Magazyn użytkowników Linuksa
dragonia magazine
numer 6 | 2006

Mandriva Power Pack 2007 Gold, Pro

Mandrakesoft jak i obecnie Mandriva kojarzone są z dystrybucjami bardzo prostymi i przyjaznymi użytkownikowi. Była to chyba pierwsza dystrybucja, która była w pełni spolonizowana. Z tych powodów często Mandriva wybierana jest przez początkujących. Jednocześnie system ten jest bardzo dopracowany i stabilny. Polityka firmy w tym zakresie jest konsekwentnie realizowana.

> str.3

Podpis cyfrowy, czemu warto używać?

Jesteśmy w dobie powszechnej informatyzacji społeczeństwa - to już truizm, powtarzany na każdym kroku. Komputery wdzierają się do wszelkich możliwych dziedzin życia. Podobnie jak w kontaktach międzyludzkich, także w Internecie istnieje potrzeba uwiarygodnienia swojej tożsamości, tak aby korespondent miał pewność, że komunikuje się z właściwą osobą.

> str.30

Wstępniak

Święta Bożego Narodzenia zbliżają się wielkimi krokami... Każda witryna sklepowa oświetlona jest różnymi światełkami, na choinkach wiszą bombki. Śniegu jeszcze nie ma, ale atmosferę świat juz dawno się odczuwa. Święta to czas magiczny! Święta to czas... spędzony z rodziną i przy komputerze :-) .

W tym numerze przeczytacie m.in. o dystrybucji linuksa KateOs III , Podpisie elektronicznym, Efektywnym zarządzaniu informacją w firmie.

Życzę miłej lektury i czekamy na Wasze opinie i sugestie dotyczące Dragonia Magazine.

Redaktor Naczelny

Piotr Krakowiak

Wszystkie materiały są objęte prawem autorskim (na zasadach licencji Creative Commons). Nie ponosimy odpowiedzialności za treść ogłoszeń. Nazwy firm, nazwy handlowe i znaki towarowe, jeśli zostały zostały użyte w publikacji, to jedynie w celach informacyjnych i są własnością poszczególnych podmiotów.

SPIS TREŚCI

SYSTEM
3 - Mandriva Power Pack 2007 Gold, Pro
5 - KateOs III
7 - Efektywne zarządzanie informacją w firmie cz.1
PROGRAMOWANIE
10 - Obiektowość
15 - C++ cz.5
20 - Wprowadzenie do Tcl\Tk cz.3
SOFTWARE
23 - Wykorzystanie programu gnuplot do do tworzenia wykresów 2D
27 - Apache cz.4
30 - Podpis cyfrowy, czemu warto używać?
32 - Thawte - jak dostać darmowy certyfikat niekwalifikowany
34 - Kilka upiększeń dla KDE
38 - VPN i PPTP - jak to zrobić?
HARDWARE
39 - Notebooki - przyszłość komputera?
KONKURS
19 - Konkurs!
ROZRYWKA
41 - Little Sheep czyli zabawa z owieczkami

Redaktor Naczelny: Piotr Krakowiak (dragon)

Redaktorzy: Tomasz Czunko (tomcash), Rafał Trójniak (ert16), Maciek Malinowski (borimir), Piotr Szewczuk, Rafał Domeracki (RD), Marcin Lipiec (lipi3c), Michał Rzepka, Tomasz Łuczak

Współpracujący: Tomasz Bednarski, Maciek Rutecki

www: http://www.dragonia.pl

forum: http://www.forum-dragonia.bee.pl

e-mail: dragonia.magazine@gmail.com

Mandriva Power Pack 2007 GOLD, PRO

Mandrakesoft jak i obecnie Mandriva kojarzone są z dystrybucjami bardzo prostymi i przyjaznymi użytkownikowi. Była to chyba pierwsza dystrybucja, która była w pełni spolonizowana. Z tych powodów często Mandriva wybierana jest przez początkujących. Jednocześnie system ten jest bardzo dopracowany i stabilny. Polityka firmy w tym zakresie jest konsekwentnie realizowana.

MACIEK 'borimir' MALINOWSKI

Mandriva nie jest bardzo starą firmą, ale na pewno można śmiało powiedzieć, że jest jedną z firm o najbardziej bujnej historii. Jej początki można datować na 1998 rok. Wówczas założono we Francji firmę Mandrakesoft. Firma około 2003 roku przeżywała znaczne problemy finansowe i przez prawie rok trwało postępowanie naprawcze, ale reorganizacja działania i zmiana stylu zarządzania pozwoliła wyjść z problemów. Zyski stawały się coraz większe i w 2004 roku Mandrakesoft połączył

LinDVD

się z brazylijską firmą Conectiva. Doszło do zmiany nazwy na Mandriva, a w 2005 roku Mandriva wykupiła firmę Lycoris.

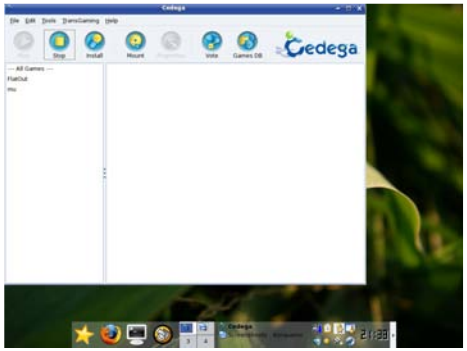
Mandriva to dystrybucja, która przygotowywana jest w kilku wersjach w zależności od wymagań użytkownika, przeznaczenia systemu jak i zasobności portfela, bowiem pewne wersje udostępniające dodatkowe elementy są płatne. Każdy bez większego problemu może znaleźć wersję najlepszą dla siebie i swoich wymagań oraz oczekiwań.

Testowałem wersję Mandriva 2007 Powerpack Gold. Dystrybucję tę zainstalowałem na moim laptopie Fujitsu-Siemens v2030. Nie miałem z procesem instalacji żadnych problemów. Instalacja przebiega bardzo sprawnie i szybko, sprzęt wykrywany jest w jej trakcie wzorowo. Na obecną chwilę jest to jedyna dystrybucja, która poprawnie wykryła układ graficzny w jaki wyposażony jest mój komputer. Jest to chipset VIA Unichrome Pro, który przysparzał mi wiele problemów do tej pory. Wreszcie mogę oglądać filmy na pełnym ekranie, poprawiła się również jakość wyświetlanego obrazu. Wykrywanie sprzętu jest moim zdaniem jednym z największych plusów Mandrivy. Z doświadczenia zauważyłem, że jest o wiele lepsze niż w SUSE, a nawet SLED.

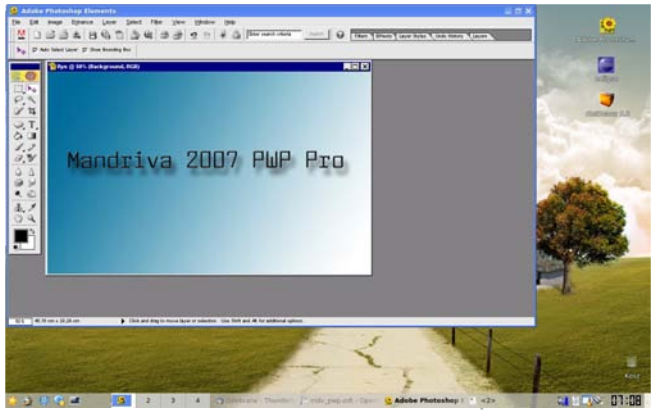
Kolejnym plusem jest jej wielka multimedialność. System dostarczany jest wraz z kodekami umożliwiającymi odtwarzanie muzyki i filmów w zamkniętych standardach, dodatkowo wersja płatna udostępniania jest z programem LinDVD, który świetnie nadaję się do odtwarzania filmów z płyt DVD. Jest to program dzięki, któremu nie jesteśmy narażeni na jakiekolwiek nieprzyjemności związane z patentami. W Polsce nie ma to obecnie większego znaczenia, ale w skali światowej jest to niewątpliwie duży plus tego systemu.

Wersja Gold dostarczana jest dodatkowo wraz z programem Cedega, który z kolei umożliwia instalację gier i innych programów przeznaczonych na platformę Windows. Dzięki niemu bez problemu zainstalujemy najnowsze gry bez potrzeby posiadania systemu z pod znaku Okien. Dla osób nie mających legalnego systemu będzie to na pewno wybawienie przed wizytą odpowiednich władz. Skorzystają również na tym osoby, które często przełączają się między dwoma systemami. Mandriva z Cedegą chroni nas od tej konieczności dzięki czemu oszczędzamy nasz czas. Oczywiście nie mamy pewności, że każda gra będzie działać i że nie pojawią się związane z jej działaniem żadne problemy, ale postęp programów typu Cedega jest bardzo dynamiczny i co jakiś czas pojawiają się uaktualnienia. Dodatkowo w internecie można znaleźć bardzo wiele informacji od użytkowników w jaki sposób dany problem obejść.

Wersja jeszcze bardziej rozbudowana - Mandriva Powerpack Pro dostarczana



Cedega



Photoshop uruchomiony za pomocą CrossOver Office

jest wraz z pełną wersją programu Cross Over Office, który umożliwia podobnie jak Cedega uruchamianie programów stworzonych dla systemu Windows, ale w odróżnieniu od Cedegi zamiast do uruchamiania gier przeznaczony jest do uruchamiania i pracy programów biurowych zaczynając od Ms Office, kończąc na Photoshopie czy CorelDraw.

Dodatkowo jak w większości dystrybucji dodano obsługę biurka 3D. Mandriva zawiera zarówno AIGLX i Xgl. Dzięki temu można powiedzieć, że rozwiązuje problemy związane z przyszłością owych rozwiązań. Oba bowiem rozwijane się równolegle przez różnych producentów, nie są ze sobą kompatybilne i obecnie ciężko cokolwiek powiedzieć, które z nich prowadzi i najprawdopodobniej stanie się w przyszłości standardem. Po pierwsze z tego powodu, że zawierają tyle samo wad co zalet, a to z powodu ich stosunkowo wczesnego etapu rozwoju.

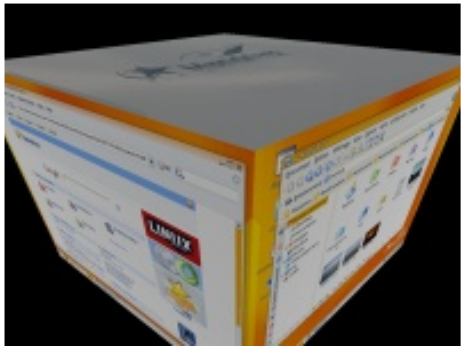
Na uwagę zasługuje również to, że system ten dostarczany jest wraz z oprogramowaniem antywirusowym znanej i bardzo solidnej firmy Kaspersky. Jak zapewniają producenci systemu - Linux jest wolny od wirusów, a program ten ma służyć jako sposób na ochronę przechodzącego przez system ruchu sieciowego, skanowania poczty dla Windows itp. Jednak możemy oczekiwać w przyszłości, że coraz większa popularność Linuksa niestety doprowadzi do pojawienia się większej liczby wirusów przeznaczonych dla tego systemu. Dlatego już wkrótce antywirus będzie niestety nie tylko dodatkiem, ale i stałym elementem systemu.

Jeśli chodzi o ochronę systemu to Mandriva dostarcza kolejne bardzo ciekawe i zarazem przydatne narzędzie mianowicie BitDefender, którego zadaniem jest chronienie naszego systemu przed trojanami, programami typu spyware i wirusami. Na pewno zdublowanie oprogramowania chroniącego system przyczyni się do jego większego bezpieczeństwa. Dodatkowo programy dostarczane wraz z Mandrivą bardzo łatwo aktualizują swoje bazy danych.

W Mandrivie dostajemy ponadto:
jądro Linux w wersji 2.6.17,
KDE 3.5.4 i GNOME 2.16,
OpenOffice 2.0.3,
Skype,
pluginy RealPlayer 10 i Flash 7, Java,
i wiele wiele innych użytecznych programów;

Wraz z systemem otrzymujemy dwie publikacje elektroniczne w formacie pdf. Pierwsza z nich jest 300 stronicowym podręcznikiem o użytkowaniu Mandrivy, a druga dotyczy posługiwania się wierszem poleceń. Obie zostały przetłumaczone na język polski.

W stosunku do wersji poprzednich zauważalna jest znaczna poprawa w przypadku prędkości uruchamiania systemu. Jednocześnie zmieniono program służący do zarządzania pakietami i instalator sys-



Biurko 3D

temu. Te dwa ostanie elementy wzbudzają największe kontrowersje wśród użytkowników. Moim zdaniem jest to kwestia przyzwyczajenia.

Można pokusić się o stwierdzenie, że Mandriva ostro walczy o użytkowników domowych i biurowych z Windows. Na obecnym etapie rozwoju (włączając Cedegę i Cross Over Office) w większości przypadków może go śmiało zastąpić i co jest najważniejsze dzięki tym programom nie musimy iść już na żadne kompromisy. Udostępnienie specyfikacji Płatnika moim zdaniem jeszcze bardziej przyspieszy zadomawianie się Linuksa w biurach, bowiem w bardzo wielu instytucjach to właśnie tylko owy Płatnik jest argumentem za pozostaniem przy Windows.

Można o niej powiedzieć, że jest systemem, który w znaczny sposób przyczyni się do rozprzestrzeniania się Linuksów na biurkach. Dostarczanie obu rozwiązań AIGLX i Xgl oraz programu antywirusowego jest świadectwem dalekowzroczności firmy. Polecam Mandrivę zarówno jako centrum multimedialne do domu, serwer w biurze jak i zamiennik Windowsa. ■

Przydatne adresy

Oficjalna strona Mandrivy
http://www.mandriva.pl

Polski klub użytkowników Mandrivy
http://strefa.mandriva.pl

Sklep Mandrivy
http://store.mandriva.pl

Nieoficjalna strona Mandrivy
http://www.mandriva.org.pl

KateOS III

Pewnego razu, kiedy marudziłem na temat braku dostępu do partycji NTFS, znajomy wspomniał coś o systemie, który rzekomo potrafi zapisywać na takich partycjach od razu po zainstalowaniu. Brzmiało ciekawie; pilnie potrzebowałem takiej funkcjonalności więc wszedłem na stronę www.kateos.org i już następnego dnia miałem trzy płytki.

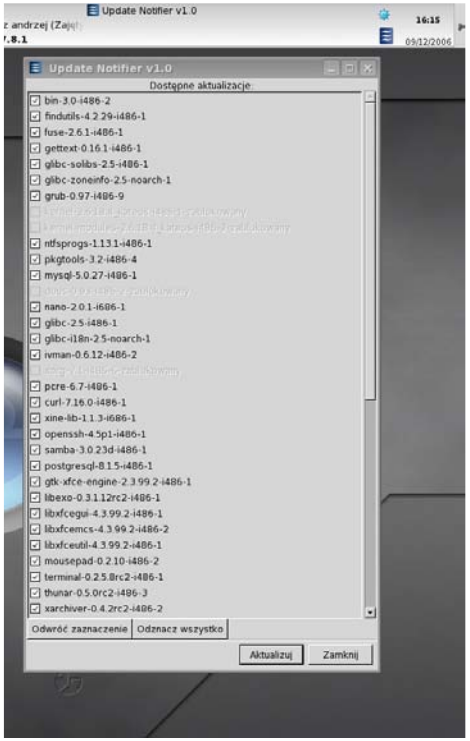
MICHAŁ RZEPKA

Była to KateOS 3.0, kolejna odsłona systemu kiedyś znanego jako Kate Linux. Tym razem nie bazowała już na Slackware, (jak dawniej) – została zbudowana od zera.

Kate 3.0 (i 3.1), w zupełnym skrócie, to dość pokaźny zestaw nowych wersji przeróżnych programów, którymi lokalnie zarządza unikalny i spotykany tylko w Kasi menadżer PKG. Operuje on na plikach TGZex – formacie którego raczej nigdzie więcej nie spotkacie. Do zdalnej aktualizacji zaś służy Updateos. Ale o nich potem.

Kiedy napisałem 'przeróżnych programów', to nie przesadzałem, Kate bowiem wykazuje bardzo dobre zbilansowanie pomiędzy typowymi zastosowaniami serwerowymi i programistycznymi, a multimediami i prostotą użycia. Już domyślna, pełna instalacja udostępnia masę programów tak różnorodną, że jeśli nie jesteś wybredny, to raczej z updateos będziesz musiał korzystać wyłącznie w celach aktualizacji. Znajdują się tu – między innymi – jądro 2.6.17.13, GIMP 2.2.13, apache2 2.2.3, PHP 5.1.6, Xorg 7.1, pakiet Open Office 2.0.3, czy kadu. Jak pokazują wersje programów, Kate stara się uchodzić za system nowoczesny.

Wszystkie pakiety kompilowane są pod i686, więc Kate odznacza się dość szybkim działaniem, nawet na słabszym sprzęcie. Właśnie ta lekkość systemu i szybkość jest kolejnym z priorytetów twórców Kasi. Dlatego standardowym środowiskiem graficznym, dostępnym już po instalacji z płyt 1 i 2, jest XFCE 4.4. Na trzeciej i czwartej płycie znajdują się kolejno KDE (w wersji 3.5.4) i Gnome. Jeśli komuś nie chce się ściągać kilku giga, może zaopatrzyć się w Miniiso, ważące około 80 MB. Ta miniatura systemu zawiera wszelkie narzędzia potrzebne do zbudowania 'pełnoprawnego' systemu, a konkretniej ściągnięcia jego elementów z



Update-notifier w akcji – milej się już chyba nie da

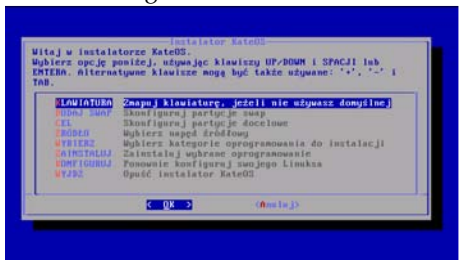
sieci.

Innym priorytetem developerów Kasi jest prostota użycia. System wymaga znacznie mniej 'zabaw' przy użytkowaniu niż na przykład Slackware, co nie znaczy jednak że prowadzi użytkownika za rączkę. Zależności są pilnowane, ale można (jeśli się koniecznie chce) z nich zrezygnować. Kate bowiem jest z założenia przeznaczona dla użytkownika który już coś o Linuksach wie. Dowodem takiego myślenia może być już sam instalator. Jest to bowiem surowo wyglądający, zmodyfikowany instalator znany ze Slackware - na szczęście dla wielu, jest on całkowicie po Polsku, a na stronie dokumentacji projektu znajduje się szczegółowy przewodnik instalacji we Flashu. Podobieństwo do Slaka pojawia się jeszcze w jednym miejscu – instalacji pakietów.

PKG potrafi bowiem instalować TGZ, 'installpkg' zaś potrafi zainstalować pakiet 'Kasiowy'. Nie jest to jednak praktyka zalecana przez twórców systemu, chociażby dlatego (ale nie tylko) że installpkg nie poinformuje użytkownika, że brakuje jakichś zależności. Aby zainstalować pakiet TGZex (zakładając, że mamy go na dysku) wystarczy napisać

```
pkg -i nazwa_pakietu
```

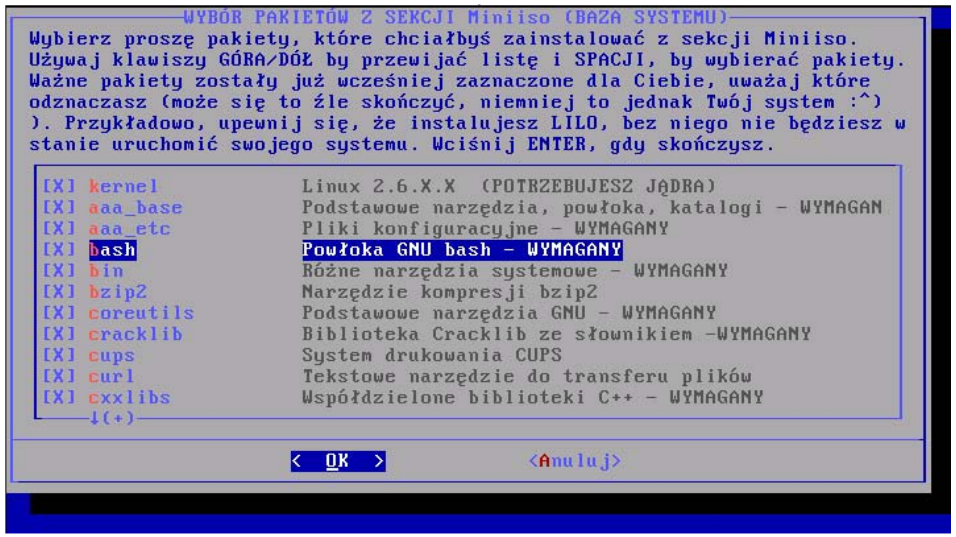
Jeśli z kolei chcemy usuwać, zamieniamy w powyższej komendzie i na r. Jak do tego dorzucimy '--with-deps', to usuniemy również pakiety zależne.
Podniesienie pakietu do wyższej wersji umożliwia argument '-u'.



Instalator KateOS – małe deja-vu?

Jeśli nie mamy pakietu na dysku, lub chcemy ściągnąć jakiś program z Internetu, z pomocą przychodzi updateos. Po instalacji systemu warto wydać polecenie:

```
updateos -usd
```



Instalator KateOS – małe deja-vu?

Co zaktualizuje listę pakietów. Teraz można już szukać, pisząc na przykład:

```
updateos -f kdegraphics
```

Jeśli zaś wiemy o co nam dokładnie chodzi, możemy napisać:

```
updateos -i apache2
```

'I' jak 'instaluj', 'f' jak 'find'.
Updateos może zaktualizować nam tylko konkretną grupę programów, za pomocą:

```
updateos -ug nazwa_grupy
```

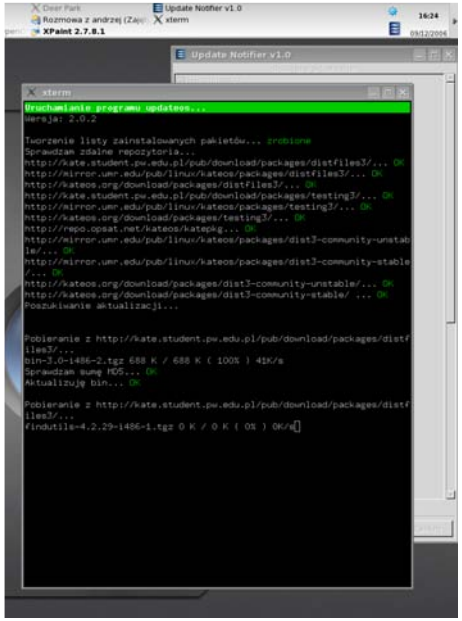
Może i od razu cały system, wystarczy go o to poprosić komendą:

```
updateos -ud
```

Tu może dać o sobie znać jedna z nieprzyjemności Kate. Otóż, może ona... nie wstać po aktualizacji, póki nie wydamy komendy 'lilo'. Można tego dokonać po aktualizacji a przed restartem systemu, można również potem, bootując Kasię za pomocą płyty instalacyjnej.
Oba wyżej opisane narzędzia działają w konsoli, ale jeśli ktoś chce, może zainstalować niedawno powstałego 'Update-notifiera'. Po jego odpaleniu, zobaczymy w tray-u logo Kate, które migając da znać, że wykryto dostępne aktualizacje. Wystarczy nań kliknąć, wybrać co chcemy aktualizować i zatwierdzić.

Kasia to produkt rodzimy. Jednak na chwilę obecną jego rozwojem - od strony paczkowania, dokumentacji i wsparcia - zajmują się ludzie z wielu krajów (samo forum KateOS podzielone jest już na część Polsko- i Angielskojęzyczną). Oficjalna dokumentacja zawiera how-to o budowaniu paczek TGZex. Jeśli zgłosisz chęć pomocy, i wykażesz się umiejętnościami, to dostaniesz autoryzację i prawo do podzielenia się ze światem własnoręcznie zrobionymi paczkami na PKG-Portal dystrybucji. Jeśli czegoś nie możesz znaleźć za pomocą updateos, warto zerknąć na PKGPortal. Jeśli nie możesz sobie poradzić na przykład z instalacją modemu od Neostrady, czy karty WiFi, to odpowiedź znajdziesz w centrum dokumentacji.

Nie da się ukryć, że KateOS rozwija się bardzo prężnie, i już od jakiegoś czasu jest wysoko oceniane, chociażby na tuxmachines.org, czy 'u nas' – na linux.pl. Mnie to osobiście nie dziwi, i nie zamierzam instalować póki co żadnego innego systemu...



Nieco trudniejszy updateos, wykonujący zlecenia nieco prostszego update-notifiera.

Przydatne adresy

```
http://kateos.org/ - strona domowa projektu
http://wikidoc.kateos.org/index.php/Main_Page - centrum dokumentacji
http://kateos.org/forum/ - forum użytkowników KateOS
http://kateos.org/?sekcja=kateIIIss - tak wygląda Kasia 3
```

Efektywne zarządzanie informacją w firmie cz.1

Co jest najważniejsze w procesie zarządzania informacją w firmie?

TOMASZ BEDNARSKI

Takie pytanie zadaje sobie większość osób odpowiedzialnych za przygotowanie i wdrożenie zmian w systemie zarządzania firmą. Odpowiedzi są różne w zależności kto nam jej udzieli. Wygoda – powiedzą pracownicy; ekonomiczność – odpowie dyrektor finansowy; bezpieczeństwo – uczuli administrator systemu lub dyrektor IT; efektywność – dyrektor zarządzający. I każdy z nich ma rację. Pogodzenie wszystkich potrzeb jest bardzo trudne i wymaga właściwego podejścia na etapie budowania koncepcji systemu informacyjnego przedsiębiorstwa. System informacyjny w nowoczesnej firmie pełni rolę podobną do tej, którą spełnia układ nerwowy w organizmie człowieka. Przekazuje sygnały między poszczególnymi organami i decyduje o ich działaniu. Podobnie jak w przypadku organizmu jeżeli jeden z organów zaczyna szwankować – zaczynają się kłopoty dotykające całości. Ważne jest zatem dostosowanie konstrukcji i sposobu działania systemu informacyjnego do specyfiki przedsiębiorstwa i jego celów nadrzędnych. Jest to kwestia oczywista. Nieoczywista jest jednak metoda osiągnięcia tego celu.

Odpowiedzi na pytanie co jest najważniejsze w procesie zarządzania firmą należy szukać stawiając kolejne pytanie: co jest najważniejsze w procesie rozwoju i przetrwania firmy w długim okresie? I tutaj możemy natrafić na wiele różnych odpowiedzi, ale jeżeli firma działa na rynku konkurencyjnym najważniejszy dla jej przyszłości jest klient. Klient zadowolony z obsługi, produktu, ceny i kontaktów z firmą w okresie użytkowania produktu/konsumowania usługi. Każdy szef firmy wie jakim trudnym zadaniem jest osiągnięcie tego celu. Warunki otoczenia, konkurencja, dostawcy, warunki w samej firmie i wreszcie potrzeby klienta stanowią bardzo zmienne i trudne środowisko. Sztuką jest takie wykorzystanie sprzyjających i zniwelowanie negatywnych czynników, aby cele firmy zostały osiągnięte. Sztuką, która potrzebuje pożywki w postaci dużej ilości informacji o odpowiedniej jakości oraz narzędzi wspomagających podejmowanie właściwych decyzji.

Proces podejmowania decyzji ma różny charakter. Jest on uzależniony od wielkości firmy, jej rozmieszczenia geograficznego, struktury organizacyjnej, sposobu podejmowania decyzji, ilości miejsc, w których podejmowane są decyzje o różnym zakresie, wadze i wpływie na pozostałe obszary funkcjonowania firmy. Można wyodrębnić różne modele funkcjonowania, dla których powstały tysiące koncepcji, modeli i rozwiązań w zakresie organizacji i zarządzania systemem informacyjnym. Mimo tego ogromu rzadko kiedy rozwiązanie stworzone dla jednej firmy sprawdzi się w innej. Dlaczego? Ponieważ każda firma inaczej radzi sobie z opanowaniem sztuki dbania o klienta. I jest to zjawisko naturalne ponieważ warunki funkcjonowania są inne dla każdej z nich. Co z tego wynika w świetle naszych rozważań nad budową systemu informacyjnego? Wynika z tego potrzeba bardzo indywidualnego podejścia do budowania koncepcji systemu zarządzania informacją już na poziomie planowania strategicznego. Stawiając zadowolenie klienta na szczycie hierarchii celów firmy (zysk i przetrwanie uznajemy za cele zależne od zadowolenia klienta) zmieniamy spojrzenie na wszystko co dzieje się w firmie i jej otoczeniu.

Analiza strategiczna odpowiada na pytanie, co jest ważne dla rozwoju firmy w długim okresie – wskazuje konkretne czynniki jakie mają decydujący wpływ na ofertę (koszty i w efekcie ceny produktów, jakość surowców i jej wpływ na jakość produktów) oraz podpowiada ich pożądane wartości. Przykładowo cena mebli z litego drewna zależy w prostej linii od cen drewna na rynku. Producent mebli powinien przeanalizować jaki wpływ na jego sytuację będą miały ruchy cen ponieważ ich zmiany określają ostateczny kształt oferty i tym samym decydują w pewnej mierze o zadowoleniu klienta. Znając kluczowe czynniki decydujące o sukcesie lub porażce przedsięwzięcia można przygotować rozwiązania, które umożliwią ich optymalne wykorzystanie. Obejmuje to również działania związane z systemem informacyjnym. Rozpoczynając projektowanie systemu informacyjnego od analizy potrzeb klientów (jako wartości nadrzędnej) oraz czynników wpływających na poziom ich zaspokojenia, możemy osiągnąć efekt w postaci narzędzia efektywnie

wspomagającego nasze dążenia. Jakie narzędzie możemy określić mianem efektywnego? Trudno to określić w złotówkach. Istnieją co prawda metody systematyzujące proces oceny efektywności systemu informacyjnego, jednak osiągane wyniki należy uznać za co najmniej nieprecyzyjne. Przyjmijmy w uproszczeniu, że system efektywny to taki, który realizuje postawione przed nim zadania, łatwo dostosowuje się zmieniających się wymagań, nie przeszkadza w prowadzeniu biznesu i nie absorbuje zbyt wiele środków finansowych. Taka definicja efektywności odpowiada mniej więcej odczuciu osoby odpowiedzialnej za zarządzanie firmą. Producenci rozwiązań informatycznych coraz lepiej to rozumieją i tworzą rozwiązania ukierunkowane na realizację przedstawionej koncepcji projektowania systemów informacyjnych. Oczywiście w praktyce okaże się, że ocena efektywności danego rozwiązania będzie bardzo różna dla różnych środowisk, w którym przyjdzie im działać. Dlatego niezbędne jest zrozumienie koncepcji jaka kryje się za procesem projektowania systemu informacyjnego w danej firmie – koncepcji dbania o zadowolenie klienta – i przygotowanie rozwiązania współtworzącego nową wartość.

Podsumowując. Sprawny system informacyjny jest niezbędny do skutecznego zaspokajania potrzeb klientów. Sprawny system informacyjny w nowoczesnej firmie nie może funkcjonować bez sprawnego systemu informatycznego. Taki system powinien mieć zdolność efektywnego wspierania wszystkich działań firmy. Na efektywność składa się:

- możliwości realizacji określonych zadań
- możliwość szybkiego dostosowania do indywidualnych potrzeb firmy
- możliwość zapewnienia bezpieczeństwa informacji
- czas absorbowany przez utrzymanie i dostosowywanie systemu
- koszt utrzymania i stosowania w długim okresie

Otwarte oprogramowanie doskonale wpisuje się w założeniom określającym miarę efektywności systemu. Jak objawia się to w praktyce? Nad tym zastanowimy się w kolejnym odcinku.

Mandriva Poland
Amazis.net Sp. z o.o.
ul. Bukowska 352
60-189 Poznań

www.mandriva.pl
www.store.mandriva.pl
Telefon: 061/ 868 27 01
e-mail: biuro@mandriva.pl

Mandriva Corporate Server 4

Corporate Server 4.0 to idealne rozwiązanie, które wspomaga system komunikacji Twojej firmy: wydajność oraz wysoka niezawodność w krytycznych punktach infrastruktury, zgodność z powszechnymi standardami, interoperacyjność, wysoki poziom bezpieczeństwa oraz długa i niezawodna praca.

Najlepsze technologie wirtualizacji

Technologia wirtualizacji staje się podstawą nowoczesnego przedsiębiorstwa. Pozwala na bardziej skuteczne użycie programowej technologii oraz pomaga zredukować koszty poprzez lepsze wykorzystanie zasobów sprzętowych.

Sposród dużej oferty oraz możliwości oferowanych komponentów, Mandriva wybrała wiodące technologie wirtualizacji:

Xen: pozwala jednocześnie na uruchomienie wielu instancji Linuksa oraz czerpie dodatkowe korzyści ze sprzętowej akceleracji dla wirtualizacji poprzez Intel® VTTM.

OpenVZ: rozwiązanie zoptymalizowane dla dużych liczb wirtualnych serwerów poprzez dużą liczbę instancji Linuksa, bez dodatkowej sprzętowej wirtualizacji.

VmWare: pozwala na stworzenie kompletnych wirtualnych maszyn.

Mandriva Corporate Server 4 wspiera najnowsze technologie:

- Dual core CPU
- SAS bus (Serial Attached SCSI)
- Procesory Intel® Xeon® dual core 5100 (Woodcrest) i 5063 (Dempsey)
- Uruchamianie poprzez SAN (Maszyny HP®)
- SAN (Storage Area Network) (NEC)
- Intel® VT
- Serwery Blade
- Nowe drukarki sieciowe Ricoh® oraz Xerox®

Usługi

Usługi sieciowe	Drukowanie i pliki	Komunikacja
• Serwer DNS • Serwer DHCP • Serwer PXE ...	• Serwer Samba • Serwer Cups • Serwer FTP • Serwer NFS ...	• Serwer Poczty • Serwer list dyskus. • Anti-spam • Anti-virus

Infrastruktura

LAMP	Uwierzytelnianie	Serwer apl. J2EE	Bazy danych
Serwer Apache, PHP, Perl, Python, Ruby, MySQL	• OpenLDAP • Kerberos	• Jonas • Tomcat • Websphere CE	• MySQL • Postgresql • DB2

Architektura	x86, x86-64	
System	Jądro 2.6.12 Szyfrowany zdalny dostęp: OpenSSH System kontroli dostępu: rsbac (1.2)	Wirtualizacja: Xen (3.0), OpenVZ System duplikacji: Ka, Dolly, Dolly+
Infrastruktura	Serwer katalogowy: OpenLDAP (2.3) Uwierzytelnianie sieciowe: Kerberos 5 Bazy danych: MySQL (5.0), Postgre (8.1)	Serwer WWW: Apache (2.2) Serwer Proxy: SQUID
Serwer	Udostępnianie plików: NFS (3 i 4) Serwer plików: SAMBA (3.0.22) Serwer druku: CUPS (1.2) Serwery SMTP: Postfix (2.2), Sendmail Serwery POP/IMAP: Cyrus-IMAP (2.2) Courier-IMAP (3.0)	Serwer instalacyjny: PXE Serwer DHCP: ISC dhcp-server Serwer FTP: Proftp, Pure-ftp Serwer DNS: BIND

Produkt Corporate Server 4.0 jest certyfikowany przez wiodących producentów oprogramowania włączając w to takie firmy jak **Oracle**, **IBM**, **Sun**, **Swsoft** i wiele innych.

Obiektowość

Na początku programowano liniowo. Przykładem języka wykorzystującego tę technikę jest BASIC. Program pisano w ten sposób, że pierwsze linijki programu znajdowały się na początku pliku, a ostatnie na jego końcu (w sumie jest to najbardziej intuicyjny sposób kodowania). W takiej samej kolejności program się wykonywał. Między poszczególnymi miejscami w programie „skakano" przy pomocy instrukcji goto. Doprowadzało to do powstania bardzo trudnego w pielęgnacji i czytaniu kodu zwanego często kodem spaghetti.

MACIEJ 'borimir' MALINOWSKI

Kolejnym krokiem było wprowadzenie procedur (funkcji). Dzięki temu można było podzielić program na logiczne części i wielokrotnie wykorzystywać tak napisany kod, po raz pierwszy wprowadzono do programowania zasadę ponownego wykorzystania kodu, dzięki temu programy stawały się krótsze. Łatwiejsze stało sie również zarządzanie nimi, praktycznie wcześniejsze liniowe programy pisane były przez jednego programistę. Od momentu zadomowienia się funkcji odpowiedzialny mógł być za to już zespół, który dzielił między siebie pracę.

Postęp następnie dotknął sposobu organizacji danych. Pojawiły sie bowiem struktury (w Pascalu zwane rekordami). Były to typy danych, które w odróżnieniu od tablic mogły składać się z wielu różnych typów danych. Przykładem może być na przykład wpis w książce telefonicznej, który oprócz nazwiska i imienia obejmuje również numer telefonu. Od tego momentu można było w bardziej naturalny sposób oddawać pewne modele ze świata rzeczywistego.

Kolejnym krokiem było wprowadzenie obiektów, o których będzie traktował ten artykuł. Technikę wykorzystującą obiekty i ich właściwości takie jak polimorfizm i dziedziczenie nazywamy Programowaniem Orientowanym Obiektowo.

Najprościej można wyjaśnić czym w zasadzie jest obiekt mówiąc o nim jako o strukturze, w której oprócz danych składowych zawarte są również funkcje składowe, które służą do operowania na obiekcie. Dzięki temu możemy w jeszcze bardziej intuicyjny niż w przypadku struktur sposób oddać pewne elementy środowiska. Taka organizacja programu zwiększa również jego przejrzystość, o wiele łatwiej się nim zarządza oraz co najważniejsze o wiele łatwiej program taki można rozbudowywać i poszerzać.

Obiekty wykorzystywane są w bardzo wielu językach programowania między innymi w Delphi, Javie, czy C++. W każdym z tych języków problem obiektowości rozwiązany jest w nieco inny sposób, ale teoria działania obiektów pozostaje praktycznie niezmieniona. Ucząc się programowania obiektowego w jednym z powyższych języków po poznaniu niewielkich różnic można przesiąść się na inny.

Zacznijmy więc od ogólnego poznania obiektów. Z tą techniką, zresztą jak z każdą inną wiąże się pewien zbiór nazw odnoszących się do poszczególnych pojęć. Najważniejszym z nich jest obiekt, nazywany jest on również egzemplarzem klasy. Klasa jest typem danych. Definiującym zarówno obiekty, z których się składa jak i również pewne funkcje. Klasę można porównać do typu samochodu Ford Escort, a obiekt do jego określonego fizycznie istniejącego egzemplarza. Klasa można powiedzieć nie istnieje fizycznie w programie, ale jedynie obiekty, które są jej egzemplarzami. Podobnie do wspomnianego powyżej Forda Escorta. Nie można powiedzieć, że jako typ samochodu jest on obecny fizycznie i namacalnie, ale już jego konkretny egzemplarz jak najbardziej.

Jak już wspomniałem klasa opisuje elementy składowe. Mogą nimi być obiekty. Mówiąc o obiektach mam na myśli zarówno typy wbudowane w dany język programowania, typy przez nas zdefiniowane jak i również inne obiekty. Niedozwolone jest jedynie zamieszczanie obiektu tej samej klasy. Jak na razie możemy myśleć o obiektach po prostu jak o strukturach. Odróżnia je jednak występowanie tak zwanych funkcji składowych (w innych językach np. PHP nazywane są mało szczęśliwie metodami).
Funkcje składowe to pewne „zachowania", które często nazywamy interfejsem danej klasy, czyli sposobami w jaki możemy na nią wpływać. Poniżej zamieszczam pewien przykład. Mamy obiekt klasy włącznik. Jest to jakiś włącznik, który nazwijmy np. mój_włącznik (kolejną ważną cechą obiektu jest to, że obiekty rozróżniamy przy pomocy pewnej unikatowej dla nich nazwy). Możemy ustawić go w stanie włączony, lub wyłączony. Jest to pewna akcja jaką można na nim wykonać, a więc jest to element interfejsu. Przykładowo klasę włączników możemy zdefiniować następująco:

```
class włącznik {
	bool stan;

	void włącz()
	{
		stan = true;
		dalsze ciało funk-
cji odpowiedzialnej za włącza-
nie
	}

	void wyłącz()
	{
		stan = false;
		dalsze ciało funk-
cji odpowiedzialnej za wyłącza-
nie
	}
};
```

Powyższa definicja klasy jest napisana w pseudokodzie. Widzimy w niej definicję obiektu zachowującego obecny stan włącznika (obiekt stan), oraz dwie funkcje, które umożliwiają operowanie na nim. Jest to definicja pewnej klasy włączników, która w rzeczywistości nie istnieje w programie. Zdefiniujmy więc sobie pewien włącznik:

```
włącznik mój_włącznik;
```

Mamy już w programie jeden obiekt klasy włącznik i od tego momentu możemy nim się posługiwać. W jaki sposób operujemy na danym obiekcie? Oczywiście używamy interfejsu, a więc jego funkcji składowych. By np. włączyć nasz włącznik możemy napisać:

```
mój_włącznik.włącz();
```

Jest to bardzo przejrzysty sposób operacji na obiekcie. Najpierw podajemy jego nazwę, a następnie po kropce funkcję, którą chcemy na jego rzecz wywołać. O poprawność wywołania funkcji składowej włącz() na rzecz obiektu mój_włącznik dba kompilator.

Możemy odwoływać się również do obiektów składowych. Na przykład po utworzeniu obiektu urządzenie, za włączanie, którego odpowiedzialny jest włącznik jest wyłączone i chcemy by było to w obiekcie uwzględnione. Można uczynić to w następujący sposób:

```
mój_włącznik.stan = false;
```

Oznacza to, że obiekt stan przyjmuje wartość false (fałsz) i tym samym uznajemy, że jest on wyłączony. Widzimy tutaj, że podobnie jak do funkcji można odnieść się do obiektów składowych danego obiektu i odpowiednio modyfikować ich wartości.

Obiekty i funkcje składowe są w pełni wartościowymi obiektami i funkcjami i mogą się pojawić w tych samych miejscach co zwykłe zmienne i funkcje w programie. Zaprezentowany sposób odnoszenia się bezpośrednio do elementu składowego włącznika nie jest dobrym pomysłem. Po pierwsze naraża nas to na pewne problemy związane z kontrolą przypisywanych do danego obiektu danych, a po drugie modyfikowanie kolejnych składowych obiektu jest niepraktyczne w przypadku gdy ich liczba w jednej klasie przekracza np. 10 (możemy zdefiniować klasę obiektów zawierającą informacje nie tylko o stanie włącznika, ale np. o tym jakiego on jest koloru, jaki jest jego rozmiar, ile razy był włączany i wyłączany, za włączanie i wyłączanie czego odpowiada) i mamy w programie np. 10 włączników to praca z nimi jest lekko mówiąc nieefektywna, a dosadniej - naraża nas na częste popełnianie pomyłek. Dlatego lepiej jest używać funkcji składowych do operacji na obiektach, po pierwsze można w niej odnosić się i zmieniać dowolną liczę parametrów klasy na raz, a dodatkowo można w nich łatwo dodać mechanizmy kontroli wprowadzanych danych. Ponieważ chcemy mieć możliwość ustawiania wartości odpowiednich dla naszego obiektu po jego konstrukcji możemy dodać kolejną funkcję składową:

```
class włącznik {
	bool stan;
	kolor kol;

	void ustaw(kolor _kol)
	{
		stan = false;
		kol = _kol;
	}

	void włącz()
	{
		stan = true;
		dalsze ciało funk-
cji odpowiedzialnej za włącza-
nie
	}

	void wyłącz()
	{
		stan = false;
		dalsze ciało funk-
cji odpowiedzialnej za wyłącza-
nie
	}
};
```

Mamy teraz dodatkową składową typu kolor, która zawiera informacje o kolorze obiektu typu włącznik, oraz funkcję, która umożliwia ustawienie domyślnych wartości stanu i koloru. Widzimy tutaj, że odwołując się do elementów składowych danej klasy wystarczy podać ich nazwy, a kompilator sprawdzi, czy występują one w danej klasie i odpowiednio nam je udostępni. Wartość zmiennej stan ustawiana jest domyślnie na wartość false, a kolor możemy podać jako parametr funkcji ustaw. Tak, więc, by utworzyć obiekt mój_włącznik typu włącznik i przypisać mu wartości domyślne możemy posłużyć się następującym sposobem:

```
włącznik mój_włącznik;
mój_włącznik.ustaw(biały);
```

Tworzymy w ten sposób obiekt włącznik i w drugiej linijce ustawiamy wartości jego obiektów składowych. Nie jest to jednak wygodne rozwiązanie. Obiekty ze swej natury powinny być równie proste jak typy składowe, a my potrzebujemy tym czasem, aż dwóch linijek by utworzyć i odpowiednio ustawić nasz obiekt. Nie jest to na pewno wygodne. Przychodzi nam z pomocą pewna specjalna funkcja składowa zwana konstruktorem. Wywoływana jest niejawnie przez kompilator w momencie tworzenia nowego obiektu. Konstruktor jest funkcją, która ma taką samą nazwę jak klasa, do której jest przypisany i nie zwraca on żadnych wartości, ale możemy do niego pewnie

dane wysłać. Tak więc nasza klasa wygląda teraz w sposób następujący:

```
class włącznik {
	bool stan;
	kolor kol;

	włącznik(kolor _kol)
	{
		stan = false;
		kol = _kol;
	}

	void włącz()
	{
		stan = true;
		dalsze ciało funk-
cji odpowiedzialnej za włącza-
nie
	}

	void wyłącz()
	{
		stan = false;
		dalsze ciało funk-
cji odpowiedzialnej za wyłącza-
nie
	}
};
```

Po tej niewielkiej zmianie możemy tworzyć obiekty w następujący sposób:

```
włącznik wł_biały = biały;
włącznik wł_czerwony(czerwony);
```

Widzimy, że pierwszy ze sposobów jest identyczny z tym jakiego używamy do nadawania wartości obiektom wbudowanym. Po prostu po znaku równości przekazujemy wartość, którą kompilator wysyła do konstruktora i nowemu obiektowi przypisany jest odpowiedni kolor. Drugi sposób jest nieco mniej prosty. Widzimy tutaj po prostu wpisanie wartości bezpośrednio po nazwie obiektu. Ta wartość również przekazywana jest do uruchamianego konstruktora, który przypisuje nam odpowiednio wartości dla składowych. Pierwszy sposób można używać tylko wówczas, gdy do konstruktora przekazujemy jedną wartość, a drugi sposób przypominający wywołanie funkcji może być korzystny gdy do konstruktora wysyłamy więcej niż jedną wartość.

W przypadku konstruktorów bardzo często stosowana jest technika zwana przeładowaniem funkcji. Oznacza to, że definiujemy sobie w klasie parę konstruktorów (każdy ma taka samą nazwę), ale różnią się one ilością i rodzajem przesyłanych do nich wartości. Możemy wówczas bardzo łatwo tworzyć obiekty, przekazując do konstruktora jak najmniej wartości. Jednocześnie konstruktor, który może zostać wywołany bez żadnego argumentu nazywany jest konstruktorem domyślnym.

Wiemy jak są tworzone obiekty, wiemy również, że nie mogą one istnieć w programie w nieskończoność tylko w pewnym momencie są likwidowane. Kiedy się to dokładnie dzieje zależne jest nie tylko od rodzaju języka programowania, którym się posługujemy, ale również od jego implementacji. Może się zdarzyć tak, że przed zlikwidowaniem włącznika możemy chcieć by urządzenie, które włączał było wyłączane, by po prostu po zakończeniu działania programu nie zużywało bez sensu prądu (umówmy się, że martwimy się o środowisko naturalne). Oczywiście możemy zawsze wywołać procedurę składową wyłącz przed likwidacją obiektu:

```
mój_włącznik.wyłącz();
```

Pojawiają się dwa problemy, bowiem wcale nie musimy, przecież wiedzieć, gdzie nasz obiekt jest likwidowany, lub po prostu nie chcemy pamiętać o tym, by zawsze wyłączyć dane urządzenie. Tutaj z rozwiązaniem przychodzi kolejny rodzaj funkcji składowej mianowicie destruktor. Jest on przeciwieństwem konstruktora. Również o jego uruchomienie nie musimy się martwić, gdyż w momencie likwidowania obiektu kompilator uruchomi go dla nas. Do destruktora nie możemy dodatkowo przesłać, ani odebrać żadnych danych, nie może być on również przeładowany, a więc dla danego obiektu jest tylko jeden destruktor. Destruktor podobnie jak konstruktor ma taką samą nazwę jak klasa, w której jest definiowany i dla odróżnienia dodatkowo przed jego nazwą znajduje się znak '~'. Teraz definicja naszej klasy wygląda w sposób następujący:

```
class włącznik {
	bool stan;
	kolor kol;

	włącznik(kolor _kol)
	{
		stan = false;
		kol = _kol;
	}

	~włącznik()
	{
		wyłącz();
	}

	void włącz()
	{
		stan = true;
		dalsze ciało funk-
cji odpowiedzialnej za włącza-
nie
	}

	void wyłącz()
	{
		stan = false;
		dalsze ciało funk-
cji odpowiedzialnej za wyłącza-
nie
	}
};
```

Dzięki utworzeniu destruktora dla kasy każdy obiekt w momencie likwidowania go będzie miał poprzez destruktor uruchamianą funkcję składową wyłącz(), po czym będzie kasowany.

Są to w zasadzie podstawowe elementy składowe każdej klasy w większości języków programowania. Pozostał jeszcze jeden bardzo ważny element – mianowicie dostęp do określonych elementów. Dostęp do elementów składowych wiąże się ściśle z bardzo ważnym zadaniem obiektów – z enkapsulacją danych. Pojęcie to oznacza, że w obiektach pewne elementy możemy chować przed użytkownikami naszej klasy. Dzięki czemu ma ona mniejszy interfejs, jest bardziej przejrzysta i zwiększa się bezpieczeństwo jej użytkowania. Czemu jest to takie ważne? Obiekty wymyślono jako sposób na usprawnienie pracy nad dużymi projektami. W projektach takich pewne zespoły

pracują nad określonym fragmentem programu. Dodają poszczególne elementy do swojej części często używając obiektów stworzonych przez inne zespoły. Nie interesuje ich sposób w jaki osiągnięto np. włączanie i wyłączanie danych urządzeń przez obiekty klasy włącznik, interesuje ich tylko to, że takie operacje można na włącznikach wykonać. Pisząc daną klasę należy robić to tak jak byśmy pisali ją dla jakiegoś innego programisty (nigdy przecież nie wiemy co się z nią stanie). Dobrze jest wówczas, pewne elementy do których użytkownik nie powinien mieć dostępu zabezpieczyć. Czemu? Choćby dlatego by nie dawać mu szans na zepsucie obiektu. Załóżmy. że posługując się naszą klasą użytkownik zrobi coś takiego:

```
mój_włącznik. włącz();
mój_włącznik.stan = false;
```

Najpierw włączy dane urządzenie, a następnie ustawi stan włącznika na wartość false, czyli wyłączony. Jest to oczywiście błąd, bo samo ustawienie stanu nic nie da. Urządzenie jest nadal włączone, ale włącznik już o tym nie wie i zaczynają się problemy. Można sobie zadać więc pytanie po co użytkownikowi dostęp do elementu składowego stan, skoro to nie on, ale funkcje składowe na nim operują? Jeśli postawimy sobie to pytanie i uznamy, że dostęp do tego elementu nie jest konieczny to dochodzimy do bardzo ważnej funkcji obiektów. Możemy bowiem pewne elementy składowe takie jak obiekty, czy funkcje uczynić prywatnymi dla obiektu. Występują one w danym obiekcie, ale użytkownik nie ma prawa do nich się odwoływać. Nie tylko je zmieniać, ale nawet czytać. Mogą na nich operować tylko funkcje składowe danego obiektu w naszym przypadku funkcje włącz i wyłącz, oraz konstruktory i destruktor. Jest to tak zwana enkapsulacja. Upraszczamy interfejs i jednocześnie zabezpieczamy nasz obiekt przed popsuciem, zmniejszając prawdopodobieństwo popełnienia błędu. Tryb prywatny nie jest jedynym trybem dostępu do obiektów. Występują również tryby chroniony i publiczny. Publiczny oznacza, że do danej składowej może odwoływać się każdy użytkownik, inne obiekty czy funkcje składowe, natomiast chroniony umożliwia odwoływanie się tylko klas dziedziczących daną klasę (o dziedziczeniu nie będę pisał). Nasza klasa wygląda teraz następująco:

```
class włącznik {
private:
	bool stan;
	kolor kol;
public:
	włącznik(kolor _kol)
	{
		stan = false;
		kol = _kol;
	}

	~włącznik()
	{
		wyłącz();
	}

	void włącz()
	{
		stan = true;
		dalsze ciało funkcji odpowiedzialnej za włączanie
	}

	void wyłącz()
	{
		stan = false;
		dalsze ciało funkcji odpowiedzialnej za wyłączanie
	}
};
```

Mamy w niej dwa obszary jeden rozciąga sie od etykiety private do public i oznacza część, która chroniona jest przed użytkownikiem, druga od public do końca definicji klasy, która dostępna jest dla wszystkich. Taki sposób organizacji klasy z podziałem jej na odpowiednie części o określonym dostępie świadczy o dobrym stylu programowania i dużej odpowiedzialności za projekt.

Znamy już wszystkie podstawowe właściwości klas i obiektów. Chcę zaznaczyć, że nie wszystkie one muszą być dostępne w każdym z języków obiektowych. W zasadzie wiele z nich posiada dodatkowe udogodnienia, które w jeszcze większym stopniu ułatwiają pracę, ale ich opis wykracza poza ramy tego artykułu.

Wiemy już jak definiować klasy i obiekty oraz jak ich używać. Nie wiemy jeszcze w jaki sposób stosować je w naszych programach. Po pierwsze klasy umożliwiają najwierniejsze oddanie rzeczywistości w programie z proponowanych do tej pory technik programowani. Przez co ich używanie i pisanie jest bardzo intuicyjne. Łatwiej jest napisać np. program symulujący działanie prostego mózgu (sieć neuronową) definiując sobie najpierw obiekty zwane neuronami, a następnie przy ich pomocy tworzyć odpowiednie elementy całej sieci. W tradycyjnym podejściu bylibyśmy zmuszeni zdefiniować szereg niepołączonych w programie, a jedynie logicznie funkcji i obiektów, Cały program przez to, że definiowany odmiennie od sposobu w jaki postrzegamy świat byłby trudny i odstraszający. Mimo tego, że programowanie obiektowe wydaje się być takie naturalne dla wielu osób, zwłaszcza tych przyzwyczajonych do programomania proceduralnego wydaje się być trudne, bo okazuje się, że najtrudniej jest robić z rzeczy skomplikowanych proste, a właśnie do tego Programowanie Orientowane Obiektowo najlepiej się nadaje. Oczywiście wszystko ma swoje granice i nie zawsze opłaca się wprowadzać obiekty, gdyż w bardzo prostych programach, typu „Hello World" mogą one całą sprawę nadmiernie komplikować.

Jak planować program obiektowy? Nie jest to proste pytanie i by być dobrym programistą obiektowym po pierwsze potrzebne jest pewne doświadczenie, a po drugie wiedza. Programu obiektowego nie można po prostu napisać, trzeba go wcześniej odpowiednio zaprojektować. Dla sporej rzeszy osób jest to duży minus Programowania Obiektowego, ale dzięki temu nasz program zamiast być bezładną plątaniną obiektów i klas stanie się perełką, z której będziemy dumni. Po pierwsze należy bardzo dogłębnie po-

znać model, który będziemy odwzorowywać. Następnie należy się zastanowić jakie klasy będą nam potrzebne np. neurony w mózgu (ważne jest by definiować tyle klas ile jest to naprawdę niezbędne do rozwiązania problemu, nie mniej nie więcej, często bowiem można tworzyć obiekty, przy których niepotrzebnie się napracujemy i jednocześnie będą one cały program niepotrzebnie gmatwać), musimy określić odpowiednie zależności między nimi i to jak nawzajem ze sobą oddziałują. Dopiero po tym możemy zacząć programować. Jest to oczywiście bardzo skrócony opis, powstało wiele książek traktujących o projektowaniu programów.

Czy obiektowość jest ostatecznym etapem rozwoju programów i czy nie da się czegoś nowego wymyślić? Odpowiedzieć jest prosta - obiektowość na pewno nie jest ostatnią rzeczą jaką można w programowaniu poprawić i na pewno nie rozwiązuje wszystkich problemów. By ją jeszcze bardziej usprawnić wprowadzono mechanizmy dziedziczenia, funkcji wirtualnych i polimorfizm. Na pewno jeszcze wiele do programowania można wnieść przykładem mogą być wciąż rozwijane szablony stosowane w C++.

Mam nadzieję, że wiele osób po przeczytaniu tego krótkiego artykułu o programowaniu obiektowym poczuje się pewniej i spróbuje swoich sił w tej dziedzinie.

Zapraszamy na FORUM

http://www.forum-dragonia.bee.pl

C++ cz.5

W tym odcinku skończymy omawianie podstawowych elementów języka C++ i od przyszłego numeru zaczniemy zajmować się już typami definiowanymi przez programistę czyli tak zwanymi klasami. Zachęcam do przeczytania mojego artykułu o Programowaniu Obiektowym w tym numerze. Jest to wstęp do techniki, którą będziemy poznawać od następnego numeru.

MACIEJ 'borimir' MALINOWSKI

Omawiane będą dzisiaj wskaźniki i nowość w C++ w stosunku do C - referencja. Czym są wskaźniki? Są to obiekty, w których przetrzymujemy adresy innych obiektów. Używając ich możemy się odnosić do danych wskazywanych przez ten adres. Pewnie wiele osób zastanawia się po co taka konstrukcja? Otóż powstała ona by usprawnić pracę z tablicami i umożliwić modyfikowanie wartości argumentów przekazanych do funkcji.

Operatorem wskaźnika jest znak '*'. Stawia się go przed nazwą wskaźnika w miejscu definicji. Przykładem wskaźnika, który może przechowywać adres obiektów typu double jest:

```
double * wskaznik;
```

Następnie możemy takiemu wskaźnikowi przypisać adres jakiegoś obiektu typu double:

```
double pi = 3.14;
double * wskaznik;
wskaznik = &pi;
```

Poznajemy kolejny operator `&`. Jest to tak zwany operator referencji. Stawia się go przed nazwą obiektu i zwraca on adres tego obiektu w pamięci. Każdy obiekt (zmienna, funkcja) po zdefiniowaniu otrzymuje określone miejsce w pamięci, w którym zapisana jest jego wartość. Kolejna ważną sprawą jest sposób odwoływania się do obiektu wskaznik. Jeśli przypisujemy mu adres odpowiedniej zmiennej to nie stosujemy `*` przy nazwie wskaźnika. Natomiast jeśli chodzi nam o wartość zmiennej, na którą wskazuje to znak '*' jest wymagany.

Co można zrobić ze wskaźnikiem (zakładając, że mamy już poprawnie przypisany mu adres)? Najważniejszą sprawą jest to, że możemy zrobić to samo co ze zwykłą zmienną typu double (a dokładnie z jej wartością). Do wartości zmiennej, którą wskazuje dany wskaźnik odnosimy się poprzez operator dereferencji '*' (ten sam, którego używaliśmy w momencie definiowania obiektu).

```
cout << *wskaznik << endl;
```

Zakładając ze wskaznik jest obiektem z powyższych przykładów na konsoli powinniśmy otrzymać wartość 3.14. Można powiedzieć, że wskaźnik jest innym sposobem odwołania się do danej zmiennej poprzez jej adres. Na razie nie widać nic ciekawego w tej możliwości gdyż do tej pory odwoływanie się w zwykły sposób nam wystarczało, ale zanim poznamy nowe możliwości muszę dokładniej opisać wskaźniki.

Po pierwsze wskaźnik nie przechowuje jedynie adresu obiektu do którego się odnosi, ale również typ tego obiektu. Dlatego w momencie definicji podaliśmy typ double dla wskaźnika wskaznik. Ważne jest to, że wskaźnik danego typu może wskazywać tylko na zmienne tego samego typu. Nie można przypisać adresu zmiennej typu char wskaźnikowi do double. Do danego wskaźnika możemy wielokrotnie w trakcie pisania programu przypisywać różne adresy zmiennych jeśli tylko typ wskaźnika na to pozwala. Definiując wskaźniki możemy określić w ich definicji posługując się modyfikatorem const czy ma to być wskaźnik do obiektu stałego (nie możemy wówczas modyfikować wartości tego obiektu poprzez ten wskaźnik), czy ma to być stały wskaźnik – wówczas można modyfikować zmienną, której adres zawiera poprzez ten wskaźnik, ale nie można zmieniać wartości adresu zapisanego w tym wskaźniku, ostatnia możliwość to stały wskaźnik do stałej, który łączy powyższe dwie cechy. A oto przykłady definicji:

```
const int * wsk_i;          //
 wskaźnik do stałej
char * const wsk_c;         //
stały wskaźnik do zmiennej
const double * const
wsk_d; // stały wskaźnik do sta-
łej
```

Poznaliśmy już sposób w jaki definiuje się wskaźnik do obiektu pojedynczego. Jak w takim razie wygląda wskaźnik do tablicy? Otóż wygląda on identycznie. Jako typ obiektu wskazywanego należy podać typ, który w tablicy jest przechowywany.

```
char imie[] = „Jan”;
char *wsk_imie = imie;
```

Widzimy definicję tablicy znaków, która przechowuje ciąg imie i kończący go znak '\0' (tablica ta ma rozmiar równy 4) oraz definicję wskaźnika, którym można wskazywać obiekty typu char. Jednocześnie mamy w tym miejscu inicjalizację wskaźnika. Przypisujemy mu adres tablicy imie. I tutaj jest pierwsza niespodzianka w momencie przypisywania adresu nie stosujemy w odniesieniu do tablicy operatora '&', ponieważ nazwa tablicy zawiera adres swojego pierwszego elementu. Może wydawać się to skomplikowane, ale jest bardzo logiczne.

Gdyż nazwę tablicy możemy traktować jako wskaźnik do jej pierwszego elementu. Gdy umiemy już definiować wskaźniki do zmiennych i tablic możemy zademonstrować jak znacznie upraszcza to pracę (Listing 1).

Listing 1

```
#include <iostream>
using namespace std;

char * strcat(char * a, char * b);//(1)

int main()
{
   char imie_nazw[100] = "Jan";    //(2)
   char * nazwisko = " Kowalski"; //(3)
   cout << strcat(imie_nazw, nazwisko) << endl;     //(4)
   return 0;              //(5)
}

char * strcat(char * a, char * b) //(6)
{
   char * poczatek = a;   //(7)
   while( *++a); //(8)
   while( *a++ = *b++ );  //(9)
   return poczatek;       //(10)
}
```

Wynik działania programu:

Jan Kowalski

1)W tym miejscu mamy deklarację funkcji strcat, której zadaniem jest dołączać tablicę znakową podaną jako drugi argument funkcji na sam koniec tablicy znaków podanej jako pierwsza. Definicja właściwa treści tej funkcji znajduje sie tutaj (6).

2)Tworzymy tablicę znaków imie_nazw o odpowiednio długim rozmiarze (by zmieścić jeszcze dopisywane nazwisko) i przypisujemy jej imię.

3)Tworzymy drugą tablicę i umieszczamy w niej nazwisko.

4)Tutaj wywołujemy naszą funkcję i przekazujemy do niej odpowiednie dwie tablice znakowe. Widzimy również, że wartość zwracana przez naszą funkcję przekazywana jest od razu do strumienia. Dzieje się tak ponieważ nasza funkcja zwraca wskaźnik typu znakowego, a strumień jest tak zaprojektowany, że przy pomocy takich wskaźników wyświetla tekst zawarty w tablicy znakowej, którą wskazuje ten wskaźnik.

5)Kończymy program zwracając z funkcji main wartość 0, czyli informację o poprawnym zakończeniu programu.

6)W tym miejscu rozpoczyna się właściwa definicja naszej funkcji. O przekazywanych do niej parametrach i odbieranych z funkcji wartości mówiliśmy już wcześniej.

7)Tworzymy nowy wskaźnik, który ustawiamy na początek naszej pierwszej tablicy i to on będzie zwracany z funkcji. Czemu tak postępujemy? Ponieważ wewnątrz funkcji będziemy zmieniać zawartość wskaźnika a i po wszystkich naszych operacjach nie będzie już wskazywać na początek tablicy.

8)Jest to bardzo zwięzła i przez to na pierwszy rzut oka przerażająca konstrukcja, ale takie formy są często spotykane w programach operujących na wskaźnikach, nie dlatego, że wskaźniki są trudne, ale dlatego, że umożliwiają tak daleko idące uproszczenie składni. W tym miejscu ustawiamy wskaźnik a pokazujący na tablicę znaków na jej koniec. Jak wiemy tablice znakowe w C++ zakończone są wartością '\0', czyli zerem. Najpierw zwiększamy wartość wskaźnika p przy pomocy operatora preinkrementacji - zwraca on wartość wskaźnika zwiększoną o jeden. Wskaźnik wskazuje na daną komórkę pamięci jeśli zwiększymy jego wartość to przesuwamy go na komórkę następną, czyli w naszym przypadku z litery J na a. Ponieważ zwracana jest wartość wskaźnika, a jak wiemy przechowuje on adres to by sprawdzić co pod danym adresem się znajduje używamy operatora dereferencji '*'. Operator ten zwróci zawartość znajdującą się pod podanym adresem. Zwraca a. Wartość ta staje się warunkiem pętli while, która wykonuje się póki do póty warunek jest prawdziwy. Ponieważ w C++ istnieje domniemanie, że prawdą jest wszystko różne od zera, a kolejne litery J, a, n są od niego różne pętla się wykonuje i zatrzyma w momencie gdy zwrócony zostanie znak '\0' równy zero. Będziemy wówczas mieć ustawiony wskaźnik na koniec tablicy znakowej.

9)W tym miejscu przypisujemy kolejne znaki z tablicy znakowej b na koniec tablicy a. Zasada jest bardzo podobna do powyższej i nie będę jej dokładnie opisywać.

10)W tym miejscu kończymy naszą funkcję zwracając początkowy adres tablicy znakowej a.

Nowością w C++ jest wprowadzenie referencji, które w znaczy sposób ułatwiają przekazywanie wartości, zwłaszcza bardzo dużych obiektów, do funkcji, ale nie tylko. Referencję można porównać do wskaźnika z tą jednak różnicą, że nie zawiera ona adresu danego obiektu w pamięci, ale bezpośrednio się do niego odwołuje. Definicja referencji musi być połączona z przypisaniem jej wartości. Musi więc nastąpić jej inicjalizacja. Oto przykład definicji:

```
double pi = 3.14;
double &ref = pi;
```

W odróżnienia do wskaźnika referencję definiuje się z operatorem referencji '&'. Ten sam operator w innym miejscu niż definicja zmiennej służy do uzyskania jej adresu. Ważną cechą referencji jest to, że w bardzo naturalny sposób można się nią posługiwać. Mianowicie identycznie jak zwykłą zmienną i de facto przy jej pomocy operujemy na wartościach zapisanych w zmiennej, która jest przypisywana w momencie definicji. Referencja używana jest zwłaszcza jako argument funkcji, ponieważ znacznie przyspiesza przekazywanie do niej wartości. Referencja bowiem zapobiega przed spowalnianiem programu w momencie przekazywania dużych obiektów do funkcji. W typowej funkcji przyjmującej argument przez wartość w momencie przekazywania do niej danych następuje wykonanie ich kopi w zmien-

nej lokalnej i to na nich funkcja operuje, dlatego po jej zakończeniu wartość przekazywanego obiektu nie ulega zmianie. Inaczej jest w przypadku referencji. Nie mamy tutaj do czynienia z kopiowaniem wartości, ale funkcja bezpośrednio poprzez referencję operuje na zmiennej do niej wysłanej. Dzięki temu program działa szybciej, ale niebezpieczne może się okazać to, że wówczas wszelkie zmiany dotyczące tej zmiennej zachodzące wewnątrz funkcji dotyczą zmiennej zewnętrznej. Zobrazujmy to programem (Listing 2):

Wynik działania programu:

```
Wartość zmiennej przed wysłaniem do funkcji przez wartość : 2
Wartość zmiennej wysłanej do funkcji przez wartość wynosi : 2
Po zwiększeniu 5
po wykonaniu funkcji : 2
Wartość zmiennej przed wysłaniem do funkcji przez referencję : 2
Wartość zmiennej wysłanej do funkcji przez referencję wynosi : 2
Po zwiększeniu 5
po wykonaniu funkcji : 5
```

Listing 2

```
#include <iostream>
using namespace std;
void wartosc(int a);
void referencja(int &a);
int main()
{ int a = 2;     //(1)
   cout << "Wartość zmiennej przed wysłaniem do funkcji przez wartość : " << a << endl;       //(2)
   wartosc(a);   //(3)
   cout << "po wykonaniu funkcji : " << a << endl << endl;  //(4)
   cout << "Wartość zmiennej przed wysłaniem do funkcji przez referencję : " << a << endl;//(5)
   referencja(a);
   cout << "po wykonaniu funkcji : " << a << endl;
   return 0;
}
void wartosc(int a)       //(6)
{
   cout << "\tWartość zmiennej wysłanej do funkcji przez wartość wynosi : " << a << endl;
   a += 3;
   cout << "\tPo zwiększeniu " << a << endl;
}
void referencja(int &a)   //(7)
{
   cout << "\tWartość zmiennej wysłanej do funkcji przez referencję wynosi : " << a << endl;
   a += 3;
   cout << "\tPo zwiększeniu " << a << endl;
}
```

(1)Tworzymy zmienną a, którą będziemy się posługiwać w trakcie działania naszego programu.
(2)Wyświetlamy wartość zmiennej a by udowodnić zmiany związane z jej wartością w ciągu działania całego programu.
(3)Wywołujemy funkcję, która odbiera zmienną przez wartość.
(4)Wyświetlamy zawartość zmiennej po wykonaniu funkcji.
(5)To samo wykonujemy dla przekazywania zmiennej do funkcji odbierającej zmienną przez referencję.
(6)Tutaj definiujemy funkcję wartosc, która odbiera zmienną w ten sam sposób jak się nazywa. Dochodzi tutaj do kopiowania wartości zmiennej i umieszczanie jej w zmiennej lokalnej. Dzięki temu funkcja pracuje na kopi zmiennej i zmiany dokonywane w jej obrębie nie mają żadnego pływu na zmienną do niej wysłaną. Dowodem tego jest wydruk działania programu.
(7)Tutaj definiujemy funkcję, która obiera zmienną przez referencję i zmiany wartości dokonane w jej obrębie dotyczą też zmiennej zewnętrznej. Z jednej strony przyspiesza to działanie programu w przypadku dużych obiektów, które muszą być często kopiowane, ale może doprowadzać do niezrozumiałego działania programu, gdy wewnątrz funkcji wartość tych obiektów w „tajemniczy" sposób ulega zmianie. Należy więc unikać zmiany wartości obiektu przekazanego do funkcji przez referencję, jeśli nie wynika to wprost z działania funkcji. Na przykład nie jest rozsądne mnożenie zmiennej przez liczbę pi wewnątrz funkcji, która tą zmienną ma wyświetlić na ekranie.

Gdy znamy już wskaźniki i referencję możemy teraz porozmawiać o zastosowaniu tych struktur. O referencjach już mówiliśmy, ale wskaźniki oprócz zastosowania do usprawnienia pracy z tablicami służą również do odnoszenia się do określonych komórek pamięci komputera oraz do tak zwanej dynamicznej rezerwacji tablic. Zdarza się, że musimy napisać program, który obsługuje duże ilości danych, ale dokładnie nie wiemy ile będzie tych danych. Jak dobrze wiemy z poprzedniego odcinka tablice w C++ mają stałą wielkość, która musi być podana w momencie ich definiowania w programie jeszcze przed kompilacją. Można powiedzieć, że jesteśmy w ten sposób mocno ograniczeni. Problem pojawia się w momencie gdy nie jesteśmy w stanie stwierdzić jak wielka tablica będzie nam w rzeczywistości

potrzebna. Możemy wówczas wybrać odpowiednio większą wielkość, uwzględniając ewentualne dodatkowe elementy na wszelki wypadek. Jednak w tym przypadku albo będziemy marnować sporą część pamięci, ponieważ nasza tablica będzie za duża, albo jeśli źle ocenimy zapotrzebowanie na pamięć okaże się za mała.

Jeśli dotykają nas takie problemy to z pomocą przychodzą dwa operatory, które służą do dynamicznej alokacji pamięci i jej zwalniania. Operator new służy do przydzielania nowego miejsca w pamięci, natomiast delete to miejsce zwalnia. W C++ nie ma odśmiecacza z tego powodu jesteśmy zmuszeni samodzielnie zwalniać alokowaną przez nas pamięć. Dzieje się tak ponieważ w C++ położono głównie nacisk na prędkość działania programów a odśmiecacz by znacznie je spowalniał. Pamięc alokujemy w bardzo prosty sposób. Po prawej stronie operatora new podajemy dla jakiego obiektu ma nam alokować miejsce, a zwraca wskaźnik do tego miejsca w pamięci. Najbardziej interesuje nas to, że operator new umożliwia alokowanie pamięci pod tablice, których rozmiar może być określany już w momencie wykonywania programu. Taką tablice musimy wówczas usunąć odpowiednim operatorem delete. Istnieją dwa rodzaje operatorów new i delete. Jedne do pracy z pojedynczymi obiektami nazywają się one new i delete, a drugie do pracy z tablicami odpowiednio new [] i delete []. Ważne jest to, żeby stosować się do kilku zasad. Mianowicie dany obiekt możemy usunąć tylko raz, dodatkowo możemy zwalniać pamięć tylko dla obiektów, które sami alokowaliśmy, oraz, że obiekty alokowane przy pomocy new usuwamy delete, a new [] - delete []. W jaki sposób zwolnić pamięć - po prostu po operatorze delete umieszczamy wskaźnik wskazujący na wcześniej alokowane komórki (Listing 3).

Listing 3

```
#include <iostream>
using namespace std;
int main()
{
   int rozmiar;
   cout << "Jaki rozmiar tablicy należy zaalokować : ";
//(1)
   cin >> rozmiar;
   char *string; //(2)
   string = new char[rozmiar];     //(3)
   cout << "Jak masz na imię : ";
   cin >> string;//(4)
   cout << "Witaj " << string << "!!\n";
   delete [] string;      //(5)
}
```

Wynik działania programu:

```
Jaki rozmiar tablicy należy zaalokować : 12
Jak masz na imię : Maciek
Witaj Maciek!!
```

(1)Wczytujemy z klawiatury rozmiar tablicy jaka będzie nam potrzebna. Widać wyraźnie, że rozmiar jej przy każdym uruchomieniu programu może być inny i w żaden sposób nie jest znany w momencie kompilacji.

(2)Tworzymy sobie wskaźnik, w którym zapiszemy adres alokowanej tablicy.

(3)W tym miejscu za pomocą operatora new [] alokujemy tablicę o rozmiarze rozmiar. Operator zwraca nam adres do tego miejsca, którego możemy używać by odnieść się do tej tablicy.

(4)Wczytujemy jakieś dane do naszej tablicy np. imię.

(5)Po zakończeniu pracy z naszą tablicą zwalniamy pamięć używając operatora delete [].

Nauczyliśmy się obchodzić jedno z poważnych ograniczeń C++, czyli konieczność posługiwania sie tablicami określonego typu. Dodatkowo operator new umożliwia nam tworzenie naprawdę dużych tablic.

Ostatnim zagadnieniem, które będziemy poruszać w dzisiejszym odcinku jest funkcja main. Wiemy już, że main jest zwykłą funkcją, więc możemy zwracać z niej wartości (przy pomocy return), ale gdzie wartości te docierają? Docierają do miejsca systemie, które nas program wywołały. W przypadku systemów Linux jest to powłoka, które używamy do uruchomiania programu. By sprawdzić co zwrócił nasz program proponuje bezpośrednio po jego wykonaniu wpisać polecenie 'echo $?' - zwróci one wartość stojącą za return w funkcji main.

Wiemy również, że funkcje nie tylko zwracają pewne wartości, ale również mogą je otrzymywać. Tak również jest w przypadku funkcji main. Zanim poznamy sposób w jaki te wartości odbiera musimy odpowiedzieć sobie na pytanie skąd te wartości są wysyłane? Odpowiedzią ponownie jest powłoka systemowa. Wartości docierające do funkcji main to te znajdujące się po nazwie programu w momencie wpisywania jego polecenia do funkcji. Zanim napiszemy pierwszy program odbierający dane z linii poleceń musimy poznać sposób zapisu argumentów w funkcji main, bo jest on ściśle określony. Dane do funkcji main docierają w postaci tablicy wskaźników do tablic znakowych, każda z tablic zawiera jeden z parametrów przekazanych do programu w linii poleceń. Dodatkowo ostatni wskaźnik ma adres zero, a pierwszy odnosi się do tablicy zawierającej nazwę programu. Jednocześnie do funkcji przekazana jest liczba parametrów. Może to wyglądać dosyć skomplikowanie, ale w rzeczywistości jest to bardzo proste. Zwyczajowo tablicę wskaźników do tablic znakowych nazywamy argv, a licznik jej elementów argc. Przykład programu (Listing 4):

Wynik działania programu:

```
[borimir@fujitsu Source]$ ./main -l raz "dwa trzy"
Argument nr. 0 >./main<
Argument nr. 1 >-l<
Argument nr. 2 >raz<
Argument nr. 3 >dwa trzy<
```

(1)Definiujemy funkcje main i jako pierwszy argument podajemy zmienną, która przechowywać będzie ilość zawar-

Listing 4

```
#include <iostream>
using namespace std;

int main(int argc, char*argv[]) //(1)
{
  for(int i = 0 ; i < argc ; i++) //(2)
    cout << "Argument nr. " << i << " >" << argv[i] << "<" << endl;
}
```

tych w tablicy wskaźników, czyli ilość wysłanych do programu argumentów. Oraz drugą, która jest tablicą wskaźników do tablic znakowych.

(2)W tym miejscu rozpoczynamy pętle, która wyświetla zawartość wszystkich tablic znakowych wskazywanych przez kolejne wskaźniki zapisane w tablicy.

Wynik działania programu jest oczywisty. Widzimy w kolejnych linijkach wypisane wartości parametrów przekazanych w momencie uruchamiania naszego programu. Pierwszą wartością jest nazwa programu. Należy zaznaczyć, że kolejne argumenty określane są na podstawie rozdzielających je spacji chyba, że umieścimy je między parą cudzysłowów.

Na tym skończymy omawianie podstawowych elementów C++ i od przyszłego numeru jak zapowiadałem zajmiemy się definiowaniem klas i wykorzystywaniem obiektów. ■

KONKURS

Do wygrania Licencja + płyta DVD z Mandrivą Power Pack 2007 Gold.

Odpowiedz na pytania wyślij wraz z podaniem imienia i nazwiska oraz adresu i kontaktu (mail, telefon) na adres **konkurs@dragonia.pl**. Pośród Prawidłowych odpowiedzi rozlosujemy Mandrivę Power Pack 2007 Gold (Licencje + płytkę DVD).

Oto pytania:

1. *Na ilu płytach DVD mieści się Slackware 11?*

2. *Wymień przynajmniej dwa darmowe programy graficzne dla linuksa.*

3. *Co oznacza skrót Mandriva CS4 ?*

Na odpowiedzi czekamy do 10 stycznia.

Życzymy szczęścia w losowaniu!!

Wprowadzenie do Tcl\Tk cz.3

W artykule przedstawiono przykładową aplikację Tcl/Tk - edytor tekstu. Mimo prostoty przykładu można pokazać przykłady programowania w języku Tcl.

TOMASZ ŁUCZAK

Wstęp do wprowadzenia do Tcl

Poniższy „wyciąg" z języka Tcl ma na celu danie podstaw do zrozumienia przykładu. Nie należy traktować że jest to zwięzły opis języka Tcl!

Zmienne

Zmienną tworzymy za pomocą polecenia:

```
set nazwaZmiennej ?wartość?
```

istniejącą zmienną można usunąć poleceniem:

```
unset nazwaZmiennej1 ?nazwaZmiennejn?
```

Do zmiennej odwołujemy się (przytaczamy jej wartość) poprzedzając jej nazwę znakiem dolara:

```
set zmiennaA $zmiennaB
```

Typy danych

W języku Tcl mamy trzy typy danych: łańcuch, lista i tablica. Elementy listy są zwykle ujęte w nawiasy { i }.

Polecenia

Składnia polecenia Tcl jest postaci:

```
polecenie ?-opcja wartość? ?argument1? ... ?argumentn?
```

Wszystkie elementy polecenia są separowane spacją, a polecenia znakiem końca wiersza lu poleceniem średnika.
Język Tcl umożliwia podstawianie poleceń, czyli zwracanych przez polecenie wartości, przez ujęcie polecenia w nawias kwadratowy [i], np:
set a [pwd]

Wyrażenia

Wyrażenia to kombinacja operatorów i operandów, dla zapewnienia właściwej kolejności obliczeń można stosować nawiasy okrągłe.
Operandami mogą być: łąńcuchy znaków, wartości logiczne, podstawienia poleceń (czyli wynik polecenia) oraz funkcje matematyczne.

Instrukcje sterujące

Instrukcja warunkowa if jest postaci:

```
if wyrażenie ?then? polecenie1 ?elseif polecenie2 else polecenie3?
```

Dla zapewnienia poprawnej interpretacji instrukcji, wygodnie jest zapisywać polecenie oraz wyrażenie warunkowe w nawiasach { i } i w oddzielnych wierszach, np:

```
if {a eq 1} then {
  set b 1
  set c 2
} else {
  set b 2; set c 1
}
```

Istnieją jeszcze inne instrukcje sterujące, znane z innych języków: switch, while, for, foreach. Nie są one wykorzystane w przykładzie, dlatego nie zostały opisane.

Procedury

Dla większej przejrzystości kodu i wykorzystaniu tego samego kawałka kodu można wykorzystać procedury. Ponadto zdefiniowane w nich zmienne są zmiennymi lokalnymi niedostępnymi poza procedurą. Postać procedury:
proc nazwa ?argumenty? polecenia
Do procedury można przekazywać argumenty. Polecenia procedury mogą mieć dostęp do zmiennych globalnych, jeśli w treści procedury podano ich nazwy po słowie kluczowym global.
Procedura wywoływana jest przez podanie nazwy i jeśli zdefiniowano procedurę z parametrami to należy podać parametry.
Wykonanie procedury zwraca wynik ostatniego wykonananego w niej polecenia.

Obsługa błędów

Do obsługi wyjątków służy polecenie catch postaci:

```
catch polecenie ?zmienna?
```

Jeśli polecenie wygeneruje błąd, polecenie catch zwróci niezerową wartość, a w zmiennej `zmienna` zawarty jest komunikat o błędzie.

Operacje na plikach

Za pomocą polecenia open uzyskujemy dostęp do plików. Polecenie jest postaci:

```
textttopen plik ?trybdostępu? ?prawa?
```

gdzie: trybdostępu określa czy to będzi zapis, odczyt, z usunięciem zawartości czy nie, a prawa to uniksowe prawa dostępu do pliku.
Zapisu i odczytu z otwartego kanału dokonujemy za pomocą poleceń puts i read.
Otwarty kanał zamykamy poleceniem close

Objaśnienia do kodu przykładu

Wykorzystane widgety zostały już przedstawione we wcześniejszych artykułach. Teraz zaprezentowano ich przykładowe zastosowanie.

Wiersz 9
Polecenie source pozwala na wczytanie i wykonanie zewnętrznego pliku z kodem w języku Tcl.

Wiersz 25
Polecenie menu wykonuje procedurę nowyPlik

Wiersz 27
Polecenie menu wykonuje procedurę zapiszPlik z parametrem 0

Wiersz 33
Polecenie menu wykonuje wbudowane polecenie widgetu.

Wiersze 42-46
Instrukcja sterująca if. Wykonywany jest polecenie w nawiasach po słowie kluczowym then jeśli wartość zmiennej zawijajWiersze jest równa 1. W przeciwnym wypadku wykonane zostanie polecenie zawarte w nawiasach po słowie kluczowym else. Polecenia wykonywane w zależności od warunku konfigurują okno edytora tak by zawijał wiersze treści lub nie zawijał.

Wiersze 52-56
Kolejna instrukcja sterująca. W wyniku jej działania główne okno jest w trybie edycji lub jest zablokowane.

Wiersze 89-105
Procedura otworzPlik bez parametrów (pusty nawias) wywołuje okno wyboru pliku (wiersze 91-92). Następnie otwiera wybrany plik (wiersz 93), jeśli otwarcie się nie powiedzie, to wywoływane jest okno dialogowe z odpowiednim komunikatem.

Jeśli zaś otwarcie się powiedzie, to do zmiennej zawartoscPliku wczytywana jest zawartośc pliku (wiersz 94) i zamykany otwarty kanał (wiersz 95). Następnie usuwana jest zawartość okna od pierwszego znaku do ostatniego (wiersz 96). Polecenie w wierszu 97 dodaje do okna zawartość pliku.

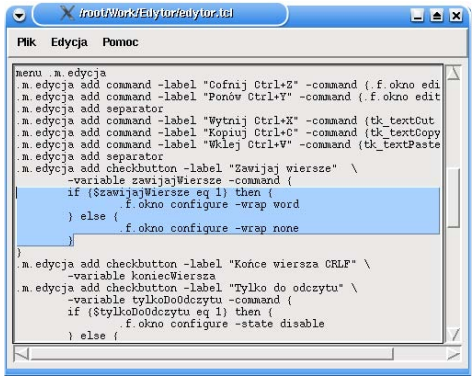
W tytule okna wyświetlona zostanie nazwa pliku (wiersz 98).

Wiersz 113
W wierszu 113 zdefiniowana jest procedura z jednym parametrem. Nazwa tego parametru wykorzystywana jest w treści procedury.

Wiersze 122 i 124
W zależności od wybranej konwencji znaków końca wiersza, dokonywana jest konfiguracja kanału.

Wiersz 126
Zapis pobranej z widgetu (wiersz118) zawartości okna edytora jest zapisywana do kanału.

Wiersze128-132
W razie niepowodzenia zapisu wyświetlany jest komunikat błędu. ■



Listing

```
1   #!/bin/sh
2   #the next line restarts using wish \
3   exec tclsh "$0" "$@"
4
5   # wywołanie biblioteki graficznej
6   package require Tk
7
8   # wczytanie i wykonanie pliku zewnętrznego
9   source procedury.tcl
10
11  # zmienne
12  set zawijajWiersze 1
13  set koniecWiersza 0
14  set nazwaPliku ""
15
16  # tworzenie menu
17  menu .m
18  # konfiguracja głównego okna z menu
19  . configure -menu .m
20  .m add cascade -label "Plik" -menu .m.plik
21  .m add cascade -label "Edycja" -menu .m.edycja
22  .m add cascade -label "Pomoc" -menu .m.pomoc
23  # zawartosc menu Plik
24  menu .m.plik
25  .m.plik add command -label "Nowy" -command nowyPlik
26  .m.plik add command -label "Otwórz" -command otworzPlik
27  .m.plik add command -label "Zapisz" -command { zapiszPlik 0 }
28  .m.plik add command -label "Zapisz jako" -command {     zapiszPlik 1 }
29  .m.plik add separator
30  .m.plik add command -label "Koniec" -command exit
31  # zawartosc menu Edycja
32  menu .m.edycja
33  .m.edycja add command -label "Cofnij Ctrl+Z" -command {.f.okno edit undo}
34  .m.edycja add command -label "Ponów Ctrl+Y" -command {.f.okno edit redo}
35  .m.edycja add separator
36  .m.edycja add command -label "Wytnij Ctrl+X" -command {tk_textCut .f.okno}
37  .m.edycja add command -label "Kopiuj Ctrl+C" -command {tk_textCopy .f.okno}
38  .m.edycja add command -label "Wklej Ctrl+V" -command {tk_textPaste .f.okno}
39  .m.edycja add separator
40  .m.edycja add checkbutton -label "Zawijaj wiersze"  \
41          -variable zawijajWiersze -command {
42          if {$zawijajWiersze eq 1} then {
43                  .f.okno configure -wrap word
44          } else {
45                  .f.okno configure -wrap none
46          }
47  }
48  .m.edycja add checkbutton -label "Końce wiersza CRLF" \
49          -variable koniecWiersza
50  .m.edycja add checkbutton -label "Tylko do odczytu" \
```

```
51          -variable tylkoDoOdczytu -command {
52          if {$tylkoDoOdczytu eq 1} then {
53                  .f.okno configure -state disable
54          } else {
55                  .f.okno configure -state normal
56          }
57  }
58  # zawartosc menu Pomoc
59  menu .m.pomoc
60  .m.pomoc add command -label "Pomoc" -command pomoc
61  .m.pomoc add command -label "O programie" -command {
62          tk_messageBox -title "O programie" \
63                  -message "Ja jestem autorem, a to jest mój pierwszy program"
64  }
65  # menu podręczne tez sie przyda
66  menu .mp
67  .mp add command -label "Cofnij" -command {.f.okno edit undo}
68  .mp add command -label "Ponów" -command {.f.okno edit redo}
69  .mp add separator
70  .mp add command -label "Wytnij" -command {tk_textCut .f.okno}
71  .mp add command -label "Kopiuj" -command {tk_textCopy .f.okno}
72  .mp add command -label "Wklej" -command {tk_textPaste .f.okno}
73  .mp add separator
74  .mp add command -label "Zapisz" -command {zapiszPlik 0 }
75  bind . <Button-3> {tk_popup .mp %X %Y}
76  # okno edytora
77  frame .f
78  text .f.okno -width 80 -height 30 \
79          -wrap word -undo 1 \
80          -xscrollcommand {.f.sbx set} \
81          -yscrollcommand {.f.sby set}
82  scrollbar .f.sbx -orient horizontal -command {.f.okno xview}
83  scrollbar .f.sby -orient vertical -command {.f.okno yview}
84  pack .f -padx 2m -pady 2m
85  pack .f.sbx -side bottom -fill x
86  pack .f.sby -side right -fill y
87  pack .f.okno
# Koniec pliku
```

Listing

```
88   proc otworzPlik {} {
89        global nazwaPliku
90        set nazwaPliku [tk_getOpenFile -title "Otwórz plik" \
91           -filetypes {{Wszystkie *} {Tcl .tcl} {Tekst .txt}}]
92        if { [catch {set input [open $nazwaPliku r]} wynik] eq 0} then {
93             set zawartoscPliku [read $input]
94             close $input
95             .f.okno delete 0.0 end
96             .f.okno insert 0.0 $zawartoscPliku
97             wm title . "$nazwaPliku"
98        } else {
99             set zawartoscPliku ""
100            tk_messageBox -title "Błšd" -icon error \
101                 -message "Nie można otworzyć pliku: $zawartoscPliku"
102    }
103    return $nazwaPliku
104 }
105
106 proc nowyPlik {} {
107    global nazwaPliku
108    set nazwaPliku ""
109    .f.okno del 0.0 end
110 }
111
112 proc zapiszPlik zapiszJako {
113    global nazwaPliku
114    global koniecWiersza
115    if { $zapiszJako eq 1 } then {
116            set nazwaPliku [tk_getSaveFile -title "Zapisz plik"]
117    }
118    set zawartosc [.f.okno get 0.0 end]
119    if { [catch {set output [open $nazwaPliku w+]} wynik] eq 0} then {
120            if {$koniecWiersza eq 1} then {
121                    fconfigure $output -translation crlf
122            } else {
123                     fconfigure $output -translation lf
124            }
125            puts -nonewline $output $zawartosc
126            close $output
127    } else {
128            set filecontents ""
129            tk_messageBox -title "Błšd" -icon error \
130            -message "Nie można otworzyć pliku: $nazwaPliku"
131    }
132 }
133
134 # Pomoc
135 proc pomoc {} {
136    tk_messageBox -title "Pomoc"  \
136        -icon info -type ok \
137        -message "Pomoc jeszcze nie jest dostępna"
138 }
139 # Koniec pliku
```

Wykorzystanie programu gnuplot do tworzenia wykresów 2D

W artykule przedstawiono sposób tworzenia wykresów 2D za pomocą programu gnuplot. Opisano tylko te możliwości programu, które autorowi były potrzebne, w szczególności kreślenie wykresów na podstawie danych pomiarowych, zaznaczenie obszarów niepewności danych pomiarowych, interpolacje, przeliczenia wartości oraz dopasowanie wyglądu wykresu do swoich potrzeb.

TOMASZ ŁUCZAK

Krótko o programie

gnuplot jest darmowym programem do kreślenia funkcji i danych. Program dostępny jest dla większości systemów operacyjnych (MSDOS, MS Windows, Unix, Linux, OS/2, MacOS, BeOS, VMS) i platform sprzętowych (PC, Atari, Amiga, RISC). Program gnuplot jest także wykorzystywany przez inne programy (np. octave) jako tzw. backend do plotowania wykresów.

Ze względu na możliwość pracy wsadowej jest nieoceniony przy przetwarzaniu dużych ilości danych.

Praca z programem

Program sterowany jest z wiersza poleceń, ale dla uproszczenia można wczytywać z plików zewnętrznych (w formacie ASCII) dane oraz korzystać ze skryptów plotowania.

Domyślnym wyjściem wykresów jest ekran, ale można je przekierować do pliku graficznego.

Struktura plików danych

Struktura pliku z danymi do plotowania jest bardzo prosta. Dane są umieszczone w kolumnach i rozdzielone znakami odstępu (spacja lub tabulator). Domyślnie dane są pobierane począwszy od pierwszej kolumny, chyba że podamy inną kolejność pobierania danych z kolumn.

Poniżej przedstawiono przykładowy plik z danymi zebranymi z pomiarów. Znaki # stanowią komentarz i dalsza część wiersza począwszy od znaku # jest ignorowana. Kolejne kolumny przykładowego pliku danych zawierają: rzędne, odcięte, niepewność wartości rzędnych i niepewność wartości odciętych.

```
# To jest plik test.dat
# X      Y    errX    errY
0.000    0    0.0004 3
0.001    104  0.0004 4
0.002    202  0.0004 5
0.003    290  0.0005 5
0.004    291  0.0005 8
0.005    310  0.0005 6
0.010    311  0.0005 9
0.020    280  0.0005 5
```

Skrypty poleceń

Polecenia możemy zapisywać do plików ASCII i wczytywać je za pomocą polecenia load w wierszu poleceń gnuplota:

```
gnuplot> load "nazwa.skryptu"
```

Można również wywoływać gnuplota z nazwą skryptu:

```
gnuplot nazwa.skryptu
```

Kreślenie wykresów

Każdy wykres składa się z kilku elementów: tytułu, opisów osi, znaczników na osiach, legendy, komentarzy oraz właściwego wykresu. Powyższe elementy, oprócz samego wykresu, są definiowane za pomocą polecenia set. Sam wykres kreślony jest za pomocą polecenia plot.

Definiowanie elementów wykresu

Na początku zdefiniujmy elementy opisu wykresu.

Tytuł wykresu

Tytuł wykresu wyświetlany jest u góry rysunku i jest wyśrodkowany.

```
set title "Pierwszy wykres"
```

Opis osi

Opisy osi wyświetlane są na zewnątrz wykresu w środku osi.

```
set xlabel "U [V]"  # opis osi X
set ylabel "I [pA]" # opis osi Y
```

Legenda

Jeśli nie podamy żadnych poleceń dotyczących legendy, to zostanie ona umieszczona domyślnie w prawym górnym narożniku rysunku.

```
set key 0.015,100   # współrzędne legendy
set key box         # legenda w ramce
```

Legendę można również wyłączyć:

```
set nokey           # bez legendy na wykresie
```

Zakresy osi

Wykresy są domyślnie automatycznie skalowane, ale być może zechcemy podać

inne zakresy osi wykresów.

```
set xr [0.0:0.022]  # zakres
osi X
set yr [0:350]      # zakres
osi Y
```

Znaczniki na osiach

gnuplot sam dobiera znaczniki na osiach. Jeśli jednak chcemy ustawić je po swojemu to oczywiście możemy.

```
set xtics (0.002, 0.004,
0.006, 0.008, 0.010, 0.015,
0.020)
```

Znaczniki na osiach możemy wyłączać i włączać, w zależności od potrzeb.

```
set noxtics         #
wyłączenie znaczników
set xtics           #
włączenie znaczników
```

Komentarze na wykresie

Na wykresach możemy również umieszczać komentarze. Komentarze są pozycjonowane na wykresie przez podanie współrzędnych.

```
set label "Punkt zalamania" at
0.0035,260
```

Możemy również wskazać, do którego miejsca odnosi się komentarz przez narysowanie strzałki. Podczas rysowania strzałki należy podać współrzędne początku i końca (ze strzałką) strzałki.

```
set arrow from 0.0032,250 to
0.0031,280
```

Wykres możemy również opatrzyć tzw. „stopką" wstawioną w lewym dolnym narożniku, zawierającą np. datę utworzenia wykresu lub autora.

```
set timestamp "Utworzono dnia
%Y-%m-%d o godzinie %H:%M
przez TLu"
```

Zmienne %Y-%m-%d i %H:%M wyświetlą nam bieżącą datę i godzinę.

Plotowanie

Wydruku wykresu 2D dokonujemy za pomocą polecenia plot, które umożliwia wykreślenie również kilku wykresów na jednym rysunku.

Wykresy mogą być wydrukowane jako punkty z pliku danych albo mogą być interpolowane odcinkami lub krzywymi sklejanymi. Można także wygładzić wykres za pomocą krzywych beziera. Sposób plotowania jest ustawiany za pomocą dyrektyw polecenia plot.

Składnia polecenia plot:

```
plot [zakres] {funkcja|plik}
[osie] [tytuł] [styl] [, kolej-
ne polecenie]
```

gdzie:

`zakres`

podajemy zakres kolejno dla osi rzędnych (X), pierwszej funkcji i kolejnych. Wartości w nawiasach są separowane znakiem dwukropka :, np.:

```
plot [-3.14:3.14] sin(x)
funkcja|''plik danych''
```

podajemy funkcję do plotowania albo nazwę pliku. Nazwa pliku musi być zawarta w cudzysłowach.

`osie`

tutaj określamy, które osie mają być wykorzystane do opisu. Domyślnie wykorzystywana jest lewa (y1) i dolna (x1) oś.

`tytuł`

Dyrektywa przeznaczona jest do definicji nazwy wykresu, która pojawi się w legendzie. Tytuł należy umieścić w cudzysłowie i poprzedzić dyrektywą title lub skrótowo t.

`styl`

Styl określa sposób rysowania wykresu i zaznaczania punktów. Styl poprzedzony jest dyrektywą with. Dyrektywa with wymaga oddzielnego opisu.

`, kolejne polecenie`

po przecinku można umieścić polecenie plotowania kolejnego wykresu, np:

```
plot sin(x) , cos(x) , tan (x)
```

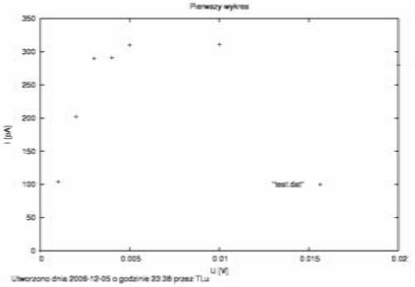
`Pierwszy wykres`

Wykorzystajmy wiedzę z poprzednich sekcji i napiszmy prosty skrypt:

```
# Plik wykres1, dane są pobierane z pli-
ku wykres.dat
set title "Pierwszy wykres" # tytuł
set xlabel "U [V]"          # opis osi X
set ylabel "I [pA]"         # opis osi Y
set key 0.015,100           # współrzęd-
ne legendy
set timestamp "Utworzono dnia %Y-%m-%d
o godzinie %H:%M przez TLu"
plot "test.dat"
```

Po uruchomieniu programu gnuplot piszemy:

```
gnuplot> load "wykres1"
```



Wykres utworzony przez skrypt wykres1 wygląda nieźle, ale nie odpowiada naszym wymaganiom. Chcemy aby punkty były połączone linią (odcinkiem, krzywą sklejaną) i były zaznaczone obszary niepewności pomiarów na wykresie.

Przeliczanie zawartości kolumn

```
# przeliczenie wartości przez stałą
($2*0.1)
plot "wykres.dat" using 1:($2*0.1) smooth
csplines title "Wykres"
```

Znaczniki wielkości błędu

gnuplot umożliwia nam naniesienie na wykres również słupków lub obrysów obszarów niepewności. By umiescić je na rysunku nasz plik z danymi danych musi zawierać kolumny zawierające te wartości.

Znaczniki można wyświetlać na trzy sposoby:

znaczniki dla rzędnych (Y),
znaczniki dla odciętych (X),
znaczniki dla rzędnych i odciętych.

Znacznik błędu można podajemy jako odchylenie od wartości lub w przypadku różnych wartości błędów in plus i in minus jako współrzędne końców obszarów odchyleń.

Ponadto znaczniki dla rzędnych i odciętych mogą być postaci słupków lub prostokątów.

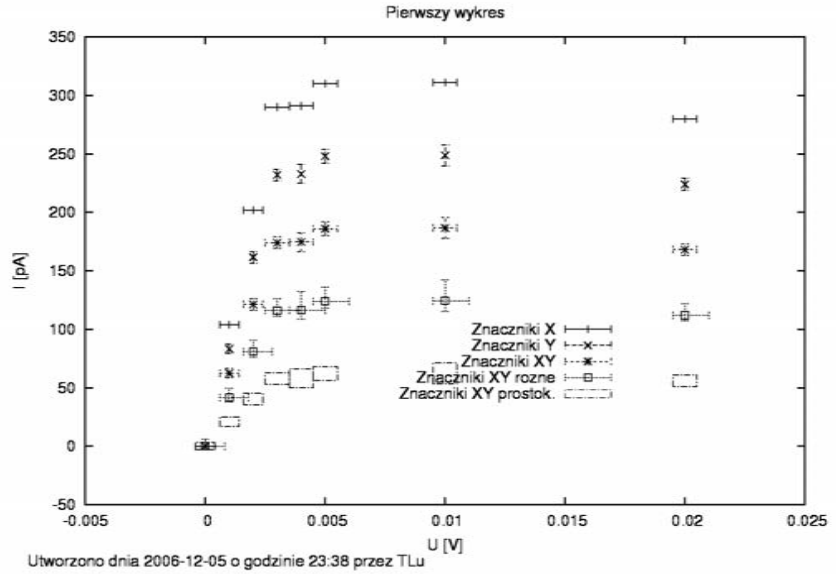
Wykres z rysunku [*] uzyskano za pomocą następującego polecenia:

```
plot "test.dat" using 1:2:3 title
"Znaczniki X" with xerrorbars , \
"test.dat" using 1:($2*0.8):4 title
"Znaczniki Y" with yerrorbars , \
"test.dat" using 1:($2*0.6):3:4 title
"Znaczniki XY" with xyerrorbars , \
"test.dat" using 1:($2*0.4):($1-
$3):($1+2*$3):(0.4*$2-$4):(0.4*$2+2*$4)
\
```

```
     title "Znaczniki XY rozne" with xy-
errorbars , \
 "test.dat" using 1:($2*0.2):3:4 \
     title "Znaczniki XY prostok."
with boxxyerrorbars
```

Tabela: Oczekiwany przez program gnuplot układ kolumn

Układ kolumn	dyrektywa
x, y, ydelta	(box)errrobars, (box)yerrorbars
x, y, ylow, yhigh	(box)errrobars, (box)yerrorbars
x, y, xdelta	(box)xerrrobars
x, y, xlow, xhigh	(box)xerrrobars
x, y, xdelta, ydelta	(box)xyerrrobars
x, y, xlow, xhigh, ylow, yhigh	(box)xyerrrobars

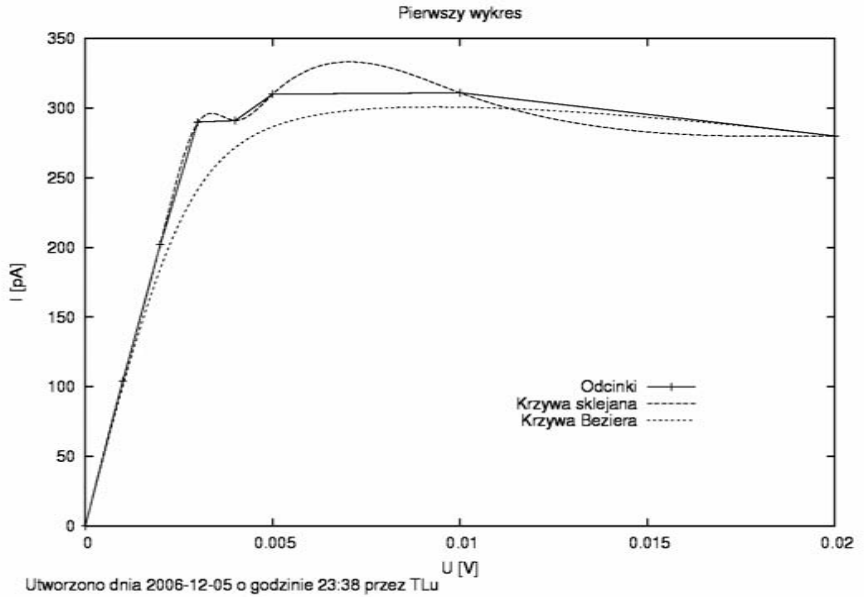


Za pomocą dyrektywy using podajemy jawnie, z których kolumn pliku danych korzystamy. Numery kolumn separujemy znakiem dwukropka :. Znak \ oznacza kontynuację bieżącego wiersza w wierszu następnym. Uwaga: po tym znaku nie może wystąpić spacja!

Należy zwrócić uwagę na liczbę kolumn wykorzystaną przy kreśleniu kolejnych wykresów.

Pierwszy wiersz zawiera polecenie plotowania zawartości pliku test.dat ze znacznikami błędów rzędnych. Do plotowania będą być wykorzystane w kolejności kolumna pierwsza, druga i trzecia.

Drugi wiersz zawiera polecenie plotowania ze znacznikami błędów odciętych. Do plotowania będą wykorzystane kolumny: pierwsza, druga przemnożona przez stałą 0.8 i kolumna czwarta pliku test.dat.



Trzeci wiersz zawiera polecenie plotowania ze znacznikami błędów rzędnych i odciętych. Do plotowania będą wykorzystane kolumny: pierwsza, druga przemnożona przez stałą 0.6, trzecia i czwarta kolumna pliku.

Czwarty i piąty wiersz zawiera polecenie plotowania ze znacznikami błędów rzędnych i odciętych o różnych wartościach niepewności in plus i in minus. Do plotowania będą użyte kolumny w kolejności: pierwsza, druga przemnożona przez stałą 0.4, pierwsza minus trzecia, pierwsza plus trzecia pomnożona przez 2, druga pomnożona razy 0.4 minus czwarta i druga pomnożona razy 0.4 plus czwarta kolumna pomnożona przez 2.

Szósty i siódmy wiersz nie wymaga już komentarza.

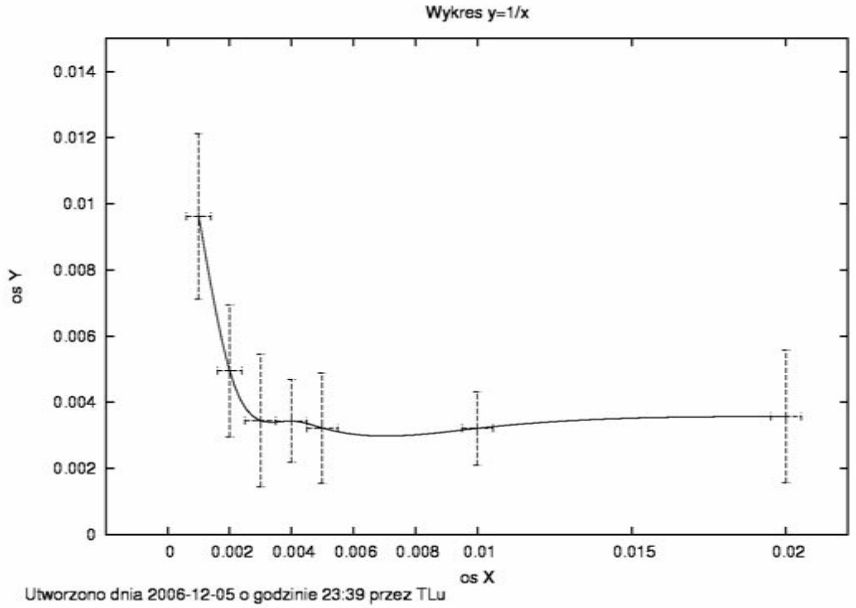
Wygładzanie wykresu

Wykres można interpolować za pomocą odcinków łączących kolejne punkty i krzywych sklejanych. Można również wygładzić wykres krzywymi Beziera.

Rysunek: Wygładzony wykres

Polecenie plotowania wygładzonego wykresu:

```
plot "test.dat" title "Odcin-
```




```
ki" with linespoints , \
"test.dat" smooth csplines ti-
tle "Krzywa sklejana" , \
"test.dat" smooth bezier title
"Krzywa Beziera"
```

Wydruk wykresu do pliku
Domyślnym terminalem wyjściowym jest x11 czyli wyjście na ekran. Jeśli chcemy otrzymać plik z rysunkiem należy zmienić terminal na jeden z poniższych:
x11 - ekran - domyślnie,
png - plik png,
latex - pliku LATEX-a,
postscript - plik Postscript,
za pomocą polecenia:

```
set terminal latex
```

Polecenie:

```
set terminal
```

wyświetla wszystkie dostępne terminale czyli możliwości zapisu do różnych formatów.
W przypadku niektórych terminali można podać dodatkowe opcje.
Należy jeszcze podać nazwę pliku wyjściowego, np.:

```
set output "wykres.png"
```

Pełny przykład
Narysujmy wykres funkcji y=1/x wygładzony krzywą sklejaną oraz z zaznaczonymi zakresami niepewności. Dane są identyczne jak podane wcześniej.

```
#set terminal x11        # wyjście na
ekran
#set terminal png        # wyjście do pli-
ku png
#set terminal latex      # wyjście do pli-
ku LaTeX
set terminal postscript # wyjście do pli-
ku PostScript, np. do druku
set output "test.ps"     # nazwa pliku
wyjściowego
set title "Wykres y=1/x"# tytuł
set xlabel "os X"        # opis osi X
set ylabel "os Y"        # opis osi Y
set xr [-0.002:0.022]    # zmiana zakre-
su osi X
set yr [0:0.015]         # zmiana zakre-
su osi Y
set nokey                # bez legendy
set timestamp "Utworzono dnia %Y-%m-%d
o godzinie %H:%M przez TLu"
set xtics (0,0.002, 0.004, 0.006,
0.008, 0.010, 0.015, 0.020)
plot "test.dat" using 1:(1/$2) smooth
csplines , \
 "test.dat" using 1:(1/$2):3:(0.01/$4)
with xyerrorbars
```


TeaM-TL

TeaM–TL, czyli
TEXLive w LinuxLive

TeaM–TL, to jedyna dystrybucja zawierająca kompletny TEXLive.

Wystarczy uruchomić komputer z płyty CD/DVD, a dostaniemy w pełni skonfigurowane środowisko do pracy z LATEXem wraz z dodatkowym nieTEXowym oprogramowaniem.

Już niebawem pojawi się nowa wersja oznaczona numerem **5.5.2**, a w niej między innymi:

- *kernel 2.6.15*
- *TEXLive 2005*
- *Emacs-21.4.2*
- *gVim-7.0.109*
- *Kile-1.9.1*
- *LyX-1.4.3*
- *ImageMagick-6.2.8.8*
- *Xfig-3.2.4*
- *FluxBox-1.0rc2*
- *AcrobatReader-7.0.8*
- *ispell-3.2.06*
- *Firefox 1.5.0.7*
- *Dillo-0.8.5*
- *aterm-1.0.0*
- *conky-1.4.2*
- *iDesk-0.8.0*
- *tablaunch-0.6*
- *torsmo-0.18*
- *xawtv-3.95*
- *xonclock-0.0.8.1*

W ramach portalu *TeaM-TL* dostępne jest forum dyskusyjne, wiki oraz system zgłaszania błędów flyspray.

Happy TEXing!

info@team-tl.livenet.pl
http://www.team-tl.livenet.pl

Apache cz.4

Kiedy włączymy naszą cudowną przeglądarke (jaka kolwiek by ona nie była), wpiszemy adres naszego ulubionego serwisu internetoweg (prawie dowolnego), naszym oczom zapewne ukaże się coś cudownego. Masa grafiki, wiele informacji, animacji, forularze i odnośniki. Jednym kliknięciem możemy sprawdzić jaka jest za oknem pogoda, jakie są wyniki wyborów, na którą mam jutro do szkoły lub poprostu przeczytać świetny dowcip. Kolejne kliknięce dostarcza nam jeszcze więcej informacji i zabawy. Jakie to wszystko piękne - ale czy to jeszcze httpd ??

RAFAŁ 'ert16' TRÓJNIAK

Dynamiczne generowanie stron

Kiedy projektowano httpd nikt jeszcze nie myślał o automatycznej generacji stron internetowych. Serwer HTTP miał za zadania w prosty, szybki i przejżysty sposób serwować pliki, poprostu przesyłać dane. Nikt wtedy nie myślał o modyfikowaniu części tych danych, a co dopiero ich tworzenia. Postęp technologiczny sprawił nam prawdziwego bzika. Aktualnie większość większych serwisów oparta jest o ogromne systemy generujące strony. Skrypty napisane w językach takich jak Perl, PHP, ASP, prekompilowane jak JAVA lub programy CGI. Protokół zapisany w RFC 2616 zapewnia doskonały pomost pomiędzy rozmaitymi aplikacjami a serwerem, niezależnie od skrytymi za nim mechanizmami. Dzięki temu mamy dzisiaj dostęp do gier poprzez przeglądarke, konfiguracje użądzeń sieciowych, informacji, poczty. Zyskujemy przenośność naszego biurka, możliwosć pracy z każdego zakatka globu w którym jest dostęp do internetu. Aktualnie całe systemy przenoszone są w sieć, włącznie z edytorami tekstu i arkuszami kalkulacyjnymi. Dlaczego więc i my nie sprubujemy swoich sił w tym wyścigu ??

mod_cgi(d)??

Prócz php często spotykane są jeszcze programy cgi. Do wykożystywania tej technologii wykożystywanych jest szereg modułow,takich jak mod_cgi, mod_cgid oraz suExec. Wykożystywanie programów CGI dostarczonych przez użytkowników jest na tyle niebezpieczne, iż udostępniamy mu o poziom więcej przestrzeni dowykożystania, niż przy języków skryptowych jak PHP. Jest to temat na osobny numer, więc będe zmuszony opisać go innym razem.

mod_php w skrócie

Instalacja mod_php od strony Apache wymaga jedynie kilku zmian. Chciałbym tutaj zaznaczyć, iż w tym opisie nie jestem w stanie omówić wszystkich zabiegów wymaganych przez aspekty bezpieczeństwa. Tego typu instalacja wystarczy w zupełności jako domowe-deweloperskie środowisko programisty w PHP. Dla systemów wieloużytkownikowych należy przemyśleć zasadność wielu operacji i ograniczyć jego prawa. wyłączyć niebezpieczne funkcje, ograniczyć dostęp do systemu plików.

Do rzeczy. Instalujemy paczke, przy czym należy uważać dla jakiej wersji serwera jest ona przeznaczona.(paczki z Apache v2 nie są kompatybilne z v1.3). Następnie musimy poinformować apache, aby załadował odpowiedni moduł poprzez umieszczenie w pliku konfiguracyjnym dyrektywy

```
LoadModule php5_module
        modules/libphp5.so
```

Kolejnym krokiem jest poinformowanie serwera które pliki ma poddawać przetwarzaniu przez moduł.

```
AddType application/x-httpd-
php .php .php3 .phtml
```

```
AddType application/x-httpd-
php-source .phps
```

Pierwsza dyrektywa informuje jakie rozszerzenia mieszczą się w kategorii MIME określajacej skryptóy php, natomiast druga źródła skryptów php. Dzięki tej informacji są one odpowiednio przetwarzane. Skrypty są wykonywane, a ich wyjście jest keirowane do klienta (najczęściej przeglądarki), natomiast źródła są przetwarzane do postaci stron HTML z kolorowaniem składni i przezentowane użytkownikom.

Na tym etapie po zrestartowaniu serwera powinien działać już interpreter PHP. Możemy to sprawdzić w bardzo prosty sposób. Tworzymy plik test.php o następującej treści:

```
<?php
phpinfo();
?>
```

Jeśli uprzednio nie zablokowaliśmy powyższej funkcji w pliku php.ini , po wybraniu adresu tego pliku w przeglądarce powinniśmy zobaczyć ekran z informacją o zainstalowanym module. Dowiemy sie tutaj o wszystkich wykrytych przez moduł rozszerzeniach, ich wersjach oraz opcjach. Funkcja phpinfo jest pierwszym kandydatem do zablokowania, chociarz większość udostępnianych przez nią informacji jest dostępna w innych miejscach.

Kolejną, bardziej kosmetyczna zmianą jest umożliwienie skryptom PHP występowanie jako indexy katalogów. Konfiguracja jest dokonywana przy pomocy

opisanej już dyrektywy DirectoryIndex.

```
DirectoryIndex index.php in-
dex.html index.html.var
```

Pamiętnik serwera

Apache, jak każdy prawdziwy demon, posiada wbudowany moduł pozwalający na zapisywanie wszystkich zapytań jakie obsługuje, oraz rozmaitych informacji na ich temat. Popularne Logi pozwalają na obserwacje wykożystania naszego serwera, obliczanie statystyk oraz (co chyba najbardziej boli) pomagają w znalezieniu sprawcy wyrządzonych szkód.

A co to ja miałem zapamiętać ?

Zawartość i format logów jest bardzo szeroko konfigurowalny. Możemy tutaj wpisywać wiele informacji, rozmieszczać je w dowolnej kolejności. Sam wygląd każdego rekordów ustalamy przez fromatujący ciąg znaków. Jest to sposób zapożyczony ze zbioru funkcji printf znanego z c++ i innych języków programowania. Każde pole zaczyna się znakiem procęta (%), po którym może nastąpić szereg parametrów, zakończony symbolem danego pola. Dla przykładu format :

```
"%!200r %200f"
```

pozwoli zalogować pierwszą linijke zapytania dla wszystkich nieudanych (status różny od 200) żądań, natomiast adres plik (flaga f) dla wszystkich udanych. Możemy użyć więcej takich opcji oddzielając je przecinkiem. Dla niektórych pul opcje podawane mogą być w nawiasach klamrowych, tak przykładowo :

Tabela pól

Opcja	Opis
%%	Znak procenta (Apache 2.0.44 i puźniejsze)
%...a	Adres IP klienta
%...A	Lokalny adres IP klienta
%...B	Rozmiar odpowiedzi w bajtach bez nagłówków http
%...b	j/w w formacie CLF. Dla zera wypisuje '-'
%...{Foobar}C	ciasteczko Foobar wysłane przez klienta
%...D	Czas poświęcony na zapytanie w milisekundach
%...{FOOBAR}e	Zarartość zmiennej FOOBAR
%...f	Nazwa pliku
%...h	Nazwa hosta klienta
%...H	Protokuł zapytania
%...{Foobar}i	Zawartość nagłówka Foobar wysłanego przez klienta
%...m	Metoda zapytania
%...{Foobar}o	Zawartość nagłówka Foobar w odpowiedzi.
%...p	Port na serwerze odbierający połączenie
%...P	PID procesu obsługującego zapytanie
%...{format}P	dla wartości pola format ▪ tid - numer wątku procesu ▪ pid - numer procesu obsługującego żądanie
%...q	Część zapytania występująca po "?" (dane przesyłane metodą GET)
%...r	Pierwsza linia zapytania
%...s	Status zapytania
%...t	Czas kiedy obsługiwano zapytanie.
%...{format}t	Czas kiedy obsługiwano zapytanie w formacie jak dla *strftime*.
%...T	Czas obsługi zapytania w sekundach
%...u	Zdalny użytkownik po uwieżytelnieniu przez moduł auth.
%...U	Żądany URL - bez danych GET
%...v	Nazwa wirtualnego serwera obsługującego żądanie
%...V	Nazwa wirtualnego serwera obsługującego żądanie
%...X	Status żądania po zakończeniu połączenia ▪ X - Przedwczesne zakończenie połączenia. ▪ + -Połączenie jest utrzymywane. ▪ - - Połączenie zamknięte.

```
"%{Referer}i"
```

pozwoli zapisać strony odnoszące się do danego pliku (nagłówek Referer). Dodatkowo wśród opcji może się pojawić specjalny znak "<" lub ">" określający część zapytania (początek lub koniec). Wykorzystywany np. w przypadku przekierowania klienta.

No tak, tylko gdzie ?

Znamy już sposób na ustalenie wyglądu każdego rekordu, nalezy się teraz zabrać za ich wykorzystanie. Logi mogą być wysyłane do plików, lub do programu przetwarzającego i zapisującego te informacje. Aby zapisać dane do pliku wystarczy podać ścieżkę pliku, znajdującą się w ServerRoot. Używanie programu do zapisywania plików jest możliwe, przez podanie jego nazwy poprzedzonego pionową kresą ("|" - pipe ". Program ten będzie uruchamiany przez serwer na tych samych prawach, oraz karmiony przez standardowe wejście danych.

Do konfiguracji tych czynnosci posłużą nam 3 dyrektywy:

* Dyrektywa LogFormat przyjmuje 2 argumenty. Definiuje ona sposób zapisywania logów, przez nadanie mu nazwy zarartej w drugim argumencie. Pierwsą opcją jest format rekordu (definiowany ciągmiem znaków w cudzysłowie) lub nazwa już zdefiniowanego formatu.

```
LogFormat "%h %l %u %t \"%r\"
%>s %b" common
LogFormat common common_copy
```

* Dyrektywa CustomLog przyjmuje 3 argumenty. Pierwszym z nich jest miejsce umieszczenia danych. Drugą opcja jest nazwa formatu wpisów, lub sam ciąg formatujący.

```
 CustomLog tajne/logi/acces-
s_log "%h %l %u %t \"%r\" %>s
%b"
 CustomLog logs/access_log com-
mon
```

* Dyrektywa TransferLog działa podobnie do CustomLog. Przyjmuje ona jeden argument będący miejscem przechowywania logów. Za format logów przyjmuje ostatni zdefiniowany dyrektywą LogFormat, ale nie nazwany format.

```
LogFormat "%h %l %u %t \"%r\"
%>s %b \"%{Referer}i\"
\"%{User-agent}i\""
TransferLog logs/access_log
```

Inne zapiski

Prucz zwykłych informacji o dostępie do plików rejestrujemy również błędy. Do umiejscowienie tego typu informacji służy dyrektywa ErrorLog przyjmujące jeden argument. Dodatkowo możemy tutaj również użyć dziennika systemowego syslog podając jako parametr syslog:TYP , gdzie TYP jest grupą do jakiej zostaną zakfalifikowane informacje od serwera.Możemy włączyć buforowanie logów poprzez dyrektywe BufferedLogs przyjmującą argument w postaci off lub on. Możemy również zbierać informacje o ciasteczkach zapisując je do pliku dzięki dyrektywie CookieLog.

Apache pozwala na wszechstronne zapisywanie swoich poczynań, co daje nam wiele kożyści. Należy pamiętać, że zbieranie takich informacji jest obowiązkiem każdego administratora.

NetBeans

http://www.netbeans.org

Podpis cyfrowy, czemu warto używać?

Jesteśmy w dobie powszechnej informatyzacji społeczeństwa - to już truizm, powtarzany na każdym kroku. Komputery wdzierają się do wszelkich możliwych dziedzin życia. Podobnie jak w kontaktach międzyludzkich, także w Internecie istnieje potrzeba uwiarygodnienia swojej tożsamości, tak aby korespondent miał pewność, że komunikuje się z właściwą osobą.

MACIEJ RUTECKI (unixy.pl)

O ile przyzwyczailiśmy się do bezpiecznych stron (szyfrowanych) bankowych, pocztowych, to jakoś ciągle jest problem ze zrozumieniem potrzeby podpisywania swoich wiadomości podpisem cyfrowym. A szkoda, zwiększa to bowiem zaufanie do nas, kiedy go używamy (niezależnie od wiedzy technicznej drugiej strony).

Tutaj należy wspomnieć różnicę pomiędzy podpisem cyfrowym a elektronicznym. O ile pierwszy jest ściśle związany z komputerami i technologiami informatycznymi, to podpis elektroniczny jest bardziej ogólnym pojęciem, obejmującym zarówno podpis cyfrowy, jak i wszelkie inne (marginalne dziś) elektroniczne sposoby poświadczania autentyczności danych [6].

Zasadniczym problemem w naszym kraju jest topornie wprowadzana informatyzacja, wszystko wskazuje na to, że przed 2008 rokiem nie mamy co liczyć na masowe wykorzystanie podpisu elektronicznego w polskiej administracji. Z drugiej strony osoby często komunikujące się z zachodnimi firmami mogą już mieć potrzebę jego używania. W tym artykule zamierzam się skupić nad zaletami podpisu elektronicznego w przypadku zwykłych użytkowników.

Nie będę zajmował się technicznymi aspektami, gdyż nie wniosą nic praktycznego do artykułu. Sam sposób otrzymania certyfikatu został opublikowany już wcześniej. Moim celem jest zachęcenie do szerszego wykorzystania podpisu cyfrowego w komunikacji elektronicznej.

Dlaczego?

Osoby, które często korzystają z takiego medium jakim jest Internet i poczta elektroniczna wcześniej czy później padły ofiarą spamerów - ludzi masowo wysyłających różnego rodzaju reklamy przez sieć. Ponieważ taki proceder jest często na granicy legalności bądź jednoznacznie sprzeczny z prawem, to autorzy reklam często podszywają się pod obce osoby. Raz miałem przyjemność dostać spam ze sfałszowanymi nagłówkami wiadomości, gdzie jako nadawca widniał mój adres e-mail...

Rodzi się pytanie; jak zapewnić innych o autentyczności swojego listu? Jak zabezpieczyć dokument przed niepowołanymi zmianami? W listach "analogowych" (papierowych) normalną rzeczą jest podpisywanie się na końcu tekstu. Czemu tego nie przenieść do korespondencji elektronicznej? Podpis elektroniczny, aby go wykorzystać w praktyce, musi być:

- przypisany jednej osobie,
- niemożliwy do pod-

robienia (w realnym czasie),
- uniemożliwiający wyparcie się go przez autora,
- łatwy do weryfikacji przez osobę niezależną,
- łatwy do wygenerowania.

Różni się jednak od podpisu tradycyjnego kilkoma właściwościami:

- może być składowany i przesyłany niezależnie od dokumentu (niemożliwe w podpisie tradycyjnym),
- jest funkcją dokumentu (czyli zmiana dokumentu unieważnia podpis elektroniczny, zapewnienie o autentyczności dokumentu jest niekiedy ważniejsze od autentyczności nadawcy),
- obejmuje cały dokument.

Rozwiązania

Najbardziej popularnymi technikami podpisywania wiadomości są PGP (np. OpenPGP) [4] oraz certyfikaty X.509 [3]. Nie chcę wdawać się w dyskusję wyższości jednego nad drugim i tylko krótko je scharakteryzuję w podpunktach.

OpenPGP:
- niezależny od dużych firm,
- zdecentralizowana baza podpisów, działa na zasadzie zaufanej sieci [2]: uczestnicy wzajemnie podpisują sobie klucze, istnieje ryzyko, że któryś z kluczy nie będzie zaufany,
- bezpłatny.

X.509:
- hierarchiczny sposób nadawania certyfikatów, istnieją główne ośrodki certyfikacji, swojego czasu certyfikat wymagał opłaty, obecnie można uzyskać go bezpłatnie,
- obsługuje go zdecydowana większość programów pocztowych.

Wygodniejszy jest w obsłudze X.509 i ten polecam czytelnikom. Obydwa rozwiązania istnieją na większość dostępnych platform i systemów operacyjnych (OpenPGP także pod DOSa) i są "wolne" technologicznie.

Czy podpis elektroniczny jest dla mnie?

Zalety podpisu cyfrowego powinni dostrzec właściciele firm rozsyłających dokumenty do współpracowników, kontrahentów czy klientów. Obecność podpisu - w takim przypadku - podnosi znacząco wiarygodność adresata (już nie może się wyprzeć wysłania dokumentu!).

A zwykli użytkownicy? Od dnia 7 sierpnia 2006 r. Narodowy Fundusz Zdrowia umożliwia przesyłanie korespondencji i dokumentów podpisanych elektronicznie. Powyższe dokumenty powinny być kierowane na adres kancelaria.elektroniczna@nfz.gov.pl. Przyjmowane będą tylko dokumenty sygnowane podpisem cyfrowym (certyfikatem elektronicznym). "Klienci" ZUS często już mają (niekiedy nieświadomie) do czynienia z podpisem cyfrowym.

Kupujesz coś w serwisie aukcyjnym lub sklepie internetowym? Nie ma lepszego potwierdzenia transakcji jak zaświadczenie autentyczności listu za pomocą podpisu cyfrowego. Na zachodzie Europy niektóre duże koncerny przy większych zakupach ignorują niepodpisane listy!

Ponadto możesz łatwo szyfrować listy. Wystarczy, że nadawca ma certyfikat, prześle do nas list podpisany tym certyfikatem i możemy szyfrować listy wysyłane do niego.

Microsoft i inne firmy chciały wprowadzić nowe metody uwierzytelniania nadawcy, zapominając zupełnie, że ich rozwiązanie uniemożliwiłoby korzystanie z list dyskusyjnych, aliasów pocztowych itp. A przecież mamy podpis cyfrowy, który łatwo dostać i używać.

Pod nasz adres pocztowy może podszyć się nie tylko spamer, ale także wirus czy trojan, czy złośliwy "kolega". Otrzymując podpisany list mamy pewność, że nie został zmieniony i jednocześnie nadawca nie może się wyprzeć jego wysłania. Zobacz [9] jak łatwo podszyć się pod inną osobę. ■

Przydatne adresy

[1] http://pl.wikipedia.org/wiki/Podpis_cyfrowy
[2] http://pl.wikipedia.org/wiki/Sie%C4%87_zaufania
[3] http://pl.wikipedia.org/wiki/X.509
[4] http://pl.wikipedia.org/wiki/PGP_%28program_kryptograficzny%29
[5] http://ipsec.pl/podpis_elektroniczny/
[6] http://www.republika.pl/podpis_elektroniczny/
[7] http://ipsec.pl/pk/index.php?title=Faktura_elektroniczna
[8] http://spam.jogger.pl/tag/podpis-cyfrowy
[9] http://spam.jogger.pl/2006/04/10/e-mail-nie-mozna-mu-ufac/

Thawte - jak dostać darmowy certyfikat niekwalifikowany

Certyfikat niekwalifikowany jest wygodnym narzędziem do podpisywania wiadomości elektronicznych. Problemem jest to, że w Polsce jest płatny (25PLN w promocji...). Ale Thawte.com znany ośrodek certyfikacji pozwala uzyskać go za darmo.

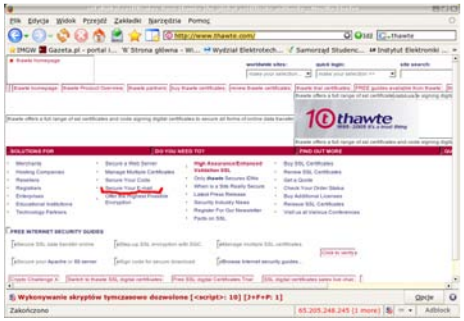
MACIEK RUTECKI (unixy.pl)

Co prawda w naszym kraju wykorzystanie podpisu elektronicznego opóźni się o dwa lata, to moim zdaniem warto taki już posiadać, szczególnie jak ktoś często prowadzi korespodencję z zagranicą. Ponadto obecność podpisu elektronicznego zwiększa zaufanie do naszej osoby (nawet jak adresat kompletnie nie zna się na tym zagadnieniu).

Niniejszy artykuł opisuje stronę techniczną pozyskania certyfikatu. Zainteresowanych zachęcam do zajrzenia na stronę Tomasza Andrzeja Nideckiego: http://spam.jogger.pl/tag/podpis-cyfrowy.

A teraz jak zdobyć certyfikat.

Wchodzimy na stronę główną http://www.thawte.com/ i wybieramy Secure Your E-mail



wybieramy Click here:

Pojawi się nowe okno, w którym musimy zaakceptować warunki umowy radzę przeczytać uważnie całą treść, aby później nie było niedomówień. Wymagany wiek do stworzenia certyfikatu to 13 lat (niepełnoletni za zgodą rodziców/opiekunów). Ponieważ nasz certyfikat jest naszą wizytówką (praktycznie jak dowód tożsamości), zatem nie powinniśmy go zgubić. Zgadzamy się jednocześnie na publikację informacji o nas i certyfikacie. Z resztą inaczej trudno będzie nas powiązać z danym certyfikatem.

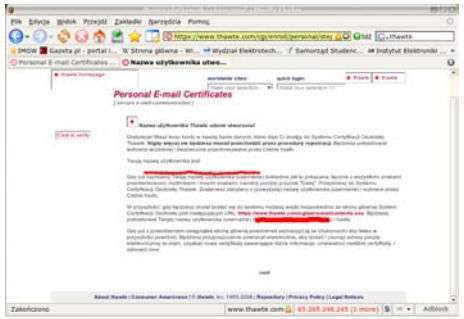
Jeśli zgadzamy się z warunkami naciskamy next na samym dole.

W następnym oknie, jeśli mamy poprawnie skonfigurowaną przeglądarkę, powitani zostaniemy komunikatami po polsku. Tutaj warto wpisywać prawdziwe dane osobowe, inaczje certyfikat będzie bezużyteczny. Radzę dwukrotnie sprawdzić wprowadzany tekst.

W kolejnym oknie poproszeni zostajemy o adres e-mail, który jest jednocześnie nazwą użytkownika w systemie Thawte. Następnie potwierdzamy stronę kodową i kraj.

W kolejnym oknie musimy wymyśleć solidne hasło, musi być mocne, w przeciwnym przypadku grozi to włamaniem i kompromitacją naszego konta i certyfikatu! Potem odpowiadamy na kilka pytań, weryfikujemy dane i czekamy na list z potwierdzeniem tego etapu rejestracji.

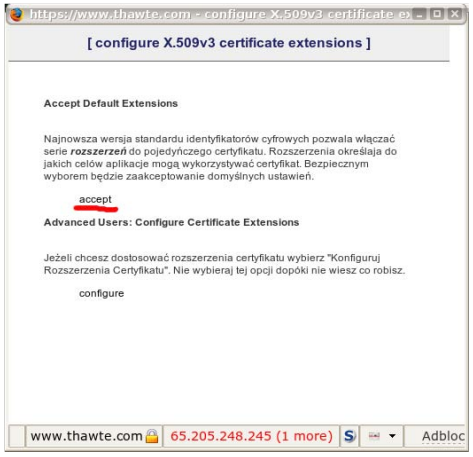
Wchodzimy na adres podany w liście. Kopiujemy dane z listu, które nakazuje nam Thawte. Po wpisaniu otrzymamy coś podobnego jak poniżej:



Naciskamy next i logujemy sie na stronie.

Wybieramy certyfikat X.509. W nowym oknie wybieramy przeglądarkę, jakiej używamy i naciskamy request:

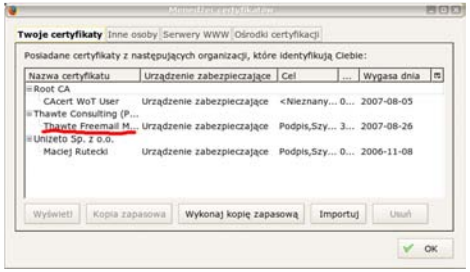
Przechodzimy do następnego okna wybieramy nasz adres e-mail. 2x next. Wybrać Accept Default Extensions -> Accept.

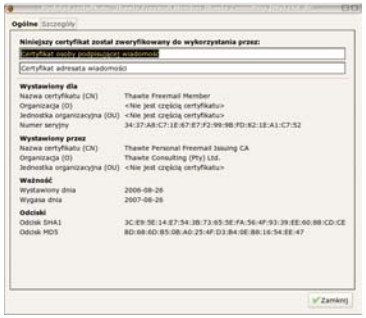


Wybieramy wysoki poziom zabezpieczeń (2048 bitów) -> next i czekamy aż wygeneruje się certyfikat.

W kolejnym oknie upewniamy się, czy nie zrobiliśmy jakiegoś błędu i naciskamy finish. Teraz czekamy, aż otrzymamy potwierdzenie. Najpierw otrzymamy e-mail informujący o złożeniu zamówienia. Dopiero później o dostępności certyfikatu.

W liście pod tytułem: Thawte Personal Cert Issuedotrzymasz adres, który musisz wpisać w przeglądarce (taki długi z dużą ilością cyfr na końcu). Po wejściu powinien zainstalować się automagicznie certyfikat. Użytkownicy Firefoxa i pokrewnych nie zobaczą żadnego komunikatu, ale wystarczy zajrzeć w Preferencje->Zaawansowane->Bezpieczeństwo->Wyświetl certyfikaty->Twoje certyfikaty i powinniśmy zobaczyć

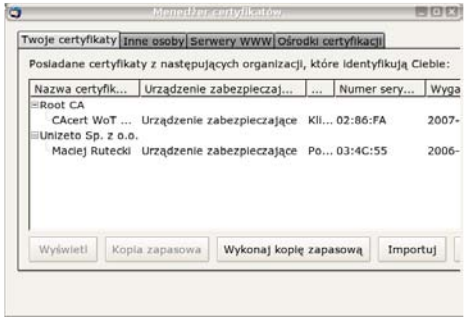




nowy certyfikat:
Zaznaczamy nasz i wybieramy wyświetl:
Teraz jak przepisać certyfikat z Firefoxa do Thunderbirda.

Zaznaczamy nasz certyfikat i wybieramy Wykonaj kopię zapasową, wybieramy hasło (musi być mocne!) i zapisujemy.

Uruchamiamy Thunderbirda: Preferencje-



>Prywatność->Zabezpieczenia->Wyświetl certyfikaty->Importuj:
Wybieramy nasz plik z certyfikatem, podajemy wcześniej zapisane hasło i po otwarciu KASUJEMY PLIK NA DYSKU Z ZAPISANYM CERTYFIKATEM!

Teraz w Thunderbirdze wybieramy Konfigurację kont->nasze konto->Zabezpieczenia. W Podpis cyfrowy i szyfrowanie poczty wybieramy nasz certyfikat. Warto zaznaczyć domyślne podpisywanie cyfrowe wiadomości.

Od tej chwili możemy testować działanie certyfikatu.

Certyfikat cyfrowy jest ważny tylko przez rok, potem trzeba go odnowić.

PLD Linux Distribution

http://pld-linux.org/

Kilka upiększeń dla KDE

KDE jest wysoce konfigurowalnym środowiskiem graficznym. Za pomocą Centrum Sterowania można zrekonfigurować niemalże każdy element systemu. Jednakże KDE zaczyna błyszczeć dopiero po doinstalowaniu dodatkowych dedykowanych programików. Chciałbym opisać kilka z nich.

KAROL "AXIO" KOZIOŁ

Do testów wybrałem szczęśliwą ósemkę: Superkaramba, Kxdocker, Kooldock, Ksmoothdock, Kompose, Taskbar2, Kbfx, Yakuake.

Part 1. Superkaramba, czyli jak dobajerzyć pulpit.

Superkaramba, czyli ulepszona następczyni Karamby, jest aplikacją potrafiącą dodać do kde-owskiego pulpitu różne aplety: monitory badające obciążenie systemu, dodatkowe paski z ikonami, kalendarze i zagary, czytnik newsów albo pogodynka. Ciekawe są aplety potrafiące współpracować z zewnętrznymi programami: XMMS czy Amarokiem. Dodatkowo tworzenie własnych tematów Karamby jest tak proste, że każda osoba z odpowiednim zacięciem powinna sobie poradzić (oczywiście nie sztuka zrobić jakiś temat, sztuka to zrobić dobry schemat). Dlatego też na początek pokaże kod mojego tematu karamby (prosty aplet bez żadnych graficzek, który ma po prostu wyświetlać kilka informacji o systemie). Tekst kodu jest oznaczony `taką czcionką`, moje komentarze objaśniające (normalnie nie ma ich w kodzie) są oznaczone *taką czcionką*.

```
karamba x=0 y=0 w=860 h=50 loc-
ked=true interval=1000
```

x,y to odległości apletu w pikselach mierzone od lewego górnego brzegu ekranu; w,h to odpowiednio szerokość i wysokość apletu; locked=true/false czyli czy aplet ma być zablokowany czy taż jest możliwość przesuwania go za pomocą myszki; interval definiuje czas odświeżania tematu (podawany w milisekundach)

```
#CPU
text x=15 y=10 value="CPU:" co-
lor=0,0,0 fontsize=13
font="Luxi Sans"
```

zmienna tekst określa że będziemy definiować obiekt tekstowy, przy czym może to być zarówno samodzielny tekst, jak i jakiś sensor (w tym wypadku jest to po prostu samodzielny tekst); oprócz zmiennej tekst możliwe są też (aczkolwiek nie stosowane tutaj przeze mnie) image (wstawia obraz), bar (sensor w kształcie miernika analogowego) oraz graph (sensor w kształcie wykresu); x,y określają położenie obiektu liczone od lewego górnego brzegu apletu; value określa treść wypisywanego tekstu; color określa oczywiście kolor tekstu, należy wpisywać kolory w postaci RGB, w tym przypadku 0,0,0 to czarny; fontsize to rozmiar użytej czcionki, zaś font określa jaka czcionka zostanie użyta do wyświetlenia tekstu

```
text x=55 y=10 sensor=CPU for-
mat="%v%" color=0,0,0 align=ri-
ght fontsize=13 font="Luxi
Sans"
```

teraz dodany pierwszy sensor systemowy; sensor=CPU oznacza że będzie to sensor mierzący zużycie procesora (inne sensory użyte poniżej to memory, disk, uptime, network); sensor będzie tekstowy i wyrównany w prawo (zmienna align=right); parametry czcionki zdefiniowane jak powyżej; format ma określać w jaki sposób sensor będzie wyświetlać dane; %v określa ze będzie to po prostu wartość obciążenia procesora (znaczek % określa że te 'v' to nie zwykła literka), natomiast dodatkowy '%' wstawia znak procenta po wyświetlanej wartości

```
#Memory
text x=105 y=10 value="Ram:"
color=0,0,0 fontsize=13
font="Luxi Sans"
text x=145 y=10 sensor=memory
format="%um of %tm MB" co-
lor=0,0,0 fontsize=13
font="Luxi Sans"
```

teraz będzie sensor mierzący ilość zajętej pamięci operacyjnej (sensor=memory); kilka słów poświęcę formatowi danych; powyższy zapis oznacza że będzie wyświetlana wartość zużytej pamięci (%um to used memory) oraz wartość całkowitej ilości pamięci (%tm czyli total memory) rozdzielone spójnikiem of; czyli na przykład 333 of 768 MB; aha, wolna pamięć to %fm (free memory)

```
text x=235 y=10 value="Swap:"
color=0,0,0 fontsize=13
font="Luxi Sans"
text x=280 y=10 sensor=memory
format="%us of %ts MB" co-
lor=0,0,0 fontsize=13
font="Luxi Sans"
```

tak jak wyżej tylko teraz mierzona jest ilość pamięci swap; formaty: %us to używany swap (used swap), %fs to niezajęty swap (free swap) natomiast %ts to całkowita ilość pamięci swap (total swap)

```
#Disk
text x=380 y=10 value="HDD:"
color=0,0,0 fontsize=13
font="Luxi Sans"
text x=420 y=10 sensor=DISK
 interval="10000" format="%u
of %t Mb (/)" color=0,0,0 font-
size=13 font="Luxi Sans" mount-
point="/"
```

następnie idzie sensor mierzący zajętość dysku (sensor=disk); nieużywana wcześniej w sensorach zmienna interval określa jak często dany sensor ma sprawdzać to co ma sprawdzać (o ile obciążenie CPU można sprawdzać co sekundę to dla zajętości dysku to nie ma większego sensu, więc lepiej wydłużyć ten okres); formaty: %u to zajęta przestrzeń dyskowa (used) zaś %t to całkowita pojemność dysku (a %f to pewnie wolna przestrzeń); i najważniejsza zmienna – mountpoint określa którą z podmontowanych partycji sensor ma sprawdzać (podajemy punkt montowania zdefiniowany w /etc/fstab)

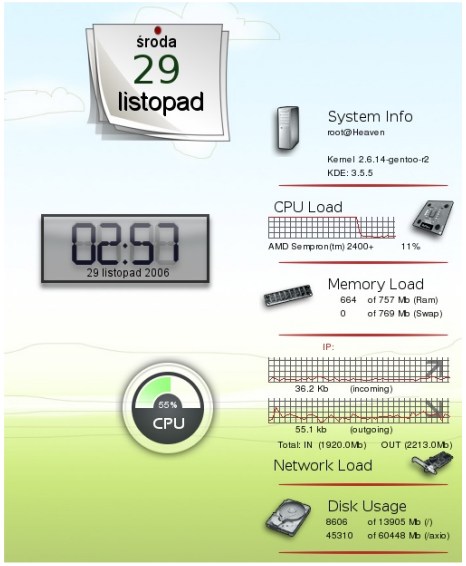
```
text x=550 y=10 sensor=DISK  in-
terval="10000" format="%u of
%t Mb (/axio)" color=0,0,0 font-
size=13 font="Luxi Sans" mount-
point="/axio"

#Uptime
text x=720 y=10 value="Upti-
me:" color=0,0,0 fontsize=13
font="Luxi Sans"
text x=775 y=10 sensor=uptime
format="%dd %H:%M" color=0,0,0
fontsize=13 font="Luxi Sans"
```

a tutaj zdefiniowałem sensor mierzący uptime systemu, czyli czas w jakim komputer pozostał bez restartu (sensor=uptime); formaty: %dd to dni %H to godziny, natomiast %M to minuty (chyba nie ma sensu mierzyć uptime w sekundach, natomiast miesiące czy lata to jednak dużo, no chyba że ktoś administruje stabilnym serwerem)

```
#Net
#in
text  x=110 y=30 sen-
sor=NETWORK color=0,0,0 for-
mat="%in kb" fontsize=13
font="Luxi Sans" DECIMALS=1
```

sensor=network czyli mierzymy obciążenie sieci internetowej; format %in określa natężenie pakietów przychodzących (incoming), natomiast niżej zdefiniowany %out określa natężenie pakietów wychodzących (outcoming) z naszego komputera, mierzonych w kilobajtach; zmienna decimal określa z jaka dokładnością (a dokładnie ile miejsc po przecinku) mają być podawane dane

```
text  x=15 y=30 value="Net (in-
coming)" color=0,0,0 fontsi-
ze=13 font="Luxi Sans"
#out
text  x=280 y=30 sen-
sor=NETWORK color=0,0,0 for-
mat="%out kb" fontsize=13
font="Luxi Sans" DECIMALS=1
```



Nagromadzenie tematów superkaramby

```
text  x=185 y=30 value="Net
(outgoing)" color=0,0,0 fontsi-
ze=13 font="Luxi Sans"
```

i to już koniec :-)

No, to jak już wiadomo jak napisać własny aplet to do roboty! Warto przy tym podglądać źródła innych, porządnie zrobionych apletów, żeby zobaczyć jak daną rzecz można zrobić (wbrew pozorom to powszechna metoda doskonalenia umiejętności programistycznych, rzekłbym nawet że najlepsza).
Strona domowa projektu: [1], zaś pod [2] jest kilka porad jak zrobić własny temat.

Part 2. Kxdocker, Kooldock, Ksmoothdock - czyli dock-panele inne niż wszystkie.

Powyższa trójka ma wspólną cechę - potrafi zastąpić standardowy pasek dokowania programów (tak zwane szybkie uruchomianie) na ładniejszy, z kilkoma cieszącymi oko efektami. Zazwyczaj są to powiększające się ikonki, czasami z różnymi ciekawymi przejściami. Paski sa półprzeźroczyste i maja fajne obwódki, lub też konfigurowalne tło. Po najechaniu kursorem na ikonkę pojawiają się "dymki" z nazwą aplikacji. Takie aplikacje stały sie popularne po tym jak Apple wprowadził pasek szybkiego uruchomiania w MacOSX. Najbardziej wzorowanym na nim jest chyba KXDocker. Posiada mnóstwo tematów do znalezienia w sieci (np. duży zbiór na stronie domowej projektu), do wyboru do koloru (mi najbardziej pasuje niebieski). Oprócz możliwości zgromadzenia w jednym miejscu najczęściej używanych aplikacji posiada także funkcyjne odpowiedniki paska zadań (aczkolwiek aplikacje wyświetlane są w postaci ikon) i tacki systemowej. Efekty powiększania ikon pod kursorem są dość ładne (trudno opisać efekty dynamiczne słowami więc uwierzcie mi na słowo). Tylko dlaczego to zużywa tyle procesora?!! Podczas bawienia się aplikacją (co polegało na suwaniu kursorem w tą i we wtą aby sprowokować efekty ikonek) chwilowe zużycie procesora sięgało, tu uwaga, nawet do 40-60%. Nawet w spoczynku program zajmował kilka procent CPU, natomiast uruchamianie aplikacji z poziomu KXDockera zajmowało zauważalnie więcej czasu niż np. z menu start. Tak wysokie użycie procesora jest według mnie niedopuszczalne (nawet jeżeli tylko przez 1-2 sekundy). Nie wiem, zawsze mogę powiedzieć że miałem jakąś szczególnie wybrakowaną wersję, ale uznaje to za błąd. Drugi mój zarzut przeciwko tej aplikacji jest taki, że dodawanie nowej pozycji na pasku jest trochę pokomplikowane. Według mnie powinny być tylko 3 opcje: widzialna nazwa programu, ścieżka uruchamialna i rodzaj wyświetlanej ikony (i tak np. jest w KsmoothDock). Tymczasem mamy jeszcze kilka opcji dodatkowych do wypełnienia, i na dodatek niezbyt jasno nazwanych. Ok, teraz przejdźmy do drugiego programu, który w końcu okazał sie moim faworytem. KsmoothDock jest może nieco mniej efektywny od poprzednio opisywanego programu, ale za to łatwy, prosty i przyjemny w użyciu. Posiada 2 tryby animacji ikon (znane normal i parabolic), jednak z czego ten drugi (niedomyślny) trochę zżera procesora. Dodatkowo program posiada funkcje taskbara,

systraya i zegara. Jak napisałem wyżej dodawanie nowych pozycji do paska jest dość proste. Można ustawić poziom przeźroczystości tak aby pasował do reszty wystroju pulpitu. Delikatna obwódka oraz wyskakujące opisy pozycji menu dopełniają całości. Ogólnie uważam za najlepszy z trójki bo łączy prostotę, funkcjonalność i estetykę. Dla formalności opiszę jeszcze ostatni program: Kooldock. Chyba najmniej dopracowany. Animacja okien wypada jakoś blado. Dodatkowo program ma problem z utrzymaniem i skalowaniem rozmiaru paska podczas powiększania ikonek, co wygląda po prostu brzydko. Programu mało używałem, więc powiem tylko że w ogólności wygląda w porządku, pod warunkiem że nie najedzie się kursorem, bo wtedy ładny efekt się rozmywa. Strony domowe projektów to odpowiednio [3], [4], [5].

Ksmoothdock

Part 3. Komposé, czyli jak efektywnie i efektownie zarządzać mnogością pulpitów.

Kompose stanowi ciekawe podejście do tematu zarządzania wirtualnymi pulpitami. Wzorowane na macowskim Exposé (gdyby Apple porządnie opatentowało swoje pomysły to gdzie byśmy byli? No gdzie?!). A owo podejście polega na tym, by w trakcie przełączania się między desktopami wejść w fullscreen i posegregować otwarte okna (obrazowane w postaci małych zrzucików okien) według pulpitu. Tak w największym skrócie. Zrzut ekranu powinien wszystko wyjaśnić – otóż na pulpicie pierwszym widnieją dwie aplikacje, na drugim jedna, na trzecim również dwie, natomiast czwarty pulpit jest pusty i widniej na dole.

Tak odbywa się przeglądanie pulpitów roboczych w Komposé

Przełączać się można zarówno na konkretny pulpit jak i do konkretnej aplikacji (można też z poziomu widoku pełnoekranowego zamknąć lub zminimalizować aplikację, lub też przerzucić ją na inny pulpit). Nie dość, że ładnie to wygląda, to i jeszcze przyjemne i wygodne w obsłudze. Mamy podgląd wszystkich okienek na wszystkich pulpitach naraz, i na dodatek jeszcze widzimy co każde robi. Wadą takich dobajerzonych rozwiązań jest to, że zwykłe przełączenie się na inny pulpit trwa dłużej niż skorzystanie ze standardowej przeglądarki pulpitów KDE. Czasami też Komposé nie zdąży odświeżyć obrazu okna, gdy zbyt często pełnoekranowy podgląd. Oczywiście można zmieniać częstotliwość odświeżania obrazów okien, ale ustawianie zbyt niskich wartości nie ma sensu bo wtedy wzrasta obciążenie procesora. Ogólnie programik jest bardzo fajny, i gdyby nie fakt że przeglądanie pulpitów w ten sposób trwa dla mnie nieco zbyt długo (sekunda do sekundy a zbierze się minuta), to bym go stale używał. Strona domowa projektu to [6].

Part 4. Taskbar 2, czyli jak można ulepszyć pasek zadań.

Standardowy pasek zadań KDE dobrze spełnia swoje zadanie, ale to nie znaczy że można go ulepszyć. I takim ulepszeniem jest Taskbar v2. Najważniejszym fiuczerem jest możliwość dołączania do programu widocznego na pasku jego mini-zrzutu okna. Czyli można podejrzeć co robiła zminimalizowana przez nas aplikacja. Taskbar 2 obsługuje przeźroczystość (co umożliwia stworzenie w pełni przeźroczystego paska), chociaż nie da się jej regulować. Innym ulepszeniem jest ustawienie wyglądu przycisków aplikacji tak by po najechaniu kursorem wydawa-

Okienko konfiguracji KBFX

ły sie wklęsłe. Plus jeszcze kilka małych lecz polepszających wygląd udoskonaleń. Programik jest mały, ale mi się bardzo podoba. Fajne są te małe screenshoty okien, chociaż w ogólności są one mało przydatne, bo aplikację na pasku rozpoznaję głównie po nazwie. Ale efekt jest. Strona domowa projektu to [7].

Part 5. Kbfx, czyli nowe spojrzenie na K-Menu.

No może nie całkiem nowe, bo coś podobnego pojawiło się już w Windows XP. A dodatkowo powiem także że w OpenSuse testują nowe menu – a zwie się Kickoff (do zobaczenia tutaj [8]). Ale miała być mowa o kbfx. Największą jego innowacyjnością i zarazem główna zaletą jest możliwość spersonalizowania sobie przycisku start. W standardowym menu KDE zawsze była jedna i ta sama ikonka, i ewentualne próby jej zmiany sprowadzały się do podmiany odpowiedniej ikony w aktualnie używanym zestawie ikon KDE. Tutaj jest inaczej. Z poziomu okna konfiguracyjnego można wybrać dowolny rysunek jako obrazowanie menu start. Obrazek może być w trzech opcjach: normal, hover (gdy kursor myszy jest nad przyciskiem lecz go nie naciska) i pressed (po naciśnięciu). Oczywiście nie szkodzi aby to był ten sam obrazek, albo np. ten sam ale ze zmienioną gamą kolorów. Wreszcie wygląd menu start można zmieniać tak samo jak tapetę czy zestawy ikon. Inną innowacyjnością w porównaniu do klasycznego K-menu jest sposób przeglądania menu. W Kbfx główne kategorie menu sa po lewej stronie, natomiast poszczególne pozycje stanowią przewijane za pomocą myszki menu po prawej (jeżeli nie pasuje nam ten sposób poruszania sie po menu, to jest możliwość powrotu do standardowych ustawień). Na dole sa też standardowo umieszczone przyciski blokowania ekranu / wylogowywania oraz linia poleceń do szybkiego uruchomienia aplikacji. W sieci można znaleźć mnóstwo tematów do Kbfx, o różnej kolorystyce, które pozwolą jeszcze bardziej spersonalizować wygląd menu. Strona domowa projektu to [9].

Part 6. Yakuake, czyli taka fajna konsola.

Yakuake można tłumaczyć jako 'Yet Another Kuake KDE terminal emulator' (wzięte ze strony domowej programu [10]). Ci co grali w Quake (albo inaczej - ci co nie grali w Quake ręka do góry) pamiętają zapewne taką fajną konsolkę wysuwaną od góry ekranu za pomocą kliknięcia magicznego przycisku. Ten program symuluje coś podobnego. Zamiast standardowych XTerma, Konsole, Aterma czy Urxvt możemy cieszyć się nieco innym terminalem. Ale po kolei. Program najlepiej ściągnąć z www.kde-apps.org. Są tam paczki dla ważniejszych dystrybucji. Download na stronie domowej jakiś taki mało przyjazny jest. Po instalacji odpalamy program lub jak kto chce to wrzucamy go do autostartu KDE. Przy uruchamianiu programu przez kilka sekund pojawia się napis typu “Press F12 to use”. I właśnie tym magicznym klawiszem uruchamiamy Yakuake. Klikamy F12 i wysuwa nam się z góry przeźroczysta konsolka, klikamy F12 raz jeszcze i konsola sie chowa. I tak można bez końca :-) chociaż oczywiście terminal służy do poważnej pracy a nie do zabawy. Program obsługuje wielość tabów (po naszemu chyba zakładek), aczkolwiek na tab-barze mieści się ich co najwyżej 15. Inną ciekawą a przydatną opcją jest możliwość przyczepienia okna Yakuake na wszystkie pulpity. Z opcji konfiguracyjnych programu należy wymienić: możliwość zmiany szerokości i wysokości okna terminala, położenia oraz czasu animacji wysuwania/wsuwania się, a także możliwość schowania tab-baru czy klawisza odpowiadającego za aktywację (jak wspomniałem wcześniej domyślnie jest nim F12). A ponieważ jestem złośliwy to także wymienię wady programu (przy czym są to bardziej moje widzimisię niż rzeczywiste wady). Po pierwsze, jeżeli ustawimy start Yakuake przy uruchamianiu KDE to po pewnym czasie znudzi/zdenerwuje nas ciągle wpatrywania się w radosny napis że aplikacja szczęśliwie wystartowała i aby ją zobaczyć wystarczy nacisnąć F12. Ja po pierwszym komunikacie już to wiedziałem. A w opcjach konfiguracji nie ma jak to deaktywować. A druga wada jest taka, że Yakuake respektuje moją wolną wolę w kwestii umieszczenia paska zadań na górze ekranu. A respektuje to w ten sposób, że wysuwając się z góry pomija pasek. Poczynawszy od góry widzę wtedy pasek i dopiero poniżej okienko aplikacji. Co akurat nie wygląda estetycznie. Nie miałbym nic przeciwko żeby program chwilowo przykrył mi pasek, co by dało lepszy efekt kolorystyczno-estetyczny. Szczególnie że zawsze mogę terminal schować bez utraty danych. Ale pewnie się tylko czepiam. Zresztą tak jak zwykle. I tym optymistycznym akcentem kończymy dzisiejszą audycję. Dobranoc Państwu. ■

Przydatne adresy

Dużo więcej screenów oraz opisy nowości można zobaczyć na stronie http://www.kde-look.org oraz na stronie http://www.kde-apps.org.

[1] http://netdragon.sourceforge.net/ssuperkaramba.html
[2] http://netdragon.sourceforge.net/sinfo.html
[3] http://www.xiaprojects.com/www/prodotti/kxdocker/main.php
[4] http://ksmoothdock.sourceforge.net/
[5] http://www.kde.cl/kooldock/
[6] http://kompose.berlios.de/
[7] http://www.uni-weimar.de/~wolff3/index_taskbar.html
[8] http://home.kde.org/~binner/kickoff/sneak_preview.html
[9] http://www.kbfx.org/
[10] http://yakuake.uv.ro

VPN i PPTP - jak to zrobić?

MARCIN 'lipiec' LIPIEC

Dane z wikipedii:

VPN - (ang. Virtual Private Network, Wirtualna Sieć Prywatna), można opisać jako "tunel", przez który płynie ruch w ramach sieci prywatnej pomiędzy klientami końcowymi za pośrednictwem publicznej sieci (takiej jak Internet) w taki sposób, że węzły tej sieci są przezroczyste dla przesyłanych w ten sposób pakietów. Taki kanał może opcjonalnie kompresować lub szyfrować w celu zapewnienia lepszej jakości lub większego poziomu bezpieczeństwa przesyłanych danych.

Określenie "Wirtualna" oznacza, że sieć ta istnieje jedynie jako struktura logiczna działająca w rzeczywistości w ramach sieci publicznej, w odróżnieniu od sieci prywatnej, która powstaje na bazie specjalnie dzierżawionych w tym celu łącz. Pomimo takiego mechanizmu działania stacje końcowe mogą korzystać z VPN dokładnie tak jak gdyby istniało pomiędzy nimi fizyczne łącze prywatne. Rozwiązania oparte na VPN powinny być stosowane w firmach, w których dosyć często pracują zdalnie ze swoich domów, na niezabezpieczonych łączach. Wirtualne Sieci Prywatne charakteryzują się dość dużą efektywnością, nawet na słabych łączach (dzięki kompresji danych) oraz wysokim poziomem bezpieczeństwa (ze względu na szyfrowanie).

PPTP (ang. Point to Point Tunneling Protocol) to protokół komunikacyjny umożliwiający tworzenie wirtualnych sieci (VPN) wykorzystujących technologię tunelowania. Polega to na zdalnym dołączaniu się do stacji roboczych lub sieci (głównie opartych na systemie operacyjnym Windows) za pośrednictwem Internetu i tworzeniu pozorów połączenia z lokalną siecią, bez wychodzenia z domu zapewniając jednocześnie zachowanie bezpieczeństwa przy zdalnym przesyłaniu danych.

Najbardziej rozpowszechniona implementacja PPTP została opracowana przez firmę Microsoft. Protokół PPTP stanowi standardowe wyposażenie systemu operacyjnego Windows od wersji 98 i NT.

W tekście tym opiszę konfigurację VPN z PPTP w oparciu o świetny system jakim jest m0m0wall. Co to za system?

M0m0wall został stworzony przez Manuela Kaspera. Jest to specjalna dystrybucja którą możemy instalować na komputerach stacjonarnych jak i na komputerach jednopłytowych takich jak (Soekris).

Konfiguracja m0n0wall'a odbywa się za pośrednictem przeglądarki i jes bardzo prosta. Zaletą tej dystrybucji jest to, że plik konfiguracyjny zapisywany jest w formacie XML.

Tworzenie i przywracanie kopii zapasowych sytemu jest także bardzo proste dzięki specjalnemu mechanizomi. W artykule tym zakładam, że użytkownik ma już zainstalowanego m0n0wall'a i pomijam opis jego instalacji.

Poprawna konfiguracja m0n0wall'a.

Do panelu konfiguracyjnego wchodzimi poprzez wpisanie w jako adres witryny, IP komputera gdzie stacjonuje m0n0wall.

Z menu wybieramy kolejno VPN => PPTP. Naszym oczom ukazuje się strona konfiguracyjna. Zauważ, że na górze widnieją dwie zakładki "Users" i "Configuration".

Przejdź do zakładki "Configuration" i zaznacz kółeczko przy "Enable PPTP server". Następnie w polu "Server address" podaj adres IP serwera PPTP. Nie wiesz co tutaj wpisać?

Pamiętaj, że adresem serwera PPTP może być adres który znajduje się bezpośrednio ponad adresem komputera z dystrybucją m0n0wall. Załóżmy, że system m0n0wall zainstalowałeś w komputerze jednopłytowym i IP tego komputerka to 10.14.28.1, więc adresem serwera PPTP powinien być ten adres: 10.14.28.2

Teraz przechodzimy do pola "Remote address range". W tym polu musisz wpisać adres sieci: np.:192.168.2.1. Jeżeli chodzi o konfigurację na zakładce "Configuration" to to już wszystko. Kliknij przycisk "Save" u dołu strony aby zapisać ustawienia.

Przechodzimy do zakładki "Users". Tutaj sprawa jest już prosta. W tej zakładce dodajemy nowego użytkownika, aby mógł on korzystać z naszej sieci.

Pole "Username" tutaj wpisz dowolna nazwe. Pole "Password" dowolne hasło, ale nie za proste! Pole "IP address" jeden adres z zakresu jaki wpisałeś w polu "Remote address range" na zakładce "Configuration".

Teraz tylko klikamy na przycisk "Apply changes".

Kolejną rzeczą jaką musimy zrobić będzie skonfigurowanie zapory sieciowej dla PPTP. A więc klikamy na znaczek "+" (plus) i konfigurujemy. Przedstawię tutaj przykładową konfigurację.

W polu "Action" określamy co dzieję się z przesyłanymi pakietami. Zalecam wybranie opcji "Pass" (oznacza że pakiety będą przepuszczane). W polu "Interface" wybierz rodzaj sieci (prawdopodobnie wybierzesz LAN).

W polu "Protocol" zalecam ustawienie TCP/UDP. Reszta pól powinna pozostać domyślna. Następnie kliknij w "Save", a następnie na "Apply changes".

Gratulacje! Właśnie skonfigurowałeś własne VPN z PPTP.

Jeżeli macie jakieś pytania odnośnie VPN i PPTP piszcie do mnie, chętnie pomogę.

Notebooki - przyszłość komputera ?

Spójrzmy prawdzie w oczy gdziekolwiek się nie oglądniemy otacza nas wszędobylska elektronika. Użytkowa, „efekciarska” czy po prostu służąca szeroko pojętej rozrywce. Młodzież słucha muzyki z „przyklejonych” do ciała przenośnych odtwarzaczy mp3, stateczni ojcowie korzystają z samochodów napędzanych silnikami sterowanymi coraz bardziej wyrafinowanymi centralnymi jednostkami zarządzającymi, kobiety mają rożne urządzenia dla wzmocnienia naturalnej urody – zabezpieczone układami wysokiej skali integracji. Wreszcie wszechstronny, stary poczciwy pecet – ciągle jest sercem do pracy i rozrywki, ale czy na pewno …

TOMASZ 'tomcash' CZUNKO

Na rynek konsumencki wkraczają przebojem coraz nowsze modele notebooków. Już nie są tak ciężkie i duże jak kiedyś. Wiele modeli może stawać w szranki z dumnymi, „dużymi” maszynami stacjonarnymi. Czas pracy na bateriach stale się zwiększa. Niedawno doszło do wręcz „absurdalnej” sytuacji – to pecety przejmują rozwiązania stosowane i sprawdzone w notebookach. Przykładem może tu być procesor Intela: Core 2 Duo. Jego energooszczędność i wydajność zarazem bije na głowę stare konstrukcje oparte o rozwiązania Pentium 4. Dziś liczy się cisza. Zmniejszenie hałaśliwości widać na każdym kroku. Jeden z czołowych producentów obudów ATX-mini/midi/mikro/… firma Mode Com

(wraz z wykupionym Logic Concept) rozpoczęła masowe stosowanie zasilaczy chłodzonych wolnoobrotowymi wentylatorami o rozmiarze 12 cm. Płyty główne są wyposażane w funkcję regula-

cji obrotów wentylatora procesora zależnie od obciążenia i warunków termicznych panujących wewnątrz obudowy. Większość z nas ma dość hałasu. Przychodząc do domu chcemy posłuchać ulubionych utworów mp3 czy oglądnąć film dvd. Jednak bardziej cenimy sobie ścieżkę dźwiękową niż szum wentylatorów. I na tym polu komputery przenośne nie mają konkurencji. Kino domowe możemy sobie urządzić gdzie jest nam w danej chwili wygodniej. Panoramiczne ekrany zapewniają bardzo dobre wrażenia wzrokowe. Podpięcie się do zewnętrznych głośników w większości notebooków jest koniecznością z racji słabych wbudowanych tychże. A hałasowi mówimy po prostu „pa pa”. Z doświadczeń kolegów oraz własnych chciałbym przedstawić krótko zalety i wady wybranych notebooków według podziału: wady i zalety danej marki (producenta). Artykuł ma na celu przedstawienie bardzo ogólnych doświadczeń w obcowaniu z seriami notebooków, aby ułatwić zorientowanie się i wybór odpowiedniej serii modelowej. Później można dobrać dokładniejsze parametry kierując się własnymi potrzebami i sugestiami znajomych i sprzedawców. „Let’s start” jak mawiają amerykanie. Acer – marka bardzo długo obecna na rynku, z bagażem doświadczeń, często niedoceniana (również przeze mnie). Kiedyś słynęła z maksymalnych konfiguracji za rozsądną cenę – niestety z jakością było już dużo słabiej. Obecnie proces produkcyjny został znacznie udoskonalony, przy zachowaniu niemal złotej proporcji ceny do wyposażenia. Do zalet zaliczyć możemy również bardzo dobrze wyprofilowany układ klawiatury i bardzo precyzyjny

touchpad. Z wad wymienić należy – średniej klasy baterie, kiepskie jakościowo zawiasy matrycy oraz średniej jakości plastiki – łatwe do zarysowania i dość szybko „starzejące się". Awaryjność – nie jest duża, jednak jest kilka lepszych pod tym względem marek. Co na to Asus ze swymi modelami ? Tu bateria jest bardzo

słabiutka. jakość wykonania i użyte materiały są na wysokim poziomie. Można mieć zastrzeżenia do wiotkości pokrywy górnej – ale mając na względzie koszt małych wymiarów i wagi, nie jest to już tak odczuwalne. Ilość dołączonego oprogramowania mniejsza niż u konkurencji – za to programy do zarządzania notebookiem bardzo rozbudowane. Wydajność Asusów jest stawiana wśród konkurencji jako klasa średnia wyższa. Ceny na przyzwoitym poziomie. Ciekawostka: Asus produkuje również tzw. kadłubki notebooków dla pasjonatów, którzy sami chcą poskładać swój laptop. Kadłubki te mają certyfikat zgodności z technologią Intel Core. Elementami dobieranymi są: procesor, pamięć RAM, dysk twardy, karty wifi na minipci. W zestawieniu nie mogło zabraknąć marki Toshiba (ulubiona marka autora ;)). Cóż tu dużo mówić – jakość, materiały, stabilność – wszystko na wysokim poziomie. Teraz łyżka dziegciu – tanie modele z serii Satelite są bardzo głośne i nie nadają się do roli domowego komputera „do wszystkiego". Przeciwieństwo ich to modele z serii Qosmio. Piękna stylistyka, ogromna moc, głośniki Harman Kardon, wydzielona klawiatura numeryczna (w wybranych modelach) – słowem komputer na ładny stolik w centralnym miejscu salonu ;). Lenovo – mocny gracz na rynku oferujący niezłe jakościowo własne konstrukcje oraz markę przejętą od IBM – ThinkPad. Podobnie jak wspomniana poprzednio Toshiba – tanie serie są bardzo dobre jakościowo i stabilne – natomiast budowane w oparciu o mało wydajne podzespoły (celeron M, zintegrowany układ graficzny, dość mało pamięci RAM). Modele topowe charakteryzują się bardzo wysoką wydajnością,

przy zachowaniu bardzo stabilnej pracy całości. Czytnik linii papilarnych zabezpieczający przed nieautoryzowanym dostępem wbudowany tuż obok touchpada to znak rozpoznawczy tej marki. Tonacja kolorystyczna – szaro-bura (tu podobieństwo do Asusa). Często niedoceniany, mający małą wpadkę z białą klawiaturą, Fujitsu Siemens Computers (FSC) – podbija półkę średnią wyższą. Najtańsze modele mają najmniej wydajne kontrolery grafiki na rynku. Czas pracy na baterii – średnia rynkowa. Natomiast komputery w cenie 3500-4500 to to co tygrysy FSC lubią najbardziej. Dobrze dobrane konfiguracje, najnowsze rozwiązania procesorowe i znakomity stosunek wydajności do ceny. Świetnie sprawujące się pod linuksem - możliwość kupienia praktycznie każdego modelu bez systemu z rodziny Windows daje nam spore oszczędności. Praktycznie każdy liczący się na rynku dystrybutor ma w swojej ofercie Fujitsu Siemens – powoduje to walkę cenową korzystną dla klienta.
Po tej krótkiej wycieczce wokół oferty obecnej na rynku zapraszam na zakupy – profesjonalny sprzedawca doradzi i dobierze szczegóły konfiguracji i warto na nim polegać. Jednak ogólny pogląd na sprawę również trzeba mieć i to miał na celu ten artykuł. W przyszłym roku pojawi się kilka bardziej szczegółowych testów sprzętu. Mając dostęp na co dzień z nowości sprzętowych postaram się opisywać wrażenia z dłuższego obcowania z notebookami i nie tylko. Oby nowy rok przyniósł znaczący przełom w branży IT – zarówno oprogramowania jak i

sprzętu. A tymczasem ciepłych i pełnych uczucia Świąt Bożego Narodzenia.

LittleSheep czyli zabawy z owieczkami

Dzisiaj zabawimy się z owieczkami :-) Małymi owieczkami. Litttle Sheep, bo tak nazywa się gierka którą chciałbym przedstawić, to prosta arcadówka, w której głównym i nadrzędnym celem jest przeprowadzenie owieczek przez niebiezpieczny i pokomplikowany topograficznie teren.

KAROL "AXIO" KOZIOŁ

Gierkę swego czasu zauważyłem na www.kde-apps.org po czym zachęcony pozytywnymi komentarzami sciągnąłem ją sobie i zainstalowałem. Gra ma prosty okienkowy interfejs. W zasadzie obsługuje się ją myszką, chociaż warto znać niektóre skróty klawiaturowe, np. na pauzę.

Zasady gry są proste - należy tak prowadzić owieczki aby doprowadzić je wszystkie (a jeżeli nie wszystkie to jak najwięcej) do mety będącej małym domkiem. Było by to prostsze, gdyby owieczki nie ruszały pędem przed siebie, a czasami też w zupełnie losowym kierunku (gdy owieczka osiągnie przeszkodę to albo zawróci, albo skręci w prawo, albo w lewo). Rzeczonymi owieczkami sterujemy za pomocą myszki. Klikając LPM na dowolny punkt planszy podzielonej na małe kwadraciki, stawiamy tam strzałeczkę - drogowskaz który wskaże przechodzącej przez ten punkt owieczce w którą stronę ma skręcić. Układajac takie drogowskazy w różnych miejscach planszy należy zmusić krnąbrne owce aby szły tam gdzie my im każemy, czyli do mety. Trzeba sie tylko śpieszyć, gdyz nasze podopieczne wcale nie czekają aż ustawimy im najlepszą trasę, tylko radośnie ruszaja do przodu. Jeżeli nie zdążymy nimi pokierować to uciekną za planszę lub się utopią lub coś je pożre...

A teraz trochę o urozmaiceniu plansz we wszelakie przeszkadzajki. Otóż mamy krzaki niejadalne i jadalne (za zjedzenie tych jadalnych chyba są dodatkowe punkty ale pewności nie mam), płoty, rowy ze smołą, rozlewiska wodne, przestrzenie piaszczyste. Ich rola jest różna. Na krzakach i piasku nie można ustawiać strzałek (tylko na trawie). Płoty oczywiście nie przepuszczają nam owieczek. Natomiast jeżeli owieczka wpadnie do smoły to staje się martwą owieczką. Nieco bardziej skomplikowana sytuacja jest z wodą. Owieczka może przejść przez zbiornik wodny, ale tylko przez jego 2 pola. Jeżeli zajdzie dalej to ginie (co jest symbolizowane przez mały szkielecik). Tak więc należy uważać jakimi ścieżkami prowadzi się swoje stado. Innym zagrożeniu utrudniającym ukończenie etapu są osy. Ale nie zwyczajne osy tylko osy-gigantu, zabijające jednym swym dosknięciem biedne owieczki. Trzeba jej unikać za wszelką cene, bo latając z jednego końca planszy na drugi potrafi wykosić nam połowe stada. Lecz nawet przed straszną osą-gigantem jest obrona (oprócz intuicyjnego prowadzenia owiec zdala od niej). Otóż gdy owieczka wyszła z wody (bo przypadkiem się nie utopiła) to przez pewien czas jest mokra i wtedy osa jej nie atakuje. Oczywiście nie zawsze mamy do dyspozycji zbiorniki wodne, dlatego należy stosować inne strategie. Na przykład na jednej planszy zapętliłem owieczki w prostokącie 2x3 (śmiesznie tak chodziły w kółko :-) i wypuszczałem je tylko wtedy gdy osa była odpowiednio daleko. Aha, i bym zapomniał - są teleporty. Każdy chyba wie do czego służą :-) Dodatkowym utrudnieniem jest to, że z początku nie wiemy który teleport jest połączony z którym, i w jakim kierunku teleportuje. Jednakże wiadomo, że jeżeli owieczka wejdzie do jednego teleportu dołem to wyjdzie innym górą. Zazwyczaj pierwsze podejście na danej planszy polega na sprawdzeniu rozkładu teleportów i dopiero za drugim razem mozemy zakończyć misję ratowania owieczek. Są jeszcze inne ograniczniki gry: czas na przejście danej planszy i liczba dostępnych strzałek, ale nimi mniej należy się przejmować. Oczywiście dobrze by było wykonać zadanie za pomocą minimalnej liczby strzałek (a tych jest zazwyczaj za dużo).

Gra nie jest niczym specjalnym, ale jest idealna aby się rozluźnić po całym dniu ciężkiej pracy. Miłą opcją jest możliwość zapisu/wczytania stanu gry, co nie zawsze w takich prostych gierkach.

Dla obowiązku podam jeszcze stronę producenta: *http://www.gamecraft.de/schafe/*

Życzę miłej gry :-)

Jeden z bardziej zaawansowanych topologicznie etapów :-)