Webサーバ設定

Apacheを利用したWebサーバ構築

Ver. 1.0

2006.7.2

Webサーバ設定　目次

- Basics
  - Webサーバの種類
  - Apacheの概要
  - 標準設定(httpd.conf)
  - その他の設定
- More
  - ユーザ認証
  - リダイレクト
  - バーチャルホスティング
  - SSL
- Tips
  - サーバ情報
  - ログ

環境

- 本講座で使用するサーバ環境
  - OS: Red Hat Enterprise Linux 4
  - Server: Apache 2.0
  - CPU: Pentium4 2.8GHz
  - Memory: 512MB
  - Network: 192.168.0.0/24

Basics

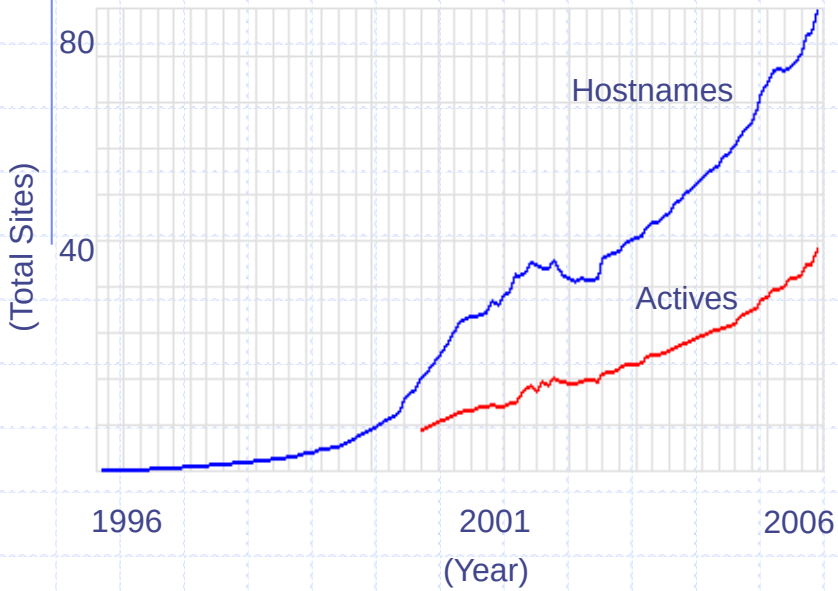
- Basics
  - Webサーバの種類
  - Apacheの概要
  - 標準設定
  - その他の設定

Webサーバの種類

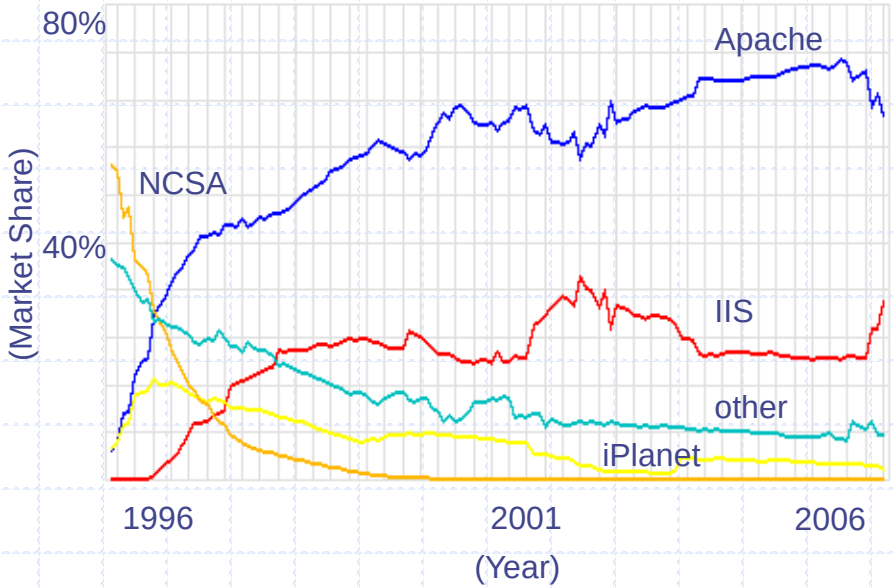
- Apache
  - WebサーバシェアNo.1、61%
  - Apache Software Foundation (ASF)によって配布されるフリーソフトウェア
  - 1.3系と2.0系がある
  - 主にUNIXで使用されている、Windows版もある
  - 自由なカスタマイズが可能
- IIS (Internet Information Server)
  - Microsoft社製
  - Windowsサーバシェア拡大とともにシェアを拡大中、29%
  - 最新版のIIS6.0はセキュリティ面も万全
  - Apacheよりもパフォーマンスで優位
  - ASP.NETを利用可
- Zeus Web Server
  - イギリスのZeus Technology社製
  - シェアは数%だが、大量のバーチャルホスティングが可能
  - ホスティング事業者やISPでよく使用されている

Webサーバシェア

- Webサイト数



- Webサーバシェア



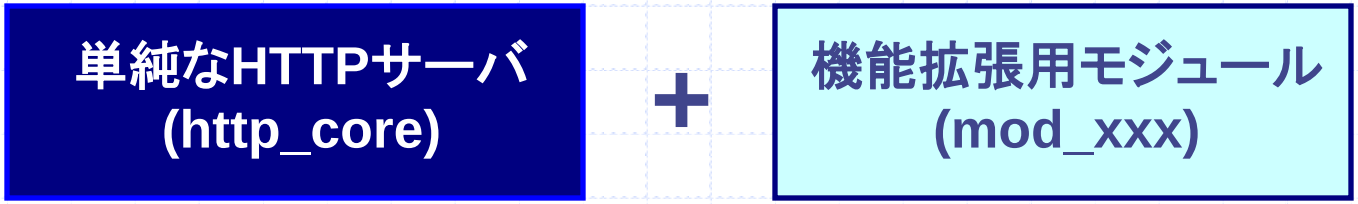
（Netcraft Web Server Surveyより2006年6月のデータ）

Apache2.0の新機能

- SSL標準対応
  - SSL対応を実装するmod_sslモジュールが標準で組み込まれた
- フィルタリング機能
  - CGIの出力をSSIに渡すなどの多段構成が可能
- IPv6のサポート
- マルチスレッドのサポート
  - マルチプロセス・マルチプロセスにより高速化が可能
  - 次の3つの動作モード
    - ✓ prefork : マルチスレッドは使わず、事前に生成したプロセス
    - ✓ worker : プロセスとマルチスレッドを使用する。スレッドの数に制限がある。
    - ✓ perchild :プロセスとマルチスレッドを使用する。スレッドの数に制限がない。
- inetd経由の起動をサポートしない
  - アクセスごとの起動は負荷が高いためサポートしなくなった
- モジュールの組み込み
  - モジュールの組み込みは自動処理
- 設定ファイル
  - 一部変更

Apache内部構造

- Apache httpdの内部構成

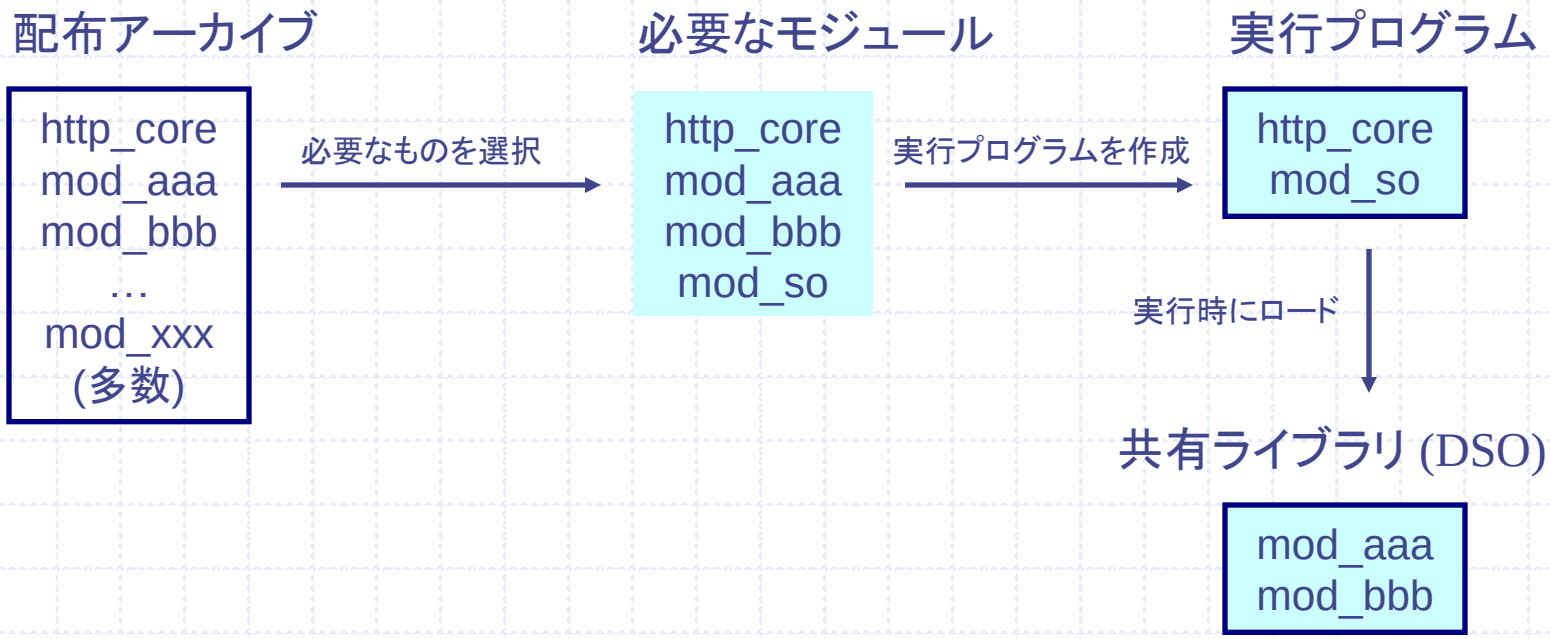


クライアントからの要求に
対するレスポンス部分

様々な付加処理

機能拡張モジュールの組み込み

- DSOによるモジュール組み込み



DSO (Dynamic Shared Object)
モジュールをすべて実行プログラムに組み込まずに、一部を共有ライブラリの形にしておき、設定ファイルの内容に応じてそれを読み込み仕組みのこと。
DSOの仕組みは、モジュール(mod_so)で実現されています。

ディレクトリ構成

- Apacheのディレクトリ構成

	バイナリパッケージからインストール	ソースファイルからインストール
Darmon Program	/usr/sbin/httpd	/usr/sbin/httpd
Pid file	/var/run/httpd.pid	/usr/local/apache2/logs/httpd.pid
Lock file	/var/lock/subsys/httpd	/var/lock/subsys/httpd
Server Root	/etc/httpd/	/usr/local/apache2/
Library Files Directory	/usr/lib/	/usr/local/apache2/lib/
Module Files Directory	/usr/lib/httpd/modules/	/usr/local/apache2/modules/
Configuration Files	/etc/httpd/conf/httpd.conf	/usr/local/apache2/httpd.conf
Control Script	/etc/init.d/httpd	/usr/local/apache2/bin/apachectl
Document Root	/var/www/html	/usr/local/apache2/htdocs
Log Files	/var/log/httpd/access_log /var/log/httpd/error_log	/usr/local/apache2/logs/access_log /usr/local/apache2/logs/error_log

※バイナリパッケージはRed Hat Enterprise Linuxを使用

- 設定ファイル構成

ファイル	機能
httpd.conf	httpdに関する基本的な設定
mime.types	サフィックス(拡張子)によるMIMEファイルタイプの設定
magic	ファイル内容(magic)によるMIMEファイルタイプの設定

デフォルト設定で動作確認

/etc/httpd/conf/httpd.conf

```
…
ServerAdmin webmaster@test.net
…
DocumentRoot "/var/www/html"
…
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
…
```

デフォルト設定で
動作可能

コンフィグテスト

```
# service httpd configtest
Syntax OK
```

起動

```
# service httpd start
```

サーバ起動時の自動起動設定

```
# chkconfig httpd on
```

変更する設定

サーバ情報

```
ServerTokens Prod
ServerSignature Off
```

セキュリティの観点から、サーバの情報はなるべく出さないほうが良いです。

待機するアドレスとポート

```
Listen 192.168.1.2:80
```

待機するアドレスを設定します。

サーバ名

```
ServerName www.test.net:80
```

DNS逆引きをしなくても良いようにします。

DocumentRoot

```
DocumentRoot "/home/webmgr/html"
```

DocumentRootを別の場所にする場合は設定します。

ディレクトリアクセス制御

```
<Directory “/home/webmgr/html”>
…
 Options FollowSymLinks
…
</Directory>
```

DocumentRootを変更した場合に、アクセス制御するディレクトリも変更します。ディレクトリインデックスの作成は許可しないようにします。

IPアドレスによるアクセス制御

```
Order deny, allow
Deny from all
Allow from 192.168.1.0/255.255.255.0
```

内部向けページの場合は、IPアドレスでアクセス制限を設けます。

変更する設定

不必要なエイリアスの削除

```
Alias /icons/ "/var/www/icons/“
Alias /manual "/var/www/manual"
```

サイトと関係のないエイリアスを削除して閲覧不可能にします。

スクリプトアイリアス

```
ScriptAlias /cgi-bin/ "/home/webmgr/cgi-bin/"
```

必要に応じてスクリプトの置かれるディレクトリを変更します。

文字エンコード

```
AddDefaultCharset EUC-JP
```

文字エンコードを変更します。

SSL無効化

/etc/httpd/conf.d/ssl.conf

```
SSLEngine off
```

SSLを利用しない場合は無効にします。

httpd.conf

- httpd.confの構造
  - Section 1: Global Environment
  - Section 2: ‘Main’ server configuration
  - Section 3: Virtual Hosts
- httpd.confの書式

ディレクティブ　設定値

<Directory>ディレクティブ

```
<Directory />
    Options FollowSymlinks
    AllowOverride None
</Directory>
```

他に<File>, <Location>, <Limit>など

モジュール機能の設定

```
<IfModule mod_userdir.c>
    UserDir public_html
</IfModule>
```

※httpd.confを編集後は必ずApacheの再起動が必要

Section1: Global Environment

/etc/httpd/conf/httpd.conf (core)

```
### Section 1: Global Environment
ServerTokens OS                         ---(1)
ServerRoot "/etc/httpd“                 ---(2)
#ScoreBoardFile run/httpd.scoreboard                ---(3)
PidFile run/httpd.pid
Timeout 300                                         ---(4)
KeepAlive Off                                       ---(5)
MaxKeepAliveRequests 100
KeepAliveTimeout 15
<IfModule prefork.c>                    ---(6)
StartServers       8
MinSpareServers    5
MaxSpareServers   20
MaxClients       150
MaxRequestsPerChild  1000
</IfModule>
Listen 0.0.0.0:80                       ---(7)
LoadModule …                            ---(8)
Include conf.d/*.conf                   ---(9)
#ExtendedStatus On                      ---(10)
```

Section1: Global Environment

(1) クライアントに返送するサーバ情報の設定

```
ServerToken Full | OS | Minor | Minimal | Major | Prod
```

クライアントに送り返すレスポンスヘッダ内に、サーバの一般的な OS 種別や、コンパイルされて組み込まれているモジュールの情報を 含めるかどうかを指定します。この設定はサーバ全体に適用され、バーチャルホスト上で有効にしたり 無効にしたりはできません。
バージョン 2.0.44 以降ではこのディレクティブは ServerSignature ディレクティブにより表示される情報も制御します。セキュリティ的にはサーバの情報をクライアントに流さないことが望ましいですのでMajorやProdあたりに設定します。

(2) 設定ファイルの置き場所の指定

```
ServerRoot <directory>
```

httpdの設定ファイルやログファイルを置くディレクトリを指定します。

(3) プロセスファイル等の置き場所の指定

```
ScoreBoardFile <filename>
PidFile <filename>
```

ScoreBoardファイル（内部のサーバプロセス情報）とProcessIDファイルのファイル名を指定します。

(4) タイムアウトに関する設定

```
Timeout <sec>
```

Apacheの定める3つの待機時間の合計を秒で指定します。
・GETリクエストを受け取るまでの時間の合計
・POST, PUTリクエストにおける、受け取るTCPパケットの間隔の合計
・レスポンスにおけるACKパケットの間隔の合計

Section1: Global Environment

(5) KeepAliveに関する設定

```
KeepAlive On | Off
MaxKeepAliveRequests <count>
KeepAliveTimeout <sec>
```

Keep-Alive（1個のコネクションで複数回のリクエストを送ること）を有効にするかどうかを指定します。
MaxKeepAliveRequestsでは、1個のコネクションが受けつけることのできるリクエストの上限の回数を指定します。0を指定すると無制限になります。
KeepAliveTimeoutでは、リクエストを受け取ってから次のリクエストを待つ時間を指定します。

(6) perforkの設定

```
StartServers <count>
MinSpareServers <count>
MaxSpareServers <count>
MaxClients <count>
MaxRequestsPerChild <count>
```

MPMがperforkの場合の動作について設定しています。デフォルトでは、MPMはperforkの動作になっています。
StartServers, MinSpareServers, MaxSpareServersは、アイドル状態にあるhttpdの子プロセスのそれぞれ、起動時、最小、最大の個数を指定します。
MaxClients, MaxRequestsPerChildは、それぞれ、httpdに接続を許可する最大のクライアント数と、1プロセスあたりが処理する最大のリクエスト数を指定します。 MaxRequestsPerChildに0を指定すると、httpdの子プロセスが無制限にリクエストを処理することになります。ただし、トラブルのもとになりますので、通常は行いません。

Section1: Global Environment

(7) 待機するアドレスとポートの指定

```
Listen [<address>:]<port>
```

リクエストを待つアドレスとポートを指定します。<address>が指定されていない場合には、すべてのインタフェースでリクエストを待つことになります。

(8) 他の設定ファイルの読み込みに関する指定

```
Include <filename>
```

他の設定ファイルを読み込む場合に、その設定ファイルの名前を指定します。

(9) モジュールの使用に関する指定

```
LoadModule <module> <file>
```

DSO (Dymamic Shared Object) によってロードするモジュールを指定します。<module>という名前のモジュールを、<file>から読み出します。

(10) ステータス情報の仕様の指定

```
ExtendedStatus On | Off
```

server-statusハンドラが呼び出された際に出力する、httpdのステータス情報の仕様を指定します。ステータス情報の仕様にはbasicとfullの2種類があります。

デフォルト設定はOffで、basicが指定されたとみなされます。Onを指定すると、fullが指定されたとみなされます。

server-statusハンドラが呼び出されるのは、URLのファイル指定が、/server-statusとなっている場合です。server-statusハンドラが呼び出されると、HTTPサーバの状態を様々な統計情報と同時に出力します。

Section2: ‘Main’ server configuration

/etc/httpd/conf/httpd.conf (主要な設定)

```
### Section 2: 'Main' server configuration
User apache                                  ---(1)
Group apache
ServerAdmin root@localhost                   ---(2)
#ServerName new.host.name:80     ---(3)
UseCanonicalName Off             ---(4)
DocumentRoot "/var/www/html"     ---(5)
```

(1) プロセス所有者の指定

```
User <user name> | #<user id>
Group <group name> | #<goup id>
```

個々のhttpdサーバプロセスの所有者であるユーザとグループを指定します。それぞれ、名前かまたはIDで指定します。IDで指定する場合は、先頭に#をつけます。

(2) サーバ管理者の指定

```
ServerAdmin <mail address>
```

サーバの管理者のアドレスを指定します。サーバがクライアントに返すエラーページに表示されます。

Section2: ‘Main’ server configuration

(3) サーバ名の指定

```
ServerName <name>
```

サーバの名前を指定します。サーバがクライアントに返すエラーページに表示されます。デフォルトではコメントアウトされていますが、この場合、リクエストを受けつけたIPアドレスから名前解決を行って得られるホスト名が使用されます。

(4) サーバ名の指定

```
UseCanonicalName On | Off
```

サーバが公開するURL、サーバ名、サーバのポート番号を設定します。“On”が指定されているとき、ServerNameディレクティブで設定されている値を利用します。“Off”が利用されているとき、クライアントから送られたホスト名とポート番号が利用されます。

(5) ドキュメントルートの指定

```
DocumentRoot <directory>
```

サーバが公開するHTMLドキュメントのルートになるディレクトリを指定します。このディレクトリは、クライアントがサーバ名だけを指定してきた場合にアクセスされるディレクトリです。
※<directory>の末尾には、/(スラッシュ)を付けないようにして下さい。

Section2: ‘Main’ server configuration

/etc/httpd/conf/httpd.conf (ディレクトリアクセスに関する設定)

```
<IfModule mod_userdir.c>
UserDir disable                    ---(1)
#UserDir public_html
</IfModule>
DirectoryIndex index.html index.html.var        ---(2)
```

(1) ユーザが公開するディレクトリの指定

```
UserDir <directory>
```

http://server/~user/fileという形式でオブジェクトが要求されたときに実際に使用するサーバ上のディレクトリを指定します。<directory>には次のパターンを指定できます。デフォルトではdisabledです。

パターン	公開されるディレクトリ	パターン例	パターン例
disabled	ユーザによる公開を禁止する。	disabled	–
public_html	/home/user/public/html以下が公開される。	public_html	/home/user/public_html/file
<absolute path>	<absolute path>/user以下が公開される。	/usr/home	/usr/home/user/file
<absolute path>*	<absolute path>中の*をユーザ名に置き換えたディレクトリ以下が公開される。	/usr/*/home	/usr/user/home/file

(2) インデックスファイルの指定

```
DirectoryIndex <file> …
```

ディレクトリ名のみでオブジェクトをリクエストしてきた場合に、デフォルトで返すべきファイルの名前を指定します。スペースで区切って複数指定ができますが、左側にあるファイルから順番に検索されていき、最初に見つかったファイルがクライアントに返されます。

この機能により、URLでファイル名を指定せずにサーバ名だけやサーバ名とディレクトリ名だけでアクセスしても、コンテンツが返されます。

Section2: ‘Main’ server configuration

/etc/httpd/conf/httpd.conf (ログの作成に関する設定)

```
HostnameLookups Off                    ---(1)
ErrorLog logs/error_log                ---(2)
LogLevel warn
LogFormat "%h %l %u %t ¥"%r¥" %>s %b ¥"%{Referer}i¥" ¥"%{User-Agent}i¥"" combined
                ---(3)
LogFormat "%h %l %u %t ¥"%r¥" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
# CustomLog logs/access_log common
CustomLog logs/access_log combined
#CustomLog logs/referer_log referer
#CustomLog logs/agent_log agent
#CustomLog logs/access_log combined
```

(1) ホスト名解決の指定

```
HostnameLookups On | Off | Double
```

サーバにアクセスしたクライアントのホスト名をログに記録する際に、DNS逆引きを有効にするか指定します。Onを指定すると逆引きが行われ、Offを指定すると行われません。Doubleを指定すると、いったん逆引きを行ってホスト名を求めた後、そのホスト名で正引きを行ってIPアドレスを求めます。得られるIPアドレスが、最初に与えられたクライアントのIPアドレスと一致しない場合、そのホストはPARANOID（身元を詐称している）とみなされます。

デフォルトではOffです。OnにしてDNSの逆引きを有効にしておくと、DNSサーバの負荷の増加と、ネットワークのトラフィック増大を招き、Webサーバのレスポンス低下を招くので注意が必要です。

Section2: ‘Main’ server configuration

(2) エラーログに関する設定

```
ErrorLog <filename>|syslog[:<facility>]
LogLevel <level>
```

ErrorLogは、エラーログの出力先を指定します。<filename>でファイル名を指定すれば、そのファイルに出力されます。Syslogと指定しsyslogファシリティを指定すれば、そのファシリティでsyslogへ出力されます。

Loglevelは、エラーログに出力すべきメッセージのレベルを指定します。<level>にはレベルを表す文字列を指定します（次表を参照）。syslogのレベルと似ていますが、あくまでもApache独自のものです。

レベル	意味
emerg	緊急事態、システムは使用不能
alert	警告、対応を直ちにする必要あり
crit	致命的な状態
error	エラー
warn	警告
notice	重要な通知
info	情報
debug	デバッグ情報

(3) カスタムログに関する設定

```
LogFormat <format> [<nickname>]
CustomLog <filename> <nickname>|<format>
```

LogFormatは、CustomLogまたはTransferLogによる出力のための書式（ログフォーマット）を定義します。<format>で指定したログフォーマットを、<nickname>という名前に割り当てるか、またはデフォルトのログフォーマットとして定義します。

Red Hat系Linuxでじゃ、combined, common, referer, agentという4つの<nickname>を利用してログフォーマットを定義しています。それぞれ、複合型、基本型、REFERERのみ、AGENTのみの記録を定義しています。

CustomLogは、デフォルトのログフォーマットかLogFormatによって作成されたログフォーマットを実際のファイルに割り当てます。または、ここで直接ログフォーマットを指定することもできます。

Section2: ‘Main’ server configuration

/etc/httpd/conf/httpd.conf (エイリアスの設定)

```
Alias /icons/ "/var/www/icons/"       ---(1)
<Directory "/var/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"---(2)
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

(1) エイリアスの指定

```
Alias <alias> <directory>|<file>
```

Aliasは、実際のデイレクトリ名に対し別名（エイリアス）を定義します。<directory>または<file>で指定されるディレクトリ・ファイルに、<alias>という別名を定義します。リクエストされるファイルにこのエイリアスが含まれている場合、実在するローカルなディレクトリに置き換えられます。DocumentRoot以下にないディレクトリでも、エイリアスを定義すればクライアントからアクセスすることが可能になります。

※<alias>の最後に/（スラッシュ）を付けた場合は、<directory>の最後にも必ず/を付ける必要があります。

(2) スクリプトエイリアスの指定

```
ScriptAlias <alias> <directory>|<file>
```

ScriptAliasはAliasと同様に、実際のデイレクトリ名に対し別名（エイリアス）を定義します。ただし、ここで設定されたディレクトリはCGIプログラムのディレクトリとして扱われます。

Section2: ‘Main’ server configuration

/etc/httpd/conf/httpd.conf (アクセス制御に関する設定)

```
<Directory />                                   ---(1)
    Options FollowSymLinks                      ---(2)
    AllowOverride None              ---(3)
</Directory>
<Directory "/var/www/html“>
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny                ---(6)
    Allow from all                  ---(7)
</Directory>
AccessFileName .htaccess                        ---(4)
<Files ~ "^¥.ht">               ---(5)
    Order allow,deny
    Deny from all
</Files>
#<Location /server-status>                      ---(8)
#    SetHandler server-status
#    Order deny,allow
#    Deny from all
#    Allow from .example.com
#</Location>
```

Section2: ‘Main’ server configuration

(1) ディレクトリ単位のアクセス制御

```
<Directory <directory>>
…
</Directory>
```

<Directory>は、<directory>で指定されるディレクトリ以下のアクセスコントロールを行います。<directory>以下のすべてのディレクトリに対して有効になります。ただし、<directory>以下のあるディレクトリに対して別の指定がされた場合には、その指定でそれ以下のディレクトリの設定が置き換えられます。
※“/”すなわちルートディレクトリに対する指定は、ローカルなファイルシステム全体に対するものです。ServerRootやDocumentRootによるディレクトリの指定とは関係ありません。

(2) 制御オプションの追加・変更・削除

```
Options [+|-]<option> …
```

Optionsは、制御オプションと呼ばれるパラメータをディレクトリに対して設定します（次表参照）。

オプション	意味
None	制御オプションをすべて無効にする。
All	制御オプションをすべて有効にする。
ExecCGI	CGIプログラムの実行を許可する。
Includes	SSIを許可する。
IncludesNOEXEC	SSIを許可するが、#exec, #cmd, #includeによるプログラムの実行は禁止する。
Indexes	ディレクトリインデックスの作成を許可する。
MultiViews	Content negotiated MultiViewsを許可する。
FollowSymlinks	ディレクトリにシンボリックリンクがあるとき、それをたどることを許可する。
SymLinksOwnerMatch	シンボリックリンクとリンク先が同じ所有者である場合のみ、それをたどることを許可する。

<option>の前に“+”を付けると、そのディレクトリで新たに指定した<option>が追加されます。“-“を付けると、そのディレクトリで新たに指定した<option>が削除されます。どちらも付けなければ、それより上位のディレクトリにおける制御オプションの指定が、すべてこのディレクトリのものに置き換えられます。

Section2: ‘Main’ server configuration

(3) アクセスコントロールファイルによるオーバライドの許可

AllowOverride <override>…

AllowOverrideは、アクセスコントロールファイルによるディレクトリごとの設定の変更をどこまで許可するか指定します（次表参照）。

指定子	意味
None	あらゆるオーバライドを無効にする（アクセスコントロールファイルを無視する）。
All	アクセスコントロールファイルのあらゆるディレクティブを有効にする。
AuthConfig	認証に関するディレクトィブを有効にする。対応するディレクティブは、AuthDBMUserFile, AuthDBMGroupFile, AuthGroupFile, AuthName, AuthUserFile, AuthTypeの6つである。
FileInfo	ドキュメントタイプを指定するディレクティブを有効にする。対応するディレクティブは、AddEncoding, AddLanguage, AddType, DefaultType, LanguagePriorityの5つである。
Indexes	ディレクトリインデックスを制御するディレクティブを有効にする。対応するディレクティブは、AddDescription, AddIcon, AddIconByEncoding, AddIconByType, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions, ReaderNameの12個である。
Limit	HTTPサーバへのアクセス制御を行うディレクティブを有効にする。対応するディレクティブは、allow, deny, orderの3つである。
Options	アクセスコントロールファイルが存在するディレクトリを制御するディレクティブを有効にする。対応するディレクティブは、Options, XBitHackの2つである。

(4) アクセスコントロールファイルの指定

AccessFileName <file>

アクセスコントロールファイルの名前を指定します。アクセスコントロールファイルは、ディレクトリ側で独自にアクセス制御を行うためのディレクティブを記述したファイルで、一般的にはユーザのディレクトリに置かれます。

Section2: ‘Main’ server configuration

(5) ファイル単位のアクセス制御

```
<Files <file>>
…
</Files>
```

<Files>は、<file>で指定されるファイル単独のアクセス制御を行います。<file>にはファイル名のパターンを指定します。ファイル名には、?と*を含むことができます。もし正規表現を使用する場合には、<file>の前に、~（チルダ）を置き、~（チルダ）とパターンの間には空白を置きます。

(6) 評価順の指定

```
Order <ordering>
```

Orderは、通常<Directory>ディレクティブや<Files>ディレクティブの内部に置かれ、Allowディレクティブと Denyディレクティブの評価順を指定します。

評価順	意味
deny, allow	まずDenyの指定が適用され、次にAllowの指定が適用される。デフォルト。
allow, deny	まずAllowの指定が適用され、次にDenyの指定が適用される。デフォルト。
mutual-failure	Allowの指定のうち、Denyにないものだけが適用される。

Section2: ‘Main’ server configuration

(7) アクセス許可リスト・アクセス拒否リストの指定

```
Allow from <host> …
Deny from <host> …
```

Allowはアクセスを許可するホストを、Denyはアクセスを拒否するホストを指定します（次表参照）。通常<Directory>ディレクティブや<Files>ディレクティブの内部に置かれ、Orderディレクティブと同時に指定されます。

指定方法	指定	例
all	すべてのホスト。	ー
ドメイン名	指定するドメインに属するホストすべて。	test.com
IPアドレス	指定するIPアドレスを持つホストすべて。 IPアドレスの共通部分以外は省略可能。	192.168.1
ネットワークアドレス/ネットマスク	指定するネットワークに属するホストすべて。	192.168.1.0/255.255.255.0
ネットワークアドレス/ビット数	指定するネットワークに属するホストすべて。	192.168.1.0/24

(8) URLによるアクセス制御

```
<Location <URL>>
…
</Location>
```

<Files>は、URLに対するアクセス制御を行います。<Directory>がローカルなディレクトリに対しアクセス制御を行うものであるのに対し、<Location>はURLに対して機能します。つまり、Alias等によって定義された別名に対して、アクセス制御ができます。

Section2: ‘Main’ server configuration

/etc/httpd/conf/httpd.conf (その他の設定)

```
ServerSignature On                          ---(1)
#ErrorDocument 500 "The server made a boo boo."          ---(2)
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
```

(1) Apacheが生成するドキュメント中のサーバ情報の指定

```
ServerSignature On | Off | EMail
```

ServerSignature ディレクティブは、サーバが生成するドキュメント (エラーメッセージ、mod_proxy における FTP のディレクトリリスト、mod_info の出力など) の最下行に付与するフッタの設定を行ないます。そのような、フッタ行を有効にしたい理由としては、プロキシが複数連なっている場合に、ユーザはどのサーバが返した エラーメッセージかを知る手段がほとんど無いからです。

Off に設定をすると、エラーの際の行が抑制されます。(そして、Apache-1.2 以前と互換の動作をします) On に設定した場合は、単にドキュメントの中に、サーバのバージョン、稼動中のバーチャルホストの ServerName の書かれた行を追加し、EMail にした場合はさらに参照されたドキュメントに対する ServerAdmin を指す "mailto:" が追加されます。

(2) ErrorDocumentの指定

```
ErrorDocument <error code> <filename>|<plain text>|<URL>
```

Apache1.3まではエラーページの内容を設定することはできませんでしたが、Apache2.0ではエラーごとに出力するHTMLファイルを指定することができます。

<error code>にはエラーコードを指定します。<filename>にはエラーごとに出力するHTMLファイル名、またはテキスト文、またはURLを指定します。

More

- More
  - ユーザ認証
  - リダイレクト
  - バーチャルホスティング
  - SSL

ユーザ認証

パスワードファイル

httpdにおけるユーザ認証のためには、ページへのアクセスを許可するユーザの名前とパスワードのリストを記録したパスワードファイルが必要です。

ここで必要なパスワードファイルは/etc/passwdとは無関係です。パスワードファイルを管理するプログラムhtpasswdを使用してパスワードファイルを作成します。

```
htpasswd [<options>] <filename> <username>
```

<filename>には、パスワードファイルの名前を指定します。名前は任意で良いですが、.htpasswdなどの隠しファイルとすることが一般的です。<username>には、追加あるいは更新するユーザの名前を指定します。パスワードファイルに登録されていない場合は追加、登録されている場合は更新となります。

<filename>は存在するファイルを指定しますが、存在しない場合には、-cオプションを指定して新規作成します。

```
# htpasswd -c /etc/httpd/.htpasswd user01
New password:
Re-type new password:
Adding password for user user01

# cat /etc/httpd/.htpasswd
user01:grlvvq5T4r5BQ
```

暗号化方式はデフォルトでMD5による暗号化（-mオプション）です。他に、DESによる暗号化（-dオプション）、SHA(Secure Hash Algorithm)ハッシュ関数による暗号化（-sオプション）、プレーンテキスト（-pオプション）があります。

ユーザ認証

■ アクセスコントロールファイル

認証が必要なオブジェクトを含むディレクトリにアクセスコントロールファイルを置き、それとパスワードファイルを結びつけます。

なお、認証の機能を使うためにはmod_authモジュールが必要です。また、Digest認証を行う場合には、mod_auth_digestが必要です。デフォルトではこれらが読み込まれる設定になっています。

```
AuthType Basic                          ---(1)
AuthUserFile /etc/httpd/.htpasswd       ---(2)
AuthGroupFile /dev/null                 ---(3)
AuthName “Please Login”                 ---(4)
<Limit GET>
require valid-user                      ---(5)
</Limit>
```

(1) 認証タイプの指定

認証タイプを指定します。BasicかDigestのいずれかを指定します。通常はBasicを使用します。DigestのMD5暗号化は、128ビットの暗号化強度を持っていてBasicより強力ですが、対応ブラウザが少ないという欠点があります。

(2) パスワードファイルの指定

パスワードファイルを指定します。

(3) グループファイルの指定

グループファイルを指定します。グループによる認証をしない場合は、/dev/nullとしておきます。

(4) メッセージの指定

ブラウザのダイアログボックスに表示するメッセージを指定します。

(5) アクセス制御の指定

```
require <entity-type> [<entity> …]
```

<entity-type>には、user, group, value-userのいずれかを指定します。valid-userを指定すると、パスワードファイルで認証できるユーザのみ許可します。userとgroupを指定した場合には、それぞれAuthUserFile, AuthGroupFileで指定されるファイルに登録されてる必要があります。

ユーザ認証

ユーザのディレクトリで認証

デフォルト設定では、ユーザのホームディレクトリに関して非常にきつい制限が施されています。ユーザーに対して個別の認証を許可するならば、httpd.conf中の該当箇所をコメンアウトします。
OptionsディレクティブのAuthConfigが指定されていれば、認証機能が有効となります。

/etc/httpd/conf/httpd.conf

```
<Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>
```

リダイレクト

■ リダイレクト

あるURLに対して行われたリクエストを、別のURLに転送する方法。
・CGI, PHP等でレスポンスヘッダにLocationヘッダを出力
・HTMLファイルに<meta http-equiv="Refresh">タグを記述
・ApacheでRedirectディレクティブを使用

```
Redirect [<status>] <old URL> <new URL>
```

<oldURL>に古いURLを、<new URL>に新しいURLを指定します。<status>に指定することができるパラメータは次表を参照してください。

パラメータ	機能
nokeepalive	Keep-Aliveを無効にする。
force-response-1.0	HTTP/1.0と応答させる。
downgrade1.0	HTTP/1.0としての要求を強制する。
permanent	リソースが永遠に移動したことを意味する。
temp	リソースが一時的に移動したことを意味する。何も指定しない場合のデフォルトパラメータ。
seeother	リソースが他のもので置き換えられたことを意味する。
gone	リソースが永遠に削除されたことを意味する。この場合<new URL>は省略する。

ただし、Redirectによる指定では、httpdがリクエストを実際に転送するわけではなく、レスポンスヘッダにLocationヘッダを返しているだけです。クライアントのブラウザがこれを見て新しいURLにリクエストを出します。
サーバが実際にリクエストを転送するようにするには、mod_rewriteモジュールを使用してURLの書き換えを行います。

バーチャルホスティング

- バーチャルホスティング
  - 1台のWebサーバで複数のドメインを運用。
  - 共有ホスティングでよく使用される。
- IPアドレスおよびポート番号によるバーチャルホスティング

/etc/httpd/conf/httpd.conf (Virtual Hosts)

```
### Section 3: Virtual Hosts
#<VirtualHost *>                                    ---(1)
#    ServerAdmin webmaster@dummy-host.example.com
#    DocumentRoot /www/docs/dummy-host.example.com
#    ServerName dummy-host.example.com
#    ErrorLog logs/dummy-host.example.com-error_log
#    CustomLog logs/dummy-host.example.com-access_log common
#</VirtualHost>
```

(1) バーチャルホストの指定

```
<VirtualHost <address>:<port>>
…
</VirtualHost>
```

<VirtualHost>ディレクティブを使用し、それぞれのサーバごとに、DocumentRootなどの設定を行います。

<address>の部分を“_default_”と指定すれば、他の<VirtualHost>で指定されていないIPアドレスをデフォルトの指定とすることができます。同様に、<port> に“*”を指定すれば、他にマッチしないポート番号をデフォルトの指定とすることができます。

IPアドレスとポート番号を指定しなかった場合は、デフォルトの設定として、<VirtualHost>ディレクティブの外に書かれた標準の設定が使用されます。

バーチャルホスティング

- ホスト名によるバーチャルホスティング

/etc/httpd/conf/httpd.conf (Virtual Hosts)

```
### Section 3: Virtual Hosts
#NameVirtualHost *:80                    ---(1)
#<VirtualHost *>                         ---(2)
#    ServerAdmin webmaster@dummy-host.example.com
#    DocumentRoot /www/docs/dummy-host.example.com
#    ServerName dummy-host.example.com
#    ErrorLog logs/dummy-host.example.com-error_log
#    CustomLog logs/dummy-host.example.com-access_log common
#</VirtualHost>
```

(1) 使用するアドレスとポートの指定

```
NameVirtualHost <address>:<port>
```

ホスト名によるバーチャルホスティングを行う場合、使用するIPアドレスとポートを指定します。

(2) バーチャルホストの指定

```
<VirtualHost <host-name>>
…
</VirtualHost>
```

<VirtualHost>ディレクティブを使用し、それぞれのサーバごとに、DocumentRootなどの設定を行います。<VirtualHost>内に書かれたServerNameディレクティブの設定によって、サーバ名の判断が行われます。

<host-name>の部分を“_default_”と指定すれば、他の<VirtualHost>の指定にマッチしなかった場合のデフォルトの指定とすることができます。

SSL

- SSLプロトコル
  - 暗号化通信プロトコル。
  - TCP/IPのトランスポート層とアプリケーション層の間で動作する。
- CA

SSLによる暗号化通信は公開鍵暗号方式を利用して行われます。
ただし、公開鍵暗号方式は、共通鍵暗号方式に比べて暗号化、復号化の時間が著しく長いという欠点があるため、SSLでは最初に公開鍵暗号方式を利用して暗号化した共通鍵を送受信し、その後の暗号化に共通鍵暗号方式を利用しています。
公開鍵暗号方式を利用した共通鍵の交換を行うとき、相手が利用している公開鍵が接続しようとしているWebサーバの本物の鍵であるかを確認する必要があります。相手のWebサーバの身元と、公開鍵が本物であるかの確認をするために利用されているのが認証局（CA; Certificate Authority）です。
Webサーバを運営している組織が、認証局から証明書を購入してWebサーバにインストールすることで対応します。

- ApacheでのSSLの利用

Apache1.3系では、SSLを利用するためにはApache本体にパッチを当てる必要がありましたが、Apache2.0系では、Apache本体に手を加えなくても、モジュール mod_ssl を利用すればSSLを利用することができます。また、SSLの設定は ssl.conf に記述され、httpd.conf を修正しなくてもSSLの設定を行うことができます。

SSLを利用するためには、WWWサーバの公開鍵と証明書が必要です。Red HatのApache2.0系では、これらを用意すれば、特に他の設定は不要でSSLを利用できます。ここでは自己証明書を作成して利用します。

SSLー自己証明書の作成

1. サーバの鍵の作成

すでに鍵がある場合には削除します。(server.crtファイルがあると、同一ファイル名では、証明書の作成ができません。)

```
# cd /etc/httpd/conf
# rm ssl.key/server.key
# rm ssl.crt/server.crt
```

鍵を作成し保存します。

```
# /usr/bin/openssl genrsa 1024 > /etc/httpd/conf/ssl.key/server.key
Generating RSA private key, 1024 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
```

サーバの鍵が /etc/httpd/conf/ssl.key/server.key という名前で作成されます。サーバの鍵をrootユーザ以外が読むことができないようにします。

```
# chmod 400 ssl.key/server.key
```

SSLー自己証明書の作成

2. サーバの証明書の作成

```
# make testcert
umask 77 ; ¥
/usr/bin/openssl req -new -key /etc/httpd/conf/ssl.key/server.key -x509 -days 365 -out
/etc/httpd/conf/ssl.crt/server.crt
You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:JP
State or Province Name (full name) [Berkshire]:Tokyo
Locality Name (eg, city) [Newbury]:Shinjuku-ku
Organization Name (eg, company) [My Company Ltd]:TestNet
Organizational Unit Name (eg, section) []:Server-Team
Common Name (eg, your name or your server's hostname) []:www.test.net
Email Address []:webmaster@test.net
```

サーバの証明書が /etc/httpd/conf/ssl.crt/server.crt という名前で作成されます。有効期間はデフォルトで1年間です。

起動

```
# service httpd start
```

SSLーCA証明書の利用

1. サーバの鍵の作成

自己証明書の場合と同様です。

2. CSRファイルの作成

```
make certreq
umask 77 ; ¥
/usr/bin/openssl req -new -key /etc/httpd/conf/ssl.key/server.key -out /etc/httpd/conf/ssl.csr/server.csr
You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:JP
State or Province Name (full name) [Berkshire]:Tokyo
Locality Name (eg, city) [Newbury]:Shinjuku-ku
Organization Name (eg, company) [My Company Ltd]:TestNet
Organizational Unit Name (eg, section) []:Server-TEAM
Common Name (eg, your name or your server's hostname) []:www.test.net
Email Address []:webmaster@test.net

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: [そのままリターン]
An optional company name []: [そのままリターン]
```

サーバの証明書が /etc/httpd/conf/ssl.csr/server.csr という名前で作成されます。

ファイルの中身は暗号化されたテキストファイルですので、認証局のウェブページの指定に従って、カット＆ペーストで貼り付けて証明書を申請します。

SSLーCA証明書の利用

3. CRTファイルの適用

認証局より電子メール等で証明書が送付されてきますので、そこに記述された、-----BEGIN CERTIFICATE-----から、-----END CERTIFICATE-----までの部分をカット&ペーストで、/etc/httpd/conf/ssl.crt/server.crt ファイルに貼り付けます。

/etc/httpd/conf/ssl.csr/server.csr

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

起動

```
# service httpd start
```

SSLーCSRに自己署名

- CSRに自己署名

通常は行うことはないと思いますが、すでに作成済みのCSRに自己署名する方法です。

```
# openssl x509 -in ssl.csr/server.csr -out ssl.crt/server.crt -req -signkey ssl.key/server.key -days 365
Signature ok
subject=/C=JP/ST=Tokyo/L=Shinjuku-ku/O=TestNet/OU=Server-
TEAM/CN=www.test.net/emailAddress=webmaster@test.net
Getting Private key
```

ssl.conf

/etc/httpd/conf.d/ssl.conf (SSLバーチャルサーバ)

```
<VirtualHost _default_:443>                    ---(1)
# DocumentRoot "/var/www/html"
# ServerAdmin you@your.address
# ServerName new.host.name:443
ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
SSLEngine on                                   ---(2)
SSLCertificateFile /etc/httpd/conf/ssl.crt/server.crt
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/server.key
</VirtualHost>
```

(1) SSLバーチャルホストの設定

```
<VirtualHost _default_:443>
…
</VirtualHost>
```

SSLは443番ポートで待ち受けているバーチャルサーバという形で実現されていますので、SSLバーチャルサーバに関する設定をします。<VirtualHost>ディレクティブを使用し、DocumentRootなどの設定を行います。デフォルトですでにある程度の設定がされています。

(2) SSLの有効化

```
SSLEngine on | off
```

onを指定するとSSLを有効化します。offを指定するとSSLを無効化します。

Tips

- Tips
  - サーバ情報
  - ログ

サーバ情報取得

/etc/httpd/conf/httpd.conf (サーバ情報)

```
<Location /server-status>                ---(1)
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from 192.168.1
</Location>
<Location /server-info>                  ---(2)
    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from 192.168.1
</Location>
```

(1) サーバ状態情報の取得

サーバの状態に関する情報を取得します。次のURLで閲覧することができます。管理者しか見れないようにIPアドレスまたはホスト名でアクセス制限するのが望ましいです。

http://servername/server-status

(2) サーバ設定情報の取得

サーバの設定に関する情報を取得します。次のURLで閲覧することができます。管理者しか見れないようにIPアドレスまたはホスト名でアクセス制限するのが望ましいです。

http://servername/server-status