Front End Server用モジュール プログラマーズガイド

http://www.nsl.tuis.ac.jp/
mailto:iseki@rsch.tuis.ac.jp

０．免責＆ライセンス

このソフトウェアは全くの無保証です．このソフトウェアの使用・改造・再配布に伴い発生する問題に関して，作者は一切の責任を負いません．全て自己責任でご使用ください．

このプログラムは商用以外ではフリーです．再配布・ライブラリの再利用（改造を含む）は自由ですが，再配布する場合は，配布されたものを完全な形で再配布してください．ライブラリの再利用では Copyrightを明記してください．

商用に利用する（このソフトウェアに対して対価を要求する）場合はご連絡ください．

1.モジュール.

FEServer はメイン処理を外部モジュールとして作成し,起動時に読み込むことができる.従って,様々な機能を外部モジュールとして定義して置けば，一つのプログラムで色々な機能を実現することが可能となる．また，似たようなプログラムを開発する場合もFEServerの通信機能を再利用することが可能であり，プログラムの作成効率が高くなる．

２．記述する関数.

モジュールを作成する場合は以下の５個の関数を必ず作成しなければならない.

```
int  init_main(void)
int  init_process(int)
int  term_process(int)
int  fe_server(int csofd, int nsofd, SSL* cssl, SSL* sssl, char* msg, int cc)
int  fe_client(int nsofd, int csofd, SSL* sssl, SSL* cssl, char* msg, int cc)
```

2-1. int init_main(void)

fesvr の起動時に行なう前処理（モジュールの初期化）を記述する．syslogのオープン，設定ファイルの読み込み等はここで行なう．戻り値としては，syslogのタイプを返す．ログを取りたくない場合は NO_SYSLOG を返し，エラーを通知する場合は負の数を返す（fesvrは停止する）．この関数中では以下の大域変数が参照できる．これらの変数は fesvr本体でも使用するので，勝手に書き換えてはいけない.

unsigned char* ServerIPaddr_num

長さ4byteのバイナリデータ．中継するサーバのIPアドレスがバイナリで格納されている.

unsigned char* MyIPaddr_num

長さ8byteのバイナリデータ．自分自身のIPアドレス4Byteに続いて，ネットマスク4Byteがバイナリで格納されている.

unsigned char* MyNetworkaddr_num

長さ4byteのバイナリデータ．自分自身のネットワークアドレスがバイナリで格納されている.

char* ServerIPaddr
　中継するサーバのIPアドレスが文字列として格納されている．
char* MyIPaddr
　自分自身のIPアドレスとネットマスクが“[IP]/[mask]”の形式で文字列として格納されている．
char* MyNetworkaddr
　自分自身のネットワークアドレスが文字列として格納されている．

2-2. int　init_process(int sock)

　クライアントからの接続要求を受けた直後に起動される関数．deamonモード（デフォルト）では，チャイルドプロセスを初期化するために使用される．アクセス制御や，これから開始される処理の準備（モジュールの開始処理）を行なう．sock は接続してきたクライアントへのソケット．戻り値は，処理が正常に終了した場合は TRUE，失敗した場合は FALSE を返す．FALSE を返すと fesvr は停止する．因みにTRUEとFALSEはマクロで以下のように定義されている．

```
#define FALSE 0
#define TRUE  (!FALSE)
```

　この関数以降，前出の大域変数に加えて以下の大域変数も参照できるようになる．

unsigned char* ClientIPaddr_num
　長さ4byteのバイナリデータ．クライアントのIPアドレスがバイナリで格納されている．
char* ClientIPaddr
　クライアントのIPアドレスが文字列として格納されている．
char* ClientName
　クライアントのIPアドレスから逆引きしたクライアント名が文字列として格納されている．逆引きに失敗した場合は　NULL である．

2-3. int　term_process(int)

　クライアントとの一つのセッションが終了する前に呼び出される関数．deamonモード（デフォルト）では，チャイルドプロセスの終了直前に呼び出される．モジュールの終了処理を記述する．sock はクライアントへのソケット．戻り値は，処理が正常に終了した場合は TRUE，失敗した場合は FALSE を返す．FALSE を返すと fesvr は停止する．

2-4. int　fe_server(int csofd, int nsofd, SSL* cssl, SSL* sssl, char* msg, int cc)

　メイン処理の一つ．サーバからクライアントへの通信をフックしている．csofd はサーバへのソケット，nsofd はクライアントへのソケットである．msg は長さ BUFSZ の配列であり，サーバからクライアントへの通信内容が格納されている．cc は msg中のデータ長を表す．
　戻り値はクライアントへ転送したデータのバイト数であるが，fesvr では厳密にはチェックしていない．ただし，負の数を返した場合は中継処理を中断する(チャイルドプロセスは終了する)．

2-5. int　fe_client(int nsofd, int csofd, SSL* sssl, SSL* cssl, char* msg, int cc)

　もう一つのメイン処理．クライアントからサーバへの通信をフックしている．nsofd はクライアントへのソケット，csofd はサーバへのソケットである．msg は長さ BUFSZ の配列であり，クライアントからサーバへの通信内容が格納されている．cc は msg中のデータ長を表す．
　戻り値はサーバへ転送したデータのバイト数であるが，fesvr では厳密にはチェックしていない．ただし，負の数を返した場合は中継処理を中断する(チャイルドプロセスは終了する)．

３．作成手順.

3-1.ヘッダファイル

まずヘッダファイルを記述する．ヘッダファイルでは最初に必ず feplg.h をインクルードする．feplg.h をインクルードしておけば，必ず記述すべき５つの関数のプロトタイプ宣言は行なわなくてもよい.

3-2. メインプログラム

メインプログラムは，最初に3-1で作成したヘッダをインクルードする．プログラム中では５つの必須関数を必ず記述しなければならない．feplg_nop.c を参考にするとよい.

3-3. ライブラリ.

FESeerver はTUISライブラリを使用している．TUISライブラリには基本ライブラリと拡張ライブラリがあり，モジュール中ではこれらのライブラリの全てを使用することができる．各ライブラリの説明は Lib/Doc/*, xLib/Doc/* を参照すること．これらのドキュメントは make コマンドによりライブラリを生成すると，自動的に生成される.

3-4. コンパイル・リンク方法.

コンパイルではOSとCPUの種別を指定し，-fPIC オプション付きでコンパイルしなければならない．-DEBUG オプションをつけるとデバッグモードでコンパイルされる.

例）gcc XXXX.c -fPIC -I../Lib -I../xLib -DLinux -DIntel -DEBUG -c -O2

リンクでは TUISライブラリと，必要ならその他のライブラリ(SSL, gdbmなど)も結合しなければならない．または，モジュールは共有ライブラリとして生成される必要があるため -shared オプション付きでリンクしなければならない.

例）gcc XXXX.o -shared -L../Lib -lbasic -L../xLib -lextend -lm -O2 -o XXXX.so

以上，詳しくは，Makefile を参照すること.