
**Konference Czech Warpstock 2003**

Pořádají
Server www.os2.cz a uživatelé OS/2

# Tvorba distribučních archivů pro instalátor WarpIN

Jaroslav Kačer

Článek konference Czech Warpstock č. CZW-001
Červen 2003

Článek konference Czech Warpstock č. CZW-001
Červen 2003

# Tvorba distribučních archivů pro instalátor WarpIN

Jaroslav Kačer

## Résumé

Tento článek představuje univerzální open-source instalátor WarpIN, který má za cíl nahradit starší a méně schopný instalátor z produkce firmy IBM a stát se všeobecně používaným instalátorem na platformě OS/2. Článek se zaměřuje zejména na problematiku vytváření instalačních balíků pro tento program, včetně členění na podbalíky, psaní instalačního skriptu a manipulace s Workplace Shellem. Prezentované postupy jsou demonstrovány na reálném příkladu instalačního balíku knihovny J-Sim.

## Informace o autorovi

E-Mail: jkacer@kiv.zcu.cz
WWW: home.zcu.cz/~jkacer
Jaroslav Kačer (* 1978) vystudoval obor Informatika a výpočetní technika na Fakultě aplikovaných věd ZČU v Plzni, v současné době působí jako doktorand tamtéž. Zabývá se převážně paralelním programováním a diskrétními simulacemi. Systém OS/2 používá každodenně od roku 1997.

Kopie tohoto článku lze nalézt na
http://www.os2.cz/warpstock/

# Obsah

# 1 Popis instalátoru WarpIN

WarpIN je univerzální instalátor, vyvíjený pro operační systém OS/2 skupinou programátorů okolo serveru Netlabs. WarpIN běží v grafickém módu, potřebuje tudíž Presentation Manager. Pro specifické úkoly během instalace softwaru potřebuje také běžící Workplace Shell, není to však vždy nezbytně nutné. WarpIN je distribuován pod licencí GPL[1], každý si ho tedy může stáhnout zdarma a libovolně upravit za podmínky, že upravené zdrojové texty budou dále k dispozici.

Vývoj začali Jens Bäckman a Ulrich Möller, nyní do týmu patří ještě Teemu Ahola a Cornelis Bockemühl. Hlavní stránka projektu se nachází na serveru Netlabs ([WWW]), odkud vede odkaz na [XWp]. Zde je možno WarpIN stáhnout, stejně jako další dílo Ulricha Möllera – XWorkplace.

WarpIN je schopen při instalaci softwaru plně nakonfigurovat OS/2 systém, včetně změn v souboru `CONFIG.SYS`, registrace tříd Workplace Shellu, vytvoření složek a objektů různých typů apod. K vlastnostem WarpINu patří i funkce Undo, kontrola závislosti balíků, spouštění skriptů psaných v Rexxu (součást archivů) i externích programů.

Předpokládá se, že pokud je WarpIN jednou na počítač nainstalován, už navždy tam nainstalovaný zůstane. Uživatelé si pak obstarávají pouze instalační archivy pro tento instalátor. Tyto archivy obsahují výhradně instalovaný software, nikoliv instalátor samotný, což samozřejmě podstatně snižuje velikost stahovaných dat. O každém nainstalovaném programu si WarpIN uchovává množství informací. Tyto informace může využít například uživatel při kontrole nainstalovaného softwaru nebo samotný instalátor při odinstalovaní některého programu.

## 1.1 Princip fungování

Jak již bylo řečeno, samotný instalátor je třeba do systému nainstalovat před tím, než je možno instalovat balíčky pro něj určené. Stačí si z [XWp] stáhnout poslední vydání WarpINu (v době psaní tohoto článku je to verze 1.0.1) v podobě jednoho `EXE` souboru a ten spustit. WarpIN se nainstaluje do zadaného adresáře, kde zabere cca 1.2 MB. WarpIN si asociuje příponu `WPI`[2], takže od této chvíle stačí kliknout myší na libovolný `WPI` soubor a software v něm obsažený se začne za asistence uživatele instalovat.

[1] GNU General Public Licence

[2] Přípona `WPI` patří intalačním archivům určeným pro WarpIN.

Jak bude řečeno v kapitole 2.1, WarpIN lze nainstalovat také spolu s dodávaným programem, pokud na cílovém systému ještě není. V takovém případě má instalační archiv příponu `EXE`. Po spuštění takového instalačního souboru se nejdříve nainstaluje WarpIN (v případě tzv. plné instalace) a následně instalovaný software.

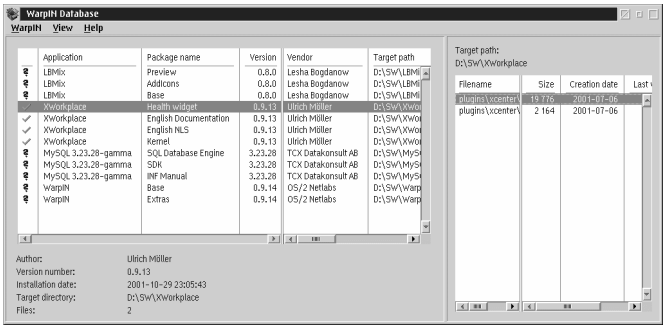
Informace o veškerém instalovaném softwaru jsou uloženy v tzv. *globální databázi* WarpINu, což je soubor `DATBAS_C.INI` ve formátu OS/2 inicializačních souborů, podobně jako například `OS2.INI` nebo `OS2SYS.INI`. V databázi jsou uloženy následující informace:

1. cesta, kam byl software nainstalován, a seznam všech zkopírovaných souborů, včetně jejich dat, jak byly uloženy v instalačním archivu, což WarpINu umožňuje zjistit, zda nebyly soubory náhodou po instalaci bez jeho vědomí přepsány;
2. datum instalace;
3. všechny změny provedené v souboru `CONFIG.SYS`, aby mohl být tento soubor v případě deinstalace obnoven do původního stavu;
4. všechny změny provedené v systémových `INI` souborech (implicitně to jsou `OS2.INI` a `OS2SYS.INI`) – změny mohou být opět vráceny;
5. všechny vytvořené objekty Workplace Shellu – při odinstalování mohou být zrušeny;
6. všechny nainstalované nebo nahrazené třídy Workplace Shellu – mohou být odregistrovány.

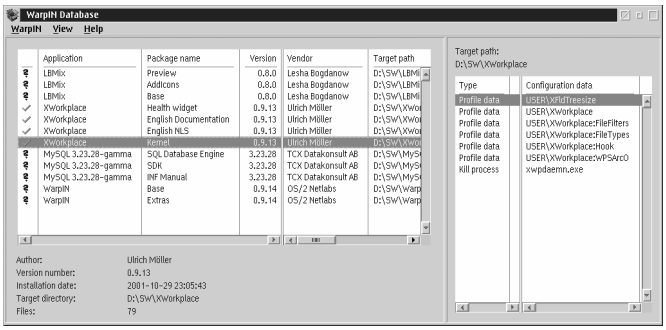
Na následujících obrázcích je ukázáno, jak WarpIN zobrazuje informace uživateli. V levé části obrazovky jsou zobrazeny všechny nainstalované aplikace a jejich balíky[3]. Jejich zobrazení lze přepínat – lze použít stromové nebo lineární (ploché) zobrazení. V pravé části jsou umístěny detailnější informace o nainstalovaném softwaru. Opět lze přepínat mezi dvěma zobrazeními – zobrazením všech nainstalovaných souborů (včetně jejich charakteristik) a zobrazením všech akcí, které se musejí provést při případné deinstalaci. Sem patří například modifikace souboru `CONFIG.SYS`, smazání objektů WPS, zastavení procesu apod.

WarpIN umožňuje uživateli zkontrolovat integritu již nainstalovaného softwaru. Pod tímto pojmem se rozumí, že software se nachází v tom stavu,

[3]O členění na balíky bude řeč v kapitole 2.2




Obrázek 1: Lineární zobrazení aplikací a jejich balíků spolu se seznamem souborů balíku *Health Widget* aplikace *XWorkplace*. Všechny balíky aplikace *XWorkplace* jsou zkontrolovány a nebyly u nich nalezeny žádné chyby.

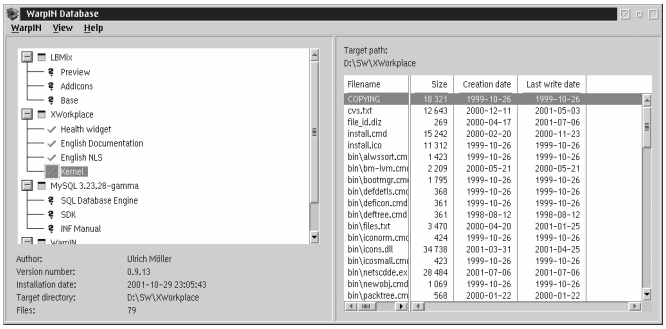



Obrázek 2: Opět lineární zobrazení aplikací a jejich balíků, nyní jsou ale v pravém panelu zobrazeny akce, které je třeba provést při odinstalování balíku *Kernel* aplikace *XWorkplace*.

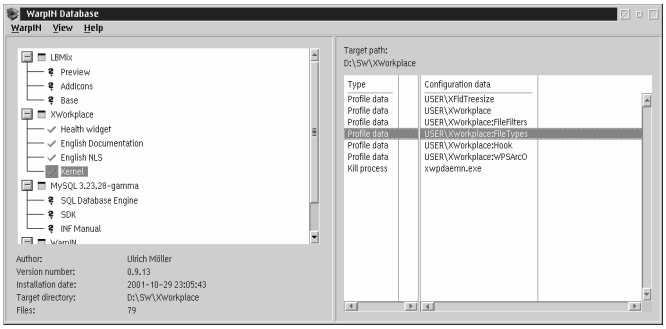
v jakém ho WarpIN nainstaloval. Tedy nebyl smazán žádný soubor, nastavení v souboru `CONFIG.SYS` ani v `INI` souborech nebyla změněna, objekty Workplace Shellu existují apod. Zkontrolovat lze jak celou aplikaci, tak i její jednotlivé balíky.

Aplikaci nebo její balíky lze kliknutím myši také odinstalovat. V tom případě se smažou všechny nainstalované soubory a vezmou se zpět všechny konfigurační změny provedené při instalaci.

Pokud uživatel provedl záměrně nějaké změny na nainstalovaném softwaru bez vědomí instalátoru nebo software dokonce ručně smazal, začne instalátor hlásit při kontrole chyby. Aby uživatel zamezil těmto chybovým hlášením, lze WarpIN přinutit, aby informace o nainstalované aplikaci nebo jejím balíku




Obrázek 3: Stromové zobrazení aplikací a balíků spolu se seznamem souborů.



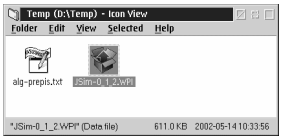

Obrázek 4: Stromové zobrazení aplikací a balíků spolu se seznamem akcí při odinstalování.

(balíkách) zapomněl. To se hodí zejména v případě ručního zásahu do instalovaného softwaru, může to být ovšem značně nebezpečné, pokud je software řádně nainstalovaný. Po smazání konfiguračních informací se totiž WarpIN přestane o nainstalovanou aplikaci starat a veškeré změny (deinstalaci) musí uživatel provést ručně, včetně editace `CONFIG.SYS`u a `INI` souborů. To může být značně namáhavá práce, navíc některá nastavení, zejména ta v `INI` souborech, nemusí uživatel vůbec objevit. Proto je třeba tuto funkci používat s rozmyslem.

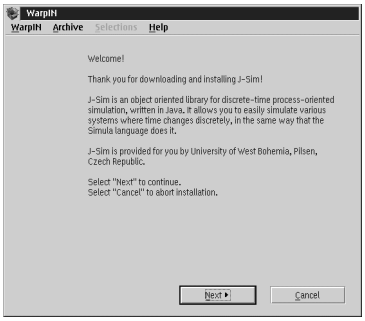
## 1.2 Módy činnosti

Nejběžnější použití instalátoru WarpIN spočívá v poklikání na nějaký stažený instalační archiv, který má příponu `WPI`. Pokud ještě nebyl nainstalován žádný balík z tohoto archivu, WarpIN se spustí v tzv. *plně instalačním módu*. Objeví se okno s úvodním pozdravem a uživatel pak klikáním na tlačítko

Další prochází nastavením instalace, vybírá balíčky, nastavuje cesty apod. Následující obrázky ukazují, jak taková typická instalace vypadá.




Obrázek 5: Nejdříve je třeba poklepat myší na stažený `WPI` soubor, čímž dojde ke spuštění WarpINu. Soubor se může nacházet v jakékoliv složce.



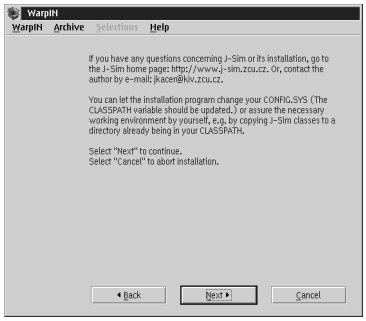

Obrázek 6: Po spuštění uvítá uživatele úvodní obrazovka instalované aplikace, která by ho měla stručně informovat.

Po výběru požadovaných balíčků se WarpIN obvykle otáže, zda má vytvořit všechny potřebné adresáře (které obvykle neexistují). Poté již následuje kopírování souborů, změny nastavení systému a manipulace s Workplace Shellem. Na závěr je uživatel informován o úspěšné instalaci a poté se program ukončí. Pokud nebyly požadovány změny `CONFIG.SYS`u, lze začít aplikaci ihned používat, v opačném případě je nutný reboot počítače. Obecně lze doporučit, aby se programátoři aplikací snažili změnám souboru `CONFIG.SYS` vyhnout a jeho změny prováděli jen pokud je to nevyhnutelné. Většina aplikací pro OS/2 je totiž konfigurovatelná i bez `CONFIG.SYS`u.

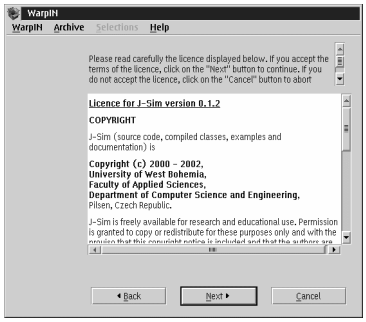
Pokud WarpIN zjistí, že některý balíček (nebo dokonce všechny) instalované aplikace je již nainstalován, bude se chovat zcela normálně[4], navíc ale nabídne uživateli následující možnosti:

- Instalovat – I když je balíček již nainstalovaný, WarpIN ho nainstaluje ještě jednou. Nějaké soubory budou tedy přepsány.

[4]To jest, rozeběhne instalační proces.




Obrázek 7: Těchto obrazovek může být i více, ale jedna nebo dvě obvykle stačí.



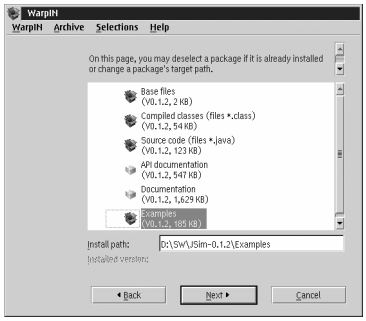

Obrázek 8: Pokud potřebujeme uživateli ukázat delší část textu, i to jde. Text, který může být formátovaný podobně jako HTML dokument, se pak zobrazí v rámu s posuvným ovladačem. Ve skutečnosti se pro formátování takového textu používá malá podmnožina HTML značek.

- Nechat tak, jak je – Balíček nebude znovu nainstalován, ale nebude ani odinstalován.
- Odinstalovat – Balíček bude odinstalován ze systému a databáze instalátoru bude aktualizována. Budou vymazány všechny soubory balíčku a proveden návrat systémové konfigurace do stavu před jeho instalací.

## 1.3 Parametry při spouštění z příkazové řádky

Při spouštění instalátoru WarpIN lze použít následující parametry:

```
warpin [-FIU] [archiv]
```




Obrázek 9: Pokud už je uživatel dostatečně informován a souhlasí s podmínkami naší tolerantní licence, může být připuštěn k výběru balíků. U každého balíku lze zvolit, zda se má instalovat či ne, dále lze zvolit i cílový adresář. Uživatel má k dispozici i krátkou nápovědu ke každému balíku po najetí myši na jeho jméno a seznam volného místa na všech dostupných discích.

Pokud zadáme jméno souboru s instalačním archivem, WarpIN ho otevře a začne instalovat stejně jako kdyby byl spuštěn poklepáním myší na ikoně archivu. Od verze WarpINu 0.9.14 lze vytvářet i *samoinstalovatelné instalační archivy*, které mají příponu `EXE`. Ty lze nainstalovat jednak přímým spuštěním (z příkazové řádky, myší z WPS), lze je ale také zadat jako parametr místo souboru `WPI`.

Pokud není zadán žádný archiv, WarpIN přejde do tzv. *databázového módu*, kdy může uživatel provádět různé údržbové práce, jako je např. deinstalace, kontrola nainstalovaného softwaru apod.

WarpIN podporuje navíc i tyto přepínače, které mírně modifikují jeho chování:

- Pokud není zadán žádný přepínač, WarpIN provede ihned po startu test, zdali už není naistalován. Pokud zjistí že ano, otáže se uživatele, zda by měla být nalezená instalace updatována nebo deaktivována.
- Přepínač `-F` (force – donutit) určuje, že případná nalezená instalace WarpINu bude vždy deaktivována a navíc budou vytvořeny objekty Workplace Shellu pro běžící instalaci. To lze využít, pokud došlo k nepředvídané chybě v objektech Workplace Shellu a již nefunguje automatické spouštění WarpINu při dvojitém kliknutí myší.
- Přepínač `-I` (ignore – ignorovat) se použije tehdy, pokud si přejeme

vynechat počáteční test. Lze použít například v situacích, kdy právě neběží Workplace Shell.

- Přepínač `-U` (upgrade – upgradovat) určuje, že WarpIN vždy přepíše již existující instalaci sebou samým, pokud taková instalace existuje. Pokud neexistuje, dojde k chybě.

# 2 Vytváření distribučních archivů

Nyní se již můžeme pustit do samotného vytváření souborů s instalačními archivy. K tomu budeme potřebovat následující:

- Nějaký dočasný adresář, kam umístíme veškeré soubory, které chceme po nainstalování na disku mít. Můžeme vytvářet i podadresáře. V případě, že chceme mít v našem archivu více balíků, je to dokonce vhodné.
- Instalační skript, což je čistý textový soubor. Jeho vytvoření bude popsáno v kapitole 2.3.
- Program, který vezme všechny požadované soubory a instalační skript a zapakuje je do jednoho výsledného archivu. Tento program je součástí distribuce WarpINu a nemusíme jej tedy shánět. Jmenuje se `WIC.EXE`[5] a práce s ním bude popsána v kapitole 2.4.

## 2.1 Typy archivů

Existují celkem čtyři typy instalačních archivů pro WarpIN. Mimo klasického pasivního archivu s příponou `WPI` je možné od verze WarpINu 0.9.14 vytvářet i tzv. *samoinstalovatelné archivy*, což jsou spustitelné soubory s příponou `EXE`. De facto se jedná také o `WPI` archivy, je k nim ale přidána spustitelná složka, tzv. *stub*.

Lze rozlišit tři typy samoinstalovatelných archivů:

1. Nejjednodušším případem je `WPI` soubor doplněný doplněný pouze stubem, jehož úkolem je najít instalaci WarpINu a té říci, aby daný archiv nainstalovala. Soubor má samozřejmě příponu `EXE`. Stub, který

[5] zkratka z WarpIN Archive Creator

se používá, je obvykle soubor `STUB.EXE` z adresáře WarpINu. Pokud není na cílovém počítači WarpIN nainstalován, instalace takového archivu skončí chybou.

Výhoda oproti klasickým `WPI` archivům není skoro žádná. Dokumentace uvádí, že se tím potlačí problémy se stahováním `WPI` souborů (zřejmě stahování s ASCII režimu). To lze ale jednoduše udělat i správnou konfigurací WWW serveru.

2. Je též možné do archivu zahrnout jakousi *minimální instalaci WarpINu*. Pokud potom není na cílovém počítači WarpIN nainstalován, z archivu se nejdříve rozbalí tato dočasná minimální instalace WarpINu a použije se pro instalaci požadovaného softwaru. Jestliže na cílovém počítači WarpIN nainstalovaný je, použije se pro možné pozdější odinstalování softwaru.

   Tento typ samoinstalovatelného archivu je vhodný v tom případě, kdy nechceme uživatele nutit, aby si před instalací našeho softwaru nainstaloval WarpIN. Pokud to ovšem neudělá, nelze provést pozdější řádnou deinstalaci, na tu je potřeba WarpIN nainstalovaný napevno.

3. Posledním typem archivu je tzv. *plná instalace WarpINu* spolu s instalovaným softwarem. WarpIN je pak nainstalován úplně stejně jako v případě jeho samostatné instalace.

   Příkladem takového archivu je distribuce samotného WarpINu. Ten je od verze 0.9.14 distribuován ve formě `EXE` souboru, dříve to byl pouze `ZIP` soubor, který bylo nutno ručně rozpakovat.

## 2.2 Složení archivu

Jak již bylo řečeno, každý archiv se skládá z jednoho nebo více balíků a z instalačního skriptu. Dále pomineme samoinstalovatelné archivy, které mají navíc stub nebo různě velkou část WarpINu, principy zbytku obsahu jsou stejné pro všechny typy archivů.

Pro kompresi archivů se používá knihovna *bzip2*, kterou používají i některé kompresní programy. Binární formát souborů ovšem není kompatibilní neboť archivy WarpINu mají navíc určité specifické informace.

### 2.2.1 Balíky

Balík je skupina několika souborů (mohou být i v podadresářích), které jsou vždy instalované společně. To znamená, že soubory daného balíku jsou buď instalovány všechny najednou anebo žádný z nich. Balík si lze představit např. jako obyčejný `ZIP` soubor, který uvnitř obsahuje další soubory.

Balík ovšem nikdy není viditelný jako samostatný soubor, protože je vždy obsažen v nějakém archivu. Slouží tak pouze k jakémusi logickému seskupení instalovaných souborů. Z existence balíčků plynou následující vlastnosti WarpINu a instalovaných aplikací:

1. Aplikaci lze vhodně rozkouskovat a nainstalovat pouze její části, o které má uživatel zájem.
2. Každému balíčku lze přiřadit vlastní část konfigurace systému, která se provádí pouze tehdy, když je příslušný balíček instalován.
3. Do globální databáze, spravované WarpINem, se zaznamenávají informace o balíčcích, nikoli o archivech. Pak lze upgradovat či deinstalovat pouze balíčky, nikoliv celé aplikace.

V instalačním archivu má každý balíček svoje číselné označení, tzv. *index balíčku*. Toto číslo přiděluje balíčku tvůrce archivu v instalačním skriptu a obvykle začíná od jedničky. Naproti tomu na cílovém systému – v globální databázi WarpINu – má každý balíček své *unikátní ID*, což je speciální řetězec, skládající se z dodavatele, pojmenování aplikace, pojmenování balíčku a tří čísel pro označení verze.

Díky rozdělení do balíčků tedy není uživatel nucen instalovat celou aplikaci. Aplikace pak může být rozdělena například na část, která je jazykově nezávislá (jeden balíček), a na část jazykově závislou, skládající se z několika balíčků, z nichž stačí nainstalovat pouze jeden.

Teoreticky je také možné updatovat již nainstalovaný balíček bez toho, aby to mělo jakýkoliv vliv na ostatní balíčky, nainstalované z téhož archivu.

Lze se také z jednoho archivu odkazovat na balíčky z druhého archivu, respektive je do archivu logicky zahrnout, aniž by tam byly umístěny fyzicky. Lze tak vytvořit hlavní archiv, odkazující na několik dalších archivů, přičemž uživatel si stáhne pouze ty archivy, které bude skutečně instalovat.

V globální databázi WarpINu jsou uchovávány informace o balíčcích, nikoli o archivech. Tím pádem tam nejsou ukládány ani instalační skripty, protože

ty se vztahují k archivům[6]. Jsou tam však uloženy konfigurační informace balíčků, které jsou v původním instalačním skriptu umístěny.

Na závěr ještě krátké shrnutí:

- Archiv je (velký) soubor, obsahující další soubory – soubory, které budou instalovány za použití balíčků, a instalační skript.
- Balíček není reprezentován jako soubor, je to pouze logické uskupení souborů určených k nainstalování a musí být umístěn v archivu.
- Každý archiv obsahuje nejméně jeden balíček a právě jeden instalační skript, popisující všechny balíčky v archivu.
- Globální databáze, spravovaná WarpINem, pracuje na úrovni balíčků, nikoli na úrovni archivů.

Výše popsaný způsob umožňuje co možná nejflexibilnější správu balíčků.

Na závěr výkladu o balíčcích si ještě uvedeme příklad, jak by mohl být pomocí WarpINu distribuován textový editor, rozdělený do několika balíčků:

1. Musí existovat jeden základní balíček, který bude obsahovat spustitelné soubory a další soubory, které jsou nezbytně nutné pro běh programu. Jak bude uvedeno v kapitole 2.2.2, balíček lze označit jako vždy povinně instalovaný, takže uživatel nemůže jeho instalaci odmítnout.
2. Bude existovat několik balíčků s národní podporou (NLS – National Language Support), z nichž si uživatel vybere jeden.
3. Budou existovat ještě další nepovinné balíčky, obsahující části aplikace, které nejsou nezbytně nutné pro chod editoru.

Všechny tyto balíčky budou umístěny v jednom archivu, který si uživatel stáhne a otevře ve WarpINu. Pokud by však výsledný archiv nabyl obřích rozměrů, může tvůrce aplikace balíčky rozdělit do více archivů. WarpIN pak bude schopen potřebné balíky, na které odkazuje hlavní balík, najít i v dalších archivech. Obecně však nelze tuto strategii doporučit, protože zbytečně dezorientuje uživatele.

[6]Jeden instalační skript se vztahuje k jednomu archivu a naopak.

### 2.2.2 Instalační skript

Každý archiv by měl obsahovat právě jeden *instalační skript*. Pokud tomu tak není, bude si WarpIN při startu stěžovat.

Instalační skript je čistý textový soubor, uchovávající informace o balíčcích obsažených v archivu a informacích, které budou uživateli prezentovány během instalace. Tyto informace jsou rozděleny do jednotlivých obrazovek. Obrazovka může obsahovat prostý text, formátovaný text ve stylu HTML umístěný do rolovací plochy nebo kombinaci obojího. Poslední obrazovka před zahájením instalace obsahuje graficky znázorněnou strukturu balíčků spolu s informacemi ke každému balíčku. Tuto obrazovku tvůrce archivu neprogramuje. Instalační skript je WarpINem čten hned po otevření instalačního archivu.

Instalační skript obsahuje také informace o názvech balíčků, závislosti mezi balíčky, jména balíků, které musí být také instalovány, informace o verzích balíků, informace pro odinstalovaní, změny souboru `CONFIG.SYS`, třídy Workplace Shellu, které musejí být registrovány, atd.

Syntax instalačního skriptu je vzdáleně podobná značkovacím jazykům jako HTML. Používá bloky začínající určitou značkou ve špičatých závorkách a končící tou samou značkou s lomítkem před. Některé značky (otevírací) mohou mít parametry.

Celý skript je umístěn v bloku mezi značkami `<WARPIN>` a `</WARPIN>`. Ten obsahuje blok `<HEAD>` ... `</HEAD>`, specifikující balíky v archivu obsažené, a blok `<BODY>` ... `</BODY>`, specifikující informační obrazovky zobrazované uživateli během instalace.

Následující fragment kódu velmi zběžně ukazuje celkovou strukturu skriptu, která bude podrobně probrána v následující kapitole.

```
<WARPIN>
  <HEAD>
    <!-- Zde je seznam balíků -->
    <PCK INDEX=celé_číslo
         PACKAGEID=popis_balíku
         TITLE=jméno_balíku
         TARGET=cílový_adresář>
    Balíček číslo 1
    </PCK>

    Zde jsou další balíčky...
```

```
  </HEAD>
  <BODY>
    <!-- Obrazovka č. 1 -->
    <PAGE INDEX=1 TYPE=typ_obrazovky>
      <NEXTBUTTON TARGET=2>~Další</NEXTBUTTON>
      <TEXT>
        Vítejte!
      </TEXT>
    </PAGE>
    <!-- Obrazovka č. 2 -->
    <PAGE INDEX=2 TYPE=typ_obrazovky>
      Informace o obrazovce...
    </PAGE>

    Zde jsou další obrazovky...

  </BODY>
</WARPIN>
```

## 2.3 Vytváření instalačního skriptu

Nyní se podíváme podrobněji na to, jak napsat správný instalační skript. Jak již víme, celý skript je umístěn mezi značkami `<WARPIN>` a `</WARPIN>`. Skládá se ze dvou částí: hlavičky a těla. Hlavička je uvozena a ukončena značkami `<HEAD>` a `</HEAD>` a obsahuje výčet balíčků s jejich vlastnostmi. Tělo je zapouzřeno značkami `<BODY>` a `</BODY>` a popisuje se v něm sled uživatelských obrazovek.

Otevírací značka `<WARPIN>` může obsahovat celkem tři nepovinné parametry:

1. `VERSION="hlavní.vedlejší[.revize]"` – Tento parametr udává, že archiv může být zpracován pouze WarpINem, jehož verze je větší nebo rovna zadané verzi. Pokud není uvedena revize, předpokládá se 0. Pokud není parametr uveden vůbec, predpokládá se verze WarpINu 0.9.0 a vyšší.

2. `OS="operační_systém"` – Tento parametr udává minimální verzi operačního systému, která musí být použita pro instalaci archivu. Hodnota parametru může být jedna z následujících: `OS2_2x` (implicitní hodnota), `OS2_3x`, `OS2_4x` nebo `OS2_45x`. Poslední hodnota je používána na

systémech OS/2 Warp 4 s instalovaným fix pakem 13 nebo na OS/2 Warp Server for e-Business, případně na systémech s Merlin/Aurora Convenience Pakem.

3. `CODEPAGE=kódová_stránka` – Tento parametr udává kódování, ve kterém je daný skript napsán. Pokud je kódování jiné než na cílovém systému, WarpIN se pokusí do tohoto kódování přepnout. Pokud se toto nezdaří, pokusí se WarpIN znaky do používaného kódování převádět. Pokud i to selže, uživatel bude varován a budou zobrazeny znaky bez konverze, tudíž nejspíše nesmysly. Implicitní hodnota tohoto atributu je kódová stránka 850.

### 2.3.1 Hlavička

Hlavička mezi značkami `<HEAD>` a `</HEAD>` je vlastně pouhým seznamem balíků, které jsou v daném archivu přítomny. Každý balík je opět uzavřen do značek `<PCK>` a `</PCK>`. Vytváření tohoto popisu balíků je hlavním námětem této kapitolky.

Místo definice balíků lze do hlavičky alternativně umístit blok `<MSG>`[7] a `</MSG>`, který nemá žádné parametry. Úkolem tohoto bloku je informovat uživatele, že balíky umístěné v tomto archivu nelze nainstalovat přímo z tohoto archivu, ale z hlavního archivu, který na balíky tohoto archivu odkazuje. Text mezi značkami by tedy měl uživatele informovat, kde požadovaný hlavní balík sežene. Tento text bude zobrazen ve standardním message boxu a poté se instalace ukončí.

Předpokládejme dále, že píšeme skript pro archiv, který je jediný pro celou naši aplikaci, nebo je určen pro hlavní archiv aplikace. Naše hlavička se tak bude skládat pouze z bloků jednotlivých balíků.

Zvládnutí syntaxe bloku pro popis balíku je pravděpodobně to nejtěžší, s čím se u psaní skriptu potkáte. Syntax je následující:

```
<PCK INDEX=celé_číslo
 PACKAGEID="dodavatel\aplikace\balík\hlavní\vedlejší[\revize]"
 TITLE="název_balíku"
 [EXTERNAL="[REQUIRED|]jméno_archivu"]
 TARGET="adresář"
 [BASE] [FIXED] [SELECT] [NODESELECT]
 [REQUIRES=celé_číslo]...
```

[7] od slova message – vzkaz

```
 [REQUIRES="dodavatel\aplikace\balík
          \hlavní\vedlejší[\revize]"]...
 [CONFIGSYS="výraz[|modifikátory...]"]...
 [REGISTERCLASS="jméno_třídy|cesta_k_DLL"]...
 [REPLACECLASS="staré_jméno_třídy|nové_jméno_třídy"]...
 [CREATEOBJECT="[REPLACE] jméno_třídy|jméno_objektu|
          umístění[|konfigurace]]"]...
 [EXECUTE="[ kontext |] spustitelný_soubor parametry"]...
 [CLEARPROFILE="profil[\aplikace[\klíč]]"]...
 [WRITEPROFILE="profil\aplikace\klíč|řetězec"]...
 [KILLPROCESS="jméno_souboru"]...
>
Popis balíku, jak ho uvidí uživatel během instalace
po najetí myší na jeho ikonu.
</PCK>
```

Na první pohled se to zdá nesmírně komplikované, ovšem většina parametrů je nepovinných, takže vytvoření popisu k nějakému jednoduchému balíku je záležitostí minut. Popis balíku mezi oběma značkami je prostý text, který uživatel uvidí v bublinové nápovědě, pokud přejede myší na ikonu balíku. Tento text se také ukládá do databáze, takže je k dispozici i při případném odinstalování.

Dále si vysvětlíme význam jednotlivých atributů:

- `INDEX=celé_číslo` – Povinný parametr, který udává číslo balíku v jeho archivu. Musí to být celé číslo větší než nula a menší než 30000[8]. Číslo se musí shodovat s číslem balíku zadaným při vytváření archivu programem `WIC.EXE`, viz kapitola 2.4. Tento index musí být v rámci jednoho archivu unikátní.

  Tento index má platnost pouze v daném archivu a není ukládán do databáze. V různých archivech tak mohou existovat různé balíky se stejným indexem. Typicky jsou všechny balíky číslovány od jedné. Pokud chceme identifikovat balík poté, co byl nainstalován, musíme použít jeho jednoznačnou identifikaci: atribut `PACKAGEID`.

- `PACKAGEID="dodavatel\aplikace\balík\hlavní\vedlejší[\revize]"` – Povinný parametr, který jednoznačně identifikuje balík v globální databázi. Díky tomuto atributu je WarpIN schopen zjistit, zdali je balík (případně jeho starší verze) na počítači již nainstalován.

[8] Hodnoty od 30000 jsou rezervovány pro vytvářeč archivů `WIC.EXE`.

Tato identifikace se musí skládat přesně z pěti nebo šesti řetězců oddělených zpětným lomítkem. Řetězce mají tento význam:

- Dodavatel je jméno společnosti nebo jednotlivce. Tento údaj se zobrazuje uživateli.
- Aplikace je jméno aplikace, ke které tento balík přísluší. Toto jméno pak seskupuje jednotlivé balíky stejné aplikace dohromady, jak je vidět při zapnutí stromového pohledu.
- Balík je jméno balíku, které se ukládá do databáze. Uživatel ho vidí jen v databázovém módu, při instalaci vidí uživatel hodnotu atributu `TITLE`.
- Hlavní udává hlavní číslo verze.
- Vedlejší udává vedlejší číslo verze.
- Revize je nepovinný parametr, udávající číslo revize balíku. Pokud není uvedeno, je nastaveno na nulu.

Příklad:

```
PACKAGEID="SuperSoft\Super textový editor\Slovník\1\0"
```

Poznámky k atributu `PACKAGEID`:

1. Při psaní identifikace se rozlišují malá a velká písmena. Tedy SuperSoft není to samé co SUPERSOFT.
2. Při použití ASCII znaků nad 127 by měl být použit atribut `CODEPAGE` u otevírací značky `<WARPIN>`. Jinak není zajištěno, že budou balíky správně porovnávány.
3. V podřetězcích lze používat mezery.

- `TITLE="název_balíku"` – Povinný parametr, udávající jméno balíku, které bude zobrazeno u jeho ikony během instalace. Může být, ale nemusí, stejné jako jeho skutečné jméno, které bude uloženo do databáze. Název balíku je také ukládán do databáze a je tedy zobrazován i během deinstalace.
- `EXTERNAL="[REQUIRED|]archiv"` – Nepovinný parametr, udávající, že popisovaný balík není umístěn v tomto archivu. Specifikovaný archiv je hledán v tom samém adresáři, jako je umístěn současný archiv. Pokud není nalezen, balík je ignorován a není nainstalovan. Pokud je ovšem

zapnutý přepínač `REQUIRED` a balík není nalezen, instalace skončí s chybou.

Při nalezení externího archivu se nečte jeho skript, pouze se kontrolují indexy balíků. To znamená, že se bude pouze hledat balík s indexem, který byl uveden v původním skriptu. Při použití více archivů najednou tak samozřejmě musejí být indexy balíků ve všech archivech jedinečné.

- `TARGET="adresář"` – Povinný parametr, určující plné jméno adresáře (např. `C:\MujSoft`), kam bude balík instalován. Uživatel ho samozřejmě může změnit, pokud není zapnut přepínač `FIXED` – viz dále.

  Lze použít i makra, která umí WarpIN expandovat. Použije se znak dolaru, za ním v závorkách jméno proměnné. WarpIN je expanduje a uživatel už uvidí expandované hodnoty. Sám WarpIN nabízí dvě vlastní makra: `WARPIN_DEFAULTAPPSPATH` pro adresář aplikací a `WARPIN_DEFAULTTOOLSPATH` pro adresář utilit. Tyto adresáře lze nastavit přímo z prostředí WarpINu. Použijeme-li znak otazníku, expanduje se na písmeno disku, ze kterého je zaveden systém.

  Příklady:

  ```
  TARGET="$(WARPIN_DEFAULTAPPSPATH)\SuperWord"
  TARGET="$(WARPIN_DEFAULTTOOLSPATH)\SuperUtility"
  TARGET="?:\OS2\DLL"
  TARGET="$(MMBASE)\DLL"
  ```

- `BASE` – Volitelný přepínač, který určuje, že na instalačním adresáři tohoto balíku závisejí instalační adresáře ostatních balíků. Jestliže uživatel změní instalační adresář u tohoto balíku, všechny ostatní balíky se zkontrolují a pokud mají začátek cesty shodný s adresářem tohoto balíku, automaticky se upraví.

  Příklad:

  ```
  <PCK INDEX=1 ... TARGET="C:\MujSoft" BASE>Balík 1</PCK>
  <PCK INDEX=2 ... TARGET="C:\MujSoft\XXX">Balík 2</PCK>
  <PCK INDEX=3 ... TARGET="D:\Jiny">Balík 3</PCK>
  ```

  Pokud nyní uživatel změní u balíku č. 1 cestu na `D:\Ahoj`, cesta u balíku č. 2 se změní na `D:\Ahoj\XXX` a cesta balíku č. 3 zůstane nezměněna.

- `FIXED` – Volitelný přepínač, který zakazuje uživateli změnit cílový adresář. To je důležité například pro soubory instalované do systémových adresářů.
- `SELECT` – Volitelný přepínač, který zajišťuje, že balík bude na začátku vybrán k instalaci.
- `NODESELECT` – Volitelný přepínač, který zajišťuje, že daný balík bude vždy nainstalován. Uživatel nemůže tento balík z instalování vyřadit. To je užitečné například pro balíky se zapnutým přepínačem `BASE`.

  Tento příznak se neukládá do databáze. Jestliže jsou vyžadovány komplikovanější závislosti mezi balíky, je vhodné použít atribut `REQUIRES`, viz níže.
- `REQUIRES="dodavatel\aplikace\balík\hlavní\vedlejší[\revize]"` nebo `REQUIRES=celé_číslo` – Nepovinný atribut udávající závislost daného balíku na jiném balíku. Jiný balík musí být potom již instalován, aby mohl být nainstalován tento balík, nebo musejí být nainstalovány oba současně.

  První verze specifikuje jedinečnou identifikaci požadovaného balíku, zatímco druhá verze se odkazuje na číslo balíku (index), který je ve stejném archivu jako tento balík. Je to pouze zkratka, WarpIN si interně uloží závislost jako v prvním případě.

  Tyto závislosti se ukládají do globální databáze a jsou tedy stále k dispozici.

  Číslo verze se nebere přesně, nýbrž jako minimální nutná verze. Tak tomu bude i v případě uvedení pouze indexu balíku ze stejného archivu. Ten se převede na jednoznačnou identifikaci balíku a jeho verze bude použita jako dolní limit závislosti.

  Jeden balík může vyžadovat více dalších balíků, atribut `REQUIRES` tak může být veden vícekrát.
- `CONFIGSYS="výraz[|modifikátory...]"` – Tento atribut přidá do souboru `CONFIG.SYS` řádku výraz. Pro jeden balík může být uvedeno více takových atributů, provedou se všechny. Jestliže je tento atribut alespoň u jednoho balíku v archivu, zobrazí WarpIN při instalaci zaškrtávací políčko, kde dovoluje uživateli změny odmítnout.

  Chování závisí na případném použití modifikátorů:

  - Modifikátor `UNIQUE` zaručuje, že v souboru bude zadaný výraz právě jednou.

- Modifikátor `ADDRIGHT` zaručuje, že hodnota napravo od rovnítka bude připojena na konec již definované hodnoty anebo bude výraz do souboru přidán.
- Modifikátor `ADDLEFT` zaručuje, že hodnota napravo od rovnítka bude připojena na začátek již definované hodnoty anebo bude výraz do souboru přidán.
- Modifikátor `ADDTOP` způsobí, že nová řádka bude přidána na začátek souboru `CONFIG.SYS`, nikoli na jeho konec.
- Modifikátory `ADDAFTER(parametr)` a `ADDBEFORE(parametr)` způsobí, že nová řádka bude vložena za, respektive před, řádku zadanou jako parametr.
- Mofifikátor `REMOVELINE` způsobí, že řádka bude ze souboru vymazána.
- Modifikátor `REMOVEPART` způsobí odstranění části řádky za rovnítkem.

Chování atributu `CONFIGSYS` je ještě komplexnější, protože modifikátory lze různě spojovat. Je proto třeba vždy konzultovat originální dokumentaci.

- `REGISTERCLASS="jméno_třídy|cesta_k_DLL"` – Tento nepovinný atribut zaregistruje SOM třídu ve Workplace Shellu. Třída musí být umístěna v `DLL` souboru `cesta_k_DLL`.

  Jestliže má být provedeno takových registrací několik, budou provedeny v tom pořadí, v jakém jsou uvedeny ve skriptu.

  Stejně jako v případě modifikace `CONFIG.SYS`u bude i zde uživatel na jedné obrazovce dotázán, zdali chce registrace opravdu provést.

  U jmen tříd Workplace Shellu se rozlišují malá a velká písmena. Použití tohoto atributu také vyžaduje, aby byl Workplace Shell aktivní. Jinak registrace selže.

  `cesta_k_DLL` musí být plná cesta k danému `DLL` souboru, pokud není umístěn v adresáři, do kterého ukazuje systémová proměnná `LIBPATH`. Opět lze použít makra, takže `cesta_k_DLL` může záviset na adresáři, do kterého byl instalován nějaký balík.

  Příklad:

```
<PCK INDEX=2 ... REGISTERCLASS="XFolder|$(1)\xfldr.dll">
```

- REPLACECLASS="staré_jméno_třídy|nové_jméno_třídy" – Tento nepovinný atribut nahradí již registrovanou třídu nějakou novou třídou. Novou třídu je nejdříve třeba řádně zaregistrovat atributem REGISTERCLASS.

  Atribut opět vyžaduje, aby běžel Workplace Shell. Opět je nutno rozlišovat velikosti písmen. Nahrazení tříd se provedou v pořadí, v jakém byla uvedena ve skriptu, stejně jako registrace.

- CREATEOBJECT="[REPLACE] jméno_třídy|jméno_objektu|umístění-[|konfigurace]]" – Tento nepovinný atribut vytvoří objekt Workplace Shellu. Funguje podobně jako funkce SysCreateObject, kterou lze volat z Rexxu.

  Atribut opět vyžaduje spuštěný Workplace Shell. Požadavky na vytvoření objektu budou provedeny ve stejném pořadí, v jakém jsou ve skriptu.

  Jednotlivé parametry mají následující význam:

  - REPLACE způsobí, že již existující stejný objekt bude nahrazen.
  - jméno_třídy udává jméno WPS třídy, jejíž instanci vytváříme. Všechny potřebné třídy jsou registrovány (viz atribut REGISTERCLASS) před vytvářením objektů.
  - jméno_objektu udává jméno objektu (titulek), jak bude zobrazeno ve Workplace Shellu. Nelze použít uvozovky a čárky.
  - umístění udává, kde bude nově vytvořený objekt umístěn. Může to být buď ID nějaké složky (např. <WP_DESKTOP> pro plochu) anebo plná cesta. Opět lze použít makra.
  - konfigurace specifikuje nastavení, například ID objektu. Zde se můžou použít čárky, je tudíž možné uvést více nastavení najednou. Doporučuje se uvést minimálně ID objektu (např. OBJECTID=<APP_OBJECT>), jinak nebude WarpIN schopen daný objekt při deinstalaci zrušit.

    konfigurace by vždy měla být ukončena středníkem a její poslední částí by měla být identifikace objektu.

  Příklad vytvoření složky a v ní objektu typu program, přičemž použijeme makro k určení plné cesty ke spustitelnému souboru:

```
<PCK INDEX=1
 ...
```

```
CREATEOBJECT="WPFolder|SuperWord Installation|
       <WP_DESKTOP>|OBJECTID=<SUPERWORDFOLDER>;"
CREATEOBJECT="WPProgram|SuperWord|
       <SUPERWORDFOLDER>|EXENAME=$(1)\bin\suprword.exe;
       OBJECTID=<SUPERWORD>;" >
```

- `EXECUTE="[ kontext |] spustitelný_soubor parametry"` – Tento nepovinný atribut vyvolá externí program poté, co jsou všechny balíky nainstalované. To se hodí tehdy, když je potřeba provést ještě nějaká nastavení systému nebo aplikace, která nejsou přímo podporavána WarpINem.

  `spustitelný_soubor` bude spuštěn v oddělené relaci příkazem `CMD.EXE /C spustitelný_soubor parametry`, tudíž lze spustit i Rexx skript. Spuštění se provede až po rozpakování všech souborů, lze tedy spustit i nějaký program, který byl právě nainstalován. Pro určení přesné cesty lze opět použít makra.

  `kontext` je parametr, který udává, co spouštěný program dělá:

  - `CONFIGSYS`: Program modifikuje `CONFIG.SYS`.
  - `REGISTERCLASS`: Program přidává nabo nahrazuje třídy Workplace Shellu.
  - `CREATEOBJECT`: Program vytváří objekty Workplace Shellu.

  V závislosti na nastavení těchto příznaků budou na jedné obrazovce zobrazena zaškrtávací políčka pro povolení daných akcí. Pokud se uživatel rozhodne provedení těchto akcí zakázat, programy nebudou spuštěny. Pokud nemá program nastaven žádný příznak, bude vždy proveden.

  U jednoho balíku lze vyvolat více externích programů.

- `CLEARPROFILE="profil[\aplikace[\klíč]]"` – Atribut určuje, která data se mají při deinstalaci daného balíku vyjmout z `INI` souboru. Netýká se to dat, která budou do profilu zapsána během instalace (viz atribut `WRITEPROFILE`), týká se to dat zapsaných do profilu během používání aplikace. Je velmi vhodné tento atribut použít, aby po se odinstalování aplikace zbytečně nezabíralo místo v `INI` souborech.

  Na jeden balík může být použito více těchto atributů.

  `profil` udává `INI` soubor, se kterým se má pracovat. Může to být `USER` pro uživatelský profil (obvykle `OS2.INI`), `SYSTEM` (obvykle `OS2SYS.INI`)

pro systémový profil anebo cesta k nějakému `INI` souboru. Opět lze použít makra.

`aplikace` udává jméno aplikace, která má být v profilu vyhledána. Pokud není zadána, bude vymazán celý profil. To samozřejmě neplatí v případě `OS2.INI` a `OS2SYS.INI`.

`klíč` udává klíč aplikace, který má být vymazán. Není-li zadán, bude vymazána celá aplikace.

- `WRITEPROFILE="profil\aplikace\klíč|řetězec"` – Atribut určuje, jaká data se mají během instalace zapsat do profilů.

  `profil`, `aplikace` a `klíč` mají stejný význam jako u atributu `CLEARPROFILE`. `řetězec` udává řetězec, který se má do profilu zapsat, to jest hodnotu, která je asociována s daným klíčem.

  Atribut lze použít vícekrát a lze používat makra.

  Pomocí tohoto atributu lze zapisovat pouze řetězce, nikoli binární data. Pokud je potřeba zapsat binární data, musí to udělat sama aplikace anebo externí program spuštěný atributem `EXECUTE`.

  Data zapsána do profilu atributem `WRITEROFILE` budou při deinstalaci automaticky odstraněna.

- `KILLPROCESS="jméno_souboru"` – Nepovinný atribut, určující proces, který má být během instalace a deinstalace zrušen pomocí funkce `DosKillProcess`.

  Těchto atributů může být více na jeden balík.

  Tento atribut je vhodný zejména pro aplikace, které používají démona, který si zamyká soubory nebo zavádí systémové PM-hooks[9] funkcí `WinSetHook`.

  Lze použít i na jakoukoliv jinou aplikaci, abychom se ujistili, že neběží. Není to ale příliš vhodné, neboť ukončovaná aplikace nedostane šanci uložit si svá data.

  Příklad:

  ```
  KILLPROCESS="npswps.exe"
  ```

[9] zaregistrovaná volání zpět ze systému do programu

### 2.3.2 Tělo

Stejně jako hlavička je seznamem balíků, tak tělo skriptu je seznamem uživatelských obrazovek. Tělo začíná značkou <BODY> a končí značkou </BODY>, každá obrazovka začíná značkou <PAGE> a končí značkou </PAGE>.

Jeden blok <PAGE> ... </PAGE> popisuje právě jednu obrazovku. Uživatel může přepínat mezi obrazovkami pomocí tlačítek, která jsou vespodu každé obrazovky. Obrazovky nemusejí ve skutečnosti jít za sebou tak, jak jsou napsány ve skriptu, u každé obrazovky je totiž uvedeno, jaká je ta příští.

Mezi značkami <PAGE> a </PAGE> je třeba uvést text, který má být zobrazen. Text se uzavírá do značek <TEXT> a </TEXT>. Text může být uveden i na obrazovkách, které nejsou textového typu.

Otevírací značka <PAGE> má tyto dva povinné atributy:

1. INDEX=celé_číslo – Udává index obrazovky, který se používá při přeskoku z jedné obrazovky na druhou tlačítkem *Další*. Každá obrazovka musí mít jedinečné číslo, které musí být větší než nula.

2. TYPE="typ_obrazovky" – Udává typ stránky. Musí to být jedna z následujících konstant:

   - TEXT: Tato konstanta určuje obrazovky, na kterých je pouze čistý text. Textových obrazovek lze vytvořit libovolné množství. Velmi se doporučuje, aby alespoň první obrazovka byla tohoto typu. Text se zapisuje mezi značky <TEXT> a </TEXT>.
   - README: Jedná se také o obrazovku zobrazující text mezi značkami <TEXT> a </TEXT>, navíc k tomu však v bloku mezi značkami <README> a </README> zobrazuje formátovaný text, který lze posouvat. Toto se využívá zejména pro zobrazení licencí nebo dlouhých sdělení typu README.

     README blok může mít nastaven atributem FORMAT jeden ze tří následujících typů: PLAIN, FLOW a HTML. Ve formátu PLAIN je text zobrazen tak, jak je uveden ve skriptu. Formát FLOW podporuje zalamování řádků a odstavce. Konečně formát HTML podporuje určitou podmnožinu HTML značek pro formátování textu. Jsou to značky: <P> pro uvození nového odstavce, <BR> pro řádkový zlom uvnitř odstavce, <PRE> a </PRE> pro vysázení bez formátování monospaced fontem, <UL>, </UL>, <OL>, </OL> a <LI> pro nečíslované a číslované seznamy a jejich položky, <DL>, </DL>,

<DD> a <DT> pro seznamy definicí, <B> a </B> pro tučné písmo, <I> a </I> pro italiku, <U> a </U> pro podtržení a konečně <CODE> a </CODE> pro kousky kódu.

- CONTAINER: Jedná se o obrazovku zobrazující všechny instalovatelné balíčky a jejich detailní informace, které se specifikují v hlavičce skriptu. Právě zde uživatel vybírá, jaké balíčky se nainstalují. Tato obrazovka musí být právě jedna. Následující obrazovkou by měla být obrazovka typu CONFIGURE.
- CONFIGURE: Obrazovka tohoto typu dává uživateli možnost vybrat si, jaké konfigurační změny povolí a jaké zakáže. Je zde i tlačítko *Create response file...*, které dovoluje vytvořit soubor s informace o balíkách a konfiguraci a později ho použít při CID instalaci. Tato obrazovka by měla vždy existovat, i když třeba není nic k nakonfigurování.

  Obrazovka tohoto typu je jediná možnost, jak provést tyto akce:

  (a) změnit CONFIG.SYS;
  (b) registrovat nebo nahradit třídy Workplace Shellu;
  (c) vytvořit Response file pro CID instalaci.

  Skutečná konfigurační data nejsou uložena u této obrazovky, ale v definici balíků v hlavičce skriptu. Když dojde řada na konfigurační obrazovku, WarpIN zjistí, jaká zaškrtávací políčka jsou potřebná a případně je zobrazí. Tak může uživatel zabránit změně konfigurace.

  Na této obrazovce by značka NEXTBUTTON měla mít atribut TARGET rovný nule, což zahájí samotnou instalaci. Případně lze zobrazit ještě jednu textovou obrazovku, která bude mít TAGRET u značky NEXTBUTTON nastavený na nulu.

Minimální sled obrazovek by tedy měl být tento:

1. úvodní obrazovka typu TEXT s pozdravem,
2. další obrazovka typu README s licencí,
3. obrazovka typu CONTAINER pro výběr balíků,
4. obrazovka typu CONFIGURE.

## 2.4 Vytváření archivu

Nyní, když již máme připravený skript (a samozřejmě i soubory, které chceme distribuovat), schází nám udělat to poslední: vytvořit archiv. K vytváření archivu poskytuje WarpIN program `WIC.EXE`, který se spouští z příkazové řádky.

`WIC.EXE` pracuje ve čtyřech módech: mód přidávání souborů, mód přidávání skriptu, mód zobrazení archivu a mód rozpakování archivu. Všechny jsou dále probrány.

### 2.4.1 Přidávání souborů do archivu

Syntax spuštění programu `WIC.EXE` pro přidání souborů je následující:

```
wic archiv -a {číslo_balíku [-r] [-cAdresář] maska...}...
       [-uexe | -U[+]]
```

Nejdříve je nutné uvést jméno archivu, který budeme vytvářet, případně upgradovat. Dále nesmíme zapomenout uvést přepínač `-a`, který značí přidávání souborů do archivu.

Pak již stačí napsat číslo balíku, který do archivu chceme přidat a masku souborů, které budou danému balíku patřit. Číslo balíku musí samozřejmě odpovídat nějakému číslu z instalačního skriptu.

Pokud použijeme přepínač `-r`, znamená to, že WIC bude rekurzivně prohledávat všechny podadresáře a přidávat z nich všechny soubory, vyhovující masce.

Použijeme-li přepínač `-c`, ihned následovaný jménem adresáře[10], WIC se před přidáním souborů přepne do tohoto adresáře. Všechny soubory pro daný balík pak musí být samozřejmě uloženy pouze v tomto adresáři nebo jeho podadresářích, nikde jinde. Naopak ale pro více balíků lze použít stejný adresář a soubory z něj rozdělit do balíků podle masky. Pozor na to, že tento adresář nesmí mít nastaven archivní atribut. Má-li ho nastaven, WIC nebude správně fungovat a nepřidá žádné soubory.

Vše mezi složenými závorkami lze opakovat, takže lze naráz přidat několik balíků. Kdykoliv najde WIC na řádce nějaké číslo, ví, že má přidat do archivu nový balík s tímto číslem.

[10] bez mezery

Přepínače `-u`, `-U` a `-U+` vytvoří spustitelný archiv namísto prostého `WPI` archivu. Po přepínači `-u` je nutno ještě uvést tzv. stub (viz kapitola 2.1). Doporučuje se uvést `STUB.EXE`, který je distribuován s WarpINem. Přepínač `-U` automaticky použije právě tento stub. Pak se také přidá balík s číslem 30000, který dovoluje provést instalaci i na systémech, kde není WarpIN nainstalován. Konečně přepínač `-U+` zahrne do archivu ještě balík 30001 a umožní tak dokonce instalaci samotného WarpINu spolu s aplikací.

### 2.4.2 Přidávání skriptu do archivu

Přidání skriptu do archivu se provede zcela jednoduše. Stačí uvést za jménem archivu přepínač `-s`, následovaný jménem skriptu[11]. Tento přepínač lze uvést i v módu přidávání souborů do archivu, takže pak lze vše provést naráz: přidání několika balíků i přidání skriptu.

### 2.4.3 Zobrazení obsahu archivu

Obsah archivu lze vypsat na obrazovku použitím přepínače `-l`, který zadáme po jméně archivu. Nezadáme-li nic dalšího, vypíše se obsah celého archivu. Zadáme-li po přepínači ještě číslo balíku, vypíše se pouze obsah tohoto balíku.

Ke každému balíku dostaneme počet souborů, celkovou velikost nezkomprimovaných souborů a velikost po zkomprimování. Navíc se ještě spočtou statistiky pro všechny balíky dohromady.

Výstup pak může vypadat třeba takto:

```
[D:\Temp]D:\SW\WarpIN\wic.exe JSim-0_1_2.WPI -l
Archive: "JSim-0_1_2.WPI"
WPI revision: 3
Stub size: 0 bytes
Package 1
  1 files, size:      1.775 bytes original
                      1.035 bytes compressed (42% compression)
Package 2
 22 files, size:     55.380 bytes original
                     30.772 bytes compressed (45% compression)
Package 3
 16 files, size:    125.965 bytes original
```

[11] tentokrát s mezerou

```
                    34.866 bytes compressed (73% compression)
Package 4
 40 files, size:    560.584 bytes original
                     96.519 bytes compressed (83% compression)
Package 5
 24 files, size:  1.667.871 bytes original
                    308.693 bytes compressed (82% compression)
Package 6
 93 files, size:    189.700 bytes original
                     94.393 bytes compressed (51% compression)
Summary: 6 package(s), 196 files total
     2.601.275 bytes original
       566.278 bytes compressed (79% compression)
```

V případě výpisu nějakého balíku je výpis podrobnější, dané statistiky se zobrazí pro každý soubor zvlášť.

### 2.4.4 Rozbalení archivu

Rozalení archivu se provede přepínačem `-x` za jménem archivu. Opět lze rozbalení aplikovat pouze na vybraný balík.

Je třeba si uvědomit, že se jedná opravdu o pouhé rozbalení, ne o plnohodnotnou instalaci. Nebude tedy provedeno žádné nastavení konfigurace, uživatel nebude informován o balíčkách apod.

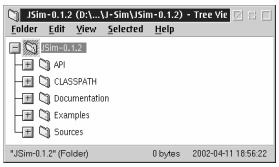
Také je nutno upozornit, že ve většině případů nebudou soubory rozbaleny stejně jako v případě instalace. Tam se totiž umisťují do nově založených adresářů, které jsou uvedeny v instalačním skriptu, většinou pro každý balík jiný adresář. V případě prostého rozbalení se ale tyto adresáře nevytvářejí a všechny soubory se umisťují do aktuálního adresáře, případně do podadresářů, které existovaly při vytváření archivu.

# 3 Příklad archivu: Distribuce simulační knihovny J-Sim

Nyní už zbývá pouze ukázat si reálný příklad vytvoření instalačního skriptu a samotného archivu. V našem příkladě použijeme knihovnu *J-Sim*, psanou v Javě. Jelikož se nejedná o OS/2 aplikaci, nebudeme zapisovat žádná data

do profilů. Budeme ovšem chtít vytvožit složku na pracovní ploše a do ní umístit pár objektů, například odkaz na HTML dokumentaci. Také musíme do proměnné `CLASSPATH` přidat adresář, kde jsou umístěné třídy knihovny.

Obrázek 10 popisuje strukturu knihovny.




Obrázek 10: Struktura knihovny J-Sim verze 0.1.2. V kořenovém adresáři je pouze soubor `Licence.TXT`.

Je velmi vhodné připravit si všechny soubory přesně do těch adresářů, v jakých budou na cílovém systému po nainstalování. Také je vhodné dodržet pravidlo, že jednomu adresáři odpovídá jeden balík. Jednak je to přehlednější, jednak do jednoho balíku nelze umístit soubory z různých podadresářů, pokud to není balík jediný. Naopak to samozřejmě jde, soubory z jednoho podadresáře mohou být rozděleny do více balíků.

## 3.1 Popis balíků

Budeme mít tedy tyto balíky:

1. Základní balík, který bude obsahovat pouze licenci. Při jeho instalaci se vytvoří složka na pracovní ploše a do ní se umístí odkaz na textový soubor s licencí. Zároveň bude tento balík udávat instalační adresář i ostatním balíkům, protože bude mít u sebe přepínač `BASE`. Balík bude povinně instalovatelný, jednak kvůli licenci, jednak také kvůli složce na ploše, kam mohou ostatní balíky přidávat objekty. samozřejmě je také dobré označit základní balík jako povinně instalovatelný, a to prostě proto, že je základní.

   Kód balíku v hlavičce našeho skriptu bude vypadat takto:

   ```
   <PCK
    INDEX=1
    PACKAGEID="University of West Bohemia\J-Sim\Base\0\1\2"
    TARGET="$(WARPIN_DEFAULTAPPSPATH)\JSim-0.1.2"
   ```

```
 TITLE="Base files"
 BASE
 SELECT
 NODESELECT
 CREATEOBJECT="WPFolder|J-Sim 0.1.2|<WP_DESKTOP>|
    OBJECTID=<JSIM_FOLDER>;"
 CREATEOBJECT="WPProgram|Licence|<JSIM_FOLDER>|
    EXENAME=e.exe;PARAMETERS=$(1)\Licence.TXT;
    OBJECTID=<JSIM_LICENCE>;"
>
This package contains J-Sim base files.
You MUST install this.
</PCK>
```

Všimněte si použití makra u druhého atributu `CREATEOBJECT`.

2. Druhý balík bude obsahovat zkompilované třídy a bude implicitně instalován do podadresáře `CLASSPATH`. Počáteční část cesty souhlasí se základním balíkem, může se tedy případně automaticky měnit podle něj. Protože třídy jsou pro knihovnu nutné (knihovna vlastně není nic jiného než tyto třídy), označíme balík jako povinně instalovaný. Navíc v souboru `CONFIG.SYS` přidáme k proměnné `CLASSPATH` adresář, do kterého se třídy nainstalují. To může uživatel samozřejmě odmítnout a zařídit si nastavení po svém.

   Kód balíku následuje:

```
<PCK
 INDEX=2
 PACKAGEID="University of West Bohemia\J-Sim\
    Classes\0\1\2"
 TARGET="$(WARPIN_DEFAULTAPPSPATH)\JSim-0.1.2\CLASSPATH"
 TITLE="Compiled classes (files *.class)"
 SELECT
 NODESELECT
 CONFIGSYS="SET CLASSPATH=$(2) | ADDRIGHT"
>
This package contains J-Sim compiled classes:
Java packages cz.zcu.fav.kiv.jsim and
cz.zcu.fav.kiv.jsim.gui.
You MUST install this.
</PCK>
```

3. Třetí balík obsahuje zdrojové kódy knihovny. Již není povinný a není třeba pro něj nic nastavovat. Nainstaluje se do podadresáře `Sources` a počáteční část cesty se opět bude měnit se změnou adresáře hlavního balíku.

```
<PCK
 INDEX=3
 PACKAGEID="University of West Bohemia\J-Sim\
    Sources\0\1\2"
 TARGET="$(WARPIN_DEFAULTAPPSPATH)\JSim-0.1.2\Sources"
 TITLE="Source code (files *.java)"
 SELECT
>
This package contains J-Sim source files:
Java packages cz.zcu.fav.kiv.jsim and
cz.zcu.fav.kiv.jsim.gui.
</PCK>
```

4. Čtvrtý balík obsahuje API dokumentaci ve formátu HTML. Opět není povinný a jeho cesta bude záviset na hlavním balíku. Povšimněte si vytvoření objektu HTML dokumentace ve složce, která byla vytvořena při instalaci hlavního balíku. Ve skutečnosti je vytvořený objekt typu program (startuje Netscape Navigator), který otevře soubor `index.html` v kořenovém adresáři dokumentace. Opět je použito makro pro určení tohoto adresáře. Lepším řešením by možná bylo použití stínového objektu souboru `index.html`, pak by se dokumentace otevírala programem, který je asociován s HTML soubory (například Mozilla).

```
<PCK
 INDEX=4
 PACKAGEID="University of West Bohemia\J-Sim\API\0\1\2"
 TARGET="$(WARPIN_DEFAULTAPPSPATH)\JSim-0.1.2\API"
 TITLE="API documentation"
 SELECT
 CREATEOBJECT="WPProgram|J-Sim API|<JSIM_FOLDER>|
    EXENAME=netscape.exe;PARAMETERS=$(4)\index.html;
    OBJECTID=<JSIM_API>;"
>
This package contains J-Sim API documentation
```

```
in HTML format, generated by javadoc.
You might need this if you program applications
for J-Sim and need a handbook for quick reference.
</PCK>
```

5. Pátý balík obsahuje uživatelskou dokumentaci a má ty samé vlastnosti jako předchozí balík. Opět instaluje odkaz na dokumentaci ve formátu HTML otevíranou Netscape Navigátorem, k tomu navíc ale vytváří stínový objekt té samé dokumentace ve formátu PostScript.

```
<PCK
 INDEX=5
 PACKAGEID="University of West Bohemia\J-Sim\
    Documentation\0\1\2"
 TARGET="$(WARPIN_DEFAULTAPPSPATH)\JSim-0.1.2\
    Documentation"
 TITLE="Documentation"
 SELECT
 CREATEOBJECT="WPProgram|HTML Documentation|
    <JSIM_FOLDER>|EXENAME=netscape.exe;
    PARAMETERS=$(5)\HTML\index.html;
    OBJECTID=<JSIM_DOC_HTML>;"
 CREATEOBJECT="WPShadow|PostScript Documentation|
    <JSIM_FOLDER>|SHADOWID=$(5)\PostScript\
    JSimDocEnglish.PS;OBJECTID=<JSIM_DOC_PS>;"
>
This package contains J-Sim documentation
in PostScript and HTML formats.
You might need this if you want to learn
how J-Sim works internally or
if you want to get familiar with principles
of discrete simulation.
</PCK>
```

6. Konečně šestý – poslední – balík obsahuje ukázkové příklady v adresáři `Examples`. Balík má opět ty samé vlastnosti jako předchozí. Do složky na pracovní ploše přidává stínový objekt adresáře, ve kterém jsou příklady umístěny. Stínové objekty tak nemusejí být omezeny pouze na soubory.

```
<PCK
 INDEX=6
 PACKAGEID="University of West Bohemia\J-Sim\
    Examples\0\1\2"
 TARGET="$(WARPIN_DEFAULTAPPSPATH)\JSim-0.1.2\Examples"
 TITLE="Examples"
 SELECT
 CREATEOBJECT="WPShadow|Examples|<JSIM_FOLDER>|
    SHADOWID=$(6);OBJECTID=<JSIM_EXAMPLES>;"
>
This package contains examples on using J-Sim.
You might need this if you want
to learn quickly how to use the library.
</PCK>
```

## 3.2 Popis obrazovek

Tak, a první část skriptu máme hotovou. Nyní již zbývá pouze popsat uživatelské obrazovky v bloku `<BODY>` ... `</BODY>`.

Jdeme na to!

1. První obrazovka je úvodní textová obrazovka, na které uživateli poděkujeme, že si chce nainstalovat náš produkt. Po stisku tlačítka *Next* se přesuneme na druhou obrazovku.

```
<PAGE INDEX=1 TYPE=TEXT>
<NEXTBUTTON TARGET=2>~Next</NEXTBUTTON>
<TEXT>
Welcome!

Thank you for downloading and installing J-Sim!

J-Sim is ... (a tak dále)

J-Sim is provided ...

Select "Next" to continue.
Select "Cancel" to abort installation.
</TEXT>
</PAGE>
```

2. Druhá obrazovka není nic převratného, opět je textového typu a trochu více představuje daný produkt. Pokračuje se třetí obrazovkou (kdo by to tušil, že).

```
<PAGE INDEX=2 TYPE=TEXT>
<NEXTBUTTON TARGET=3>~Next</NEXTBUTTON>
<TEXT>
If you have any questions ...

You can let the installation program ...

Select "Next" to continue.
Select "Cancel" to abort installation.
</TEXT>
</PAGE>
```

3. Třetí obrazovka je už jiné kafe. Je typu `README`, obsahuje tedy posuvnou oblast s delším textem. V tomto případě je oblast typu `HTML`, můžeme tedy použít formátování pomocí omezené podmnožiny HTML značek. A samozřejmě stále můžeme zobrazit čistý text nad touto oblastí, který je uzavřen v bloku `<TEXT>` ... `</TEXT>`.

```
<PAGE INDEX=3 TYPE=README>
<NEXTBUTTON TARGET=4>~Next</NEXTBUTTON>
<TEXT>
Please read carefully the licence displayed below.
If you accept ...

Select "Next" to continue.
Select "Cancel" to abort installation.
</TEXT>

<README FORMAT=HTML>
<B><U>Licence for J-Sim version 0.1.2</U></B>
<P>
<B>COPYRIGHT</B>
<P>
J-Sim (source code, compiled classes, ...) is
<P>
<B>Copyright (c) 2000 - 2002</B>,<BR>
```

```
<B>University of West Bohemia</B>,<BR>

...

</README>
</PAGE>
```

4. Konečně na čtvrté obrazovce pustíme uživatele k tomu, aby si vybral balíky a zvolil cesty, kam se budou instalovat. Použijeme tedy obrazovku typu `CONTAINER`. Opět mu pomůže lehkou nápovědou v podobě textu a poté pokračujeme na pátou obrazovku.

```
<PAGE INDEX=4 TYPE=CONTAINER>
<NEXTBUTTON TARGET=5>~Next</NEXTBUTTON>
<TEXT>
On this page, you may deselect a package if it is already
installed or change a package's target path.
</TEXT>
</PAGE>
```

5. Zobrazíme ještě jednu textovou obrazovku, abychom uživatele informovali, že když stiskne tlačítko Install, už to opravdu začne. Povšimněte si, že u značky `NEXTBUTTON` je parametr `TARGET` roven nule. To znamená, že WarpIN začne s instalací. Jaksi jsme ovšem zapomněli na obrazovku typu `CONFIGURE`, kde uživatel může povolit nebo zakázat změny v `CONFIG.SYS`u. V příští verzi tam určitě bude :-)

```
<PAGE INDEX=5 TYPE=TEXT>
<NEXTBUTTON TARGET=0>I~nstall</NEXTBUTTON>
<TEXT>
Press "Install" to begin installing J-Sim 0.1.2
on your computer.
</TEXT>
</PAGE>
```

## 3.3 Použití programu `WIC.EXE`

Nyní již máme i kompletní instalační skript. Zbývá pouze vytvořit distribuční archiv. To se provede následujícím ďábelským příkazem (pozor, vše je ve skutečnosti na jedné řádce):

```
D:\SW\WarpIN\wic.exe D:\Temp\JSim-0_1_2.WPI
   -a
   1 -cJSim-0.1.2 Licence.TXT
   2 -cJSim-0.1.2\CLASSPATH -r *
   3 -cJSim-0.1.2\Sources -r *
   4 -cJSim-0.1.2\API -r *
   5 -cJSim-0.1.2\Documentation -r *
   6 -cJSim-0.1.2\Examples -r *
   -s JSim.WIS
```

Spustíme program `WIC.EXE` a řekneme mu, že chcem pracovat s archivem `JSim-0_1_2.WPI` v adresáři `D:\Temp`. Archiv samozřejmě v tomto čase ještě neexistuje. Poté parametrem `-a` řekneme, že do archivu chceme něco přidávat.

Vytvoříme první balík a přepínačem `-c` se zanoříme do adresáře `JSim-0.1.2`. Tam vezmeme soubor `Licence.TXT` a přidáme ho do archivu. Pak vytvoříme druhý balík, zanoříme se do adresáře `JSim-0.1.2\CLASSPATH` a z něj rekurzivně přidáme všechny soubory. Postup u dalších balíků je už úplně stejný.

Nezapomeňte si ověřit, že žádný z adresářů nemá nastaven archivní atribut!

Nakonec přepínačem `-s` přidáme do archivu instalační skript `JSim.WIS` a jsme hotovi! Nyní již zbývá pouze umístit archiv `JSim-0_1_2.WPI` na WWW nebo na FTP, aby si ho uživatelé mohli stáhnout.

I zde je třeba opatrnosti. Z FTP je nutno archivy stahovat výhradně v *binárním módu*. WWW server lze nakonfigurovat tak, aby binární mód přenosu vnutil. Například u serveru Apache lze přidat do konfiguračního souboru nebo do lokálního konfiguračního souboru `.htaccess` následující řádku:

```
AddType application/x-warpin .wpi
```

# 4 Závěr

WarpIN je bezesporu jeden z nejlepších instalátorů, jaký kdy pro OS/2 existoval. Je uživatelsky velmi příjemný, má intelegentně řešené členění na balíky a závislosti mezi nimi, umožňuje plně konfigurovat systém a v případě potřeby tuto konfiguraci obnovit do stavu před instalací. Má též výbornou podporu

pro různé kódové stránky včetně Unikódu. Vytváření archivů je přímočaré a vcelku jednoduché, zaškolený začátečník[12] by neměl strávit vytvářením archivu více než hodinu.

WarpIN je navíc open source projekt, každý má tedy zdarma přístup ke zdrojovým textům a může je volně upravovat.

Úplným závěrem bych rád popřál WarpINu i všem uživatelům OS/2, aby neměli nouzi o aplikace, které budou tímto skvělým nástrojem instalovány.

# Reference


[UsrGd] WarpIN Team: **WarpIN User's Guide**.
Soubor `wpi_user.inf`.

[PrgGd] WarpIN Team: **WarpIN Programmer's Guide and Reference**.
Soubor `wpi_prog.inf`.

[WWW] OS/2 Netlabs: **WarpIN Home Page**.
WWW stránka `http://warpin.netlabs.org/`.

[XWp] Ulrich Möller: **XWorkplace Home Page**.
WWW stránka `http://www.xworkplace.org/`.


[12] např. někdo, kdo si právě přečetl tento článek