

Kurs administrowania systemem Linux

Zajęcia nr 13: Porty, pakiety i dystrybucje

Instytut Informatyki Uniwersytetu Wrocławskiego

29 maja 2025

Tworzenie

- Narzędzia do tworzenia: kompilatory
- Narzędzia do współpracy: systemy kontroli wersji

Pozyskiwanie

- Archiwa z kodem źródłowym `*.tar.gz` (przeważnie gotowy *release*)
- *Download* z VCS, np. CVS lub Git (przeważnie wersja rozwojowa)

Instalacja

- Ustalona prosta procedura

Klient-server

- RCS (Revision Control System), 1982, obecnie w ramach projektu GNU
- CVS (Concurrent Versioning System), 1986, licencja GNU
- Subversion (SVN), 2000, licencja Apache

Rozproszone

- BitKeeper, 2000
- Darcs, 2003 (Haskell)
- GNU Bazaar, 2006 (Canonical)
- Git, 2005, Torvalds

- GNU Compiler Collection (GCC)
- GNU Debugger (GDB)
- GNU Binutils (ld, gas, gprof, readelf, elfedit, objdump i in.)
- GNU Coreutils (ls, install i wiele in.)
- GNU Bison
- GNU make
- GNU m4
- GNU build system (autotools): Autoconf, Automake i Libtool

Źródła oprogramowania od strony użytkownika

```
./configure  
make  
sudo make install
```

Skrypt configure:

- Sprawdza obecność niezbędnych programów, w tym kompilatorów, serwisów, bibliotek, plików nagłówkowych oraz typów i struktur w nich zdefiniowanych.
- Sprawdza konfigurację systemu, możliwości kompilatora itp. i wybiera właściwe wersje kodu do kompilacji.
- Generuje odpowiedni plik Makefile.
- Dawniej pisany ręcznie. Obecnie generowany automatycznie *przez autora programu* za pomocą GNU Autotools.
- Cele Makefile: default, install, uninstall, check, clean, distclean i in.
- Instalacja zwykle za pomocą programu install (GNU Coreutils).

Główny system plików / (montowany przez initramfs)

`/bin`, `/sbin` podstawowe programy wykonywalne (odp. dla wszystkich użytkowników i *root*-a)

`/lib`, `/lib64`, `/lib32`, ... podstawowe biblioteki współdzielone

`/etc` pliki konfiguracyjne

`/root` katalog domowy użytkownika *root*

`/sys`, `/proc`, `/dev`, `/run` psudosystemy plików

Katalogi, które mogą być wydzielone w postaci osobnych systemów plików montowanych w późniejszych fazach rozruchu

`/usr` drugi poziom hierarchii (`/usr/{bin,sbin,lib*,share,include,src,...}`)

`/usr/local` trzeci poziom hierarchii — lokalny

`/tmp` pliki tymczasowe

`/var` część zmienna systemu

`/home` katalogi domowe użytkowników

`/mnt`, `/media` punkty montowania (tymczasowy, urządzeń wymiennych)

`/srv`, `/opt` dane dla serwerów, oprogramowanie opcjonalne

Gdzie instalować, żeby nie namieszać?

- `./configure --prefix=/usr/local/`
- `./configure --prefix=/opt/prognome-X.Y.Z/` plus zbiór `ln -s` z `/usr/local/`

Pseudopakietyzacja

- `checkinstall --pkgname=prognome --pkgversion=X.Y.Z make install`
- `dpkg -L prognome`
- `dpkg -r prognome` albo `apt remove prognome`
- Działa dla deb, rpm, tgz (Slackware)
- Uwaga na pułapki!

GNU Build System (Autotools)

- GNU Autoconf
- GNU Automake
- GNU Libtool
- Niezbędne pakiety pomocnicze:
 - GNU M4
 - GNU Autoconf-archive
- Plus zbiór bibliotek ułatwiających przenoszenie kodu (gnulib, libiberty i in.)

Twórca programu

- `autoscan: *.c, *.h → configure.ac`
- `autoheader: configure.ac → config.h.in`
- `automake: configure.ac, config.h.in, Makefile.am → Makefile.in`
- `aclocal: configure.ac → aclocal.m4`
- `autoconf: configure.ac, aclocal.m4 → configure`

Użytkownik programu

- `configure: () → config.status`
- `config.status: config.h.in, Makefile.in → config.h, Makefile`
- `make: *.c, *.h, config.h → program`

Autotools

- Duży, skomplikowany system, trudny do nauczenia.
- Wiele starego kodu, sporo ograniczeń.
- Dodawanie własnych testów wymaga użycia M4, który jest nieintuicyjny i mało znany.
- Ukrywa problem przenośności kodu w nieczytelnym i długim skrypcie.
- Ogranicza przenośność do systemów posiadających Bash-a.

Alternatywy

- CMake
- Meson
- SCons
- pkg-config (zarządzanie bibliotekami)

Jak wprowadzać i publikować zmiany?

GNU Diffutils

- `cmp`
- `diff`
- `diff3`
- `sdiff`
- `patch`

`patch` pozwala nałożyć zmiany na pliki źródłowe otrzymane za pomocą `diff`:

- `diff -u file.orig file > patch.diff`
- `patch < patch.diff`
- `patch -R < patch.diff`

Cele

- Automatyzacja instalacji systemu
- Automatyzacja instalacji i konfiguracji oprogramowania
- Automatyzacja aktualizacji oprogramowania
- Wsparcie dla konfiguracji i współdziałania komponentów systemu
- Modularyzacja oprogramowania
- Organizacja *współdzielenia komponentów* (biblioteki współdzielone itp.)

Składowe dystrybucji

- Instalatory
- *Pakiety źródłowe* (*porty*, *upstream* + *patches*): automatyzacja kompilacji kodu źródłowego
- *Pakiety binarne*: prekompilowane komponenty systemu
- Dokumentacja i wsparcie

Zalety

- Wspiera filozofię *Open Source*.
- Upraszcza dostęp do kodu źródłowego (analizowanie działania programu itp.).
- Ułatwia wprowadzanie drobnych poprawek do programów.
- Pozwala na optymalizację kodu wynikowego dla konkretnego modelu procesora.
- Ułatwia współdzielenie bibliotek.

Wady

- Wymaga zainstalowania pełnego zbioru narzędzi deweloperskich (kompilatory, pliki nagłówkowe itd.).
- Utrudnia i wydłuża proces instalacji i aktualizacji oprogramowania.

Optymalizacja zależna od architektury na przykładzie GCC

Wybrane opcje `-march` i `-mtune` dla architektury 386 i x86-64

pentium pentium-mmx pentiumpro pentium2 pentium3 pentium3m pentium-m
pentium4 pentium4m prescott nocona core2 nehalem westmere sandybridge
ivybridge haswell broadwell bonnell k6 k6-2 k6-3 athlon athlon-tbird
athlon-4 athlon-xp athlon-mp k8 opteron athlon64 athlon-fx k8-sse3
opteron-sse3 athlon64-sse3 amdfam10 barcelona bdver1 bdver2 bdver3 bdver4
btver1 btver2 winchip-c6 winchip2 c3 c3-2 geode

- `-march=arch` — wybiera zestaw instrukcji dla konkretnego procesora; implikuje `-mtune=arch`.
- `-march=arch` — włącza optymalizacje dla konkretnej architektury.
- `-march=native` — generuje kod specyficzny dla procesora na którym odbywa się kompilacja.
- Domyślnie na x86-64: `-march=x86-64 -mtune=generic`.

Optymalizacja w GCC — przykład

```
-march=x86-64
```

```
-mtune=generic -march=x86-64
```

```
-march=native
```

```
-march=ivybridge -mmmx -mno-3dnow -msse -msse2 -msse3 -mssse3 -mno-sse4a  
-mcx16 -msahf -mno-movbe -mno-aes -mno-sha -mpclmul -mpopcnt -mno-abm  
-mno-lwp -mno-fma -mno-fma4 -mno-xop -mno-bmi -mno-bmi2 -mno-tbm -mavx  
-mno-avx2 -msse4.2 -msse4.1 -mno-lzcnt -mno-rtm -mno-hle -mno-rdrnd -mf16c  
-mfsgsbase -mno-rdseed -mno-prfchw -mno-adx -mfxsr -mxsave -mxsaveopt  
-mno-avx512f -mno-avx512er -mno-avx512cd -mno-avx512pf -mno-prefetchwt1  
--param l1-cache-size=32 --param l1-cache-line-size=64 --param  
l2-cache-size=3072 -mtune=ivybridge
```


Dystrybucje skoncentrowane na udostępnianiu kodu źródłowego (binaria jako opcja)

- Gentoo Linux (prekompilowane pakiety binarne tylko dla wybranych programów)

Dystrybucje udostępniające zarówno źródła, jak i binaria (porty i pakiety)

- *BSD (ale aktualizacje i łaty bezpieczeństwa czasem tylko dla portów)

Dystrybucje skoncentrowane na udostępnianiu pakietów binarnych (źródła jako opcja)

- Większość dystrybucji Linuksa (Debian, RHEL, Fedora, CentOS, OpenSuse, Arch, Slackware i in.)
- TrueOS (FreeBSD repack)

Dystrybucje dla wielu architektur

- Debian GNU/Linux (oficjalne: amd64, arm64, armel, armhf, i386, mips, mips64el, mipsel, ppc64el, s390x; nieoficjalne: alpha, hppa, hurd-i386, kfreebsd-amd64, kfreebsd-i386, m68k, powerpc, powerpcspe, ppc64, sh4, sparc64, x32)
- Gentoo Linux (oficjalne: i386, x86-64, ia-64, pa-risc, powerpc, powerpc970, sparc64, alpha; nieoficjalne: mips, ps3, s390, arm, sh4)
- RHEL, Fedora
- Slackware
- *BSD (szczególnie NetBSD, ale też OpenBSD)

Dystrybucje tylko dla x86-64

- Arch Linux (nieoficjalny port dla arm)
- CentOS
- QubesOS (tylko x86-64 + UEFI)

Większość dystrybucji tylko dla architektury Intelu porzuciła wsparcie dla i386.

Zawartość pakietu

- Archiwum instalowanych plików
- Metadane

Formaty i systemy pakietów

- Deb (Debian i pochodne)
- RPM (*Red hat Package Manager*): RHEL, Fedora, CentOS, OpenSuse i pochodne
- Portage (Gentoo)
- Pacman (Arch)
- tgz/txz (Slackware)
- pkg (FreeBSD), pkgsrc (NetBSD i wiele innych), pkgng, DPorts (DragonflyBSD)

System pakietów na przykładzie Debiana

Pakiet źródłowy: archiwum `foo_X.Y.Z-N.tar.xz` zawierające:

- `foo_X.Y.Z-N.dsc` — plik z metadanymi
- `foo_X.Y.Z.orig.tar.bz2` — oryginalny *upstream release* `X.Y.Z`
- `foo_X.Y.Z-N.debian.tar.xz` — dodatki Debiana, *patchlevel* `N`

Uwagi:

- Z jednego pakietu źródłowego może powstać wiele pakietów binarnych.
- *Download:* `apt-get source foo`.
- Utworzenie pakietów binarnych: `dpkg-buildpackage`.

Pakiet binarny: archiwum `ar` o nazwie `foo_X.Y.Z-N_arch.deb` zawierające:

- `debian-binary` — plik tekstowy zawierający wiersz `2.0`
- `control.tar.gz` — archiwum zawierające metadane i skrypty (`gz` lub `xz`)
- `data.tar.gz` — archiwum plików do zainstalowania (`gz`, `bz2`, `lzma`, `xz`)

control.tar.gz

- `control` — główny plik z metadanymi
- `conffiles` — lista plików konfiguracyjnych
- `shlibs` — lista wymaganych bibliotek współdzielonych
- `md5sums` — lista skrótów MD5 instalowanych plików
- `postinst` — skrypt wykonywany przed instalacją
- `preinst` — skrypt wykonywany po instalacją
- `config` — skrypt do (re-)konfiguracji
- `prerm` — skrypt wykonywany przed usunięciem
- `postrm` — skrypt wykonywany po usunięciu
- `templates` — i18n komunikatów
 - Katalog `/var/lib/dpkg/info/`
 - Pliki z `control.tar.gz` pakietu `foo_X.Y.Z-N.deb` są kopiowane do `/var/lib/dpkg/info/foo.*` (bez `control`)
 - Plik `/var/lib/dpkg/status`

Pola wymagane:

- Package: *name*
- Version: *[epoch:]upstream-version[-debian-revision]*
- Maintainer: *Imię Nazwisko <email>*
- Description: short (60 znaków) + long

Pola opcjonalne:

- Section: *utils, net, mail, text, x11, ...* (zob. Debian Policy Manual, *debian-policy*)
- Priority: *required, important, standard, optional, extra, ...*
- Essential: *yes/no*
- Architecture: *arch, any, all*
- Source: nazwa pakietu źródłowego
- Depends:, Pre-Depends:, Recommends:, Suggests: *lista pakietów*
- Breaks:, Conflicts:, Replaces:, Provides: *lista pakietów*

Lista pakietów

Specyfikacja pakietu:

`nazwa-pakietu` $([< > >= <= =]$ *wersja*)

Lista pakietów:

- , — koniunkcja
- | — alternatywa, łączy *silniej* niż koniunkcja
- brak możliwości nawiasowania

Przykład:

```
libc6 (>= 2.14), libfreetype6 (>= 2.2.1),  
libgcc1 (>= 1:4.1.1), libgl1-mesa-glx | libgl1,  
libqtcore4 (= 4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1),  
libqtgui4 (= 4:4.8.6+git64-g5dc8b2b+dfsg-3+deb8u1),  
libstdc++6 (>= 4.1.1), libx11-6, libxrender1
```

Lista pakietów jako formuła logiczna

$$(p_1^1 \vee p_2^1 \vee \dots p_{k_1}^1) \wedge (p_1^2 \vee p_2^2 \vee \dots p_{k_2}^2) \wedge \dots (p_1^n \vee p_2^n \vee \dots p_{k_n}^n)$$

Lista pakietów jako formuła logiczna

$$(p_1^1 \vee p_2^1 \vee \dots p_{k_1}^1) \wedge (p_1^2 \vee p_2^2 \vee \dots p_{k_2}^2) \wedge \dots (p_1^n \vee p_2^n \vee \dots p_{k_n}^n)$$

Resolver: wybór pakietów do instalacji/usunięcia

- Lista pakietów jest formułą w koniunkcyjnej postaci normalnej.
- Należy znaleźć wartościowanie zbioru takich formuł, które:
 - spełnia jak najwięcej pakietów, które użytkownik chce zainstalować,
 - nie spełnia jak najmniej pakietów już zainstalowanych (tylko full-upgrade),
 - spełnia wszystkie formuły ze zbiorów *Depends* itd.,
 - nie spełnia żadnej z formuł ze zbioru *Conflicts* itd.
- Rozwiązanie problemu jest bardzo bliskie zwykłemu SAT-solverom.
- W Debianie apt i aptitude mają różne algorytmy.
- Polecenia **upgrade** (d. safe-upgrade) i **full-upgrade** (d. dist-upgrade) apt-a i aptitude.
- W RHEL/Centos/Fedora: zarządca pakietów rpm → yum → dnf (dandified yum).

Specjalne typy pakietów

- Pakiety wirtualne
(nie istnieją; ich nazwy występują w Provides: i Depends:/Recommends:/Suggests:)
- Metapakiety (puste pakiety z Depends:)
- Transitional (specjalne metapakiety)
- Tasks: tasksel [--test] [--list-tasks | install [*task* | standard]]
tasksel [--test] --new-install
- Uwaga: zwykle nie warto używać tasksel, także wywoływanego przez instalator:
`debconf-apt-progress -- apt-get -q -y -o APT::Install-Recommends=true -o APT::Get::AutomaticRemove=true -o APT::Acquire::Retries=3`
- debootstrap instaluje pakiety required i important. Resztę (standard, optional) lepiej instalować ręcznie.

Tworzenie własnych pakietów

- Kompilowanie pakietów źródłowych: apt-build
- Tworzenie własnych pakietów:

```
mkdir foo-1.0
cp -Rp foo-binary/* foo-1.0/
cd foo-1.0
dh_make --indep
find ../foo-binary > debian/install
debuild -i -us -uc -b
cd ..
dpkg -i foo_1.0-1_all.deb
```

dpkg — Debian packages

- narzędzia: `dpkg`, `dpkg-*`, `dselect`[†]
- zadania: instalowanie i odinstalowywanie pakietów, sprawdzanie zależności

APT – Advanced Package Management

- narzędzia: `apt-*` (`apt`), `aptitude`
- zadania: zarządzanie dostępem do repozytoriów, wyznaczanie planu instalacji/aktualizacji/deinstalacji zachowującego zależności. Możliwe różne *resolvery*.

Status pakietu

- **State:** not-installed, config-files, half-installed, unpacked, half-configured, triggers-awaited, triggers-pending, installed
- **Selection state:** install, hold, deinstall, purge
- **Flags:** reinst-required

Zarządzanie pakietami w dpkg/dselect

- `/var/lib/dpkg/available`
- `/var/lib/dpkg/triggers`

Narzędzia

- `dpkg`, `dpkg-reconfigure`
- `dpkg -S plik`, `dpkg -L pakiet`, `dpkg -l [pakiet]`

apt, apt-*, aptitude

- Zgodne narzędzia
- apt: command line, interactive
- apt-*: command line, traditional, scripts
- aptitude: command line + ncurses, rozbudowany *dependency resolver* i duże możliwości wyszukiwania pakietów
- Dodatkowe atrybuty pakietów: auto/manual, new
- /var/lib/apt, /var/lib/aptitude, /etc/apt
- /etc/apt/sources.list[.d]
- /etc/apt/apt.conf

Konfiguracja APT-a

- Katalog z konfiguracją: `/etc/apt/`
- Repozytoria: `sources.list(5)`, `sources.list.d/`
- Konfiguracja: `apt.conf(5)`, `apt.conf.d/`
- Sprawdzanie konfiguracji: `apt-config(8)`, np. `apt-config dump`

Przykładowy plik konfiguracyjny APT-a

```
APT {  
    Install-Recommends "false";  
    Install-Suggests "false";  
    AutoRemove {  
        RecommendsImportant "false";  
        SuggestsImportant "false";  
    };  
    Get {  
        Purge "true";  
        AutomaticRemove "true";  
    };  
};
```


- Pakiety: `url /pool/{main,contrib,non-free}` — katalogi z pakietami źródłowymi i binarnymi, np.
`pool/main/a/at/at_3.1.20{.orig.tar.gz,-3.dsc,-3.debian.tar.xz,-3_amd64.deb}`
Wiele różnych wersji pakietów. Wersje pakietów nie są związane z konkretną dystrybucją.
- Metadane: `url /dists/` — katalogi z dystrybucjami, np.
`trixie, trixie-backports, trixie-proposed-updates, trixie-updates`
Linki symboliczne: `stable, oldstable, testing, unstable`
Pliki: `ChangeLog*, InRelease, Release, Release.gpg` — podpisy cyfrowe.
Katalogi: `main, contrib, non-free`.
Pliki: `Release, Packages.*`
Opisy wielojęzyczne w: `Translation-*`

Konfiguracja APT-a a repozytoria

Wpis w `sources.list`:

```
deb http://ftp.pl.debian.org/debian/ trixie main contrib
```

przekłada się na adresy:

```
http://ftp.pl.debian.org/debian/dists/trixie/{main,contrib}/binary-amd64/Packages.xz
```

Pakiety znajdują się w katalogu `pool`, np. `nano_6.3-1_amd64.deb` jest pod adresem

```
http://ftp.pl.debian.org/debian/pool/main/n/nano/nano_6.3-1_amd64.deb
```

Ponadto:

- Korzystanie z wielu repozytoriów: *apt pinning*
- Klonowanie repozytoriów: `debmirror(1)` i in.
- Repozytoria `security`.

```
Types: deb deb-src
URIs: https://deb.debian.org/debian
Suites: trixie trixie-backports trixie-proposed-updates trixie-updates
Components: main contrib non-free non-free-firmware
Enabled: yes
Signed-By: /usr/share/keyrings/debian-archive-keyring.gpg
```

```
Types: deb deb-src
URIs: https://security.debian.org/debian-security
Suites: bookworm-security
Components: main contrib non-free non-free-firmware
Enabled: yes
Signed-By: /usr/share/keyrings/debian-archive-keyring.gpg
```

Zob. `sources.list(5)`, `deb822(5)`, RFC 822.

Aptitude search patterns

<code>?=var</code>	
<code>?not(pat)</code>	<code>!pat</code>
<code>?action(act)</code>	<code>~aact</code>
<code>?all-versions(pat)</code>	
<code>?and(pat₁, pat₂)</code>	<code>pat₁ pat₂</code>
<code>?any-ver(pat)</code>	
<code>?architecture(arch)</code>	<code>~rarch</code>
<code>?archive(arch)</code>	<code>~Aarch</code>
<code>?automatic</code>	<code>~M</code>
<code>?bind(var, pat)</code>	<code>?var:tn[(args)]</code>
<code>?broken</code>	<code>~b</code>
<code>?broken-dep</code>	<code>~Bdep</code>
<code>?broken-dep(pat)</code>	<code>~DB[dep:]pat</code>
<code>?broken-reverse-dep(pat)</code>	<code>~RBdep:pat</code>
<code>?conflicts(pat)</code>	<code>~Cpat</code>
<code>?config-files</code>	<code>~c</code>
<code>?dep(pat)</code>	<code>~D[dep:]pat</code>

<code>?description(desc)</code>	<code>~ddesc</code>
<code>?essential</code>	<code>~E</code>
<code>?exact-name(name)</code>	
<code>?false</code>	<code>~F</code>
<code>?for var: pat</code>	
<code>?garbage</code>	<code>~g</code>
<code>?installed</code>	<code>~i</code>
<code>?maintainer(maint)</code>	<code>~mmaint</code>
<code>?multiarch(multiarch)</code>	
<code>?narrow(filter, pat)</code>	<code>~S filterpat</code>
<code>?name(name)</code>	<code>~nname, name</code>
<code>?new</code>	<code>~N</code>
<code>?obsolete</code>	<code>~o</code>
<code>?or(pat₁, pat₂)</code>	<code>pat₁ pat₂</code>
<code>?origin(orig)</code>	<code>~Oorig</code>
<code>?provides(pat)</code>	<code>~Ppat</code>
<code>?priority(prio)</code>	<code>~pprio</code>

<code>?reverse-dep(pat)</code>	<code>~R[dep:]pat</code>
<code>?reverse-broken-dep(pat)</code>	<code>~RBdep:pat</code>
<code>?section(sec)</code>	<code>~ssec</code>
<code>?source-package(name)</code>	<code>~ename</code>
<code>?source-ver(ver)</code>	
<code>?tag(tag)</code>	<code>~Gtag</code>
<code>?term(kw)</code>	
<code>?term-prefix(kw)</code>	
<code>?true</code>	<code>~T</code>
<code>?task(task)</code>	<code>~ttask</code>
<code>?upgradable</code>	<code>~U</code>
<code>?user-tag</code>	
<code>?version(ver)</code>	<code>~Vver</code>
<code>?virtual</code>	<code>~v</code>
<code>?widen(pat)</code>	<code>~Wpat</code>

Przykłady

- `'~i !~M !~prequired !~pimportant'`
- `'~pstandard !~i'`
- `'~c'`
- `'~i ~Brecommends'`
- `'~RBrecommends:~i'`
- `'~ahold'`
- `'~aremove'`
- `'~S ~i ~Aoldstable !~Astable'`
- `'~m"Microsoft Corporation"'`
- `'!~Odebian !~v'`