コンパイルとインストール

この文書で扱う範囲は、Unix や Unix に類似したシステムでの Apache のコンパイルとインストールです。 Windows における コンパイルとインストールに関しては「Microsoft Windows で Apache を使う [1]」をご覧下さい。 その他のプラットホームに関しては「プラットホーム [2]」をご覧下さい。

Apache 2.0 の設定とインストールの環境は、Apache 1.3 とは 完全に異なるものになりました。簡単にインストールできるように、 Apache 1.3 では特製スクリプトを使っていました。 Apache 2.0 では他の Open Source プロジェクトと同様の環境に するために libtool と autoconf を使うようになっています。

トピック

参照

- 起動 [3]
- 停止と再起動 [4]

概要 (せっかちな人向け)

ダウンロード	$ lynx http://www.apache.org/dist/httpd/httpd-2_1_NN.tar.gz
展開	$ gzip -d httpd-2_1_NN.tar.gz $ tar xvf httpd-2_1_NN.tar
設定	$ ./configure --prefix=PREFIX
コンパイル	$ make
インストール	$ make install
カスタマイズ	$ vi PREFIX/conf/httpd.conf
テスト	$ PREFIX/bin/apachectl start

NN は最新のマイナーバージョンナンバーに、 PREFIX はインストールするサーバでのファイルシステムのパスに、 置き換えてください。PREFIX を指定しなかった場合は、 デフォルトの /usr/local/apache2 になります。

Apache HTTPD のコンパイルとインストールに必要なものをはじめとして、 編集とインストールプロセスでのそれぞれの項は 次に詳しく記述されています。

必要なもの

Apache のビルドには次のものが必要です:

ディスクスペース
: ディスクに少なくとも 50 MB の一時的な空き容量があるように 気を付けてください。インストール後は Apache は 10 MB 程度の ディスクスペースを占めます。実際に必要になるディスクスペースは、 設定オプションやサードパーティー製モジュールをどう選択するかによって 大きく変わるでしょう。

ANSI-C コンパイラとビルドシステム
: ANSI-C コンパイラをインストールしておいて下さい。お薦めは Free Software Foundation (FSF) [5] による GNU C compiler (GCC) [6] です (バージョン 2.7.2 で大丈夫です)。GCC がない場合は、 少なくとも提供されているコンパイラが ANSI 準拠であることを確認しておいて下さい。 それから、変数 PATH には make といった基本的なビルドツールが含まれている必要があります。

時刻を正確にする
: HTTP プロトコルの要素は日時の時刻で表現されています。ですから、 正確な時刻にシンクロさせる機能をシステムに設定することを吟味してみて下さい。 Network Time Protocol (NTP) をベースとした ntpdate や xntpd プログラムが この目的によく用いられます。NTP ソフトウェアや公開 NTP サーバに関する詳細は、Usenet ニュースグループ comp.protocols.time.ntp [7] や NTP ホームページ [8] をご覧下さい。

Perl 5 [9] [オプション]
: 提供されているスクリプト幾つか、例えば apxs [10] や dbmmanage [11] は Perl で書かれているので、Perl 5 インタプリタが必要になります (5.003 以降)。 "configure"スクリプトでこのようなインタプリタが見つからなくても、 別に不具合はありません。もちろん、Apache 2.0 のコンパイルとインストールはできます。 これらのサポートスクリプトが使えなくなるだけです。 Perl インタプリタを複数インストールしている場合 (ベンダーの Perl 4 と自分で入れた Perl 5 がある場合など) は、--with-perl オプション (下記参照) を使って ./configure が適切なものを確実に選ぶようにすると良いでしょう。

ダウンロード

Apache は Apache Software Foundation ダウンロードサイト [12]や近くの ミラーサイト [13]からダウンロードできます。

alpha で終わるバージョン番号は、 動くか動かないかよく分からない、 早期のプリテストバージョンであることを示しています。 beta で終わるバージョンはそれよりは信頼性のある リリースですが、テストとバグフィックスを重ねる段階のものです。 Apache HTTP サーバの入手可能な中で最高の製品リリースを ダウンロードしたいのであれば、ファイル名に alpha も beta も付かない最新のバージョンを選んでください。

ダウンロード後、特にミラーサイトを使った場合は、 ダウンロードしたものが Apache HTTP サーバの完全で改竄されていないバージョンであることを 検証することが重要です。これはダウンロードした tarball の PGP 署名を テストすることによって検証します。 これは二つの段階があります。 まず KEYS ファイルを Apache 配布サイト [12] からダウンロードしてください。(KEYS ファイル自体が 改竄されていないことを確実にするために、以前

の Apache 配布から取り出したファイルを使ったり、 公開鍵サーバから鍵を取り込むのもいいでしょう。) 自分の個人キーホルダーに鍵を取り込むには、 次のコマンド (pgp のバージョンに依存) を使います:

```
$ pgp < KEYS
```

または、

```
$ gpg --import KEYS
```

次のステップでは、tarball を PGP 署名でテストします。 PGP 署名は必ず メインの Apache ウェブサイト [12]から取得してください。 署名ファイルの名前は、ソースファイルの tarball と同じ名前に .asc を付加したものになっています。 以下のコマンド (やっぱり pgp のバージョンに依存) のどれか一つで、 配布ファイルをチェックすることができます:

```
$ pgp httpd-2_1_NN.tar.gz.asc
```

または、

```
$ gpg --verify httpd-2_1_NN.tar.gz.asc
```

すると、こんなメッセージを受けるはずです。

```
Good signature from user "Martin Kraemer <martin@apache.org>".
```

自分のキーホルダーに格納されている信頼関係が原因で、"鍵とその鍵の署名者が検証できなかった" という旨のメッセージを同時に受けるかもしれません。 しかし、KEYS ファイルの信憑性を受け入れるならば問題ではありません。

展開

Apache HTTPD の tarball からソースファイルを展開して取り出すとは、 単なる圧縮の解除と tar の展開です:

```
$ gzip -d httpd-2_1_NN.tar.gz
$ tar xvf httpd-2_1_NN.tar
```

配布用のソースコードがある現在いるディレクトリの下に、 新しいディレクトリが作られます。 サーバをコンパイルする段階に進む前に、そのディレクトリに cd で移動してください。

ソースツリーを設定する

次のステップは、あなたのプラットホームと 個人的な要求に合うように Apache ソースツリーを設定することです。 これは配布ディレクトリのルートディレクトリにある、configure スクリプトで行ないます。 (Apache ソースツリーの CVS 版をダウンロードした

開発者は、次のステップに進む前に autoconf とlibtool をインストールして buildconf を実行する必要があるでしょう。 公式リリースではこの作業は必要ありません。)

デフォルトオプションを使ってソースツリーを全て設定する のであれば、単純に ./configure とタイプしてください。 デフォルトオプションを変更できるように、configure には様々な変数やコマンドラインオプションが用意されています。 一般的に、環境変数が ./configure の前に設置されて、 他のオプションはその後に設置されます。 ここで最も重要なオプションは Apache をどこにインストールするかを決める location prefix です。なぜなら、Apache が正しく動作するためには この場所用に設定されていないといけないからです。 しかし、お好みにより利用できるオプションはもっと沢山あります。

ちょっとどんなことができるかを見せましょう。 ここで典型的な例として、/sw/pkg/apache というインストールツリーでコンパイラとフラグを指定して、 さらに二つの追加モジュール mod_rewrite と mod_speling を後で DSO メカニズムでロードするようにコンパイルしてみます:

```
$ CC="pgcc" CFLAGS="-O2" ¥
./configure --prefix=/sw/pkg/apache ¥
--enable-rewrite=shared ¥
--enable-speling=shared
```

configure を実行したら、システムの機能を テストしたり、後でサーバをコンパイルするために必要な Makefile を生成したりするのに数分間かかるでしょう。

Apache の全ての設定フラグを見る最も簡単な方法は、 ./configure --help を実行する方法です。 引数や環境変数に関する簡単な記述が出力されます。

環境変数

autoconf でのビルドでは幾つかの環境変数を使ってビルド環境を 設定します。一般的に、これらの変数で Apache をビルドする際の 方法が変わったりしますが、できあがるサーバの機能には影響ありません。 これらの変数は configure を呼び出す直前の環境中に 置くことができますが、通常は上に示した例のように configure のコマンドラインでもっと簡単に指定できます。

CC=...
: C コンパイラのコマンド名。

CPPFLAGS=...
: その他の C プリプロセッサやコンパイラのオプション。

CFLAGS=...
: C コンパイラのデバッグ・最適化オプション。

LDFLAGS=...
: その他のリンカに渡されるオプション。

LIBS=...
: リンカに渡すライブラリの位置情報 ("-L"と"-l"オプション) 。

INCLUDES=...
: ヘッダファイルの探索ディレクトリ ("-Idir") 。

TARGET=... [デフォルト値: apache]
: ビルドする実行ファイルの名前。

NOTEST_CPPFLAGS=...

NOTEST_CFLAGS=...

NOTEST_LDFLAGS=...

NOTEST_LIBS=...
: これらの変数は "NOTEST でない版" のものと同じ機能を 持っています。しかし、これらの変数は autoconf がテストを行った後のビルドプロセスにおいてのみ、適用されます。 これを使うと、テスト中に問題を起すけれども 最終的には必要になるフラグを含めることができるようになります。

SHLIB_PATH=...
: コンパイラとリンカに渡す、共有ライブラリへのパスを指定する オプション。

autoconf 出力オプション

--help
: 使用可能な全オプションなど使い方をプリントします。 実際に設定はされません。

--quiet
: "cheking..." といったメッセージがプリントされないように します。

--verbose
: 検査される全ファイル名を含め、設定プロセス中に 情報をたくさんプリントします。

パス名

Apache をインストールするパス名を設定するには、 現在二通りの方法があります。まず一つ目は、 ディレクトリを指定して、その下にデフォルトの配置で Apache のインストールを行う方法です。

--prefix=PREFIX [デフォルト値: /usr/local/apache2]
: Apache のファイル群がインストールされるディレクトリを 指定します。

アーキテクチャに依存したファイルを、 異なるディレクトリに配置するようにもできます。

--exec-prefix=EPREFIX [デフォルト値: PREFIX]
: アーキテクチャ依存のファイルを配置すべき ディレクトリを指定します。

二つ目の方法は、もっと柔軟にインストールパスの配置を 設定する方法で、config.layout ファイルを 使います。この方法を使うことによって、 Apache のインストール中に、それぞれのファイルのための配置を 個々に指定できるようになります。config.layout ファイルには設定例が幾つか含まれていますし、 お好みの設定を次の例に従って作り出すこともできます。 このファイル中では、異なる配置は <Layout FOO>...</Layout> セクションでグループ化され、 FOO といった名前で参照されます。

--enable-layout=LAYOUT
: インストールパスを指定するため、 config.layout ファイル中での、 指定された名前のレイアウトを使用します。

モジュール

Apache はモジュール化されたサーバです。 ごくごく基本的な機能だけが、コアサーバに含まれています。 拡張機能は様々なモジュールの形で提供されます。 設定プロセス中では、

どんなモジュールをサーバで使うように コンパイルするか選ばなければなりません。このマニュアルにあるモジュールの一覧[14]を 参照できます。 "Base" ステータス[15]の モジュールはデフォルトで含まれますし、 使いたくないのであれば、わざと無効にしなければ なりません（例えば mod_userdir ）。 他のステータスのモジュールは、 使いたければ有効にしなければなりません（例えば mod_expires）。

Apache と一緒にコンパイルして使うようにするには、 二通りの方法があります。一つめはモジュールを スタティックコンパイルする方法です。この場合は、 Apache のバイナリに恒久的に組み込まれることになります。 これの代わりに、もしオペレーティングシステムが動的共有 ライブラリ (DSO) (訳注: Dynamic Shared Object) を 提供していて autoconf がそれを認識できる 場合は、モジュールをダイナミックコンパイルする方法があります。DSO モジュールは Apache のバイナリとは別に 保存され、mod_so で提供される 実行時設定ディレクティブを使って 組み込んだり取り外したりできます。 動的モジュールを実際に一つもコンパイルすることなく サーバが DSO をロードできるようにするには、 --enable-so と明示的にすることができます。

--enable-MODULE[=shared]
: MODULE モジュールをコンパイルして インクルードします。識別子 MODULE は モジュール文書に記載されているモジュール 識別子[16]から "_module" を取り除いた文字列です。 DSO としてモジュールをコンパイルする場合は、 =shared オプションを付加してください。

--disable-MODULE
: 通常はコンパイルされてインクルードされる MODULE モジュールを除去します。

--enable-modules=MODULE-LIST
: スペース区切りの MODULE-LIST に列挙されたモジュールをコンパイルして インクルードします。

--enable-mods-shared=MODULE-LIST
: スペース区切りの MODULE-LIST を ダイナミックロード (DSO) できるモジュールとして コンパイルとインクルードをします。

--enable-modules や --enable-mods-shared オプションに使う MODULE-LIST は、普通はスペース区切りの モジュール識別子のリストです。 例えば mod_dav と mod_info を有効にする場合は、次のどちらかが使えます。

```
./configure --enable-dav --enable-info
```

または、同等の

```
./configure --enable-modules="dav info"
```

これに加えて、特別なキーワード all や most を使って、 一度に全てあるいはほとんどのモジュールを加えることができます。 その後で好きなモジュールを --disable-MODULE オプションを使って取り除くことができます。 例えば、mod_info を除く全てのモジュールを DSO モジュールとして組み込む場合は、次のようにします。

```
./configure --enable-mods-shared=all --disable-info
```

標準的なモジュールに加えて、Apache 2.0 では Multi-Processing Modules[17] (MPM) を選

択してインクルードします。ただ一つだけの MPM をコンパイルのプロセスで含める必要があります。 個々のプラットホーム向けのデフォルトの MPM は MPM 文書[17]に一覧がありますが、 configure コマンドで置き換えることができます。

--with-mpm=NAME
: MPM モジュール NAME を選択します。

DBM

mod_authn_dbm と mod_rewrite の DBM RewriteMap 等、 Apache 機能の幾つかでは情報の検索に単純な key/value データベースを使用します。Apache には SDBM がソースコードごと含まれていますので、このデータベースはいつでも利用可能です。 他のタイプのデータベースを使用したい場合は、次の configure オプションが利用できます。

--with-gdbm[=path]

--with-ndbm[=path]

--with-berkeley-db[=path]
: もしパス (path) が指定されなかった場合は、Apache は通常の検索パス上でインクルードファイルとライブラリを探します。 明示的に path を指定すると、Apache は path/lib と path/include を検索して関連ファイルを探します。path にはコロンで区切って複数のインクルード、ライブラリパスを指定できます。

Suexec

Apache には suexec [18] と呼ばれる 補助プログラムがあります。 このプログラムはユーザの CGI プログラムを隔離するために 使用することができます。しかしながら、suexec を適切に設定しなければ、 セキュリティ上致命的な問題をかかえる場合があります。 そのため、この機能を実装する前に suexec 文書[18]をよく読んで一考しておきましょう。

ビルド

これで Apache の様々なパーツをビルドすることができます。 次のコマンドを単純に実行するだけです:

```
$ make
```

基本的な設定をするのに、Pentium III/Linux 2.2 のシステムでおおよそ 3 分程度かかりますが、 あらかじめご了承下さい。 また、時間はハードウェアや有効にしたモジュールの数に 大きく依存するでしょう。

インストール

さて、設定したインストール PREFIX (前述の --prefix オプションを参照) 以下にパッケージをインストールする段階になりました。 次のコマンドを実行してください:

```
$ make install
```

アップグレードする場合は、インストールでは設定ファイルや ドキュメントファイルの上

書きは行いません。

カスタマイズ

次に PREFIX/conf/ 以下にある 設定ファイル [19]を編集して、 Apache HTTP サーバをカスタマイズします。

```
$ vi PREFIX/conf/httpd.conf
```

docs/manual/ [20] の Apache マニュアルをざっと見てください。 または、 http://httpd.apache.org/docs-2.1/ にあるマニュアル最新版、設定ディレクティブ [21]に当たってみてください。

テスト

次のコマンドを実行して Apache HTTP サーバを開始 [3]できます:

```
$ PREFIX/bin/apachectl start
```

URL http://localhost/ を通して最初のドキュメントに対する リクエストを発行する事ができるはずです。これで見える ウェブページは DocumentRoot 以下に置かれたもので、通常は PREFIX/htdocs/ でしょう。 サーバを再び停止 [4]するには、 次のコマンドを実行します:

```
$ PREFIX/bin/apachectl stop
```

URI References

[1] http://httpd.apache.org/docs-2.1/platform/windows.html
[2] http://httpd.apache.org/docs-2.1/platform/
[3] http://httpd.apache.org/docs-2.1/invoking.html
[4] http://httpd.apache.org/docs-2.1/stopping.html
[5] http://www.gnu.org/
[6] http://www.gnu.org/software/gcc/gcc.html
[7] news:comp.protocols.time.ntp
[8] http://www.eecis.udel.edu/~ntp/
[9] http://www.perl.org/
[10] http://httpd.apache.org/docs-2.1/programs/apxs.html
[11] http://httpd.apache.org/docs-2.1/programs/dbmmanage.html
[12] http://www.apache.org/dist/httpd/
[13] http://www.apache.org/dyn/closer.cgi/httpd/
[14] http://httpd.apache.org/docs-2.1/mod/
[15] http://httpd.apache.org/docs-2.1/mod/module-dict.html#Status
[16] http://httpd.apache.org/docs-2.1/mod/module-dict.html#ModuleIdentifier
[17] http://httpd.apache.org/docs-2.1/mpm.html

[18] http://httpd.apache.org/docs-2.1/suexec.html
[19] http://httpd.apache.org/docs-2.1/configuring.html
[20] http://httpd.apache.org/docs-2.1/
[21] http://httpd.apache.org/docs-2.1/mod/directives.html