CGIの裏側見せます

Apache mod_cgi.c と秘密の部屋

小山浩之

oyama@cpan.org

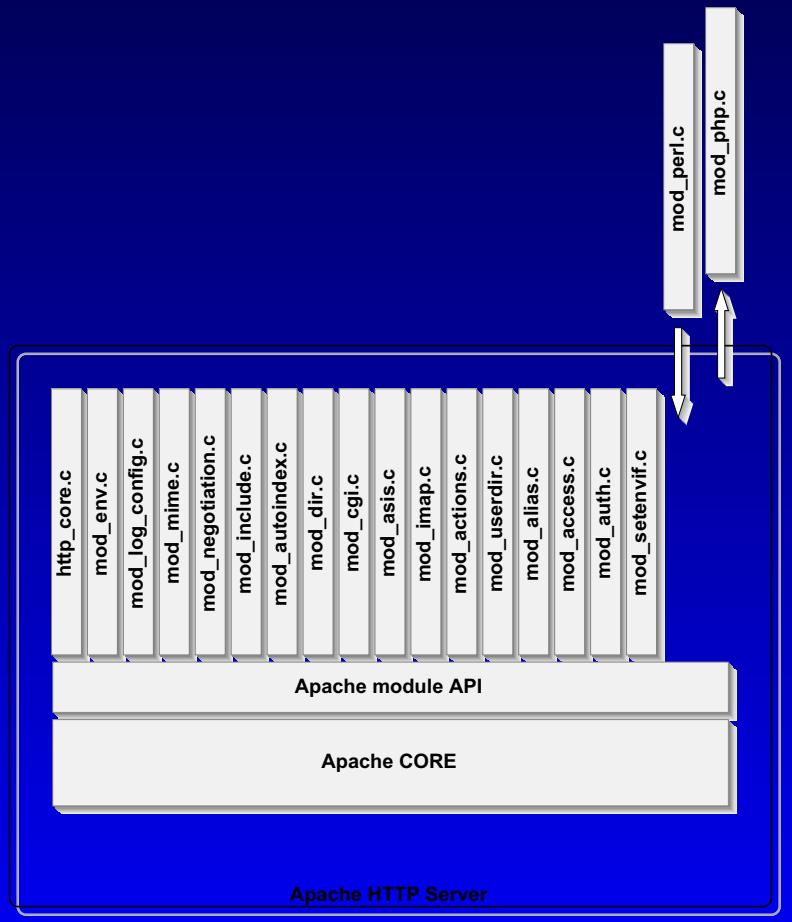
Shibuya Perl Mongers

おしながき

- Apache の構造
- CGI の動作概略
- 入力の流れ
- 出力の流れ
- 小ネタ

Apache の構造

- 機能は個別の module で実装
- module 無しの Apache は激しく役立たず



CGIの動作

- URIに対応した実行ファイルをpipeで開く
- リクエストの情報は環境変数で引き渡す
- リクエストBodyをpipeで渡す
- 実行結果をpipeで受け取る
- レスポンスヘッダを解釈した上でブラウザに送信する

この動きを実装したmod_cgi.cは600行足らずの小さなモジュール。

入力 - その1

- cgi-script ハンドラへのリクエスト (AddHandler cgi-script .cgi)
- ファイルの確認
- subprocess_env にリクエスト情報をセット (あとで環境変数になる)
- pipe を作成して fork する

入力 - その2

- subprocess_env に CGI 固有のリクエスト情報をセット (あとで環境変数になる)
- 環境変数を設定 (というか作成)
- スクリプトのあるディレクトリに chdir
- STDERR を error log に繋ぐ (dup2)
- プログラムを実行 (execle, execve)
- Socket からリクエスト Body を読み出し pipe に入力

設定される環境変数 - その1

- リクエストヘッダのを直接設定
  CONTENT_TYPE, CONTENT_LENGTH
- その他のリクエストヘッダは
  HTTP_*
- Apache 内部情報より
  SERVER_SIGNATURE, SERVER_SOFTWARE,
  SERVER_NAME, SERVER_ROOT,
  REMOTE_HOST, REMOTE_ADDR,
  DOCUMENT_ROOT, SERVER_ADMIN,
  SCRIPT_FILENAME, REMOTE_PORT,
  REMOTE_USER, AUTH_TYPE,
  REMOTE_IDENT,
  REDIRECT_QUERY_STRING,
  REDIRECT_URL

設定される環境変数 - その2

- CGI用
  GATEWAY_INTERFACE = "CGI/1.1",
  SERVER_PROTOCOL, REQUEST_METHOD,
  QUERY_STRING, REMOTE_URI,
  SCRIPT_NAME, PATH_INFO,
  PATH_TRANSLATED

POSTされたデータの流れ

- Socketから8192バイトずつreadしてpipeに投入
  → HUGE_STRING_LENの値に依存
- read し終わったらflush
- プログラムのSTDINを閉じる

Socketからread しpipeに送信する間300秒通信が途絶えるとタイムアウトする。

→ DEFAULT_TIMEOUTもしくはTimeoutディレクティブの設定値に依存

出力 - その1

- レスポンスヘッダをスキャン
- Location ヘッダを受け取った場合は REDIRECT して終了
- HTTP レスポンスヘッダをクライアントに送信
- プログラムの STDOUT をクライアントに送信
- プログラムの STDOUT を閉じる
- プログラムの STDERR を読み飛ばす
- プログラムの STDERR を閉じる

拾われる Response Header

- 特別扱い
  `Set-Cookie, Content-type, Status, Location, Content-Length, Transfer-Encoding, Last-Modified, Set-Cookie`

これらを除くヘッダはすべてそのまま出力される。

出力したデータの流れ

- プログラムの出力をバッファリング無し & ノンブロックで 8192 バイトずつ受信
  → IOBUFSIZE の定義に依存
- ブラウザへの Socket を flush
- ブラウザへの Socket に送信

この間プログラムの出力が 300 秒滞った場合
Timeout
→ DEFAULT_TIMEOUT もしくは Timeout ディレクティブの設定値に依存。

クライアントが切断した場合は送信処理を中断。

Apacheで設定可能な要素

- LimitRequestLine → リクエスト行の最大長 (デフォルト 8190)
- LimitRequestFieldsize → リクエストヘッダの最大長 (デフォルト 8190)
- LimitRequestFields → リクエストヘッダの最大数 (デフォルト 100)
- LimitRequestBody → リクエストボディの最大長 (デフォルト 無制限)
- RLimitCPU → 利用可能な CPU 時間の上限
- RLimitMEM → 利用可能なメモリの上限

小ネタ - その1

- POST されたデータの受信中にプログラムが STDIN を閉じると?
- 送信中にプログラムが固まると?
- 送信中にブラウザが切断すると?
- ブラウザが STOP ボタンを押したのを検知したい

小ネタ - その1

- POST されたデータの受信中にプログラムが STDIN を閉じると?
  → Apache は残ったデータを読み捨てる。
- 送信中にプログラムが固まると?
  → 300 秒後にタイムアウトして終了。
- 送信中にブラウザが切断すると?
  → 送信処理を中止
- ブラウザが STOP ボタンを押したのを検知したい
  → pipe が切れるので、`$SIG{PIPE}` にシグナルハンドラを登録する。

小ネタ - その2

- "Location: /path/to/file"は不正でしょ？
- 客のCGIスクリプトが馬鹿メモリ喰いで困る

小ネタ - その2

- "Location: /path/to/file"は不正でしょ？
  → いいえ、Apache さんは内部リダイレクトとして良い塩梅に処理する。
- 客の CGI スクリプトが馬鹿メモリ喰いで困る
  → RLimitMEM などでリソースを制限する。

小ネタ - その3

- CGI 側で特定の機能を潰したい。→ 潰したい関数を上書きするライブラリを用意し、SetEnv LD_PRELOAD /path/to/libdisable.so

```
$ cat disable_socket.c
#include <stdio.h>
int socket(int domain, int type, int protocol)
{
    fprintf(stderr, "Ha Ha Ha! cannot use socke!\n");
    return -1;
}

$ gcc -shared -o libdisable.so disable_socket.c
```

！未承諾広告！

2003年8月頃
Apacheモジュールプログラミングをネタに本が出ます。

(ごめんなさい、Perlじゃなくて C です...)