Uniwersytet w Białymstoku

Wydział Matematyczno-Fizyczny

Instytut Matematyki

Adam Piwnikiewicz

System zarządzania hostami wirtualnymi HTTP

Praca dyplomowa napisana pod kierunkiem
dr hab. K. Prażmowskiego, prof. UwB

Białystok 2003

Składam serdeczne podziękowania
mgr Mariuszowi Żynelowi
za pomoc przy pisaniu tej pracy.

Adam Piwnikiewicz

Spis treści

Wstęp

W erze dot-com, rozpoczętej przed kilku laty, kiedy to technologie internetowe trafiły pod strzechy, nie ma osoby na świecie, która nie słyszałaby o Internecie. Co ciekawe sieć Internet utożsamiana jest z usługą World Wide Web, w skrócie WWW, która na dobre zadomowiła się w naszym życiu. Dzieje się tak, dlatego, że dominującą rolę wśród wszystkich usług oferowanych w Internecie jest właśnie WWW. Szacuje się, że ponad 80 procent wszystkich usług przypada właśnie na WWW. Widać, zatem jak ważną role pełnią serwery HTTP w erze dot-comu.

Usługa WWW jest stosunkowo młodą usługą w sieci Internet. Mogła się ona upowszechnić dopiero wtedy, gdy powszechne stały się okienkowe systemy operacyjne i odpowiednie graficzne przeglądarki. Usługa WWW oparta jest na technologii klient- serwer. Rolę klienta w tym wypadku pełni *przeglądarka WWW*, zwana też *przeglądarką internetową*. Komunikacja pomiędzy klientem a serwerem odbywa się zgodnie z protokołem HTTP (ang. *Hyper Text Transfer Protocol*). Stąd bierze się określenie *serwer HTTP* lub *serwer WWW*.

Podstawowym typem dokumentów przesyłanych na WWW jest hipertekst. To, co go wyróżnia spośród innych dokumentów to możliwość umieszczania odnośników do innych dokumentów, nie koniecznie znajdujących się na tym samym serwerze WWW. Powstaje w ten sposób sieć powiązań dokumentów, co prawdopodobnie było źródłem dla określenia World Wide Web. Serwer HTTP jest to oprogramowanie realizujące usługę WWW. Jego głównym zadaniem jest udzielanie odpowiedzi na żądanie klienta i przekazywanie stosownych dokumentów.

Zdecydowana większość serwerów pracujących w sieci Internet to serwery pracujące na maszynach UNIX. Do głównych typów serwerów WWW możemy zaliczyć: Serwer Apache, Serwer NCSA i Serwery Netscape. Z wymienionych serwerów najbardziej popularnym i najczęściej stosowanym jest Apache. Poza tym jest to oprogramowanie *open-source* tzn. jest rozpowszechniane wraz z kodem źródłowym bez pobierania opłat [1]. To właśnie uzasadnia powody dla których w niniejszej pracy zajmujemy się serwerem Apache, a dokładniej Apache 1.3.26.

Do głównych zadań administratora serwera HTTP należy między innymi:

- konfiguracja serwera,

- konfiguracja oprogramowania współpracującego, takiego jak CGI, dodatkowe moduły itp.,
- przeglądanie dzienników systemowych,
- tworzenie statystyk,
- usuwanie usterek.

Powyższe zadania wymagają dużych nakładów pracy i o ile nie uda się całkowicie zlikwidować tego problemu, to możemy spróbować go zminimalizować. Aby tego dokonać należy:

- udokumentować różne warianty wirtualnych hostów HTTP, podać przykłady konfiguracji, rozważyć argumenty za i przeciw, którymi należy się kierować przy wyborze wariantu,
- zaprojektować strukturę katalogów wirtualnego hosta HTTP z uwzględnieniem mechanizmów CGI, tworzenia dzienników oraz statystyk,
- opracować oprogramowanie wspomagające czynności wykonywane przez administratora (cykliczne tworzenie statystyk, zakładanie i usuwanie serwerów wirtualnych),

Co zostało zrobione w niniejszej pracy.

Określenie *serwer* stosuje się zarówno w odniesieniu do sprzętu jak i oprogramowania, aby wyrazić specjalne jego zastosowania i możliwości. Termin *wirtualny serwer* odnosi się do sytuacji, gdy na jednej maszynie znajduje się więcej niż jeden serwer, co w znaczący sposób ułatwia prace administratorom i pozwala zaoszczędzić na sprzęcie. W przypadku serwera Apache możliwe są dwa rodzaje serwerów wirtualnych, co zostanie dokładnie opisane w pierwszym rozdziale. Rozważam tam wady i zalety obydwu wariantów. Zawiera on także konkretne przykłady typowych sytuacji oraz najczęściej stosowanych przez administratorów rozwiązań. W drugim rozdziale rozważamy kwestię organizacji serwera HTTP; strukturę katalogów oraz przechodzimy do programów wspomagających pracę administratora serwera HTTP.

Rozdział 1

Wirtualne serwery HTTP

1.1 Zasady działania

Istnieją dwa warianty wirtualnych serwerów HTTP. Zasadnicza różnica między tymi dwoma typami uwypuklona jest już w ich nazwach; mianowicie dla *IP-based Virtual Host*, czyli wirtualnego serwera opartego o adres IP jest potrzebny oddzielny adres IP, w przypadku zaś *Name-based Virtual Host*, czyli wirtualnego serwera opartego o nazwę domenową, serwer kieruje się nazwami domen z nagłówka HTTP. Tak, więc by umieścić na jednej maszynie więcej niż jeden wirtualny serwer oparty o adres IP należy przypisać każdemu z nich oddzielny numer, co zmusza nas do zamontowania tylu interfejsów ile wirtualnych serwerów mamy zamiar umieścić. Sytuacja, gdy do jednej karty sieciowej przypisanych jest więcej niż jeden adres IP jest możliwa, lecz niezbyt wydajna, dlatego też stosuje się ją tylko w wyjątkowych wypadkach. Jak choćby sytuacja, w której jeden z interfejsów, na którym znajduje się jakaś ważna wirtualna strona, jest uszkodzony i nie mamy możliwości zainstalowania kolejnego interfejsu, a stworzenie wirtualnej strony opartej o nazwę nie rozwiązuje problemu. Wówczas przypisanie dodatkowego adresu jednej karcie może być tymczasowym rozwiązaniem. Widać, więc że jedynym wydajnym sposobem na zwiększenie liczby wirtualnych serwerów na jednej maszynie, bez instalowania odpowiedniej ilości interfejsów to konfiguracja wirtualnych serwerów opartych o nazwy domenowe. Okazuje się jednak, że są sytuacje, w których wirtualne serwery oparte o IP są wygodniejszym rozwiązaniem. Oto niektóre z nich:

- wirtualne serwery korzystające z protokołu SSL — wynika to ze specyfiki działania tego protokołu. Połączenie SSL, wygląda następująco:
  - inicjacja połączenia,
  - wymiana certyfikatów,
  - żądanie dokumentu i odpowiedź serwera.

    Wynika z tego, że certyfikat jest podawany klientowi zanim przeglądarka przekaże serwerowi, od jakiej nazwy (wirtualnego serwera)

chce pobrać witrynę. Przez to właśnie dla każdego połączenia przypada tylko jeden certyfikat, czyli na jeden adres serwera można użyć tylko jednego certyfikatu, co oznacza jeden SSL na jednym adresie. Adres w tym przypadku to kombinacja IP: port.

- niektóre starsze przeglądarki nie są kompatybilne z wirtualnymi serwerami opartymi o nazwy, ponieważ dla ich poprawnego działania klient musi wysłać nagłówek `Host`.

Aby lepiej zrozumieć ideę wirtualnych serwerów opartych o nazwy domenowe musimy odwołać się do podstaw protokołu HTTP. HTTP w warstwie używa protokołu TCP/IP. Umożliwia wymianę dokumentów hipertekstowych między serwerem a przeglądarką. HTTP działa wykorzystując zasadę żądanie-odpowiedź (ang. *request-response*). Każdy komunikat w protokole HTTP, czyli żądanie i odpowiedź, składa się z linii żądania lub statusu, listy nagłówków (ang. *headers*) oraz treści komunikatu (ang. *content*). Linia żądania nie może być pusta, w niej określona jest metoda, ścieżka do żądanego dokumentu oraz wersja stosowanego protokołu. Najpowszechniejsze metody protokołu HTTP to GET i POST. Oto przykład kompletnego zapytania z metodą GET:

```
GET /index.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/2.02 Gold
Host: www.uwb.edu.pl
Accept:image/gif,*/*
```

Przeglądarka w nagłówku `Host` podaje nazwę domenową serwera (w tym wypadku: `www.uwb.edu.pl`), na którym znajduje się żądany dokument (tutaj: `index.html`). Właśnie dzięki nagłówkowi serwer wie, jakiego wirtualnego serwera opartego o nazwę dotyczy żądanie. Ponadto podawana jest wersja protokołu, którego używa przeglądarka. W tym wypadku jest to HTTP/1.0. Jeśli `index.html` istnieje odpowiedź przeglądarki może wyglądać następująco:

```
HTTP/1.1 200 OK
Date: Sun, 08 Jun 2003 13:04:15 GMT
Server: Apache/1.3.27 (Unix) PHP/4.2.3
Last-Modified: Wed, 28 May 2003 09:39:09 GMT
Content-Length: 5171
Connection: close
Content-Type: text/html

<doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<body>
```

```
Wirtualne serwery
<\body>
<\html>
```

Kolejne wiersze oznaczają odpowiednio: udane połączenie wraz z protokołem, którego używa serwer, datę połączenia, serwer, który odpowiada, datę ostatniej modyfikacji dokumentu, długość treści komunikatu podana w znakach (bajtach), typ zwracanego dokumentu, i na końcu treść dokumentu. Metoda POST różni się od metody GET możliwością przesyłania większej ilości różnego rodzaju danych do serwera.

Nagłówek `Host` nie jest obowiązkowy w wersji 1.0, ale niemal wszystkie przeglądarki wysyłają ten nagłówek. Bez niego nie możliwe jest funkcjonowanie wirtualnych serwerów opartych o nazwy. Dlatego też w wersji 1.1 nagłówek `Host` jest obowiązkowy.

1.2 Konfiguracja

Po skompilowaniu i zainstalowaniu serwera pliki konfigurujące znajdują się w katalogu `/opt/etc/httpd`. Należą do nich:

httpd.conf Jest to główny plik konfiguracyjny Apache'a. W pliku tym zapisane są m.in. informacje na temat komputera, portu, trybu pracy, logów, zasobów udostępnionych przez serwer. Tutaj również przeprowadzamy konfigurację serwerów wirtualnych WWW uruchamianych przy pomocy Apache.

srm.conf Plik ten konfiguruje sposób zarządzania żądaniami serwera, wskazuje na katalogi, które zawierają informacje oferowane przez serwer i różne elementy potrzebne do formatowania i prezentowania informacji.

acces.conf Służy do definiowania kontroli dostępu dla serwera i dostarczanych przez niego informacji.

Wszystkie trzy pliki mają podobną strukturę. Są one plikami ASCII, komentarze zaczynają się od znaku `#` i wszystkie posiadają obszerny komentarz do poszczególnych dyrektyw. Większość dyrektyw w plikach jest zapisana w formie kluczy, po których następuje wartość przypisana do tego klucza. Użycie trzech plików konfiguracyjnych jest podyktowane zachowaniem kompatybilności ze standardem serwera NCSA. Od wersji 1.3.4 zaleca się umieszczenie wszystkich parametrów konfiguracyjnych w jednym pliku `httpd.conf`, co ułatwia zarządzanie serwerem. Pozostałe pliki (`srm.conf i acces.conf`) nadal istnieją, ale zawierają informacje, że istnieją wyłącznie ze względów historycznych a całą konfigurację należy przeprowadzić w pliku http.conf. Konfigurację serwera przeprowadzamy "ręcznie" edytując odpowiednie pliki. W

naszym przypadku jest to jeden plik `httpd.conf`. Oto niektóre dyrektywy, które pozwolą nam na wyodrębnienie różnic pomiędzy hostami IP-based a Name-based:

`Listen` - dyrektywa definiująca adresy oraz porty, na których ma nasłuchiwać serwer,

`DocumentRoot` - określa miejsce w systemie plików gdzie znajduje się zawartość serwera,

`NameVirtualHost` - tę dyrektywę w odróżnieniu od pozostałych konfiguruje się tylko dla wirtualnych serwerów opartych na nazwach domenowych; w niej musimy przypisać adres lub zbiór adresów IP na serwerze by mógł on realizować polecenia dla tych hostów,

`<VirtualHost> </VirtualHost>` - ten blok obejmuje grupę dyrektyw odnoszących się do wyszczególnionego wirtualnego serwera,

`ServerName` - dzięki tej dyrektywie możemy przypisać wirtualnemu serwerowi dowolną nazwę domenową,

`ServerAlies` - pozwala na przypisaniu wirtualnemu serwerowi więcej niż jednej nazwy.

Poniższy przykład przedstawia podobieństwa i różnice pomiędzy dwoma typami wirtualnych hostów. Użyto tylko istotnych dyrektyw. Kolejne wiersze zostały ponumerowane dla orientacji.

```
 1  Port 80                              Port 80

 2  NameVirtualHost 212.33.73.194
 3  <VirtualHost 212.33.73.194>          <VirtualHost 212.33.73.194>
 4  SerwerName math.uwb.edu.pl           SerwerName math.uwb.edu.pl
 5  DocumentRoot /www/math               DocumentRoot /www/math
 6  </VirtualHost>                       </VirtualHost>
 7  <VirtualHost 212.33.73.194>          <VirtualHost 212.33.73.195>
 8  SerwerName matfiz.uwb.edu.pl         SerwerName matfiz.uwb.edu.pl
 9  DocumentRoot /www/matfiz             DocumentRoot /www/matfiz
10  </VirtualHost>                       </VirtualHost>
```

W wierszu pierwszym obu konfiguracji `Port 80` oznacza port, na którym ma nasłuchiwać serwer, domyślnie jest on ustawiony na 80 i nie należy go zmieniać. Dalej widać, że dyrektywę `NameVirtualHost` konfiguruje się tylko dla wirtualnych serwerów opartych o nazwy; jest to konieczne by serwer wiedział do jakiego adresu odnoszą się nazwy wirtualnych hostów. Następne dwa wiersze w bloku `VirtualHost` określają nazwę domenową hosta i położenie dokumentów. W wierszu szóstym widać fundamentalną różnicę pomiędzy serwerami wirtualnymi opartymi o nazwy i adresy IP. W pierwszym przypadku

użyty jest ten sam adres, który był podany w `NameVirtualHost`, natomiast w drugim przypadku występuje inny adres IP.

1.3 Różne typy konfiguracji wirtualnych serwerów

Znamy już zasadniczą różnicę w konfiguracji obydwu typów wirtualnych serwerów, ale by można było docenić wady i zalety tych wariantów musimy wiedzieć, co to jest `demon httpd`. Terminem `demon`(ang. *deamon*) określa się program, który do działania nie potrzebuje ciągłej komunikacji z użytkownikiem, co oznacza, że demon zaczyna startować razem z uruchamianym systemem. Przyjętą konwencją jest dodawanie do nazwy demona litery d, i tak demon Apach'a nazywa się `httpd`, a jego funkcja to nasłuchiwanie na odpowiednich portach i komunikacja z klientem zgodnie z protokołem HTTP. Tu trzeba podkreślić, że konfiguracja demonów na jednym serwerze stwarza dodatkowe możliwości. Możliwe jest przypisanie jednego demona wszystkim adresom IP na maszynie, jak również określenie różnych demonów różnym adresom. Przy podejmowaniu decyzji należy rozważyć kwestię wydajności i bezpieczeństwa.

Zanim przejdziemy do przykładów objaśnijmy kilka dodatkowych dyrektyw konfiguracji Apache:

`<Directory> </Directory>` – blok obejmujący grupę dyrektyw dotyczących ustalonego katalogu na serwerze, które między innymi określają prawa dostępu do tego katalogu,

`Options` – nakazuje demonowi stosowanie pewnych wskazanych w tej dyrektywie opcjonalnych funkcji serwera; te funkcje to:

- `All` – ustawione są wszystkie opcje; ustawienie domyślne,
- `None` – nie są udostępniane żadne opcje,
- `Indexes` – starsze wersje demona automatycznie wyświetlały zawartość katalogu, kiedy tylko przywoływano adres URL, zaś w wywoływanym katalogu nie istniał żaden plik indeksowy określony przez wpis `DirectoryIndex` (np. index.html),
- `FollowSymLinks` – demon umożliwia posługiwanie się symbolicznymi odnośnikami umieszczonymi w tym katalogu; oznacza to możliwość przekraczania struktury przeznaczonej na dokumenty serwera,
- `FollowSymLinksIfOwnerMatch` – właściciel odnośnika i miejsca docelowe muszą być identyczni,
- `ExecCGI` – możliwość wykonywania skryptów CGI,

`deny` – pozwala na zablokowanie połączeń z niechcianych hostów, adresów lub domen,

`allow` – wskazanie hostów, adresów, lub domen, którym zezwalamy na łączenie się z naszym serwerem.

Wszystkie niżej przedstawiane przykłady konfiguracji serwera Apache zostały przetestowane w Instytucie Matematyki na serwerze `theta`. Uzyskane w drodze testów wyniki zostały zastaosowane w praktyce na serwerze `math`.

1.3.1 Typowe konfiguracje wirtualnych serwerów opartych o adresy IP

Przykład 1: Dwa różne IP i jeden demon

Poniższy przykład to sytuacja, gdy dwa wirtualne hosty obsługiwane są przez jednego demona. Dlatego też do konfiguracji potrzebny nam jest jeden plik `httpd.conf`.

Plik `http.conf`

```
<VirtualHost 212.33.73.195:80>
  ServerName theta.uwb.edu.pl
  DocumentRoot /www/theta
    <Directory /www/theta
      Options Indexes FollowSymLinks
      Order allow,deny
      Allow from all
    </Directory>
</VirtualHost>


<VirtualHost 212.33.73.196:80>
  ServerName omega.uwb.edu.pl
  DocumentRoot /www/omega
    <Directory /www/omega>
      Options Indexes FollowSymLinks
      Order allow,deny
      Allow from all
    </Directory>
</VirtualHost>
```

Adresom 212.33.73.195, 212.33.73.196 przypisujemy odpowiednio nazwy domenowe: `theta.uwb.edu.pl`, `omega.uwb.edu.pl`. Dyrektywę `Listen` pozostawiamy nie skonfigurowaną. Oznacza to, że `httpd` (w tym wypadku jeden) będzie nasłuchiwał na standardowych portach (80 i 443) wszystkich interfejsów w systemie. Dla obu wirtualnych serwerów dyrektywa `Allow` skonfigurowana jest tak samo. `Allow from all` oznacza, że udostępniamy połączenie wszystkim, dlatego też dyrektywa `Deny` nie została skonfigurowana.

Przykład 2: Dwa różne IP i dwa demony

Inaczej jest w przykładzie następnym, w którym dwa demony obsługują różne adresy IP. Musimy pamiętać, że każdy demon jest inaczej skonfigurowany, co oznacza, że potrzebne są różne pliki konfiguracyjne. W tym celu można skopiować `httpd.conf` na pliki o różnych nazwach, w naszym przypadku na `httpd1.conf` oraz `httpd2.conf`. i dokonać niezbędnych modyfikacji

Plik `httpd1.conf`

```
Listen 212.33.73.195:80

<VirtualHost 212.33.73.195:80>
  ServerName theta.uwb.edu.pl
  DocumentRoot /www/theta
   <Directory /www/theta>
     Options Indexes FollowSymLinks
     Order allow,deny
     Allow from all
   </Directory>
</VirtualHost>
```

Plik `httpd2.conf`

```
Listen 192.168.0.1:80

<VirtualHost 192.168.0.1:80>
  ServerName omega.uwb.edu.pl
  DocumentRoot /www/omega
   <Directory /www/omega>
    Options Indexes FollowSymLinks
    Order allow,deny
    Deny from all
    Allow uwb.edu.pl
   </Directory>
</VirtualHost>
```

Dyrektywa `Listen` w pliku `httpd1.conf` sprawia, że jeden z demonów nasłuchuje i odpowiada na adres 212.33.73.195:80 przypisany do komputera o nazwie `theta.uwb.edu.pl`, natomiast konfiguracja `Listen` w pliku `httpd2.conf` powoduje, że drugi z demonów obsługuje adres 192.168.0.1:80 przypisany do `omega.uwb.edu.pl`.

Podsumowanie

W obu przykładach wybrano bardziej niezawodny wariant serwerów wirtualnych. Nie ma tutaj znaczenia, jakiej używa się przeglądarki i jakiej wersji protokołu HTTP. Połączenie się z odpowiednim hostem odbywa się niezależnie od nich. W łatwy sposób możemy w sieci zaimplementować bezpieczny wariant HTTP, a mianowicie szyfrowany HTTPS przy użyciu technologii SSL.

Rozwiązanie z przykładu 2 jest wydajniejsze od tego z przykładu 1; a to, dlatego, że jeden demon obsługuje tylko jeden adres IP. To rozwiązanie jest również bezpieczniejsze. Procesy demonów są tutaj odizolowane od siebie, więc nawet, gdy w jednym z nich wystąpią błędy krytyczne czy włamanie, to drugi będzie działał niezależnie.

Z tego powodu taki scenariusz stosuje się, gdy jedna maszyna znajduje się w dwu różnych sieciach, często jedną z nich jest publiczny Internet, a drugą prywatny Intranet. Oczywiście wyższy sposób bezpieczeństwa w takiej sytuacji uzyskamy rozdzielając role serwera publicznego i prywatnego między różne maszyny, ale pociąga to za sobą znacznie wyższe koszty finansowe oraz koszty administracji i konserwacji. W tym przypadku `omega.uwb.edu.pl` należy do prywatnego Intranetu. Dlatego też nie chcemy żadnych połączeń (`Deny fram all`), prócz połączeń z sieci lokalnej (`Allow from uwb.edu.pl`)

1.3.2 Typowe konfiguracje wirtualnych serwerów opartych o nazwy domenowe

Przykład odnosi się do sytuacji, gdy dysponujemy jednym numerem IP, a potrzebujemy dwóch wirtualnych serwerów. Jedyne rozwiązanie to przypisanie dwóm domenom tego samego adresu IP. Jednak przy konfiguracji wirtualnych serwerów opartych o nazwy musimy pamiętać, że do obsługi jednego adresu nie możemy przypisać więcej niż jednego demona. Tak, więc tym razem do konfiguracji użyjemy tylko jednego pliku `httpd.conf`.

Plik `httpd.conf`

```
VirtualSerwerName 212.33.73.195:80

<VirtualHost 212.33.73.195:80>
  ServerName theta.uwb.edu.pl
  DocumentRoot /www/theta
   <Directory /www/theta>
     Options Indexes FollowSymLinks
     Order allow,deny
     Allow from all
   </Directory>
</VirtualHost>
```

```
<VirtualHost 212.33.73.195:80>
  ServerName omega.uwb.edu.pl
  DocumentRoot /www/omega
   <Directory /www/omega>
     Options Indexes FollowSymLinks
     Order allow,deny
     Allow from all
   </Directory>
</VirtualHost>
```

Ponownie jeden demon pozwala nam na pominięcie w konfiguracji dyrektywy `Listen`, oczywiście użyta jest dyrektywa `VirtualSerwerName` bez której nie możliwe byłoby przypisanie różnym domenom tego samego adresu. Używając tylko jednego adresu do wirtualnych serwerów opartych o nazwy domenowe musimy się liczyć z niebezpieczeństwem, jakie niesie ze sobą obsługiwanie ich przez jednego demona. To rozwiązanie jest najczęściej stosowane w przypadku sieci publicznej Internet.

1.3.3 Użycie dwóch wariantów wirtualnych serwerów na jednej maszynie

Poza sytuacją opisaną w 1.3.1 gdy jedna maszyna - serwer WWW - znajduje się w różnych sieciach, konieczne jest zastosowanie mieszanych wariantów serwerów wirtualnych, gdy chcemy użyć protokołu HTTPS.

Plik `http1.conf`

```
# Konfigurujemy Apache bez obsługi SSL dla stron
#, które go nie używają

  Listen 212.33.73.195:80

  VirtualSerwerName 212.33.73.195:80

  <VirtualHost 212.33.73.195:80>
    ServerName theta.uwb.edu.pl
    DocumentRoot /www/theta
      <Directory /www/theta>
        Options Indexes FollowSymLinks
        Order allow,deny
        Allow from all
      </Directory>
  </VirtualHost>
```

```
  <VirtualHost 212.33.73.195:80>
    ServerName omega.uwb.edu.pl
    DocumentRoot /www/omega
     <Directory /www/omega>
       Options Indexes FollowSymLinks
       Order allow,deny
       Allow from all
     </Directory>
  </VirtualHost>


# Następnie konfigurujemy Apache dla wirtualnego
# serwera z SSL

   Listen 212.33.73.195:443

  <VirtualHost 212.33.73.195:443>
    DocumentRoot /www/mathmail
    ServerName mathmail.uwb.edu.pl
    SSLEnable
    SSLCACertificatePath /opt/etc/apache/ssl
    SSLCACertificateFile /opt/etc/apache/ssl/cacert.pem
    SSLCertificateFile /opt/etc/apache/ssl/newcert.pem
    SSLCertificateKeyFile /opt/etc/apache/ssl/newreq.pem
  </VirtualHost>
```

Adres 212.33.73.195:80 przypisany jest dla dwóch wirtualnych serwerów opartych o nazwy domenowe, natomiast adres 212.33.73.195:443 przypisany jest dla wirtualnego serwera opartego o IP, dodatkowo host ten obsługuje protokół SSL. Należy zauważyć, że numer IP jest taki sam we wszystkich adresach. Zastosowanie dyrektywy `Listen` nakazuje serwerowi HTTP nasłuchiwanie portów 80 i 443 pod adresem 212.33.73.195. Ponieważ serwer Apache jako adres traktuje parę IP:port, możliwe jest, więc posiadanie zarówno wirtualnych hostów opartych o nazwy jak i adresy. W powyższym przykładzie port 443 przypisany jest serwerowi używającemu protokołu HTTPS. Dzięki temu, z punktu widzenia serwera Apache, host mathmail ma inny adres niż hosty theta i omega. Natomiast theta i omega mają ten sam adres

1.3.4 SSL a wirtualne serwery oparte o nazwy domenowe

Jak już było wcześniej wspominane specyfikacja protokołu SSL nie pozwala na bezpośrednie użycie więcej niż jednego serwera z protokołem SSL na jednym

adresie. Zmusza to administratorów do używania oddzielnego IP dla każdego z serwerów obsługujących SSL. Zmiana w adresie portu nie rozwiązuje problemu gdyż konieczne jest wtedy pamiętanie tego portu i wpisywanie go w przeglądarce Przykładowo, jeśli przypiszemy adresy w następujący sposób:

- `mathmail.uwb.edu.pl` — 212.33.73.195:443, oraz
- `mathintranet.uwb.edu.pl` — 212.33.73.195:8443,

to by połączyć się z `mathmail.uwb.edu.pl`, konieczne będzie wpisanie w przeglądarkę adresu `mathmail.uwb.edu.pl:443`, co oczywiście nie jest zbyt wygodne. Jest jednak możliwość skonfigurowania serwera tak by obsługiwał więcej niż jedną witrynę SSL na jednym adresie.

Plik `httpd.conf`

```
# Apache bez obsługi SSL

  Port 80
  Listen 80
  NameVirtualHost 212.33.73.195:80

  <VirtualHost 212.33.73.195:80>
    ServerName mathmail.uwb.edu.p
    Redirect / https://mathmail.uwb.edu.pl:443/
  </VirtualHost>

  <VirtualHost 212.33.73.195:80>
    ServerName mathintranet.uwb.edu.pl
    Redirect / https://mathintranet.uwb.edu.pl:8443/
  </VirtualHost>


# Apache z SSL:

  Port 8443
  Listen 8443

  <VirtualHost 212.33.73.195:443>
    DocumentRoot  /www/mathmail
    ServerName mathmail.uwb.edu.pl
    SSLEnable
    SSLCACertificatePath /opt/etc/apache/ssl
    SSLCACertificateFile /opt/etc/apache/ssl/cacert.pem
    SSLCertificateFile /opt/etc/apache/ssl/mathmail.pem
```

```
    SSLCertificateKeyFile /opt/etc/apache/ssl/mathmail.pem
</VirtualHost>


<VirtualHost 212.33.73.195:8443>
    DocumentRoot  /www/mathintranet
    ServerName mathintranet.uwb.edu.pl
    SSLEnable
    SSLCACertificatePath /opt/etc/apache/ssl
    SSLCACertificateFile /opt/etc/apache/ssl/cacert.pem
    SSLCertificateFile /opt/etc/apache/ssl/mathintranet.pem
    SSLCertificateKeyFile /opt/etc/apache/ssl/mathintranet.pem
</VirtualHost>
```

Powyższa konfiguracja spowoduje, że po wpisaniu w przeglądarce adresu `mathmail.uwb.edu.pl` serwer Apache przekieruje przeglądarkę na adres `mathmail.uwb.edu.pl:443`, czyli na wirtualny serwer obsługujący protokół SSL. Możliwe jest to dzięki dyrektywie `Redirect`, która przeadresowuje klienta na podany w niej adres. Tak, więc niewielkim kosztem udało się połączyć wirtualne serwery oparte o nazwy z protokołem SSL.

Rozdział 2

Zarządzanie serwerem HTTP

2.1 Planowanie serwera HTTP dla ISP

Rozpowszechnienie się usługi WWW wiąże się z powstaniem nowych ofert w Internecie, jedną z nich jest wynajmowanie wirtualnych serwerów prywatnym firmom. Wiele firm by utrzymać się na rynku musi zabiegać o klientów nie tylko w wymiarze lokalnym. Do udostępniania swojej oferty na kraj, a nawet świat idealnym narzędziem jest Internet. Dlatego też wiele firm decyduje się na wynajęcie wirtualnego serwera, którego nazwa domenowa może być idealnym nośnikiem reklamowym, a prezentacja usług łatwa i szczegółowa. Za wynajęciem serwera wirtualnego przemawiają stosunkowo duże koszty utrzymania własnego serwera i łącza internetowego.

Takie usługi oferowane są przez `ISP` (ang. *Internet Service Provider*), co w wolnym tłumaczeniu oznacza dostawca usług internetowych. Mimo, że na świecie i w Polsce jest wielu dostawców usług internetowych, to jest niewiele darmowego oprogramowania wspomagającego pracę administratorów ISP.

Aby zapewnić sprawną administracje serwerem HTTP, na którym umieszczono 50 a nawet 100 hostów wirtualnych oraz aby umożliwić automatyczne wykonywanie czynności administracyjnych, niezbędne jest zorganizowanie przejrzystej jednostki i regularnej struktury całego serwisu. Dotyczy to rozmieszczenia plików i katalogów bezpośrednio związanych z wirtualnymi hostami. Ważne jest również rozmieszczenie programów i ich plików konfiguracyjnych, jeśli używa się do obsługi wirtualnych hostów.

Tak, więc przy planowaniu struktury całego serwera należy między innymi:

- oddzielić dokumenty od konfiguracji serwera,
- wydzielić miejsce na programy CGI,
- wydzielić miejsce na dzienniki systemowe,
- oddzielić pliki związane z różnymi domenami.

Oddzielenie dokumentów od konfiguracji serwera, chociaż wydaje się tylko zabiegiem kosmetycznym w rzeczywistości jednak sprawia, że struktura serwera jest bardziej czytelna, co z kolei może nas uratować przed wieloma pomyłkami. W tym celu wszystkie pliki, tzn. dokumenty HTML, dzienniki systemowe, statystyki oraz pliki konfiguracyjne, związane z jednym wirtualnym hostem grupujemy w jednym katalogu o nazwie takiej jak nazwa domenowa tego hosta. Natomiast wszystkie takie katalogi trzymamy w jednym miejscu. W naszym przypadku tym centralnym katalogiem jest `/export/home1/httpd`. Na serwerze `math.uwb.edu.pl` w katalogu `/export/home1/httpd` znajdują się między innymi podkatalogi `math.uwb.edu.pl` oraz `matfiz.uwb.edu.pl` odpowiadające najważniejszym hostom wirtualnym na tym serwerze.

Każdy katalog związany z konkretnym wirtualnym hostem zawiera pięć kolejnych podkatalogów:

`htdocs` — Tutaj znajdują się wszystkie dokumenty składające się na stronę internetową związaną z danym wirtualnym hostem. Zazwyczaj modyfikację plików w tym katalogu powierza się wynajmującemu wirtualny serwer. W zależności od zainstalowanego na serwerze oprogramowania strona może być statyczna lub dynamiczna. Może ona być napisana w języku PHP lub Meta-HTML.

`cgi-bin` — Umieszczane tu pliki to programy CGI obsługujące wirtualny serwer. W przeciwieństwie do dokumentów na stronie, z programami tymi wiąże się problem bezpieczeństwa. Niestety dość łatwo jest zrobić lukę w programie CGI, która pozwoli hakerom na włamanie się do serwera lub uszkodzenie jego zawartości. Dlatego też, katalog ten nie powinien być udostępniany wynajmującemu wirtualny serwer.

`logs` — Serwer Apache będzie umieszczał tu swoje dzienniki:

- `acces.log` - zawiera informacje o udanych połączeniach; odnotowany jest tutaj adres IP klienta, dokładna data połączenia, rodzaj żądania, pełna ścieżka do żądanego dokumentu, status wyniku żądania oraz ilość wysłanych bajtów;
- `error.log` - gromadzi informacje o błędach, jakie wystąpiły podczas udzielania odpowiedzi przez serwer na żądania klientów; spisywane są podobne informacje jak w przypadku `access.log`.

Plik `access.log` jest niezbędny do tworzenia statystyk.

Natomiast na podstawie `error.log` administrator może wyłapać różne problemy. Do najważniejszych należy błąd 404 – nie znaleziono strony, albo 501 – błąd w konfiguracji serwera.

Oba pliki rosną w trakcie używania serwera HTTP. Jeśli dana strona jest często odwiedzana, to plik `access.log` może osiągać duże rozmiary w

krótkim czasie. Rozmiar pliku `error.log` zależy od ilości błędów, jakie wystąpią w związku z daną domeną. Niezbędna jest rotacja obu tych plików, gdyż duże pliki spowalniają pracę serwera HTTP. Częstotliwość rotacji powinna być dostosowana do ilości odwiedzin.

Rotacja polega na tym, że plik dziennika jest usuwany, lub zmieniana jest jego nazwa. W przypadku usuwania tracimy możliwość wglądu do dziennika, dlatego też, lepiej jest przechowywać stare dzienniki w postaci skompresowanej i oznaczone datą, najlepiej w nazwie pliku.

`usage` — Jest to katalog, w którym przechowywane są statystyki wirtualnego hosta. Dobrze przygotowane statystyki służą nie tylko najbardziej zainteresowanemu, którym jest osoba wynajmująca serwer, ale również administratorowi. Na podstawie statystyk można określić, jaka część serwisu cieszy się największym powodzeniem, które strony są odwiedzane najczęściej. Na podstawie wyszukiwanych słów kluczowych, dzięki którym użytkownik dostał się za pośrednictwem wyszukiwarek na naszą stronę, możemy określić, czego klienci oczekują od naszego serwisu.

Konfiguracje poszczególnych wirtualnych hostów można trzymać w jednym pliku `httpd.conf`. Gdy ilość wirtualnych hostów jest duża, to lepiej jest wydzielić oddzielny plik dla każdego z hostów, co ułatwi administratorowi kontrolę i modyfikowanie konfiguracji. W tym celu należy stworzyć podkatalog `vhosts` w katalogu `/opt/etc/httpd` i użyć w `httpd.conf` następującej dyrektywy:

```
Include /opt/etc/httpd/vhosts
```

która to powoduje, że wszystkie pliki znajdujące się we wskazanym w dyrektywie katalogu (tutaj: `/opt/etc/httpd/vhosts`) zostaną włączone do konfiguracji serwera Apache, tzn. plik `httpd.conf` zostanie rozszerzony o wpisy zawarte w tych plikach.

2.2 Programy administracyjne

W celu ułatwienia zarządzania serwerami wirtualnymi, w ramach tej pracy, zostały przygotowane następujące skrypty: `httpvhostadd`, `httpvhostmod` oraz `httpvhostdel`. Mogą one być wykorzystane w sytuacji, gdy serwer posiada jeden adres IP i stosuje się wirtualne hosty oparte o nazwy.

Ponieważ parametry serwera oraz struktura katalogów może być inna na każdej maszynie, na której tych programów chcielibyśmy używać, konieczne było wydzielenie pewnych stałych w tych programach:

`IPADDRESS` – w tej stałej przechowywany jest pełny adres serwera, tzn. para IP:Port; domyślną wartością jest *, co oznacza że adres IP zostanie pobrany przez serwer Apache z ustawień systemowych serwera na którym pracuje;

`HTTPDROOTDIR` – tu podana jest ścieżka do katalogu, w którym przechowywane są strony, logi oraz statystyki poszczególnych serwerów wirtualnych, domyślna wartość `/export/home/httpd`;

`APACHECONFDIR` – w tym katalogu umieszczone są pliki konfiguracyjne poszczególnych serwerów wirtualnych, domyślna wartość `/opt/etc/httpd/vhosts`;

`WEBALIZERCACHE` – ścieżka do pliku, w którym gromadzone są dane z DNS'ów podczas generowania statystyk, domyślna wartość `/opt/var/httpd/dns_cache.db`;

`VHOSTLIST` – plik tekstowy zawierający informacje o założeniu, ostatniej modyfikacji oraz ewentualnym usunięciu wszystkich wirtualnych hostów, pojedynczy wiersz tego pliku zawiera nazwę hosta, oddzielone tabulacją daty założenia, modyfikacji i usunięcia; daty podane są z dokładnością do minuty; brak daty modyfikacji oznacza że konfiguracja hosta nie była zmieniana;
`/opt/etc/httpd/vhostlist`.

2.2.1 Dodanie serwera wirtualnego

Nowy serwer wirtualny dodaje się przy pomocy programu `httpvhostadd`.

Na początku skryptu shell'owego deklarowane są cztery stałe, które zostały opisane wcześniej. Program uruchamia się podając jako argument nazwę wirtualnego serwera. W skrypcie występuje ona jako $1. Jest to argument obowiązkowy, dlatego, też na początku skryptu sprawdzana jest obecność tego parametru. Jeśli nie podano argumentu zwracany jest błąd i następuje wyjście z programu. Dalej wykonywane są następujące kroki:

1. sprawdzenie czy istnieje katalog `HTTPROOTDIR/$1`, jeśli tak, to zwracany jest błąd i program jest przerywany;
2. sprawdzenie czy istnieje katalog `HTTPROOTDIR`, jeśli go nie ma to jest tworzony;
3. stworzenie w `HTTPROOTDIR` katalogu o nazwie $1;
4. utworzenie w `HTTPROOTDIR/$1` podkatalogów opisanych w rozdziale drugim;
5. sprawdzenie czy istnieje katalog `APACHECONFDIR`, w którym umieszcza się opisy poszczególnych wirtualnych hostów, jeśli tego katalogu jeszcze nie ma, to jest tworzony;
6. utworzenie w katalogu `HTTPROOTDIR/$1` pliku `webalizer.conf` konfigurującego program Webalizer dla danego wirtualnego serwera. Poszczególne dyrektywy w tym pliku mają następujące znaczenie:

- `LogFile` – określa plik dziennika serwera HTTP do tworzenia statystyk,
- `OutputDir` – określa katalog, w którym umieszczane są statystyki,
- `incremental` – ustawienie na 'yes' powoduje, że statystyki są tworzone rosnąco,
- `Host` – nazwa wirtualnego serwera,
- `DNSCache` – plik, w którym gromadzone są dane z DNS'ów, do dalszego wykorzystania,
- `Quiet` – przy ustawieniu na 'yes' Webalizer nie wypisuje żadnych komunikatów,
- `HideSite`, `HideReferrer`, `groupURL` – grupowanie wpisów w dzienniku,

7. utworzenie pliku `nph-engine` w katalogu `HTTPROOTDIR/$1/cgi-bin`, który jest plikiem konfiguracyjnym Meta-HTML;
8. utworzenie pliku konfiguracyjnego serwera wirtualnego, który będzie dołączany do konfiguracji serwera HTTP;
9. utworzenie pliku `index.html` w katalogu `HTTPROOTDIR/$1/usage` aby usprawnić przeglądanie statystyk serwera HTTP;
10. dopisanie daty utworzenia wirtualnego hosta do rejestru hostów;
11. restart serwera HTTP, tak, aby dokonane zmiany były aktywne.

2.2.2 Modyfikacja serwera wirtualnego

Może się zdarzyć, że konieczna jest modyfikacja konfiguracji któregoś z wirtualnych serwerów. Zadanie to ułatwia skrypt `httpvhostmod`. Jako argument program ten pobiera nazwę serwera wirtualnego. Po uruchomieniu wyświetlony zostanie plik konfiguracyjny danego hosta w domyślnym edytorze, zdefiniowanym w stałej środowiskowej `EDITOR`.

W skrypcie sprawdzany jest tylko jeden warunek czy istnieje podany serwer wirtualny. Odpowiednio zmieniana jest data i godzina ostatniej modyfikacji w rejestrze wirtualnych hostów. Po zakończeniu modyfikacji serwer Apache jest restartowany.

2.2.3 Usunięcie serwera wirtualnego

Usunięcie hosta wirtualnego ułatwia skrypt `httpvhostdel`. Uruchamia się go podając nazwę hosta wirtualnego, którego chcemy usunąć. Sprawdzane jest jedynie czy istnieje dany serwer wirtualny, który ma być usunięty. Zanim zostaną usunięte pliki hosta tworzone jest kompletne archiwum w katalogu, z

którego uruchamiany jest program. W rejestrze wirtualnych hostów odnotowywane jest usunięcie danego wirtualnego serwera. Następnie usuwane są dokumenty, statystyki oraz logi z `HTTPDROOTDIR/$1` jak również konfiguracja z `APACHECONFDIR`.

2.3 Programy uruchamiane cyklicznie

W systemie Solaris cykliczne wywołanie programu następuje dzięki mechanizmowi zwanemu `cron`. Każdy z użytkowników, w szczególności `root` może zlecić systemowi zadania do jednorazowego lub cyklicznego wykonania. W tym celu należy wykonać odpowiedni wpis do tablicy `crontab` przy pomocy polecenia

```
crontab -e
```

Polecenie to umożliwia edycję tablicy `crontab` przy pomocy domyślnego edytora ze zmiennej `EDITOR`. Syntaktyka tablicy opisana jest w [5].

2.3.1 Tygodniowe przetwarzanie dzienników

Czas tworzenia statystyk określamy poprzez funkcję systemu UNIX `crontab`, w której to możemy zadać dzień miesiąc, a nawet godzinę uruchamiania wybranego przez nas programu. Przykładowy wpis w tablicy `crontab` może wyglądać następująco:

```
30 3 * * *  /opt/sbin/processhttpdlogs
```

co oznacza że, statystyki będą generowane codziennie o godzinie 3.30.

Za tworzenie statystyk odpowiada skrypt `processhttpdlogs`.

Program ten wykonywany jest na wszystkich wirtualnych serwerach. Na początku skryptu serwer Apache jest wyłączany, a następnie nazwa pliku `access.log` zmieniana jest na `access.tmp`, natomiast plik `error.log` zmieniany jest na `error-DATE.log`, gdzie `DATE` to bieżąca data określająca rok, miesiąc i dzień. Wykonywane jest to dla wszystkich wirtualnych hostów. Po tym zabiegu Apache jest restartowany. Później program Webalizer analizuje dziennik połączeń `access.tmp` i przypisuje, o ile jest to możliwe, adresom IP nazwy domenowe. Następnie generowane są statystyki. Po utworzeniu statystyk nazwa pliku acces.tmp jest zmieniana na acces-Date.log. Na koniec oba dzienniki są pakowane programem `gzip`.

2.3.2 Roczne przetwarzanie dzienników

Program `processhttpdlogs-ny` jest bardzo podobny do `processhttpdlogs`. Różnica polega na tym, że po wygenerowaniu statystyk dla wszystkich wirtualnych hostów, wszystkie tygodniowe logi z roku, który się właśnie kończy,

są łączone w jeden duży roczny log. Oczywiście jest on pakowany programem gzip. Ponadto tworzony jest nowy katalog, w którym będą gromadzone statystyki za rok rozpoczynający się. Do tej zmiany dostosowywana jest konfiguracja programu Webalizer. Na koniec do strony startowej statystyk (index.html) dopisywany jest odnośnik do statystyk w rozpoczynającym się roku.

Istotą tego skryptu jest wyłączenie serwera Apache jeszcze przed rozpoczęciem się nowego roku i wygenerowanie statystyk za rok mijający tak, aby zamknąć ten rok i rozpocząć nowy. W tym celu konieczny jest następujący wpis w tablicy crontab:

```
59 23 31 12 *  /opt/sbin/processhttpdlogs-ny
```

Spowoduje to uruchamianie programu `processhttpdlogs-ny` na minutę przed rozpoczęciem się każdego nowego roku.

2.4 Wykorzystanie praktyczne

Wszystkie skrypty będące integralną częścią tej pracy zostały zainstalowane na serwerze `math` w Instytucie Matematyki UwB. Programy adminisracyjne wykorzystywane są przez opiekuna tego serwera. Statystyki HTTP na tym serwerze generowane są w każdą niedzielę rano o godzinie 3:30. Porę te dobrano tak, aby serwer był w tym czasie możliwie najmniej obciążony innymi zadaniami. Odpowiedni wpis w tablicy `crontab` wygląda następująco:

```
30 3 * * 0  /opt/sbin/processhttpdlogs
```

Statystyki można oglądać pod adresem

```
http://math.uwb.edu.pl/usage
```

Na zakończenie każdego roku na serwerze `math` urachamiany jest program `processhttpdlogs-ny`.

Napisane skrypty zostały również zainstalowane i są używane na kilku komercyjnych serwerach, będących własnością prywatnych firm działających w Białymstoku: Solution, Amikom S.C., BCMB, Komandor.

Dodatek A

Skrypty

httpvhostadd

```
#!/bin/sh
#
# Add HTTP virtual host.
#
# Last modified: May 23, 2003
#
# Copyright (c) 2003 Adam Piwnikiewicz & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

SELF=`basename $0`

IPADDRESS=*
HTTPDROOTDIR=/export/home1/httpd
APACHECONFDIR=/opt/etc/httpd/vhosts
WEBALIZERCACHE=/opt/var/httpd/dns_cache.db
VHOSTLIST=/opt/etc/httpd/vhostlist

CURYEAR=`date +%Y`
NOW=`date +"%Y-%m-%d %H:%M"`

# Check input parameters
if [ $# -ne 1 ]; then
    echo "Usage: $SELF virtual_server_name"
    exit 1;
fi

if [ -d $HTTPDROOTDIR/$1 ]; then
    echo "ERROR: $1 already exists"
    exit 2;
fi

# Create directory structure
if [ ! -d $HTTPDROOTDIR ]; then
```

```
    mkdir -p $HTTPDROOTDIR
fi

mkdir $HTTPDROOTDIR/$1
mkdir $HTTPDROOTDIR/$1/cgi-bin
mkdir $HTTPDROOTDIR/$1/htdocs
mkdir $HTTPDROOTDIR/$1/logs
mkdir $HTTPDROOTDIR/$1/usage
mkdir $HTTPDROOTDIR/$1/usage/$CURYEAR

if [ ! -d $APACHECONFDIR ]; then
 mkdir -p $APACHECONFDIR
fi

# Configure Webalizer
cat >$HTTPDROOTDIR/$1/webalizer.conf << EOF
LogFile         $HTTPDROOTDIR/$1/logs/access.tmp
OutputDir       $HTTPDROOTDIR/$1/usage/$CURYEAR
Incremental     yes
HostName        $1
DNSCache        $WEBALIZERCACHE
Quiet           yes
HideSite        *$1
HideReferrer    $1
GroupReferrer   $1
GroupURL        /rdm/*                          rdm
GroupURL        /$CURYEAR/*         usage

EOF

# Setup Meta-HTML engine
cp -p /opt/metahtml/bin/ispengine $HTTPDROOTDIR/$1/cgi-bin/nph-engine

# Configure Meta-HTML
cat >$HTTPDROOTDIR/$1/cgi-bin/engine.conf << EOF
;;; engine.conf: -*- Meta-HTML -*-  Typical startup file for (nph-)Engine.
;;;
;;; Copyright (c) 1996 Brian J. Fox
;;; Author: Brian J. Fox (bfox@ai.mit.edu) Sat Sep 14 03:51:33 1996.

;;; You set the variable MHTML::DOCUMENT-ROOT to tell the engine
;;; where your documents are being served from.  If the location
;;; of the engine is "/usr/local/metahtml/doc/cgi-bin/nph-engine", then
;;; the default document root is "/usr/local/metahtml/doc".  This may or
;;; may not be correct for your site.  Simply set the variable here if
;;; it is not.
<set-var mhtml::document-root = $HTTPDROOTDIR/$1/htdocs>

;;; The user that you would like the engine to run under.
;;; This only has an effect if the engine is already running as the
;;; root user.
;;; <set-var mhtml::default-user = nobody>

;;; If you wish to allow your users to store documents underneath
;;; their home directories, then set this variable to be the name of
;;; the subdirectory under which users store their documents.  For
;;; example, if the URL "/~bfox/welcome.mhtml" should be served from
;;; a subdirectory of Brian Fox's home directory called "public_html",
;;; then you would do:
;;;
;;; <defvar mhtml::~directory  public_html>

;;; Use cookies that appear permanent to the client.
<set-var mhtml::permanent-cookies = true>

;;; Top level cgi-directory.  Actually an array of possible cgi directories.
```

```
<set-var mhtml::cgi-directories[0]=/cgi-bin/>

;;; Analogous to MHTML::PROLOGUE-DOCUMENT is MHTML::PER-PAGE-FUNCTION.
;;; You probably only need one or the other, but not both.  If this
;;; variable is set to the name of a defun, defsubst, or defmacro,
;;; then that function is called with no arguments.  Here is the
;;; default that we recommend -- it makes all posted data innocuous.
;;;
;;; If you turn this off, you will have to make sure that your pages
;;; are written in a secure fashion.  Please see the manual
;;; description of <get-var-once> for more details.
;;;
<defun mhtml::engine-per-page-function>
  <mhtml::post-secure>
   ;;;
   ;;; Code that sets up the URL rewriter.  This has to be a per-page function
   ;;; because we haven't computed the URL of the document at the time
   ;;; this engine.conf file is read, so variables such as mhtml::http-prefix
   ;;; aren't set yet, and will be changed at the time the page location is
   ;;; computed.
   <when <get-var rewriter::engine-webpath>>
     <set-var
       mhtml::http-prefix = <get-var mhtml::http-prefix
                                     rewriter::engine-webpath>
       mhtml::http-prefix-sans-sid = <get-var mhtml::http-prefix-sans-sid
                                          rewriter::engine-webpath>
       mhtml::url-to-dir = <get-var mhtml::http-prefix mhtml::relative-prefix>
       mhtml::url-to-dir-sans-sid = <get-var mhtml::http-prefix-sans-sid
                                         mhtml::relative-prefix>
       mhtml::full-url = <get-var mhtml::http-prefix mhtml::location>>
     <copy-var *meta-html*::rewriter::a *meta-html*::a>
     <copy-var *meta-html*::rewriter::img *meta-html*::img>
     <copy-var *meta-html*::rewriter::form *meta-html*::form>
   </when>
</defun>

<set-var mhtml::per-page-function = mhtml::engine-per-page-function>
<set-var mime-type::.asp=metahtml/interpreted>

EOF

# Configure virtual host in Apache
cat >$APACHECONFDIR/$1 << EOF
################################################################################
##
## Virtual host: $1
##
<VirtualHost $IPADDRESS>
    ServerName $1
    DocumentRoot $HTTPDROOTDIR/$1/htdocs
    ErrorLog $HTTPDROOTDIR/$1/logs/error.log
    CustomLog $HTTPDROOTDIR/$1/logs/access.log combined env=!whistle
    Alias /usage "$HTTPDROOTDIR/$1/usage"

    <Directory "$HTTPDROOTDIR/$1/htdocs">
       Options -ExecCGI -Indexes -Includes FollowSymLinks
       AllowOverride AuthConfig FileInfo
       Order allow,deny
       Allow from all
    </Directory>

    <Directory "$HTTPDROOTDIR/$1/usage">
       Options -ExecCGI -Indexes -Includes FollowSymLinks
       Order allow,deny
       Allow from all
    </Directory>
```

```
    ScriptAlias /cgi-bin/ $HTTPDROOTDIR/$1/cgi-bin/
</VirtualHost>

EOF

# Add usage/index.html file
cat >$HTTPDROOTDIR/$1/usage/index.html << EOF
<html>
<body bgcolor=white>

<h3>Statystyki serwera HTTP</h3>

<ul>
  <li><a href="$CURRENTYEAR">$CURRENTYEAR</a>
</ul>

</body>
</html>

EOF

# Update vhost registry file
cat >>$VHOSTLIST << EOF
$1      $NOW
EOF

# Restart Apache
/etc/init.d/apache restart
```

httpvhostmod

```
#!/bin/sh
#
# Reconfigure HTTP virtual host.
#
# Last modified: May 23, 2003
#
# Copyright (c) 2003 Adam Piwnikiewicz & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

SELF=`basename $0`

APACHECONFDIR=/opt/etc/httpd/vhosts
VHOSTLIST=/opt/etc/httpd/vhostlist

NOW=`date +"%Y-%m-%d %H:%M"`

# Check input parameters
if [ $# -ne 1 ]; then
  echo "Usage: $SELF virtual_server_name"
  exit 1;
fi

$EDITOR $APACHECONFDIR/$1

# Update vhost registry file
sed "s/^\($1     [-0-9: ]*\).*/\1         $NOW/" $VHOSTLIST > /tmp/vhlist.tmp
mv /tmp/vhlist.tmp $VHOSTLIST

# Restart Apache
/etc/init.d/apache restart
```

httpvhostdel

```
#!/bin/sh
#
# Remove HTTP virtual host.
#
# Last modified: Jun 26, 2003
#
# Copyright (c) 2003 Adam Piwnikiewicz & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

SELF=`basename $0`

HTTPDROOTDIR=/export/home1/httpd
APACHECONFDIR=/opt/etc/httpd/vhosts
VHOSTLIST=/opt/etc/httpd/vhostlist

NOW=`date +"%Y-%m-%d %H:%M"`

# Check input parameters
if [ $# -ne 1 ]; then
  echo "Usage: $SELF virtual_server_name"
  exit 1;
fi

# Backup vhost content and its configuration
gtar czpf $1.tgz -C $HTTPDROOTDIR $1 -C $APACHECONFDIR/.. vhosts/$1

# Remove complete directory structure
rm -rf $HTTPDROOTDIR/$1

# Remove virtual host configuration file in Apache dir
rm -f $APACHECONFDIR/$1

# Update vhost registry file
sed "s/^\($1 .*\)/\1 $NOW/" $VHOSTLIST > /tmp/vhlist.tmp
mv /tmp/vhlist.tmp $VHOSTLIST

# Restart Apache
/etc/init.d/apache restart
```

processhttpdlogs

```
#!/bin/sh
#
# Rotate Apache logs. Run at most daily.
#
# Last modified: May 23, 2003
#
# Copyright (c) 2003 Adam Piwnikiewicz & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

HTTPDROOTDIR=/export/home1/httpd
DATE=`date '+%Y-%m-%d'`

# Stop httpd daemon
/etc/init.d/apache stop >/dev/null

# Rotate logs
for d in $HTTPDROOTDIR/*
do
    (cd $d/logs
        mv access.log access.tmp
        mv error.log error-$DATE.log)
done

# Restart httpd daemon
/etc/init.d/apache start >/dev/null

# Gather reverse DNS lookups
for d in $HTTPDROOTDIR/*
do
    (cd $d; /opt/bin/webazolver -Q -N 8)
done

# Generate statistics
for d in $HTTPDROOTDIR/*
do
    (cd $d; /opt/bin/webalizer -Q)
done

# Rename & gzip log files
for d in $HTTPDROOTDIR/*
do
    (cd $d/logs
        mv access.tmp access-$DATE.log
        gzip -q access-$DATE.log
        gzip -q error-$DATE.log)
done

exit 0
```

processhttpdlogs-ny

```
#!/bin/sh
#
# Rotate Apache logs at the end of the year.
#
# Last modified: May 23, 2003
#
# Copyright (c) 2003 Adam Piwnikiewicz & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

HTTPDROOTDIR=/export/home1/httpd
DATE=`date '+%Y-%m-%d'`
CURYEAR=`date +%Y`
NEWYEAR=`expr $CURYEAR + 1`

# Stop httpd daemon
/etc/init.d/apache stop >/dev/null

# Rotate logs
for d in $HTTPDROOTDIR/*
do
    (cd $d/logs
        mv access.log access.tmp
        mv error.log error-$DATE.log)
done

# Restart httpd daemon
/etc/init.d/apache start >/dev/null

# Gather reverse DNS lookups
for d in $HTTPDROOTDIR/*
do
    (cd $d; /opt/bin/webazolver -Q -N 8)
done

# Generate statistics
for d in $HTTPDROOTDIR/*
do
    (cd $d; /opt/bin/webalizer -Q)
done

# Rename & gzip log files
for d in $HTTPDROOTDIR/*
do
    (cd $d/logs
        mv access.tmp access-$DATE.log
        for f in access-$CURYEAR-*.gz; do
            gunzip $f;
        done
        for f in access-$CURYEAR-*.log; do
            cat $f >> access-$CURYEAR.log
            rm $f
        done
        gzip -qf access-$CURYEAR.log
```

```
        for f in error-$CURYEAR-*.gz; do
            gunzip $f;
        done
        for f in error-$CURYEAR-*.log; do
            cat $f >> error-$CURYEAR.log
            rm $f
        done
        gzip -qf error-$CURYEAR.log)
done


for d in $HTTPDROOTDIR/*
do
    (cd $d
        # Create new directory for statistics
        mkdir usage/$NEWYEAR

        # Modify Webalizer configuration
        sed "s/$CURYEAR/$NEWYEAR/g" webalizer.conf > /tmp/webalizer.conf.tmp
        mv /tmp/webalizer.conf.tmp webalizer.conf

        # Modify usage/index.html
        sed '/<\/ul>/i\
  <li><a href=\"NEWYEAR\">NEWYEAR</a>' usage/index.html \
            | sed "s/NEWYEAR/$NEWYEAR/g" > /tmp/index.html.tmp
        mv /tmp/index.html.tmp usage/index.html
    )
done

exit 0
```

Spis literatury

[1] Apache HTTP Server, `http://Apache.org`.

[2] Apache Virtual Host documentation, `http://httpd.apache.org/docs/vhosts/index.html`.

[3] Hypertext Transfer Protocol – HTTP/1.0, `http://www.w3.org/Protocols/rfc1945/rfc1945`.

[4] Hypertext Transfer Protocol – HTTP/1.1, `http://www.w3.org/Protocols/rfc2068/rfc2068`.

[5] Solaris 9 4/03 System Administrator Collection, System Administration Guide: Advanced Administration, Scheduling System Tasks (Tasks), `http://docs.sun.com/db/doc/816-4553/6maop1hbg?a=view`.