

Dzień z życia administratora: Checkinstall

Droga ucieczki

Większość programów, którymi zajmujemy się w tej kolumnie, dostępna jest jedynie w postaci kodu źródłowego. Oczywiście nie ma w tym nic złego, ponieważ ich kompilacja i instalacja jest zazwyczaj bardzo prosta. Jednak większość programistów zapomina o tym, że czasem potrzebne jest również polecenie `make uninstall...`

CHARLY KÜHNAST



Programowanie dystrybuowane w postaci kodu źródłowego jako archiwum tar zazwyczaj instalujemy w następujący sposób: czytamy pliki `README` oraz `INSTALL` (o ile istnieją), a następnie wykonujemy magiczne polecenia `./configure`; `make`; `make install` i wszystko powinno działać.

Niestety, szukając i testując nowe programy, instaluję wiele aplikacji, które okazują się być zupełnie nieprzydatne. Dlatego chciałbym mieć możliwość łatwego pozbycia się takich programów. Gdzie są binaria skopiowane poleceniem `make install`? Dlaczego twórca programu nie umieścił w pliku `Makefile` opcji `uninstall`? Rozwiązaniem tego problemu jest `Checkinstall`.

Installwatch

Zamiast uciążliwego ręcznego usuwania niepotrzebnych programów, wykorzystuję do tego celu narzędzie `Installwatch` [1], które informuje mnie, jakie pliki (i gdzie) zostaną skopiowane po wykonaniu polecenia `make install`. Mimo że `Installwatch` jest użyteczny, nie zwalnia nas jednak od konieczności ręcznego usuwania niepotrzebnych plików. Dlatego lepszym rozwiązaniem jest `Checkinstall` [2] – narzędzie oparte na `Installwatch`, ale stanowiące istotny krok naprzód.

Największą różnicą jest to, że `Checkinstall`

nie tylko monitoruje, gdzie zostaną skopiowane pliki wykonywalne przy instalacji, ale także przechwytuje kopiowane pliki i tworzy z nich niewielki pakiet RPM, DEB albo TGZ dla Slackware. Pakiet stworzony w ten sposób można instalować przy użyciu narzędzia (np. `rpm`, `dpkg...`) specyficznego dla danej dystrybucji. Podobnie jest z usuwaniem pakietów – wystarczy użyć standardowego narzędzia do obsługi pakietów.

Praktyczna aplikacja

Chcąc wypróbować działanie `Checkinstall`, użyłem go do kompilacji i instalacji pakietu (`./configure`; `make`; `make install`) `DNRD` (jest to `DNS-Proxy` [3]). Po rozpakowaniu archiwum przechodzimy do nowo powstałego katalogu `dnrd-2.13` i wykonujemy polecenia:

```
./configure
make
```

A teraz, zamiast wpisać polecenie `make install`, uruchamiamy `Checkinstall`:

```
checkinstall --type=rpm
```

`Checkinstall` najpierw zapytał mnie, czy chcę umieścić dokumentację w katalogu `/usr/doc/` – dlaczego nie? Następnie zostało uruchomione polecenie `make install`. Program prze-

chwytuje polecenia wykonywane na systemie plików przez `cp` czy `mv` i przy użyciu przechwyconych danych buduje pakiet RPM. Mogę teraz zainstalować taki pakiet, a jeśli będę się chciał pozbyć `DNRD` w przyszłości, będę mógł to zrobić poleceniem `rpm --erase nazwapakietu`.

Co stanie się, jeśli nasze magiczne polecenie to nie `make install`, ale `make all`? Po prostu dodajemy to nietypowe polecenie do listy opcji `Checkinstall`:

```
checkinstall --type=rpm make all
```

Ponieważ zazwyczaj korzystam z domyślnych ustawień, jeśli chodzi o położenie ścieżek, używam parametru `--default`. Dzięki temu `Checkinstall` nie zamięcza mnie za każdym razem takimi pytaniami. Użytkownicy Slackware muszą jednak uważać na pewną pułapkę – jeśli używasz Slackware w wersji 8.1 lub nowszej, musisz określić parametr `--newslack`, aby `Checkinstall` nie używał starego formatu pakietów TGZ. ■

INFO

[1] `Installwatch`: <http://asic-linux.com.mx/~izto/installwatch.html>

[2] `Checkinstall`: <http://checkinstall.izto.org>

[3] `DNRD`: <http://dnrd.sourceforge.net>