# Hypertext Preprocessor

Part - 4


インストール

# 第 4 部

PHP

Hypertext Preprocessor

HP

# Chapter-1 インストールの前に

## 1.1 前提事項

Linuxのディストリビューション（カーネルに各種ソフトウェアパッケージを組み合わせた配布物または商品）においては、RedHat、Kondara、Vine、TurboLinuxなどのいわゆるRedHat系といわれるカテゴリが存在します。これらのディストリビューションでは、各種アプリケーションのインストールにおいて、RPM（Redhat Package Manager）という管理システムを使っています。RPMは簡単にパッケージのインストール／アンインストールができ、makeなどの細かい作業をしなくてもよいという優れた特徴があります。その反面、最新バージョンに追随しにくい（最新バージョンが出ても、誰かがそのバージョン用のRPMパッケージを作ってくれるのを待つか、自分で作らなければならない）というデメリットも存在します。

特にPHP4に関しては、登場してまだあまり間がないことや、国際化の作業が継続中であるということもあり、安定して動作するRPMパッケージがまだ存在しないようです。そこで本書では、ソース（tar.gz形式のアーカイブ。いわゆるtar ball）から各ソフトウェアをインストールする方法について説明します。この方法であれば、本書の出版後に各ソフトウェアのバージョンが上がっても、バージョン番号を読み替える程度で常に最新版をダウンロードしてインストールすることができるようになるでしょう。また、Linux以外のOSを使っている人にとっても、以下の手順は参考になるでしょう。

オープンソース系であればどんなソフトウェアでも同じですが、インストール途中で何かトラブルが起こったら（できればトラブルが起こる前に）、まずはソース配布物に含まれるREADMEやINSTALLといったテキストファイルを読んでください。ヒントになることが書かれています。

以下で紹介するコマンドの実行例では、シェルのプロンプトが**#**で終わっている場合はroot権限で実行することを、**$**で終わっている場合は特定の一般ユーザ権限で実行することを表しています。

# 1.2 必要な開発ツールの確認

各ソフトウェアをインストールするにあたっては、以下に示すいくつかのツールがインストールされていることが前提となります。これらがインストールされていない場合は、ソースを展開して生成されるINSTALLファイルに目を通してみてください。

以下に示すコマンドのバージョン表示やインストールの例は、原稿執筆時にプラットフォームとして使用したVine Linux 2.0の実行結果です。ほかのディストリビューションやプラットフォームでは異なっていることもあります。Vine Linux 2.0を使用している場合には、必要なツール群はすべてインストールされています。Linuxを使う限りにおいては、最近のディストリビューションであればほかのものでもほぼ大丈夫でしょう。なお、RPMなどの各コマンドの使い方については、**man rpm**などを参照してください。

### ● GNU make

```
hotta@star:~$ make -v
GNU Make version 3.77, by Richard Stallman and Roland McGrath.
Copyright (C) 1988, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98
        Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

Report bugs to <bug-make@gnu.org>.

hotta@star:~$ rpm -qf /usr/bin/make
make-3.77-6
```

バージョン3.75以降のGNU makeが必要です。プラットフォームによっては、システム標準のmakeコマンドのほかに別途、gmakeという名前でGNU makeがインストールされているかもしれません。この場合は**gmake -v**で確認できます。gmakeを使う場合は、以下適宜makeをgmakeと読み替えてください。

### ● gcc（C コンパイラ）

これは必須ではありませんが、各種商用UNIXに付属しているCコンパイラの中には、コンパイルできないものもあるかもしれません。

```
hotta@star:~$ gcc -v
Reading specs from /usr/lib/gcc-lib/i386-redhat-linux/egcs-2.91.66/specs
gcc version egcs-2.91.66 19990314/Linux (egcs-1.1.2 release)

hotta@star:~$ rpm -qf /usr/bin/gcc
egcs-1.1.2-24vl1
```

- **readline（コマンドライン編集用ライブラリ）**
- **history（コマンドヒストリ保持用ライブラリ）**

これらが組み込まれていると、psqlコマンドでコマンドライン編集やコマンドヒストリ機能が使えるようになり、使い勝手が格段に向上します。

```
hotta@star:~$ ls /usr/lib/lib{readline,history}*
/usr/lib/libhistory.a       /usr/lib/libreadline.a
/usr/lib/libhistory.so@     /usr/lib/libreadline.so@
/usr/lib/libhistory.so.3@   /usr/lib/libreadline.so.3@
/usr/lib/libhistory.so.3.0  /usr/lib/libreadline.so.3.0

hotta@star:~$ ls /usr/include/readline/*
/usr/include/readline/chardefs.h  /usr/include/readline/readline.h
/usr/include/readline/history.h   /usr/include/readline/tilde.h
/usr/include/readline/keymaps.h

hotta@star:~$ rpm -qa|grep readline
readline-2.2.1_jp-5
readline-devel-2.2.1_jp-5
```

# 1.3 インストール済みのパッケージの削除

この節の話題はRedHat系のLinuxを使っている人だけに関係する話です。それ以外の人は読み飛ばしてもらってかまいません。

RedHat系を使っている場合は、必要なソフトウェアが（ある意味、中途半端に）インストールされていることがあります。このまま本書の推奨手順で同じソフトウェアをソースから構築してしまうと、新旧バージョン間のコンフリクト（衝突）が起こり、トラブルの元になります。お勧めは、それらプリインストールされたものをばっさり削除してしまうことです。もしこれらプリインストールされたソフトウェアを使ってすでに何らかのデータを登録済みであれば、それらのバックアップを取っておいてください。

まず、インストールしようとしているソフトウェアがプリインストールされているかどうかを調べます。こちらの環境では以下のようになっているとします[*1]。

*1 この例は擬似的に作成したもので、かならずしもVine Linux 2.0のデフォルトの状態を表すものではありません。

```
root@star:~# rpm -qa|grep postgres
postgresql-7.0.2-2vl2
postgresql-devel-7.0.2-2vl2
postgresql-jdbc-7.0.2-2vl2
postgresql-odbc-7.0.2-2vl2
postgresql-perl-7.0.2-2vl2
postgresql-python-7.0.2-2vl2
postgresql-server-7.0.2-2vl2
postgresql-tcl-7.0.2-2vl2
postgresql-test-7.0.2-2vl2
postgresql-tk-7.0.2-2vl2
root@star:~# rpm -qa|grep apache
apache-1.3.12-2vl3
apache-devel-1.3.12-2vl3
apache-manual-1.3.12-2vl3
root@star:~# rpm -qa|grep php
php-3.0.15-2
php-imap-3.0.15-2
php-ldap-3.0.15-2
php-manual-3.0.15-2
php-pgsql-3.0.15-2
```

インストールされているこれらのパッケージをすべて削除します。この作業はroot権限で行います。

```
root@star:~# rpm -qa|grep postgres|xargs rpm -e
root@star:~# rpm -qa|grep apache|xargs rpm -e
root@star:~# rpm -qa|grep php|xargs rpm -e
```

以下のようなエラーが出ることがあります。

```
/var/lib/pgsql/data を削除できません - ディレクトリが空でありません
```

これは、ユーザが作成したデータや設定ファイルなどは削除しないようにガードがかかっているためです。必要なものはバックアップを取ってから、手作業で削除してください。

きれいに削除されたか、もう一度確認します。なにも出力されなくなれば大丈夫です。

```
root@star:~# rpm -qa|grep postgres
root@star:~# rpm -qa|grep apache
root@star:~# rpm -qa|grep php
```

Hypertext Preprocessor

HP

# Chapter-2 基本構成のインストール

この章では、PostgreSQL+Apache（DSO）+PHP3 国際化バージョン（Apache モジュール）という組み合わせのインストールを行います。

## 2.1 PostgreSQLのインストール

### 2.1.1 専用アカウントの作成

PostgreSQLはセキュリティ保護のために、root権限では起動しないようになっています。また、インストールやデータベースの作成といったPostgreSQLの運用管理をする場合にも専用のアカウントで行うことが推奨されています。ここでは慣例にしたがって、postgresという一般ユーザをadduserコマンドで作成し、このアカウントでインストール作業のほとんどを行っています。

```
root@star:~# adduser postgres
root@star:~# passwd postgres
Changing password for user postgres
New UNIX password: （パスワード）
Retype new UNIX password: （パスワード）
passwd: all authentication tokens updated successfully
```

### 2.1.2 作業用ディレクトリの作成

ソースの展開用およびインストール用としてディレクトリを作成します。

```
root@star:~# mkdir -p /usr/local/src/postgresql-7.0.2
root@star:~# chown postgres /usr/local/src/postgresql-7.0.2
root@star:~# mkdir -p /usr/local/pgsql
root@star:~# chown postgres /usr/local/pgsql
```

### 2.1.3 CD-ROM のマウント

本書付属のCD-ROMをマウントします。Vine Linuxでは以下の方法でマウントできますが、ほかのプラットフォームでは異なる場合もあります。

```
root@star:~# mount /mnt/cdrom
```

これ以降、CD-ROMの基点は**/mnt/cdrom**であることを前提に作業します。必要に応じて適宜読み替えてください。

### 2.1.4 ソースの展開とパッチの適用

これ以降はユーザpostgresで作業を行います。

```
root@star:~# su - postgres
postgres@star:~$  cd /usr/local/src/
postgres@star:/usr/local/src$ tar xvzf /mnt/cdrom/arc/postgresql-7.0.2-patched.tar.gz
postgres@star:/usr/local/src$ zcat /mnt/cdrom/arc/psqlj-7.0.2.diff.gz | patch -p0
```

最後のステップはオプションです。このパッチにより、会話型SQLクライアントであるpsqlの日本語版psqljが作られます。

### 2.1.5 configure の実行

ほかの多くのオープンソース・ソフトウェアと同様に、PostgreSQLでもconfigureスクリプトが用意されています。これは各種プラットフォーム間の差異を吸収し、コンパイル（make）の際に必要となる適切なMakefile（GNUmakefile）を自動生成してくれるものです。メジャーなプラットフォームの場合は以下のオプションだけでGNUmakefileが作られます。

```
postgres@star:/usr/local/src$ cd postgresql-7.0.2/src/
postgres@star:/usr/local/src/postgresql-7.0.2/src$ ./configure --enable-multibyte=EUC_JP
```

**--enable-multibyte=EUC_JP**オプションは、多バイト拡張機能を有効にします。PostgreSQLのバージョン6.4からは、データベースの作成時に明示的に文字コードを指定することができるようになりました。このオプションでは、コード指定がない場合のデフォルト文字コードとしてEUCを指定しています。残念ながらconfigureでエラーとなってしまっ

た場合、**--with-template**オプションを使って明示的にプラットフォームを指定してみるとうまくいくかもしれません。詳細は**./configure --help**で表示されるヘルプを参照してください。

## 2.1.6 コンパイルおよびインストール

コンパイルを行います。マシンスペックにより数分から数十分かかります。

```
postgres@star:/usr/local/src/postgresql-7.0.2/src$ make
(中略)
All of PostgreSQL is successfully made. Ready to install.
```

**/usr/local/pgsql**配下にバイナリをインストールします。

```
postgres@star:/usr/local/src/postgresql-7.0.2/src$ make install
```

**/usr/local/pgsql/doc**配下にhtml形式のマニュアルをインストールします。

```
postgres@star:/usr/local/src/postgresql-7.0.2/src$ cd ../doc/
postgres@star:/usr/local/src/postgresql-7.0.2/doc$ make install
```

以上でインストールは完了です。なお、添付CD-ROMの**/docs/pgsql**配下には日本語マニュアルも入っています。

## 2.1.7 環境設定

管理者ユーザであるpostgres、およびPostgreSQLを利用するすべてのユーザについて、コマンドサーチパスと環境変数の設定を行います。シェルとしてbashを使っている場合は~**/.bashrc**に、csh／tcshを使っている場合は~**/.cshrc**に以下の設定を追加します。

●~/.bashrc への追加設定

```
PATH="$PATH":/usr/local/pgsql/bin
export POSTGRES_HOME=/usr/local/pgsql
export PGLIB=$POSTGRES_HOME/lib
export PGDATA=$POSTGRES_HOME/data
export MANPATH="$MANPATH":$POSTGRES_HOME/man
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH":"$PGLIB"
```

●~/.cshrc への追加設定

```
set path = ($path /usr/local/pgsql/bin)
setenv POSTGRES_HOME /usr/local/pgsql
setenv PGLIB $POSTGRES_HOME/lib
setenv PGDATA $POSTGRES_HOME/data
if ($?MANPATH) then
    setenv MANPATH "$MANPATH":$POSTGRES_HOME/man
else
    setenv MANPATH $POSTGRES_HOME/man
endif
if ($?LD_LIBRARY_PATH) then
    setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH":"$PGLIB"
else
    setenv LD_LIBRARY_PATH $PGLIB
endif
```

**source ~ /.bashrc**（または**source ~ /.cshrc**）を実行して、設定を反映させます。

このあと実際にいくつかのプログラムを起動します。各プログラムには多くの引数が用意されていますが、ここでは動作確認に必要な最小限の使用にとどめます。

## 2.1.8 データベースの初期化

データベースを使用する前に、データベースの入れ物を用意する作業を行います。具体的には$PGDATAディレクトリに、管理用データベースとユーザデータベースのひな型をコピーします。

```
postgres@star:~$ initdb
```

なお、initdbを起動したユーザ（通常はpostgres）がそのデータベース領域の所有者となります。このあとユーザが作成するデータベースやテーブルの実体は、特に指定しない限り**$PGDATA/base**配下に置かれます。

## 2.1.9 postmasterの起動

PostgreSQLのデーモンプログラムであるpostmasterを起動します。postmasterコマンドで有効なオプションを以下に示します。

```
Usage: postmaster [options]
  -B nbufs     共有バッファ数のセット
  -D datadir   データディレクトリのセット
  -S           サイレントモード(ttyから切り離す)
  -a system    この認証システムを使用する
  -b backend   指定したバックエンドサーバを使用する
  -d [1-5]     デバッグレベルの指定
  -i           UNIXソケットに加えTCP/IPソケットからの接続を受けつける
  -N nprocs    バックエンドの最大数(1..1024, デフォルト 32)
  -n           異常終了後に共有メモリを再初期化しない
  -o option    各バックエンドサーバへ 'option' を渡す
  -p port      接続を待つポート番号
  -s           バックエンドのうち一つが死んだらほかのバックエンドにもSIGSTOPを送る
```

実際には、postmasterプログラムの起動スクリプトpg_ctlを使って起動すると便利です。pg_ctlの起動オプションを以下に示します。

```
pg_ctl [-w][-D DIR][-p PATH] [-o "OPTS"] start
pg_ctl [-w][-D DIR][-m s|f|i] stop
pg_ctl [-w][-D DIR][-m s|f|i] [-o "OPTS"] restart
pg_ctl [-D DIR] status
```

```
-h|--help    ヘルプを表示する
-w           既存のpostmasterプロセスが終了するのを待ってから起動する
-D DIR       $PGDATAディレクトリを指定する
-p PATH      postmasterが存在するパスを指定する
-m s|f|i     シャットダウン時のモード指定
                 s(mart)      スマートなシャットダウン(SIGTERMを送る)
                 f(ast)       早いシャットダウン(SIGINTを送る)
                 i(mmediate)  即時シャットダウン(SIGQUITを送る)
-o "OPTS"    postmasterプログラムに渡す起動時オプションを指定する
start        postmasterを起動する
restart      postmasterを再起動する
stop         postmasterを終了する
status       postmasterの動作状況を得る
```

**-o** オプションが指定されない場合、pg_ctlはpostmasterに渡すべきオプションを**/usr/local/pgsql/data/postmaster.opts.default**ファイルから読み込みます。

```
chmod +w /usr/local/pgsql/data/postmaster.opts.default
```

としてpostmaster.opts.defaultを書き込み可能としてから、適当なエディタでこのファイルにデフォルトのオプションを書き込んでおきましょう。一例を挙げます。

```
-S      Webサーバと DBサーバが同一マシンの場合。postmasterが動作しているホストからの接続のみを許可する
-S -i   他のホストからの接続を許す場合
```

そしてpg_ctlコマンドでpostmasterを起動します。

```
postgres@star:~$ pg_ctl -w start
Waiting for postmaster starting up...done.
postmaster successfully started up.
```

## 2.1.10 アクセス制御

postmasterの起動時オプションに**-i**をつければ別ホストからの接続を許可するようになりますが、さらにアクセス制御ファイル**/usr/local/pgsql/data/pg_hba.conf**を適切に設定する必要があります。別ホストからの接続を行わない場合は、デフォルトのままでかまいません。

とりあえずどこからでもアクセスできるようにしたい場合は、140行目付近にある

```
#host         all         0.0.0.0         0.0.0.0           trust
```

のコメントをはずしてください。

**trust**のところを**passwd**もしくは**crypt**に代えると、パスワード認証ができるようになります。

- **passwd**

普通のファイルにユーザ名とパスワードを書いておく方法。詳しくはpg_hba.conf(5)とpg_passwd(1)manページを参照してください。

- **crypt**

PostgreSQLのデータベースでパスワードを管理する方法。詳しくはpg_hba.conf(5)とalter_user(l)manページを参照してください。

## 2.1.11 regression test(回帰テスト)

次に、PostgreSQLがシステムに正しくインストールされたかどうかを確認するためのregression testを行います。regression testは標準SQL機能に加え、PostgreSQLで用意されている拡張機能のテストも行います。

```
postgres@star:~$ pg_ctl -w start (postgresが動作していない場合のみ)
postgres@star:~$ cd /usr/local/src/postgresql-7.0.2/src/test/regress/
postgres@star:/usr/local/src/postgresql-7.0.2/src/test/regress$ make all runtest
```

このあと数行のメッセージの後テストが開始され、順次結果が表示されます。テスト完了までに手元のマシンでは15分ほどかかりました。

```
=============== running regression queries...        =================
boolean .. ok
char .. ok
name .. ok
varchar .. ok
text .. ok
int2 .. ok
int4 .. ok
(以下略)
```

上記のように各機能に対しての動作チェックが行われ、逐次okまたはfailedが表示されます。この表示結果は、テスト終了後に**regress.out**というファイルに保存されます。これで全項目がokとなれば申し分ありませんが、そうでないこともあるかもしれません。参考までに、手元の環境(Vine Linux 2.0)ではすべてokになりました。

テストが失敗する原因は、テスト方法に起因するものがほとんどです。regression testでは、結果として期待される文字列をテキストファイル(**expected/*.out**)として前もって用意しておき、これらと実行結果をファイルに落としたものとの単純な差分を取ることでokかどうかを判定しています。差分が発生したらその項目をfailedとするとともに、差分を**regression.diffs**というファイルに保存します。このファイルをみてfailedが妥当であるかどうかを判断します。failedになる典型的なパターンとしては以下のものが挙げられます。いずれもプラットフォーム間の差異に起因するものです。

- **エラーメッセージの違い(perror(3)の出力)**
- **浮動小数点の表現方法の違い**
- **乱数を使ったテストにおける違い**

詳細は同ディレクトリにあるREADMEファイルを参照してください。

# 2.2 Apacheのインストール

ここから先は再びroot権限で作業を行います。

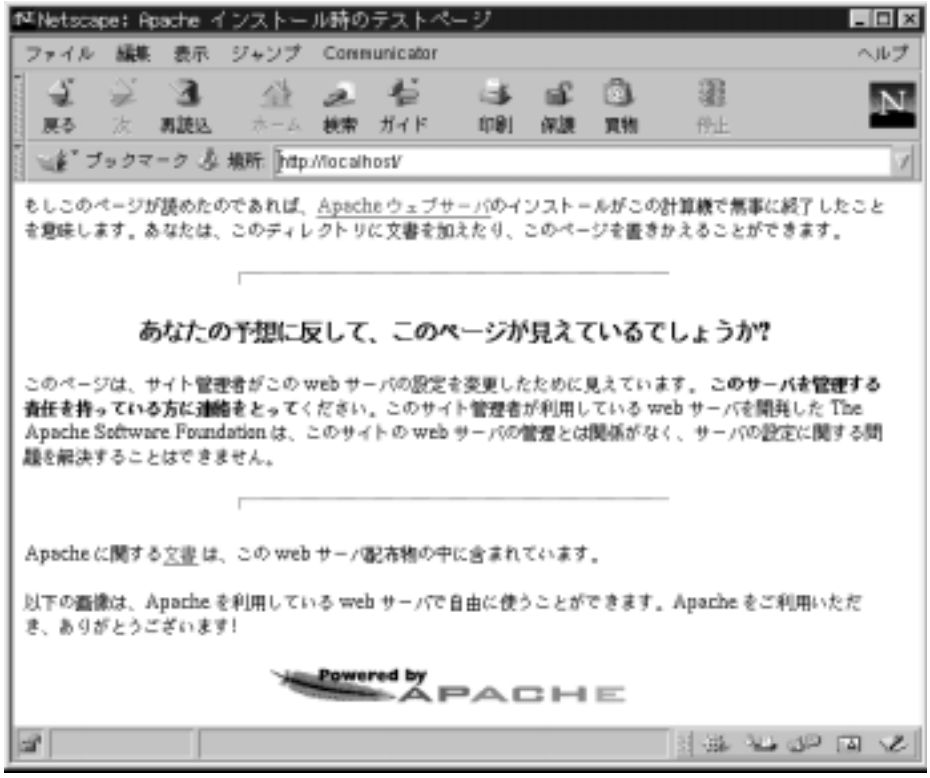
```
root@star:~# cd /usr/local/src/
root@star:/usr/local/src# tar xvzf /mnt/cdrom/arc/apache_1.3.12.tar.gz
root@star:/usr/local/src# cd apache_1.3.12/
root@star:/usr/local/src/apache_1.3.12# OPTIM="-O2" ./configure --enable-module=so
root@star:/usr/local/src/apache_1.3.12# make
root@star:/usr/local/src/apache_1.3.12# make install

+--------------------------------------------------------+
| You now have successfully built and installed the      |
| Apache 1.3 HTTP server. To verify that Apache actually |
| works correctly you now should first check the         |
| (initially created or preserved) configuration files   |
|                                                        |
|   /usr/local/apache/conf/httpd.conf                    |
|                                                        |
| and then you should be able to immediately fire up     |
| Apache the first time by running:                      |
|                                                        |
|   /usr/local/apache/bin/apachectl start                |
|                                                        |
| Thanks for using Apache.       The Apache Group        |
|                                http://www.apache.org/  |
+--------------------------------------------------------+
```

このように表示されればApacheのインストールは成功です。ではApacheを起動してみましょう。

```
root@star:/usr/local/src/apache_1.3.12# /usr/local/apache/bin/apachectl start
/usr/local/apache/bin/apachectl start: httpd started
```

手元のブラウザから、**http://localhost/**にアクセスしてみましょう。**図4-1**の画面が表示されたらインストール成功です。

図4-1 インストール直後のApache




# 2.3 PHP3国際化バージョンのインストール

PHP3の国際化バージョンをインストールします。次章ではPHP4のインストールを行いますが、PHP3とPHP4は混在可能です。ここではPHPをApacheのDSO（Dynamic Shared Object）としてインストールします。

```
root@star:/usr/local/src/apache_1.3.12# cd ..
root@star:/usr/local/src# tar xvzf /mnt/cdrom/arc/php-3.0.15-i18n-ja.tar.gz
root@star:/usr/local/src# cd php-3.0.15-i18n-ja/
root@star:/usr/local/src/php-3.0.15-i18n-ja# ./configure ¥
  --with-pgsql --enable-track-vars --with-apxs=/usr/local/apache/bin/apxs ¥
  --enable-i18n --enable-mbregex  --enable-versioning --without-gd
root@star:/usr/local/src/php-3.0.15-i18n-ja# make
root@star:/usr/local/src/php-3.0.15-i18n-ja# make install
```

これで実行イメージがコピーされました。後は、アクセスされるコンテンツに関する拡張子の関連づけを行います。**/usr/local/apache/conf/httpd.conf**の中に

```
#AddType application/x-httpd-php3 .php3
#AddType application/x-httpd-php3-source .phps
```

という2行があるので、行頭の**#**(コメント行)をはずしてください。これによって、**.php3**というファイルがアクセスされると、ApacheによりPHP3スクリプトであると判断され、PHP3実行ルーチンに処理が移ります。また、PHP3スクリプトに**.phps**という拡張子をつけてブラウザからアクセスすると、PHP3ソースをカラーで見やすく表示してくれるようになります。

これで設定は終了です。Apacheを再起動してみましょう。

```
root@star:~# /usr/local/apache/bin/apachectl restart
/usr/local/apache/bin/apachectl restart: httpd restarted
```

環境によっては、

```
Syntax error on line 203 of /usr/local/apache/conf/httpd.conf:
Cannot load /usr/local/apache/libexec/libphp3.so into server:
  Can't find shared library "libpq.so.2.0"
/usr/local/apache/bin/apachectl start: httpd could not be started
```

というエラーが表示されるかもしれません。これは、PHP3の実体である**libphp3.so**が、PostgreSQLのAPIライブラリ**libpq.so.2.0**を読み込めなかったことを示しています。この場合、以下のいずれかを試してみてください。

**●LD_LIBRARY_PATHを指定する**

**bashの場合**

```
root@star:~# LD_LIBRARY_PATH=/usr/local/pgsql/lib ¥
  /usr/local/apache/bin/apachectl start
```

**csh/tcshの場合**

```
root@star:~# env LD_LIBRARY_PATH=/usr/local/pgsql/lib ¥
  /usr/local/apache/bin/apachectl start
```

● **/etc/ld.so.conf に以下の１行を追加する**

```
/usr/local/pgsql/lib
```

そのあとで**ldconfig**コマンドを実行します。

次にPHP3のサンプルを表示することにより動作を確認します。**/usr/local/apache/htdocs/phpinfo.php3**という名前のスクリプトをエディタで作ってみてください。中身は以下の1行のみです。

```
<?  phpinfo();  ?>
```

できたら、ブラウザからこのスクリプトを呼び出してみます。

```
http://localhost/phpinfo.php3
```

**図4-2**の画面が表示されれば、インストールは成功です。

**図４-２　phpinfo()**

なお、PHP3には**php3.ini**という設定ファイルが存在します（存在しない場合はデフォルトの動作をします）。PHP3のデフォルトの動作を変更したい場合は、以下のように設定ファイルを作成しておいてください。

```
root@star:~# cp /usr/local/src/php-3.0.15-i18n-ja/php3.ini-dist ¥
 /usr/local/lib/php3.ini
```

**php3.ini**についての内容は、第3部「4.2　PHPオプションおよび情報」を参照してください。

## 2.4 自動起動の設定

マシンをブートする際に自動的にPostgreSQL+Apache+PHPを起動するには、以下の設定を行います。ここはrootで作業してください。

**/etc/rc.d/rc.local**に、以下の設定を追加します。

```
if [ -x /usr/local/pgsql/bin/pg_ctl ]; then
    su postgres -c "/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data start"
fi
if [ -x /usr/local/apache/bin/apachectl ]; then
    /usr/local/apache/bin/apachectl start
fi
```

このあとリブートを行い、再起動後正常にpostmasterとhttpdが起動しているのを確認しておきましょう。

Hypertext Preprocessor

HP

# Chapter-3 オプション構成のインストール

**この章では、第2章でインストールした内容に加え、PHP4のApacheモジュール、PHP3のコマンドラインバージョン、およびPHP4コマンドラインバージョンのインストールについて説明します。なお、第2章までの作業はすべて完了しているものとします。**

**本章における作業は、特に必須ではありません。たとえばPHP4を組み込まなければ、本書で紹介した機能のうち、セッション管理などPHP4で取り入れられた新機能が使えないに過ぎません。**

## 3.1 PHP4

PHP4に関しては、国際化バージョンの開発が始まったばかりということもあり、まだ日本語関係のモジュールがそろっていません。ここではオリジナルのPHP4をインストールする手順を紹介します。PHP4用の国際化関連アドインモジュールについては、第3部「Chapter-7 国際化（日本語）関連」を参照してください。

インストールの手順は以下の通りです。

```
root@star:/usr/local/src# tar xvzf /mnt/cdrom/arc/php-4.0.1pl2.tar.gz
root@star:/usr/local/src# cd php-4.0.1pl2
root@star:/usr/local/src/php-4.0.1pl2# ./configure ¥
  --with-pgsql --enable-track-vars --with-apxs=/usr/local/apache/bin/apxs ¥
  --enable-versioning --without-gd --enable-trans-sid
root@star:/usr/local/src/php-4.0.1pl2# make
root@star:/usr/local/src/php-4.0.1pl2# make install
```

次に、**/usr/local/apache/conf/httpd.conf**にある

```
#AddType application/x-httpd-php .php
```

という行のコメントをはずして有効にします。PHP4におけるスクリプトの拡張子は**.php**となります。最後にApacheを再起動します。

```
root@star:~# /usr/local/apache/bin/apachectl restart
/usr/local/apache/bin/apachectl restart: httpd restarted
```

PHPのサンプルを表示することによりPHPの動作を確認します。前項で作成した**/usr/local/apache/htdocs/phpinfo.php3**を**phpinfo.php**にコピーしてアクセスするだけです。

```
root@star:/usr/local/apache/htdocs# cp phpinfo.php3 phpinfo.php
```

ブラウザからこのスクリプトを呼び出してみます。

```
http://localhost/phpinfo.php
```

**図4-3**の画面が表示されれば、インストールは成功です。

**図4-3** phpinfo()

PHP4には**php.ini**という設定ファイルが存在します（存在しない場合はデフォルトの動作をします）。PHP4のデフォルトの動作を変更したい場合は、以下のように設定ファイルを作成しておいてください。

```
root@star:~# cp /usr/local/src/php-4.0.1pl2/php.ini-dist ¥
 /usr/local/lib/php.ini
```

**php.ini**についての内容は、第3部「4.2　PHPオプションおよび情報」を参照してください。

## 3.2 PHP3コマンドラインバージョン

第1部のコラム「コマンドライン版PHP」でも紹介しましたが、PHPのコマンドラインバージョンはCGIプログラムとして動作させることができます。またApacheとは独立したバイナリのため、たとえばシェルスクリプトのような使い方をしたり、Cron[*2]と組み合わせて定期的に作業を行わせるなど、Web以外のアプリケーションにも使えます。インストール方法は以下の通りです。

*2 UNIXにおけるジョブスケジューラ。

```
root@star:~# cd /usr/local/src/php-3.0.15-i18n-ja
root@star:/usr/local/src/php-3.0.15-i18n-ja# make distclean
```

make distcleanは、前回指定したconfigure環境（ここではApacheモジュールとして構築するための情報）をクリアするコマンドです。

```
root@star:/usr/local/src/php-3.0.15-i18n-ja#  ./configure ¥
    --with-pgsql --enable-track-vars --enable-i18n --enable-mbregex ¥
    --enable-force-cgi-redirect --without-gd
root@star:/usr/local/src/php-3.0.15-i18n-ja# make
root@star:/usr/local/src/php-3.0.15-i18n-ja# cp php /usr/local/bin/php3
```

これでインストールは終了です。ここでは後述のPHP4とコンフリクトしないように、php3というコマンド名でインストールしました。では動作確認を行ってみましょう。

```
hotta@star:~$ echo '<? echo date("Y/m/d H:i:s "),"¥n"; ?>' | php3
X-Powered-By: PHP/3.0.15
Content-type: text/html

2000/08/03  17:46:49
```

正しく現在の時刻が表示されたでしょうか？　このように、コマンドラインバージョンはフィルタとしても使用できます。

## 3.3 PHP4コマンドラインバージョン

PHP4のコマンドラインバージョンの特徴は、前項のPHP3とほとんど同じです。こちらはphp4というコマンドでインストールしてみましょう。

```
root@star:~# cd /usr/local/src/php-4.0.1pl2/
root@star:/usr/local/src/php-4.0.1pl2# make distclean
root@star:/usr/local/src/php-4.0.1pl2# ./configure ¥
       --with-pgsql --enable-track-vars --enable-force-cgi-redirect ¥
       --without-gd
root@star:/usr/local/src/php-4.0.1pl2# make
root@star:/usr/local/src/php-4.0.1pl2# make install
root@star:/usr/local/src/php-4.0.1pl2# cd /usr/local/bin/
root@star:/usr/local/bin#mv php php4
```

これでインストールは終了です。では動作確認を行ってみましょう。

```
hotta@star:~$ echo '<? echo date("Y/m/d H:i:s "),"¥n"; ?>' | php4
X-Powered-By: PHP/4.0.1pl2
Content-type: text/html

2000/08/03  17:46:49
```

正しく現在の時刻が表示されたでしょうか？　出力はPHP3とほとんど変わりませんが、HTTPヘッダを見るとphp4が動いていることがわかります。