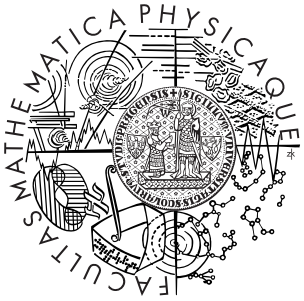
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Tomáš Benhák

Snímky webových stránek

Katedra aplikované matematiky

Vedoucí bakalářské práce: Mgr. Eva Jelínková
Studijní program: Informatika, správa počítačových systémů

2008

Děkuji paní Mgr. Evě Jelínkové za odborné vedení mé práce, za rady a za čas, který mi během vypracovávání této práce věnovala.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne Tomáš Benhák

Obsah

Název práce: Snímky webových stránek
Autor: Tomáš Benhák
Katedra (ústav): Katedra aplikované matematiky
Vedoucí bakalářské práce: Mgr. Eva Jelínková
e-mail vedoucího: Eva.Jelinkova@mff.cuni.cz

Abstrakt: Tato práce vytváří program pro Linux, který generuje snímky webových stránek bez použití okenního systému. K vykreslení webových stránek je použita komponenta GtkMozEmbed webového prohlížeče Mozilla.

Součástí práce je úprava knihovny GTK+ tak, aby grafický výstup probíhal do zvoleného souboru bez účasti okenního systému.

Title: Screenshots of Webpages
Author: Tomáš Benhák
Department: Department of Applied Mathematics
Supervisor: Mgr. Eva Jelínková
Supervisor's e-mail address: Eva.Jelinkova@mff.cuni.cz

Abstract: The aim of the thesis is to create a program for Linux that generates screenshots of webpages independently on the windowing system. The GtkMozEmbed component of the Mozilla browser is used.

A part of the thesis is a modification of the GTK+ library so that the graphic output is directed to a chosen file indepently of the windowing system.

Kapitola 1

Úvod

Webové prohlížeče zobrazují stránky na obrazovce počítače. Snímkem webové stránky je myšleno zobrazení webové stránky do obrázku.

Možností využití je mnoho. Můžeme si představit například vyhledávač (jmenovitě seznam.cz), který zobrazuje miniaturu webové stránky vedle každého odkazu.

Většina dnes existujících programů funguje pouze pod operačním systémem Microsoft Windows, avšak tato práce je věnovaná řešením pro operační systém Linux.

Pro operační systém Linux dnes existuje hned několik funkčních řešení. Všechny tyto programy mají jednu společnou vlastnost – ke své funkci nutně potřebují X server.

Tato práce má dvě části. První částí je vhodně upravit knihovnu GTK+, aby podporovala grafický výstup do obrázku. Druhou částí je vytvořit program za použití upravené knihovny GTK+, umožňující dávkově vytvářet snímky webových stránek bez interakce s uživatelem, a který pracuje bez asistence X serveru.

Program pořizující snímky webových stránek, nazvaný Webshot, je postaven na webovém prohlížeči Mozilla, konkrétně na widgetu gtkmozembed knihovny GTK+, který je součástí prohlížeče ve formě sdílené knihovny. Tento widget vykresluje stránku pomocí funkcí knihovny GTK+. Stránka není vykreslována na obrazovku, ale do obrázku, protože se k vykreslování používá upravená knihovna GTK+.

Práce je rozdělena celkem do osmi kapitol. Druhá kapitola je věnována současným možnostem pořizování snímků webových stránek. Ve třetí kapitole se seznámíme se základní terminologií. Čtvrtá kapitola je věnovaná základům knihovny GTK+, v páté kapitole budou pak popsány provedené změny. V šesté kapitole se zmíním o webovém prohlížeči Mozilla, sedmá kapitola popisuje program Webshot. V závěru budou shrnuty výsledky práce.

Kapitola 2

Možnosti vytváření snímků

V této kapitole nastíním současné možnosti vytváření snímků webových stránek, nejen pod operačním systémem Linux.

Programy pro Linux

Jak jsem se již v úvodu zmínil, pro operační systém Linux již existují programy pro vytváření snímků webových stránek. Některé z nich bych rád uvedl. Jsou jimi gnome-web-photo a khtml2png.

gnome-web-photo

Tento program umožňuje vytvářet snímky webových stránek plné délky – ne pouze viditelnou část na obrazovce. Pomocí gnome-web-photo lze vytvářet miniatury webových stránek, podporuje rovněž dávkové zpracování. Snímek dokáže uložit do obrázku různých formátů.

Ve své implementaci využívá rovněž widgetu gtkmozembed, který však není využit pro vykreslení stránky. K samotnému vykreslování je použita metoda (komponenty Presentation Shell) RenderDocument, která dokáže vykreslit stránku pomocí knihovny Cairo.

I když samotné vykreslování neprovádí X server, je přesto nutný pro funkci tohoto programu.

khtml2png

Tento program využívá k pořizování snímků knihovnu khtml. Dokáže pořídit snímky celých webových stránek, včetně možnosti vytváření miniatur. Snímky je možné uložit do různých formátů. Ke své funkci vyžaduje X server, který provádí veškeré kreslení. Velkou nevýhodou je, že grafický výstup probíhá přímo na obrazovku počítače. Tomuto efektu však lze zabránit použitím virtuálního X serveru.

Další možnosti

Programy pro MS Windows Prakticky všechna řešení pro operační systém Windows využívají k vykreslování snímků prohlížeč Internet Explorer. Mezi nejznámější patří program *WebShot*, shodou okolností používající stejný název, jako program, který je součástí této práce. Tento komerční program poskytuje velké množství funkcí, více na `http://www.websitescreenshots.com/`.

Webové systémy Kromě výše uvedeních programů je možné k pořízení snímků využít rovněž různé webové systémy, například *webshotspro.com*. Tento systém umožňuje vytvářet miniatury webových stránek, avšak není zdarma. Ke snímkování využívá již zmiňovaného programu Webshot.

Dalším, tentokrát volně použitelným, systémem je třeba *thumbalizr.com*, který generuje snímek podle parametrů zadaných v url a rovnou jej zobrazí[1]. Pokud se již v databázi snímek nachází, je ihned zobrazen, jinak je zobrazeno hlášení o tom, že byl snímek zařazen do fronty. Obdobnou službu zdarma nabízí i systém *thumbnail.cz*.

Pluginy Pro pořízení snímků jednotlivých stránek je možné využít řešení ve formě pluginů do webového prohlížeče. Pro prohlížeč Internet Explorer je to například plugin *ieSnapshotter*, pro prohlížeč Mozilla Firefox existuje plugin pod názvem *FireShot*.

[1] ukázka použití: `<img src="http://api.thumbalizr.com/?url=http://www.atlas.cz&width=250">`

Kapitola 3

Terminologie

3.1 X Window systém

X Window systém (zkráceně X) je grafický systém používaný v Unixových systémech. Jeho první verze byla vydána již roku 1984. Dnešní verze, nazývaná též X11, vyšla roku 1987.

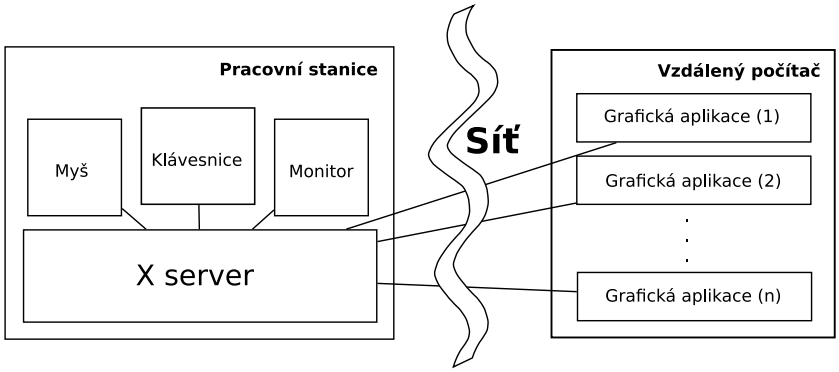
X využívá architekturu klient-server. X server je počítač, který zobrazuje grafickou část aplikace a zároveň se stará o interakci s uživatelem pomocí vstupně-výstupních zařízení. Lze si jej představit jako pracovní stanici, typicky s monitorem, myší a klávesnicí, u kterého sedí uživatel a používá klientskou aplikaci (klienta), spuštěnou na vzdáleném počítači.

Pokud na serveru nastane nějaká událost, například stisknutí klávesy na klávesnici, je o tom klient informován a může na tuto akci zareagovat. Naopak X server dává klientovi k dispozici svou obrazovku, na kterou může pomocí metod, které X server poskytuje, kreslit.

Komunikace mezi X serverem a X klientem probíhá pomocí X protokolu, typicky přes síťové sockety. Pokud klient i server běží na stejném počítači (dnes je to typické použití), komunikují pomocí lokálních socketů a v některých případech i sdílené paměti, což vykazuje vyšší výkon.

3.2 Xlib

Teoreticky by bylo možné, aby si aplikace přímo otevřela síťové spojení se serverem a komunikovala přímo s ním. V praxi se toho nevyužívá. Pro komunikaci s X serverem se používá klientská knihovna Xlib.



Obrázek 3.1: Architektura klient-server

Samotná Xlib poskytuje pouze jakési zaobalení X protokolu a umožňuje provádět jen jednoduché operace, například vytvořit okno, vykreslit přímku a zároveň umožňuje zpracovávat události zaslané X serverem, jako je pohyb myši nebo stisknutí klávesy na klávesnici.

3.3 Toolkity

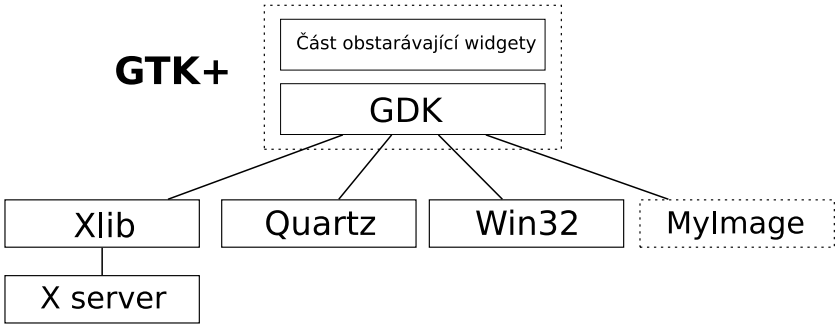
Dnešní aplikace nevyužívají knihovnu Xlib přímo – musely by se starat o mnoho detailů. Knihovna Xlib sama o sobě neposkytuje žádné typy widgetů, tedy prvků uživatelského rozhraní, jako jsou například scrollbary, tlačítka aj. Pokud by je aplikace chtěla využívat, musela by se sama postarat o jejich vykreslení pomocí funkcí Xlibu a zpracovat události s tím spojené, jako je stisknutí tlačítka myši, pohyb myši aj. Z tohoto důvodu existují nadstavby, tzv. toolkity.

Toolkity jsou knihovny poskytující jednotlivé prvky grafického uživatelského rozhraní (scrollbary, tlačítka, různá menu aj.) Ve své implementaci ovšem využívají knihovny Xlib. Velice populárním toolkitem je knihovna GTK+.

3.4 Knihovna GTK+

The Gimp Toolkit, neboli GTK+, je toolkit umožňující vytváření grafického uživatelského rozhraní. Přestože byla původně knihovna navržena pro X Window system, je multiplatformní a zahrnuje podporu také pro MS Windows a Mac/OS. Toolkit GTK+ byl původně vytvořen pro grafický program GIMP v roce 1997.

Knihovna GTK+ je rozdělena na dvě části. První částí je část zprostředkovávající widgety, druhou částí je knihovna GDK (GTK Drawing Kit). Část starající se o widgety



Obrázek 3.2: Struktura knihovny GTK+

nevyužívá ve své implementaci knihovnu Xlib přímo, ke všem voláním využívá abstrakci X Window okenního systému, knihovnu GDK.

Jelikož widgetová část nevyužívá přímo Xlibu, ale abstrakci v podobě knihovny GDK, je možné knihovnu GTK+ přenést na jiné okenní systémy pouze úpravou knihovny GDK pro daný okenní systém. Knihovna GDK poskytuje podporu pro většinu dnes používaných okenních systémů: X Window, okenní systém MS Windows, okenní systém MacOS (Quartz), podporuje rovněž vykreslování přímo do framebufferu grafické karty.

Pro více informací odkazuji na domovskou stránku projektu GTK+ [1].

3.5 Grafická knihovna Cairo

Cairo je 2D grafická knihovna, pomocí níž provádí knihovna GTK+ většinu kreslení. Cairo poskytuje metody pro kreslení základních útvarů (úsečka, obdélník, ...), které kreslí za pomoci Cairo contextu (cairo_t). Cairo context obsahuje informace o tom, jak se mají věci kreslit, například barvu a styl čáry.

Plocha, na kterou knihovna Cairo kreslí, se nazývá surface (cairo_surface_t). Surface může být reprezentován například bufferem (obrázkem) v operační paměti. Tento buffer je pak možné uložit do PNG souboru. Obrázek je v operační paměti uložen po jednotlivých pixelech. Cairo podporuje tyto formáty pixelů:

CAIRO_FORMAT_ARGB32 Každý pixel je uložen v 32 bitech. Horních 8 bitů je vyhrazeno pro informaci o transparenci, zbylé bity obsahují hodnotu RGB.

CAIRO_FORMAT_RGB24 Stejné jako předchodí formát, neposkytuje však žádnou informaci o transparenci – horních 8 bitů je nevyužito.

CAIRO_FORMAT_A8 Každý pixel zabírá 8 bitů – nese informaci o transparenci.

CAIRO_FORMAT_A1 Každý pixel je reprezentován jedním bitem – nese informaci o transparenci.

Detailnější informace lze nalézt na domovské stránce knihovny Cairo [4].

Kapitola 4

Principy knihovny GTK+

Knihovna GTK+ se skládá ze dvou částí, první část obstarávající samotné widgety je platformně nezávislá. Druhou částí je knihovna GDK, kterou část widgetů využívá jako prostředníka ke komunikaci s X serverem. Jak již bylo dříve řečeno, knihovna GDK nabízí podporu pro více grafických systémů.

Implementace konkrétního grafického systému v knihovně GDK se nazývá backend knihovny GDK. Knihovna obsahuje backendy pro X11, Windows GDI, Quartz a DirectFb.

4.1 Základní pojmy

GObject Knihovna GTK+ využívá při implementaci objektového systému Glib zvaného též GObject, což umožňuje využití principů objektového programování i v jazyce C. Základy modelu GObject lze nalézt na [1]. Pokud bude řeč o objektu, bude jím myšlen objekt modelu GObject.

4.2 Základní objekty knihovny GDK

GdkScreen je objekt reprezentující fyzickou obrazovku (screen). Obsahuje informace, jako je například rozlišení obrazovky.

GdkDisplay je objekt reprezentující X display, tedy pracovní stanici obsahující klávesnici, myš a monitor.

GdkDrawable reprezentuje něco, do čeho lze kreslit (drawable). Knihovna GDK obsahuje dva odvozené typy, jsou jimi GdkWindow a GdkPixmap. GdkDrawable obsahuje metody pro kreslení základních útvarů, jako je například obdélník nebo úsečka.

GdkWindow je objekt reprezentující obdélníkovou část obrazovky. Okna tvoří hierarchickou strukturu. Pozice oken se udává vůči jejich rodiči. Potomci jsou částí svého rodiče, nikdy nepřesáhnou jeho plochu. Knihovna GDK poskytuje několik typů oken:

GDK_WINDOW_ROOT reprezentuje hlavní okno (nejvyšší v hierarchii oken). Je vytvářeno pouze jednou při inicializaci okenního systému.

GDK_WINDOW_TOPLEVEL reprezentuje klasické okno v okenním systému. Window manager většinou umístí dekorace kolem tohoto typu okna.

GDK_WINDOW_DIALOG je podobné top level oknu, avšak window manager kolem něj zpravidla neumisťuje dekorace.

GDK_WINDOW_CHILD reprezentuje obdélníkovou část top level okna.

GDK_WINDOW_INPUT_ONLY není okno v klasickém slova smyslu. Jedná se o okno určené pouze k příjmu událostí, nemá žádnou grafickou podobu.

GdkPixmap je objekt reprezentující off-screen drawable, z pohledu X umístěný na serveru.

GdkImage je objekt reprezentující oblast kreslení umístěnou z pohledu X u klienta. Jelikož není odvozen od typu GdkDrawable, neimplementuje ani kreslící funkce. K pixelům se přistupuje přímo. Formát pixelů je implementačně závislý, viz GdkVisual.

GdkGC je objekt popisující, jak se mají věci kreslit. Popisuje například barvu pozadí nebo šířku či styl čáry. Všechny kreslící operace knihovny GDK mají objekt GdkGC jako jeden z argumentů.

GdkVisual je objekt poskytující nízkoúrovňové informace o grafickém hardwaru, jako je bitová hloubka nebo způsob převodu pixelů na jednotlivé barevné složky a naopak. Tato informace slouží především ke zjištění, v jakém formátu jsou uloženy pixely v GdkImage. Knihovna GDK rozlišuje visualy typů:

GDK_VISUAL_STATIC_GRAY Každý pixel reprezentuje přímo odstín šedi.

GDK_VISUAL_GRAYSCALE Každý pixel je index v tabulce, která jej převádí na odstín šedi.

GDK_VISUAL_STATIC_COLOR Každý pixel ukazuje do colormapy, která jej převádí na jednotlivé složky. Colormapa je statická a nejde modifikovat.

GDK_VISUAL_PSEUDO_COLOR Je podobný jako předchozí typ visualu, avšak colormapa je modifikovatelná aplikací.

GDK_VISUAL_TRUE_COLOR Každý pixel obsahuje přímo červenou, zelenou a modrou složku. Způsob, jakým se pixel převede na jednotlivé složky, definuje použitý visual.

GDK_VISUAL_DIRECT_COLOR Tento typ visualu je podobný předchozímu GDK_VISUAL_TRUE_COLOR, avšak barvy se nepřevádějí z pixelu, ale přes colormapu.

GdkColor představuje jednu barvu. Obsahuje jednotlivé složky barvy (červenou, zelenou a modrou) a hodnotu pixelu. Hodnota pixelu se liší v závislosti na použitém typu visualu. Hodnota pixelu je zjištěna při registraci barvy funkcí `gdk_color_alloc`.

GdkColormap představuje seznam barev, které je možno použít. Než je možné barvu použít, je třeba ji zaregistrovat v colormapě objektu drawable, do kterého chceme touto barvou kreslit, funkcí `gdk_color_alloc`.

4.3 Backend knihovny GDK

Jak již bylo řečeno, knihovna GDK poskytuje jednotný interface pro několik grafických systémů. Není však třeba pro každý systém implementovat všechny funkce knihovny GDK. Pro každý grafický systém je třeba implementovat danou množinu funkcí a objektů, které jsou pak začleněny do knihovny.

Všechny objekty mají veřejnou a privátní verzi [8]. Veřejnou verzí je například GdkWindow, odpovídající privátní verzí je GdkWindowImplX11, která obsahuje navíc položky specifické pro daný backend.

Pro každý backend existuje složka v adresáři gdk, kde jsou umístěny jeho zdrojové kódy. Název této složky zpravidla koresponduje s názvem backendu, v případě backendu X11 je to například složka x11.

Hlavní věci, které jsou v režii backendu:

- volba dostupných visualů
- implementace kreslení
- implementace obrázků
- implementace oken
- implementace klávesnice a myši

V backendu je třeba implementovat velké množství dalších funkcí a objektů, které zde nebudu uvádět, především z důvodu, že by se jednalo o dlouhý, nic neříkající seznam. Kompletní seznam lze vyčíst ze zdrojových kódů každého backendu.

4.4 Implementace GdkWindow a GdkPixmap

GdkWindow a GdkPixmap jsou základní dva typy drawablů, které knihovna GDK nabízí. GdkWindow i GdkPixmap implementují všechny metody zděděné po GdkDrawable společně pro všechny backendy. Jak je tedy možné, že se při volání metody `draw_rectangle` objektu GdkWindow objeví obdélník na obrazovce? Volání `draw_rectangle` končí ve skutečnosti voláním stejné metody privátní verze objektu. Na privátní verzi ukazuje ukazatel impl ve veřejné verzi. Privátní verze jsou vytvářeny v konstruktoru veřejných verzí objektů voláním funkce, kterou backend poskytuje. Na to, jak jsou privátní verze v backendu implementovány, se podíváme v další kapitole.

Kapitola 5

Změny v knihovně GTK+

V této kapitole popíšu změny, které jsou náplní jedné z částí této práce. Úpravy v knihovně GDK spočívají v přidání nového backendu, který umožňuje knihovně GDK grafický výstup do obrázku, resp. bufferu v operační paměti. Nový backend byl nazván myimage. Knihovnu GTK+ využívající backend myimage budu dále nazývat GTK MyImage. Situaci ilustruje obrázek 3.2.

S knihovnou GTK MyImage lze používat většinu dnes existujících GTK programů bez nutnosti úpravy jejich zdrojových kódů. Výjimku tvoří programy napsané pro konkrétní okenní systém, například programy, které přímo komunikují s X serverem pomocí knihovny Xlib, čehož se občas využívá především pro účel optimalizace. Prakticky ve všech případech lze ovšem najít ekvivalentní náhradu využívající pouze knihovny GDK.

5.1 Implementace

V této sekci budu popisovat pouze hlavní změny, které bylo nutné v knihovně GTK+ provést.

Dostupné visualy

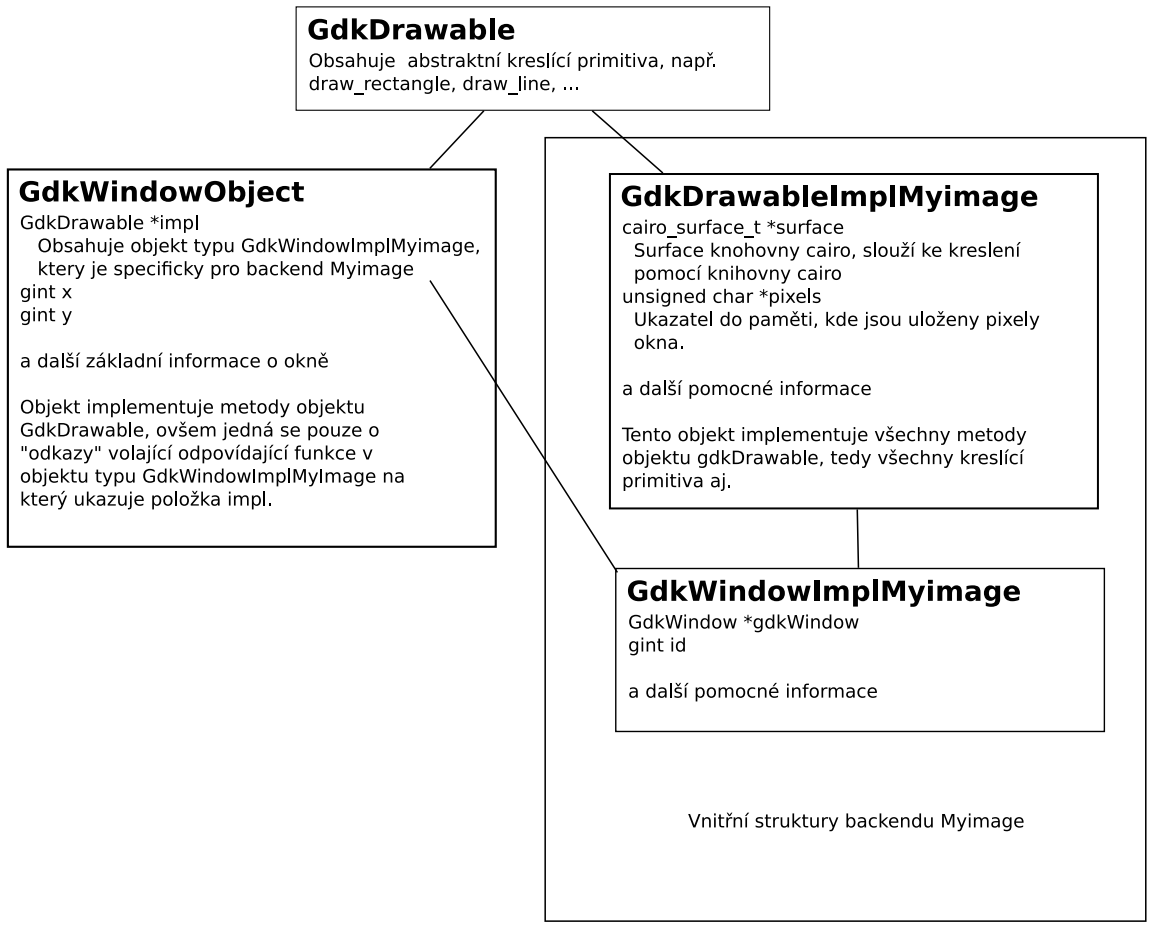
Backend myimage poskytuje pouze jeden typ visualu a to GDK_VISUAL_TRUE_COLOR o bitové hloubce 32. Výběr bitových hloubek byl omezen knihovnou Cairo. Knihovna GTK+ od verze 2.8 poskytuje metodu, která vytváří Cairo context ke každému typu drawable. Z tohoto důvodu byl omezen výběr bitových hloubek a formát pixelu obrázku. Backend myimage využívá formátu pixelu CAIRO_FORMAT_ARGB32 – každý pixel

má velikost 32-bitů, horních 8 bitů je vyhrazeno informaci o transparenci pixelu, zbylé bity reprezentují RGB barvu pixelu.

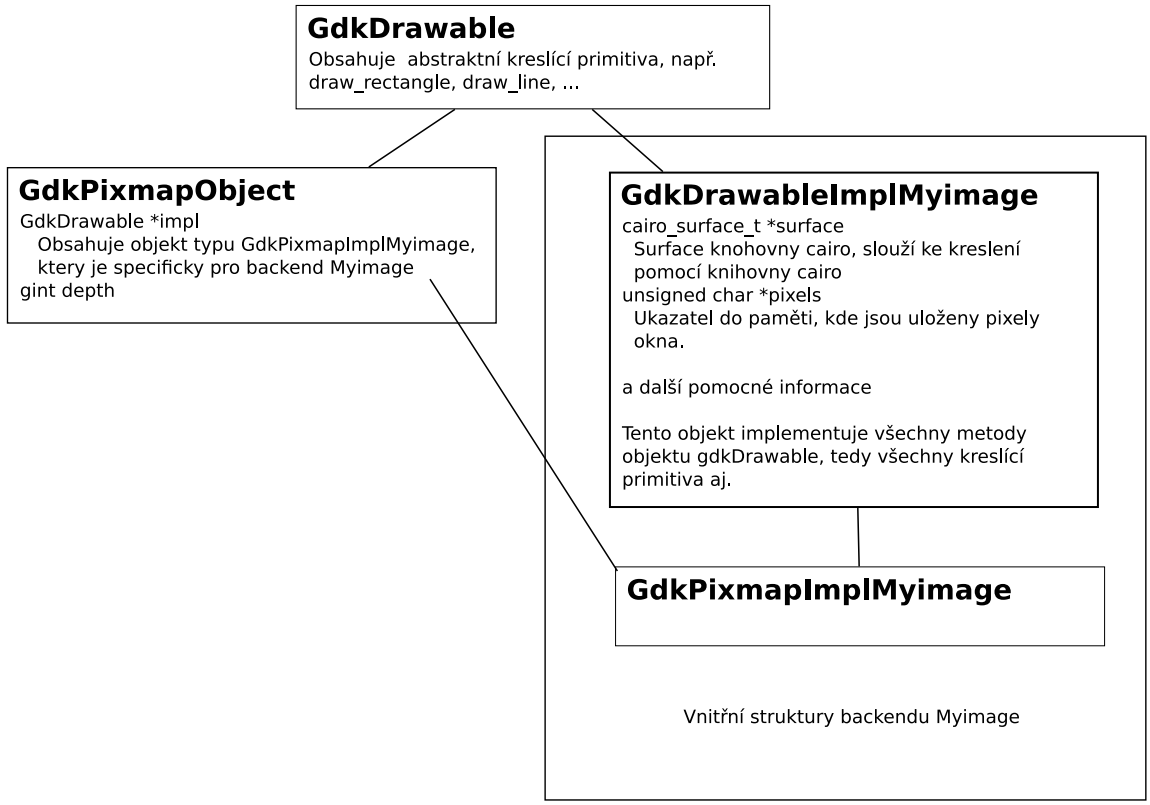
Implementace drawablů

Vraťme se k objektům GdkWindow a GdkPixmap. Jsou to dva základní typy drawablů, které knihovna GDK nabízí. Tyto objekty obsahují ukazatel na své privátní verze. V případě GdkWindow tento ukazatel ukazuje na objekt typu GdkWindowImplMyImage, v případě GdkPixmap na objekt typu GdkPixmapImplMyImage.

GdkWindowImplMyImage a GdkPixmapImplMyImage jsou odvozené od typu GdkDrawableImplMyImage, ve kterém jsou implementovány všechny metody zděděné po GdkDrawable.



Obrázek 5.1: GdkWindowImplMyImage



Obrázek 5.2: GdkPixmapImplMyImage

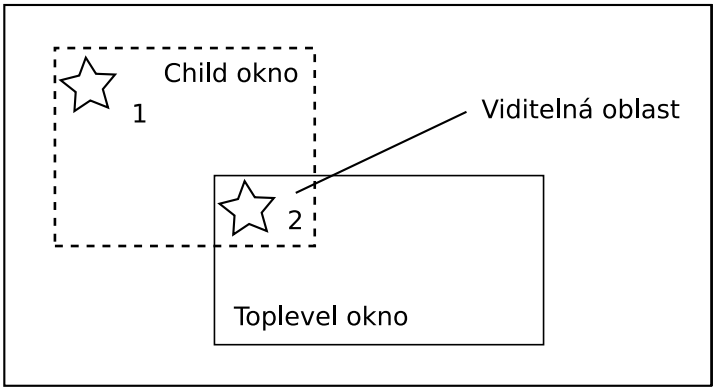
GdkDrawableImplMyImage Privátní verze GdkDrawable. Reprezentuje drawable backendu MyImage. Obsahuje ukazatel na surface knihovny Cairo, do kterého je kresleno a ukazatel na místo v paměti, kde jsou umístěny pixely bufferu v operační paměti. V objektu jsou implementovány všechny metody z rodičovské třídy GdkDrawable.

Implementace oken

Okno je v backendu myimage reprezentováno objektem GdkWindowImplMyImage, viz obrázek 5.1. Společně s vytvořením okna je nutné vytvořit příslušný surface knihovny Cairo. Nyní se podíváme, jakým způsobem se vytváří surface pro jednotlivé typy oken.

Root okno Vytváří se pouze jednou při inicializaci okenního systému. Cairo surface je vytvořen nad alokovaným prostorem v paměti o požadovaných rozměrech.

Toplevel okna Pro každé toplevel okno je alokován prostor v operační paměti, nad kterým je vytvořen Cairo surface. Toplevel okna nejsou tedy kreslena přímo do surfacu root okna, pokud tedy chce někdo vytvořit snímek celé aplikace, musí se všechna toplevel okna nakopírovat do root okna, poté se uloží obsah root okna.



Obrázek 5.3: Ilustrace child oken

Child okna Child okno představuje obdélníkovou část toplevel okna, proto je taky tento typ oken vytvořen nad pamětí daného toplevel okna jako jeho část. S implementací tohoto typu oken byly problémy způsobené faktem, že child okno může mít i zápornou pozici vůči svému rodiči. Situace je ilustrována obrázkem 5.3. Jediným řešením je vytvořit surface pouze nad viditelnou částí toplevel okna, to má ovšem jeden problém, co když bude chtít aplikace kreslit pomocí knihovny Cairo do child okna na pozici (0,0)? Problém je dostatečně popsán obrázkem. Východiskem z této situace je možnost knihovny Cairo posunout bod, vůči němuž jsou brány souřadnice. Pokud je tedy child okno v záporné pozici vůči jeho rodiči, je voláním funkce `cairo_surface_set_device_offset` nastaven bod, vůči němuž budou brány souřadnice.

Implementace pixmapů

Pixmap je reprezentovaný objektem GdkPixmapImplMyImage, viz obrázek 5.2. Jelikož je podporován pouze jeden visual, je podporován pouze odpovídající formát pixmapu, tedy 32-bitový.

Implementace obrázků

Obdobně jako u pixmapů jsou implementovány pouze 32-bitové obrázky.

Implementace kreslení

GdkDrawableImplMyImage je objekt, který musí implementovat všechny kreslící operace. Některé kreslící operace jsou implementovány přímo nad pixely daného drawablu, ale při implementaci některých je použit interface knihovny Cairo. Přímá implementace bez použití knihovny Cairo byla provedena zejména u klíčových operací, jelikož vykazuje výrazně vyšší výkon.

Kreslení úsečky

Kreslení úseček je implementováno přímo. Ke kreslení se používá Bresenhamův algoritmus [5]. Pro rozpoznání míst, kde úsečka nemá být kreslena je použit Cohen-Sutherlandův algoritmus [6].

Kreslení drawablů

Jelikož knihovna nemá povinnost implementovat kreslení jednoho drawablu do jiného s jinou bitovou hloubkou (navíc knihovna implementuje pouze jeden typ visualu), je kreslení realizované velice efektivně, pouze pomocí přesunu bloku paměti.

Kreslení obrázků

Jelikož formát obrázku je implementačně závislý a používá stejný visual, jako všechny drawably, je obrázek napřed převeden na typ GdkPixmap a pak je použita metoda pro kreslení drawablů. Obdobně je implementováno i kopírování drawablu do obrázku.

Kreslení textu

Knihovna GTK+ kreslí text pomocí knihovny Pango do Cairo surfacu daného drawablu. O kreslení textu se tedy nestará backend, ale přímo knihovna GDK ve spolupráci s knihovnou Pango. Funkce `gdk_draw_string` a `gdk_draw_text` jsou zastaralé a nejsou implementované.

Ostatní kreslící funkce

Ostatní funkce jsou implementovány za použití knihovny Cairo.

Uložení snímku do souboru	s název souboru
Změna pozice myši	e 3 x y
Stisknuté tlačítko myš	e 4 x y
Uvolněné tlačítko myši	e 7 x y
Vypsaní obsahu root okna	d

Obrázek 5.4: Příkazy TCP serveru

5.2 TCP server

Backend knihovny GDK zprostředkovává vstupně-výstupní zařízení (myš, klávesnice), tyto funkce jsou užitečné i pro backend myimage především pro účel debugování. Zároveň je vhodné, aby knihovna uměla uložit snímek spuštěného GTK programu. Bylo nutné najít vhodné řešení, jak tyto funkce implementovat do backendu myimage. Jako řešení byl zvolen TCP server, který zajišťuje všechny výše zmíněné požadavky.

Pomocí serveru implementovaného v knihovně GDK lze:

- Používat klávesnici a myš.
- Vypsat obsah root okna aplikace.
- Uložit snímek do souboru na filesystému serveru.

Server je spuštěn při inicializaci knihovny GDK na TCP portu 5463. V tabulce 5.4 jsou shrnuty všechny příkazy serveru. Port, na kterém bude server poslouchat, je možné změnit při spuštění programu parametrem `--myimage-port`. Pokud není server potřeba, je možné jej vypnout parametrem `--myimage-disable-server`.

Pro účel debugování byl vytvořen program nazvaný myimage-klient, který do jisté míry umožňuje interaktivní používání aplikace. K dorozumění s knihovnou GTK využívá TCP serveru.

Kapitola 6

Mozilla

Mozilla je opensource webový prohlížeč použitelný na více platformách (cross-platformní). Opensource znamená, že je zdrojový kód dostupný každému a každý jej může modifikovat.

Projekt Mozilla je jeden z největších opensource projektů vůbec a počet řádků zdrojového kódu se počítá v milionech. Jelikož je složité se v takto rozsáhlém projektu orientovat, je zdrojový kód Mozilly rozdělen na jisté funkční celky – komponenty.

6.1 Stručný pohled na XPCOM

XPCOM (Cross Platform Component Object Model) je řešení použité v Mozille, které umožňuje rozdělit kód do menších funkčních celků – komponent.

Komponenta je znovupoužitelný modulární kus kódu, který implementuje jisté předem dané rozhraní (interface). Rozhraním se myslí daná množina metod, včetně parametrů a návratových hodnot.

Jedno rozhraní může mít několik implementací. Bude řeč například o rozhraní nsIRenderingContext, implementujícím kreslení v Mozille, které má implementaci pro každý podporovaný grafický systém.

6.2 Hlavní adresáře

Zdrojový kód Mozilly je členěn do adresářů, podle funkce, kterou plní. Pro nás budou zajímavé pouze některé z nich, jsou jimi adresáře Gfx a Widget.

Widget Tento adresář obsahuje prvky uživatelského rozhraní, jako jsou scrollbary apod.

Gfx Adresář GFX obsahuje platformně závislou část (implementaci) komponent pro kreslení základních geometrických útvarů, textů a obrázků. Neposkytuje žádné widgety (prvky uživatelského rozhraní), ale poskytuje základní metody pro kreslení. Adresář Gfx obsahuje podadresář pro každý implementovaný grafický systém (GTK+, Qt, ...)

Gfx/gtk Obsahuje implementaci kreslících komponent využívajících knihovnu GTK+.

6.3 Grafika v Mozilla

Mozilla obsahuje oddělené implementace widgetů (prvků uživatelského rozhraní) pro různé grafické systémy (adresář widget). Kromě oddělené implementace widgetů jsou rovněž pro každý grafický systém implementovány rozhraní nsIDeviceContext a nsIRenderingContext.

Device Context Device context slouží ke komunikaci s grafickým systémem. Device Context je v podstatě množina metod pro dotazování se grafického systému na:

- Rozlišení výstupního zařízení.
- Vytváření Rendering Contextu dostupného pro dané zařízení.

Rendering Context Hlavním úkolem Rendering Contextu je vykreslení základních geometrických útvarů, rovněž umožňuje kreslení obrázků a textů.

6.4 Nutné změny

Ideálně by Mozilla zkompilovaná pro grafický systém GTK využívala pouze rozhraní knihovny GTK, ve skutečnosti tomu tak není. Implementace kreslících komponent Mozilly pro knihovnu GTK předpokládá využití backendu X11 a Mozilla se občas snaží komunikovat přímo s X serverem.

Aby Mozilla šla zkompilovat s knihovnou GTK MyImage, bylo nutné nahradit všechny pokusy o komunikaci s X serverem pomocí knihovny Xlib ekvivalenty v knihovně GDK. Většina závislostí se nachází ve složce gfx, menší část ve složce widget.

Tuto práci již z větší části udělal někdo jiný a s drobnými úpravami bylo možné použít již existující patch, viz [7].

Tento patch byl původně určen pro backend directfb a je dostupný pouze pro Mozillu 1.5.0.3. V současné době již existuje nový patch pro beta verzi Mozilly 3.0 přidávající podporu backendu directfb. Tento nový patch lze rovněž snadno upravit pro backend MyImage.

6.5 Nový kreslící model

Implementace Rendering Contextu ve starších Mozillách neumožňovaly v prostředí Unixu vykreslovat stránky jinak, než prostřednictvím toolkitu Gtk+ nebo Qt, které nutně využívají X serveru. S postupem času se kreslící model v Mozille (resp. renderovacím jádru Gecko) značně změnil. Nový kreslící model v Mozille nazvaný Thebes, využívá ke kreslení knihovnu Cairo.

Model Thebes byl pro testovací účely součástí i starších renderovacích jader, avšak až jádro Gecko 1.9, které je použito v Mozille 3.0, provádí veškeré kreslení pomocí knihovny Cairo.

Stále je sice možné využít knihovny GTK MyImage ke kreslení bez použití X serveru, avšak nový model kreslení v Mozille poskytuje novou cestu, jak stejného efektu docílit. Více o tomto novém kreslícím modelu se lze dočíst na webových stránkách Mozilly.

Kapitola 7

Program webshot

Program Webshot je náplní druhé části práce. Webshot slouží k pořizování snímků webových stránek (dále jen stránek), které následně uloží do souboru ve formátu PNG. K jeho dalším vlastnostem patří například možnost vytváření miniatur stránek. V kombinaci s knihovnou GTK MyImage je možné měnit také rozlišení pomyslné obrazovky, na kterou se stránka vykreslí.

7.1 Princip programu

K vykreslování stránek využívá program webového prohlížeče Mozilla Firefox, dále jen Mozilla. Mozilla poskytuje různé možnosti embeddingu (začlenění do aplikací), včetně možnosti začlenění prohlížeče do okna GTK programu, a to přes widget knihovny GTK. Tento widget se jmenuje gtkmozembed a poskytuje všechny potřebné funkce webového prohlížeče. Na tomto widgetu je postaven například webový prohlížeč Galeon.

Idea programu je jednoduchá. Po spuštění se vytvoří GTK okno, u kterého se změní velikost tak, aby zabíralo celou plochu obrazovky. Toto okno obsahuje jediný widget gtkmozembed, který vykresluje stránku. Ze vstupu se přečte adresa stránky, které chceme udělat screenshot, a soubor, kam má být snímek uložen. Poté, co se stránka načte, tedy je zobrazena ve widgetu, se uloží obsah okna do PNG souboru.

7.2 Implementace

Cyklus programu

Celý program je implementován jako stavový automat. Po nezbytné inicializaci se program nachází ve stavu DATA. Tento stav indikuje, že má být načtena adresa a název

souboru ze vstupu (data).

Po přečtení dat se přejde do stavu NAČÍTÁM – začne se načítat požadovaná stránka. Program čeká na stáhnutí a načtení stránky. Aby nedošlo k situaci, že se program zablokuje na stránce, která se načítá příliš dlouho nebo by se nikdy nenačetla, je hned po přechodu do stavu NAČÍTÁM nastaven timeout, tedy maximální doba, po kterou může být stránka načítána. Pokud timeout vyprší, přejde se do stavu DATA, což umožní načítání nové stránky.

Poté, co je stránka načtena, se přejde do stavu NAČTENO, ve kterém se zastaví timeout na načítání stránky a přejde se do stavu SNÍMKUJI, ve kterém se uloží snímek načtené stránky do souboru.

Po uložení stránky se opět přejde do stavu DATA. Tento cyklus se opakuje, dokud nejsou přečtena všechna data.

Uložení snímku

Uložení snímku probíhá přes interface knihoven GTK+ a Cairo následujícím způsobem:

```
cr = gdk_cairo_create(okno_prohlizece);
cairo_surface = cairo_get_target(cr);
cairo_surface_write_to_png(cairo_surface, "/snimek.png");
```

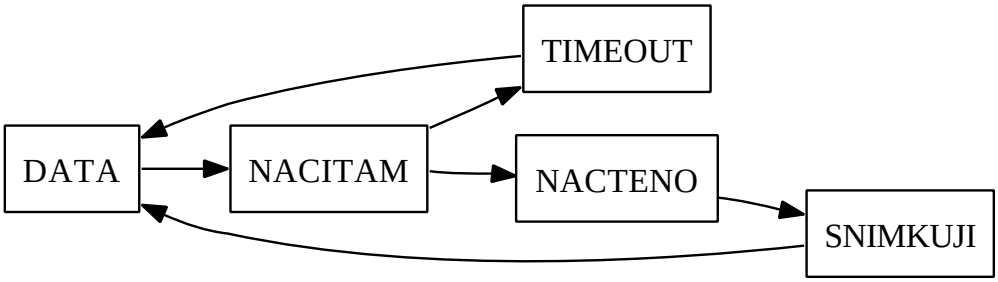
Jak je z úryvku kódu patrné, program Webshot nevyužívá k uložení snímku žádných rozšíření knihovny GTK MyImage. K uložení postačí původní rozhraní knihovny GTK+, avšak implementace těchto funkcí nevyužívá X serveru.

Stránka je ukládána ve chvíli příchodu události onload. Metoda nefunguje v případě, že stránka obsahuje rámy, jelikož je událost onload generována pro každý rám, proto jsou rámy vypnuty. Z podobného důvodu jsou vypnuty rovněž meta redirecty[1]. Určité procento stránek tedy nebude vůbec zobrazeno.

Přístup na zabezpečené stránky a dialogy

Mozilla se přirozeně snaží komunikovat s uživatelem a předkládá mu různé výstrahy ve formě dialogů, které uživatel musí potvrdit, aby se požadovaná stránka načetla. Toto chování je pro program webshot nevhodné. Tento problém se projeví například při přechodu na zabezpečenou stránku, kdy uživatel musí potvrdit, že s tím souhlasí. Některé dialogy jdou vypnout přímo v profile souboru Mozilly, to ale nepokrývá zdaleka

[1] přesměrování pomocí meta tagu:
`<meta http-equiv="refresh" content="5;url=http://firma.com">`



Obrázek 7.1: Jednotlivé stavy programu

všechny případy, například, pokud stránka využívá neplatný certifikát. Mozilla avšak poskytuje elegantní řešení tohoto problému.

Řešení spočívá v nahrazení dvou komponent, které vytváří ony dialogy, upravenými komponentami, které "vše potvrdí" bez toho, aby se ptaly uživatele. Jsou to komponenty Bad Cert Listener a Promt Service.

Bad Cert Listener Tato komponenta je využívána v případě, že je nalezen problém s certifikátem při přístupu na zabezpečenou stránku. Bad Cert Listener má různé metody pro různé typy problémů s certifikátem. Pokud například stránka využívá prošlý certifikát, Mozilla vyvolá metodu této komponenty ConfirmCertExpired. Původní implementace zobrazí dialog, aby s tím byl uživatel srozuměn, a čeká na potvrzení. Nová implementace povolí přístup bez toho, aby zobrazovala jakýkoli dialog, což je přesně to, co se od programu Webshot očekává.

Promt Service Komponenta zajišťující všechny dialogy v Mozille. Pokud chce Mozilla dát uživateli najevo nějakou skutečnost, například, že zadal špatný formát adresy, využije k tomu metodu Alert této komponenty. Původní implementace této metody zobrazí okénko s popisem problému a čeká na stisknutí ok. Poté, co uživatel stiskne v dialogu ok, tak vrátí zprávu typu "uživatel stiskl ok". Nová implementace nezobrazí okno s varováním, ale rovnou vrátí "uživatel stiskl ok". Tím se vyhneme zobrazení dialogů.

7.3 Profile soubor

Profile soubor je soubor, ve kterém lze upravit chování Mozilly. Každý uživatel má vlastní profile soubor $HOME/.mozilla/webshot/profile.js. V profile souboru použitém

programem Webshot jsou vypnuta zbytečná varování, například při přechodu z nezabezpečené na zabezpečenou stránku a naopak. Pokud by v profile souboru nebyla tato varování vypnuta, byla by zachycena komponentou Prompt Service.

7.4 Použití

Parametry programu

-m	Vytvořit miniatury stránek.
-w	Šířka miniatury.
-h	Výška miniatury.

Obrázek 7.2: Parametry programu webshot.

Změna rozlišení

Program sám o sobě neumožňuje změnit rozlišení[2]. Pokud je program používán s knihovnou GTK MyImage, je možné změnit rozlišení parametry `--myimage-width` a `--myimage-height`, které se zadají přímo jako parametry programu.

Vstup programu

Program čte data ze standardního vstupu. Pokud program napíše `ready`, nachází se ve stavu, kdy je možné zadat data. Data se zadávají ve formátu: adresa (mezera) soubor (enter). Poté, co program data zpracuje a uloží snímek opět vypíše `ready`.

Zadávat data ručně samozřejmě není vždy ideální. Tento způsob zadávání dat ovšem umožňuje zpracování připraveného textového souboru, který obsahuje na každém řádku dvojici (adresa, soubor). Tento soubor je pak možné přesměrovat na standardní vstup programu, viz příklad:

```
Soubor weby.txt:
firma1.com /tmp/firma1.png
firma2.com /tmp/firma2.png
...
firmaN.com /tmp/firmaN.png
```

[2]rozlišením se myslí rozlišení pomyslné obrazovky, na kterou se stránka vykreslí

```
Přesměrování vstupu:
webshot < weby.txt
```

Ukončení programu

Program se ukončí po příchodu znaku konce souboru, který při vstupu z klávesnice lze vyvolat stisknutím `CTRL+D`.

Kapitola 8

Závěr

8.1 Co zabralo nejvíce času?

Nejvíce času zabrala snaha o pochopení cizího kódu, především knihovny GTK+ a návrh úprav, které je třeba provést. Samotná implementace byla méně časově náročná. Průběh také velmi zpomalovala doba sestavení tak velkých projektů, jako je knihovna GTK+ a prohlížeč Mozilla byť při těch nejmenších změnách ve zdrojovém kódu.

8.2 Splnění cíle

Tato práce měla dva cíle. Prvním cílem bylo vhodně upravit knihovnu GTK+, aby podporovala grafický výstup do obrázku. Druhým cílem bylo vytvořit program za použití této knihovny, umožňující dávkově zpracovávat snímky webových stránek bez interakce s uživatelem, a který by pracoval bez asistence X serveru.

Do knihovny GTK+ byl přidán nový backend, který umožňuje knihovně GTK+ grafický výstup do obrázku. Za použití této knihovny byl pak vytvořen program Webshot, který umožňuje dávkově, bez interakce s uživatelem vytvářet snímky webových stránek bez použití X serveru.

Jisté procento stránek však není schopen program Webshot vykreslit, což je způsobeno metodou zjištění, kdy je stránka kompletně vykreslena. Snímek se uloží ve chvíli příchodu události onload. Tuto událost Mozilla generuje pro každý rám, který stránka obsahuje, rovněž se s generováním události nečeká na přesměrování pomocí meta redirectu. Aby se zabránilo příchodu více událostí onload a uložení snímku ve chvíli, kdy je načten pouze jeden rám a stránka není ještě kompletně načtena, jsou rámce vypnuty. Rovněž jsou vypnuta přesměrování pomocí meta redirectu.

Dalším problémem je postupné narůstání využití operační paměti při generování velkého množství snímků. Tento problém je způsoben webovým prohlížečem Mozilla, který neuvolňuje všechny alokované obrázky a pixmapy. Aby nedošlo k zahlcení operační paměti, je třeba program po určitém počtu snímků restartovat, čímž dojde k uvolnění alokovaných zdrojů.

Literatura

[1] *GTK+, Glib, GDK dokumentace*
http://www.gtk.org/

[2] *Stránky webového prohlížeče Mozilla*
http://www.mozilla.org/

[3] *XPCOM reference*
http://www.xulplanet.com/references/xpcomref/

[4] *Dokumentace knihovny Cairo*
http://cairographics.org/manual/

[5] *Bresenhamův algoritmus*
http://en.wikipedia.org/wiki/Bresenham's_line_algorithm

[6] *Cohen-Sutherlandův algoritmus*
http://en.wikipedia.org/wiki/Line_clipping

[7] *Patch pro použití GTK nad DirectFb s Mozillou*
https://bugzilla.mozilla.org/show_bug.cgi?id=357946

[8] *Slidy na přednášku Programování pro X Window System*
http://www.ms.mff.cuni.cz/~beran/vyuka/X/

Dodatek A

Příručky

A.1 Instalační příručka

V této části popíši kompletní postup instalace programu webshot s použitím přiloženého CD.

Instalace sestává ze tří částí v tomto pořadí:

- instalace knihovny GTK MyImage (případně závislostí)
- instalace Mozilly
- instalace programu Webshot

A.1.1 Instalace knihovny GTK MyImage

Instalaci budu popisovat včetně všech závislostí, pokud některé z nich již máte v systému, není třeba je instalovat znovu. Knihovna GTK+ závisí na těchto knihovnách `Glib>=2.12.0`, `Pango>=1.12.0` (s podporou backendu Cairo), `ATK>=1.9.0`, `Cairo>=1.2.0` (s podporou PNG), `PNG>=1.2.4.`

Prvním krokem je nastavení prefixu, tedy složky, kam se bude vše instalovat.

```
PREFIX="$HOME/bc"
export PREFIX
mkdir -p $PREFIX
```

Jelikož instalace bude probíhat do oddělené složky, musíme dát linkeru a programu pkgconfig vědět, kde má hledat příslušné knihovny a soubory.

```
LD_LIBRARY_PATH=$PREFIX/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH

PKG_CONFIG_PATH=$PREFIX/lib/pkgconfig:$PKG_CONFIG_PATH
export PKG_CONFIG_PATH

PATH=$PREFIX/bin:$PATH
export PATH
```

Připojíme instalační CD:

```
mount /cdrom
```

Instalace závislostí

Instalace Glib

```
cp /cdrom/dep/glib-2.12.13.tar.gz .
tar -xvzf glib-2.12.13.tar.gz
cd glib-2.12.13
./configure --prefix=$PREFIX
make && make install
cd ..
```

Instalace knihovna PNG

```
cp /cdrom/dep/libpng-1.2.28.tar.bz2 .
tar -xvjf libpng-1.2.28.tar.bz2
cd libpng-1.2.28
cp scripts/makefile.linux makefile

# Editujte makefile a změňte proměnnou prefix na ${PREFIX}

make && make install
cd ..
```

Instalace knihovny Cairo

```
cp /cdrom/dep/cairo-1.4.14.tar.gz .
tar -xvzf cairo-1.4.14.tar.gz
```

```
cd cairo-1.4.14
./configure --prefix=$PREFIX \
            --enable-png=yes \
            --enable-xlib=no \
            --enable-xlib-xrender=no
make && make install
cd ..
```

Instalace knihovny Atk

```
cp /cdrom/dep/atk-1.9.1.tar.bz2 .
tar -xvjf atk-1.9.1.tar.bz2
cd atk-1.9.1
./configure --prefix=$PREFIX
make && make install
cd ..
```

Instalace knihovny Pango

```
cp /cdrom/dep/pango-1.16.5.tar.gz .
tar -xvzf pango-1.16.5.tar.gz
cd pango-1.16.5
./configure --prefix=$PREFIX
# na posledním řádku jsou shrnuty podporované backendy,
# musí být podporovaný backend Cairo, pokud je uveden,
# je vše v pořádku a můžeme knihovnu nainstalovat

make && make install
cd ..
```

Instalace knihovny GTK MyImage

```
cp /cdrom/src/gtk-2.10.13_myimage.tar.gz .
tar -xvzf gtk-2.10.13_myimage.tar.gz
cd gtk+-2.10.13
./configure --with-gdktarget=myimage \
            --without-libjpeg \
            --without-libtiff \
            --prefix=$PREFIX
```

```
make && make install
cd ..
```

V tuto chvíli by měla být knihovna GTK MyImage nainstalována. Vyzkoušíme její funkčnost spuštěním programu gtk-demo:

```
$PREFIX/bin/gtk-demo

telnet 127.0.0.1 5463
s /tmp/test.png
```

Pokud soubor /tmp/test.png obsahuje grafický výstup programu gtk-demo, je knihovna správně nainstalována a můžeme přistoupit k instalaci prohlížeče Mozilla.

A.1.2 Instalace prohlížeče Mozilla Firefox

Konfigurační soubor Mozilla čte konfigurační parametry ze souboru .mozconfig, tento soubor je umístěný v kořenovém adresáři Mozilly. Případná změna konfiguračních parametrů, například prefixu, se provádí v tomto souboru.

```
CC="gcc-3.4"
export CC
CXX="g++-3.4"
export CXX

cp /cdrom/src/mozilla-1.5.0.3_myimage.tar.gz .
tar -xvzf mozilla-1.5.0.3_myimage.tar.gz
cd mozilla

# ujistěte se, že máte nastavenou proměnnou PREFIX

./configure
make && make install

LD_LIBRARY_PATH=$PREFIX/lib/firefox-1.5.0.3:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

A.1.3 Instalace programu webshot

```
cp /cdrom/src/webshot-v3.0b.tar.gz .
tar -xvzf webshot-v3.0b.tar.gz
cd webshot
./configure --prefix=$PREFIX
make && make install
```

Po instalaci nakopírujte do adresáře $HOME/.mozilla/webshot soubor profile.js.

A.2 Uživatelská příručka knihovny GTK MyImage

A.2.1 Kompilace programu s knihovnou

```
g++ ‘pkg-config --libs --cflags gtk+-2.0‘ program.cpp
./a.out
```

Knihovně je možné předávat parametry při spuštění programu, jsou jimi:

`--myimage-width`
: Nastaví šířku obrázku (výchozí hodnota 1024).

`--myimage-height`
: Nastaví výšku obrázku (výchozí hodnota 768).

`--myimage-port`
: Nastaví port, na kterém bude spuštěn TCP server (výchozí hodnota 5463).

`--myimage-disable-server`
: Nespustí se TCP server.

A.2.2 TCP server

Při inicializaci knihovny GTK MyImage se spouští TCP server na portu 5463, přes který je možné s knihovnou komunikovat. Server rozumí těmto příkazům:

Uložení snímku do souboru	s název souboru
Změna pozice myši	e 3 x y
Stisknuté tlačítko myš	e 4 x y
Uvolněné tlačítko myši	e 7 x y
Vypsaní obsahu root okna	d

A.2.3 Uložení snímku

Použití Cairo Surface

Tuto metodu lze použít pouze pro jedno okno aplikace.

```
GdkDrawableClass *w_priv = GDK_DRAWABLE_GET_CLASS(GDK_WINDOW(w));
cairo_surface_t *surface = w_priv->ref_cairo_surface(GDK_WINDOW_OBJECT(w)->impl);
cairo_surface_write_to_png(surface, "/tmp/file.png");
```

Přes TCP server

Přes TCP server je možné knihovně sdělit, aby uložila snímek programu do souboru na serveru:

```
telnet 127.0.0.1 5463
s /tmp/program.png
```

Dodatek B

Obsah přiloženého CD

/src Zdrojové kódy knihovny GTK MyImage, Mozilly Firefox a programu Webshot.
/dep Závislosti (Pango, Atk, Glib, Cairo, PNG).
/doc Programátorská a uživatelská dokumentace.
/bc Elektronická verze bakalářské práce.