Systemy Linux i *BSD oraz wolne oprogramowanie

Tomasz Lewicki

WWSIS, Wrocław

maj 2007

Linux jako jądro systemu

Fiński programista Linus Torvalds napisał jądro systemu Linux w 1991 r. na bazie kodu źródłowego systemu operacyjnego Minix. Jądro miało być implementacją systemu Unix na procesory Intel i386 (obecnie działa na procesorach o różnych architekturach, również na systemach wieloprocesorowych). Początkowo Linus pracował nad nim sam, ale po krótkim czasie do pisania kodu przyłączyły się kolejne osoby. W tej chwili tzw. *developerów* są dziesiątki.

Nie ma zgodności co do pochodzenia nazwy Linux. Zazwyczaj tłumaczy się to jako zlepek słów *Linus* (imię twórcy) i *Unix* (czasem *Minix*). Czasami używa się też skrótu rekurencyjnego **L**inux **I**s **N**ot **U**ni**x** („Linux to nie Unix").

Linux jest znakiem towarowym należącym do Linusa Torvaldsa.

Linux a GNU/Linux

Można również spotkać się z pisownią GNU/Linux. GNU to kolejny skrót rekurencyjny: **G**NU is **N**ot **U**nix („GNU to nie Unix"). Określenie GNU/Linux jest forsowane przez organizację Free Software Foundation – założyciela projektu GNU – i wskazuje, że GNU/Linux to jądro Linux z towarzyszącym mu oprogramowaniem GNU oraz podkreśla wkład projektu GNU w narzędzia użyte do budowy rozmaitych dystrybucji Linuksa.

Przeciwnicy powyższego podejścia wskazują, że przeciętna dystrybucja Linuksa zawiera nie tylko oprogramowanie napisane w ramach projektu GNU, ale i dużo innego kodu powstałego poza tym projektem. Proponują oni, by kompletny system operacyjny z oprogramowaniem nazywać po prostu Linux.

Dystrybucje Linuksa

Mówiąc „dystrybucja Linuksa” mamy na myśli jądro (*kernel*) Linuksa wraz z modułami jądra, oprogramowaniem na różnych licencjach (najczęściej wolnych) oraz dokumentacją, zebrane w formie skompilowanej, wstępnie skonfigurowanej i zapisanej na nośniku w taki sposób, by po instalacji można było używać elementów tego zestawu w sposób możliwie bezproblemowy.

Dystrybucje powstają głównie dlatego, iż samodzielne zebranie, skompilowanie i skonfigurowanie potrzebnego oprogramowania zabiera mnóstwo czasu i pociąga za sobą spore koszty. Gotowa dystrybucja stanowi ogromne ułatwienie dla użytkowników końcowych, niezależnie od celu, dla jakiego decydują się oni na używanie Linuksa.

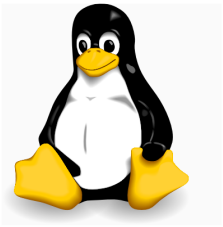
Znane i popularne dystrybucje

Poniższe zestawienie uwzględnia tylko najbardziej znane dystrybucje:

- dla biznesu: RedHat Enterprise, SUSE Enterprise, Mandriva Corporate
- dla domu: Fedora (oparta na RedHat), openSUSE (oparta na SUSE), Ubuntu i jego klony (oparte na Debianie), Mandriva PowerPack
- dla administratorów i zaawansowanych użytkowników: Slackware (najdłużej istniejąca), Debian, PLD, Gentoo, CentOS (oparty na RHEL), System Rescue, Back Track, Network Security Tools
- dla szkół: Edubuntu, Linux Edu CD, Skole, LTSP
- prezentacja możliwości: Knoppix (pierwszy Linux w wersji *live*)

Inne dystrybucje o uniwersalnym przeznaczeniu to np. Damn Small Linux, Linspire, Mepis czy SlaX . Niektóre dystrybucje wydawane są zarówno jako wersje instalacyjne i wersje uruchamiane bezpośrednio z nośnika, tzw. *live* (CD, DVD czy USB).

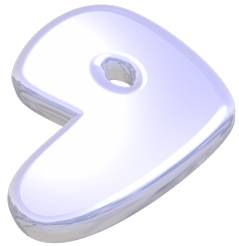
Różne symbole graficzne (logo) ze świata Linuksa

Linux ogólnie: 

GNU: 

SUSE: 

Debian: 

Gentoo: 

PLD: 

RedHat: 

Fedora: 

Ubuntu: 

Rodzina systemów BSD

Skrót BSD rozwija się w Berkeley Software Distribution – odmiany Uniksa napisanej na Uniwersytecie Kalifornijskim w Berkeley, pochodzącej od rozszerzeń tworzonych dla systemu operacyjnego firmy AT&T. Historia systemów z rodziny BSD jest długa i burzliwa. Różne zawirowania prawne spowodowały nawet wstrzymanie rozwoju pewnych gałęzi tej rodziny, ale dzisiaj system ma się dobrze i rozmaite jego odmiany są rozwijane zarówno przez firmy komercyjne (np. Darwin firmy Apple), jak i przez entuzjastów wolnego oprogramowania (np. FreeBSD, OpenBSD czy NetBSD). Istnieją również wersje *live* systemu BSD, np. FreeSBIE czy DragonFly.

UWAGA: *BSD i Linux to dwa odmienne systemy operacyjne!

Rodzina systemów BSD – c.d.

Rodzina BSD rozwija się obecnie dzięki społeczności programistów i użytkowników skupionych wokół trzech głównych gałęzi:

- NetBSD – wywodzi się z systemu 386BSD. Jego główna cecha to ogromna liczba platform sprzętowych (około 50, w tym wiele egzotycznych i rzadko spotykanych), na jakich może działać NetBSD
- FreeBSD – pochodzenie jak w przypadku NetBSD. Cechuje się stabilnością, niezawodnością i wysokim stopniem bezpieczeństwa. Jest znany z bliskiej ideałowi implementacji stosu TCP/IP, działa na nim wiele znanych serwisów internetowych
- OpenBSD – powstał w wyniku rozłamu w zespole tworzącym NetBSD. System ten jest bardzo ceniony za bezkompromisowość w dziedzinie bezpieczeństwa – znaleziono tylko dwie zdalne luki w domyślnej instalacji w całej historii istnienia systemu (od 1996 r.)

Symbole systemów z rodziny BSD

FreeBSD:  oraz 

OpenBSD: 

NetBSD: 

Zarządzanie oprogramowaniem

Różne dystrybucje stosują rozmaite podejście do kwestii zarządzania oprogramowaniem, tzn. jego dodawania, konfigurowania i usuwania. Najwygodniejsze w użytkowaniu są tzw. **pakiety**, czyli skompilowane i wstępnie skonfigurowane „paczki" zawierające programy wykonywalne, biblioteki, dokumentację, pliki konfiguracyjne, ikony itp. Pakiety mogą mieć różną postać w zależności od dystrybucji (*RPM* w RedHat czy SUSE, *DEB* w Debianie).

Innym sposobem rozpowszechniania oprogramowania są archiwa w formacie *.tar.gz* (*.tgz*) – to podejście zastosowano w Slackware. System FreeBSD stosuje kolekcje tzw. **portów**, kompilowanych w trakcie instalacji. Taki pomysł na zarządzanie oprogramowaniem został włączony do linuksowej dystrybucji Gentoo.

Repozytoria

Zasoby oprogramowania są przechowywane w tzw. **repozytoriach**, utrzymywanych przez twórców danej dystrybucji. Oprogramowanie jest tam podzielone tematycznie, z rozróżnieniem na kolejne wersje dystrybucji oraz architekturę procesora. Jako repozytorium bywa też traktowany nośnik danych lub zasób w sieci lokalnej.

Repozytoria znacznie ułatwiają zarządzanie oprogramowaniem. Dla dużych dystrybucji z silnym wsparciem społeczności skupionej wokół różnych projektów najrozsądniejszym wyjściem jest instalacja oprogramowania w postaci gotowych pakietów wprost z codziennie aktualizowanych repozytoriów. Dzięki temu łatwo jest zachować porządek w systemie operacyjnym. Jeśli przy instalacji systemu określimy repozytoria dla niego, aktualizacja do najnowszych wersji oprogramowania i bibliotek systemowych jest banalnym, niemalże automatycznym zadaniem.

Zależności

Linux i systemy uniksowe w ogólności są zbudowane na zasadzie modularnej: z małych elementów, z których każdy ma przypisane pewne niewielkie zadanie do wykonania. Pozwala to m. in. na oszczędzenie pracy twórcom oprogramowania. Zamiast „wyważać otwarte drzwi" pisząc po raz kolejny np. bibliotekę realizującą pewne czynności, programiści korzystają z bibliotek napisanych przez kogoś innego i przetestowanych przez tysiące użytkowników. Dzięki temu jedna biblioteka może być **współdzielona** przez dziesiątki programów i innych bibliotek, które są od niej **zależne**.

Dzięki takiemu podejściu unika się powielania rozwiązań, oszczędza miejsce na nośniku i zapobiega bałaganowi w plikach systemowych. Czasami może to jednak doprowadzić do tzw. **piekła zależności**, czyli niezgodności instalowanej właśnie wersji pakietu z innymi pakietami, które są już zainstalowane na dysku. W sukurs użytkownikowi przychodzą repozytoria i menadżery oprogramowania danej dystrybucji.

Oprogramowanie w postaci źródłowej

W przypadku mniej popularnych projektów zdarza się, że dla wielu dystrybucji nie są dostępne gotowe, skompilowane pakiety. Prawie zawsze dostępne są jednak pliki źródłowe do samodzielnej kompilacji – taka sytuacja jest wypadkową ilości osób pracujących nad projektem, czasu poświęcanego na jego rozwój, zainteresowania projektem ze strony użytkowników, a czasem po prostu lenistwa autorów.

Oprogramowanie w postaci źródłowej jest w znacznym stopniu niezależne od platformy sprzętowej i architektury procesora. Na użytkowniku spoczywa ciężar określenia parametrów wejściowych, odpowiednie skonfigurowanie własnego systemu, kompilacja bibliotek i plików wykonywalnych oraz umieszczenie ich w odpowiednich miejscach w systemie operacyjnym. Często zdarza się, że twórcy oprogramowania automatyzują dużą część procesu instalacji takiego oprogramowania.

Oprogramowanie w postaci binarnej

Bywa, że oprogramowanie nie jest dostępne w postaci źródłowej i/lub pakietów, zaś autorzy rozpowszechniają je wyłącznie w postaci skompilowanych programów i bibliotek. Ten sposób instalacji jest doskonale znany z systemów Microsoft Windows. Takie podejście wymuszają sami producenci oprogramowania, którzy z rozmaitych przyczyn nie chcą ujawniać kodu źródłowego swoich programów.

Przykłady:

- sterowniki do kart graficznych NVidia
- komunikator Skype
- czytnik plików PDF Acrobat Reader firmy Adobe
- maszyna wirtualna Java firmy Sun
- odtwarzacz plików Flash firmy Macromedia
- przeglądarka internetowa Opera

Kilka zdań o licencjach

Dystrybucje Linuksa i systemy *BSD mogą istnieć dzięki tzw. **wolnym licencjom** – takim, które pozwalają na tworzenie i wykorzystywanie oprogramowania do dowolnych celów oraz dzielenie się nim bez ograniczeń. Linux wykorzystuje głównie licencję GNU GPL (General Public License; Ogólna Licencja Publiczna), a systemy z rodziny BSD mają własną licencję BSD (Berkeley Software Distribution License).

Są to najbardziej znane licencje ze świata wolnego oprogramowania, czyli **FOSS** (**Free/Open Source Software**), określanego również skrótem FLOSS (Free Libre/Open Source Software).

Licencja GNU GPL

Została sformułowana w 1998 r. (wersja 1), poprawiona i uzupełniona w 1991 r. (wersja 2). Obecnie trwają ostatnie prace nad wersją trzecią.

Założenia:

- użytkownik ma prawo korzystać z oprogramowania w dowolnym celu
- użytkownik może badać, w jaki sposób program działa i zmieniać go
- użytkownik ma prawo rozpowszechniać kopie oprogramowania
- użytkownik może wprowadzać do oprogramowania własne poprawki i upubliczniać je

Drugi i ostatni warunek implikują konieczność dostarczania kodu źródłowego dzieł opartych na licencji GNU GPL każdemu zainteresowanemu owym kodem odbiorcy programu, co nie jest tożsame z OBOWIĄZKIEM publicznego udostępniania kodu.

Licencja GNU GPL – c.d.

GPL bywa nazywana „licencją wirusową”, ponieważ wymaga, by każda praca powstała na bazie dzieła objętego licencją GPL była również objęta licencją GPL („zarażona” GPL). Oznacza to, że kod programu lub biblioteki objętej licencją GPL nie może być włączony do dzieła na innej licencji.

Sytuacja odwrotna jest dopuszczalna: kod programu objętego inną niż GPL licencją może być włączony do dzieła objętego GPL. Wtedy całość również ma być objęta GPL. Oczywiście licencja włączanego fragmentu musi zezwalać na włączanie do innych programów czy bibliotek.

FOSS w zastosowaniach serwerowych i administracyjnych (nieliczne przykłady)

- serwery WWW: Apache (ok. 60% serwerów Web w Internecie)
- serwery plików i drukarek: Samba+CUPS, ProFTPd, vsFTPd
- bazy danych: PostgreSQL, MySQL, SQLite, Firebird
- serwery pocztowe: Postfix, Sendmail, Exim, Qmail
- filtry pakietów: IPTables, IPFW, PF
- zarządzanie użytkownikami i dostępem do zasobów: OpenLDAP, FreeRADIUS
- systemy IDS/IPS, audyt bezpieczeństwa: Snort, Nessus
- inne: systemy Wiki, Squid, ClamAV, Spamassassin, Bind

Przykłady dzieł na wolnych licencjach

- systemy operacyjne: rozmaite dystrybucje Linuksa, *BSD, OpenSolaris
- języki programowania: GNU C/C++, Perl, Python, GFortran
- środowiska programistyczne: Mono, Eclipse
- systemy wyświetlania: XFree86/X.Org
- środowiska graficzne: KDE, Gnome, Xfce, Blackbox
- przeglądarki: Mozilla, Firefox, Konqueror
- klienci *e-mail*: Thunderbird, Evolution, KMail

Przykłady programów na wolnych licencjach – c.d.

- grafika i skład tekstu: Gimp, Blender, LaTeX, Scribus, Inkscape
- pakiety biurowe: OpenOffice, KOffice, Gnome Office
- edytory: Vi, Emacs, Quanta Plus, Abiword
- komunikatory: Kadu, TleenX, Jabber
- kodeki audio i wideo: Ogg, XviD
- systemy zarządzania treścią: Drupal, Plone, Zope, Joomla!, Moodle
- gry: Freeciv (klon Civilization), Battle for Wesnoth, Doom, Quake I–III
- inne: OpenSSH, GnuPG

LAMP

Istnieje zgrabne określenie na platformę serwerową użytą do budowania serwisów Web o różnej skali, złożoną z „cegiełek” objętych wolnymi licencjami. Skrót **LAMP** pochodzi od pierwszych liter nazw owych czterech elementów:

Linux	system operacyjny
Apache	serwer WWW
MySQL	baza danych
PHP, Perl, Python	języki skryptowe

LAMP

Istnieje zgrabne określenie na platformę serwerową użytą do budowania serwisów Web o różnej skali, złożoną z „cegiełek” objętych wolnymi licencjami. Skrót **LAMP** pochodzi od pierwszych liter nazw owych czterech elementów:

Linux	system operacyjny
Apache	serwer WWW
MySQL	baza danych
PHP, Perl, Python	języki skryptowe

LAMP

Istnieje zgrabne określenie na platformę serwerową użytą do budowania serwisów Web o różnej skali, złożoną z „cegiełek” objętych wolnymi licencjami. Skrót **LAMP** pochodzi od pierwszych liter nazw owych czterech elementów:

Linux	system operacyjny
Apache	serwer WWW
MySQL	baza danych
PHP, Perl, Python	języki skryptowe

LAMP

Istnieje zgrabne określenie na platformę serwerową użytą do budowania serwisów Web o różnej skali, złożoną z „cegiełek” objętych wolnymi licencjami. Skrót **LAMP** pochodzi od pierwszych liter nazw owych czterech elementów:

Linux	system operacyjny
Apache	serwer WWW
MySQL	baza danych
PHP, Perl, Python	języki skryptowe

LAMP

Istnieje zgrabne określenie na platformę serwerową użytą do budowania serwisów Web o różnej skali, złożoną z „cegiełek” objętych wolnymi licencjami. Skrót **LAMP** pochodzi od pierwszych liter nazw owych czterech elementów:

Linux	system operacyjny
Apache	serwer WWW
MySQL	baza danych
PHP, Perl, Python	języki skryptowe

LAMP

Istnieje zgrabne określenie na platformę serwerową użytą do budowania serwisów Web o różnej skali, złożoną z „cegiełek” objętych wolnymi licencjami. Skrót **LAMP** pochodzi od pierwszych liter nazw owych czterech elementów:

Linux	system operacyjny
Apache	serwer WWW
MySQL	baza danych
PHP, Perl, Python	języki skryptowe

LAMP

Istnieje zgrabne określenie na platformę serwerową użytą do budowania serwisów Web o różnej skali, złożoną z „cegiełek” objętych wolnymi licencjami. Skrót **LAMP** pochodzi od pierwszych liter nazw owych czterech elementów:

Linux	system operacyjny
Apache	serwer WWW
MySQL	baza danych
PHP, Perl, Python	języki skryptowe

LAMP

Istnieje zgrabne określenie na platformę serwerową użytą do budowania serwisów Web o różnej skali, złożoną z „cegiełek” objętych wolnymi licencjami. Skrót **LAMP** pochodzi od pierwszych liter nazw owych czterech elementów:

Linux	system operacyjny
Apache	serwer WWW
MySQL	baza danych
PHP, Perl, Python	języki skryptowe

LAMP

Istnieje zgrabne określenie na platformę serwerową użytą do budowania serwisów Web o różnej skali, złożoną z „cegiełek” objętych wolnymi licencjami. Skrót **LAMP** pochodzi od pierwszych liter nazw owych czterech elementów:

Linux	system operacyjny
Apache	serwer WWW
MySQL	baza danych
PHP, Perl, Python	języki skryptowe

Zalety wolnego oprogramowania

- każdy, kto chce testować i rozwijać oprogramowanie ma taką możliwość dzięki dostępności kodu źródłowego
- FOSS opiera się głównie na otwartych standardach – ich konsekwentne stosowanie umożliwia osiągnięcie pełnej kompatybilności różnych produktów
- dzięki pracy wielu *developerów* błędy są dostrzegane i naprawiane szybciej niż w przypadku zamkniętego oprogramowania („im więcej oczu patrzy, tym mniej istotne są błędy")
- łatwo uzyskać wsparcie z różnych źródeł (społecznościowych [*community*] i komercyjnych)
- projekty mogą dzielić się i rozwijać niezależnie – wygra lepszy

Wady wolnego oprogramowania

- projekty rozwijają się w różnym tempie, niektóre zostają zarzucone na pewnym etapie rozwoju, ewentualnie dają początek innym
- zazwyczaj nie ma nikogo, kto narzucałby tempo pracy i rozliczał wyniki – nad projektami w ogromnej większości pracują wolontariusze
- ludzie zasilają i opuszczają projekty z różnych przyczyn – kod może mieć różną jakość
- członkowie zespołu pracującego nad danym projektem mogą kierować się odmiennymi celami i różnie zapatrywać na elementy projektu
- nikt nie gwarantuje, że rozpoczęty projekt zakończy się sukcesem

„I kto za to płaci...?”, czyli źródła finansowania

- firmy, którym zależy na stabilnym, wydajnym, energicznie rozwijanym oprogramowaniu
- firmy, które poważnie rozpatrują tzw. całkowity koszt posiadania (TCO = Total Cost of Ownership)
- producenci sprzętu, którym zależy na zwiększeniu sprzedaży
- ludzie i firmy, którzy zarabiają na FOSS (wdrożenia, opieka, szkolenia, pomoc techniczna, dodatki pisane na zamówienie) – sposób okazania wdzięczności i... podtrzymanie źródeł swojej egzystencji
- *developerzy* bywają zatrudniani przez duże firmy, by pracowali nad określonymi projektami lub pisali sterowniki dla sprzętu
- społeczność, czyli użytkownicy (gadżety, płyty, dobrowolne datki, wpływy z reklam na stronach WWW poświęconych FOSS)
- projekty wspomagają się wzajemnie, również finansowo

Przydatne adresy

- http://www.opensource.org
- http://www.gnu.org
- http://www.distrowatch.com
- http://www.jakilinux.org
- http://7thguard.net
- http://pl.wikipedia.org/wiki/Kategoria:FLOSS
- http://www.linux.pl
- http://linuxnews.pl
- http://www.linuxsoft.cz/pl
- http://bsdzine.org
- http://www.meetbsd.pl
- http://www.bsd4u.org