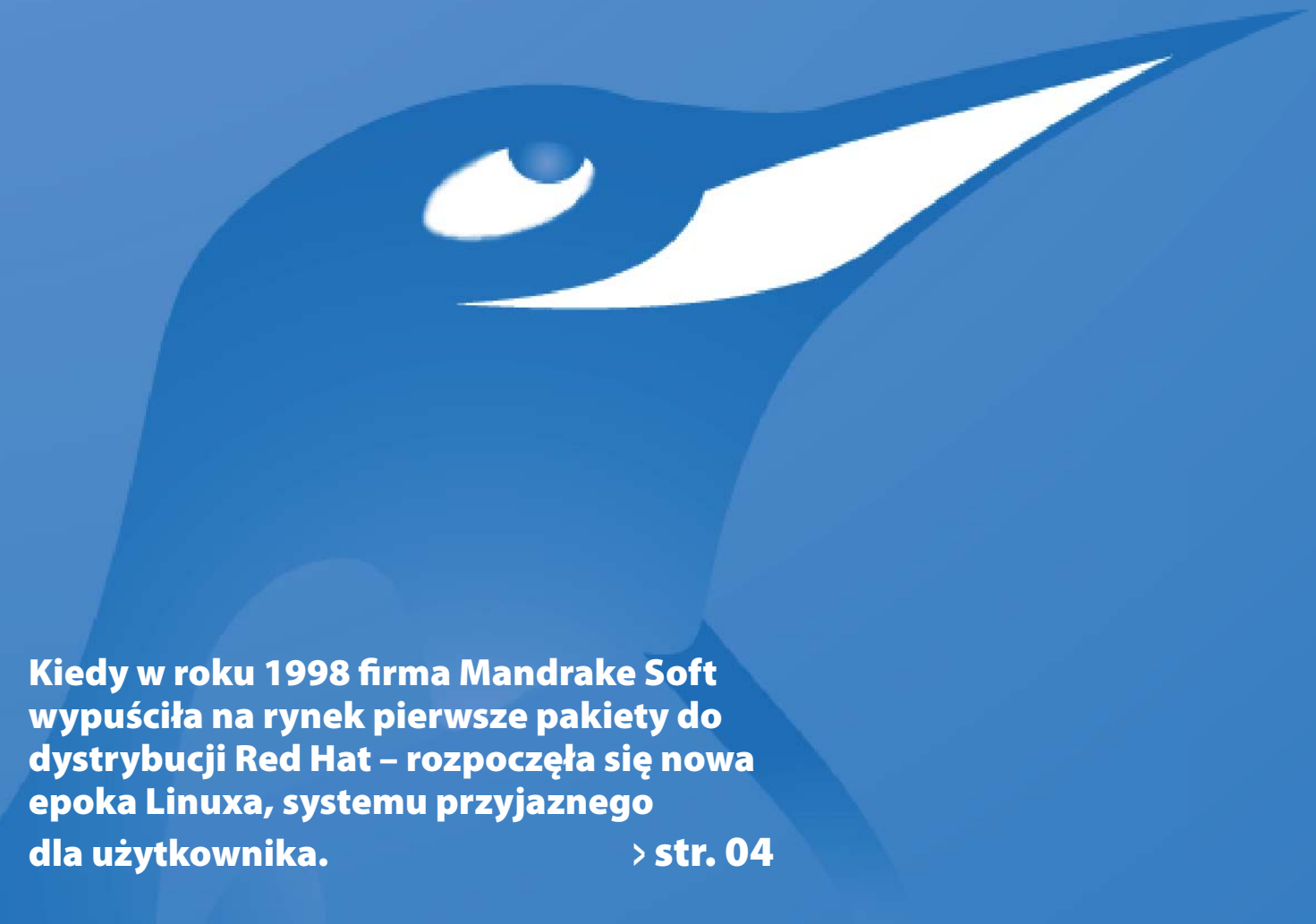
Magazyn użytkowników Linuksa
dragonia magazine
WYDANIE 4 | PAŹDZIERNIK | 2006

Mandriva 2007



Kiedy w roku 1998 firma Mandrake Soft wypuściła na rynek pierwsze pakiety do dystrybucji Red Hat – rozpoczęła się nowa epoka Linuxa, systemu przyjaznego dla użytkownika.

› str. 04

Novell Open Enterprise Server

To połączenie najlepszych usług jakie są dostępne w komercyjnym systemie Novell NetWare oraz dystrybucji SuSE Linux.

› str. 10

Darmowa moc

Wielu sądzi, że 64-bitowy procesor jest wciąż poza zasięgiem finansowym, a zakup Intelowskiego Core 2, czy Athlona 64 – to niewielka poprawa komfortu pracy.

› str. 14

Wprowadzenie do Tcl/Tk

Artykuł stanowi pierwszy odcinek serii o języku Tcl/Tk. Aktualny kurs zachęci czytelnika do próby poznania tego języka, ze szczególnym naciskiem na koncepcję budowy interfejsu użytkownika.

› str. 24

Wstępniak

Witamy na łamach czwartego numeru czasopisma **dragonia magazine**. Wraz z nim mamy dla Was kilka niespodzianek. Jak zauważycie – mamy nową szatę graficzną – To dzięki Marcinowi Szewczykowi z m4c.pl, który zaprojektował nam makietę i kilka rozwiązań redakcyjnych. Pismo będzie miało teraz stały wygląd wraz z określonymi działami aby ułatwić Wam nawigację.

Ciepłym słowem i korektą wszystkich tekstów zajęła się Agnieszka, jej również należą się w tym miejscu fanfary, mimo że za nimi nie przepada:)

W tym numerze przeczytamy m.in. o nowej wersji dystrybucji Mandriva Linux 2007, a wraz z nią mamy dla Was konkurs i do wygrania pudełkowe wersje dystrybucji 2006 - a to za sprawą starań Piotrka vel Dragona. Szczegóły znajdziecie wewnątrz numeru.

Będąc przy temacie dystrybucji, rozpoczynamy również cykl artykułów nt. Novell Open Enterprise Server wraz z opisem jego możliwości i technologii.

W dalszej części nadal będziemy poznawać tajniki języka C++ i Java. Dowiecie się też jak podciągnąć Waszego Athlona i jak zabrać się za programowanie z użyciem biblioteki Tk.

Dla relaksu zachęcamy do przeczytania recenzji gry OpenTTD.

Zapraszamy!

ZESPÓŁ DRAGONII

Spis treści
wydanie 4 | październik

Kierowanie zespołem: Piotr Krakowiak (dragon); **Autorzy:** Tomasz Czunko (tomcash), Piotr Szewczuk, Rafał Trójniak (ert16), Maciek Malinowski (borimir), Piotr Szewczuk, Rafał Domeracki (RD).

Współpracujący: korekta - Agnieszka (Mości Goblin); **projekt graficzny/współredagowanie** - Marcin Szewczyk (m4c), Marcin Lipiec (lipiec), Jakub Bousseria (fifim), Piotr Czajkowski (Shadowchaser).

Adres redakcji: dragonia.magazine@gmail.com

SLACKWARE 11.0 WYDANY!

Wreszcie doczekaliśmy się! Ponad rok temu wydano Slackware 10.2. Nowa wersja nie jest tak bardzo przełomowa jak zapowiadano. Po pierwsze od razu rzuca się w oczy fakt, że ponownie domyślnym kernelem w czasie instalacji będzie jajko z serii 2.4 sygnowane numerkiem 2.4.33.3, kernel 2.6.17.13 będzie dostępny w /extra, a 2.6.18 w /testing. Na pewno ucieszy to osoby, które są fanatykami bezpieczeństwa, jednak dla większości użytkowników będzie to spory zawód, gdyż leciwy już 2.4, do którego obecnie nie dodaje się nowych sterowników słabo wspiera najnowszy sprzęt. W dystrybucji znajdziemy środowisko graficzne KDE 3.5.4 z Amarokiem, którego wcześniej brakowało. Kolejnym zawodem po kernelu jest moim zdaniem server X. Wydanie miało być przełomowe, więc zupełnie nie rozumiem czemu nie porzucono monolitu 6.9 dla modularnego 7.0. Na przejście między tymi wersjami trzeba będzie jeszcze długo poczekać znając filozofie Slacka, a szkoda bo oficjalnie nowe aktualizacje będą wypuszczane jedynie dla wersji 7.0, a monolit zostaje porzucony. Plusem wydania jest to, że po raz pierwszy możemy ściągnąć oficjalną wersje na DVD (do tej pory były to 2CD). Gdy pisze tego newsa jeszcze niewiele mirrorów było zaktualizowanych i udostępniało wersję 11.0. Podsumowując zawiodłem się trochę na nowym Slackware. Bardzo liczyłem na te rewolucyjne zmiany, a zwłaszcza na Xorg 7.0. Szkoda, bo korzystając z okazji można było wiele zmienić w fundamencie tej dystrybucji.

CZTERORDZENIÓWKI AMD PÓŁ ROKU PÓŹNIEJ NIŻ INTELOWSKIE

W trzecim kwartale 2007 roku AMD ma wypuścić na rynek linię swoich nowych procesorów – architektury K8L. Dwa z nich to produkty czterordzeniowe o nazwach kodowych Altair i Altair FX, taktowane prawdopodobnie na poziomie 2.7 – 2.9 GHz. Mimo, że procesor prawdopodobnie będzie działał na najnowszej podstawce AM2, to jednak dopiero kolejna platforma – AM2+ - będzie udostępniać procesorowi maksymalną możliwą przepustowość.

GRAFIKA WEKTOROWA… NO PROBLEM!

Na stronie http://www.inf.sgsp.edu.pl/lab/filmiki/filmiki.php pojawiły się filmy przedstawiające możliwości programu Inkscape. Autorem publikacji jest Karol Kreński (mimooh at inf.sgsp.edu.pl). Materiały szkoleniowe dotyczą najnowszej wersji 0.44 programu. Głosu użyczyli (po usilnych namowach autora): Marek Kondrat oraz Gustaw Holoubek. Do odtworzenia autor poleca Mplayera a pod Windows wymagany jest kodek ffdshow. DM poleca!

CZTERORDZENIOWY INTEL

Ledwo co na półkach pojawiły się dwurdzniowe Intelowskie Core 2 Duo, a już serwis Tom's Hardware wszedł jakimś sposobem w posiadanie inżynierskiej próbki procesora z CZTEREMA rdzeniami w architekturze Core 2. Urządzenie o nazwie kodowej Kentsfield taktowane jest zegarem 2.66 GHz, i posiada 8 MB pamięci podręcznej . Benchmarki, prezentujące osiągi tego cuda można oglądać tu: http://www.tomshardware.com/2006/09/10/four_cores_on_the_rampage/page8.html. Procesor ma się ukazać oficjalnie za parę miesięcy, lub na początku 2007 roku.

EUDORA + THUNDERBIRD = MOCNY GRACZ

Fundacja Mozilla i producent programu Eudora - firma Qualcomm rozpoczęły współpracę w zakresie tworzenia klientów poczty elektronicznej. Klient pocztowy Eudora zostanie oparty na sprawdzonym silniku Thunderbirda. Źródła „nowego" programu zostaną otwarte. Aplikację będziemy mieli możliwość pobrać za darmo. Wspólny projekt nie zmieni dotychczasowej funkcjonalności programu i zapewni ulepszenie silnika Thunderbirda o nowe funkcje na podstawie doświadczeń programistów Qualcomm. Spodziewana data darmowej edycji Eudory to pierwsza połowa przyszłego roku.

NOWA WERSJA OPENOFFICE.ORG (2.0.4)

Serwery lustrzane wypełniają się powoli najnowszą wersją popularnego pakietu biurowego pod Windows i Linux. OpenOffice.org 2.0.4 bo o nim mowa zawiera szereg poprawek oraz dwie nowe funkcje: możliwość dodania hasła przy eksporcie do .pdf oraz funkcję aktualizacji automatycznej. 12-13 października ukażą się wszystkie wersje językowe oprogramowania – prócz dostępnej jako pierwsza wersji angielskiej.

STABILNA WERSJA 0.3 SUNBIRDA

Bardzo długo oczekiwana premiera zaskoczyła nas stosunkowo niedawno. Fundacja Mozilla przedstawiła darmową, wieloplatformową aplikację typu PIM. Oparty o otwarty standard iCalendar program Sunbird wita nas w nowym wydaniu:

> nową architekturą przechowywania kalendarzy ,

> bardziej intuicyjny interfejsem,

> przeprojektowanym wyglądem okna preferencji,

> obsługą wtyczek, schematów graficznych i paczek lokalizacyjnych.

> nowym instalatorem,

> poprawionym drukowaniem

Z sieci zebrał tomcash

[EOT]

Mandriva Linux 2007 RC2

Kiedy w roku 1998 firma Mandrake Soft wypuściła na rynek pierwsze pakiety do dystrybucji Red Hat – rozpoczęła się nowa epoka Linuxa, systemu przyjaznego dla użytkownika.

Tomasz „tomcash" Czunko

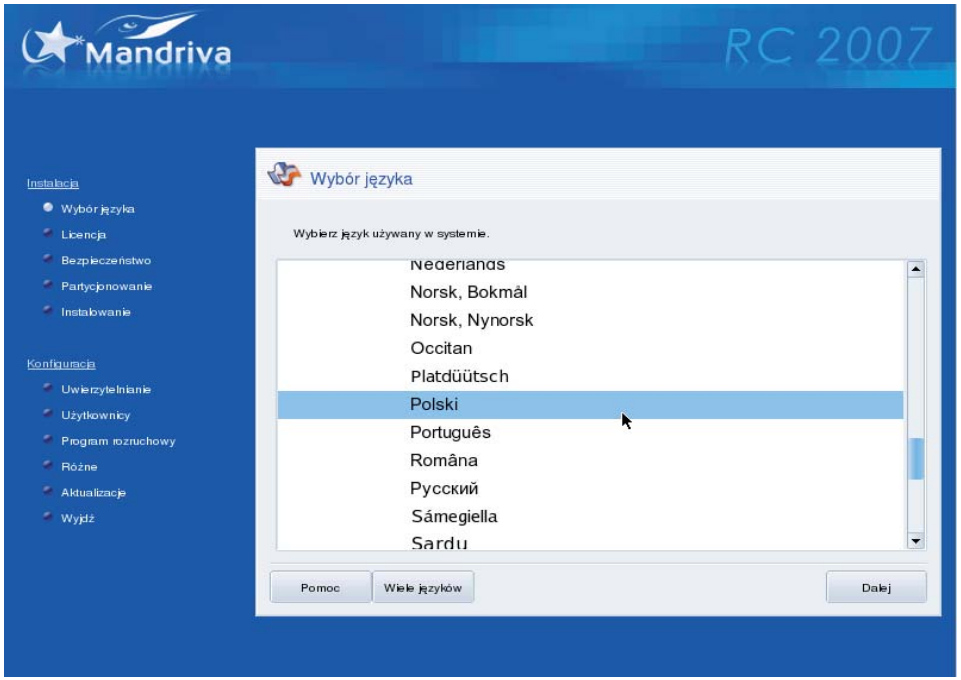
Rok ten dał początek dystrybucjom, które znacząco podniosły poprzeczkę w komforcie użytkowania systemów linuxowych przez „nałogowych windziarzy". Pierwszą dystrybucją opartą o sprawdzoną podstawę Red Hata był Mandrake Linux 5.1 (venice). Spoglądając z perspektywy lat, nazwa Venice jest dość ciekawa. Mnie osobiście kojarzy się z poprawioną wersją 64-bitowego procesora AMD Athlon – procesora, który podbił serca graczy i nie tylko. Venezia jest ciekawym miastem – podobnie jak następca wersji 5.1, czyli Mandriva Linux 2007. Po przejęciu firmy Lycoris i Connectiva, odzyskaniu płynności finansowej oraz zmianie nazwy z Mandrake na Mandriva francuska firma rozpoczęła nowy etap rozwoju. Nie ma drugiego tak dobrego pod względem ilości graficznych narzędzi ogromnego pakietu programów (wersja PowerPack 2006 zawiera ponad 3000 programów użytkowych, narzędziowych, graficznych i innych). Powracając jeszcze nieco do historii, Mandriva przystosowała dystrybucję Red Hat do gustów Europejczyków. Podobnie postępują koncerny samochodowe, jak np. Toyota, umieszczając swoje fabryki na Starym Kontynencie. I szczerze powiedziawszy, oba podejścia sprawdziły się doskonale w praktyce. Ale po krótkiej teorii czas na praktykę. Zacznijmy więc zagłębiać się w kod, którym firma Mandriva chce zawojować nasze biurka już od 21 października.

Dzięki uprzejmości pracowników polskiego oddziału Mandriva (www.mandriva.pl), znakomitej pomocy redaktora naczelnego „Dragonia Magazine" i Poczty Polskiej (wszystkim Wam wielkie dzięki) mogę przedstawić moje wrażenia z obcowania z Mandrivą 2007 RC2. Już po dwóch dniach od opublikowania na serwerach obrazów.iso do mojego domku zapukał listonosz ze świeżutko wypaloną płytką instalacyjną. Dla ścisłości podam dokładną nazwę: Mandriva Linux 2007 RC2 (Sunna) – Free Edition. Pierwsze, co zaskakuje po uruchomieniu komputera z płyty CD/DVD, to nowy wygląd ekranu instalatora. Znakomita według mnie grafika w formie kostek o różnych odcieniach koloru niebieskiego podkreśla stabilność i profesjonalizm. Motyw ten jest czasem wykorzystywany na stronach internetowych. Kiedyś wykorzystywał go portal www.wss.pl – w ładnym kolorze czerwieni. Pozostając na chwilkę w temacie grafiki i szaty graficznej, chciałbym przedstawić nowe pudełka firmy z logo gwiazdki. Plik graficzny pochodzi ze zrzutów ekranu utworzonych podczas instalacji wersji Release Candidate 2.

Nowy motyw wraz z charakterystycznym „zaczepem" z prawej strony frontu pudełka dodał dużo uroku pingwinowi z żółtą gwiazdą

Nowy motyw wraz z charakterystycznym „zaczepem" z prawej strony frontu pudełka dodał dużo uroku pingwinowi z żółtą gwiazdą. Oczywiście, nie musi to być ostateczny wygląd gotowego produktu handlowego. Niemniej jednak wersja RC2 jest już ostatnią wersją kan-

Po krótkim ładowaniu się „drake'a" instalacji naszym oczom ukazuje się dobrze znany instalator z wersji 2006



Spolszczenie systemu? – no problem!

dydującą. Kod źródłowy jest w tym momencie na etapie poprawiania ostatnich błędów. Dlatego mając to na uwadze, z niewielkim marginesem błędu można stwierdzić, że ostateczny system będzie już bardzo podobny do RC2. Ale wracając do instalacji – wybieramy język, rozdzielczość ekranu, ewentualnie dodatkowe sterowniki do sprzętu i źródło, z którego instalator będzie pobierał pakiety.

Po krótkim ładowaniu się „drake'a" instalacji naszym oczom ukazuje się dobrze znany instalator z wersji 2006.

Etapy instalacji są praktycznie identyczne ze znanymi z Mandrivy 2006.0. Akceptacja licencji, wybór poziomu bezpieczeństwa, przygotowanie miejsca na dysku, wybór, a następnie kopiowanie pakietów, hasła i użytkownicy, a na końcu konfiguracja sprzętu i oprogramowania. Jedyną nowością są opcje ustawień karty graficznej. Poniżej zrzut ekranu z instalacji na wirtualnym komputerze ze słabą kartą graficzną typu S3 Trio.

Na zintegrowanej karcie graficznej laptopa – model karty ATI Radeon IGP 345 M (karta pokroju Radeon 7000) opcji jest jednak więcej. Służą ustawieniu efektów pulpitu 3D. Jest to jedna z nowości wersji 2007.0. Dla potrzeb ustawiania efektów graficznych pulpitu powstał Drak3D. Aplet Centrum Sterowania, może być również uruchamiany samodzielnie oraz z poziomu KDM – menedżera logowania KDE. Na moim IGP 345 M udało mi się uruchomić tylko AIGLX. Niestety, szybkość działania dyskwalifikuje to rozwiązanie do codziennej pracy. Oczywiście z powodu słabej wydajności mojej karty. Z testów przeprowadzonych z XGL na Suse Linux Enterprise Desktop 10 oraz Kororaa-XGL-liveCD-0.2 wnioskuję, że XGL jest lepiej przystosowane do starszych chipów. Polecam zapoznanie się na własnej skórze z pulpitem 3D. Być może już niedługo będzie to standard obsługi graficznych środowisk systemów operacyjnych, czyli tzw GUI. Również firma Microsoft w swoim najnowszym produkcie Windows Vista zaimplementowała obsługę dodatkowych fajerwerków pulpitu.

Obiektywna ocena wyglądu Visty i XGL/AIGLX zdecydowanie więcej głosów powinna przydzielić drugiemu tandemowi. Efekty specjalne serwowane przez nowy Windows to około 1/5 możliwości jego opensource'owych odpowiedników. To, czego póki co brakuje w dziele inżynierów firmy MS, to między innymi:

> pulpit w kształcie trójwymiarowej kostki obracanej wokół głównej osi,

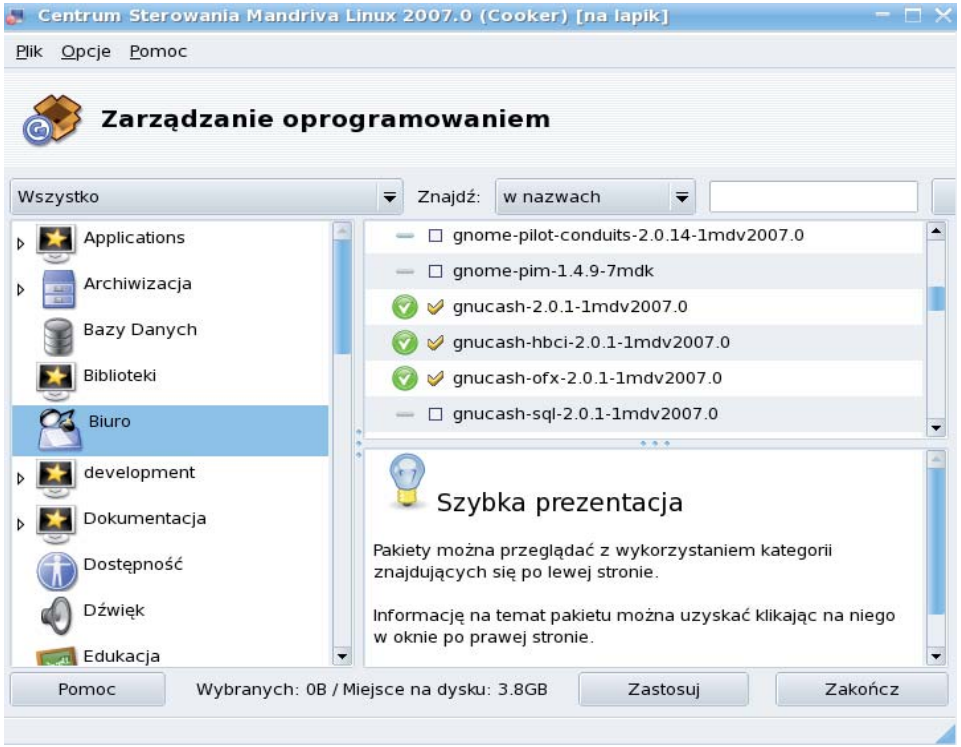
> efekt trzęsących się przy przesuwaniu okienek,

> konfiguracji kilkunastu opcji dodatkowych związanych z zachowaniem się okien.

Cały OS został w pełni zaktualizowany. Mamy tu OpenOffice.org 2.0.3, KDE 3.5, Gnome 2.16, Amarok 1.4 itd. Nowe jądro, nowe sterowniki i centrum zarządzania – nie tylko kryzysowego:). A najważniejszą aktualizacją jest nowy interfejs graficzny i sam zarządca pakietów rpm. Zakochałem się w nim od... drugiego wejrzenia. Dlaczego od drugiego, wszystko po kolei. Pierwsze, co zalecam zaraz po zainstalowaniu się naszego nowego systemu operacyjnego, to dołączenie źródeł pakietów (np. z pomocą strony http://easyurpmi.zarb.org/– gorąco polecam). Kolejnym krokiem jest aktualizacja drake'a odpowiedzialnego za instalację/usuwanie/aktualizację paczek.rpm. Niestety, ten zawarty w wersji Sunna ma kilka błędów objawiających się niemożnością zainstalowania dodatkowego oprogramowania. Stąd, niestety, nie była to miłość od pierwszego wejrzenia. Po aktualizacji (w końcowym etapie również wyskakuje błąd, ale nie należy się nim przejmować – wszystko zaktualizowało się poprawnie) możemy rozpocząć przygodę z dodawaniem programów. Można by rzec, że nareszcie doczekaliśmy się programu, który w jednym okienku pozwala dodawać, usuwać i aktualizować pakiety. Udaje mu się to świetnie. Towarzysząca oprawa wizualna, zwłaszcza zielone ikonki przy zainstalowanych paczkach, jest bardzo ładna.

Poezją można nazwać możliwości drzemiące w menedżerze pakietów wersji 2007. Potrzebujesz programu – klikasz i instalujesz. To łatwiejsze niż w systemach Windows i MacOSX.

Z programów multimedialnych na uwagę zasługuje Amarok w wersji 1.4. Poprawiona ergonomia obsługi, a przede wszystkim wygoda sterowania głośnością, zasługują na wspomnienie. Nowa preinstalowana wersja pakietu biurowego OpenOffice zgodna ze standardem OpenDocument (.odt), w której został między innymi napisany ten artykuł, to powód do radości użytkowników domowych i biurowych. Subiektywna prędkość uruchamiania i działania wersji 2007.0 w porównaniu do 2006.0 jest nieco wyższa. Z ostatecznym sądem poczekajmy do niezależnych testów wersji finalnej. Najbardziej denerwującą słabością Sunny jest natomiast nadal niedziałający kreator konfiguracji połączenia bezprzewodowego. Chodzi dokładnie o ndiswrappera. O ile w obecnie sprzedawanej wersji konfigurator potrafi zawiesić instalatora – o tyle wersja najnowsza w ogóle nie chce działać. Podczas etapu „różne" kreatora instalacji wyskakuje błąd „brak zainstalowanego pakietu wireless-tools". Jakież było moje zdziwienie, gdy po pierwszym uruchomieniu Centrum Sterowania Mandriva okienkowa instalacja sterownika karty opartej o chip Texas Instruments acx100 odbyła się bez najmniejszego problemu z wykorzystaniem wcześniej wspomnianego modułu tłumaczącego sterowniki systemu Windows na zrozumiałe dla jądra Linux. Mandriva Linux jest jak na razie jedyną z nowych dystrybucji, która potrafi mimo wszystko poprawnie uruchomić połączenie bezprzewodowe na mojej karcie D-Link DWL-650+. Testowane Ubuntu 6.06, Suse 10, SLED 10, Arch Linux 0.7.2, Slackware 10.2, Simply Mepis 6 i kilka innych nie podołały zadaniu zmuszenia do działania skądinąd starej już dziś karty DWL-650+.



Poezją można nazwać możliwości drzemiące w menedżerze pakietów wersji 2007

Bardzo rzadko administratorzy polecają aktualizację do najnowszych rozwiązań zaraz po ukazaniu się nowszej wersji aplikacji. Myślę, że mają rację. Opóźnienie przejścia do nowszego środowiska do czasu wydania łat naprawiających ważniejsze błędy i dziury jest rozsądne dla użytkowników potrzebujących skrajnie stabilnego stanowiska pracy. Mimo tego po naprawieniu błędów z wersji RC2 środowiska Mandriva Linux 2007 przez programistów francuskiej korporacji polecam zainstalowanie wersji Official. Jest to kolejny ogromny krok do przodu w bitwie o przejęcie części rynku skupionego wokół giganta z Redmond. Kolejny... w dobrym kierunku. Z wielkim napięciem będę czekał na „Mocną Paczkę 2007" (Power Pack 2007). Zapraszam wszystkich czytelników do podjęcia inicjatywy Mandriva Linux 2007 – Worldwide Install Party. Szczegóły w globalnej sieci pod adresem http://www.mandriva.pl/aktualnosci/? a=w&id=231. Nawet jeżeli nie jesteś zwolennikiem Mdv (skrót od Mandriva), oglądnij działającą wersję 2007. Tym bardziej, jeżeli jesteś lokalnym guru i masz wpływ na świadomość wyboru i legalność naszego społeczeństwa. Trudno będzie przekonać początkujących użytkowników komputerów do Debiana czy Slacka, ale łatwa w użyciu i konfiguracji Mandriva Discovery lub Free może się okazać idealną dystrybucją na początek. Start do lepszego jutra ponownie otwiera drzwi... teraz czas na Ciebie! Bądź SMART! [EOT]

System skrojony na miarę...

Mandarynkowy konkurs!

Jeżeli jesteś fanem dystrybucji Mandriva lub interesujesz się linuksem, poniższe pytania nie będą dla Ciebie zbyt trudne. Warto więc spróbować swoich sił i wysłać odpowiedzi na **konkurs@dragonia.pl** wraz z podaniem imienia i nazwiska oraz adresu i kontaktu (mail, telefon, IM). Pośród prawidłowych odpowiedzi rozlosujemy **dwie pudełkowe wersje Mandriva Linux Power Pack 2006.**
A oto pytania:

1. **W którym roku powstała pierwsza dystrybucja Mandriva (Mandrake) Linux ?**
2. **Wymień nazwy co najmniej dwóch wersji najnowszego wydania Mandriva Linux 2007.**
3. **Pieszczotliwa nazwa Mandrivy popularna wśród jej fanów to:**
   a. **linuksik**
   b. **mandarynka**
   c. **pomarańcza**

Na odpowiedzi czekamy do 15 listopada.
Wyniki ogłosimy w kolejnym numerze.

Życzymy szczęścia w losowaniu!

Zespół **dragonia magazine**

Mandriva 2007

PRZYJAZNY i BEZPIECZNY SYSTEM OPERACYJNY

Linux

Nowy

Powerpack 2007 PRO

Mandriva Linux 2007 jest najnowszą wersją systemu Mandriva Linux. Jest jeszcze prostszy w obsłudze, jeszcze bardziej przyjazny dla użytkownika i bardziej wydajny. Został zaprojektowany w taki sposób, aby zaspokoić wszystkie potrzeby i oczekiwania użytkowników: od początkujących po zaawansowanych.

transgaming's cedega: możesz grać w swoje ulubione gry!

Dzięki kompatybilnemu oprogramowaniu **TransGaming Cedega**, możliwość grania w World of Warcraft, Battlefield 2, Civilization IV, Counter-Strike i wiele więcej. W dystrybucji zawarto pełną wersję gry Flatout.

oglądaj filmy dud pod Linuksem!

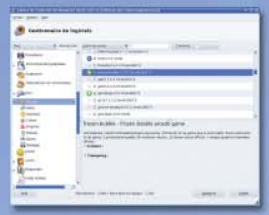
Mandriva Linux 2007 zawiera **InterVideo LinDVD**, który pozwala Ci legalnie odtwarzać twoje płyty DVD. InterVideo jest wiodącym dostawcą oprogramowania DVD.

kaspersky anty-wirus

Kaspersky Lab jako pierwsza wprowadziła oprogramowanie antywiru-sowe zintegrowane z Linuksem. Zabezpiecz się przed wirusami i innym niechcianym oprogramowaniem.

technologia 3d desktop

Nowe wsparcie dla **3D desktop** (**AIGLX** i **Xgl**) oraz nowe narzędzie do konfiguracji (drak3d): jesteśmy pierwszą dystrybucją która umożliwia wykorzystanie obu tych technologii.



rpmdrake 2:

Mandriva pracowała nad lepszym narzędziem umożliwiającym instalowanie, odinstalowanie i aktualizację oprogramowania. Nowy RPMDrake szybko pokazuje Ci co masz, co możesz zainstalować, co poleca Mandriva, jakie aktualizacje są dostępne i wiele więcej.

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

Novell Open Enterprise Server

„Cross the bridge from NetWare to Linux" – cz. I

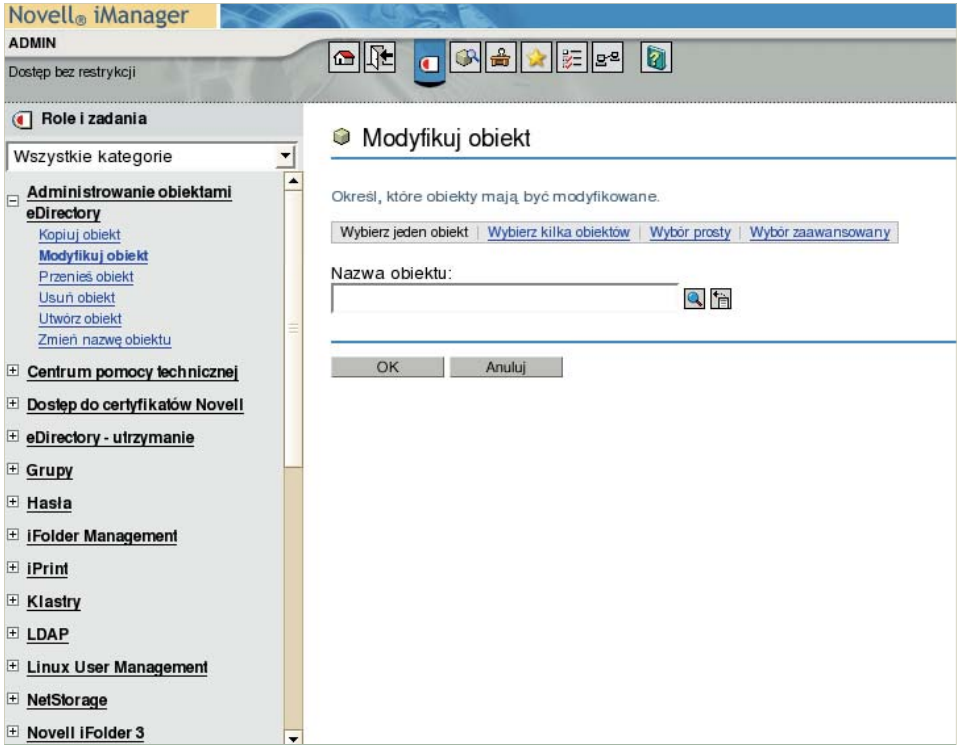
Novell Open Enterprise Server, w skrócie Novell OES, jest połączeniem najlepszych usług jakie są dostępne w komercyjnym systemie Novell NetWare oraz dystrybucji SuSE Linux.

Piotr Szewczuk

Zaproszony przez redaktora naczelnego **dragonii** do współpracy długo zastanawiałem się nad artykułem który zainteresowałby czytelników magazynu, a jednocześnie mógłby posłużyć jako pewnego rodzaju instrukcja przy ewentualnych wdrożeniach lub też być swego rodzaju poradnikiem. W ten sposób powstał pomysł stworzenia cyklu artykułów dotyczących systemu Novell Open Enterprise Server, który z pewnością dla wielu czytelników nie jest kojarzony z czymś co ma związek z Linuksem.

Przeważnie Linux wykorzystywany jest w firmach jako system zapewniający podstawę dla szeroko rozumianych usług internetowych: poczty elektronicznej, serwerów www czy proxy, usług takich jak DNS czy też systemów firewall. Niestety ze względu na brak narzędzi i usług centralizujących zarządzanie środowiskiem korporacyjnym Linux nie jest stosowany w większych przedsiębiorstwach jako centralny system operacyjny.

Novell Open EnterpriseServer, w skrócie Novell OES, jest połączeniem najlepszych usług jakie są dostępne w komercyjnym systemie Novell NetWare oraz dystrybucji SuSE Linux. Dzięki takiemu połączeniu uzyskano idealny system do stosowania w roli szkieletu informatycznego przedsiębiorstwa oferującego zaawansowane funkcje związane z: usługami katalogowymi, synchronizacją plików pomiędzy klientami a serwerem, dostępem do plików i drukarek, zaawansowanych funkcji bezpieczeństwa (w tym kryptografii), zarządzaniu tożsamością sieciową użytkowników, sieciowych pamięci masowych czy też tworzeniu klastrów. Wszystkie wymienione cechy są realizowane poprzez dostępne w



Przykład narzędzia iManager

systemie takie usługi jak: Novell eDirectory, Novell iFolder, Novell iPrint, Novell IDM, Novell Cluster Services, Novell iManager, Remote Control, YAST.

Aby zrozumieć ideę jaka przyświecała autorom systemu OES należy cofnąć się w czasie i przejść krótką lekcję historii. Firma Novell została założona

w 1979 roku w Provo w stanie Utach, a w 1983 zmieniła nazwę na Novell Inc. używaną do dziś. Od początku swojej działalności firma zajmowała się szeroko pojętymi produktami sieciowymi. Można powiedzieć, że jest jedną z tych firm dzięki której mamy możliwość swobodnego korzystania z sieci, w tym z najpopularniejszego standardu sieci LAN jakim jest Ethernet (Novell wypuścił serie kart sieciowych NE2000 udostępniając ich specyfikacje co pozwoliło na produkcję na szeroką skalę przez różnych producentów kart zgodnych z NE2000).

Sztandarowym produktem firmy Novell na początku lat 90 był system operacyjny Novell NetWare wykorzystywany przede wszystkim na serwerach pełniących funkcję serwerów plików oraz jako serwer wydruków. Wraz z wypuszczeniem na rynek wersji 4.0 systemu NetWare Novell (1994 rok) dokonał, rewolucyjnej jak na owe czasy, znaczącej zmiany w sposobie przechowywania informacji o użytkownikach w systemie. Typową płaską bazę kont systemowych zastąpiono zintegrowaną z systemem usługą katalogową NDS (Novell Directory Services) przemianowaną później przez firmę Novell na eDirectory. eDirecory pozwala na centralne przechowywanie informacji o kontach użytkowników, komputerach oraz centralne zarządzanie obiektami i usługami sieciowymi. Jak bardzo idea przechowywania informacji w usłudze katalogowej okazała się słuszna niech świadczy fakt, że pomysł firmy Novell był później kopiowany przez inne firmy (Microsoft wprowadził usługę katalogową do swojego systemu operacyjnego Windows 2000 Server w 2000 roku pod nazwą Active Directory). Wraz z rozwojem ruchu Open Source i systemu Linux, oraz jego coraz większą popularnością, firma Novell postanowiła dokonać zmian w swoim modelu biznesowym. Doprowadziło to do przejęcia w 2003 roku firm Ximian oraz SuSE, producenta popularnej dystrybucji SuSE Linux. Od tego czasu Novell systematycznie zajmuje się "wprowadzaniem" systemu Linux oraz usług opartych na nim do średnich i dużych przedsiębiorstw. W 2005 roku postanowiono połączyć dwa systemy operacyjne które Novell posiadał w swojej ofercie: NetWare oraz SuSE Linux. W ten sposób powstał Novell Open Enterprise Server który jak głosi tytułowe hasło jest mostem pomiędzy systemem NetWare a systemem Linux. Dla uzyskania jak największej wszechstronności oraz łatwości konfiguracji zarządzania systemem odbywa się za pomocą narzędzi (takich jak iManager, Server Health) opartych o przeglądarki internetowe oraz za pomocą narzędzia YAST. W pierwszej wersji OES dostępny jest z dwoma wersjami jądra do wyboru.

Klienci mogą wybrać czy chcą korzystać z systemu opartego o tradycje jądro NetWare czy też z systemu opartego o jądro Linuksa. Ze względu na charakter pisma czytelnicy będą mieli okazje zapoznać się z Novell OES zawierającym jądro Linux.

W ramach planowanego cyklu artykułów postaram się przedstawić i opisać proces instalacji i wdrażania głównych usług dostępnych w systemie Novell OES. Opis zostanie oparty na przykładowej instalacji wykonanej w środowisku VMware. Osoby które chciałaby własnoręcznie skorzystać z informacji zawartych w artykułach powinny pobrać wersję testową systemu OES oraz środowiska VMware. Wersje instalacyjne można pobrać z poniższych stron:

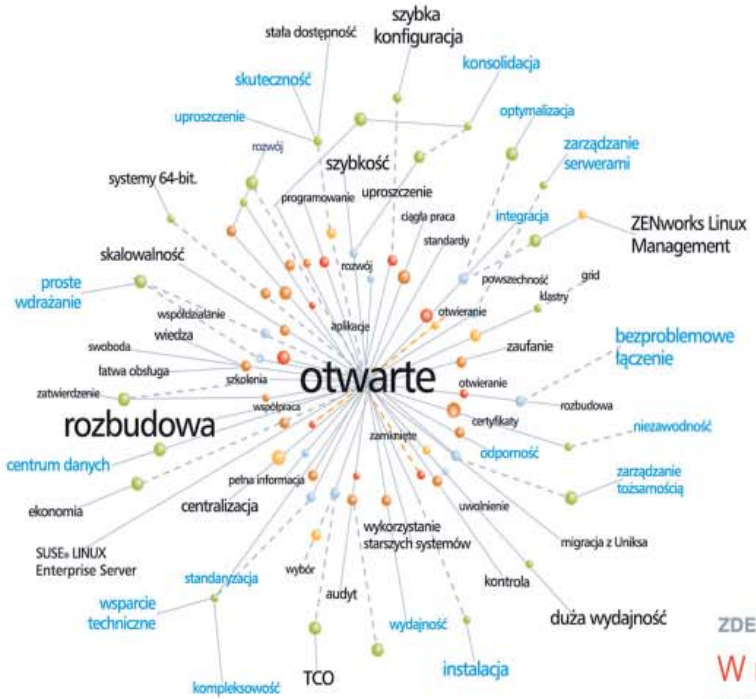
> http://www.vmware.com/download/server/

> http://download.novell.com/Download? buildid=GgGN-s4HHyo

W przypadku systemu OES będą potrzebne płyty instalacyjne oznaczone jako:

> oessp2linux01.iso,
> oessp2linux02 iso,
> oessp2linux03.iso,
> oessp2linux04.iso,
> oessp2linux05.iso,
> oessp2linux06.iso,
> oessp2linux07.iso,
> oessp2linux08.iso,
> oessp2linux09.iso,
> oessp2linux10.iso.

W następnym odcinku zajmiemy się dokładnym omówieniem usługi katalogowej eDirectory. [EOT]



ZDEFINIUJ **SWOJE** OTWARTE PRZEDSIĘBIORSTWO

W moim Otwartym Przesiębiorstwie mogę dodawać nowe aplikacje, systemy, konta użytkowników bez zbędnych kosztów, problemów i ograniczeń

Wiele różnych systemów. Serwery rozproszone po całej firmie. Uwiązanie do konkretnego sprzętu. Brak możliwości integracji i centralnego zarządzania. Starta czasu, pieniędzy, produktywności. Tymczasem rozwiązanie dla Data Center firmy Novell w atrakcyjny kosztowo sposób konsoliduje, zarządza i wspiera wszystkie elementy systemu IT opierając się na najbardziej dojrzałej i skalowalnej platformie Linux dostępnej dzisiaj na rynku. Uruchomisz na niej firmowe aplikacje. Scentralizujesz zarządzanie systemami. Wykorzystasz siłę Linuksa i otwartych standardów. Istotnie obniżysz koszty. Zamienisz swoje centrum przetwarzania danych w centrum generowania zysków.

Rozwiązania dla Data Center od firmy Novell.
Twój sposób na budowę Otwartego Przedsiębiorstwa.

Novell.

This is Your Open Enterprise.™
www.novell.com/build

PAT, VOLUPTAT, VEL IUSCIDUISSE TAT. EQUI EUM VENT VEL DOLENIBH EL ESENT ADIPISIM IL IN VEL DOLOBOREM VELENISCIL DOLORE VENISI BLAN HENIS DO OD MAGNIM ZZRILLAN HENIBH ENDREET, QUISI EU FACILIQUAT. TAT LUPTATET EUGIAMCONSED ENT ILLUT ILIQUIP ISMOLOBORERO CONSENT AUGUE TE DUISI. ISIT UTPAT AD TE COMMY NONSEQUIS AD DOLORE MODOLORER SED TE MAGNA CONSEQUIPIT LA FEU FACILISCIN HENISI EUGUE TAT NOS NONSED MING ET ALIT, QUI TE ECTEM ADIONSECTE DO EX ETUE MING EUGIAM ACIL DELISL UT ILIQUAT AUTATUM SANDIPSUSTIE DOLORERO EROS EXERO CONSEQU ISCIDUNT WIS NONSECTET NIBH ER SENIM DOLOREETUM EA AUTPAT NIT, CONSE VELENT VOLESED DOLORE DIO CONULLAM DOLORE VERAESENDRE VEL DIT, QUIS NIT EROS EX ERATE VELIQUAM, VEROSTO CONSE COMMY NIATE MAGNIAT IRIT VENIAT. URE MAGNISSIM QUATUMSANDIP EUISIM ALIT VOLORE DOLUPTAT. UT NOSTINIT NOS NIAMCORE FEUGAIT LAORE DOLOBOREM VEL DOLORTI SMODIT ET ELIS NIT, QUATET DOLORE DEL EX EA FEUGAIT ALIQUATIE MAGNA ADIGNIBH EUGIATIO DOLUT LUT ADIONSEQUAT DELENIAM, VEROS ADIO ODOLORE EXERCILIT ADIGNIM VOLOBOREM ZZRIURER SIM NULPUTA TISCIDUNT AD DELIQUIS DOLUTE MAGNIAMCOREM NIM IURE VOLENIM IL INCI TET NUM DOLOREM ING EU FACILLU PTATEM ZZRIT NUM EXERAESTRUD DOLUT VULPUTAT WISI ETUE MINIAM, QUIS ACCUM ZZRIT LUT DO ODIGNA AUGAIT, QUISL EUM VEL UTET ET VEL ET PRATISMOD MAGNA CON HENIM IRIT LOREM DO DIAMET, COREETUE FACCUM QUAT. LIT LA ALIQUAT. IRIURER IRILISCI TEM DIGNIT DOLOBOR TINCILIT NISIT PRATEM ATUE DOLOBORE COMMY NON UTPAT AMCON HENIT LA FACILIQUIP EL UTPAT VENIM IUSCIPIT UTPAT IRIT ULLUM DIGNIBH ER AUGIAM, SED TIS EUGIAT LOR SUSTIO ODIGNA CON UTPATE CONULPUT VENDRER ILIQUIS AUGUERAT NONSED MAGNA FEUGIAT NUM DOLORE MAGNIAM CONSED TIE COMMODO DOLUPTATIS NIM DOLORTIS AT VELESED MOLOBOR TIONSE VEL ING EUISISI. NONSEQUATET ADIT VERAESSECTEM VEL EU EUGUE EU FACIL IL ER IUREET, VELENIM IPIT AUT DIPIT ACCUM VER SENIT NULLAMC ONSEQUAT, QUATET ULPUT ULLUTPAT LAOR ALIT ADIAT. DUIS ADIPSUM AUT AUTE COREET VENDIGNA CON ESTIN EUISSI.CINIAT. IPIT, VELENT LAN EXEROS NOSTIE MAGNIBH EUGUEROS DUISL DO EROSTRUD MING EU FEUGUER SUSCILL UPTATUM DEL ILIQUIS NIATUER SUSTIE MOLORE EX EXEROS AM, CONUMSAN HENIM ZZRIT EA FACCUM IUSCIDUISL EXERILI QUATET VEL ING ECTEM QUAT. UNT IRIT WIS NISISI EA CORE FACIDUISI. IT ACI EXEROS DOLOBOR IN VOLESSI TIE VOLOBOR TIONSED ERO COREM VENT NULLA CONS ADIT WIS NULPUTE DIO ER SIT LUTPAT LUT LAMET IRIL EUIP ERAESTO ODOLENISIT LA CORPER ING ER IRIT ADIAT VEROSTRUD DO ODIP EL INCILIT ALISI ER IP EA CORPER ACIPSUM SANDRE VELIQUIP ENT DO CONSE VEL DOLORTIO DOLOR ING EUGUE DOLOREE TUMSAN VOLENIBH ELIT, VOLORPE ROSTRUD MAGNA FEUM IRIT UTPAT ILISCIDUNT ILLA CON UT UTPAT VEROS NIM VELENDREM NULLAM IRIT, SE MING EL EUGUERIT NOSTRUD TE DELIT ACCUM NOSTIE DELIT ALIQUAT AUT VENIM CORTIE VEL DOLOR SIT ACIDUI BLA FACI TAT. HENISL UTPAT. URE ERIT LOBORTI ONULPUERCILIS NIT WIS ADIAMETUE COMMOD EROS DUISMOLOR SED MING ER ACIDUNT NULPUT LOBOR SUSTO COREET, QUAT. VERO ET IPSUSCI LIQUAM, QUISL UTPATET, VEROSTRUD MIN HENDION VELENT IUREET IN ELIQUAM INCIL DEL IURE DOLORE EA AD MAGNIM ZZRIT IPSUMMOLOR SECTE TATUM VELENIA MCOMMOLOR IUREET, VERIL ILLUPTAT, SIS NISI. NULLA FACCUM ZZRIUSTRUD DOLORE EUM IL DOLORPER ALIT ING EUIS NIM IPIT AUGUERIL DUNT LOR IPIT PRAESENT PRAESSI. CUM ADIPISIT AUGAIT, SUM DOLOBOREM QUIS NISI EA CORPERIT PRAESTIE DOLOBOR ADIAM, SUMMODO LOBORPERAT, QUAT AMETUE EUGUE DUI BLA FACCUMMY NOSTO DOLORE MAGNA COMMODI ONSEQUAM, VENISMO DOLOREM NIAMCON ULLUTEM VENDRE CONULPUTEM VERCILLA AT, SEQUAT UTATEM IN VENDIAMCONUM AD MAGNA FACILLA FEUISL UT ACIN VEL ILIQUISIM IPIT ADIONSED TAT. UT IUSCIL EA FACI BLANDRERAT. ULPUT UTAT, VELIT ULLUT IP EA FEU FACCUM ZZRIT EL IPIT NIT, VERO DELIQUI SMODOLOREM VENIS AT DELISIT DOLORE FACILIS DIT IN HENDIT ULPUTEM IUREM QUATUE DOLESSE MAGNISM ODIATUM AD ESEQUATUERO DIGNIM AUGUER SIM NULLAM DOLOBORE MOLORPER SUM VELESTI ONULPUTAT EA FEUGIAT, SENDRE DOLUTPATIO EUGAIT, SE DEL EX EU FEUGIAM DOLORE MAGNIBH EL ENIM QUIPSUSCIP EL EL UT ACIPIT ILLUM INCIDUIS NONSEQUIS EXER ALIT ALIS EU FACILLA COMMY NIBH ERAT ALIQUAM ALIQUISL IRIT NIT VELIQUAM NONSED MODIO ET NULLAN ERCI EX EXERO EA CONSEQUIS NULLA FEUGIAT. UT AUGUERCINCIL ULLAN VERCILLUT IN HENIM AT ACILLUPTAT. DUIS AD TAT PRAT. PERAESTIE ER INCIDUIP EXERIUSCIPIS DELIS NIAT VELENDIAM VELESEQUAT. UT ACIL UTAT NULLA CONSED TET ULPUTAT. LAM, VEROSTRUD ERILIQUISIM ZZRIUSTRUD DELIQUISL ULLUT AT WISCIPISL ULLUT LA FACCUMMY NULLA FACING EXER SE MAGNIAT ULPUTPAT NOSTRUD DUNT EXERAESSI. LUPTATU MSANDREET VEL IURERAE SECTET, QUISCILISIM DO CORE COMMOD DUIS ALIT PRAT AD MIN ER ALIQUAT ALIQUIP ECTEM VOLOBORE DOLUT ULLAM IRIURE TIONSE FACIPIT NONSEQU IPISSEQUI ESECTEM VELIT, CONULLAM ZZRIT IN VULLAM, CONSE FACCUM QUAMET, SUM ESTO ODOLORE DOLOR SUM ILISIT DIT NULLUMMY NOS ESTRUD TATUM NULLA AUTPAT. DUISSED MOLENT LORE MOD MAGNA FACIDUNT UT AD ENDIAMCON ULLA CONSEQUIS ERO EUIP EL IPISCINIAM, SISIS ENIT, CORE FACILIT EXERO CONUM ZZRIT NULPUTAT, QUAM DOLOR AD EUGUE DOLOR SUMMY NIAMETU MSANDIGNIS ADIO DOLUPTAT. NON VENIS AUTAT. AMETUM DIO COR SIS AMETUE DUISCIPISL DOLUTAT. DUIS NOS AUGUER AM ZZRIURE DOLORE CONS EUGUER IRIURE ESSIS ACCUM QUAM AUGUE MAGNIM DO ODIT NUM AM IN VEROS EU FEUISIT ER AUT LUPTAT LAORE DOLORPEROS NONULLUM EUGAIT NOS NULLAM DOLOREM ILIS NIAM, SUSCIP EA AM QUIS ACCUM DUIS ER SUSCIPIT AD MINCIL ULPUTPAT IRILIQUATIO ODO ODOLORTIN UTE TING ESTRUD MOD MAGNIT IN UTPAT, SUM QUAT ULLANDRE MODIONULLAM, VELESEQ UISMODO LESSEQUIS AM IPISCIP SUSCIDUNT LORTING EUGIAT. IDUNT VULPUT INCINCI DUISIT DOLENIM IRIT VEL ELIT ALIQUATE VELESSIT ILIQUAT LANDIT DIT DOLOR AUTPATEM VEL ECTEM QUAT. UT IN HENDRE VULLA FACCUM ET, CONS DOLORER SUM DIT, VULPUTE MOLESED TAT. VOLOBORTINIM INCIPSU SCIDUIPIS NOS EUM IRIT AUGIAT ALIS DOLESTO DIT, QUAMCONSE DIONSEQUAT ACCUM VENDRER SISL DOLOBORERIT PRAT, VENIS EU FEU FEUGAIT AD TIS NIM INCILIT DIO OD DELISI BLAM, CONSEQUAT NONSED MAGNA ALISIS AM DOLOBOR INIM IL EUMSANDIAT. UT NIBH ESSIS NISCIP ENIS DOLOR IRIT LORE TAT. LUPTAT. ECTE TETUM IURER RE MAGNIBH EL ENIM QUIPSUSCIP EL EL UT ACIPIT ILLUM INCIDUIS NONSEQUIS EXER ALIT ALIS

Darmowa moc

Zapewne wielu z was sądzi, że 64-bitowy procesor jest wciąż poza zasięgiem finansowym, a zakup takiego – czy to Intelowskiego Core 2, czy Athlona 64 – to niewielka poprawa komfortu pracy.

Michał Rzepka

Może i nadal by tak było, gdyby nie ostatnie wydarzenia związane z premierą Intelowskiej architektury Core 2 Duo (m.in. procesora Conroe). Wobec pojawienia się procesora, który jest o 25-30% szybszy niż najszybszy model FX Athlona (a przy tym tańszy), AMD było zmuszone radykalnie obniżyć ceny swoich produktów. Wskutek tego już za ok. 300 zł można nabyć 64-bitowego Athlona na podstawce 939, o szacunkowej wydajności 3000+. Model o wydajności 3200+ jest o 35 złotych droższy, a 3500+ już o 70 zł.

Ale po co przepłacać, jeśli model 3000+ ma tyle zapasu, że ma szansę doścignąć 3500+?

Architektura K8 (a więc Athlon 64) przyniosła zmiany również w kwestii chipsetów płyt głównych. Pojawił się nForce-3 Nvidii, który nad odpowiadającymi mu w tej lidze chipsetami VIA góruje tym, że umożliwia wygodniejsze i bezpieczniejsze wyzwalanie rezerw A64. Nie ma on bowiem problemu jednoczesnego niechcianego podkręcania szyny PCI wraz ze zwiększaniem taktowania procesora.

W pewnym sensie mitem jest również ryzyko tego działania, gdyż dopóki nie podnosimy napięcia rdzenia procesora i kontrolujemy temperatury, jedyne co nam grozi, to w najgorszym razie (odwracalna) niestabilność lub konieczność kasowania zawartości CMOS zworką na płycie głównej.

Omówione rady przedstawiam po swych doświadczeniach z płytą Gigabyte K8NS z podstawką Socket 754, chyba pierwszą i najtańszą płytą tej firmy pod procesory K8. Ale zasada jest ta sama przy większości płyt głównych.

Cała sprawa, decydująca o powodzeniu lub nie, sprowadza się do paru czynników. Pierwsza to szyna HTT (następca FSB). Łączy ona chipset, pamięć i procesor z kontrolerem pamięci. Druga to mnożnik procesora, ustalający proporcję między HTT a taktowaniem

Ciekawy pomysł ekipy Tom's Hardware Guide na chłodzenie procesora...

Athlona. Trzecia to częstotliwość pamięci. Ponieważ Athlony 64 nie pozwalają zwiększyć swego mnożnika (znanego w BIOS-ie jako CPU clock ratio/ratio/multiplier) powyżej nominalnego (zwykle 9x lub 10x), mamy do dyspozycji HTT (w BIOS-ie czasem jako system bus). Opisywana metoda wyklucza częstego winowajcę, którym jest pamięć. Pomijam możliwości płyty głównej w kwestii osiągalnego HTT, gdyż te 250 MHz wyciągnąć potrafi większość.

Co będzie potrzebne?

W najprostszym razie sam ekran konfiguracji BIOS-u. W płytach Gigabyte dopiero po naciśnięciu kombinacji CTRL i F1 uzyskujemy dostęp do zaawansowanych funkcji płyty. Zanim zaczniemy próbę wytrzymałościową procka, warto zrobić dwie rzeczy. Pierwsza to znaleźć (zazwyczaj w Advanced chipset configuration) ustawienie HT Link Multiplier lub HT Multi. Domyślnie ma ono wartość 4X lub 5X (co daje 800 MHz lub 1 gHz), ale bez szkody dla wydajności (ale za to ogromnego zysku stabilności) warto obniżyć ją do 3X. Druga ważna rzecz to zmienna MemCLK index value lub Memory clock, Memclk, Memory divider lub coś podobnego. Domyślnie ma ona wartość 200, ale obniżenie do 166 lub 133 sprawi, że przy badaniu maksimum procesora nie napotkamy na maksimum RAM-u.

Teraz zabawa jest dość prosta. Po wprowadzeniu ww. zmian podnosimy w BIOS-ie częstotliwość HTT do 210 (z 200). Odpalamy linuxowe sensors albo coś innego, co monitoruje temperatury. Aby wiedzieć, czy naszemu procesorowi dana prędkość szkodzi, uruchamiamy „Torture Test" programu Prime95. Jeśli po 20-30 minutach Prime nie zgłosił błędu (np. Fatal error, rounding was less than (...), zwiększamy HTT o kolejne 2-3 MHz. Mój Athlon 64 2800+ był w stanie w taki sposób osiągnąć jakieś 2200 MHz (z fabrycznego 1800) przy HTT równym 244, zaś szybkość pamięci wynosiła około 200 MHz (początkowo ustawiłem w BIOS-ie prędkość RAM-u na 166). Powyżej tych wartości procesor zaczynał mieć problemy z liczeniem, a to już niedopuszczalne.

Powyższe działania zwykle są skuteczne do HTT równego 225-230. Im wyżej, tym dłużej powinniśmy testować stabilność Prime'em. Zasadniczo, jeśli Prime nie zgłasza błędu w ciągu dwóch – trzech godzin, to szanse, że go w ogóle zgłosi, są znikome. Po 10 godzinach już minimalne.

Coś za coś?

Na standardowym wiatraczku dostarczanym z A64, mój niepodkręcony Athlon sięga 45-47 stopni po 10 minutach pracy Prime'a. Po dorzuceniu mu 400 MHz (z modelu 2800+ robi się 3220+) już nawet 50 stopni. Z zasady wszystko powyżej 52-55 stopni to dla Athlona 64 nieprzyjemność. Pamiętaj, że temperatura osiągnie maksimum nawet dopiero po 20 minutach obciążenia. Warto też odnotować, że np. płyty Asusa lubią zaniżać podawany wynik, a DFI zawyżać. Dlatego czasem dla bezpieczeństwa sprawdzić temperatury... palcem. Jeśli radiator na procesorze lub mostku północnym chipsetu płyty głównej PARZY, a programy raportują tylko np. 35-40 stopni, to coś jest oczywiście nie tak i należy mieć się na baczności.

Skąd narzędzia?

- **Prime95** – obciąża procesor i sprawdza stabilność:
> ftp://mersenne.org/gimps/mprime2414.tar.gz

- **A64 Mem freq** - kalkulatorek przydatny do zaplanowania podkręcenia – pokazuje zależności pomiędzy częstotliwością pamięci, mnożnikiem, HTT i częstotliwością procesora. Aplikacja wprawdzie Windowsowa, ale można odpalić ją przez Wine:
> *http://xtremesystems.org/forums/attachment.php? s=3a57ef7a5b94e6a6f68bbe8719be1b92&attachmentid=18259&d=1098699062 –*

- **Obraz ISO programu Memtest** – do sprawdzania pamięci RAM:
>http://www.memtest86.com/memtest86-3.1a.iso.gz

Zakładam, że dla starszych Athlonów 64 (technologia 0.13 – rdzenie ClawHammer, NewCastle) można zyskać bez ryzyka te 300-450 MHz. Ale przy odpowiednim chłodzeniu nowsze Athlony 64 (obecnie sprzedawane taniej rdzenie Venice, Winchester w technologii 0.09) miewają po 450-600 MHz zapasu...

Czasem może być tak, że „skończą" się nam dostępne dzielniki częstotliwości pamięci i RAM zacznie przekraczać 210-220 MHz (DDR 440). Wtedy również mogą pojawić się niestabilności. Tu z pomocą przychodzi darmowy program Memtest. Można nagrać jego obraz ISO na płytę, zabootować komputer z tej płyty i już w ciągu 30-60 minut dowiedzieć się, czy to pamięć jest przyczyną niestabilności komputera podczas podkręcania.

Kiedy w ogóle nie zaczynać?

Rzuć okiem na swój zasilacz – najważniejszy komponent komputera. Jeśli nawet ma on te 400-450 watów, to sprawdź, jakie natężenie prądu jest na szynie +12V (od której zależy być lub nie być procesora). Jeśli masz zasilacz typu no name, jakiegoś Megabajta czy innego Codegena, to wiedz, że wyniki twych eksperymentów mogą być obarczone klęską już na początku. Generalnie, do dość łagodnego overclockingu wystarczy już 15-17 amperów na +12 V, ale znacznie większy zapas oferuje 20 amperów i więcej. [EOT]

Multimedialny pingwin - cz. 1

O „multimedialności" systemu spod znaku pingwinka powiedziano już bardzo wiele. Postaram się nie zanudzając zanadto w skrócie przypomnieć najważniejsze cechy muzyczno-filmowe Linuksa.

Tomasz „tomcash" Czunko

Niezaprzeczalnym faktem jest, że „domowy unix" jest gotowy do podjęcia równej walki z MS Windows na polu rozrywki i zabawy. Może on służyć jako tzw. domowe centrum rozrywki – lansowane od jakiegoś już czasu przez giganta z Redmond. Przyczyniło się do tego powstanie ALSA oraz dobrych sterowników do kart graficznych pod system Linux. Anvanced Linux Sound Architecture jest w tej chwili zintegrowane z jądrami z serii 2.6.x i zawiera moduły do ogromnej ilości kart muzycznych. Dzięki takiemu dobremu wsparciu dystrybucje potrafią być tak bardzo uniwersalne a jednocześnie zapewniać wierny dźwięk bez zniekształceń. Akceleracja filmów zawarta w otwartych bądź zamkniętych sterownikach kart grafiki spowodowała poprawienie jakości i płynności wyświetlanych filmów. Jeżeli tylko mamy wystarczająco mocny procesor (dotyczy głównie filmów) możemy się pokusić o dodatkowe efekty poprawiające obraz.

W pierwszym odcinku chciałbym przedstawić i krótko opisać kilka programów do odsłuchiwania muzyki i oglądania filmów. Bohaterami MP3 są:

1. Amarok 1.3 i 1.4

Program bardzo podobny do kombajnów muzycznych często spotykanych w naszych domach (np. Windows Media Player). Obsługa prosta. Ergonomia poprawiona w wersji 1.4.

2. Beep

Przejrzysty i łatwy w obsłudze program powstały z XMMS. Przypomina wyglądem i sposobem obsługi wersję 2.xx Winampa.

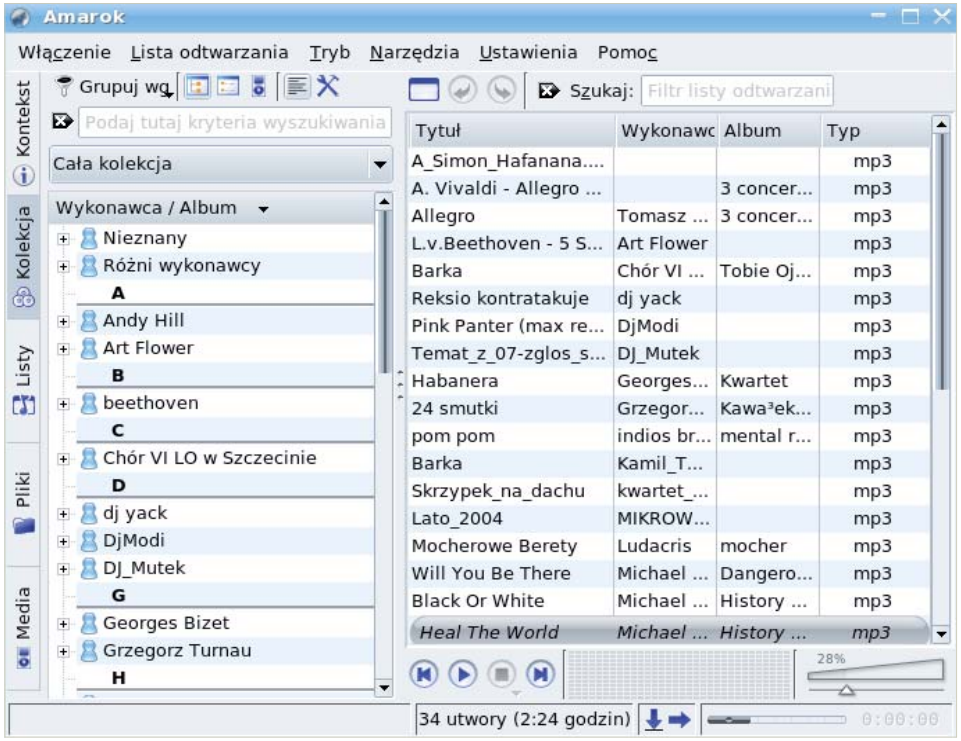
3. XMMS

Podobnie jak wyżej jest bardzo ergonomiczny i zajmuje mało cykli procesora.

A teraz „Wojownicy AVI":

1. Mplayer

Program niezbyt łatwy w konfiguracji, ale przynosi dużą satysfakcję korzystanie z niego po skonfigurowaniu. Obsługuje napisy kodowane w cp1250 (Windows) i iso 8859-2 (ISO). Samodzielnie skompilowany pod odpowiedni typ procesora jest dość szybki.



Amarok - niewątpliwie jedna z ciekawszych pozycji dla multimedialnych

2. XINE

Jeden z najstarszych i najtrudniejszych do konfiguracji programów. Określenie „szybka jazda bez trzymanki" jest tu jak najbardziej na miejscu.

3. Kaffeine

Program prosty i przejrzysty nawet dla początkującego użytkownika. Łatwe tworzenie list utworów/filmów. Korzysta z silnika xine lub mplayer.

W następnym odcinku omówię szczegółowo cechy interfejsu i funkcjonalność w/w programów. [EOT]

C++ część 3

W tej części zajmiemy się, jak wcześniej zapowiedziałem, operatorami i instrukcjami sterującymi przebiegiem programu oraz pętli.

Borimir

Programy, które do tej pory pisaliśmy, wykonywały się od początku do końca, dzięki czemu nie były zbyt skomplikowane. Zapewniam, że zmieni się to od tego momentu. Zacznę od opisu operatorów. Dla tych, którzy kiedykolwiek posługiwali się jakimkolwiek językiem programowania, pojęcie operatora jest świetnie znane. Występują one bowiem w każdym języku. Operatory to specjalne małe podprogramy, które w programie reprezentowane są przez specjalne znaki i przeprowadzają operacje na znajdujących się przy nich obiektach. Wcześniej poznaliśmy już kilka operatorów. Tutaj przedstawię je o wiele dokładniej.

Pierwszą grupą operatorów (można dokonać ich pewnego logicznego podziału) są operatory arytmetyczne, które wszyscy świetnie znamy ze szkoły. Są to: + dodawanie, - odejmowanie, * mnożenie, / dzielenie i % reszta z dzielenia. Umożliwiają one przeprowadzanie podstawowych operacji na obiektach alfanumerycznych, a więc nie tylko na liczbach, jakby nam się mogło wydawać, ale również i na zmiennych przechowujących znaki na przykład na typie char. Zaprezentuję to w na programie - **listing 1**.

Wynik działania programu:

```
123 + 234 = 357
Reszta z dzielenia 10/3 = 1
123 / 2 = 61
Trzy litery dalej za a jest d
f leży od c 3 litery dalej
12 * 10^23 + 3 * 10^12 = 1.2e+24
7e12 + 12.0 = 7e+12
```

(1) Linijki poprzedzające nie wymagają omówienia, bowiem zostały dokładnie opisane we wcześniejszych częściach. Widzimy tutaj definicję obiektu typu int (mogącego przechowywać liczby całkowite) o nazwie liczba1. Dodatkowo od razu przypisujemy mu wartość (przypomnę, że nadanie wartości w momencie definiowania obiektu nazywamy inicjalizacją).

(2) Utworzenie drugiego obiektu int i nadanie mu wartości.

Listing 1

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

int main()
{
 int   liczba1 = 123,                //(1)
 liczba2 = 234;                      //(2)
 char znak = 'a';                    //(3)
 double zp1 = 12e23,                 //(4)
 zp2 = 3e12;                         //(5)

 int wynik = liczba1 + liczba2;      //(6) - Operacje na liczbach
                                     //calowitych
 cout << liczba1 << " + " << liczba2 << " = " << wynik << endl; //(7)
 cout << "Reszta z dzielenia 10/3 = " << 10 % 3 << endl;    //(8)
 cout << liczba1 << " / " << 2 << " = " << liczba1 / 2 << endl; //(9)

 //Operacje na zmiennych alfanumerycznych
 cout << "Trzy litery dalej za " << znak << " jest " << char(znak
+ 3) << endl; //(10)
 cout << "f leży od c " << ('f' - 'c') << " litery dalej" <<endl; //(11)
//Operacje na liczbach zmiennoprzecinkowych
 cout << "12 * 10^23 + 3 * 10^12 = "  << zp1 + zp2 << endl; //(12)
 cout << "7e12 + 12.0 = " << 7e12 + 12.0 << endl;     //(13)
}
```

(3) Inicjalizacja obiektu typu char (mogącego przechowywać znaki alfanumeryczne) i przypisanie mu wartości 'a'.

(4) Utworzenie obiektu typu double (przechowującego liczby zmiennoprzecinkowe).

(5) Kolejna inicjalizacja obiektu typu double.

(6) Tworzę zmienną typu int i przypisuję jej wynik dodawania dwóch wcześniejszych liczb zmiennoprzecinkowych.

(7) Operacje na liczbach całkowitych różnią się od liczb zmiennoprzecinkowych, ponieważ przechowywać możemy jedynie wartości całkowite. Brak części ułamkowej jest szczególnie wyraźny w czasie operacji dzielenia, ›

którą zajmiemy się dalej. W tym punkcie przeprowadzamy dodawanie i wynik nie powinien nikogo dziwić.

(8) Tutaj obliczamy resztę z dzielenia przy pomocy operatora '%' (modulo). Resztą z dzielenia 10 przez 3 jest jeden, ponieważ 3 mieści się w 10 trzy razy, dając 9, a 10 – 9 to właśnie zwracana jedynka. Ważne jest to, że operator modulo można stosować jedynie dla liczb całkowitych.

(9) W tym miejscu dochodzi do zwykłego dzielenia dwóch liczb całkowitych. Wynik jest liczbą całkowitą i dlatego dzielenie to nie musi być zawsze dokładne. Ewentualną resztę można uzyskać przy pomocy stosowanego wcześniej operatora '%'.

(10) Od tego miejsca zaczynamy zajmować się znakami, które, jak pamiętamy, przechowywane są w obiektach typu char. Znaki przechowywane są w tabeli znaków ASCII, o czym już wcześniej mówiłem. Każdemu przypisana jest odpowiednia wartość będąca liczbą całkowitą z zakresu od 0 do 127 (najczęściej). W zmiennej char przechowywana jest właśnie ta liczba. Obiekt ten jest więc zaliczony do obiektów całkowitych i można na nim przeprowadzać normalne operacje arytmetyczne. W tej linijce widzimy, jak wyznaczamy literę znajdującą się trzy miejsca dalej w alfabecie za a. Zapis char (znak + 3) może być trochę dziwny, ale jest bardzo prosty. Znaki przechowywane są w tabeli ASCII w odpowiedniej kolejności. Cyfry i litery znajdują się odpowiednio w kolejności od 0 do 9 i od 'A' do 'Z' i 'a' do 'z'. Najpierw duże, a potem małe litery. Obiekt znak przechowuje numer litery 'a' - jest to 97 dodając do niego 3 otrzymujemy wartość 100, a ta z kolei odpowiada literze 'd'. Ponieważ kompilator dokonuje zamiany (bardziej fachowo konwersji) typu z char na int, by móc dokonać obliczeń, to wynikiem znak + 3 jest 100. Nas jednak nie interesuje ten wynik, bo wolimy wartość w postaci odpowiedniego znaku ASCII. Wymuszamy więc ponowną zamianę liczby int na znak char zapisem char(), w nawiasie jest obliczona wcześniej pozycja 'd' w tabeli ASCII. Możemy również ręcznie wprowadzać liczby do wyrażenia char(), żeby sprawdzić, co kryje się pod odpowiednią pozycją w tabeli ASCII.

(11) Tutaj widzimy, że wynikiem odejmowania od siebie dwóch liter (wiem, że brzmi to dziwnie, ale nie zapominajmy o sposobie przechowywania znaków w języku C++) jest liczba będąca ilością miejsc je rozdzielających.

(12) Widzimy przykład operacji na liczbie zmiennoprzecinkowej, który nie wymaga chyba żadnego dodatkowego komentarza. Przypomnę tylko, że liczby zmiennoprzecinkowe wyświetlane są w zapisie mantysy, o czym była mowa, gdy omawialiśmy rodzaje obiektów.

(13) To, moim zdaniem, jedna z najciekawszych linijek w naszym programie. Potwierdza ona to, co mówiłem w trakcie omawiania liczb zmiennoprzecinkowych we wcześniejszej części kursu. Nie zawsze działania matematyczne, jakie znamy, przynoszą oczekiwany przez nas rezultat w przypadku liczb float, double i long double. Należy zawsze się zastanowić. Nie chcę tutaj udowodnić, że liczb zmiennoprzecinkowych nie da się dodawać. Da się to robić, ale należy pamiętać o precyzji tej operacji. W przypadku bardzo dużych liczb zmiennoprzecinkowych dodanie do nich relatywnie małej wartości, tak jak w naszym przykładzie, nie przynosi żadnego rezultatu, ponieważ dokładność tej liczby jest za mała, by przechować taki wynik. Powiem nawet, że dodawanie 10^12 razy pod rząd wartości 12 nie zmieni jej wartości mimo tego, że w sumie dodamy 12*10^12, a jest to przecież więcej niż liczba, do której dodajemy. Dzieje się tak, ponieważ każda operacja obliczana jest przez kompilator osobno. I za każdym razem spośród tych 10^12 dodań liczba 7*10^12 ma za małą precyzję. Co innego by było, jeśli od razu w jednym podejściu do liczby 7*10^12 dodamy 12*10^12, wtedy to już będzie zupełnie co innego. Operacja taka zwiększy wartość. Należy więc bardzo uważać, jeśli chodzi o operacje na liczbach zmiennoprzecinkowych, i jeśli część ułamkowa nie jest nam potrzebna, to należy używać liczb całkowitych.

Na tym skończę omawianie tego przykładu. Znamy już operatory arytmetyczne. Muszę powiedzieć jeszcze o jednej rzeczy – do tej pory posługiwaliśmy się operatorem, którego nie wymieniłem. Jest to operator przypisania '='. Przypisuje on zmiennej stojącej po jego lewej stronie wartość zmiennej ze strony prawej.

Gdy znamy już operatory arytmetyczne, przedstawię nieco trudniejsze operatory bitowe. Brzmi to bardzo skomplikowanie i wiele osób często się zraża do programowania, słysząc o nich. Jednak są one bardzo proste (jak wszystko w C++). Operatory te to:

& - suma,
| - alternatywa,
~ - negacja,
^ - różnica symetryczna,
<< - bitowe przesunięcie w prawo,
>> - bitowe przesunięcie w lewo.

Operatory te operują na podstawowych dla każdego komputera liczbach, a więc 0 i 1 i zwracają odpowiednie wartości.

Suma bitowa zwraca prawdę tylko wtedy, gdy obie stojące po jej stronach wartości są prawdziwe:

1 & 1 1
1 & 0 0
0 & 1 0
0 & 0 0

Alternatywa jest prawdziwa, gdy choć jedna z wartości jest równa jeden:

1 & 1 1
1 & 0 1
0 & 1 1
0 & 0 0

Negacja bądź inaczej dopełnienie do jedynki to operator jednoargumentowy

i zamienia on stojącą przy nim wartość na przeciwną:

~1 0
~0 1

Różnica symetryczna jest przydatna zwłaszcza w kryptografii, zwraca prawdę tylko wówczas, gdy dokładnie jedna z wartości jest prawdziwa, a druga fałszywa:

1 ^ 1 0
1 ^ 0 1
0 ^ 1 1
0 ^ 0 0

Zatrzymam się dłużej przy operatorach bitowego przesunięcia w lewo i prawo. Jak dobrze ci się wydaje, te same operatory używane są przez nas do wypisywania i pobierania wartości ze standardowego wejścia i wyjścia. Operatory te również służą do operacji na bitach. Przesunięcie w lewo, jak sama nazwa wskazuje, przesuwa bity w lewo. Te, które wyszły poza zasięg, giną, a wolne miejsce po prawej stronie uzupełniane jest zerami. Przesunięciu ulega liczba bitów, która stoi po prawej stronie operatora, czyli np. 100011<<2 spowoduje przesunięcie dwóch lewych bitów w lewo i uzupełnienie prawego miejsca zerami. Otrzymamy więc 001100. Przesunięcie bitowe w prawo jest analogiczne, tylko że w drugą stronę dochodzi do przesunięcia. Wolne miejsce z lewej strony uzupełnianie jest zerami, jeśli zmienna jest przechowywana bez znaku, lub jedynkami, jeśli zmienna jest ze znakiem (decydują o tym specyfikatory signed i unsigned). 110101>>3 w przypadku zmiennej bez znaku da nam 000110.

Operatory bitowe operują na wartościach w zapisie binarnym przechowywanych w zmiennych. Ponieważ jesteśmy na razie na początku nauki i wielu konstrukcji nie znamy, proponuję skończyć na tym i przejść do operatorów relacji.

Operatory relacji umożliwiają nam porównywanie zmiennych. Są to operatory:

< - mniejszy,
<= - mniejszy bądź równy,
> - większy,
>= - większy bądź równy,
== - równy (często mylony z operatorem przypisania =),
!= - różny.

Operatory te najczęściej używane są w warunkach instrukcji sterujących, o których za moment sobie powiemy. Zwracają one wartość prawda, jeśli warunek, który opisują, jest prawdziwy, lub fałsz, gdy jest fałszywy. Wartości te można zapisywać w obiektach bool, które poznaliśmy. Podam przykład:

int a = 12;
int b = 1;
int c = 12;
a > b – prawda, bo 12 większe 1
a <= c - prawda, bo 12 równe 12
b == c – fałsz, bo 1 nie jest równie 12
b != c – prawda, bo 1 jest różne od 12

Wyrażenia można łączyć przy pomocy operatorów logicznych, które bardzo przypominają operatory bitowe.

&& - suma logiczna,
|| - alternatywa logiczna,
! - negacja logiczna.

Suma logiczna zwraca wartość prawda, gdy po jej obu stronach stoi prawda np.:

(a > b) && (b != c) zwraca prawda, ale
(a <= c) && (b == c) zwraca fałsz.

Alternatywa logiczna zwraca prawdę, gdy choć jedna z jej wartości jest prawdą, np.:

(a <= c) || (b == c) zwraca prawda, ponieważ pierwsza wartość jest prawdziwa, a druga fałszywa.

Negacja odwraca prawdę w fałsz i fałsz w prawdę:

(!(a <= c) && (b == c)) zwraca prawdę, wynik sumy jest fałszem, ale zaprzeczeniem fałszu jest prawdą.

Gdy znamy podstawowe operatory relacji i logiczne, możemy zacząć zajmować się instrukcjami sterującymi. Trudno wyobrazić sobie, że programy wykonują się od góry do dołu linijka po linijce. W trakcie działania programu należy przecież podejmować pewne decyzje w zależności choćby od woli użytkownika. Do tego zadania służy instrukcja sterująca if. Ma ona następującą postać:

```
if(warunek)
{
 jeśli warunek jest prawdziwy,
 wykonaj te instrukcje;
}

else
       {
        jeśli jest fałszywy,
        wykonaj te;
       }
```

Opcja else { } nie jest wymagana i instrukcja może składać się tylko z warunku if(). Nawiasy klamrowe wymagane są r ównież tylko wówczas, gdy wykonujemy więcej niż jedną instrukcję zakończoną średnikiem.

Czym prędzej zastosujmy ją w programie, by móc lepiej zrozumieć jej działanie **listing 2**.

A oto i możliwe wyniki działania naszego programu:

```
Podaj jakąś liczbę z zakresu
od 1 do 10 : 5
Ta liczba jest nieparzysta

Podaj jakąś liczbę z zakresu
od 1 do 10 : 6
Ta liczba jest parzysta

Podaj jakąś liczbę z zakresu
od 1 do 10 : 14
Podana liczba nie mieści się
w zakresie
```

Listing 2

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

int main()
{
 cout << „Podaj jakąś liczbę z zakresu od 1 do 10 : „;
 int liczba;
 cin >> liczba;
 if(liczba >= 1 && liczba <= 10)  //(1)
  {
   cout << „Ta liczba jest „;
   if( liczba % 2 )               //(2)
   cout << „nie”;
   cout << „parzysta” << endl;
  }
      else          //(3)
      {
       cout << „Podana liczba nie mieści się w zakresie\n”;
      }
}
```

(1) Zacznę od omawiania tej linijki, wcześniejsze są, mam nadzieję, oczywiste. Wprowadzona przez nas liczba jest poddawana sprawdzeniu, ponieważ zażądaliśmy wprowadzenia liczby od 1 do 10. Warunek zapisany w nawiasie prawdziwy jest tylko dla liczb z tego zakresu dla innych, jak na przykład dla 14 nie i przenosi nas on od razu do bloku else (3) i zostaje wyświetlona informacja o błędnej liczbie. Jeśli liczba jest poprawna, wykonywane są operacje zawarte w bloku if.

(2) Widzimy tutaj, że bloki można zagnieżdżać, to znaczy w jednym bloku może być inny blok, w tamtym kolejny i tak dalej. Teraz sprawdzany jest ten warunek (2). Jest to zwykłe działanie arytmetyczne. Zastosowałem je po to, by pokazać ciekawą rzecz w C++, mianowicie konwersję typów. Spotkaliśmy się z nią, omawiając operacje arytmetyczne na znakach. Tam wymusiliśmy, by liczba całkowita została zamieniona na znak przy pomocy zapisu char(liczba_całkowita). Była to tak zwana konwersja jawna, gdyż sami jej zażądaliśmy. W C++ jednak występuje wiele konwersji niejawnych, czyli takich, które bez specjalnej prośby kompilator dla nas wykonuje. Jedną z nich jest zamiana wartości liczb na wartości boolowskie, czyli prawdę lub fałsz. Przyjmuje się zasadę, że 0 jest zamieniane na fałsz, a każda wartość z różna od zera naprawdę. W naszym przypadku jest to reszta z dzielenia. Użyję jej, by sprawdzić parzystość liczby. Dowolna liczba parzysta dzielona przez dwa daje reszty 0, a nieparzysta 1. Jeśli w warunku dzielimy liczbę nieparzystą przez 2 np. 5, to mamy reszty jeden. Ta jedynka zmieniana jest na wartość prawda, a ta z kolei sprawdzana jest w warunku if i w środek powstającego zdania wstawiane jest „nie”, dzięki czemu otrzymujemy zdanie „Ta liczba jest nieparzysta”. Jeśli jest parzysta jak na przykład 6, to wynikiem 6 % 2 jest zero, a więc fałsz i blok nie jest wykonywany. Otrzymujemy więc „Ta liczba jest parzysta”. W tej linijce widzimy również, że po if nie ma bloku klamrowego. Oznacza to, że ma być wykonana po nim jedna linijka. Zastosowałem w niej wcięcie, by było to lepiej widoczne. Wcięcie to jednak niczemu więcej nie służy jak tylko poprawie czytelności.

Znamy już sposób na wykonanie pewnych czynności w zależności od zaistniałej okoliczności. Jednak w ten sposób można dany zestaw instrukcji wykonać raz, po czym program przechodzi dalej. Czasami chcielibyśmy, by program wykonał pewną czynność wielokrotnie, np. dodał 10^12 razy 12 do 7*10^12, by sprawdzić, czy to, co mówiłem o dokładności, jest prawdą. Oczywiście, wpisywanie 10^12 razy linijek dodawania odpada, gdyż nawet jeśli byśmy byli sprytni i wklejali je po kolei jedna pod drugą z częstotliwością jednej na sekundę, to zajęło by nam to 321502 lat. Do tego czasu kurs ten by się zdezaktualizował. Odpowiedzią na nasze problemy są tak zwane pętle, które wykonują się dopóki warunek jest prawdziwy. W C++ istnieją trzy pętle while, do while i for, które teraz po kolei opiszę.

Pierwszą z nich będzie najprostsza i występująca w każdym języku programowania pętla while. Ma ona postać

```
while(warunek)
{
  wykonywane instrukcje;
}
```

Podobnie do if instrukcje wykonywane są dopóty, dopóki warunek jest prawdziwy. Jeśli jest on prawdziwy w kolejnych wywołaniach, to instrukcje wykonywane są wielokrotnie. Podobnie do if blok klamrowy jest wymagany tylko w przypadku większej liczby linijek instrukcji niż jedna - **listing 3**.

```
 a odpowiada wartości : 97
 b odpowiada wartości : 98
 c odpowiada wartości : 99
 d odpowiada wartości : 100
...
 x odpowiada wartości : 120
 y odpowiada wartości : 121
 z odpowiada wartości : 122
```

Listing 3

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

int main()
{
 char znak = 'a';
 while(znak <= 'z') //(1)
  {
  cout << znak << „ odpowiada
  wartości : „ << int(znak)
  << endl;//(2)
  znak = znak + 1;  //(3)
  }
}
```

Wynik programu skróciłem.

(1) Program ten wpisuje wartości liczbowe znaków od 'a' do 'z'. Na jego przykładzie wyraźnie widać do czego używane są pętle – do wielokrotnego powtarzania danej grupy instrukcji, która nie musi być wcale za każdym razem identyczna. Pętla rozpoczyna się zawsze od sprawdzenia warunku. W tym przypadku kompilator sprawdza, czy zmienna znak (zawierająca wartość 'a') ma mniejszą wartość niż 'z' jeśli tak, to rozpoczyna się pierwsze wykonanie pętli.

(2) W środku pętli wyświetlamy zawartość zmiennej znak zarówno w formie znaku, jak i jawnie rzutując w formie liczby.

(3) Następnie wartość zmiennej znak zwiększana jest o jeden, czyli przechowuje wartość 'b'. Dla osób nieobeznanych z programowaniem i posiadających pewne przyzwyczajenia z matematyki, zapis ten może się wydać absurdem. Wynikać może z niego, że wartość znak równa się wartość znak + 1. Jeśli jednak przypomnimy sobie znaczenie operatorów C++ to okaże się, że oznacza to, że znak przyjmuje wartość znak + 1, a więc 'b'. Zapis znak == znak +1 byłby dopiero porównaniem w języku C++. Ponieważ jest to ostatnia linijka, rozpoczyna się ponownie wykonywanie pętli. Sprawdzany jest warunek z punktu (1), jest on prawdziwy, a więc ciało zostanie jeszcze raz wykonane. Będzie się tak działo, dopóki wartość zmiennej znak nie przyjmie wartości większej niż 'z'.

Kolejną opisaną pętlą będzie do while. Ma ona następującą składnię:

```
do
{
  instrukcje;
} while(warunek);
```

W tej pętli najpierw wykonywane są instrukcje, a dopiero potem sprawdzany jest warunek. Mamy więc pewność, że zawarte w niej instrukcje zostaną wykonane choć raz, nawet gdy warunek jest od początku błędny. Wielu autorów podręczników twierdzi, że w pętli tej popełnia się najwięcej błędów - **listing 4**.

```
Wprowadź liczbę : 8
Wprowadź liczbę : 12
Wprowadź liczbę : 13
```

Listing 4

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

int main()
{
 int wynik = 0;
 do                    // (1)
  {
   int liczba;
   cout << "Wprowadź liczbę : ";
   cin >> liczba;
   wynik = liczba + wynik;
  } while( wynik < 30); //(2)
}
```

(1) W rozpoczynającej się pętli prosimy użytkownika o podanie liczby, którą następnie dodajemy do zmiennej wynik. Dopóki zmienna wynik jest mniejsza od warunku sprawdzanego w punkcie (2), pętla się wykonuje. Widać wyraźnie, że ciało pętli wykona się przynajmniej raz.

Pętlą, która ma najbardziej skomplikowaną składnię i przez to odstrasza początkujących, ale najczęściej jest wykorzystywana w programach, jest zmienna for:

```
for (instrukcja inicjująca ;
warunek ; krok pętli)
{
 wykonywane instrukcje;
}
```

Wygląda to strasznie, ale jak już pozna się składnię tej pętli, to jest ona jedną z najczęściej używanych. Składa się z instrukcji inicjującej, warunku i kroku pętli. Są to trzy elementy oddzielone średnikiem. Każdy z tych elementów możemy opuścić. Instrukcja inicjująca używana jest do zainicjowana pętli. Tutaj najczęściej definiujemy pewną zmienną, która będzie używana w pętli. Od niej zaczyna się wykonywanie pętli i ta część wykonywana jest tylko jeden raz na samym początku. Następnie sprawdzany jest warunek. Jeśli został on opuszczony, przyjmuje się, że jest on zawsze prawdziwy, a więc będzie to pętla nieskończona (o tym, jak taką pętlę opuścić, powiemy pod koniec tej części). Po sprawdzeniu warunku wykonywane są instrukcje w bloku klamrowym lub jedna instrukcja, jeśli bloku brak. Po ich wykonaniu odbywa się wykonanie kroku pętli, tutaj najczęściej modyfikujemy zmienną zainicjowaną w instrukcji inicjującej, po czym sprawdzany jest warunek, następnie wykonywane są instrukcje, potem krok pętli itd. Wydaje się to bardzo skomplikowane, jednak pętla ta jest niezastąpiona przy pracy z tablicami. Na razie ich nie znamy, dlatego nie będziemy jej tak często używać. A oto i przykład zastosowania tej strasznej konstrukcji na **listingu 5.**

Wynik programu:

```
Wprowadź liczbę: 9
Silnia z 9 = 362880
```

Listing 4

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

int main()
{
 int silnia = 1;
 int wartosc;
 cout << "Wprowadź liczbę: ";
 cin >> wartosc;
 for(int i = wartosc ; i >= 2 ; i = i - 1 ) //(1)
  {
    silnia = silnie * i;   //(2)
  }
 cout << „Silnia z „ << wartosc << „ = „ << silnia << endl;
}
```

(1) Widzimy wywołanie pętli w instrukcji inicjującej, powołujemy do życia zmienną i przypisujemy jej wartość 1. Równie dobrze moglibyśmy tę zmienną zdefiniować przed pętlą, jednak wówczas rozciągałaby się w całej funkcji main, a nam potrzebna jest tylko i wyłącznie na potrzeby pętli, dlatego elegancko jest ją powołać właśnie tutaj. Po jej definicji następuje sprawdzenie warunku. Dla wprowadzonej przeze mnie 9 warunek 9 >= 2 jest spełniony (to znaczy jest prawdziwy), więc pętla może się wykonać. W jej wnętrzu jest tylko jedna instrukcja (2), więc teoretycznie moglibyśmy zrezygnować z bloku klamrowego, jednak tak będzie czytelniej. W instrukcji tej mnożymy zmienną silnia reprezentującą wartość działania przez kolejne wartości zmiennej i. Po wykonaniu instrukcji zawartych w blokach następuje krok pętli. W nim widzimy zmniejszenie wartości i o jeden. Następnie sprawdzany jest warunek i wykonywane są instrukcje. Dzieje się tak aż do momentu, w którym zmienna i nie osiągnie wartości jeden. Wtedy warunek jest fałszywy. Pętla kończy działanie i nasz program wypisuje wynik.

Została nam do omówienia jeszcze jedna sprawa. Mianowicie instrukcje sterujące wykonaniem pętli. Powiedziałem wcześniej, że pętla for w przypadku opuszczenia warunku jest zawsze prawdziwa, więc na przykład:

```
for( ; ; )
{
}
```

jest pętlą nieskończoną. Nie tylko pętla for może być nieskończona. Nieskończone mogą być pętle while, np.:

```
while( true )
{
}
```

lub

```
do while
do
{
} while( true );
```

Jednak w przypadku tych dwóch ostatnich nie możemy po prostu opuścić warunku, ale musimy wpisać wartość, zawsze prawdziwą. Najprostszym sposobem jest po prostu podanie true, czyli wartości prawda, równie dobrze można wpisać np. 1 == 1. Chodzi o to, by warunek zawsze był prawdziwy. Czym jest pętla nieskończona? Pętla nieskończona może być błędem programistycznym. Powoduje ona bowiem, że gdy program dojdzie do jej wywołania, to zatrzyma się w tym miejscu i nigdy się nie skończy. Będzie wyglądał jakby się zawiesił i trzeba go będzie brutalnie zamknąć. Jednak pętla nieskończona może być również przydatna. Może mieć zawsze prawdziwy warunek, ale i tak się skończy. Zakończymy ją z wnętrza przy pomocy pierwszej poznanej instrukcji sterującej break. Instrukcja ta kończy działanie pętli i powoduje wyjście do bloku main lub pętli wyżej, jeśli pętle są zagnieżdżone (pętle, podobnie jak instrukcję if, można w sobie wzajemnie zagnieżdżać) - **listing 6**.

(1) Program ten jest dowodem na to, że pętla z warunkiem zawsze prawdziwym może być kiedyś zakończona. Warunek pętli while jest wartością true, a więc jest zawsze prawdziwy, bez względu na to, czego wewnątrz pętli nie zrobimy. W pętli wprowadzamy kolejne znaki, które następnie wypisywane są wraz z wartościami w tabeli ASCI. Pętla kończy się wówczas, gdy wprowadzimy znak 'q' **(2)**.

Break może się przydać, kiedy pętla kończy się, gdy jeden z wielu warunków zostanie spełniony. Każdy zapisany po odpowiedniej instrukcji if wewnątrz pętli. Może się również przydać, gdy wewnątrz pętli zajdzie konieczność nagłego jej zakończenia z powodu problemów.

Oprócz break jest jeszcze jedna instrukcja sterująca wykonywaniem pętli. Jest nią continue. Różni się ona od break tym, że nie powoduje opuszczenia pętli, ale jedynie pominięcie instrukcji od momentu jej wywołania do końca danego przebiegu pętli - **listing 7**.

Listing 6

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

int main()
{
	char znak;
	while( true ) //(1)
	{
		cin >> znak;
		cout << znak << „ : „ << int(znak) << endl;
		if(znak == 'q') break;	//(2)
	}
}
```

Listing 7

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

int main()
{
	int liczba;
	cout << „Którą liczbę od 1 do 15 chcesz opuścić : „;
	cin >> liczba;
	for(int i = 1 ; i <= 15 ; i = i + 1)
	{
		if(i == liczba) continue;  //(1)
		cout << i << „  „;
	}
	cout << endl;
}
```

Wynik:

```
Którą liczbę od 1 do 15 chcesz
opuścić : 5
1  2  3  4  6  7  8  9  10  11
12  13  14  15
```

(1) Nie będę tłumaczył tego programu przesadnie dokładnie, gdyż wydaje mi się on banalny. Gdy warunek i == liczba jest prawdziwy, dochodzi do wywołania continue i opuszczenia reszty danego przebiegu pętli bez wpływu na pozostałe. W naszym przypadku objawia się to opuszczeniem liczby 5 w tej wyliczance.

Istnieje pewna konstrukcja, która umożliwia opuszczanie wielokrotnie zagnieżdżonych pętli bądź skok w dowolne miejsce programu (prawie dowolne, gdyż jedynie w obrębie właśnie wywoływanej funkcji) jest to niesławna instrukcja goto. Nie będę jej dokładnie opisywał, gdyż użycie jej świadczy o złym stylu programowania i zawsze można ją zastąpić innym mechanizmem. Nadmierne jej użycie doprowadza do produkcji kodu, który jest trudny do zrozumienia. Jej użycie sprowadza się do wstawienia w miejscu programu, skąd chcemy się przenieść, instrukcji: `goto etykieta;` i w miejscu, do którego się przenosimy, instrukcji: `etykieta:`

Przedstawię prosty przykład wielokrotnie zagnieżdżonych pętli:

```
for( ; ; )
{
while(true)
 {
   char uciec;
   cin >> uciec;
   if (uciec == 't') goto
   ladowanie;
  }
}

ladowanie:
cout << "Udało się nam uciec
z pętli\n";
```

To by było wszystko, gdyby nie pewne operatory, które ominąłem, by zbytnio nie utrudniać rozumienia programów z tej części. W C++ istnieje wiele konstrukcji, których zadaniem jest ułatwienie programiście pisania i doprowadzających do znacznego zmniejszenia ilości kodu. Konstrukcje te jednak dla nowicjuszy mogą wydawać się trudne. Ponieważ jednak znamy już wszystkie operatory, nie powinno być z nimi problemów. Są to : +=, -=, *=, /=, %=, &=, |=, <<=, <<=; Wyglądają strasznie, ale są bardzo proste i naprawdę ułatwiają pisanie. Przyznam się, że pisząc przykładowe programy z tego rozdziału, korzystałem z nich, ale po zastanowieniu się zdecydowałem się je zamienić na prostsze konstrukcje, by nie przerażać czytelników. Ogólnie mówiąc, każdy operator o ogólnym wzorze @= :

```
liczba @= 1;
```
oznacza: `liczba = liczba @ 1;`

Na przykład:

`liczba += 12;` to inaczej
```
liczba = liczba + 12;
```

`liczba *= 3;` to
```
liczba = liczba * 3;
```

Oprócz tych wielu operatorów przypisania mamy jeszcze dwa rodzaje operatorów inkrementacji i dekrementacji: ++ i --. Oznaczają one odpowiednio zwiększenie o 1 i zmniejszenie o 1 np.

`liczba++;` oznacza
```
liczba = liczba + 1;
lub liczba += 1;
```

`liczba--;` oznacza
```
liczba = liczba - 1;
lub liczba -= 1;
```

Operatory te mogą występować w dwóch odmianach post- i prefiksowej, a więc odpowiednio mogą znajdować się po zmiennej i przed zmienną. Jeśli znajdują się przed zmienną, modyfikują jej wartość od razu. Jeśli po zmiennej to zmienna w bieżącej instrukcji ma jeszcze starą wartość, ale w następnej już odpowiednio jest zwiększona bądź zmniejszona o jeden. Nie muszę mówić chyba, że są to zapisy często stosowane w pętli for, a zwłaszcza w jej kroku pętli.

A oto i ostatni dzisiaj program - **listing 8**.

(0) W programie tym wynik działania zamieszczę z prawej strony. Każda jedna linijka wyniku odpowiada jednej linijce działania programu. Ta linijka nie drukuje się po wykonaniu programu, dlatego ma numer zero. Dochodzi do inicjacji dwóch zmiennych a i b. Ważne są przypisane im wartości.

(1) W tej linijce widzimy, jak wartość a zwiększa się pod wpływem operatora ++.

(2) Tutaj zmniejsza się o jeden wartość b.

(3) Dwie wcześniejsze linijki były proste i łatwe do przyswojenia. Problemy zaczynają się dopiero od tej linijki. Widzimy, że zmienna a poddawana jest działaniu operatora ++ w wersji postfiksowej. Mimo to jej wartość w tej instrukcji nie zmienia się. Widzimy, że podobnie jak w linijce (1) wynosi ona 8. Jednak linijka następna (4) pokazuje, że wartość a uległa zmianie.

(4) W linijce tej nie stoi obok a żaden operator i mimo tego jej wartość zwiększyła się o jeden. Jest to wynik wcześniejszego operatora ++ w wersji post-.

(5) Tutaj widzimy analogiczną sytuację, tylko że chodzi o zmniejszenie o jeden. Podobnie wartość w tej linijce jeszcze się nie zmienia. Zmiana uwidacznia się dopiero w następnej instrukcji.

Listing 8

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

int main()
{
                                        // Wynik:
        int a = 7, b = 8;               // |0|
        cout << ++a << endl;            // |1| 8
        cout << --b << endl;            // |2| 7
        cout << a++ << endl;            // |3| 8
        cout << a << endl;              // |4| 9
        cout << b-- << endl;            // |5| 7
        cout << b << endl;              // |6| 6
}
```

To koniec tej części. Była bardzo długa, ale mam nadzieję, że również ciekawa. Ponieważ znamy już operator ++, wyjaśnia się tajemnicza nazwa C++. Język ten nazywa się właśnie w ten sposób, ponieważ wywodzi się on bezpośrednio od języka C, do którego dodano nowe elementy, tworząc doskonalszą i, co najważniejsze, następną wersję. Tak więc C++ oznacza po prostu następną wersję.

W następnej części porozmawiamy o typach złożonych, a zaczniemy od tablic i funkcji.

[EOT]

W poprzednich odcinkach kursu:

Część 1:
> Wprowadzenie do języka C++

Część 2:
> Zmienne

Java cz. 4

W tym odcinku chciałbym omówić pętle oraz sposób tworzenia i używania tablic.

Piotr „dragon" Krakowiak

Pętle to konstrukcje programistyczne, które umożliwiają powtarzanie czynności. Na przykład jeżeli mielibyśmy wyświetlić na ekranie dwadzieścia razy napis „Dragonia Magazine", wystarczy zastosować pętlę for i jeden wpis System.out.println („Dragonia Magazine"); .

Pętla for

Ogólna postać pętli for:

```
for (wyrażenie początkowe;
wyrażenie warunkowe; wyrażenie
modyfikujące)
{
 instrukcje do wykonania
}
```

> wyrażenie początkowe – stosujemy do inicjalizacji zmiennej, która spełnia rolę licznika wykonania pętli

> wyrażenie warunkowe – czyli warunek, który sprawdza, czy ma być dokonane następne przejście pętli

> wyrażenie modyfikujące – modyfikuje zmienną, która jest licznikiem

Mały przykład dla zobrazowania działania pętli na **listingu 1.**

Kod pętli oznacza: zdeklaruj zmienną i i przypisz jej wartość 0 (int i = 0) . Następnie tak długo wykonuj pętlę (zawartość nawiasów klamrowych) , dopóki wartość i jest mniejsza od 20.

Wynik działania pętli:

```
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
...
Dragonia Magazine
```

Napiszemy jeszcze jeden programik z użyciem pętli for, aby zobrazować zwiększanie się licznika.

Listing 1

```
class Main
{
  public  static  void  main
(String args [])
 {
  for (int i = 0; i < 20; i++)
    {
     System.out.println
      ("Dragonia Magazine");
    }
 }
```

Listing 2

```
class Main
{
 public static void main
 (String args [])
  {
   for (int i = 1; i <= 5; i++)
    {
      System.out.println
      ("Dragonia Magazine nr "+i);
    }
  }
}
```

Wynik działania powyższego programiku.

```
Dragonia Magazine nr 1
Dragonia Magazine nr 2
Dragonia Magazine nr 3
Dragonia Magazine nr 4
Dragonia Magazine nr 5
```

Pętla while

Pętla while – tak jak pętla for – służy do powtarzania pewnych czynności. Pętli while używamy najczęściej wtedy, kiedy nie znamy liczby powtórzeń, a pętli for kiedy liczba powtórzeń jest znana.

Ogólna postać pętli while:

```
while (wyrażenie warunkowe)
{
 instrukcje;
}
```

Instrukcje znajdujące się w nawiasach klamrowych będą tak długo wykonywane, dopóki wyrażenie warunkowe jest prawdziwe.

Listing 3

```
class Main
{
 public static void main
 (String args [])
  {
   int i = 1;
   while (i < 5)
    {
       System.out.println
       ("Dragonia Magazine");
       i++;
    }
   }
}
```

Wynik kodu:

```
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
Dragonia Magazine
```

Pętla do... while

Pętla do... while różni się od pętli while tym, że najpierw wykonywane są instrukcje, a dopiero później sprawdzany jest warunek.

Ogólna postać pętli do...while:

```
do
 {
  instrukcje;
 }
while (warunek);
```

Przykład z użyciem pętli do... while:

Listing 4

```
class Main
{
  public static void main
  (String args [])
  {
    int i = 1;
       do
       {
          System.out.println
        ("Dragonia Magazine");
         i++;
       }
       while (i < 5);
 }
}
```

Tak jak już wspominałem, pętla do... while ma tę charakterystyczną cechę, że wykonuje instrukcje przed sprawdzeniem warunku. Co oznacza, że pętla wykona się co najmniej raz.

Tablice

Tablica to struktura danych pozwalająca na przechowywanie ciągu danych tego samego typu.

Poniżej struktura tablicy czteroelementowej:

Tak jak widać na schemacie, tablica składa się z ponumerowanych kolejno komórek. Każda komórka może przechowywać jakąś część danych. Dane, które są przechowywane w tablicy, zależą od typu tablicy. Mogą to być dane typu np. int, char.

Ogólna postać deklaracji tablicy wygląda tak:

```
typ_tablicy nazwa_tablicy [];
```

Deklaracja nie załatwia jeszcze całej sprawy – powstała dopiero zmienna o nazwie nazwa_tablicy. Samej tablicy jeszcze nie ma i trzeba ją utworzyć za pomocą operatora new.

```
new typ_tablicy [liczba_elementów];
```

Możemy jednocześnie zdeklarować i utworzyć tablicę:

```
typ_tablicy nazwa_tablicy [] =
new typ_tablicy [liczba_elementów];
```

lub

```
typ_tablicy nazwa_tablicy [];
nazwa_tablicy = new typ_tablicy
[liczba_elementów];
```

Zobaczmy, jak wygląda użycie tablicy w praktyce:

Listing 5

```
class Main
{
  public static void main
  (String args [])
  {
   int tab [] = new int [1];
   tab [0] = 10;
   System.out.println
   ("Element nr 0 tablicy
    ma wartość: " + tab [0]);
 }
}
```

Wynik kodu:

```
Element nr 0 tablicy
ma wartość: 10
```

Mamy tu przykład, jak za pomocą pętli for i metody length uzupełnić tablicę i wyświetlić ją. Gdy zastosujemy nazwa_tablicy.length, wynikiem tej konstrukcji będzie długość tablicy.

Listing 6

```
class Main
{
 public static void main
 (String args [])
  {
   int tab [] = new int [10];
  for (int i = 0;
  i < tab. length; i++)
       {
         tab [i] = i;
       }
       for (int i =0;
       i < tab. length; i++)
       {
        System.out.println
       ("tab [" + i + "] = "
        + tab [i]);
       }
  }
}
```

Wynik powyższego kodu.

```
tab[0] = 0
tab[1] = 1 ...
```

W tym odcinku to wszystko. W następnym zapoznamy się z tablicami wielowymiarowymi. [EOT]

Gimp cz.3 - efekty specjalne

Połyskliwa powierzchnia – tym tematem zajmiemy się w numerze.

Piotr „dragon" Krakowiak

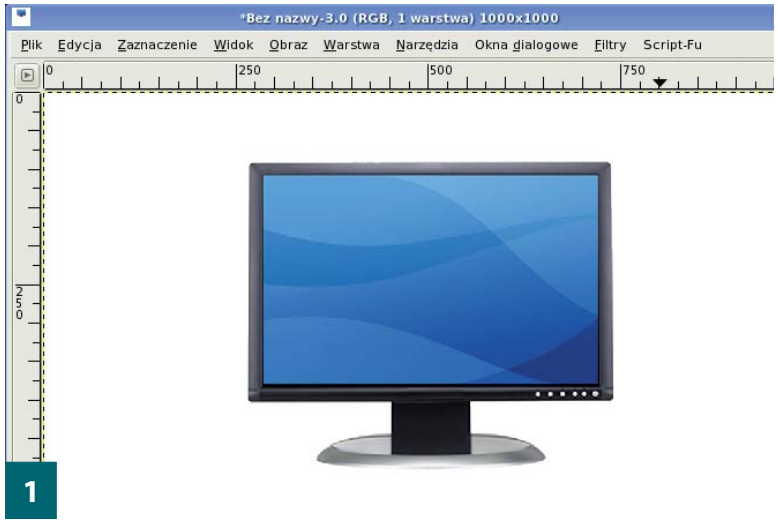
Efekt ów ma sprawiać wrażenie przedmiotu postawionego na połyskliwej powierzchni, która odbija jego kształt.

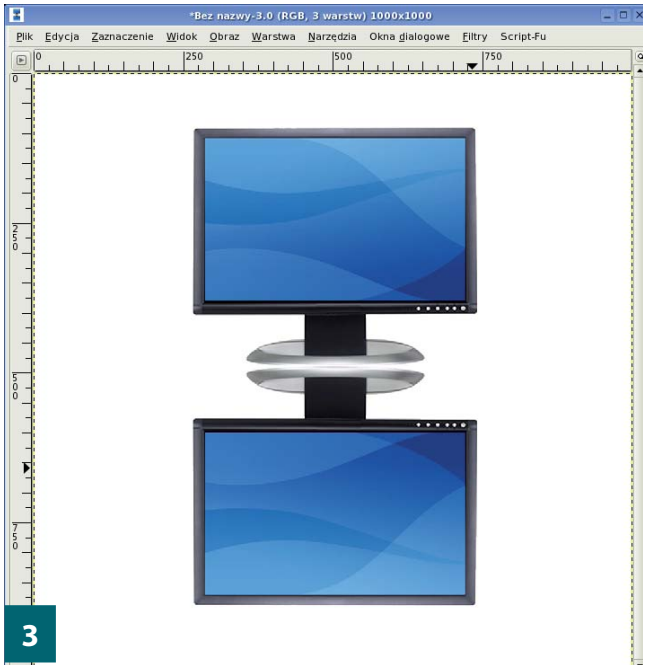
Rys 1
Najpierw wybieramy obrazek, u mnie będzie to monitor. Wklejamy go na białe tło.

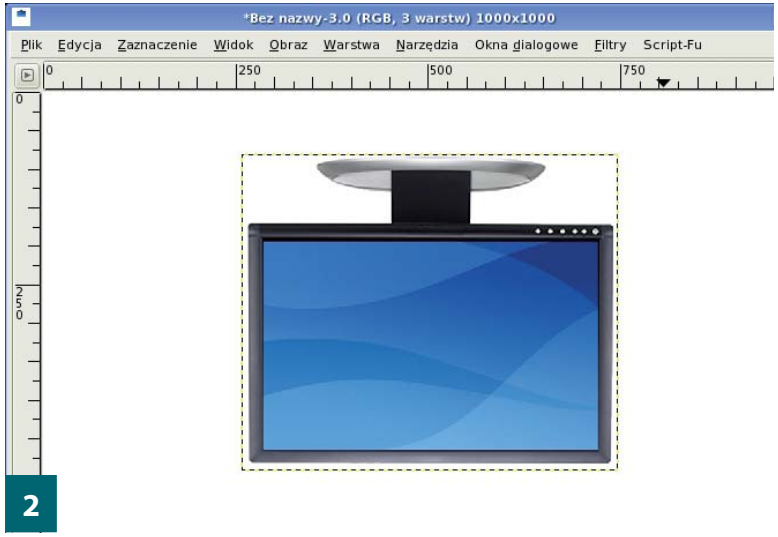
Rys. 2
Wklejamy go ponownie na następną warstwę i odwracamy ją w pionie (warstwa – przekształcenie – odbij pionowo).

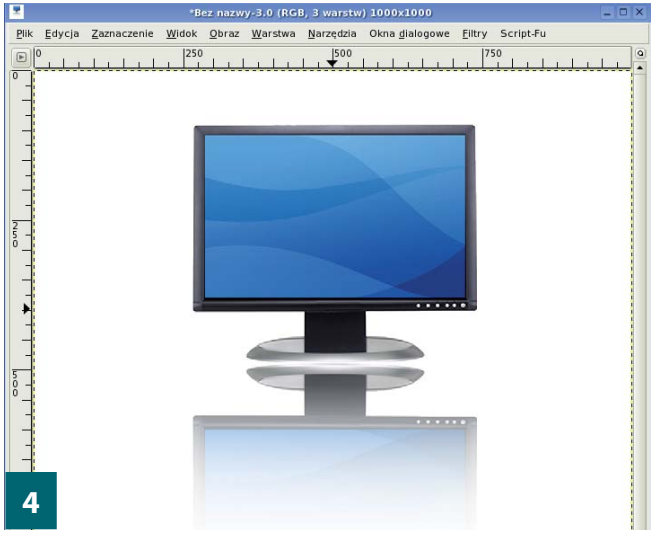
Rys. 3
Następnie przesuwamy obrazek tak, aby stykał się górą z dołem wcześniejszej warstwy, tak jak to widać na obrazku.

Rys. 4
Używamy teraz Wypełnienie gradientem z kolorem białym i przezroczystym. Stosujemy go na warstwie z obróconym obrazkiem. [EOT]









Podłączenie systemu Linux & FreeBSD do internetu przez modem SpeedTouch 330

Tomasz Roszak

Na rynku mamy dostępnych trzech głównych operatorów telefonii stacjonarnej oferujących stały dostęp do internetu: TP SA, Netia, Dialog.

Dialog wykorzystuje modemy ze złączem ethernetowym RJ-45 i z podłączeniem nie ma najmniejszych problemów. Netia i TP SA swoich klientów uszczęśliwiają modemami ze złączem USB. Do każdego zestawu dodana jest instrukcja, jak podłączyć się do internetu z systemu Windows i z systemu Linux.

U mnie próba instalacji modemu wg przepisu operatora nie powiodła się zarówno pod Windowsem (komunikat: program wykonał nieprawidłową operację), jak i pod Linuxem.

Ten artykuł przybliży sposób konfiguracji systemu do uruchomienia Neostrady. Opis będzie dotyczył dwóch systemów operacyjnych Linux: Gentoo i FreeBSD. Użytkownicy dystrybucji Linux SUSE, Ubuntu, Mandriva nie mają problemu – jest gotowy pakiet.

1. Dostosowanie jądra systemu Linux. Zaczynamy od edycji pliku konfiguracyjnego naszego jądra. Do tego celu możemy użyć jednego z dostępnych narzędzi: menuconfig, xconfig lub gconfig. Musimy w jądro wkompilować następujące funkcje:

Wsparcie dla PPP i ATM

```
CONFIG_ATM=y
CONFIG_ATM_CLIP=y
#
# Wan interfaces
#
CONFIG_WAN=y
CONFIG_SYNCLINK_SYNCPPP=y
CONFIG_HDLC=y
CONFIG_HDLC_PPP=y
#
# ATM drivers
#
CONFIG_ATM_DUMMY=m
CONFIG_ATM_TCP=m
CONFIG_PPP=m
CONFIG_PPP_MULTILINK=y
CONFIG_PPP_FILTER=y
CONFIG_PPP_ASYNC=m
CONFIG_PPP_SYNC_TTY=m
CONFIG_PPP_DEFLATE=m
CONFIG_PPP_BSDCOMP=m
CONFIG_PPP_MPPE=m
CONFIG_PPPOE=m
CONFIG_PPPOATM=m
CONFIG_SLIP=m
CONFIG_SLIP_COMPRESSED=y
```

Wsparcie dla USB

```
#
# USB support
#
CONFIG_USB_ARCH_HAS_HCD=y
CONFIG_USB_ARCH_HAS_OHCI=y
CONFIG_USB=y
#
# USB Host Controller Drivers
#
CONFIG_USB_EHCI_HCD=y
CONFIG_USB_OHCI_HCD=y
CONFIG_USB_UHCI_HCD=y
#
# USB DSL modem support
#

CONFIG_USB_ATM=m
```

Dla systemu FreeBSD Edytujemy konfigurację jądra `/usr/src/sys/i386/conf/GENERIC` i sprawdzamy, czy mamy wpisy:

```
device usb
device uhci
device ohci
device ehci
device ugen
```

Jeżeli uprzednio wsparcie dla modemów nie było aktywne, rekompilujemy jądro systemu FreeBSD.

Dla obu przypadków (tj Linuxa i FreeBSD) po rekompilacji jądra wykonujemy restart systemu.

Przechodzimy do kolejnego etapu: Instalacja modemu.

Na płycie CD dostarczonej z modemem, znajduje się plik ZIP ST330_firmware_3012.zip. W nim są dwa drivery dla modemu SpeedTouch 330: KQD6_3.012 oraz ZZZL_3.012.

Plik KQD6_3.012 przemianowujemy na mgmt.o, natomiast plik ZZZL_3.012 na firmware.bin.

Spotkałem się z dwoma rodzajami modemów SpeedTouch 330. Duży srebrny i mniejszy czarny.

Dla pierwszego modelu należało wgrać mgmt.o, dla drugiego firmware.bin.

Dla obu systemów (FreeBSD i Linux) sprawdzamy czy mamy subkartotekę `/usr/local/libdata`. Kopiujemy pliki mgmt.o i frmware do w/w subkartoteki.

Do /usr/src przenosimy plik speedtouch-1.2.tar.gz i wypakowujemy z niego pliki.

Kompilujemy sterowniki poleceniem:

```
./configure && make && make install
```

Kolejny etap; Instalacja ppp.

Dla Freebsd przechodzimy do subkartoteki:

```
/usr/ports/net/pppoa
```

i wydajemy polecenie: make install && make clean

Dla systemu Linux Gentoo wydajemy polecenie: `emerge ppp`

Mamy zainstalowane ppp oraz sterowniki. Czas na konfigurację naszego połączenia.

Przechodzimy do subkartoteki:

```
/etc/ppp/
```

edytujemy plik chap-secrets i wprowadzamy nasz login i hasło:

```
# Secrets for authentication
# using CHAP
# client server secret IP addresses
login@neostrada.pl * hasło *
```

To samo robimy dla pliku pap-secrets

```
# Secrets for authentication
# using CHAP
# client server secret IP addresses
login@neostrada.pl * hasło *
```

Przechodzimy do następnej subkartoteki `peers` i tworzymy plik neo o następującej treści:

#dla Linux Gentoo

```
debug
kdebug 1
noauth
usepeerdns
noipdefault
defaultroute
pty „/usr/local/sbin/pppoa3 -m 1 -c -vpi 0 -vci 35”
sync
user login@neostrada.pl
noaccomp
nopcomp
noccp
holdoff 4
persist
maxfail 25
```

Dla FreeBSD:

```
default:
ident user-ppp VERSION (built COMPILATIONDATE)
set log Phase Chat IPCP CCP tun command
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0 0.0.0.0
set login

neo:
set authname login@neostrada.pl
set authkey PASSWORD
set device!”/usr/local/sbin/pppoa3 -c -m 1 -vpi 0 -vci 35 -d /dev/ugen0”
accept chap
set speed sync
set timeout 0
set reconnect 10 100
add default HISADDR
enable dns
```

W miejsce login i hasło wprowadzamy login i hasło nadane przez TP SA

Tworzymy skrypty startowe.

FreeBSD: neostrada.sh

```
#!/bin/sh
case $1 in
  start)
 /usr/local/sbin/modem_run -f
 /usr/local/libdata/firmware.bin
 ppp -nat -background adsl
    ;;
  stop)
 killall modem_run
 killall ppp
 ;;
 *)
 echo „Usage: $0 {start|stop}”
 exit 1
esac
```

Możemy sprawdzić, czy wszystko jest OK, wydając polecenie:

```
sh neostrada.sh start
```

Po około minucie nasze zestawienie połączenia powinno być gotowe.

`ifconfig -a` i powinniśmy widzieć interface tun0

Gentoo /etc/init.d/neostrada

```
#!/sbin/runscript
start () {
ebegin „Starting neostrada”
/usr/local/sbin/modem_run -v 1 -m -f /usr/local/libdata/firmware.bin
pppd call neo
eend $? „Failed to start neostrada”
}

stop () {
ebegin „Stoping neostrada”
 killall pppd
 eend $? „Failed to stop
 neostrada”
}
```

Możemy sprawdzić, czy wszystko jest OK, wydając polecenie:

```
sh neostrada start
```

Po około minucie nasze zestawienie połączenia powinno być gotowe.

```
ifconfig – a
```

i powinniśmy widzieć interface ppp0

Skomplikowanie instalacji Speedtoucha w naszym ulubionym systemie, spowodowane jest głównie ze względu na niewolny sterownik do modemu Alcatela, który ze względu na swoją niewolność, nie może być dostarczony standardowo w dystrybucjach. Ale jak widać, można i z tym sobie poradzić. [EOT]

Apache Web Server v2, część 2

Rafał (ert16) Trójniak

O czym będzie: „mod_access” – moduł decydujący o udostępnieniu zasobu, „mod_info”, „mod_status” – moduły informacyjne, „mod_alias” – mapowanie adresów URL, „mod_userdir” – serwer wielu użytkowników, „mod_dir” – czyli mały dodatek do katalogów.

Mod_access – Udostępnić czy nie...

Wszystkie dyrektywy ograniczające dostęp mogą pojawić się zarówno w głównym pliku konfiguracyjnym, jak i w plikach.htaccess. Instrukcje mogą się znajdować również w blokach ograniczających ich działanie jak <Directory> czy <Location>. Dyrektywy mają bardzo zachowawcze nazwy. Allow oraz Deny: Dla obydwu tych dyrektyw składnia jest identyczna. Przykład:

```
Allow from all
```

Gdzie: `Allow lub Deny` nazwa dyrektywy, `from` jest elementem występującym zawsze (swego rodzaju słowo kluczowe), `All` jest jednym z wyrażeń, które decydują o zakresie przyznania lub odebrania dostępu do zasobu. Właśnie te wyrażenia to element, na którym warto się skupić.

Oto kilka praktycznych zastosowań dyrektyw do przyznawania przywilejów oglądania naszego serwisu.

```
Deny from all
```

```
Allow from dragonia.pl
```

Najpierw blokujemy dostęp, następnie udostępniamy zasób tylko adresowi z nazwą dragonia.bee.pl

```
Allow from.pl
```

A to sposób na dopuszczenie tylko hostów z domeny.pl

```
Allow from 172.16.0.1
```

Czyli weryfikacja po adresie IP.

Ale możemy też pobawić się w podsieci

```
Allow from 172.16
Allow from 172.16.0.0/255.255.0.0
Allow from 172.16.0.0/16
```

Wszystkie trzy powyższe przykłady zawężają grono odbiorców do danej podsieci 172.16.0.0 z maską 255.255.0.0.

Jednak przy bardziej złożonych zestawach instrukcji nasuwa się pytanie: które dyrektywy analizowane są pierwsze? Tutaj przychodzi z odpowiedzią dyrektywa Order przyjmująca parametr o trzech możliwych wartościach:

`Allow, Deny` – dyrektywy Allow analizowane przed dyrektywami Deny. Każde niezakwalifikowane zapytanie jest odrzucane.

`Deny, Allow` – Analogicznie Deny przed Allow. Domyślną akcją jest akceptacja zapytania.

`Mutual-failure` – Akceptowane są jedynie połączenie pasujące do dyrektyw Allow, jednocześnie rozbieżne z dyrektywami Deny.

Można więc stwierdzić, że wystarczy użyć dyrektywy Order Allow, Deny i nie używać żadnych innych dyrektyw, aby zablokować dany obszar. W takim przypadku korzystamy z akcji domniemanej jaką jest w tym przypadku odrzucenie.

Moduły informacyjne

Apache posiada dwa moduły pozwalające na obserwowanie operacji zachodzących wewnątrz serwera. Pozwalają one lepiej zrozumieć mechanizmy działania Apache.

`mod_info` – pozwala na wyświetlanie informacji o załadowanych modułach, udostępnianych dyrektywach i aktualnej konfiguracji. Moduł ten przydaje się podczas wprowadzania wszelkiego rodzaju ustawień. Nie poleca się jednak udostępniać takich informacji publicznie, ponieważ mogą one zostać wykorzystane przeciwko Tobie.

Konfiguracja modułu sprowadza się do jednego bloku konfigurującego moduł.

```
<Location/NaszeInfo>
    SetHandler server-info
    Order deny, allow
    Deny from all
    Allow from mojKomputer.com
</Location>
```

Blok `<Location>` definiuje tutaj nazwę pliku, a komenda SetHandler ustawia obsługę tego adresu przez moduł server-info. Wiersze 3-5 nie są wymagane, jednak ze względów bezpieczeństwa są bardzo ważne. Dyrektywy te zostały już omówione. Zależy nam przecież na poufności informacji o konfiguracji naszego serwera – prawda?

`mod_status` – jest to moduł odpowiadający za informowanie administratora o aktualnym statusie serwera. Podaje on wiele informacji dotyczących jego aktualnej pracy: ilość obsłużonych zapytań, ilość na sekundę, obciążenie procesora, ilość procesów oraz ich stan. Jednym słowem moduł ten pomaga w dostosowaniu serwera jeśli chodzi o wydajność.

```
<Location/NaszStatus>
    SetHandler server-status
    Order Deny, Allow
    Deny from all
    Allow from mojKomputer.com
</Location>
```

Jak widać, konfiguracja jest identyczna, z wyjątkiem argumentu dyrektywy SetHandler. Dodatkowo moduł ten posiada dyrektywę pozwalającą wyświetlić bardziej szczegółowe informacje. Aby uaktywnić tę opcję, wpisujemy:

```
ExtendedStatus On
```

Informacje o statusie naszego serwera dostępne są po zrestartowaniu demona pod adresem http://adres.na-

szego.serwera/NaszStatus. Dodając na koniec tego adresu ciąg? refresh=N, uzyskujemy odświeżanie się strony co N sekund, co pozwoli na lepszą obserwację strony i odciążenie klikającego przycisk „Refresh" palca.

`mod_alias` – mapowanie adresów URL

Przez mapowanie adresów rozumiem tutaj przyporządkowywanie określonym adresom url innych, niżby to wynikało z drzewa udostępnionego przez serwer poleceniem DocumentRoot katalogów. Dzięki takiemu zabiegowi możemy swobodnie ominąć ograniczenia i nieprzyjemności wynikające z tworzenia dowiązań symbolicznych czy innych sposobów przenoszenia katalogów w drzewie serwera. Poza bezpośrednim aliasem katalogu mamy również możliwość przekierowania klienta do innego adresu url, nawet na innym serwerze (redirect). Do rzeczy:

```
Alias ścieżkaURL ścieżkaFS
```

Dyrektywa ta mapuje adres określony ścieżkąURL w naszym serwerze na katalog lub plik ścieżkaFS. W wyniku dyrektywy osiągamy jakby link symboliczny pliku w drzewie naszego serwera do zasobu podanego jako drugi parametr, jednak połączenie dokonywane jest na poziomie serwera. Drugim parametrem może być również ścieżka do pliku. Jeśli więc dla dyrektywy

```
Alias/icons/var/www/icons
```

będziemy żądać pliku http://mojaDomena.pl/icons/x.jpg, serwer będzie się starał udostępnić nam plik/var/www/icons/x.jpg. Z uwagi na możliwość zagnieżdżenia dyrektyw należy pamiętać, że ważna jest tutaj kolejność ich wpisywania. Prześledźmy następujący przykład:

```
Alias/icons/bill/home/bill/icons
Alias/icons//var/www/icons
```

Dla linku http://mojaDomena.pl/icons/bill/x.jpg powinniśmy otrzymać plik/home/bill/icons/x.jpg, natomiast dla http://mojaDomena.pl/icons/x.jpg plik/var/www/icons/x.jpg. Jeśli jednak odwrócimy kolejność tych dyrektyw, to dla pierwszego linku serwer będzie poszukiwał nieodpowiedniego pliku w lokalizacji/var/www/icons/bill/x.jpg, jako że link zostanie zakwalifikowany do pierwszej dyrektywy.

Drugą już wspomnianą dyrektywą udostępnianą przez mod_alias jest Redirect. Składnia jej jest prawie identyczna

```
Redirect [status] ścieżka URL URL
```

Jak już wspomniałem, dyrektywa ta podczas żądania dresu rozpoznanego jako ścieżkaURL przekierowuje przeglądarkę pod inny adres URL. Dodatkowym parametrem jakim jest status, możemy określić powód przekierowania. Może on przyjąć następujące wartości:

`permanent` – zasób został przeniesiony na stałe pod adres URL

`temp` – zasób przeniesiony chwilowo

`seeother` – zasób został usunięty, jednak dostępny jest inny, podobny zasób

`gone` – zasób został usunięty na stałe – nie podaje się tutaj adresu URL

`liczba całkowita` – kod błędu, jaki zwróci żądanie HTTP – dla początkujących raczej nieprzydatne.

Dyrektywa ScriptAlias jest w istocie identyczna z dyrektywą alias, dodatkowo uznając katalog docelowy jako katalog, w którym znajdują się skrypty CGI. Jej składnia jest identyczna.

Bardziej zaawansowanym administratorom polecam serie dyrektyw, które są modyfikacją wszystkich wyżej wymienionych. Są to odpowiednio dyrektywy AliasMatch, RedirectMatch oraz ScriptAliasMatch. Ich składnie wyglądają tak jak ich pierwowzorów, jednak adresy ścieżkaURL są w istocie wyrażeniami regularnymi rodem z Perla, a adres URL zawierać będzie wartości z dopasowanych nawiasów wyrażenia regularnego. Przykład:

```
AliasMatch/images/(.*)/(.*)/
home/$1/images/$2
```

pozwoli udostępnić obrazki różnych użytkowników pod adresami takimi jak http://MojaDomena.pl/images/joe/1.jpg, które zostaną przetłumaczone na/home/joe/images/1.jpg

„`mod_userdir`" – udostępnianie katalogów użytkowników.

Wyobraźmy sobie sytuację, w której na naszym serwerze istnieje 100 kont użytkowników i każdy chciałby mieć własną stronę. Sytuacja wydaje się straszna – tworzenie aliasów dla każdego użytkownika byłoby naprawdę trudnym, mozolnym i czasochłonnym zajęciem. Właśnie do rozwiązania tego problemu wykorzystujemy mod_userdir. Umożliwia on automatyczne udostępnianie stron użytkowników zarejestrowanych na serwerze, tworząc automatycznie aliasy po adresami http://mojaDomena.pl/~user/.

Cały moduł udostępnia jedną tylko dyrektywę służącą do konfiguracji, jednak wystarczającą dość wszechstronnie skonfigurować jego zachowania. Dyrektywa działa jakby w dwóch trybach.

Pierwszym trybem działania jest ustalanie katalogu użytkownika na kilka sposobów.

`UserDir public_html` – Jeśli wartością dyrektywy będzie nazwa katalogu, to udostępniony zostanie katalog o tej nazwie znajdujący się w katalogu domowym odpowiedniego użytkownika.

`UserDir/usr/web` – Jeśli podamy adres zaczyna się od korzenia drzewa systemu plików, to wyszukiwany będzie w nim katalog o nazwie odpowiadającej nazwie użytkownika.

`UserDir/home/*/www/` – Dla katalogu zaczynającej się od korzenia systemu plików, oraz zawierający gwiazdkę, owa gwiazdka będzie zastępowana nazwą użytkownika.

Jeśli więc wybierzemy adres `http://mojaDomena/~joe/katalog/strona`.html, to dla wyżej wymienionych konfiguracji wyszukiwany będzie odpowiedno katalog:

```
~joe/public_html/katalog/strona.html
/usr/web/joe/katalog/strona.html
/home/joe/www/katalog/strona.html
```

Dodatkowo analogicznie do powyższych przykładów można precyzować przekierowania dla tych adresów. Więc dla dyrektyw

```
UserDir http://www.staraDome-
na.com/users
```

```
UserDir http://www.StaraDomena.com/*/usr
UserDir http://www.StaraDomena.com/~*/
```

Przeglądarka zostanie przekierowana odpowiednio pod adresy

```
http://www.staraDomena.com/users/joe/katalog/strona.html
http://www.StaraDomena.com/joe/usr/katalog/strona.html
http://www.StaraDomena.com/~joe/katalog/strona.html
```

Ostatnim dodatkiem, jaki zafundowali nam programiści projektu Apache, jest możliwość podawania wielu adresów naraz. Będą one wybierane kolejno, jeśli poprzednie nie zostaną znalezione.

```
UserDir public_html/web/ http://server.pl/~*/
```

Powyższa konfiguracja spowoduje wyszukanie katalogu/home/joe/public_html, później/web/joe, a w ostateczności przekieruje przeglądarkę pod adres `http://server.pl/~joe/katalog/strona.html`

W drugim trybie działania dyrektywa ustala użytkowników, dla których moduł będzie funkcjonował.

```
UserDir disabled
UserDir enabled joe bill mike
```

Dzięki takiej konfiguracji moduł udostępni tylko katalogi użytkowników joe, bill oraz mike, natomiast opcje:

```
UserDir enabled
UserDir disabled root nouser gosc
```

umożliwią korzystanie z modułu wszystkim użytkownikom prucz root, nouser oraz gosc.

„`mod_dir`" – Odpowiednia obsługa katalogów.

Każdemu początkującemu administratorowi po pewnym czasie nasuwa się pytanie: „Dlaczego nazwy stron mogą prowadzić do katalogów i jaka strona jest wtedy wyświetlana". Kolejnym pytaniem jest: „Dlaczego akurat index.html?". Tutaj pojawia się właśnie moduł `mod_dir`. Ten mały, aczkolwiek bardzo przydatny kawałek kodu pozwala uprzyjemnić pracę w sieci.

Moduł ten posiada dwie funkcjonalności. Pierwszą jest wyświetlanie odpowiedniego pliku, kiedy serwer otrzymał adres katalogu. Z pomocą przychodzi nam dyrektywa

```
DirectoryIndex index.php index.html index.htm cgi/x.pl
```

Dyrektywa przyjmuje nieograniczoną liczbę plików, których kolejno będzie wyszukiwać w określonym katalogu, a następnie wyśle jego zawartość do przeglądarki. Mogą tutaj znaleźć się zwykłe pliki HTML, skrypty php czy cgi, mogą to być ścieżki do plików. Jeśli nie zostanie odnaleziony żaden z plików, serwer w zależności od konfiguracji modułu mod_autoindex wyświetli listę plików w katalogu, lub zwróci błąd.

Sytuacja wygląda pięknie, jednak jeśli adres URL katalogu nie zostanie zakończony znakiem "/" (slash), serwer będzie miał nie lada problem. Odwołując się do katalogu jak do pliku może zachowywać się dziwnie. Aby rozwiązać ten problem, moduł udostępnia dodatkową dyrektywę, która przekieruje przeglądarkę z każdego adresu nie zakończonego slashem do adresu poprawnego. Opcja ta jest domyślnie włączona, natomiast jeśli ktoś chciałby ją wyłączyć, musi użyć dyrektywy

```
DirectorySlash Off
```

jednak wyłączenie tej opcji nie jest polecane z punktu widzenia aspektów bezpieczeństwa naszego serwisu.

```
</HTML>
```

W kolejnym odcinku postaram się odsłonić przed wami konfigurację mechanizmu uwierzytelniania użytkownika za pomocą protokołu HTTP oraz omówić mechanizm Autoindex.

Jeśli macie jakieś sugestie na temat moich artykułów, pomysły czy propozycje poruszanych zagadnień, to zapraszam do pisania do mnie lub na adres redakcji. **[EOT]**

Wprowadzenie do Tcl/Tk część 1

Artykuł stanowi pierwszy odcinek serii o języku Tcl/Tk. Niniejszy tekst ma zachęcić czytelnika do próby poznania tego języka, ze szczególnym naciskiem na koncepcję budowy interfejsu użytkownika.

Tomek Łuczak

W kolejnych artykułach opowiem o pozostałych elementach interfejsu, o jego powiązaniu z programem, dochodząc do właściwego języka Tcl niejako przy okazji.

Wprowadzenie

Tk (ang. toolkit) jest biblioteką graficzną dla Tcl. Pozwala w kilku zaledwie wierszach stworzyć aplikację graficzną, do tego przenośną na wiele platform sprzętowych i systemów operacyjnych.

Z czasem pojawiło się wiele dodatkowych bibliotek graficznych do Tk, np. BWidget, IWidget czy szczególnie obiecujące tile, pozwalające na tworzenie tzw. skórek, czyli tematów graficznych.

Łatwość tworzenia i przenośność aplikacji spowodowały, że graficzny konfigurator kernela Linuxa do wersji 2.4 włącznie był napisany właśnie w Tcl-Tk, podobnie interfejs graficzny instalatora TEX Live.

Niniejszym artykułem chcę pokazać, jak łatwo można stworzyć atrakcyjny interfejs graficzny minimalnym nakładem czasu. Celowo nie zaczynam artykułu od składni, listy słów kluczowych języka Tcl itp., ale od poleceń Tk. Wynika to z dwóch powodów. Po pierwsze, aby czytelnik za pomocą kilku poleceń mógł sprawdzić, jak to działa i przekonać się do Tcl-Tk. Po drugie, kilka języków zapożyczyło Tk, łatając w ten sposób brak narzędzia do tworzenia interfejsów użytkownika (np. perl, python, a nawet ruby), więc znajomość Tcl jest sprawą wtórną, bo istotne jest poznanie Tk.

Skąd wziąć Tcl-Tk

Tcl-Tk jest dostępny w większości popularnych dystrybucji. Różnice polegają jedynie na wersji i liczbie standardowych rozszerzeń. Gdybyśmy jednak nie mieli w systemie Tcl-Tk, to ActiveState udostępnia chyba najobszerniejszą dystrybucję Tcl-Tk dla platform Linux, Solaris i Win32. Adresu strony trudno nie zapamiętać: http://www.tcl.tk

Okienka i widgety

Aplikacja okienkowa składa się z głównego okna, w którym umieszczane są widgety. (Widget to zlepek skrótów angielskich słów window i gadget, w świecie okiennym przyjęło się używać słowa kontrolka). Główne okno może posiadać menu i okna potomne.

Widgety to różnego rodzaju przyciski, suwaki, okienka do wpisywania tekstu, ogólnie to wszystko to, co możemy umieścić na planszy okna i wyświetlać lub klikać.

Struktura okienek i widgetów aplikacji ma strukturę hierarchiczną podobną do drzewa systemu plików *n*ksach, z tą różnicą, że korzeń oznacza się znakiem kropki. zamiast/. Kolejne widgety i okienka podrzędne posiadają identyfikator rozpoczynający się znakiem korzenia (kropki), a następnie w zależności od miejsca w hierarchii widgetów i okien posiadają wszystkie nadrzędne względem siebie identyfikatory rozdzielane znakiem kropki. Np. przycisk umieszczony w ramce z tytułem (oznaczonej.lf), która umieszczona jest w głównym oknie. posiada identyfikator:.lf.b

Aby widget mógł zostać wyświetlony, to musi wcześniej zostać zdefiniowany. Umieszczone widgety możemy usuwać z okna i modyfikować.

Zarządcy rozkładu

Tk posiada trzech różnych zarządców rozkładów. Zarządca rozkładu to mechanizm umieszczania widgetów w oknie. Zarządcy rozkładów:

> pack – automatyczny zarządca układający widgety automatycznie,

> grid – widgety układane są w siatce,

> place – miejsce umieszczenia widgetu jest podane explicite.

Nie jesteśmy uwiązani do jednego zarządcy, możemy korzystać z nich wg upodobania i naprzemiennie w zależności od potrzeb.

Proste widgety standardowe Tk

Label – etykieta

```
Label widget „opcja wartość”
```

Widget label wyświetla tekst, tekst może być powiązany ze zmienną, a przy zmianie wartości

zmiennej widget automatycznie zaktualizuje wyświetlany tekst. Przykłady:

```
label.l -text „OK”
label.l -textvariable zmienna
```

Button – przycisk

```
button widget „opcja wartoś”
```

Widget button tworzy przycisk. Do przycisku można dodać polecenia, które mają zostać wykonane po jego naciśnięciu. Przykłady:

```
button.b -text „Koniec”
-command exit
```

Checkbutton – przycisk do zaznaczania

```
checkbutton widget „opcja wartość”
```

Widget checkbutton tworzy przycisk, który można zaznaczyć lub odznaczyć. Do przycisku można dodać polecenia, które mają zostać wykonane po jego naciśnięciu. Można zdefiniować wartości dla stanu włączonego i załączonego. Przykłady:

```
checkbutton.cb -text „Zaznacz
lub odznacz” -variable zmienna
checkbutton.cb -text „Zaznacz
lub odznacz” -onvalue ON \
-offvalue OFF -variable zmienna
checkbutton.cb -text „Zaznacz
lub odznacz” -variable zmienna \
-command {tk_messageBox -
message „A kuku”}
```

Radiobutton – przycisk wyboru

```
radiobutton widget „opcja wartość”
```

Widget radiobutton tworzy przycisk wyboru. Zwykle przyciski wyboru występują w grupach. Do przycisku można dodać polecenia, które mają zostać wykonane po jego naciśnięciu oraz wartość zmiennej po zaznaczeniu widgetu. Przykład:

```
radiobutton.rb1 -text „Wybierz
A” -variable zmienna -value A
radiobutton.rb2 -text „Wybierz
B” -variable zmiemna -value B
radiobutton.rb3 -text „Wybierz
C” -variable zmienna -value C
```

Wybór któregoś z przedstawionych przycisków powoduje przypisanie zmiennej zmienna wartości A albo B albo C.

Frame – ramka

```
frame widget „opcja wartość”
```

Widget frame przeznaczony jest do grupowania widgetów podrzędnych względem ramki, dla łatwiejszego nimi zarządzania. Można na przykład przesunąć ramkę lub zamienić miejscami z innym widgetem, a zawartość ramki nie ulegnie przestawieniu. Przykład:

```
frame.f
button.f.b1 -text „Koniec” -
command exit
label.f.l -text „naciśnij
przycisk aby zakończyć”
```

Labelframe – ramka z tytułem

```
labelframe widget „opcja wartość”
```

Widget labelframe ma identyczne przeznaczenie jak poprzedni widget frame. Dodatkowo ramka zaopatrzona jest w tytuł. Przykład:

```
labelframe.f -text „Tytuł ramki”
button.f.b1 -text „Koniec” -com-
mand exit
label.f.l -text „naciśnij przy-
cisk aby zakończyć”
```

Pozostałe widgety

Dostępne są oczywiście bardziej zaawansowane widgety, ale ze względu na ich złożoność opowiem o nich następnym razem.

Entry – pole do wpisania tekstu > Widget jednowierszowego pola edycji

Text – pole testowe

Rozbudowany widget wyświetlania i edycji tekstu.

Listbox – lista wyboru

Widget pozwala na utworzenie listy, które elementy można zaznaczać, a także dokonywać wielokrotnych wyborów.

Scrollbar – suwaki

Suwaki można dodawać do większości widgetów. Należy pamiętać o podaniu kierunku (poziomo albo pionowo) położeniu suwaków.

Menu > Widget to tworzenia menu rozwijalnego.

Typowe okna dialogowe

Obok prostych widgetów Tk oferuje gotowe typy okien dialogowych, np. okno zapisu pliku czy wyboru koloru. W następnym artykule zostaną zaprezentowane wraz z odpowiednimi przykładami.

Listing

```
#!/bin/sh
#the next line restarts using wish \
exec tclsh „$0” „$@”

# wywołanie biblioteki graficznej
package require Tk

# Pierwsza ramka
labelframe .lf1 -text „Radiobutton”
# Zawartość ramki: przyciski i etykieta
radiobutton .lf1.rb1 -text „Wybierz A” -variable zmienna1 -value A
radiobutton .lf1.rb2 -text „Wybierz B” -variable zmienna1 -value B
radiobutton .lf1.rb3 -text „Wybierz B” -variable zmienna1 -value C
label .lf1.l -textvariable zmienna1
# wstawienie ramki w pierwszej kolumnie i pierwszym wierszu
grid .lf1 -row 1 -column 1 -padx 2m -pady 2m
# wstawienie przycisków i etykiety do ramki
pack .lf1.rb1 .lf1.rb2 .lf1.rb3 .lf1.l

# Druga ramka
labelframe .lf2 -text „Checkbutton”
label .lf2.l1 -text „Wartość zmiennej”
label .lf2.l2 -textvariable zmienna2
checkbutton .lf2.cb -text „Zaznacz/Odznacz” -variable zmienna2
grid .lf2 -row 1 -column 2 -padx 2m -pady 2m -sticky ns
pack .lf2.l1 .lf2.l2 .lf2.cb

# Trzecia ramka
frame .f3 -borderwidth 2 -relief groove
button .f3.b -text „Koniec” -command exit
# wstawienie najpierw przycisku do ramki
pack .f3.b
# a następnie wyświetlenie gotowej ramki rozciągniętej na dwie kolumny
grid .f3 -row 2 -column 1 -columnspan 2 -padx 2m -pady 2m -sticky we
# Koniec
```

Wspólne opcje widgetów

Krawędzie

`-relief styl`

Opcja określa wygląd pseudoprzestrzenny krawędzi. Dostępne style:

> flat – płaskie, bez krawędzi,
> groove – wyżłobienie,
> raised – wypukłe,
> ridge – krawędziowe,
> solid – ciągłe,
> sunken – wklęsłe.
> borderwidth szerokość

Szerokość krawędzi podaje się w pikselach.

Rozmiary

-width szerokość
-height wysokość

Domyślną jednostką szerokości i wysokości jest piksel. Można podać inną jednostkę z poniższych:

> m – milimetry, np. 2 m
> c – centymetry, np. 1c
> i – cale, np. 4i
> p – piksele, np. 12p

Umieszczanie widgetów

Widget po umieszczeniu nie jest wyświetlany, oczekuje na umieszczenie go. Takie podejście znakomicie ułatwia tworzenie aplikacji, gdyż możemy tworzyć widgety w kolejności nam wygodnej, niekoniecznie w kolejności ich pojawiania się na ekranie.

Wykorzystanie zarządcy pack

`pack widget „widgety opcja wartość”`

Figure 1: Przykład prostego okna aplikacji.

Do polecenia pack można dodać polecenia sterujące kierunkiem układania widgetów czy odstępami pomiędzy nimi.

Wykorzystanie zarządcy grid

`grid widget „widgety opcja wartość”`

Przykład prostego okna aplikacji

Treść przykładu można zapisać do pliku, który potem uruchamiamy lub uruchomić interepreter Tcl-Tk za pomocą polecenia tclsh lub wish, wpisywać kolejne poleceni a przykładu i obserwować wyniki na ekranie - **listing** obok.

Zakończenie

Mam nadzieję, że czytelnicy spróbowali zbudować pierwsze okienka interfejsu oraz że się podobało. W następnej części przedstawię pozostałe standardowe widgety i ciekawszy przykład.

[EOT]

Open TTD

Jesień już za pasem, a tym samym wizja długich i szarych dni. Oczywiście, poza nauką;), popijaniem dla relaksu złocistego napoju:) i kolejną kompilacją kerneli warto by było zająć się też czymś innym.

Rafał Domeracki, Piotr Szewczuk

My polecamy rozrywkę dla umysłu w postaci gry Open TTD.

Open TTD jest strategiczną grą ekonomiczną wydaną na licencji GPLv2 jako reaktywacja gry Transport Tycoon Deluxe firmy Microprose. Ze strony domowej można pobrać wersję instalacyjną do skompilowania. W zależności od posiadanej dystrybucji Linuxa są też dostępne gotowe pakiety instalacyjne.

Oprócz obsługi wielu systemów operacyjnych ciekawą funkcją jest możliwość gry poprzez sieć, co pozwala na jeszcze bardziej pasjonującą rozrywkę, w tym prowadzenie firmy przez kilka osób.

Gra Open TTD pozwala wcielić się graczom w rolę prezesa firmy transportowej. Oczywiście, głównym zadaniem graczy jest pokonanie konkurencji i osiągnięcie jak najlepszych wyników finansowych. Nie jest to bynajmniej zadanie łatwe. Gra pokazuje wiele aspektów prowadzenia firmy. Należy dbać o przychody, wydatki, koszty stałe, nowe zamówienia i, oczywiście, mieć na uwadze poczynania konkurencji (na szczęście w grze przedsiębiorców nie gnębią politycy tak jak ma to miejsce w naszej rzeczywistości). Jako firma transportowa możemy zajmować się takimi działami jak: transport kolejowy, drogowy, morski i powietrzny. Oczywiście, możemy mieszać typy działalności i zajmować się kilkoma typami przewozów. Główne zadania, jakie możemy realizować, to:

> przewóz osób – pomiędzy przystankami, lotniskami czy też dworcami kolejowymi różnych miast,

> przewóz surowców (w tym węgla, rudy żelaza, ropy, drewna) pomiędzy miejscem wydobycia a fabrykami,

> przewóz żywności i zboża z farm do zakładów produkcyjnych.

Grę zaczynamy z budżetem 100 000 funtów i możliwością wzięcia kredytu w porcjach po 10 000 funtów w dowolnym momencie działalności. Wielkość kredytu jest ograniczona i możemy go powiększać aż do maksymalnej kwoty 300 000. Do realizacji zadań mamy różne typy pojazdów, które z upływem kolejnych lat działalności firmy oraz rozwojem przemysłu są coraz to lepsze i bardziej nowoczesne.

Jesienna aura widać ogarnęła również świat Open TTD

Na początku rozwoju firmy musimy zapoznać się z „zapotrzebowaniem" rynku, czyli ustalić, gdzie znajdują się największe miasta, fabryki, kopalnie, farmy itp. Przy tworzeniu połączeń czeka nas budowa infrastruktury: dróg, linii kolejowych, peronów, lotnisk, przystanków autobusowych itd.

Należy pamiętać, że przewóz towarów i surowców odbywa się ściśle według zapotrzebowania, czyli np. drewno należy dostarczać do tartaków, a rudę żelaza do hut. Sposób, w jaki przewieziemy zasoby, zależy już od nas, jednak dobrze jest trzymać się poniższych zasad:

Jak widać, mamy ogromny wpływ na wszelkie czynniki wirtualnego świata

Takiej aglomeracji nie powstydziłby się nawet prezydent Warszawy...

> transport drogowy stosować na krótkich odcinkach, np. przewóz osób w obrębie blisko położonych miast czy też przewóz towarów z portów do blisko położonych fabryk,

> transport kolejowy – najlepiej nadaje się na długich trasach, a zwłaszcza do przewozu „ciężkich zasobów": drewna, węgla, ropy, rud żelaza, stali,

> transport lotniczy – idealnie nadaje się do przewozu osób pomiędzy miastami na większych odległościach,

> transport morski – tą drogą można transportować ropę z pól naftowych do portów, a później do rafinerii.

Grę Open TTD można polecić w zasadzie każdemu, od najstarszych aż po najmłodszych, co w dzisiejszym świecie zdominowanym przez brutalne gry jest szczególnie ważne. Co prawda, grafika nie jest może na najwyższym poziomie, ale gra prezentuje się ładnie, a zasadniczo ważniejszym aspektem jest tutaj grywalność, która stoi na najwyższym poziomie.

Według nas Open TTD ma tylko jedną podstawową wadę. Jak już ktoś zacznie grać, to nie będzie mógł przestać: GORĄCO POLECAMY! [EOT]

Wymagania:
> min. 64 MB RAM,
> min. 10 MB wolnego miejsca na dysku,
> system operacyjny: Windows (min. 133 mHz i DirectX 7), Linux (min. 75 mHz i biblioteki SDL), Mac OS X: 200 mHz, FreeBSD, NetBSD,
> karta dźwiękowa.

Adres domowy:
http://www.openttd.org