Jak napisać system (auto)aktualizacji oprogramowania? (sheadovas/artykuly/programowanie/jak-napisac-system-autoaktualizacji-oprogramowania/)

Maj 27, 2017 / Programowanie (sheadovas/category/artykuly/programowanie/)

Hej, w dzisiejszym wpisie chciałbym wspólnie z Wami najpierw zastanowić się nad tym jak można zrealizować (automatyczne) aktualizowanie napisanego programu.

W dzisiejszych czasach cenimy sobie wygodę jaką dają automatyczne aktualizatory (w ogóle jest takie słowo?) oprogramowania, dzisiaj mało kto wyobraża sobie sytuację w której w celu pobrania aktualizacji musiałby wejść na stronę producenta, ściągnąć najnowszą paczkę i ponownie zainstalować program.

Jak już wspomniałem we wstępie dzisiaj chciałbym się z Wami zagłębić (poniekąd eksperymentalnie) w temat automatycznych aktualizacji, czy to gier, czy to programów.

Auto-aktualizacje

W tej części chciałbym się głównie skupić „na teorii”, tzn chciałbym abyśmy odpowiedzieli sobie na kilka pytań i zauważyli kilka „szkół” pisania tego typu systemów do aktualizowania oprogramowania.

Czy każdy program powinien posiadać (auto)aktualizacje?

Ciężko jest odpowiedzieć na to pytanie jednoznacznie (a odpowiedź jest dość oczywista). Moim zdaniem taki system powinien posiadać każdy „większy” program, tzn. taki który zamierzamy aktywnie rozwijać także po jego wydaniu.

Wyjątkiem od powyższej reguły powinno być to oprogramowanie, które korzysta z sieci, tak aby w razie potrzeby i luki w bezpieczeństwie można było ją załatać.

Wspólne cechy aktualizatorów

Jeżeli przyjrzymy się oprogramowaniu na rynku to zauważymy kilka wspólnych cech:

1. Najbardziej oczywista: wymagany jest jakiś ogólnodostępny magazyn plików (różnych wersji oprogramowania)
2. Programy posiadają (uwaga! brzmi poważnie): system wersjonowania oprogramowania
3. Aktualizacją zajmuje się dodatkowy program.

Krótki komealntarz:

(1) oczywiste ;)

(2) to zazwyczaj prosty plik tekstowy „VERSION” trzymający numer aktualnej wersji programu. System wersji można także osiągnąć choćby licząc hashe (coś jak git), czy inne twory, ale tym się nie będziemy zajmowali.

(3) jest to konieczny zabieg w przypadku gdy aktualizujemy główny plik wykonywalny, a jak wiadomo: nie można podmienić aktualnie używanego pliku (a przynajmniej nie powinno się tego robić).

Auto-aktualizacje (pseudokod)

Co oczywiste istnieje wiele „szkół” pisania takich aplikacji, jednak tutaj chciałbym zaprezentować w postaci pseudokodu chyba najbardziej popularny schemat (app – główny program, updater – program pomocniczy, „aktualizator”):

```
Pseudokod aktualizacji programu
```

```
# po uruchomieniu app
1. Wczytaj ustawienia, pobierz z sieci najnowszą dostępną wersję
   z repozytorium aktualizacji: ftps://myapp.com/update/version/latest/VERSION
2. Porównaj z lokalną wersją pliku VERSION
 2.1 Jeżeli jest identyczna to kontynuuj normalną pracę programu
 2.2 Jeżeli są różne to wyświetl okienko z pytaniem do użytkownika o zgodę
     o natychmiastową aktualizację
  2.2.1 Jeżeli się zgadza to uruchom updater i zakończ swoje działanie
  2.2.2 (opcjonalnie) zapisz informację o aktualizacji do pliku,
        przed kolejnym uruchomieniem zrób aktualizację (bez pytania o zgodę)

# updater
1. Pobierz listę aktualizacji potrzebnych do przejścia na najnowszą
   (alternatywnie: wygeneruj listę zmian potrzebnych do przejścia na
   najnowszą wersję)
2. Pobierz listę patch'y potrzebnych do zaaplikowania
   (np. listę plików do pobrania)
3. Pobierz aktualizacje do pliku tymczasowego,
   np. /tmp/myapp/<random_hash>/update/
4. Po pobraniu aktualizacji zweryfikuj poprawność poprawnych aktualizacji
   np. przy użyciu funkcji hashujących
 4.1 Jeżeli pobrano jakieś pliki błędnie, to pobierz je jeszcze raz
     i idź do kroku (3)
5. Zaaplikuj aktualizacje (podmień pliki)
6. Uruchom główną aplikację (app)
7. Zakończ swoje działanie (updater) i opcjonalnie wyświetl listę zmian
```

Sama lista kroków jest dość szczegółowa i mam nadzieję że jasna. Pokazuje też dwie koncepcję: tej w której musimy aplikować wszystkie aktualizacje po kolei oraz taką, która posiada skrypt na serwerze tworzący mega-paczki aktualizacji.

Pierwsze podejście (wszystkie aktualizacje po kolei) jest fajne, jeżeli aktualizacje są dość małe i użytkownik podąża za aktualizacjami, w innym przypadku fajnie jest zrealizować także wersję ze skryptem do przygotowywania aktualizacji. Dzięki temu użytkownik może dostać skompresowaną wersję aktualizacji (dzięki czemu sumaryczna aktualizacja będzie znacznie mniejsza).

Podsumowanie

W tym artykule pokazałem dwa podejścia tworzenia aktualizacji (w pseudokodzie). Mam też nadzieję, że „tajemna” wiedza dotycząca tworzenia takich systemów jest już dużo bardziej oczywista.

Tradycyjnie zachęcam do pisania komentarzy, śledzenia bloga przez social-media. Dajcie znać, czy chcielibyście zobaczyć taki system w praktyce ;)

Code ON!