

通信プロトコル実装実習

Armadillo800EVAによるCGI

2022 年 月 日 ()

中国職業能力開発大学校

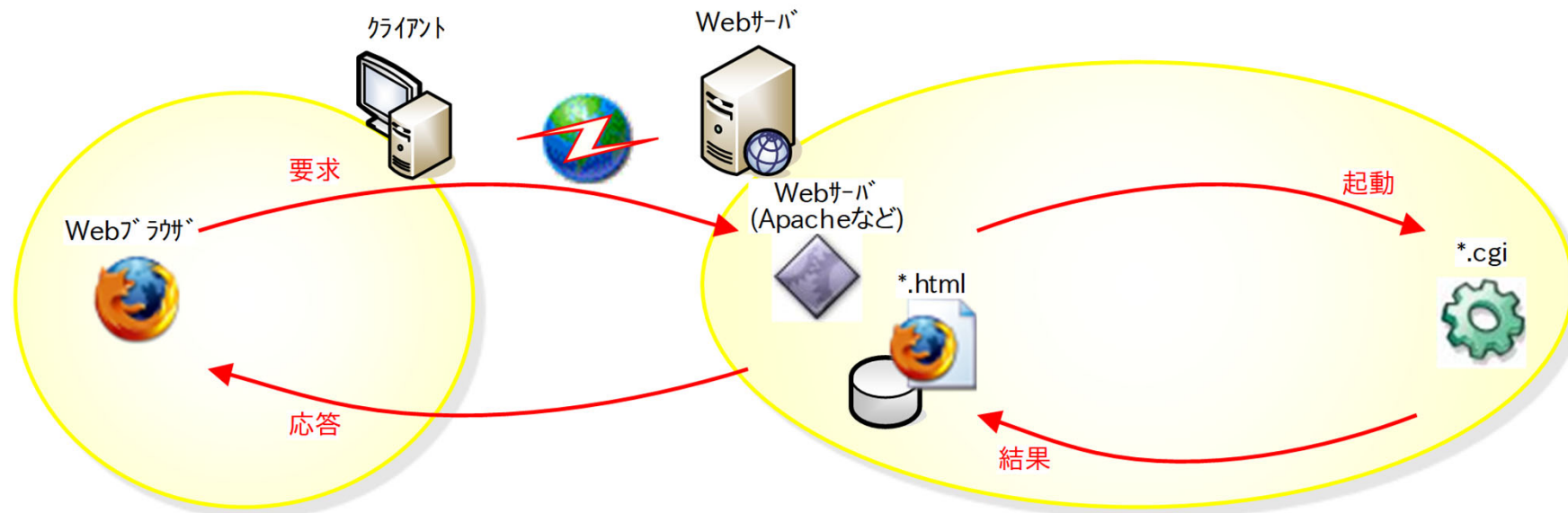
1号棟4階 1405室

CGIの仕組み

CGI

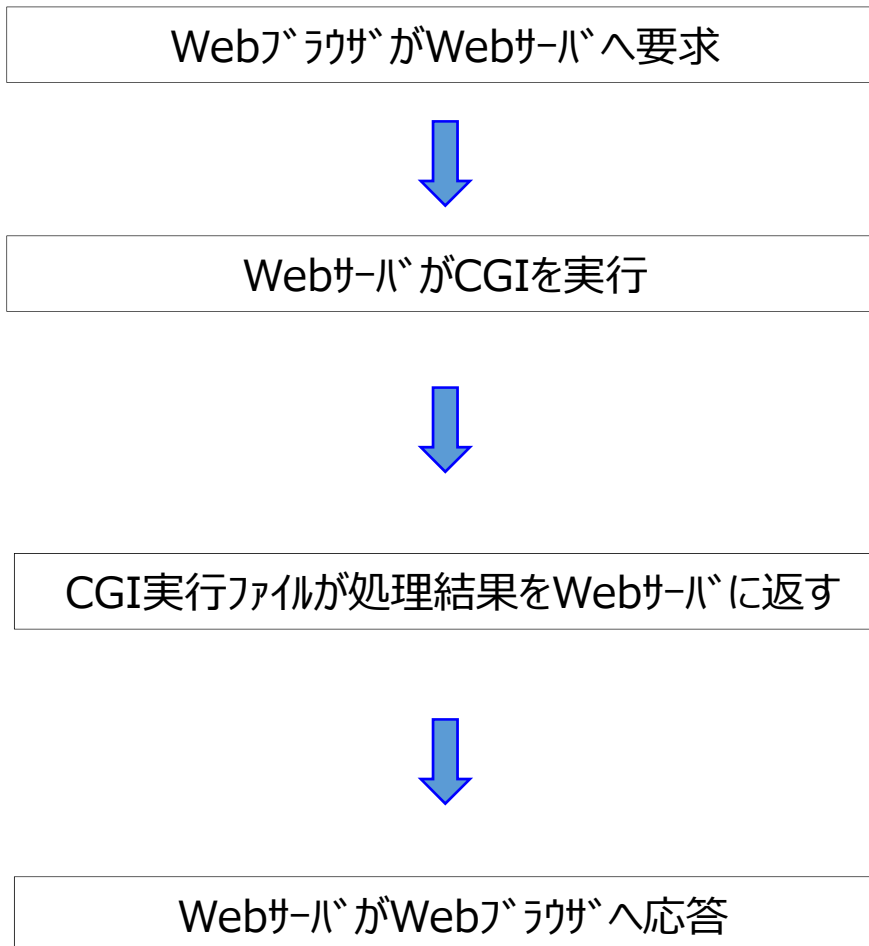
📌 CGIはCommon GateWay Interfaceの略称

- 📌 Webサーバー(Apacheなど)が、Webブラウザ(IEやFireFoxなど)からの要求に応じて、プログラムを起動する仕組みのこと



- 📌 Webサーバーは、Webブラウザからの要求に対してCGIを起動
- 📌 CGIからの処理結果をHTMLでWebブラウザに応答
- 📌 CGIは、Webサーバーから起動される実行ファイル(例として*.cgiや*.pl)なので、開発言語に依存しない
- 📌 全て標準入出力関数による文字列処理
- 📌 エンタープライズ系のサイトではPerl言語がCGIの開発言語として主流
- 📌 組込み分野では、実行ファイルサイズを小さくするため C言語 で開発されていることが多い

CGIの処理の流れ



- Webブラウザは特に「CGIを起動する」とは意識しない
- `` や `` や `<form action="〜">` タグで記述されたアドレス(URL)を単にWebサーバへ要求するだけ
- Webサーバは受け取ったURLを解釈
- CGI実行ファイルかどうか判断
- たいいてい拡張子`*.cgi`や`*.pl`
- Webサーバの設定によって異なる
- Webサーバをレンタルする場合、そのWebサーバの利用規約によって、CGIをサポートしていない場合もある
- CGI実行ファイルは、`printf`関数などの標準出力によって処理結果をHTML形式で書き出す
- CGIソース次第で、複数のファイルを読み込んで連結したり、カウンター値を演算してその結果を返すことができる

CGIへの引数

🔗 リンクタグを利用する方法

🔗 ` ~ `

🔗 リンクをクリックすると cgilink.cgi が起動し、第1引数data1で abc、第2引数data2で 123 を渡す

🔗 cgiled1a.html

🔗 GETメソッドを利用する方法

🔗 `<form action="http://10.22.91.114/cgi-bin/cgigetmethod.cgi" method="GET"> ~ </form>`

🔗 引数名と引数を格納する仕組みが必要

🔗 文字数に制限がある

🔗 URLに引数が表示される

🔗 cgiled1bget

🔗 POSTメソッドを利用する方法

🔗 `<form action="http://10.22.91.114/cgi-bin/cgigetmethod.cgi" method="POST"> ~ </form>`

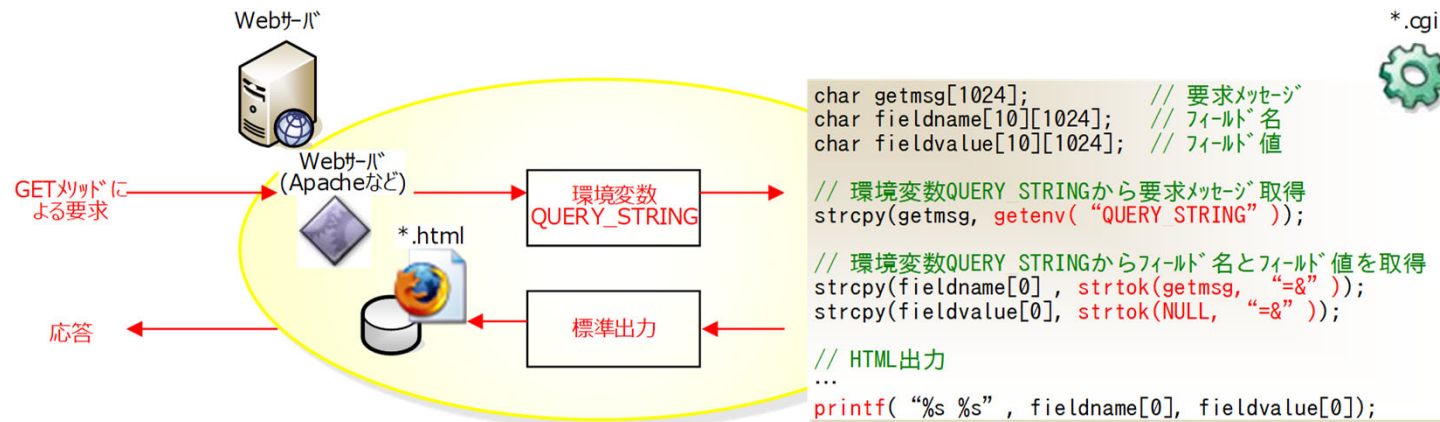
🔗 引数名と引数を格納する仕組みが必要

🔗 文字数に制限はない

🔗 URLに引数は表示されない

🔗 cgiled1cpost

GETメソッド



GETメソッドの引数

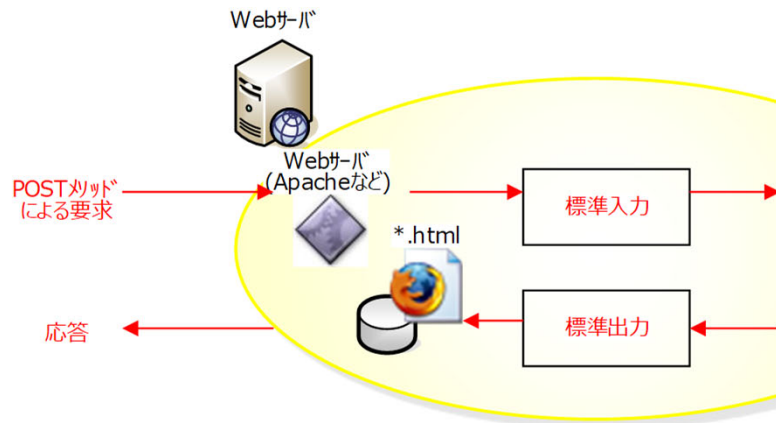
- 環境変数 **QUERY_STRING** に格納
- *.c は `getenv("QUERY_STRING")` で取得
- `getenv` の戻り値が文字列へのポインタなので、`strcpy()` でバッファへコピー
- QUERY_STRING** は “変数名=データ&変数名=データ” の文字列ポインタ
- `strtok` で 変数名 と データ を切り分ける

2バイト文字（日本語など）をGETメソッドの引数にした場合

- shift-JIS や UTF-8 などの文字コードによって変わる
- 日本語 (UTF-8) : “%E6%97%A5%E6%9C%AC%E8%AA%9E”
- 日本語 (shift-JIS) : “%93%FA%96%7B%8C%EA”
- 文字化けしないようにするには、***.html の保存形式** と `<meta charset= “UTF-8” >` と **Webブラウザの文字コード** を一致させる
- 適切な エンコード処理 および デコード処理 を実装する
- エンコードマニアックス

<http://www.encodemaniac.com/>

POSTメソッド



```

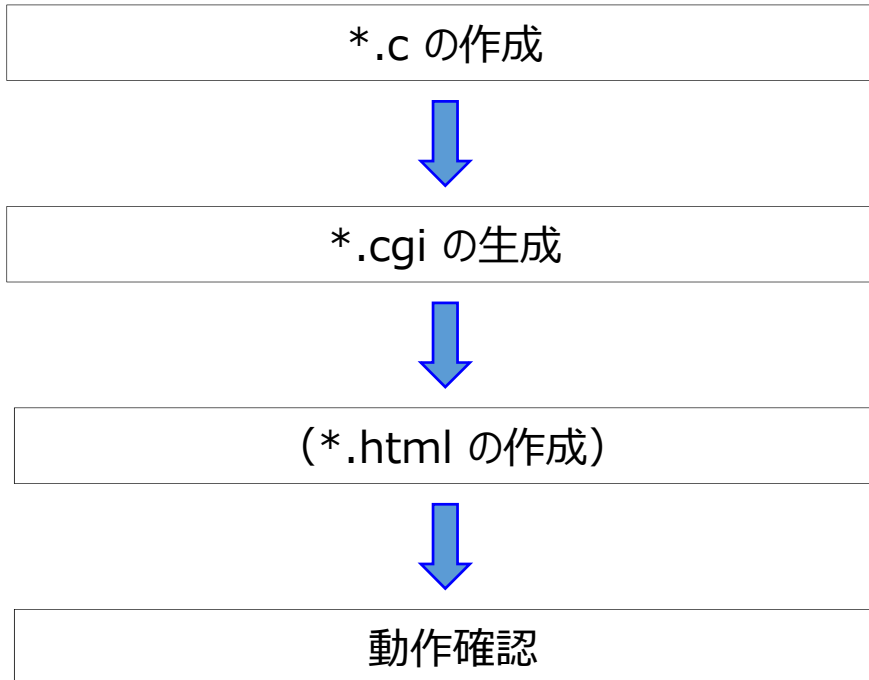
*.cgi
// 要求メッセージサイズ取得
len = getenv( "CONTENT_LENGTH" );
// POST要求メッセージサイズの領域確保
pbuf = (char *)malloc(len+1);
// 要求メッセージ取得
scanf( "%s", pbuf );
strcpy(postmsg, pbuf);
// フィールド名とフィールド値を取得
strcpy(fieldname[0], strtok(postmsg, "&"));
strcpy(fieldvalue[0], strtok(NULL, "&"));
// HTML出力
...
printf( "%s %s", fieldname[0], fieldvalue[0] );
  
```

POSTメソッドの引数

- 📌 *.c は scanf() で取得
- 📌 getenv("CONTENT_LENGTH") でサイズ取得
- 📌 バッファ領域は malloc で取得したサイズ +1 を確保
- 📌 POSTメソッドの引数は “変数名=データ&変数名=データ” の文字列ポインタ
- 📌 strtok で 変数名 と データ を切り分ける
- 📌 2バイト文字は、GETメソッドと同様に処理をする
- 📌 エンコードマニアクス

<http://www.encodemanix.com/>

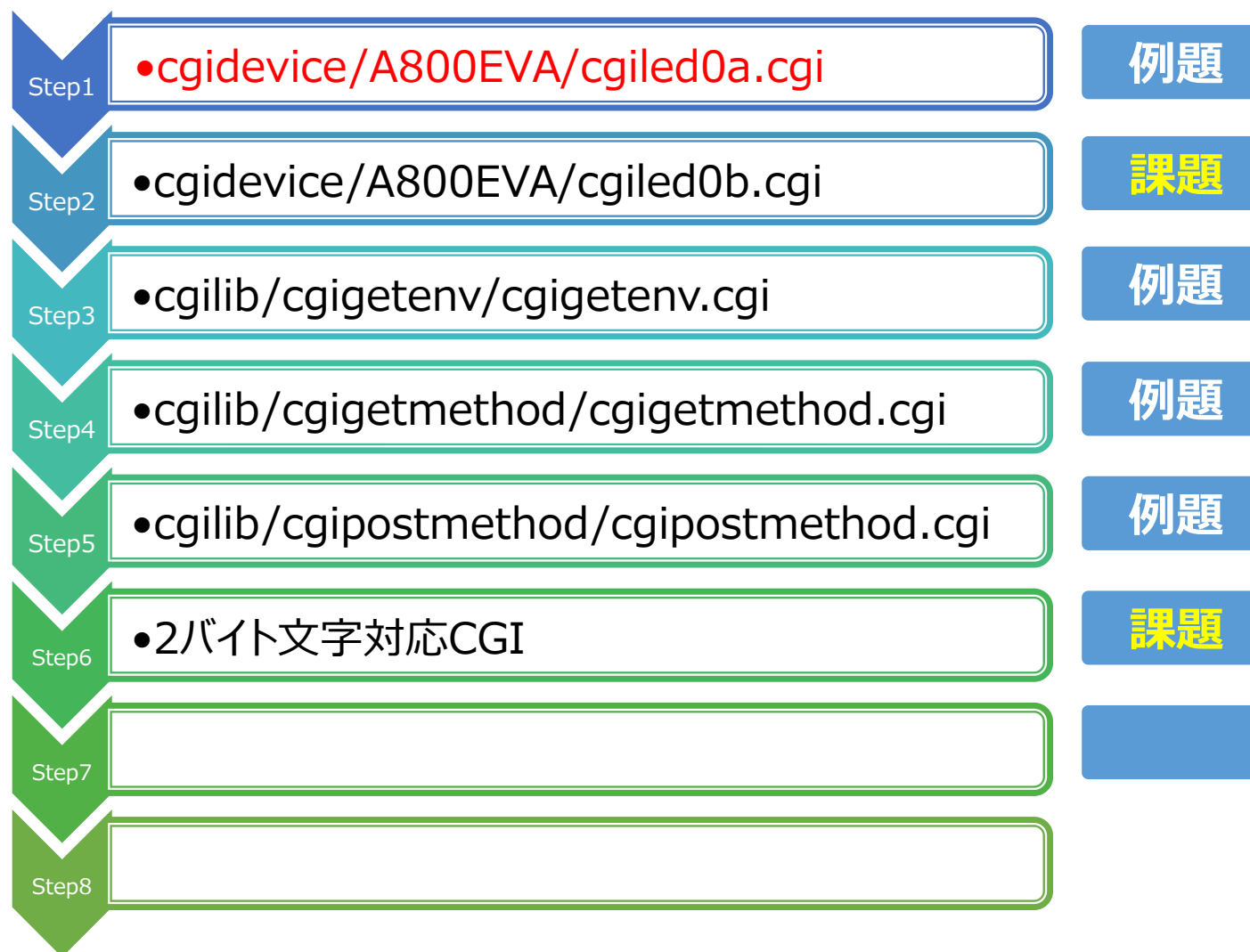
CGIの開発の流れ



- いきなり *.c でHTML を記述するのが大変な場合、あらかじめWebブラウザからの応答で想定する *.html を作成した方が良い
- 作成した *.html を printf 文 に編集し直す
- make ファイルの編集
- CGI の動作確認をするため、入力テスト用の *.html を作成
- ない場合もある
- CSiDE などを利用しデバッグする

CGIプログラミング

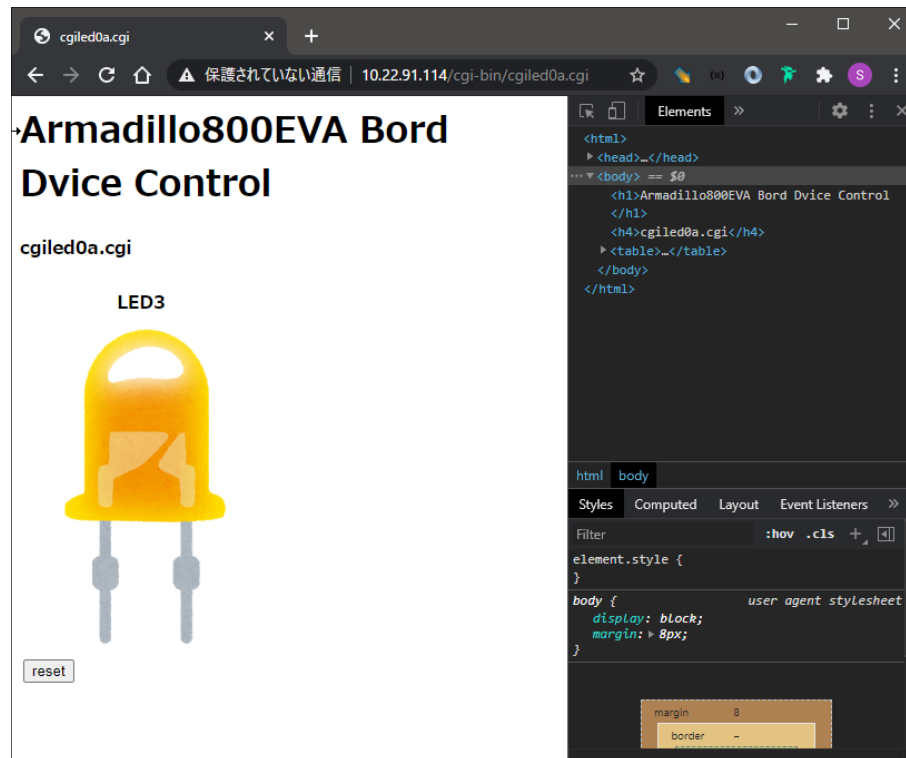
流れ



[例題] cgiled0a.cgi

レイアウト例

<http://10.22.91.114/cgi-bin/cgiled0a.cgi>



機能

- [/var/www/html/image/ledy_on.png](#) の表示
- reload ボタンの配置
- table で構成

cgiled0a.html

- *.cgi のたたき台なので作成しなくてよい
- 以下の HTML を printf文 で記述していく
- printf文では “ を ¥ ” と記述しなければならない

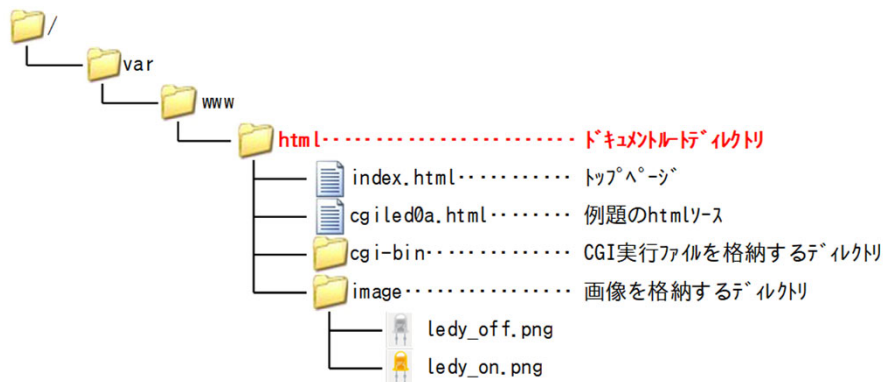
```
<!DOCTYPE html>
<html lang="jp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>cgiled0a.html</title>
</head>

<body>
  <h1>Armadillo800EVA Bord Dvice Control</h1>
  <h4>cgiled0a.html</h4>
  <table>
    <tr><th>LED3</th></tr>
    <tr><td>
      
    </td></tr>
    <tr><td><button type="reset"
onclick="location.reload()">reset</button></td></tr>
  </table>
</body>
</html>
```

作業ディレクトリの作成

📁 画像の保存場所

- 📁 **html** ディレクトリ下に **image** ディレクトリ作成
- 📁 /work/linux/nfsroot/var/www/html/image へ *.png をコピー



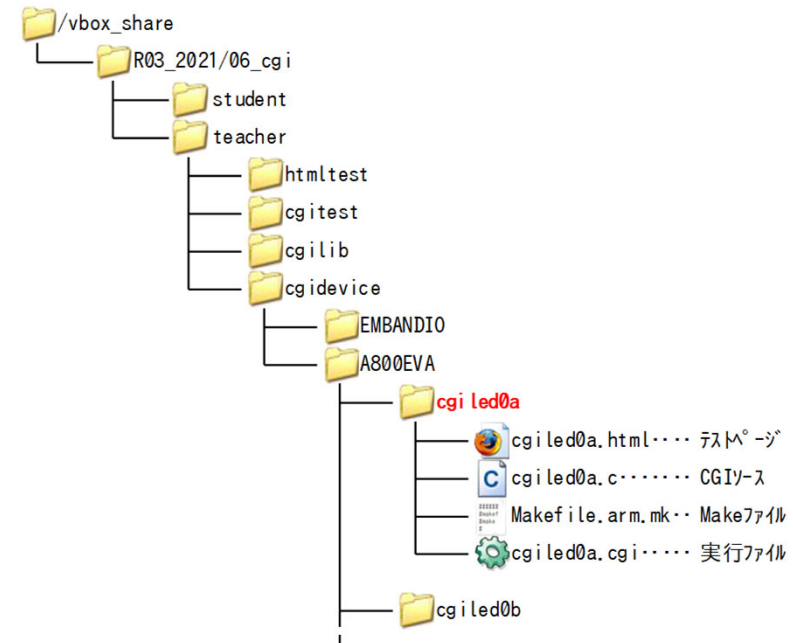
📁 パーMISSIONの確認

```
root@debian:/var/www/html# ls -al
(省略)
drwxrwxrwx 2 samba samba 4096 Jan  7 09:23 image
-rwxrw-rw- 1 samba samba 236 Jan  1 2000 index.html

root@debian:/var/www/html# ls -al ./image/
(省略)
-rwxrw-rw- 1 samba samba 20631 Aug 23 2019 ledy_off.png
-rwxrw-rw- 1 samba samba 38445 Aug 23 2019 ledy_on.png
```

📁 *.c と *.mk の作成場所

- 📁 Windowsで作業しても良い



```
user@vbubuntu:~$ cd
/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a
user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ ls -la
合計 32
drwxrwx--- 1 root vboxsf 4096 1月 11 19:02 .
drwxrwx--- 1 root vboxsf 4096 1月 10 14:25 ..
-rwxrwx--- 1 root vboxsf 3319 1月 10 13:12 Makefile.arm.mk
-rwxrwx--- 1 root vboxsf 1010 1月 11 18:42 cgiled0a.c
-rwxrwx--- 1 root vboxsf 7292 1月 11 19:02 cgiled0a.cgi
-rwxrwx--- 1 root vboxsf 526 1月 10 13:20 cgiled0a.html
-rwxrwx--- 1 root vboxsf 3844 1月 11 19:02 cgiled0a.o
```

[例題] cgiled0a.c

```
//=====
// URL: http://10.22.91.114/cgi-bin/cgiled0a.cgi
//
//                                     2021/1/10 S.Fujimoto
//=====
#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>

int main( int argc, char **argv )
{
    printf("Content-type:text/html;charset=UTF-8¥n¥n");
    printf("<html>¥n");
    printf("<head>¥n");
    printf("<title>cgiled0a.cgi</title>¥n");
    printf("</head>¥n");

    printf("<body>¥n");
    printf("<h1>Armadillo800EVA Bord Dvice Control</h1>¥n");
    printf("<h4>cgiled0a.cgi</h4>¥n");
    printf("<table>¥n");
    printf("<th>LED3</th>¥n");
    printf("<tr><td><img src=¥\"../image/ledy_on.png¥\"></td></tr>¥n");
    printf( "¥<tr><td><button type=¥\"reset¥\" onclick=¥\"location.reload("
) ¥>reset</button></td></tr>¥n");
    printf("</table>¥n");

    printf("</body>¥n");
    printf("</html>¥n");
    return 0;
}
// End of file
```

🔗 バウンダリの ¥n¥n を忘れないこと

🔗 <table> ~ </table> タグ

🔗 <http://www.htmq.com/html/table.shtml>

🔗 <button> ~ </button> タグ

🔗 <http://www.htmq.com/html5/button.shtml>

[例題] Makefile.arm.mk

```
#####
# makefilename
#     make [-f name][option]
#
# メイク(構築・ビルド)      make -f makefilename
# 再構築(リビルド)         make -f makefilename rem
# *.oと実行ファイル削除    make -f makefilename clean
# 指定場所へのコピー      make -f makefilename install
#
#                                     2021/1/10 S.Fujimoto
#####
# サフィックスルールの適用
#   サフィックスルールを適用させオブジェクトファイル(*.o)はソースファイル(*.c)から生成させる
#   既存のサフィックスルールを削除し新たに定義する
.SUFFIXES:
.SUFFIXES: .o .c .h

#同一ディレクトリ内にサーバーとクライアントの *.c が存在するため対象を明確にする
#複数ファイルで構成する場合、SRC2 = xxx.c と SRCS = $(SRC1) $(SRC2) を追加
TARGET =  cgiLed0a
SRC1    =  cgiLed0a.c
SRCS    =  $(SRC1)

#オブジェクトファイルの指定(サフィックスルールを適用)
OBJS    =  $(SRCS:.c=.o)

#使用するコンパイラとオプションの指定
CC      =  arm-none-linux-gnueabi-gcc
CFLAGS  =  -gdwarf-2 -O0
LDFLAGS =
#LDFLAGS = -lpthread
#MYLIB   =

#暗黙ルールの設定
#   暗黙ルールを(明示的に)決めると、cファイルから.oファイルを生成することができる
#   .oのターゲットに対し、依存関係のファイルをターゲットの主ファイル名+.cと設定しコマンド適用
#   $<: 依存関係のコンポーネント(サフィックスルールのみ利用可) この場合、cファイルのこと
#   $@: ターゲット名(ルールとサフィックスルールで利用可)この場合、oファイルのこと
.c.o:
    $(CC) -c $(CFLAGS) -o $@ $<
```

```
#####
# これよりコンパイル定義を記述
#
#   makeコマンド`引数なし(オプションなし)'で実行すると、makefile上で最初に記述した
#   ルールがmake対象となる
#####
#タミターゲットの明示
.PHONY: all rem clean install

#最初のルール
all :$(TARGET).cgi

#remakeコマンド`
rem:clean $(TARGET).cgi

#cleanコマンド`
#   '-rm'の'-r'(リフン)はエラーコードを無視する(makeが途中で止まらないように)
clean:
TAB rm -f $(TARGET).cgi $(OBJS)

#ソースファイルから実行形式ファイルを作成
#   $@: ターゲット名(ルールとサフィックスルールで利用可)
#   $?: ターゲットより後で更新されたコンポーネント名
$(TARGET).cgi:$(OBJS)
TAB $(CC) -o $@ $(LDFLAGS) $?

#installコマンド`
install:
TAB cp -p $(TARGET).cgi /work/linux/nfsroot/debug/06_cgi/student
TAB cp -p $(TARGET).cgi /work/linux/nfsroot/var/www/html/cgi-bin
TAB cp -p $(TARGET).cgi /media/sf_debug_share/R03_2021/06_cgi/student
TAB cp -p $(SRC1) /media/sf_debug_share/R03_2021/06_cgi/student
TAB chmod 777 /work/linux/nfsroot/var/www/html/cgi-bin/$(TARGET).cgi
#   cp -p $(TARGET).css /work/linux/nfsroot/var/www/html
#   chmod 777 /work/linux/nfsroot/var/www/html/$(TARGET).css
# End of file
```

[例題] cgiled0a.cgi 動作確認方法

📁 作業ディレクトリへ移動

```
user@vbubuntu:~$ cd /media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a
```

📁 ビルド

```
user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ make  
-f Makefile.arm.mk rem  
rm -f cgiled0a.cgi cgiled0a.o  
arm-none-linux-gnueabi-gcc -c -gdwarf-2 -O0 -o cgiled0a.o cgiled0a.c  
arm-none-linux-gnueabi-gcc -o cgiled0a.cgi cgiled0a.o
```

📁 インストール

```
user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ sudo  
make -f Makefile.arm.mk install  
[sudo] password for user:  
cp -p cgiled0a.cgi /work/linux/nfsroot/debug/06_cgi/student  
cp -p cgiled0a.cgi /work/linux/nfsroot/var/www/html/cgi-bin  
cp -p cgiled0a.cgi /media/sf_debug_share/R03_2021/06_cgi/student  
cp -p cgiled0a.c /media/sf_debug_share/R03_2021/06_cgi/student  
chmod 777 /work/linux/nfsroot/var/www/html/cgi-bin/cgiled0a.cgi
```

【補足】 ../var/www/html/image/* .pngのコピー

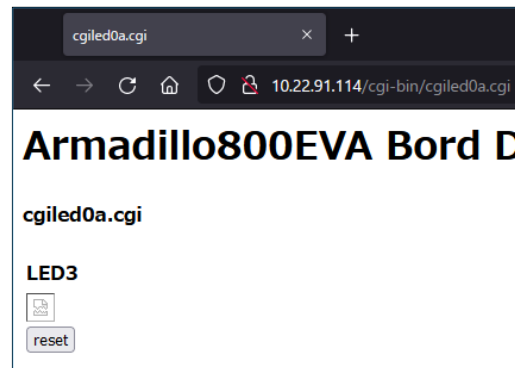
🐉 VSCode で Ubuntu20.04 に リモートSSH で接続したときのコピー作業

```
user@vbubuntu:~$ cd /media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a
user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ cp ../
../image/ledy_on.png /work/linux/nfsroot/var/www/html/image/

user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ cp ../
../image/ledy_off.png /work/linux/nfsroot/var/www/html/image/

user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ ls -
la /work/linux/nfsroot/var/www/html/image/
合計 72
drwxrwxr-x 2 user user 4096 1月 25 16:17 .
drwxrwxrwx 4 root root 4096 1月 25 15:09 ..
-rwxrwx--- 1 user user 30348 1月 25 16:17 ledy_off.png
-rwxrwx--- 1 user user 29763 1月 25 16:17 ledy_on.png
```

🐉 このままだと 画像が表示されない



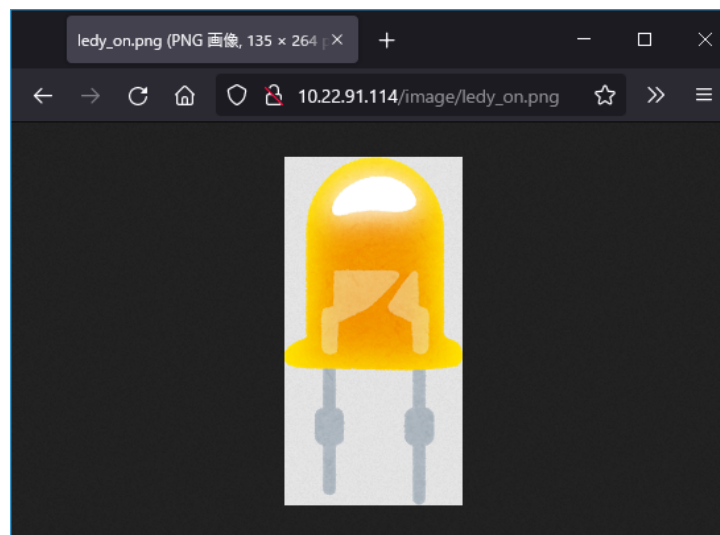
🐉 パーミッションを変更する

【補足】*.cgi 実行しても 画像 が表示されない

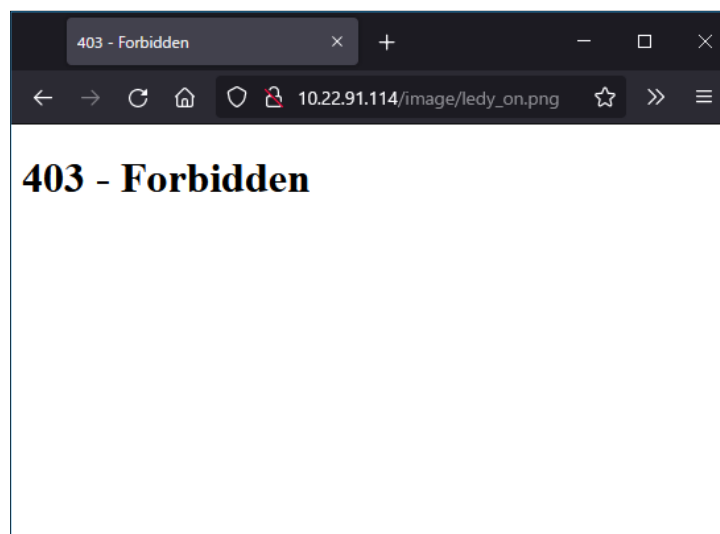
🔗 画像単独で表示できるか試す

🔗 http://10.22.91.114/image/ledy_on.png

🔗 表示したなら、*.c を見直す



🔗 表示できないならパーミッションを確認する



【補足】クロスコンパイルエラーについて

- VSCode で Ubuntu20.04 に リモートSSH で接続したときに、クロスコンパイルエラー

```
user@vbubuntu:~$ cd /media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a
user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ make
-f Makefile.arm.mk rem
arm-none-linux-gnueabi-gcc -c -gdwarf-2 -00 -o cgiled0a.o cgiled0a.c
make: arm-none-linux-gnueabi-gcc: コマンドが見つかりませんでした
make: *** [Makefile.arm.mk:40: cgiled0a.o] エラー 127
```
- 原因は環境変数 \$PATH が通っていないから
- ホームディレクトリにある .profile を確認

```
user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ cat
~/.profile
(省略)
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
export PATH=$PATH:/opt/arm-2010q1/bin
export CROSS_COMPILE=/opt/arm-2010q1/bin/arm-none-linux-gnueabi-
```
- クロスコンパイラが登録されていることを確認
- 一時的に PATH を通す作業なので、ターミナルを新規で開くたびに必要

```
user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ source ~/.profile
```
- make

```
user@vbubuntu:/media/sf_vbox_share/R03_2021/06_cgi/teacher/cgidevice/A800EVA/cgiled0a$ sudo
make -f Makefile.arm.mk install
[sudo] password for user:
cp -p cgiled0a.cgi /work/linux/nfsroot/debug/06_cgi/student
cp -p cgiled0a.cgi /work/linux/nfsroot/var/www/html/cgi-bin
cp -p cgiled0a.cgi /media/sf_debug_share/R03_2021/06_cgi/student
cp -p cgiled0a.c /media/sf_debug_share/R03_2021/06_cgi/student
chmod 777 /work/linux/nfsroot/var/www/html/cgi-bin/cgiled0a.cgi
```

[例題] cgiled0a.cgi 動作確認方法

📌 CGI の公開ディレクトリへ移動

```
root@debian:~# cd /var/www/html/cgi-bin/
```

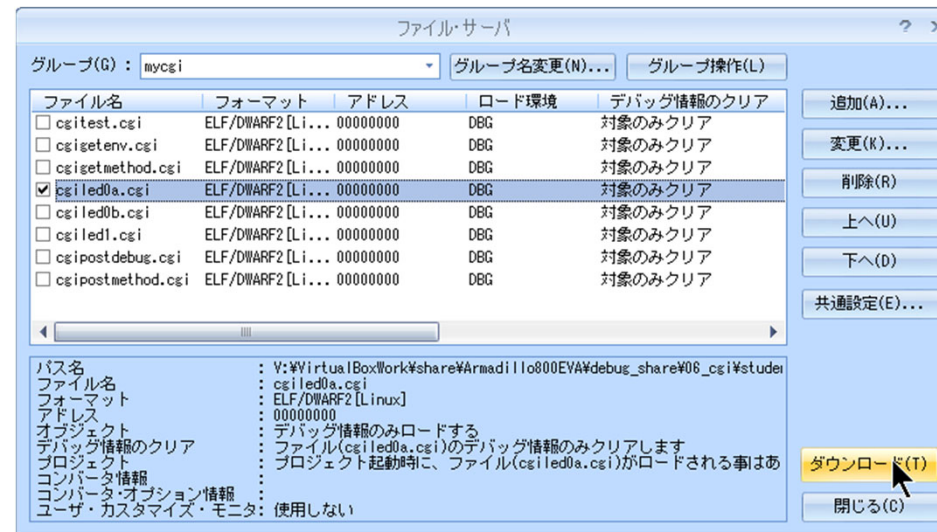
📌 パーMISSIONの確認

```
root@debian:/var/www/html/cgi-bin# ls -al
total 76
drwxrwxrwx 2 root root 4096 Jan 10 08:49 .
drwxrwxrwx 4 root root 4096 Jan 10 08:49 ..
-rwxrwxrwx 1 root 999 7292 Jan 11 10:02 cgiled0a.cgi
-rwxrwxrwx 1 root 999 6875 Jan 3 07:35 cgitest.cgi
```

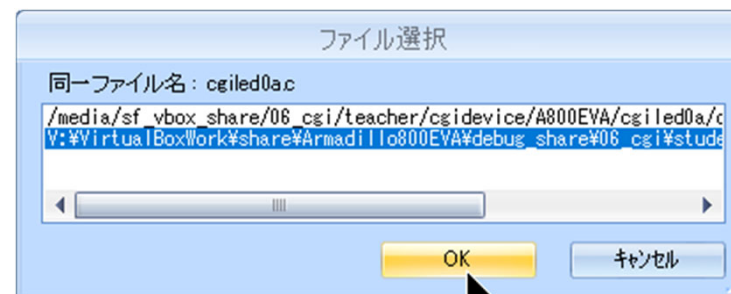
📌 メニュー[ファイル] -> [ロード]

📌 グループ名: mycgi

📌 ロードするファイル名: cgiled0a.cgi

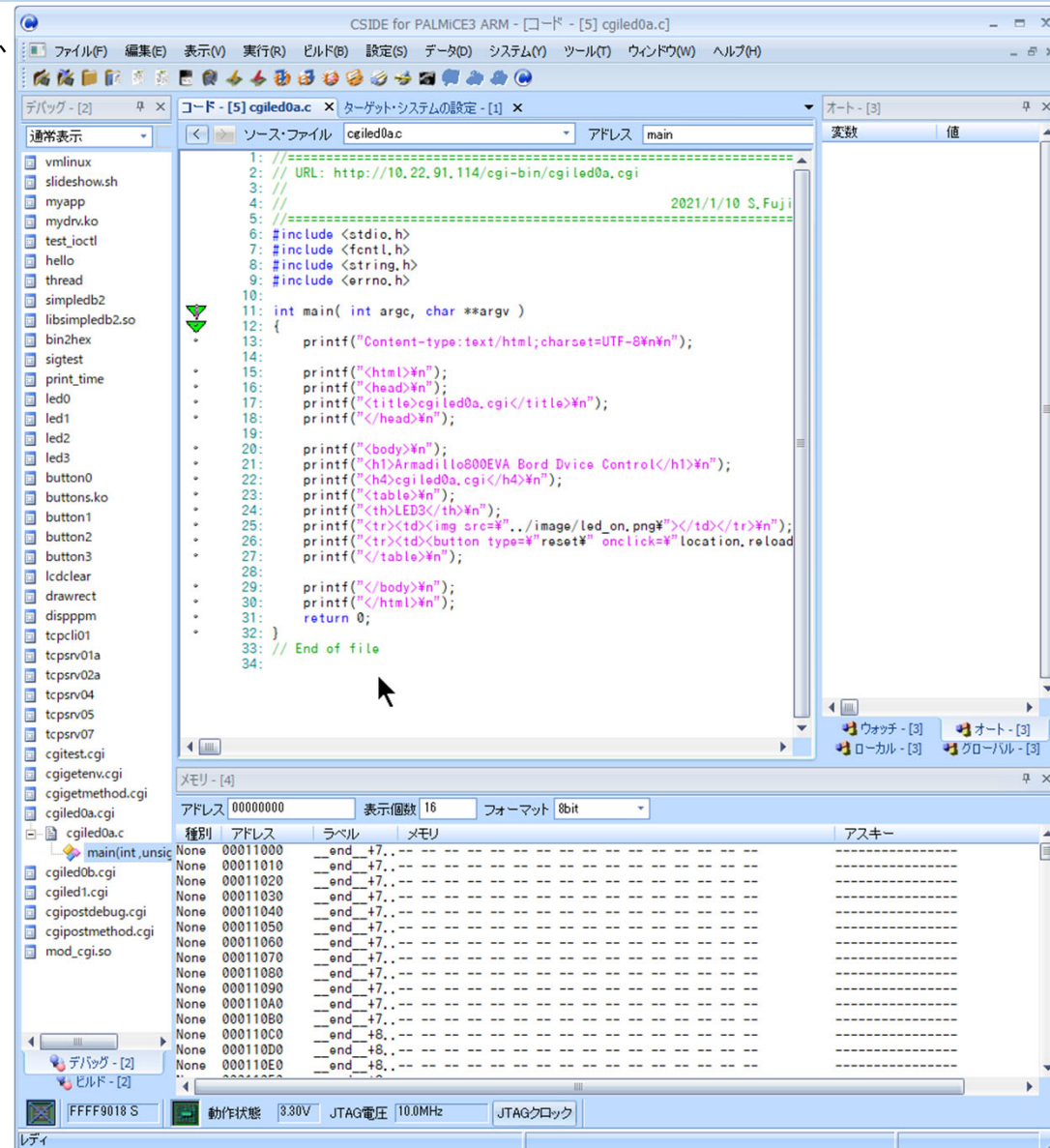


📌 もしファイルロードの選択となったら、debug_share のディレクトリ以下のソースを選択すること



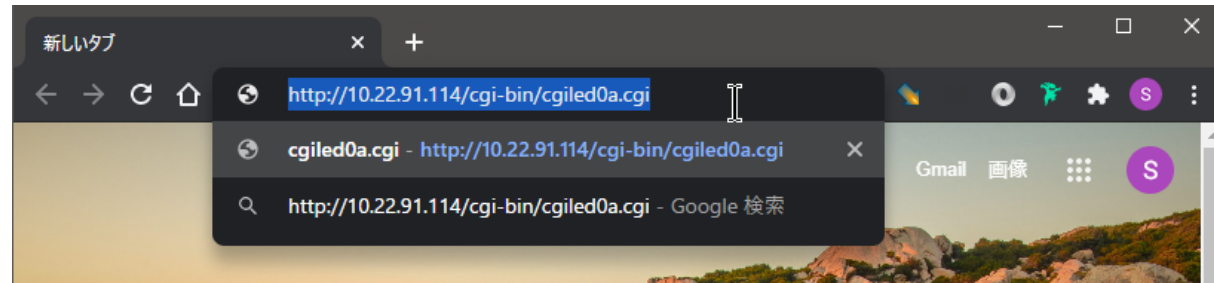
[例題] cgiled0a.cgi 動作確認方法

- 🔗 cgiled0a.cgi の表示がうまくいかない場合は、CSIDE を再起動する



[例題] cgiled0a.cgi 動作確認方法

Webブラウザ起動しアドレス欄に入力



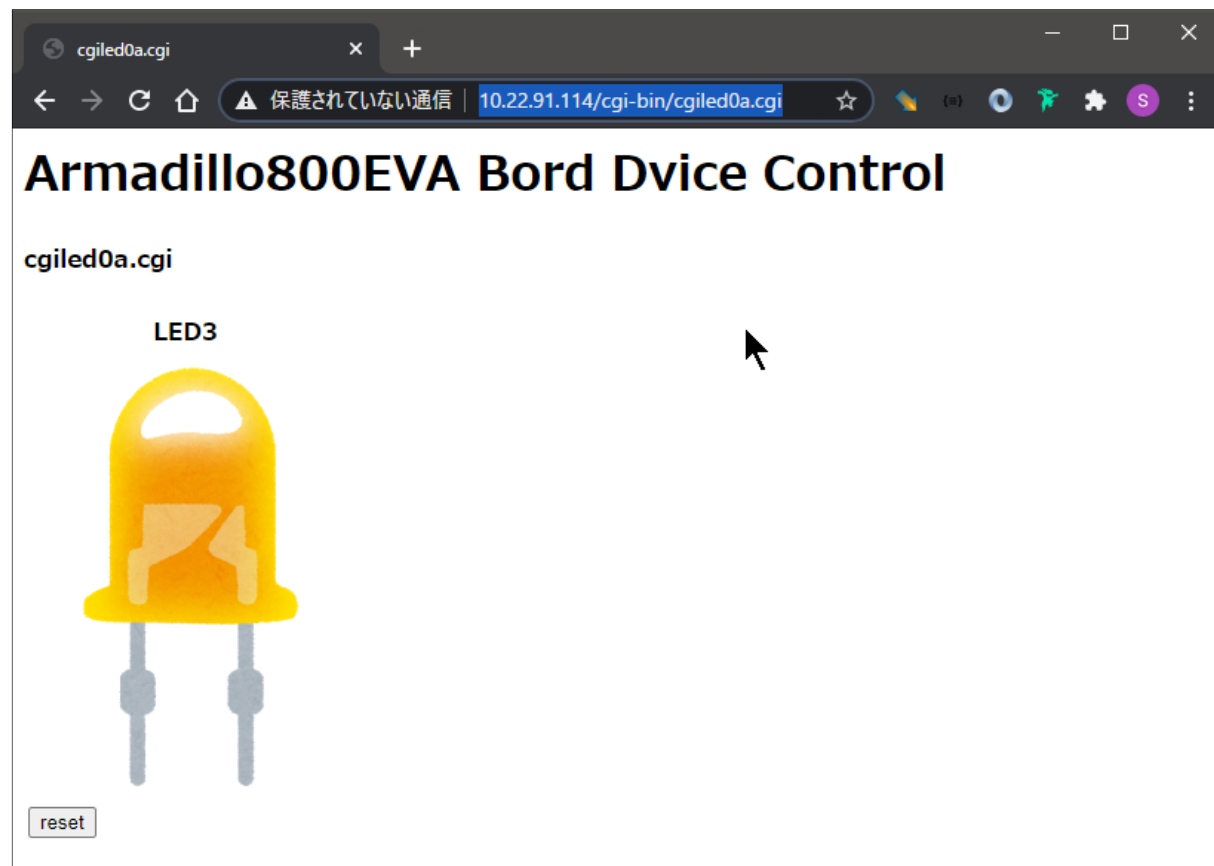
Armadillo800EVA が動作していないと反応しない

くるくる回転していること

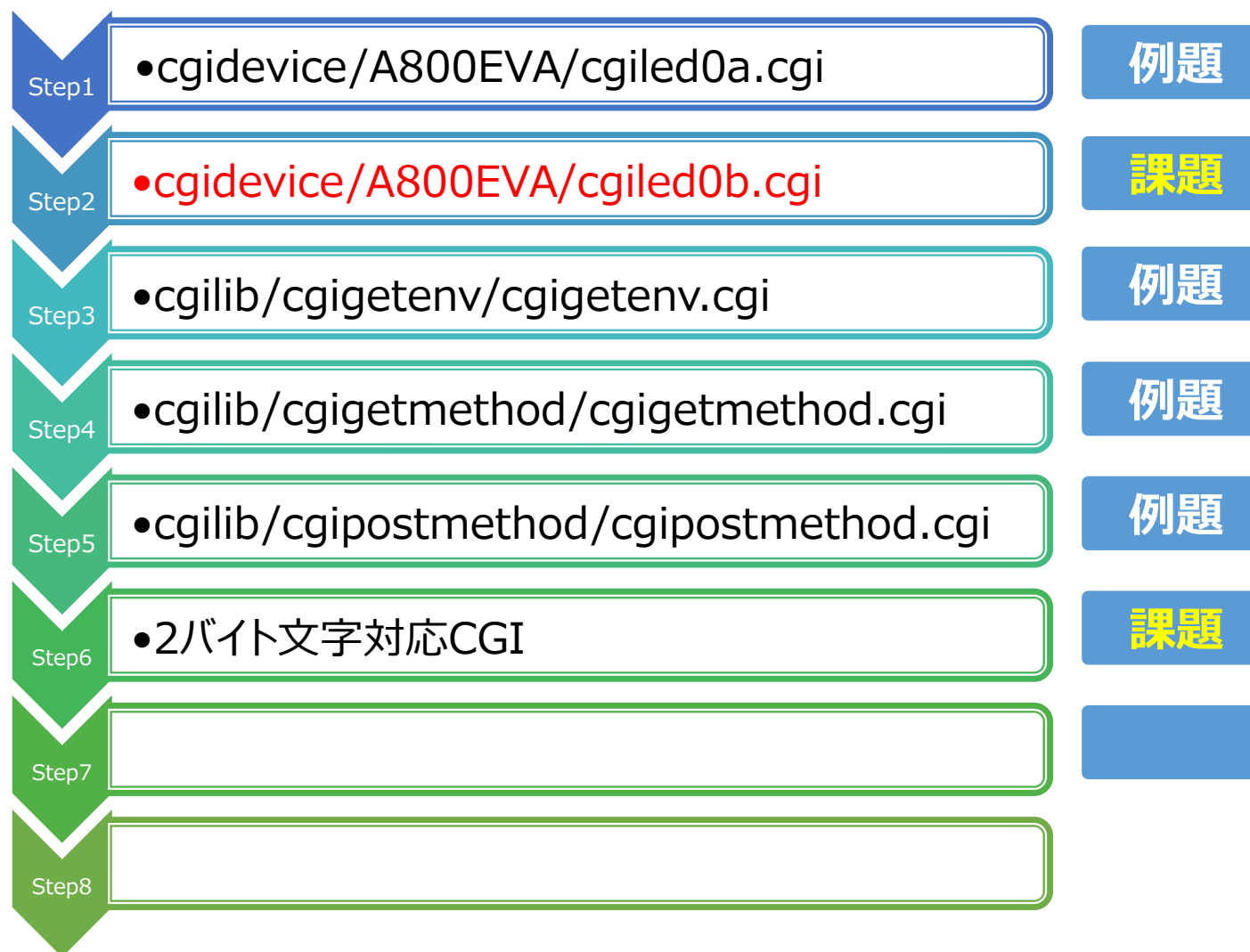


reset ボタンをクリックすると reload (再更新) する

再更新しても何も変化はない



流れ



[課題] cgiled0b.cgi

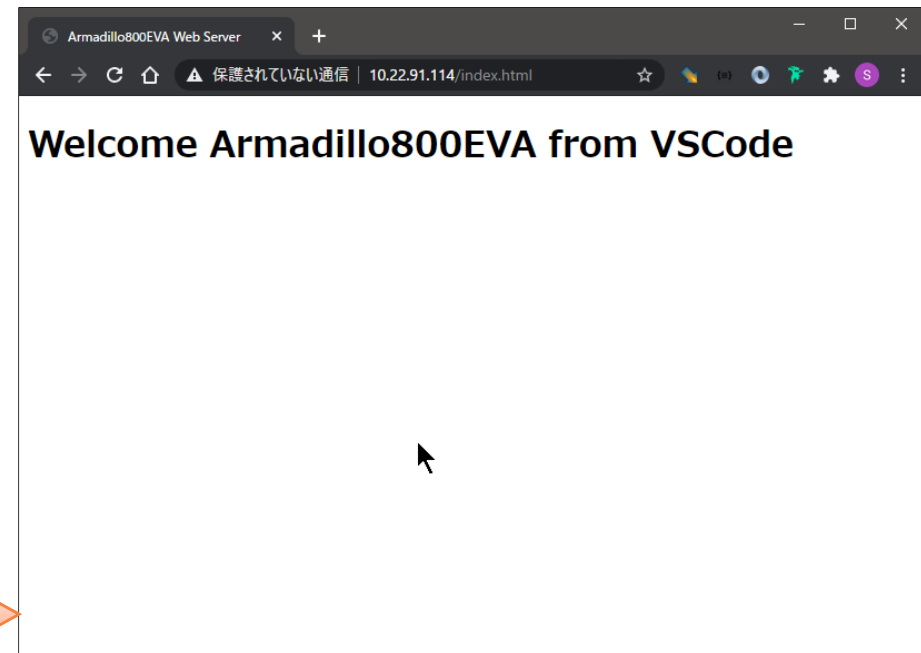
レイアウト

<http://10.22.91.114/cgi-bin/cgiled0b.cgi>



ボタンをクリックすると
画面遷移

index.html ボタンをクリックした結果



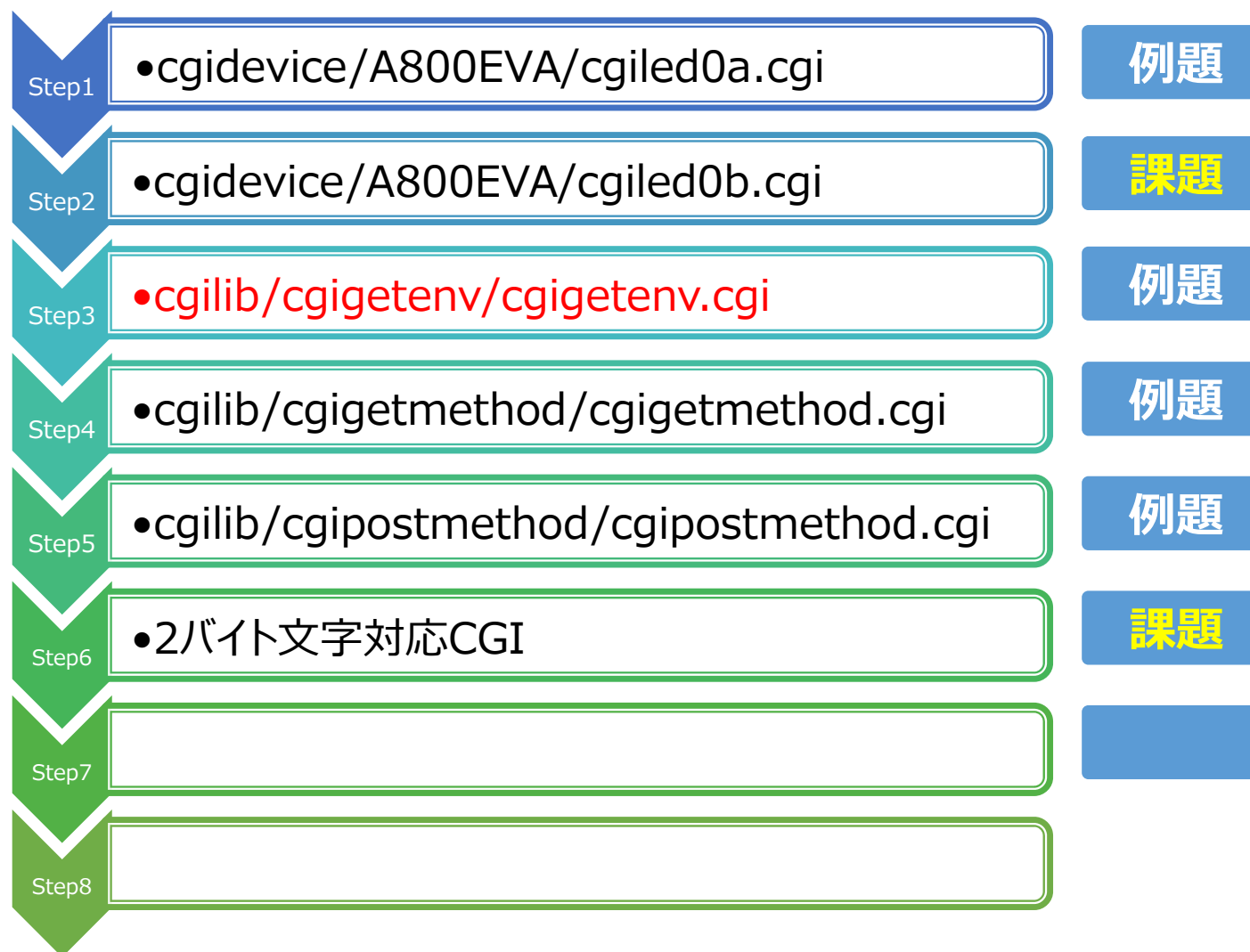
機能

- ボタンの配置
- index.htmlボタンをクリックすると index.html へ遷移
- cgitest.cgiボタンをクリックすると cgitest.cgi へ遷移

ヒント

- submit
- onclick="location.href='./index.html'"

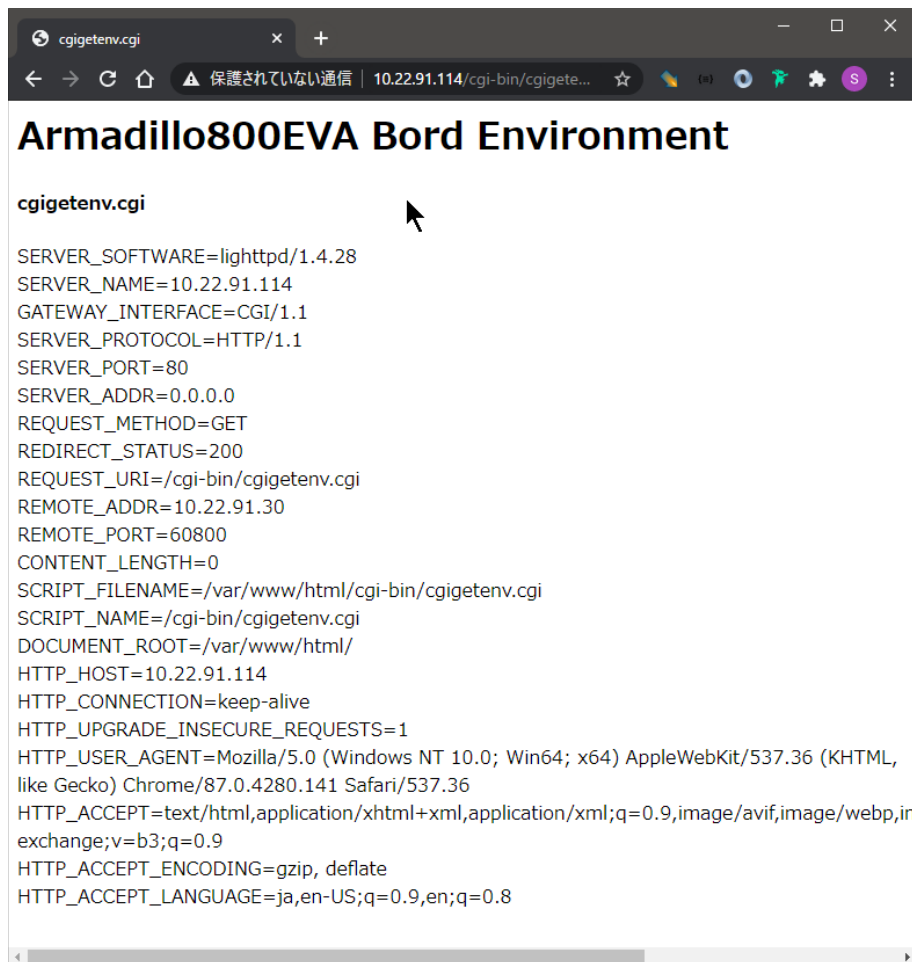
流れ



[例題] cgigetenv.cgi

- 🐼 Armadillo800EVA のWebサーバー環境を取得する

🐼 <http://10.22.91.114/cgi-bin/cgigetenv.cgi>



The screenshot shows a web browser window with the address bar displaying "cgigetenv.cgi" and the URL "http://10.22.91.114/cgi-bin/cgigetenv.cgi". The page title is "Armadillo800EVA Bord Environment". The main content area displays the output of the "cgigetenv.cgi" script, listing various environment variables and their values. A mouse cursor is visible over the text "cgigetenv.cgi".

```
cgigetenv.cgi

SERVER_SOFTWARE=lighttpd/1.4.28
SERVER_NAME=10.22.91.114
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
SERVER_PORT=80
SERVER_ADDR=0.0.0.0
REQUEST_METHOD=GET
REDIRECT_STATUS=200
REQUEST_URI=/cgi-bin/cgigetenv.cgi
REMOTE_ADDR=10.22.91.30
REMOTE_PORT=60800
CONTENT_LENGTH=0
SCRIPT_FILENAME=/var/www/html/cgi-bin/cgigetenv.cgi
SCRIPT_NAME=/cgi-bin/cgigetenv.cgi
DOCUMENT_ROOT=/var/www/html/
HTTP_HOST=10.22.91.114
HTTP_CONNECTION=keep-alive
HTTP_UPGRADE_INSECURE_REQUESTS=1
HTTP_USER_AGENT=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.141 Safari/537.36
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,in
exchange;v=b3;q=0.9
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_ACCEPT_LANGUAGE=ja,en-US;q=0.9,en;q=0.8
```

- 🐼 取得方法

🐼 `extern char** environ;`
🐼 `getenv()`

[例題] cgigetenv.c

```
//=====
// URL: http://10.22.91.114/cgi-bin/cgigetenv.cgi
//
//                                     2021/1/10 S.Fujimoto
//=====
#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>

int main( int argc, char **argv )
{
    extern char** environ;
    char** env;

    printf("Content-type:text/html;charset=UTF-8¥n¥n");

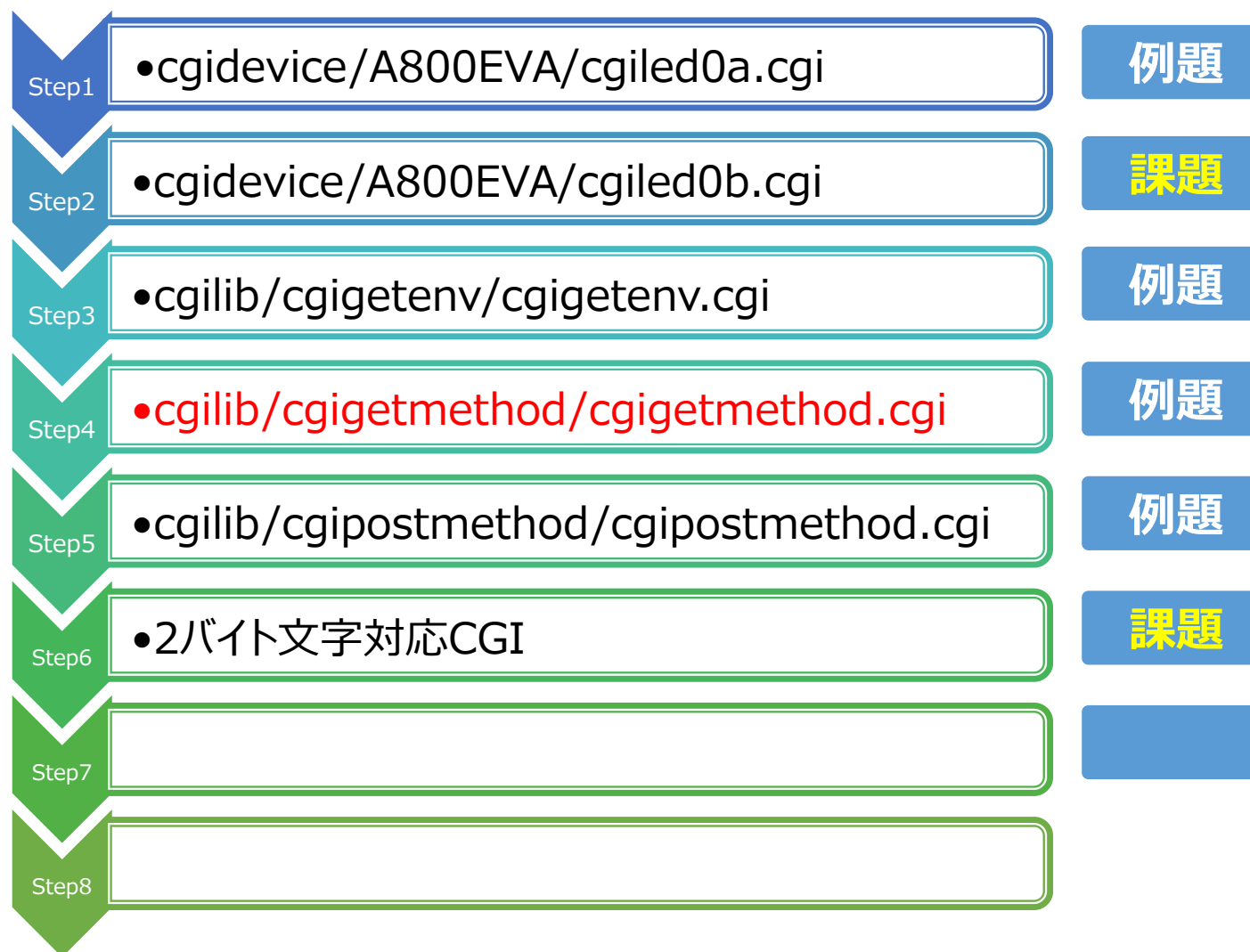
    printf("<html>¥n");
    printf("<head>¥n");
    printf("<title>cgigetenv.cgi</title>¥n");
    printf("</head>¥n");

    printf("<body>¥n");
    printf("<h1>Armadillo800EVA Bord Environment</h1>¥n");
    printf("<h4>cgigetenv.cgi</h4>¥n");

    for ( env=environ ; *env != NULL ; env++ ) {
        printf("%s<br>¥n", *env );
    }

    printf("</body>¥n");
    printf("</html>¥n");
    return 0;
}
// End of file
```

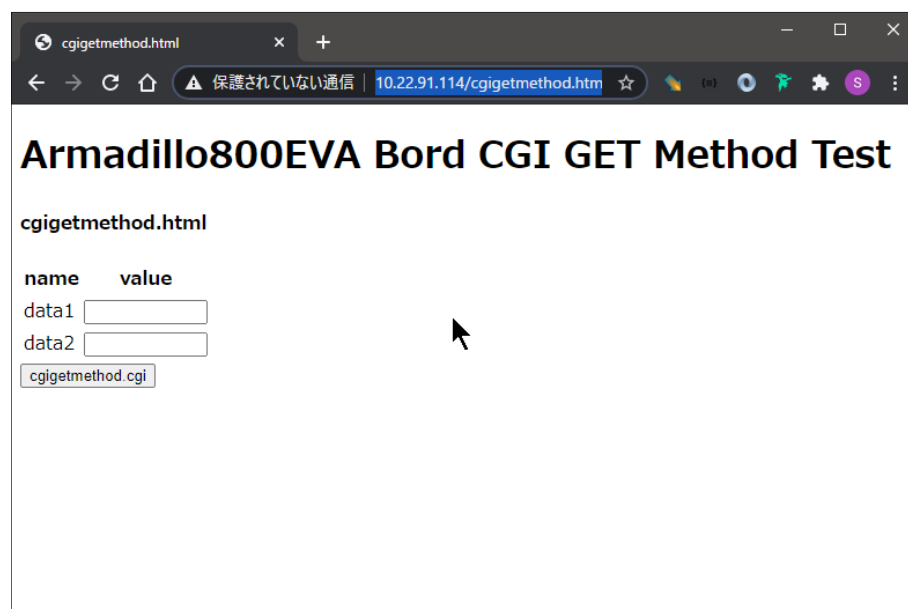
流れ



[例題] cgigetmethod.cgi

📁 cgigetmethod.html

🌐 <http://10.22.91.114/cgigetmethod.html>



📁 cgigetmethod.c

🌐 <http://10.22.91.114/cgi-bin/cgigetmethod.cgi?data1=aaa&data2=123>



📁 機能

🌐 data1 および data2 テキストボックスにデータ入力

📁 data1: aaa data2: 123

🌐 GETメソッドで cgigetmethod.cgi ヘデータを渡す

🌐 cgigetmethod.cgi は データを表示

🌐 URLに注目

📁 ?data1=aaa&data2=123

[例題] cgigetmethod.html

```
<!DOCTYPE html>
<html lang="jp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>cgigetmethod.html</title>
</head>

<body>
  <h1>Armadillo800EVA Bord CGI GET Method Test</h1>
  <h4>cgigetmethod.html</h4>

  <form action="http://10.22.91.114/cgi-bin/cgigetmethod.cgi" method="GET">
    <table>
      <tr><th>name</th><th>value</th></tr>
      <tr><td>data1</td> <td><input type="text" name="data1" size="10"></td></tr>
      <tr><td>data2</td> <td><input type="text" name="data2" size="10"></td></tr>
    </table>
    <button type="submit" onclick="location.href='cgi-bin/cgigetmethod.cgi'">cgigetmethod.cgi</button>
  </form>
</body>
</html>
```

[例題] cgigetmethod.c

```
//=====
// URL: http://10.22.91.114/cgi-bin/cgipostmethod.cgi
// (2バイト文字処理未対応)
//
//                                     2021/1/10 S.Fujimoto
//=====
```

```
#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>

int main( int argc, char **argv )
{
    extern char** environ;
    char** env;

    char getmsg[1024];          // GETリクエストによる要求メッセージを格納
    char fieldname[10][1024];  // フィールド名を格納
    char fieldvalue[10][1024]; // フィールド値を格納

    // 環境変数QUERY_STRINGにセットされたGETリクエストメッセージをgetenv関数で取得
    if( getenv( "QUERY_STRING" ) != NULL ) {
        strcpy( getmsg, getenv("QUERY_STRING") );
    } else {
        strcpy( getmsg, "NO_DATA" );
    }
}
```

```
// getmsgに格納されているメッセージは、フィールド名=フィールド値&フィールド名=フィールド値
// ...なので、strtok関数で分離する
strcpy( fieldname [0], strtok(getmsg, "&") );
strcpy( fieldvalue[0], strtok(NULL, "&") );
strcpy( fieldname [1], strtok(NULL, "&") );
strcpy( fieldvalue[1], strtok(NULL, "&") );

printf("Content-type:text/html;charset=UTF-8\r\n\r\n");
printf("<html>\r\n");
printf("<head>\r\n");
printf("<title>cgigetmethod.cgi</title>\r\n");
printf("</head>\r\n");

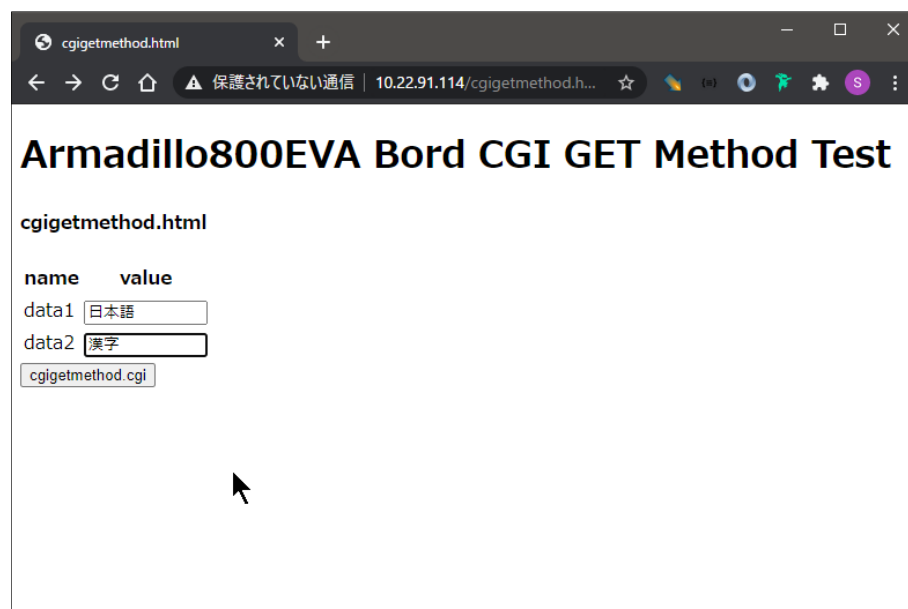
printf("<body>\r\n");
printf("<h1>Armadillo800EVA Bord CGI GET Method Result</h1>\r\n");
printf("<h4>cgigetmethod.cgi</h4>\r\n");
printf("QUERY STRING:%s<br>\r\n", getmsg );
printf("1th Fieldname:%s Fieldvalue:%s<br>\r\n", fieldname[0],
fieldname[0]);
printf("2th Fieldname:%s Fieldvalue:%s<br>\r\n", fieldname[1],
fieldname[1]);

printf("</body>\r\n");
printf("</html>\r\n");
return 0;
}
// End of file
```

[例題] cgigetmethod.cgi (2バイト文字)

📌 入力HTMLレイアウト例

📌 <http://10.22.91.114/cgigetmethod.html>



Armadillo800EVA Bord CGI GET Method Test

cgigetmethod.html

name	value
data1	日本語
data2	漢字

📌 機能

📌 data1 および data2 テキストボックスにデータ入力

📌 data1: 日本語 data2: 漢字

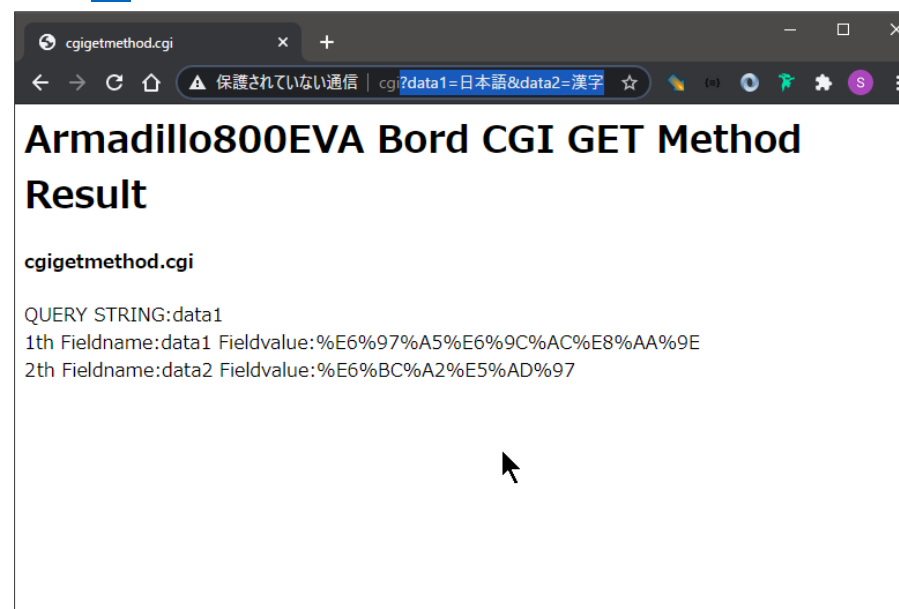
📌 GETメソッドで cgigetmethod.cgi ヘデータを渡す

📌 cgigetmethod.cgi は データを表示

📌 charset、*.htmlの保存形式、Webブラウザの文字コードは全てUTF-8

📌 出力cgiレイアウト例

📌 <http://10.22.91.114/cgi-bin/cgigetmethod.cgi?data1=%E6%97%A5%E6%9C%AC%E8%AA%9E&data2=%E6%BC%A2%E5%AD%97>



Armadillo800EVA Bord CGI GET Method Result

cgigetmethod.cgi

QUERY STRING:data1

1th Fieldname:data1 Fieldvalue:%E6%97%A5%E6%9C%AC%E8%AA%9E

2th Fieldname:data2 Fieldvalue:%E6%BC%A2%E5%AD%97

📌 URLに注目

- 📌 Webブラウザはデコード処理を実装しているので、適切に変換している
- 📌 一方、*.cgiはデコード処理を実装していないので、GETメソッドの引数をWebブラウザの文字コード (UTF-8) で表示している
- 📌 日本語 (UTF-8) : %E6%97%A5%E6%9C%AC%E8%AA%9E

📌 Webブラウザではなく、あるプログラムからGETメソッドで引数を*.cgiに渡す場合、エンコード処理が必要

URLエンコードとデコード

🔗 エンコードマニアックスのサイトで調べてみる

🔗 <http://www.encodemaniacs.com/>



🔗 URLエンコード Webブラウザ -> *.cgi

🔗 URLデコード *.cgi -> Webブラウザ

🔗 data1=日本語&data2=漢字 を URLエンコード

🔗 UTF-8

🔗 data1%3D%E6%97%A5%E6%9C%AC%E8%AA%9E%26data2%3D%E6%BC%A2%E5%AD%97

🔗 Shift-JIS

🔗 data1%3D%93%FA%96%7B%8C%EA%26data2%3D%8A%BF%8E%9A

🔗 data1%3D%E6%97%A5%E6%9C%AC%E8%AA%9E%26data2%3D%E6%BC%A2%E5%AD%97 を URLデコード

🔗 UTF-8

🔗 data1=日本語&data2=漢字

🔗 Shift-JIS

🔗 data1=偽・潜り隠?data2=狸「蟄

🔗 data1%3D%93%FA%96%7B%8C%EA%26data2%3D%8A%BF%8E%9A を URLデコード

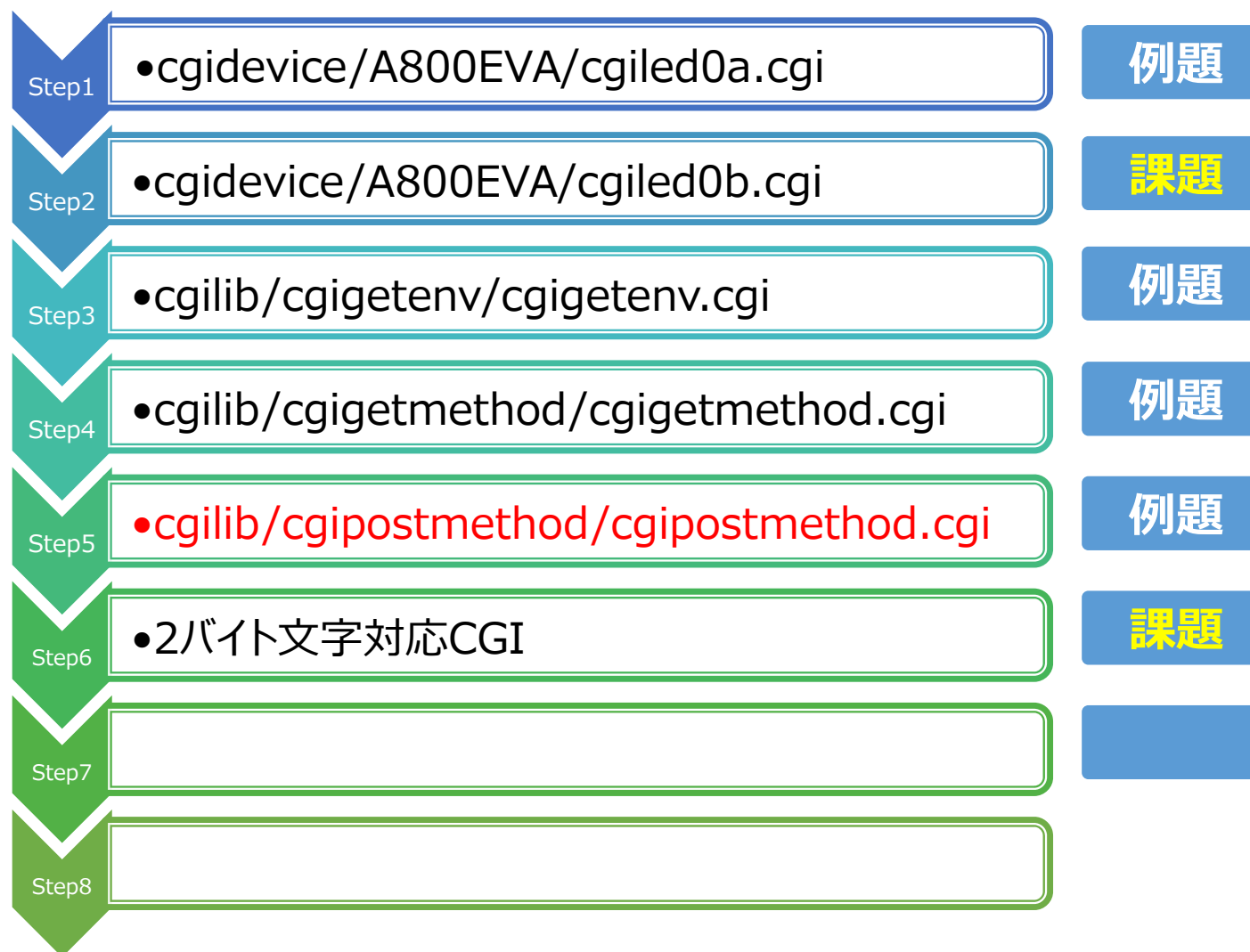
🔗 UTF-8

🔗 data1%3D%93%FA%96%7B%8C%EA%26data2%3D%8A%BF%8E%9A

🔗 Shift-JIS

🔗 data1=日本語&data2=漢字

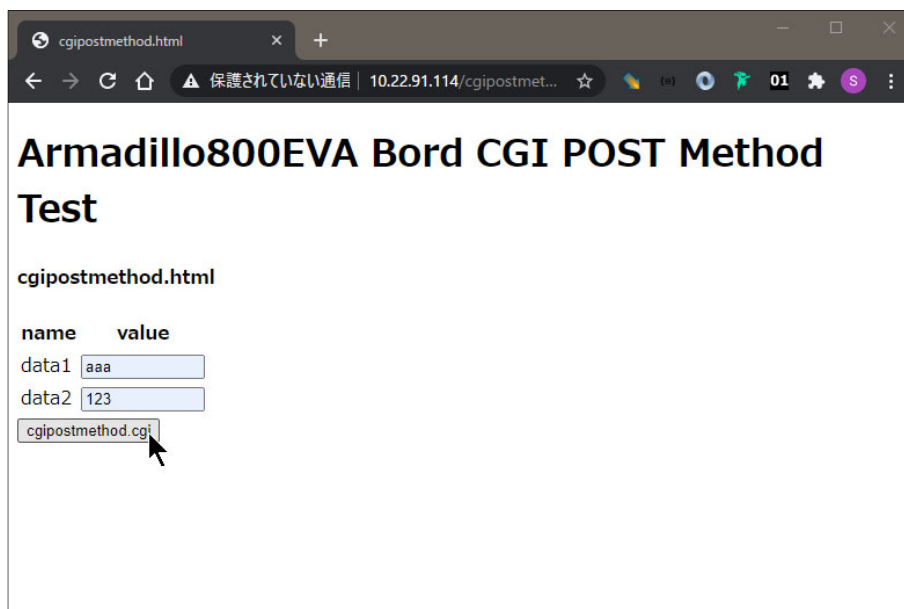
流れ



[例題] cgipostmethod.cgi

📌 入力HTMLレイアウト例

📌 <http://10.22.91.114/cgipostmethod.html>



Armadillo800EVA Bord CGI POST Method Test

cgipostmethod.html

name	value
data1	aaa
data2	123

📌 出力cgiレイアウト例

📌 <http://10.22.91.114/cgi-bin/cgiitpostmethod.cgi>



Armadillo800EVA Bord CGI POST Method Result

cgipostmethod.cgi

1th Fieldname:data1 Fieldvalue:aaa
 2th Fieldname:data2 Fieldvalue:123

📌 機能

📌 data1 および data2 テキストボックスにデータ入力

📌 data1: aaa data2: 123

📌 POSTメソッドで cgipostmethod.cgi ヘデータを渡す

📌 cgipostmethod.cgi は データを表示

📌 文字コード、*.htmlの保存形式、Webブラウザの文字コードは全てUTF-8

📌 URLに注目

📌 GETメソッドのような引数表示がない

[例題] cgipostmethod.html

```
<!DOCTYPE html>
<html lang="jp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>cgipostmethod.html</title>
</head>

<body>
  <h1>Armadillo800EVA Bord CGI POST Method Test</h1>
  <h4>cgipostmethod.html</h4>

  <form action="http://10.22.91.114/cgi-bin/cgiposttmethod.cgi" method= "POST">
    <table>
      <tr><th>name</th><th>value</th></tr>
      <tr><td>data1</td> <td><input type="text" name="data1" size="10"></td></tr>
      <tr><td>data2</td> <td><input type="text" name="data2" size="10"></td></tr>
    </table>
    <button type="submit" onclick="location.href='cgi-bin/cgipostmethod.cgi'">cgipostmethod.cgi</button>
  </form>
</body>
</html>
```

[例題] cgi postmethod.c

```
//=====
// URL: http://10.22.91.114/cgi-bin/cgigetmethod.cgi
// (2バイト文字処理未対応)
//
// 2021/1/10 S.Fujimoto
//=====
#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>

int main( int argc, char **argv )
{
    char postmsg[1024];          // POSTメソッドによる要求メッセージを格納
    char fieldname[10][1024];   // フィールド名を格納
    char fieldvalue[10][1024];  // フィールド値を格納
    int len;                    // POSTメッセージ長
    char *plen;                 // POSTメッセージ長へのポインタ
    char *pbuf;                 // POSTメッセージ格納領域へのポインタ

    // POSTメソッドによる要求メッセージ長の取得
    if( ( plen = getenv( "CONTENT_LENGTH" ) ) == NULL ) {
        printf("No contents./n");
        exit(1);
    }

    // POSTメソッドによる要求メッセージの格納領域確保
    len = atoi( plen );
    if( len > sizeof( postmsg ) ) {
        printf("buffer over flow raise. %n");
        exit(1);
    }
    pbuf = (char *)malloc( len+1 );

    // POSTメソッド要求メッセージ取得
    scanf("%s", pbuf);
    pbuf[len] = '\0';

    // postmsgに格納されているメッセージは、フィールド名=フィールド値&フィールド名=フィールド値&...なので、strtok関数で分離する
    strncpy( postmsg, pbuf, len+1 );
    strcpy( fieldname[0], strtok(postmsg, "&") );
    strcpy( fieldvalue[0], strtok(NULL, "&") );
    strcpy( fieldname[1], strtok(NULL, "&") );
    strcpy( fieldvalue[1], strtok(NULL, "&") );

    // HTMLヘッダ部
    printf("Content-type:text/html;charset=UTF-8¥n¥n");
    printf("<html>¥n");
    printf("<head>¥n");
    printf("<title>cgipostmethod.cgi</title>¥n");
    printf("</head>¥n");

    // HTMLボディ部
    printf("<body>¥n");
    printf("<h1>Armadillo800EVA Bord CGI POST Method Result</h1>¥n");
    printf("<h4>cgipostmethod.cgi</h4>¥n");
    printf("1th Fieldname:%s Fieldvalue:%s<br>¥n", fieldname[0], fieldvalue[0]);
    printf("2th Fieldname:%s Fieldvalue:%s<br>¥n", fieldname[1], fieldvalue[1]);

    printf("</body>¥n");
    printf("</html>¥n");

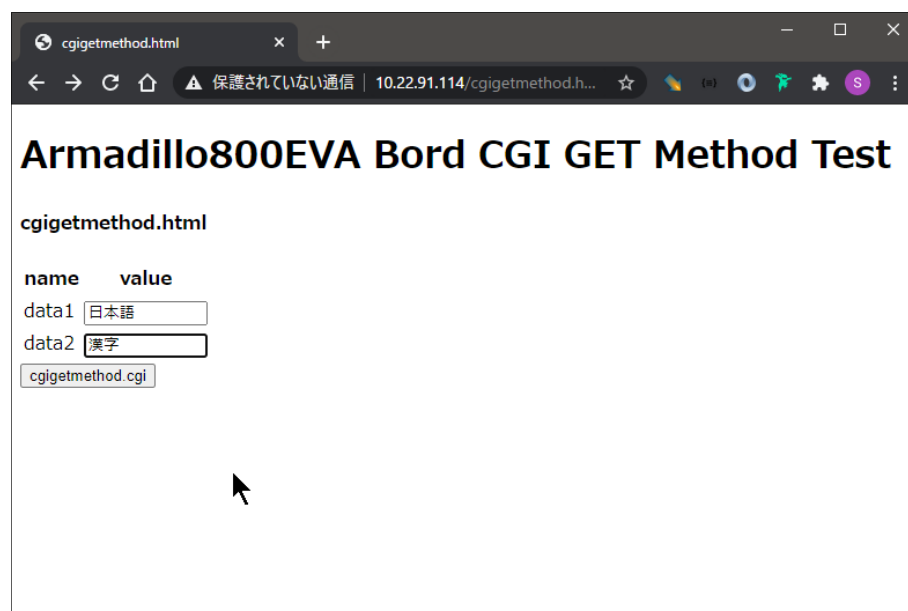
    free(pbuf);

    return 0;
}
// End of file
```

[例題] cgipostmethod.cgi (2バイト文字)

📌 入力HTMLレイアウト例

📌 <http://10.22.91.114/cgigetmethod.html>



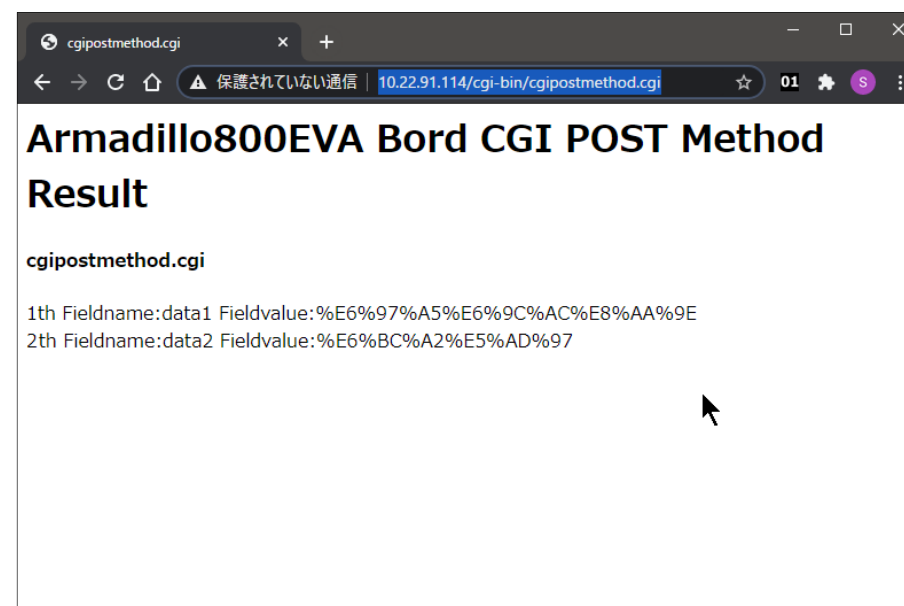
Armadillo800EVA Bord CGI GET Method Test

cgigetmethod.html

name	value
data1	日本語
data2	漢字

📌 出力cgiレイアウト例

📌 <http://10.22.91.114/cgi-bin/cgipostmethod.cgi>



Armadillo800EVA Bord CGI POST Method Result

cgipostmethod.cgi

1th Fieldname:data1 Fieldvalue:%E6%97%A5%E6%9C%AC%E8%AA%9E
 2th Fieldname:data2 Fieldvalue:%E6%BC%A2%E5%AD%A9

📌 機能

📌 data1 および data2 テキストボックスにデータ入力

📌 data1: 日本語 data2: 漢字

📌 GETメソッドで cgigetmethod.cgi ヘデータを渡す

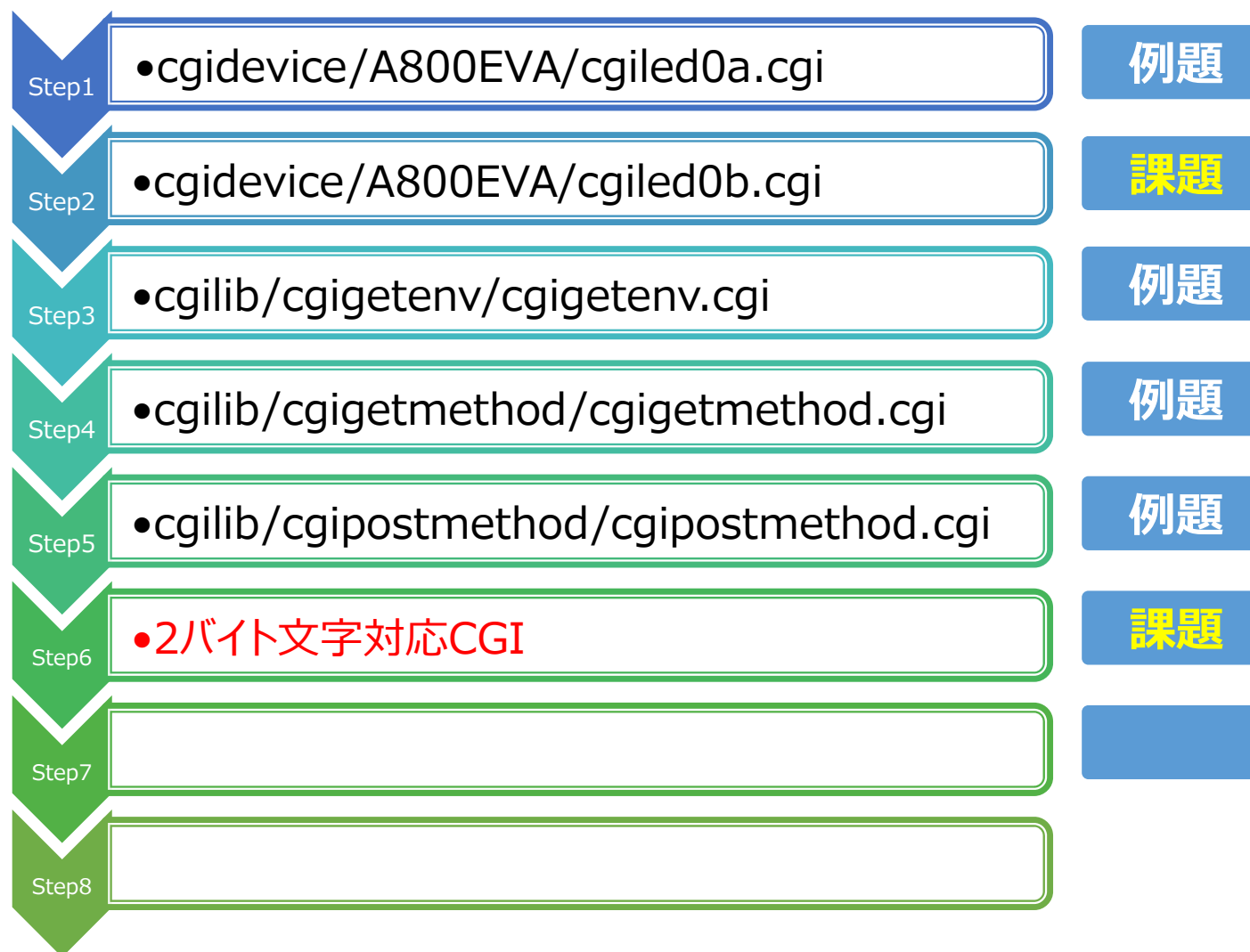
📌 cgigetmethod.cgi は データを表示

📌 charset、*.htmlの保存形式、Webブラウザの文字コードは全てUTF-8

📌 *.cgiはデコード処理を実装していないので、POSTメソッドの引数をWebブラウザの文字コード (UTF-8) で表示している

📌 日本語 (UTF-8) : %E6%97%A5%E6%9C%AC%E8%AA%9E

流れ



[課題] 2バイト文字対応CGI cgifpostdebug.cgi

🐉 入力HTMLレイアウト例

🐉 <http://10.22.91.114/cgifpostmethod.html>

🐉 機能

🐉 data1 および data2 テキストボックスにデータ入力

🐉 data1: 日本語 data2: 漢字

🐉 POSTメソッドで cgifpostdebug.cgi ヘデータを渡す

🐉 cgifpostdebug.cgi は データを表示

🐉 charset、*.htmlの保存形式、Webブラウザの文字コードは全てUTF-8

parse 関数仕様

🔗 parse関数

関数I/F	<code>int parse(char *pst, long max, char *pfname[], char *pfval[], int *pfnumen)</code>	
関数責務	Webブラウザからの引数(文字列)を分離し、フィールド名を第3引数*pfnameへ、フィールド値を第4引数*pfvalに格納する	
引数	<code>char *pst</code>	: 分離文字列
	<code>long max</code>	: 文字列サイズ
	<code>char *pfname[]</code>	: フィールド名を格納するバッファ領域 (ポインタ配列へのポインタ)
	<code>char *pfval[]</code>	: フィールド値を格納するバッファ領域 (ポインタ配列へのポインタ)
	<code>int len</code>	: フィールド数
復帰値	0: 正常 -1: NULL	
使用方法	<code>void uip_init(void);</code>	
補足	strtokを利用しても良い 【利用例】 <pre> char postmsg [MAXLEN]; // POSTメソッドによる要求メッセージを格納 char *pfieldname [MAXLEN]; // フィールド名格納領域へのポインタ char *pfieldvalue[MAXLEN]; // フィールド値格納領域へのポインタ int nfield; // フィールド数 // postmsgに格納されているメッセージは、フィールド名=フィールド値&フィールド名=フィールド値&...なので、parse関数で分離させる parse(postmsg, MAXLEN, pfieldname, pfieldvalue, &nfield); </pre>	

decode 関数仕様

decode関数

関数I/F	<code>int decode(char *pst, long len)</code>	
関数責務	第1引数*pstの文字列を16進数2桁の文字列へ変換する	
引数	<code>char *pst</code>	: 変換前の文字列へのポインタ
	<code>int len</code>	: 文字列サイズ
復帰値	0: 正常 -1:NULL	
使用方法	<code>void uip_init(void);</code>	
補足	変換前の文字列 “%93” 変換後の文字 0x93 の 1バイトデータ 【利用例】 <pre> for (i=0 ; i<nfield ; i++) { decode(pfieldname[i], strlen(pfieldname[i])); decode(pfieldvalue[i], strlen(pfieldvalue[i])); #ifdef DEBUG == 0 printf(“%s ---> %s
/n”, pfieldname[i], pfieldvalue[i]); #endif } </pre>	

[例題] cgifpostdebug.html

```
<!DOCTYPE html>
<html lang="jp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>cgipostdebug.html</title>
</head>

<body>
  <h1>Armadillo800EVA Bord CGI POST Method Debug</h1>
  <h4>cgipostdebug.html</h4>

  <form action="http://10.22.91.114/cgi-bin/cgifpostdebug.cgi" method="POST">
    <table>
      <tr><th>name</th><th>value</th></tr>
      <tr><td>data1</td> <td><input type="text" name="data1" size="10"></td></tr>
      <tr><td>data2</td> <td><input type="text" name="data2" size="10"></td></tr>
    </table>
    <button type="submit" onclick="location.href='cgi-bin/cgifpostdebug.cgi'">cgipostdebug.cgi</button>
  </form>
</body>
</html>
```

[課題] cgipostdebug.c

```
//=====
// URL: http://10.22.91.114/cgi-bin/cgipostdebug.cgi
//
// 要求文字列(shift-
jis):"data1=%93%FA%96%7B%8C%EA&data2=%8A%BF%8E%9A&%8E%C0%8Ds=%8E%C0%8Ds"
// 要求文字列(utf-
8) : "data1=%E6%97%A5%E6%9C%AC%E8%AA%9E&data2=%E6%BC%A2%E5%AD%97&%E5%AE%9F%E
8%A1%8C=%E5%AE%9F%E8%A1%8C"
//
// POSTリクエストにて"日本語", "漢字"をホタルをクリックによりHTTPサーバーへ送信。
// HTTPサーバーは、標準入力関数scanfによりクライアントからの要求文字列を取得
//
// 漢字コードの参考URL http://ash.jp/code/codetbl2.htm
//
// デバッグ時は、#define DEBUG 1 に設定
//
// parse関数の処理(shift-jis)
// pfieldname[0]:data1 pfieldvalue[0]:%93%FA%96%7B%8C%EA("日本語")
// pfieldname[1]:data2 pfieldvalue[1]:%8A%BF%8E%9A("漢字")
//
// parse関数の処理(utf-8)
// pfieldname[0]:data1 pfieldvalue[0]:%E6%97%A5%E6%9C%AC%E8%AA%9E("日本語")
// pfieldname[1]:data2 pfieldvalue[1]:%E6%BC%A2%E5%AD%97("漢字")
//
// decode関数の処理(shift-jis)
// "%93%FA"の文字列('日')を、93FAの16進数へデコード
// "%96%7B"の文字列('本')を、967Bの16進数へデコード
//
// decode関数の処理(utf-8)
// "%E6%97%A5"の文字列('日')を、E697A5の16進数へデコード
// "%E6%9C%AC"の文字列('本')を、E6AC9Cの16進数へデコード
//
// 2007/12/16 S.Fujimoto 新規作成
// 2009/ 2/17 S.Fujimoto デバッグ用 shift-jisとutf8の文字列を追加
// 2021/ 1/10 S.Fujimoto Armadillo800EVA へ移植
//=====

#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>

// make時(コンパイル時) -DDEBUGオプションがない場合
#ifdef DEBUG
#define DEBUG 0 // デバッグ時1
#endif

#if DEBUG == 1
#define printdc(varc) printf(#varc":'%c' H'%x D'%d ￥n", varc, varc, varc)
#define printds(vars) printf(#vars":'%s' ￥n", vars)
#define printdi(vari) printf(#vari":'%d' ￥n", vari)
#define printdl(varl) printf(#varl":'%ld' ￥n", varl)
#define printdarr(addr, arrname, cnt, varc)/
printf("%p | "#arrname "[%2d]":'%c' H'%x D'%d ￥n", addr, cnt, varc,
varc, varc)
#endif

#define MAXLEN 2048

// プロトタイプ宣言
int parse(char *, long, char *[], char *[], int *);
int decode(char *, long);
```

[課題] cgipostdebug.c

```
int main( int argc, char **argv )
{
    char *pbuf;
    char *plen;
    char postmsg [MAXLEN]; // POSTメソッドによる要求メッセージを格納
    char *pfieldname [MAXLEN]; // フィールド名格納領域へのポインタ
    char *pfieldvalue[MAXLEN]; // フィールド値格納領域へのポインタ
    int nfield; // フィールド数
    long i;
    long len;

    #if DEBUG
        printf("--Debug mode--/n");
        printf(" Test string:");
        strcpy(postmsg,
"data1=%93%FA%96%7B%8C%EA&data2=%8A%BF%8E%9A&%8E%0%8Ds=%8E%0%8Ds");
        // strcpy(postmsg,
"data1=%E6%97%A5%E6%9C%AC%E8%AA%9E&data2=%E6%BC%A2%E5%AD%97&%E5%AE%9F%E8%A1%8C
=%E5%AE%9F%E8%A1%8C");
        printds(postmsg);
    #else
```

// POSTメソッドによる要求メッセージ長の取得

// POSTメソッドによる要求メッセージの格納領域確保

// POSTメソッド要求メッセージ取得

// ハット記号postmsgへコピー

// HTMLヘッダ部

#endif

// HTMLボディ部

// postmsgに格納されているメッセージは、フィールド名=フィールド値&フィールド名=フィールド値・・・なので、parse関数で分離させる

for (i=0 ; i<nfield ; i++) {

}

#if DEBUG

#endif

free(pbuf);

return 0;

}

[課題] cgipostdebug.c

```
int parse(char *pst, long max, char *pfname[], char *pfval[], int *pfnum)
{
    //-----
    // HTMLフォームからの送信データをフィールド名とフィールド値に切り分ける関数
    // 第1引数 : parseする文字列
    // 第2引数 : サイズ
    // 第3引数 : フィールド名を格納するバッファへのポインタ
    // 第4引数 : フィールド値を格納するバッファへのポインタ
    // 第5引数 : フィールド数
    // 戻り値 0 : 正常終了 -1 : NULL
    //-----

    int i=0; // カンマ
    int pos=0; // 文字列のインデックス位置

    *pfnum = 0;
    if(pst[0] == '¥0') return -1; // parseする先頭文字が¥0の場合終了
    pfname[0] = pst; // 文字列をフィールド名を格納するバッファへ

    // parse文字列が¥0でなく最大サイズ未満の場合
    return 0;
}
```

[課題] cgipostdebug.c

```
int decode(char *pst, long len)
{
    //-----
    // URLデコード関数
    //   第1引数 : デコード前の文字列 -> デコード後の文字列
    //   第2引数 : サイズ(文字列の長さ)
    //   戻り値 0 : 正常終了 -1 : 異常終了
    //-----
    int i, j;
    char buf,*s1;

    if(len == 0) return(-1);          // サイズが0の場合、終了

    return 0;
}
// End of file
```