

Old Man GURU Magazine

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

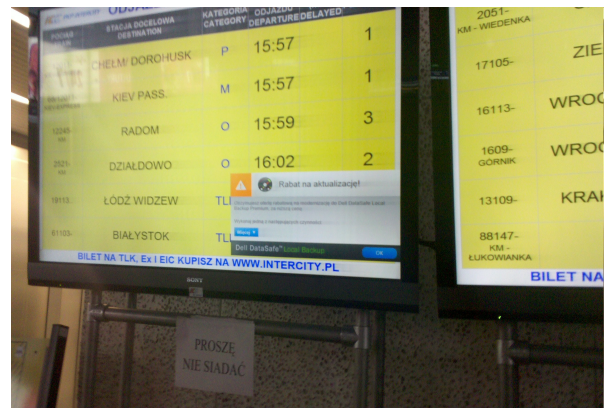
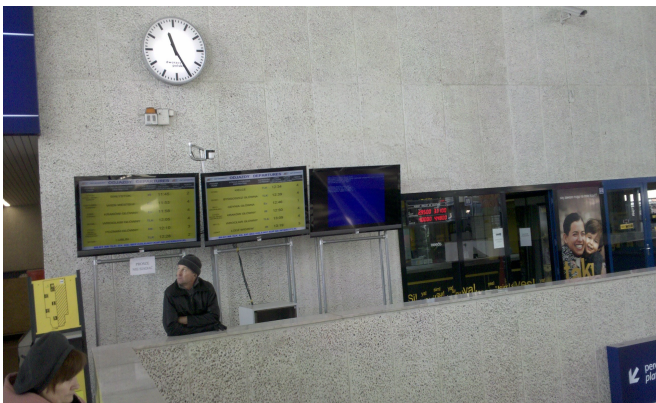
Numer 24/2011

12 grudzień 2011

Zróbmy sobie terminal !

Zapewne wiele osób się skrzywi i stwierdzi – a po co? A jednak mam nadzieję, że ten tekst przynajmniej niektórych zainteresuje, ponieważ mowa w nim będzie na nowoczesnej końcówce sieciowej dysponującej pełnymi możliwościami graficznymi, która może być wykorzystywana w bardzo różnych celach – od zdalnej pracy na serwerach firmowych aż po stację wyświetlania informacji lub korzystania z pełnowartościowej przeglądarki internetowej. Przecież jeśli korzystamy z gmail lub google documents tak naprawdę wykorzystujemy nasz komputer jako terminal.

W tym miejscu nie mogłem sobie odmówić zamieszczenia zdjęć zrobionych komórką na remontowanym Dworcu Centralnym w Warszawie. Nie muszę chyba nikomu wyjaśniać jaki napis znajdował się niebieskim ekranie zamiast rozkładu jazdy. Reklama „Rabatu na aktualizację” zasłaniająca perony oraz godziny odjazdu pociągów na drugim zdjęciu posiada podpis „DataSafe”. Czy naprawdę o takie bezpieczeństwo danych chodziło twórcom tego systemu prezentacyjnego?



Terminal, którego budowę proponuję PT Czytelnikom powinien nas w pełni zabezpieczyć przed takimi niespodziankami. W przeciwieństwie do oprogramowania komputera PC nie możemy liczyć na szybką interwencję użytkownika lub administratora.

Dobry terminal niezależnie od celu, do jakiego będziemy go wykorzystywać musi więc pracować niezawodnie i być w pełni zabezpieczonym przed niespodziankami – na przykład przed nagłą utratą zasilania. Po jego powrocie terminal powinien rozpocząć pracę bez potrzeby jakiegokolwiek interwencji ze strony osób obsługujących system.

Określiśmy więc pierwsze – i w moim przekonaniu chyba najważniejsze założenie naszego projektu i powinniśmy się zastanowić w jaki sposób możemy je zrealizować.

Ponieważ zapewne wykorzystamy typowy sprzęt klasy PC ważnym jest, aby BIOS naszej platformy sprzętowej umożliwił wprowadzenie ustawienia zapewniającego automatyczne uruchomienie komputera po pojawieniu się napięcia zasilającego. Niestety to jednak nie wystarczy, ponieważ typowy dyskowy system operacyjny (zarówno Windows jak i Linux czy MacOS) pracują w oparciu o systemy plikowe, które mogą łatwo ulec uszkodzeniu w przypadku nagłego zaniku zasilania i nie należy się łudzić, że w każdym przypadku ich struktura zostanie automatycznie poprawnie odbudowana. Co więc możemy zrobić?

Proponuję rozwiązanie dość radykalne – przenieść cały system plikowy terminala do pamięci RAM. W przypadku zaniku zasilania nic nie będziemy musieli odbudowywać – po prostu załadujemy obraz tego systemu na nowo.

Takie rozwiązania są oczywiście znane – wykorzystuje się je zarówno w wersjach znanych jako „Live CD” lub też w procedurach odtwarzania uszkodzonych systemów operacyjnych. Nic nie stoi na przeszkodzie, aby zastosować je w naszym terminalu, którego start będzie mógł wyglądać np. następująco:

- Po włączeniu zasilania BIOS uruchamia komputer,
- startuje program ładujący (z dowolnego nośnika lub dostarczony za pomocą sieci komputerowej),
- do pamięci RAM zostają załadowane – jądro systemu oraz kompletny obraz systemu plikowego,
- terminal rozpoczyna pracę.

Jeśli nastąpi zanik zasilania cała procedura po prostu zostanie zrealizowana ponownie.

Proszę zwrócić uwagę, że dostęp do nośnika (np. niewielkiego dysku twardego, pamięci flash lub nawet krążka CD) jest nam niezbędny tylko na czas niezbędny do startu systemu i w dodatku tylko w trybie tylko do odczytu. Nie ma żadnej potrzeby udostępniania nośnika (nawet w trybie ro) w trakcie normalnej pracy terminala. Że tak jest można łatwo sprawdzić korzystając z systemu Linux, w którym często stosuje się osobną partycję dla systemu plikowego boot – np.:

```
# mount
/dev/sda1 on /boot type ext2
/dev/sda2 on / type ext3
/dev/sda3 on /home type ext3
w rzeczywistym systemie wynik komendy mount może zawierać więcej informacji
```

Po uruchomieniu systemu można przecież bez żadnych konsekwencji odłączyć (odmontować) system plikowy /boot, bo jest on niezbędny jedynie do startu systemu.

Dość popularną metodą jest skorzystanie z initrd. Mechanizm taki wykorzystywany jest np. przez dystrybucje Puppy Linux, Tiny Core Linux itp. oraz w wielu urządzeniach Embedded Linux (w tym także w terminalach ABA-X3). Polega on na wykorzystaniu initrd (Initial RAM disk), który najczęściej jest systemem tymczasowym ułatwiającym załadowanie głównego systemu jako ostatecznego obrazu systemu root (/) umieszczanego w RAM dysku.

Realizujący taką funkcję wpis w pliku menu.lst programu ładującego Grub może wyglądać na przykład tak:

```
title ABAX-3
kernel      /kernel vga=0x311 ro root=/dev/ram ramdisk_size=40000
initrd      /root.gz
```

Oczywiście nie musimy się ograniczać do rezerwacji jedynie 40 MB w pamięci RAM dla systemu root. Pojemności pamięci RAM współczesnych komputerów są na tyle duże, że można temu systemowi plików przydzielić znacznie większą (nawet kilkaset MB) przestrzeń. Kompresja obrazu systemu plikowego root nie jest konieczna – w terminalach ABA-X3 stosowano także niewielkie pamięci flash (32 MB) i oszczędność miejsca była jak najbardziej wskazana. Dziś ceny tych pamięci są niewielkie, zaś pojemności duże.

Wiemy więc już, w jaki sposób będziemy ładować obraz głównego systemu plikowego do pamięci RAM. Pozostaje nam przygotowanie tego obrazu. Ogólny opis znajdziemy na stronie <http://www.ibm.com/developerworks/linux/library/l-initrd/index.html> jednak oczywiście możemy przygotować ten obraz w dowolny sposób. Dziś również możemy być o wiele bardziej „rozzutni” gdyż dysponujemy znacznie większymi pojemnościami pamięci. Dla wielu dystrybucji system plikowy o wielkości 500 MB okazuje się wystarczający. Oczywiście warunkiem ograniczenie ilości pakietów, ale przecież terminal nie potrzebuje ich tak wielu. Wspomniany już Puppy Linux zadowala się przecież znacznie skromniejszą ilością miejsca. Wykorzystanie wyników takich projektów jak na przykład BusyBox (www.busybox.net) pozwala na pracę systemu z bardzo niewielkim systemem plikowym root.

Sposób przygotowania obrazu głównego systemu plikowego systemu plikowego jest bardzo prosty. Po sprawdzeniu jego działania (np. wykorzystując maszynę wirtualną kvm) przygotowujemy tymczasowy system plikowy o potrzebnej nam objętości (np. 200 MB) i po jego zamontowaniu kopiujemy tam całą zawartość przetestowanego systemu root. Następnie odmontowujemy system tymczasowy i wykonujemy na nim fsck. Zawartość sprawdzonego systemu plikowego kopiujemy w całości do pliku (np. o nazwie root) komendą dd:

```
dd if=/dev/<nazwa_urzadzenia of=/tmp/root
```

i przeprowadzamy kompresję:

```
gzip /tmp/root
```

Otrzymany plik root.gz kopiujemy do systemu plikowego boot naszego terminala, zaś w pliku menu.lst programu grub dokonujemy odpowiednich wpisów.

Kilka szczegółów w których „diabeł siedzi”:

Oczywiście w tak przygotowanym systemie dość trudno jest dokonywać zmian, ponieważ konieczne byłoby przygotowywanie za każdym razem nowego obrazu systemu. Można sobie jednak z tym poradzić w bardzo prosty sposób tworząc plik zawierający odpowiedni zestaw zmiennych konfiguracyjnych. Może on (na przykład) zawierać dowolną ilość linii w postaci:

```
...  
ZMIENNA=<wartość>  
export ZMIENNA  
...
```

Plik ten powinien być wykonany (jako skrypt) w początkowym stadium startu systemu (np. przez sysinit). W ten sposób odpowiednie zmienne (np. opisujące konfigurację interfejsu sieciowego) będą mogły być wykorzystane przy uruchamianiu systemu. Niezbędne jest oczywiście, aby plik z zestawem zmiennych był dostępny naszemu systemowi – a więc jeśli w naszym terminalu przewidujemy zastosowanie pamięci masowej (np. DiskOnModule, PenDrive, karta SD itp.) lub posiada on nawet niewielki dysk twardy (który zazwyczaj mamy do dyspozycji w starych komputerach) to oprócz systemu plikowego boot (jedna partycja) zawierającego program ładujący grub oraz obraz systemu root.gz (initrd) najprościej jest założyć drugą partycję z systemem plikowym, w którym będziemy przechowywać dane konfiguracyjne i ew. przeznaczone do modyfikacji skrypty startowe (o których później). System plikowy zawierający konfigurację musi być oczywiście dostępny (w trybie tylko do odczytu!) już w początkowym okresie startu systemu.

Plik ze zmiennymi konfiguracyjnymi pozwala oczywiście na przekazanie wartości tylko tych zmiennych, które są wykorzystywane w procesie uruchamiania systemu. Liczba tych zmiennych musi być ustalona już na etapie tworzenia obrazu głównego systemu plikowego. Często jednak mamy do czynienia z koniecznością wprowadzenia modyfikacji oprogramowania terminala, której nie przewidzieliśmy podczas przygotowywania obrazu initrd. Celowe jest wówczas wprowadzenie w głównym systemie możliwości wykonania zewnętrznego skryptu na zakończenie startu systemu. W wielu dystrybucjach wykorzystywany jest w tym celu plik rc.local. Warto w nim umieścić odwołanie do pliku (skryptu) umieszczonego (razem ze skrypcem zawierającym zmienne startowe) w partycji konfiguracyjnej naszego terminala. W ten sposób uzyskamy możliwość wykonywania dowolnych programów oraz wprowadzania zmian w sposobie pracy terminala bez potrzeby tworzenia za każdym razem nowego obrazu głównego systemu plikowego root.

Drugą niewątpliwą zaletą takiego rozwiązania jest możliwość pobierania plików oraz skryptów konfiguracyjnych z serwera sieciowego (np. za pomocą programu wget) – a tym

samym realizacja zdalnego zarządzania naszym terminalem.

W terminalu ABA-X3 takim skrypcem startowym jest plik o nazwie start.sh. Aby uzyskać możliwość edycji tego skryptu należy go umieścić albo w niewielkiej partycji na nośniku dostępnym lokalnie (np. w pamięci flash) albo pobierać go z serwera sieciowego.

Terminal z oprogramowaniem przygotowanym w taki sposób może być wykorzystywany do różnych celów. Zadaniem głównego systemu plikowego jest przygotowanie środowiska umożliwiające korzystanie z protokołów terminalowych – RDP, ICA, XDMCP, NX, VMware View itp. W wielu przypadkach w terminalach jest instalowana lokalna przeglądarka WWW (z obsługą Javy oraz możliwością instalacji dodatków i wtyczek), przeglądarki plików PDF oraz DjVu a nawet programy komunikacyjne SKYPE lub inne komunikatory.

Ponieważ przeglądarki (oraz inne programy) są dość często uaktualniane korzystne jest umieszczenie ich poza głównym systemem plikowym. Uzasadnia to także ich wielkość oraz liczba niezbędnych bibliotek. Możemy w tym celu wykorzystać odrębny system plikowy w trzeciej partycji na lokalnym nośniku (jak w terminalu ABA-X3) lub skorzystać z urządzeń loop (jak w rozwiązaniu 2X OS). W przypadku dużej liczby niezbędnych programów lokalnych rozwiązanie typu NetBOOT lub wykorzystujące zdalne systemy plikowe (NFS lub CIFS) nie wydaje się zbyt korzystne za względu na obciążanie sieci.

Ostatnim problemem, który powinniśmy uwzględnić jest katalog domowy użytkownika. Terminal nie powinien być urządzeniem personalizowanym, lecz wiele programów zapisuje dane konfiguracyjne. Na przykład klient NX zapisuje w katalogu pliki sesji z parametrami połączeń. Podobnie postępują programy klienckie innych protokołów. Pliki te powinny być zapisane w sposób trwały przez administratora systemu – odpowiednim więc miejscem dla tych plików jest partycja konfiguracyjna.

Problem są jednak obszary dla pamięci notatnikowych (cache) oraz pliki z historią (przeglądarki WWW) i danymi autoryzacyjnymi. Powinny być one kasowane po zakończeniu sesji i niedostępne dla kolejnych użytkowników. Wszelkie dane niezbędne podczas trwania sesji korzystnie jest zapisywać w pamięci RAM.

Jeśli na lokalnym nośniku mamy wystarczającą ilość miejsca możemy je wykorzystać na obszar swap.

Podsumowując:

- system plikowy /boot zawierający program startowy oraz obraz systemu / ładowany do pamięci RAM (jako initrd) nie powinien być w ogóle montowany przy normalnym uruchamianiu terminala. W razie potrzeby administrator może go przecież podmontować korzystając z linii komend.
- System plikowy zawarty w partycji konfiguracyjnej powinien być montowany jak najwcześniej (sysinit), lecz w trybie „tylko do odczytu” (ro).
- Dodatkowy system plikowy (lub systemy plikowe) z programami lokalnymi powinny być także montowane w trybie ro, lecz już po starcie systemu (local).
- Jeśli terminal realizuje wiele sesji oraz korzysta z wielu lokalnych programów korzystne będzie zapewnienie odpowiedniego obszaru swap.

Tak przygotowany terminal interakcyjny lub wyświetlacz informacji albo kiosk internetowy zapewni pracę bez niespodzianek, które zasygnalizowałem na początku...