VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

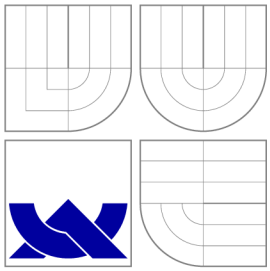
AUTOMATICKÁ INSTALACE A KONFIGURACE OPERAČNÍHO SYSTÉMU GNU/LINUX, DISTRIBUCE GENTOO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

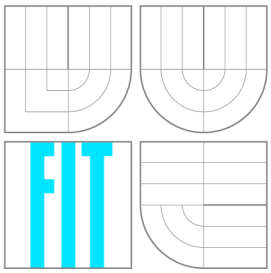
AUTOR PRÁCE
AUTHOR

MARTIN KŘÍŽEK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

AUTOMATICKÁ INSTALACE A KONFIGURACE OPERAČNÍHO SYSTÉMU GNU/LINUX, DISTRIBUCE GENTOO

AUTOMATED INSTALLATION AND CONFIGURATION OF OPERATING SYSTEM GNU/LINUX, DISTRIBUTION GENTOO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KŘÍŽEK

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. MAREK RYCHLÝ, Ph.D.

BRNO 2010

Abstrakt

Úkolem této práce je navrhnout a implementovat nástroj, který uloží souhrnnou konfiguraci již instalovaného Gentoo Linux systému a nástroj, který automaticky provede novou instalaci systému se stejnými balíčky a stejnou konfigurací na stejné či jiné hardwarové konfiguraci.

Abstract

This bachelor's thesis deals with designing and implementation of a tool which stores a configuration of installed Gentoo Linux system and a tool that performs automated installation of a new system of the same configuration and with the same packages on both the same and a different hardware configuration.

Klíčová slova

GNU, Linux, Gentoo, Python, Portage, Emerge, Ebuild, automatická instalace, automatická konfigurace

Keywords

GNU, Linux, Gentoo, Python, Portage, Emerge, Ebuild, automated installation, automated configuration

Citace

Martin Křížek: Automatická instalace a konfigurace operačního systému GNU/Linux, distribuce Gentoo, bakalářská práce, Brno, FIT VUT v Brně, 2010

Automatická instalace a konfigurace operačního systému GNU/Linux, distribuce Gentoo

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Mgr. Marka Rychlého, Ph.D., a že jsem uvedl všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Křížek
19. května 2010

Poděkování

Děkuji vedoucímu práce, Mgr. Marku Rychlému, PhD., za podnětné připomínky a odbornou pomoc, kterou mi během mé práce ochotně poskytoval. Dále pak děkuji svým rodičům za podporu a zázemí, které mi ke studiu vytvořili.

Obsah

Kapitola 1

Úvod

Distribuce Gentoo Linux je známa hlavně tím, že klade důraz na vysokou konfigurovatelnost a rychlost, díky tomu, že všechny balíčky jsou kompilované přímo ze zdrojových kódů.

Je také známa svou netypickou instalací. Uživatel musí nastavit vše ručně, což je náročnými uživateli bráno kladně. Na druhou stranu, instalace může zabrat mnoho hodin. Cílem této práce bude vytvořit prostředky, které zautomatizují instalaci systému Gentoo Linux při zachování konfigurace zdrojového systému a tak uživatel nepřijde o výhodu konfigurovatelnosti systému.

Kapitola *Gentoo Linux* v krátkosti popisuje historii operačního systému GNU/Linux a historii a filozofii distribuce Gentoo Linux.

Kapitola nazvaná *Portage* seznamuje čtenáře s balíčkovacím systémem Portage, který distribuce Gentoo Linux používá k instalaci aplikací.

Instalace Gentoo Linux je kapitola popisující současný stav instalace a konfigurace distribuce Gentoo Linux.

V kapitole *Návrh* je popsáno, jaké problémy bylo nutné řešit při navrhování nástrojů pro automatickou instalaci a konfiguraci distribuce Gentoo Linux.

Kapitola *Implementace* stručně popisuje implementaci nástrojů.

V kapitole *Testování a dosažené výsledky* se definují případy testování nástrojů a výsledky těchto testování.

Kapitola *Další možný vývoj* diskutuje možná rozšíření, která by byla vhodná, v případě budoucího vývoje, zvážit a implementovat za účelem vylepšení použitelnosti nástrojů.

Poslední kapitola *Závěr* obsahuje zhodnocení práce a její přínos.

Kapitola 2

Gentoo Linux

2.1 GNU/Linux

Operační systém GNU/Linux se skládá z jádra operačního systému zvaného Linux a z kolekce svobodného software GNU. GNU je rekurzivní akronym pro „GNU's Not Unix“[1]. Projekt GNU byl založen Richardem Stallmanem ve snaze vytvořit ničím neomezený a volně šiřitelný unixový operační systém. V rámci projektu GNU se podařilo vytvořit mnoho softwarových nástrojů, především to byla kolekce překladačů GCC[2] či textový editor Emacs. Jediným chybějícím článkem bylo jádro operačního systému. Tento nedostatek měl být vyřešen začátkem vývoje jádra GNU/Hurd, kterému se bohužel dodnes nedostalo verze, která by byla schopna konkurence. Na počátku 90. let, nezávisle na projektu GNU, začal finský student Linus Torvalds pracovat na svém jádru operačního systému, které bylo pojmenováno Linux. Torvalds prohlásil, že hlavním důvodem byla příliš vysoká cena za UNIX, který na jeho univerzitu dodávala společnost Sun Microsystems. [10]

GNU/Linux se rozšířil hlavně díky softwaru, který byl pro něj primárně vytvořen. Hlavním takovým softwarem je webový server Apache. V dnešní době však roste počet instalací i na domácích a přenosných počítačích. Uživatelé si mohou vybrat z mnoha distribucí, každá z nich se zaměřuje na jinou část trhu. Jednou z těchto distribucí je Gentoo Linux.

2.2 Historie

Daniel Robbins pracoval na novém balíčkovacím systému pro distribuci Stampede Linux. Ačkoliv měl podporu hlavních vývojářů, ostatní jeho vývoj neocenili. Rozhodl se proto, že vytvoří zcela novou distribuci s pracovním názvem Enoch Linux. Cílem byl vytvořit distribuci, která bude klást důraz na výkon. Toho dosáhl vytvořením balíčkovacího systému, který nevyužíval předkompilované balíčky, ale programy kompiloval ze zdrojových kódů přímo na cílovém počítači. [12] Poté co se Enoch Linux rozrostl, Robbins jej přejmenoval na Gentoo Linux. Po problémech s instalací operačního systému GNU/Linux na jeho nový počítač, Robbins přešel na systém FreeBSD. [13] Pár měsíců od přechodu na FreeBSD se Robbins rozhodl vrátit k systému GNU/Linux a pokračovat ve vývoji distribuce Gentoo Linux. Po návratu k vývoji, Robbins implementoval několik vlastností z balíčkovacího systému z FreeBSD do balíčkovacího systému Portage v distribuci Gentoo Linux. [14]

[1] „GNU Není Unix“
[2] GNU Compiler Collection

2.3 Filozofie

Gentoo Linux je volně šiřitelný operační systém založen na jádru Linux, existuje však i varianta založena na jádru FreeBSD. Gentoo Linux klade důraz na konfiguraci a výkon, což mu umožňuje balíčkovací systém Portage, díky kterému může uživatel ze své instalace vytvořit webový server, vývojářskou stanici či multimediální systém. Právě proto je Gentoo Linux označována jako metadistribuce. Obsahuje pouze jádro operačního systému a základní systém s GNU utilitami, zbytek obsahu systému je čistě na uživateli. [4]

Distribuce je často kritizována za časově náročné instalace balíčků. Toto je pochopitelně způsobeno tím, že balíčky nejsou předkompilované, ale stahují se na počítač v podobě zdrojových kódů, které jsou následně kompilovány, což může trvat v případě rozsáhlého balíčku i několik hodin. Není to však vždy nutné. Balíčky jako např. `app-office/openoffice` jsou dodávány i v binární podobě, `app-office/openoffice-bin`, a tak instalace zabere místo několika hodin jen několik minut. Uživatel však ztrácí možnost optimalizací kompilátoru. Balíčkovací systém Portage nabízí uživatelům možnost paralelní a distribuované kompilace pro urychlení procesu kompilace.

Mnoho uživatelů se domnívá, že tato distribuce není vhodná pro začátečníky. Toto však není zcela pravda. Díky velice kvalitně zpracované dokumentaci je možné naučit se principům distribuce či dokonce operačního systému GNU/Linux velice rychle. Toto však samozřejmě nemusí být záměr uživatele a v takovém případě je opravdu vhodné se podívat po nějaké více uživatelsky přátelské distribuci. Gentoo Linux je dostupný na několika platformách včetně x86, x86-64, Intel Itanium, PowerPC či SPARC [17].

Kapitola 3

Portage

3.1 Úvod

Portage je systém správy balíčků v distribuci Gentoo Linux. Systém je napsán v jazycích Python a Bash a tedy, co se týče zdrojových kódů, snadno přístupný uživateli. Skládá se ze dvou hlavních částí, systémů *ebuild* a *emerge*. Systém *ebuild*, přímé rozhraní pro systém Portage, se stará o kompilaci balíčků a jejich instalaci, zatímco *emerge* poskytuje rozhraní pro *ebuild*, tedy řeší závislosti, spravuje hlavní Portage strom apod. Přestože se nástroje *ebuild* a *emerge* dají velice pohodlně používat z prostředí příkazové řádky, existují pro ně i grafické rozhraní, např. Porthole. (viz. příslušné manuálové stránky – `man ebuild` a `man emerge`)

3.2 Portage strom

Strom Portage je kolekce ebuildů, což jsou soubory obsahující informace, které říkají systému Portage jak daný balíček nainstalovat či jaké další balíčky jsou pro jeho instalaci nebo správu potřeba. Výchozí adresář pro uložení souborů *ebuild* je `/usr/portage`, avšak je možné jej zvolit nastavením proměnné PORTDIR v souboru `/etc/make.conf`.

Strom Portage obsahuje adresáře reprezentující kategorie balíčků, ve kterých jsou adresáře reprezentující samotné balíčky. V adresářích balíčků jsou pak jednotlivé ebuildy, jeden pro každou verzi balíčku, dále pak soubor `Manifest`, obsahující kontrolní součty souborů v adresáři a soubor `Changelog` popisující změny v jednotlivých verzích balíčku. Cesta k ebuildu bude vypadat např. takto: `/usr/portage/app-editors/vim/vim-7.2.359.ebuild`, kde `app-editors` je kategorie balíčku, `vim` je balíček a `vim-7.2.359.ebuild` je *ebuild*, a jde o balíček textového editoru Vim ve verzi `7.2.359`.

Zdrojové kódy balíčků se ukládají do adresáře `distfiles/` v adresáři uloženém v proměnné `PORTDIR`. Se „stárnutím“ systému tento adresář nabývá na velikosti. Zdrojové kódy se stahují až před samotnou kompilací.

Aby měl Portage přístup k nejnovějším balíčkům, např. při jejich instalaci či vyhledávání, je nutné strom Portage pravidelně aktualizovat, tedy stáhnout na pevný disk aktuální *ebuildy*. Aktualizace probíhá pomocí nástroje *emerge* a jeho přepínače `--sync`, který využívá systémovou utilitu *rsync*. Pokud uživatel není schopen *rsync* použít, je možné použít přepínač `--websync`, který stáhne z webu nejaktuálnější strom a nainstaluje do systému.

3.3 Uložiště informací o nainstalovaných balíčcích

Informace o již nainstalovaných balíčcích jsou v adresáři `/var/db/pkg`, který obsahuje adresáře reprezentující kategorie balíčků, ve kterých jsou adresáře reprezentující samotné verze nainstalovaných balíčků. Každý z těchto adresářů obsahuje soubory s informacemi o použitých přepínačích kompilátoru při kompilaci balíčku či použitých USE flags (viz. 3.5). Dále také obsahuje informace o celém Portage prostředí, např. textový popis balíčku, apod. Názvy balíčků instalované uživatelem přímo, tedy se nejedná o závislosti, se ukládají do souboru `/var/lib/portage/world`.

3.4 Správa balíčků

V Gentoo Linux se balíčky spravují pomocí nástroje *emerge*. Instalace je jednoduchá, ostatně jako na většině linuxových distribucí. Stačí napsat do příkazové řádky `emerge balíček` a systém udělá již vše potřebné, od vyřešení závislostí, přes stažení zdrojových kódů a kompilaci, až po instalaci nových souborů do systému. Je užitečné vědět, které balíčky se nainstalují jako závislosti. K tomu slouží přepínač `-p`, resp. `--pretend`, který vypíše co a jak bude s balíčkem instalováno. Uživatel tak může mít instalaci pod kontrolou a případně zabránit instalaci nechtěných balíčků.

Je doporučeno systém aktualizovat pravidelně, aby měl nejnovější verze balíčků, např. z důvodů bezpečnosti. Také se tak dá vyhnout problémům s přechody na vyšší, „major“, verze programů, které často bývají problematické, jmenujme alespoň X server. Dva roky neaktualizovaný systém může být při následné aktualizaci pro správce pohromou. Aktualizaci provedeme příkazem `emerge -u world`, který vyhledá a nainstaluje aktualizace balíčků v souboru `/var/lib/portage/world`. Použitím přepínače `-D` docílíme aktualizace všech balíčků v systému. Užitečný je také přepínač `-N`, který zkontroluje změnu USE flags a pokud k nějaké změně došlo, překompiluje balíčky podporující přidaný, resp. odebraný USE flag (viz. 3.5).

Odebrání balíčků ze systému už tak bezproblémové není. Příkaz `emerge -C balíček` sice daný balíček odebere, ale nezkontroluje, zda na něm závisí nějaký jiný balíček, což může způsobit nefunkčnost daného balíčku. Útěchou nám může být to, že pokud se pokusíme odebrat systémový balíček, např. `sys-apps/coreutils` obsahující utility jako *cp*, *mv*, *rm*, *ls* a další, dostaneme varování a máme možnost proces přerušit.

Pro odstranění již nepotřebných balíčků, typicky závislostí, můžeme použít přepínač `--depclean` nástroje *emerge*. Ještě předtím je však nutné mít aktualizovaný systém, tedy `emerge -DuN world`. Pak však musíme znovu přeložit balíčky, které jsou dynamicky linkovány s odebranými knihovnami, což umožňuje nástroj *revdep-rebuild*, který je součástí balíčku `app-portage/gentoolkit`. [19]

3.5 USE Flags

Konfigurace balíčků probíhá přes tzv. USE flags, které se ukládají do konfiguračního souboru `/etc/make.conf` do proměnné `USE`. Jsou to textové řetězce, podle kterých systém Portage pozná podporu toho, co má či nemá při kompilaci zdrojových kódů balíčku přidat. Jako příklad nám může posloužit USE flag `jabber`, který přidá do všech aplikací, které toto umožnují, např. instant messanger klient, podporu komunikačního protokolu Jabber. Seznam dostupných USE flags jsou k nalezení v souboru `/usr/portage/profiles/use.desc`.

Proměnná `USE` v konfiguračním souboru `/etc/make.conf` platí pro všechny balíčky v systému. Systém Portage však umožňuje použít USE flag pouze pro vybrané aplikace, a to přidáním záznamu do konfiguračního souboru `/etc/portage/package.use`. Takový záznam má tvar `kategorie/jméno USE_flag`, např. `kde-base/kopete jabber`. Nechceme-li podporovat určitou funkcionalitu, napíšeme před daný USE flag znak „-“, tedy např. `-gnome`, čímž zabráníme instalaci knihoven prostředí okenního manažeru Gnome.

3.6 Maskování verzí

Maskování verzí určuje, které balíčky vývojáři považují za stabilní, tedy použitelné pro práci a které ne. Tento systém, řekněme, připomíná systém známý např. z distribuce Debian, kde existují větve Stable, Unstable, Testing, popř. Experimental [2]. V Gentoo Linux existují balíčky nemaskované, maskované a tzv. Hard Masked. Balíčky nemaskované jsou označeny jako stabilní, balíčky maskované běžně fungují jako stabilní, avšak nejsou ještě dostatečně otestovány. Hard Masked jsou balíčky, u kterých je známo, že obsahují problémy a jsou uvedeny v souboru `package.mask` v adresáři `/usr/portage/profiles/`, kde jsou opatřeny komentářem od vývojáře popisující problém balíčku.

Chceme-li používat to, co bychom v distribuci Debian nazvali Unstable, nastavíme proměnnou `ACCEPT_KEYWORDS` v konfiguračním souboru `/etc/make.conf` na `~`arch, kde arch je např. `x86`, `amd64`, `ppc` apod. Tímto ale možnosti Portage nekončí. Chceme-li používat stabilní balíčky, ale chceme mít např. vždy nejnovější verzi jádra v systému, využijeme možnosti souboru `/etc/portage/package.keywords`, kde, např. na platformě x86_64, přidáme řádek `sys-kernel/gentoo-sources ~amd64`.

Pro lepší přehlednost můžeme mít místo souboru `package.keywords` stejnojmenný adresář. Do takového adresáře, např. `/etc/portage/package.keywords/kde/`, uložíme soubory se stejnou strukturou jako soubor `package.keywords`, který budeme používat pro odmaskování balíčků týkajících se prostředí KDE[1]. Všechny takové soubory budou pak seřazeny a systém Portage k nim bude přistupovat jako k jednomu.

Máme-li dostatek odvahy, můžeme použít i balíčky označené jako Hard Masked. Docílíme toho vložením jejich názvu ve tvaru `kategorie/jméno` do konfiguračního souboru `/etc/portage/package.unmask`.

Řekněme, že máme problém s určitou verzí balíčku, např. `net-im/psi-0.13`. Můžeme systému Portage říci, že tuto verzi má ignorovat, a to tak, že přidáme řádek `=net-im/psi-0.13` do konfiguračního souboru `/etc/portage/package.mask`. Samozřejmě můžeme použít i ostatní porovnávací znaménka. Použijeme-li `>=net-im/psi-0.13`, řekneme tím, že si nepřejeme, aby Portage instaloval verzi 0.13 a vyšší. [8]

3.7 Overlays

Overlays jsou neoficiální stromy balíčků. Strukturou se neliší od hlavního Portage stromu. Spravovány mohou být ručně či pomocí nástroje *layman* z balíčku `app-portage/layman`. Nástroj *layman* uchovává cesty ke svým stromům v proměnné `PORTDIR_OVERLAY` v konfiguračním souboru `/usr/local/portage/layman/make.conf`[2]. Cesty k uživatelským overlays jsou uloženy v proměnné `PORTDIR_OVERLAY` v konfiguračním souboru `/etc/make.conf`.

[1] K Desktop Environment

[2] od verze 1.3.0 je tento adresář `/var/lib/layman`

Overlays jsou užitečné tehdy, když chceme mít v systému *ebuild*, který není v hlavním Portage stromu, ale kdybychom jej do hlavního stromu nakopírovali, při `emerge --sync` by jej systém Portage odstranil, protože by se neshodoval s oficiálním Portage stromem. Overlays jsou také vhodná pro testování.

3.8 Systémový profil

Systémový profil určuje implicitní hodnoty `USE`, `CFLAGS` a dalších důležitých proměnných. Další vlastností je znepřístupnění určitých verzí balíčků pro systém. Dostupné systémové profily můžeme vypsat příkazem `eselect profile list`, profil pak vybereme příkazem `eselect profile set` *`číslo profilu`*. Tyto profily jsou spravovány vývojáři distribuce Gentoo Linux.

3.9 Konfigurační soubory

Konfigurační soubory systému a aplikací jsou uloženy v adresáři `/etc`, popř. jiných, např. `/usr/local`. Pokud uživatel chce změnit chování aplikace, změní příslušně jeho konfigurační soubor. Protože však vývojáři aplikací přidávají v nových verzích také nové vlastnosti, mění se i struktura konfiguračních souborů. Často pak nastává situace, kdy po instalaci aplikace se má nahrát jeho nový konfigurační soubor, ale kdyby se tak stalo, přepíše se tím jeho uživatelem upravená verze, což je z hlediska uživatele nepřípustné. Systém Portage uchovává MD5[3] kontrolní součet konfiguračních souborů z adresářů `CONFIG_PROTECT` v souborech `/var/lib/portage/config` a `/var/db/pkg/`*`kategorie/balíček`*`/CONTENTS`. Když pak nastane kolize konfiguračních souborů po instalaci balíčku, systém Portage zkontroluje právě pomocí zmíněných MD5 kontrolních součtů, zda byl konfigurační soubor změněn, tedy uložený MD5 kontrolní součet se liší od toho skutečného. A pokud ano, upozorní uživatele a nový konfigurační soubor uloží pod jménem `._cfg0000_<skutečné_jméno_souboru>`. Toto však probíhá pouze v adresářích z proměnné `CONFIG_PROTECT`. Pokud uživatel chce, aby systém Portage automaticky nahradil konfigurační soubory v podadresáři `CONFIG_PROTECT`, musí jej přidat do proměnné `CONFIG_PROTECT_MASK`. Uživatel může sloučení konfiguračních souborů v adresáři `CONFIG_PROTECT` provést ručně, nicméně systém Portage obsahuje nástroje *etc-update* a *dispatch-conf*, které tyto změny provádějí interaktivně. [18]

3.10 SLOT

Je časté, že balíčky jsou závislé na určité verzi knihovny. Některé si vystačí s verzí nižší, zatímco další potřebují pro svou funkčnost verzi knihovny vyšší. V takovém případě systém Portage umožňuje, aby bylo v systému nainstalováno současně více verzí stejné knihovny. Každá pak bude nainstalovaná v tzv. `SLOT`u. Číslo `SLOT`u, do kterého bude systém Portage knihovnu instalovat, je uveden přímo v souboru `.ebuild` daného balíčku knihovny. [8]

[3] Message-Digest algorithm

Kapitola 4

Instalace Gentoo Linux

4.1 Úvod

Tato kapitola popisuje současný proces instalace distribuce Gentoo Linux [17] tak, aby bylo možné v závěru této práce zhodnotit, v jaké míře se podařilo instalaci a konfiguraci automatizovat.

4.2 Výběr instalačního média

Při výběru instalačního prostředí má uživatel víceméně dvě volby, Gentoo Linux instalační CD a instalace z již běžícího systému. Gentoo Linux instalační CD umožňuje uživateli zavést připravené prostředí systému Gentoo Linux. Instalační CD detekuje hardware a nahraje do jádra příslušné moduly. Po zavedení systému jsme schopni začít samotnou instalaci. Instalace z běžícího systému probíhá po vstupu do nového prostředí pomocí programu *chroot*.

4.3 Konfigurace sítě

V případě, že systém je zapojen do sítě přes Ethernet s DHCP serverem je velice pravděpodobné, že instalační CD nastavilo síť automaticky. V případě, že síť nefunguje, instalační CD poskytuje několik nástrojů, které uživatel může k nastavení sítě použít, jako např. *net-setup*, *pppoe-setup* či *ifconfig*.

4.4 Příprava disků

Pro vytvoření, změnu a odstranění oddílů disků může uživatel využít příkazové prostředí nástroje *fdisk*. Vytvoření souborových systémů lze pak provést programy, jako např. *mke2fs* nebo *mkfs*. Aby bylo možné pracovat s daty na oddílech, je nutné připojit souborový systém na daném oddílu do adresářové struktury systému a to pomocí programu *mount*. Aby se oddíly připojily při zavádění systému automaticky, uživatel musí do souboru `/etc/fstab` zapsat informace o daných oddílech. Takové informace jsou např. jméno zařízení, číslo oddílu, adresář připojení a další argumenty jako např. uživatelská práva.

Chceme-li např., aby se oddíl `/dev/sda1` se souborovým systémem `ext2` automaticky při zavedení systému připojil do adresáře `/boot`, přidáme do souboru `/etc/fstab` následující řádek: `/dev/sda1 /boot ext2 defaults 1 2`.

4.5 Instalace instalačních souborů

Před samotným kopírováním souborů je vhodné nastavit aktuální čas a datum programem *date*, aby se předešlo hlášením při zavádění systému, že čas modifikace souborů je v budoucnosti. Následuje stažení `stage3` archivu a jeho rozbalení do kořenového adresáře nového systému.

`Stage3` archiv obsahuje minimální Gentoo Linux prostředí. Existují také archivy `stage2` a `stage1`, které už nejsou oficiálně podporovány. V čem se tedy jednotlivé etapy, `stage1`, `stage2`, `stage3` liší? U `stage1` uživatel zkompiluje celý systém včetně překladače a knihoven jazyka C sám, ve `stage2` je toto již předkompilováno, ale je nutné zkompilování systémových nástrojů a ve `stage3` jsou již předkompilovány i systémové nástroje. [7]

Dalším krokem je stažení archivu s Portage stromem a jeho rozbalení do adresáře uloženém v proměnné `PORTDIR`. Ještě je třeba provést nastavení možností kompilace, tedy proměnných `CFLAGS`, `CHOST` a `MAKEOPTS` (viz. 5.2.4) v souboru `/etc/make.conf`.

4.6 Instalace základního Gentoo systému

Aby systém Portage věděl odkud má stahovat jednotlivé ebuildy a zdrojové kódy, musíme k tomu, např. pomocí nástroje *mirrorselect*, vybrat servery obsahující potřebné soubory. Ještě před vstupem do nového prostředí (*chroot*) je nutné připojit souborové systémy `/proc` a `/dev` a zkopírovat nastavení DNS[1] serveru.

Dalším krokem je aktualizace Portage stromu příkazem `emerge --sync`. Následuje výběr systémového profilu nástrojem `eselect` a nastavení USE flags v konfiguračním souboru `/etc/make.conf`.

4.7 Konfigurace jádra

V distribuci Gentoo Linux mají uživatelé dvě možnosti, jak jádro operačního systému nakonfigurovat a následně nainstalovat. Oběma postupům však předchází stažení zdrojových kódů jádra. Toto zařídíme spuštěním např. `emerge sys-kernel/gentoo-sources`.

První možností je ruční instalace, oblíbená mezi zkušenějšími uživateli. Konfigurace probíhá mj. v textovém prostředím po spuštění příkazu `make menuconfig`, a následným spuštěním `make && make modules_install` jádro zkompilujeme.

Druhou možností, jak jádro konfigurovat, je použitím balíčku `sys-kernel/genkernel`. Jedná se o automatickou konfiguraci a instalaci jádra, určena především pro začínající uživatele.

4.8 Konfigurace systému

V této části uživatel vytváří, jak již bylo zmíněno, soubor `/etc/fstab`, nastavuje síť systému změnou souboru `/etc/conf.d/net`, nastavuje heslo superuživatele či třeba rozvržení klávesnice.

[1] Domain Name System

4.9 Instalace nezbytných systémových nástrojů

Je pravděpodobné, že nástroje v dodaném `stage3` archivu uživateli nebudou vyhovovat, v této části je tedy prostor pro instalaci nástrojů, které vyhovují uživateli. Takové nástroje mohou být `app-admin/syslog-ng`, `sys-process/vixie-cron` či `net-misc/dhcpcd`.

4.10 Konfigurace zavaděče

Další důležitou částí instalace je instalace a konfigurace zavaděče, abychom systém mohli vůbec spustit. Gentoo Linux nabízí několik různých zavaděčů, z nichž ty nejznámější jsou `sys-boot/grub` a `sys-boot/lilo`.

4.11 Dokončení instalace

Poslední věcí, kterou je vhodné udělat, je vytvořit účty uživatelů pro denní použití a případně, pro pořádek, smazat stažené archivy z kořenového adresáře. Je pak na uživateli jaké balíčky si do svého nového systému nainstaluje.

Kapitola 5

Návrh

5.1 Sbírání konfigurace systému

5.1.1 Konfigurace systémů

V rámci konfigurace systému je nutné uložit všechny adresáře v proměnné `CONFIG_PROTECT` systému Portage. V případě, že uživatel bude instalovat cílový systém na stejný hardware jako zdrojová instalace, bude uložena i konfigurace jádra, tedy soubor `.config` ve všech podadresářích adresáře `/usr/src/`, které obsahují zdrojové kódy různých verzí jádra. Další akcí je smazání souboru `/etc/fstab` (viz. 5.2.1).

5.1.2 Uživatelská konfigurace aplikací

Uložení konfigurace aplikací může být náročné z hlediska velikosti výsledného archivu s konfigurací, což může způsobovat problémy při přenášení konfigurace, a tak je tato možnost volitelná. Jako samotné konfigurace se budou uvažovat soubory a adresáře začínající znakem „.“ v uživatelských podadresářích adresáře `/home`.

5.1.3 Overlays

Častou součástí instalací jsou také overlays vytvořené uživatelem. Ty je pochopitelně také nutné přenést. Cesty k těmto overlays jsou k nalezení v proměnné `PORTDIR_OVERLAY` v konfiguračním souboru `/etc/make.conf`.

Pokud se v systému nachází balíček `app-portage/layman`, který slouží pro automatickou správu overlays, je nutné také přenést jeho konfigurační soubor `make.conf` v adresáři `/usr/local/portage/layman/`[1].

5.1.4 Nainstalované balíčky

Informace o nainstalovaných balíčcích jsou v souboru `world` v adresáři `/var/lib/portage/` a v adresáři `/var/db/pkg` (viz. 3.3).

O nainstalovaných balíčcích budeme uchovávat kategorii, do které daný balíček náleží, jeho jméno a verzi. Tyto informace budeme ukládat ve, Gentoo zaběhlém a tedy pro uživatele čitelném, formátu: `kategorie/jméno-verze`. Každý nainstalovaný balíček bude zapsán v souboru `packages` na jednom řádku.

[1] od verze 1.3.0 je tento adresář `/var/lib/layman`

5.1.5 Informace o systému

Systém obsahuje několik informací, které budou vyžadovány pro funkčnost samotné instalace a bude je proto nutné uložit.

První takovou informací je cesta k profilu systému, která se na různých architekturách liší.

Další informací je architektura, v Gentoo Linux `keyword`, zdrojové instalace, např. `amd64`. Ta potom bude užitečná při zjišťování, zda se zdrojová a cílová platforma liší a v případě, že ano, bude nutné provést potřebné změny. První změnou je přepsání proměnné `ACCEPT_KEYWORDS` (viz. 3.6). Dále pak v souboru, popř. adresáři `package.keywords` v adresáři `/etc/portage/` musí být změněny všechny `keyword` a také cesta k profilu systému.

Musíme také uložit název aktuálně běžícího jádra, abychom věděli, v případě více instalovaných verzí, které zkompilovat (viz. 5.2.7).

V případě, že si uživatel nevybere volbu uložení konfigurace aplikací, pak se v adresáři `/home` při následné instalaci nevytvoří uživatelské adresáře, proto je nutné si jejich názvy také uložit a při instalaci je vytvořit.

Abychom věděli, kam máme nahrát Portage strom, musíme uchovat i obsah proměnné `PORTDIR`.

Poslední informací, kterou budeme ukládat je obsah proměnné `GENTOO_MIRRORS`, jejíž obsahem je jedna či více adres URL[2] počítače, ze kterého se stahují aktualizace Portage stromu a zdrojové kódy. Z těchto adres budou pak při instalaci staženy `stage3` archiv a aktuální Portage strom.

Všechny tyto informace budou uloženy v konfiguračním souboru `gentoo.conf`, který může mít např. následující podobu:

```
PROFILE   = "../usr/portage/profiles/default/linux/amd64/10.0/"
KERNEL    = "linux-2.6.30-gentoo-r5"
HOMES     = "uzivatel1 uzivatel2"
ARCH      = "amd64"
PORTDIR   = "/usr/portage"
MIRRORS   = "http://ftp.fi.muni.cz/pub/linux/gentoo/"
```

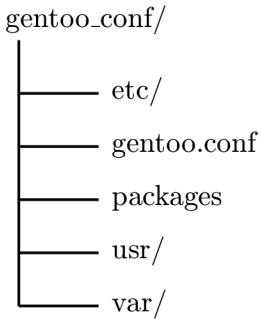
Příklad 5.1: Ukázka konfiguračního souboru `gentoo.conf`

5.1.6 Uložiště konfigurace

Získanou konfiguraci systému bude nutné uložit nejlépe tak, aby byla snadno přenositelná mezi počítači. Bude také vhodné, aby velikost konfigurace byla co nejmenší, tedy použít nějakou kompresi. Samozřejmě, dalším požadavkem je otevřenost vybraného úložiště a dostupnost programů, které dokáží archiv rozbalit. Všechny tyto požadavky splňuje kombinace programů *tar* a *gzip*.

Souhrnná konfigurace systému bude uložena v archivu s názvem `gentoo_conf.tar.gz` (viz. 5.1).

[2] Uniform Resource Locator



Obrázek 5.1: Ukázka možné adresářové struktury archivu s uloženou konfigurací

5.2 Automatická instalace a konfigurace systému

5.2.1 Manuální části instalace

Je zřejmé, že ne všechno lze provést automaticky bez komplikací, či automaticky nelze provést vůbec.

Nastavení sítě je jednou z těchto akcí. Samozřejmě, existují nástroje jako *net-setup*. Nicméně v případě, že uživatel bude pro připojení do sítě Internet používat, řekněme, netypický ADSL modem, který k zprovoznění vyžaduje zásahy do jádra, např. formou záplat[3], bude automatizace nastavení sítě pro většinu zařízení téměř nemožná. Bude vhodné předpokládat, že si uživatel síť nastaví sám.

Další, a poslední akcí, kterou bude muset uživatel provést před samotnou instalací, je vytvoření oddílů na pevném disku, instalace souborových systémů, jejich připojení a nastavení jejich automatického připojení v souboru `/etc/fstab`. Toto automatizovat by šlo jen velmi těžce, jestli vůbec, protože každý uživatel má svou osobní představu jakým způsobem oddíly na pevném disku nastavit.

5.2.2 Stažení a instalace systémových souborů

Stažení aktuálního stromu Portage bude probíhat bez problému, protože jeho archiv se jmenuje vždy stejně, `portage-latest.tar.bz2`. Toto neplatí u `stage3` archivu, který se liší v názvu architekturou a datem vydání, např. `stage3-amd64-20100101.tar.bz2`. Bude nutné najít pomocí regulárního výrazu, ve zdrojovém kódu HTML[4] stránky obsahující odkaz na `stage3` archiv, jeho název. `Stage3` archiv se rozbalí do kořenového adresáře systému a archiv s Portage stromem pak do adresáře, jehož jméno je uloženo v proměnné `PORTDIR`.

5.2.3 Instalace konfigurace systému

Ve chvíli kdy máme na disku základní systém, můžeme zkopírovat konfiguraci ze zdrojové instalace, uloženou v archivu `gentoo_conf.tar.gz`, do kořenového adressáře systému. Vytvoříme adresáře uživatelů v adresáři `/home`, jejichž jména jsou uložena v proměnné `HOMES` konfiguračního souboru `gentoo.conf` (viz. 5.1.5). Pokud se zdrojová a cílová architektura liší, musíme změnit názvy `keyword` v proměnné `ACCEPT_KEYWORD` a v souboru, resp. adresáři `/etc/portage/package.keywords` (viz. 3.6).

[3] angl. patch
[4] HyperText Markup Language

5.2.4 Nastavení voleb pro kompilaci

Procesor zdrojového a cílového systému se může, a často bude, lišit. Proto je nutné nastavit volby pro kompilaci podle daného procesoru. Takové volby jsou `CFLAGS`, nastavení kompilátoru jazyka C, `CHOST`, která definuje architekturu, pro kterou má kompilátor překládat a konečně `MAKEOPTS` určující optimalizace překladu, jejíž hodnota by měla být (viz. [17]) o jednu větší než počet jader procesorů v počítači.

Všechny tyto volby se dají nalézt, resp. odvodit ze souboru `/proc/cpuinfo`:

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 15
model name      : Intel(R) Core(TM)2 Duo CPU T7300 @ 2.00GHz
stepping        : 10
cpu MHz         : 800.000
cache size      : 4096 KB
...
flags           : fpu vme de pse tsc msr pae... sacpi mmx fxsr sse sse2 ss...
...
```

Příklad 5.2: Část souboru /proc/cpuinfo

Každé jádro, resp. procesor má jeden záznam. Spočítáním těchto záznamů a přičtení jedničky dostaneme hodnotu proměnné `MAKEOPTS`.

Webová stránka (viz. [5]) poskytuje seznam funkčních kombinací `CFLAGS` a `CHOST` pro různé procesory. Vytvořením XML[5] souboru obsahující informace o procesoru a jejich `CFLAGS` a `CHOST` (viz. 5.3) můžeme tyto proměnné nastavit vyhledáním informací `cpu_family`, `vendor_id`, `model` a `model name` ze souboru `/proc/cpuinfo` ve vytvořeném XML souboru. Pokud `CFLAGS` nebo `CHOST` nebudou nalezeny, uživatel budou dotázán, aby je zadal ručně.

Soubor `/proc/cpuinfo` využijeme také pro doplnění hardwarově závislých `USE` flags. Protože známe seznam dostupných `USE` flags (viz. 3.5), můžeme zkontrolovat zda existuje nějaký, který odpovídá vlastnosti procesoru uvedených na řádku začínajícím na `flags:` v souboru `/proc/cpuinfo` (viz. 5.2).

5.2.5 Nastavení systémového profilu

Systémový profil nebude nutné měnit, pokud se architektura zdrojové a cílové instalace nebude lišit. Pokud se však lišit bude, bude nutné zaměnit původní `keyword` v cestě k profilu (viz. 5.4) za ten z cílové instalace.

5.2.6 Stažení a instalace systémových balíčků

V této sekci jako systémový balíček budeme považovat balíček, který je nutné mít nainstalovaný v systému ještě před samotnou kompilací jádra. Mezi takové balíčky patří samotné jádro, případně `sys-kernel/genkernel` pro automatickou kompilaci jádra. Dalším systémovým balíčkem může být `app-portage/layman` pro automatickou správu `overlays`,

[5] Extensible Markup Language

```
<root>
    ...
    <AuthenticAMD>
        ...
        <cpu>
            <cpu_family>6</cpu_family>
            <model>4</model>
            <model_name>AMD Athlon(tm) Processor</model_name>
            <chost>i686-pc-linux-gnu</chost>
            <cflags>-march=athlon-tbird -O2 -pipe -fomit-frame-pointer</cflags>
        </cpu>
        ...
    </AuthenticAMD>
    ...
    <GenuineIntel>
        ...
        <cpu>
            <cpu_family>6</cpu_family>
            <model>15</model>
            <model_name>Intel(R) Core(TM)2 Duo CPU</model_name>
            <bit32>
                <chost>i686-pc-linux-gnu</chost>
                <cflags>-march=prescott -O2 -pipe -fomit-frame-pointer</cflags>
            </bit32>
            <bit64>
                <chost>x86_64-pc-linux-gnu</chost>
                <cflags>-march=nocona -O2 -pipe</cflags>
            </bit64>
        </cpu>
        ...
    </GenuineIntel>
    ...
</root>
```

Příklad 5.3: XML soubor pro vyhledávání voleb pro kompilaci

```
/usr/portage/profiles/default/linux/amd64/10.0
```

Příklad 5.4: Ukázka cesty k systémovému profilu

protože i samotné jádro může mít uživatel nainstalované z jiné `overlay` než z oficiálního Portage stromu. O tom jaké verze systémových balíčků se budou instalovat rozhodne záznam v souboru `packages` (viz. 5.1.4). Výjimkou může být balíček `sys-kernel/genkernel`, který nemusí být ve zdrojové instalaci přítomen a tak se zvolí jeho nejaktuálnější verze, samozřejmě s ohledem na `keywords` (viz. 3.6). Toto může nastat, když ve zdrojové instalaci bylo jádro zkompilováno ručně, ale při instalaci na nový systém byla zvolena možnost automatické kompilace jádra.

5.2.7 Kompilace jádra

V případě, že přenášíme instalaci na počítač se stejnou hardwarovou konfigurací můžeme využít soubor s konfigurací jádra `.config` pro jeho kompilaci. Pokud ovšem verze jádra ze zdrojové instalace byla z hlavního Portage stromu, případně jiné `overlay`, odstraněna, nainstaluje se nejnovější dostupná verze a je nutné automaticky odsouhlasit implicitní hodnoty (`make oldconfig`) v nově přidaných možnostech konfigurace.

Je velice pravděpodobné, že ve zdrojové instalaci bude instalováno více jader. V takovém případě se přenesou konfigurace (`.config`) všech jader. Gentoo Linux vyžaduje (viz. [3]), aby symbolický odkaz `/usr/src/linux` ukazoval na adresář se zdrojovými kódy aktuálně běžícího jádra. Tuto informaci využijeme k tomu, aby skript věděl, které jádro zkompilovat, zatímco instalace zdrojových kódů bude provedena u všech verzí ze zdrojové instalace.

Jestliže byla zvolena možnost instalace jádra pomocí balíčku `sys-kernel/genkernel`, je nejdříve nutné odstranit ze souboru `/etc/modules.autoload.d/kernel-2.6` automaticky nahrávané moduly při startu systému, protože již nemusí být v cílové instalaci dostupné.

5.2.8 Stažení a instalace balíčků

Balíčky instalované ve zdrojové instalaci jsou uloženy v souboru `packages` (viz. 5.1.4). Ještě před jejich samotnou instalací je nutné provést dvě akce. Musíme zkontrolovat, zda je balíček stále dostupný v hlavním Portage stromu, resp. jiné `overlay`. Pokud není, vyhledá se nejnovější verze dle systémových nastavení, tedy `keywords` (viz. 3.6). Může se stát, že žádný novější balíček neexistuje nebo existuje, ale je maskován. V takovém případě se daný balíček neinstaluje. Toto však může mít za následek selhání instalace, protože jiný balíček může ten nedostupný vyžadovat. Další příklad selhání instalace může být, když uživatel nainstaluje balíček, který je poté zamaskován vývojáři. Je proto doporučeno přenášet aktualizovaný systém.

Dále se pak zkontroluje, zda je balíček v souboru `/var/lib/portage/world`, pokud ne, instaluje se s parametrem `--oneshot` nástroje *emerge*, abychom zabránili jeho přidání do `world` a zachovali tak konfiguraci zdrojové instalace.

Systém Portage se pokusí nahrát (viz. 3.9) implicitní verze konfiguračních souborů. Skript však konfigurační soubory nahrál ze zdrojové instalace. Implicitní konfigurační soubory budou pojmenovány ve tvaru `._cfg0000_<skutečné_jméno_souboru>` (viz. 3.9), ty smažeme.

Jestliže dojde k situaci, ve které některý z balíčků nebyl instalován, např. z důvodu jeho nedostupnosti na cílovém systému, a je o něm uložen záznam v souboru `world`, musí být tento záznam odstraněn. K tomu využijeme nástroj *emaint*, který je součástí Portage. Spustíme jej následovně: `emaint --fix world`.

Pokud zdrojová instalace obsahuje starší verze balíčků, než jsou obsaženy ve `stage3` archivu, dojde k instalaci nižší verze těchto balíčků a některé z nich mohou způsobit problémy.

```
default 0
timeout 5

title Gentoo Linux 2.6.30-r5
root (hd0,0)
kernel /boot/kernel-2.6.30-gentoo-r5 root=/dev/sda3
```

Příklad 5.5: Ukázka konfigurace GRUBu pro ruční kompilaci jádra

```
default 0
timeout 5

title Gentoo Linux 2.6.30-r5
root (hd0,0)
kernel /boot/kernel-gen root=/dev/ram0 init=/linuxrc ramdisk=8192 real_root=/dev/sda3
initrd /boot/initramfs-gen
```

Příklad 5.6: Ukázka konfigurace GRUBu při použití `sys-kernel/genkernel`

Některé balíčky, jako např. `sys-devel/gcc`, a změna jejich verze může způsobit nekompatibilitu či nefunkčnost systému. Jsou však vždy instalovány do nového `SLOT` (viz. 3.10) a tak zde nebezpečí nehrozí. Problém však nastane u knihovny jazyka C, balíček `sys-libs/glibc`. Po změně její verze většina balíčků, dynamicky linkovaných s touto knihovnou, přestane fungovat. Následný proces je problematický ([1], [6]). Jako rozšíření bychom tento problém mohli vyřešit načtením, před začátkem instalace balíčků, všech balíčků obsažených ve `stage3` a ty pak neinstalovat. Ztratili bychom však jistotu, že cílová instalace bude odpovídat instalaci zdrojové. Dalším možným řešením do budoucna by bylo zautomatizovat změnu verze takto problematických balíčků.

5.2.9 Instalace zavaděče

Abychom byli schopni nainstalovat zavaděč GRUB, musíme mít k dispozici informace, na jakém oddílu je kořenový adresář systému, jak se jmenuje soubor s jádrem a na jakém oddílu je umístěn adresář `/boot`. Jména potřebných oddílů získáme ze souboru `/etc/fstab`. Jméno souboru s jádrem známe, protože jsme jej po kompilaci sami pojmenovali. Musíme si také uvědomit, že jestliže použijeme `genkernel` (viz. 5.6) pro kompilaci jádra, bude se konfigurace GRUBu lišit od ruční kompilace (viz. 5.5).

V případě, že uživatel nebude chtít instalovat zavaděč přímo do MBR[6], bude vhodné, aby toto nástroj umožňoval.

[6] Master Boot Record

Kapitola 6

Implementace

6.1 Volba implementačního jazyka

Pro implementaci byl vybrán jazyk Python [16], který obsahuje rozsáhlou a velmi užitečnou standardní knihovnu [15], která umožňuje se při vývoji soustředit přímo na daný problém a pro rutinní záležitosti využít právě možnosti této knihovny. Je to skriptovací jazyk umožňující rychlý vývoj a v neposlední řadě velká část systému Portage je právě v jazyce Python napsána, což umožňuje využití funkcí systému Portage [9]. Dalším důvodem je snadná dostupnost v instalačním prostředí, např. Live CD, která interpret jazyka Python obsahují právě kvůli Portage. V době psaní této práce byla v distribuci Gentoo Linux označena verze `2.6.4-r1` jako stabilní. Proto byl právě balíček `dev-lang/python-2.6.4-r1` zvolen pro vývoj nástrojů.

6.2 Interakce s uživatelem

Instalační CD distribuce Gentoo linux je dostupné, a často aktualizované, především v podobě minimálního instalačního CD, tedy bez jakéhokoliv grafického prostředí. Instalace této distribuce probíhá výhradně v prostředí příkazové řádky. Interakce s uživatelem bude tedy probíhat v příkazové řádce přes argumenty jednotlivých skriptů. Grafická uživatelská rozhraní by bylo možné implementovat v budoucnu, avšak jeho využití by bylo minimální a na uživatelském komfortu by nepřidalo.

6.3 Modul `aiclib`

Modul `aiclib` je nespustitelný a poskytuje pomocné funkce pro ostatní moduly. V této sekci budou popsány jeho zajímavé části.

6.3.1 Správa konfiguračního souboru `gentoo.conf`

Konfigurační soubor `gentoo.conf` obsahuje informace o zdrojovém systému, které jsou nutné pro jeho přenesení (viz. 5.1.5). Modul `aiclib` obsahuje funkce `save_conf` a `get_conf` pro jeho správu. První jmenovaná funkce vezme pole, naplňené získanými informacemi, předané parametrem a jeho prvky zapíše do souboru `gentoo.conf`. Funkce `get_conf` pak analogicky načte informace ze souboru `gentoo.conf` a vrátí je v poli.

6.3.2 Správa stavu sbírání konfigurace a instalace

Procesy sbírání konfigurace a samotná instalace je časově náročná. Dojde-li k chybě nebo uživatel jeden z těchto procesů přeruší, bylo by zbytečné celý proces opakovat. Pro tyto případy obsahuje modul `aiclib` funkce `set_lastrun` a `get_lastrun`. Identifikační číslo naposledy dokončené akce je uloženo v souboru `lastrun`. Konstanty identifikující jednotlivé akce jsou uloženy v modulech, které tyto akce vykonávají. Např. modul `aicfetch` obsahuje konstantu `STATE_CONF`, která identifikuje akci sbírání konfiguračních souborů systému. Funkce `set_lastrun` tedy uloží identifikační číslo akce, předané parametrem funkce, do souboru `lastrun`, zatímco funkce `get_lastrun` jej z tohoto souboru přečte a vrátí. Příslušný modul pak na základě návratové hodnoty funkce `get_lastrun` rozhodne, zda má následující akci vykonat či má pokračovat akcí následující. Když danou akci vykoná, zavolá funkci `set_lastrun` a předá jí identifikační číslo této akce.

6.3.3 Výpis chybových hlášení

Nastane-li chybový stav, např. nepodaří se otevřít soubor, je nutné takovou informaci oznámit uživateli. Modul `aiclib` poskytuje funkce `print_info` a `print_err`. Prvně zmiňovaná funkce slouží k vypsání informace o stavu aplikace na standardní výstup, která má pro uživatele význam orientační, např. jakou akci aplikace právě vykonává. Funkce `print_err` vypíše na standardní chybový výstup zprávu o chybě a aplikace se ukončí.

6.3.4 Operace s balíčky

Funkce `find_installed_version` slouží k nalezení verze balíčku instalovaného ve zdrojovém systému. Funkce prohledá soubor s balíčky `packages` (viz. 5.1.4), které byly ve zdrojovém systému instalovány a vrátí všechny instalované verze balíčku v poli. Tato funkce je důležitá, chceme-li instalovat balíček ještě před instalací všech balíčků ze zdrojové instalace (viz. 5.2.6).

Funkce `get_available_package` slouží pro nalezení nejnovější dostupné verze podle `keywords` (viz. 3.6). Tato funkce pro svůj výsledek využívá výpisu nástroje *emerge*, s přepínačem `-s`, který slouží pro vyhledávání (viz. 6.1).

```
$ emerge -s gentoo-sources
Searching...
[ Results for search key : gentoo-sources ]
[ Applications found : 1 ]

* sys-kernel/gentoo-sources
      Latest version available: 2.6.30-r5
      Latest version installed: 2.6.30-r5
      Size of files:    58,140 kB
      Homepage:         http://dev.gentoo.org/~mpagano/genpatches
      Description:      Full sources including the Gentoo patchset for the 2.6 kernel tree
      License:          GPL-2
```

Příklad 6.1: Výstup hledaní pomocí `emerge -s`

6.3.5 Převod architektury na `keyword`

Zkratku hardwarové architektury získáme příkazem `uname -m`, v jazyce Python zavoláním `platform.machine()`. Výsledkem těchto volání může být např. `x86`, `x86_64` či `sparc`. Ne vždy se však toto shoduje s tzv. `keyword` (viz. 3.6). Funkce `get_arch` převádí hardwarovou zkratku na `keyword`, tedy např. `x86_64` převede na `amd64`. Tuto funkci využijeme při přenášení instalace na odlišnou architekturu od té zdrojové.

6.4 Modul `aicfetch`

Modul `aicfetch` slouží jako nástroj pro získání konfigurace systému. Při jeho spuštění probíhá ukládání konfiguračních souborů. Jeho výstupem je archiv `gentoo_conf.tar.gz` (viz. 5.1.6).

Nejdříve se získá identifikační číslo naposledy vykonané akce (viz. 6.3.2), pak se provedou zbylé akce, případně všechny. První akcí je uložení adresářů v systémové proměnné `CONFIG_PROTECT` (viz. 3.9 a 5.1.1).

Dále pak, pokud si tak uživatel zvolil, proběhne uložení konfiguračních souborů aplikací z domovských adresářů uživatelů (viz. 5.1.2).

Další volitelnou akcí je zkopírování uživatelských `overlays` spolu s konfiguračním souborem `/usr/local/portage/layman/make.conf`, resp. `/var/lib/layman/make.conf` balíčku `app-portage/layman`, pokud je instalován (viz. 3.7 a 5.1.3).

Následuje uložení názvů balíčků nainstalovaných v systému do souboru `packages` (6.2). Informace o instalovaných balíčcích jsou v adresáři `/var/db/pkg` (viz. 3.3 a 5.1.4).

```
app-admin/apache-tools-2.2.14
app-admin/eselect-1.2.9
app-admin/eselect-boost-0.3
...
```

Příklad 6.2: Formát souboru `packages`

Ještě před zabalením uložených souborů do výsledného archivu se vytvoří konfigurační soubor `gentoo.conf` (viz. 5.1.5)

6.5 Modul `aicinstall`

Modul `aicinstall` slouží jako nástroj pro automatickou instalaci a konfiguraci systému Gentoo Linux. Jeho vstupem je archiv `gentoo_conf.tar.gz`, vytvořený nástrojem `aicfetch`, a výstupem nainstalovaný systém.

6.5.1 Příprava instalačních souborů

Nejdříve se získá identifikační číslo naposledy vykonané akce (viz. 6.3.2), pak se provedou zbylé akce, případně všechny. První akcí je zavolání funkce `unpack_conf`, která rozbalí archiv `gentoo_conf.tar.gz` s konfigurací do budoucího kořenového adresáře systému.

Následuje pak stažení archivu s aktuálním Portage stromem a základního Gentoo systému, archivu `stage3`. Stažení těchto archivů zprostředkovávají funkce `download_portage`,

resp. `download_stage3`. Zavoláním funkcí `install_portage` a `install_stage3` dojde k rozbalení těchto archivů do budoucího kořenového adresáře systému. (viz. 5.2.2)

Dalšími akcemi je pak vytvoření domovských adresářů uživatelů, funkcí `create_homes` a zkopírování konfiguračních souborů do `CONFIG_PROTECT` adresářů, funkcí `copy_conf`. V případě, že architektura cílové instalace se bude lišit od té zdrojové je nutné změnit `keywords`, což umožňuje funkce `reset_keyword`. (viz. 5.2.3)

6.5.2 Instalace

Ještě před samotnou instalací je nutné vstoupit do nového prostředí pomocí programu `chroot`. Protože vstupujeme do nového prostředí přímo ze skriptu v jazyce Python, musíme aktualizovat proměnnou `sys.path` sami. V této proměnné jsou uchovány cesty k modulům. My do ni přidáme cestu `/usr/lib/portage/pym`, kde se nachází modul `portage`.

Nejdříve se nastaví proměnné potřebné pro kompilaci, tedy `CFLAGS`, `CHOST`, `MAKEOPTS` a `USE` v konfiguračním souboru `/etc/make.conf` podle hardwaru, na který se systém instaluje. Hodnoty těchto proměnných jsou k nalezení v souborech `cflags.xml` a `/proc/cpuinfo` (viz. 5.2.4). Tato nastavení se provedou po zavolání funkcí `set_cflags` a `set_use_flags`.

Funkce `set_profile` se provede pouze pokud se architektura zdrojové a cílové instalace liší. Nastavení systemového profilu se provede záměnou `keyword` v cestě k profilu (viz. 5.2.5).

Následuje synchronizace Portage stromu spuštěním `emerge --sync`, případně pokud si tak uživatel zvolí `emerge-webrsync`. Pokud je ve zdrojové instalaci přítomen balíček `app-portage/layman`, nainstaluje se a spustí se `layman -f`, synchronizace všech `overlays`.

Pokud se uživatel rozhodne mít celý systém instalovaný pro svůj hardware, instalace pokračuje kompilací balíčků ze `stage3` (viz. 4.5 a [11]).

Dalším krokem je volání funkce `emerge_syspkg`, která nainstaluje systémové balíčky (viz. 5.2.6), které jsou především nutné pro kompilaci jádra operačního systému.

Následuje kompilace jádra operačního systému. Podle uživatelem zvolené možnosti se zavolá funkce `install_kernel` či `install_genkernel`. V rámci `install_kernel` je mj. nutné zavolat `make oldconfig` a nástrojem *yes* nastavit implicitní hodnoty v případě, že verze jádra ze zdrojové instalace není ve stromu Portage dostupná. (viz. 5.2.7)

Důležitou částí je samotná instalace balíčků ze zdrojové instalace, jejichž jména a verze jsou uloženy v souboru `packages` (viz. 5.1.4). Ještě před instalací balíčku se pomocí funkce `check_package` zkontroluje, zda je balíček přítomen ve stromu Portage. Je zcela běžné, že vývojáři balíček odstraní. Funkce `check_package` používá funkce `pkg_in_system`, která hledá v `PORTDIR` a jiných overlays ebuild pro daný balíček a `get_available_package` (viz. 6.3.4) v případě, že funkce `pkg_in_system` daný balíček nenašla. Pokud je balíček, např., na zdrojové architektuře označen jako stabilní, může se stát, že na cílové architektuře bude maskován. Toto řeší funkce `is_masked`, která se podívá do souboru `.ebuild` daného balíčku, zda je dostupný. Instalaci balíčků provádí funkce `install_packages`.

Další volitelnou akcí je odstranění již nepotřebných balíčků a překompilování balíčků, které závisely na těch odstraněných, `emerge --depclean && revdep-rebuild` (viz. 3.4). Uživatel bude upozorněn, že pro spuštění nástroje *revdep-rebuild* je potřeba instalace balíčku `app-portage/gentoolkit`. Pokud je tento balíček přítomen ve zdrojovém systému, je instalována právě tato verze, pokud není, je instalována aktuální verze podle `keywords` (viz. 3.6).

Posledním krokem k dokončení instalace je zavolání funkce `install_grub`, která provede instalaci zavaděče. Jeho konfiguraci provede na základě souboru `/etc/fstab`, kde jsou potřebné informace o oddílech na pevném disku. (viz. 5.2.9).

Kapitola 7

Testování a dosažené výsledky

7.1 Náročnost testování

Testování nástrojů pro automatickou instalaci a konfiguraci operačního systému Gentoo Linux je náročné hned z několika důvodů. Většina instalací obsahuje několik stovek balíčků. Každý z nich může způsobit specifické problémy. Především pracovní stanice mají instalovány X server a grafický okenní manažer, které zkompilovat trvá i několik hodin. Další aspekt, který testování nástrojů jako celek komplikuje, je nedostupnost prostředků pro instalaci. Není snadné dostat se alespoň k několika architekturám, které Gentoo Linux podporuje.

7.2 Specifikace testů

Kromě obligátního testování jednotlivých funkcí nástrojů samostatně, byly navrženy dva testy pro přenesení instalace:

1. přenesení instalace z architektury `x86` na architekturu `x86_64`, s odlišnou hardwarovou konfigurací,
2. přenesení instalace z architektury `x86` na architekturu stejnou, se stejnou hardwarovou konfigurací.

7.3 Prostředí testování

Testy byly provedeny na dvou sestavách, jejichž parametry jsou uvedeny v následující tabulce:

Architektura	x86	x86_64
Procesor	AMD Athlon 1.1 GHz	Intel Core 2 Duo 2.0 GHz
Operační paměť	256 MB	2 GB
Pevný disk	30 GB, 7200 ot./s	120 GB, 5400 ot./s
Grafická karta	nVidia MX 200	VirtualBox
Poznámka	–	Instalováno ve VirtualBox

Tabulka 7.1: Testovací sestavy

7.4 Test: Přenesení instalace na jinou architekturu

V tomto testu ověříme funkčnost automatické kompilace jádra operačního systému pomocí balíčku `sys-kernel/genkernel`, změnu `keywords`, přenastavení systémového profilu či odstranění modulů jádra, které se načítají při zavádění systému. Je zřejmé, že nastanou problémy s funkčností odlišného hardwaru. Instalace proběhla spuštěním `aicinstall.py -g -a 64`, tedy použili jsme `sys-kernel/genkernel` a instalovali jsme 64 bitový systém.

7.4.1 Výsledek

Při automatické instalaci a konfiguraci byly úspěšně nahrazeny balíčky, které na zdrojové platformě dostupné byly, na cílové však nikoliv. Systém Portage automaticky nainstaloval balíčky z kategorie `app-emulation`, které nabízejí předkompilované 32 bitové knihovny.

Jak se předpokládalo, nastaly i problémy. Balíček `sys-apps/coldplug`, načítající automaticky moduly při startu systému, se nepodařilo nainstalovat. Tento balíček nebyl přítomen ve zdrojové instalaci a je instalován se zvolením automatické kompilace jádra [17]. Důvodem bylo blokování od balíčku `sys-fs/udev`.

Dále pak, podle předpokladu, nenaběhl X server a tedy i grafický okenní manažer. Toto bylo způsobeno odlišnou konfigurací grafické karty na cílovém počítači. Snadné řešení takového problému by byla editace proměnné `VIDEO_CARDS`, která říká jaké ovladače grafické karty se mají nainstalovat. Editace proměnných `USE` či `VIDEO_CARDS` by proběhla v případné fázi editace uložené konfigurace systému. S tím také souvisí poslední problém, neúspěšná instalace balíčku `media-video/nvidia-settings`.

7.5 Test: Přenesení instalace na stejnou architekturu

V tomto testu ověříme kompilaci jádra operačního systému pomocí programu `make` a konfigurace uložené ze zdrojového systému. Dále si pak ověříme předpoklad, že by během instalace neměly nastat problémy, a že by cílový systém měl být téměř identický jako ten zdrojový, protože instalujeme na totožný hardware. Instalace proběhla spuštěním `aicinstall.py -c -a 32`, tedy na konci kompilace proběhne `emerge --depclean && revdep-rebuild` (viz. 3.4) a instalovali jsme 32 bitový systém.

7.5.1 Výsledek

Při automatické instalaci a konfiguraci byly úspěšně odstraněny balíčky, které již nejsou ve stromu Portage. Předpoklad, že by instalace měla proběhnout bez problémů se potvrdil, systém naběhl a vše funguje jako na zdrojové instalaci. Jediným problémem se ukázala nemožnost, jako v prvním testu, instalace balíčku `media-video/nvidia-settings`. Přerušení instalace zbylých balíčků, selháním instalace `media-video/nvidia-settings`, se vyřešila vypsáním si již instalovaných balíčků a balíčků ze souboru `packages` a následná instalace jejich rozdílu.

Kapitola 8

Další možný vývoj

Do budoucna by bylo vhodné implementovat podporu i jiných zavaděčů než velmi rozšířeného GRUB, jako např. LILO (`sys-boot/lilo`) a další. Postup automatizace jiného zavaděče by byl analogický. Odvození zaváděcího a kořenového oddílu je již implementováno, stačilo by tedy spustit instalaci zavaděče a vytvořit a patřičně konfigurovat konfigurační soubor. (viz. 5.2.9)

Když se uživatel rozhodne přenášet instalaci na počítač, který nedisponuje určitým hardwarem (např. bluetooth), bylo by vhodné mít nástroj umožňující odstranění balíčku, který by daný hardware využíval. Tento nástroj by interaktivně umožňoval balíček odebrat ze souboru `packages` (viz. 5.1.4) a vyřešil by, pomocí Portage, závislosti. Ebuildy totiž neumožňují způsob označení balíčku jako hardwarově závislý a tak automatizace této akce je téměř nemožná.

Ve stromu Portage existují jádra speciálně upravená pro určitou architekturu, jako např. MIPS (`sys-kernel/mips-sources`) či SPARC (`sys-kernel/sparc-sources`[1]). V případě, že by se na takovou architekturu systém přenášel, bylo by vhodné uživateli nabídnout jádro specializované pro tuto architekturu. Současné řešení instaluje vždy balíček s jádrem `sys-kernel/gentoo-sources`, který je podporován, a tedy upraven, pro všechny architektury, které distribuce Gentoo Linux podporuje. Toto rozšíření by tedy nemělo za následek velké zvýšení použitelnosti nástrojů pro automatickou instalaci a konfiguraci distribuce Gentoo Linux.

S přibývajícími možnostmi konfigurace nástrojů by bylo vhodné nahradit jejich argumenty konfiguračními soubory. V těchto souborech by uživatel specifikoval způsob instalace a mohl by je využívat opakovaně.

[1] Tento balíček však není delší dobu podporován.

Kapitola 9

Závěr

Tato práce se zabývala automatickou instalací a konfigurací operačního systému GNU/ Linux, distribuce Gentoo. Byl popsán balíčkovací systém Portage a současný postup instalace a konfigurace této distribuce. Dále byly navrženy a implementovány nástroje, které automatickou instalaci a konfiguraci provedou. Nakonec byly nástroje otestovány a byl diskutován další možný vývoj nástrojů do budoucna.

Testování ukázalo, že nástroje zvládnou přenést instalaci na stejnou architekturu. Při přenášení systému na jinou architekturu je potřeba provést více akcí, nicméně samotná instalace proběhne, dle testování, v pořádku.

Navržené nástroje trpí i nedostatky. Instalujeme-li systém na jinou hardwarovou konfiguraci, není možné zjistit, zda je balíček hardwarově závislý. Takový balíček pak bude na cílový systém instalován zbytečně. Řešení takového problému není obtížné, ale bude nutný zásah uživatele. Řešení bude spočívat v editaci uložené konfigurace, typicky obsah proměnných `USE` a `VIDEO_CARDS`.

Vyvinuté nástroje byly zveřejněny jako projekt s otevřeným zdrojovým kódem a jsou přístupné ostatním uživatelům na adrese http://code.google.com/p/gentoo-aic/.

Literatura

[1] COIE, R.: *Gentoo 1.4 Upgrade Guide* [online]. July 2, 2005 [cit. 2010-04-09].
URL http://www.gentoo.org/doc/en/new-upgrade-to-gentoo-1.4.xml

[2] Debian Project: *Debian Releases* [online]. February 12, 2010 [cit. 2010-02-20].
URL http://www.debian.org/releases/index.en.html

[3] DRAKE, D.: *Gentoo Linux Kernel Upgrade Guide* [online]. June 22, 2007 [cit. 2010-03-28].
URL http://www.gentoo.org/doc/en/kernel-upgrade.xml

[4] Gentoo Foundation, Inc.: *About Gentoo* [online]. September 17, 2007 [cit. 2010-02-20].
URL http://www.gentoo.org/main/en/about.xml

[5] Gentoo Linux Wiki: *Safe Cflags* [online]. March 26 2010 [cit. 2010-03-28].
URL http://en.gentoo-wiki.com/wiki/Safe_Cflags

[6] Gentoo Linux Wiki: *Downgrade Glibc* [online]. March 3, 2009 [cit. 2010-04-09].
URL http://en.gentoo-wiki.com/wiki/Downgrade_Glibc

[7] Kolektiv: *Gentoo Handbook česky*. 2007.
URL http://www.root.cz/knihy/gentoo-handbook-cesky/

[8] KOTĚŠOVEC, J.: *Poznejte své Gentoo* [online]. 6. 12. 2004 [cit. 2010-02-20].
URL http://www.root.cz/clanky/poznejte-sve-gentoo/

[9] MAUCH, M., et al.: *Gentoo Linux Portage Development* [online]. March 24, 2010 [cit. 2010-04-11].
URL http://www.gentoo.org/proj/en/portage/

[10] RAYMOND, E. S.: *Umění programování v UNIXu*. Computer Press, Brno, 2004, ISBN 80-251-0225-4.

[11] ROBBINS, D.: *Gentoo Linux Frequently Asked Questions* [online]. October 25, 2008 [cit. 2010-03-28].
URL http://www.gentoo.org/doc/en/faq.xml#stage12

[12] ROBBINS, D.: *Making the distribution, Part 1* [online]. October 9, 2005 [cit. 2010-05-06].
URL http://www.gentoo.org/doc/en/articles/making-the-distro-p1.xml

[13] ROBBINS, D.: *Making the distribution, Part 2* [online]. October 9, 2005 [cit. 2010-05-06].
URL http://www.gentoo.org/doc/en/articles/making-the-distro-p2.xml

[14] ROBBINS, D.: *Making the distribution, Part 3* [online]. October 9, 2005 [cit. 2010-05-06].
URL http://www.gentoo.org/doc/en/articles/making-the-distro-p3.xml

[15] ROSSUM, G. V.: *The Python Standard Library* [online]. April 10, 2010 [cit. 2010-04-11].
URL http://docs.python.org/library/

[16] ROSSUM, G. V.: *The Python Tutorial* [online]. April 10, 2010 [cit. 2010-04-11].
URL http://docs.python.org/tutorial/

[17] VERMEULEN, S.: *Gentoo Handbook* [online]. June 14, 2009 [cit. 2010-03-28].
URL http://www.gentoo.org/doc/en/handbook/

[18] VERMEULEN, S., et al.: *Additional Portage Tools* [online]. December 29, 2009 [cit. 2010-02-20].
URL
http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=3&chap=4

[19] VERMEULEN, S., et al.: *A Portage Introduction* [online]. March 2, 2010 [cit. 2010-03-10].
URL
http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=2&chap=1

Příloha A

Obsah CD

`./doc` – programová dokumentace
`./ibp` – text bakalářské práce ve formátu PDF
`./src` – zdrojové kódy nástrojů
`./tex` – LaTeX zdrojové soubory práce

Příloha B

Manual

B.1 aicfetch.py

Použití: aicfetch.py [PARAMETRY]

Parametry:

-h, --help	Zobrazí nápovědu
-v, --verbose	Vypíše informace informace za běhu programu
-n, --new	Nenaváže na předešlé sbírání konfigurace
-o, --overlays	Uloží uživatelské `overlays`
-u, --userconf	Uloží konfigurační soubory aplikací z domovských adresářů uživatelů

B.2 aicinstall.py

Použití: aicinstall.py [PARAMETRY]

Parametry:

-h, --help	Zobrazí nápovědu
-v, --verbose	Vypíše informace informace za běhu programu
-n, --new	Nenaváže na předešlou instalaci
-c, --clean	Spustí emerge --depclean && revdep-rebuild po instalaci systému (!!! vyžaduje emerge app-portage/gentoolkit !!!)
-g, --genkernel	Použije sys-kernel/genkernel pro kompilaci jádra
-w, --webrsync	Použije emerge-webrsync místo emerge --sync to sync the portage tree
-a A, --arch=A	Nainstaluje 32 nebo 64 bitový systém
-b B, --boot=B	Umístí GRUB jinde než do MBR (Master Boot Record), např.: ”hd0,1”
-s, --bootstrap	Provede bootstrap systému

B.3 Přenesení instalace

Pro bezproblémovou funkčnost nástrojů je nutné je spouštět s interpretem jazyka Python ve verzi 2.6.4 a vyšší.

- Na zdrojovém počítači:
  - spustíme skript `aicfetch.py` (vyžaduje skript `aiclib.py`), který vygeneruje archiv `gentoo_conf.tar.gz`.
- Na cílovém počítači:
  - zavedeme systém z Gentoo Linux instalačního CD[1] či zahájíme instalaci z již instalovaného systému,
  - nastavíme připojení do sítě Internet, např. `dhcpcd eth0`,
  - vytvoříme pomocí programu `fdisk`[2] oddíly pevného disku,
  - programem `mount` vytvořené oddíly připojíme do kořenového adresáře `/mnt/gentoo`,
  - do adresáře `/mnt/gentoo` nakopírujeme soubory `aicinstall.py`, `aiclib.py`, `cflags.xml`, archiv `gentoo_conf.tar.gz` a vytvořený soubor `fstab`[3],
  - vstoupíme do adresáře `/mnt/gentoo` příkazem `cd /mnt/gentoo`,
  - spustíme skript `aicinstall.py`.

[1] http://www.gentoo.org/main/en/where.xml
[2] http://tldp.org/HOWTO/Partition/fdisk_partitioning.html
[3] http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=1&chap=8#doc_chap1