný stroj[43].

[42] UUID- jedná se o 128-bitové číslo sloužící k jednoznačné identifikaci objektu v systému
[43] Dbus-uuidgen. [online]. [cit. 2014-11-27]. Dostupné z: http://dbus.freedesktop.org/doc/dbus-uuidgen.1.html

```
exec dbus-daemon --system --fork
```

Hlavní částí jobu je stanza e*xec*, která bude vykonána po *pre-script* sekci. Pomocí *exec* stanzy dojde ke spuštění aplikace *dbus-daemon*, skrze kterou je více programů schopno mezi sebou komunikovat. U této částí se navíc nachází přídavné nastavení *--system* a *--fork*. První z nich zajistí, že bude použitý standardní konfigurační soubor pro *message bus*. Druhé nastavení, *--fork,* zajistí, že pří provedení tohoto procesu dojde také k vytvoření přesné kopie d-bus démona, který bude označen jako potomek.

```
post-start exec kill -USR1 1
```

Předposlední částí tohoto skriptu je *post-start* sekce, ve které je obsažen příkaz *kill*. Ten je určen k odeslání signálu stanovenému *PID* a je ve tvaru *kill -signal PID*. Signály, které mohou být použity, jsou dostupné v systému pomocí příkazu *man 7 signal*. Ve většině případu je tento příkaz použit pro ukončení určitého procesu. Pokud by byl odeslán například příkaz *kill 377*, tedy příkaz bez nastavení *-signal*, byl by použit defaultní signál *SIGTERM*, který by ukončil proces s daným *PID*. V tomto případě náleží *PID 377 dbus* procesu. Zde je použit uživatelsky definovaný signál *USR1* spolu s *PID 1*, kterým je označen *INIT*.

```
post-stop exec rm –f /var/run/dbus/pid
```

Poslední řádka skriptu pouze zajistí, že pomocí příkazu *rm* dojde ke smazání souboru *pid* v adresáři */var/run/dbus*. Příkaz *rm* obsahuje přídavné nastavení *–f*. Pokud by soubor *pid* neexistoval, byl by příkaz pro smazání stejně proveden. O případném nesmazání daného souboru uživatel nebude informován. Tato *post-stop* část bude provedena poté, co je hlavní část jobu ukončena. Také může být spuštěna v případě, že job nedokázal spustit hlavní proces nebo pokud *pre-start* sekce nebyla úspěšná.

5.2.2 Script 2 - rsyslog.conf

Druhým příkladem *Upstart* skriptu je job *rsyslog.conf*, který slouží jako náhrada za syslog k logování zpráv z aplikací v unixových systémech.

```
description    "system logging daemon"

start on filesystem
stop on runlevel [06]

expect fork
respawn

pre-start script
    /lib/init/apparmor-profile-load usr.sbin.rsyslogd
end script

script
    . /etc/default/rsyslog
     exec rsyslogd $RSYSLOGD_OPTIONS
end script
```

V rsyslog jobu je použito dohromady 9 stanz – *description, start on, stop on, expect fork, respawn, pre-start script, script* a v poslední řadě *exec*, který je obsažen ve stanze *script*.

Stanza *description* pouze informuje uživatele, že tento job slouží ke sledování aplikaci v *UNIX* systému.

```
Start on filesystem
Stop on runlevel [06]
```

job bude automaticky spuštěn poté, co jsou připojeny všechny souborové systémy. Zastaven bude v případě změny úrovně běhu na *0* nebo *6*. Nule je v Ubuntu přiřazeno vypnutí systému, zatímco číslem *6* je označen jeho restart. Dále používané úrovně běhu v *UNIX* systému jsou například 1 a 2. První z nich označuje jednouživatelský mód. *Runlevel 2* je v Ubuntu nastaven jako defaultní a slouží ke grafickému víceuživatelskému režimu.

Pomocí *expect fork* bude Upstart, stejně jako v předchozím příkladu očekávat, že dojde k zavolání funkce *fork()* právě jednou. Tím bude zajištěno správné *PID* pro tento job.

Pokud by během provádění jobu došlo k nějaké chybě, je pomocí stanzy *respawn* zajištěno, že bude docházet k jeho opětovnému spuštění, dokud nebude řádně ukončen.

```
pre-start script
/lib/init/apparmor-profile-load usr.sbin.rsyslog
end script
```

Předtím než bude provedena hlavní část skriptu, dojde k vykonání *pre-start script* sekce, která je zde založena na modulu *Apparmor*. *Apparmor* je bezpečnostní modul, použitý k vylepšení jádra *UNIX* systému, pomocí kterého mohou být programy omezeny jen na určité zdroje. Toto omezení je provedeno pomocí profilů, nacházejících se v adresáři */etc/apparmor*, které obsahují potřebný soubor zásad[44]. Názvy profilů jsou odděleny *"."* a odkazují na soubory obsažené v adresáři */usr/sbin*. Profil *usr.sbin.firefox* by označoval soubor *fixefox* z adresáře */usr/sbin*.

V tomto případě je *rsyslog* omezená aplikace. Jelikož má tato aplikace i svůj *Upstart* job, je třeba, aby tento job načetl soubor s danými zásadami. Toho je dosaženo právě pomocí příkazu */lib/init/apparmor-profile-load*. Druhá část tohoto příkazu, *usr.sbin.rsyslog*, označuje název souboru zásad, který má být načten. *usr.sbin.rsyslog* je profil pro *rsyslog* soubor, nacházející se v adresáři *usr/sbin/rsyslog*.

Nakonec bude provedena hlavní *script* sekce, která je v tomto jobu složena ze dvou řádek

```
. /etc/default/rsyslog
exec rsyslogd $RSYSLOGD_OPTIONS
```

[44] Apparmor. [online]. [cit. 2015-01-23]. Dostupné z: http://manpages.ubuntu.com/manpages/precise/en/man7/apparmor.7.html

Příkaz *“.“* má podobný význam jako příkaz *exec*. Pomocí tohoto příkazu je přečten a vykonán příkaz z určitého souboru. Oproti *exec* se však v určitých podmínkách liší. Zatímco *exec* potřebuje k přečtení a provedení příkazu určitého souboru přístupová práva, příkaz *”.”* je schopen přečíst soubor a provést příkaz v něm obsažený bez toho, aby k tomuto souboru měl daná přístupová práva. Dále musí být rozlišeny dvě různé varianty.

. /etc/default/rsyslog a *./etc/default/rsyslog*. Kdyby byla v tomto skriptu obsažena druhá možnost, *./etc/default/rsyslog* (bez mezery), nebyl by příkaz proveden. Tento příkaz je schopen vykonat soubor pouze v případě, že je obsažen v podřazených adresářích. K provedení příkazu *./etc/default/rsyslog* by tak došlo pouze v případě, že by současný adresář byl *root@majkl-VirtualBox:/#.* V případě, že byl současný adresář jiný, byla by zobrazena hláška *bash: ./etc/init/z: No such file or directory*.

Díky příkazu *. /etc/default/rsyslog* však může být přečten i soubor, který není obsažen přímo v podřazeném adresáři. K jeho vykonání by tak došlo i v případě, že by aktuální adresář byl například */etc#.*

Pomocí první řádky je tedy proveden příkaz nacházející se v daném souboru *rsyslog*. V tomto případě je v souboru obsažen příkaz *RSYSLOGD_OPTIONS="-c5",* kdy znak *-c* označuje režim kompatibility. Znakem *-c5* je označen nativní mód pro *rsyslog*.

5.2.3 Script 3 - Odeslání IP adresy na e-mail

Třetím skriptem, který bude rozebrán je *mail.conf*, který slouží k odeslání IP adresy připojeného počítače na určenou e-mailovou adresu.

```
description    "vypis IP adresy na email"
author "Michal Kuchta <kill3rnio@google.com>"

start on runlevel [12345]

pre-start script
```

```
     echo "IP adresa PC je : " >
/etc/init/ipadresa.txt
      ifconfig | grep 'Bcast' | grep -o -P 'addr.{16}'
>> /etc/init/ipadresa.txt
end script

script
     mail -s "Pocitac se prihlasil a ma tuto adresu"
kill3rnio@gmail.com < /etc/init/ipadresa.txt
end script

post-stop script
     rm -f /etc/init/ipadresa.txt
end script
```

Hlavní částí tohoto vytvořeného skriptu jsou 4 stanzy – *start on, pre-start script, script* a *post-stop script*. Pomocí první z nich, *start on*, je dosaženo toho, aby byl skript spuštěn při všech úrovních běhu kromě úrovní běhu 0, která označuje ukončení systému, a úrovně 6, kterou je označen jeho restart.

Základem tohoto skriptu je *pre-start script* sekce, díky které dojde ke zjištění IP adresy daného počítače. Pomocí první řádky je vložen daný *echo* text do souboru *ipadresa.txt*, který je uložen v adresáři */etc/init*. Druhá část této sekce je složena z příkazů *ifconfig* a *grep*. Příkaz *ifconfig* obsahuje informace o síťovém rozhraní. V samotném *ifconfig* příkazu je však obsaženo mnoho informací a je tak potřeba, aby byl tento výstup omezen. Toho je docíleno použitím 2x příkazu *grep*, sloužícího k vyhledávání textu, pomocí kterého bude zobrazena jen potřebná informace, tedy IP adresa počítače. V případě, že by byl použit jenom první *grep* příkaz, vypadal by výstup takto.

```
etc/init# ifconfig | grep 'Bcast'
inet addr:192.168.1.106  Bcast:192.168.1.255
Mask:255.255.255.0
```

Zobrazena má však být pouze IP adresa a je tak potřeba tento výstup upravit. K tomu slouží druhý *grep* příkaz, obsahující přepínače *–o* a *–P*.

První zmíněný přepínač, *-o*, zobrazí pouze vyhledávaný text, který je v tomto případě *addr*. Pomocí přepínače *–P* je vypsán text *addr* a k němu dalších 16 znaků, které v tomto případě zobrazí pouze požadovanou IP adresu. Výsledek poté bude vypadat takto.

```
ifconfig | grep 'Bcast' | grep -o -P 'addr.{16}'
addr:192.168.1.106
```

Tento výstup je poté, pomocí znaku *"**>>**"*, přidán do souboru *ipadresa.txt*.

Ve stanze *script* je obsažen příkaz *mail*, pomocí kterého dojde k odeslání výše zmíněné IP adresy na stanovený mail. Operátor *„<"* zkopíruje obsah souboru *ipadresa.txt* do tohoto emailu. Nastavení *mail –s* slouží k vložení textu do předmětu zprávy.

Přes *post-stop* sekci dojde, v okamžiku ukončení skriptu, ke smazání souboru *ipadresa.txt* z adresáře */etc/init*.

5.2.4 Script 4 – Výpis všech scriptů

Skript vypíše seznam všech skriptů, které jsou obsaženy v adresáři */etc/init/* do příslušného textového souboru. Tento soubor je následně zabalen a jsou mu nastavená příslušná práva.

```
description    "script pro vypis všech scriptu "
author "Michal Kuchta <kuhta.m@google.com>"

start on runlevel [2345]

pre-start script
     mkdir -p /zaloha
     rm –f /zaloha/soubor.gz
end script

script
     ls -l /etc/init/*.conf > /zaloha/soubor.txt
     gzip -c /zaloha/soubor.txt > /zaloha/soubor.gz
     chmod 700 /zaloha/soubor.gz
end script
```

```
post-stop script
     rm /zaloha/soubor.txt
end script
```

Skript bude automaticky spuštěn při úrovních běhu 2, 3, 4 a 5. *Pre-start* sekce slouží k přípravě prostředí pro daný skript. Nejprve dojde k vytvoření adresáře */zaloha*, do kterého bude uložen soubor s obsaženými skripty. Pokud tento adresář již existuje, je pomocí parametru *–p* blokována chybová hláška. Druhý příkaz, v případě existence souboru *soubor.gz*, smaže tento soubor. Pokud tento soubor nebude existovat, bude přes příkaz *rm –f* ignorován.

Hlavní sekce *script* vytvoří příslušný soubor, následně ho zabalí a nastaví mu daná práva. Soubor s výpisem všech skriptů je vytvořen pomocí příkazu *ls -l /etc/init/*.conf > /zaloha/soubor.txt*. K zabalení je využit příkaz *gzip -c*, který je již obsažený v *UNIX* systému. Parametr *–c* nechá originální soubor beze změny. Práva pro tento zabalený soubor jsou nastavena přes příkaz *chmod 700*, který všem uživatelům, kromě vlastníka souboru, zablokuje veškerý přístup k tomuto souboru.

Post-stop script sekce pouze vymaže soubor, vytvořený přes příkaz *ls,* kterým je soubor *soubor.txt*.

5.3 Systemd

V *Systemd* zaváděních nejsou použity klasické *init* skripty, ale tzv. *.service* jednotky, které jsou jejich přímou náhradou.

Každá *.service* jednotka je složena dohromady ze tří sekcí – [*Unit*], obsahující všeobecné informace o jednotce, [*Service*], obsahující informace a službě a jejich procesech, a [*Install*], který nese instalační volby o jednotce. V jednotlivých sekcích jsou poté obsaženy jednotlivé příkazy, které jsou ve tvaru *argument=hodnota*. Každý jeden příkaz se nachází na vlastním řádku. Příkazy navíc nejsou ukončeny žádným speciálním znakem.

V případě, že je příkaz moc dlouhý a nevejde se na řádku, je oddělen znakem *"\"*. Pokud řádek začíná znakem *"#"* je zcela ignorován[45].

Argumentů pro sekci *[Unit]* je hned několik a patří mezi ně například *Description*, *Wants*, *After* nebo *Before*.

Volba *Description* slouží k popisu jednotky a měla by obsahovat pouze krátkou informaci pro uživatele o této službě.

Documentation obsahuje adresy k dokumentaci jednotky. Podporovány jsou adresy typu *"http://"*, *"https://"*, *"file:"*, *"info:"* a *"man:"*.

Mezi používané argumenty patří i *Requires*. Ten obsahuje název jednotky, která bude aktivována ve stejný okamžik s touto jednotkou. V případě, že se jednotka, uvedená v hodnotě *Requires*, neaktivuje určitou chybou nebo dojde k jejímu zastavení, dojde také k ukončení hlavní jednotky. Obdobou volby *Requires* je možnost *Wants*, která, stejně jako volba *Requires*, obsahuje název jednotky, která bude spuštěna. Neúspěšné spuštění této jednotky však nemá žádný vliv na hlavní jednotku, a tak nedojde k jejímu zastavení. Jednotky, určené v parametru *Requires* a *Wants*, nemusí být spuštěny přesně ve stejný okamžik jako hlavní jednotka.

To je docíleno použitím parametrů *After* a *Before*, které určují pořadí spuštění jednotek. Pokud jednotka *bluetooth.service* bude obsahovat nastavení *After=syslog.service*, bude jednotka *bluetooth.service* spuštěna poté, co dojde ke spuštění jednotky *syslog.service*. V případě použití možnosti *Before* dojde ke spuštění hlavní jednotky těsně před jednotkou, která je uvedena v hodnotě *Before*. Více jednotek může být obsaženo v těchto volbách, je však potřeba, aby byly tyto hodnoty odděleny mezerou.

[45] VYSKOČIL, Michal. Systemd – jednotky a závislosti: Stručný přehled syntaxe. [online]. [cit. 2015-03-07]. Dostupné z: http://www.abclinuxu.cz/clanky/systemd-jednotky-a-zavislosti

Aby nebylo více jednotek spuštěno společně, je možné využít parametru *Conflicts*, který v případě startu první jednotky zastaví druhou jednotku a naopak.

OnFailure obsahuje název jednotky, která v případě, že hlavní jednotka bude ve stavu *„failed“*, bude aktivována.

Argumenty *RefuseManualStart* a *RefuseManualStop* obsahují buď hodnotu *True* nebo *False*. Pokud je hodnota nastavená na *True*, je možné aktivovat nebo deaktivovat danou jednotku pouze skrze závislost na jiné jednotce. Defaultně je nastaven stav *False*. [*Unit*] sekce může vypadat následovně:

```
[Unit]
Description=Bluetooth service
Documentation:man:bluetoothd(8)
Requires=syslog.service sluzba.service
After=syslog.service
```

Druhou sekcí Systemd jednotek je [*Service*], obsahující například parametry *Type*, *ExecStart* nebo *ExecStop*.

Nejdůležitější částí této sekce je *Type*, který definuje typ služby pro danou jednotku. Může obsahovat jednu z 6 možností – *simple*, *oneshot*, *forking*, *dbus*, *notify* nebo *idle*.

Pokud má *Type* hodnotu *Simple*, je jako hlavní proces služby proces obsažený v parametru *ExecStart*. Možnost o*neshot* je obdobou možnosti *simple*, s tím, že závislá jednotka je spuštěna až poté, co je daný proces ukončen.

V případě nastavení hodnoty na *forking* bude systém očekávat, že bude *ExecStart* volat při spuštění také *fork()*. Rodičovský proces bude ukončen, jakmile je spuštění kompletní a až budou všechny komunikační kanály nastaveny. *Child* proces poté bude pokračovat jako hlavní

démonizovaný proces. Pokud je nastavena hodnota na *forking*, měl by být obsažen v sekci *[Service]* také argument *PIDFile*, ve kterém je obsažena úplná cesta k *PID* souboru hlavního procesu tohoto démona. Tento soubor je následně přečten v okamžiku načtení služby.

Argument *Type* může mít také hodnotu *dbus*, u které se očekává, že démon získá jméno na D-Bus sběrnici tak, jak je určeno v argumentu *BusName*. Závislé jednotky jsou spuštěny poté, co je získáno jméno D-Bus sběrnice. Jednotka obsahující toto nastavení navíc implicitně získá závislost na jednotce *dbus.socket*.

Pokud je nastavena hodnota na *notify*, budou závislé služby spuštěny poté, co démon odešle notifikační zprávu skrze zavolání *sd_notify(3)*.

Zbývající možnost, *idle*, provede službu až po odeslání všech úloh.

BusName obsahuje jméno D-Bus sběrnice, díky které je daná služba dostupná. Pokud je tento argument obsažen v *[Service]* sekci, má parametr *Type* hodnotu *dbus*.

V sekci [*Service*] mohou být využity také parametry určené k provedení příkazů. Těmi mohou být *ExecStart*, *ExecStop* nebo *ExecStartPre*.

ExecStart obsahuje příkaz, spolu s jeho argumenty, který bude proveden v okamžiku spuštění jednotky. Pokud je nastaven parametr *Type=oneshot*, může být tento příkaz použit v dané jednotce vícekrát, ale také nemusí být využit vůbec. Jednotlivé parametry jsou provedeny postupně. *ExecStart* musí být použit v případě, že má parametr *Type* hodnotu jinou než *oneshot*, ale musí obsahovat právě jeden příkaz, který má být proveden.

ExecStartPre a *ExecStartPost* jsou obdobou parametru *ExecStart*. Liší se však v době provádění příkazu. Zatímco *ExecStartPre* provádí příkaz před

hlavním *ExecStart*, *ExecStartPost* provádí příkaz až po něm. Oproti *ExecStart* mohou být tyto parametry využity vícekrát, nezávisle na hodnotě parametru *Type*.

ExecStop obsahuje příkazy, které jsou provedeny pro zastavení běžící jednotky. Pokud příkazy, obsažené v *ExecStop*, probíhají, budou ukončeny pomocí parametru *KillMode*. Tento parametr je volitelný.

Parametr *Restart* určuje, kdy by měla být restartována služba, pokud je proces jednotky ukončen nebo zabit. Využívané hodnoty jsou *no*, *always*, *on-succes* nebo například *failure*. Defaultně je nastavena hodnota na *no*, kdy při zastavení služby nikdy nedojde k restartu služby. V případě nastavení na *always* bude služba restartována vždy. Při *on-succes* dojde k restartu služby pouze v případě, že je služba dobře ukončena a je tedy její návratová hodnota rovná 0. Nastavení *on-failure* je opakem *on-succes*. K restartu služby dojde v případě, že návratová hodnota nemá hodnotu 0 nebo pokud byla služba ukončena pomocí některého ze signálů.

Poslední částí Systemd jednotek je *[Install]* sekce, která je využita pouze pro nástroj *systemctl* při instalaci jednotky. Navíc se všechny parametry této jednotky vztahují jen k příkazům *systemctl enable* a *systemctl disable*. Může obsahovat parametry *Alias*, *WantedBy*, *RequiredBy* a *Also*.

V *Alias* jsou obsaženy jména, pod kterými by měla být jednotka také nainstalována. Tyto jména musí mít stejný sufix jako hlavní soubor jednotky. Jakmile je zadán příkaz *systemctl enable*, dojde k vytvoření symbolických odkazů. Tyto odkazy jsou odstraněny po zadání příkazu *systemctl disable*.

WantedBy obsahuje jména jednotek, pro které budou pří instalaci jednotky pomocí příkazu *systemctl enable* vytvořeny symbolické odkazy

v adresáři *.wants/*. V případě spuštění jednotky, uvedené v parametru *WantedBy*, dojde ke spuštění hlavní jednotky.

RequiresBy má podobný význam jako *WantedBy* s tím, že jsou symbolické odkazy vytvořeny v adresáři *.requires/*. Navíc je přidána závislost z *Wants* nebo *Requires* z uvedené jednotky do současné jednotky.

Ostatní jednotky mohou být instalovány nebo odinstalovány spolu s instalací hlavní jednotky, a to pomocí parametru *Also*. Pokud je zadán příkaz *systemctl enable*, dojde spolu s instalací hlavní jednotky i k instalaci jednotky, obsažené v *Also*. K odinstalaci této jednotky dojde po příkazu *systemctl disable*.

.service skript je ovládán pomocí příkazu *systemctl argument nazev_jednotky*. Jako argument může být použit *status*, *reload, restart*, *start* a *stop*. Pokud má být skript spuštěn při startu systému, musí být použit příkaz *systemctl enable nazev_jednotky*. Přes *systemctl disable nazev_jednotk*y je při startu systému služba vypnuta.

Vlastně psaný Systemd *.service* skript by měl být nastavena přístupová práva. Aby byl skript nalezen je potřeba provést příkaz *systemctl daemon-reload*. Tento příkaz musí být proveden i v případě, že došlo ke změně v *.service* skriptu.

5.3.1 Script 1 - dbus.service

dbus.service skript slouží, stejně jako u *Sysvinit* skriptu a *Upstart* jobu, ke komunikaci aplikací s „démony“.

```
[Unit]
Description=D-Bus System Message Bus
Requires=dbus.target
After=syslog.target

[Service]
ExecStart=/bin/dbus-daemon --system --address=systemd:
--nofork --nopidfile --systemd-activation
ExecReload=/bin/dbus-send --print-reply --system --
type=method_call --dest=org.freedesktop.Dbus /
org.freedesktop.Dbus.ReloadConfig
OOMScoreAdjust=-900
```

V *[Unit]* sekci jsou obsaženy celkem 3 položky – *Description*, *Requires* a *After*. *Description* slouží pouze jako krátká informace o tomto skriptu. Spolu s touto jednotkou bude aktivován také *dbus.target*, který je obsažen v parametru *Requires*. Jednotka navíc bude spuštěna až po *syslog.target*.

[Service] sekce obsahuje *ExecStart* a *ExecReload*. Pomocí *ExecStart* je spuštěn *dbus-daemon* s tím, že je použit standartní konfigurační soubor pro tuto sběrnici. Přes parametr *--adress* je nastavena adresa na které se bude naslouchat. Tato adresa následně přepíše původní adresu, která je nastavena v konfiguračním souboru. Pomocí *--nofork* je tato sběrnice nucena *neforkovat* i v případě, že je v konfiguračním souboru nastaveno, že by měla *forkovat*. Dbus-daemon také nebude zapisovat *PID* soubor i kdyby byl nakonfigurován v daném konfiguračním souboru. *--systemd-activation* povolí aktivaci *systemd-style* jednotky. Tato jednotka je užitečná pouze pro *Systemd* spolu se správcem relací v Linuxu.

Pomocí parametru *ExecReload* dojde k provedení příkazu a znovunačtení jeho konfigurace v jednotce. Zde je proveden příkaz *dbus-send*, který slouží k posílání zpráv na sběrnici. V tomto případě odesílá zprávy systémové sběrnici. Nastavení *--print-reply* slouží k využití bloku pro odpověď odeslané zprávě. Tato odpověď je poté zobrazena

v čitelné podobě. *desc=* určuje názvy připojení, na kterých bude zpráva obdržena.

Řádek *OOMScoreAdjust* určuje hodnotu pro zabití běžícího procesu v případě nedostatku paměti. Tato hodnota je v rozmezí od -1000 do 1000, kde první zmíněná hodnota zablokuje toto nastavení. Čím blíže se nastavená hodnota blíží 1000, tím je větší šance, že v případě nedostatku paměti dojde k zabití tohoto procesu. Zde má tento parametr hodnotu -900, takže šance, že doje k zabití procesu, je velmi nízká.

5.3.2 Script 2 – rsyslog

Rsyslog.service skript slouží k logování zpráv v *UNIX* systému.

```
[Unit]
Description=System Logging Service
;Requires=syslog.socket

[Service]
Type=notify
EnvironmentFile=-/etc/sysconfig/rsyslog
ExecStart=/sbin/rsyslogd -n $SYSLOGD_OPTIONS
StandardOutput=null

[Install]
WantedBy=multi-user.target
;Alias=syslog.service
```

Description v *[Unit]* sekci krátce popisuje tento *rsyslog* skript. Druhý parametr v této sekci, *require*s, začíná znakem *";"*, který má stejný význam jako znak *#*. Tento řádek bude systém ignorovat.

Pomocí *Type=notify* bude systém očekávat, že démon, v případě spuštění jednotky, odešle notifikační zprávu pomocí *sd_notify(3)*. Parametr *EnvironmentFile* čte proměnné z textového souboru. Zde je přečten soubor *rsyslog*, který je obsažen v adresáři */etc/sysconfig*. Znak *"-"*, obsažený před adresou textového souboru bude, v případě neexistence tohoto souboru, blokovat výpis chybové hlášky. V souboru *rsyslog* je obsažena proměnná

SYSLOGD_OPTIONS=" ". Přes *ExecStar*t dojde k provedení souboru *rsyslogd*. Klasicky běží *rsyslog* v pozadí po startu systému. Přes parametr *–n* dojde k zablokování automatického běhu tohoto démona v pozadí.

StandartOutput kontroluje, kam je připojen *file descriptor 1 (STDOUT)* prováděného procesu. Pokud chceme otevřít určitý soubor, systém vytvoří položku reprezentující tento soubor. Touto položkou je *file descriptor 1* neboli *standart output*. Pro každý otevřený soubor je vytvořen vždy další *descriptor*. Možnost *null* připojuje standartní výstup na */dev/null*. Kdykoliv bude něco zapsáno do tohoto souboru, dojde k jejímu smazání.

Při použití příkazu *systemctl enable* dojde k vytvoření symbolického odkazu pro *multi-user.target* v adresáři *.wants/*. Pokud bude spuštěn *multi-user.target*, dojde ke spuštění *rsyslog* jednotky. Parametr *Alias* je zde obsažen za znakem *";"* a bude tedy ignorován.

5.3.4 Script 4 – Odeslání IP adresy na e-mail

mail.service jednotka slouží ke zjištění IP adresy přihlášeného počítače.

```
#mail.service

[Unit]
Description= Vypis IP adresy na email

[Service]
Type=oneshot
ExecStart=/bin/bash /etc/systemd/system/mail.sh
ExecStartPost=/usr/bin/rm –f /ipadresa.txt
```

ExecStart parametr provede příkazy, které jsou obsaženy v souboru *mail.sh*. Tento soubor zjistí IP adresu přihlášeného počítače, kterou následně odešle na určenou e-mailovou adresu. Jakmile je odeslán e-mail s IP adresou je pomocí *ExecStartPost* vymazán soubor *ipadresa.txt*.

Hlavní částí této *.service* jednotky je soubor *mail.sh*, který obsahuje skript ke zjištění IP adresy a jejího odeslání na danou e-mailovou adresu. Tvar tohoto skriptu se nijak neliší od *Sysvinit* skriptu. Na začátku souboru je, stejně jako u *Sysvinit* skriptů, obsažen tzv. *shebang*, který určuje použitý interpret pro tento skript.

```
#mail.sh

#! /bin/sh
echo "IP adresa PC je:" > /ipadresa.txt
ifconfig | grep 'broad' | grep -o -P 'inet.{16}' >>
/ipadresa.txt
mutt -s "Pocitac se prihlasil a ma tuto adresu"
kuchta.m@gmail.com < /ipadresa.txt
```

Soubor *ipadresa.txt*, do kterého bude uložena IP adresa počítače, je vytvořen pomocí *echo* příkazu. Do tohoto souboru je nejprve uložen obsah *echo* příkazu. Nejdůležitější částí v tomto scriptu je příkaz *ifconfig*, přes který je zjištěna informace o IP adrese připojeného počítače. Samotný výpis pomocí *ifconfig* však obsahuje spoustu informací. Aby byla vypsána jenom IP adresa je potřeba tento výpis zredukovat. To je dosaženo pomocí dvou *grep* příkazů, které slouží k vyhledání textu. První *grep* příkaz vyhledá a vypíše IP adresu spolu s její maskou. Druhý *grep* příkaz slouží k omezení výpisu prvního *grep* příkazu. Příkaz *grep -o -P 'inet.{16}'* vypíše pouze text *inet* spolu s dalšími 16 následujícími znaky, které vypíší jenom samotnou IP adresu. Tato adresa je následně uložena do již vytvořeného *ipadresa.txt* souboru.

E-mail s IP adresou počítače je odeslán na e-mailovou adresu přes příkaz *mutt*. Atribut *–s* určuje předmět e-mailové zprávy. Pomocí příkazu *</* je do těla e-mailové zprávy přidán obsah souboru *ipadresa.txt*.

5.3.4 Script 4 - Výpis všech scriptů

Stejně jako u čtvrtého *Upstart* a *Sysvinit* scriptu slouží tato *.service* jednotka (script) k vypsání všech *.service* souborů, které jsou obsaženy

v adresáři */usr/lib/systemd/system*. Soubor bude vytvořen v adresáři */zaloha*, kde bude následně zabalen pomocí *gzip* příkazu. Tomuto souboru budou následně nastavena práva pro přístup, čtení a zápis.

```
[Unit]
Description= script pro vypis vsech scriptu

[Service]
Type=oneshot
ExecStartPre=/usr/bin/mkdir –p /zaloha
ExecStartPre=/usr/bin/rm –f /zaloha/soubor.gz
ExecStart=/bin/bash /etc/systemd/system/vypis.sh
ExecStartPost=/usr/bin/chmod 700 /zaloha/soubor.gz
ExecStop=/usr/bin/rm –f /zaloha/soubor.txt
```

[Unit] sekce obsahuje pouze parametr *Description*, který informuje uživatele o vlastnostech tohoto scriptu.

Druhou a zároveň nejdůležitější sekcí je *[Service]*, která obsahuje důležité příkazy pro provedení tohoto scriptu. *ExecStartPre* parametry slouží k přípravě prostředí pro tento skript. Pomocí prvního dojde, v případě neexistence daného adresáře, k jeho vytvoření. Druhý *ExecStartPre* parametr smaže soubor *soubor.gz* z adresáře */zaloha*, který zde může již být obsažen.

Hlavní částí tohoto skriptu je *ExecStart*, který vytvoří soubor s výpisem všech skriptů. K provedení této částí je potřeba vytvořit další soubor, který tento příkaz provede, protože samotná *Systemd .service* jednotka není schopná tento příkaz vykonat. K vykonání příkazů z tohoto souboru je docíleno pomocí *ExecStart=/bin/bash nazev_souboru*.

Soubor *vypis.sh* obsahuje následující příkazy.

```
#! /bin/sh
ls –l *.service usr/lib/systemd/system>zaloha/soubor.txt
gzip –c /zaloha/soubor.txt > /zaloha/soubor.gz
```

Pomocí prvního příkazu je vytvořen soubor *soubor.txt* v adresáři */zaloha*, do kterého je pomocí *ls* příkazu vypsán seznam skriptů. Příkaz *gzip* tento soubor zabalí do souboru *soubor.gz*. Skrze *ExecStartPost* jsou nastavena práva souboru *soubor.gz*. *chmod 700* povolí všechna práva vlastníkovi souboru a ostatním uživatelům zakáže veškerý přístup k tomuto souboru. Jakmile je skript vykonán, je přes parametr *ExecStop* smazán vytvořený soubor *soubor.txt* z adresáře */zaloha*.

Oproti *Upstart* nebo *Sysvinit* skriptům nefungují v *Systemd* jednotkách samotné příkazy *mkdir*, *rm*, *chmod* a další. Příkaz *ExecStart=mkdir /zaloha* nebude fungovat. Nejprve je potřeba definovat cestu k tomuto souboru a až poté může být tento skript vykonán. Pokud má být vytvořen adresář nebo provedena změna práv je potřeba znát absolutní cestu k *mkdir* a *chmod* souborům, které jsou obsaženy v adresáři */usr/bin/*. Příkaz tedy musí vypadat následovně: *ExecStart=/usr/bin/mkdir /zaloha*.

6 Zhodnocení výsledků

V této bakalářské práci jsem se věnoval rozdílům při psaní skriptů mezi *init* systémy *Sysvinit*, *Upstart* a *Systemd*, které jsou používány jako systémy zavádění ve většině linuxových distribucí. Podle zjištěných informací o psaní těchto skriptů jsem došel k následujícím závěrům:

1) Sysvinit *init* systém byl před několika lety užívaný ve většině linuxových distribucí, avšak pro jeho neschopnost spustit více služeb současně je nahrazován *Upstart* a *Systemd init* systémem. Tyto dokáží spustit více služeb najednou, a díky tomu je start systému rychlejší než u *Sysvinit*.

2) Pro psaní vlastních skriptů není *Sysvinit* zcela ideální. V případě, že chce uživatel vytvořit vlastní jednoduchý skript, který bude například zjišťovat existenci souboru a v závislosti na tom provede předem určenou akci, nenastává problém. Pokud však bude *Sysvinit* skript provádět více akcí, dojde k nepřehlednosti celého kódu (skript bude na několik desítek řádek). K porovnání může být použit *rsyslog* skript, který je již obsažený ve všech těchto *init* systémech. Samotná funkce tohoto skriptu je u všech zavádění naprosto stejná. Zatímco *Systemd* a *Upstart* skript je maximálně na 15řádcích, *Sysvinit* skript tuto velikost mnohonásobně převyšuje.

3) *Sysvinit init* systém neobsahuje žádné sekce, a příkazy jsou tak provedeny postupně. V případě nedbalosti programátora při psaní skriptu může dojít k provedení příkazů ve spatném pořadí, které mohou následně vést k chybě při vykonávání skriptu nebo k provedení kódu v jiném pořadí, což může vést k nemalým problémům. Přehlednosti kódu naopak využívají zavádění *Systemd* a *Upstart*. Upstart obsahuje jednotlivé *script* sekce, díky kterým je přesně určeno, kdy se má která část kódu vykonat. V těchto sekcích jsou navíc podporovány všechny *bash* příkazy pro Linux.

4) *Systemd* provádí příkazy na podobném principu jako *Upstart* s tím, že příkazy, které se mají vykonat, nejsou obsaženy v jednotlivých *script* sekcích, ale jsou obsaženy v [*Service*] sekci. Zde je následně pomocí parametrů *ExecStart=*, *ExecStop=* nebo *ExecStartPre=* určeno, kdy má být příkaz, obsažený v těchto parametrech, proveden.

5) Formát *Systemd service* jednotky (skriptu) se částečně liší od *Upstart* a *Sysvinit init* skriptů. V *service* jednotce není možné psát příkazy tak, jak jsou běžně psány v ostatních *init* systémech. Parametr *ExecStart=rmdir /soubor* nebude nikdy proveden. Nejdříve je potřeba definovat cestu k souboru *rmdir*, který je obvykle obsažen v adresáři */usr/bin*. Aby byl *Systemd* schopen vykonat skript, musí být definována cesta k *bash rm* souboru. Service jednotka musí mít tedy tvar *ExecStart=/usr/bin/rmdir /soubor*.

6) *Systemd* nedokáže provést příkaz, který obsahuje více parametrů. Pokud má být odeslán e-mail se souborem pomocí *mail* příkazu, nebude *Systemd* schopen odeslat tuto přílohu, protože nedokáže vykonat celý příkaz tento příkaz. To je však možné vyřešit pomocí vykonání příkazu z *bash* souboru, pomocí kterého je možné provést příkazy s více parametry.

7) *Systemd* není schopen vykonat e-mailový skript spolu s přílohou skrze příkaz *mail*. Oproti *Sysvinit* a *Upstart* skriptu zde bylo potřeba využít příkaz *mutt*, přes který již tento skript fungoval. Jednalo se však o ojedinělý případ. Při testování vykonání různých příkazů s více parametry byl *Systemd* schopen vykonat, pomocí *bash* souboru, všechny testované skripty.

	Sysvinit	Upstart	Systemd
Používaný na systému	***Debian***	***Ubuntu***	***Fedora***
Podpora paralelního spuštění služeb	***NE***	***ANO***	***ANO***
Možnost provedení *bash* souboru	***ANO***[46]	***ANO***	***ANO***
Provedení příkazu s více parametry	***ANO***	***ANO***	***NE***[47]
Potřeba nastavení práv k provedení skriptu	***ANO***	***ANO***	***NE***

Tabulka 3 Porovnání jednotlivých *init* systémů

[46] *Sysvinit init* systém provádí skripty pouze přes *bash* soubory.
[47] *SystemD* je schopen provést příkaz s více parametry pouze skrze *bash* soubor. V samotné *.service* jednotce toho není schopen.

7 Závěr

Tato práce se zaměřila na význam *initu* samotného a také na historický vývoj *init* systémů spolu s jejich hlavními znaky. Pomocí porovnání jednotlivých *init* systémů, zejména při rozdílech psaní skriptů v těchto systémech, se pak práce snažila odpovědět na otázku, jaký je nejvhodnější *init* systém?

Vzhledem ke zjištěním poznatkům bylo zjištěno, že nelze jednoznačně určit nejvhodnější *init* systém. *Sysvinit init* systém lze však, na základě zjištěných informací (viz kapitola *Zhodnocení výsledků*), jasně zvolit jako nejhorší z porovnávaných *init* systémů, a to zejména kvůli neschopnosti spuštění více služeb současně a podpory skriptů pouze přes *bash* soubory. *Init* systémy *Upstart* a *Systemd* se ve svém konceptu liší velice málo, a je tedy těžké určit lepší z nich, protože v některých případech psaní skriptů převládá *Upstart* a v některých *Systemd*. Nejblíže k ideálnímu *init* systému však má, na základě zjištěných poznatků (viz kapitola *Zhodnocení výsledků*), *Systemd init* systém, zejména díky snadnějšímu psaní *shell* skriptů a znatelnému zrychlení startu systému.

8 Seznam pramenů a literatury

MOLIČ, Jan. Inicializace aneb od Initu k Runitu. Dostupné z: http://www.root.cz/clanky/inicializace-aneb-od-initu-k-runitu/

Introduction to Linux: Processes. In: [online]. [cit. 2014-10-27]. Dostupné z: http://www.tldp.org/LDP/intro-linux/html/sect_04_02.html

System V Definition. [online]. [cit. 2014-10-21]. Dostupné z: http://www.linfo.org/system_v.html

Systemd – úvod a představení System V init: Na počátku byl (System V) init. [online]. [cit. 2014-10-26]. Dostupné z: http://www.abclinuxu.cz/clanky/systemd-uvod-a-predstaveni-system-v-init#na-pocatku-byl-system-v-init

HAGARA, Ladislav. Source Mage GNU/Linux: Není init jako init: BSD init. [online]. [cit. 2014-11-11]. Dostupné z: http://www.root.cz/clanky/source-mage-gnu-linux-init/

Jak startuje systém: Proces init. In: [online]. [cit. 2015-03-31]. Dostupné z: http://www.linuxexpres.cz/praxe/jak-startuje-system

System Startup. [online]. [cit. 2014-11-11]. Dostupné z: https://www.freebsd.org/doc/en/articles/linux-users/startup.html

Upstart - event-based init deamon. In: [online]. [cit. 2014-03-14]. Dostupné z: http://upstart.ubuntu.com/index.html

KRČMÁŘ, Petr. Upstart zpřehlední a zrychlí start systému. Root.cz [online]. [cit. 2013-10-11]. Dostupné z: http://www.root.cz/clanky/upstart-zprehledni-a-zrychli-start-systemu/

HUNT, James a Clint BYRUM. Upstart Intro, Cookbook and Best Practises: Concepts and Terminology. In: [online]. [cit. 2014-11-15]. Dostupné z: http://upstart.ubuntu.com/cookbook/#concepts-and-terminology

HUNT, James a Clint BYRUM. Upstart Intro, Cookbook and Best Practises: Upstart's Design: Why It Is Revolutionary. In: [online]. [cit. 2014-11-15]. Dostupné z: http://upstart.ubuntu.com/cookbook/#upstart-s-design-why-it-is-revolutionary

HUNT, James a Clint BYRUM. Job Lifecycle: Starting a job. [online]. [cit. 2015-01-19]. Dostupné z: http://upstart.ubuntu.com/cookbook/#starting-a-job

HUNT, James a Clint BYRUM. Job Lifecycle: Stopping a Job. [online]. [cit. 2015-01-19]. Dostupné z: http://upstart.ubuntu.com/cookbook/#stopping-a-job

HUNT, James a Clint BYRUM. Event Types: Hooks. In: [online]. [cit. 2015-03-05]. Dostupné z: http://upstart.ubuntu.com/cookbook/#hooks

HUNT, James a Clint BYRUM. *Upstart Intro, Cookbook and Best Practises: Event* [online]. [cit. 2014-11-16]. Dostupné z: http://upstart.ubuntu.com/cookbook/#event

VYSKOČIL, Michal. Systemd – principy: Co je systemd. [online]. [cit. 2014-11-16]. Dostupné z: http://www.abclinuxu.cz/clanky/systemd-principy

Concepts [online]. [cit. 2014-11-15]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.html

Description [online]. [cit. 2014-11-15]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.unit.html

Systemd.service: Descripion. In: [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.service.html

Systemd.device: Descripion. In: [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.device.html

Systemd.socket: Descripion. In: [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.socket.html

Systemd.mount: Descripion. In: [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.mount.html

Systemd.automount: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.automount.html

Systemd.snapshot: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.snapshot.html

Systemd.timer: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.timer.html

Systemd.swap: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.swap.html

Systemd.path: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.path.html

Systemd.slice: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.slice.html

Systemd.scope: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.scope.html

Systemd.target: Descripion. [online]. [cit. 2015-03-24]. Dostupné z: http://www.freedesktop.org/software/systemd/man/systemd.target.html

Getting Started with systemd: Terminology. In: [online]. [cit. 2014-11-16]. Dostupné z: https://coreos.com/docs/launching-containers/launching/getting-started-with-systemd/

Working with systemd Targets. In: [online]. [cit. 2014-11-16]. Dostupné z: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/sect-Managing_Services_with_systemd-Targets.html

Creating your own init skripts. In: [online]. [cit. 2014-11-16]. Dostupné z: http://www.softpanorama.org/Commercial_linuxes/Startup_and_shutdown/creating_your_own_init_skripts.shtml

Soubor /dev/null [online]. [cit. 2015-03-06]. Dostupné z: http://www.abclinuxu.cz/slovnik/soubor-dev-null

HUNT, James a Clint BYRUM. Upstart Intro, Cookbook and Best Practises: Concepts and Terminology. In: [online]. [cit. 2014-11-15]. Dostupné z: http://upstart.ubuntu.com/cookbook/#concepts-and-terminology

HUNT, James a Clint BYRUM. Upstart Intro, Cookbook and Best Practises: Expect. [online]. [cit. 2015-01-14]. Dostupné z: http://upstart.ubuntu.com/cookbook/#expect-daemon

Dbus-uuidgen. [online]. [cit. 2014-11-27]. Dostupné z: http://dbus.freedesktop.org/doc/dbus-uuidgen.1.html

Apparmor. [online]. [cit. 2015-01-23]. Dostupné z: http://manpages.ubuntu.com/manpages/precise/en/man7/apparmor.7.html

VYSKOČIL, Michal. Systemd – jednotky a závislosti: Stručný přehled syntaxe. [online]. [cit. 2015-03-07]. Dostupné z: http://www.abclinuxu.cz/clanky/systemd-jednotky-a-zavislosti

9 Seznam tabulek