Apache serwer www

oraz programy związne

Maciej Mazur

Oleksak Sławomir

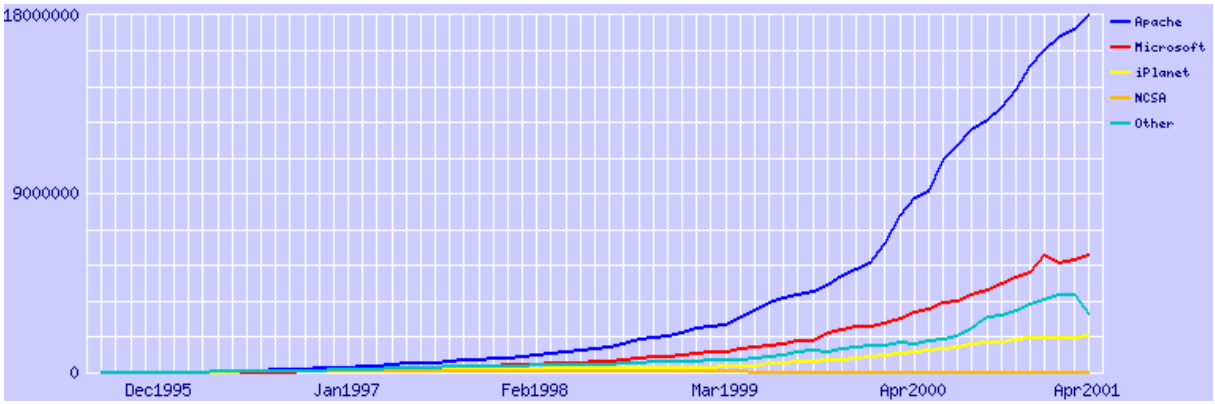
Wstęp

Przeciętny użytkownik internetu zazwyczaj korzysta jedynie z dwóch usług z bogatej oferty jaką daje internet. Pierwszą rzeczą jest poczta elektroniczna, natomiast drugą są strony www. Najczęściej nikt nie zastanawia się jak to się dzieje, że po wpisaniu adresu otrzymujemy na ekranie monitora odpowiednią stronę, a jeśli już to kręg zainteresowań kończy się na języku HTML w którym napisana jest strona WWW. Jednakże bardzo interesującą sprawą jest program który pracując na serwerze odpowiada na żądania użytkowników i wysyła im odpowiednie dane. Demony http w dzisiejszych czasach są bardzo skomplikowanymi programami które muszą nie tylko wysyłać pliki ale również wykonywać skrypty lub programy CGI, programy PHP itp., odpowiadać za bezpieczeństwo przesyłanych danych, oraz wiele innych rzeczy. Dodatkowo demon taki musi być bezpieczny, gdyż aby pracować na porcie 80 musi mieć prawa administratora co powoduje ze jest ulubionym celem ataków hakerów.

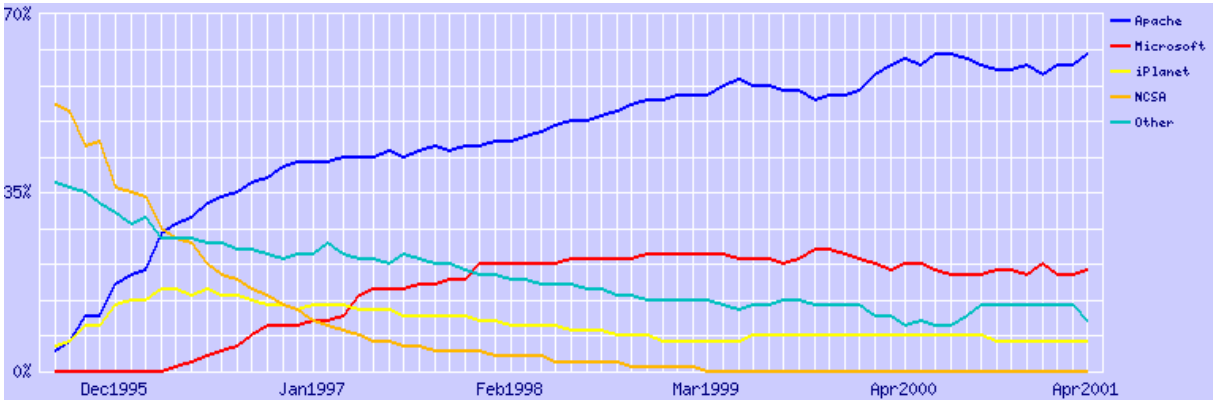
Porównanie serwerów

Dostępnych jest wiele różnych serwerów www, przy czym praktycznie monopolistą jest Apache.
Według netcraft (www.netcraft.com) w kwietniu 2001 Apache miało druzgocącą przewagę nad kolejnym w stawce Internet Information Serwer firmy microsoft.

Serwer	Marzec 2001	udział	Kwiecień 2001	udział	zmiana
Apache	17238004	60.25%	17932251	62.55%	2.30%
Microsoft-IIS	5648960	19.74%	5916724	20.64%	0.90%
Netscape-Enterprise	1750429	6.12%	1762872	6.15%	0.03%
Zeus	738068	2.58%	779209	2.72%	0.14%



Ilość działających serwerów WWW na świecie



Podział rynku przez serwery WWW.

iPlanet jest sumą ilości następujących serwerów iPlanet-Enterprise, Netscape-Enterprise, Netscape-FastTrack, Netscape-Commerce, Netscape-Communications, Netsite-Commerce, oraz Netsite-Communications.
Na sukces Apache złożyło się przede wszystkim to, iż jest on bardzo dobrym serwerem www. Dodatkowo jest to serwer darmowy co powoduje, że w połączeniu z systemem Linux można bardzo małym kosztem postawić profesjonalny serwer WWW. Kolejną zaletą Apache jest to, że istnieją wersje dla praktycznie wszystkich systemów operacyjnych. Oprócz najpopularniejszych czyli unixów, poprzez windows, na beos kończąc. Dodatkowo Apache uchodzi za jeden z najbezpieczniejszych programów działających pod unixami, a dziury w Apache są znajdywane niezwykle rzadko.
Oczywiście aby demon pracował stabilnie dobrze było by aby również system operacyjny nie powodował niespodzianek, więc większość Apache'ow działa na unixach najczęściej Linux oraz BSD, oczywiście serwer microsoftu pod kontrolą windows

Skoro Apache jest wiodącym serwerem www, w dalszej części pracy zostanie opisany właśnie on, chyba ze zostanie zaznaczone, że wspomniany został inny produkt.

Serwer Apache

Apache jest serwerem WWW, powstałym na bazie NCSA httpd 1.3 (przede wszystkim posiada poprawione wiele błędów, a także wiele lepszą wydajność). Nazwa jest pewnego rodzaju skrótem od słów "A PAtCHy server", z czego wynika, że w dużej mierze do jego powstania przyczyniły się "łatki" (patche) stworzone dla NCSA httpd. W dokumentacji podano również, że powodem stworzenia APACHE'a były pewne obawy co do polityki licencyjnej NCSA. Apache jest (i w zamierzeniach twórców ma pozostać) produktem darmowym.
Pierwsza publicznie dostępna wersja Apache ujrzała światło dzienne w kwietniu 1995, była to wersja 0.6.2
W kwietniu 2001 administratorzy mają do wyboru dwie alternatywne wersje Apache, jest to wersja 1.3.19, oraz 2.0.16 beta. Wersja z serii 1.3.x jest na tyle przetestowana, że bez obaw można na niej polegać, natomiast nowa generacja z serii 2.0.x funkcjonuje na razie w fazie beta, jednakże jest już na tyle stabilna, że główna strona www.apache.org korzysta już z nowej wersji.

Kompilacja Apache

Kompilacja Apache'a zawiera się w trzech krokach: Na początku wybieramy moduły Apache'a jakie chcemy załączyć do serwera. Następnie tworzymy konfigurację odpowiednią dla Naszego systemu i kompilujemy
Apache'a.

Wszystkie konfiguracje Apache'a dokonywane są w katalogu `src` dystrybucji Apache'a.

a. Wybieramy moduły które chcemy wkompilować do Apache'a w pliku konfiguracyjnym. Odkomentowujemy linie odpowiadające za moduły które chcemy załączyć (między liniami Modułów na końcu pliku), albo dodajemy nowe linie odpowiadające za moduły dodatkowe. Zaawansowani użytkownicy mogą zakomentować domyślnie ustawione moduły jeżeli są pewni, że nie będą ich potrzebowali (należy być jednak ostrożnym, wiele z domyślnych modułów jest potrzebnych do prawidłowej i bezpiecznej pracy serwera). Należy również przeczytać instrukcje w pliku konfiguracyjnym jeżeli będziesz chciał ustawić Rule linie.
Ustawień można również dokonać podając odpowiednie opcje do programu configure.

b. Konfigurujemy Apache'a odpowiednio do maszego systemu operacyjnego. Normalnie można uruchomić skrypt konfigurujący (Configure) aczkolwiek gdyby nie zadziałał lub mamy specjalne wymagania (np. chcemy załączyć odpowiednią bibliotekę do modułu)

możemy być zmuszeni do edycji jednej lub więcej niżej podanych opcji w pliku konfiguracyjnym (Configure): EXTRA_CFLAGS, LIBS, LFLAGS, INCLUDES.

Uruchamiamy skrypt konfiguracyjny:

```
% Configure
Using 'Configuration' as config file
 + configured for <whatever> platform
 + setting C compiler to <whatever> *
 + setting C compiler optimization-level to
<whatever> *
%
```

(*: Depending on Configuration and your system, Configure make not print these lines. That's OK).

Powyższy skrypt generuje plik Makefile który jest potrzebny w trzecim kroku kompilacji. Skrypt ten generuje również plik Makefile w katalogu pomocniczym, potrzebnym do kompilacji opcjonalnych programów pomocniczych.

(Jeżeli potrzebujemy używać kilku konfiguracji, możemy podać opcje przy Configure i wskazać własny plik konfiguracyjny, np. Configure -file Configuration.ai)

c. Wykonujemy polecenie

```
make
```

Moduły które zostały umieszczone w dystrybucji Apache'a są przetestowane i regularnie używane przez członków Apache development group. Dodatkowe moduły rozprowadzane przez członków lub dodatkowe moduły ze szczególnymi potrzebami albo funkcjami są dostępne pod http://www.apache.org/dist/contrib/modules/ . Umieszczone są tam instrukcje na stronie łączącej te moduły w główny kod Apache'a.

Kompilacja Programów Zawartych w Dystrybucji

Apache zawiera sporą liczbę programów, które nie są domyślnie kompilowane. Znajdują się one w katalogu support dystrybucji. Aby skompilować te programy, przechodzimy do tego katalogu i piszemy

```
make
```

Instalacja

Powinniśmy mieć plik binarny o nazwie httpd w katalogu `src`. Binarna dystrybucja Apache'a powinna zawierać ten plik.

Następny krok to instalacja i konfiguracja. Apache jest zaprojektowany tak aby był konfigurowany i uruchamiany w tym samym katalogu w którym został skompilowany. Jeżeli

chcemy uruchamiać go z innego miejsca, tworzymy katalog i kopiujemy `conf`, `logs` i `icons` do tego katalogu. Można także przenieść wszystkie potrzebna pliki do wybranej lokalizacji domyślnie (/usr/local/apache). Aby tego dokonać piszemy:

```
make install
```

Następny krok to edycja pliku konfiguracyjnego serwera. Polega to na ustawieniu różnych katalogów w trzech centralnych plikach konfiguracyjnych. Domyślnie pliki te są umieszczone w katalogu `conf` i są to:
`srm.conf`, `acces.conf` i `httpd.conf`. Aby pomóc Ci zacząć konfigurację w katalogu conf dystrybujci Apache'a znajdują się pliki, `srm.conf-dist`, `acces.conf-dist` i `httpd.conf-dist`.
Skopiuj lub zmień nazwy tych plików na nazwy bez końcówki -dist. Następnie przeedytuj każdy z plików. Czytaj uważnie komentarze zawarte w każdym pliku. Nieodpowiednie ustawienie tych plików prowadzi do złej pracy serwera lub niezabezpieczonej w odpowiedni sposób pracy serwera. Powinieneś również mieć plik `mime.types` w katalogu conf.

Pierwszy edytuj `httpd.conf`. W pliku typ ustawia się główne atrybuty serwera, numer portu, uruchamianie jako użytkownik itp. Następnie przejdź do edycji pliku `srm.conf`; tutaj ustawia się główny katalog przechowywanych dokumentów html, specjalne funkcje takie jak server-parsed HTML lub internal imagemap parsing, itp. Na koniec edytuj plik access.conf i ustaw podstawowe prawa dostępu.

Dodatkowo oprócz tych trzech plików, pracę serwera można ustawić poprzez plik .htaccess w każdym z katalogów do których serwer ma dostęp.

Uruchamianie

Aby wystartować serwer po prostu uruchamiamy httpd. Httpd odczyta plik konfiguracyjny httpd.conf znajdujący się tam gdzie podano w czasie kompilacji (domyślnie jest to /usr/local/Apache/conf/httpd.conf).
Jeżeli plik ten znajduje się w innym miejscu możesz podać prawdziwą ścieżkę dostępu z argumentem -f np.

```
/usr/local/Apache/bin/httpd -f /etc/Apache/conf/httpd.conf
```

Jeżeli wszystko pójdzie dobrze natychmiast wrócimy do linii komend. Oznacza to, że serwer jest już podniesiony i działa. Jeżeli jednak pójdzie coś źle podczas inicjalizacji serwera na ekranie pojawi się informacja o błędzie. Jeżeli serwer już działa, możesz użyć przeglądarki www aby połączyć się z serwerem i przeczytać dokumentację. Jeżeli uruchamiamy przeglądarkę www na tym samym
komputerze gdzie uruchomiony jest serwer i używa on standardowo portu 80, stosowny URL jaki powinieneś podać przeglądarce jest:

```
http://localhost/
```

Uwaga, kiedy serwer zostanie uruchomiony utworzy odpowiednią liczbę procesów child do zarządzania i kierowania prośbami połączeń. Jeżeli uruchomiliśmy Apache'a jako użytkownik root proces parent będzie kontynuowany do uruchomienia jako root podczas gdy children zmienią użytkownika jak podano w pliku httpd.conf.

Jeżeli uruchomimy httpd i będzie on wskazywał, że nie jest w stanie "połączyć" się z określonym adresem to będzie to wskazywało, że port który podaliśmy w czasie konfiguracji Apache'a jest wykorzystywany przez inny proces, lub uruchamiamy httpd jako zwykły użytkownik który próbuje używać portu poniżej 1024 (domyślnie jest ustawiony port 80).

Jeżeli serwer nie uruchomi się, czytamy informacje o błędzie która zostaje wyświetlona w czasie uruchamiania httpd. Należy także sprawdzić plik error_log aby uzyskać dodatkowe informacje. (w domyślnej konfiguracji znajduje się on w katalogu logs).

Jeżeli chcemy aby serwer uruchamiał się po restarcie systemu, należy dodać httpd do swoich plików startowych (normalnie są to rc.local lub plik w katalogu rc.N). To powinno wystartować Apache'a jako użytkownik root.

Httpd jest zwykle uruchamiany jako demon który wykonuje ciągłą kontrolę nad prośbami połączeń. Jest to możliwe przez wywoływanie Apache'a przez demaona internetowego inetd za każdym razem gdy jest wykonywane połączenie do serwisu HTTP, jednak nie jest to zalecane.

Opcje linii komend

Następujące opcje są rozpoznawane przez httpd:

`-d serverroot`
Ustawia początkową wartość dla ServerRoot zmienną dla serverroot. To może być z góry ustawione w pliku konfiguracyjnym. Domyślnie jest /usr/local/Apache/httpd.

`-f config`
Wykonuje polecenia zawarte w config przy uruchamianiu. Jeżeli config nie istnieje w katalogu /, to brana jest ścieżka dostępu dotycząca ServerRoot'a. Domyślnie jest conf/httpd.conf.

`-X`
Uruchamia httpd w trybie pojedynczego procesu, przeznaczone jest to do wewnętrznego debbugingu; demon nie sprawdza czy pochodzi to z terminala czy od któregoś z procesów children. NIE używaj tej funkcji jeżeli prowadzisz zwykły serwis www.

`-v`
Wyświetla numer wersji httpd i kończy działanie.

`-h`
Podaje listę poleceń łącznie z wymaganymi argumentami i miejscem gdzie należy je podać aby były ważne. (Nowość w Apache'u 1.2)

`-l`
Podaje listę wszystkich modułów wkompilowanych w serwerze.

`-?`
Wyświetla listę opcji httpd i kończy działanie.

Pliki konfiguracyjne

Serwer powinien czytać trzy pliki konfiguracyjne. Jakakolwiek komenda konfiguracyjna powinna znaleźć się w jednym z tych plików. Nazwy tych plików zależą od konkretnych ustawień serwera; Ustawiane są one poprzez ServerRoot lub poprzez parametr -d w linii komend. Przyjęte nazwy tych plików to :

conf/httpd.conf
Zawiera parametry kontrolujące pracę demona serwera. Nazwa może być podana w linii komend, służy do tego parametr -f.
conf/srm.conf
Zawiera parametry kontrolujące specyfikację dokumentów które serwer dostarcza dla klienta. Nazwa może być pobrana z ResourceConfig.
conf/access.conf
Zawiera wskazówki na temat dostępu do dokumentów. Nazwa może być pobrana z AccessConfig.

Serwer również czyta plik zawierający mime types; plik ten ustawia się poprzez TypesConfig i domyślnie znajduje się w conf/mime.types.

Logi

plik pid

W czasie uruchamianie demona, zapisuje on numer procesu http parent do pliku logs/httpd.pid. Plik ten może być zmieniony z parametrem PidFile. Numer procesu wykorzystywany jest przez administratora do
restartowania i zatrzymywania demona. Sygnał HUP lub USR1 powoduje odczytanie plików konfiguracyjnych a sygnał TERM powoduje zatrzymanie pracy demona. Po więcej informacji zajrzyj do Zatrzymywanie
u uruchamianie Apache'a.

Jeżeli proces nie zostanie zatrzymany w sposób normalny, potrzebne będzie zatrzymanie procesów children httpd.

Logi z błędami

Serwer domyślnie zapisuje wiadomości z błędami w logu, logs/error_log Nazwa pliku może być ustawiona poprzez użycie parametru z ErrorLog; można ustawić osobne logi dla wirtualnych hostów

Log Transfer

Serwer standardowo zapisuje każdą prośbę o przesłanie pliku, domyślnie jest to logs/access_log. Nazwa może być ustawiona z parametrem TransferLog. można ustawić osobne logi dla wirtualnych hostów.

- Aby zatrzymać Apache'a należy wysłać do procesu parent sygnał TERM. PID tego procesu jest zapisany w pliku httpd.pid w katalogu logs (chyba, że masz inaczej skonfigurowane).

Nie należy kilować procesu child ponieważ będzie on odnowiony przez proces parent. Typowa komenda zatrzymująca serwer to:

```
kill -TERM 'cat /usr/local/Apache/logs/httpd.pid'
```

Wysyłanie sygnału TERM do procesu parent powoduje natychmiastową próbę zatrzymania wszystkich własnych procesów children. Może to zająć kilka sekund aby zatrzymać wszystkie własne procesy children. Następnie proces parent zatrzymuje sam siebie. Wszystkie prośby połączenia w czasie zatrzymywania i potem są odrzucane.

- Wysyłanie sygnału HUP do procesu parent powoduje zatrzymanie procesów children podobnie jak sygnał TERM ale nie zatrzymuje samego procesu parent. Następuje ponowne odczytanie plików konfiguracyjnych, otwarcie logów, uruchomienie nowych procesów children i kontynuacja pracy serwera.
Użytkownicy modułu status module powinni zaobserwować że statystyka serwera zostaje wyzerowana, kiedy zostanie wysłany sygnał HUP.
Jeżeli Twoje pliki konfiguracyjne zawierają błędy i wydasz polecenie zrestartowania Twój proces parent nie zrestartuje się i zakończy błędnie pracę. Zobacz niżej metodę która zapobiega temu.

- Sygnał USR1 powoduje, że proces parent zawiadamia proces children aby zakończył pracę po skończeniu odpowiadania na prośby połączeń (lub do natychmiastowego zakończenia pracy jeżeli dany proces nie serwuje żadnej usługi). Proces parent odczytuje ponownie pliki konfiguracyjne i otwiera logi. Jeżeli jeden z procesów child zakończy pracę, proces parent uruchomi nowy proces child w miejsce poprzedniego. Nowy proces child rozpoczyna pracę natychmiast po wywołaniu go.

Użytkownicy modułu status module powinni zaobserwować, że statystyka serwera nie zostaje wyzerowana podczas wysłania sygnału USR1. Kod ten został napisany aby zminimalizować czas w którym serwer nie jest aktywny i nie odpowiada na żądania połączeń (połączenia powinny być kolejkowane przez system, także nie powinny nigdzie zaginąć) a także żeby respektował Twoje polecenia które mają usprawnić działanie serwera. W czasie wykonywania kodu, utrzymywany jest scoreboard wykorzystywany do utrzymania ścieżek między wszystkimi procesami children.

Moduły

W systemach UNIX istnieje mechanizm zwykle zwany dynamicznym łączeniem/ładowaniem modułów (Dynamicznie Dzielonych Obiektów - ang Dynamic Shared Objects /DSO/) który umożliwia budowanie części programów w specjalnym formacie, które to części można później ładować w czasie wykonywania.
Takie ładowanie można zazwyczaj wykonać na 2 sposoby: Automatycznie przez systemowy program zwany ld.so podczas startu programu, lub ręcznie, kiedy program zechce załadować sobie moduł poprzez funkcje systemowe dlopen()/dlsym().
Moduły (biblioteki dynamiczne) ładowane pierwszym sposobem znajdują się najczęściej w katalogu /usr/lib mają rozszerzenie so lub so.1.2 gdzie numery są numerem wersji. W trakcie tego ładowania system dba o poprawne przypisanie nazw do miejsca w pamięci (resolving symbols).

Drugim sposobem moduły zazwyczaj znajdują się w katalogu danej aplikacji, mogą mieć dowolne rozszerzenia. W tym przypadku program musi samodzielnie wykonać funkcję dlsym(), która pozwoli na zlokalizowanie w pamięci załadowanego kodu.
Apache używa drugiego sposobu ładowania. Praktycznie cały serwer może zostać załadowany z modułów oprócz dwóch. mod_so, oraz http_core. Pierwszy odpowiada właśnie za ładowanie modułów, natomiast drugi jest rdzeniem serwera.

Aby korzystać z modułów należy przy kompilacji Apache podać opcję do konfiguratora --enable-shared, lub dodać AddModule w pliku Configure

Aby uprościć tworzenie modułów dla Apache, powstał program apxs (APache eXtenSion). Jest on używany do budowania modułów dodatkowych, nie dostarczanych razem z serwerem. Umożliwia on tworzenie modułów nawet bez dostępnych źródeł Apache.

Zalety modułów:

- serwer jest bardziej elastyczny, ponieważ można wybrać te cechy które chce się aby występowały, natomiast usunąć niewykorzystywane. Powoduje to, iż oprogramowanie jest mniejsze i bardziej dopasowane do potrzeb.
- serwer może być łatwo rozszerzony o dodatkowe moduły nawet po instalacji.
- program apxs umożliwia tworzenie modułów nawet bez źródeł Apache

Wady modułów:

- dynamiczne ładowanie do pamięci nie jest możliwe we wszystkich typach systemów operacyjnych
- serwer jest około 20% wolniejszy przy starcie, oraz około 5% wolniejszy podczas wykonywanie na niektórych platformach, gdyż niekiedy trzeba dodatkowych readresacji które są niepotrzebne przy statycznym linkowaniu.

Moduły umieszczone w standardowej dystrybucji serwera Apache wersja 1.3.x

Core - Podstawowe funkcje zawsze dostępne w dystrybucji (kontrolują również inne moduły)
`mod_access` - Kontrola dostępu do plików w zależności od adresu IP i/lub nazwy komputera klienta. Użycie tego modułu pozwala na dokładną kontrolę użytkowników, np. administrator może zezwolić na wykonywanie skryptów CGI tylko pracownikom firmy.
`mod_actions` - Odpowiada za wykonywanie skryptów CGI w zależności od typu danych lub sposobu pobrania.
`mod_alias` - Pozwala mapować (udostępniać) część systemu plików w katalogu głównym Apache'a, umożliwia też przekierowanie adresów URL. Część plików może znajdować się poza katalogiem lub nawet na innym komputerze w sieci.
`mod_asis` - Deklaracja plików, które mogą być wysyłane bez nagłówków HTTP (pliki *.asis).
`mod_auth` - - Moduł odpowiedzialny za uwierzytelnianie użytkowników na podstawie zdefiniowanych plików tekstowych.
`mod_auth_anon` - Pozwala anonimowym użytkownikom na dostęp do danych podlegających weryfikacji dostępu.

mod_auth_db - Uwierzytelnianie za pomocą plików DB (Berkeley).
mod_auth_dbm - Uwierzytelnianie za pomocą plików DBM.
mod_auth_digest - Uwierzytelnianie za pomocą MD5
mod_autoindex - Automatyczne tworzenie indeksów (wyświetlenie zawartości) dla katalogów, które nie mają standardowych plików index.*htm*
mod_cern_meta - Emulacja plików CERN HTTPD, pozwala dodawać dodatkowe nagłówki do wszystkich plików.
mod_cgi - Prawdopodobnie najpopularniejszy moduł. Pozwala wykonywać skrypty CGI po stronie serwera i zwracać wyniki klientowi.
mod_digest - Uwierzytelnianie za pomocą algorytmu MD5.
mod_dir - Podstawowe operacje na katalogach. Zwykle używany do uzupełniania adresu, np. http://serwer.pl/plik zostanie zastąpiony poprawnym wywołaniem http://serwer.pl/plik/.
mod_env - Moduł odpowiedzialny za przekazywanie zmiennych środowiskowych do skryptów CGI/SSI.
mod_example - Demonstracja możliwości interfejsu programowego, Apache API.
mod_expires - Dodaje znacznik Expires (strona wygasa, traci ważność) do stron WWW przesyłanych klientowi, ważne dla często zmienianych serwisów, które powinny być zawsze aktualne.
mod_headers - Pozwala na dowolną modyfikację nagłówków HTTP.
mod_imap - Wsparcie dla map plików graficznych (.map), używane po stronie serwera WWW. Zastępuje program CGI imagemap.
mod_include - Pozwala włączać zawartości plików lub wyniki działania skryptu do zwykłych plików HTML i zwracać ich zawartość klientowi.
mod_info - Odpowiedzialny za informację o ustawieniach serwera Apache.
mod_isapi - Pozwala używać rozszerzeń serwerowych ISAPI. Tylko dla Apache'a w wersji dla Windows.
mod_log_agent - Zapisywanie w logach nazw i wersji przeglądarek internetowych klientów.
mod_log_config - Konfigurowalne logowanie zdarzeń, pliki log zapisywane są w formacie Common Logfile Format.
mod_log_referer - Logowanie odwołań do plików umieszczonych na serwerze.
mod_mime - Określenie typu pliku na podstawie rozszerzenia.
mod_mime_magic - Określenie typu pliku na podstawie kilku bajtów jego zawartości.
mod_mmap_static - Pozwala określić pewne niezmienne pliki, które zostaną umieszczone w pamięci serwera Apache w celu szybszego dostępu.
mod_negotiation - Odpowiedzialny za uzgadnianie najlepszej reprezentacji danych w przeglądarce klienta. Wprowadzony ze względu na zgodność z HTTP/1.1
mod_proxy - Apache staje się serwerem proxy dla stron WWW, przyspiesza dostęp do często używanych danych, gdy serwer WWW (komputer) jest wykorzystywany do zapamiętywania danych.
mod_rewrite - Moduł o ogromnych możliwościach. Pozwala modyfikować adresy URL „w locie" za pomocą wyrażeń regularnych.
mod_setenvif - Pozwala modyfikować zmienne środowiskowe na podstawie wywołania (danych klienta).
mod_so - Ładowanie dodatkowych modułów podczas działania serwera.
mod_speling - Moduł odpowiedzialny za poprawianie pomniejszych błędów w adresach URL.

`mod_status` - Wyświetla bieżący stan serwera Apache.
`mod_userdir` - Ustawienia dotyczące katalogów domowych użytkowników.
`mod_unique_id` - Generuje unikalny identyfikator dla każdego żądania.
`mod_usertrack` - Śledzenie zachowania użytkowników za pomocą Cookies (tzw. ciasteczka), szczególnie przydatne dla np. stałych klientów w sklepach internetowych lub do określania preferencji użytkownika.
`mod_vhost_alias` - Dynamiczna konfiguracja wirtualnymi hostami

CGI

Standard interfejsu służącego wymianie informacji między serwerami WWW a zewnętrznymi programami. CGI to najstarszy sposób tworzenia interaktywnych stron WWW i dostępu do baz danych za pomocą przeglądarki WWW.

CGI pozwala realizować to, czego nie można osiągnąć za pomocą samego tylko języka HTML i protokołu HTTP. Na podstawie wywołania ze strony WWW serwer uruchamia zewnętrzny program i przekazuje do niego parametry (wartości zmiennych, rodzaj przeglądarki, adres klienta, itp) otrzymane z przeglądarki użytkownika. W kolejnym kroku program przetwarza otrzymane parametry i zwraca wynik w postaci strony HTML, która wysyłana jest do klienta. Dzięki takiemu rozwiązaniu możliwa jest interakcyjna wymiana danych na drodze serwer-przeglądarka i tworzenie dynamicznie zmieniających się stron WWW.

Najpoważniejszą wadą programów CGI jest ich niewielka wydajność. Każde wywołanie CGI ze strony WWW zmusza serwer do uruchomienia programu, przekazania parametrów, odczekania na zakończenie programu, pobrania i wysłania wyników jego działania. Czasochłonność wykonania tych wszystkich operacji sprawia, iż programy CGI tworzone są w postaci niewielkich modułów realizujących proste, pojedyncze zadania. W przypadku systemów opierających się na wielu programach CGI dochodzą do tego trudności z synchronizacją pracy poszczególnych modułów. Z tych powodów stosowanie CGI zalecane jest jedynie w przypadku mniejszych projektów.

Programy lub skrypty CGI tworzy się przy pomocy popularnych języków programowania, mogą to być języki kompilowane, jak również interpretowane. Warunkiem jest, aby dane były pobierane i wysyłane z zachowaniem reguł określonych przez standard CGI oraz aby kod wynikowy (w przypadku programów kompilowanych) lub skrypt można uruchomić w systemie, pod kontrolą którego pracuje host.

Domyślnie nie jest możliwe uruchamianie programów CGI w domyślnej konfiguracji Apache, należy lekko zmodyfikować pliki konfiguracyjne.

W pliku /usr/local/Apache/httpd.conf odkomentowujemy lub wpisujemy jeśli nie istnieje następującą linię:

```
AddHandler cgi-script .cgi
```

To sprawi, że serwer będzie obsługiwał CGI. Następnie należy dodać alias cgi, by zamiast ścieżki /usr/local/Apache/cgi-bin/ móc podawać /cgi-bin/ np. http://127.0.0.1/cgi-bin/. Dodajmy do tego samego pliku następującą linię.

```
Alias /usr/local/Apache/cgi-bin/ /cgi-bin/
```

Teraz musimy pozwolić na wykonywanie skryptów CGI z katalogu /usr/local/Apache/cgi-bin/. W pliku access.conf powinny się znaleźć te oto linie:

```
<Directory /usr/local/Apache/cgi-bin>
     AllowOverride None
     Options ExecCGI
</Directory>
```

Aby sprawdzić czy CGI są wykonywanie na naszym serwerze, stąd możesz ściągnąć dwa skrypty wyświetlające informacje o systemie. Pobierz: Test-CGI i PrintENV. Oba są dystrybuowane z Apachem, ale w niektórych źródłach może ich nie być. Umieszczamy je w katalogu cgi-bin, nadajemy prawa dostępu na 755 (chmod 755), uruchamiamy przeglądarkę i wpisujemy http://127.0.0.1/cgi-bin/printenv. Jeśli wyświetlą się nam informacje o naszym komputerze, wszystko gra! Bardzo częstą pomyłką, dzięki której skrypty nie działają jest zła ścieżka do perla. Wydaj polecenie which perl i sprawdź czy ścieżka podana w skrypcie jest taka sama jak u Ciebie. Jeśli nie musisz ją poprawić.

Skrypty i programy CGI można dzisiaj spotkać w wielu witrynach - jest to pierwsza i tym samym dobrze opanowana przez programistów technologia. Jednak pojawienie się nowych, wydajniejszych techniki tworzenia dynamicznych witryn internetowych zepchnęło CGI na dalszy plan. Obecnie coraz częściej wykorzystywane są konkurencyjne PHP, osadzony Perl, ASP natomiast CGI sporadycznie stosuje się jeszcze w mniejszych projektach.

PHP

SSI (Server Side Include)
Technika tworzenia dynamicznych stron WWW, w której obiekty strony (tekst, grafika, itp.) dodawane są przez serwer "w locie" tuż przed wysyłaniem dokumentu do klienta.

PHP: Hypertext Preprocessor jest językiem skryptowym działającym po stronie serwera i zagnieżdżonym w kodzie HTML. Tak więc cały kod PHP umieszczony jest pomiędzy kodem html, a wykonywany na serwerze. Znakomicie nadaje się do zbierania danych z formularzy, generowania dynamicznie zmieniających się stron www, operowanie na cookies (ciasteczkach) oraz przedstawiania i operacji na bazach danych. Ponadto PHP ma wsparcie dla innych protokołów sieciowych np. pop3.

kompilacja PHP
PHP statyczne i dynamiczne

PHP możemy skompilować na dwa sposoby: pierwszy, statyczny, na stałe wbudowany w serwer www (w naszym wypadku w Apache) i dynamiczny jako moduł ładowany przez serwer www. Znacznie korzystniejszym rozwiązaniem jest drugi sposób, gdyż pozwala na lepszą kontrolę nad PHP i przy kompilacji nowej wersji PHP nie jest wymagana rekompilacja serwera Apache.

Dynamiczne kompilowanie PHP 4 dla Apache.

Zalecam taką kompilację. Potrzebujemy serwera Apache zainstalowanego albo z pakietu rpm, albo skompilowanego ręcznie, co też zaraz uczynimy. Ważne jest, aby Apache był skompilowany z modułami http_core.c i mod_so.c, dlatego nie wyłączajmy ich przy konfiguracji!

Wchodzimy do katalogu ze źródłami PHP i wydajemy polecenia:

```
./configure --with-mysql --with-pgsql --with-
apxs=/usr/local/Apache/bin/apxs
```

--with-mysql i --with-pgsql wydajemy tylko wtedy gdy chcemy mieć wsparcie PHP dla którejś z powyższych baz danych. --with-apxs=/usr/local/Apache/bin/apxs określa gdzie w naszym systemie znajduje się apxs, jeśli instalowaliśmy Apacha z rpm ścieżka ta będzie inna.

Kompilujemy PHP:

```
make
make install
```

Teraz edytujemy plik konfiguracyjny serwera Apache (httpd.conf). Dodajemy w nim linie:

```
LoadModule PHP4_module libexec/libPHP4.so
AddModule mod_PHP4.c
AddType application/x-httpd-PHP .PHP
```

Uruchamiamy ponownie serwer Apache.

Statyczne kompilowanie PHP 4 z Apache.

Najpierw musimy skompilować PHP ze wsparciem dla apacha. Przechodzimy do katalogu ze źródłami Apache i wydajemy polecenie:

```
./configure
```

Dalej przechodzimy do katalogu ze źródłami PHP i wydajemy polecenia:

```
./configure --with-mysql --with-pgsql \
```

```
            --with-Apache=ścieżka/do/źródeł/Apache \
            --enable-track-vars
        make
        make install
```

Wracamy teraz do źródeł Apache i wpisujemy:

```
        ./configure                                    --activate-
module=src/modules/PHP4/libPHP4.a
        make
        make install
```

Plik libPHP4.a nie istnieje, ale powyższa linia jest jak najbardziej prawidłowa. Plik ten zostanie "automagicznie" utworzony.

Przechodzimy do źródeł PHP i kopiujemy plik PHP.ini-dist:

```
        cp PHP.ini-dist /usr/local/lib/PHP.ini
```

Następnie edytujemy plik konfiguracyjny Apache (httpd.conf lub srm.conf) i dopisujemy jedna linię:

```
        AddType application/x-httpd-PHP .PHP
```

Uruchamiamy serwer Apache

Statyczne kompilowanie PHP 3 z Apache.

Najpierw musimy skompilować PHP ze wsparciem dla Apacha. Przechodzimy do katalogu ze źródłami PHP i wydajemy polecenie:

```
        ./configure --with-Apache=/ścieżka_do_źródeł_apacha
```

Inne opcje konfiguracyjne nie są nam potrzebne, więc pozostawiamy je standardowo. Teraz kompilujemy i instalujemy PHP:

```
        make
        make install
```

Teraz musimy skopiować plik inicjujący aplikacje:

```
        cp PHP3.ini-dist /usr/local/lib/PHP.ini
```

Mamy już PHP, ale musimy go jeszcze wrzucić do Apacha. Kompilujemy go z zaznaczeniem, że ma używać modułu PHP, który w tej chwili leży w źródełkach Apacha w katalogu src/modules/PHP3 i nosi nazwę libPHP3.a. (dla tych co nie czytali Apache+cgi w

konfiguracji podamy też pasującą nam ścieżkę gdzie Apache ma leżeć na dysku). Wydajmy polecenie z katalogu ze źródełkami Apacha:

```
./configure --prefix=/usr/local/Apache \
      --activate-module=src/modules/PHP3/libPHP3.a
```

Następnie znowu kompilujemy i instalujemy Apacha.

```
make
make install
```

Teraz trzeba go skonfigurować do interpretowania PHP. Odkomentujmy w pliku httpd.conf linię :

```
AddType application/x-httpd-PHP3 .PHP3
```

Działa?
Uruchom serwer i sprawdź czy PHP4/PHP3 działa pisząc prostą stronę PHP, w której zagnieżdżony skrypt, wyświetli tekst: Cześć! Jestem skryptem PHP!

```
<html>
     <head>
          <title>Strona testowa PHP</title>
     </head>
     <body>
          <?PHP echo "Cześć! Jestem skryptem PHP!";
            phpinfo();
          ?>
     </body>
</html>
```

Zapisujemy ten plik pod nazwą z rozszerzeniem .PHP3 dla PHP3 i z rozszerzeniem .PHP dla PHP4, władu gdzieś na serwer np. do własnego public_html, otwórz przeglądarkę i wpisz http://127.0.0.1/~login/nazwa.rozszerzenie. Jeśli wyświetli się prawidłowy tekst (bez znaczników!), oraz tabele z informacjami na temat serwera oraz PHP to znaczy, że PHP działa!

SuExec

Cecha suEXEC umożliwia uruchamianie CGI oraz SSI z innym identyfikatorem użytkownika, niż ten z którym pracuje serwer. Użytkownicy będą mieć taki sam identyfikator jaki mają oni sami. Możliwość taka istnieje tylko w systemach operacyjnych które mają zaimplementowane operacje setuid oraz setgid.

Wykorzystanie tej możliwości może zredukować niebezpieczeństwo jakie niesie uruchamianie programów z prawami serwera, jednakże niewłaściwa konfiguracja suExeca może spowodować liczne problemy oraz stworzyć lukę w bezpieczeństwie całego systemu.

Niebezpieczeństwa jakie niesie uruchamianie skryptów które mają id serwera powoduje, że złośliwy użytkownik może zniszczyć skrypty innych użytkowników, pokasować im dane które wykorzystują do swoich stron www itp.

Cała idea polega na zastosowaniu programu który działa z prawami roota. Apache kiedy dostaje żądanie uruchamiania programu CGI, uruchamia wcześniej ten dodatkowy program który następnie uruchamia właściwy program już z prawami użytkownika.

Konfiguracja oraz instalacja suExeca.

Istnieją dwie metody konfiguracji tego programu jedną jest edycja plików nagłówkowych, a później instalacja binariów w odpowiednie miejsce ręcznie, drugą możliwością jest instalacja z pomocą konfiguratora (APACI APache AutoConf-style Interface)

Opcje konfiguratora:

`--enable-suexec` - Opcja ta aktywuje mechanizm suExec, która nie jest instalowana domyślnie, przy wybraniu tej opcji musi być wybrana przynajmniej jedna opcja `--suexec-xxxx`
`--suexec-caller=UID` - Podanie identyfikatora użytkownika z którym pracuje normalnie Apache.
`--suexec-docroot=DIR` - Podanie korzenia drzewa katalogów w którym znajdują się pliki www serwowane przez Apache
`--suexec-logfile=FILE` - Opcjonalne podanie pliku do logwania dla suExeca, normalnie program loguje do domyślnych plików używanych przez serwer.
`--suexec-userdir=DIR` - Podanie katalogu w domowym drzewie użytkownika w którym znajdującej się programy będą uruchamiane przez suExec, domyślnie public_html
`--suexec-uidmin=UID` - Podanie minimalnego numeru UID na jaki suExec może zmienić prawa wykonywania. Należy podać najniższy numer ID z pośród numerów użytkowników, dla większości systemów stosuje się 100, albo 500
`--suexec-gidmin=GID` - Podanie minimalnego numeru GID na jaki suExec może zmienić prawa wykonywania. Najczęściej stosuje się 100.
`--suexec-safepath=PATH` - Podanie ścieżki do bezpiecznych programów, z których mogą korzystać CGI. Domyślnie "/usr/local/bin:/usr/bin:/bin".

Po kompilacji i domyślnej instalacji w katalogu /usr/local/Apache/sbin/ powinien pojawić się program suexec. Należy wówczas sprawdzić, czy jego właścicielem jest root, oraz czy ma ustawioną flagę suid

Prawidłowe uruchomienie się Apache z suidem owocuje pojawieniem się w logach następującej linijki:

[notice] suEXEC mechanism enabled (wrapper: /katalog/w/którym/znajduje/sie/suexec)

Jedynym sposobem aktywowania suExec'a jest użycie dyrektyw User oraz Group w definicji VirtualHost. Należy ustawić tam wartości różne od tych z którymi pracuje serwer. Jeśli nie ma tych wpisów w plikach konfiguracyjnych, wtedy używane jest ID serwera.

Dla CGI w katalogach domowych użytkowników nie ma specjalnych dyrektyw konfiguracyjnych. suExec będzie działał jeśli jest możliwość uruchamiania skryptów dla użytkowników.

SSL

SSL (Secure Sockets Layer) - jest to protokół kryptograficzny opracowany przez firmę Netscape. Początkowo stosowany tylko do przeglądarek i serwerów WWW, w dniu dzisiejszym jest najpopularniejszym sposobem na szyfrowanie informacji w internecie, wykorzystywanym także w innych programach korzystających z protokółu TCP/IP. Protokół SSL rozszerza standardowe metody transmisji o uwierzytelnianie klienta oraz serwera, zapewnienie poufności i integralności przesyłanych danych. (szyfrowanie oraz "skróty wiadomości").

Opiera się on o technologie certyfikatów. Certyfikat jest to klucz publiczny oraz nazwa podmiotu (osoba lub serwer), do którego dany klucz należy a wszystko to podpisane przez CA (czyli urząd certyfikacji). Certyfikat może zawierać także dodatkowe informacje zarówno o wystawcy certyfikatu jak i jego właścicielu. Taki sposób dystrybucji klucza publicznego zapobiega podszywaniu się poprzez podkładanie własnego klucza. Urząd certyfikacji powinien być instytucją powszechnie szanowaną oraz obdarzaną zaufaniem. Za podpisanie certyfikatu CA pobiera opłaty, dlatego też można samemu podpisać certyfikat (tzw. Root Level Certificate Authority) odbywa to się jednak kosztem zmniejszonego zaufania ze strony potencjalnych użytkowników naszego serwera - ponieważ przeglądarka przed rozpoczęciem właściwej transmisji danych poinformuje użytkownika, że ma on do czynienia z certyfikatem podpisanym przez nieznanego CA.

W początkowej fazie połączenia serwer i klient wymieniają certyfikaty, oraz przy pomocy zawartych w nich kluczy publicznych, uzgadniają metodę szyfrowania - tym razem już symetryczną - oraz szyfr. W ten sposób bezpiecznie przekazywany jest klucz metody symetrycznej. Parametry te są z reguły stałe podczas całego połączenia możemy jednak wymusić ich ponowną renegocjację.

Instalacja serwera Apache z modułem SSL

Oto kolejne kroki, które należy wykonać aby zainstalować w pełni działający serwer Apache wykorzystujący mod_sll:

1. Należy pobrać najnowsze wersje mod_ssl oraz OpenSSL'a (wymaganego przez mod_ssl do działania). Najlepiej zrobić to korzystając ze stron organizacji tworzących to oprogramowanie

www.modssl.org/source/
www.openssl.org/source/

przy pobieraniu mod_ssl'a należy zwrócić uwagę na wersję modułu odpowiednią do wersji pobranego serwera Apache.

2. Następny krok to rozpakowanie pobranego oprogramowania. Najlepiej zrobić to poprzez przekopiowanie pobranych plików do katalogu /usr/src/ a następnie wydanie poniższych poleceń dla każdego z nich:

```
gzip -d nazwa_pliku.tar.gz
tar -xf nazwa_pliku.tar
```

3. Teraz kompilujemy i instalujemy OpenSSL'a: będąc w katalogu openssl-0.9.x wydajemy polecenia:

```
./config - skrypt który automatycznie utworzy Makefile zależny od zainstalowanych komponentów systemu
make - kompilacja OpenSSL
make install - instalacja skompilowanych plików
```

4. Następnie musimy odpowiednio skonfigurować Apacha (jego kompilację) tak aby wykorzystywał moduł mod_ssl, w tym celu przechodzimy do katalogu zawierającego źródła mod_ssl'a i wydajemy polecenie: (wygodnie jest pomagać sobie klawiszem [tab] w celu dokańczania nazw katalogów)

```
./configure --with-Apache=../Apache_1.3.x --with-ssl=../openssl-0.9.x --prefix=/usr/local/Apache
```

5. Teraz przechodzimy do katalogu ze źródłami Apacha, mamy tutaj skonfigurowany (wcześniejszym poleceniem) serwer Apache, gotowy do kompilacji, której dokonujemy poleceniem:

```
make
```

Certyfikat z którego możemy korzystać tylko do testów generujemy poleceniem:

```
make certificate
```

Nie ma sensu zastanawiać się nad wpisywaniem poprawnych wartości w poszczególne pola ponieważ i tak za chwile wygenerujemy nowy certyfikat odpowiedni dla naszego serwera. Ostatnim poleceniem jest:

```
make install
```

Teraz mamy gotowy do pracy serwer Apache (zainstalowany w katalogu /usr/loacal/Apache), można usnąć źródła poleceniami:

```
rm -rf nazwa_katalogu
```

Jak wygenerować certyfikat SSL dla serwera Apache + mod_ssl?

Openssl jest zbiorem narzędzi służącym do wygenerowania klucza prywatnego serwera, żądania podpisania certyfikatu czy też samodzielnego podpisania certyfikatu. Powinien być zainstalowany w katalogu /usr/local/ssl/bin. Należy samodzielnie dopisać ten katalog do zmiennej PATH lub używać pełnej ścieżki dostępu przy uruchamianiu polecenia openssl.

Aby wygenerować klucz publiczny serwera wydajemy polecenie:

```
openssl genrsa -des3 -out server.key 1024
```

zostaniemy poproszeni o podanie hasła, które będzie chroniło nasz klucz prywatny serwera oraz o powtórne wpisanie go w celach bezpieczeństwa. Jako wynik otrzymamy plik server.key zawierający 1024 bitowy klucz prywatny. Wpisane hasło będziemy musieli podać przy każdorazowym starcie serwera Apache co jest trochę niewygodne, nie pozwala nam na zdalny restart maszyny. Możemy zrezygnować z hasła chroniącego klucz prywatny usuwając z polecenia dyrektywę -des3. Musimy być jednak pewni, że plik z kluczem będzie dostępny tylko dla root'a oraz, że serwer jest wystarczająco dobrze zabezpieczony przed włamaniem. Przejęcie przez kogoś tego klucza spowoduje, że nasz certyfikat stanie się całkowicie bezużyteczny.

Kolejne polecenie:

```
openssl req -new -key server.key -out server.csr
```

wygeneruje nam plik server.csr zwany Certificate Sign Request, który przesyłamy do urzędu certyfikacji, który podpisuje go własnym kluczem prywatnym podnosząc tym samym wiarygodność klucza publicznego naszego serwera. (o urzędach certyfikacji można poczytać w dalszej części tego dokumentu).

W trakcje tworzenia CSR będziemy musieli podać odpowiedź na kilka pytań z czego najważniejszym (choć inne też są ważne dla CA - urzędu certyfikacji) jest pytanie o Common name. Jako odpowiedź należy podać nazwę domenową serwera, czyli taką jaka jest zawarta w pliku httpd.conf jako ServerName oraz taką jaką użytkownicy będą wpisywali w przeglądarce.

Do czasu weryfikacji naszych danych przez CA możemy samodzielnie podpisać certyfikat. Spowoduje to że serwer będzie działał prawidłowo, ale przeglądarka użytkownika poinformuje go o fakcie iż certyfikat jest podpisany przez urząd nie należący do grupy "zaufanych". Dokonujemy tego poleceniem:

```
openssl x509 -req -days 60 -in server.csr -signkey server.key -out server.crt
```

Jeżeli nie mamy zamiaru korzystać z urzędu certyfikacji tylko chcemy samodzielnie podpisać certyfikat całą procedurę możemy sprowadzić do jednego polecenia:

```
openssl req -new -x509 -out server.crt -keyout server.key
```

aby utworzyć klucz prywatny nie zabezpieczony hasłem należy dodać do powyższego polecenia dyrektywę -noodes.

Instalacja certyfikatu serwera sprowadza się do przegrania plików z kluczem i certyfikatem gdzieś do drzewa Apache/conf i dokonania odpowiedniej zmiany w pliku hhtpd.conf w sekcji

SSL w wierszach SSLCertificateKeyFile i SSLCertificateFile (muszą się tam znajdować ścieżki i nazwy odpowiednio plików z kluczem i z certyfikatem)

Urzędy Certyfikacji.

Urzędy Certyfikacji (CA - Certificate Authority) są instytucjami podpisującymi nasze klucze publiczne celem nadania im wiarygodności. Są to instytucje powszechnie uznane jako godne zaufania. Te najbardziej popularne są już standardowo wpisane do przeglądarek internetowych obsługujących połączenia szyfrowane. Możemy przejrzeć listę urzędów dostępnych w danej przeglądarce (oczywiście obsługującej protokół https). I tak dla Internet Explorera należy wybrać narzędzia->opcje internetowe a następnie w zakładce zawartość znajdziemy przycisk certyfikaty. Dla Netscape Comunicatora (4.x) możemy je obejrzeć naciskając przycisk Security, a następnie klikając łącze Certificates.

Celem uzyskania certyfikatu musimy przesłać CSR (jak wygenerować ten plik można przeczytać powyżej) oraz inne wymagane dokumenty przez urząd, na podstawie których zostanie potwierdzona nasza tożsamość i wiarygodność. średni czas oczekiwania na podpis jest w granicach jednego tygodnia do dwóch a sam certyfikat jest ważny przez rok.

Koszt uzyskania certyfikatu w organizacji Thawte wynosi około 125 $ dla zwykłego szyfrowania oraz 300 $ przy wykorzystaniu mocniejszej kryptografii.

W Polsce certyfikacją kluczy publicznych zajmuje się np. Centrum Certyfikacji (www.certum.pl/pl/) koszt uzyskania certyfikatu jest zależny od prowadzonej przez nas działalności i tak uzyskanie certyfikatu dla serwera nie komercyjnego jest darmowe. Dla zwykłego serwera kosztuje 500 PLN a dla serwera obsługującego transakcje finansowe 1000 PLN (mocniejsza kryptografia - podobnie jak w Thawte). Przedłużenie certyfikatu po jego wygaśnięciu kosztuje połowę tego co jego wydanie.

Instytucja ta nie znajduje się w grupie zaufanych CA dla przeglądarek takich jak Internet Explorer czy Netscape Navigator ale zawsze certyfikat taki jest bardziej wiarygodny niż podpisany samemu (root level certificate). Aczkolwiek można go dodać do listy zaufanych aby pozbyć się każdorazowego samodzielnego potwierdzania certyfikatu.

Negocjacja zawartości

Możliwości negocjacji zawartości zostały dodane do Apache w związku z uaktualnieniem protokołu HTTP do wersji 1.1.
Daje to możliwości wyboru najlepszej reprezentacji zasobu na podstawie ustawień przeglądarki internetowej dla danych typów, języków naturalnych, kodowania oraz zestawu znaków. Dodatkowo jest kilka cech aby umożliwić bardziej inteligentne realizowanie żądań przeglądarki które nie zawierają informacji o "negocjacji zawartości"

Negocjacja zawartości jest realizowana przez moduł mod_negotiation który jest kompilowany domyślnie.

Zasób może być dostępny w kilku różnych reprezentacjach. Na przykład mogą być dostępne różne wersje językowe, typy plików multimedialnych, itp.

Jedną drogą wyboru zasobu jest umieszczenie linków do wszystkiego i pozwolenie internaucie wyboru. Jest jednak możliwość wyboru tego przez serwer automatycznie. Jest to możliwe ponieważ przeglądarki mogą wysłać jako część każdego żądania informacje na temat preferowanej reprezentacji zasobu. Przykładowo przeglądarka może zaznaczyć ze najlepiej jeśli strona będzie w języku polskim, a jeśli to nie możliwe to w angielskim. Protokół przewiduje, ze takie informacje umieszczane są w nagłówku żądania. Dla przykładu:
Accept-Language: pl
Oczywiście takie żądanie będzie zrealizowane tylko wówczas jeśli jest dostępna możliwość przesłania strony w języku polskim.
Przykładem bardziej złożonego żądania może być mieszanka różnych typów i języków.
Accept-Language: pl; q=1.0, en; q=0.5
Accept: text/html; q=1.0, text/*; q=0.8, image/gif; q=0.6, image/jpeg; q=0.6, image/* 0.5, */*; 0.1

Apache obsługuje następujące typy negocjacji:
Accept: - typy
Accept-Language: - język
Accept-Charset: - zestaw znaków
Accept-Encoding: - kodowanie

Od wersji 1.3.4 Apache obsługuje także eksperymentalny protokół negocjacji opisany w RFC 2295 oraz RFC 2296

Dostęp do zasobów:
Zasób jest definiowany jako plik do którego dostęp jest przez URL.
Apache zapewnia dostęp do zasobów na podstawie nazwy. Każda reprezentacja zasobu ma swoją nazwę dla przykładu:
foo.en.html
foo.html.en
foo.html - język domyślny (niesprecyzowany)

Proxy i cache

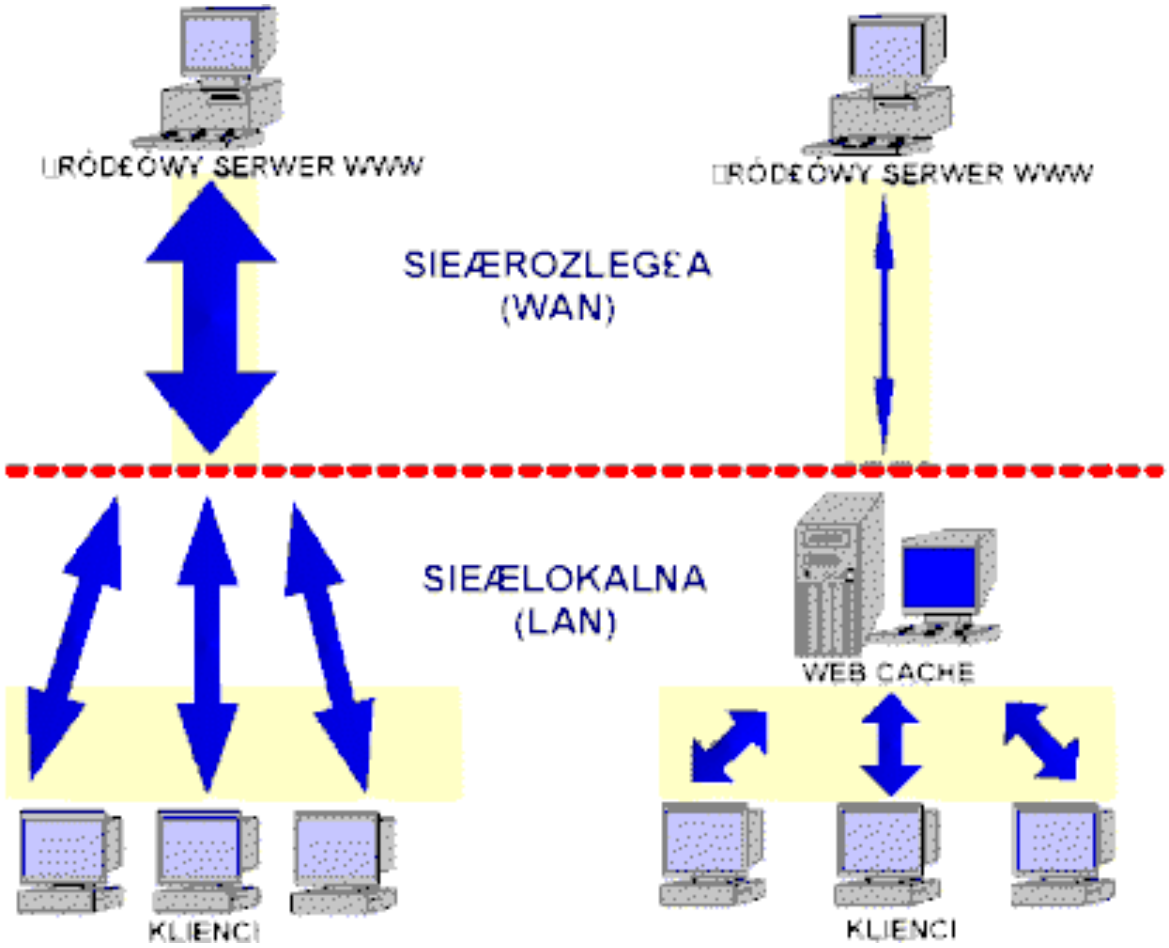
Serwer proxy jest to serwer pośredniczący w komunikacji pomiędzy klientem a serwerem WWW. Serwer proxy może spełniać dwie role. Jedna to taka że cachujący serwer proxy tworzy bazę danych ostatnio używanych dokumentów i przy następnym zapytaniu klienta o dokument który znajduje się w tej bazie, serwer proxy może przesłać do klienta przechowywany dokument bez kontaktowania się po raz kolejny z serwerem WWW. Druga rola serwera proxy jest taka, że stanowi on logiczne odizolowane od siebie serwera WWW i klienta, w ten sposób możemy zabezpieczyć się przed niepożądanymi próbami połączeń, a także odizolować własna sieć lokalną po przez postawienie firewalla.

Serwer proxy w internecie może działać w dwie strony jako "forward" i "revers" proxy serwer, lub też może działać jako te dwa typy serwera jednocześnie.
Forward proxy serwer działa w ten sposób że retransmituje zapytania od klientów z sieci wewnętrznej do serwerów WWW znajdujących się na zewnątrz tej sieci, a także zapisuje (cachuje) strony z serwerów WWW, zmniejszając w ten sposób ruch wychodzący i zmniejszając obciążenie naszego łącza ze światem.

Rewers proxy serwer natomiast przekazuje zapytania od klientów z zewnątrz to serwerów WWW znajdujących się w sieci lokalne, jest to użyteczne wtedy gdy chcemy trochę odciążyć nasz serwer WWW od częstych zapytań, cachując najczęściej odwiedzane strony przez klientów. W ten sposób oczywiście możemy też postawić nasze serwery za firewallem i ograniczyć do nich dostęp, co w pewnym stopniu chronić je będzie przed włamaniami.

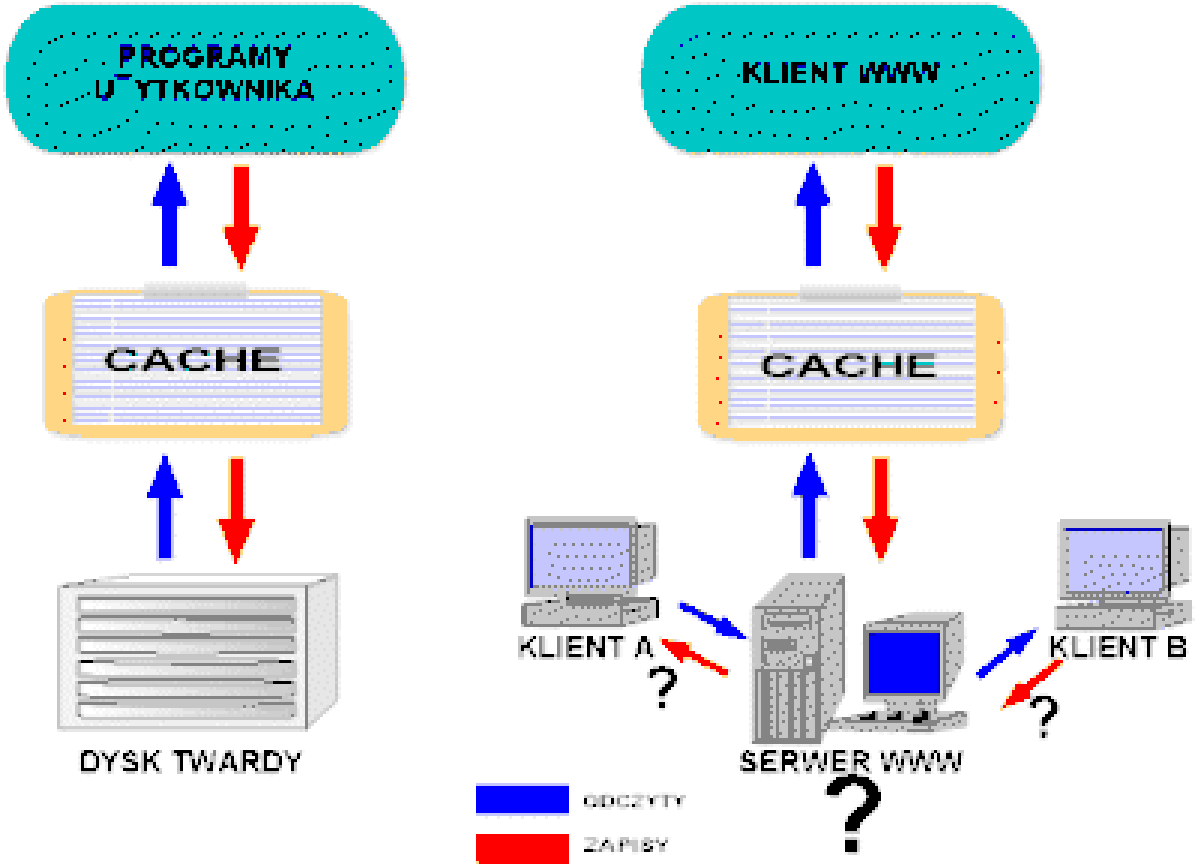
Nietrudno zauważyć, że środowisko sieciowe stwarza dodatkowe możliwości dla pamięci podręcznych. Połączenie oprogramowania proxy z web cache daje dodatkowy efekt. Przeglądarka zamiast kierować połączenie do serwera źródłowego, kieruje je do serwera pośredniczącego - w naszym wypadku nie tylko przekazującego żądaną transakcję do właściwego serwera WWW, ale również inteligentnie buforującego zapytania. Poprzez współdzielenie zasobów w obrębie jednego, wspólnego serwera, ich wykorzystanie jest dużo lepsze.

Ideę web cachingu przedstawia poniższy rysunek:



Idea web cache: serwer web cache mający połączenie z klientami szybkim łączem sieci lokalnej pośredniczy w wymianie danych z serwerami WWW skracając czas dostępu do odległych zasobów i zmniejszając obciążenie połączeń ze światem.
Zysk wydaje się być oczywisty, chociaż - jak się zaraz okaże - osiągany jest kosztem pewnych, nie do końca czystych kompromisów. Porównajmy środowisko pracy podręcznej pamięci dyskowej i web cache'a.

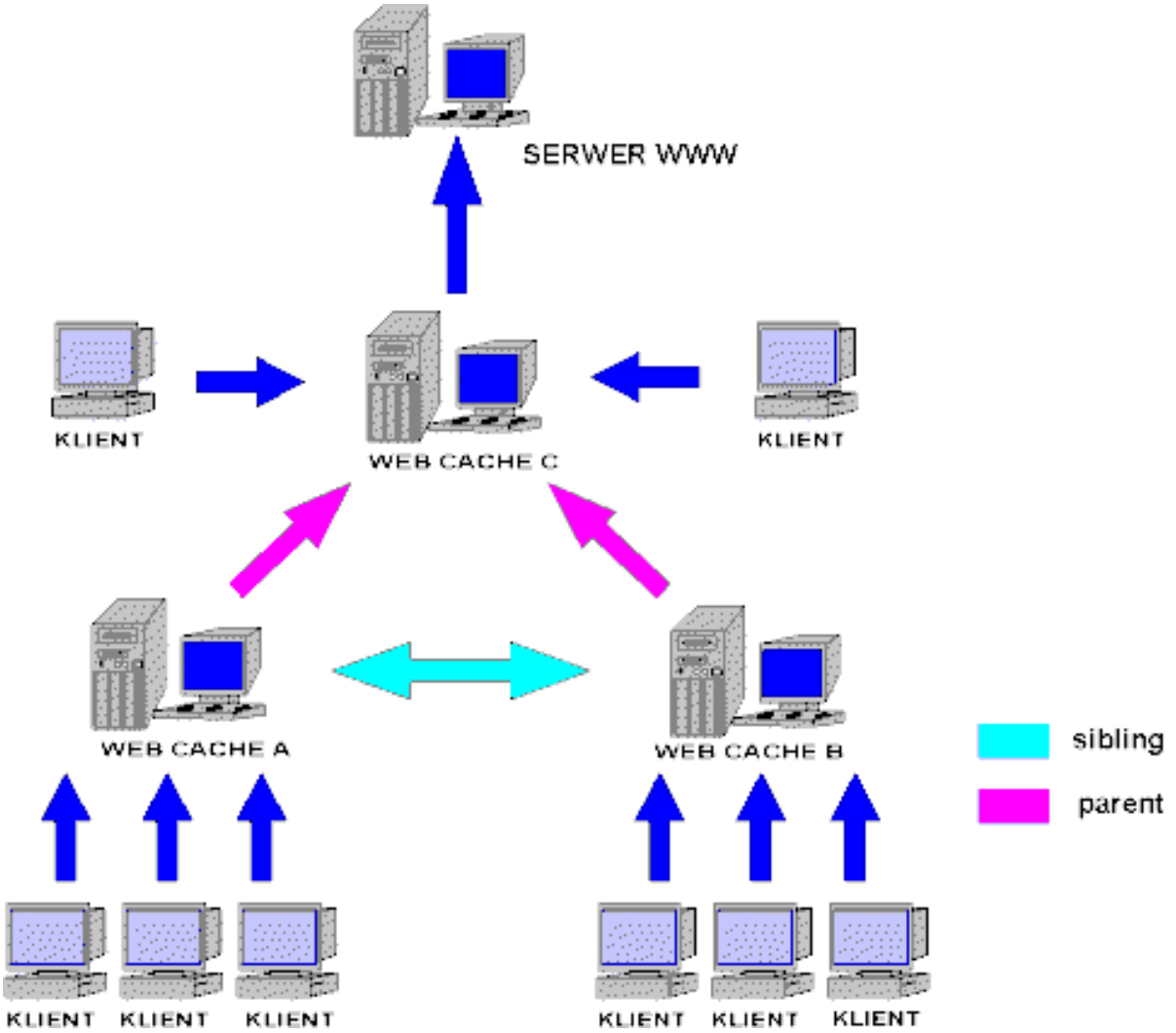
Przedstawia to następujący rysunek:



Porównanie środowisk pracy pamięci podręcznej systemu plików i web cache; web cache pracuje w środowisku nie będącym pod jego całkowitą kontrolą, stąd niebezpieczeństwo niezgodności jego zasobów z oryginałem. Nie można mieć żadnej pewności, że przechowywana lokalnie kopia odpowiada oryginałowi, ponieważ ten mógł od momentu skopiowania ulec zmianie. W chwili obecnej nie istnieje żaden mechanizm replikacji danych zapewniający stuprocentową zgodność tych obiektów - poza każdorazowym sprowadzeniem obiektu z serwera źródłowego, co przeczy samej idei cachingu. Ustalenie punktu równowagi pomiędzy ilością odwołań do oryginalnego obiektu a aktualnością jego kopii to jeden z podstawowych problemów web cachingu. Istnieją już oczywiście algorytmy, które zapewniają rozsądny kompromis. Jest to tylko przybliżone rozwiązanie tego problemu.

Serwery web cache potrafią współpracować i wymieniają ze sobą informacje na temat posiadanych przez siebie lokalnie zasobów.

W jaki sposób się to dzieje przedstawia to mniej więcej poniższy rysunek:



Przedstawione są tu różnego rodzaju relacje między serwerami cachującymi. Relacja parent (rodzic), a także sibling (rodzeństwo)

Serwery A i B znajdujące się relatywnie blisko siebie i mniej więcej w tej samej odległości od zasobów światowych są dla siebie rodzeństwem, natomiast serwer C znajduje się bliżej zasobów światowych, dlatego dla obu pozostałych instalacji jest on rodzicem. Relacje odzwierciedlają wymianę informacji pomiędzy serwerami na temat aktualnie przechowywanych w ich pamięciach podręcznych kopii obiektów WWW oraz wymianę samych obiektów.

Wyjaśnijmy tu pokrótce różnicę między rodzicami a rodzeństwem. Serwer web cache może zaspokoić żądanie klienta na kilka sposobów: z własnej pamięci podręcznej, z pamięci podręcznej któregoś z serwerów kooperujących (czy może skoligaconych? jeśli trzymać się przyjętej terminologii) oraz z serwera źródłowego. Wydaje się być oczywiste, że do transakcji między serwerami dochodzi tylko wtedy, gdy jeden z nich posiada zasób interesujący z punktu widzenia klienta innego cache'a. Jest to prawda, ale tylko dla relacji typu sibling. Relacja typu parent jest podobna do tej, jaka istnieje między przeglądarką WWW a serwerem

proxy lub web cache - ponieważ parent pośredniczy w transakcjach - także wtedy, gdy nie posiada żądanego obiektu w swojej pamięci podręcznej.

Używając relacji parent można kierować ruchem WWW w sposób niezależny od routingu - daje to dodatkowe możliwości zarządzania ruchem. Z tego powodu web caching bywa również określany jako application level routing.

Serwery web cache potrafią współpracować i wymieniają ze sobą informacje na temat posiadanych przez siebie lokalnie zasobów.

Internet Cache Protocol to pierwszy zaawansowany i wdrożony protokół wymiany danych między serwerami web cache. Oparty o protokół transportowy UDP pracuje w schemacie pytanie-odpowiedź. Używający ICP serwer web cache nie mogąc zaspokoić żądania klienta z własnej pamięci podręcznej rozsyła zapytanie do swoich siblings i parents po czym postępuje następująco:
Jeśli otrzyma co najmniej jedną odpowiedź twierdzącą, to żądany obiekt zostaje pobrany z tego serwera, który odpowiedział najszybciej. Najszybsza odpowiedź pozwala domniemywać, że żądany obiekt, w przypadku, gdy wszystkie odpowiedzi są negatywne, obiekt jest sprowadzany za pośrednictwem tego serwera parent, który odpowiedział najszybciej, jeśli nie otrzymano odpowiedzi od parent, a tylko co najwyżej negatywne odpowiedzi od siblings, obiekt zostaje sprowadzony z serwera źródłowego.

Wady ICP: Przede wszystkim czas oczekiwania na odpowiedzi może być długi, szczególnie wtedy, gdy są to odpowiedzi negatywne - pytający musi doczekać do momentu, kiedy wszyscy zapytani “krewni” odpowiedzą lub do wyznaczonego limitu - dopiero wtedy może sprowadzać obiekt z serwera źródłowego (chyba, że użyjemy opisanej powyżej modyfikacji włączając serwer źródłowy do puli decyzyjnej). Nawet jeśli otrzymamy odpowiedź twierdzącą, to na przesłanie zapytania i odpowiedzi zużywany jest cenny czas. ICP jest poza tym dość kosztowne - każda transakcja to wysłanie kilku do kilkunastu zapytań i odpowiedzi. Im bardziej obciążony jest nasz serwer web cache - tzn. im więcej przyjmuje żądań od klientów - tym więcej pytań i odpowiedzi ICP musi obsłużyć. Ilość zapytań ICP wzrasta z liczbą serwerów kooperujących.

Zupełnie inaczej ma się rzecz w przypadku skróconych spisów treści zwanych cache digests. Jest to coś podobnego do kompresji spisu treści ze stratą - tj. możliwe jest, że na podstawie cache digest otrzymamy informację fałszywą o obecności danego obiektu w pamięci podręcznej, podczas gdy naprawdę go tam nie ma lub o jego nieobecności - podczas gdy w rzeczywistości jest. Alternatywą dla kilkuprocentowego ryzyka otrzymania błędnej informacji jest duża kompresja spisu - dla kilku gigabajtów danych jest to kilkaset kilobajtów spisu treści. Web caches wymieniają tego typu spisy treści np. co godzinę. Co prawda w ten sposób zatracają informację o tym, czy w danej chwili dany “krewny” jest dostępny, zyskują za to kilka innych cennych zalet. Korzystanie z cache digests jest po prostu szybsze - cała procedura sprawdzenia zawartości sąsiednich serwerów odbywa się lokalnie w szybko dostępnej pamięci operacyjnej, nie angażując stosunkowo wolnego medium, jakim jest sieć. Ilość informacji o treści “krewnych”, jaką nasz serwer będzie z nimi wymieniał będzie proporcjonalna do zgromadzonych zasobów, nie zaś do obciążenia serwera.

Różnorodność serwerów proxy:

Od modułu proxy serwera WWW Apache poczynając, kończąc na specjalizowanych komputerach cache box będących w istocie dedykowanymi serwerami web cache. Te ostatnie zapewne zniechęcą przeciętnego użytkownika przede wszystkim wąską specjalizacją i niezbyt przystępną ceną. Podobnie jest np. z komercyjnym oprogramowaniem Net Cache firmy Network Appliance cena raczej nie predestynuje go do zastosowań indywidualnych, edukacyjnych czy małego biznesu. Swoje produkty w tej dziedzinie mają także tacy potentaci jak IBM (Web Traffic Express), Novell (Border Manager FastCache), Intel (Quick Web), Cisco Systems (Cache Director System) czy wreszcie Netscape i Micropsoft (w przypadku obu firm produkt nosi tę samą nazwę Proxy Server). Obok dużych pakietów sporo jest również propozycji o nieco mniejszym kalibrze, w tym programów napisanych w Javie czy PERL-u.
Dobrym rozwiązaniem dla początkujących wydaje się Squid. Serwer ten jest rozwijany przez zespół National Laboratory for Applied Network Research (NLANR) i grupy niezależnych programistów. Do zalet tego oprogramowania należy niewątpliwie fakt bezpłatnego rozpowszechniania oraz wsparcie dla protokołów komunikacji między serwerami web cache: ICP, cache digests, HTCP. Ze względu na dostępność tekstów źródłowych jest on również podstawową platformą dla badaczy i twórców nowych rozwiązań technicznych w dziedzinie web cachingu, przez co zawiera implementacje najnowszych pomysłów w tej materii. Squid przeznaczony jest oficjalnie dla platform unixowych, chociaż istnieją również udane implementacje dla innych systemów operacyjnych. Oczywiste wydaje się tu wsparcie dla Linuxa i FreeBSD, co sprawia, że uruchomienie własnego serwera web cache kosztuje nas w najgorszym przypadku tyle, ile użyty do tego celu sprzęt.

Wirtualne serwery

Serwery wirtualne to metoda umożliwiająca pojedynczej instancji serwera i udostępnienie wielu adresów URI. Za pomocą serwerów wirtualnych dana instancja serwera Apache "maskuje" rzeczywiste odniesienia, co w konsekwencji daje złudzenie, iż w rzeczywistości mamy do czynienia z większą liczbą różnych serwerów dostępnych pod różnymi adresami domenowymi.

Klasyczne podejście polegało na tym, iż albo instalowało się pojedynczą instancję serwera na oddzielnych komputerach, albo też uruchamiało się wiele instancji pojedynczego serwera na jednej maszynie.

By rozwiązać te niedogodności, opracowano metodę serwerów wirtualnych w oparciu o adres IP. Przy tej metodzie dany serwer może odpowiadać na zapytania skierowane do wielu adresów IP. Z drugiej jednak strony wirtualne serwery oparte o adres IP wymagają (poza konfiguracją samego serwera Apache) umieszczenia w maszynie wielu kart sieciowych (i/lub stworzenia stosownej liczby dowiązań do już posiadanej liczby owych kart), przypisaniu im różnych adresów IP, ewentualnie ustawieniu trasowania, w końcu zaś powiadomieniu DNS-a o tym fakcie:

```
www.domena1.com    IN  A 192.168.0.1
www.domena2.com    IN  A 192.168.0.2
```

Protokół HTTP/1.1 wprowadził metodę umożliwiającą serwerowi określenie, jakiego adresu dotyczy zapytanie przeglądarki. Począwszy od wersji 1.1 serwer Apache obsługuje to

podejście, zwane zwykle metodą serwerów wirtualnych w oparciu o nazwę. Warto jednak mieć na uwadze, iż niektóre (nie tylko) co starsze przeglądarki nie będą udostępniały możliwości dostępu do serwera Apache skonfigurowanego do obsługi serwerów opartych o nazwę.

Jedną z podstawowych zalet takiego podejścia jest redukcja kosztów. Wiele serwisów WWW może być przechowywanych i obsługiwanych na jednej maszynie, przez co zmniejsza się ogólny koszt sprzętu (pamięć, karty sieciowe itd.) i/lub oprogramowania. Zamiast dedykowanego sprzętu, mamy do czynienia ze współdzieleniem tego samego sprzętu pomiędzy wiele wirtualnych miejsc.

Tak jak i poprzednio będziemy stosowali wyłącznie i tylko jeden plik konfiguracyjny, mianowicie `$APACHE/conf/httpd.conf`. Uczyńmy zatem stosowne wpisy dla naszych wirtualnych serwerów:

Przykładowo jeżeli na naszym serwerze chcemy mieć dwa serwisy WWW o adresach www.domena1.com, www.domena2.com musimy umieścić w pliku `$APACHE/conf/httpd.conf` następujący wpis:

```
NameVirtualHost 192.168.0.1:80

<VirtualHost 192.168.0.1>
ServerAdmin webmaster@mail.domena1.com
ServerName www.domena1.com
DocumentRoot /www/domena1/htdocs
TransferLog /var/log/www/access_log.domena1
ErrorLog /var/log/www/error_log.domena1
</VirtualHost>

<VirtualHost 192.168.0.1>
ServerAdmin webmaster@mail.domena2.com
ServerName www.domena2.com
DocumentRoot /www/domena2/htdocs
TransferLog /var/log/www/access_log.domena2
ErrorLog /var/log/www/error_log.domena2
</VirtualHost>
```

NameVirtualHost - dyrektywa wymagana przy konfiguracji w oparciu o metodę nazw symbolicznych. Określa ona adres do którego odnoszą się wirtualne serwery (w powyższym przykładzie 192.168.0.1). Choć adres można podawać w postaci symbolicznej zalecane jest stosowanie adresów IP. Jeśli przydziela się wirtualne serwery do wielu adresów IP, wówczas należy powtórzyć tę dyrektywę oddzielnie dla każdego adresu IP.

Uwaga: zastosowanie tej dyrektywy powoduje, iż wszystkie zapytania skierowane pod ten adres będą się odnosiły wyłącznie i tylko do serwerów wirtualnych o parametrach określonych przez `<VirtualHost ... > ... </VirtualHost>`. Trzeba więc dodać sekcję `<VirtualHost>` dla wszystkich zarządzanych serwerów, włączając w to "serwer główny".

<VirtualHost ... > ... </VirtualHost> - deklaracja ta wskazuje na początek i koniec sekcji poświęconej danemu serwerowi wirtualnemu. Atrybutem w tej sekcji jest adres wirtualnego serwera. Adresem może być adres IP lub adres symboliczny. Z pewnych względów zaleca się stosowanie adresów IP. Opcjonalnie można zastosować numer portu.

ServerName - dyrektywa przypisująca symboliczną nazwę wirtualnego serwera. Opcjonalna, choć również z pewnych istotnych względów zalecana.

Czasami istnieje potrzeba, by dany serwer wirtualny był widoczny pod wieloma nazwami (np. pajeczyna.domena1.com). Potrzeba taka może być podyktowana choćby względami marketingowymi czy przyzwyczajeniami klientów. Druga strona medalu (zwłaszcza w intranetach) jest taka, iż użytkownicy przyzwyczaili się wpisywać krótką nazwę (bez domeny, np. WWW). Rozwiązanie tego problemu stanowi dyrektywa ServerAlias. Stanowi ona ogólny sposób określania wielu nazw dla jednego i tego samego serwera wirtualnego. Przykładowo:

```
<VirtualHost 192.168.0.1>
...
ServerName www.domena1.com
ServerAlias www.pajeczyna.domena1.com
...
</VirtualHost>

<VirtualHost 192.168.0.1>
...
ServerName www.domena2.com
ServerAlias *.domena2.com
...
</VirtualHost>
```

Drugi z wpisów ilustruje zasadę, iż równie dobrze można stosować znaki globalne. Możliwość takich ustawień została wprowadzona w Apachu 1.3.13.

Dla każdego dowiązania trzeba posiadać stosowne wpisy w plikach konfiguracyjnych DNS-a.

Klastry WWW

Gdy rozmiar naszego serwisu WWW, zaczyna osiągać bardzo duże rozmiary i nie możemy sobie pozwolić na przestój serwera, to są dwie możliwości żeby poradzić sobie z tym problemem:

Proste rozbudowywanie serwera na którym jest nasz serwis WWW (takie jak dokupywanie pamięci, szybszego procesora, lub szybkiego dysku) jest często nie wystarczające i bardzo kosztowne.

Zainstalowanie większej ilości serwerów i podzielenie obciążenie generowanego przez klientów pomiędzy nich. Z racji tego iż ruch jest dzielony na

poszczególne serwery to, serwery te nie muszą być drogimi superkomputerami, tylko wystarczą komputery o niewielkich mocach obliczeniowych, adekwatnych do zadania wykonywanego przez nich.

Klastry serwerów WWW są interesującym rozwiązaniem z kilku powodów:

- Serwery te mogą być tanie więc łatwe do zastąpienia.
- Przy "padnięciu" jednego z serwerów serwis WWW nadal będzie dostępny.
- Jeżeli będziemy chcieli upgrejdować nasz system, to wystarczy dokupić jeszcze jeden serwer i nie ma potrzeby upgrejdowania czy rekonfiguracji już istniejących serwerów.

Są dwa podstawowe rozwiązania stworzenia serwerów WWW. Dzielenie obciążenia po przez konfiguracje DNS'a, lub poprzez odpowiednią konfiguracje Apache. W każdym z tych rozwiązań jest kilka metod do uzyskania pożądanego przez nas efektu.

Rozwiązania poprzez odpowiednią konfiguracje dns'a

Serwer zapasowy, poprzez przekierowanie przez Secondary DNS
Jest to najprostsze rozwiązanie z wykorzystaniem konfiguracji DNS'a. Rozwiązanie to pozwala nam na stworzenie zapasowego serwera WWW. W rozwiązaniu tym wykorzystujemy to iż zawsze mamy do dyspozycji dwa serwery DNS'a (primary i secondary). Przeważnie w obydwóch tych serwerach mamy takie same rekordy dla nazw serwerów WWW. Podstawowym założeniem w tym rozwiązaniu jest to iż primary serwer DNS jest na tym samym komputerze co nasz serwis WWW. Natomiast secondary name server na komputerze gdzie mamy zapasowy serwis WWW.
Przykładowe wpisy w Name serverach będą wyglądały wtedy następująco:

primamary name server:
```
www.alpha-complex.com. IN A 204.148.170.3
```

secondary:
```
www.alpha-complex.com. IN A 204.148.170.203
```

W normalnej sytuacji kiedy inne name-server'y żądają podania to podawany jest pierwszy numer IP, dopiero gdy z jakiś przyczyn pierwszy numer IP jest niedostępny (na przykład gdy serwer jest wyłączony), to wtedy na zapytanie DNS'a o numer IP serwera WWW odpowie secondary name server i poda drugi numer IP i klient będzie miał możliwość połączenia się z naszym serwisem WWW, nie zauważając praktycznie żadnej różnicy.
Istotną sprawą jest to żeby ustawić odpowiednio niski TTL (na przykład 30 minut) tak aby zewnętrzne serwery DNS nie trzymały w swoim cachu zbyt długo pobranego numeru IP z naszego name servera.
Używanie tego rozwiązania prowadzi za sobą wiele różnych problemów, takich jak na przykład: autoryzacja dla danego użytkownika, obsługa sesji (poprzez skrypty CGI) i co najważniejsze to, ze zapasowy serwer WWW wykorzystywany jest tylko wtedy gdy pierwszy serwer jest całkowicie niedostępny, a także jeśli przestanie działać httpd deamon a działał będzie nadal serwer DNS na tym komputerze to, strona będzie niedostępna pomimo tego że istnieje zapasowy serwer WWW.

Dzielenie obciążenia poprzez Round-Robin DNS
Od ukazania się wersji BIND'a 4.9, wprowadzono możliwość skonfigurowania DNS'a tak aby rozdzielał on obciążenie pomiędzy kilka serwerów (nazwane to zostało round-robin DNS). Rozwiązanie to funkcjonuje do dzisiaj. Polega ono na tym iż w przypisujemy w naszym serwerze DNS'owym przypisujemy jednemu adresowi internetowemu kilka numerów IP.

Przykładowo:

```
www.alpha-complex.com. 60 IN A 204.148.170.1
www.alpha-complex.com. 60 IN A 204.148.170.2
www.alpha-complex.com. 60 IN A 204.148.170.3
```

Kiedy nasz serwer DNS dostanie zapytanie o numer IP adresu www.alpha-complex.com, BIND zwróci jeden z tych trzech adresów IP i zanotuje który tych trzech IP zwrócił. Przy następnym zapytaniu dnsowym zwróci kolejny z tych numerów IP, gdy zwróci ostatni z nich, zacznie zwracać od początku poszczególne numer IP
Wiedząc że inne serwery DNS'owe cachuja sobie otrzymane od naszego serwera odpowiedzi to musimy ustawić TTL odpowiednio niski aby obciążenie rozłożyć jak najbardziej równomiernie, nie mniej jednak nie może to być zbyt mała wartość żeby zapytania DNS'owe nie obciążały nadmiernie sieci (sugerowana jest wartość około 60 minut).
Rozwiązanie to ma także zastosowanie do innego rodzaju serwerów nie tylko do serwerów WWW). Ma też kilka wad. Przede wszystkim nie zapewnia ono równomiernego rozłożenia obciążenia poszczególnych serwerów WWW, a także jeżeli jeden z naszych serwerów WWW będzie niedostępny to nadal będzie zwracany jego numer IP przez nasz serwer DNS. Zaletą tego rozwiązania jest to że aby je wykorzystać wystarczy ze wpiszemy kilka linijek do pliku serwera DNS'owego.

Sprzętowe równoważenie obciążenia.
Wiele firm (na przykład Cisco), wytwarza produkty do zastosowań w klastrach sieciowych na poziomie TCP/IP. Są one bardzo efektywne, ale także bardzo drogie.

Tworzenie klastra WWW przy pomocy Apach'a.

Apache dostarcza kilka prostych metod na stworzenie klastra serwerów WWW, przy wykorzystaniu mod_rewrite i mod_proxy. Pozwala to ominąć problemy na które natykamy się przy tworzeniu klastra przy pomocy DNS'u, "ukrywając nasz klaster za serwerem proxy. Jedną z zalet tego rozwiązania jest to że wykorzystuje ona Apache, a co za tym idzie jest to rozwiązanie darmowe.
Rozwiązanie to przedstawia się w następujący sposób: na jednym z serwerów stawiamy serwer proxy, który
kieruje nasze zapytania do jednego z serwerów właściwych. Naszemu serwerowi proxy przypisujemy nazwę www.alpha-complex.com, a serwery właściwe przykładowo nazywamy od www1 do www6.

Rozwiązanie to można podzielić na dwie części.

Używając mod_rewrite losowo kierujemy zapytanie klienta do jednego z naszych serwerów właściwych.
Używając mod_proxy ProxyPassReverse maskujemy prawdziwy adres jednego z naszych serwerów właściwych i zapytanie klienta idzie do naszego serwera nie bezpośrednio tylko przez nasz serwer proxy.

Część pierwsza wymaga od nas stworzenia odpowiedniego pliku (random file map) zawierającego prostą linie:

```
# /usr/local/Apache/rewritemaps/cluster.txt
#
# Random map of back-end web servers
www www1|www2|www3|www4|www5|www6
```

Kiedy używamy tego pliku to za wartość WWW podstawiana jest losowo jedna z wartości (od www1 do www6)

Następnie musimy w konfiguracji naszego serwera proxy wpisać kilka poleceń modułu mod_rewrite używających tego pliku

```
# switch on URL rewriting
RewriteEngine on
# define the cluster servers map
RewriteMap                                                cluster
rnd:/usr/local/Apache/rewritemaps/cluster.txt
# rewrite the URL if it matches the web server host
RewriteRule ^http://www.(.*)$ http://{cluster:www}.$2 [P,L]
# forbid any URL that doesn't match
RewriteRule .* - [F]
```

Wykorzystując mod_rewrite możemy w ten sposób obsłużyć więcej niż jeden klaster sieciowy:

Plik `/usr/local/Apache/rewritemaps/cluster.txt`:

```
www www1|www2|www3|www4|www5|www6
secure secure-a|secure-b
users admin.users|normal.users
```

RewriteRule:

```
# rewrite the URL based on the hostname asked for. If nothing
matches,
# default to 'www1':
RewriteRule  ^http://([^.]+).(.*)$  http://{cluster:$1|www1}.$2
[P,L]
```

Część druga wykorzystuje mod_proxy do przekierowywania zapytań klientów na właściwy serwer WWW.
Bez takiego rozwiązania klient otrzymuje odpowiedź na swoje zapytanie z lokalizacją zaczynającą się od www1 lub www3 zamiast WWW, można to poprawić wykorzystując ProxyPassReverse:

```
ProxyPassReverse / http://www1.alpha-complex.com
ProxyPassReverse / http://www2.alpha-complex.com
...
ProxyPassReverse / http://www6.alpha-complex.com
```

Kompletna konfiguracja Apache tworząca klaster WWW przez proxy wygląda w następujący sposób:

```
# Apache Server Configuration for Clustering Proxy
### Basic Server Setup
# The proxy takes the identity of the web site...
ServerName www.alpha-complex.com
ServerAdmin webmaster@alpha-complex.com
ServerRoot /usr/local/Apache
DocumentRoot /usr/local/Apache/proxysite
ErrorLog /usr/local/Apache/proxy_error
TransferLog /usr/local/Apache/proxy_log
User nobody
Group nobody

# dynamic servers load their modules here...
# don't waste time on things we don't need
HostnameLookups off

# this server is only for proxying so switch off everything
else
Options None
AllowOverride None

# allow a local client to access the server status
order allow,deny
deny from all
allow from 127.0.0.1
SetHandler server-status


### Part 1 - Rewrite
# switch on URL rewriting
RewriteEngine on
# Define a log for debugging but set the log level to zero to
disable it for
# performance
```

```
RewriteLog logs/proxy_rewrite
RewriteLogLevel 0
# define the cluster servers map
RewriteMap                                                    cluster
rnd:/usr/local/Apache/rewritemaps/cluster.txt
# rewrite the URL if it matches the web server host
RewriteRule ^http://www.(.*)$ http://{cluster:www}.$2 [P,L]
# forbid any URL that doesn't match
RewriteRule .* - [F]


### Part 2 - Proxy
ProxyRequests on
ProxyPassReverse / http://www1.alpha-complex.com/
ProxyPassReverse / http://www2.alpha-complex.com/
ProxyPassReverse / http://www3.alpha-complex.com/
ProxyPassReverse / http://www4.alpha-complex.com/
ProxyPassReverse / http://www5.alpha-complex.com/
ProxyPassReverse / http://www6.alpha-complex.com/
# We don't want caching, preferring to let the back end
servers take the load,
# but if we did:
#
#CacheRoot /usr/local/Apache/proxy
#CacheSize 102400
```

Wykorzystując ten sposób możemy nasze serwery umieścić za firewallem sieci wewnętrznej która jest lepiej chroniona przed włamaniami z zewnątrz.
Wadą przyjęcia tego rozwiązania jest to iż obciążenie na poszczególne serwery nie jest rozkładane równomiernie. Można to naprawić w ten sposób iż zastępujemy plik (random file map) zewnętrznym programem, który stara się inteligentnie pokierować ruchem, zgadując który z podanych serwerów jest w danym momencie najmniej obciążony. Program ten powinien być bardzo prosty żeby zbytnio nie oddziaływał na działanie całego systemu.

Inne rozwiązania klastrów WWW:

Eddie
Projekt Eddie jest projektem open-source, zapoczątkowany i sponsorowany przez Ericssona. Projekt ma na celu rozwój zaawansowanych rozwiązań klastrowych dla Linuxa, FreeBSD, oraz Solarisa. WindowsNT został później włączony do projektu. System składa się z dwóch pakietów: rozrzerzonego serwera DNS który zastępuje standartodego demona BIND i zapewna prawdziy load balancing, oraz inteligentnej bramy HTTP która pozwala serwerom WWW pracować w klastrze. Do pakietu dołączona jest przykładowa konfiguracja Apache, i dostępne są pakiety dla linuxa pod adresem http://www.eddieware.org

TurboCluster
Rozwiązanie dostępne darmowo, rozwijane dla potrzeb TurboLinuxa
http://community.turboLinux.com/cluster/.

Sun Cluster
System Solarisa najbardizej zainteresuje te osoby które posiadają oprogramowanie Sun'a , niestety nie jest to darmowy produkt.
http://www.sun.com/clusters/.

Freequalizer
Freequalizer jest darmową wersją Equalizera produkowanego przez Coyote Point Systems i zaprojektowanego dla systemu FreeBSD (Equalizer, wersja komercyjna, jest przeznaczony do uruchamiania na na sprzęcie produkowanym przez tą firmę). Narzędzia do monitorowania pracy oprogramowania są częścią pakietu.
http://www.coyotepoint.com.

Apache 2.0

Utrzymanie pozycji lidera na rynku serwerów www wymaga jednak od programistów ASF (Apache Software Foundation) ciągłej pracy nad serwerem. World Wide Web w chwili obecnej jest najbardziej dynamicznie rozwijającym się rynkiem, i zaledwie chwilowy przestój może owocować stratami nie do nadrobienia. Serwer Apache oczywiście zawdzięcza swoją wiodącą pozycję intensywnemu rozwojowi, który dotyczy zarówno optymalizacji już napisanego kodu jak i implementacji nowych cech. Jednak od pewnego czasu zespół programistów pracujących n ad projektem Apache zamiast na doraźnych celach, koncentruje się na pracach nad nową wersją serwera która będzie oznaczona numerem 2.0.

Mimo, że wersja 2.0 nadal pozostaje w stadium "beta", to w miarę wyraźnie jest już zarysowany kształt nowego "apacza". O ile w zasadzie nic nowego nie dzieje się w stabilnej i sprawdzonej rodzinie Apache 1.3.x, to w Apache 2.0 aż kipi od nowości.

Znakomita większość wprowadzanych zmian dotyczy kodu źródłowego, a ściślej jego organizacji. Całkowicie usunięto kod odpowiedzialny za obsługę konkretnych platform formując jednocześnie osobny komponent APR (Apache Portable Run-Time) odpowiedzialny właśnie za poprawną pracę serwera na różnych platformach. Całokształt zagadnień związanych z APR w ostateczności przekłada się na szybszą, wydajniejsza pracę oraz ułatwienie w pracy nad rozwojem samego serwera.

Na tym jednak nie kończą się zmiany w kodzie źródłowym. Zapewne najbardziej widoczną zmianą w nowym serwerze jest wprowadzenie MPM (Multi-Processing Modules). Apache 2.0 bowiem może obsługiwać napływające połączenia na kilka sposobów posiłkując się kombinacją procesów potomnych, czy też wątków. Od platformy na jakiej pracuje serwer, oraz od specyfiki jego działania zależeć będzie prawidłowy dobór MPM. Poprawna decyzja z kolei będzie leżeć u podstaw wydajnej (lub wręcz przeciwnie) pracy serwera, toteż znajomość zasad działania MPM jest w nowym Apache wręcz krytyczna.

Bezsprzecznie jednym z podstawowych atutów serwera Apache są moduły. Jak można było się spodziewać: nie obyło się bez rewolucji i w tej części serwera. Przede wszystkim spore zmiany w kodzie odbiły się na samych modułach. Bez uprzednich zmian żaden z modułów dostarczanych z Apache v1.3 nie skompiluje się poprawnie w przypadku Apache 2.0. Wszystkie moduły są powoli portowane na potrzeby nowego Apache'a. Koszt "zerwania z kompatybilnością" w znacznym stopniu rekompensuje benefit w postaci łatwiejszego i bardziej przejrzystego API modułów. Razem ze znanymi już z wersji 1.3 modułami pojawiły się nowe, które m.in. pozwalają na bardzo wydajny mechanizm automatycznej konfiguracji serwerów wirtualnych (mod_vhost_alias), czy też wydajną pracę skryptów CGI pod kontrolą nowych MPM (mod_cgid).

Zmian doczekało się logowanie dostępu do serwera. Dotychczas logowaniem zajmował się sam serwer co nie za specjalnie dobrze odbijało się na jego wydajności (zwłaszcza przy "HostnameLookups On"). Teraz pracą tą obarczone są inne programy, które oczywiście stanowią cześć dystrybucji Apache'a. Serwer uwolniony z jeszcze jednego obowiązku może zając się tym czym powinien, czyli udostępnianiem stron.

Wszystkie wprowadzone zmiany, a także te które dopiero trafią do nowego Apache'a na uwadze mają ten sam cel: Apache 2.0 ma być jeszcze szybszy, jeszcze wydajniejszy i bezpieczny niż aktualnie dostępny Apache 1.3.

Dodatek

Strony domowe producentów serwerów WWW

Apache - http://www.apache.org/
Apache-SSL-US - http://apachessl.c2.net/
Stronghold - http://stronghold.ukweb.com/
Apache-NeoWebScript - http://www.neosoft.com/neowebscript/
NCSA - http://hoohoo.ncsa.uiuc.edu/
Netscape Commerce Server -
http://home.netscape.com/comprod/netscape_commerce.html
Netscape Communications Server
- http://home.netscape.com/comprod/netscape_commun.html
Netscape FastTrack
- http://home.netscape.com/comprod/server_central/product/fast_track/index.html
Netscape Enterprise
- http://home.netscape.com/comprod/server_central/product/enterprise/index.html
CERN - http://www.w3.org/hypertext/WWW/Daemon/
Microsoft-Internet-Information-Server
- http://www.microsoft.com/iis/
thttpd - http://www.acme.com/software/thttpd/
WebSite - http://website.ora.com/
WebSitePro - http://website.ora.com/
Process - http://www.process.com/
MacHTTP - http://www.starnine.com/machttp/
WebSTAR - http://www.webstar.com/
HomeDoor - http://www.opendoor.com/
RushHour - http://www.maxum.com/RushHour/
NetPresenz - http://www.stairways.com/netpresenz/index.html
AolServer - http://www.aolserver.com/
Commerce-Builder - http://www.ifact.com/
WN - http://hopf.math.nwu.edu/
Oracle - http://www.oracle.com
EMWAC - http://emwac.ed.ac.uk/
WebQuest - http://www.questar.com/
Open-Market-WebServer - http://www.openmarket.com/products/webserver.html
Open-Market-Secure-WebServer
- http://www.openmarket.com/products/secureweb.html
GoServe - http://www2.hursley.ibm.com/goserve/
Plexus - http://www.bsdi.com/server/doc/plexus.html
EIT - http://wsk.eit.com/
Spry - http://server.spry.com
OSU - http://kcgl1.eng.ohio-state.edu/www/doc/serverinfo.html
Roxen - http://www.roxen.com
Phttpd - http://www.signum.se/phttpd
Lotus Domino - http://domino.lotus.com/
Zeus - http://www.zeus.co.uk

JavaWebServer - http://jeeves.javasoft.com
ZBServer - http://www.zbserver.com/
Frontier - http://www.frontiertech.com/products/superweb.htm
Gosite-SSL - http://www.gosite.com/
Aserve - http://www.phone.net/aws/
Apache for OS/2 - http://www.slink.com/ApacheOS2/
OS2HTTPD - ftp://ftp.netcom.com/pub/kf/kfan/overview.html
PowerWeb - http://www.compusource.co.za/powerweb/
IBM Internet Connection Server
- http://www.ics.raleigh.ibm.com/
IBM Internet Connection Secure Server
- http://www.ics.raleigh.ibm.com
Alibaba - http://alibaba.austria.eu.net/
Boulevard - http://www.resnova.com/Boulevard/
WebForOne - http://www.resnova.com/WebForOne/
Webshare
- http://www.beyond-software.com/Products/EWeb/Webshare/webshare.html
EnterpriseWeb
- http://www.beyond-software.com/Products/EWeb/eweb.html
Cosmos - http://www.ris.fr
Web Commander - http://www.luckman.com/wc/webcom.html
American SiteBuilder - http://www.american.com/product1.html
GLACI HTTPD - http://www.glaci.com/info/glaci-httpd.html
Common Lisp Hypermedia Server (CL-HTTP)
- http://www.ai.mit.edu/projects/iiip/doc/cl-http/home-page.html
W4-Server - http://130.89.224.16/
I/NET - http://www.inetmi.com
WebDisk - http://www.ararat.com/
Web Server 4D - http://www.mdg.com/
HyperWave - http://www.hyperwave.com
TeleFinder - http://www.tfbbs.com
Viking - http://www.robtex.com/viking/
VM:Webserver - http://www.vm.sterling.com/
OmniHTTPd - http://www.omnicron.ab.ca/httpd/
Xitami - http://www.imatix.com/html/xitami/index.htm
Avenida - http://www.avenida.co.uk
Spinnaker - http://www.telegrafix.com
Wildcat - http://www.santronics.com/
vqServer - http://www.vqsoft.com//vq/server/index.html
NSL'server - http://www.nsl.net/
Goahead - http://www.goahead.com/
StWeb - http://www.stweb.org/
HTTPi - http://stockholm.ptloma.edu/httpi/
VSWebServer - http://www.vswebcenter.com/vswebserver/webserve.html
Caudium - http://caudium.net/
WebLogic - http://www.bea.com/products/servers_application.shtml

Główne opcje Apache w plikach konfiguracyjnych

Poniżej przedstawione dyrektywy służą do ustawiania głównych właściwości Apache'a i są one zawsze takie same.

AccessConfig directive

Składnia: AccessConfig *nazwa_pliku*
Domyślnie: `AccessConfig conf/access.conf`
Kontekst: server config, virtual host
Status: core
Serwer czyta ten plik aby uzyskać więcej wskazówek co do ustawień ale po przeczytaniu pliku ResourceConfig . *Nazwa_pliku* jest taka jak w ustawieniu ServerRoot. Opcje tą możemy wyłączyć używając:

```
AccessConfig /dev/null
```

Historycznie, ten plik zawierał się w sekcji <Directory>; faktycznie może on zwierać teraz dowolną dyrektywę dozwoloną w *server config* (w pliku konfiguracyjnym serwera).

AccessFileName directive

Składnia AccessFileName *nazwa_pliku*
Domyślnie: `AccessFileName .htaccess`
Kontekst: server config, virtual host
Status: core
Kiedy serwer przesyła dokument dla klienta szuka pliku z określonymi prawami dostępu z nazwą dokumentu w każdym katalogu podanym w scieżce dostępu do tego dokumentu, jeżeli pliki z prawami dostępu są w danym katalogu. Na przykład:

```
AccessFileName .acl
```

przed odesłaniem dokumentu /usr/local/web/index.html, do klienta serwer przeczyta /.acl, /usr/.acl, /usr/local/.acl i /usr/local/web/.acl aby sprawdzić co ma dalej zrobić z tym dokumentem, chyba, że funkcja ta będzie wyłączona poleceniem

```
<Directory />
AllowOverride None
</Directory>
```

AddModule directive

Składnia AddModule *moduł moduł ...*
Kontekst: server config
Status: core
Zgodność: AddModule jest dostępny tylko w wersji Apache'a 1.2 i późniejszych
Server może posiadać skompilowane moduły, które nie są aktywne. Ta opcja może być użyta do załączenia tych modułów. Serwer rozprowadzany jest z listą załadowanych aktywnych modułów. Lista ta może być wyczyszczona poprzez opcję ClearModuleList .

AllowOverride directive

Składnia AllowOverride *override override ...*
Domyślnie: `AllowOverride All`
Kontekst: directory
Status: core

Kiedy serwer odnajdzie pliki .htaccess (określone w AccessFileName) musi wiedzieć, które dyrektywy zadeklarowane w tym pliku mogą unieważnić wcześniej podane informacje o dostępie.
Override może być ustawiony jako `None`, w typ przypadku serwer nie będzie czytał pliku, `All` w typ przypadku serwer zezwoli na wszystkie dyrektywy (opcje) lub jedną albo ktorąś z niżej podanych:

AuthConfig
: Zezwala na używanie dyrektyw autoryzacji. (AuthDBMGroupFile, AuthDBMUserFile, AuthGroupFile, AuthName, AuthType, AuthUserFile, require, itp.).

FileInfo
: Zezwala na używanie dyrektyw kontrolujących typy dokumentów (AddEncoding, AddLanguage, AddType, DefaultType, ErrorDocument, LanguagePriority, itp.).

Indexes
: Zezwala na używanie dyrektyw kontrolujących katalogowanie katalogu (AddDescription, AddIcon, AddIconByEncoding, AddIconByType, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions, ReadmeName, itp.).

Limit
: Zezwla na używanie dyrektyw kontrolujących dostęp do hosta (allow, deny and order).

Options
: Zezwala na używanie dyrektyw kontrolujących określone cechy katalogu (Options i XBitHack).

AuthName directive

Składnia AuthName *auth-domain*
Kontekst: directory, .htaccess
Override: AuthConfig
Status: core
Dyrektywa ta ustawia nazwę dziedziny autoryzacji dla katalogu. Dziedzina przesyłana jest do klienta i stąd użytkownik wie na jaki login i hasło musi się zalogować. Musi być to skoordynowane z ustawieniami AuthType, dyrektyw require i dyrektywami takimi jak AuthUserFile i AuthGroupFile ażeby działało.

AuthType directive

Składnia AuthType *type*
Kontekst: directory, .htaccess
Unieważnia: AuthConfig
Status: core
Dyrektywa ta wybiera typ autoryzacji użytkownika, który ma mieć dostęp do katalogu. Tylko `Basic` jest aktualnie zaimplementowany. Musi być to skoordynowane z ustawieniami AuthType, dyrektyw require i dyrektywami takimi jak AuthUserFile i AuthGroupFile ażeby działało.

BindAddress directive

Składnia BindAddress *saddr*
Domyślnie: `BindAddress *`
Kontekst: server config
Status: core
A Unix(R) http serwer może nasłuchiwać żądań połączeń dla każdego adresu IP danej maszyny lub tylko dla jednego adresu IP *Saddr* może być

- *
- adresem IP
- domeną internetową

Jeżeli przyjmuje wartość * , wtedy serwer nasłuchuje żądań połączeń dla wszystkich adresów IP, w innym przypadku nasłuchuje tylko dla określonego IP.
Ta opcja może być użyta jako alternatywna metoda do obsługi virtual hosts (wirtualnych hostów) zamiast sekcji <VirtualHost>.
Zobacz również: kwestie DNS
Zobacz również: Ustawianie adresów i portów z których korzysta Apache

ClearModuleList directive

Składnia ClearModuleList
Kontekst: server config
Status: core
Zgodność: ClearModuleList jest dostępna tylko w wersji Apache'a 1.2 i późniejszych
Serwer rozprowadzany jest z listą załadowanych aktywnych modułów. Dyrektywa ta czyści tą listę. Oczywiście zakładając, że następnie lista będzie odnowiona poprzez dyrektywę AddModule.

DefaultType directive

Składnia DefaultType *mime-type*
Domyślnie: `DefaultType text/html`
Kontekst: server config, virtual host, directory, .htaccess
Override: FileInfo
Status: core
Czasami bywa tak, że serwer pytany jest o typ przesyłanego dokumentu a dokument ten nie może być określony poprzez typy MIME.
Serwer musi poinformować klienta o typie dokumentu i jeżeli typ jest nieznany sewer używa domyślnego typu `DefaultType`. Na przykład:

```
DefaultType image/gif
```

mógłby być stosowny dla katalogu, który zawiera wiele gif'ów ale w nazwach nie mają rozszerzenia .gif.

<Directory> directive

Składnia <Directory *directory*> ... </Directory>
Kontekst: server config, virtual host
Status: Core.
<Directory> i </Directory> jest używane do zawaracia w jednej grupie dyrektyw, które będa miały zastosowanie do katalogów i podkatalogów danego katalogu. Jakakolwiek dyrektywa która jest zawarta w kontekście może być użyta. *Directory* spełnia również rolę ścieżki dostępu do katalogu albo jest ciągiem wild-card. W ciągu wild-card, '?' oznacza pojedyńczy znak a '*' oznacza dowolny ciąg znaków. Na przykład:

```
<Directory /usr/local/httpd/htdocs>
Options Indexes FollowSymLinks
</Directory>
```

Apache 1.2 i nowsze: Rozszerzone wyrażenia mogą być użyte z dodatkowym znakiem ~. Na przykład:

```
<Directory ~ "^/www/.*/[0-9]{3}">
```
oznacza katalogi w /www/ zawierające trzy cyfry.
Jeżeli wszelkie sekcje katalogów pasują do katalogu zawierającego dokument to dyrektywa wskazuje na najkrótszą pasującą nazwę katalogu i wczytuje dyrektywy z plików .htaccess. Na przykład:

```
<Directory />
AllowOverride None
</Directory>

<Directory /home/*>
AllowOverride FileInfo
</Directory>
```

kroki dostępu do dokumentu `/home/web/dir/doc.html` są następujące:

- Zastosowanie dyrektywy `AllowOverride None` (wyłączając pliki `.htaccess`).
- Zastosowanie dyrektywy `AllowOverride FileInfo` (dla katalogu `/home/web`).
- Zastosowanie jakiejkolwiek dyrektywy FileInfo w `/home/web/.htaccess`

Uwaga domyślnie w Apache'u ustawiony jest dostęp do <Directory /> `Allow from All (dla wszystkich)`. To oznacza, że Apache będzie przesyłal dowolny plik podany w URL'u. Zalecane jest, że zmienisz to ustawieinie poprzez

```
<Directory />
    Order Deny,Allow
    Deny from All
</Directory>
```

i unieważnisz to dla katalogu który *chcesz* usdostępnić. Zobacz stronę Security Tips po więcej szczegółów.
Sekcja katalogów standardowo występuje w pliku access.conf, ale może występować w dowolnym z plików konfiguracyjnych. Dyrektywa <Directory> nie może być umieszczona i nie może znajdować się w sekcji <Limit>.

DocumentRoot directive

Składnia DocumentRoot *nazwa_katalogu*
Domyślnie: `DocumentRoot /usr/local/etc/httpd/htdocs`
Kontekst: server config, virtual host
Status: core
Dyrektywa ta ustawia katalog z którego httpd będzie pobierał (serwował) pliki. Jeżeli ustawienie nie pasuje do ustawionego np. Aliasu, serwer dołączy ścieżkę z URL'a do podanego dokumentu aby mógł mieć dostęp do dokumentu. Przykład:

```
DocumentRoot /usr/web
```

wówczas dostęp do `http://www.my.host.com/index.html` kierowany jest do `/usr/web/index.html`.
Są błędy w mod_dir które stwarzają problemy wówczas gdy DocumentRoot ma wstawiony slash na końcu ("DocumetRoot /usr/web/") także lepiej tego unikać.

ErrorDocument directive

Składnia ErrorDocument *error-code document*
Context server config, virtual host, directory, .htaccess
Status: core

Uniważnia: FileInfo
Zgodność: Konteks katalogu i .htaccess dostępne są tylko w Apache'u 1.1 i późniejszych.
W przypadku wystąpienia błędu, Apache może być skonfigurowany tak aby zrobił coś z niżej wymienionych punktów,

1. pokazać prostą informację o błędzie
2. pokazać przerobioną wiadomość
3. przeadresować do lokalnego URL aby wstrzymał problem/błąd
4. przeadreswoać do zewnętrznego URL aby wstrzymał problem/błąd

Pierwsza opcja jest ustawiona domyslnie, podczas gdy opcje 2-4 sa skonfigurowane przy użyciu dyrektywy `ErrorDocument`.
Messages w tym kontekście zaczynają się zwykłym pojedyńczym cudzysłowem (`"`), jeżeli nie wchodzą w treść informacji. Apache czasami oferuje dodatkowe informacje odnośnie danego problemu/błędu.
URL'e mogą zaczynać sie od slash'a (/) dla lokalnych URL'i, lub moga być podane w pełnej formie. Przykład:

```
ErrorDocument 500 http://foo.example.com/cgi-bin/tester
ErrorDocument 404 /cgi-bin/bad_urls.pl
ErrorDocument 401 /subscription_info.html
ErrorDocument 403 "Sorry can't allow you access today
```

Kiedy wyszczególnisz `ErrorDocument` który prowadzi do odległego URL'a (będziesz podawał adresy z "http" na początku) Apache będzie wysyłał przekierowanie do klienta z informacją gdzie ma znaleźć dokument, nawet wtedy gdy dokument będzie znajdował się na tym samym serwerze... To zawiera kilka implikacji, najważniejsza zaistnieje **jeżeli użyjesz dyrektywy "ErrorDocument 401" powinna ona odsyłac do istniejącego lokalnego dokumentu.**Rezultaty pochodzą w gruncie rzeczy z podstawowych własności protokołu HTTP.
Zobacz również: documentation of customizable responses.

ErrorLog directive

Składnia ErrorLog *nazwa_pliku*
Domyślnie: `ErrorLog logs/error_log`
Kontekst: server config, virtual host
Status: core
Dyrektywa error log ustawia nazwę pliku gdzie serwer będzie przechowywał informacje o błędach, o ile będą. Jeżeli nazwa pliku nie będzie poprzedzona slash'em (/) to automatycznie będzie to odpowiadało temu, że plik ten znajduje się w miejscu określonym przez dyrektywę ServerRoot. Na przykład:

```
ErrorLog /dev/null
```

Wyłącza zapisywanie błędów w logu.
ŚRODKI BEZPIECZEŃSTWA: Zajrzyj do security tips po więcej szczegółów na temat dlaczego Twój serwer jest narażony jeżeli prawa zapisu do katalogu gdzie przechowywane są logi mają inni użytkownicy aniżeli "użytkownik" który uruchomił serwer.

<Files>

Składnia <Files *nazwa_pliku*> ... </Files>
Kontekst: server config, virtual host, htaccess
Status: core
Zgodność: dostępne tylko w wersji Apache 1.2 i wyżej.

Dyrektywa <Files> pozwala na kontrolę dostepu poprzez nazwę pliku. Jest ona porównywalna z dyrektywą <Directory> i <Location>. Dyrektywy które mają być zastosowane z dyrektywą <Files> powinny być zawarte pomiędzy znacznikami `<Files>` *nazwa_pliku* powinna zawierać nazwę pliku, lub łańcuch wild-card, gdzie `?' oznacza pojedyńczy znak, a `*' oznacza dowolny ciąg znaków. Rozszerzone regularne wyrażenia mogą być również użyte z dodatkowym znakiem ~. Na przykład:

```
<Files ~ "\.(gif|jpe?g|png)$">
```

powinno odpowiadać większości używanych formatach graficznych w Internecie.
Uwaga w odróżnieniu od `<Directory>` i `<Location>` , sekcje `<Files>` mogą być użyte w pliku .htacces. To zezwala użytkownikom kontrolowanie dostępu do ich własnych plików. Kiedy dyrektywa `Files` użyta jest w pliku .htaccess i *nazwa_pliku* nie rozpoczyna się od znaku `/`, do katalogu który jest tam umieszczony zostanie automatycznie dodany przedrostek.

Group directive

Składnia Group *unix-group*
Domyślnie: `Group #-1`
Kontekst: server config, virtual host
Status: core
Dyrektywa Group ustawia grupę na której rządania będzie odpowiadał serwer. Aby korzystać z tej dyrektywy, serwer musi być uruchomiony w trybie stand-alone z prawami użytkownika root (administratora). *Unix-group* jest jedną z:
nazwa grupy
    Odesłanie wziąwższy pod uwagę nazwę grupy.
lepiej używać numeru grupy.
    Odesłanie do grupy poprzez numer.
Zalecane jest, że ustawisz nową grupę wykorzystywaną tylko do uruchamiania serwera. Niektórzy administratorzy używają użytkownika `nobody`, ale nie zawsze jest to możliwe lub wskazane.
Uwaga: jeżeli uruchomisz serwer jako użytkownik non-root (nie-administrator) nie będzie można zmienić na odpowiedna do tego grupę i działanie serwera będzie kontynuowane w grupie użytkownika który uruchomił serwer.
Uwaga specjalna: Użycie tej dyrektywy w <VirtualHost> wymaga poprawnego skonfigurowania suEXEC wrapper. Jeżeli jest użyta wewnątrz <VirtualHost> w ten sposób, tylko grupa która ma prawa na uruchamianie skryptów CGI jest afektowana. Żądania połączenia ale nie żądania uruchomienia skryptów CGI są nadal wykonywane w grupie ustawionej dyrektywą Group.

HostNameLookups directive

Składnia HostNameLookups *boolean*
Domyślnie: `HostNameLookups on`
Kontekst: server config, virtual host
Status: core
Dyrektywa ta załącza sprawdzanie DNS'u i dzięki temu nazwy host'ów mogą być logowane. Mając ustawioną tą dyrektywę na `on` załącza ona również wykorzystywanie nazw w <Limit> kontrolując dostęp.
Obciążone serwery powinny ustawić tą dyrektywę na `off`, ponieważ sprawdzanie DNS'u zabiera dużo czasu. *Logresolve* znajdujący się w katalogu */support* może być użyty do sprawdzenia IP zalogowanych nazw hostów offline.

IdentityCheck directive

Składnia IdentityCheck *boolean*
Domyślnie: `IdentityCheck off`
Kontekst: server config, virtual host
Status: core
Dyrektywa załącza (RFC1413) rozpoczęcie zapisu logowania połączeń użytkowników gdy po stronie użytkownika uruchomiony jest ident (identyfikacja) lub coś podobnego. Informacje te przechowywane są w pliku log.
Informacje te powinny być traktowane tylko informacyjnie.
Dyrektywa ta może powodować częste opóźnienia podczas dostępu do Twojego serwera gdy każda prośba połączenia będzie sprawdzana i logowana. Kiedy jest ustawiony skomplikowany firewall żądania połączenia mogą sie nie powieść dodaj wtedy 30 sekund opóźnienia. Dyrektywa ta nie jest użyteczna na publicznych serwerach dostępnych w Internecie.

<IfModule>

Składnia <IfModule [!]*module-name*> ... </IfModule>
Domyślnie: None
Kontekst: all
Status: Core
Zgodność: ScriptLog dostępny tylko w wersji 1.2 i nowszych.
Sekcja <IfModule *test*>...</IfModule> jest używana do zaznaczenia dyrektyw warunkowych. Dyrektywy umieszczone wewnątrz IfModule wykonywane są tylko kiedy *test* jest prawdziwy. Jeżeli *test* jest fałszywy, wszystko pomiędzy startem i końcem znaczników jest ignorowane.
test w sekcji <IfModule> może przyjąć jedną z dwóch postaci:

- *module name*
- !*module name*

Pierwszy z formatów oznacza, że dyrektywy umieszczone pomiędzy znacznikami startu i końca wykonywane są tylko wtedy gdy moduł o nazwie *module name* jest skompilowany w Apache'u. Drugi natomiast działa odwrotnie i dyrektywy są wykonywane gdy moduł **nie** jest skompilowany w Apache'u.
Argument *module name* jest nazwą modułu, która została nadana plikowi w czasie kompilacji, w którym umieszczony jest moduł. Na przykład: `mod_rewrite.c`.
<IfModule> może być użyte do przetestowania modułów.

KeepAlive

Składnia (Apache 1.1) KeepAlive *max-requests*
Domyślnie: (Apache 1.1) `KeepAlive 5`
Składnia (Apache 1.2) KeepAlive *on/off*
Domyślnie: (Apache 1.2) `KeepAlive On`
Kontekst: server config
Status: Core
Zgodność: KeepAlive dostępny tylko w Apache'u 1.1 i nowszym.
Dyrektywa ta załącza wspomaganie Keep-Alive .
Apache 1.1: Ustaw *max-requests* na maksymalną liczbę odpowiedzi na prośby połączeń, które Apache ma przyjmować. Limit jest nałożony aby zapobiec okupowaniu przez klientów Twojego serwera. Aby wyłączyć dostęp ustaw tą dyrektywę na `0`.

Apache 1.2 i nowsze: Ustaw na "On" aby załączyć ciągłe połączenia, "Off" aby wyłączyć. Zobacz również dyrektywę MaxKeepAliveRequests.

KeepAliveTimeout

Składnia KeepAliveTimeout *seconds*
Domyślnie: `KeepAliveTimeout 15`
Kontekst: server config
Status: Core
Zgodność: KeepAliveTimeout dostępne tylko w wersji 1.1 i późniejszych.
Liczba sekund w których Apache czeka na kolejne połączenia przed zerwaniem połączenia. Wartośc timeout określa dyrektywa `Timeout`.

Listen

Składnia Listen [*IP address*:]*port number*
Kontekst: server config
Status: core
Zgodność: Listen dostępne tylko w wersji 1.1 i późniejszych.
Dyrektywa Listen instruuje Apache'a do nasłuchiwania więcej niż tylko jednego adresu IP lub portu; domyślnie odpowiada na prośby połączeń na wszystkich interfejsach IP, ale tylko na jednym porcie ustawionym przez dyrektywę Port
.

<Limit> directive

Składnia <Limit *method method ...* > ... </Limit>
Kontekst: any
Status: core
<Limit> i </Limit> używane są aby zawrzeć grupę dyrektyw kontrolujących dostęp, które będą zastosowane tylko do określonej metody dostępu, gdzie *method* jest ważną metodą HTTP. Jakakolwiek dyrektywa poza inną niż <Limit> lub <Directory> może być wykorzystana; większość będzie miała znaczenia na <Limit>. Na przykład:

```
<Limit GET POST>
require valid-user
</Limit>
```

Jeżeli dyrektywa kontroli dostępu pojawi sie poza dyrektywą <Limit> to zastosowane zostaną wszystkie metody kontroli dostępu.

<Location>

Składnia <Location *URL*> ... </Location>
Kontekst: server config, virtual host
Status: core
Zgodność: Location dostępne tylko w wersji 1.1 i nowszych.
Dyrektywa <Location> uwzględnia kontrolę na poziomie adresu URL. Jest podobna do dyrektywy <Directory>, i powinna być używana razem z dyrektywą </Location> Dyrektywy które są stosowane z adresem URL powinny być umieszczone wewnątrz. Sekcje `<Location>` są rozpatrywane według kolejności występowania w pliku konfiguracyjnym, ale po uwzględnieniu sekcji <Directory> i po przeczytaniu pliku `.htaccess`.
Apache 1.2 i wyższe: Rozszerzone regularne wyrażenia również mogą być używane poprzez dodanie znaku `~`. Na przykład:

```
<Location ~ "/(extra|special)/data">
```
Pasuje do URL'i które zawierają łańcuch "/extra/data" lub "/special/data".
`Location` jest funkcjonalne w połączeniu z dyrektywą SetHandler. Na przykład, aby załączyć stan próśb połączeń, ale przeznaczony tylko dla przeglądarek z foo.com, możesz użyć:

```
<Location /status>
SetHandler server-status
order deny,allow
deny from all
allow from .foo.com
</Location>
```

LockFile

Składnia LockFile *filename*
Domyślnie: `LockFile logs/accept.lock`
Kontekst: server config
Status: core
Dyrektywa LockFile ustawia scieżkę dostępu do pliku blokującego używanego gdy Apache skompilowany jest jednym z USE_FCNTL_SERIALIZED_ACCEPT lub USE_FLOCK_SERIALIZED_ACCEPT. Dyrektywa ta normalnie powinna pozostać jako domyślna wartość. Główny powód zmian jest jeżeli katalog `logs` zamontnowany jest jako NFS, plik blokujący powinien jednak być trzymany lokalnie o ile jest to możliwe. PID głównego procesu serwera jest automatycznie dodawany do pliku.

MaxClients

Składnia MaxClients *number*
Domyślnie: `MaxClients 256`
Kontekst: server config
Status: core
Dyrektywa MaxClients ustawia limit równoległych połączeń, które mogą być zrealizowane; w zależności od tej dyrektywy uruchomionych zostanie nie więcej niż MaxClient liczba procesów child.

MaxKeepAliveRequests

Składnia MaxKeepAliveRequests *number*
Domyślnie: `MaxKeepAliveRequests 100`
Kontekst: server config
Status: core
Zgodność: Dostępne tylko w wersji 1.2 i nowszych.
Dyrektywa MaxKeepAliveReauests ustawia limit zezwoleń połączeń gdy KeepAlive jest załączona. Jeżeli jest ustawiona na "`0`", zezwolona liczba połączeń będzie nielimitowana. Zalecamy ustawić wartość na jak największą pod względem wydajności serwera.

MaxRequestsPerChild directive

Składnia MaxRequestsPerChild *number*
Domyślnie: `MaxRequestsPerChild 0`
Kontekst: server config
Status: core

Dyrektywa MaxRequestsPerChild ustawia limit połączeń indywidualnie dla każdego procesu child. Po liczbie połączeń określonych przez MaxRequestsPerChild, proces child zostaje zatrzymany. Jeżeli MaxRequestsPerChild jest równy 0, to proces nigdy nie przestanie działać. Ustawienie MaxRequestsPerChild na wartość nie-zerową daje dwie korzyści:

- ogranicza przypadkowe zużywanie pamięci RAM;
- dzięki ograniczeniu czasu życia procesu, redukujemy liczbę niepotrzebnych procesów.

MaxSpareServers directive

Składnia MaxSpareServers *number*
Domyślnie: `MaxSpareServers 10`
Kontekst: server config
Status: core
Dyrektywa MaxSpareServers ustawia maksymalną liczbę bezczynnych procesów child. Bezczynny proces child jest jendnym z procesów nie utrzymujących połączenia. Jeżeli jest więcej bezczynnych procesów niż MaxSpareServers, wtedy nadrzędny proces będzie wyłączał niepotrzebne procesy.
Dokładne ustawienie tego parametru może być potrzebne na bardzo pbleganych site'ach. Ustawianie tego parametru na jak największą wartość jest w większości przypadków złym pomysłem.
Zobacz również MinSpareServers and StartServers.

MinSpareServers directive

Składnia MinSpareServers *number*
Domyślnie: `MinSpareServers 5`
Kontekst: server config
Status: core
Dyrektywa MinSpareServers ustawia minimalną liczbę bezczynnych procesów child. Bezczynny proces child jest jendnym z procesów nie utrzymujących połączenia. Jeżeli jest mniej bezczynnych procesów niż MinSpareServers, wtedy nadrzędny proces utowrzy nowe procesy child, z maksymalną prędkością 1 proces/sekundę.
Dokładne ustawienie tego parametru może być potrzebne na bardzo pbleganych site'ach. Ustawianie tego parametru na jak największą wartość jest w większości przypadków złym pomysłem.
Zobacz również MaxSpareServers and StartServers.

Options directive

Składnia Options *[+|-]option [+|-]option ...*
Kontekst: server config, virtual host, directory, .htaccess
Override: Options
Status: core
Dyrektywa Options kontroluje, które z właściwości serwera są dostępne w konkretnym katalogu.
option może być ustawione na `None`, w tym przypadku żadna z dodatkowych właściwości serwera nie jest załączona, albo jedna lub więcej z poniższych:
All
Wszystkie opcje z wyjątkiem MultiViews.
ExecCGI
Zezwolone jest uruchamianie skryptów CGI.
FollowSymLinks

Server będzie zwracał uwagę na symbliczne linki w katalogu. **Uwaga**: mimo działania tej opcji, nie zostanie zmieniona ścieżka dostępu ustawiona w sekcji `<Directory>`.

Includes
: Includes są zezwolone.

IncludesNOEXEC
: Includes są zezwolone, ale komendy #wykonawcze i #zawierające skrypty CGI są wyłączone.

Indexes
: Jeżeli adres URL odwołuje się do katalogu w którym nie ma DirectoryIndex (np. index.html) to serwer zwróci zformatowany listing katalogu.

MultiViews
: Ustalona zawartość MultiViews jest dozwolona.

SymLinksIfOwnerMatch
: Serwer będzie korzystał z linków symblicznych które wskazują na plik lub katalog tego samego użytkownika którego jest link.

PidFile directive

Składnia PidFile *filename*
Domyślnie: `PidFile logs/httpd.pid`
Kontekst: server config
Status: core
Dyrektywa PidFile ustawia plik w którym server zapisuje numer procesu id demona. Jeżeli nazwa nie rozpoczyna się od slach'a (/) to zostanie przybrana wartość względem ServerRoot. Plik PidFile wykorzystywany jest tylko w trybie pracy standalone.

Port directive

Składnia Port *numer*
Domyślnie: `Port 80`
Kontekst: server config
Status: core
Numer jest liczbą z przedziału od 0 do 65535; niektóre numery portów (szczególnie poniżej 1024) są zarezerwowane dla określonych protokołów. Zobacz `/etc/services`, w pliku tym są już definowane niektóre porty; standardowy port protokołu http to 80.
Dyrektywa Port ma dwa postępowania, pierwsze które jest konieczne dla zachowania kompatybilności z NCSA backwards (które powoduje zagmatwanie w kontekście Apache'a).

- W przypadku nieobecności jakiejkolwiek z dyrektyw Listen lub BindAddress określających numer portu, dyrektywa Port ustawia port sieciowy na którym prowadzi nasłuch serwer. Jeżeli są określone dyrektywy Listen lub BindAddress określające `numer portu:` wtedy dyrektywa Port nie ma wpływu na port na którym serwer prowadzi nasłuch.
- Dyrektywa Port ustawia zmienną `SERVER_PORT` (dla CGI i SSI),i używana jest ona kiedy serwer musi wygenerować URL odsyłający do siebie samego (na przykład w czasie tworzenia wewnętrznego przekierowania do siebie).

Jeżeli nie ma żadnych zdarzeń ustawienie Portu wpływa na jakie porty wirtualnych hostów serwer ma reagować, dyrektywa VirtualHost sama w sobie jest wykorzystywana do tego. Podstawowe zachowanie dyrektywy Port powinno być podobne do dyrektywy ServerName. Dyrektywa ServerName i Port razem określają co powinieneś wziąść pod uwagę przy *kanonicznym* adresie serwera.

Port 80 jest jednym ze specjalnycj portów UNIX'a. Wszystkie numery portów poniżej 1024 są zarezerwowane dla systemu, zwykły użytkownik (nie-root) nie może ich wykorzystywać; zamiast tego może używać wyższych numerów portów. Aby użyc portu 80, musisz wystartować serwer z konta root'a. Po przypisaniu portu ale przed zaakceptowaniem połączeń, Apache zmieni prawa na najmniej uprzywilejowanego użytkownika ustawionego przez dyrektywę User.
Jeżeli nie możesz używać portu 80, wybierz inny wolny port. Zwykli użytkownicy muszą wybrać port powyżej 1023, taki jak 8000.
BEZPIECZEŃSTWO: jeżeli uruchamiasz serwer jako root, upewnij się że nie ustawiłeś dyrektywy User jako root. Jeżeli uruchomisz serwer jako root, Twój serwer będzie otwarty na ataki hackerów.

require directive

Składnia require *entity-name entity entity...*
Kontekst: directory, .htaccess
Override: AuthConfig
Status: core
Dyrektywa ta wybiera którzy "zaufani" użytkownicy mają dostęp do katalogu. Zezwolone składnie to:

- require user *userid userid* ...
  Tylko określeni użytkownicy (z nazwy, imienia) mają dostęp do katalogu.
- require group *group-name group-name* ...
  Tylko użytkownicy danej grupy mają dostęp do katalogu.
- require valid-user
  Wszyscy ważni użytkownicy mają dostęp do katalogu.

Jeżeli `require` pojawi się w dyrektywie <Limit> ograniczony jest wtedy dostęp określony metodą, w przeciwnym wypadku ograniczony jest dostęp wszystkimi metodami. Przykład:

```
AuthType Basic
AuthName somedomain
AuthUserFile /web/users
AuthGroupFile /web/groups
<Limit GET POST>
require group admin
</Limit>
```

ResourceConfig directive

Składnia ResourceConfig *nazwa_pliku*
Domyślnie: `ResourceConfig conf/srm.conf`
Kontekst: server config, virtual host
Status: core
Serwer będzie czytał ten plik po więcej dyrektyw po przeczytaniu pliku httpd.conf. *Nazwa_pliku* jest relatywana w stosunku do dyrektywy ServerRoot. Ta opcja może być wyłączona poprzez użycie:

```
ResourceConfig /dev/null
```

Historycznie, ten plik zawiera większość dyrektyw poza dyrektywami konfiguracyjnymi i sekcji <Directory>; faktycznie może zawierać dowolną dyrektywę serwera dopuszczoną w *konfiguracji serwera*.
Zobacz również AccessConfig.

RLimitCPU directive

Składnia RLimitCPU *# or 'max' [# or 'max']*
Domyślnie: `Unset uses operating system defaults`
Kontekst: server config, virtual host
Status: core
Zgodność: RLimitCPU dostępne tylko w Apache'u 1.2 i nowszych
Pobiera 1 lub 2 parametry. Pierwszy parametr ustawia "łagodny" limit zasobów dla wszystkich procesów a drugi parametr ustawia maksymalny limit zasobów. Obojętnie który parametr może być liczbą, lub *max* sygnalizującym serwerowi, że limit powinien być ustawiony na maksymalny dozwolony przez konfigurację systemu operacyjnego. Dyrektywa ta wymaga uruchamiania serwera jako root.
Limit zasobów CPU wyrażany jest w ilości sekund przypadających na proces.
Zobacz również RLimitMEM or RLimitNPROC.

RLimitMEM directive

Składnia RLimitMEM *# or 'max' [# or 'max']*
Domyślnie: `Unset uses operating system defaults`
Kontekst: server config, virtual host
Status: core
Zgodność: RLimitMEM dostępne tylko w Apache'u 1.2 i nowszych
Pobiera 1 lub 2 parametry. Pierwszy parametr ustawia "łagodny" limit zasobów dla wszystkich procesów a drugi parametr ustawia maksymalny limit zasobów. Obojętnie który parametr może być liczbą, lub *max* sygnalizującym serwerowi, że limit powinien być ustawiony na maksymalny dozwolony przez konfigurację systemu operacyjnego. Dyrektywa ta wymaga uruchamiania serwera jako root.
Limity zasobów pamięci wyrażane są w ilości bajtów przypadających na proces.
Zobacz również RLimitCPU or RLimitNPROC.

RLimitNPROC directive

Składnia RLimitNPROC *# or 'max' [# or 'max']*
Domyślnie: `Unset uses operating system defaults`
Kontekst: server config, virtual host
Status: core
Zgodność: RLimitNPROC dostępne tylko w Apache'u 1.2 i nowszych
Pobiera 1 lub 2 parametry. Pierwszy parametr ustawia "łagodny" limit zasobów dla wszystkich procesów a drugi parametr ustawia maksymalny limit zasobów. Obojętnie który parametr może być liczbą, lub *max* sygnalizującym serwerowi, że limit powinien być ustawiony na maksymalny dozwolony przez konfigurację systemu operacyjnego. Dyrektywa ta wymaga uruchamiania serwera jako root.
Limit procesów kontroluję liczbę procesów przypadających na użytkownika.
See also RLimitMEM or RLimitCPU.

SendBufferSize directive

Składnia SendBufferSize *bytes*
Kontekst: server config
Status: core
Serwer powinien ustawić rozmiar bufora TCP do określonej liczby bajtów.

ServerAdmin directive

Składnia ServerAdmin *email-address*
Kontekst: server config, virtual host
Status: core
Dyrektya ServerAdmin ustawia adres poczty elektronicznej pod który serwer będzie wysyłał wiadmości o błedach zwracanych klientom.
Można wstawić dedykowany adres, na przykład:

```
ServerAdmin www-admin@foo.bar.com
```

ServerAlias directive

Składnia ServerAlias *host1 host2 ...*
Kontekst: virtual host
Status: core
Zgodność: ServerAlias dostępne tylko w Apache'u 1.2 i nowszych.
Dyrektywa ServerAlias ustawia alternatywne nazwy dla hosta, potrzebne przy korzystaniu z Host-header based virtual hosts.

ServerName directive

Składnia ServerName *fully-qualified domain name*
Kontekst: server config, virtual host
Status: core
Dyrektywa ServerName ustawia nazwę hosta danego serwera; wykorzystywane jest to tylko kiedy tworzone jest przekierowanie URL'i. Jeżeli nazwa nie jest określona, wtedy serwer będzie próbował znaleść ją z własnego adresu IP; aczkolwiek może to źle pracować, lub może nie zwracać preferowanej nazwy hosta. Na przykład:

```
ServerName www.wibble.com
```

powinno być użyte jeżeli nazwa kanoniczna aktualnego komputera była `monster.wibble.com`.

ServerRoot directive

Składnia ServerRoot *directory-filename*
Domyślnie: `ServerRoot /usr/local/etc/httpd`
Kontekst: server config
Status: core
Dyrektywa ServerRoot ustawia katalog w którym "żyje" serwer. Domyślnie powinien zawierać podkatalogi `conf/` i `logs/`. Względnie ścieżki do innych plików konfiguracyjnych. Zobacz również the `-d` option to httpd.

ServerType directive

Składnia ServerType *type*
Domyślnie: `ServerType standalone`
Kontekst: server config
Status: core
Dyrektywa ServerType określna w jaki sposób serwer jest uruchamiany przez system. *Type* jest jednym z
inetd
Serwer będzie uruchomiony poprzez systemowy proces inetd; komenda uruchamiające

serwer jest dodana do `/etc/intetd.conf`

standalone

Serwer będzie uruchomiony jako proces demon; komenda uruchamiająca serwer dodana jest do systemowych skryptów startujących. (`/etc/rc.local` lub `/etc/rc3.d/...`)

Inetd jest mniej wykorzystywaną metodą z powyższych dwóch. Dla każdego połączenia http, uruchamiana jest nowa kopia serwera; po zakończeniu połączenia, kopia jest zamykana. Płaci się przez to "wyższą cenę" dla każdego połączenia, ze względu na warunki bezpieczeństwa, niektórzy administatorzy preferują tą metodę.

Standalone jest najbardzie popularną metodą ustawianą dla ServerType, jest ona bardziej wydajna. Serwer uruchamiany jest tylko raz, i obsługuje wszystkie późniejsze połączenia. Jeżeli zamierzasz uruchomić oblegany serwer, standalone jest prawdopodobnie jedynym rozwiązaniem.

StartServers directive

Składnia StartServers *number*
Domyślnie: `StartServers 5`
Kontekst: server config
Status: core

Dyrektywa StartServers ustawia liczbę procesów child tworzonych w czasie uruchamiania serwera. Liczba procesów jest dynamicznie kontrolowana w zależności od przeładowania serwera, z reguły nie ma powodów do zmiany tego parametru.

Zobacz również MinSpareServers i MaxSpareServers.

TimeOut directive

Składnia TimeOut *number*
Domyślnie: `TimeOut 300`
Kontekst: server config
Status: core

Dyrektywa TimeOut obecnie definiuje ilość czasu w którym Apache będzie oczekiwał na trzy rzeczy:

1. Całkowita ilość czasu jaka potrzebna jest do odebrania żądania GET.
2. Całkowita ilość czasu pomiędzy potwierdzeniem pakietów TCP na żądania POST lub PUT.
3. Całkowita ilość czasu pomiędzy ACKs (potwierdzeniami) transmisji pakietów TCP w odpowiedzi.

User directive

Składnia User *unix-userid*
Domyślnie: `User #-1`
Kontekst: server config, virtual host
Status: core

Dyrektywa User ustawia userid na jakie serwer będzie odpowiadał. Żeby korzystać z tej dyrektywy, standalone serwer musi być zainicjowany jako root. *Unix-userid* jest jednym z:

A username

Odpowiada użytkownikowi określonemu przez nazwę (imię).

lepiej używać numberu użytkownika.

Odpowiada użytkownikowi określonemu przez jego numer.

Użytkownik nie powinien posiadać żadnych uprawnień, które zezwalałyby na dostęp do plików nie przeznaczonych do ogladania przez resztę świara, i podobnie użytkownik nie

powinien mieć możliwości wykonywnia programów które nie odpowiadają httpd. Zalecane jest ustawienie nowego użytkownika i grupy przeznaczonej tylko do uruchamiania serwera. Niektórzy administratorzy wykorzystują użytkownika `nobody`, ale nie zawsze jest to możliwe lub wskazane.

<VirtualHost> directive

Składnia <VirtualHost *addr*[:*port*] ...> ... </VirtualHost>
Kontekst: server config
Status: Core.
Zgodność: Non-IP address-based Virtual Hosting dostępne tylko w wersji Apache'a 1.1 i nowszych.
Zgodność: Multiple address support dostępne tylko w wersji Apache'a 1.2 i nowszych.
<VirtualHost> i </VirtualHost> są wykorzystywane do zawarcia grupy dyrektyw które będą stosowane tylko do określonego wirtualnego hosta. Jakakolwiek dyrektywa może być użyta wewnątrz dyrektywy VirtualHost. Kiedy serwer otrzyma żądanie przesłania dokumentu konkretnego wirtualnego hosta, użyje wtedy dyrektyw konfigurujących zawartych w sekcji <VirtualHost> *Addr* może być

- Adresem IP wirtualnego hosta.
- Pełnoprawną domeną przyznaną dla adresu IP wirtualnego hosta.

Przykład:

```
<VirtualHost 10.1.2.3>
ServerAdmin webmaster@host.foo.com
DocumentRoot /www/docs/host.foo.com
ServerName host.foo.com
ErrorLog logs/host.foo.com-error_log
TransferLog logs/host.foo.com-access_log
</VirtualHost>
```

Każdy Wirtualny Host musi posiadać osobny adres IP lub różne nazwy serwerów, w drugim przypadku serwer musi być tak skonfigurowany aby akceptował pakiety IP dla różnych adresów. (Jeżeli komputer nie ma wielu interfejsów sieciowych, dobrym rozwiązaniem jest zastosowanie komendy `ifconfig alias` (o ile Twój system operacyjny zezwala na to)).
Specjalna nazwa `_default_` może być określona w którym przyapdku wirtualny host będzie pasował do dowolnego adresu IP który nie jest wyraźnie zdefiniowany, nasłuchiwany w innym wirtualnym hoście. W przypadku braku _default_ wirtualnego hosta, główny plik konfiguracyjny składa się ze wszystkich definicji poza sekcją VirtualHost.
Możesz określić `:port` dla danego wirtualnego hosta. Jeżeli port nie jest określony przyjmuje domyślną wartość portu jaka jest ustawiona dla całego serwera. Możesz również określic `:*` które będą pasować do wszystkich portów na danym adresie. (Jest to zalecane kiedy używasz `_default_`.)

Zródła

http://httpd.apache.org/docs/

http://dreamnet.help.pl/linux/apache-ssl-server.php
http://helion.pl/ksiazki/recenzje/apache.htm
http://helion.pl/ksiazki/spisy/apache.htm
http://jakarta.apache.org/
http://pcworld.pl/artykuly/0778.html
http://php.weblogs.com/tuning_apache_unix
http://student.uci.agh.edu.pl/~pawha/www/index.html
http://www.adso.com.pl/%7Erob/LinuxHOWTO/Apache-Overview-HOWTO.html#toc1
http://www.apacheweek.com/features/ap2
http://www.bea.com/products/servers_application.shtml
http://www.devshed.com/Books/ProApache/
http://www.fantomaster.com/faarticles/modrewrite01.txt (aż do)
http://www.fantomaster.com/faarticles/modrewrite04.txt
http://www.filg.uj.edu.pl/~lb/apache/
http://www.jtz.org.pl/Inne/Apache/
http://www.kegel.com/nt-linux-benchmarks.html
http://www.linux.sky.pl/teksty/apache.html
http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO.html
http://www.linuxdoc.org/HOWTO/mini/Apache+SSL+PHP+fp.html
http://www.linuxdoc.org/HOWTO/mini/Apache-mods.html
http://www.linuxfan.com.pl/artykuly/apache.html
http://www.netcraft.com/survey/
http://www.netcraft.com/survey/servers.html
http://www.pckurier.pl/archiwum/artykuly/golachowski_krzysztof/2000_02_51/
http://www.pckurier.pl/archiwum/artykuly/szafranski_bohdan/2000_06_24/
http://www.pckurier.pl/webmaster/2000/czerwiec/teczynski/serwirt.html
http://www.software.com.pl/konferencje/Apache2000/wyk/d2w2.asp
http://www.stormerhosting.net/support/serverhelp/webserver/
http://www.szpital.grudziadz.net/squid/Welcome.html
http://www.webdevelopersjournal.com/software/apache_web_server.html
http://zls.mimuw.edu.pl/~alx/WWW/serwery.php
http://asp2php.naken.cc/
http://www.eddieware.org
http://community.turboLinux.com/cluster/
http://www.sun.com/clusters/
http://www.coyotepoint.com