1 Ogólny opis zadania

Należy przygotować pakiet instalacyjny `interp4cmds.tgz` oprogramowania stworzonego w ramach dotychczasowych etapów. Pakiet instalacyjny powinien być pakietem typ GNU stworzonym za pomocą narzędzi takich jak `automake`.

1.1 Uszczegółowienie

Pakiet powinien zawierać wszystkie elementy właściwe dla pakietów typu GNU, a więc pliki takie jak `README`, `INSTALL`, `LICENSE`, skrypt `configure` itp.

Konstrukcja skryptu `configure` musi zapewnić sprawdzenie istnienia w systemie kompilatora oraz biblioteki xerces-c. W wersji podstawowej pakiet powinien umożliwić *zbudowanie* programu `interp4cmds` i zainstalowanie go domyślnie w katalogu `/usr/local/bin`. Skrypt `configure` musi zapewniać możliwość zmiany docelowej instalacji poprzez podanie odpowiedniego parametru opcji `--prefix`.

Oprócz zasadniczego programu, pakiet musi utworzyć co najmniej jedną wtyczkę i zainstalować je w katalogu `$PREFIX/lib`, zmienna `$PREFIX` jest zadawana przy wywoływaniu skryptu `./configure` (jej domyślną wartością jest: `/usr/local`).

Uwaga: To zadanie w całości musi być zrealizowane w ramach zajęć laboratoryjnych. Dostarczenie pakietu instalacyjnego po zajęciach będzie powodowało obniżenie oceny. Bardzo proszę o należyte przygotowanie się do zajęć, aby mogły zakończyć się pełnym powodzeniem. W tym celu należy zapoznać się z materiałami do wykładu oraz z zamieszczonymi do tego zadania przykładami.

2 Uproszczenia

W wersji uproszczonej można zrezygnować z utworzenia wtyczki i jej instalacji.

3 Rozszerzenie

3.1 Mało ambitna wersja rozszerzona

Pakiet powinien umożliwiać instalację dokumentacji w podkatalogu `$PREFIX/doc/interp4cmds` (domyślnie podkatalog `/usr/local/doc/interp4cmds`).

3.2 Ciekawsza wersja rozszerzona

Oprócz dokumentacji pakiet powinien utworzyć wszystkie wtyczki i domyślnie zainstalować je w katalogu `/usr/local/lib`.

4 Przykłady

Do zadania zostały dostarczone przykłady prostych pakietów instalacyjnych. W kartotece z przykładami (panamint: bk/edu/zamp/z4) znajdują się odpowiednio podkartoteki: `1-simple`, `2-inc-src`, `package-interp4cmds` oraz `package-interp4cmds-2libs`. Zawierają one coraz bardziej rozbudowane przykłady tworzenia pakietów instalacyjnych. Przykład z ostatniej kartoteki należy potraktować jako zalążek rozwiązania zadania. Pokazane jest w nim jak tworzyć zasadniczy program oraz dwie biblioteki dzielone.

5 Jak stworzyć własny pakiet

Najwygodniej jest użyć przykładu `package-interp4cmds-2libs` jako zalążka. Bezpośrednio po jego rozpakowaniu trzeba wykonać następujące kroki:

1. Z podkarteki `inc` należy usunąć dotychczasowe pliki nagłówkowe, tzn. pliki: `module1.hh`, `module2.hh` i `wspolny_plik.hh`. W ich miejsce należy umieścić pliki nagłówkowe z podkartoteki `inc` zasadniczego programu oraz bibliotek.

2. W podkartotece `inc` znajduje się plik `Makefile.am`. Jego pierwotny wpis ma postać:

   ```
   noinst_HEADERS= module1.hh \
                   module2.hh \
                   wspolny_plik.hh
   ```

   Jest to lista plików nagłówkowych, które nie podlegają instalacji w systemie, ale zostaną umieszczone w pakiecie instalacyjnym. Chcąc ułatwić sobie pracę i nie robić tego ręcznie można posłużyć się następującym poleceniem:

   ```
   ls -1 *.hh | sed 's/$/ \\/' > /tmp/lista_plikow.txt
   ```

   Następnie zawartość listy należy odpowiednio przekopiować do pliku `Makefile.am`. Uwaga: trzeba pamiętać, aby w ostatniej linii usunąć z jej końca znak '\'.

3. Należy do podkartoteki `src/` przekopiować pliki źródłowe programu. W pliku `src/Makefile.am` należy dopisać źródła własnych modułów i usnąć nazwy przykładowych modułów, tzn.: `main.cpp`, `module1.cpp`, `module2.cpp`.

4. Do podkartoteki `src-lib/` należy przekopiować pliku źródłowe plugin'ow (bibliotek współdzielonych).

5. W pliku `inc/Makefile.am`, tak jak to już było robione wcześniej w przypadku plików nagłówkowych, należy dopisać nazwy plików źródłowych bibliotek i usunąć nazwy przykładowych bibliotek tzn. `libtool1.la` i `libtool2.la`. Zamiast nich należy wpisać nazwy własnych bibliotek. Pamiętać jednak należy, aby użyć rozszerzenia `*.la`, a nie np. `*.so`. W dalszych sekcjach należy odpowiednio zmodyfikować rdzenie nazw, tak aby odpowiadały aktualnym bibliotekom.

6. Po wprowadzonych zmianach należy wykonać polecenia, których lista znajduje się poniżej. Są one podane dla przypadku, gdy modyfikacje zostały przeprowadzone bezpośrednio po rozpakowaniu pakietu.

   1. `libtoolize`
   2. `autoreconf`
   3. `./configure`
   4. `make`
   5. `make distcheck`

   Jeśli dokonamy ponownie nowych zmian, to wracamy do kroku *2.*, tzn. uruchonienie `autoreconf`.

   Jeśli wszystko poszło dobrze, to powinien zostać stworzony pakiet:
   `interp4cmds-2libs-1.0.tar.gz`
   pod warunkiem, że nie została zmieniona nazwa pakietu w pliku `configure.ac`.

6 Weryfikacja poprawności pakietu instalacyjnego

Pragnąć zweryfikować poprawność stworzonego pakietu należy go rozpakować i zainstalować program i biblioteki. Zakładając, że pakiet instalacyjny, np.
`interp4cmds-2libs-1.0.tar.gz`,
jest w kartotece `HOME`, należy wykonać następujące kroki:

```
tar xzf interp4cmds-2libs-1.0.tar.gz
cd  interp4cmds-2libs-1.0/
./configure --prefix=${HOME}/test
make
make install
```

Jeśli wszystko poszło dobrze, to należy przejść do katalogu, w którym wcześniej był uruchamiany program i w którym znajdują się wszystkie niezbędne pliki, tzn. plik konfiguracji, plik gramatyki i plik z opisem działań. Jeśli zrealizowany program współpracował z serwerem graficznym, to należy go uruchomić. Następnie uruchamiamy sam program z odpowiednimi opcjami, przy czym odwołujemy się do kartoteki, w której program został zainstalowany, np.

```
~/test/bin/interp4cmds plik_configuracji.xml plik_dzialan.cmd
```

Jeśli program pracuje tak samo, jak wcześniej, gdy był wywoływany bezpośrednio z kartoteki roboczej, to kończy ten etap pełnym sukcesem.