

モジュール設計仕様書	システム	Visual Studio Code	コードディタ、コンソール付だからテストもできる	作成日	2020/4/6
補足	モジュール	Hello World	こんにちは、World	作成者	hosiyama

DFD: gawk データ整形などの処理で威力を発揮する

```

1 # $JIS $Workfile: hello.awk $$Revision: $$Date: $
2 # $NoKeywords: $
3 #
4 # Usage(ps,cmd):gawk -f ../gawk/hello.awk /dev/nul
5 # Usage(ps,cmd):gawk -f ../gawk/hello.awk /dev/con
6 # /dev/con(コンソール入力)の終了(EOF)は^Z( Ctrl+Z)
7 # Usage(sh,ps,cmd):gawk -f ../gawk/hello.awk --
8 # --(コンソール入力)の終了(EOF)はshなら^D( Ctrl+D), ps,cmdなら^Z( Ctrl+Z)
9 # Usage(sh):gawk -f ../gawk/hello.awk /dev/null
10 #
11
12 BEGIN{
13     print "こんにちは、World","はじめに";
14 }
15 {
16     print "こんにちは、World","その " FNR ":" $0;
17 }
18 END{
19     print "こんにちは、World","おわりに";
20 }

```

\$gawk --version (Windows版) ※便宜上gawkと呼んでいる
GNU Awk 3.1.5
Copyright (C) 1989, 1991-2005 Free Software Foundation.

\$ awk --version (Unix(WSL Ubuntu))版 ※便宜上awkと呼んでいる
GNU Awk 4.1.4, API: 1.1 (GNU MPFR 4.0.1, GNU MP 6.1.2)
Copyright (C) 1989, 1991-2016 Free Software Foundation.

awkはテキストデータに対して、区切り文字(項目、行)を任意に設定でき、しかも正規表現が使えるので、データ整形に有用。(CSV:Comma Separated Valueの略で、","は区切り文字の代表。tab区切りが実用的かも)
awkは開発者3人の頭文字とか。

Unixといえばawk & sed そして vi である。昔の話になってしまったか。最初にedラインエディタが作られ、Stream版としてsed、Visual版としてviが生まれたとか。とにかく正規表現(Regular Expression)は画期的である。

なお、Windows版のUnix環境として、MinGW(Minimalist GNU for Windows)も導入した。なので、
GNU bash, version 3.1.23(1)-release (i686-pc-msys)
Copyright (C) 2005 Free Software Foundation, Inc.
を便宜上sh
GNU bash, バージョン 4.4.19(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2016 Free Software Foundation, Inc.
をbashと呼んでいる。

- ^DでEOF入力すると、ENDブロックが実行される。
- 誤って^Zとするとバックグラウンド(bg)に移行するので戻すときはfgコマンドで。
- /dev/nul,/dev/nullではいきなりEOFなのでBEGIN,ENDブロックのみ実行される

モジュール設計仕様書	システム	Visual Studio Code	コードディタ、コンソール付だからテストもできる	作成日	2020/4/6
補足	モジュール	Hello World	こんにちは、World	作成者	hosiya

DFD: clisp 1980年代のAIブームで活用された記号処理言語

The image shows a Visual Studio Code editor with a file named 'hello.lisp' open. The code in the editor is as follows:

```

1  ;; SJIS $Workfile: $$Revision: $$Date: $
2  ;; $NoKeywords: $
3  ;;
4  ;; usage:clisp hello.lisp
5  ;; (load 'hello.lisp)
6  ;; (hello)
7  ;; (exit)
8
9  (defun hello ()
10 |   (format t "こんにちは、World")
11 | )
12
13 ;comment next line & save file for load only
14 (hello)
15

```

The terminal window shows the following commands and output:

```

PS J:\dev\CMD\clisp> clisp hello.lisp
こんにちは、World
PS J:\dev\CMD\clisp>

```

A callout box provides detailed instructions and observations:

- >clisp --version
GNU CLISP 2.49 (2010-07-07) (built on STSst063.jenty.by [150.0.0.63])
Software: GNU C 3.4.5 (mingw-vista special r3)
- clispインタプリタで実行すると…
PS J:\dev\CMD\clisp> clisp
①関数定義(defun)後に関数(hello)実行するlispファイルをloadするケース……OK
[1]> (load 'hello.lisp)
;; Loading file hello.lisp ...
こんにちは、World
;; Loaded file hello.lisp
T
[2]>
- ②コンソール入力(「選択範囲を実行」)すると漢字が化ける……NG、保留中
[1]> (defun hello ()
 (format t "AWorld")
)
HELLO
[2]> (hello)
AWorld
NIL
[3]>
- ③関数実行をコメントアウトしload(defunのみ)実行後、(hello)を実行……OK
[1]> (load 'hello.lisp)
;; Loading file hello.lisp ...
;; Loaded file hello.lisp
T
[2]> (hello)
こんにちは、World
NIL
[3]>

The status bar at the bottom indicates: 行 4、列 26 (16 個選択) スペース: 2 Shift JIS CRLF Lisp

モジュール設計仕様書	システム	Visual Studio Code	コードディタ、コンソール付だからテストもできる	作成日	2020/4/6
補足	モジュール	Hello World	こんにちは、World	作成者	hoshiyama

DFD: C++ ネイティブコードを生成するオブジェクト指向言語

The screenshot shows the Visual Studio Code interface with the following components:

- Explorer:** Shows the project structure including `helloworld.cpp` and `helloworld.exe`.
- Code Editor:** Displays the source code in `helloworld.cpp`:


```

1 #include <iostream>
2 #include <vector>
3 #include <string>
4
5 using namespace std;
6
7 int main()
8 {
9     vector<string> msg {"こんにちは", "C++", "World", "from", "VS Code", "and the C++ extension!"};
10
11     for (const string& word : msg)
12     {
13         cout << word << " ";
14     }
15     cout << endl;
16 }
      
```
- Terminal:** Shows the command `ls` and its output:


```

PS J:\dev\Projects\helloworld> ls

ディレクトリ: J:\dev\Projects\helloworld

Mode                LastWriteTime         Length Name
----                -
d-----          2020/04/05   19:27         .vs
d-----          2020/04/05   19:35        .vscode
-a----          2020/04/05   19:44           303 helloworld.cpp
-a----          2020/04/05   19:46          149497 helloworld.exe

PS J:\dev\Projects\helloworld> ./helloworld.exe
こんにちは C++ World from VS Code and the C++ extension!
PS J:\dev\Projects\helloworld>
      
```
- Debug Console:** Shows a memory dump of the `cout` output, confirming the string content:


```

{...}
  npos: 4294967295
  > _M_dataplus
    _M_string_length: 5
  <anonymous union>
    <_M_local_buf
      [0]: 87 'W'
      [1]: 111 'o'
      [2]: 114 'r'
      [3]: 108 'l'
      [4]: 100 'd'
      [5]: 0 '\000'
    }
      
```

Using GCC with MinGW
<https://code.visualstudio.com/docs/cpp/config-mingw>
 を参考に、
 helloworld.cppの”Hello”を”こんにちは”に変更した。
 デバッグコンソールでは、ソースコードがSJISだと字化けする。

モジュール設計仕様書	システム	Visual Studio Code	コードディタ、コンソール付だからテストもできる	作成日	2020/4/14
補足	モジュール	Hello World	こんにちは、World	作成者	hosiyama

DFD: C ネイティブコードを生成する構造化言語

The screenshot shows the Visual Studio Code interface with a C program named `helloworld.c` open. The code is as follows:

```

1 //SJIS $Workfile: helloworld.c $$Revision: 2 $$Date: 20/04/13 23:40 $
2 //
3 // helloworld.c:   こんにちは
4 //
5 // Usage: ./helloworld.exe
6 //
7
8 #include <stdio.h>
9 #include <string.h>
10
11 int
12 main()
13 {
14     char *msg[]= {"こんにちは", "C", "World", "from", "VS Code", "and the C/C++ extension!", NULL};
15
16     /* ----- procedure */
17
18     // こんにちは
19     for(char **pp=msg;*pp!=NULL;pp++){
20         printf("%s%s", (pp==msg?"": " "), *pp);
21     }
22     printf("\n");
23
24 }

```

The terminal window shows the execution of the program:

```

PS G:\Projects\ccpp> ./helloworld.exe
こんにちは C World from VS Code and the C/C++ extension!
PS G:\Projects\ccpp>

```

A callout box points to the terminal output with the text: "CPPと同様のメッセージを出してみた。「タスクの実行...」でgccすればOKのようだ。"

At the bottom of the terminal, the status bar shows: 行 24、列 3 タブのサイズ: 4 Shift JIS CRLF C Win32

モジュール設計仕様書	システム	Visual Studio Code	コードディタ、コンソール付だからテストもできる	作成日	2020/4/6
補足	モジュール	Hello World	こんにちは、World	作成者	hosiyama

DFD: Java バイトコードを生成するオブジェクト指向言語

The screenshot shows the Visual Studio Code interface with the following elements:

- Explorer:** Shows the file structure with 'HelloWorld.java' selected.
- Editor:** Displays the Java code for 'HelloWorld.java'. The code is as follows:

```
1  /**
2   * UTF8 $Workfile: $$Revision: $$Date: $
3   * $NoKeywords: $
4   *
5   * HelloWorld
6   */
7
8  public class HelloWorld {
9      public static void main(final String args[]) {
10         System.out.println("こんにちは、World");
11     }
12 }
13
```
- Terminal:** Shows the output of the program: 'こんにちは、World'.
- Debug Console:** Shows the output of the program: 'こんにちは、World'.
- Terminal:** Shows the prompt: 'PS J:\dev\Projects\Java>'.

A callout box highlights the code on line 10: `System.out.println("こんにちは、world");`. A tooltip shows the variable `String[0]@8` next to the `world` string.

デバッグコンソール 問題 出力 ターミナル

こんにちは、World
PS J:\dev\Projects\Java>

Writing Java with Visual Studio Code
<https://code.visualstudio.com/docs/java/java-tutorial>
を参考に、
"Hello"を"こんにちは"に変更した。
デバッグもいけそう。

行 10、列 1 スペース: 4 UTF-8 LF Java

モジュール設計仕様書	システム	Visual Studio Code	コードディタ、コンソール付だからテストもできる	作成日	2020/4/6
補足	モジュール	Hello World	こんにちは、World	作成者	hosiyama

DFD: C# Microsoft社.NET環境の開発言語

The screenshot shows the Visual Studio Code interface with the following components:

- Explorer (Left):** Shows the file structure for the project, including folders like `.vscode`, `bin`, `obj`, and `Debug`, and files like `csproj` files and `project.assets.json`. The file `HelloWorld.cs` is selected.
- Editor (Center):** Displays the source code of `HelloWorld.cs`. The code is as follows:

```
1 //SJIS $Workfile: $$Revision: $$Date: $
2 //$NoKeywords: $
3
4 using System;
5
6 namespace csharp
7 {
8     class HelloWorld
9     {
10         static void Main(string[] args)
11         {
12             Console.WriteLine("こんにちは、World");
13         }
14     }
15 }
```
- Terminal (Bottom):** Shows the command `dotnet run` being executed, resulting in the output `こんにちは、World`.

```
PS J:\dev\Projects\csharp> dotnet run
こんにちは、World
PS J:\dev\Projects\csharp>
```
- Debugger (Bottom):** Shows the execution flow with a breakpoint set at line 12. A tooltip for the `args` parameter shows its value as `{string[0]}`.
- Call Stack (Bottom):** Shows the current method `static void Main(string[] args)` at line 12.
- StatusBar (Bottom):** Shows the current file is `csharp`, and the cursor is at line 8, column 21.

A callout box with a blue border and a white background points to the debugger area, containing the text: `デバッグもいけそう。`