# C 言語演習支援システム プロトタイプ
# 簡易マニュアル Ver 0.1

知能メディア工学研究室
M2 西 輝之

平成 19 年 10 月 30 日

## 1　はじめに

このマニュアルは開発段階で書いた簡易的なもので所謂叩き台です。各ソースコード中の関数など細かい仕様については書いていません。

バージョン管理を行った上での改良は自由ですが、変更を加えた場合はマニュアルに適宜反映させる、バージョン管理でログを残すなどして下さい。

## 2　プロトタイプの概要

Emacs 上で C 言語のソースコードをコンパイルや実行を行い、出力されたエラーメッセージを XML に変換。その後指定された CGI へと送信します。テスト用 CGI は受け取った XML を XHTML に変換し、テーブル形式で提示します。

## 3　ファイル構成

### 3.1　prot_clang ディレクトリ

プロトタイプシステムのメイン

**analysis_client.el** エラーメッセージを解析モジュールへ送信 (elisp)

**analysis_serv.rb** コンパイラ、デバッガ共用のエラーメッセージの解析モジュール (ruby)

**compile.sh** コンパイルを行うシェルスクリプト

**compile_err.txt** コンパイルエラーのサンプル

**functions.el** コンパイル用シェルスクリプト、http 通信モジュールの実行 (elisp)

**http_bridge.rb** http 通信モジュール (ruby)

**readme.txt** 説明書みたいな物

**run_err.txt** 実行エラーのサンプル

### 3.2 sample_src ディレクトリ

C 言語サンプルソース群

***.c** サンプルソース群

### 3.3 test_cgi

テスト用 CGI

**functions.js** Ajax もどき用 JavaScript

**index.html** 表示用 html

**result.txt** エラーメッセージ格納用

**style.css** スタイルシート

**test.xsl** xml から xhtml に変換する XSLT スタイルシート

**test_cgi.cgi** エラーメッセージの保存、xhtml の変換を行う CGI

### 3.4 manual

使い方や仕様が記述されたマニュアル（本ファイルを含む）

```
*.tex
*.pdf
*.dvi
```
など

## 4 実行方法

1. エラーメッセージ解析モジュールと、http 通信モジュールを起動します。コンソール一つにつきモジュール 1 つしか起動出来ないので 2 つのコンソールを用意して下さい。(emacs shell 上で起動しても構いません)

   ```
   ./analysis_serv.rb
   ./http_client.rb
   ```

2. emacs で拡張用 elisp 関数を評価します。emacs で、

   - analysis_client.el
   - functions.el

   の 2 つの elisp ファイルを読み込み、1 つの関数が定義されている括弧の外で C-x C-e で関数を評価できます。

例）
```
(defun analysis-client ()

…中略…

) □←ここで C-x C-e
```

詳しくは elisp の書籍や Web サイトを参考のこと。

3. 対象とする C 言語のソースファイルを読み込み、拡張した elisp の関数を用いてコンパイルを行います。この時、対象とするファイルを開いているバッファに切り替えた（カレントバッファにした）状態でコンパイルを行って下さい。

   例）ファイルを開いた emacs 上で
   ```
   M-x compile-files
   ```

   この関数を実行することで、コンパイル、エラーメッセージの解析、テスト用 CGI への送信が完了します。各項目が正しく実行されたか確認する場合には、モジュールを実行させているコンソールに出力されたログを確認して下さい。

# 5 主な仕様

## 5.1 解析モジュール (analysis_serv.rb)

このモジュールはソケット通信によるプロセス間通信を用いて実現されています。通信を行う際は、ポート 7120 番に接続すること。不都合がある場合はポート番号を変更するとよいでしょう。

受け取ったメッセージがコンパイラから物か、デバッガからの物かを判別するために簡易的なプロトコルを用いています。

- 動作モードの送信
- メッセージの送信
- モジュールが非同期で動作するため、メッセージの送信が完了した事を通知

まず初めに、どのモードで動作させるかをモジュールへと送信します。

**コンパイラーのメッセージの場合** mode_compiler

**デバッガーのメッセージの場合** mode_debugger

次に改行を送信した後に実際のエラーメッセージを送信します。送信が完了したら改行に続き”EOF”の 3 文字を送信する。これがメッセージの送信が完了した事の合図となります。

今回はプロトタイプのためプロトコルを簡易化しましたが、厳密な動作にはモジュールがその動作を正しく完了したかなどを通知する仕組みが必要となると思われます。

### 5.2 http 通信モジュール (http_client.rb)

このモジュールはソケット通信によるプロセス間通信を用いて実現されています。通信を行う際は、ポート 7300 番に接続すること。不都合がある場合はポート番号を変更するとよいでしょう。

このモジュールも非同期で動作するため、簡易的なプロトコルを用いています。解析したメッセージ（xml）の送信が完了したら、解析モジュール (analysis_serv.rb) 同様改行に続き”EOF” の 3 文字を送信します。

このモジュールでも、モジュールの状態を通知するようなプロトコルが必要になると思われます。

## 6 全体の流れ

ファイル単位での、全体的な流れを図 1 に示します。

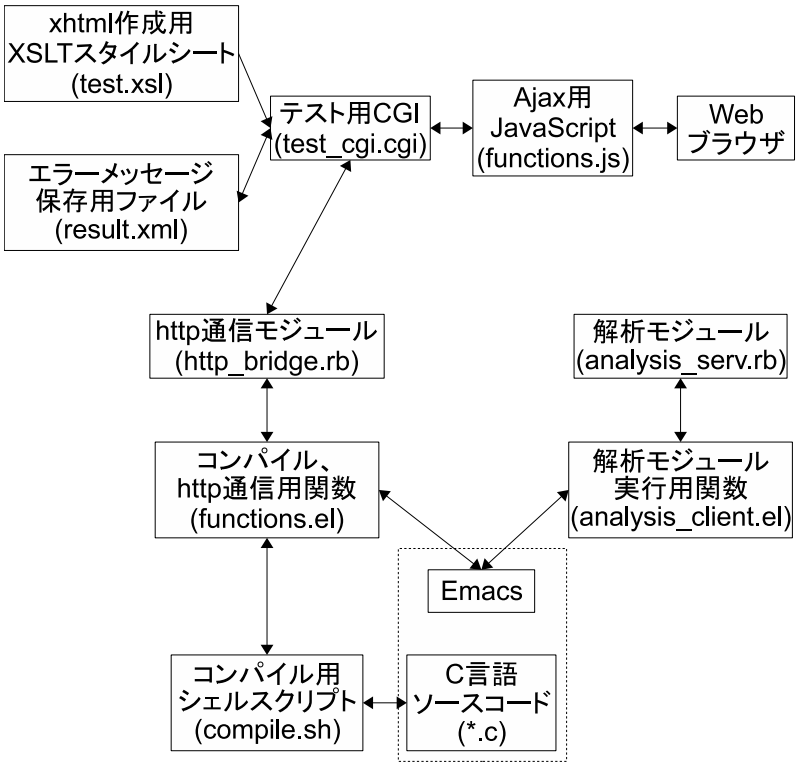


図 1: 各ファイル間の関係