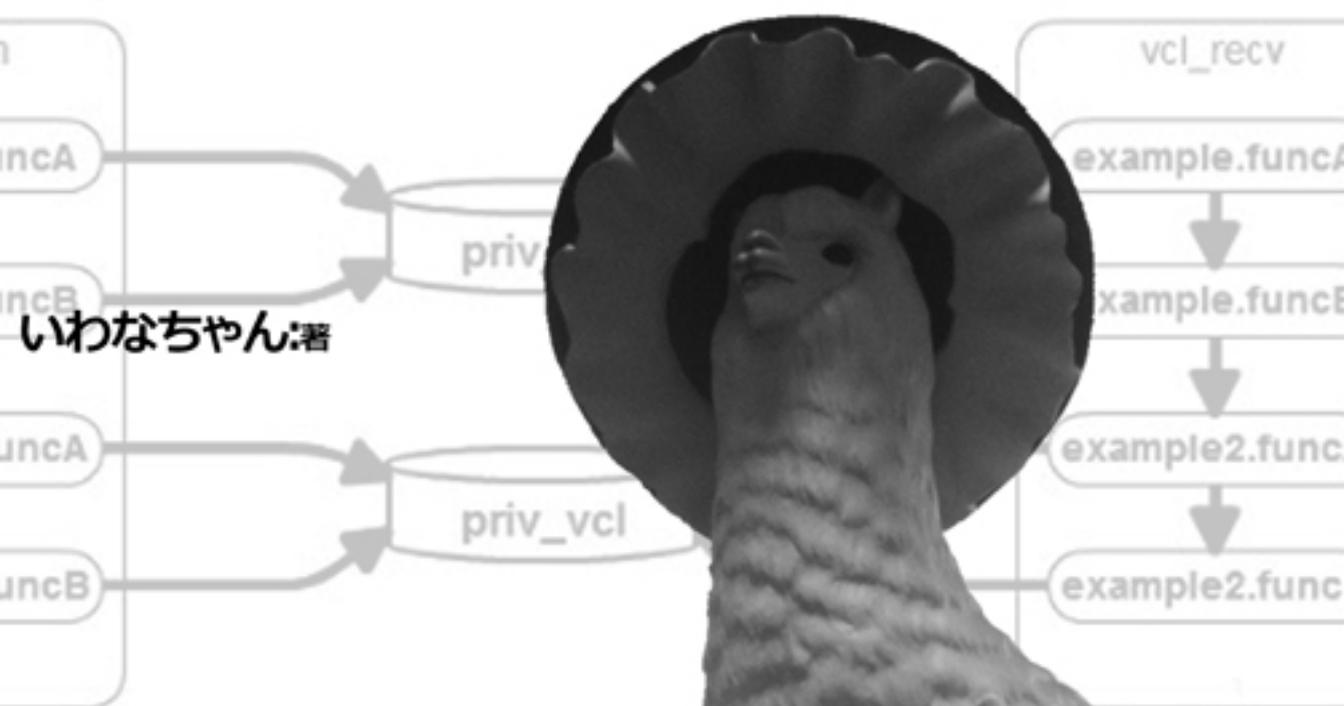


reverse proxy

Varnish Cache

inline-C/VMOD
GUIDEBOOK

inline-C/VMOD ガイドブック



ポイントはそれぞれのVMOD 毎に
で設定した値をfetchする
のはpriv_callです。を参照



目次

本書の対象者.....	2
Varnish の高度なカスタマイズ.....	2
インライン C を使う際に必要な基礎知識.....	4
VCL のダンプ.....	4
Varnish のソースを読む.....	7
インライン C で変数操作を行う方法.....	12
組み込み関数を使う.....	18
インライン C で外部の共有ライブラリを利用する場合.....	22
VMOD を使う際に必要な基礎知識.....	24
外部の VMOD を使ってみる(vmod_example).....	24
vmod に関数を追加してみる.....	25
vmod_example.vcc (vmod.vcc).....	26
vmod_example.c (vmod.c).....	27
VMOD で使用可能な変数の型.....	28
セッションワークスペースの使い方.....	34
プライベートポインタの使い方.....	36
VMOD で外部の共有ライブラリを使う.....	39
VMOD で正規表現を使う.....	40
VMOD のデバッグの仕方(varnishtest).....	41
おしらせ.....	42
あとがき.....	42

またインライン C で扱った関数と同様に、必ず第一引数は `sp` となります。

```
int vmod_len(struct sess *sp, const char *p){
    return(strlen(p));
}
```

第二引数以降は受け取る変数によって変わってきます。

`init_function` については後述します。

VMOD で使用可能な変数の型

VMOD で利用可能な型はほぼ VCL と同じです。しかし一部非推奨だったり特殊な型があったりします。

VCLでの型名	Cでの型名	戻値	引数	備考
BACKEND	struct director *	○	○	記述なし
BOOL	unsigned	○	○	非推奨
DURATION	double	○	○	
INT	int	○	○	
IP	struct sockaddr_storage *	△	○	非推奨
STRING	const char *	○	○	
TIME	double	○	○	
HEADER	enum gethdr_e, const char *	×	○	非推奨
REAL	double	○	○	
STRING_LIST	const char *, ...	×	○	
PRIV_VCL	struct vmod_priv *	△	○	
PRIV_CALL	struct vmod_priv *	△	○	
VOID	void	○	×	

戻値が△なのは、書き込みできる変数が存在せず、使い道がさほど思いつかなかった為です。また非推奨のものは公式のドキュメントに記載されていたものです。それぞれの変数を引数・戻値に持つ簡単な関数を作って解説します。

BACKEND

バックエンドの情報を格納しています。引数、戻り値共に指定可能です。

```
■VCC
Function BACKEND tbackend(BACKEND)

■C
struct director * vmod_tbackend(struct sess *sp, struct director *p){
    return p;
}

■VCL
set req.backend = example.tbackend(req.backend);
```

もし `director` のメンバーにアクセスする場合は以下のヘッダを `include` する必要があります。

```
#include "bin/varnishd/cache.h"
#include "bin/varnishd/cache_backend.h"
```

例として指定したバックエンドが正常の場合はそれを返却し、そうでない場合は現在選択されているバックエンドを返却します。

```
■ VCC
Function BACKEND gethealthydirector(BACKEND)

■ C
struct director * vmod_gethealthydirector(
struct sess *sp, struct director *p){
    if(VDI_Healthy(p, sp)){
        return p;
    }
    return sp->director;
}

■ VCL
set req.backend = example.gethealthydirector(client_2);
```

`VDI_Healthy` は、バックエンドの状態を返却します。
利用するには以下のヘッダを `include` する必要があります。

```
#include "bin/varnishd/cache.h"
```

BOOL

真偽が入っている型です。

```
■ VCC
Function BOOL tbool(BOOL)

■ C
unsigned vmod_tbool(struct sess *sp, unsigned p){
    return p;
}

■ VCL
set req.esi = example.tbool(req.esi);
```

inline-C/VMOD ガイドブック

本書の対象者

Varnishの高度なカスタマイズ

インラインCを使う際に必要な基礎知識

VCLのダンプ

Varnishのソースを読む

インラインCで変数操作を行う方法

組み込み関数を使う

インラインCで外部の共有ライブラリを利用する場合

VMODを使う際に必要な基礎知識

外部のVMODを使ってみる(vmod_example)

vmodに関数を追加してみる

vmod_example.vcc (vmod.vcc)

vmod_example.c (vmod.c)

VMODで使用可能な変数の型

セッションワークスペースの使い方

プライベートポインタの使い方

VMODで外部の共有ライブラリを使う

VMODで正規表現を使う

VMODのデバッグの仕方(varnishstest)

おしらせ

あとがき