Linux – Zarządzanie Oprogramowaniem - INSTRUKCJA

1 Pakiety i pliki źródłowe

Pakiety binarne

W dystrybucji Ubuntu mają rozszerzenie .deb Pakiet binarne zawierają pliki wykonywalne, konfiguracyjne, strony man oraz info, informacje o prawach autorskich i inną dokumentację. Pakiety te są rozpowszechniane w specyficznym dla systemu Ubuntu (i Debian) formacie archiwum i posiadają rozszerzenie *.deb. Do rozpakowania oraz instalacji (czyli wkopiowania plików składających się na aplikację w odpowiednie miejsca), można wykorzystać narzędzie apt-get lub dpkg.

Pakiety źródłowe

Zawierają źródła aplikacji, wraz ze skryptami, które umożliwiają poprawną konfigurację, kompilację oraz rozprowadzenie aplikacji w systemie. Najczęstszym powodem wykorzystywania tej formy instalacji programu jest brak „paczki" dla danej dystrybucji, chęć zainstalowania najnowszej, rozwojowej wersji danego programu lub konieczność dostosowania programu do specyficznych wymagań sprzętowych bądź systemowych. W pakiecie zawierającym pliki źródłowe znajduje się zwykle plik .dsc opisujący źródłowy pakiet (włącznie z nazwami następnych plików), .orig.tar.gz zawierający oryginalne, niezmodyfikowane źródło w skompresowanym gzipem archiwum tar i plik .diff.gz zawierający zazwyczaj zmiany związane z systemem Ubuntu wprowadzone w oryginalnym źródle.

2 Instalacja oprogramowania z wykorzystaniem programu APT.

Advanced Packing Tool

APT (APT –Advanced Packing Tool) to narzędzie służące do szybkiej i automatycznej instalacji pakietów, zawierające mechanizm zarządzania zależnościami pomiędzy pakietami.
System zarządzania pakietami wykorzystuje do tego celu własną bazę danych – znajdują się w niej informacje, które z pakietów są już zainstalowane na komputerze użytkownika, a które są dostępne. APT został zaprojektowany do współpracy z plikami *.deb. Istnieje również zmodyfikowana wersja apt-rpm, która współpracuje z plikami *.rpm.

Repozytoria

Narzędzie APT do części swoich operacji używa pliku zawierającego listę źródeł, z których mogą być pobierane pakiety. Tym plikiem jest `/etc/apt/sources.list`. W pliku tym znajduje się lista repozytoriów (serwerów na których gromadzone są pakiety). Dodanie danego repozytorium do pliku *sources.list* pozwala nam ściągnąć każdy ze znajdujących się w nim programów. Możemy dodawać repozytoria oficjalne (markowanymi przez Ubuntu), oraz nieoficjalne (repozytoria poszczególnych programów bądź grup developerskich, jak również osób które mają dostęp do szybkich i pojemnych serwerów na których udostępniają wybrane programy).
W każdym z repozytoriów mogą znajdować się pakiety dla różnych wersji dystrybucji systemu, Przykładowo dla dystrybucji Ubutnu 16.04 (Xenial Xerus) wpisujemy xenial, a dla dystrybucji 18.04 (Bionic Beaver) wpisujemy bionic.
Trzecia informacja jaką musimy zawrzeć w pliku *sources.list* - istotna do zdefiniowania repozytorium – to sekcja (ang. *component*). Zarówno w Debianie jak i w Ubuntu pakiety zostały posortowane w pewne działy, dzięki którym łatwiej jest je zidentyfikować. W przypadku obydwu dystrybucji do wyboru mamy sekcje *main, restricted, universe* oraz *multiverse*.

Sekcje (components)

Repozytorium podzielone jest na cztery komponenty: main, restricted, universe i multiverse. W sekcjach tych znajduje się oprogramowanie podzielone ze względu na sposób licencjonowania danego oprogramowania i możliwość wspierania tego oprogramowania przez zespół Ubuntu.
Standardowa instalacja Ubuntu jest zbiorem oprogramowania dostępnego w ramach sekcji main i restricted.

Sekcja „**Main**"
Zawiera aplikacje będące wolnym oprogramowaniem, które może być rozprowadzone bez ograniczeń i jest w całości wspierane przez Ubuntu.

Sekcja „**Restricted**”
Zawiera popularne aplikacje, wspierane przez Ubuntu, jakkolwiek nie zawsze są to aplikacje dostępne na zasadzie licencji open – source.

Sekcja „**Universe**”
Zawiera aplikacje dostępne w ramach darmowego oprogramowania jednakże nie wspierane przez Ubuntu.

Sekcja „**Multiverse**”
Zawiera aplikacje, niedostępne w ramach darmowego oprogramowania oraz nie wspierane przez Ubuntu.

	Wolne oprogramowanie	Zamknięte oprogramowanie
Wspierane	Main	Restricted
Niewspierane	Universe	Multiverse

Dzięki takiemu podziałowi możemy dokładnie wybrać pakiety z których będziemy korzystać, bez konieczności osobnego wgłębiania się w warunki licencji każdej jednej aplikacji. Przykładowo ograniczając plikowi sources.list dostęp do repozytoriów do sekcji main i universe mamy 100% pewność, że nasz system jest w 100% OpenSource i w 100% legalny.

Format wpisu w pliku *sources.list*

Wpisy w tym pliku mają następujący format:

```
deb http://witryna.http.org/ubuntu dystrybucja sekcja1 sekcja2 sekcja3
deb-src http://witryna.http.org/ubuntu dystrybucja sekcja1 sekcja2 sekcja3
```

Na przykład:
deb http://pl.archive.ubuntu.com/ ubuntu bionic main restricted
deb-src http://pl.archive.ubuntu.com/ubuntu bionic main restricted

Pierwszym słowem w każdej linii jest deb lub deb-src, które wskazuje typ archiwum. Mówi ono czy archiwum zawiera pakiety binarne (deb), czyli skompilowane już pakiety, których zazwyczaj używamy czy pakiety źródłowe (deb-src). Inne (poza **http**) typy repozytoriów z których może korzystać apt to: **ftp, file, cdrom, ssh.**

Po zmodyfikowaniu pliku sources.list i zapisaniu zmian, należy wykonać polecenie apt-get update. Jest to konieczne aby APT porał listy dostępnych pakietów ze źródeł wymienionych tym pliku.

Polecenie apt-get

Polecenie apt-get jest dostępne jedynie z konta roota lub dla uprawnionych użytkowników poprzez sudo. Najczęściej wykorzystywanymi poleceniami programu jest komenda *install, remove* oraz *update*.

apt-get update
Pobranie aktualnych informacji o najnowszych pakietach.

apt-get install *nazwa_pakietu*
Zainstalowanie pakietu.

apt-get remove *nazwa_pakietu*
Usunięcie pakietu.
Uwaga: zależności (na przykład dodatkowe biblioteki zainstalowane po to, aby aplikacja funkcjonowała poprawnie) nie są usuwane wraz z pakietem – nawet jeżeli są wykorzystywane tylko przez usuwany pakiet. Nie są za pomocą polecenia apt-get remove nazwa_pakietu nie można również usunąć plików konfiguracyjnych danej aplikacji

apt-get --purge remove nazwa_pakietu
Usunięcie pakietu wraz z jego plikami konfiguracyjnymi.

apt-get dist-upgrade
Aktualizacja całego systemu:

Dodatkowe opcje:
-d - tylko pobranie pliku, bez instalacji
-s - symulacja instalacji
–reinstall - usunięcie i ponowna instalacja pakietu znajdującego się juz w systemie

Polecenie apt-cache

Program używany przez APT do zarządzania bazą danych. Pozwala na odczytywanie takich informacji jak np. podsumowanie pakietu lub wyszukiwanie go spośród listy repozytoriów.

apt-cache show *nazwa_pakietu*
Wyświetlenie informacji na temat pakietu. Kolejno: nazwa pakietu, priorytet, sekcja, rozmiar po zainstalowaniu, właściciel pakietu, architektura pod jaką pakiet został zoptymalizowany, wersja, zależności, pliki zalecane do instalacji, ścieżka na serwerze repozytorium, rozmiar pakietu, suma MD5, opis pakietu, adres pod który można zgłaszać błędy, system dla którego został dany pakiet przygotowany.

apt-cache stats
Wyświetlenie statystki repozytoriów

apt-cache depends *nazwa_pakietu*
Wyświetlenie informacji od jakich innych pakietów zależy dany pakiet.

Polecenie apt-cdrom

apt-cdrom add
Polecenie które umożliwia zdefiniowanie napędu cd-rom jako repozytorium w pliku sources.list.

3 Instalacja oprogramowania z wykorzystaniem programu dpkg

Jeżeli pakiet, który chcemy zainstalować znajduje się już na dysku można do tego celu wykorzystać polecenie dpkg. Polecenie to również dba o zależności.

dpkg –i *nazwa_pakietu*
Zainstalowanie określonego pakietu

dpkg --list
Wyświetlenie krótkiego podsumowania dla każdego pakietu, zawierające dwuznakowe oznaczenie statusu (wyjaśnione w nagłówku), nazwę pakietu, wersję, która jest *zainstalowana*, oraz krótki opis.

dpkg --status *nazwa_pakietu*
Szczegółowe informacje na temat wybranego pakietu

dpkg -r *nazwa_pakietu*
Usunięcie pakietu z systemu wraz z pakietami powiązanymi (jeśli w trakcie usuwania pakietu potwierdzimy chęć ich usunięcia)

dpkg --purge nazwa_pakietu
Usunięcie pakietu wraz z jego plikami konfiguracyjnymi.

4 Instalacja oprogramowania z plików źródłowych

Źródła najczęściej ściąga się w postaci archiwów tar.gz lub tar.bz2. Aby zainstalować taki program najpierw trzeba rozpakować archiwum. Można to zrobić w środowisku graficznym - klikając prawym przyciskiem na archiwum i wybierając rozpakuj, w konsoli z wykorzystaniem programu midnight commander „wchodząc” za pomocą enter-a do danego archiwum i przekopiowując jego zawartość do drugiego kona lub też w konsoli:
Dla archiwów tar poleceniem: tar xfz nazwa_archiwum.tar.gz
Dla archiwów tar.bz2 poleceniem: tar xfj nazwa_archiwum.tar.bz2

Po rozpakowaniu następuje właściwa instalacja pakietów. Zwykle podzielona jest ona na 3 etapy:

- Konfiguracji
- Kompilacji
- Instalacji

Konfiguracja

./configure

Moduł **configure** ustala wstępną konfigurację, sprawdza czy są wszystkie składniki potrzebne do instalacji programu. Często skrypt configure używany jest z opcją **--**prefix zmieniającą domyślny katalog do instalacji plików programu.
Na przykład polecenie **./**configure **--**prefix=/usr, ustala że pliki wykonywalne programu zainstalowane zostaną w katalogu */usr/bin*, biblioteki w */usr/lib* itd.
Wszystkie dostępne opcje wyświetlamy poleceniem **./**configure –help**.**

Ponadto skrypt *configure* generuje dodatkowo jeszcze inne pliki. W skrócie można to przedstawić tak:

- jeden lub kilka plików *Makefile* w każdym katalogu lub podkatalogu
- skrypt *config.status*
- tekstowy plik *config.log* (w przypadku wystąpienia błędu tu należy szukać szczegółowej przyczyny)
- następny skrypt *config.cache* (opcja)
- pliki nagłówkowe "C" (*.h) specyficzne dla danego systemu (opcja)

Kompilacja

make
Rozpoczyna się właściwa kompilacja programu, **make** odnajduje w katalogu plik ***Makefile*** , odczytuje z niego kolejne polecenia i wykonuje. Czas wykonania zależy od wielkości programu i mocy komputera (kiedyś przy bardzo dużych plikach i słabym komputerze kompilacja mogła trwać nawet do kilku godzin.).

Instalacja

make install
Instalacja polega na przekopiowaniu skompilowanych składników aplikacji, bibliotek, plików konfiguracyjnych i innych plików potrzebnych do działania programu w odpowiednie miejsca w systemie plików.

Jeżeli na którymś etapie instalacji występują błędy, odczytaj komunikaty i zlokalizuj przyczynę błędu. Typowe problemy to brak plików nagłówkowych, niewłaściwe wersje bibliotek lub ich brak.
Inne często (choć nie zawsze) dostępne opcje w pliku **makefile** to:

`clean` - sprząta po kompilacji, jest to dobry obyczaj ;)
`uninstall` - usuwa program – niestety nie zawsze występuje, co utrudnia deinstalację

Pakiet build-essentials

Aby móc instalować oprogramowanie ze źródeł konieczne jest aby w systemie zainstalowany był pakiet build-essentials. Najprościej zrobić to wykonując polecenie:
apt-get install build-essential

5 Instalacja oprogramowania z plików źródłowych z wykorzystaniem programu APT

Aby korzystać z tej możliwości w pliku sources.list należy umieścić repozytorium zawierające pliku źródłowe (słowo kluczowe deb-src na początku linii).

Aby pobrać pakiet źródłowy należy wykonać następującą komendę:
apt-get source *nazwa_pakietu*

Komenda ta pobierze trzy pliki: .orig.tar.gz, .dsc i .diff.gz. W przypadku pakietów tworzonych specjalnie dla Ubuntu nie będzie pobierany ostatni plik, a pierwszy zwykle nie ma w nazwie słowa "orig".

Plik .dsc jest używany przez program dpkg-source do rozpakowania źródeł pakietu w katalogu *pakiet-wersja*. Wewnątrz każdego pobranego pakietu istnieje katalog ubuntu/, zawierający pliki niezbędne do stworzenia pakietu .deb.

Aby automatycznie zbudować pakiet w czasie pobierania jego źródeł, do komendy apt-get należy dodać opcję –b:

apt-get -b source *nazwa_pakietu*

Tworzenie „paczek" z pakietów źródłowych ma tą zaletę nad kompilacją ze źródeł, iż ułatwia nam w późniejszym czasie odinstalowanie aplikacji oraz łatwiejsze zainstalowanie pozostałych bibliotek zależnych.

Jeśli nie utworzono pakietu .deb w czasie pobierania jego źródeł, można zrobić to później za pomocą polecenia:
dpkg-buildpackage -rfakeroot -uc -b
uruchomionego wewnątrz katalogu, który został utworzony dla pakietu po jego pobraniu. Aby zainstalować pakiet zbudowany przy pomocy powyższej komendy, należy bezpośrednio wykorzystać program dpkg:
dpkg -i *nazwa_pakietu*

Zwykle specyficzne pliki nagłówkowe i biblioteki współdzielone muszą być obecne, aby można było skompilować pakiet źródłowy. Wszystkie pakiety źródłowe mają pole w plikach kontrolujących, które nazywa się *'Build-Depends'*. Pole te wskazuje, które dodatkowe pakiety są niezbędne, aby można było zbudować pakiet z jego źródeł.
Za pomocą APT w prosty sposób można pobrać te pakiety.
apt-get build-dep *nazwa_pakietu*

Wykorzystując komendę apt-get build-dep zostaną zainstalowane pakiety potrzebne do prawidłowego zbudowania pakietu będącego argumentem tego polecenia. Polecenie to jednakże nie szuka pakietu ze źródłami programu, który ma być skompilowany, dlatego niezbędne jest wcześniejsze pobranie go za pomocą polecenia:
apt-get source *nazwa_pakietu*

Aby sprawdzić jakie pakiety potrzebne są do zainstalowania danej paczki można użyć polecenia apt-cache i odnaleźć informacje konieczne informacje w sekcji *Build-Depends*
apt-cache showsrc *nazwa_pakietu*

6 Przeszukiwanie struktury katalogów

Pliki w systemach linux używane są do przechowywania danych użytkowników oraz reprezentują m. in. niektóre urządzenia systemowe, istotne jest zatem sprawne wyszukiwanie i lokalizowanie plików w strukturze katalogów. Ważne jest, aby każdy użytkownik po zainstalowaniu danej aplikacji potrafił dowiedzieć się gdzie w strukturze katalogów znajdują się pliki składające się na aplikację. W przeciwieństwie do systemów z rodziny Windows, gdzie zazwyczaj wszystkie pliki związane z danym instalowanym programem znajdują się w jednym katalogu (którego nazwa podana jest w trakcie instalacji), w systemach linux pliki składające się na aplikację kopiowane są w różne miejsca struktury katalogów w zależności od funkcji jakie spełniają – pliki konfiguracyjne kopiowane są do katalogu /etc lub /usr/local/etc, pliki wykonywane do katalogów /bin, /usr/bin, /usr/local/bin, biblioteki do katalogu /lib lub /usr/lib, dokumentacja do /usr/share/man, itd…

Wyszukiwanie plików można realizować na kilka sposobów, w zależności od charakteru poszukiwanego pliku i kryteriów wyszukiwania.

which *nazwa_pliku*
Polecenie lokalizuje binarkę (plik wykonywalny) programu, który jest argumentem tego polecenia. W wyniku otrzymujemy pełną ścieżkę do pliku wykonywalnego. Polecenie działa w ten sposób, że przeszukuje ścieżki zawarte w zmiennej PATH pod kątem plików wykonywalnych o nazwie równej parametrowi polecenia.

whereis *nazwa_pliku*
Polecenie lokalizuje binarkę, pliku źródłowe oraz dokumentację (manuala) programu który jest argumentem polecenia. W wyniku otrzymujemy zestaw ścieżek. Działanie tego polecenia polega na przeszukiwaniu standardowych katalogów linuxa w poszukiwaniu plików o nazwie podanej jako parametr polecenia.

Przeszukiwane miejsca to:

/{bin,sbin,etc}

/usr/{lib,bin,old,new,local,games,include,etc,src,man,sbin,
X386,TeX,g++-include}

/usr/local/{X386,TeX,X11,include,lib,man,etc,bin,games,emacs}

locate *nazwa_pliku*
Program wyszukuje pliki, których nazwa zgodna jest ze wzorcem podanym jako argument polecenia. W wyniku otrzymujemy pełne ścieżki do znalezionych plików. Korzystając z locate wyniki otrzymujemy niemal natychmiast, ponieważ program ten korzysta z własnej bazy plików (a nie przeszukuję strukturę katalogów). Baza ta (indeks nazw plików), przeważnie jest aktualizowana raz na dobę - oznacza to, że czasem w wyniku nie zostaną uwzględnione zmiany, które zaszły w systemie plików i katalogów po ostatniej aktualizacji spisu (aktualizacji może zawsze dokonać administrator systemu wykorzystując polecenie updatedb). Zaletą tego programu jest fakt, iż zapytania można budować wykorzystując operatory uogólniające wówczas należy umieścić wzorzec w cudzysłowie.

find [*katalog*] [*krytera*]
Program find umożliwia przeszukiwanie struktury katalogów systemu w poszukiwaniu plików (oraz katalogów) spełniających zadane kryteria. Pierwszym argumentem wywołania polecenia find jest nazwa katalogu, od którego ma się rozpocząć poszukiwanie; drugi argument dotyczy kryteriów jakie mają dotyczyć wyszukiwania. Najczęściej używane kryteria to nazwa pliku, rozmiar, czas modyfikacji, typ urządzenia.
Jednym z ciekawych parametrów tego polecenia jest parametr exec w postaci:
-exec <*polecenie*> [{}] \;,
który pozwala wykonać dowolne polecenia na wyszukanych pozycjach -wówczas należy zastosować jako argument polecenia znaki {}.

Przykładowo:
find /usr/bin –name apt-ge* -exec rm {} \;
przeszuka katalog `/usr/bin` w poszukiwaniu plików lub katalogów o nazwie zaczynającej się na „apt-ge”, a następnie wykona operację usunięcia na wszystkich odnalezionych pozycjach.

ldd *nazwa_programu*
Większość programów instalowanych w systemie linux do działania wykorzystuje własne biblioteki lub też biblioteki współdzielone. Aby dowiedzieć się z jakich bibliotek korzysta dany program można do tego celu wykorzystać polecenie ldd. W wyniku wykonania tego polecenia otrzymujemy listę bibliotek. Każda linijka wyświetlana w wynikowej liście bibliotek bardzo często posiada taką formę:
nazwa_bibliteki_1 => nazwa_biblioteki
Taka forma zapisu oznacza, że plik będący po lewej stronie strzałki jest linkiem symbolicznym do właściwej biblioteki, której nazwa znajduje się po prawej stronie strzałki.
Uwaga: jeżeli korzystamy z polecenia ldd jako argument musi zostać podana pełna ścieżka programu (polecenie ldd nie przeszukuje żadnych ścieżek, ani w szczególności zmiennej PATH w poszukiwaniu pliku będącego jego argumentem).

Źródła i materiały dodatkowe:

APT –Wiki : https://wiki.debian.org/Apt