Old Man GURU Magazine

Wychodzi bardzo nieregularnie, kiedy wydaje mi się,
że mam coś ciekawego lub pożytecznego do napisania...

Numer 2/2009 **25 sierpień 2009 r.**

Ratujmy polski Internet!!!

Wolność wypowiedzi w Internecie staje się coraz bardziej zagrożona. Pojawiają się coraz częściej projekty, które na szczęście są jak dotychczas odrzucane monitorowania zawartości serwisów oraz umieszczanych na nich wpisów internautów.

Niestety, sami internauci dostarczają coraz więcej argumentów zwolennikom wprowadzenia cenzury w sieci. Motywowane jest to wzrostem liczby obraźliwych wpisów naruszających dobra osobiste znanych postaci, stosowaniem słów uznanych za obraźliwe lub wręcz przekleństw i tak zwanych "przerywników" itp. Akcja "Chamstwo hula w Internecie" rozpoczęta (po wycofaniu się przez p.Dorotę Świeniewicz z gry w reprezentacji Polski na skutek obraźliwych wpisów na forach internetowych) przez "Gazetę Wyborczą" zatacza coraz szersze kręgi - ostatnio (18 sierpnia 2009) wypowiedział się w tej sprawie także Rzecznik Praw Obywatelskich RP - Janusz Kochanowski.

Szlachetne w swej intencji próby powstrzymania chamstwa, które rzeczywiście stało się powszechne w wypowiedziach umieszczanych w internecie oraz nagłośnienie tego problemu są jednak bardzo niebezpieczne - ponieważ mogą stać się argumentem za wprowadzeniem cenzury sieciowej - co więcej z pełnym przyzwoleniem większości społeczeństwa zmanipulowanego przez media.

Internet nie jest powodem szerzenia się chamstwa. Wystarczy przejść się po parku, przejechać autobusem lub tramwajem aby zapoznać się z całym "bogactwem" języka polskiego. Język wielu mediów (zarówno części prasy, jak i radia telewizji) również pozostawia wiele do życzenia. Artykuły prasowe i wypowiedzi telewizyjne - zarówno dziennikarzy, jak i zapraszanych gości nie są także pozbawione wycieczek osobistych przekraczających granice przyzwoitości.

Internet jest (i będzie) taki, jacy ludzie z niego korzystający - trudno oczekiwać, aby zamieszczane w nim wypowiedzi były wolne od epitetów,a nawet przekleństw używanych na co dzień przez znaczną część naszego społeczeństwa.

Akcja przeciwko chamstwu w internecie może mieć jednak bardzo poważne konsekwencje dla wolności wypowiedzi - a w konsekwencji dla wolności w ogóle! Musimy się przed tym obronić. Jedyną metodą jest piętnowanie osób naruszających zasady netetykiety, ignorowanie wszelkich postów lub innego typu wpisów, które zawierają przekleństwa lub inne słowa czy określenia obraźliwe, a także niewybredne ataki personalne. Osoby umieszczające takie wpisy powinny się spotykać z totalnym brakiem zainteresowania. Jedynie w taki sposób stracą motywację do prezentowania tworów swojego "dowcipu". Zasada "nie karmić Trolla" powinna być wobec takich osób bezwzględnie stosowana.

Podobny bojkot powinniśmy zastosować wobec serwisów internetowych, które naruszają dobre obyczaje. Jeśli nie będą one wzbudzać zainteresowania - to po prostu same znikną!

Jako osoba reprezentująca Fundację żywotnie zainteresowaną zachowaniem wolności wypowiedzi w Internecie pozwalam sobie na publikację tego apelu w nadziei, że większość użytkowników tego wspaniałego i ciągle niezależnego medium swobodnej wymiany informacji będzie w stanie zmarginalizować niebezpieczeństwo związane z pojawiającymi się chamskimi wypowiedziami oraz serwisami.

Tomasz Barbaszewski
Przewodniczący Rady Ekspertów FWiOO (www.fwioo.pl)

Troszkę o X Window

Sam X Window Server to jakby czyste płótno, na którym dopiero możemy budować środowisko graficzne podobnie jak malarz sztalugowy. Jakie? To już zależy od potrzeb, które ma ono spełnić, jednakże musimy pamiętać, że korzystać ze środowiska przygotowanego przez X Server mogą korzystać jedynie odpowiednio przygotowane programy zwane X Klientami (X Clients).

X klienci mogą działać na tym samym komputerze (pamiętajmy, że UNIX/LINUX to systemy wielozadaniowe) lub na dowolnym komputerze dostępnym w sieci. Oznacza to, że na nasz X Server możemy skierować wyjście i wejście programu działającego na zupełnie innej maszynie, innej wersji systemu itp. Jedynym warunkiem jest jego zgodność z protokołem komunikacji X Window.

Jest to bardzo ważna cecha X Window - w przeciwieństwie do innych, podobnych z wyglądu ekranu rozwiązań X Window był od początku swego istnienia systemem sieciowym, którego zadaniem było udostępnienie wszystkich zasobów sieci użytkownikowi tak zwanego X terminala. W systemie X Window użytkownik wybierając program za pomocą ikony lub pozycji menu nie musi wiedzieć, na jakiej maszynie w sieci uruchamiany jest wykorzystywany przez niego program - jeśli tyko posiada uprawnienia do jego wykorzystywania przejmuje nad nim pełną kontrolę i pracuje w taki sam sposób, jakby był to program lokalny. O całość komunikacji sieciowej "martwi się" X Window.

Stąd właśnie pochodzi nazwa naszej końcówki - ABA-X (oraz kolejne wersje 2 i 3) - ponieważ jest to właściwie X terminal wyposażony w dodatkowe programy i protokoły komunikacyjne (np. RDP i ICA do komunikacji z MS Windows).

Najprostszym przypadkiem jest uruchomienie jednego X Klienta. Wówczas nie będą nam potrzebne żadne inne programy - poza pracującym X Serwerem i uruchamianym X Klientem. Na początek załóżmy, że pracują one na tym samym komputerze. Oto bardzo prosty przykładzik:

```
$ X :3 &
$ xclock
```

i wynik jego wykonania:

Zegarek możemy umieścić w dowolnym miejscu:

```
xclock -geometry +50+100 -display :3
```

lub wyświetlić zegarek na całym ekranie:

xclock -geometry 800x600 -display :3

a może ładniejsze będą czerwone wskazówki?

xclock -geometry 800x600 -hd red -display :3

Ale koniec z tą zabawą - zabieramy się do pracy:

soffice -display :3

Z niektórymi programami (np. przeglądarka Firefox) trzeba postąpić nieco inaczej, ponieważ nie obsługują one opcji -display. Wówczas przed uruchomieniem należy ustawić zmienną systemową, która skieruje wyjście programu do określonego okna:

```
$ DISPLAY=:3
$ export DISPLAY
$ firefox
```

A co jeśli nasz X Klient ma działać na innym komputerze w sieci? Wówczas należy go uruchamiać w następujący sposób:

xclock -geometry 800x600 -hd red -display 10.1.1.25:3

czyli uzupełnić zmienną DISPLAY o prefix będący adresem IP lub kwalifikowaną nazwą maszyny, na której pracuje X Serwer! I tylko tyle!
Celowo pominąłem tu problem autoryzacji dostępu do X Serwera, o której będzie w dalszych odcinkach o X Window.

Jak można łatwo wywnioskować z powyższego opisu podstawową właściwością X Window jest elastyczność - umożliwia to pracę zarówno z lokalnymi, jak i zdalnymi (sieciowymi) X klientami - w zasadzie w identyczny sposób. Po prostu w przypadku lokalnych X Klientów jako nazwa IP jest po prostu przyjmowana domyślnie nazwa "localhost"

Powyższe przykłady opisują, w jaki sposób można przekazać sterowanie uruchamianego programu wybranemu X serwerowi - niezależnie od tego, czy działa on na tej samej maszynie, czy na innym komputerze dostępnym w sieci. Jednak najczęściej mamy do czynienia z odwrotną sytuacją - użytkownik X terminala chce uruchomić program na innej maszynie dostępnej w sieci, ale w taki sposób, aby dysponować jego wejściem i wyjściem na swym X terminalu.

Ponieważ w tej części opisu zająłem się "czystymi" X Window (bez dodatkowych ułatwiających obsługę środowiska graficznego programów zarządzających oknami, systemami menu, obsługą pulpitu itp.) zrealizujemy to zadanie za pomocą najprostszych komend.
Jak zapewne wszyscy wiedzą, aby wydać jakąkolwiek komendę w systemie musimy dysponować interpreterem komend użytkownika tej maszyny, której chcemy wydać polecenie.

Najprostszym (i bezpiecznym) rozwiązaniem jest wykorzystanie Secure Shell (SSH). Oczywiście na zdalnej maszynie musi być uruchomiony serwer ssh (sshd). Wówczas wydana na terminalu (przykładowy IP=10.1.1.10) komenda:

ssh -X tomekb@10.1.1.25 xclock -display 10.1.1.10:0

zażąda uruchomienia programu xclock (znów ten zegarek) na maszynie o numerze IP 10.1.1.25 z uprawnieniami użytkownika tomekb i przekazanie jego wejścia i wyjścia na pierwszy (0) X Serwer maszyny 10.1.1.10 (czyli naszego X terminala).

Następnie zostaniemy zapytani o hasło (do maszyny 10.1.1.25!), po którego podaniu na naszym ekranie (maszyna 10.1.1.10) pojawi się zegarek - ale warto pamiętać, że jest on wynikiem pracy programu na maszynie 10.1.1.25!

Niektóre programy użytkowe (Firefox, OpenOffice itp.) nie dopuszczają bezpośredniego przekierowania z użyciem opcji -display i w takiej sytuacji musimy ją ustawić przed wywołaniem ssh, która przekaże jej aktualną wartość zdalnej maszynie automatycznie:

```
DISPLAY=10.1.1.10:0
export DISPLAY
ssh -X tomekb@10.1.1.25 firefox
```

Ktoś może zapytać, po co ta cała zabawa? Przecież mamy gotowe środowiska graficzne - GNOME, KDE itp. Tak, ale jeśli na przykład chcemy uruchomić jedynie przeglądarkę (np. Firefox) w trybie pełnoekranowym to czy na pewno potrzebujemy całego wielkiego KDE? Czy nie wystarczy sam X serwer i Firefox (pracujący lokalnie lub zdalnie)? A jeśli jeszcze zaopatrzymy naszą przeglądarkę w odpowiednie dodatki - np. FullScreen to korzystając z powyższych przykładów będziemy mogli uruchomić terminal umożliwiający korzystanie jedynie ze stron WWW (oraz aplikacji pracujących w jej środowisku).

W identyczny sposób możemy postąpić z innymi programami - choćby z popularnym OpenOffice.

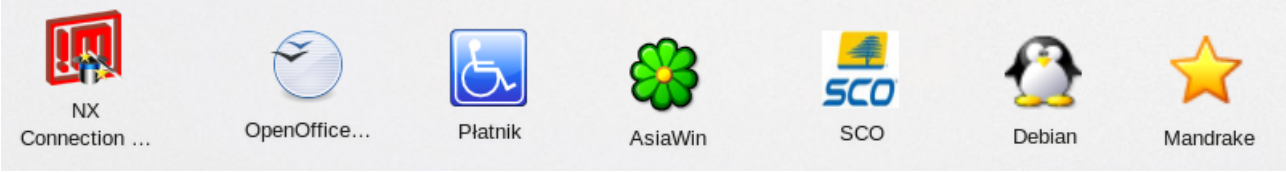
Oczywiście nie będziemy wymagać od użytkownika wprowadzania powyższych komend. Po prostu dopiszemy je do odpowiednich plików startowych naszego systemu w taki sposób, aby całe środowisko użytkownika było uruchamiane automatycznie po włączeniu zasilania...

UWAGI:
Wszystkie powyższe komendy przetestowałem w praktyce na systemie Open SuSE 11.1. Starałem się jednak wykorzystywać składnię niezależną od systemu (np. zamiast export DISPLAY=10.1.1.10:0 ustawienie zmiennej systemowej w 2 krokach, tak, jak to jest przyjęte w starszych powłokach systemu UNIX).

Przy konstruowaniu przykładów zwracałem uwagę jedynie na ich prostotę pomijając względy bezpieczeństwa! Wszystkich pragnących zastosować te rozwiązania w praktyce proszę o zastosowanie się do zaleceń dokumentacji systemu X Window oraz SSH.

W następnej części w numerze 3 zajmiemy się okienkami i uruchamianiem programów (zarówno lokalnych, jak i zdalnych) za pomocą systemu POP-UP Menu.

A oto (dla zachęty) fragment ekranu mojego notebooka (Open SuSE LINUX 11.1):



Wybór dowolnej ikonki powoduje uruchomienie żądanego programu lub połączenia z wybranym serwerem.

Dla użytkownika (czyli w tym przypadku dla mnie) nie ma żadnego znaczenia, czy uruchamia program lokalny - czy zdalny. Po prostu "się klika" i już. Na przykład ikonka "Płatnik" powoduje otwarcie okna z programem ZUS Płatnik, który oczywiście uruchamia się na maszynie z systemem MS Windows. Po zakończeniu pracy z "Płatnikiem" jego okno po prostu znika. Właściwie skonfigurowany system po prostu pozwala użytkownikowi pracować i korzystać z zasobów sieci niezależnie od tego, na jakim serwerze są one udostępnione.

To właśnie (od bardzo dawna) potrafi X Window!

Terminal MS Windows pod Linuksem

Od dłuższego już czasu dostępne jest oprogramowanie, które umożliwia komputerom pracującym pod kontrolą Linuksa korzystanie z oprogramowania pracującego na serwerach usług terminalowych Microsoftu.

Ze względu na popularność "jedynie słusznego systemu) w wielu przypadkach zadaniem terminala jest tylko dostęp do programów przeznaczonych jedynie dla środowiska MS Windows wymagających oczywiście środowiska graficznego.

Obsługę protokołu RDP (Remote Desktop Protocol, który jest odmianą protokołu T.128 ITU (Międzynarodowa Unia Telekomunikacyjna) zapewnia opracowane przez Matta Chapmana http://www.cse.unsw.edu.au/~matthewc/ oprogramowanie rdesktop obecnie dostępne w wersji 1.6.0 na stronie www.rdesktop.org .
Jest ono udostępniane na licencji GPL - niestety,wielu dostawców terminali i cienkich klientów "zapomina" o obowiązku wymienienia Autora oraz zapewnieniu dostępu do kodu źródłowego, co jest oczywistym obowiązkiem zawartym w licencji...

Przyjrzyjmy się jednak bliżej wymaganiom i możliwościom tego oprogramowania:

Rdesktop wymaga do pracy środowiska X Window - tak więc, jak już napisałem w I numerze "OldMan GURU" przed uruchomieniem programu rdesktop musimy uruchomić X Window serwer. Kolejnym krokiem jest poprawne ustawienie zmiennej DISPLAY i uruchomienie programu rdesktop z odpowiednimi opcjami - a jest tych opcji sporo!

Spróbujmy najważniejsze z nich uszeregować:

Dane o użytkowniku:
Nazwa użytkownika: -u <Logon_name>
Domena: -d <domain_name>
Hasło dostępu -p <password>: Ta opcja działa tylko dla wersji RDP powyżej 5, tak więc automatyczne logowanie jest możliwe tylko do serwerów MS Windows 2003 i późniejszych. Dostęp do MS Windows 2000 zawsze wymaga podania hasła.
Warto także pamiętać, że jeśli zapiszemy linie komend (lub odpowiadający jej plik konfiguracyjny), to hasło dostępu do serwera MS Windows zostanie zapisane w jawnej (niezaszyfrowanej!) postaci.

Środowisko użytkownika:
Katalog docelowy: -c <nazwa_katalogu>
Uruchamiany program: -s <nazwa_programu>
Z tymi opcjami wiąże się szereg pytań. Jeśli nie zostaną one wprowadzone to katalogiem docelowym będzie domowy katalog użytkownika określony w jego profilu, oraz zostanie uruchomiony pulpit użytkownika systemu MS Windows.
Jeżeli chcemy dopuścić korzystanie tylko z jednej aplikacji to wprowadzenie opcji -s <nazwa_programu> spowoduje, że ten program zostanie uruchomiony automatycznie, a po zakończeniu jego pracy połączenie RDP zostanie rozłączone. W ten sposób realizowany jest tryb pracy "Application publishing".

Identyfikatory połączenia:
Opcje te pełnią istotną rolę jeśli chcemy korzystać z mapowania zdalnych zasobów i urządzeń:
Nazwa klienta RDP: -n <nazwa> jest to nazwa pod którą MS Windows będzie "widział" klienta RDP,
Nazwa okna RDP: -T <tytuł_okna>

Ekran i klawiatura:
Wymiary okna: -g <800x600> lub -f (pełny ekran),
Mapa klawiatury: -k <pl, en-us, cz...>
Liczba kolorów: -a <8, 16, 24 lub 32> (dla Windows 2000 tylko 8!)

Parametry sieciowe:
Kompresja transmisji RDP: -z
Dostępne pasmo sieciowe: -x <modem, SDI, LAN>
Wyłączenie szyfrowania: -e (w obu kierunkach), -E (tylko klient -> serwera)

Mapowanie urządzeń:
Ważne - urządzenia są mapowane za pomocą kanałów wirtualnych, które powstają po nawiązaniu sesji RDP i dziedziczą uprawnienia użytkownika sesji!

Włączenie mapowania: opcja -r może być używana kilkakrotnie i dotyczy:
-r comport:COM1=/dev/ttyS0 (lub odpowiednio COM2),
-r disk:<share_name>=<mount_directory> np. -r disk:cdrom=/mnt/cdrom
Można mapować dyski USB Storage (PenDrive, aparaty foto itp., ale należy zachować ostrożność przy ich usuwaniu - najlepiej je po prostu wcześniej odmontować (można przygotować odpowiedni skrypt).
-r lptport:LPT1=/dev/lp0 (lub LPT2),
-r sound:[local[driver:[device]],remote] sterowniki obejmują alsa i oss,
-r scard: "<linux_name>"="<Windows_name>"
Korzystanie z kart chipowych lub tokenów wymaga biblioteki PCSC (wystarcza wersja Lite).

Uff, a to jeszcze nie wszystko, ponieważ pominąłem mniej popularne opcje. Linie komendy rdekstop może się więc ciągnąć i ciągnąć - np.:

rdesktop -kpl -a24 -utomekb -dABA -f -c/user/dokumenty -soffice -nterm1 -TWin2003 -z -xlan -r lptport:LPT1=/dev/lp0 -r disk:Term1_USB=/dev/usb/1 -r sound:local 10.1.1.221:3256

Komu chciałoby się za każdym razem pisać coś takiego!

Przyjęto więc zasadę konfigurowania sesji RDP za pomocą konfiguratora graficznego, a następnie zapis wszystkich parametrów sesji pod wybraną nazwą. Jest to powszechnie stosowana praktyka - podobnie rozwiązano konfigurację sesji protokołów ICA, NX itp. Można zapisać dowolnie wiele konfiguracji - a uruchamiać tą, która aktualnie jest potrzebna.
Rozwiązanie to ma jeszcze jedną zaletę - pliki z opisami sesji są jednak niewielkie. Można je łatwo pobierać z serwerów sieciowych (nawet przez sieci rozległe) lub przesyłać na terminale (stacje robocze). Administrator może więc zmieniać konfigurację terminala lub grup terminali nie ruszając się z miejsca...
Niestety, większość "konfiguratorów" uwzględnia bardzo niewiele możliwości oferowanych przez protokół RDP - a tym samym oprogramowanie rdesktop.

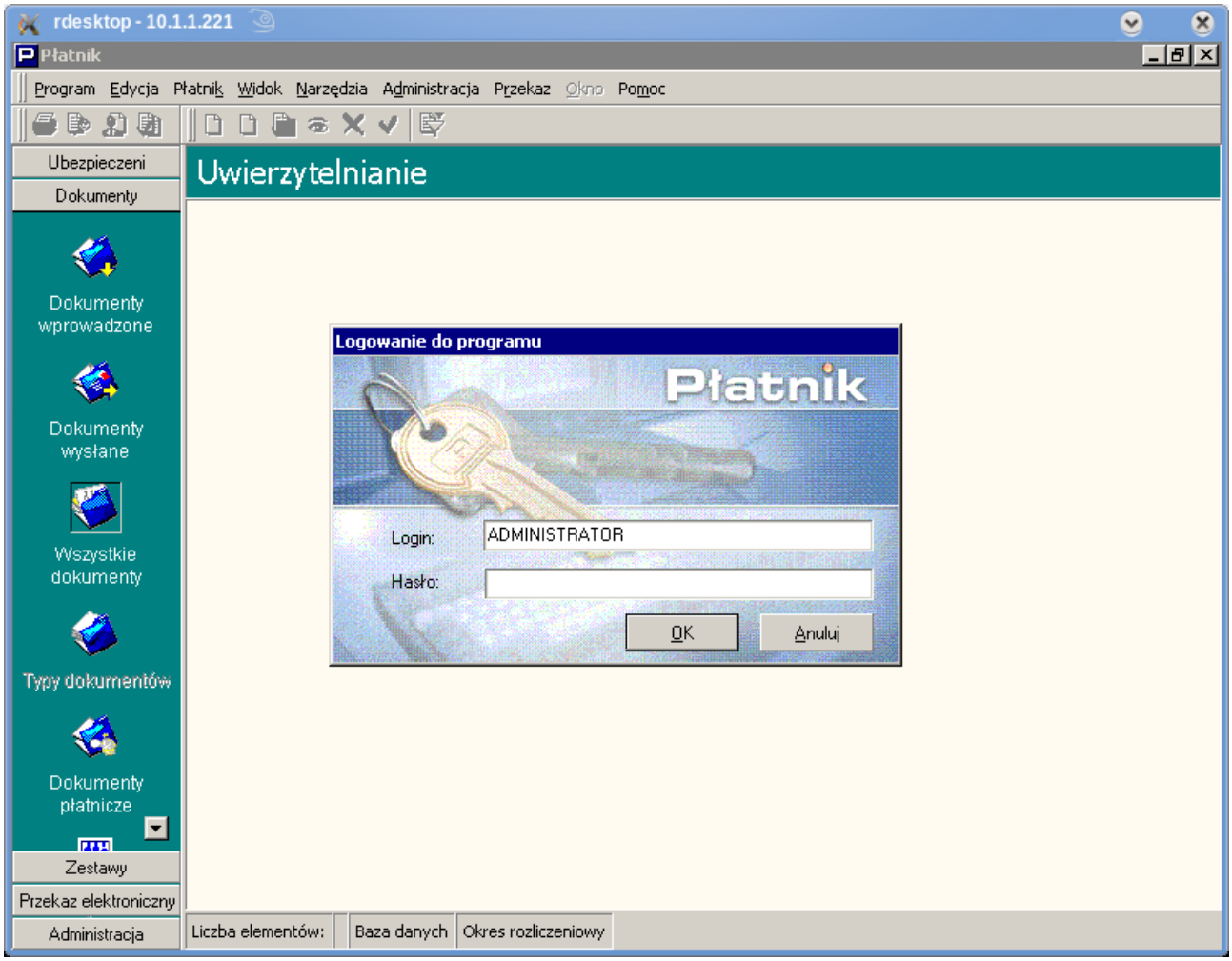
Gnome-RDP
Pozwala ustawić tylko najważniejsze opcje - zaś sesje zapisuje w formacie .db . Nie można ich więc poprawić "ręcznie".

tsclient
Również nie oferuje zbyt bogatego zestawu opcji (choć więcej niż gnome-rdp), wymaga sporej liczby bibliotek. Największą zaletą tsclient jest jawny format zapisu sesji, co daje możliwość "ręcznego" uzupełnienia opcji.

ABA-X3
Oparty o prosty formularz RDP i skrypty CGI. Pozwala ustawić dużą liczbę opcji (z mapowaniem wielu urządzeń oraz obsługą SMART CARD włącznie). Wymaga serwera WWW (może być bardzo prosty - np. zawarty w BusyBox). Zapewnia dostęp lokalny i sieciowy (z WAN włącznie). Zapis sesji w formacie jawnym z możliwością wprowadzania poprawek.

A oto rezultat wykonania komendy:

rdesktop -c "C:\Program Files\PROKOM Software SA\Płatnik 7" -s "C:\Program Files\PROKOM Software SA\Płatnik 7\P2.exe" 10.1.1.221
czyli słynny program "Płatnik" na ekranie komputera z Linuksem.



W tym przypadku "Płatnik" (lub jakikolwiek program pracujący na serwerze MS Windows) otwiera się w oknie na pulpicie KDE. Nic jednak nie stoi na przeszkodzie, aby uruchomić płatnika na pełnym ekranie - np. dostępnym na 8 konsoli Linuksa <Ctrl><Alt>F8. Trzeba jednak pamiętać, aby najpierw uruchomić tam X serwer w tle - ale o tym już było.

Na końcu podłączamy naszą komendę pod ikonkę KDE (patrz wyżej) i możemy zapomnieć o jej składni.

Mała pułapka - program rdesktop nie może wystartować nim X serwer "zdąży wstać". Jak to sprawdzić - no cóż, kłania się "skryptologia". Mało ambitnym rozwiązaniem może być wstawienie polecenia sleep:

X :2 &
sleep 20
rdesktop .....

Ale to sposób dla naprawdę mało ambitnych - prawdziwi Guru's sprawdzą, czy program X serwer już skończył swój start. Jak - no właśnie, jak to zrobić? Oto moja zagadka.

"Skryptologia" - powrót z zapomnienia...

Prawie każdy programista wie, że powłoka w systemach Linux/Unix to interpreter komend, a którym można pisać całkiem przyzwoite programy zwane potocznie skryptami. Jednak sztuka "skryptologii" na przełomie wieków XX i XXI zaczęła zanikać.

Pomimo, że "Na nowo narodzona" (Born Again) powłoka bash, której nazwa oddaje w dość przewrotny sposób cześć autorowi najpopularniejszej powłoki systemu UNIX (Bourne shell) uzyskała szereg nowych, atrakcyjnych właściwości w porównaniu ze swym pierwowzorem, to nowocześniejsze języki interpretowane takie jak Perl, Python czy TcL i oczywiście PHP wydały się na tyle atrakcyjne, że zapomniano prawie zupełnie o poczciwej powłoce systemowej.

Skrypty powłokowe oczywiście cały czas stosowano, jednak powierzano im albo bardzo proste zadania, albo wykorzystywano podczas startu samego systemu do uruchamiania jego funkcji lub modułów.

Stało się tak między innymi dlatego, że znacząco potaniały twarde dyski! Wielu moich rozmówców nie potrafi sobie dziś wyobrazić, że na moim PC (jednym z najszybszych wówczas na uczelni!) na dysku twardym o pojemności 40 MB (tak, nie ma tu pomyłki - 40 Megabajtów) z powodzeniem zmieściły się dwa systemy operacyjne - DOS 3.30 oraz SCO XENIX! A gdzie miejsce na interpreter np.Perla, który na moim Linuksie (Open SuSE 11.1) ma wielkość 36,2 MB? O modułach dodatkowych już nie wspomnę.

No tak, ale kto by się tym przejmował mając do dyspozycji najmniejszy z obecnie dostępnych dysków, którego pojemność to "jedynie" 80 GB?! Przecież taki Perl zajmie mniej niż promil dostępnej pojemności...

Sytuację zasadniczo zmieniły systemy typu Embedded. Kiedy zaczęły one powstawać fotografia cyfrowa jeszcze raczkowała i pamięci typu FLASH były bardzo drogie. Ponownie zaczęliśmy się więc liczyć z każdym megabajtem. Pierwsza generacja naszych terminali ABA-X zadowalała się pamięcią FLASH o pojemności 8 MB (i zawierała środowisko graficzne - oczywiście X Window). Później historia (jak to ona) zaczęła się powtarzać - ABA-X2 wymagała już 32 MB flasha. Udało nam się utrzymać tą wielkość także dla 3 generacji jako minimum (chodziło o zachowanie możliwości uaktualnienia bez zmian sprzętowych) - ale wymagania klientów rosły - a pamięci FLASH taniały...

Jednak w dalszym ciągu (z wielu względów) praktycznie wszyscy dostawcy systemów tego typu starają się stosować RAM dyski do organizacji systemów plikowych. A tu pojemności są znacznie mniejsze, a konkurencja programów (lokalna przeglądarka, JAVA itp.) w wyścigu do pamięci jest spora.

Powłokę systemową (nawet ograniczoną jak powłoka o przewrotnej nazwie ash z BusyBoxa) jednak trzeba mieć - no bo i jak w prosty sposób uruchomić bez niej system?

Przypomnieliśmy więc sobie wszyscy o starych, dobrych skryptach powłokowych. I okazały się one całkiem atrakcyjne i efektywne. Być może ich możliwości są mocno ograniczone, ale wielu uważa też Fortran za prymitywny, przestarzały i niegodny zainteresowania język - cóż, w mechanice funkcjonuje takie powiedzenie: "Im gorszy majster, tym trzeba mu zapewnić lepsze narzędzia".

Powłoka zapewnia nam nie tylko wykonywanie instrukcji prostych lub złożonych, lecz także instrukcji warunkowych, różnego rodzaju pętli, funkcji i procedur zewnętrznych - a nawet obliczeń zmiennoprzecinkowych (pod warunkiem dostępu do kalkulatora systemowego bc, który ma tylko 300 kB). Odpowiednie sterowanie strumieniami wejścia/wyjścia pozwala pisać w powłoce (nawet ash) także skrypty CGI (taki mechanizm zastosowałem w konfiguratorze WWW dla terminali ABA-X3). A co najważniejsze - skrypty powłokowe zajmują w systemie tyle miejsca, ile same "ważą" - ponieważ do ich wykonania nie jest potrzebny żaden dodatkowy interpreter!

Nie musimy nic kompilować, konsolidować - po prostu piszemy w dowolnym (oczywiście honor każe korzystać z vi !) edytorze i od razu sprawdzamy, czy nasz pomysł działa. Jeśli się pomylimy - poprawki można wprowadzać i kontrolować, a sam kod debugować na bieżąco! I w dodatku w bardzo łatwy sposób możemy zadbać o to, aby nasze skrypty funkcjonowały na każdym systemie.

A więc - Niech Żyje i Rozkwita nam SKRYPTOLOGIA !