□□□□□□ 16.2 - C□□□□□□□□□□□□□□□

C□□から□び□されるプロシージャーのある□くの□□なユーティリティが□□するので、AdaからC□□をどのように□び□すかを□っておくのは□いアイディアです。このセクションは、プログラマーが□□の□□□なレベルでC□□について□っていることを□□しています。;
もしプログラマーがC□□について□らないのであれば、このセクションは□み□すだけで□□です。

□□に、ここでAdaとC□□をどのように□□させるかについての□□□なルールを□します。これは、RM B.3(63)
を□□としています。

1. Adaのプロシージャーは、voidを□すC□□に□□します。
2. Adaの□□は、voidでない□り□を□すC□□に□□します。
3. Adaの□□は、Cのポインター□の□□の□□に□□します。
4. □□なスカラ□は□integer、floatsとアクセス/ポインター□□は、□の□□の□□な□に□□します。

Ada
95は、C□□とのインターフェースを□□にするための□□□□パッケージのセットを□□しています。□□パッケージは、「Interfaces.C」で、AdaにおけるCの□に□する□□を□んでいます。C□□のint、long、unsigned、doubleの□が□まれています。C□□のfloat□は、「C_float」とAdaでは□ばれており、それは、AdaのFloat□と□□しないようにするためです。(AdaのFloatとC□□のfloatは、□□のように□えますが、□□はそうではないのです).

「char_array」□は、Cのcharacter□□を□□します。□くのC□□の□□は、character□□が□□なキャラクター「nul」で□わっていることを□□しています。□C□□では「□」と□かれます□。Adaの□□□は、□□、ヌルターミネートではないので、To_C□□とTo_Ada□□がAdaの□□□□とC□□のchar_array□の□□を□います。

Interfaces.C.StringsとInterfaces.C.PointersというC□□スタイルの□□□とポインター□のための□と□□を□□するための□□パッケージが□□します。□に、「Interfaces.C.Strings」は、「chars_ptr」□を□□しており、これはC□□の□□□な□「char*」に□□し、C□□の□□□へのポインター□に□□されます。□たとえば、キャラクター□□へのポインター□などです□

このパッケージは、また□□も□□しています。:

1. □□*Null_Ptr*、これはC□□の*(char*)NULL*に□□します。
2. プロシージャー*Free*、これはC□□の*free()*に□□します。
3. □□*Value*、これはchars_ptrを□□にとり、□□のAda□□□を□します。この□□は、ヌルポインター□が□された□□には、Dereference_Error□□を□げます。

□□の□を□して、プログラマーがこれらを□□にどのように□□するかを□ていくことにしましょう。この□は、「 package CGI 」からとったもので、AdaをWWWのコモン･ゲートウェイ･インターフェース□CGI□と□み□わせたものです。プログラマーがオペレーティングシステムから□□□□の□を□りたいと□えていると□□しましょう。そして、プログラマーは、□□のC□□がこれを□□することを□じて、□□□□の□を□ろうとしていることにします。C□□では、この□□は「getenv」と□ばれており、これは□のように□□されています。 (□の□□によります [Kernighan and Ritchie 1988, edition 2, page 253]):

```
char *getenv(char *name);
```

これは、かなりそのままAdaに□□することができます。:

```
function getenv(Variable : chars_ptr) return chars_ptr;  pragma Import(C, getenv);
```

これは□□しますが、Adaプログラムにおいてinとoutの□「chars_ptr」の□□された□を□□しなければならない□□には□□です。おそらく、ラッパープログラムを□いて、Ada□□□をC□□□(chars_ptr
)に□□し、まさその□をするようにする□がより□いでしょう。では、そのように□□するAdaの□□を□□します。:

```
with Interfaces.C.Strings; use Interfaces.C.Strings;  -- ...   function Get_Environment(Variable : String) return String is  -- Return the value of the given environment variable.  -- If there's no such environment variable, return an empty string.   function getenv(Variable : chars_ptr) return chars_ptr;   pragma Import(C, getenv);   -- getenv is a standard C library function; see K&R 2, 1988, page 253.   -- it returns a pointer to the first character; do NOT free its results.  Variable_In_C_Format : chars_ptr := New_String(Variable);   Result_Ptr : chars_ptr := getenv(Variable_In_C_Format);   Result : String := Value_Without_Exception(Result_Ptr); begin  Free(Variable_In_C_Format);  return Result;  end Get_Environment;
```

□□セクションにおいて、たくさんの□□□□□が□□することを□えておいてください。これは、□□の□□を□□するためであり、シンプルなAda□□□はそれらが□□□□されると□□の□さを□つためなのです。Value_Without_Exceptionと□ばれる□□の□□への□び□しが□□します。;
これは、□□ヌルのCのポインターを□□□にしようとしたときに、□□が□がるためです。そして、□たちはそれを□の□□□に□□したいと□えたわけです。このことは、□たちがそのような□□を□□しなければならないということを□□します。; ここで、その□□を□します。:

```
   function Value_Without_Exception(S : chars_ptr) return String is   -- Translate S from a
C-style char* into an Ada String.   -- If S is Null_Ptr, return "", don't raise an exception.   begin
if S = Null_Ptr then return "";      else return Value(S);     end if;   end Value_Without_Exception;
 pragma Inline(Value_Without_Exception);
```

さて、□たちはAdaにおいて□□□□を□□に□□できるようになりました。□えば、REQUEST_METHODという□□□□の□を□□したいのであれば、□のようにします。:

```
    Request_Method_Text : String := Get_Environment("REQUEST_METHOD");
```

□たちがカバ―していない□□に、C□□の□□□があります。Adaのレコ―ドとC□□の□□□は、はっきりと□□しています。しかし、どのように□□に□□しているのでしょうか。Ada RMは、Adaのレコ―ドが□に□□するC□□の□□□の□□に□し、ポインタ―として□されるべきとアドバイスを□えていますが、□□はしていません。
このため、□□□□まれではあるでしょうが□、C□□の□□が□□□を□□しとして□□しているようなケ―ス□□□□へのポインタ―の□わりにコピ―が□される□において、プログラマ―は、ポインタ―を□□の□□□に□□するような□しいC□□の□□を□き、そして、Adaからその□しいC□□の□□を□び□すようにします。しかしながら、これは□□なアドバイスです。GNATコンパイラ―は、このアドバイスに□っていません。 -
その□わり、GNATはAdaのレコ―ドを□□し□コピ―□します。どちらのプロ―チも□□□なものですが、□□なことにそれらは□なっています。Adaのレコ―ドを□すための□も□□なアプロ―チは、□に「access to record」の□を□すことです。-
それらが、スカラ―なのであれば、それらは□てのAdaコンパイラ―において□しく□されることが□□されています。

□□: http://www.adahome.com/Tutorials/Lovelace/s16s2.htm