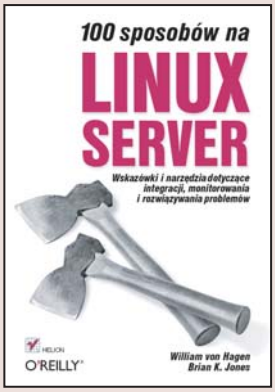
# 100 sposobów na Linux Server. Wskazówki i narzędzia dotyczące integracji, monitorowania i rozwiązywania problemów




Autorzy: William von Hagen, Brian K. Jones
Tłumaczenie: Małgorzata Czart, Piotr Maciejek, Leszek Sagalara
ISBN: 83-246-0490-1
Tytuł oryginału: Linux Server Hacks, Volume Two: Tips & Tools for Connecting, Monitoring, and Troubleshooting
Format: B5, stron: 480

Linux zdobywa coraz większą popularność. Wszędzie, gdzie mamy do czynienia z zarządzaniem sieciami komputerowymi, bezpieczeństwem danych czy potrzebą nieprzerwanej, stabilnej pracy systemu, bardzo często będziemy mieli do czynienia z serwerem linuksowym. System Linux nieustannie rzuca wyzwanie największym producentom oprogramowania i stanowi bardzo poważną alternatywę wobec konkurencyjnych rozwiązań.

Autorzy książki „100 sposobów na Linux Server. Wskazówki i narzędzia dotyczące integracji, monitorowania i rozwiązywania problemów" przedstawiają ogrom praktycznej wiedzy z zakresu administracji serwerem Linux Obaj pracowali jako administratorzy systemów, zatem doskonale wiedzą, z jakimi problemami spotykają się na co dzień użytkownicy Linuksa. Dzięki poradom zawartym w książce nawet zaawansowani użytkownicy odkryją wiele nowych sposobów usprawniających ich pracę.
W przejrzystej formie przedstawiono instrukcje opisujące krok po kroku rozwiązania, dzięki którym Twój system będzie wydajny, bezpieczny i użyteczny. Wszystko krótko i na temat!

W książce znajdziesz między innymi:

- Kontrolowanie procesu uwierzytelniania w Linuksie
- Administrowanie pamięcią masową w sieci z użyciem kwot dyskowych, klonowania, snapshotów i systemów RAID
- Instalacja i konfiguracja serwera Kerberos
- Podłączanie graficznego interfejsu użytkownika do zdalnych systemów
- Odzyskiwanie danych z uszkodzonych systemów plików

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

# Spis treści

ROZDZIAŁ CZWARTY

# Narzędzia i wskazówki dla administratora systemu

## Sposoby 29. – 45.

Pod spokojną, opanowaną powierzchownością wytrawnego administratora systemu kryje się szalony naukowiec, który żyje i oddycha tylko po to, by być pierwszym, który odkryje kolejny tajemniczy sekret dotyczący informacji lub metod działania nieznanych do tej pory nikomu poza wąską grupą uzależnionych od kofeiny hakerów.

Przyczyną tej niepohamowanej żądzy nie jest chęć zdobycia uznania, lecz coś znacznie bardziej praktycznego: efektywność. Jeśli można wykonać coś lepiej, szybciej lub bez potrzeby interwencji człowieka, administrator systemu będzie stale szukał okazji do wprowadzenia takiego rozwiązania.

W niniejszym rozdziale przyjrzymy się niektórym narzędziom i technikom, jakie być może są nieznane Czytelnikom, a które mogą znacznie zwiększyć wydajność. Bez względu na to, czy będzie to skrót na pulpicie uruchamiający połączenie z komputerem centralnym, sposób wydawania poleceń na wielu komputerach jednocześnie czy metoda wpisywana krótszych poleceń lub mniejszej liczby znaków w edytorze Vim — przedstawimy tu narzędzia i techniki, które z niewolnika systemu uczynią prawdziwego Władcę Bitów.

Sprawność techniczna to wspaniała sprawa, ale na dzisiejszym konkurencyjnym rynku pracy coraz bardziej liczą się tzw. „miękkie umiejętności", takie jak komunikacja z ludźmi czy wielozadaniowość. Dlatego też przyjrzymy się również tej stronie administrowania systemem, omawiając kwestie związane z zarządzaniem czasem, a także metody *rozmawiania* z zarządem.

SPOSÓB 29.

### Jednoczesne wykonywanie poleceń na wielu serwerach

Uruchomienie tego samego polecenia w tym samym czasie w wielu systemach upraszcza zadania administracyjne i zmniejsza problemy związane z synchronizacją.

Jeśli mamy wiele serwerów o podobnej lub identycznej konfiguracji (np. węzły klastra), często trudno jest sprawić, aby ich zawartość i konfiguracja pozostawały identyczne. Jeszcze trudniej jest wtedy, gdy trzeba zmodyfikować konfigurację z poziomu wiersza poleceń, wiedząc, że to samo polecenie trzeba będzie wykonać w większej liczbie systemów (lepiej najpierw napić się kawy). Można pokusić się o napisanie skryptu automatycznie

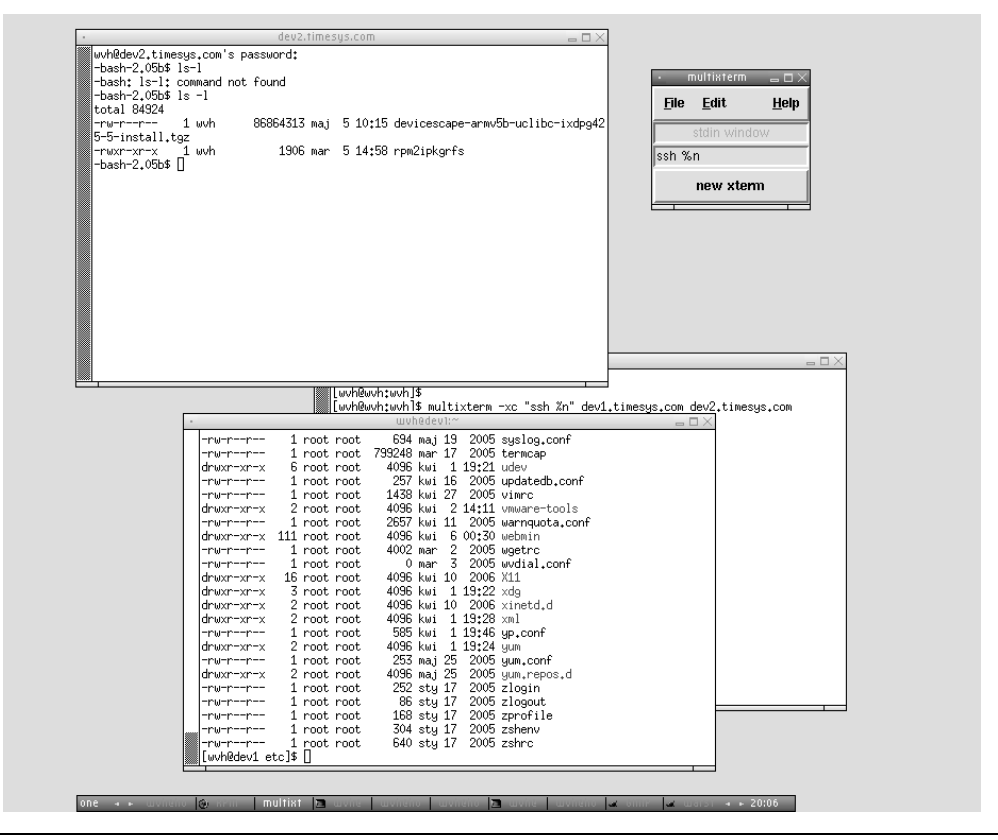
wykonującego to zadanie, ale czasem napisanie skryptu jest trudniejsze niż sama praca do wykonania. Na szczęście istnieje sposób umożliwiający jednoczesne wykonanie poleceń na wielu komputerach.

Świetnym rozwiązaniem tego problemu jest wspaniałe narzędzie o nazwie *multixterm*, które umożliwia równoczesne otwarcie okien terminali *xterm* dowolnej liczby systemów, wpisanie polecenia w jednym głównym oknie i wykonanie go w każdym z otwartych okien programu *xterm*. Brzmi interesująco? Wpisujemy raz, wykonujemy wielokrotnie — brzmi jak nowy zestaw instrukcji potokowej.

*multixterm* można pobrać ze strony *http://expect.nist.gov/example/multixterm.man.html*. Wymaga on do działania *expect* i *tk*. Najbardziej popularnym sposobem jego uruchomienia jest następujące polecenie:

```
# multixterm -xc "ssh %n" host1 host2
```

Polecenie to otwiera połączenie `ssh` do `host1` i `host2` (rysunek 4.1). Wszystko, co wpiszemy w pole *stdin window* (które może mieć różne kolory w zależności od wybranego schematu kolorów), zostanie wysłane do obu widocznych na rysunku okien.




*Rysunek 4.1. Okna xterm i okno kontrolne multixterm*

Jak widać na przykładzie polecenia, opcja `-xc` oznacza wykonanie polecenia, a po niej podajemy polecenie, które ma zostać wykonane na każdym komputerze, objęte znakami cudzysłowu. Jeśli określone polecenie zawiera symbol wieloznaczny `%n`, wówczas każdy z wymienionych dalej komputerów zostanie tu kolejno podstawiony przy wykonywaniu polecenia. Tak więc w naszym przykładzie *multixterm* wykona dwa polecenia (`ssh host1` i `ssh host2`) — każde z nich w osobnym oknie *xterm*.

## Zobacz również

- `man multixterm`
- „Szybkie połączenia przez telnet (SSH) z pulpitu" **[sposób 41]**
- „Odłączanie konsoli bez kończenia sesji" **[sposób 34]**

*— Lance Tost*

SPOSÓB 30.

# Bezpieczna współpraca z bezpiecznym Wiki

Jak pozbyć się wszystkich spraw dotyczących kodowania, obsługi, wykrywania błędów i zarządzania witrynami, które związane są ze współpracą nad projektami, za pomocą narzędzia umożliwiającego użytkownikom utworzenie własnych witryn.

Ostatnią rzeczą, jakiej potrzebuje zajęty administrator WWW próbujący zarządzać systemem, jest kolejny użytkownik żądający zbudowania i utrzymywania następnej witryny lub instalacji jeszcze jednego systemu zarządzania treścią. Zamiast tego można w ciągu kilku sekund uruchomić system oparty na Wiki, który będzie gotowy do tworzenia i edycji treści przez użytkowników bez naszej dalszej pomocy.

Rozwiązania typu Wiki rozwinęły się wokół idei, że zawartość może być dostępna do edycji przez każdego, komu zdarzy się wejść na stronę i zauważy tam pomyłkę lub ma coś do dodania. Administratorzy WWW oraz administratorzy systemów byli nieufni wobec takiego pomysłu, który wydawał się tylko czekać na nadużycia spamerów, wandali komputerowych i im podobnych. Jeśli ktoś po raz ostatni zetknął się z rozwiązaniami Wiki, gdy było to jeszcze modne słówko na głównych stronach internetowych witryn informacyjnych, i uznał je za niemożliwe do zarządzania lub czekające tylko na zagrożenia dla bezpieczeństwa (jak ja), namawiam do przyjrzenia się temu ponownie.

MediaWiki jest głównym oprogramowaniem Wiki, w oparciu o które działa witryna Wikipedii (*http://wikipedia.org/*). Wikipedia jest najsłynniejszą witryną Wiki na świecie i szczyci się tym, że jej zasoby są tworzone przez każdego[1] i dla każdego. Czyż może być lepsza rekomendacja dla aplikacji Wiki?

Witryna Wiki musi być niezabezpieczona, dostępna dla wszystkich. Obecnie można skonfigurować MediaWiki do uwierzytelniania przez wewnętrzny serwer LDAP, całkowicie wyłączając możliwość anonimowej edycji, co ogranicza ewentualne zniszczenia,

[1] Obecnie te możliwości zostały nieco ograniczone (2006 r.), przynajmniej w stosunku do niektórych fragmentów Wikipedii — *przyp. red.*

jakie można spowodować w witrynie. Oprócz tego MediaWiki bardzo ułatwia śledzenie zmian na stronach witryny i daje możliwość przywrócenia starszych kopii stron.

Dlaczego więc używać Wiki, skoro zamierzamy zamknąć do niej dostęp? Otóż Wiki to przykład wspaniałego rozwiązania zarządzania treścią, z jednego powodu: nie ma tu absolutnie żadnych przyjętych założeń, jeśli chodzi o przeznaczenie naszej witryny. Wiele opartych na LAMP[2] wolnodostępnych rozwiązań zarządzania treścią powstało *głównie* z myślą o witrynach informacyjnych — rozszerzenia umożliwiające inne zastosowania, takie jak blogi, fora, witryny pogodowe czy repozytoria plików, są dodawane później, często przez członków danych społeczności użytkowników. Jeżeli nie planujemy utworzenia witryny informacyjnej, musimy znaleźć sposób na to, aby witryna naszego systemu zarządzania treścią działała tak, jak chcemy. Jeśli korzystamy w tym celu z rozszerzeń, to nie można po prostu dokonać aktualizacji i założyć, że wszystko działa poprawnie.

Używałem wielu wolnodostępnych systemów zarządzania treścią i mogę powiedzieć, że w zależności od potrzeb prawdopodobnie znajdziemy dla siebie coś odpowiedniego, ale jeżeli obsługujemy użytkowników uniwersyteckich, dział badawczo-rozwojowy czy wydziały wewnętrzne, wówczas każda grupa projektowa może mieć inne potrzeby. Prosta konstrukcja przekazująca możliwości kształtowania struktury i formatowania zawartości przez twórców treści i użytkowników jest potężnym narzędziem, a dzięki możliwości nakładania różnego rodzaju ograniczeń w zakresie dostępu i edycji możemy zapewnić spokój sobie i użytkownikom. Jeżeli użytkownicy będą się domagać całkowicie otwartej witryny, a my się na to zgodzimy, nie będzie z tym żadnego problemu. Lecz jeśli mamy inne wymagania w zakresie bezpieczeństwa, to z MediaWiki możemy je z łatwością wdrożyć.

MediaWiki umożliwia uwierzytelnianie przez serwer LDAP lub połączenie z bazą danych, a dostępne są również poprawki umożliwiające zastosowanie innych metod uwierzytelniania występujących w naszym środowisku. Oprócz tego możemy zdecydować się na wprowadzenie ograniczeń dostępu, aby jedynie zarejestrowani i zalogowani użytkownicy mogli edytować strony, utworzyć więcej otwartych witryn, gdzie każdy będzie mógł edytować strony, lub zupełnie wyłączyć rejestrację dla witryny przeznaczonej na dokumentację personelu wewnętrznego.

## Instalacja MediaWiki

Inną korzystną cechą MediaWiki jest łatwość instalacji. Nie wymaga ona PHP, a jej kreator usilnie zaleca stosowanie bazy danych MySQL. W zależności od opcji, z których chcemy korzystać (np. miniaturki obrazków czy uwierzytelnianie LDAP), może zajść konieczność skompilowania PHP z określonymi bibliotekami, jednak wymagania niezbędne do uruchomienia prostej witryny są całkiem niewielkie.

[2] LAMP — skrót oznaczający zestaw oprogramowania open source wykorzystywany w serwerach WWW, a pochodzący od pierwszych liter słów: Linux, Apache, MySQL, Perl (PHP, Pyton) — *przyp. tłum.*

Jeżeli utrzymujemy już własną witrynę (np. mamy uprawnienia użytkownika root), wówczas instalacja zajmuje dosłownie sekundy. Wszystko, co trzeba zrobić, to rozpakować archiwum tar zawierające dystrybucję do głównego katalogu dokumentów naszego serwera WWW i przejść do witryny! MediaWiki rozpozna, że jest to nasza pierwsza wizyta i poprosi o przeprowadzenie instalacji. Po podaniu hasła administratora MySQL MediaWiki utworzy nowego użytkownika, nową bazę danych i wszystkie niezbędne tabele, co stanowi 90% procesu instalacji.

Jeżeli jednak uruchamiamy witrynę MediaWiki, która utrzymywana będzie na zdalnym serwerze, wówczas nie trzeba będzie podawać hasła administratora ani hasła administracyjnego MySQL. W tym przypadku tworzymy najpierw bazę danych MediaWiki, a następnie wskazujemy przy instalacji tę bazę danych w celu utworzenia jej tabel. Niestety, nie mogę powiedzieć, jak to wykonać, gdyż każdy serwer prowadzący usługi WWW zapewnia do tego inne narzędzia wspomagające.

Po pomyślnie przeprowadzonej instalacji zostanie nam przedstawiony odnośnik prowadzący do naszej nowej witryny.

## Konfiguracja MediaWiki

Instalacja była prosta, ale jak teraz zablokować witrynę? Istnieją na ten temat tony dokumentacji, ale podsumuję tu niektóre z najważniejszych opcji dla administratorów.

Przede wszystkim chodzi o dostęp do witryny. Często nie używa się Wiki z powodu błędnego przekonania, że takich witryn nie da się zabezpieczyć. Nieprawda!

Już w wersji 1.4 MediaWiki można było za pomocą pliku konfiguracyjnego i (lub) instrukcji SQL zmieniać funkcje dostępne dla różnych typów użytkowników. Z kolei wersja 1.5 zawiera całkiem pokaźną kolekcję potencjalnych ról, jakie mogą zostać przydzielone użytkownikom za pomocą grup. Omawiamy tutaj wersję 1.5, gdyż prawdopodobnie będzie ona już wydana, gdy ta książka ukaże się w druku.

Pracuję na wersji 1.5rc4, którą zarządza się głównie za pomocą przeglądarki internetowej. Są tu oddzielne strony służące do dodawania, usuwania i blokowania użytkowników. Istnieje też strona do zmiany grup, do których należą użytkownicy, co ma wpływ na ich uprawnienia podczas odwiedzania witryny. Oprócz tego dostępne są również wtyczki pomagające utworzyć zestawienie zawierające użytkowników i wszystkie znane adresy IP, jakich używają, oraz realizujące inne funkcje niedostępne w głównej dystrybucji.

Nie ma jeszcze jednak interfejsu do zmiany uprawnień dla grup oraz usuwania i dodawania grup. W tym celu należy mieć dostęp do powłoki poleceń serwera WWW lub utworzyć lokalną kopię pliku *LocalSettings.php*, zmodyfikować go i skopiować z powrotem na swoje miejsce, aby zmiany odniosły skutek. Plik ten jest prosty w edycji, a dokumentacja omawiająca dokonywane zmiany jest więcej niż wystarczająca, ale pokażę na przykładach jedną czy dwie szybkie zmiany, jakie być może zechcemy wprowadzić.

Jeśli chcemy tylko zmienić grupę, do której należy dany użytkownik, logujemy się jako użytkownik o uprawnieniach administracyjnych i przechodzimy do odnośnika *Strony specjalne*. W górnej części ekranu zobaczymy zastrzeżone strony specjalne, które będą widoczne jedynie wtedy, gdy zalogowany jest administrator. Ta sekcja zawiera odnośnik do strony zarządzania prawami użytkownika, która aktualnie jest tylko interfejsem do zmiany członkostwa grupy dla określonego użytkownika.

Jeśli chcemy utworzyć grupę, musimy poddać edycji plik *LocalSettings.php* i ustawić uprawnienia dostępne dla grupy. Żeby zobaczyć ustawienia grup domyślnych, musimy sprawdzić dokumentację lub otworzyć plik *includes/DefaultSettings.php* w katalogu instalacyjnym. Do pliku *LocalSettings.php* dodajemy poniższe wiersze w celu utworzenia grupy o nazwie *nowagrupa* z uprawnieniami do odczytu i edycji, ale nie do usuwania:

```
$wgGroupPermissions['nowagrupa']['edit'] = true;
$wgGroupPermissions['nowagrupa']['read'] = true;
$wgGroupPermissions['nowagrupa']['delete']= false;
```

Jak widać, nie ma tu jawnej funkcji „utwórz grupę". Przypisanie praw nieistniejącej grupie, co właśnie zrobiłem, spowoduje jej utworzenie. Gdy przejdziemy teraz do strony zarządzania prawami użytkownika, grupa ta będzie wymieniona na liście dostępnych grup.

Pamiętajmy, że istnieją też ustawienia globalne, dotyczące *wszystkich* grup (reprezentowanych w konfiguracji przez `*`). Oto kilka domyślnych ustawień dla grup z pliku *DefaultSettings.php*:

```
$wgGroupPermissions['*'    ]['createaccount']   = true;
$wgGroupPermissions['*'    ]['read']            = true;
$wgGroupPermissions['*'    ]['edit']            = true;
```

Jeśli chcemy nadpisać te ustawienia, wystarczy umieścić podobne wiersze w pliku *LocalSettings.php*, ustawiając według własnego uznania odpowiednie wartości na `true` lub `false`. Plik *LocalSettings.php* unieważnia wszystkie odpowiadające ustawienia znajdujące się w pliku *DefaultSettings.php*.

Taki model zapewnia nam elastyczność, możemy na przykład całkowicie wyłączyć możliwość tworzenia kont przez anonimowych użytkowników lub pozwolić im jedynie na odczyt bądź wprowadzić wymóg logowania w celu dokonania jakichkolwiek edycji. Istnieją też dodatkowe uprawnienia, jakie możemy nadać użytkownikom, czyniąc z nich quasi-administratorów, co pozwoli im tworzyć konta dla innych użytkowników, usuwać pliki czy przywracać poprzednie edycje.

## Struktura danych

Skoro mamy już rozwiązaną sprawę dostępu użytkowników, prawdopodobnie najważniejszą decyzją dotyczącą uruchomienia Wiki będzie organizacja zawartości w witrynie oraz wynalezienie najlepszego sposobu na odwzorowanie w zawartości witryny elementów organizacyjnych dostępnych w MediaWiki. Istnieje wiele przydatnych narzędzi, z jakich możemy skorzystać, a wszystkie one mają charakter dość ogólny — bez założeń dotyczących przeznaczenia witryny.

Jest wiele sposobów użycia różnych elementów organizacyjnych. Jeśli dysponujemy tylko jedną grupą projektową, może mieć ona własną Wiki poświęconą jej projektowi. Jednak potencjalnie możemy utworzyć kilka projektów współdzielących pojedynczą witrynę Wiki za pomocą oddzielnych przestrzeni nazw w witrynie. **Przestrzeń nazw** (ang. *namespace*) to element danych najwyższego poziomu, jaki jest dostępny w MediaWiki. W obrębie przestrzeni nazw występują kategorie, dzięki którym osoby zarządzające projektem mogą rozdzielić swoje witryny na rożne części odpowiadające ich potrzebom.

Aby ktoś nie myślał, że strony witryny muszą być całkowicie statycznymi dokumentami, przyjrzyjmy się opcji szablonów MediaWiki, która pozwala osadzać dokumenty w obrębie różnych stron. Daje to dużą elastyczność, np. gdy strona główna ma być jedynie zbiorem innych dokumentów umieszczonym na stronie głównej. Osoby zarządzające różnymi stronami szablonów będą mogły następnie aktualizować ich zawartość, a zmiany zostaną odzwierciedlone na stronie głównej bez wpływu na szablony utworzone przez innych użytkowników.

SPOSÓB 31.

## Edycja pliku konfiguracyjnego GRUB za pomocą grubby

Jak, korzystając z gotowego narzędzia do edycji pliku grub.conf, oszczędzić sobie sporo wpisywania (i pomyłek).

System, który się nie uruchamia, będzie niezdolny do pracy, a jest wiele środowisk, w których plik *grub.conf* dostarczony z dystrybucją wymaga dostosowania. Gdy przeprowadzamy szybką instalację zespołu serwerów lub testujemy nowe wersje jądra w serwerze WWW, możemy odłożyć na bok swoje umiejętności pisania skryptów i skorzystać z *grubby*, prostego narzędzia uruchamianego z wiersza poleceń, które pomoże nam zmodyfikować definicje jądra.

Oto przykład bardzo prostej definicji jądra z pliku *grub.conf* w serwerze Red Hat Enterprise Linux:

```
title Red Hat Enterprise Linux AS (2.4.21-32.0.1.EL)
        root (hd0.0)
        kernel /vmlinuz-2.4.21-32.0.1.EL ro root=LABEL=/initrd /
        initrd-2.4.21-32.0.1.EL.img
```

Jest to dość standardowy wpis określany w dokumentacji programu GRUB mianem definicji systemu operacyjnego (ponieważ GRUB potrafi uruchomić niemal każdy istniejący system operacyjny). Czasami zachodzi konieczność dokonania zmian w pliku *grub.conf* w celu przesłania do jądra argumentów startowych. Na przykład uruchomiliśmy zespół serwerów i dodajemy do nich później konsolę szeregową, a GRUB nie dodaje automatycznie argumentów niezbędnych do przekierowania wyjścia na urządzenie terminala szeregowego. Omówione to zostało w [sposobie 76], „Scentralizowany dostęp do systemu za pomocą konsoli szeregowej”.

Zwykle oznacza to konieczność ręcznej edycji pliku *grub.conf* w celu wstawienia argumentów — chyba że zdarzyło się nam zetknąć z *grubby*. Oto polecenie, które (wydane jako root) dodaje dla wszystkich jąder argumenty wymagane do przekierowania konsoli:

```
# grubby --update-kernel=ALL --args=console=ttyS0,9600
```

Słowo kluczowe `ALL` działa z kilkoma opcjami. W tym przypadku dodaje argumenty do każdego jądra w pliku konfiguracyjnym. Innym słowem kluczowym jest `DEFAULT`, które zmienia wyłącznie wiersz dotyczący domyślnego jądra, oznaczonego w pliku *grub.conf* parametrem `default`.

*grubby* może również zmieniać opcje samego bootloadera *grub*. Za pomocą poniższych poleceń możemy dodać nowe jądro do pliku *grub.conf* i uczynić je domyślnym:

```
# grubby --add-kernel=/boot/vmlinuz-2.4.21-new --make-default
# grubby --set-default=/boot/vmlinuz-2.4.21-32.0.1.ELsmp
```

Użyłem tu opcji `--make-default`, aby uczynić `vmlinuz-2.4.21-new` jądrem domyślnym. Jeżeli za pomocą *grubby* nakażemy zmianę domyślnego jądra na takie, które nie występuje w pliku konfiguracyjnym, wówczas parametr `default` zostanie całkowicie usunięty z pliku bez żadnego ostrzeżenia. Dlatego, jeżeli polecenie dodania nowego jądra zakończy się niepowodzeniem, należy wydać kolejne polecenie przywracające poprzednio zdefiniowane jądro domyślne za pomocą parametru `--set-default`.

Zatem w jaki sposób ma to nam oszczędzić wpisywania? Przecież zmiana domyślnego jądra jest równie prosta jak zmiana pojedynczej cyfry w pliku *grub.conf*, prawda? No cóż, zgadza się, zakładając że wykonujemy to na pojedynczym komputerze. Jednak jeśli za pomocą skryptu z trzeba zaktualizować plik *grub.conf* na wszystkich zarządzanych komputerach podczas instalacji, gdzie ustawiamy własne jądro jako domyślne, wówczas do wykonania takiej pracy o wiele bardziej wolę użyć programu *grubby*, niż korzystać z *sed*, *awk*, *vi*, *ed* i (lub) Perla. W takich przypadkach zaoszczędzi to nam wpisywania, nie wspominając o tym, że uchroni to nas przed ponownym odkrywaniem Ameryki!

## SPOSÓB 32. Lepsze wykorzystanie klawisza tabulacji

Programowalne uzupełnianie w powłoce *bash* może uzupełniać znacznie więcej niż tylko nazwy plików.

Autouzupełnianie za pomocą klawisza tabulacji w powłoce *bash* nie jest niczym nowym i nie wyobrażam już sobie życia bez tego. Wpisuję na przykład `ls fo<tab><tab>` i otrzymuję listę pięciu czy sześciu plików rozpoczynających się na „fo" — to może być bardzo przydatne. Mamy skrypt o długiej nazwie, przy wpisywaniu której łatwo o pomyłkę? Wystarczy wpisać klika początkowych liter i nacisnąć dwukrotnie klawisz tabulacji, a *bash* spróbuje za nas dokończyć nazwę. Pozwala to wspaniale oszczędzić czas. Ja w każdym razie bardzo tęsknię za tym mechanizmem, gdy zaloguję się na inny komputer z Uniksem, gdzie powłoką jest *csh*, w której domyślnie nie ma uzupełniania za pomocą klawisza tabulacji (zamiast nazw plików pojawiają się wtedy znaki kontrolne).

W wersji 2.04 do powłoki *bash* wprowadzone zostało uzupełnianie „programowalne". Umożliwia to wprowadzenie osobliwych i wspaniałych możliwości do procedury startowej powłoki *bash* (która zapisana jest zwykle w plikach *~/.bashrc* i *~/.bash_profile*).

Procedura startowa powłoki *bash* zależna jest od ustawień środowiska powłoki — *bash* swoje informacje startowe może pobierać z globalnego pliku */etc/bashrc*, plików *~/.bash_profile*, *~/.bashrc*, *~/.profile* i pliku *~/.login*.

Oto krótki przykład:

```
complete -f -X '!*.@(sxw|stw|sxg|sgl|doc|dot|rtf|txt|htm|html|odt|ott|odm)'
oowriter
```

Wygląda enigmatycznie, ale to naprawdę zupełnie proste. `complete` to wbudowane słowo kluczowe powłoki *bash*, które powoduje, że powłoka próbuje uzupełnić tekst przed kursorem w wierszu poleceń. Opcja `-f` oznacza, że chodzi o uzupełnianie nazw plików. Opcja `-X` określa wzorzec, jaki zostanie użyty do przeprowadzenia dopasowania. Ponieważ powłoka analizuje cały wiersz, należy zawsze pamiętać o objęciu wzorca znakami apostrofu, aby się upewnić, że nie nastąpi tu żadne rozwinięcie powłoki, powodując dziwne zachowania po naciśnięciu klawisza tabulacji.

Sam wzorzec przyjmuje następującą postać:

```
!*.@(rozszerzenie)
```

Początkowy wykrzyknik wskazuje w tym kontekście, że przy przeprowadzaniu uzupełniania nazw plików elementy niepasujące do podanego wzorca zostaną pominięte. Ciąg `*.@(`*`rozszerzenie`*`)` oznacza „wszystkie elementy, po których następuje kropka, a następnie dokładnie jedno spośród wszystkich podanych tu rozszerzeń". Znaki `|` na liście rozszerzeń są odpowiednikiem operatora logicznego „lub". Jeśli jakieś elementy będą pasować do wzorca, zostaną one wymienione na liście plików wygenerowanej po dwukrotnym naciśnięciu klawisza tabulacji.

Ostatnie słowo w tym wierszu (w tym przypadku `oowriter`) określa polecenie, którego dotyczy cały ten wiersz. Innymi słowy, instrukcje w wierszu `complete` będą wykonywane tylko wtedy, gdy wykonywanym poleceniem będzie `oowriter`.

Jeśli chcemy, możemy napisać tysiące takich wierszy, ale obmyślenie wszystkich rzeczy, jakie powinny być uzupełniane, prawidłowe wpisanie wszystkich wyrażeń regularnych, a następnie przetestowanie tego, aby się upewnić, czy zwracane są właściwe nazwy plików, zabrałoby nam wieczność. Zamiast tego możemy pobrać gotowy plik autorstwa Iana MacDonalda, twórcy pakietu `bash-completion`, dostępnego pod adresem *http://www.caliban.org/bash/index.shtml#completion*. Pakiet ten składa się głównie z prostej dokumentacji oraz pliku zawierającego olbrzymi zbiór „sztuczek" związanych z uzupełnianiem w powłoce *bash*. Wersja, którą ostatnio pobrałem, zawierała ponad 200 skrótów!

Wiele spośród tych skrótów to bardzo proste wzorce uzupełniania plików powiązane z określonymi aplikacjami, co jest bardziej przydatne, niż mogłoby się wydawać. Wpisanie `tar xvzf f<tab><tab>` i otrzymanie tych plików z rozszerzeniem *tar.gz* to wspaniała sprawa, ale skróty umożliwiające po wpisaniu polecenia `ssh` uzupełnianie nazw hostów (z naszego pliku *known_hosts*) czy obiektów docelowych w pliku Makefile po wpisaniu polecenia `make` potrafią naprawdę oszczędzić sporo czasu administratorom, którzy często muszą zdalnie administrować lub budować oprogramowanie ze źródeł.

Wspaniałe jest w tym rozwiązaniu to, że jedyną zależność stanowi tu sama powłoka *bash* — cała reszta zależy od nas! Jeśli mamy uprawnienia dostępu na poziomie administratora, możemy w katalogu */etc/profile.d* utworzyć plik o nazwie *bash_complete.sh* i wkleić do niego kod ustawiający uzupełnianie powłoki *bash* w sensowny sposób. Kod ten pochodzi wprost z pliku dystrybucyjnego *README* powłoki *bash*:

```
bash=${BASH_VERSION%.*}; bmajor=${bash%.*}; bminor=${bash#*.}
if [ "$PS1" ] && [ $bmajor -eq 2 ] && [ $bminor '>' 04 ] && [ -f
/etc/bash_completion ]; then # interactive shell
    # Source completion code
    . /etc/bash_completion
fi
unset bash bmajor bminor
```

Powyższy kod wykonuje proste sprawdzenie, aby się upewnić, że wersja powłoki *bash* obsługuje programowalne uzupełnianie, a następnie kontroluje, czy uruchomiona jest interaktywna powłoka, po czym wczytuje plik uzupełniania programowalnego powłoki *bash*.

Umieszczenie tego kodu w katalogu */etc/profile.d* lub pliku globalnym */etc/bashrc* pozwoli wszystkim użytkownikom komputera czerpać korzyści z programowalnego uzupełniania powłoki *bash*.

Z drugiej strony, jeśli chcemy sami z tego korzystać lub przesłać to na swoje konto „powłokowe" na serwerze WWW, możemy wkleić ten sam powyższy kod do swojego pliku *~/.bashrc*.

## Zobacz również

- *http://www.caliban.org/bash/index.shtml#completion*

SPOSÓB 33.

# Utrzymywanie działających procesów po wyjściu z powłoki

Polecenia kontrolujące procesy, takie jak `nohup` i `disown`, ułatwiają uruchamianie długotrwałych procesów i utrzymywanie ich działania nawet po wylogowaniu.

Przypuśćmy, że uruchomiliśmy na swoim serwerze narzędzie diagnostyczne lub monitorujące, albo kompilujemy bardzo duży program, a proces ten musi trwać godziny, dni lub dłużej. A co, jeśli taki proces będzie musiał działać nawet po naszym wylogowaniu lub jeśli nasza sesja powłoki zakończy się niezależnie od nas? Można sobie z tym poradzić za pomocą poleceń `nohup` i `disown`.

Po uruchomieniu sesji powłoki wszystkie procesy, jakie uruchamiamy z wiersza poleceń, są procesami potomnymi tej powłoki. Jeśli się wylogujemy lub nasza sesja zakończy się nieoczekiwanie na skutek awarii lub z innego powodu, wówczas sygnał zabicia procesu `SIGHUP` (sygnał zerwania łączności) zostanie wysłany także do wszystkich procesów potomnych w celu ich zakończenia.

Można to obejść przez nakazanie procesowi (procesom), którego działanie chcemy utrzymać, ignorowania sygnałów SIGHUP. Są na to dwa sposoby: użycie polecenia `nohup` (ang. *no hangup*) w celu uruchomienia polecenia w środowisku, gdzie będzie ono ignorować pewne sygnały zakończenia, lub przez polecenie powłoki *bash* `disown`, aby uczynić określone zadanie działające w tle niezależnym od aktualnej powłoki.

## Wykonywanie poleceń przy użyciu nohup

Polecenie `nohup` zapewnia szybki i łatwy mechanizm umożliwiający utrzymanie działania procesu bez względu na to, czy jego proces macierzysty jest wciąż aktywny. Aby skorzystać z tej możliwości, należy uruchomić polecenie, poprzedzając je poleceniem `nohup`:

```
$ nohup polecenie
```

Spowoduje to wykonanie określonego polecenia i utrzymanie jego działania nawet po zakończeniu jego sesji macierzystej. Jeżeli nie przekierujemy wyjścia tego procesu, wówczas będzie ono — wraz z komunikatami o błędach (`stout` i `sterr`) — przesyłane do pliku o nazwie *nohup.out* w aktualnym katalogu. Jeśli taki plik nie będzie mógł zostać tam utworzony, wówczas zostanie utworzony w katalogu macierzystym użytkownika, który wydał to polecenie.

Możemy monitorować informacje zapisywane do pliku *nohup.out*, stosując polecenie `tail`:

```
$ tail -f ~/nohup.out
```

Możemy też jawnie skierować wyjście naszego polecenia do określonego pliku. Na przykład poniższy wiersz polecenia uruchomi określone polecenie w tle, przesyłając jego wyjście do pliku o nazwie *mój_tekst_wyjściowy.txt* w naszym katalogu macierzystym i kontynuując jego działanie nawet po zakończeniu jego sesji macierzystej:

```
$ nohup polecenie > ~/mój_tekst_wyjściowy.txt &
```

Jeżeli nie chcemy zapisywać wyjścia określonego polecenia, możemy się go pozbyć (i nie tworzyć pliku *nohup.out*) przez przekierowanie wyjścia do */dev/null*, będącego czymś w rodzaju linuksowego kosza na śmieci:

```
$ nohup polecenie > /dev/null &
```

Spowoduje to uruchomienie polecenia lub programu w tle, zignorowanie jego wyjścia przez wysyłanie go do */dev/null* i kontynuację jego działania nawet po zakończeniu jego sesji macierzystej.

Jeżeli użyliśmy polecenia `nohup` do utrzymania działania procesu po zakończeniu jego procesu macierzystego, wówczas nie będzie już można go ponownie podłączyć, jeśli zechcemy później go zamknąć. Jednak polecenie `nohup` chroni proces jedynie przed sygnałem `SIGHUP`. Wciąż można je zakończyć, używając „polecenia-młota" `kill` (`kill -9 PID`).

## Użycie polecenia disown do zadań działających w tle

Jeśli korzystamy z powłoki *bash*, możemy nakazać istniejącemu zadaniu ignorowanie sygnałów SIGHUP za pomocą `disown`, wbudowanego polecenia powłoki *bash*:

```
$ disown -h numer_zadania
```

Nakazuje ono zadaniu, które już działa w tle, utrzymanie działania po zamknięciu procesu macierzystego. Numer zadania można odszukać za pomocą polecenia powłoki `job`. Użycie wbudowanej opcji `-h` sprawi, że po wydaniu polecenia `disown` działające zadanie nie zostanie usunięte z tabeli zadań, lecz utrzyma swoje działanie po zakończeniu aktualnej powłoki. Możemy wciąż ponownie podłączyć się do procesu za pomocą standardowego mechanizmu *bash* `%numer-zadania`. Jeżeli użyjemy `disown` bez opcji, wówczas działające zadanie zostanie usunięte z tabeli zadań: wciąż będzie kontynuować działanie po wylogowaniu, ale nie będziemy w stanie się do niego podłączyć.

Możemy też użyć polecenia `disown` do utrzymywania działania wszystkich zadań uruchomionych w tle:

```
$ disown -ar
```

Nakazuje to wszystkim uruchomionym zadaniom utrzymywanie działania nawet po zamknięciu bieżącej powłoki.

## Zobacz również

- `man bash`
- `man nohup`

*— Jon Fox*

SPOSÓB 34.

# Odłączanie konsoli bez kończenia sesji

Jak uruchomić długotrwałe zadanie w pracy i podłączyć się do niego zdalnie z domu lub w czasie podróży.

Załóżmy taką sytuację: jesteśmy konsultantem systemów linuksowych, którego harmonogram pracy jest bardzo napięty. Jest dziewiąta rano, a my już od godziny siedzimy gdzieś nad instalacją bardzo dużej bazy danych, ale za godzinę musimy być w innym miejscu. Instalacja potrwa jeszcze na tyle długo, że nie zdążymy gruntownie wszystkiego przetestować, utworzyć odpowiednich baz danych i ustawić właściwych ograniczeń w celach bezpieczeństwa. Co zrobimy?

Możemy porozmawiać z naszym klientem i powiedzieć mu, że wrócimy później, aby wszystko dokończyć. Innym rozwiązaniem może być jednak uruchomienie zadania w sesji *screen* i późniejsze zalogowanie się z miejsca, w którym się znajdziemy, celem dokończenia pracy. Trzeba tu zaznaczyć, że nie wymaga to żadnej kompilacji dodatkowego oprogramowania w komputerze, gdyż *screen* jest zazwyczaj instalowany lub łatwo

dostępny w postaci pakietów dla dowolnej używanej przez nas dystrybucji. Więcej informacji na temat programu *screen*, łącznie z informacją o możliwości jego pobrania, znajdziemy na stronie głównej projektu GNU poświęconej programowi *screen*: *http://www.gnu.org/software/screen/*.

Rozpoczęcie pracy z programem *screen* jest nadzwyczaj proste. Wystarczy otworzyć swój ulubiony emulator terminala i wydać poniższe polecenie:

```
$ screen
```

Znajdziemy się w nowej powłoce, uruchomionej wewnątrz sesji *screen*. Wciąż możemy komunikować się z programem *screen* z wnętrza powłoki w podobny sposób jak komunikujemy się z wnętrza powłoki z dowolną aplikacją terminala konsoli. Kombinacją klawiszy używaną do wysłania sygnału wejściowego do programu *screen*, zamiast do powłoki uruchomionej wewnątrz sesji programu *screen*, jest *Ctrl+A*. W przypadku *screen* *Ctrl+A* przypomina w działaniu klawisz *Escape* w *vi* — zwraca on uwagę aplikacji, abyśmy mogli nakazać jej jakieś działanie. Na przykład, aby uzyskać szybki dostęp do listy poleceń programu *screen*, należy nacisnąć *Ctrl+A*, a następnie *?*.

W efekcie powinna pojawić się lista wielu poleceń, jakie można przekazać do programu *screen*. Jeżeli taka lista się nie pojawi, należy się upewnić, czy rzeczywiście jesteśmy w sesji *screen*. Wykonujemy to przez wywołanie programu *screen* z opcją `-list`. Powinniśmy ujrzeć komunikat podobny do poniższego:

```
$ screen -list
There is a screen on:
        28820.pts-2.willy    (Attached)
1 Socket in /var/run/screen-jonesy
```

Jak wynika z powyższego komunikatu, mamy tu uruchomioną sesję programu *screen*, do której jesteśmy aktualnie podłączeni. Identyfikator procesu (ID) to 28820 i jesteśmy przypisani do pseudoterminala o numerze 2. Spróbujmy teraz uruchomić nowe zadanie, które będziemy mogli kontynuować później z innego miejsca. Prostym sposobem na przetestowanie tego jest otwarcie pliku np. w edytorze Vim. Po otwarciu pliku naciskamy *Ctrl+A*, a następnie *d*, co spowoduje odłączenie nas od sesji programu *screen* i powrót do poprzedniej powłoki.

Teraz możemy już wyjść i udać się na następne spotkanie. Może w kolejnym miejscu przyjdzie nam zainstalować system operacyjny, co da nam trochę czasu podczas instalowania pakietów. Odpalamy laptopa, łączymy się za pomocą SSH z komputerem, na którym uruchomiona jest nasza sesja programu *screen*, i wpisujemy `screen -r`, aby ponownie podłączyć się do działającej sesji. Jeżeli mamy więcej niż jedną uruchomioną sesję programu *screen*, wpisujemy `screen -r pid`, gdzie *pid* jest identyfikatorem procesu sesji programu *screen*, do której chcemy się podłączyć (można go odczytać z wyjścia polecenia `screen -list`, które było omawiane wcześniej).

Oczywiście próba skojarzenia identyfikatora procesu sesji programu *screen* z tym, co się w danej sesji dzieje, może wydawać się nieco zniechęcająca, zwłaszcza przy wielu uruchomionych sesjach. Zamiast tego możemy przy uruchamianiu nadać sesji jakąś znaczącą

nazwę. Jeśli np. uruchamiany program *screen* w celu rozpoczęcia długotrwałej kompilacji oprogramowania, możemy wpisać po prostu `screen -S make`, wówczas przy następnym podłączeniu wystarczy wpisać `scren -r make`, zamiast pamiętać o tym, do jakiego identyfikatora procesu musimy się podłączyć.

## Skrypty dla programu screen

Jeżeli zarządzamy więcej niż kilkoma komputerami, pewnie znaleźliśmy już jakąś metodę automatyzacji procesu łączenia się z niektórymi podzbiorami naszych komputerów usługowych po zalogowaniu czy to za pomocą ikony na pulpicie, czy też w jakiś inny, bardziej zautomatyzowany sposób niż ręczne otwieranie okien terminala i wpisywanie poleceń w celu połączenia się z każdym z tych komputerów. Jeśli używamy kluczy SSH (sposób 66. w książce *Serwer linuksowy okiem hakera*), możemy utworzyć prosty skrypt powłoki do zautomatyzowania za nas tego procesu. Oto przykład takiego skryptu powłoki:

```
#!/bin/bash
screen -d -m -S svr1 -t jonesy@svr1 ssh server1.linuxlabolatory.org
screen -d -m -S svr2 -t jonesy@svr2 ssh server2.linuxlabolatory.org
screen -d -m -S svr3 -t jonesy@svr3 ssh server3.linuxlabolatory.org
```

Zapisujemy ten skrypt w swoim katalogu *~/bin* i upewniamy się, że nadaliśmy mu atrybut wykonywalności.

Skrypt działa przez wywołanie *screen* z opcjami `-d -m`, co nakazuje programowi *screen* rozpoczęcie sesji, ale nie podłączanie do niej. Zauważmy też, że użyta tu została opcja `-S` służąca do określenia nazwy sesji, więc kiedy zajdzie potrzeba podłączenia się np. do sesji *svr1*, wystarczy tylko wpisać `screen -r svr1`.Oprócz tego użyłem opcji `-t` do określenia tytułu powłoki, który widoczny będzie na pasku tytułowym emulatora terminala, co pomaga się zorientować, gdzie jesteśmy.

Uruchomienie powyższego skryptu spowoduje otwarcie sesji SSH na (w tym przypadku) komputerach *server1*, *server2* i *server3*. Kuszącym pomysłem może wydawać się umieszczenie tego w swoich skryptach startowych powłoki. *Nie wolno tego robić!* W środowiskach, gdzie katalogi macierzyste (a więc także skrypty startowe powłoki) są współdzielone pomiędzy komputerami, może to wywołać niekończący się strumień zapętlonych sesji SSH.

## Zobacz również

- *http://www.gnu.org/software/screen/*
- Rob Flickener, *Serwer linuksowy okiem hakera,* wydawnictwo RM, Warszawa, 2003.
- „Zastosowanie polecenia script do zaoszczędzenia sobie czasu i szkolenia innych” **[sposób 35]**

SPOSÓB 35.

# Zastosowanie polecenia script do zaoszczędzenia sobie czasu i szkolenia innych

Standardowe polecenie script zapewnia powtarzalność i świetnie nadaje się do szkolenia początkujących administratorów.

Studenci uczęszczający na zajęcia z informatyki pewnie spotkali się już z poleceniem `script`. Profesorzy często dają zadania wymagające dostarczenia zapisu całej interaktywnej sesji powłoki, więc studenci wykonują to zazwyczaj przez wydanie na początku swojej sesji polecenia `script`. Powoduje to kopiowanie do pliku (o domyślnej nazwie *typescript*) wszystkich zachodzących operacji wejścia-wyjścia. Przy zakończeniu wystarczy wpisać `exit` i można już zwrócić wykładowcy plik *typescript*.

Jednak zastosowania polecenia `script` wykraczają poza salę lekcyjną. W niektórych ściśle produkcyjnych środowiskach wszystko, co zachodzi w rzeczywistych (nietestowych), w pełni produkcyjnych systemach, musi być „powtarzalne" — innymi słowy, zarejestrowane, gruntownie przetestowane i udokumentowane do takiego stopnia, aby osoba zarządzająca zmianami, bez przygotowania w zakresie Uniksa, była w stanie to zrobić. Narzędziem, które można wykorzystać do utworzenia dokumentacji, jest `script`. Wciąż trzeba będzie napisać procedurę poza procesem rejestracji, używając firmowego standardu kodowania, ale potem można będzie zarejestrować sesję, gdzie zostało wywołane dane narzędzie, i przekazać zapis personelowi zarządzającemu zmianami, aby mógł on z kolei zobaczyć, co dokładnie należy zrobić.

Wyjątkowo przydatną cechą polecenia `script` jest możliwość zapisu informacji czasowych do oddzielnego pliku. Dzięki temu cała sesja terminala może zostać później odtworzona za pomocą polecenia `scriptreplay` z wykorzystaniem tych samych przedziałów czasowych co oryginalna sesja! To świetne rozwiązanie dla nowych użytkowników mających trudności z zapamiętaniem sposobu wykonania zadań, których nie mieliśmy czasu im zapisać.

Oto krótka sesja z użyciem tych dwóch poleceń:

```
$ script -t 2> timing
Script started, file is typescript
$ ls
Desktop  hax  hog.sh  My Computer ostg  src
S pwd
/home/jonesy
$ file hax
hax: empty
$ exit
exit
Script done, file is typescript
$ scriptreplay timing
$ ls
Desktop  hax  hog.sh  My Computer ostg  src
S pwd
/home/jonesy
$ file hax
hax: empty
```

```
$ exit
exit
$ ls
Desktop  hax  hog.sh  My Computer ostg  src
S pwd
/home/jonesy
$ file hax
hax: empty
$ exit
exit
```

Użycie opcji `-t` nakazuje poleceniu `script` kierowanie informacji czasowych na standardowe wyjście, więc przekierowujemy je do pliku (tutaj: *timing*) do wykorzystania później. Następnie wydajemy polecenie `scriptreplay`, wskazując mu plik *timing*. Nie trzeba tu wskazywać pliku z zapisem sesji, gdyż domyślnie polecenia to szuka pliku o nazwie *typescript*, który jest domyślnym plikiem zapisu dla polecenia `script`.

Rejestrowane jest każde naciśnięcie klawisza, więc jeśli coś pomylimy i naciśniemy później klawisz backspace w celu skasowania kilku znaków, zostanie to odzwierciedlone przy odtwarzaniu sesji! Trzeba również pamiętać, że odtwarzanie sesji będzie działać poprawnie jedynie na terminalu, w którym został utworzony plik zapisu sesji.

Jeśli chcemy komuś pokazać sposób wykonania jakiegoś zadania w czasie rzeczywistym, istnieje inny sposób, gdzie pomocne może być polecenie `script`. Należy utworzyć nazwany potok i przekierować do niego wyjście. Ktoś inny, zalogowany zdalnie, może następnie przyłączyć potok i zobaczyć, co się dzieje w trakcie danego procesu.

Działa to tak, że najpierw należy utworzyć nazwany potok za pomocą polecenia `mkfifo`:

```
$ mkfifo out
```

a następnie wydać polecenie `script` z opcją `-f`, co spowoduje opróżnianie całego wyjścia do potoku przy każdym zapisie. Bez tej opcji to nie zadziała. Ostatnim argumentem polecenia `script` jest plik, do którego zostanie wysyłane wyjście:

```
$ script -f out
```

Jesteśmy teraz w sesji, która wygląda i zachowuje się całkiem zwyczajnie, ale inna osoba może się zalogować z dowolnego miejsca i uruchomić następujące polecenie, aby obserwować nasze działania:

```
$ cat out
```

Wszystkie nasze działania będą widoczne dla tego użytkownika. Jest to nieco łatwiejsze od pamiętania, jak skonfigurować wieloużytkownikowe sesje programu *screen*!

## Zobacz również

- „Odłączanie konsoli bez kończenia sesji” **[sposób 34]**

SPOSÓB 36.

# Zautomatyzowana instalacja Linuksa po uruchomieniu komputera

Jak za pomocą demonów serwera działających w naszym środowisku oraz prostej konfiguracji PXE uczynić instalację tak prostą jak włączenie komputerów docelowych.

Wiele dystrybucji posiada jakąś formę automatycznej instalacji. W SUSE jest AutoYaST, Debian ma FAI (ang. *Fully Automated Install*), Red Hat ma kickstart itd. Narzędzia te działają zwykle na zasadzie analizy pliku konfiguracyjnego lub szablonu używającego słów kluczowych przekazujących programowi instalacyjnemu sposób skonfigurowania komputera. W większości z nich istnieje możliwość zastosowania własnych skryptów w celu przeprowadzenia zadań, które nie są wykonywane przez szablon instalacyjny.

Ostateczny rezultat to ogromna oszczędność czasu. Choć przygotowanie i przetestowanie szablonu wymaga na początku poświęcenia pewnej ilości czasu, to gdy już się z tym uporamy, możemy za pomocą pojedynczego pliku szablonu przeprowadzić instalację na wszystkich komputerach tej samej klasy lub błyskawicznie zmodyfikować plik szablonu w celu dostosowania zautomatyzowanej instalacji do komputera stanowiącego „specjalny przypadek". Przykładowo, szablon dla serwera WWW może zostać szybko zmodyfikowany przez usuniecie odniesień do Apache i zastąpienie ich na przykład Sendmailem.

Jedynym minusem zautomatyzowanej instalacji jest to, że jeśli w miejscu, gdzie mają nastąpić czynności automatyzujące, nie ma żadnej obsługiwanej infrastruktury, wówczas trzeba przeprowadzić rozruch z płyty CD lub innego nośnika i wykonać kilka poleceń w celu uruchomienia procesu instalacyjnego. Byłoby to naprawdę wspaniałe, gdyby instalacja Linuksa była tak prosta, że wymagałaby jedynie przejścia się po serwerowni (lub laboratorium czy dowolnym pomieszczeniu, w którym znajduje się wiele komputerów wymagających instalacji), włączenia wszystkich komputerów i wyjścia z pomieszczenia. Przyjrzyjmy się, w jaki sposób można to (i nie tylko to!) wykonać.

W podanych tu przykładach do zautomatyzowania instalacji będę korzystał z mechanizmu kickstart systemu Red Hat (Fedora), ale inne narzędzia mogą z powodzeniem dać podobne lub identyczne wyniki.

## Kroki przygotowawcze

Lista składników, jakie należy skonfigurować, może nieco przerażać, ale jest to o wiele prostsze, niż wygląda, a gdy już raz to przeprowadzimy, wówczas automatyzacja procesu instalacyjnego i powtarzalność takiej instalacji to drobnostka. Jednak zanim cokolwiek zrobimy, musimy się upewnić, że komputery, na których ma się odbywać instalacja, zaopatrzone są w karty sieciowe obsługujące mechanizm PXE (ang. *Preboot eXecution Environment*). Jest to standardowy mechanizm rozruchowy obsługiwany przez oprogramowanie sprzętowe wgrane do karty sieciowej naszego serwera. Większość kart sieciowych klasy serwerowej, a nawet najnowsze karty sieciowe komputerów biurkowych, obsługuje mechanizm PXE. Można to sprawdzić, wchodząc w ustawienia BIOS-u i szukając opcji włączającej

PXE lub uważnie obserwować komunikaty rozruchowe, aby zobaczyć, czy są tam ustawienia dla rozruchu PXE. Na wielu systemach rozruch przez PXE można wykonać przez naciśnięcie klawisza funkcyjnego w czasie startu.

### Konfiguracja DHCP

Jeżeli jesteśmy pewni, że nasze komputery obsługują PXE, możemy przystąpić do konfiguracji naszego serwera DHCP (lub BOOTP). Usługa ta odpowiada na rozgłaszania PXE nadchodzące z węzłów docelowych przez dostarczenie adresów IP wraz z nazwą pliku rozruchowego oraz adresem komputera macierzystego, z którego może zostać pobrany plik rozruchowy. Oto typowy wpis dla komputera docelowego:

```
host pxetest {
    hardware ethernet 0:b:db:95:84:d8;
    fixed-address 192.168.198.112;
    next-server 192.168.101.10;
    filename "/tftpboot/linux-install/pxelinux.0";
    option ntp-servers 192.168.198.10, 192.168.198.23;
}
```

Wszystkie powyższe wiersze są doskonale przewidywalne dla wielu środowisk. Jedynie dwa wyróżnione wiersze odnoszą się to tego, co zamierzamy przeprowadzić. Po dostarczeniu tych informacji do klienta, wie on, jakiego ma zażądać pliku oraz na jakim serwerze się on znajduje.

W tym momencie jesteśmy już w stanie uruchomić klienta, nakazać mu rozruch PXE i zobaczyć, czy pobierze adres IP oraz czy powiadomi nas o tym, jaki to adres. W przypadku implementacji PXE nieprzekazującej żadnych komunikatów możemy uzyskać potwierdzenie, sprawdzając dzienniki DHCP serwera. Zakończone powodzeniem żądanie DHCP oraz odpowiedź powinny wyglądać mniej więcej tak jak w poniższym pliku dziennika:

```
Aug 9 06:05:55 livid dhcpd: [ID 702911 daemon.info] DHCPDISCOVER from
00:4O:96:35:22:ff (jonesy-thinkpad) via 172.16.1.1
Aug 9 06:05:55 livid dhcpd: [ID 702911 daemon.info] DHCPOFFER on
192.168.198.101 to 00:4O:96:35:22:ff (jonesy-thinkpad) via 192.168.198.100
```

### Konfiguracja serwera TFTP

Gdy komputer jest już w stanie pobrać adres IP, następną rzeczą, jaką spróbuje wykonać, będzie przekazanie do pliku rozruchowego swojej konfiguracji startowej. Zostanie ona umieszczona na serwerze TFTP. Serwer TFTP jest dołączony do wielu dystrybucji lub jest łatwo dostępny. W zależności od naszej dystrybucji może on być (choć nie musi) uruchamiany przez *inetd* lub *xinetd*. Jeśli jest on uruchamiany z *xinetd*, powinniśmy być w stanie włączyć tę usługę przez zmodyfikowanie pliku */etc/xinetd.d/in.tftpd* i zmianę wartości opcji `disable` na `no`. Po zapisaniu zmian i ponownym uruchomieniu *xinetd* usługa zostanie włączona. Jeżeli nasz system uruchamia TFTP przez *inetd*, należy się upewnić, że w pliku */etc/inetd.conf* występuje wpis dotyczący demona TFTP i nie jest on oznaczony jako komentarz. Jeśli nasz system uruchamia serwer TFTP jako stałego demona, wówczas należy się tylko upewnić, że demon TFTP jest automatycznie uruchamiany przy starcie systemu.

Następnie należy utworzyć strukturę katalogów dla naszych plików rozruchowych, jąder oraz plików konfiguracyjnych. Oto podstawowa (bez żadnych dodatków i zawierająca tylko to, co absolutnie konieczne) struktura katalogów, która zostanie omówiona za chwilę:

```
/tftpboot/
    linux-install/
    pxelinux.0
    vmlinuz
    initrd.img
    pxelinux.cfg/
        default
```

Najpierw uruchamiamy polecenie, które szybko utworzy opisaną wyżej hierarchię katalogów:

```
$ mkdir -p /tftpboot/linux-install/pxelinux.cfg
```

Opcja `-p` programu `mkdir` tworzy niezbędne katalogi nadrzędne wymienione w ścieżce, jeżeli jeszcze nie istnieją. Skoro katalogi są już na swoim miejscu, kolej na pliki! Pierwszym jest plik, który zostanie zażądany przez klienta: *pxelinux.0*. Plik ten jest prostym bootloaderem umożliwiającym systemowi jedynie pobranie pliku konfiguracyjnego, z którego uzyska on informację dotyczącą tego, jakie jądro i obraz RAM-dysku należy pobrać w celu kontynuacji. Sam plik można uzyskać z pakietu *syslinux*, który jest łatwo dostępny dla prawie wszystkich dystrybucji na świecie. Należy więc pobrać ten pakiet (lub wersję źródłową), zainstalować go (lub rozpakować archiwum tar) i skopiować plik *pxelinux.0* do pliku */tftpboot/linux-install/pxelinux.0*.

Po dostarczeniu pliku do klienta następnym wykonywanym przez niego działaniem będzie odszukanie pliku konfiguracyjnego. Należy tu zwrócić uwagę na fakt, że *pxelinux.0* dostarczany z pakietem *syslinyx* domyślnie zawsze szuka swojego pliku konfiguracyjnego w *pxelinux.cfg*. Ponieważ nasz serwer DHCP określa jedynie plik rozruchowy, a musimy mieć różne pliki konfiguracyjne dla każdego komputera uruchamianego przez PXE, szuka on pliku konfiguracyjnego w sposób następujący:

1. Szuka pliku o nazwie używającej jego własnego adresu MAC w formacie szesnastkowym zapisanej dużymi literami, poprzedzonego typem jego ARP w postaci szesnastkowej, gdzie wszystkie pola są rozdzielone myślnikami. Tak więc, posługując się naszym przykładem, w przypadku komputera docelowego o adresie MAC 00:40:96:35:22:ff plik będzie nosił nazwę *01-00-40-96-35-22-FF*. Cyfry 01 w pierwszym polu to zapisany w postaci szesnastkowej typ ARP (ARP typu 1).
2. Następnie szuka on pliku o nazwie używającej adresu IP klienta zapisanej dużymi literami w formacie szesnastkowym. Projekt *syslinux* dostarcza plik binarny o nazwie *gethostip* służący do odszukania tej wartości, co jest znacznie wygodniejsze niż obliczanie tego w pamięci. Po przekazaniu mojego adresu IP polecenie to zwraca `COA8C665`.

3. Jeśli żaden z powyższych plików nie istnieje, klient powtarza szukanie w poszukiwaniu odpowiedniego pliku, wykonując pętlę zmieniająca kolejny znak od końca reprezentacji szesnastkowej swojego adresu IP (*COA8C66*, *COA8C6*, *COA8C*, *COA8…* itd.).
4. Jeśli również to nie przyniesie rezultatu, na koniec klient będzie szukał pliku o nazwie *default*. Jeśli go nie będzie, przerwie działanie.

W naszym prostym ustawieniu testowym umieściliśmy na swoim miejscu plik o nazwie *default*, ale w większych konfiguracjach możemy tu umieścić pliki konfiguracyjne dla każdej klasy komputerów, na których będzie przebiegać instalacja. Na przykład, jeśli mamy do zainstalowania 40 serwerów WWW i 10 serwerów baz danych, nie musimy tworzyć 50 plików konfiguracyjnych — wystarczy utworzyć jeden o nazwie *serwery-www* i jeden o nazwie *serwery-bd*, a następnie utworzyć dowiązania symboliczne dla każdego z komputerów docelowych albo za pomocą polecenia *gethostip*, albo przez dodanie typu ARP do adresu MAC, jak opisano to powyżej.

Bez względu na to, w jaki sposób wybierzemy, plik konfiguracyjny musimy poinformować klienta, jakiego jądra ma użyć oraz jakie opcje mają być przesłane do jądra podczas startu. Jeśli brzmi to znajomo, to dobrze, ponieważ przypomina to bardzo konfigurację LILO lub GRUB. Oto nasz domyślny plik konfiguracyjny *default*:

```
default linux

label linux
    kernel vmlinuz
    append ksdevice=eth0 load_ramdisk=1 prompt_ramdisk=0 network
    ks=nfs:mójserwer:/kickstart/Profiles/pxetest
```

Dodałem tutaj do jądra szereg opcji. Opcje `ksdevice` oraz `ks=` są specyficzne dla mechanizmu instalacyjnego kickstart systemu Red Hat; informują one klienta odpowiednio o urządzeniu używanym do instalacji sieciowej (w razie, gdyby było ich więcej niż jedno) oraz o tym, jak i skąd uzyskać szablon kickstartu. Opcja `ks=` wskazuje, że instalacja będzie przebiegać przez NFS z serwera *mójserwer*. Szablonem kickstartu jest */kickstart/Profiles/pxetest*.

Klient jednak nie ruszy dalej, dopóki nie pobierze jądra i obrazu RAM-dysku. Nakazaliśmy mu użycie jądra *vmlinuz* i domyślnego obrazu RAM-dysku, którym jest zawsze *initrd.img*. Oba te pliki położone są w tym samym katalogu co *pxelinux.0*. Pliki te można uzyskać z płyty instalacyjnej, która będzie wykorzystana do instalacji. W tym przypadku, ponieważ chodzi o system Red Hat, należy przejść do katalogu *isolinux* na płycie CD i skopiować stamtąd obrazy jądra i RAM-dysku do katalogu */tftpboot/linux-install*.

## Uruchomienie

Nasz komputer jest przygotowany do PXE, nasz serwer DHCP jest skonfigurowany do przekazania komputerom docelowym niezbędnych informacji, a serwer TFTP jest przygotowany do dostarczenia komputerowi pliku rozruchowego, pliku konfiguracyjnego, jądra

oraz obrazu RAM-dysku. Pozostało już tylko uruchomić komputery! Dla jasności, poniżej znajduje się lista zachodzących działań:

1. Uruchamiamy komputer i naciskamy klawisz funkcyjny w celu poinformowania komputera, że ma on wykonać rozruch przez PXE.
2. Klient wysyła sygnał rozgłoszeniowy i (miejmy nadzieję) otrzymuje adres IP wraz z nazwą i położeniem pliku rozruchowego.
3. Klient kontaktuje się z serwerem TFTP, prosi o plik rozruchowy i (miejmy nadzieję) otrzymuje go.
4. Plik rozruchowy zostaje uruchomiony, po czym ponownie kontaktuje się z serwerem TFTP w celu pobrania pliku konfiguracyjnego za pomocą omawianej wcześniej procedury. W naszym przypadku pobrany zostanie plik o nazwie *default* zawierający informacje uruchomieniowe.
5. Klient pobiera jądro i obraz RAM-dysku określone w pliku *default* i rozpoczyna procedurę kickstart, używając serwera NFS podanego w wierszu `append` parametrów jądra.

## Szybkie rozwiązywanie problemów

Oto kilka z problemów, jakie mogą wystąpić, oraz sposoby zaradzenia im:

- Jeśli wystąpią błędy TFTP ACCESS VIOLATION, mogą one być spowodowane niemal wszystkim. Jednak pierwsze, co należy sprawdzić, to to, czy serwer TFTP może uzyskać dostęp do pliku (używając klienta TFTP) oraz czy konfiguracja DHCP dla komputera docelowego zawiera tylko parametr `filename` określający *pxelinux.0*, a nie zawiera parametru BOOTP `bootfile-name`.
- Jeśli wystąpi błąd przy pobieraniu pliku rozruchowego i otrzymamy komunikat „TFTP open timeout" lub inny komunikat związany z przekroczeniem czasu oczekiwania, należy dla pewności sprawdzić, czy serwer TFTP przyjmuje połączenia z komputera klienta.
- Jeśli błąd wystąpi przy pobieraniu adresu IP, poszukajmy wskazówek w plikach dziennika DHCP, wyszukując adres MAC klienta za pomocą polecenia `grep`. Jeśli go nie znajdziemy, oznacza to, że pakiety rozgłoszeniowe klienta nie docierają do serwera DHCP. W takim przypadku przyczyną mogą być reguły firewalla (lub ACL).
- Jeśli nie można uzyskać pliku konfiguracyjnego kickstartu, upewnijmy się, czy mamy odpowiednie uprawnienia do zamontowania źródła NFS, czy prosimy o właściwy plik oraz czy nie popełniliśmy gdzieś literówki!
- Jeśli wszystko zawiedzie, a mamy możliwość przetestowania za pomocą identycznego komputera lub innego jądra, zróbmy to, ponieważ przyczyną może być uszkodzenie sterownika lub karty sieciowej. Na przykład w pierwszym jądrze, którego użyłem w teście, uszkodzony był sterownik sieciowy b44 i nie mogłem pobrać pliku kickstart.

SPOSÓB 37.

# Zmiana laptopa w prowizoryczną konsolę

Za pomocą programu minicom i kabla (lub dwóch, jeśli nasz laptop nie jest wyposażony w port szeregowy) możemy podłączyć się do portu konsoli dowolnego serwera.

Jest wiele sytuacji, w których możliwość podłączenia się do portu szeregowego konsoli serwera może nas wybawić z opresji. W mojej codziennej pracy czasem robię tak dla wygody, gdyż mogę wprowadzać polecenia na konsoli serwera, a tym samym czasie przeglądać dokumentację, która jest zwykle dostępna jedynie w formacie PDF (czego nie mogę zrobić na zwykłym terminalu). Przydaje się to również wtedy, gdy wykonujemy zadania na komputerze, który nie jest jeszcze podłączony do żadnej konsoli lub jesteśmy gdzieś u klienta i chcemy od razu rozpocząć pracę bez konieczności uczenia się niuansów rozwiązań zastosowanych w konkretnej konsoli na serwerze klienta.

## Przedstawiamy minicom

Jak to jest możliwe? Istnieje stare jak świat rozwiązanie dostarczane w postaci pakietu binarnego w każdej chyba dystrybucji Linuksa. Nosi ono nazwę *minicom*. Jeśli musimy przeprowadzić kompilację ze źródeł, możemy je pobrać ze strony *http://alioth.debian.org/projects/minicom/*. *minicom* może robić mnóstwo wspaniałych rzeczy, ale ja używam go w celu dostarczenia serwerowi interfejsu konsoli przez połączenie szeregowe za pomocą kabla null-modem (zwanego inaczej szeregowym kablem krosującym).

Tak naprawdę to jedno wielkie kłamstwo. Mój laptop, jak się okazało, nie ma portu szeregowego! Nawet nie sprawdziłem tego przy zamawianiu, ale odkryłem, że wiele nowszych laptopów również go nie posiada. Jeśli jesteśmy w podobnej sytuacji, to bez obaw! Wszędzie w sklepach internetowych dostępne są przejściówki USB-port szeregowy. Wystarczy tylko umieścić taka przejściówkę w porcie USB, podłączyć do niej kabel null-modem, a drugi koniec kabla umieścić z porcie szeregowym serwera i… możemy działać.

Skoro już zadbaliśmy o sprawy sprzętowe, możemy przejść do konfiguracji programu *minicom*. W systemach Debian domyślnym katalogiem konfiguracyjnym jest */etc/minicom*. W systemach Red Hat pliki konfiguracyjne przechowywane są zwykle w */etc* i nie mają swojego własnego katalogu. Dostosowanie konfiguracji wykonuje się przez wydanie (jako root) polecenia:

```
# minicom -s
```

Spowoduje to otwarcie interfejsu tekstowego, w którym będziemy mogli zmienić niezbędne opcje. Konfiguracja jest domyślnie zapisywana w pliku o nazwie *minirc.dfl*, ale możemy użyć opcji menu *Zapisz setup jako*, aby podać inną nazwę pliku konfiguracyjnego. Możemy tak zrobić w celu przygotowania kilku plików konfiguracyjnych dla różnych potrzeb — profil używany przy uruchomieniu może zostać przesłany do programu *minicom* jako pojedynczy argument.

Na przykład, jeśli wydam polecenie `minicom -s`, a mam już swój domyślny profil przechowywany w pliku *minicom.dfl*, mogę zmienić szybkość transmisji z domyślnej 9600 na 115 200 bodów, a następnie zapisać ten profil pod nazwą *szybki*. Plik utworzony w taki sposób będzie miał nazwę *minicom.szybki*, ale przy uruchamianiu wystarczy podać samą nazwę profilu, a nie nazwę pliku, jak w poniższym przykładzie:

```
# minicom szybki
```

Oczywiście zakładamy tu, że zwykły użytkownik ma dostęp do tego profilu. Istnieje plik dostępu użytkowników o nazwie *minicom.users*, w którym możemy określić, którzy użytkownicy będą mieć dostęp do danych profilów.

Nieco prostszym sposobem uzyskania działającej konfiguracji jest podpatrzenie jej u kogoś innego. Oto podstawowa konfiguracja dla programu *minicom*. Choć jest bardzo prosta, to tak naprawdę jest to jedyna konfiguracja, jaka kiedykolwiek była mi potrzebna:

```
# Wygenerowano przez minicom - użyj menu w minicomie do zmiany ustawień.
pu port             /dev/ttyUSB0
pu baudrate         9600
pu bits             8
pu parity           N
pu stopbits         1
pu minit
pu mreset
pu mconnect
pu mhangup
```

Załączyłem tu opcje przechowywane domyślnie, mimo że nie są one wykorzystywane. Te nieużywane ustawienia dotyczą sytuacji, w których *minicom* musi nawiązać połączenie modemowe. Zauważmy, że w tym pliku konfiguracyjnym używanym przeze mnie urządzeniem szeregowym (urządzeniem lokalnym, przez które komunikuje się *minicom*) jest */dev/ttyUSB0*. Urządzenie to zostało utworzone i przypisane przez moduł jądra o nazwie *usbserial*. Jeżeli używamy przejściówki USB-port szeregowy i nic nie wskazuje na to, że urządzenie takie na porcie szeregowym zostało wykryte i przypisane do właściwego urządzenia systemowego przez jądro systemu, wówczas należy sprawdzić, czy mamy ten moduł. Prawie każda współczesna dystrybucja zawiera moduł *usbserial*, który jest ładowany dynamicznie w razie potrzeby, ale jeśli będziemy kompilować swoje własne jądro, pamiętajmy o dodaniu tego modułu! W pliku konfiguracyjnym jądra Linuksa opcja `CONFIG_USB_SERIAL` powinna mieć wartość `y` lub `m`. Nie może być oznaczona jako komentarz.

Następnym ustawieniem jest `baudrate`. Wartość ta powinna być taka sama u klienta i na serwerze. W tym przypadku wybrałem 9600 nie dlatego, że chcę mieć terminal działający w żółwim tempie, ale dlatego, że taka jest prędkość skonfigurowana na serwerach, do którym zwykle się podłączam. Jest to wystarczająca szybkość dla większości działań niewymagających śledzenia ogromnych plików dziennika, aktualizowanych kilka razy na sekundę.

Następne trzy ustawienia określają sposób, w jaki dane klienta zostaną wysyłane do serwera. W tym przypadku pojedynczy znak będzie składał się z ośmiu bitów, bez bitu parzystości i jednego bitu stopu. Takie ustawienie (określane jako „8N1") jest zdecydowanie najbardziej powszechnie stosowanym ustawieniem w asynchronicznej komunikacji szeregowej. Ustawienia takie są na tyle standardowe, że nigdy jeszcze nie musiałem ich zmieniać w moim pliku *minicom.conf* — tak naprawdę jedynym ustawieniem, jakie zmieniałem, było `baudrate`.

## Testowanie

Gdy mamy już konfigurację na miejscu, możemy podłączyć kabel null-modem lub przejściówkę USB-port szeregowy do swojego laptopa i połączyć drugi koniec kabla do portu szeregowego konsoli na serwerze. Jeśli robimy to po raz pierwszy, przypominam, że port szeregowy konsoli na serwerze to 15-pinowe gniazdo męskie wyglądające podobnie do męskiej wersji standardowego portu VGA. Prawdopodobnie będzie to jedyne wolne gniazdo, do którego będzie można podłączyć kabel null-modem! Jeśli są takie dwa, na ogół gniazdo znajdujące się u góry (przy ustawieniu pionowym) lub po lewej (przy ustawieniu poziomym) będzie urządzeniem *ttyS0* serwera, a pozostałe będzie urządzeniem *ttyS1*.

Gdy laptop i serwer są już ze sobą fizycznie połączone, następną rzeczą do wykonania jest włączenie aplikacji terminala i uruchomienie *minicom*:

```
$ minicom
```

Polecenie to uruchomi *minicom* z jego domyślną konfiguracją. Należy zauważyć, że na wielu systemach po uruchomieniu samej aplikacji nic się nie wydarzy: trzeba będzie nacisnąć klawisz *Enter* raz lub dwa, aby uzyskać dostęp do znaku zachęty.

## Rozwiązywanie problemów

Rzadko mam kłopoty z używaniem w ten sposób programu *minicom*, zwłaszcza gdy do zapewnienia komunikacji po stronie serwera używany jest *agetty*, ponieważ *agetty* jest dość tolerancyjny i potrafi dostosować się do takich rzeczy jak siedmiobitowe kodowanie znaków czy do innych nietypowych ustawień. W razie gdybyśmy nie otrzymali żadnego wyjścia lub gdyby wyjście było zniekształcone, sprawdźmy dla pewności, czy prędkość transmisji u klienta odpowiada prędkości transmisji na serwerze. Skontrolujmy też, czy jesteśmy podłączeni do właściwego portu szeregowego! Spróbujmy wpisać na serwerze poniższe polecenie, aby uzyskać szybkie podsumowanie ustawień serwera:

```
$ grep agetty /etc/inittab
co:2345:respawn:/sbin/agetty ttyS0 9600 vt100-nav
$
```

Komunikat ten pokazuje, że *agetty* działa tu na urządzeniu *ttyS0* z ustawieniem 9600 bodów. Ostatnia opcja, `vt100-nav`, umieszczona tu została przez program instalacyjny systemu Fedora, który domyślnie ustawia wpis *inittab*, jeśli coś jest podłączone do portu

konsoli w trakcie instalacji. Opcja `vt100-nav` ustawia zmienną środowiskową `TERM`. Jeżeli pominiemy to ustawienie, większość systemów linuksowych przyjmie jako ustawienie domyślnie `vt100`, co jest na ogół prawidłowe. Jeśli chcemy, możemy nakazać programowi *minicom* użycie alternatywnego terminala po stronie klienta za pomocą opcji `-t`.

Jeśli przy uruchamianiu *minicom* napotkamy problemy, upewnijmy się, czy nie chodzi o ograniczenia w pliku konfiguracyjnym dotyczące tego, kto może używać domyślnego profilu.

SPOSÓB 38.

## Dokumentacja przydatna dla leniwych

Dokumentacja w formie stron WWW jest wspaniała, ale niezbyt dostępna z wiersza poleceń. Jednak strony podręcznika systemowego mogą być z nami zawsze.

Znam niewielu administratorów, którzy byliby wielbicielami tworzenia i utrzymywania dokumentacji. To nie jest zabawne. Poza tym nie ma w tym nic heroicznego. Znajomi administratorzy nie będą poklepywać po plecach i gratulować utworzenia wspaniałej dokumentacji. Co więcej, trudno jest zobaczyć korzyści, jakie odnoszą z dokumentacji końcowi użytkownicy, jeśli opracowaliśmy dokumentację do czegoś, z czego będą korzystać jedynie administratorzy, a jeśli jakiś administrator pisze dokumentację, prawdopodobnie i tak już wszyscy jego znajomi znają to, co opisuje!

Cóż, to tylko jeden punkt widzenia. Jednakże faktem jest, że fluktuacja kadr istnieje, a nawet się zwiększa. Możliwe, że z powodu zwiększenia lub wymiany personelu trafią do nas nowi administratorzy, i trzeba ich będzie przeszkolić w kwestiach dotyczących naszych programów narzędziowych, skryptów, procesów, procedur i różnych sposobów specyficznych dla naszej organizacji. Ten proces nauki jest również częścią adaptacji nowego administratora w zespole i powinien być tak łatwy, jak to możliwe, dla dobra zespołu i naszego własnego.

Podczas moich podróży zauważyłem, że ostatnia rzecz, o jakiej marzą administratorzy, to pisanie dokumentacji. Jedyną czynnością znajdująca się u nich niżej na liście zadań, jakie pragną zrobić, jest pisanie dokumentacji WWW. Próbowałem zapoznać ich z edytorami HTML pracującymi w trybie WYSIWYG, ale nie chcieli tego. Administratorzy Uniksa są całkiem zadowoleni z używania w swojej pracy narzędzi uniksowych („Daj mi Vim, albo daj mi umrzeć!").

Kolejną rzeczą, której administratorzy zwykle nie chcą poznać, są takie narzędzia jak LaTeX, SGML czy *groff*, służące do tworzenia oficjalnej dokumentacji. Całkiem zadowala ich zwykły tekst, który jest łatwy do wpisywania i do zrozumienia dla każdego, kto zetknął się z „surowym" plikiem. Cóż, znalazłem w końcu narzędzie, które pozwala administratorom tworzyć strony podręcznika systemowego ze zwykłych plików tekstowych i jest całkiem niezłe. Nazywa się *txt2man*.

Oczywiście razem z nim dostarczana jest strona podręcznika, która zawiera więcej informacji, niż jest to konieczne do efektywnego korzystania z tego narzędzia. Jest to prosty skrypt powłoki, któremu podajemy nasz plik tekstowy wraz z dowolnymi opcjami, jakie chcemy mu przekazać w celu uzyskania lepszych rezultatów końcowych, i uzyskujemy w efekcie doskonale użyteczną stronę podręcznika systemowego. Tak to działa.

Mam skrypt o nazwie *cleangroup*, który napisałem w celu zaprowadzenia porządku po osobach, które odeszły z naszego działu (zobacz również informacje zawarte w sposobie 77., „Czyszczenie NIS po odejściu użytkowników"). Analizuje on naszą mapę NIS i pozbywa się wszystkich odwołań do użytkowników, którzy nie istnieją już w mapie haseł NIS. To przydatny skrypt, ale ponieważ sam go utworzyłem, nie istnieje żaden powód, dla którego dwaj nowi administratorzy przyjęci na pełny etat mieliby wiedzieć o jego istnieniu i zastosowaniu. Dlatego utworzyłem nowy katalog dla stron podręcznikowych i rozpocząłem pracę nad pisaniem dokumentacji dla wszystkich napisanych lokalnie narzędzi, o których powinni wiedzieć nowi administratorzy. Oto tekst, jaki wpisałem w celu utworzenia strony podręcznikowej:

```
NAZWA
  cleangroup - usuwa użytkowników z grup, jeśli konto nie istnieje
SKŁADNIA
  /usr/local/adm/bin/cleangroup plik_grupy
OPIS
  cleangroup to skrypt Perla sprawdzający każdy UID w pliku grup względem
  mapy haseł YP. Jeżeli użytkownik tam nie istnieje, zostaje usunięty
  z grupy.

  Jedynym argumentem pliku jest plik grup, jest on wymagany.

ŚRODOWISKO
        LOGOWANIE      Do pomyślnego uruchomienia skryptu wymagane są
                       prawa administratora na głównym YP

BŁĘDY
  Tak. Z pewnością.

AUTOR
  Brian Jones jonesy@linuxlaboratory.org
```

Nagłówki pisane wersalikami są znane każdemu, kto czytał strony podręcznika systemowego. Zapisałem ten plik pod nazwą *cleangroup.txt*, a następnie uruchomiłem polecenie tworzące stronę podręcznikową o nazwie *cleangroup.man*:

```
$ txt2man -t cleangroup -s 8 cleangroup.txt > cleangroup.man
```

Gdy otworzymy tę stronę podręcznikową za pomocą polecenia `man`, w lewym i prawym rogu w górnej części pojawi się tytuł i nazwa sekcji określone w wierszu poleceń odpowiednio przez opcje `-t` i `-s`. Oto końcowy rezultat:

```
cleangroup(8)                    cleangroup(8)

NAZWA
       cleangroup - usuwa użytkowników z grup, jeśli konto nie istnieje
SKŁADNIA
       /usr/local/adm/bin/cleangroup plik_grupy
```

```
OPIS
       cleangroup to skrypt Perla sprawdzający każdy UID w pliku grup
       względem mapy haseł YP. Jeżeli użytkownik tam nie istnieje, zostaje
       usunięty z grupy.

       Jedynym argumentem pliku jest plik grup, jest on wymagany.

ŚRODOWISKO
       LOGOWANIE
                Do pomyślnego uruchomienia skryptu wymagane są prawa
                administratora na głównym YP
BŁĘDY
       Tak. Z pewnością.

AUTOR
        Brian Jones jonesy@cs.princeton.edu
```

Jeśli ktoś nie wie, dlaczego wybrałem sekcję 8. stron podręcznikowych, należy tu wyjaśnić, że sekcji tych nie ustala się całkowicie dowolnie. Dla różnych klas poleceń przeznaczone są różne sekcje. Oto szybki ich przegląd:

1. Polecenia użytkownika, takie jak `ls` i `man`.
2. Wywołania systemowe, takie jak `gethostname` i `setgid`.
3. Wywołania bibliotek, takie jak `isupper` i `getchar`.
4. Pliki specjalne, takie jak *fd* i *fifo*.
5. Pliki konfiguracyjne, takie jak *ldap.conf* i *nsswitch.conf*.
6. Gry i wersje demonstracyjne.
7. Różne.
8. Polecenia wykonywane normalnie przez użytkownika root, takie jak `MAKEDEV` i `pvscan`.

W niektórych systemach jest też sekcja 9., zawierającą dokumentację jądra. Jeśli planujemy utworzenie swojej własnej sekcji stron podręcznikowych, spróbujmy wybrać istniejącą sekcję, która nie jest używana, lub umieścić swoje strony podręcznikowe w jednej z istniejących sekcji. Aktualnie polecenie `man` wczytuje tylko katalogi *manX* (gdzie *X* jest pojedynczą cyfrą), więc `man42` nie jest prawidłową sekcją stron podręcznikowych.

Choć wynikowa strona podręcznikowa nie różni się zbytnio od pliku tekstowego, to daje ona korzyść w postaci możliwości odczytania jej za pomocą standardowych narzędzi, i każdy będzie wiedział, o co nam chodzi gdy powiemy „sprawdź cleangroup w man 8". To o wiele prostsze niż „wejdź do intranetu, kliknij *Dokumentację*, przejdź do *Systemy*, potem *Linux-Unix*, następnie *Konta użytkowników* i kliknij znajdujący się tam plik PDF".

Jeśli ktoś sądzi, że *txt2man* może obsłużyć jedynie najprostsze strony podręcznikowe, niech sprawdzi jego wbudowaną pomoc, którą może przesłać do samego siebie; wynikowa strona podręcznikowa stanowi całkiem dobry przykład tego, co *txt2man* może zrobić ze zwykłego tekstu. W celu sprawdzenia uruchamiany polecenie (wzięte wprost ze strony podręcznikowej *txt2man*):

```
$ txt2man -h 2>&1 | txt2man -T
```

Spowoduje ono przesłanie wyjścia pomocy z powrotem do *txt2man*, a dzięki opcji `-T` będziemy mogli przejrzeć wyjście za pomocą polecenia `more` lub innego, które jest ustawione w zmiennej środowiskowej `PAGER`. Opcja ta stanowi również szybki sposób przejrzenia stron podręcznikowych, nad którymi pracujemy, w celu sprawdzenia, czy wszystko zostało poprawnie sformatowane, zamiast utworzenia strony podręcznikowej, otwarcia jej, wyszukania błędów, zamknięcia i ponownego otwarcia w edytorze. Spróbujmy!

## SPOSÓB 39. Wykorzystanie potęgi Vima

Opcje rejestracji i makr klawiaturowych mogą pomóc w realizacji monotonnych zadań.

Każdy administrator w pewnym momencie swojej kariery spotka się z sytuacją, gdy nie jest jasne, czy dane zadanie można wykonać szybciej za pomocą polecenia Vima `.` (kropki) i jednego lub dwóch naciśnięć klawiszy przy każdej zmianie czy też za pomocą skryptu. Często administratorzy nadużywają polecenia `.`, gdyż wydaje im się, że zajmuje to mniej czasu niż wymyślanie pasującego wyrażenia regularnego w skrypcie Perla, *sed* czy *awk*.

Jeżeli jednak wiemy, jak korzystać z opcji rejestracji w Vimie, możemy zastosować makra w locie, przy minimalnej ilości uderzeń klawiszy. Co więcej, jeśli mamy zadania, które stale musimy przeprowadzać w Vimie, możemy utworzyć dla nich makra klawiaturowe dostępne przy każdym otwarciu naszego edytora. Przyjrzyjmy się temu!

### Rejestracja makr w Vimie

Najlepiej będzie wyjaśnić to na przykładzie. Mam plik, który powstał w wyniku zrzucenia wszystkich danych z mojego katalogu LDAP. Składa się on w wpisów LDIF wszystkich użytkowników w moim środowisku.

Jeden wpis wygląda tak:

```
dn: cn=jonesy,ou=People,dc=linuxlaboratory,dc=org
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: evolutionPerson
uid: jonesy
sn: Jones
cn: Brian K. Jones
userPassword: {crypt}eRnFAci.Ie2Ny
loginShell: /bin/bash
uidNumber: 3025
gidNumber: 410
homeDirectory: /u/jonesy
gecos: Brian K. Jones,STAFF
mail: jonesy@linuxlaboratory.org
roomNumber: 213
fileas: Jones, Brian K.
telephoneNumber: NONE
labeledURI: http://www.linuxlaboratory.org
```

```
businessRole: NONE
description: NONE
homePostalAddress: NONE
birthDate: 20030101
givenName: Brian
displayName: Brian K. Jones
homePhone: 000-000-0000
st: NJ
l: Princeton
c: UStitle: NONE
o: Linuxlaboratory.orgou: Systems Group
```

W pliku tym jest mniej więcej 1000 wpisów. Muszę na końcu każdego wiersza `labeledURI` dodać wartość *~nazwa_użytkownika*. Będzie to odpowiadało zmianom wprowadzonym w naszym środowisku, gdzie każdemu użytkownikowi przydzielona została przestrzeń WWW dostępna w jej katalogu macierzystym o adresie *http://www.linuxlibrary.org/~nazwa_użytkownika*. Niektóre wpisy składają się z większej liczby wierszy niż inne, więc nie ma tu zbyt dużo konsekwencji i przewidywalności, które ułatwiłyby mi pracę. Pewnie można by tu napisać jakiś skrypt powłoki lub Perla, który by się tym zajął, ale w rzeczywistości nie trzeba wcale opuszczać Vima, żeby to wykonać. Najpierw zarejestrujemy makro. Pierwszym krokiem będzie wpisanie (w trybie poleceń) `q`*`n`*, gdzie *`n`* jest etykietą rejestru. Prawidłowe etykiety rejestrów to wartości 0-9 i a-z. Gdy to zrobimy, rozpocznie się rejestracja, a Vim przechowa w rejestrze każde naciśnięcie klawisza, więc piszmy ostrożnie! Ponowne wpisanie `q` zatrzyma proces rejestracji.

Poniżej przedstawiam moje naciśnięcia klawiszy, łącznie z rozpoczęciem i zatrzymaniem rejestracji:

```
qz
/uid:<Enter>
ww
yw
/labeledURI<Enter>
A
/~
<Esc>
p
q
```

Pierwszy wiersz rozpoczyna rejestrację i wskazuje, że moje naciśnięcia klawiszy zostaną przechowane w rejestrze `z`. Następnie szukam ciągu `uid:` (`/uid:`), przesuwam się dwa słowa na prawo (`ww`) i pobieram (kopiuję do bufora Vima) to słowo (`yw`). Teraz mam już nazwę użytkownika, która ma zostać wklejona na koniec adresu URL znajdującego się już w pliku. Szukam teraz atrybutu `labeledURI` (`/labeledURI`), wskazuję zamiar dopisania czegoś na końcu wiersza (`A`), wpisuję `/~` (ponieważ znaki te muszą się tam znaleźć, a nie są one częścią identyfikatora użytkownika), a następnie naciskam klawisz *Esc* w celu przejścia w tryb poleceń i natychmiast naciskam klawisz `p`, aby wkleić skopiowaną nazwę użytkownika. Na koniec naciskam klawisz `q` w celu zatrzymania rejestracji.

Teraz w rejestrze `z` przechowywany jest ciąg zawierający zapisane klawisze, który możemy obejrzeć, wpisując następujące polecenie:

```
:register z
"z /uid: ^Mwwyw/labeledURI: ^MA/~^[p
```

Są tu także znaki sterujące (`^M` oznacza *Enter*, a `^[` to *Escape*), więc, jak widać, jest tu wszystko, co wpisałem. Teraz w dowolnym momencie mogę wywołać ten ciąg naciśnięć klawiszy, wpisując (w trybie poleceń) `@z`. W pliku, na którym pracuję, jest 935 wpisów (taki wynik otrzymałem za pomocą polecenia `wc -l`), z których jeden zmieniłem już wcześniej, więc umieszczam kursor poniżej tego zmienionego wiersza i wpisuję `934@z`, co spowoduje dokonanie niezbędnych zmian w każdym wpisie tego pliku. Niestety, nie znalazłem innego sposobu na uruchomienie makra do końca pliku bez określania liczby.

## Tworzenie skrótów klawiaturowych

Tak się składa, że naprawdę bliska jest mi idea edytorów HTML działających w trybie WYSIWYG. Podoba mi się to, że nie trzeba się martwić o składnię znaczników. W tym zakresie edytory takie zapewniają porządną warstwę abstrakcji, umożliwiając mi skoncentrowanie się bardziej na treści niż na formie. Wspaniałe jest to, że korzystając z nich, nie trzeba pamiętać znaczników dla takich elementów jak znaki większości i mniejszości czy spacje nierozdzielające.

Niestety, żadne z tych wspaniałych narzędzi nie pozwala mi korzystać ze skrótów klawiaturowych Vima do poruszania się w obrębie pliku. Nie proszę nawet o funkcję szukania i zastępowania czy inne funkcje związane z rejestrami oferowane przez Vima — chodzi jedynie o prostą możliwość poruszania się za pomocą klawiszy `h`, `j`, `k` oraz `l` i może klika innych udogodnień. Minęło sporo czasu, zanim zrozumiałem, że nie muszę już iść na kompromis! Mogę mieć pełną moc Vima i korzystać z niej w środowisku, gdzie formatowanie, choć nie jest całkiem niewidoczne, nie wymaga żadnej aktywności umysłowej.

Oto doskonały przykład pewnego sposobu, w jaki codziennie używam skrótów klawiaturowych Vima. Mam do napisania dokumentację mojej pracy w postaci HTML. Zawsze, gdy jakiś mój dokument zawiera polecenie do uruchomienia, obejmuję je znacznikami `<code></code>`. Zdarza się to dość często, gdyż dokumentacja, którą tworzę w pracy, przeznaczona jest dla administratorów systemu. Innymi dwoma często używanymi przeze mnie znacznikami są znaczniki akapitu `<p></p>` oraz znaczniki `<h2></h2>`, którymi oznaczam poszczególne fragmenty dokumentacji. Oto wiersz, który wpisałem do mojego pliku *~/.vimrc*, dzięki czemu wpisanie znaczników kodu jest tak łatwe jak naciśniecie klawisza *F12* na klawiaturze:

```
imap <F12> <code> </code> <Esc>2F>a
```

Słowo kluczowe `imap` oznacza, że odwzorowanie to będzie aktywne tylko w trybie wstawiania. Zrobiłem to celowo, gdyż zawsze jestem już w trybie wstawiania, kiedy potrzebuję tych znaczników. Następnie podany jest klawisz do odwzorowania, którym w tym przypadku jest *F12*. Po nim występują właściwe znaczniki, które zostaną wstawione. Gdybym na tym zakończył, naciśnięcie klawisza *F12* w trybie wstawiania spowodowałoby wstawienie moich znaczników z kursorem pozostawionym po ich prawej stronie. Ponieważ jestem zbyt leniwy, aby samemu przesuwać kursor między znaczniki,

dlatego na końcu tego odwzorowania umieściłem kolejne naciśnięcia klawiszy. Najpierw przechodzę do trybu poleceń za pomocą klawisza *Esc*. Fragment `2F>` każe szukać (wstecz od miejsca położenia kursora) drugiego wystąpienia znaku >, a następnie `a` umieszcza kursor (z powrotem w trybie wstawiania) po znaku >. Nawet nie zdaję sobie sprawy z tego, że opuszczam tryb wstawiania — wszystko przebiega zupełnie gładko!

SPOSÓB 40.

## Jak przenieść swoje umiejętności pisania skryptów PHP do wiersza poleceń

PHP jest tak proste, że programistą WWW mógłby zostać nawet trzylatek. Teraz możemy przenieść te umiejętności do wiersza poleceń!

Obecnie nie jest łatwo znaleźć osobę utrzymującą się z pracy związanej w dowolny sposób z komputerami, która nie miałaby nigdy kontaktu z PHP. Programowanie stron WWW w PHP jest łatwiejsze od pisania skryptów CGI w Perlu choćby dlatego, że do uruchomienia skryptów PHP nie trzeba ich kompilować. Uzależniłem się od PHP od samego początku, ale nie zajmuję się już wiele programowaniem WWW. Odkryłem jednak, że PHP jest bardzo przydatnym narzędziem w tworzeniu skryptów wiersza poleceń, a nawet poleceń mieszczących się w pojedynczym wierszu.

Wchodząc na witrynę PHP.net zawierającą listę funkcji (*http://www.php.net/manual/pl/funcref.php*), możemy sprawdzić, co PHP ma nam do zaoferowania. Jak szybko się przekonamy, wiele cech PHP doskonale nadaje się do programowania wiersza poleceń. PHP posiada wbudowane funkcje do współdziałania z *syslog*, tworzenia demonów oraz wykorzystania strumieni i gniazd. Posiada nawet zestaw funkcji POSIX, takich jak `getwuid` i `getpid`.

W niniejszym sposobie będę korzystał z PHP5 dostarczanego z dystrybucją Fedora Core 4. PHP jest łatwo dostępne w formatach binarnych dla systemów SUSE, Debian, Red Hat, Fedora, Mandrake i innych popularnych dystrybucji. Niektóre dystrybucje nie przeszły jeszcze na PHP5, ale pewnie stanie się tak prędzej czy później.

Oczywiście kod stosowany w tym sposobie będzie miał dla nas ograniczone zastosowanie. Celem jest tu poszerzenie horyzontów przez wykorzystanie posiadanych już umiejętności programowania w PHP i wykorzystanie ich do czegoś tak niekonwencjonalnego jak administracja systemem.

### Kod

Rzućmy okiem na kod. Pierwszy skrypt jest naprawdę prosty; to uproszczona wersja skryptu, z którego korzystam w celu uniknięcia konieczności stosowania standardowego narzędzia *ldapsearch* z całym mnóstwem opcji. Na przykład, jeśli chcemy na określonym serwerze znajdującym się w innym dziale wyszukać użytkowników o nazwisku Jones i uzyskać w wyniku nazwę wyróżnioną (`dn`) każdego z tych użytkowników, musimy wpisać:

```
$ ldapsearch -x -h  ldap.linuxlaboratory.org  -b"dc= linuxlaboratory,dc=org "
'(sn=Jones)' dn
```

Okropność. Jeszcze gorzej, gdy musimy często wpisywać takie wyszukiwania. Można by napisać skrypt powłoki, ale odkryłem, że PHP jest w stanie doskonale poradzić sobie z tym zadaniem bez jakiejkolwiek potrzeby używania narzędzia *ldapsearch* w systemie. W dodatku uniwersalność PHP jest dużym plusem — każda osoba w moim zespole zetknęła się z PHP, ale niektórzy wolą pisać w *tcsh*, który na tyle różni się od *ksh* czy *bash*, że trudno się w nim połapać. Pamiętajmy, że napisany dziś kod może stać się problemem dla kogoś innego, jeśli pojawi się jakiś katastrofalny błąd w czasie, gdy my będziemy płynąć na statku, popijając margaritę, z dala od przekaźników sieci telefonii komórkowej. Oto mój skrypt, który nazwałem *dapsearch*:

```
#!/usr/bin/php
<?php

$conn=ldap_connect("ldap.linuxlaboratory.org")
 or die("Connect failed\n");

$bind = ldap_bind($conn)
 or die("Bind failed\n");

$answer = ldap_search($conn, "dc=linuxlaboratory,dc=org", "($argv[1])");
$output = ldap_get_entries($conn, $answer);

for ($i=0; $i < count($output); $i++) {
        if(!isset($output[$i])) break;
        echo $output[$i]["dn"]."\n";
}
echo $output["count"]." entries returned\n";
?>
```

W powyższym kodzie warto zwrócić uwagę na kilka rzeczy. Pierwszy wiersz zawiera ścieżkę dostępu do pliku binarnego, który uruchamia kod, poprzedzoną standardowym symbolem `#!`, podobnie jak w przypadku dowolnego innego skryptu powłoki lub Perla. Jeżeli wprowadzamy kod na komputerze biurkowym z zamiarem późniejszego umieszczenia go w komputerze, który nie znajduje się pod naszą kontrolą, możemy zastąpić ten wiersz następującym:

```
#!/usr/bin/env php
```

W ten sposób nie przyjmujemy żadnych założeń co do położenia pliku binarnego PHP w konkretnym katalogu, przeszukując zamiast tego zmienną `PATH`, co może być bardziej wiarygodne.

Oprócz tego możemy zauważyć, że w skrypcie występują znaczniki `<?php` i `?>`, podobnie jak w przypadku skryptów WWW. Może to być przydatne, gdy mamy statyczny tekst , który chcemy wprowadzić na ekran, ponieważ zamiast stosować instrukcję `echo`, wystarczy umieścić taki tekst poza znacznikami. Po prostu zamykamy znacznik, wpisujemy nasz tekst, a następnie otwieramy nowy zestaw znaczników. Analizator składniowy wypisze wówczas nasz tekst, a następnie rozpocznie analizę kodu PHP przy ponownym otwarciu znaczników.

Jak widać, uprościłem nieco pewne rzeczy, wprowadzając na stałe atrybut, jaki ma zostać zwrócony (atrybut `dn`), jak również serwer, z którym się łączę. Skrypt ten można łatwo zmienić, umożliwiając podanie tych informacji w wierszu poleceń. Wszystko, co podamy w wierszu poleceń, znajdzie się w tablicy `argv`.

## Uruchomienie kodu

Po zapisaniu powyższego skryptu należy nadać mu atrybut wykonywalności, a następnie uruchomić, podając szukany atrybut. We wcześniejszym poleceniu *ldapsearch* szukałem atrybutów nazw wyróżnionych wszystkich użytkowników o nazwisku „Jones". Oto (znacznie skrócone) polecenie, jakie wprowadzam teraz w celu uzyskania tych informacji:

```
$ dapsearch sn= Jones
```

Polecenie to wywołuje skrypt i przesyła do niego filtr wyszukiwania, który występuje w kodzie jako `$argv[1]`. Może to wyglądać dziwnie dla osób programujących w Perlu, gdzie pojedynczy argument oznacza się przez `@_`, `$_` albo `$argv[0]`. W PHP `$argv[0]` zwraca nazwę uruchomionego polecenia, zamiast pierwszego argumentu przekazywanego w wierszu poleceń.

Skoro mówimy już o tablicy `argv`: mogą wystąpić błędy, jeśli użyjemy tej opcji w instalacji PHP, która nie posiada domyślnie włączonych tablic `argv` i `argc`. W takim przypadku należy dokonać prostej zmiany — otwieramy nasz plik *php.ini* (plik konfiguracyjny samego analizatora składni PHP) i zmieniamy wartość `register_argc_argv` na `on`.

SPOSÓB 41.

## Szybkie połączenia przez telnet (SSH) z pulpitu

Skrót pulpitu i prosty skrypt powłoki to świetny zestaw do szybkiego połączenia przez telnet (SSH) ze zdalnym systemem.

Wielu z nas pracuje z dużą liczbą serwerów i zmuszonych jest często się na nie logować i wylogowywać. Za pomocą apletów uruchamiania aplikacji KDE lub GNOME i prostych skryptów powłoki możemy utworzyć skróty pulpitu umożliwiające szybkie połączenie z dowolnym komputerem za pomocą różnych protokołów.

W tym celu tworzymy skrypt o nazwie *connect*, nadajemy mu atrybut wykonywalności i umieszczamy w katalogu znajdującym się w naszej domyślnej ścieżce dostępu. Skrypt ten powinien wyglądać następująco:

```
#!/bin/bash

progname='basename $0'

type="single"

if [ "$progname" = "connect" ] ; then
        proto=$1
        fqdn=$2
```

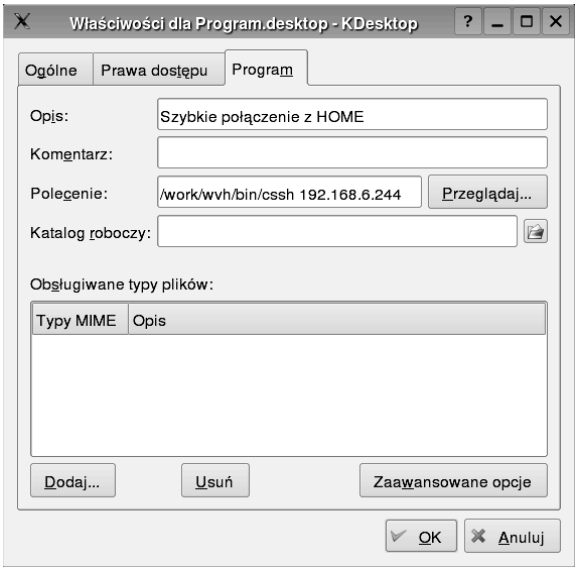
```
        shift
        shift
elif [ "$progname" = "ctelnet" ]; then
        proto="telnet"
        fqdn=$1
        shift
elif [ "$progname" = "cssh" ]; then
        proto="ssh"
        fqdn=$1
        shift
elif [ "$progname" = "mtelnet" ]; then
        proto="telnet"
        fqdn=$1
        hosts=$*
        type="multi"
elif [ "$progname" = "mssh" ]; then
        proto="ssh"
        fqdn=$1
        hosts=$*
        type="multi"
fi
args=$*

#
# Usuń znaki komentarza przed poleceniem xterm i oznacz jako komentarz poniższą klauzulę
# warunkową if/else/if, jeśli chcesz korzystać gdzieś z xterma
#
# xterm +mb -sb -si -T "${proto}::${fqdn}" -n ${host} -bg black -fg yellow -e ${proto} ${fqdn} ${args}
#
# Zmienia z Konsole na gnome-console i określa poprawne opcje, jeśli KDE nie zostało
# zainstalowane
#
if [ "$type" != "multi" ]; then
        konsole -T "${proto}::${fqdn}" --nomenubar --notoolbar ${extraargs} -e
${proto} ${fqdn} ${args}
else
        multixterm -xc "$proto %n" $hosts
fi
```

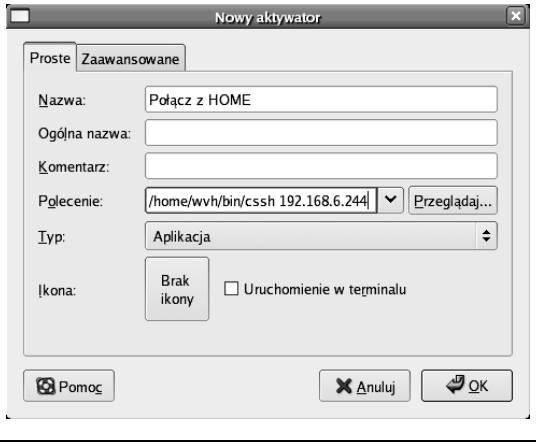
Po utworzeniu skryptu i uczynieniu go wykonywalnym należy utworzyć do niego w tym samym katalogu dowiązania symboliczne o nazwach *cssh*, *ctelnet*, *mssh* i *mtelnet*. Jak widać, używany protokół i polecenia oparte są na sposobie wywołania skryptu.

Aby użyć tego skryptu w KDE, należy kliknąć prawym przyciskiem myszy na pulpicie i wybrać *Utwórz nowe>Skrót do programu*. Pojawi się okno dialogowe widoczne na rysunku 4.2. Wpisujemy tu nazwę skryptu, który chcemy wykonać, oraz adres komputera, z którym chcemy się połączyć, i klikamy *OK*.

Aby użyć tego skryptu w GNOME, należy kliknąć prawym przyciskiem myszy na pulpicie i wybrać *Utwórz aktywator*. Pojawi się okno dialogowe widoczne na rysunku 4.3. Wpisujemy tu nazwę skryptu, który chcemy wykonać, oraz adres komputera, z którym chcemy się połączyć, i klikamy *OK*.




*Rysunek 4.2. Tworzenie skrótu do programu w KDE*




*Rysunek 4.3. Tworzenie skrótu do programu w GNOME*

Za pomocą dowolnej z tych metod sprawnie utworzymy skrót na pulpicie, dzięki któremu będziemy mogli szybko nawiązać połączenie ze zdalnym systemem przez kliknięcie skrótu na pulpicie.

## Zobacz również

- „Jednoczesne wykonanie poleceń na wielu serwerach” **[sposób 29]**

*— Lance Tost*

# SPOSÓB 42. Szybsza kompilacja

Podczas kompilacji za pomocą demona kompilacji skrośnej można wykorzystać moc wszystkich komputerów.

Wielu użytkowników innych dystrybucji naśmiewa się z wielbicieli Gentoo, ponieważ muszą oni spędzać mnóstwo czasu na kompilacji wszystkich swoich programów. I choć te wszystkie kompilacje zajmują godziny i dni, to użytkownicy Gentoo wciąż zachwalają swoją dystrybucję jako najszybszą ze wszystkich. Z powodu stałej konieczności kompilowania użytkownicy Gentoo nauczyli się korzystać z kilku sztuczek mających na celu przyspieszenie tego procesu, łącznie z użyciem *distcc* do utworzenia klastra komputerów do kompilacji. *distcc* jest demonem kompilacji skrośnej umożliwiającym połączenie mocy obliczeniowej innych komputerów w sieci z zainstalowanym systemem Linux w celu kompilacji kodu. Jest on bardzo łatwy w konfiguracji oraz użytkowaniu i powinien dać identyczny rezultat jak kompilacja całkowicie lokalna. Mając trzy komputery o podobnej szybkości, powinniśmy być wstanie skompilować 2,6 razy szybciej. Na stronie domowej *distcc* (*http://distcc.samba.org*) zamieszczono wyrazy uznania użytkowników korzystających z tego programu. Używając tego sposobu, możemy sprawić, aby *distcc* działał z każdą dystrybucją Linuksa, dzięki czemu kompilacja całego KDE czy GNOME przebiegnie szybko i łatwo.

*distcc* nie wymaga od komputerów w naszym zespole kompilacyjnym posiadania współdzielonych systemów plików, zsynchronizowania zegarów, czy nawet tych samych bibliotek i plików nagłówkowych. Warto się jednak upewnić, że jest na nich zainstalowana ta sama główna wersja samego kompilatora.

Przed rozpoczęciem pracy z *distcc* musimy najpierw wiedzieć, jak wykonać równolegle polecenie `make` w trakcie budowania kodu. Służy do tego opcja `-j` polecenia `make`:

```
dbrick@rivendell:$ make j3; make j3 modules
```

Spowoduje to utworzenie trzech procesów potomnych, które maksymalnie wykorzystają naszą moc obliczeniową i zapewnią, że zawsze będzie w kolejce coś do skompilowania. Ogólna zasada przy szacowaniu ilości równoległych procesów *make* polega na podwojeniu liczby procesorów i zwiększeniu tej wartości o jeden. Tak więc w systemie jednoprocesorowym podajemy opcję `-j3`, a w systemach dwuprocesorowych `-j5`. Gdy zaczniemy używać *distcc*, za wartość bazową `-j` powinniśmy przyjąć całkowitą liczbę procesorów w naszym zespole kompilacyjnym. Jeśli mamy dostępnych osiem procesorów, wówczas użyjemy `-j17`.

## Korzystanie z distcc

Najnowszą wersję *distcc* można pobrać ze strony głównej programu: *http://distcc.samba.org/download.html*. Pobieramy archiwum, rozpakowujemy je i wydajemy standardowe polecenia kompilujące:

```
dbrick@rivendell:$ tar -jxvf distcc-2.18.3.tar.bz2
dbrick@rivendell:$ cd distcc-2.18.3
dbrick@rivendell:$ ./configure && make && sudo make install
```

Program należy zainstalować na każdym komputerze, który chcemy włączyć do zespołu kompilacyjnego. Na każdym z tych komputerów należy uruchomić demona *distcc*:

```
root@bree:# distccd --daemon --N15
root@moria:# distccd --daemon --N15
```

Demony te będą nasłuchiwać na porcie TCP 3632 w oczekiwaniu na instrukcje i kod z komputera lokalnego (tego, dla którego przeprowadzamy właściwą kompilację). Wartość `-N` określa poziom uprzejmości, aby kompilacja skrośna nie zakłócała operacji lokalnych. Dalsze opcje znajdziemy na stronie podręcznikowej programu *distcc*.

Po stronie klienta musimy powiadomić *distcc*, z których komputerów będziemy korzystać w kompilacji skrośnej. Możemy to zrobić przez utworzenie zmiennej środowiskowej:

```
dbrick@rivendell:$ export DISTCC_HOSTS='localhost bree moria'
```

Podajemy tu `localhost`, aby włączyć nasz lokalny komputer do procesu kompilacji. Jeśli nasz komputer jest wyjątkowo wolny lub dysponujemy dużą liczbą procesorów do rozdzielenia kompilacji, możemy rozważyć pominięcie komputera lokalnego. Zamiast nazw możemy używać adresów IP komputerów. Jeśli nie chcemy ustawiać zmiennej środowiskowej, powinniśmy utworzyć w naszym katalogu macierzystym plik *distcc* przechowujący te wartości:

```
dbrick@rivendell:$ mkdir ~/.distcc
dbrick@rivendell:$ echo "localhost  bree moria" > ~/.distcc/hosts
```

Aby uruchomić kompilację skrośną, wystarczy w poleceniu `make` podać opcję `CC=dist`:

```
dbrick@rivendell:$ make j7 CC=distcc
```

Jak widać, kompilacja skrośna na wielu komputerach jest bardzo prosta. Więcej informacji na temat programu znajduje się na stronach podręcznikowych dla *distcc* i *dictccd*, łącznie z tym, jak przeprowadzić ograniczenie liczby równoległych procesów *make* na konkretnym komputerze w zespole.

## Kompilacja skrośna na komputerach z systemem Windows

Choć pewne zdolne osoby odkryły bardzo interesujący sposób kompilacji skrośnej na komputerach z zainstalowanym systemem Windows przy użyciu pakietu Cygwin, to istnieje łatwiejszy sposób wykonania tego samego zadania za pomocą dystrybucji live CD o nazwie distccKnoppix, którą można pobrać ze strony *http://ftp.caliu.info/pub/distribucions/knoppix-remaster/distccknoppix/*. Upewnijmy się, że pobraliśmy wersję, która ma ten sam główny numer wersji `gcc` co nasz komputer lokalny.

Aby użyć systemu distccKnoppix, wystarczy uruchomić komputer z płyty CD, zapisać jego adres IP, a następnie wpisać go do swojego pliku *distcc* lub do zmiennej środowiskowej zgodnie z podanymi wyżej instrukcjami. Miłego kompilowania!

*— David Brickner*

# SPOSÓB 43. Unikanie pospolitych pomyłek

Jak zakończyć karierę początkującego administratora i zostać prawdziwym guru?

Bez względu na to, jak jesteśmy doświadczeni i jak wszechpotężni czujemy się w naszej roli, i tak w końcu zdarzą się nam pomyłki. Niektóre z nich mogą być dość poważne. Przez niektóre może nam „wypaść" z kalendarza cały weekend. Klucz do sukcesu w administrowaniu serwerami to minimalizacja ryzyka, posiadanie planu awaryjnego oraz wysiłki zmierzające do ograniczenia zniszczeń, jakie mogłyby zostać wywołane przez ewentualne pomyłki. Przedstawimy tu niektóre pospolite pomyłki, których należy unikać na drodze do uzyskania statusu prawdziwego guru.

## Nie wzywaj imienia roota nadaremno

Spróbujmy zapomnieć o koncie root. Oto krótkie porównanie korzystania z konta root przez wytrawnego weterana i początkującego administratora.

Solidni, doświadczeni administratorzy czasami zapominają, że muszą mieć uprawnienia użytkownika root do wykonania niektórych zadań. Oczywiście zdadzą sobie z tego sprawę, gdy tylko w terminalu pojawi się komunikat informujący o błędzie, ale konieczność wydania polecenia `su - root` czasem wylatuje im z głowy. Nic wielkiego. Wystarczy, że przełączą się na konto root, wydadzą polecenie i wyjdą z powłoki. Jeśli muszą wydać jedynie pojedyncze polecenie, np. `make install`, prawdopodobnie wpiszą coś takiego:

```
$ su -c 'make install'
```

Spowoduje to zapytanie o hasło roota i — jeśli będzie ono poprawne — uruchomienie polecenia i powrót do powłoki zwykłego użytkownika.

Z drugiej strony, początkujący administrator ma pewnie z pięć otwartych terminali na jednym komputerze, we wszystkich będąc zalogowanym jako root. Początkujący administratorzy nie biorą pod uwagę korzystania z terminali, na których nie byliby zalogowani jako root, bo przecież „musisz mieć uprawnienia roota, żeby cokolwiek zrobić". To wyjątkowo nieodpowiednie zachowanie, które może prowadzić do naprawdę opłakanych skutków. Pamiętajmy, aby nie wchodzić na konto roota bez potrzeby!

Budowanie oprogramowania jest tu dobrym przykładem. Po pobraniu pakietu źródłowego rozpakowujemy go w miejscu, do którego mamy (jako użytkownik) dostęp. Następnie, jako zwykły użytkownik, wydajemy polecenia `./configure` i `make`. Jeśli instalujemy pakiet do swojego katalogu *~/bin*, możemy uruchomić `make install` z własnego konta. Uprawnienia roota potrzebne są tylko podczas instalacji do katalogów, w których tylko on ma prawa do zapisu, np. */usr/local*.

Prawie padłem z wrażenia, kiedy pewnego dnia poznałem całkiem nowe znaczenie „wzywania imienia roota nadaremno". Nie chodzi tu tylko o niepotrzebne uruchamianie polecenia jako root. Dotyczy to również przejścia na konto roota w celu przyznania nieuprzywilejowanego dostępu do rzeczy, do których dostęp powinien mieć tylko root!

Byłem zalogowany na komputerze klienta (oczywiście jako zwykły użytkownik), myszkując tu i tam, ponieważ użytkownik zgłosił pojawianie się dziwnych komunikatów logowania. Jednym z moich ulubionych poleceń przy śledzeniu takich spraw jest `ls -lahrt /etc`, które zwraca długą listę całej zawartości katalogu posortowaną w odwrotnej kolejności według czasu modyfikacji. W tym przypadku ostatnim elementem na liście (a tym samym ostatnio modyfikowanym) było `/etc/shadow`. Nie byłoby w tym nic dziwnego, gdyby ktoś dodał ostatnio jakiegoś użytkownika na lokalnym komputerze, ale tak się złożyło, że w tym przedsiębiorstwie używano NIS+, a uprawnienia zostały zmienione na pliku!

Zadzwoniłem pod numer, jaki mi podano w razie, gdybym coś znalazł. Początkujący administrator przyznał, że sam to zrobił, ponieważ pisał skrypt, który potrzebował dostępu do tego pliku. Ehh…

## Nie bądź zbyt wygodny

Początkujący administratorzy często lubią dostosowywać swoje środowisko. Lubią popisywać się wszystkim, czego się ostatnio nauczyli, więc stosują własne ustawienia menedżera okien, własne ustawienia logowania, własną konfigurację poczty, własne skrypty tunelowania w celu wykonywania pracy z komputerów domowych i oczywiście własne powłoki i własną inicjalizację powłoki.

To ostatnie może przyprawić o ból głowy. Jeśli mamy na swoim komputerze ustawionych milion aliasów i jakiś inny zestaw komputerów, który montuje nasz katalog macierzysty (tym samym zapewniając dostęp do inicjalizacji powłoki), wszystko będzie działać bez zarzutu dla tego zestawu komputerów. Jednak bardziej prawdopodobny jest wariant, gdzie mamy do czynienia ze środowiskiem mieszanym, w którym obok systemów Linux występują również inne odmiany Uniksa posiadające standardowe aliasy i ogólnosystemowe profile powłoki, które były tam na długo przed nami.

Jeśli zmodyfikujemy powłokę, będziemy musieli przetestować, czy wszystko, co zrobiliśmy, działa tak, jak oczekiwaliśmy na wszystkich administrowanych przez nas platformach systemowych. Lepiej więc po prostu pozostawić powłokę administracyjną stosunkowo niezmienioną. Oczywiście trzeba ustawić prawidłowe zmienne środowiskowe i utworzyć trzy czy cztery aliasy; można też (jeśli chcemy) utworzyć własny znak zachęty, ale uważajmy, aby nie poniosło nas do tego stopnia, by zaopatrzyć się we wszelkiego rodzaju polecenia uzupełniające powłoki *bash*, wyświetlać obciążenie systemu w oknie terminala czy stosować funkcje powłoki do utworzenia własnej zachęty powłoki. Dlaczego?

No cóż, nie możemy zakładać, że wszędzie będzie działać ta sama wersja powłoki lub że powłoka została zbudowana z tymi samymi opcjami we wszystkich wersjach na wielu platformach! Ponadto nie zawsze będziemy się logować z własnego komputera. Pomyślmy, co może się zdarzyć, jeśli pomyłkowo skonfigurujemy swój plik inicjacyjny do wyprowadzania różnych komunikatów na pasku tytułowym naszego terminala bez sprawdzenia, skąd się łączymy. Gdy pierwszy raz zalogujemy się z terminala nieinteligentnego (ang. *dumb terminal*), zdamy sobie sprawę, że nie był to dobry pomysł. Może to skończyć się tym, że nasz znak zachęty stanie się tak długi, że nie zmieści się na ekranie!

Podobnie jak wersje i opcje budowania naszej powłoki mogą różnić się w zależności od komputera, taka sama różnica może wystąpić w przypadku „standardowych" poleceń — i to znaczna! Na przykład uruchomienie polecenia `chown -R` daje zupełnie odmienny wynik w systemie Solaris niż w Linuksie. Solaris podąży za łączem symbolicznym i utrzyma śledzenie, pomijając hierarchię katalogów i rekursywnie zmieniając prawa własności plików w miejscach, o których istnieniu zapomnieliśmy. W Linuksie to się nie zdarzy. Aby Linux zachował się w ten sam sposób, trzeba podać mu jawnie opcję `-H`. Istnieje wiele poleceń objawiających różne zachowanie w rozmaitych systemach operacyjnych, więc bądźmy czujni!

Należy także przetestować własne skrypty powłoki na rożnych platformach, aby się upewnić, że polecenia wywoływane z poziomu skryptów zachowują się zgodnie z naszymi oczekiwaniami w każdym środowisku, w jakim mogą zostać uruchomione.

## Nie wydawaj poleceń bez zastanowienia

W wielu środowiskach istnieją ścisłe zasady dotyczące instalacji oprogramowania, sposobu tworzenia nowych systemów i wprowadzania ich do eksploatacji itd. Jednak jest też tysiące miejsc, w których nie wprowadzono takich zasad, co szczerze mówiąc, jest nieco przerażające.

Brak funduszy na wprowadzenie odpowiedniego środowiska testowego to jedna sprawa, czym innym jest natomiast rażące lekceważenie możliwości usług produkcyjnych. Przeprowadzając instalacje oprogramowania, zmiany konfiguracyjne, masową migracje danych itp., wyświadczamy sobie dużą przysługę (a właściwie kilka):

*Zapisujmy procedurę w formie skryptu!*
: Procedurę należy zapisać w postaci skryptu, umieszczając w nim odpowiednie działania sprawdzające, aby się upewnić, że wszystko uruchomi się poprawnie bez przyjmowania jakichkolwiek założeń. Należy sprawdzić poprawne wykonanie każdego kroku przez wykonaniem następnego.

*Napiszmy skrypt procedury powrotnej.*
: Jeśli przeniesiemy wszystkie dane, zmienimy konfigurację, dodamy użytkownika uprawnionego do uruchomienia programu i zainstalujemy aplikację, a coś pójdzie nie tak, wówczas nie chcielibyśmy chyba spędzić kolejnych 40 minut na przywracaniu wszystkiego do stanu wyjściowego. Jeśli w dodatku nastąpi to na etapie produkcyjnym, możemy wpaść w panikę, co może spowodować błędy w ocenie, pomyłki przy wpisywaniu i pogorszy tylko sytuację. Piszmy skrypty!

Proces zapisywania procedury do skryptu zmusza nas do myślenia o konsekwencjach naszych działań, co może przynieść zaskakujące rezultaty. Byłem kiedyś w jednej czwartej skryptu, zanim zdałem sobie sprawę z występowania niespełnionej zależności, której nikt nie wziął pod uwagę. Zaoszczędziło mi to sporo czasu i pozwoliło uniknąć ewentualnego doprowadzania wszystkiego do porządku.

## Pytaj

Najlepszą wskazówką, jaką mogę dać każdemu administratorowi, jest sugestia uświadomienia sobie własnej niewiedzy. Nie zakładajmy, że znamy każdy możliwy efekt uboczny wszystkich naszych działań. Pytajmy. Jeśli jakiś doświadczony administrator będzie na nas patrzył jak na idiotę, pozwólmy mu na to. Lepiej być uważanym za idiotę z powodu ciągłego zadawania pytań niż zostać uznanym za idiotę dlatego, że się o coś nie zapytaliśmy!

SPOSÓB 44.

# Jak wprowadzić Linuksa do firmy

Czego nie należy robić przy próbie wprowadzenia Linuksa do serwerowni.

Spójrzmy prawdzie w oczy: nie możemy skorzystać z książki *100 sposobów na na Linux Server*, dopóki nie mamy serwera, na którym można by te sposoby wypróbować. Nauczyłem się na własnych błędach i błędach innych, że ideały uznawane przez społeczność nie mają żadnego znaczenia w firmowej sali posiedzeń; lepiej przedstawić je osobom podejmującym decyzje. Jeśli używamy Linuksa w domu i korci nas wprowadzenie go do swojej serwerowni, warto zapoznać się z najczęściej popełnianymi błędami, jakich należy unikać przy przedstawianiu pewnych kwestii związanych z wprowadzeniem Linuksa do naszego środowiska.

## Nie mów o kosztach

Jeśli zwrócimy się do kierownictwa i napomkniemy o tym, że Linux jest darmowy, prawdopodobnie wyrządzimy sobie niedźwiedzią przysługę — z wielu powodów. Przede wszystkim, jeżeli pokażemy dyrektorowi działu informatycznego witrynę Debiana (zapewne jedynej „całkowicie wolnej w każdy sposób" dystrybucji Linuksa) i poprosimy go o przyjrzenie się jej, gdyż będzie to nowy system operacyjny jego serwera, zapyta nas o odnośnik do pomocy technicznej, Jeśli pokażemy mu forum internetowe, gotów pomyśleć, że urwaliśmy się z choinki.

Kanały IRC, listy dyskusyjne i fora poświęcone Linuksowi zapewniły mi lepsze wsparcie dla każdej technologii, komercyjnej czy nie, niż sami producenci. Jednak bez wydania pieniędzy na pomoc producenta nasz dyrektor przypuszczalnie będzie miał wrażenie, że jego firma nie będzie miała możliwości nacisku na samego producenta i będzie pozbawiona wynikającego z umowy wsparcia technicznego. Nie będzie nikogo, kto ponosiłby odpowiedzialność, żadnego dobrze usposobionego inżyniera oddelegowanego przez producenta do pomocy w migracji i żadnego „gardła do duszenia", gdy coś pójdzie źle.

Szczerze mówiąc, nie można go winić za takie myślenie — on tylko próbuje utrzymać swoją posadę. Pomyślmy, co by się zdarzyło, gdyby doszło do jakiegoś katastrofalnego wydarzenia i zostałby wezwany na spotkanie z wszystkimi grubymi szychami i poproszony o zrelacjonowanie sytuacji, powiedziałby „Napisałem o problemie na forum *http://www.linux.com.pl/forum/*, więc stale tam zaglądam. W międzyczasie wysłałem też e-mail na listę dyskusyjną, o której jeden z informatyków powiedział, że można tam uzyskać pomoc…". Zostałby natychmiast zwolniony!

Działy informatyczne są skłonne wydać pieniądze na oprogramowanie, które wykona swoją pracę. Są też skłonne zapłacić za wsparcie markowego, certyfikowanego producenta. To nie są zmarnowane pieniądze. Biorąc pod uwagę, że platforma systemowa jest tylko częścią znacznie większej technologii, pieniądze wydane na oprogramowanie i wsparcie techniczne stanowią inwestycję zapewniającą pomyślny rozwój. Jeżeli z jakichś powodów koszty ulegną obniżeniu (mniejsza ilość osobogodzin wymagana do utrzymania, zwiększenie wydajności) — to świetnie, ale „darmowe" nie jest niezbędne, oczekiwane, a nawet niekoniecznie dobre.

Nie na tym polega największa siła Linuksa, więc podkreślanie „darmowości" jest niesprawiedliwe wobec osób, które go tworzą i utrzymują. Koszt Linuksa sprawił, że stał się on tym, czym jest dziś, nie mówiąc o obniżeniu bariery dla nowych użytkowników pragnących poznać środowisko uniksowe oraz programistów, którzy rozwinęli technologicznie Linuksa, przenosząc na ten system takie niezawodne aplikacje jak Sendmail i Apache, czyniąc z niego opłacalną platformę, którą firmy chętnie zaadaptowały na pewną niewielką skalę. Podkreślanie argumentu finansowego sugeruje, że jest to najlepsza rzecz w Linuksie, pomijając wszystkie inne jego zalety.

## Nie mów o Linuksie w oderwaniu od kontekstu

Nie ma sensu (w najlepszym razie) mówić o uruchomieniu Linuksa, nie mówiąc o tym w kontekście rozwiązania, które — w porównaniu z obecnym — byłoby bardziej przydatne lub efektywniejsze.

Aby Linux został uznany za opłacalną platformę, musimy od czegoś zacząć. Może to być wprowadzenie nowej technologii lub zastąpienie istniejącej usługi. Aby znaleźć najlepszy sposób na wprowadzenie Linuksa, ważne jest poznanie wszystkich aspektów naszego środowiska. Jeśli wiemy, że kierownictwo jest wysoce niezadowolone z używanego w firmie komunikatora, nie musi to oznaczać, że Jabber będzie dla nich dobrym rozwiązaniem. Marudzenie szefowi, że powinniśmy przejść na Jabbera i wszystko będzie świetnie, zaprowadzi nas donikąd, gdyż nie przedstawiliśmy żadnych faktów na temat Jabbera, które skłoniłyby szefa do uznania, że taki pomysł ma jakąkolwiek wartość. Poza tym postawi nas to w złym świetle, ponieważ przedstawianie tak ogólnych twierdzeń sugeruje, że uważamy się za wiedzących wszystko, co należy wiedzieć na temat przyjętych w firmie rozwiązań dotyczących komunikatorów.

Czy jesteśmy przygotowani na trudne pytania? Czy choć pomyśleliśmy, jakie one mogą być? Czy znamy szczegóły obecnego rozwiązania? Czy wiemy, czego może wymagać przejście na inne rozwiązanie? *Dowolne* inne rozwiązanie? Czy wiemy wystarczająco wiele na temat Jabbera, aby nad nim zapanować, czy też zamierzamy siedzieć przy konsoli z książką na temat Jabbera otwartą na stronie 4., gdy wejdzie nasz szef, aby zobaczyć, jak przebiega nasz wielki i ważny projekt dotyczący wszystkich użytkowników?

„Linux jest lepszy" nie jest wiarygodnym twierdzeniem. „Współdzielenie plików w Linuksie może być lepszym rozwiązaniem na poziomie działów, ponieważ może ono obsłużyć wszystkie występujące u nas platformy systemowe" brzmi lepiej. Ale należy dążyć do

czegoś w rodzaju „Widziałem wdrożenie tej usługi na platformie linuksowej obsługującej 1500 użytkowników na trzech platformach klienckich przy stosunkowo niewielkich nakładach administracyjnych, podczas gdy my obsługujemy 300 klientów na jednej tylko platformie i musimy resetować system dwa razy w tygodniu. W międzyczasie musimy utrzymywać całkowicie oddzielny system świadczący takie same usługi innym platformom klienckim". Pierwsza cześć tego stwierdzenia jest czymś, co można usłyszeć na forum początkujących użytkowników Linuksa. Ostatnia cześć wzbudza zaufanie i dotyka czegoś, co interesuje dyrektorów IT — konsolidacji serwerów.

Gdy rozmawiamy z osobami podejmującymi decyzje na tematy dotyczące zastosowania Linuksa w nowej technologii lub w zastąpieniu usług, ważne jest zrozumienie tego, co jest dla nich wartościowe w aktualnym rozwiązaniu. Jeżeli rozwiązanie polegające na wdrożeniu aktualnie stosowanego komunikatora było spowodowane niskim kosztem licencji grupowej i współpracuje on z istniejącym oprogramowaniem klienckim bez konieczności dokonywania zmian w firewallu i wyborze trasy — odpowiednio się przygotujmy. Czy istniejące oprogramowanie klienckie może się komunikować z serwerem Jabbera? Czy istnieje na miejscu infrastruktura umożliwiająca przesłanie oprogramowania do wszystkich naszych klientów?

Bardzo łatwo powiedzieć, że Linux jest świetny. Znacznie trudniej jest porównać go z istniejącym rozwiązaniem i uzasadnić koszty migracji dyrektorowi, którego głównym zmartwieniem jest odzyskanie pieniędzy, zwrot inwestycji, wydajność pracowników i osobogodziny.

## Nie stosuj Linuksa do rzeczy, do których się nie nadaje

Linux jest przystosowany do wykonywania olbrzymiej ilości różnych zadań, które są obecnie przeprowadzane za pomocą komercyjnych programów o niskiej jakości i wysokich kosztach (zbyt wiele trzeba miejsca i czasu, by to tu szerzej opisać — podpowiedzi znajdziemy w dalszej części książki). Nie ma jednak żadnego powodu, aby używać Linuksa do zadań, których *nie może* on wykonać, gdyż spowoduje to jedynie uczucie niesmaku u osób, których pierwszy kontakt z Linuksem okaże się kompletnym fiaskiem.

Zastosowania Linuksa są całkowicie zależne od miejsca jego wykorzystania. Jeśli mamy duży personel nietechniczny składający się z pracujących w terenie sprzedawców wyposażonych w laptopy, którzy łączą się za pomocą VPN z bezprzewodowych punktów dostępowych rozsianych po całym świecie, i oprócz tego kilka starszych pań odbierających przez cały dzień telefony w biurze, wówczas pulpit naszego komputera może nie być dobrym miejscem dla Linuksa.

W drugiej strony, jeśli mamy telefonistkę na centrali zbudowanej w latach 20. ubiegłego wieku, a siłą napędową naszej firmy jest komunikacja telefoniczna, wówczas oparte na Linuksie rozwiązanie w postaci Asterix PBX może być bardzo przydatne i doceniane!

Chodzi o to, aby wygrać swoją bitwę. Nawet w środowiskach Uniksa możemy napotkać na opory w stosunku do Linuksa, ponieważ niektóre marki Uniksa przez dziesiątki lat wykonywały prace, które niektórzy napaleńcy chcą teraz wykonywać za pomocą Linuksa. W niektórych przypadkach nie istnieje absolutnie żaden powód do zmiany.

Bazy danych Sybase całkiem dobrze działają na serwerach Sun od dziesiątek lat. Nadającą się do użytku wersję swojego flagowego produktu Sybase wydało dopiero niedawno. Nie jest to obszar, na którym chcielibyśmy przeprowadzać migrację (nowe wdrożenie może być — lub nie — całkiem inną historią). Z drugiej strony, niektóre cechy linuksowego demona *syslog* są lepsze niż w przypadku stosowania Solarisa w roli centralnego komputera dzienników. Na podstawie pewnych projektów programistycznych można stwierdzić, że oprogramowanie można budować i testować na Linuksie z powodzeniem. Linux określany jest czasem mianem implementacji Uniksa, więc możemy użyć tego argumentu, aby pomóc w uzasadnieniu przejścia w tym kierunku. Tak więc odróbmy swoją pracę domową i wygrajmy swoją bitwę!

## Bądź cierpliwy

Osobiście wolę, aby wdrożenie było bliskie doskonałości, niż gdyby miało być zrobione na wczoraj. Najlepsze byłoby połączenie tych dwóch elementów, ale jak uczy historia, jest to nierealne.

Nie porywajmy się z motyką na słońce. Niech Linux pojawi się na komputerach klienckich, u naszego szefa i u naszych użytkowników. Postawmy serwer pocztowy, zainstalujmy SpamAssasin, *procmail* i portal webmail na serwerze Apache. Następnie utrzymujmy go, optymalizujmy i zabezpieczmy. Jeśli wykonamy to wszystko, Linux zacznie zdobywać uznanie w naszym środowisku. Możemy utworzyć serwer list dyskusyjnych, zbudować opartą na LDAP książkę adresową, z której użytkownicy będą mogli uzyskać informacje adresowe w swoich aplikacjach pocztowych. Jeżeli dobrze to rozegramy, za rok ludzie zaczną sobie zdawać sprawę, że uruchomienie tych wszystkich usług wymagało poświęcenia stosunkowo niewielu zasobów i że wszystko „po prostu działa". Gdy będą już gotowi do większych zmian, do kogo się zwrócą? Do faceta, który chciał u pewnej starszej pani zastąpić maszynę do pisania dwumonitorowym komputerem z systemem Linux?

Pomyśl. Zadzwonią do nas.

SPOSÓB 45.

# Priorytety w pracy

Choć chyba nikt w firmie nie potrzebuje bardziej kursu zarządzania czasem niż administratorzy systemu, to są oni niejednokrotnie ostatnimi osobami, które usiłują zorganizować swoje życie zawodowe.

Większość administratorów jest zdania, że nie można ze wszystkim w pracy nadążyć, mając do dyspozycji jedynie 40 godzin w tygodniu. Często trzeba pracować wieczorami i w weekendy. Czasami jest to fajne, np. gdy zaczynamy pracować z nowymi technologiami. Spójrzmy prawdzie w oczy — większość administratorów lubi komputery i często pracuje na nich nawet w swoim wolnym czasie. Jednak praca po 60 godzin tygodniowo,

miesiąc po miesiącu, to nic przyjemnego. Nie będzie czasu na upragnione życie towarzyskie, a i firma nie będzie miała z nas pożytku, jeśli będziemy wciąż w podłym nastroju z powodu niewyspania czy braku czasu. A pracy wciąż przybywa i w końcu nigdy nie będziemy w stanie zmieścić się ze wszystkim w ciągu zwykłego czasu pracy. Dlatego potrzebny jest sposób na ustalenie priorytetów dla naszych działań. Wiem, w rzeczywistości nie chodzi o sposób dotyczący serwerów linuksowych, ale związane jest to z życiem administratora systemu, co oznacza, że powinno to dotyczyć każdego, kto czyta tę książkę.

## Priorytety zadań

Zarządzanie naszymi zadaniami to nie tylko upewnienie się, że zrobiliśmy wszystko w odpowiednim czasie. Pomaga ono także lepiej przewidzieć, kiedy dana praca może zostać zakończona, a co ważniejsze, nasi klienci będą zadowoleni, gdyż nasza praca lepiej spełni *ich* oczekiwania co do tego, kiedy ich prośby zostaną spełnione. Następne punkty omawiają metody, jakie możemy zastosować w uporządkowaniu swoich zadań.

### Wykonywanie zadań w określonej kolejności

Jedną z metod uporządkowania swoich zadań jest niepoświęcanie temu zbyt wiele czasu. Podejmijmy po prostu decyzje i poczynając od góry listy zacznijmy je po kolei realizować, jedną po drugiej. W czasie, który poświęcilibyśmy na rozważania, od czego zacząć, można by pewnie wykonać kilka drobniejszych prac. Oprócz tego, ponieważ pierwszymi pozycjami na liście są zadania, których nie zdążyliśmy wykonać poprzedniego dnia, będziemy często pracować najpierw nad najstarszymi pozycjami.

Wykonywanie zadań z listy do zrobienia w zapisanej kolejności to doskonały sposób na uniknięcie ich odwlekania. Cytując reklamę Nike: „Po prostu zrób to".

Jeżeli nasza lista jest na tyle krótka, że możemy wykonać wszystkie zapisane na niej pozycje w ciągu jednego dnia, taki system nabiera większego sensu — jeśli nie ma znaczenia, czy dane zadanie wykonane zostanie rano czy pod koniec pracy, kto by się przejmował tym, czy została zachowana określona kolejność? Oczywiście nie zawsze tak jest…

### Ustalanie priorytetów w oparciu o oczekiwania klientów

Zdradzę tu mały sekret, który podpatrzyłem od Ralpha Loury'ego, gdy był on moim szefem w Bell Labs. Jeśli mamy listę zadań, wykonanie ich w dowolnej kolejności zajmuje (w przybliżeniu) taką samą ilość czasu. Jeśli jednak wykonamy je w kolejności opartej na oczekiwaniach klientów, będą nas oni postrzegać jako osoby szybciej pracujące. Taka sama ilość czasu dla nas, lepsze postrzeganie przez klientów. Nieźle, co?

A zatem jakie są oczekiwania naszych klientów? Z pewnością wszyscy klienci chcieliby, aby wszystkie ich prośby były spełnianie natychmiast, ale w rzeczywistości zdają oni sobie sprawę, że pewne rzeczy zajmują trochę czasu. Oczekiwania użytkownika mogą być nierealne i są często spowodowane niezrozumieniem technologii, ale wciąż istnieją.

Oczekiwania użytkowników możemy podzielić na kilka kategorii:

*Prośby, które należy spełnić szybko.*
Przykładem mogą tu być prośby o zresetowanie hasła, przydzielenie adresu IP czy usuniecie chronionego pliku. Wspólnym elementem tych próśb jest to, że często związane są one z drobnymi zadaniami, które wstrzymują zadania poważniejsze. Wyobraźmy sobie frustrację oczekującego użytkownika, gdy nie może on nic zrobić, dopóki hasło nie zostanie zresetowane, a nam zabiera to całe godziny.

*Zadania typu „pospiesz się i czekaj" powinniśmy mieć z głowy jak najszybciej.*
Zadania, które są wstępem do innych zadań, powinny być wykonane szybko. Na przykład zamówienie niewielkiego elementu sprzętowego wymaga zwykle trochę pracy związanej z wysłaniem zamówienia kupna, po czym następuje długie oczekiwanie na nadejście przesyłki. Potem możemy dany element zainstalować. Jeśli okres oczekiwania ma wynosić dwa tygodnie, oczekuje się, że zamówienie zostanie złożone szybko, aby dwa tygodnie nie przerodziły się w trzy.

*Prośby, których spełnienie zajmuje sporo czasu.*
Przykładem może tu być instalacja nowego komputera, utworzenie usługi „od zera" lub cokolwiek, co wymaga wykonania zakupu. Nawet jeśli sprzedawca oferuje całodobową wysyłkę, większość ludzi akceptuje fakt, że całodobowa nie oznacza „natychmiastowa".

*Wszystkie pozostałe prace wstrzymujemy do naprawienia przyczyny przestoju*
Ostatnią kategorią są przestoje. Nie tylko istnieje oczekiwanie, że podczas przestoju zostaną wstrzymane wszystkie inne zadania w celu rozwiązania tej kwestii; istnieje także oczekiwanie, że cały zespół będzie nad tym pracował. Klienci zazwyczaj nawet nie wiedzą, że w obrębie zespołu administratorów istnieją działy robocze.

Teraz, gdy lepiej rozumiemy oczekiwania klientów, czy możemy jakoś wykorzystać tę wiedzę? Przypuśćmy, że nasza lista zadań do wykonania zawiera pozycje przedstawione na rysunku 4.4.

| Zadania | Opis | Oczekiwanie | Faktyczna praca | Czas zakończenia |
|---|---|---|---|---|
| T1 | Zresetowanie hasła | 1 minuta | 10 minut | 9:10 |
| T2 | Utworzenie konta nowego użytkownika | następny dzień | 20 minut | 9:30 |
| T3 | Instalacja nowego serwera | następny dzień | 4 godziny (+1 godzina na przerwę obiadową) | 14:30 |
| T4 | Dodanie nowego obszaru CGI do serwera WWW | 1 godzina | 30 minut | 15:00 |
| T5 | Zamówienie pakietu oprogramowania | 1 godzina | 1 godzina | 16:00 |
| T6 | Usunięcie drobnych błędów NetNews | 10 minut | 25 minut | 16:25 |
| T7 | Przydzielenie adresu IP | 2 minuty | 5 minut | 16:30 |

*Rysunek 4.4. Zadania bez priorytetów ustalanych w oparciu o oczekiwania użytkowników*

Jeżeli wykonamy te zadania w zapisanej tu kolejności, ukończenie ich wszystkich w ciągu sześciu i pół godziny porządnej pracy (plus godzina przerwy obiadowej) będzie całkiem zadowalającym wynikiem, i to bez specjalnego wysiłku. Tym lepiej dla nas.

Jednak wrażenie użytkowników dotyczące tempa wykonywania przez nas zadań będzie dla nas niekorzystne. Osoba, która wystosowała prośbę „T7", będzie musiała czekać cały dzień na co coś, co według niej powinno zająć dwie minuty. Gdybym to ja był tym klientem, byłbym bardzo zdenerwowany. Z powodu braku adresu IP instalacja nowego elementu wyposażenia laboratorium opóźniła się o cały dzień.

(Właściwie bardziej prawdopodobny jest taki scenariusz, w którym sfrustrowany i niecierpliwy klient nie zechce czekać cały dzień. Będzie pingować adresy IP, dopóki nie znajdzie takiego, który jest akurat nieużywany i „czasowo pożyczy sobie" ten adres. Gdyby to był nasz pechowy dzień, wybrany adres mógłby wywołać z czymś konflikt i spowodować przestój, co zrujnowałoby nam cały dzień. Ale to tylko taka dygresja…)

Zmieńmy kolejność zadań, opierając się na wyobrażeniach użytkowników dotyczących tego, ile czasu powinny one zabrać. Czynności postrzegane jako zabierające niewiele czasu lub pilne zadania zostaną przesunięte na początek i wykonane z samego rana. Pozostałe zadania zostaną wykonane później. Rysunek 4.5 przedstawia przeorganizowane zadania.

| Zadania | Opis | Oczekiwanie | Faktyczna praca | Czas zakończenia |
|---|---|---|---|---|
| T1 | Zresetowanie hasła | 1 minuta | 10 minut | 9:10 |
| T7 | Przydzielenie adresu IP | 2 minuty | 5 minut | 9:15 |
| T5 | Zamówienie pakietu oprogramowania | 1 godzina | 1 godzina | 10:15 |
| T4 | Dodanie nowego obszaru CGI do serwera WWW | 1 godzina | 30 minut | 10:45 |
| T2 | Utworzenie konta nowego użytkownika | następny dzień | 20 minut | 11:05 |
| T3 | Instalacja nowego serwera | następny dzień | 4 godziny (+1 godzina na przerwę obiadową) | 16:05 |
| T6 | Usunięcie drobnych błędów NetNews | 10 minut | 25 minut | 16:30 |

*Rysunek 4.5. Zadanie uszeregowane w oparciu o oczekiwania klientów*

Rozpoczniemy dzień od wykonania dwóch zadań (T1 i T7), które są przez użytkowników oczekiwanie jak najszybciej, gdyż wstrzymuje to inne, większe projekty. Wypełnijmy te zadania w oczekiwanym czasie, a wszyscy będą zadowoleni.

## Priorytety projektów

Poprzedni punkt opisywał sposób przydzielania priorytetów poszczególnym zadaniom. Teraz przedstawię techniki przydatne przy ustalaniu priorytetów dla projektów.

### Ustalanie priorytetów przez wpływ

Powiedzmy, że wraz z innymi administratorami podczas „burzy mózgów" wymyśliliśmy 20 dużych projektów na następny rok. Jednak nasz budżet i ilość personelu pozwala na wykonanie tylko kilku z nich. Które projekty powinniśmy wybrać?

Kuszący wydaje się pomysł wybrania łatwych projektów i wykonania ich w pierwszej kolejności. Wiemy, jak je wykonać i nie ma wokół nich wiele kontrowersji, więc przynajmniej będziemy wiedzieć, że zostaną ukończone.

Kuszący wydaje się również pomysł wybrania projektów sprawiających przyjemność, politycznie bezpiecznych lub projektów, które stanowią oczywisty kolejny krok wypływający z poprzednich projektów.

Zignorujmy te pokusy i wyszukajmy projekty, które będą miały największy pozytywny wpływ na cele naszej organizacji. Właściwie lepiej jest wykonać jeden wielki projekt, który będzie miał duży pozytywny wpływ, niż wiele łatwych projektów, które będą powierzchowne — widziałem to wiele razy. Ponadto cały zespół pracujący nad jednym celem pracuje lepiej niż gdy każdy ma oddzielny projekt, ponieważ pracujemy lepiej, kiedy pracujemy wspólnie.

Oto kolejny sposób, w jaki możemy na to spojrzeć. Wszystkie projekty możemy zaliczyć do jednej z czterech kategorii przedstawionych na rysunku 4.6.

| | **Łatwe (mały wysiłek)** | **Trudne (duży wysiłek)** |
|---|---|---|
| **Duży pozytywny wpływ** | A | B |
| **Powierzchowny wpływ** | C | D |

*Rysunek 4.6. Wpływ projektu a poniesiony wysiłek*

Wykonanie projektów z kategorii A wygląda na oczywisty wybór. Łatwy projekt, który będzie miał duży pozytywny wpływ to rzadkość, a jeśli jakimś cudem pojawi się przed nami możliwość takiego projektu, wykonanie go będzie zawsze dobrym wyborem (uwaga: bądźmy ostrożni, ponieważ ich status A może być złudzeniem!).

Oczywiste jest również unikanie projektów z kategorii D. Projekty, które są trudne i nie zmienią wiele, nie powinny być podejmowane.

Jednak większość projektów znajduje się w kategoriach B lub C, a w naturze ludzkiej leży wybieranie łatwych projektów C. Możemy wypełnić nasz rok łatwymi projektami, spisać wiele osiągnięć i odejść z twarzą. Jednak firmy odnoszące sukcesy szkolą kadrę kierowniczą, by nagradzała pracowników wybierających projekty z kategorii B — trudne, ale niezbędne.

Jeśli myślimy o tym w kategoriach zwrotu inwestycji, ma to sens. Zamierzamy w tym roku wydać pewną ilość pieniędzy. Czy wydamy je ma wiele małych projektów, z których żaden nie przyniesie dużego wpływu? Nie, powinniśmy szukać tego, co będzie miało największy pozytywny wpływ i całe swoje inwestycje przeznaczyć na ten wysiłek.

Ważne jest, aby się upewnić, że projekty o dużym wpływie są zgodne z celami naszej firmy — ważne dla firmy, ale i dla nas. W ten sposób będziemy wyżej oceniani.

### Ustalanie priorytetów dla próśb naszego szefa

Jeśli nasz szef poprosi nas o zrobienie czegoś, i jest to szybkie zadanie (a nie jakiś wielki projekt), zróbmy to od razu. Na przykład: jeśli nasz szef poprosi nas o sprawdzenie, na ilu komputerach zainstalowana jest starsza wersja Windows, wróćmy do niego z porządnym wyliczeniem w ciągu kilku minut.

Musimy zrozumieć, że takie prośby zazwyczaj wynikają z faktu, że nasz szef układa większy plan lub budżet (być może szacunkowy koszt zainstalowania na wszystkich komputerach najnowszej wersji Windows), a my możemy zabrać mu cały dzień, nie podając szybko odpowiedzi.

Dlaczego ma to takie znaczenie? Cóż, nasz szef decyduje o tym, co się wydarzy przy najbliższej analizie pensji. Czy trzeba mówić coś więcej? Może i tak. Nasz szef ma ograniczoną kwotę pieniędzy na przyznanie podwyżek. Jeśli ktoś dostanie więcej, to ktoś inny musi dostać mniej. Pewnie chcielibyśmy, aby przy przeglądaniu listy członków zespołu nasz szef spojrzał na nasze nazwisko i pomyślał: „To ten, który tak szybko dał mi wyliczenie ilości komputerów z przestarzałymi systemami Windows. On zawsze daje mi szybko wszystko, czego potrzebuję". Czy chcielibyśmy, aby nasz szef pomyślał: „Przez niego budżet opóźnił się o cały dzień, bo musiałem czekać na te obliczenia"? Albo jeszcze gorzej: „Głupio wyglądałem przed moim szefem, gdy się spóźniłem, a to przez to, że musiałem czekać, aż ten [tutaj możemy wstawić swoje nazwisko] przyniesie mi te informacje. Taki-a-taki nie dostanie w tym roku podwyżki". Uszczęśliwianie swojego szefa zawsze jest dobrym pomysłem!

## Podsumowanie

Zarządzanie swoimi priorytetami pozwoli nam spełnić oczekiwania klientów i wykonać pracę o największym wpływie w wyznaczonym czasie. Jednak ustalanie priorytetów to tylko jedno z rozwiązań zarządzania czasem. Choć więcej pomysłów można znaleźć w książkach poświęconych ogólnie zarządzaniu czasem, sugeruję tu skromnie przeczytanie mojej książki, *Time Management for System Administrators* (O'Reilly).

*— Tom Limoncelli*