Apache MPM 共通ディレクティブ

説明:	二つ以上のマルチプロセッシングモジュール (MPM) で実装されているディレクティブのコレクション
ステータス:	MPM

トピック

ディレクティブ

CoreDumpDirectory ディレクティブ

説明:	Apache がコアダンプする前に移動を試みるディレクトリ
構文:	CoreDumpDirectory directory
デフォルト:	デフォルトの設定は説明文を読んでください
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, perchild, prefork, mpm_winnt

Apache がコアダンプする前に移動を試みるディレクトリを制御します。 デフォルト値は ServerRoot ディレクトリですが、このディレクトリはサーバの実行されているユーザ権限で書き込み可能であるべきではないので、通常はコアダンプは書き込まれません。 デバッグのためにコアダンプが必要であれば、 このディレクティブを使って他の位置にコアダンプを書き出すようにできます。

Group ディレクティブ

説明:	リクエストに応答する際に所属するグループ
構文:	Group unix-group
デフォルト:	Group #-1
コンテキスト:	サーバ設定ファイル, バーチャルホスト
ステータス:	MPM
モジュール:	worker, perchild, prefork

Group ディレクティブで、 リクエストに応答する際に所属しておくグループを設定します。 このディレクティブを使用するためには、スタンドアロン型の サーバは最初に root 権限で起動されている必要があります。 unix-group は次のうちのいずれかです:

グループ名
: グループを名前で参照します。

に続いてグループ番号
: グループを番号で参照します。

サーバを実行するために特定の新しいグループを設定することを お薦めします。nobody を使用する管理者もいますが、 これは常に可能であったり望ましい訳ではありません。

```
Group www-group
```

注意: サーバ開始を非 root ユーザで行った場合は、 指定されたグループに変わることができなくて、代わりに 起動を行ったユーザの所属するグループとして実行され続けます。

特記事項: このディレクティブを <VirtualHost> で使用することはサポートされなくなりました。Apache 2.0 で suEXEC ラッパー [1]を実現したい場合は、 SuexecUserGroup を使用してください。 セキュリティ: セキュリティ上の注意点に関しては、 User をご覧下さい。

Listen ディレクティブ

説明:	サーバが listen するIP アドレスとポート番号
構文:	Listen [IP-address:]portnumber
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, perchild, prefork, mpm_winnt

Listen ディレクティブは Apache が特定の IP アドレスやポート番号だけを listen するように指定します。 デフォルトでは全ての IP インターフェースのリクエストに応答します。 Listen ディレクティブは現在では必須のディレクティブとなりました。 もし設定ファイルになければ、サーバは起動に失敗します。 これは以前のバージョンの Apache から変更のあった部分です。

Listen ディレクティブでは、特定のポートあるいは アドレスとポートの組み合わせから入ってくるリクエストに対して 応答するように指定します。 もしポート番号だけが指定された場合は、サーバは全インターフェースの 指定されたポート番号に対して listen します。 IP アドレスがポートとともに指定された場合は、 サーバは指定されたポートとインターフェースに対して listen します。

複数のアドレスとポートに対して listen するように、 複数の Listen ディレクティブを使うこともできます。 サーバは列挙されたアドレスとポート全てからのリクエストに対して 応答します。

例えば、サーバが 80 番ポートと 8000 番ポートの両方の コネクションを受け入れる場合は、次のようにします。

```
Listen 80
Listen 8000
```

二つの特定のインターフェースとポート番号からのコネクションを 受け入れるようにするには、次のようにします。

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

IPv6 アドレスは角括弧で囲まなければなりません。 例えば次の例のようにです。

```
Listen [fe80::a00:20ff:fea7:ccea]:80
```

参照

- DNS の問題 [3]
- Apache が使用するアドレスとポートの設定 [4]

ListenBackLog ディレクティブ

説明:	保留状態のコネクションのキューの最大長
構文:	ListenBacklog backlog
デフォルト:	ListenBacklog 511
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, perchild, prefork, mpm_winnt

保留状態のコネクションのキューの最大長です。 一般的には調整する必要はありませんし、調整は望ましくありません。 しかし、TCP SYN フラッドアタックの状況下におかれる場合に、 増やした方が望ましいシステムもあります。 listen(2) システムコールのバックログパラメータを ご覧下さい。

この値は OS により、小さな数に抑えられます。 値は OS 毎に異なっています。また多くの OS では、 バックログとして指定されている値ちょうどまで使っているわけではなく、設定されている値に基づいて（通常は設定値よりも大きな値を）使っていることに注意してください。

LockFile ディレクティブ

説明:	受付を直列化するためのロックファイルの位置
構文:	LockFile filename
デフォルト:	LockFile logs/accept.lock
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, perchild, prefork

USE_FCNTL_SERIALIZED_ACCEPT または USE_FLOCK_SERIALIZED_ACCEPT のいずれかを使って Apache をコンパイルした際に使用することになる、 ロックファイルのパスを LockFile は設定します。 このディレクティブは通常はデフォルト値のままにしておくべきです。 これを変更する際の主な理由は、logs ディレクトリが NFS マウントされたものであるという理由です。 ロックファイルはローカルディスクに保存しなければならない からです。 メインサーバプロセスの PID が自動的にファイル名に付加されます。

セキュリティ:

/var/tmp といった、誰でも書き込めるディレクトリにファイルを 置かない方がよいです。なぜなら、サーバが起動時に作成する ロックファイルの作成自体を妨害することによって、 誰でもサービス拒否アタックを引き起こすことができるからです。

MaxClients ディレクティブ

説明:	リクエストに応答するために作成される 子プロセスの最大個数
構文:	MaxClients number
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, prefork

MaxClients ディレクティブは、 応答することのできる同時リクエスト数を設定します。 MaxClients 制限数を越えるコネクションは通常、 ListenBacklog ディレクティブで設定した数までキューに入ります。 他のリクエストの最後まで達して子プロセスが空くと、 次のコネクションに応答します。

スレッドを用いないサーバ (すなわち prefork) では、MaxClients は、リクエストに応答するために起動される 子プロセスの最大数となります。 デフォルト値は 256 で、これを増加させたい場合は、 ServerLimit の値も増加させる必要があります。

スレッドを用いるサーバ (すなわち worker) では、MaxClients は、クライアントに応答できるスレッドの総数を制限します。 デフォルト値は ThreadsPerChild を 16 倍した値になります。MaxClients を 16 プロセス以上必要な値まで増加させたい場合は、 ServerLimit も増加させる必要があります。

MaxMemFree ディレクティブ

説明:	主メモリアロケータが free() を呼ばずに保持し続けられるメモリの 最大量
構文:	MaxMemFree number
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, prefork, mpm_netware

MaxMemFree ディレクティブは free() を呼ばずに 主アロケータが保持できる空のメモリの最大値をキロバイト単位で設定します。 設定されていないか、零に設定されているときは、無制限になります。

MaxRequestsPerChild ディレクティブ

説明:	個々の子サーバが稼働中に扱うリクエスト数の上限
構文:	MaxRequestsPerChild number
デフォルト:	MaxRequestsPerChild 10000
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, perchild, prefork, mpm_winnt

MaxRequestsPerChild ディレクティブは、 個々の子サーバプロセスが扱うことのできるリクエストの制限数を 設定します。MaxRequestsPerChild 個のリクエストの後に、子プロセスは終了します。 MaxRequestsPerChild が 0 に設定されている場合は、プロセスは期限切れにより終了することはありません。

MaxRequestsPerChild を非ゼロに制限することには、二つの利点があります:

- (偶発的な) メモリーリークが起こった場合に プロセスが消費するメモリの総量を制限できる
- プロセスに有限のライフタイムを設定することで、 サーバ負荷が下がった時にプロセス数を少なくすることができる

> 注意:
> KeepAlive リクエストの場合は、 一つ目のリクエストだけがこの制限に該当します。実効的には、一つの子プロセスあたりのコネクション数を 制限するように挙動が変化します。

MaxSpareThreads ディレクティブ

説明:	アイドルスレッドの最大数
構文:	MaxSpareThreads number
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	mpm_netware, perchild, worker

アイドルなスレッドの最大数です。異なる MPM ではそれぞれ、 このディレクティブは異なる取り扱われ方をされます。

perchild では、 デフォルトは MaxSpareThreads 10 です。 この MPM はアイドルスレッド数を、それぞれの子プロセスごとに監視します。 子プロセスにアイドルスレッドが多すぎる場合は、 サーバはその子プロセスに含まれるスレッドを終了し始めます。

worker では、 デフォルトは MaxSpareThreads 500 です。 この MPM はアイドルスレッド数をサーバ全体で監視します。 サーバでアイドルスレッド数が多すぎる場合は、 この数字よりも少ない数になるまで子プロセスを終了します。

mpm_netware では、 デフォルトは MaxSpareThreads 100 です。 この MPM はシングルプロセスで実行されますので、 スペアスレッド数もサーバ全体で勘定します。

参照

- MinSpareThreads
- StartServers

MaxThreadsPerChild ディレクティブ

説明:	子プロセス毎のスレッド数の最大数
構文:	MaxThreadsPerChild number
デフォルト:	MaxThreadsPerChild 64
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, perchild

子プロセス毎に含まれるスレッド数の最大値です。 子プロセス毎にスレッド数変化する MPM では、 このディレクティブは、子プロセス内に生成されるスレッド数の 最大値を設定します。デフォルト値よりも大きい値にするのであれば、 コンパイル時に定義された HARD_THREAD_LIMIT を変更して、サーバを再コンパイルする必要があります。

MinSpareThreads ディレクティブ

説明:	リクエストに応答することのできる アイドルスレッド数の最小数
構文:	MinSpareServers number
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	mpm_netware, perchild, worker

リクエストに応答するスレッド数の最小値です。 異なる MPM ではそれぞれ、 このディレクティブは異なる取り扱われ方をします。

perchild では、 デフォルトは MinSpareThreads 5 で、 アイドルスレッド数を子プロセス毎に監視します。 もし子プロセスに十分な数のスレッドがなければ、 サーバはその子プロセスに新しいスレッドを作り始めます。

worker では、 デフォルトは MinSpareThreads 250 で、 アイドルスレッド数をサーバ全体で監視します。 もしサーバに十分な数のアイドルスレッドがなければ、 アイドルスレッド数がこの数よりも大きくなるまで 新しい子プロセスが生成されます。

mpm_netware では、 デフォルトは MinSpareThreads 10 で、 シングルプロセス MPM ですので、サーバ全体で管理されます。

参照

- MaxSpareThreads
- StartServers

NumServers ディレクティブ

説明:	同時に起動している子プロセスの総数
構文:	NumServers number
デフォルト:	NumServers 2
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	perchild

同時に起動している子プロセスの数です。 このディレクティブを使用する MPM は動的に新しい子プロセスを 生成することは行わないので、サイト全体に来るリクエスト全てを 十分扱える程度に大きな数に設定しておく必要があります。

PidFile ディレクティブ

説明:	デーモンのプロセス ID をサーバが記録するためのファイル
構文:	PidFile filename
デフォルト:	PidFile logs/httpd.pid
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, perchild, prefork, mpm_winnt

PidFile ディレクティブで、 デーモンのプロセス ID をサーバが記録するファイルを設定します。 もしファイル名がスラッシュ (/) で始まらない場合は、 ServerRoot からの相対的なものとして扱われます。

例
```
PidFile /var/run/apache.pid
```

サーバが ErrorLog や TransferLog を閉じて開き直したり、設定ファイルを 再読込したりさせるために、サーバにシグナルを送ることができると 便利なことがあります。 これは SIGHUP (kill -1) シグナルを PidFile に書かれているプロセス ID に送ることでできます。

PidFile には、ログファイルの設置位置や セキュリティ [2] と全く同じ注意点があります。

ScoreBoardFile ディレクティブ

説明:	子プロセスと連携するためのデータを保存する ファイルの位置
構文:	ScoreBoardFile file-path
デフォルト:	ScoreBoardFile logs/apache_status
コンテキスト:	サーバ設定ファイル
ステータス:	MPM

モジュール:	worker, perchild, prefork

Apache は親プロセスと子プロセス間の通信にスコアボードを用います。 この通信機能にファイルを必要とするアーキテクチャもあります。 ファイルが指定されていなければ、Apache はまずメモリ上（匿名共有メモリ）にスコアボードを作ろうとし、それが失敗すると ディスク上にファイル（ファイルベースの共有メモリ）を作ろうとします。 このディレクティブを指定すると、Apache は必ずディスクにファイルを生成します。

例
```
ScoreBoardFile /var/run/apache_status
```

ファイルベースの共有メモリは、サードパーティー製のアプリケーションで スコアボードに直接アクセスする必要がある場合に役に立ちます。

ScoreBoardFile を使う場合、 RAM ディスク上に置くとスピードが向上するでしょう。 しかし、ログファイルの設置位置や セキュリティ [5] と同様の注意点があるので、注意してください。

参照

- Apache の停止と再起動 [6]

SendBufferSize ディレクティブ

説明:	TCP バッファサイズ
構文:	SendBufferSize bytes
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, perchild, prefork, mpm_winnt

サーバは TCP バッファサイズを指定されたバイト数に設定します。 高速で高レイテンシな環境で（例 100ms 程度、大陸横断高速通信路など）古い一般的な OS のデフォルト値を増やすのに非常に便利です。

ServerLimit ディレクティブ

説明:	設定可能なサーバプロセス数の上限
構文:	ServerLimit number
デフォルト:	ServerLimit 256 (prefork), ServerLimit 16 (worker)
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, prefork

prefork MPM の場合は、このディレクティブは Apache プロセス稼働中における MaxClients に設定可能な上限値を設定することになります（訳注: prefork の場合は同時クライアント数 = サーバプロセス数なので）。 worker MPM の場合には、このディレクティブは

ThreadLimit ディレクティブと組み合わせて、 Apache プロセス稼働中における MaxClients に設定可能な上限値を設定することになります。 再起動中にこのディレクティブを変更しても無視されますが、 MaxClients は再起動中に修正することができます。

このディレクティブを使用する際は特に注意してください。 ServerLimit が必要以上に大きな値に 設定された場合は、余計な未使用共有メモリが割り当てられます。 ServerLimit と MaxClients がシステムの扱える範囲を越えた設定値になっていると、 Apache は起動しないか、起動しても不安定になるでしょう。

prefork MPM では、 MaxClients を 256 よりも大きな値に設定する必要がある時にだけ使用してください。 希望の MaxClients 数とくらべて、必要以上に大きな値を指定することは避けてください。

worker MPM では、 MaxClients と ThreadsPerChild の設定で 16 サーバプロセス以上必要になる場合にのみ使用してください。 希望の MaxClients と ThreadsPerChild とくらべて、必要となるサーバプロセス数以上に大きな値を 設定することは避けてください。

StartServers ディレクティブ

説明:	起動時に生成される子サーバプロセスの数
構文:	StartServers number
デフォルト:	StartServers 5
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker

StartServers ディレクティブは、 起動時に生成される子サーバプロセスの数を設定します。 プロセス数は負荷に応じて動的に制御されますので、 通常はこの値を調整する理由はあまりないでしょう。

参照

- MinSpareThreads
- MaxSpareThreads

StartThreads ディレクティブ

説明:	起動時に生成されるスレッドの数
構文:	StartThreads number
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	mpm_netware, perchild

起動時に生成されるスレッドの数です。 スレッド数は負荷に応じて動的に制御されますので、 通常はこの値を調整する理由はあまりないでしょう。

perchild でのデフォルトは StartThreads 5 で、このディレクティブは起動時に プロセス毎のスレッド数を追跡します。

mpm_netware でのデフォルトは StartThreads 50 で、 この場合プロセスは一つしかないので、 起動時にリクエストに応答するスレッドの総数となります。

ThreadLimit ディレクティブ

説明:	設定可能な子プロセス毎のスレッド数の上限を 設定します
構文:	ThreadLimit number
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	mpm_winnt, worker

このディレクティブは Apache プロセス稼働中における ThreadsPerChild に設定可能な上限値を設定します。再起動時にこのディレクティブの値を 変更しても無視されますが、ThreadsPerChild は再起動中に、このディレクティブで指定された上限値まで 変更することができます。

このディレクティブを使用する際は特に注意してください。 ThreadLimit が ThreadsPerChild よりもずっと大きな値に設定された場合は、 余計な未使用共有メモリが割り当てられてしまいます。 ThreadLimit が ThreadsPerChild の両方がシステムの扱える範囲を超えている場合は、 Apache は起動しないか、起動したとしても不安定になるでしょう。このディレクティブの値は今使用している Apache の ThreadsPerChild の予想上限値を 超えた値には設定しないでください。

ThreadLimit のデフォルト値は worker のときは 64 で、mpm_winnt のときは 1920 です。

ThreadsPerChild ディレクティブ

説明:	子プロセスそれぞれに生成されるスレッド数
構文:	ThreadsPerChild number
デフォルト:	ThreadsPerChild 50
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	worker, mpm_winnt

このディレクティブは、それぞれの子プロセスで生成される スレッド数を設定します。 子プロセスは開始時にこれらのスレッドを生成して、 その後は生成しません。mpm_winnt のような、 子プロセスが一つしかないような MPM を利用しているのであれば、 この値はサーバの負荷全体を十分取り扱える程度に、 大きくなければなりません。worker のような、 子プロセスが複数あるような MPM を利用しているのであれば、 サーバの通常負荷を十分扱える程度に、 スレッド総数が多くなければなりません。

User ディレクティブ

説明:	リクエストに応答する際に用いるユーザ ID
構文:	User unix-userid
デフォルト:	User #-1

コンテキスト:	サーバ設定ファイル, バーチャルホスト
ステータス:	MPM
モジュール:	worker, perchild, prefork

User ディレクティブは サーバがリクエストに応答する際に用いるユーザ ID を設定します。 このディレクティブを使用するためには、スタンドアロン型の サーバは最初に root 権限で起動されている必要があります。 unix-userid は次のどちらかです:

ユーザ名
: ユーザを名前で参照します。

に続いてユーザ番号
: ユーザを番号で参照します。

このユーザは、外部に見せるように意図していないファイルに、 アクセス可能になってしまうような権限を持つべきではないですし、 同様に httpd リクエストに対して応答するように意図していない 実行コードを、実行できるような権限を持つべきではないです。 サーバを実行するために特定の新しいユーザとグループを 設定することをお薦めいたします。 nobody ユーザを使用する管理者もいますが、 これが常に望ましいわけではありません。 なぜなら nobody ユーザは、システムで 他の役割を担っているかも知れないからです。

注意: 非 root ユーザでサーバを起動した場合は、 より低い権限のユーザに変わることに失敗して、 代わりに起動を行ったユーザ権限のまま実行され続けるでしょう。 root 権限で開始した場合親プロセスが root 権限で実行され続けますが、これは正常です。

特記事項: このディレクティブを <VirtualHost> で使用することはサポートされなくなりました。 suexec [7] 向けにサーバを設定するのであれば、 SuexecUserGroup を使用してください。

> セキュリティ
>
> 自分が何をやっているのか正確に把握していない、 そしてその危険性を把握していないのであれば、 User (や Group) を root に設定しないでください。

URI References

[1] http://httpd.apache.org/docs-2.1/suexec.html
[2] http://httpd.apache.org/docs-2.1/misc/security_tips.html#serverroot
[3] http://httpd.apache.org/docs-2.1/dns-caveats.html
[4] http://httpd.apache.org/docs-2.1/bind.html
[5] http://httpd.apache.org/docs-2.1/misc/security_tips.html
[6] http://httpd.apache.org/docs-2.1/stopping.html
[7] http://httpd.apache.org/docs-2.1/mod/mod_suexec.html