原文はこちらをご覧下さい。

wxCmdLineParser

wxCmdLineParser はコマンドラインを構文解析するためのクラスである。

wxCmdLineParser is a class for parsing the command line.

以下の機能を持っている。

1. オプション、スイッチ、パラメータを識別し、オプションをグルーピングする。
2. 省略形、標準形オプションの両方を認める。
3. コマンドライン記述から、自動的に使用方法のメッセージを生成する。
4. オプション値などの型（数値、日付など）をチェックする。

It has the following features:
+ distinguishes options, switches and parameters; allows option grouping
+ allows both short and long options
+ automatically generates the usage message from the command line description
+ does type checks on the options values (number, date, ...).

以下の手順で使用する。

1. 構文解析するためのコマンドラインや、その説明を渡してこのクラスのオブジェクトを作成する。あるいは、後で AddXXX() 関数を使用する。
2. Parse() を呼ぶ。
3. 結果を取得するために Found() を使用する。

To use it you should follow these steps:
+ construct an object of this class giving it the command line to parse and optionally its description or use AddXXX() functions later
+ call Parse()
+ use Found() to retrieve the results

以下のドキュメントでは、以下の専門用語が使用されている。

スイッチ	指定されるかどうかの Boolean 型のオプションであり、値を持たない。以下に記載されているようなより一般的なオプションと区別するために、スイッチという言葉を使用している。例えば、-v は " 冗長モードの有効 " を意味するスイッチである。
オプション	ここでは、スイッチと異なり、値 0 にもなりうるものを意味している。例えば、-o: ファイル名は、出力ファイル名の指定を許可するオプションである。

パラメータ	要求されたプログラムの引数である。

In the documentation below the following terminology is used:

switch,This is a boolean option which can be given or not, but which doesn't have any value. We use the word switch to distinguish such boolean options from more generic options like those described below. For example, -v might be a switch meaning "enable verbose mode".

option,Option for us here is something which comes with a value 0 unlike a switch. For example, -o:filename might be an option which allows to specify the name of the output file.

parameter,This is a required program argument.

plugin::pdf::PDFParser=HASH(0x534d58) 基本クラス
基本クラスは無い。

plugin::pdf::PDFParser=HASH(0x534d58) インクルードファイル
<wx/cmdline.h>

plugin::pdf::PDFParser=HASH(0x534d58) 定数
構造体 wxCmdLineEntryDesc は１つのコマンドラインスイッチ、オプション、パラメータのいずれかを記述するために使用される。この構造体の配列が SetDesc() に渡される。また、AddXXX() 関数の引数は、この構造体の対応するフィールドと同じ意味である。

The structure wxCmdLineEntryDesc is used to describe the one command line switch, option or parameter. An array of such structures should be passed to SetDesc(). Also, the meanings of parameters of the AddXXX() functions are the same as of the corresponding fields in this structure:

```
struct wxCmdLineEntryDesc
{
    wxCmdLineEntryType kind;
    const wxChar *shortName;
    const wxChar *longName;
    const wxChar *description;
    wxCmdLineParamType type;
    int flags;
};
```

コマンドラインの種別は kind フィールドで示され、以下の定数のいずれかである。

The type of a command line entity is in the kind field and may be one of the following constants:

```
enum wxCmdLineEntryType
{
    wxCMD_LINE_SWITCH,
    wxCMD_LINE_OPTION,
    wxCMD_LINE_PARAM,
    wxCMD_LINE_NONE         // リストの終端に使用される
}
```

shortName フィールドは、スイッチやオプションの短縮形の名称に使用される。longName は、対応する標準形の名称であるか、オプションが標準名を持たない場合は NULL を指定する。パラメータの場合は、どちらも使用されない。短縮形も標準形も、オプション名には文字、数字、アンダースコア（_）だけを含めることができる。

The field shortName is the usual, short, name of the switch or the option. longName is the

corresponding long name or NULL if the option has no long name. Both of these fields are unused for the parameters. Both the short and long option names can contain only letters, digits and the underscores.

description はプログラムの構文を説明するヘルプメッセージを作成するために、Usage() メソッドで使用される。

description is used by the Usage() method to construct a help message explaining the syntax of the program.

type には、オプションまたはパラメータに対して、以下のいずれかを指定することができる。

The possible values of type which specifies the type of the value accepted by an option or parameter are:

```
enum wxCmdLineParamType
{
    wxCMD_LINE_VAL_STRING,  // default
    wxCMD_LINE_VAL_NUMBER,
    wxCMD_LINE_VAL_DATE,
    wxCMD_LINE_VAL_NONE
}
```

最後に、flags フィールドは、以下のビットマスクを組み合わせることができる。

Finally, the flags field is a combination of the following bit masks:

```
enum
{
    wxCMD_LINE_OPTION_MANDATORY = 0x01, // 必須オプション
    wxCMD_LINE_PARAM_OPTIONAL   = 0x02, // 省略可能
    wxCMD_LINE_PARAM_MULTIPLE   = 0x04, // 複数指定可能
    wxCMD_LINE_OPTION_HELP      = 0x08, // ヘルプ要求に対するオプション
    wxCMD_LINE_NEEDS_SEPARATOR  = 0x10, // 値の前にセパレータが必要
}
```

デフォルトでは（つまり、フラグが 0 の時）、オプションは任意である。そして、AddParam() を呼ぶことにより、フラグをデフォルト値から変更して、１つ以上のパラメータを指定できるようになる。すなわち、オプションを必須にするには、wxCMD_LINE_OPTION_MANDATORY を指定すればよく、パラメータを任意にするには、wxCMD_LINE_PARAM_OPTIONAL にすればよい。さらに、wxCMD_LINE_PARAM_MULTIPLE を指定すれば、プログラムで受け取るパラメータを可変にできる。しかし、コマンドライン記述内の最後のパラメータだけが得られる。このフラグを使用する場合、実際に指定されたパラメータの数を取得するために、Parse を読んだ後に GetParamCount を呼ぶ必要があるかも知れない。

Notice that by default (i.e. if flags are just 0), options are optional (sic) and each call to AddParam() allows one more parameter - this may be changed by giving non-default flags to it, i.e. use wxCMD_LINE_OPTION_MANDATORY to require that the option is given and wxCMD_LINE_PARAM_OPTIONAL to make a parameter optional. Also, wxCMD_LINE_PARAM_MULTIPLE may be specified if the programs accepts a variable number of parameters - but it only can be given for the last parameter in the command line description. If you use this flag, you will probably need to use GetParamCount to retrieve the number of parameters effectively specified after calling Parse.

最後のフラグである wxCMD_LINE_NEEDS_SEPARATOR は、オプション名と、その値との間にセパレータ（コロン、イコール、空白など）を指定する際に使用する。デフォルトでは、セパレー

タは要求されない。

> The last flag wxCMD_LINE_NEEDS_SEPARATOR can be specified to require a separator (either a colon, an equal sign or white space) between the option name and its value. By default, no separator is required.

plugin::pdf::PDFParser=HASH(0x534d58) 参考

wxApp::argc および wxApp::argv
console sample

plugin::pdf::PDFParser=HASH(0x534d58) 関数郡

Construction
Customization
Parsing command line
Getting results
wxCmdLineParser::wxCmdLineParser
wxCmdLineParser::wxCmdLineParser
wxCmdLineParser::wxCmdLineParser
wxCmdLineParser::wxCmdLineParser
wxCmdLineParser::wxCmdLineParser
wxCmdLineParser::wxCmdLineParser
wxCmdLineParser::ConvertStringToArgs
wxCmdLineParser::SetCmdLine
wxCmdLineParser::SetCmdLine
wxCmdLineParser::~wxCmdLineParser
wxCmdLineParser::SetSwitchChars
wxCmdLineParser::EnableLongOptions
wxCmdLineParser::DisableLongOptions
wxCmdLineParser::AreLongOptionsEnabled
wxCmdLineParser::SetLogo
wxCmdLineParser::SetDesc
wxCmdLineParser::AddSwitch
wxCmdLineParser::AddOption
wxCmdLineParser::AddParam
wxCmdLineParser::Parse
wxCmdLineParser::Usage
wxCmdLineParser::Found
wxCmdLineParser::Found
wxCmdLineParser::Found
wxCmdLineParser::Found
wxCmdLineParser::GetParamCount
wxCmdLineParser::GetParam

構築

Parse を呼ぶ前に、解析したいコマンドラインと、有効なスイッチ、オプション、およびパラメータのルール（以降、これをコマンドライン記述と呼ぶ）を wxCmdLineParser オブジェクトに指定する必要がある。

要求される情報をいつ指定するかは完全に自由であり、唯一の制約は、Parse を呼ぶ前に済ませておくということだけである。

解析したいコマンドラインを指定するためには、コマンドラインを引数に持つコンストラクタ（wxCmdLineParser(argc, argv)、または通常の wxCmdLineParser）のうちの１つを使用するか、または、デフォルトコンストラクタを使用する場合には後で SetCmdLine を呼んでも良い。

同じことがコマンドライン記述についても言える。コマンドライン記述もコンストラクタ（コマンドラインを引数に持たないものと、両方指定できるものがある）で指定するか、または、構築後に SetDesc を使用するか、AddSwitch、AddOption、及び AddParam メソッドを組み合わせて指定することもできる。

コンストラクタや SetDesc は、コマンドライン記述を含む（通常は const static な）テーブルを使用する。受け付けるオプションを実行時に決定したい場合には、上記 AddXXX() 関数のうちのどれかを使用する。

> Construction
>
> Before Parse can be called, the command line parser object must have the command line to parse and also the rules saying which switches, options and parameters are valid - this is called command line description in what follows.
>
> You have complete freedom of choice as to when specify the required information, the only restriction is that it must be done before calling Parse.
>
> To specify the command line to parse you may use either one of constructors accepting it (wxCmdLineParser(argc, argv) or wxCmdLineParser usually) or, if you use the default constructor, you can do it later by calling SetCmdLine.
>
> The same holds for command line description: it can be specified either in the constructor (without command line or together with it) or constructed later using either SetDesc or combination of AddSwitch, AddOption and AddParam methods.
>
> Using constructors or SetDesc uses a (usually const static) table containing the command line description. If you want to decide which options to accept during the run-time, using one of the AddXXX() functions above might be preferable.

カスタマイズ

wxCmdLineParser は、アプリケーションによって変更可能な幾つかのグローバルオプションを持っている。このセクションに記載した全ての関数は、Parse の前に呼ばなければならない。

１つめのグローバルオプションは、長い名称の（GNU スタイルとしても知られている）オプションに関するサポートである。長い名称のオプションは、２つのハイフン（"--"）で始まるオプションであり、例えば --verbose のようなオプションである。言い換えれば、それらは一般的に、完全な単語であり、省略できない。長い名称のオプションは多くのアプリケーションで使用されるため、デフォルトでは使用可能であるが、DisableLongOptions で無効にすることもできる。

別のグローバルオプションは、オプションの開始を指定する文字の集合である（これを指定しなければ、コマンドライン内の単語は全てパラメータとして認識される）。Unix では常に '-' が使用

されるが、Windows では少なくとも '-' と '/' の両方が使用される。'+' も使用するアプリケーションも存在する。デフォルトは、現在のプラットフォームに合わせたものが使用されるが、SetSwitchChars メソッドで変更することができる。

最後に、SetLogo は、Usage 関数によって得られる説明文の前に、アプリケーション固有のテキストを表示するために使用することができる。

Customization

wxCmdLineParser has several global options which may be changed by the application. All of the functions described in this section should be called before Parse.

First global option is the support for long (also known as GNU-style) options. The long options are the ones which start with two dashes ("--") and look like this: --verbose, i.e. they generally are complete words and not some abbreviations of them. As long options are used by more and more applications, they are enabled by default, but may be disabled with DisableLongOptions.

Another global option is the set of characters which may be used to start an option (otherwise, the word on the command line is assumed to be a parameter). Under Unix, '-' is always used, but Windows has at least two common choices for this: '-' and '/'. Some programs also use '+'. The default is to use what suits most the current platform, but may be changed with SetSwitchChars method.

Finally, SetLogo can be used to show some application-specific text before the explanation given by Usage function.

コマンドラインの解析

コマンドライン記述が構築され、解析対象のオプションが設定されたあと、ようやく Parse メソッドを呼ぶことができる。コマンドラインが正しく解析された場合、0 が返る。ヘルプオプションが指定されている場合、-1 が返る（プログラムは通常、このあと終了するため、場合分けしている）。コマンドライン解析時にエラーが発生すると、負の数が返る。

0 または -1 が返る場合、標準の wxWidgets ログ関数を使用して、対応するエラーメッセージと使用方法が wxCmdLineParser 自身によりロギングされる。

Parsing command line

After the command line description was constructed and the desired options were set, you can finally call Parse method. It returns 0 if the command line was correct and was parsed, -1 if the help option was specified (this is a separate case as, normally, the program will terminate after this) or a positive number if there was an error during the command line parsing.

In the latter case, the appropriate error message and usage information are logged by wxCmdLineParser itself using the standard wxWidgets logging functions.

結果の取得

Parse を呼んだあと（戻り値が 0 なら）、多重定義された Found() のうちの１つを使用して、解析結果にアクセスできる。

単純なスイッチの場合、スイッチが与えられたかどうかを知るためには、単に Found を呼ぶだけでよい。オプションやパラメータの場合、用意した変数に関連する値も返すような Found() を呼べばよい。全ての Found() 関数は、スイッチやオプションがコマンドラインに見つかった場合には true を返し、指定されなかった場合には false を返す。

Getting results

After calling Parse (and if it returned 0), you may access the results of parsing using one of overloaded Found() methods.

For a simple switch, you will simply call Found to determine if the switch was given or not, for an option or a parameter, you will call a version of Found() which also returns the associated value in the provided variable. All Found() functions return true if the switch or option were found in the command line or false if they were not specified.

wxCmdLineParser::wxCmdLineParser

wxCmdLineParser()

デフォルトコンストラクタ。後で SetCmdLine を使用しなければならない。

Default constructor. You must use SetCmdLine later.

wxCmdLineParser::wxCmdLineParser

wxCmdLineParser(int argc, char** argv)

wxCmdLineParser(int argc, wchar_t** argv)

解析するコマンドラインを渡すコンストラクタ。伝統的な（Unix の）コマンドラインフォーマットである。引数 argc と argv は、main() 関数と同じ意味を持つ。

２つ目の多重定義されたコンストラクタは、Unicode ビルドのときにのみ使用できる。１つ目のコンストラクタは、ANSI と Unicode の両方の環境で使用できる。なぜなら、幾つかのプラットフォームでは､Unicode プログラムであっても ASCII 文字列をコマンドライン引数に渡すことができるからである。

Constructor specifies the command line to parse. This is the traditional (Unix) command line format. The parameters argc and argv have the same meaning as for main() function.

The second overloaded constructor is only available in Unicode build. The first one is available in both ANSI and Unicode modes because under some platforms the command line arguments are passed as ASCII strings even to Unicode programs.

wxCmdLineParser::wxCmdLineParser

wxCmdLineParser(const wxString& cmdline)

解析するコマンドラインを Windows フォーマットで指定するためのコンストラクタ。引数 cmdline は WinMain() の引数と同じ意味を持つ。

Constructor specifies the command line to parse in Windows format. The parameter cmdline has the same meaning as the corresponding parameter of WinMain().

wxCmdLineParser::wxCmdLineParser

wxCmdLineParser(const wxCmdLineEntryDesc* desc)

wxCmdLineParser と同じであるが、コマンドライン記述も指定する。

Same as wxCmdLineParser, but also specifies the command line description.

wxCmdLineParser::wxCmdLineParser

wxCmdLineParser(const wxCmdLineEntryDesc* desc, int argc, char** argv)

wxCmdLineParser と同じであるが、コマンドライン記述も指定する。

Same as wxCmdLineParser, but also specifies the command line description.

wxCmdLineParser::wxCmdLineParser

wxCmdLineParser(const wxCmdLineEntryDesc* desc, const wxString& cmdline)

wxCmdLineParser と同じであるが、コマンドライン記述も指定する。

Same as wxCmdLineParser, but also specifies the command line description.

wxCmdLineParser::ConvertStringToArgs

static wxArrayString ConvertStringToArgs(const wxChar *cmdline)

完全なコマンドライン文字列を単語に分解する。単語は空白（半角スペース、タブ文字）によって区切られる。単語の中に空白を含める場合は、入力する文字列内でダブルクォーテーションで括ればよく、ダブルクォーテーションを含める場合にはバックスラッシュで括ればよい。（最後の分の後半、ソースを見る限り、バックスラッシュ＋ダブルクォーテーション（つまり、「\\"」という文字列）を、ダブルクォーテーションとして単語に含めるようです）。

Breaks down the string containing the full command line in words. The words are separated by whitespace. The quotes can be used in the input string to quote the white space and the back slashes can be used to quote the quotes.

wxCmdLineParser::SetCmdLine

void SetCmdLine(int argc, char** argv)

void SetCmdLine(int argc, wchar_t** argv)

コンストラクタでコマンドラインを指定しなかった場合に、後から解析するコマンドラインを設定する。２つ目の多重定義された関数は、Unicode ビルド時にのみ使用可能である。

Set command line to parse after using one of the constructors which don't do it. The second overload of this function is only available in Unicode build.

plugin::pdf::PDFParser=HASH(0x534d58) 参考
wxCmdLineParser

wxCmdLineParser::SetCmdLine

void SetCmdLine(const wxString& cmdline)

コンストラクタでコマンドラインを指定しなかった場合に、後から解析するコマンドラインを設定する。

Set command line to parse after using one of the constructors which don't do it.

plugin::pdf::PDFParser=HASH(0x534d58) 参考
wxCmdLineParser

wxCmdLineParser::~wxCmdLineParser

~wxCmdLineParser()

オブジェクトにより確保されたりソースを開放する。
注意：デストラクタは仮想関数ではないため、このクラスを継承してはならない。

Frees resources allocated by the object.
NB: destructor is not virtual, don't use this class polymorphically.

wxCmdLineParser::SetSwitchChars

void SetSwitchChars(const wxString& switchChars)

switchChars には、オプションやスイッチの開始文字として取りうる全ての文字を含める。デフォルトは、Unix では "-"、Windows では "-/" である。

switchChars contains all characters with which an option or switch may start. Default is "-" for Unix, "-/" for Windows.

wxCmdLineParser::EnableLongOptions

void EnableLongOptions(bool enable = true)

標準形式のサポートを有効化／無効化する。
標準形式は（まだ）POSIX 対応していないため、それらを無効化することだけが許されている。

Enable or disable support for the long options.
As long options are not (yet) POSIX-compliant, this option allows to disable them.

plugin::pdf::PDFParser=HASH(0x534d58) 参考

Customization and AreLongOptionsEnabled

wxCmdLineParser::DisableLongOptions

void DisableLongOptions()

EnableLongOptions(false) と同じ。

Identical to EnableLongOptions(false).

wxCmdLineParser::AreLongOptionsEnabled

bool AreLongOptionsEnabled()

標準形オプションが有効な場合には true を返し、そうでなければ false を返す。

Returns true if long options are enabled, otherwise false.

plugin::pdf::PDFParser=HASH(0x534d58) 参考
EnableLongOptions

wxCmdLineParser::SetLogo

void SetLogo(const wxString& logo)

logo は Usage メソッドで表示される特殊な文字列である。

logo is some extra text which will be shown by Usage method.

wxCmdLineParser::SetDesc

void SetDesc(const wxCmdLineEntryDesc* desc)

コマンドライン記述を構築する。
wxCMD_LINE_NONE を最後の要素に持つテーブルからコマンドライン記述を取得する。

Construct the command line description
Take the command line description from the wxCMD_LINE_NONE terminated table.

使用例：

```
static const wxCmdLineEntryDesc cmdLineDesc[] =
{
    { wxCMD_LINE_SWITCH, "v", "verbose", "be verbose" },
    { wxCMD_LINE_SWITCH, "q", "quiet",   "be quiet" },

    { wxCMD_LINE_OPTION, "o", "output",  "output file" },
    { wxCMD_LINE_OPTION, "i", "input",   "input dir" },
    { wxCMD_LINE_OPTION, "s", "size",    "output block size", wxCMD_LINE_VAL_NUMBER },
    { wxCMD_LINE_OPTION, "d", "date",    "output file date", wxCMD_LINE_VAL_DATE },
```

```
        { wxCMD_LINE_PARAM,  NULL, NULL, "input file", wxCMD_LINE_VAL_STRING, wxCMD_LINE_PARAM_MULTIPLE
    },

        { wxCMD_LINE_NONE }
    };

    wxCmdLineParser parser;

    parser.SetDesc(cmdLineDesc);
```

wxCmdLineParser::AddSwitch

void AddSwitch(const wxString& name, const wxString& lng = wxEmptyString, const wxString& desc = wxEmptyString, int flags = 0)

コマンドライン記述に対して、desc で渡された記述と flag で渡されたフラグ、lng で渡された長いオプション名称（デフォルトでは空文字であり、長いオプション名を持たない）を指定しつつ、スイッチ name を追加する。

Add a switch name with an optional long name lng (no long name if it is empty, which is default), description desc and flags flags to the command line description.

wxCmdLineParser::AddOption

void AddOption(const wxString& name, const wxString& lng = wxEmptyString, const wxString& desc = wxEmptyString, wxCmdLineParamType type = wxCMD_LINE_VAL_STRING, int flags = 0)

コマンドライン記述に対して、type で渡された値（デフォルトでは文字列）で、長いオプション名称 lng（デフォルトでは空文字であり、長いオプション名を持たない）を指定しつつ、オプション name を追加する。

Add an option name with an optional long name lng (no long name if it is empty, which is default) taking a value of the given type (string by default) to the command line description.

wxCmdLineParser::AddParam

void AddParam(const wxString& desc = wxEmptyString, wxCmdLineParamType type = wxCMD_LINE_VAL_STRING, int flags = 0)

コマンドライン記述に対して、type で渡されたパラメータを追加する。

Add a parameter of the given type to the command line description.

wxCmdLineParser::Parse

int Parse(bool giveUsage = true)

コマンドラインを解析する。正常時は 0 を返し、ヘルプメッセージが登録されている状態で "-h" や "--help" オプションが指定されると -1 を返し、シンタックスエラーが発生すると正の数を返す。

Parse the command line, return 0 if ok, -1 if "-h" or "--help" option was encountered and the help message was given or a positive value if a syntax error occurred.

plugin::pdf::PDFParser=HASH(0x534d58) 引数

giveUsage

true の場合（デフォルト）、コマンドライン解析中にシンタックスエラーが発生したり、ヘルプが要求された時に、使用方法を取得する。false の場合、シンタックスエラーが発生したときに、エラーメッセージを表示するだけであり、使用方法を表示する場合には必要に応じて呼び出し元が Usage を使用する。

If true (default), the usage message is given if a syntax error was encountered while parsing the command line or if help was requested. If false, only error messages about possible syntax errors are given, use Usage to show the usage message from the caller if needed.

wxCmdLineParser::Usage

void Usage()

全てのプログラムオプションを記述する、標準的な使用方法メッセージを得る。事前に指定されたオプションやパラメータの記述を使用する。そのため、その記述が本当に指定されたものでなければ、その結果文字列は、ユーザにとって有用ではなくなる。

Give the standard usage message describing all program options. It will use the options and parameters descriptions specified earlier, so the resulting message will not be helpful to the user unless the descriptions were indeed specified.

plugin::pdf::PDFParser=HASH(0x534d58) 参考

SetLogo

wxCmdLineParser::Found

bool Found(const wxString& name) const

渡されたスイッチが見つかれば true を返し、そうでなければ false を返す。

Returns true if the given switch was found, false otherwise.

wxCmdLineParser::Found

bool Found(const wxString& name, wxString* value) const

文字列型のオプションを見つけたら true を返し、与えられたポインタ（NULL であってはならない）にその値を格納する。

Returns true if an option taking a string value was found and stores the value in the provided pointer (which should not be NULL).

wxCmdLineParser::Found

bool Found(const wxString& name, long* value) const

整数型のオプションを見つけたら true を返し、与えられたポインタ（NULL であってはならない）にその値を格納する。

Returns true if an option taking an integer value was found and stores the value in the provided pointer (which should not be NULL).

wxCmdLineParser::Found

bool Found(const wxString& name, wxDateTime* value) const

日付型のオプションを見つけたら true を返し、与えられたポインタ（NULL であってはならない）にその値を格納する。

Returns true if an option taking a date value was found and stores the value in the provided pointer (which should not be NULL).

wxCmdLineParser::GetParamCount

size_t GetParamCount() const

見つけたパラメータの数を返す。wxCMD_LINE_PARAM_MULTIPLE フラグが指定された場合に、意味をなす。

Returns the number of parameters found. This function makes sense mostly if you had used wxCMD_LINE_PARAM_MULTIPLE flag.

wxCmdLineParser::GetParam

wxString GetParam(size_t n = 0u) const

N 番目のパラメータの値を返す（当分は文字列を返す）。

Returns the value of Nth parameter (as string only for now).

plugin::pdf::PDFParser=HASH(0x534d58) 参考
GetParamCount