Ръководство за web server Apache

Автор: Мартин Петров
Благодарности: www.dhstudio.eu

Съдържание

2.2.2. Инсталиране и конфигуриране на модули за Apache

2.2.2.1. Инсталиране на mod_php
2.2.2.1.1. Инсталиране на PHP от програмен код
2.2.2.1.2. Инсталиране на PHP от пакет
2.2.2.1.2.1. Инсталиране на PHP от DEB пакет
2.2.2.1.2.2. Инсталиране на PHP от RPM пакет
2.2.2.1.3. Инсталиране на PHP чрез други инсталатори
2.2.2.1.3.1. Инсталиране на mod_php чрез apt или apt-get
2.2.2.1.3.2. Инсталиране на PHP чрез yum

2.2.2.2. Инсталирането на mod_perl
2.2.2.2.1. Инсталиране на mod_perl от пакет
2.2.2.2.1.1. Инсталиране на mod_perl от DEB пакет
2.2.2.2.1.2. Инсталиране на mod_perl от RPM пакет
2.2.2.2.2. Инсталиране на mod_perl чрез други инсталатори
2.2.2.2.2.1. Инсталиране на mod_perl чрез apt или apt-get
2.2.2.2.2.2. Инсталиране на mod_perl чрез yum

2.2.2.3. Инсталирането на mod_python
2.2.2.3.1. Инсталиране на mod_python от пакет
2.2.2.3.1.1. Инсталиране на mod_python от DEB пакет
2.2.2.3.1.2. Инсталиране на mod_python от RPM пакет
2.2.2.3.2. Инсталиране на mod_python чрез други инсталатори
2.2.2.3.2.1. Инсталиране на mod_python чрез apt или apt-get
2.2.2.3.2.2. Инсталиране на mod_python чрез yum

2.2.2.4. Инсталиране и конфигуриране на mod_cband
2.2.2.4.1. Инсталиране на mod_cband под Дебиан 4.0 (Etch)
2.2.2.4.2. Директиви на mod_cband
2.2.2.4.3 Примерни конфигурации на mod_cband
2.2.2.4.4 Извеждане на уеб статистика чрез mod_cband

2.3. Сигурност на Web сървър Apache
2.3.1. Опции на сървъра за документи и директории. Дефиниране настройки за контрол на достъп.
2.3.2. Конфигуриране контрол на ниво директория
2.3.3. Използване на iptables с цел повишаване нивото на сигурност на системата

2.4. Мониторинг и поддръжка на Apache

ГЛАВА 1. СТРУКТУРА И КОМПОНЕНТИ НА WEB СЪРВЪР

1.1. Същност на Web сървър. Видове и функции

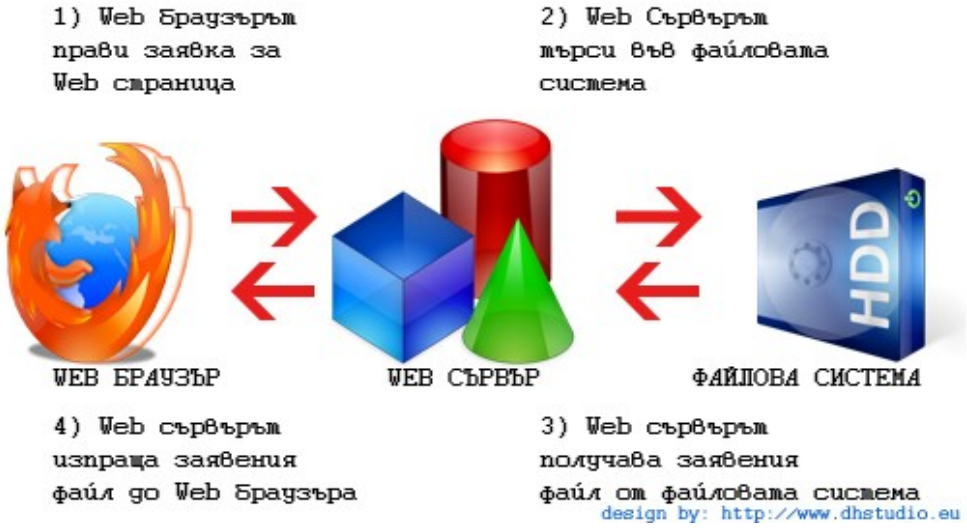
В забързаният свят на електронната търговия, с всяка минута се появяват нови технологии. Интернет прави света все по- малък и все по-тясно свързан. Накратко Web сървърите представляват хранили на файлове на Web сайтове в Интернет.

Технически погледнато, Web сървърът приема, обработва и отговаря на HTTP заявки. Тези заявки се подават от Web браузърите, които клиентските компютри използват за комуникиране, изпращане и получаване на информация по Интернет. Взаимоотношението между Web сървъра и Web клиента се нарича взаимоотношение клиент-сървър.

Повечето Web сървъри следват една и съща логика на работа. Процесът на работа може да се раздели на четири стъпки, като участниците в процеса са Web сървърът и Web браузърът. Стъпките са следните:

1. Клиентският компютър използва Web браузър за свързване с Web сървъра и прави заявка за Web страница.
2. При получаването на заявката, Web сървърът открива съответстващия файл или програма във файловата система.
3. След като открие файла, Web сървърът го извлича от файловата система.
4. Web сървърът изпраща файла на Web браузъра, след което той се изпълнява на клиентския компютър. Процесът на заявка и отговор е показан на фигура 1.1.

Процесът заявка- отговор между Web сървър и Web браузър



Фиг. 1.1

След кратко описание процеса на комуникация между Web сървър и Web браузър е необходимо да се знае, че за да се свърже компютър към Web сървър трябва да се зададе валиден локатор на ресурси (Uniform Resource Locator, URL). Web браузърът, инсталиран на клиентския компютър, използва URL (или още URL адрес), за да изпрати заявката до сървъра. Този URL съдържа уникален домейн, който различава сайтовете един от друг. Например, за да се извика страницата с често задавани въпроси, която е качена на официалния сайт на Apache, трябва да се зададе следния URL адрес: http://www.apache.org:80/faq.html. Този URL адрес може да се раздели на следните четири части:

Протокол: Всеки URL започва със задаване на протокола, който ще бъде използван за комуникация по мрежата. В посочения по- горе URL е използван протоколът HTTP (Hyper-Text Transfer Protocol). HTTP е най- използваният протокол за връзка с Интернет.

Име на сървър: След протокола всеки URL си задава името на сървъра, което в случая е www.apache.org

Порт: Номерът на порт се задава след името на сървъра и е предшестван от двоеточие. Ако портът не бъде зададен, той по подразбиране е с номер 80 за всички HTTP заявки, които се изпращат към Web сървъра.

Име на файл: Накрая се задава името на файла, който трябва да бъде разгледан. В този случай името е faq.html . Можете да се определи и точно местоположение на файла, ако той се намира в друга директория.

След като вече се изгради представа за това как Web браузърът и Web сървърът комуникират, нека да се види как Web сървърът позволява разглеждането на Web страницата от браузъра на клиентския компютър. Процесът протича по следния начин:

1. Докато изпраща заявката, браузърът преобразува името на сървъра (http://www.apache.org) в IP адрес. Браузърът се свързва със сървъра, използвайки този IP адрес.
2. След като установи връзка със сървъра посредством IP адреса, клиентският компютър изпраща заявки до сървъра. Връзката се установява при порт 80, който по традиция е запазен за HTTP операции, които протичат в мрежата.
3. Обикновено заявката, изпратена от клиента, е тип GET. Заявката GET търси файла с име faq.html

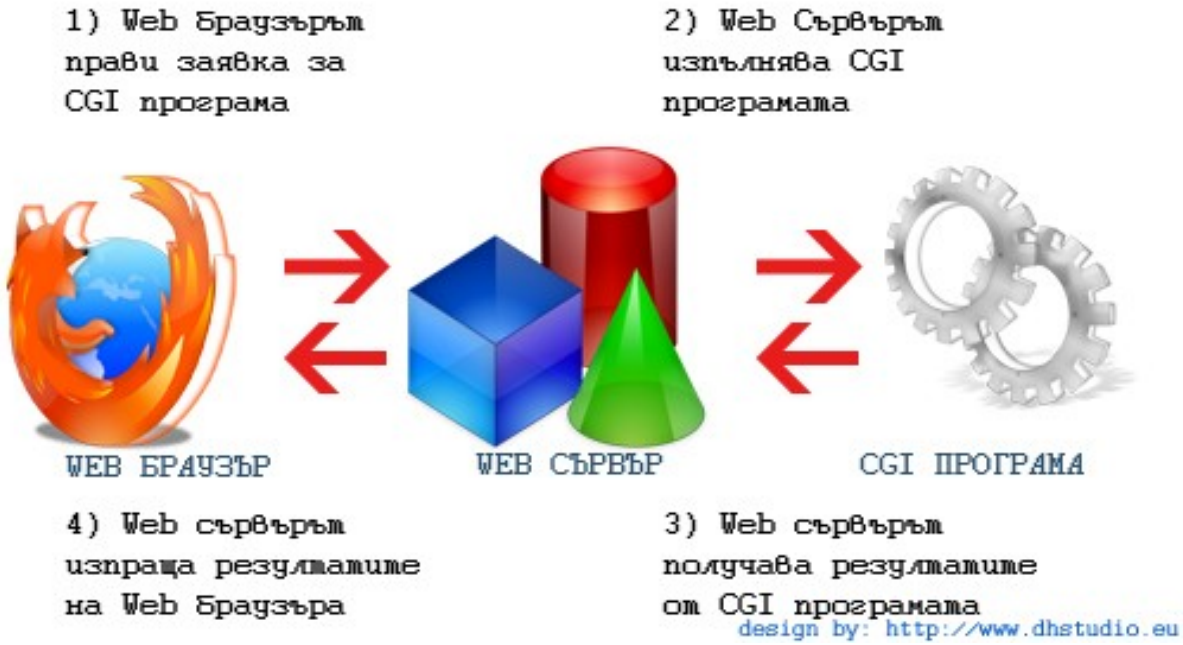
4. След като Web сървърът приеме заявката за файла, съответното съдържание на файла се изпраща до клиента, заявил страницата. В конкретния случай съдържанието е HTML текст.
5. Браузърът на клиентския компютър превежда HTML текста и показва Web страницата. Web браузърът форматира страницата в съответствие с HTML таговете, които участват в програмния код на HTML файла.

Съдържанието на даден Web сайт може да бъде както *статично* , така и *динамично.* В ранните години на Интернет, страниците се създаваха с HTML и съдържаха предимно *статично съдържание.* Web страниците, които се създават с HTML , се наричат *статични страници,* защото след като станат готови за изпращане до клиента, те не могат да бъдат променяни, преди да достигнат до клиента. Това направи Web страниците изключително сковани, по- малко интерактивни и не създаде динамичното съдържание и днес популярността на една Web страница зависи от това доколко интерактивна и визуално привлекателна е тя. Динамичното съдържание помага на програмиста да реализира тази цел. Съдържанието на динамичните Web страници може да бъде променено, докато потребителят все още общува със сървъра. С други думи, динамичните Web страници се създават на основата на входящата информация от потребителите.

CGI (Common Gateway Interface) скриптовете са мощно средство за създаването на динамично съдържание в Интернет. Ето как работи един CGI скрипт:

1. Потребителят прави заявка за CGI Програма, която е разположена на Web сървъра
2. Web сървърът извиква CGI скрипта.
3. Web сървърът възпроизвежда резултатите от CGI програмата и ги изпраща до Web браузъра на потребителя. Обикновено резултатите от CGI програмите са в HTML формат, така че Web браузърът да може да ги интерпретира. Фигура 1.2 показва как протича този процес:

Показване на динамични Web страници.



Фигура 1.2

1.2. Видове Web сървъри

Софтуерът може да бъде свободен (с отворен код) или лицензионен (със затворен код). Свободният софтуер не е просто безплатен. Неговият код може свободно да се разпространява, променя и приспособява за определени цели. От друга страна лицензионния софтуер също може да бъде безплатен (в някои случаи). Въпреки това няма достъп до програмния код и не може да се променя или адаптира конкретни нужди. В повечето случаи трябва да се плати, за да се получи лиценз за лицензионния софтуер.

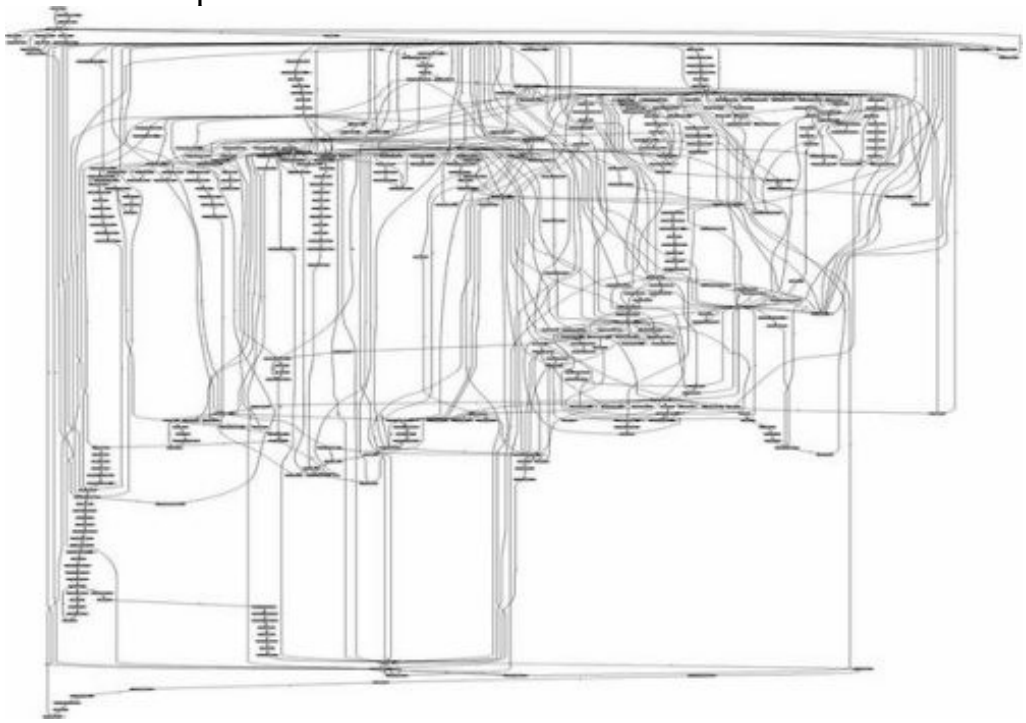
Web сървърите могат да се класифицират на свободно разпространяеми и лицензионни. Първата група включва Web сървърите, които могат да се придобият, без да се разпространяват или променят. Втората група включва Web сървърите, за които потребителите трябва да заплатят лиценз (в повечето случаи) и чийто програмен код те не могат да променят или адаптират за своите нужди.

1.2.1. Свободни Web сървъри

Най известният сред свободните (с отворен код) Web сървъри е Web сървър- Apache. Въпреки че Apache е най- широко използваният Web сървър в тази категория, популярността на няколко други Web сървъра бързо расте. Ето няколко от най- известните Web сървъри от тази група:

- Apache

Apache е най- популярният Web сървър с отворен код. Apache е очевидният избор на множество организации поради множеството си полезни възможности. Web сървърът Apache предоставя скорост, преносимост, стабилност и сигурност. Можете да го изтеглите безплатно от официалната му Web страница: http://httpd.apache.org/ . На следващата фигура може да се наблюдава управлението на системните извиквания от Apache:



Фигура 1.3

- Boa

През 1991 г. Пол Филипс създава Web сървъра Boa. Лари Дулитъл обаче, който в момента отговаря за неговата поддръжка, е направил значителни подобрения. Web сървърът Boa е идеален за администратори на Web сървъри, чиято основна цел е скоростта и сигурността. Въпреки това постигането на тази цел не е лесно, тъй като за да използват Boa , администраторите трябва да пожертват част от функционалността. За разлика от други Web сървъри, Boa е еднозадачен HTTP сървър. Той нито се разклонява за различните HTTP заявки, нито пък разклонява всяка отделна връзка. Можете да изтеглите Boa от функционалния му Web сайт: http://www.boa.org

- Red Hat Content Accelerator (RHCA)

Red Hat Content Accelerator (RHCA) е мощен HTTP сървър с отворен код, работещ в режим на ядрото на операционната система GNU/ Linux. Той може да бъде изтеглен от официалната Web страница на Red Hat .

RHCA може да обслужва статично и да кешира динамично съдържание. За повече информация за сървъра RHCA можете да посетите следната Web страница: http://www.redhat.com/docs/manuals/tux/TUX-2.2-Manual/intro.html

- **Mathopd**

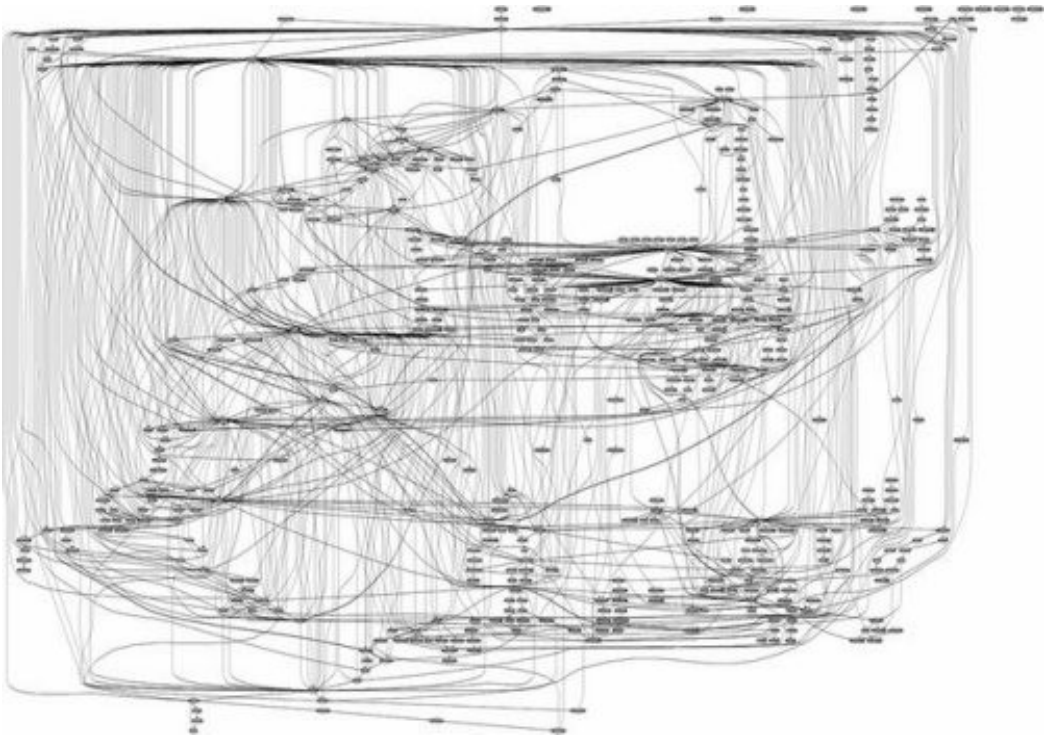
Mathopd е един интересен Web сървър, който е разработен с цел предоставяне на минимален набор от възможности. Можете да отчитате това като недостатък или като ограничен дизайн, но Mathopd работи само с операционните система и Unix и GNU/Linux. Ако единственият критерий, който използвате при избора на сървър, е производителността, можете да сте сигурни, че Mathopd се нарежда сред най- добрите алтернативи. Програмният код на Web сървъра е написан така, че той може да обслужва множество едновременни връзки. Освен това Mathopd заема минимален обем от паметта на компютъра. Можете да изтеглите този Web сървър безплатно от следната страница: http://mathop.diva.nl/download.html

1.2.2. Лицензионни Web сървъри

Всички Web сървъри, за които е необходимо закупуването на лиценз, преди да можете да ги използвате, спадат към групата на лицензионните Web сървъри. Администраторите на Web сървъри, които се интересуват повече от използването на лицензионни Web сървъри, могат да избират от дълъг писък. IIS, IBM, Zeus, Roxen, iPlanet и Stronghold са най-известните сървъри в тази категория. Сега ще разгледаме някои лицензионни Web сървъри:

- **Microsoft IIS**

Microsoft Internet Information Server е популярен Web сървър, който съществува от няколко години. Microsoft IIS е предварително инсталиран в операционните системи Windows NT, Windows 2000 и Windows XP. IIS е мощен Web сървър, който притежава множество интересни характеристики. Единственият му недостатък е, че не е преносим и ви принуждава да използвате някоя от следните платформи: Windows NT, Windows 2000 или Windows XP. IIS не работи с други операционни системи като Unix, GNU/ Linux и Solaris. Повече информация за IIS може да се открие на адрес: http://www.microsoft.com
На следващата фигура може да се наблюдава управлението на системните извиквания от IIS:



Фигура 1.4

- **IBM**

IBM HTTP Server е IBM версията на Web сървъра Apache. Преди няколко години IBM обяви, че възнамерява да поддържа Web сървъра Apache за своите решения за електронна търговия. Резултатът е, че IBM HTTP Server включва множество решения като например IBM Websphere Application Server. Можете да прочетете повече за този Web сървър на Web страницата: http://www3.ibm.com/software/webservers/httpservers/

- **Zeus**

Web сървърът Zeus е идеален за платформите GNU/Linux и Unix. Въпреки че повечето потребители предпочитат да използват безплатния софтуер, Zeuse една добра негова алтернатива. Част от неговите свойства са поддържането на Apache/NCSA/httpd съвместимост и лесното му конфигуриране. Единственият недостатък е заплащането за използване на този Web сървър. Може да научите повече за него на следната Web страница: http://www.spec.org/osg/web96

- **IPlanet**

Web сървърът iPlanet несъмнено е един от най- добрите Web сървъри с гарантирана висока производителност. Този продукт е резултат от съвместните усилия на Sun и Netscape. Web сървърът iPlanet най- често се използва за високопроизводителен мултипроцесорен хардуер. Тъй като е разработен основно за програмисти на Java, максимална полза от този сървър могат да извлекат тези, които разработват приложения на

програмния език Java. Преносимостта на iPlanet не е проблем. След платформите, на които работи iPlanet, са Solaris, GNU/Linux, Windows и други. Днес Web сървърът iPlanet се нарича Sun ONE. Повече информация за този Web сървър може да се получи на адрес: http://www.sun.com/software/iplanet/products/iplanet_application/home_ias.html

- **Stronghold**

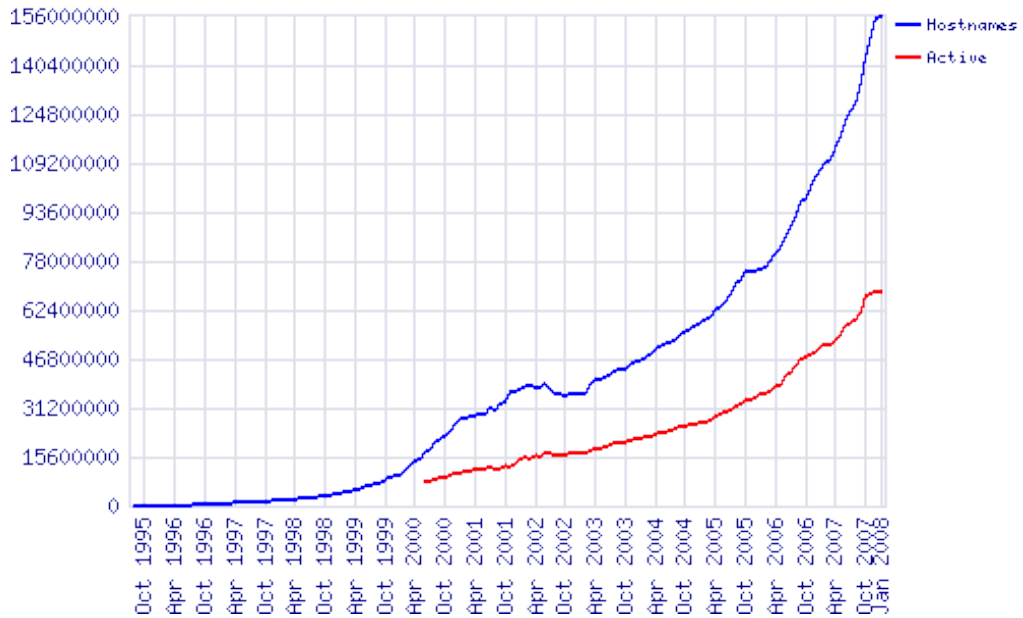
Web сървър идеален за организации с големи финансови средства. Web сървърът Stronghold е разработен аналогично на сървъра Apache, но е разработен върху Secure Socket Layer (SSL) за гарантирането на максимална сигурност. За повече информация: http://www.redhat.com/software/apache/stronghold/

След разглеждане на популярните Web сървъри е редно да се разгледа направеното проучване от Netcraft за ползваемостта на Web сървърите.

През януари 2008 година Netcraft проучва отговорите от 155.583.825 сайта, отразявайки много по-бавен растеж на само 354 хиляди места в сравнение с миналия месец където увеличението беше 5.4 милиона.

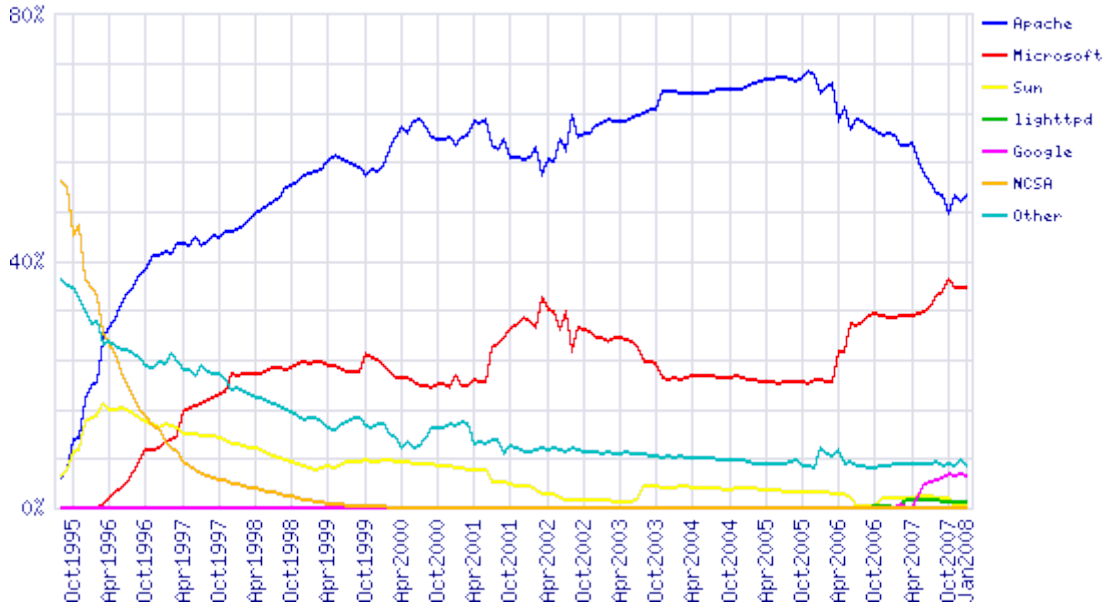
Apache продължава своето възстановяване след като претърпя падение на своя дял през миналите осемнадесет месеца. Неговият дял е бил отрицателно засегнат през онзи период от нарасналия брой blog сайтове в изследването на големите доставчици като Майкрософт и Гугъл, използвайки техния софтуер.

Всички сайтове по всички домейни Август 1995- Януари 2008



Фигура 1.5

Пазарен дял за ТОП сървъри по всички домейни Август 1995 – Януари 2008-02-21



Фигура 1.6

Топ разработчици

Разработчик	Декември 2007	Процент	Януари 2008	Процент	Промяна
Apache	76,945,640	49.57%	78,735,581	50.61%	1.04
Microsoft	55,509,223	35.76%	55,709,926	35.81%	0.05
Google	8,558,256	5.51%	8,290,471	5.33%	-0.18
lighttpd	1,521,250	0.98%	1,536,981	0.99%	0.01
Sun	588,997	0.38%	557,673	0.36%	-0.02

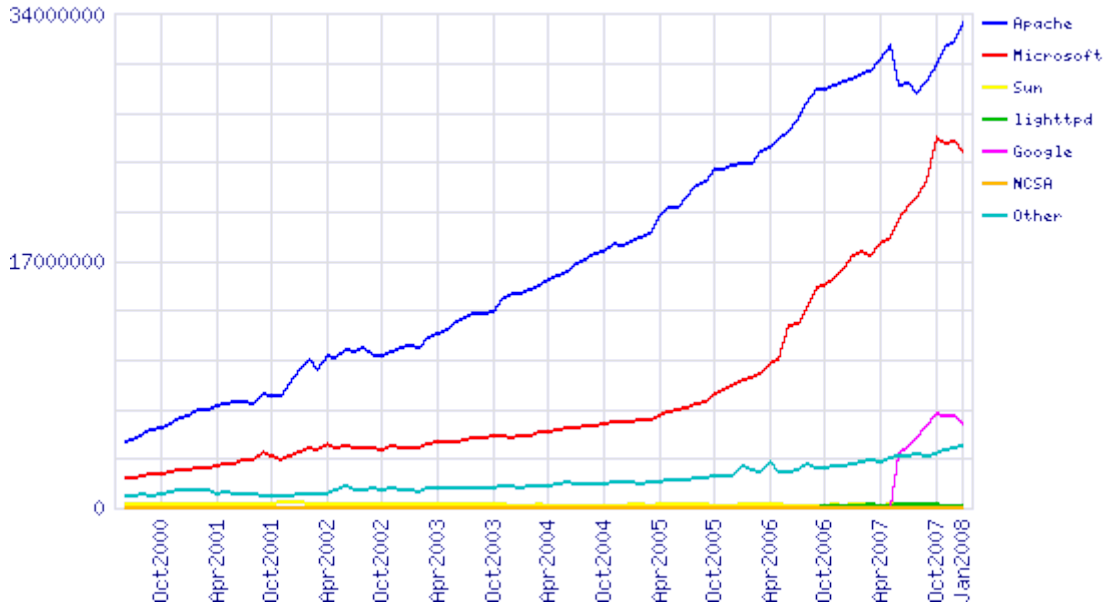
Фигура 1.7

Активни сайтове

Разработчик	Декември 2007	Процент	Януари 2008	Процент	Промяна
Apache	32,124,072	47.20%	33,344,184	48.84%	1.64
Microsoft	25,176,286	36.99%	24,612,182	36.05%	-0.94
Google	6,343,369	9.32%	5,745,468	8.42%	-0.90
Sun	191,919	0.28%	185,858	0.27%	-0.01
lighttpd	73,703	0.11%	75,114	0.11%	0.00

Фигура 1.8

Общо за всички активни сървъри по всички домейни Юни 2000- Януари 2008



Фигура 1.9

Изследването е публикувано от Netcraft в 2:08 ч. на 28 януари 2008 г.

1.3. Функции на Web сървър

Web сървърите изпълняват множество функции извън приемането на HTTP заявките. Тези функции могат да бъдат обобщени до следните:

- **Контрол на достъпа.**

Потребителите трябва да могат да използват сами тези ресурси на Web сървъра, за коти имат разрешение. Всеки потребител трябва да има достъп само до тази група от файлове, които му принадлежат При условие че са подходящо конфигурирани, Web сървърите позволяват само на упълномощените потребители да използват файловата система на сървъра. Този механизъм се нарича *контрол на достъпа.* Контролирането на достъпа може да стане по различни начини, като например настройка на подходящите разширения за файлове, директории и прилагането на ограничения, свързани с името на хоста/IP адреса. Контролирането на достъпа може да се извърши чрез проверка на идентичността, при която потребителят трябва да въведе валидно потребителско име и парола, за да използва ресурсите на сървъра.

- **Обработка на страниците от страна на сървъра.**

Обработката е процес, при който Web сървърът заменя имената на полетата със съответните стойности от информацията, въведена от потребителя. След като обработи документи, Web сървърът го изпраща на клиентския компютър.

- **Поддържане на журнали.**

За да се подсигурят срещу възможни капани или дефекти в производителността и работата на Web сървърите, последните редовно използват механизъм за поддържане на журнали. Тези журнали помагат на администраторите на Web сървъри да проследят функционирането на Web сървъра, да анализират проблемите (ако такива съществуват) и да приемат коригиращи действия за отстраняването на тези проблеми. Обикновено журналите се поддържат за наблюдение на успешните и неуспешните опити за достъп и грешките.

- **Изпълнение на CGI скриптове и други програми**

Друга важна черта на Web сървърите е, че те позволяват изпълнението на CGI скриптове и програми. Тези CGI скриптове и програми обикновено се използват за оценка на информацията, която се въвежда в Интернет формат.

1.4. Протокол Hypertext Transfer Protocol (HTTP)

Концепцията за HTTP е толкова обхватна, че само за нея може да се напише отделна книга. Този раздел покрива само най- важните концепции, отнасящи се до HTTP протокола, и е ограничен до това, което трябва да знаете във връзка с функционирането на Web сървърите.

Първата версия на HTTP, HTTP/0.9 е напусната през 1991 г. Това обаче не е официална версия. Протоколът е прост и се използва за пренос на необработени данни по Интернет. Web сървърите, които приемат HTTP заявки по това време, отговарят на прости заявки от рода на :`11W``2`12
GET /inex.html

В този случай, ако файлът с име index.html е в главната директория с документи на Web сървъра, съдържанието на тази директория ще бъде показано на Web страницата. Ако този файл не бъде открит, Web сървърът ще върне съобщение за грешка. Не след дълго тази версия остарява и през май 1996 г. Е пусната заявка за коментари (Request For Comments- RFC) за протокола HTTP 1.0. Тази версия включва нова възможност. Тя използва т.нар. заглавия (headers) . Заглавието е текст, който се прикрепва към пакетите от данни и служи за описване на прехвърляните данни. Най – срещаното заглавие изглежда така:
Content-Type: text/html

Това заглавие показва, че пренасяните данни съдържат текст на програмния език HTML. Преди изпращането на заявката към Web сървъра, браузърът прикрепва заглавия към пакетите от данни. Последната версия на HTTP е HTTP 1.1. Тази версия е официално обявена през юни 1999 г. С пускането на RFC 2616.

HTTP/1.1 е последната версия на HTTP и включва някои нови възможности. Повечето от днешните Web сървъри, включително Apache и IIS, поддържат HTTP/1.1. Новите възможности които предоставя HTTP/1.1 са следните:

- Разпознаване на името на хоста

Разпознаване на името на хоста е една от най- важните характеристики, вградени във версията HTTP/1.1 При предходните версии на HTTP, Web сървърът никога не познава името на хоста, определен в дадения URL. При HTTP/1.1 всяка заявка трябва да зададе името на хоста. Например, ако направите заявка за URL- http://www.apache.org, сървърът трябва да може да разпознае името на хоста www.apache.org. Това позволява на администратора на Web сървъра да избегне определянето на уникални IP адреси за всеки домейн. Това предполага, че няколко домейна ще могат да споделят един и същ IP адрес. Да разгледаме следния случай. Един потребител прави заявка за страница www.apache.org, а друг за www.php.net. В този случай съответните страници ще бъдат възпроизведени и показани на браузърите на всеки от потребителите, дори и в случаите когато те са всъщност два домейна, използващи един и същ IP адрес. Това става възможно благодарение на разпознаването на името на хоста. По- старите версии на HTTP щяха да се ръководят от IP адреса за показването на съдържанието на всеки от домейните. Това от своя страна изисква определянето на уникални адреси за избягване на конфликти.

- Нови методи за заявки

С появата на HTTP/1.1 бяха въведени нови четири метода за подаване на заявки. Протоколът HTTP/1.1 поддържа тези методи, освен всички други традиционни методи като GET, HEAD и POST. Тези методи за заявки са следните:

- OPTIONS
- DELETE
- PUT
- TRACE

- Постоянни връзки

Изключително ефективният механизъм на постоянните връзки е въведен при версията HTTP/1.1 . Web страниците, които днес се хостват в Интернет, включват нещо повече от съдържание. Една Web страница може

да бъде съставена от няколко свързани документа. Тези свързани документи могат да включват графични файлове, shockwave презентации, звукови файлове и дори видеоклипове. Когато зарежданата страница съдържа такива свързани документи, всеки от тях трябва да бъде зареден чрез отделна връзка. За да свали документите, браузърът трябва да бъде зареден чрез отделна връзка. За да свали документите, браузърът първо се свързва с Web сървъра и извлича документа, преди да прекъсне връзката. Този процес отнема доста време. Концепцията за постоянната връзка позволява на браузъра да зареди няколко свързани документа чрез една връзка, вместо да установява нова връзка със зареждането на всеки свързан документ. Въпреки, че връзката е само една, всички свързани документи се зареждат последовалтено един след друг. По подразбиране всички установени с протокола HTTP/1.1 връзки са постоянни, освен ако специално настроите браузъра си по друг начин.

- Кодиран трансфер чрез блокове

Кодираният трансфер чрез блокове е идеален механизъм, използван от протокола HTTP/1.1 за предаване на съобщения. Блоковият трансфер позволява на подателя на съобщението да раздели тялото на съобщението на случайни малки блокове. След разделянето на тялото на съобщението на блокове, всеки от тях се изпраща поотделно до получателя. Дължината на всеки блок е предварително прикрепена към него, а накрая на съобщението се подава блок с нулева дължина, който бележи неговия край. За да обозначи трансфера чрез блокове, подателят използва частта Transfer-Encoding от заглавието. Да разгледаме следния пример:

Transfer-encoding: chunked

В този пример зададеният параметър “chunked” (“чрез блокове”) означава, че подателят ще предава съобщението на блокове.
Освен предаването на съобщението чрез блокове, този механизъм помага на подателя да буферира отделните блокове на съобщението, вместо да буферира цялото съобщение. Така не се налага буферирането на цялото съобщение преди изпращането му до получателя. Веднага след като блокът се буферира, той бива изпратен на потребителя. Това спестява много време и получателят не трябва да чака дълго, преди да получи част от съобщението.

- Заявки за част от информацията

HTTP/1.1 позволява на браузъра да подаде заявка за части от даден документ. Задаването на обхвата от байтове позволява на браузъра да подаде заявка за една част от документ. Освен заявяването на части от документ, обхватите от байтове могат да се използват и за подновяване на

прекъснат трансфер. Можете да зададете обхвата в частта Range не заглавието, както е показано долу:

Range: bytes=400-600

Когато зададете обхвата 400-600, всъщност давате заявка частта от документа която заема последните 200 байта от целия документ.

- Кеширане

Кеширането е много полезен механизъм, който позволява запазването на Web страниците, които вече са свалени на клиентския компютър. Когато потребителят се опита да разгледа изтеглената Web страница, тя се зарежда от кеш паметта. Този механизъм спестява значително време и широчина на линията при тегленето на данни, като премахва необходимостта от повторен трансфер на вече извлечени данни. Друго преимущество на кеширането е намаляването на натоварването на Web сървъра. Повечето Web браузъри, прокси сървъри и Web сървъри поддържат кеширането от страна на клиента.

Протоколът HTTP 1.1 внася ново измерение в кеширането на данни, като същевременно запазва оригиналните възможности за поддържане на кеширането на протокола HTTP, версия 1.0, HTTP/1.1 има множество нови възможности. HTTP/1.1 предлага не просто нови възможности, а и една подобрена версия на възможностите за кеширане, съдържащи се в HTTP/1.0

Според спецификациите на протокола HTTP/1.1, всеки запис в кеш паметта е *валиден* до изтичане на определен срок. След този срок записът се обявява за *остарял.* Всички остарели записи трябва да бъдат валидирани отново от Web сървъра, преди да бъдат върнати като отговор на клиентката заявка.

HTTP/1.1 отменя начина, по който HTTP/1.0 валидира даден запис в кеш паметта. При HTTP/1.0 кешът използва частта If-Modified-Since за валидацията на запис в кеш паметта. Тази част използва абсолютно време с разделителна способност от една секунда, което предизвиква грешки при неуспешна синхронизация на часовниците. HTTP/1.1 се справя с проблемите при кеширането, като използва концепцията на *информационен етикет*. Информационният етикет представлява низов ключ на елемент от кеша, който може да се променя от сървъра-източник на информацията. При условие че етикетите на два отговора са подобни, според спецификацията може да се заключи, че те са идентични. Web сървърът с вграден HTTP/1.1 протокол използва частта Etag. Протоколът HTTP/1.1 включва заглавия като If-None-Match, If-Unmodified-Since и If-match. Клиентите могат да представят един или повече информационни етикета за определен ресурс, като използват частта If-None-Match

- Разширяемост към други протоколи

Протоколът HTTP/1.1 съдържа ново поле в заглавието наименование Upgrade. Целта на това поле е да се избегнат проблемите с несъвместимостта на протоколите. С бързия напредък на технологията, вероятно скоро ще станем свидетели на появата на нов протокол. Частта *Upgrade* гарантира лесното инсталиране на приложения, дори в случай на въвеждане на нов протокол. Когато клиентът изпрати заявка до Web сървъра, той може да зададе другите протоколи, които поддържа, като използва частта *Upgrade*. По този начин сървърът ще има възможността да превключи към другия протокол, в случай че клиентът не поддържа протокола, който сървърът е използвал в началото.

1.5. История, характеристики и архитектура на Web сървър Apache

1.5.1. История на Apache

Apache е HTTP Web сървър, разработен от група доброволци. Основата на Web сървъра Apache е HTTP сървър за публични домейни, разработен от Роб МакКул в Националния център за свръх изчислителни приложения (National Center for Supercomputing Applications) към Университета на Илинойс, Урбана Шемпейн. Скоро след създаването му множество Web специалисти започват да създават свои собствени разширени версии на Web сървъра, като внасят някои неизбежни поправки на програмни грешки. По това време липсва само свободно разпространение. През февруари 1994 г. Брайън Бехлендорф и Клиф Сколник започват да се свързват с хора от целия свят чрез електронни пощенски списъци, за да наберат мнения за сървъра Apache. Няколко доброволци си сътрудничат и написват програмния код на Apache. Името на Web сървъра Apache произхожда от фразата “a patchy” (програмистите, които написват кода на Apache, правят подобрения към кода под формата на пачове). По- късно се учредява групата Apache. Тя се състои от осемте основни члена на екипа от разработилите го програмисти: Брайън Бехлендорф, Рой Фийлдинг, Роб Хартил, Дейвид Робинсън, Клиф Сколник, Ранди Тербуш, Робърт Тау и Андрю Уилсън. Резултатът от общите им усилия е първото издание на Web сървъра Apache, 0.6.2 през април 1995 г.

Въпреки успеха на този сървър, търсенето на съвършенството е достатъчен стимул за тези програмисти да преоформят Web сървъра. След фундаментални промени и добавянето на много нови възможности, през декември 1995 г. Излиза нова версия на сървъра- Apache 1.0. Тъй като тази версия преминава през множество бета тестове преди нейното официално издание, тя се оказва по- стабилна от предшестващата я версия.

През 1999 г. членовете на групата Apache създават *Софтуерната фондация Apache*. Тази фондация е създадена, за да осигури финансова и правна подкрепа за разработването на Web сървъра Apache.

Web сървърът Apache, който вече е ориентир при разработването на Web сървъри, печели тази позиция благодарение на усилената работа на програмисти от целия свят. Фактът, че програмния код на Apache е отворен, позволява на програмисти да разглеждат кода и да предложат ценни идеи за подобряването на неговия дизайн. Това движение все още съществува и се нарича *Проект за HTTP сървъра Apache*. Проектът за сървъра Apache е инициатива за подобряване на сървъра, която позволява на хората, независимо от тяхното местоположение, да дадат своя принос във вид на допълване на програмния код, идеи или документация.

1.5.2. Характеристики на Apache

- Свободно разпространяван

Можете да се сдобиете с Apache безплатно от неговата официална Web страница (http://httpd.apache.org) . Всичко, от което се нуждаете, за да изтеглите софтуера, е Интернет връзка. Няколко огледални сайта позволяват изтеглянето на софтуера от сървър, който е разположен по близо до вас. Връзките към тези огледални сайтове са изброени на страницата на Apache. Можете да изберете подходящата огледална страница в зависимост от вашето местоположение.

- Леснота на инсталиране

Apache има доста популярна характеристика, а именно леснотата при инсталиране. Apache може да бъде инсталиран само с няколко команди(в GNU/Linux-терминал, MS Windows-command prompt).
Apache се разпространява в две форми:

1) Предварително компилирана форма.

Предварително компилирана форма (двоичен код) на софтуера е налична под формата на RPM (RedHat Package Manager) файл за дистрибуцията на Linux от Red Hat. Освен това са налични и други формати за дистрибуцията на Linux от Red Hat. Освен това са налични и други формати за диструбиция на двоичен предварително компилиран софтуер. Форматите се различават за различните операционни системи. Двоичните кодове са налични и за различни дистрибуции на една и съща операционна система. Например, за дистрибуцията на Linux от Mandrake, Debian и Red Hat има отделни двоични версии. Този вид инсталация е идеална за хората, които имат елементарни познания за операционната система Linux.

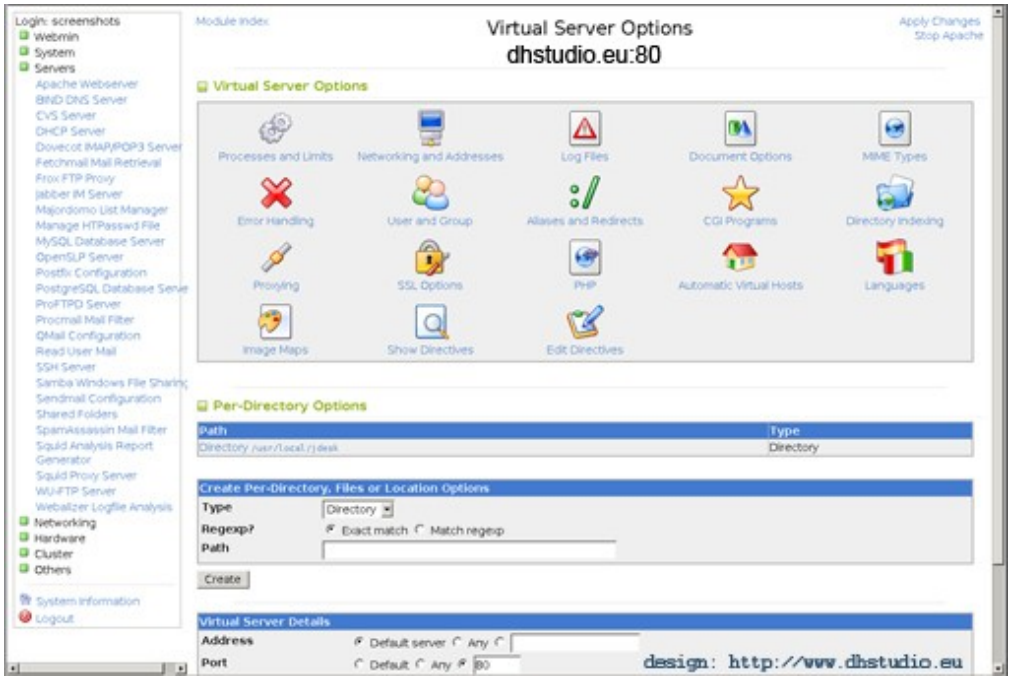
2) Програмен код

Можете да инсталирате тази форма, като използвате група команди от конзолата на Linux. Тази форма на инсталация изисква компилиране на файл с програмния код. Експертите предпочитат тази инсталация, тъй като тя позволява задаването на няколко опции за инсталиране и настройването на параметрите на инсталацията.

- Леснота на конфигуриране

Сървърът Apache се конфигурира изключително лесно, стига да можете да използвате текстов редактор в Linux. Въпреки че опциите за инсталиране са много, основната конфигурация е елементарна. Трябва да конфигурирате само един файл, за да накарате Apache да проработи. Всичко, което трябва да направите, за да конфигурирате Apache, е да посочите няколко местоположения за файлове, да зададете име на вашия сървър и да определите главната директория за документи, която ще съдържа файловете за Web страниците.

По- лесен вариант е използването на графичен интерфейс за конфигуриране на Apache. Съществуват няколко графични инструмента, осигурени от различни организации , които служат за конфигурирането на Apache. Един от популярните инструменти е Webbing. Можете да го изтеглите от: http://www.webmin.com/. Визуализация на *Webmin* може да се наблюдава в следващата фигура:



Фиг 1.10

- Висока производителност и ниско натоварване на ресурсите

Сървърът Apache работи с предварително резклоняващ модел. Предварително разклоняващ модел е този, при който разклоняването (стартирането на процес) става дори преди изпращането на заявка до сървъра. Резултатът е, че процесите на сървъра чакат, за да отговорят на заявките. Някои други Web сървъри се основават на различен модел. Те стартират процеса едва когато получат заявката. След отговора на заявката процесът спира. След като същият или друг потребител изпрати нова заявка, целият процес се стартира отначало. Резултатът е многократно спиране и рестартиране на процесите с получаването на всяка нова заявка. Повтарящото се стартиране на процес може да се окаже скъпо струващо решение, особено в среда, базирана на Unix. Решението на този проблем е Apache. Когато използват Apache, сървърите могат да бъдат предварително настроени да очакват заявките. В този случай Apache може да отвръща по-бързо на подадените заявки, отколкото другите Web сървъри, които не са предварително разклонени.

- Отворен код

Често хората тълкуват погрешно понятието *отворен код.* Терминът отворен код означава, че програмният код на определено софтуерно приложение е наличен и безплатен. Това определение обаче не показва дали този код може да бъде променян или използван многократно. Да вземем за пример програмния език Java. Програмният код на този език е наличен. Въпреки това програмният код, написан на Java, не може да се променя според нуждите на потребителите. Дори незначителната промяна на програмния код изисква от потребителите необходимите разрешения. Не такъв е случаят с Web сървъра Apache. Всеки потребител може да се сдобие, да използва, да променя или да презаписва програмния код на Web сървъра Apache според собствените си нужди. От това следва, че програмистите могат да видоизменят кода и да проектират Web сървъра съобразно собствените си изисквания.

- Дизайн, позволяващ приспособимост

Вече подчертахме, че програмният код на Apache може да се променя. Това свойство помага при приспособяването на дизайна на Apache. Освен това можете да програмирате собствени модули за Apache и по този начин да разширите неговата функционалност. Можете да приспособите Apache към вашите потребности, използвайки модула Apache Module API. Можете да използвате Apache Module API, за да разработите модули на езиците C и

Purl. След това можете да добавите тези модули към Apache, за да добавите нови функции към първоначално вградените в него функции.

- Сигурност

Apache притежава набор от опции за сигурност, които го правят по-слабо уязвим към заплахи. В характеристиките на сигурността на Apache се включват механизми за проверка на идентичността (автентикация) и Secure Sockets Layer (SSL). Apache поддържа широка гама от сървъри за бази от данни, включително сървърите Oracle и MySQL. Тези сървъри за бази от данни могат да съхраняват огромни списъци с потребители и са ефективен механизъм за проверка на идентичността.

Ако желаете, можете да добавите поддръжка на SSL към Apache, като се прибави модула mod_ssl. SSL е изпробван от времето и универсално приет механизъм, използвам от Web сървърите в Интернет за гарантиране на сигурността. SSL използва цифрови сертификати и криптиране за подсигуряване на трансфера на данни по Интернет.

Освен това няма широко разпространени вируси, които да представляват заплаха за Apache. Въпреки популярността му, Apache има съвсем малко открити уязвими страни в сравнение с другите Web сървъри

- Поддържане на HTTP/1.1

Apache е Web сървър, съвместим с протокола HTTP/1.1. Това означава, че той поддържа всички нови възможности на HTTP/1.1 като поддръжка на виртуални хостове, непрекъснати връзки, кеширане на ресурси, зареждане на клиентски файлове, разширено докладване на грешки и преговори на съдържанието. Поддържането на протокола HTTP/1.1 прави Apache по-гъвкав и предпочитан Web сървър.

- Поддържане на програмни езици

Apache поддържа дълъг списък от скриптови езици, работещи на сървъра, които могат да се използват за програмиране в мрежата. Езиците, поддържани от Apache, са PHP(Hypertext Preprocessor), ASP (Active Server Pages), JSP (Java Server Pages) и много други.

- Прокси кеширане

С течение на времето Apache започва да се използва и като прокси сървър за общи цели. С въвеждането на модула mod_proxy, вече можете да използвате Apache като ефективен механизъм за прокси кеширане. Когато използвате Apache за прокси кеширане, Web сървърът може да кешира

съдържанието, изтеглено от отдалечени сървъри. Това е изключително полезно за клиентски компютри в intranet, защото спестява време за изтегляне на данни и оптимизира мрежовия трафик. След кеширане на Web съдържанието не се налага повторно свързване със сървъра-източник при изпращането на заявка от клиентския компютър. Вместо това се използва запазеното в кеш паметта съдържание.

- Общи динамични обекти

Apache поддържа механизъм, наречен общи динамични обекти. Този механизъм позволява зареждането на допълнителни модули по време на работа на Web сървъра. Това означава, че не е необходимо да компилирате предварително целия сървър всеки път, когато решите да добавите или отстраните специфични за Apache възможности.

- Поддръжка на Windows

Apache поддържа платформи, базирани на Windows, като Windows NT, Windows 2000, Windows 98 и Windows 95. Въпреки, че поддръжката е вградена и Apache работи на тези платформи, те не се смятат за среди с идеална производителност.

- Мащабност

Можете да се настрои Apache така, че да се хостват множество Web страници на един сървър. Това става със създаването на множество виртуални хостове. Прилагането на виртуални хостове е идеално за предлагането на множество Web услуги от един компютър. Въпреки че имената на хостовете и URL адресите на Web страниците се различават, те могат да функционират на един компютър. Тази опция е полезна за Интернет доставчиците, които желаят да хостват множество сайтове при минимални разходи.

- Преносимост

Преносимостта е друга важна характеристика на Web сървъра Apache. Като резултат от неговата преносимост, можете да го инсталирате на почти всяка платформа, включително Linux, UNIX, Windows, Solaris, BeOs, както и на mainframe.

1.5.3. Архитектура на Apache

Дизайнът на Apache е модулен дизайн (дизайн, който дава възможност за разширяване на функциите, при необходимост, чрез добавянето на модули). Този дизайн е с предимство пред монолитния дизайн (негъвкав дизайн, който не позволява лесното разширяване на функциите).

Съществуват много Web сървъри, които са по- бързи от Apache. Тези сървъри са лишени от всякаква функционалност с цел увеличаване на скоростта им, за разлика от Apache където той притежава дълъг списък от опции и функции.

Сега ще разгледаме особеностите при функционирането на Web сървър Apache. За целта трябва да се запознаем с много нишковите и много процесните Web сървъри.

С пристигането на HTTP заявката, Web сървърът започва да търси заявения от клиента ресурс. Докато Web сървърът е зает, същите или други клиентски компютри могат да изпратят нови заявки. Web сървърът може да пренебрегне или да обработи успоредно тези заявки.

Web сървърите, които пренебрегват или нареждат на опашката получените нови заявки, се наричат *едно нишкови*. Това означава, че те не могат да се справят с натоварения при Web сървъра трафик. Тези Web сървъри обаче са идеални за сайтове, които имат слабо натоварен или умерен трафик. Някои добри примери за едно нишкови сървъри са Web сървърите httpd,Medusa и Zeus.Web сървърите, които обслужват едновременно новопостъпилите заявки, могат да обработват тези заявки по два начина. Те могат да започнат нов процес или да стартират нова нишка от първоначалния процес. Web сървърите, които започват нов процес при всяка нова заявка, се наричат *многопроцесни* Web сървъри, докато тези, които образуват нова нишка в рамките на основния процес, се наричат *многонишкови* Web сървъри.

IIS на платформата MS Windows е пример за многонишков Web сървър. От друга страна, Apache на платформата GNU/Linux е пример за многопроцесен Web сървър. Въпреки това, поради ограниченията на платформата MS Windows (а именно, липсата на поддръжка за *разклоняване*- стандартен начин за стартиране на процес при GNU/Linux), Apache работи само в многонишков режим на MS Windows платформата.

След като разгледахме какво представляват многопроцесните сървъри сега ще разгледаме принципът на работа на процесите. Вече споменахме, че стартирането на Apache изисква няколко процеса. Стартирането включва два процеса: *родителски* и *дъщерен*. Родителският процес е главният процес, от който могат да се родят множество дъщерни процеси. Родителският процес е този, който получава всяка заявка, изпратена до сървъра Apache. След като получи заявката, родителският процес я препраща до един от дъщерните процеси. Дъщерният процес обслужва заявката, като й връща отговор. Това поведение има своето смислено основание- сигурността.

Администраторът стартира родителския процес на Web сървъра при системите, използващи GNU/Linux. Поради факта, че администраторът на системата има пълен достъп до нея, то родителският процес не обработва директно заявките, изпратени на клиента. Да разгледаме следния случай, при който определен клиент изпраща злонамерена заявка и родителският процес я обработва. Управляваният като администратор родителски процес ще притежава необходимите права за извършването на каквато и да е операция на компютъра, което ще направи системата уязвима. Ако обаче заявката се прехвърли на процес с ограничени права в рамките на система, причиняването на време става невъзможно. Причината е, че дъщерните процеси се управляват като потребител с ограничени привилегии.

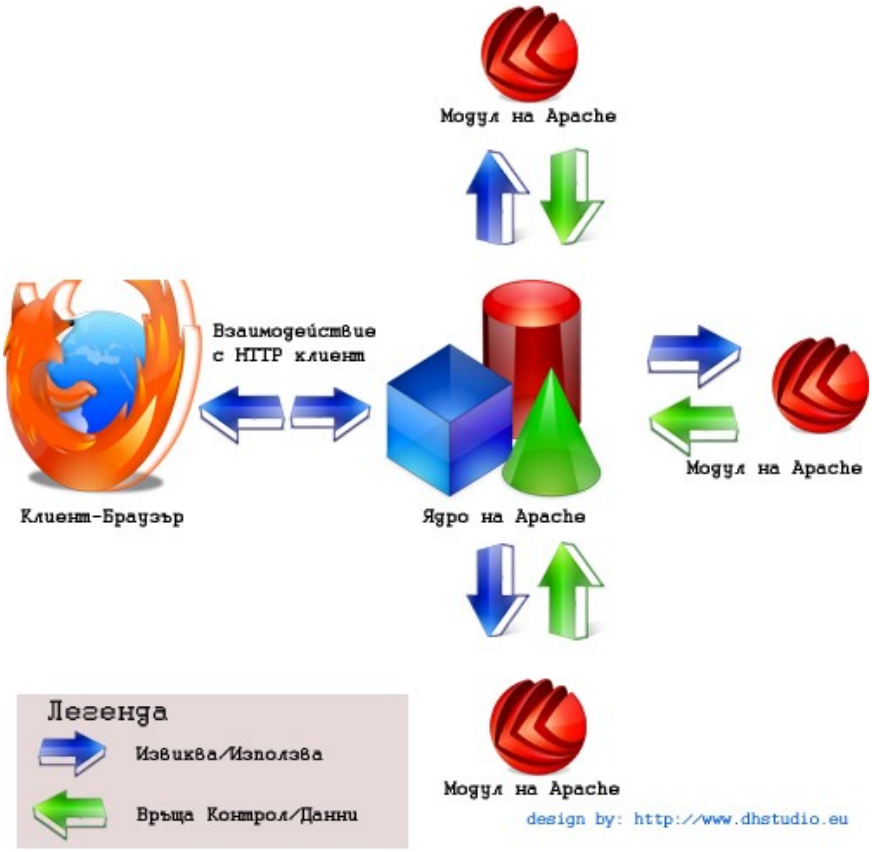
Друга важна част от архитектурата на Apache е процесът на обработка на заявките. Процесът може да се раздели на следните последователни етапи:

1. Проверка на единния идентификатор на ресурса (Uniform Resource Identifier- URI). На този етап Web сървърът Apache първо анализира изпратената от клиента заявка и определя заявената от клиента информация. След това Web сървърът Apache открива мястото на съхранение на заявената информация.
2. Проверка на данните за идентичността (authentication ID). Този етап включва потвърждаването на поверителните идентификационни данни на потребителя. С други думи, през този етап Web сървърът Apache проверява самоличността на потребителя.
3. Проверка на разрешението за достъп. На този етап се проверява дали потребителят е упълномощен да разглежда или да използва заявените ресурси.
4. Проверка на другите видове достъп. На този етап Web сървърът използва други механизми, за да провери разрешенията за достъп на потребителя.
5. Определяне на MIME типа. След като провери разрешенията за достъп на потребителя, Apache определя MIME типа на заявения ресурс.
6. Изпращане на отговор до клиента. Накрая Apache изпраща отговор на клиента. Това действие от страна на сървъра зависи от зададения метод при заявката.
7. Регистриране на заявката. Следва вписване на заявката в журнала за бъдещи справки.

Apache управлява тези етапи с използването на *модули*.Модулите са неразделна част от Apache и му придават така наречения модулен дизайн.

На теория, архитектурата на Apache се състои от две основни части: *ядро* и *модули*. Тези части на Apache са в основата на модулния му дизайн. Ядрото използва модули, когато Apache получи заявка. В

замяна модулите осигуряват ядрото с контрол и данни. Фигура. 1.3 представя взаимодействието на ядрото и модулите.



Фигура 1.11 Взаимодействие на ядрото и модулите с HTTP клиент

След като разгледахме толкова голяма част за архитектурата на Apache е редно да разгледаме и ядрото на Apache. Както името подсказва, ядрото е сърцето на Apache и включва основните му функции. Освен това ядрото отговаря за изпълнение на помощните функции. Една от помощните функции определя броя на ресурсите, които се разпределят за всяка заявка. Ядрото е съставено от компоненти, които имат следните цели:

- **Разпределение на ресурси.** Компонентът *carol.c* е отговорен за разпределението на ресурсите. След подходящото разпределение на ресурсите, този компонент наблюдава разпределените ресурси.
- **Функции на ядрото.** Компонентът http_core.c помага на Apache за изпълнението на минимума от функции. Това означава той да има достатъчно функции, за да обслужва документи (т.е. той помага на Apache да изпрати заявените документи), макар и не по най- полезния начин.
- **Управление на родителския процес.** Компонентът http_main.c управлява стартирането на Apache. Освен това той управлява поведението на сървъра. Основните задачи на този компонент са

да накара сървъра да изчака или приеме връзки. Това става с помощта на сървърният цикъл, който е част от този компонент. Накрая този компонент отговаря за управлението на изчакванията на сървъра.

- **Управление на обработчика на протокола.** Компонентът protocol.c отговаря за трансфера на данни по време на взаимодействието между клиента и Apache. Този компонент съдържа процедури, които позволяват осъществяването на директна комуникация с клиента.
- **Обработване и възлагане на ресурси.** Компонентът http_request.c управлява начина на обработване на получените заявки. Освен това той предава контрола до съответните модули.
- **Четене и управление на конфигурацията.** Компонентът http_config.c се използва главно за обработка на файловете за конфигурацията и предприемане на подходящите действие в съответствие с получените резултати.

След като бе разгледано ядрото на Apache, е редно да се разгледат и модулите към Apache. Модулът е обект, който може да се използва за прилагането, разширяването или промяната на функции на Web сървъра Apache. Web сървърите с модулен дизайн нямат ограничения за възможностите, които можете да добавите от тях. Освен това програмистите могат да създадат специално модули за посрещането на строго специфичните нужди на клиента.

Извод: Web сървърите представляват хранилища на файловете на Web сайтовете в Интернет. Те биват няколко вида, а именно: свободни и лицензионни. В лицензионните най- популярния е Web сървъра разработен от Microsoft IIS, а от свободните най- популярния е Apache.

Apache е HTTP Web сървър, разработен от група доброволци. Основата на Web сървъра Apache е HTTP сървър за публични домейни, разработен от Роб МакКул в Националния център за свръх изчислителни приложения (National Center for Supercomputing Applications) към Университета на Илинойс, Урбана Шемпейн.

Освен факта, че Web сървър Apache е най- използвания Web сървър в света е факт, че Apache прилага много висока сигурност и производителност.

ГЛАВА 2. ИНСТАЛИРАНЕ И КОНФИГУРИРАНЕ НА APACHE

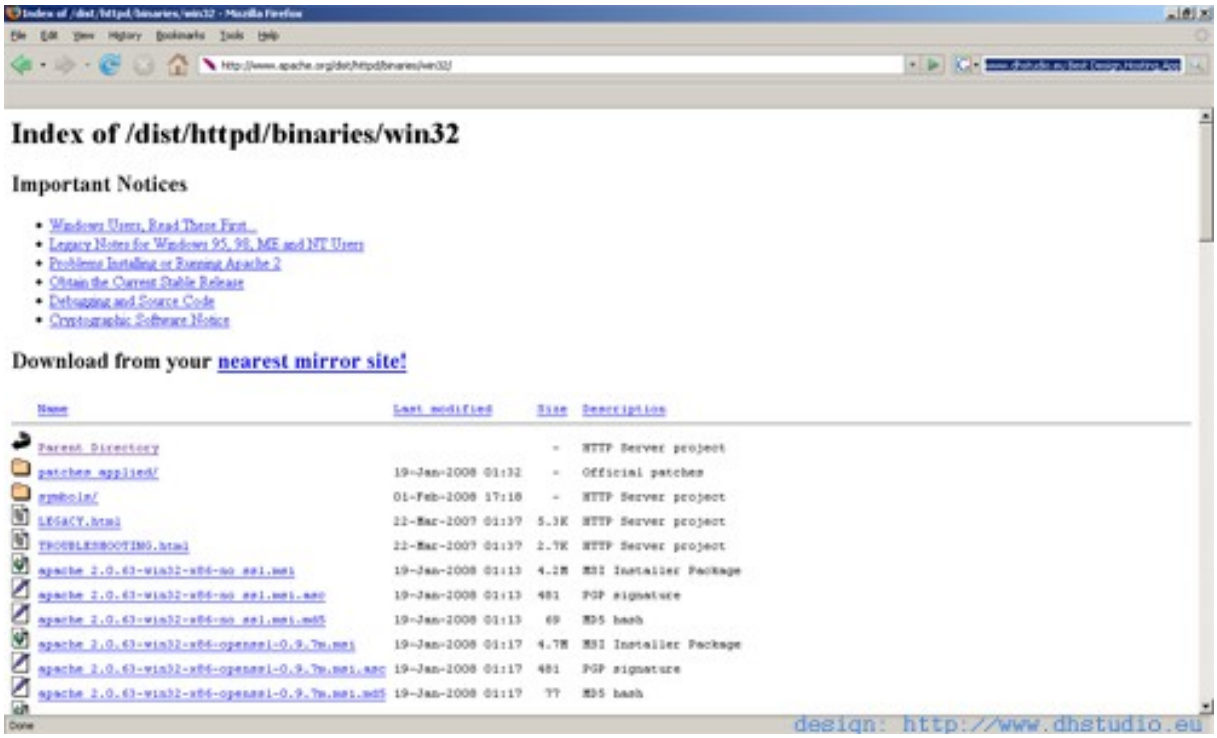
2.1. Инсталиране на Apache

Web сървърът Apache е част от повечето дистрибуции на GNU/Linux като Debian,Red Hat, Fedora, Ubuntu и други. Apache е мултиплатформен Web сървър т.е. има възможност да бъде инсталиран и на MS Windows. Дефиниране еднозначно описание на инсталиране на Apache е невъзможно понеже има възможност да бъде инсталиран от двоичен пакет или от програмен код(source code). Инсталирането на Apache чрез използването на двоичен код е най- срещаният и лесен метод. Двоичните кодове са предварително компилирани и могат да бъдат инсталирани, без да е необходимо компилиране на програмния код. Двоичният код за операционната система GNU/Linux, разпространявана от Red Hat е наличен под формата на *RPM(RPM Package manager)*, а за Debian, Ubuntu е под формата на *DEB*. Предварително инсталираните опции не позволяват на администраторите на Web сървъри да адаптират инсталацията според своите специфични изисквания от ниско ниво. RPM и DEB инсталацията представлява стандартна инсталация, т.е. инсталация по подразбиране, при която се инсталират стандартните пакети за Apache.На платформа MS Windows Apache също може да бъде инсталиран от двоичен код.

2.1.1. Инсталиране на Apache на платформа MS Windows

Преди инсталирането на Apache първо трябва да бъде наличен на конкретния сървър. Инсталационният файл е наличен на официалната страница на Apache. Инсталационният файл е с разширение *.msi* (Microsoft System Installer). След като изтеглите необходимите файлове, просто трябва да щракнете двукратно на инсталационния файл и да следвате инструкциите на екрана, за да извършите инсталацията. Програмистите, разработили Apache, се опитват да подобрят поддържането на Apache и от други операционни системи като Windows 95, Windows 98, Windows ME. Въпреки, че може да инсталирате Apache и на тези системи, поддържането на Apache от тези системи все още не се смята за достатъчно качествено.

Двоичните дистрибуции на Apache се намират в следната Web страница: http://www.apache.org/dist/httpd/binaries/win32/ . На фигура 2.1.1.1 е показана Web страницата на официалния Web сайт от където може да бъде достъпен Apache.

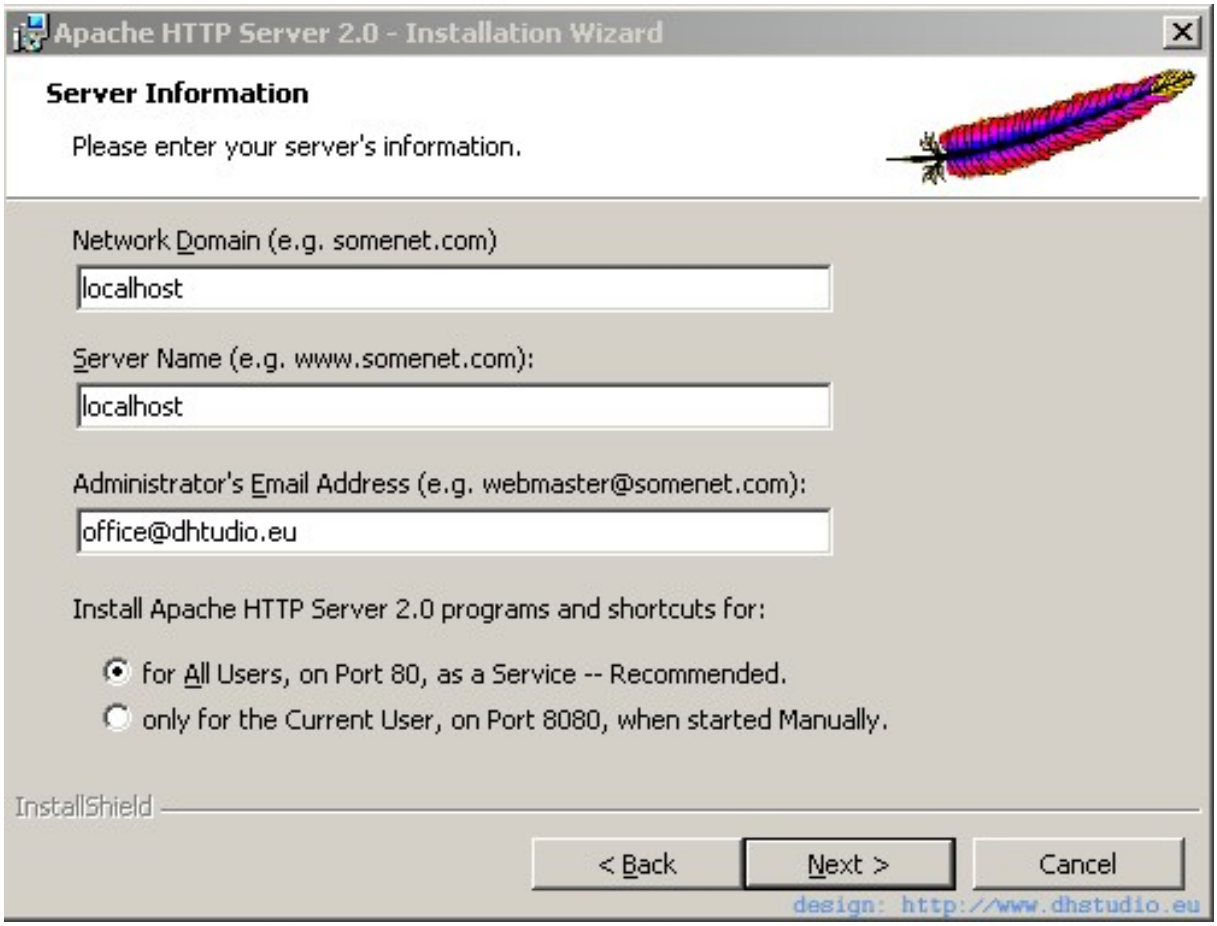


Фигура 2.1.1.1 Web страницата, показваща двоичните файлове на Apache MS Windows

След като разгледахме основните понятия в инсталиране на Apache под MS Windows нека го разгледаме по- подробно. Ще наблюдаване инсталация на Apache под MS Windows XP:

1) Кликнете на свалените файл на Apache, ще се появи екран.
2) Появява се първи екран за начало на инсталация. Потвърждение с бутон [Next]
3) Извеждат се общите условия за ползване на Apache. Потвърждение и след това натиснете бутона [Next].
4) Следващият екран е описание на Apache, там отново потвърдете [Next]
5) На този екран – Фигура 2.1.1.2 е необходимо да се въведат основни настройки :

Конфигурационен екран от инсталация на Apache под MS Windows XP



Фиг. 2.1.1.2

6) Тук избираме опцията Complete и отново се потвърждава с [Next]
7) Избираме директория където да инсталираме Apache. Изберете лесно достъпна директория(Пример: C:\) .След като натиснете [Next], сървъра ще бъде инсталиран в папка Apache, там където сте задали вие.(при мен в C:\ е направена директория Apache.)
8) На следващия екран натиснете бутона Install и инсталацията ще започне.
9) Инсталацията на Apache приключва с натискане на бутон [Finish] .

2.1.2. Инсталиране на Apache на платформа GNU/Linux

Apache може да бъде инсталиран на GNU/Linux чрез използването на RPM, DEB пакети или чрез компилиране от програмен код(source code)

2.1.2.1. Инсталиране на Apache чрез RPM пакет

За инсталиране на apache от RPM пакет е необходимо да се използва командата *rpm*. Командата *rpm* има множество възможни опции, но

повечето от тях са предназначени за разработчици създаващи пакети. За мрежов администратор командата *rpm* може да бъде редуцирана до три използването на три основни аргументи:

- rpm –i Apache_XX.rpm . Аргумента –I се използва за инсталиране на пакет
- rpm –e Apache_XX.rpm . Аргумента –e се използва за премахване на пакет.
- rpm –q Apache_XX.rpm . Аргумента –q се използва за извеждане на списък на софтуерните пакети, които вече са инсталирани на компютъра. Има възможност също да се използва и комбинацията между двата аргумента –qa за да изведе списък на всички инсталирани пакети.

След изясняване на основните аргументи за работа с командата *rpm*, нека сега инсталиране Apache пакета с командата:

rpm –I httpd-2.0.59-1.i386.rpm

Web сървърът Apache е инсталиран успешно от RPM пакет. Този тип инсталация е възможен за GNU/Linux дистрибуции: RedHat, Fedora и други RPM базирани дистрибуции.

2.1.2.2. Инсталиране на Apache чрез DEB пакет

Двоичните пакети с разширение *.deb* се използват от Debian, Ubuntu и други дистрибуции. Пакетите *.deb* могат да се инсталират чрез командата dpkg-Пакетен инсталатор за Debian.
Пример за инсталиране на пакет, чрез командата dpkg:

dpkg –i apache_2.0.49.deb

Друг метод за инсталиране на *.deb* пакет е чрез приложението *apt* или по-точно apt-get. Пример за използване на apt-get:
apt-get install apache

2.1.2.3. Инсталиране на Apache от програмен код

Компилирането на Apache от програмен код не става толкова лесно, колкото инсталирането чрез RPM или DEB пакети. Преди да се компилира Apache е необходимо да се отчетат определени пред инсталационни изисквания. Изискванията са различни за различните операционни системи. Ще наблегнем само на пред инсталационните изисквания на

GNU/Linux, тъй като по- нататък в този раздел се разглеждат стъпките при компилирането на Apache на GNU/Linux

Преди да стартирате инсталацията на Apache на компютър с GNU/Linux е необходимо да се проверят следните изисквания:

- Минималното изискване за временно пространство на диска е 12 MB. След инсталацията Apache заема приблизително 3MB памет. Въпреки изразходваната памет може да варира в зависимост от инсталираните модули.
- Преди да започнете инсталацията, трябва да инсталирате ANSI-C компилатор. Препоръчително е да се използва GNU C (gcc)
- Друга задължителна предпоставка е наличието на интерпретатора Perl 5. Apache използва определени поддържащи скриптове, като например apxs и dbmmanage, които са написани на Perl. Apache може да бъде компилиран и без да е инсталиран интерпретаторът Perl 5. Въпреки това , след инсталирането няма да можете да използвате тези скриптове. Поддръжката на общи динамични обекти (DSO) е въпрос на избор. DSO механизмът позволява зареждането на модули по време на работа.

2.1.2.3.1. Изтегляне на програмен код

След като се проверят всички пред инсталационни изисквания е необходимо да бъде изтеглен програмния код на Apache. Той е наличен на официалната Web страница на Apache. Файлът, който е под формата tar за GNU/Linux и е компресиран с помощта на GNU Zip, е наличен с разширения .tar.go, .tags или .be2. Програмния код на Apache може да бъде изтеглен от официалния сайт на Apache:

http://www.apache.org/dist/httpd/

2.1.2.3.2. Разпакетиране на програмния код на Apache

След като изтеглите програмния код, трябва да го разпакетирате. При системите работещи на GNU/Linux, програмният код обикновено се съхранява в директорията /usr/local/src. За предпочитане е да се разпакетира програмния код в тази директория. Раз пакетирането на програмния код се извършва чрез следната команда:

tar zxvf httpd-2.0.61.tar.gz

След изпълнението на тази команда, всички компресирани файлове се разкомпресират.

2.1.2.3.3. Добавяне на потребител и група по подразбиране

По подразбиране, при GNU/Linux сървърът Apache се управлява от потребител с име www-data. Можете обаче да настроите така Apache, че името на потребителя да е различно от www-data. Нека предположим че променяме стандартния потребител www-data на web. За да се извърши тази промяна първо трябва да добавим потребител и група web чрез следните команди:

```
# Тази команда ще добави група web
groupadd web
# Тази команда ще добави потребител web в група web
useradd –g web web
```

2.1.2.3.4. Използване на скрипт за конфигуриране configure и команда make

Преди да се започне компилиране на Apache трябва да се познава configure. Този файл е разположен в директорията от най- високо равнище на програмния код на Apache. Този файл не се използва за компилирането на Apache. Въпреки това той е важен, тъй като определя капацитета на системата и открива необходимите поддържащи файлове. Скриптът configure е изключително полезен по следните причини:

- Той не позволява инсталирането на Apache при наличието на проблем.
- Той дава инструкциите за разрешаването на проблемите, ако такива съществуват.
- Той определя най- добрата комбинация от опции за системата, на която искате да инсталирате Apache. Комбинацията от опции може да се различава за различните системи.
- Той използва програмата make за създаването на файлове, които са необходими за компилирането и инсталирането на Apache.

При използването на скрипта configure могат да се зададат няколко опции. Ако изберете стандартната инсталация на Apache, няма нужда да задавате допълнителни опции. Web администраторите обаче предпочитат да приспособяват инсталацията, като определят всички възможни опции. Така те инсталират Apache според собствените си изисквания. Опциите могат да бъдат зададени, за да разширят функциите на Apache или дори да отменят настройките по подразбиране на Apache. Разгледайте следната команда, която конфигурира Apache:

./configure –with-layout=RedHat

Това е основният начин за компилиране на Apache. Аргументът –with-layout=RedHat може да се използва, за да посочи, че файловете трябва да се инсталират на местата, които Red Hat Linux използва по подразбиране. Тази опция е идеално за системните администратори, които искат файловете да бъдат разположение на обичайните местоположения. Тази опция е полезна и за потребители, които не знаят какви опции трябва да се зададат, когато инсталират Apache.

Нека сега разгледаме configure с повече от един аргумент, а именно:

./configure –prefix=/usr/local/apache \
--enable-module=most \
--disable-module=auth_dbm \
--enable-shared=max

Аргументът *–prefix=/usr/local/apache* се използва за указването на директорията от файловата система, в която Apache да бъде инсталиран. В този случай е указана следната директория /usr/local/apache. Това е стандартно зададената директория, в която ще бъде инсталиран Apache. Въпреки това в този аргумент можете да зададете и различна директория. Има няколко причини, може да бъде необходимо да се инсталира Apache в директория, различна от стандартно зададената. Една от тях е да запазите стандартно зададените местоположения, използвани от специфична дистрибуции я на GNU/Linux. Освен това, някои хора инсталират множество версии на Apache с тестови цели. За тази цел те инсталират различните версии на Apache на различни места.

Аргументът *–enable-module=most* указва активирането на всички модули, които са включени в стандартната дистрибуция на Apache и които се поддържат от всички платформи. Включването на този аргумент обаче изключва някои модули, които не се поддържат от всички платформи. Препоръчително е при използването на аргумента –enable-module=most изрично да зададете допълнителните модули, които желаете да включите в инсталацията.

Аргументът *–enable-shared=max* се използва за да активира използването на общи динамични обекти по време на компилирането на модулите. Всички модули, с изключение на http_core module и mod_so , се компилират с помощта на като динамични обекти. Тези модули изрично трябва да се свържат статично с Apache. Това е важно, тъй като функционирането на модула http_core осигурява основните директиви за управлението на сървъра Apache. Модулът mod_so се използва за разрешаване на използването на динамични обекти от сървъри. Поддържането на динамични обекти беше вградено в Apache с излизането на версия 1.3. По този начин е възможно активирането и деактивирането на модулите по време на работа, без да е необходимо ново

прекомпилиране на ядрото на Apache. Това е изключително полезно, тъй като деактивирането на модулите намалява размера на изпълнимия файл на Apache. Така можете лесно да стартирате множеството отделни инсталации на Apache в определен момент, като използвате ограничено количество памет.

След изброяване на често използваните аргументи е редно да бъдат изброени в Приложение 1.

След като вече са изброени всички аргументи с кратко описание ще се разгледа една примерна конфигурация за компилиране на Apache:

./configure –prefix=/usr/local/apache \
--server-uid=web \
--server-gid=web \
--htdocsdir=/var/www \
--cgidir=/usr/lib/cgi-bin/ \
--enable-module=most \
--enable-shared=max

Нека се разгледат допълнителни аргументи.

Аргумента –server-uid=web (Web е стойност на този аргумент) показва, че сървърът Apache ще работи с идентификация на потребителя web.

Опцията –server-gid=web показва, че сървърът Apache ще работи с идентификация на групата web.

Опцията –htdocsdir показва, че стандартните файлове за Web сайта ще бъдат разположени в директорията /var/www/. Опцията

Опцията –cgidir=/usr/lib/cgi-bin/ показва, че CGI файловете по подразбиране ще бъдат разположени в директорията /usr/lib/cgi-bin/

Преди изпълнението на ./configure е необходимо да се познава и файла config.layout. Този файл съдържа определена жизнено важна информация, която помага в процеса на компилирането.

Файлът config.layout е важен конфигурационен файл. Той съдържа информация, която Apache използва, за да открие крайните инсталационни пътища за файловете по време на компилацията. Процесът на компилиране включва инсталирането на няколко файла на различни местоположения. Информацията за тези местоположения също се съдържа в този файл под формата на *именувани образци*. Именуваните образци представляват спецификации за пътищата до директориите, които Apache използва по време на компилацията, за да копира необходимите файлове. Спецификациите на директориите се наричат именувани образци, тъй като всеки запис се идентифицира със системно име. По време на инсталацията файловете се копират на определени стандартно зададени

местоположения. Тези местоположения могат да се различават за различните системи. Когато стартирате скрипта configure, Apache опитва да определи операционната система, която се използва. При успешно определяне на операционната среда, Apache използва съответните образци във файла config.layout, за да определи правилната информация за пътищата. При условие, че Apache не успее да определи операционната система,а се използва стандартно зададеният образец. Това е стандартен образец на Apache за пътищата. Стандартно зададените пътища от този образец са следните:

```
# Classical Apache path layout.
<Layout Apache>
   prefix:       /usr/local/apache2
   exec_prefix:  ${prefix}
   bindir:       ${exec_prefix}/bin
   sbindir:      ${exec_prefix}/bin
   libdir:       ${exec_prefix}/lib
   libexecdir:   ${exec_prefix}/modules
   mandir:       ${prefix}/man
   sysconfdir:   ${prefix}/conf
   datadir:      ${prefix}
   installbuilddir: ${datadir}/build
   errordir:     ${datadir}/error
   iconsdir:     ${datadir}/icons
   htdocsdir:    ${datadir}/htdocs
   manualdir:    ${datadir}/manual
   cgidir:       ${datadir}/cgi-bin
   includedir:   ${prefix}/include
   localstatedir: ${prefix}
   runtimedir:   ${localstatedir}/logs
   logfiledir:   ${localstatedir}/logs
   proxycachedir: ${localstatedir}/proxy
</Layout>
```

От съдържанието на файла config.layout се вижда, че всички записи в този файл са пътища на директории. Някои имена на пътища са определени въз основа на определените в преди тях пътища. Забележете, че всички изброени пътища са производни на определената като prefix директория. Следователно можете да използвате опцията –prefix, за да промените стандартно-зададените пътища при инсталирането на Apache. Винаги преглеждайте образеца, който скриптът configure ще използва при компилирането на Apache.

След като се изясни голяма част от метода за компилиране нека сега се премине към компилирането на Apache чрез командата: *make.* Помощният

инструмент *make* е стартова площадка за етапа на реалната инсталация на Apache. Преди да се изпълни make първо се изпълнява ./configure (със съответните аргументи). Когато се изпълни *make* всъщност се компилира Apache. След като Apache е компилиране, то трябва да бъде инсталиран с командата *make install*. Тази команда всъщност прехвърля поддържащите файлове и двоичните файлове в подходящите директории на файловата система. Тези директории са стандартно зададените местоположения, освен ако са зададени други директории. Повече файлове се копират в главната директория на Apache, която сте задали с аргумента *--prefix*

След инсталиране на Apache ще се разгледа важна команда за управлението на Apache, а именно: *apache2ctl.* По- долу ще бъдат изброени някои от аргументите на apache2ctl:

start – Стартиране на Apache
stop – Спиране на Apache
restart- Рестартиране на Apache, а ако е стартиран изпраща SIGHUP или го стартира ако не е стартиран
fullstatus – Извежда пълния статус на Apache (изисква lynx и зареден модул mod_status)
status – Извежда малък статус на Apache (изисква lynx и зареден модул mod_status)
graceful – Рестартиране с изпращане на сигнал SIGUSR1 или ако не е стартиран го стартира.
configtest – Тества конфигурационния синтаксис

След компилиране и инсталиране на Apache ще се тества конфигурационния файл, чрез apach2ctl:

apache2ctl configtest
Syntax OK

Получено съобщение означава, че конфигурационния файл на Apache не съдържа грешки, която означава, че Apache може да бъде стартиран:

apache2ctl start

След инсталиране и стартиране на Apache може да се тества по няколко метода. Първия метод е в терминал, чрез използване на командата *lynx* или по- конкретно:

lynx http://localhost

Другия метод е чрез отдалечен компютър и браузър да се напише като URL адрес IP адреса на машината на която инсталирахме Apache. Пример: http://82.137.123.111

Трети вариант е, чрез използване на командите *telnet* или *nc:*
nc 82.137.123.111 80
GET index.html

telnet 82.137.123.111 80
GET index.htm1

2.2. Конфигуриране на Apache

За конфигуриране на Apache е необходимо да се редактира файла apache2.conf , за да се зададат email адрес на Web администратор в *ServerAdmin* и хост името на сървъра в *ServerName* . На системата GNU/Debian файлът apache2.conf се намира в директорията /etc/apache2/apache2.conf . При друга операционна система конфигурационният файл може да се съхранява в друга директория. Бърз метод чрез които може да се открие местоположението на файла apache2.conf е командата:

find / -name apache2.conf

Тази команда *find* претърсва всяка директория от коренната (/) надолу за файл с име apache2.conf и извежда резултата от търсенето.

След като се открие файла apache2.conf се използва текстов редактор за да се поставят валидни стойности за *ServerAdmin* и *ServerName* в конфигурацията. По- горните настройки на директивите е редно да изглеждат по следния начин:

ServerAdmin webmaster@dhstudio.eu
ServerName www.dhstudio.eu

След конфигуриране на някои от основните директиви е добре да се разгледа директивата DocumentRoot указващата директорията от където да се взимат Web файловете. Много често DocumentRoot директорията е /var/ www/. Друг вариант за узнаване Web директорията е, чрез изпълнение на следната команда:

grep ^DocumentRoot /etc/apache2/sites-enabled/000-default| awk '{print $2}'

Претърсва се този файл понеже в директория /etc/apache2/sites-enabled се намират конфигурационните файлове на всички Web сайтове като по подразбиране е *000-default*. По подразбиране Apache търси файл с име index.htm, index.html или index.php и го използва като начална страница, ако не е изискана конкретна страница.

2.2.1. Конфигурационен файл apache2.conf и включващите се в него файлове.

Файлът apache2.conf представлява ASCII текстов файл. Коментарите започват с "#" (диез), а файлът е добре документиран с коментари. Повечето от командите във файловете се записват във формата на директива, следвана от стойността, която се задава от директивата.

Например: Listen 443

Тази директива задава на Listen стойност 443.

Освен основните директиви, файлът apache2.conf включва контейнери, които ограничават обхвата на съдържащите се в тях директиви. Например, за да ограничите приложението на определени директиви в конкретна директория, можете да създадете контейнер *Directory* за тези директиви. Има пет различни типа контейнери във файла apache2.conf:

<Directory път> Директивата *Directory* създава контейнер за директиви, които се прилагат за директорията, идентифицирана от път. Всички конфигурационни директиви, които се намират след директивата *Directory* и преди следващата конструкция *</Directory>*, се прилагат само за зададената директория.

<Files име-на-файл> Директивата *Files* създава контейнер за директиви, които се прилагат за файла, идентифициран от име-на-файл. Всички конфигурационни директиви, които се намират след директивата *Files* и преди следващата конструкция *</Files>*, се прилагат само за зададения файл. Име-на-файл може да сочи към повече от един файл, ако съдържа знаковете за заместване * и ?. Освен това, ако директивата *Files* е следвана от незадължителния знак ~ (тила), полето име-на-файл се интерпретира като регулярен израз.

<Location документ> Директивата *Location* създава контейнер за директиви, които се прилагат за конкретен документ. Всички конфигурационни директиви, които се срещат след директивата *Location* и

преди следващата конструкция *</Location>*, се прилагат само за конкретния документ.

<IfDefine аргумент> Директивата *IfDefine*
създава контейнер за директиви, които се прилагат в конфигурацията само ако зададеният аргумент съществува на командния ред на apache. Например редът *<IfDefine HAVE_SSL>* маркира началото на контейнер от директиви, които се използват само ако низът *–DHAVE_SSL* присъства на командния ред за apache. Контейнерът *IfDefine* завършва със следващата конструкция *<IfDefine>*

<IfModule модул> Директивата *IfModule* създава контейнер за директиви, които се прилагат в конфигурацията само ако зададеният модул е зареден. Например директивите, които се намират между *<IfModule mod_userdir.c>* и *</IfModule>*, се използват само ако е зареден модулът *mod_userdir*.

Директориите и файловете могат лесно да бъдат разбрани. От друга страна, документите са специфични за Web сървъра. Екранът информация, който се появява в отговор на една Web заявка, е документ. Той може да бъде изграден от много файлове от различни директории. Контейнерът Location осигурява лесен начин за указване на сложен документ като едно цяло.

Файлът apache2.conf е дълъг 668 реда. По- голяма част от неговото съдържание представлява информация от разработчиците на Apache за неговото конфигуриране. Файлът е пълен с коментари, които обясняват предназначението на конфигурационните директиви, а допълнителната информация за директивите е достъпна в онлайн документацията на адрес *apache.org* . Конфигурационният файл организира директивите по обхват: директиви за глобалното обкръжение, основни сървърни директиви и директиви за виртуални хостове. Този начин на организация е идеален за apache, когато той обработва файла, но не толкова подходящ за разчитане от човек. Поради тази причина ще се съберат директивите в свързани групи, за да се направят отделните директиви по- разбираеми, защото след като се разберат отделните директиви, ще се разбере и цялата конфигурация.

2.2.1.1 Зареждане на динамични споделени обекти

Двете директиви, които се срещат най- често във файла apache2.conf са LoadModule и AddModule.Тези две директиви участват в доста голяма част от конфигурационния файл apache2.conf .Всичките тези редове

конфигурират Dynamic Shared Object (DSO) модулите, използвани от Web сървъра Apache.

Apache е съставен от много софтуерни модули. Подобно на модулите на ядрото, DSO модулите могат да бъдат компилирани в Apache или да бъдат зареждани по време на изпълнение. Пример за извеждане на всички заредени модули:

```
dhstudio# apache2 -l
Compiled in modules:
  http_core.c
  mod_so.c
```

Някои системи могат да имат много модули, компилирани в демона Apache. Нека разгледаме компилираните модули:
http_core . Това е основният модул. Той винаги е статично свързан в ядрото на Apache. Осигурява основните функции, с които трябва да разполага всеки Web сървър Apache. Този модул е задължителен. Всички други модули са по избор.
mod_so.c . Този модул осигурява поддръжка по време на изпълнение на Dynamic Shared Object модули. Той е необходим, ако възнамерявате динамично да свързвате други модули по време на изпълнение. Ако модулите се зареждат чрез файла apache2.conf, този модул трябва да бъде инсталиран в Apache, за да осигури поддръжка на тези модули. По тази причина той често се свързва статично към ядрото на Apache.

Във файла apache2.conf се използват и много други динамично заредени модули. Използват се две директиви за разрешаване на динамично зареждане модули. Първо, всеки модул се идентифицира от директива LoadModule. Например за да се зареди модула за управление на файлов- дневник, в който се отчита типовете браузъри на потребителите, трябва да се въведе следния ред във файла apache2.conf:

```
LoadModule agent_log_module /modules/mod_log_agent.so
```

Директивата LoadModule е следвана от името на модула и пътя до самия модул. В допълнение към директивата LoadModule, конфигурацията на някои дистрибуции идентифицира всеки модул с директивата AddModule. Тази директива добавя името на модула към списъка от модули, които се зареждат по време на изпълнение. Списъкът от модули включва всички незадължителни модули- тези, които са компилирани в сървъра, и тези, които се зареждат динамично- с изключение на основния модул, който е задължителен. Например за добавяне на модул agent_log_module към списък от модули, се добавя следния ред във файла apache2.conf:

AddModule mod_log_agent.c

В директивата *AddModule* се задава името на сорс файла на зареждания модул. Това не е името на модула от реда *LoadModule*, а името на сорс файла, от който е създаден обектния модул. То е същото като името на обектни файл, с изключение на разширението. На реда *LoadModule* името на обектния файл е mod_log_agent.so. Тук името на сорс файла е *mod_log_agent.c*. Изпълнимите модули, наречени *споделени обекти (shared objects)*, използват разширението *.so*, а модулите в списъка за добавяне, написани на езика C, използват разширението *.c*

Следващата таблица № 1 р съдържа списък с описание на често използваните модули:

Описание на често използвани Apache модули Таблица № 1

Модул	**Предназначение**
mod_access	Задава контрол на достъпа на база хост и домейн
mod_actions	Асоциира CGI скриптове към MIME файлови типове
mod_alias	Указва директории с документи извън дървото на документи
mod_asis	Дефинира файлови типове, връщани без хедъри
mod_auth	Разрешава автентикацията на потребители
mod_auth_anon	Разрешава анонимно влизане.
mod_auth_db	Разрешава използване на автентикация чрез база данни
mod_autoindex	Разрешава автоматично генериране на индекси
mod_bandwidth	Задава ограничение за пропускателната способност на сървъра
mod_cgi	Разрешава изпълнението на CGI програми
mod_dav	Осигурява протоколните разширения WebDAV
mod_dir	Контролира форматирането на листингите на директории
mod_env	Позволява CGIи SSI да наследяват всички променливи на обкръжението на шела
mod_expires	Задава датата за Expires хедъра
mod_headers	Разрешава нестандартни хедър за отговор
mod_imap	Обработва файлове за карти-изображения (images maps)
mod_include	Обработва SSI файлове
mod_info	Разрешава използване на манипулатора за сървърна информация
mod_log_agent	Сочи към файла- дневник за агентите.
mod_log_config	Разрешава използване на нестандартни формати за дневниците

mod_log_referer	Сочи към referrer дневника, който записва информация за отдалечението сайтове, препращащи към вашия сайт
mod_mime	Осигурява поддръжка за MIME файлове
mod_negotiation	Разрешава договаряне на MIME съдържание
mod_perl	Осигурява поддръжка за MIME файлове
mod_php	Осигурява поддръжка за езика PHP
mod_php3	Допълнителна поддръжка за PHP
mod_php4	Допълнителна поддръжка за PHP
mod_put	Осигурява поддръжка за трансфер на файлове от клиента към сървъра с помощта на командите PUT и DELETE
mod_python	Осигурява поддръжка за езика Python
mod_rewrite	Разрешава асоцииране URI към име на файл
mod_roaming	Осигурява поддръжка на Netscape Roaming Access
mod_setenvif	Задава променливи на обкръжението от информацията от клиента.
mod_so	Осигурява поддръжка по време на изпълнение на споделени обекти (DSO)
mod_ssl	Осигурява поддръжка за Secure Sockets Layer
mod_status	Осигурява Web базиран достъп до отчет със сървърна информация.
mod_throttle	Ограничава максималната пропускателна способност на отделни потребители.
mod_userdir	Дефинира къде потребителите могат да създават публични Web страници
mod_vhost_alias	Осигурява поддръжка за виртуални хостове, базирани на имена.

Фигура 2.1

2.2.1.2. Основни директиви на Apache

Няколко директиви дефинират основната информация за самия сървър.

ServerAdmin дефинира e-mail адреса на администратора на Web сървъра. В подразбиращата се конфигурация е зададен като r*oot@localhost* понеже акаунтът *root* винаги съществува и винаги има име на хост *localhost*. Пример за коректна стойност на директивата *ServerAdmin*:

ServerAdmin webmaster@dhstudio.eu

В този пример се използва класически e-mail адрес на Web администратора webmaster@dhstudio.eu като стойност за *ServerAdmin*.

Директивата *ServerName* дефинира името на хост, което се връща на клиентите, когато те четат данни от този сървър. При някои дистрибуции директивата *ServerName* е коментирана, т.е. няма стойност като в такъв случай на клиентите се изпраща "реалното" име на хост. По този начин, ако името, зададено на първия мрежов интерфейс, е amri.dhstudio.eu, то ще се изпраща на клиентите, когато стойността за Servername не е дефинира.

Директивата *UseCanonicalName* контролира дали да се използва стойността, дефинирана от *ServerName*. *UseCanonicalName* дефинира как apache ще обработва URL- адресите, които сочат към самите себе си. Когато тя е зададена на on, както е в конфигурацията на някои дистрибуции, за име на сървъра се използва стойността, дефинирана в директивата *ServerName*. Ако е зададена на off, се използва стойността, която идва от заявката на клиента. Ако текущия Web сайт използва множество имена на хостове, може *UseCanonicalName* да бъде сетнат на off, така че потребителят да вижда в отговора името, което очаква.

Опцията *ServerRoot* дефинира директорията, която съдържа сървърните файлове на apache. Тя е различна от *DocumentRoot*-директорията на файловете с информацията, които сървърът предоставя на клиентите. В Debian и на много други системи това е /etc/apache2/. Директорията.

Опцията *ServerType* дефинира как се стартира сървърът. Ако сървърът се стартира от стартов скрипт по време на първоначалното зареждане, опцията трябва да има стойност *standalone*. Ако сървърът се стартира при поискване от *inetd* или *xinetd*, на *ServerType* се задава *inetd*. През по- голямата част от времето Web сървърите биват търсени често, поради което е най- добре да се стартират по време на първоначалното зареждане. Но е възможно някой потребител да иска да създаде малък, рядко използван Web сайт на GNU/Linux система. В този случай вероятно ще се поиска сървърът да се стартира от *inetd* или *xinetd*.

Port дефинира номер на TCP порт, използван от сървъра. Стандартният номер е 80. В някои случаи частните Web сървъри работят на други номера на портове като 8080 и 8000 са популярните алтернативни портове. Ако се промени номера на порта от 80 на 8080, след това потребителите на конкретния Web сайт трябва да бъдат информирания за промяната като ако стария адрес е бил *http://www.dhstudio.eu* то новия ще бъде: http://www.dhstudio.eu:8080

Когато ServerType има стойност inetd, обикновено е желателно да зададете за Port стойност, различна от 80. Причината за това, че портовете под 1024 са "привилегировани" портове. Ако се използва порт 80, apache трябва да бъде стартиран от inetd с потребителски идентификатор root. Това е потенциален проблем на сигурността, защото един нарушител може да получи възможност да придобие root достъп посредством Web сайта. Когато ServerType е standalone, няма проблем да се използва порт 80,

защото първоначалния apache процес не осигурява услуги на клиентите директно. Вместо това , за осигуряване на услуги той стартира няколко други HTTP демона, които не се изпълняват с root привилегии.

2.2.1.3. Конфигурация на сървър с повече от един IP адрес.

Web сървър, който е свързан към повече от една физическа мрежа, се нарича multi-homed сървър. Такъв сървър има повече от един IP адрес. В този случай системата трябва да знае на кой адрес трябва да слуша за входящи заявки към сървъра. Две конфигурационни опции управляват това:
Bind Address – Указва на apache кой адрес трябва да използва за взаимодействия със сървъра. Подразбиращата се стойност е * , което означава, че сървърът трябва да отговаря на Web заявките, отправени към всеки от неговите валидни IP адреси. Ако на командния ред на Bind Address е зададен конкретен адрес, се обработват само заявките, отправени към този адрес.
Listen- Указва на apache кои допълнителни адреси и портове трябва да бъдат наблюдавани за заявки за Web услуга. Двойките адрес-порт се отделят с двоеточие. Например, за да наблюдавате порт 8080 на IP адрес 172.16.2.13, въведете 172.16.2.2.13:8080. Ако портът е въведен без адрес се използва адресът на сървъра. Ако директивата Listen не се използва, apache наблюдава само порта, дефиниран от директивата Port.

2.2.1.4. Дефиниране на местоположенията на информацията

Директивата DocumentRoot, дефинира директорията, съдържаща документите на Web сървъра. По причини за сигурност това не е същата директория, която съдържа конфигурационните файлове. Местоположението на конфигурационните файлове на сървъра се дефинира от директивата ServerRoot. Пример:

DocumentRoot "/var/www"
ServerRoot "/etc/apache2"

Директивите PidFile и ScoreBoardFile дефинират пътищата до файловете свързани със статусите на процесите. PidFile е файлът, в който apache съхранява своя идентификатор на процес, а ScoreBoardFIle е файлът, в който apache записва информация за статуса на процеса. Ако ScoreBoardFile не е дефиниран, Apache използва сегмент от споделената памет вместо файл, което подобрява производителността.

Директивата Alias и директивата ScriptAlias асоциират URL път към директорията на сървъра. Например:

Alias /icons/ "/usr/share/icons"
ScriptAlias /cgi-bin/ "/var/lib/cgi-bin/"

Първият ред асоциира URL пътя /icons/ към директорията /usr/share/icons. По този начин заявката www.dhstudio.eu/icons се асоциира към www.dhstudio.eu/usr/share/icons/. Директивата ScriptAlias функционира точно по същия начин, като директивата Alias, с изключение на това, че директорията, към която сочи, съдържа изпълними CGI приложения.

Директивата UserDir разрешава възможността отделните потребители да поддържат лични Web страници и задава директорията, която съдържа тези страници. Обикновено UserDir има стойност public_html. При тази подразбираща се настройка потребителите създават директория с име public_html в своите лични директории, където съхраняват своите лични Web страници. Когато например пристигне заявка за www.dhstudio.eu/~amri , тя се асоциира към www.dhstudio.eu/home/amri

Една алтернатива на това е да се дефинира пълен път , например /home/userpages, на командния ред на UserDir. След това администраторът създава директорията и разрешава на всеки потребител да съхранява лични страници в поддиректории на тази директория. Заявката за www.dhstudio.eu/~amri се асоциира към www.dhstudio.eu/home/userpages/amri/. Предимствата на този подход са, че той подобрява сигурността, като ви улеснява при наблюдението на съдържанието на страниците на потребителите. Освен това той позволява лесно да контролирате лицата, които могат да публикуват страници като при него публикуването не е разрешено за всички потребители.

Опцията DirectoryIndex дефинира името на файла, който се извлича, ако заявката на клиента не включва име на файл. Пример:
DirectoryIndex index.html index.htm index.shtml index.php index.php5 index.php4 index.php index.phtml index.cgi

При дефинираната стойност за DocumentRoot и тази стойност, ако сървърът получи заявка за http://www.dhstudio.eu/amri, той първо се опитва да открие файл с име /var/www/amri/index.html. Добре е да се обърне внимание на това, че DocumentRoot се добавя пред всяка заявка, а DirectoryIndex се добавя след всяка заявка, която не завършва с име на файл. Ако сървърът открие файл с това име, той го предоставя на клиента. Ако не открие файла, сървърът се опитва да открие index.htm, след това index.shtml и така поред до index.cgi. Ако нито един от файловете , дефинирани от то DirectoryIndex, не бъде намерен, apache изпраща на клиента листинг на директорията.

2.2.1.5. Създаване на индекс с нестандартно форматиране

Ако опцията FancyIndexing е зададена в директивата IndexOptions, apache създава директорийни листинг, които включват графики, връзки и други възможности. Следните опции дефинират графиките и възможностите, използвани в нестандартно форматираните листинги на директории:

IndexIgnore: Извежда списък на файловете, който не трябва да бъдат включени в листинга на директорията.Файлове могат да бъдат зададени по име, по част от името, по разширение или с помощта на стандартни знакове за заместване (wildcards).

HeaderName: Дефинира името на файл, съдържащ информация, която трябва да бъде визуализирана в горния край на листинга на директориите.

Readname: Дефинира името на файл, съдържащ информация, която трябва да бъде визуализирана в долния край на листинга на директориите.

AddIcon: Дефинира файла на икона, използвана за представяне на файл на базата на неговото разширение.

DefaultIcon Дефинира файл на икона, използвана за представяне на файл, на който не е зададена икона от някоя друга опция

AddIconByEncoding: Дефинира файла на икона, използвана за представяне на файл на базата на неговото MIME кодиране.

AddIconByType: Дефинира файла на икона, използвана за представяне на файл на базата на неговия MIME тип.

2.2.1.6. Дефиниране на типове файлове

MIME файловите типове и файлови разширения играят важна роля. Те помагат на сървъра да определи какво да направи с даден файл. Задаването на MIME опции е също важна част от apache2.conf. Тези опции са следните:

DefaultType: Дефинира MIME типа, който се използва когато сървърът не може да определи типа на даден файл. По подразбиране този тип е text/html. При тази настройка, когато даден файл няма файлово разширение, сървърът приема, че това е HTML файл.

AddEncoding: Асоциира тип MIME кодиране към файлово разширение. Пример:

AddEnconding: x-compress Z
AddEncoding: x-gzip gz tgz

Първата директива асоциира файловото разширение .Z към типа MIME кодиране x-compress. Вторият ред асоциира файловите разширения .gz и .tgz към типа MIME кодиране x-gzip.

AddLanguage : Асоциира MIME езиков тип към файлово разширение .

LanguagePriority: Задава езиковото кодиране в случай, че клиентът не задава предпочитание.
AddType : Асоциира MIME файлов тип към файлово разширение

AddHandler : Асоциира манипулатор на файл към файлово разширение. *Манипулаторът на файл (file handler*) представлява програма, която знае как да обработи файла. Просто примери за това са cgi-script , манипулаторът за CGI файлове и *server-parsed*, който управлява Server Side Includes (SSI).

2.2.1.7. Управление на дъщерните процеси

Според замисъла на оригиналния Web сървър на NCSA (National Center for Supercomputer Appliations), сървърът стартира нови процеси, за да обработва отделните заявки. Това предизвикваше силно натоварване на CPU, когато сървърът беше зает и влияеше негативно върху скоростта на отговорите му. Дори беше възможно цялата система да бъде претоварена от процеси на HTTP демона.

Apache използва друг подход. По време на първоначалното зареждане се стартират множество сървърни процеси. Работното натоварване се поделя между всички тези процеси. Ако всички действащи apache процеси станат заети, биват стартирани резервни процеси за поделяне на работата.

Пет опции в конфигурационния файл apache2.conf контролират начина на управление на дъщерните процеси на сървъра. Опциите, които контролират управлението на тези резервни процеси, са следните:

MinSpareServers: Дефинира минималния брой на незаетите сървърни процеси, които трябва да бъдат поддържани. В доста голяма брой от дистрибуциите този брой е 5. При тази настройка друг процес за поддържане на минималния брой бездействащи процеси се създава, когато

броят на бездействащите httpd процеси спадне до 5. Ако сървърът се бави при отговор поради периоди на висока активност е необходимо да бъде увеличена стойността на *MinSpareServers*

MaxSpareServers: Дефинира максималния брой на бездействащите сървърни процеси, които могат да бъдат поддържани. По време на пикова активност могат да бъдат създадени няколко httpd процеси, за да обработват една клиентка заявка. С намаляване на активността, процесите стават бездействащи.

StartServers: Дефинира броя на постоянните httpd процеси, стартирани по време на първоначалното зареждане. Ако броят им е 8, то ефектът от тази директива може да бъде видян в резултат от изпълнението на командата “ps”, където се показва, че се изпълняват девет httpd демона. Един от тях е родителският процес, който управлява множеството процеси, но не обслужва клиентски заявки. Другите осем са дъщерните процеси, които обработват клиентските заявки за данни.

MaxClients: Дефинира максималния брой на клиентите, които могат да бъдат обслужвани. Заявките над този максимален брой се подреждат в опашка, където изчакват освобождаването на сървъра. В някои дистрибуции тази опция има стойност 150. Подразбиращата се стойност, използвана когато *MaxClients* не е дефинирана е, стойността , зададена от *HARD_SERVER_LIMIT* при компилирането на Apache. Обикновено тази стойност е 256. MaxClients не допуска сървърът да заеме всички системни ресурси, когато получи прекалено голям брой заявки от клиенти.

MaxRequestsPerChild: Дефинира броя клиентски заявки, които един дъщерен процес може да обработи, преди да се наложи да бъде прекратен. MaxRequestsPerChild се използва, когато операционната система или библиотеките имат утечки в паметта, които причиняват проблеми за постоянно работещите процеси. Apache препоръчва настройката 10000, ако вашата система има проблеми с утечки в паметта. При условие, че на *MaxRequestsPerChild* се зададе стойност 0, което означава “неограничено” то един дъщерен процес може да продължи да обработва клиентски заявки през цялото време докато системата е включена и работи, освен ако не се знае със сигурност, че библиотеката, която е използвана за да се компилира Apache има утечка в паметта.

Директивата *User* и *Group* дефинират UID и GID идентификаторите, с които се изпълняват групата httpd процеси. Когато httpd се стартира по време на първоначалното зареждане, той се изпълнява като root процес, обвързва се към порт 80 и след това стартира група от дъщерни процеси, които осигуряват действителните Web услуги. На тези дъщерни процеси се

задават UID и GID идентификаторите, дефинирани във файла apache2.conf. UID и GID идентификаторите трябва да осигуряват най- малката възможна системна привилегия за Web сървъра. На повечето GNU/Linux системи това е потребителят www-data и групата www-data. Алтернатива на използването на www-data е да се създаде потребителски ID и групов ID точно за httpd. Предимството на създаването на специални GID и GID за Web сървър е, че може да се използва позволението за група, за да се осигури допълнителна защита.

2.2.1.8. Директиви за настройка на производителността

Директивата *KeepAlive* разрешава или забранява използването на постоянни конекции. Когато не се използват постоянни конекции, клиентът трябва да установява нова конекция към сървъра за всяка хипервръзка, която трябва да следва. Тъй като HTTP работи върху TCP, всяка HTTP заявка изисква установяването на TCP конекция. Това добавя допълнително време за всеки файл, който се извлича. При постоянната конекция сървърът изчаква , за да види дали клиентът има допълнителни заявки, преди да я затвори. Следователно клиентът не трябва да създава нова конекция, за да заяви нов документ. *KeepAliveTimeout* дефинира броя секунди, които сървърът поддържа конекцията отворена, изчаквайки да разбере дали клиентът има допълнителни заявки.

MaxKeepAliveRequests: Дефинира максималния брой заявки, които ще бъдат приемани по една keep-alive конекция, преди да бъде отправена заявка за нова TCP конекция. Подразбиращата се стойност на Apache е 100. Задаването на *MaxKeepAliveRequests* на стойност 0 позволява неограничен брой заявки . Малко потребители отправят заявки за трансфер на 100 документа, затова тази стойност на практика създава постоянна конекция, приложима за всички нормални случаи. Освен това, ако клиентът наистина отправя заявки за трансфер на повече от 100 документи, това може да е индикатор на проблем с клиентската система. В този момент вероятно е добра идея да се отправя нова заявка за конекция.

Timeout : Дефинира колко секунди сървърът да изчаква завършването на един трансфер. Стойността трябва да бъде достатъчно голяма за размера на файловете, които сайта изпраща и за ниската производителност на модемните връзки на клиентите. Но ако е зададена прекалено висока стойност, сървърът ще поддържа отворен конекции за клиенти, които може да са офлайн.

BrowserMatch: Намалява производителността в интерес на съвместимостта. Пример за употреба на BrowserMatch:

BrowserMatch “Mozilla/2” nokeepalive
BrowserMatch “MSIE 4\.0b2” nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch “RealPlayer 4\.0” force-response-1.0
BrowserMatch “Java/1\.0” force-response-1.0
BrowserMatch “JDK/1\.0” force-response-1.0

Директивите *BrowserMatch* представят информацията по начин, съвместим с възможностите на различни Web браузъри. Например един браузър може да е в състояние да обработва HTTP 1.0, но не и HTTP 1.1. В този случай на реда *BrowserMatch* се използва *downgrade-1.0* , което кара сървъра да използва само HTTP 1.0, когато работи с този браузър. В по-горния пример за двата браузъра са изключени keep-alive конекциите. За единия браузър се предлага само HTTP 1.0 по време на конекцията, а за четирите различни браузъри отговорите са форматирани така, че да бъдат съвместими с HTTP 1.0.

2.2.1.9. Директиви за кеширане

Няколко директиви контролират как сървърът управлява своя *кеш*. Кешът представлява локално поддържано копие на една Web страница на сървъра. Когато се използват защитни стени, директният достъп до Web често е блокиран. Потребителите се свързват с прокси сървър от локалната мрежа, а прокси сървърът се свързва с отдалечения Web сървър. Не винаги прокси сървърите поддържат кеширани копия на Web страниците, но кеширането увеличава производителността, като намалява трафика, който се изпраща по WAN мрежата и заявките за популярни Web сайтове. Директивите които контролират кеширането, са следните:

ProxyRequests: Задаването на стойност *on* за тази опция превръща Web сървъра в прокси сървър.. По подразбиране стойността е *off*.

ProxyVia: Разрешава или забранява използването на Via: хедъри с които се проследява откъде идват кешираните страници.

CacheRoot: Дефинира директорията, в която се записват кешираните Wev страници. За да не се прави тази директория достъпна за запис от потребителя www-data, се създава специален потребител за httpd.

CacheSize: Дефинира максималния размер на кеша в килобайтове. Стойността по подразбиране е 5KB, което е много малък размер. Много системни администратори считат 100 MB за по- приемлива стойност.

CacheGcInterval : Дефинира интервала по време, на който сървърът намалява размера на кеша. Той се дефинира в часове и по подразбиране е 4. При подразбиращите се настройки сървърът намалява размера на кеша до 5 килобайта на всеки четири часа.

CacheMaxExpire : Дефинира максималния брой часове, за които документът се запазва в кеша без да се изисква опреснено копие от отдалечения сървър. Подразбиращата се стойност е 24 часа. При подразбиращата се стойност един кеширан документ може да остане необновен максимум един ден.

CacheLastModifiedFactor : Дефинира колко дълго един документ да бъде кеширан на базата на това кога за последен път е бил модифициран. Подразбиращата се стойност е 0.1. При нея, ако бъде извлечен документ, модифициран преди 10 часа, той се поддържа в кеша само за един час преди да бъде изискано определено копие. Презумцията е, че ако един документ се променя често, времето на неговата последна модификация ще бъде близко до текущото. Следователно документите, които се променят често, се кешират само за кратък период от време. Независимо от тази настройка, нищо не се кешира по- дълго от CacheMaxExpire.

CacheDefaultExpire : Дефинира подразбиращ се период за кеширане за протоколите, които нямат такава стойност. По подразбиране стойността е един час.

NoCache : Дефинира списък от хост имена на сървъри, страниците на които няма да бъдат кеширани. Ако се знае, че даден сървър съдържа информация, която постоянно се променя, не е редно информацията от конректния сървъра да бъде кеширана. Ако се зададе името на този сървър в директивата *NoCache*, заявките ще се изпращат директно до него, а отговорите няма да се записват в кеша.

2.2.2. Инсталиране и конфигуриране на модули за Apache

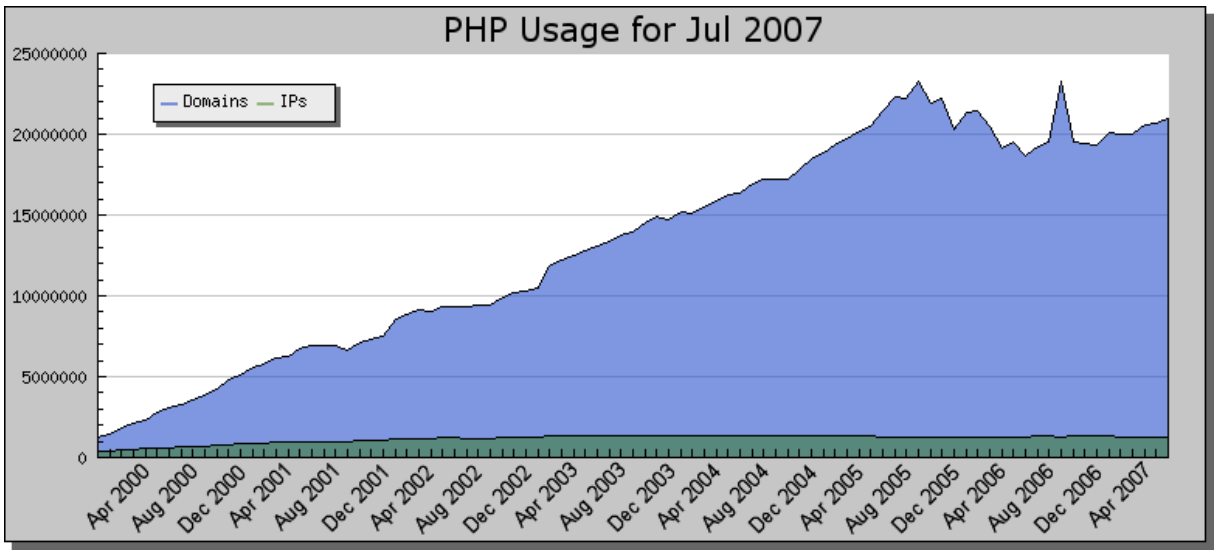
2.2.2.1. Инсталиране на mod_php

mod_php е един от най- използваните модули за Apache днес тъй като интерпретира и изпълнява *.php приложения. Преди да се инсталира PHP респективно mod_php е необходимо да се придобие представа за PHP.

PHP е сървърен скрипт език, проектиран специално за Уеб нужди. PHP кодът може да се вгради в HTML страница и ще се изпълнява при

всяко нейно посещение. Кодът се интерпретира от Уеб сървъра и генерира HTML или други изходни данни, които посетителят на сайта вижда.

PHP е създаден през 1994 г., като първоначално е бил творение само на един човек- Рамус Лерфорд. По- късно е оценен и приет и от други талантливи хора и бива сериозно пренаписан три пъти за да се превърне в познатият ни днес широко разпространен, завършен продукт. Към октомври 2002 година PHP се използва повече от 9 милиона домейна по целия свят, като техният брой бързо расте. Може да се види какъв е текущият брой на http://www.php.net/usage.php или от фигурата по- долу, като за момента PHP се ползва от 20,917,850 домейна, 1,224,183 IP адреси:



Фигура: 2.2

PHP е продукт с отворен код. Това означава, че имате пълен достъп до оригиналния код- може да го използвате,променяте и разпространявате, всичко това безплатно.

Първоначално значение на абревиатурата PHP е Personal Home Page, но по- късно се променя в съответствие със установената Gnu практика за рекурсивни имена (Gnu = Gnu's not Unix) и сега значението му е PHP hypertext Preprocessor. Страницата на PHP може да се посети на адрес http://www.php.net

Модула може да бъде инсталиран по няколко метода както Apache, а именно от пакет(двоичен) или от програмен код(source).
Преди да се инсталира mod_php първо да се изтегли последната версия на PHP която може да бъде достъпна от : http://www.php.net

2.2.2.1.1 Инсталиране на PHP от програмен код

След като се изтегли последната версия на PHP е необходимо тя да бъде разпакетирана/архивирана със следната команда:

tar zxvf php_XX.tar.gz

След това в PHP директорията с програмния код трябва да се създаде Makefile като се изпълни следната команда:

```
./configure \
        --with-apxs2=/usr/local/apache/bin/apxs \
        --with-mysql \
        --prefix=/usr/local/apache/php \
        --with-config-file-path=/usr/local/apache/php \
        --enable-force-cgi-redirect \
        --with-zlib \
        --with-gettext \
        --with-gdbm
```

Най- важните опции са –with-apxs2 и –prefix. Опцията –with-mysql добавя MySQL поддръжка, --with-config-file мести php.ini, --with-zlib позволява използването на gzip-type компресия, --with-gettext е за интернационализиране и –with-gdbm позволява достъп до GDBM бази от данни. След като е създаден файла *Makefile* е необходимо да се изпълни командата: *make*. След изпълнението на командата *make*, трябва да се инсталира, чрез изпълнение на командата *make* с аргумент *install*:

make install

След като PHP е инсталиран на системата, от директорията с програмния код се копира конфигурационния файл на PHP:

cp -p php.ini-recommended /usr/local/apache/php/php.ini

Сега е необходимо да се добавят следните редове в конфигурационния файл на Apache */etc/apache2/apache2.conf*:

```
# Use for PHP 5.x:
LoadModule php5_module        modules/libphp5.so
AddHandler php5-script php

# Add index.php to your DirectoryIndex line:
DirectoryIndex index.html index.php
```

AddType text/html php

 # PHP Syntax Coloring
 # (optional but useful for reading PHP source for debugging):
AddType application/x-httpd-php-source phps

След инсталиране и конфигуриране на PHP поддръжката е необходимо Apache сървъра да бъде рестартиран за да се активира PHP поддръжката:

apache2ctl restart

2.2.2.1.2 Инсталиране на PHP от пакет

Инсталирането на PHP от двоичен пакет доста улеснява повечето администратори понеже не е необходимо прекомпилиране

2.2.2.1.2.1 Инсталиране на PHP от DEB пакет

Пакетът DEB е характерен за дистрибуции като Debian и Uduntu. Инсталирането се извършва чрез следната команда:

dpkg –i php_x.deb

2.2.2.1.2.2 Инсталиране на PHP от RPM пакет

Пакетът RPM е характерен за дистрибуции като RedHat и Fedora. Инсталирането се извършва чрез следната команда:

rpm –ivh php_x.rpm

2.2.2.1.3 Инсталиране на PHP чрез други инсталатори

2.2.2.1.3.1 Инсталиране на PHP чрез apt или apt-get

Доста от GNU/Linux дистрибуциите имат инсталатора apt(Advanced Packaging Tool). APT има реализация във дистрибуции като: Debian, RedHat, Fedora, Knoppix и др. Инсталирането на PHP и mod_php чрез *apt* става със следната команда:

apt-get install php5 libapache2-mod-php5

2.2.2.1.3.2 Инсталиране на PHP чрез yum

Доста от GNU/Linux дистрибуциите имат инсталатора *yum*. *Yum* има реализация във дистрибуции като: RedHat, Fedora и др. Инсталирането на PHP и mod_php чрез *yum* става със следната команда:

yum install php

2.2.2.2. Инсталирането на mod_perl

Преди да се инсталира mod_perl, първо е необходимо да се придобие представа за програмния език Perl.

Perl (произнася се "Пърл", съкращение от Practical Extraction and Report Language) е универсален, интерпретируем език за програмиране, създаден от Лари Уол през 1987 година. Лари Уол е трябвало да създава отчети за системата, която е поддържал тогава и понеже не е имал подходящ инструмент за целта му се е наложило да си създаде сам такъв, а именно Perl.

mod_perl модул за Уеб Сървър позволяващ създаване и изпълняване на уеб базиран и Perl приложения. Страницата на mod_perl може да се посети на адрес http://perl.apache.org/

2.2.2.2.1. Инсталиране на mod_perl от пакет

Инсталирането на *mod_perl* от двоичен пакет доста улеснява повечето администратори понеже не е необходимо прекомпилиране

2.2.2.2.1.1. Инсталиране на mod_perl от DEB пакет

Пакетът DEB е характерен за дистрибуции като Debian и Uduntu. Инсталирането се извършва чрез следната команда:

dpkg –i mod_perl_x.deb

2.2.2.2.1.2. Инсталиране на mod_perl от RPM пакет

Пакетът RPM е характерен за дистрибуции като RedHat и Fedora. Инсталирането се извършва чрез следната команда:

rpm –ivh mod_perl_x.rpm

2.2.2.2.2. Инсталиране на mod_perl чрез други инсталатори

2.2.2.2.2.1 Инсталиране на mod_perl чрез apt или apt-get

Доста от GNU/Linux дистрибуциите имат инсталатора apt(Advanced Packaging Tool). APT има реализация във дистрибуции като: Debian, RedHat, Fedora, Knoppix и др. Инсталирането на Perl и mod_php чрез *apt* става със следната команда:

apt-get install libapache2-mod-perl2

2.2.2.2.2.2 Инсталиране на mod_perl чрез yum

Доста от GNU/Linux дистрибуциите имат инсталатора *yum*. *Yum* има реализация във дистрибуции като: RedHat, Fedora и др. Инсталирането на Perl и mod_perl чрез *yum* става със следната команда:

yum install `mod_perl`

2.2.2.3. Инсталирането на mod_python

Преди да се инсталира mod_python, първо е необходимо да се придобие представа за програмния език Python.

Питон (англ. Python, произнася се Пайтън) е интерпретируем, интерактивен, обектно-ориентиран език за програмиране, създаден от Гуидо ван Росум в началото на 90-те години. Кръстен е на телевизионното шоу на BBC „Monty Python's Flying Circus". Често бива сравняван с Tcl, Perl, Scheme, Java и Ruby.

Питон предлага добра структура и поддръжка за разработка на големи приложения. Той притежава вградени сложни типове данни като гъвкави масиви и речници, за които биха били необходими дни, за да се напишат ефикасно на C.

Програмния език Питон позволява разделянето на една програма на модули, които могат да се използват отново в други програми. Също така притежава голям набор от стандартни модули, които да се използват като основа на програмите. Съществуват и вградени модули, които обезпечават такива неща като файлов вход/изход (I/O), различни системни функции, сокети (sockets), програмни интерфейси към GUI-библиотеки като Tk, както и много други.

Тъй като Питон е език, който се интерпретира, се спестява значително време за разработка, тъй като не са необходими компилиране и свързване (linking) за тестването на дадено приложение. Освен това, бидейки интерпретируем език с идеология сходна с тази на Java, приложение,

написано на него, е сравнително лесно преносимо на множеството от останали платформи (или операционни системи).

Програмите, написани на Питон, са доста компактни и четими, като често те са и по-кратки от еквивалентните им, написани на C/C++. Това е така, тъй като:

* наличните сложни типове данни позволяват изразяването на сложни действия с един-единствен оператор.
* групирането на изразите се извършва чрез отстъп, вместо чрез начални и крайни скоби или някакви други ключови думи (друг език, използващ такъв начин на подредба, е Haskell).
* не са необходими декларации на променливи или аргументи.

mod_python модул за Уеб Сървър позволяващ създаване и изпълняване на уеб базиран и Python приложения. Страницата на mod_python може да се посети на адрес http:// www.modpython.org/

2.2.2.3.1. Инсталиране на mod_python от пакет

Инсталирането на *mod_python* от двоичен пакет доста улеснява повечето администратори понеже не е необходимо прекомпилиране

2.2.2.3.1.1. Инсталиране на mod_python от DEB пакет

Пакетът DEB е характерен за дистрибуции като Debian и Uduntu. Инсталирането се извършва чрез следната команда:

dpkg –i mod_python_x.deb

2.2.2.3.1.2. Инсталиране на mod_python от RPM пакет

Пакетът RPM е характерен за дистрибуции като RedHat и Fedora. Инсталирането се извършва чрез следната команда:

rpm –ivh mod_python_x.rpm

2.2.2.3.2. Инсталиране на mod_python чрез други инсталатори

2.2.2.3.2.1 Инсталиране на mod_python чрез apt или apt-get

Доста от GNU/Linux дистрибуциите имат инсталатора apt(Advanced Packaging Tool). APT има реализация във дистрибуции като: Debian, RedHat, Fedora, Knoppix и др. Инсталирането на Python и mod_python чрез *apt* става със следната команда:

apt-get install libapache2-mod-python

2.2.2.3.2.2. Инсталиране на mod_python чрез yum

Доста от GNU/Linux дистрибуциите имат инсталатора *yum*. *Yum* има реализация във дистрибуции като: RedHat, Fedora и др. Инсталирането на Python и mod_python чрез *yum* става със следната команда:

yum install `mod_pytnon`

2.2.2.4. Инсталиране и конфигуриране на mod_cband

mod_cband е модул за Apache 2 които предоставя възможност за определяне скоростта на виртуален хост(подобно на mod_bandwidth) , определяне брой заявки за секунда, броя на осъществените връзки (подобно на mod_limitipconn), ограничаване на трафика (подобно на mod_curb) и др. mod_cband се разработва от Lukasz Dembinski, Sergey V. Beduev, Kyle Poulter, J. Kendzorra и Adam Dawidowski. Модула се разпространява под GNU GPL лиценз. mod_cband намира най- често приложение в хостинг компании които ограничават трафика на техните потребители (пример: 5 GB за 1 месец.)

2.2.2.4.1. Инсталиране на mod_cband под Дебиан 4.0 (Etch)

Инсталирането се извършва чрез следната команда:

apt-get install libapache2-mod-cband

Излиза прозорец в които е необходимо да се избере дали ще се зареди mod_cband. Ако се избере “yes” mod_cband ще се зареди при рестартиране на Apache, ако се избере ‘No” няма да бъде инсталиран.

2.2.2.4.2. Директиви на mod_cband

Позволени единици при различните директиви:

Трафик
> **kbps, Mbps, Gbps** - Измерват се в "Бит за секунда", респективно: 1024, 1024*1024 и 1024*1024*1024 бита за секунда
> **kb/s, Mb/s, Gb/s** - Измерват се в "Байт за секунда", респективно: 1024, 1024*1024 and 1024*1024*1024 байта за секунда
> По подразбиране kbps

Скорост
> **K, M, G** - Измерват се в Байт, респективно: 1000, 1000*1000 and 1000*1000*1000 байта.
> **Ki, Mi, Gi** - Измерват се в Байт, респективно: 11024, 1024*1024 and 1024*1024*1024 байта.
> По подразбиране K

Период
> **S, M, H, D, W** – Измерват се в секунди: Секунди, Минути, Часове, Дни, Седмици; респективно: 1, 60, 3600, 86400, 604800 секунди
> По подразбиране S

Име: CBandDefaultExceededURL

Описание: URL под подразбиране къде mod_cband ще пренасочва всички заявки до виртуалния хост или потребител когато конфигурирания трансфер е надвишен.

Контекст: Server config
Синтаксис: CBandDefaultExceededURL URL
Пример: CbandDefaultExceededURL www.dhstudio.eu/exceeded.html

Име: CbandDefaultExceededCode
Описание: HTTP код които се изпраща до посетителя когато конфигурирания трафик е надвишен
Контекст: Server config
Синтаксис:CBandDefaultExceededCode HTTP_CODE
Пример: CBandDefaultExceededCode 509

Име: CBandScoreFlushPeriod
Описание: Определя периода след които данните за статистиката ще се записват. Може да се използва за подобряване бързодействието на mod_cband
Подразбиране: 1
Контекст: Server config

Синтаксис: CBandScoreFlushPeriod номер_на_заявки
Пример: CBandScoreFlushPeriod 100
Данните за статистиката ще се записват след 100 заявки

Име: CBandSpeed
Описание: Определя максималната скрост за виртуалния хост
Контекст: <Virtualhost>
Синтаксис: CBandSpeed kbps rps max_conn
kbps- максимална скорост за трансфер в [kMG]bps или [kMG]B/s
rps- Максимален брой заявки за секунда
max_conn- Максимален брой на едновременните връзки
Пример: CBandSpeed 1024 10 30

Определя максимална скорост 1024kbps, максимално 10 заявки за секунда и максимално 30 едновременно осъществени връзки.
Бележка: Тази характеристика е възможна от версия 0.9.6.0

Име: CBandRemoteSpeed
Описание: Определя максималната скрост за други отдалечени клиенти
Контекст: <Virtualhost>
Синтаксис: CBandRemoteSpeed kbps rps max_conn
kbps- максимална скорост за трансфер в [kMG]bps или [kMG]B/s
rps- Максимален брой заявки за секунда
max_conn- Максимален брой на едновременните връзки

Пример: CBandRemoteSpeed 20kb/s 3 3
Определя максимална скорост 20kB/s, максимално 3 заявки за секунда и максимално 3 едновременно осъществени връзки от друг отдалечен клиент.

Бележка: Тази характеристика е възможна от версия 0.9.6.1-rc2

Име: CBandClassRemoteSpeed
Описание: Определя максималната скорост за други отдалечени клиенти от един клас.
Контекст: <Virtualhost>
Синтаксис: CBandClassRemoteSpeed class_name kbps rps

class_name- име на дефинирания клас
kbps- максимална скорост за трансфер в kbps или kB/s
rps- Максимален брой заявки за секунда
max_conn- Максимален брой на едновременните връзки

Пример: <CBandClass googlebot_class>
CBandClassDst 66.249.64/24
CBandClassDst 66.249.65/24
CBandClassDst 66.249.79/24
</CBandClass>

CBandClassRemoteSpeed googlebot_class 20kb/s 2 3
Определя максимална скорост 20kB/s, максимално 2 заявки за секунда и максимално 3 едновременно осъществени връзки от друг отдалечен клиент клас googlebot_class.

Бележка: Тази характеристика е възможна от версия 0.9.6.1-rc2

Име: CBandRandomPulse
Описание: Спира или стартира произволния пулс генератор за изпращаната информяция. Случайния пулс генератор е част от осъществяването на ограничаване по скорост в mod_cband.
Контекст: Global
Синтаксис: CBandRandomPulse On/Off

Име: CBandLimit
Описание: Определя трафик за виртуален хост
Контекст: <Virtualhost>
Синтаксис: CBandLimit limit
limit- Размер на позволения трафик, възможни единици: K (kilo), M (mega), G (giga), Ki (kibi), Mi (mebi), Gi (gibi)

Пример: CBandLimit 10M
Определя трафик от 10 * 1000 * 1000 bytes

CBandLimit 10Mi
Определя трафик от 10 * 1024 * 1024 bytes

Бележка: Значението на K,M и G е променено с версия 0.9.6.0. Проверете вашия конфигурационен файл.

Име: CBandClassLimit
Описание: Определя трафик за виртуален хост чрез клас
Контекст:<Virtualhost>
Синтаксис: CBandClassLimit class_name limit

class_name- име на дефиниран клас
limit- Размер на позволения трафик, възможни единици: K (kilo), M (mega), G (giga), Ki (kibi), Mi (mebi), Gi (gibi)

Бележка: Значението на K,M и G е променено с версия 0.9.6.0. Проверете вашия конфигурационен файл.

Име: CBandExceededURL
Описание: URL където mod_cband ще пренасочва всички заявки до виртуалния хост или потребител когато конфигурирания трансфер е надвишен. Ако не определите URL тогава ще бъде изпратен http код 503 (Service Unavailable)

Контекст: <Virtualhost>
Синтаксис: CBandExceededURL URL

Пример: CbandDefaultExceededURL www.dhstudio.eu/exceeded.html

Име: CBandExceededSpeed

Описание: Определя максималната скорост на виртуалния хост когато конфигурирания трафик е надвишен.

Контекст: <Virtualhost>
Синтаксис: CBandExceededSpeed kbps rps max_conn
kbps- максимална скорост за трансфер в kbps или kB/s
rps- Максимален брой заявки за секунда
max_conn- Максимален брой на едновременните връзки

Бележка: Тази характеристика е възможна от версия 0.9.6.0

Име: CBandScoreboard
Описание: Определя файл за записване на статистика
Контекст: <Virtualhost>
Синтаксис: CBandScoreboard path

Бележка: Този файл трябва да има възможност за промяна от потребител apache

Име: CBandPeriod
Описание: Определя период след които да бъде изчистен файл със статистики за виртуалния хост.
Контекст: <Virtualhost>
Синтаксис: CBandPeriod period
period – възможни единици: S (seconds), M (minutes), H (hours), D (days), W (weeks)

Пример: CBandPeriod 1W
CBandPeriod 14D
CBandPeriod 60M

Име: CBandPeriodSlice

Описание: Определя период на разделяне
Подразбиране: slice_len = limit
Контекст: slice_len = limit
Синтаксис: CBandPeriodSlice slice_length

Пример: CBandLimit 100G
CBandPeriod 4W
CBandPeriodSlice 1W

Периода ще бъде разделен на 4 малки парчета(4W/1W=4).Всяко парче има 100G/4=25G трафик.След една седмица ограничението ще бъде 50G, след две седмици 75W и т.н.

Име: <CBandUser>
Описание: Дефинира нов cband потребител
Контекст: Server config
Синтаксис: <CBandUser user_name>

Име: CBandUserSpeed
Описание: Определя максимална скорост за cband потребител
Контекст: <CBandUser>
Синтаксис: CBandUserSpeed kbps rps max_conn
kbps- максимална скорост за трансфер в kbps или kB/s
rps- Максимален брой заявки за секунда
max_conn- Максимален брой на едновременните връзки

Пример: CBandUserSpeed 100kb/s 10 5

Определя максимална скорост 100kB/s, максимално 10 заявки за секунда и максимално 5 едновременно осъществени връзки.

Бележка: Тази характеристика е възможна от версия 0.9.6.0

Име: CBandUserLimit
Описание: Определя максимален трафик за cband потребител
Контекст:<CBandUser>
Синтаксис: CBandUserLimit limit
limit- Размер на позволения трафик, възможни единици: K (kilo), M (mega), G (giga), Ki (kibi), Mi (mebi), Gi (gibi)

Пример: CBandUserLimit 10M

Определя 10 * 1000 * 1000 bytes трафик

CBandUserLimit 10Mi
Определя 10 * 1024 * 1024 bytes трафик

Бележка: Значението на K,M и G е променено с версия 0.9.6.0. Проверете вашия конфигурационен файл.

Име: CBandUserClassLimit
Описание: Определя трафик за cband потребители чрез клас
Контекст:<CBandUser>
Синтаксис: CBandUserClassLimit class_name limit

class_name- име на дефиниран клас
limit- Размер на позволения трафик, възможни единици: K (kilo), M (mega), G (giga), Ki (kibi), Mi (mebi), Gi (gibi)

Бележка: Значението на K,M и G е променено с версия 0.9.6.0. Проверете вашия конфигурационен файл.

Име: CBandUserExceededURL
Описание: Определя URL където mod_cband ще пренасочва всички заявки до потребителски виртуални хостове които са с надвишен трафик. Ако не определите URL тогава ще бъде изпратен http код 503 (Service Unavailable)

Контекст:<CBandUser>
Синтаксис: CBandUserExceededURL URL

Име: CBandUserExceededSpeed
Описание: Определя максималната скорост на виртуалния хост когато конфигурирания трафик е надвишен. При използване на тази директива не трябва да използвате CBandUserExceededURL.
Контекст:<CBandUser>
Синтаксис: CBandUserExceededSpeed kbps rps max_conn
kbps- максимална скорост за трансфер в kbps или kB/s
rps- Максимален брой заявки за секунда
max_conn- Максимален брой на едновременните връзки

Бележка: Тази характеристика е възможна от версия 0.9.6.0

Име: CBandUserScoreboard
Описание: Определя файл за записване на статистика
Контекст: <CBandUser>
Синтаксис: CBandUserScoreboard path

Бележка: Този файл трябва да има възможност за промяна от потребител apache

Име: CBandUserPeriod
Описание: Specifies a period after which a user's usages are cleared
Контекст: <CBandUser>
Синтаксис: CBandUserPeriod period
period – възможни единици: S (seconds), M (minutes), H (hours), D (days), W (weeks)

Пример: CBandUserPeriod 1W
CBandUserPeriod 14D
CBandUserPeriod 60M

Име: CBandUserPeriodSlice
Описание: Specifies a period slice length
Подразбиране: slice_len = limit
Контекст: <CBandUser>
Синтаксис: CBandUserPeriodSlice slice_length

Пример: CBandUserLimit 100G
CBandUserPeriod 4W
CBandUserPeriodSlice 1W

Периода ще бъде разделен на 4 малки парчета(4W/1W=4).Всяко парче има 100G/4=25G трафик.След една седмица ограничението ще бъде 50G, след две седмици 75W и т.н.

2.2.2.4.3 Примерни конфигурации на mod_cband

- Ограничаване по скорост

#Дефиниране на клас

<CBandClass local_traffic>
CBandClassDst 172.16.0.0/16
CBandClassDst 192.168.0.0/16
</CbandClass>

<VirtualHost *>
ServerName dhstudio.eu
ServerAlias www.dhstudio.eu
DocumentRoot /var/www/
ErrorLog /var/www/error.log
LogLevel warn
CustomLog /var/www/access.log combined
<Directory "/var/www/">
Options Indexes MultiViews
AllowOverride None
Order allow,deny
Allow from all
</Directory>

#Максимална скорост 1024 kbps за този виртуален хост
#Максимално 10 заявки за секунда за този виртуален хост
#Максимално 30 отворени връзки за този хост
CBandSpeed 1024 10 30

#Максимална скорост от 20kb/s, 2 заявки за секунда, 3 отворени връзки за клас local_traffic
CBandClassRemoteSpeed local_traffic 20kb/s 2 3
</VirtualHost>

Бележка: Когато създавате класове трябва да ги описвате преди директивата <VirtualHost>. Когато създавате ограничения на виртуален хост трябва да бъдат описани след директивата ServerName.

- Ограничаване по трафик

CBandDefaultExceededURL http://www.dhstudio.eu/bandwidth_exceeded.html
<VirtualHost *>
ServerName dhstudio.eu

ServerAlias www.dhstudio.eu
DocumentRoot /var/www/
ErrorLog /var/www/error.log
LogLevel warn
CustomLog /var/www/access.log combined
<Directory "/var/www/">
Options Indexes MultiViews
AllowOverride None
Order allow,deny
Allow from all
</Directory>

#Възможен трафик за този виртуален хост 100 MB
CBandLimit 100000
#Когато трафика е надвишен пренасочва към
http://www.dhstudio.eu/bandwidth_exceeded.html
CBandExceededURL http://abc.org/bandwidth_exceeded.html

 # Файл със статистика
CBandScoreboard /var/www/cband/dhstudio.scoreboard
 # Период след които файла със статистика ще бъде занулен (30 минути)
#(само във версии >=0.9.5-rc2)

CBandPeriod 30M
</VirtualHost>

Бележка: Когато създавате класове трябва да ги описвате преди директивата <VirtualHost>. Когато създавате ограничения на виртуален хост трябва да бъдат описани след директивата ServerName.

2.2.2.4.4 Извеждане на уеб статистика чрез mod_cband

При инсталирането в конфигурационния файл на mod_cband(/etc/apache2/mods-enabled/cband.conf) се описва от къде може да бъде изведена статистика за ограничените виртуални хостове:

<Location /cband-status>
SetHandler cband-status
</Location>

<Location /cband-status-me>
SetHandler cband-status-me
</Location>

Може да се види статистиката от:

http://www.site_cband.com/cband-status
http://www.site_cband.com/cband-status-me

Ако е необходимо статистиката може да бъде представена под XML формат:

http://www.site_cband.com/cband-status?xml
http://www.site_cband.com/cband-status-me?xml

В този си вид статистиката е свободна за гледане от абсолютно всеки. Възможна е употребата на защита чрез парола например:

Файл: /etc/apache2/mods-enabled/cband.conf

<IfModule mod_cband.c>

<Location /cband-status>
Order allow,deny
Allow from all

SetHandler cband-status
AuthType basic
AuthName "Cband Authentication"
Require valid-user
AuthUserFile /var/www/.htpasswd
</Location>

<Location /cband-status-me>
SetHandler cband-status-me
</Location>

</IfModule>

След като редактирахте конфигурационния файл на mod_cband е необходимо да се създаде потребител:

htpasswd -c /var/www/.htpasswd cband

mod_cband е лесен за инсталиране и конфигуриране, а освен това и работи чудесно. Той се ползва най- често от хостинг компании, но също и от всеки които има необходимост да ограничи скоростта или трафика на виртуален хост или хостове.

2.3. Сигурност на Web сървър Apache

Web сървърите са изложени на всички проблеми свързани със сигурността. Освен всички обичайни заплахи, като проникване в мрежата и атаки чрез отказ на услугата (denial of service -DOS), Web сървърите отговарят за защитата на целостта на информацията разпространявана от сървъра и за защитата на информацията, изпращана от клиента до сървъра. Сигурността е една от най- важните части от конфигурирането на Apache.

2.3.1 Опции на сървъра за документи и директории. Дефиниране настройки за контрол на достъп.

Директивата *Options* определя кои сървърни опции са разрешени за документите. Поставянето на директивата Options в контейнер Directory ограничава нейния обхват само до тази конкретна директория. Пример:

<Directory />
Options FollowSymLinks
AllowOverride None

</Directory>

<Directory "/var/www">
Options Indexes FollowSymLinks
AllowOverride None
Order allow,deny
Allow from all
</Directory>

<Directory "/var/www/icons">
Options Indexes MultiViews
AllowOverride None
Order allow,deny
Allow from all
</Directory>

<Directory "/var/www/cgi-bin">
AllowOverride None
Options None
Order allow,deny
Allow from all
</Directory>

<Directory /usr/share/doc>
Order deny,allow
Deny from all
Allow from localhost
Options Indexes FollowSymLinks
</Directory>

Примера дефинира сървърни опции за пет директории: коренната (/) и директориите */var/www/, /var/www/icons, /var/www/cgi-bin u /usr/share/doc* . Примерът показва четири възможни стойности за директивата Options: *FollowSymLinks, Indexes, None u MultiView*. Директивата *Options* има няколко възможни настройки:

All Разрешава използването на всички сървърни опции

ExecCGI Разрешава изпълнението на CGI скриптове от тази директория. Опцията ExecCGI позволява CGI скриптове да бъдат изпълнявани от директории, различни от директорията, посочена в директивата ScriptAlias.

FollowSymLinks Разрешава използването на символни връзки. Ако тази опция е разрешена, сървърът третира една символна връзка като документ от директорията.

Includes Разрешава използването на Server Side Includes (SSI)

IncludeNOEXEC Разрешава Server Side Includes (SSI), Които не включват #exec и #include команди

Indexes Разрешава генериран от сървъра листинг на директорията, ако не бъде намерен файлът index.html

MultiViews Разрешава договаряне на езика на документите.

None Не разрешава никакви сървърни опции. Това осигурява най- високото ниво на сигурност.

SymLinksIfOwnerMatch Разрешава използването на символни връзки, ако файлът, към който сочи връзката, е собственост на потребителя, собственик на самата връзка

В предходния пример има контейнери за директории които съдържат директиви *AllowOverride*. Тези директиви ограничават обема на конфигурационния контрол, даван от отделните директории.

В допълнение към директивите Options и AllowOverride в примера , контейнерите Directory съдържат директиви Order, Allow и Deny. Тези три директиви позволяват дефиниране контрол на достъп на ниво хост. Настройките за контрол на достъп на ниво хост са следните:

Order Дефинира реда, в който правилата за контрол на достъпа се прилагат. Командният ред *order deny,allow* указва на apache първо да приложи правилото дефинирано от директивата *Deny* и след това да разреши изключения от това правило на базата на правилото дефинирано от директивата *Allow*.

Deny from Идентифицира хостове, на които не е разрешено да осъществяват достъп до Web документите, намиращи се в тази директория. Хостът може да бъде идентифициран с пълно или частично хост име, IP адрес или домейн. Ключовата дума *all* блокира всички хостове.

Allow from Идентифицира хостове, на които е позволено да осъществяват достъп до документите. Хостът може да бъде идентифициран с помощта на пълно или частично хост име или IP адрес. Ако се използва име на домейн,

то указва всички хостове от домейна. Ключовата дума *all* разрешава всички хостове.

2.3.2 Конфигуриране контрол на ниво директория

Директивата *AccessFileName .htaccess* от apache2.conf(/etc/apache2/apache2.conf) разрешава използването на конфигурационен файл на ниво директория и указва, че името на конфигурационния файл е *.htaccess*. Ако сървърът открие файл с това име в директорията , той прилага конфигурационните команди, дефинирани в него, за директорията, на базата на директивата *AllowOverride*. Файлът *.htaccess* е подобен на контейнер *Directory*. По- същия начин , по който директивите в един контейнер *Directory* се прилагат за конкретна директория, директивите във файла *.htaccess* се прилагат само за директорията, в която се намира файлът. Директивите във файла .htaccess потенциално са същите като тези, използвани във файла apache2.conf, който дефинира конфигурацията на цялата система.

Директивата *AllowOverride* има шест ключови думи, които задават нивото на конфигурационния контрол, предоставен на файла *.htaccess*:

None Не позволява предефиниране на конфигурацията. В резултат от това *None* забранява файла *.htaccess*

All Разрешава файлът *.htaccess* да предефинира всичко, дефинирано в конфигурационните файлове на Apache, за които предефинирането е разрешено. Това е същото като задаването на всичките четири останали ключови думи: *AuthConfig, FileInfo, Indexes* и *Limit*

AuthConfig Разрешава файлът *.htaccess* да дефинира директиви за автентикация на потребители.

FileInfo Позволява файлът да използва директиви, които контролират типовете документи.

Indexes Разрешава *.htaccess* да конфигурира индексите с нестандартно форматиране.

Limit Позволява на файла да конфигурира настройки за контрол на достъпа на ниво хост.

В допълнение към тези ключови думи, чрез *AllowOverride* могат да се разрешават отделни команди. Пример за разрешаване на една

директория да дефинира свои собствени асоциации на файлови разширения:

AllowOverride AddType

Директивите Options и AllowOverride контролират достъпа до възможностите на сървъра и настройките за предефиниране на конфигурацията, което помага за запазване на информацията от повреди.

2.3.3 Използване на iptables с цел повишаване нивото на сигурност на системата

IPTABLES е помощно средство което се използва за конфигуриране на защитната стена в GNU/Linux. Като част от структурата на *netfilter* в ядрото на GNU/Linux, *iptables* заменя ipchains.

В приложение № 2 се онагледява запитна стена, сравнително надеждно конфигурирани правила за iptables които инкрементират сигурността на системата.

2.4. Мониторинг и поддръжка на Apache

Няколко директиви в конфигурацията на Apache конфигурират поддръжката на файлове-дневници(*Log Files*). Директивата ErrorLog дефинира пътя до файла-дневник за грешки.

Директивата *LogLevel* дефинира какви типове събития се записват в дневника за грешките. За *LogLevel* има осем възможни настройки: *debug, info, notice, warn, error, crit, alert u emerg*. Нивата на дневника са кумулативни. Настройката *debug* осигурява информация за дебъгване и всички други типове информация за дневника. Настройката warn осигурява предупреждения, грешки, критични съобщения, както и съобщения за внимание и спешни случаи. Debug кара файла да нараства много бързо. Emerg запазва файла малък като извежда информация само за сериозни проблеми. warn е добър компромис между достатъчното детайли и прекалено многото детайли.

Директивата TransferLog дефинира пътя до дневника, в който httpd записва информация за активността на сървъра. Дневниците осигуряват и информация за това от кого се използва сървъра, колко е използван и колко добре обслужва потребителите.

Директивите LogFormat дефинират формата на записите във файла-дневник. Файловете, в които се записват тези записи, се дефинират от директивите CustomLog.

Файловете-дневници на Apache отговарят на формата Common Log Format (CLF). CLF е стандарт, използван от всички разработчици на Web сървъри. Използването на този формат означава, че дневниците, генерирани от сървърите Apache, могат да бъдат обработвани от всеки инструмент за анализ на дневници, който също поддържа този стандарт.

Форматът на един стандартен CLF запис се дефинира със следната директива LogFormat в следния пример:

LogFormat "%h %l %u %t \"%r\" %>s %b" common

Един CLF запис съдържа седем полета, всяко представено от параметър в директивата LogFormat:

%h Записва в дневника IP адреса или хост името на клиента. Ако HostnameLookups има стойност on, това е напълно определеното име на хоста на клиента. Опцията HostnameLookups доста чест е изключена (off) понеже намалява производителността на сървъра във включено състояние.

%l Записва в дневника потребителското име, зададено на потребителя при клиента. Потребителското име се извлича с помощта на протокола identd. Повечето клиенти не използват identd и не осигуряват тази информация, затова това поле обикновено съдържа "-" идентифициращо липсващата стойност.

%u Записва в дневника потребителското име, използвано за достъп до Web страница, защитена с парола. Това име трябва да отговаря на името, дефинирано във файл създаден с цел автентикация.

%t Записва в дневника дата и час

%r Записва в дневника първия ред на заявката, който е често URL адресът на заявения документ. Знаковете *\"* служат единствено за поставяне на кавички в изходните данни.

%>s Записва в дневника състоянието на последната заявка. Това е трицифреният код на отговор, който сървърът връща на клиента. Знакът > е либерален знак, който ще се появява във файла- дневник преди кога на отговора.

%b Записва в дневника броя на изброените байтове.

Форматът на директивата LogFormat е ограден в кавички. Етикетът "common" не е част от формата. Това е произволен низ, използван за

обвързване на директивата LogFormat с директивата CustomLog. Пример за “комбиниран” формат:

LogFormat “%h %l %u %t \”%r\” %>s %b \”%{Referer}i\” \”%{User-Agent}i\”” combined

Apache може да записва в дневник съдържанието на приетите или изпратените хедъри. В примера е указано да се записва приетата стойност от хедъра на клиента User-Agent “ *%{User-Agent}i* “ . “i” индицира, че това е входящ хедър. За записване на изходящ хедър се използва “o” . Хедърът User-Agent съдържа името на браузъра, използван от клиента. Хедърът Referer съдържа името на отдалечение сървър, свързан към Web сървъра.

Apache поддържа условно записване в дневника, при което се записват зададени полета само когато са изпълнени определени условия. Условията, които могат да бъдат тествани, са кодовете на състояние, връщани от сървъра. Кодовете за състояние са:

202: OK Заявката е валидна
302: Found Заявеният документ е намерен
304: Not Modified Заявеният документ не е модифициран
400: Bad Request Заявката е невалидна
401: Unauthorized На клиента или потребителя е отказан достъп
403: Forbidden Изискваният достъп не е разрешен
404: Not Found Изискваният документ не съществува
500: Server Error има неустановена грешка на сървъра
503: Out of Resources (Service Unavailable) Сървърът има недостатъчни ресурси за изпълнение на заявката.
501: Not Implemented Исканата възможност на сървъра не е достъпна
502: Bad Gateway Клиентът е задал невалиден шлюз (gateway)

За да се направи дадено поле условно се поставя код на състояние за полето в записа *LogFormat*. Пример за записване в дневника името на браузъра само ако браузърът изисква услуга, която не е имплементирана в Apache. Комбинира се кога за състояние *Not Implemented (501)* с хедъра *User-Agent* по следния начин:

%501{User-Agent}i

Ако тази стойност се съдържа в LogFormat, името на браузъра се записва в дневника само когато кодът на състоянието е 501. Може също да се ползва удивителен знак “!” за да се зададе че трябва да се записва дадена стойност само когато кодът на състоянието не е конкретна стойност

(удивителният знак “!” индицира логическата операция “not”. Пример за употреба на удивителния знак “!”:

%!200,302,304{Referer}i

Този конкретен условен запис извежда съобщение когато някои отдалечена страница има остаряла връзка, сочеща към Web сървъра. Този пример показва също, че е възможна употребата на повече от един код за състояние.

Web сървърите са важна част от Интернет на всяка организация. GNU/Linux е отлична платформа за Web използващ софтуера Apache. Apache е най- популярният Web сървър в Интернет. Под GNU/Linux той може ефективно да поддържа Web сайта на голяма организация.

Apache се разпространява под свой собствен лиценз - Apache License. Той има малки разлики с GNU GPL, но текущата чернова на GPL версия 3 има секция (7a), която позволява съвместимост с лицензи, подобни на този на Apache.