Syuhitu.org

主筆　The text editor for Solaris
Plugin Development Guide

目次

１．このマニュアルについて

１．１　概要

このマニュアルは、Sun Solaris 用テキストエディタ主筆で使用する、プラグインを開発する上での規約・ＡＰＩについて記述している。

１．２　対象となる読者

このマニュアルは、主筆で使用するプラグインの開発者を対象としている。

１．３　前提知識

このマニュアルでは、読者は下記の事項に関する一般的な知識を有することを前提としている。

- Unix 上の C 言語によるソフトウェア開発一般
- Unix におけるシステム管理一般
- 主筆の操作・管理一般

１．４　商標などについて

このマニュアルに記載されている会社名、商品名、製品名などは、一般に各社の商標または登録商標です。

２．規約

２．１　形式

主筆のプラグインは、動的共有ライブラリとして作成されます。一般に拡張子は so となります。
Forte C++を使用する場合は、コンパイル時にスイッチ「-G」を使用してください。

２．２　ヘッダファイル

主筆のプラグインを作成するに当たり必要となる宣言は「PluginFuncID.h」に記述されています。

２．３　公開する関数

主筆のプラグインは下記の形式の関数を公開しなければなりません。

```
void 関数名( PFT_GetAPIFunction pGetAPIFunction );
```

「関数名」はプラグイン設定ファイルの FunctionName により指定されます。主筆はメニューが選択されると、指定されたライブラリをロードし、指定された名前の関数を呼び出します。

この呼ばれた関数からリターンすると、プラグインの処理は終了します。

引数 pGetAPIFunction には下記の形式の関数のアドレスが渡されます。

```
void* pGetAPIFunction( int FunctionType );
```

この関数は、主筆が公開する API のアドレスを返します。求める関数の種類を引数 FunctionType に指定してください。

２．４　グローバル変数

グローバル変数は使用しないでください。

複数回の呼び出しにまたがって、共有の値を参照したい場合には PFID_ALLOCWORKMEMORY の API 関数を用いて、作業用メモリ領域を確保してください。

３．ＡＰＩ

主筆のプラグインは、主筆から呼びだされた関数に渡された引数により、主筆が公開するＡＰＩの関数ポインタを取得することができます。

３．１　宣言

「PluginFuncID.h」には、各関数ポインタの型と関数の種別を示す定数以外に、下記の値とデータ型が定義されています。

３．１．１　データ型

PFT_BOOL	真偽値型です。

３．１．２　定数

PFT_TRUE	真を示す定数です
PFT_FALSE	偽を示す定数です

３．２　ＡＰＩ

主筆は、プラグインに対し下記のＡＰＩ関数を提供しています。

種別	型	概要
PFID_GETCHAR	PFT_GetChar	文字を取得します。
PFID_SETCHAR	PFT_SetChar	文字を設定します。
PFID_GETSTRING	PFT_GetString	文字列を取得します。
PFID_REPLACE	PFT_Replace	文字列を置換します。
PFID_GETLINECOUNT	PFT_GetLineCount	行数を取得します。
PFID_GETCHARCOUNT	PFT_GetCharCount	指定した行に含まれる文字数を取得します。
PFID_GETCURSORPOSITION	PFT_GetCurPosition	カーソルの位置を取得します。
PFID_SETCURSORPOSITION	PFT_SetCurPosition	カーソルの位置を設定します。
PFID_GETSELECTIONRANGE	PFT_GetSelectionRange	選択範囲を取得します。
PFID_SETSELECTIONRANGE	PFT_SetSelectionRange	選択範囲を設定します。
PFID_GETCONFIGVALUE	PFT_GetConfigValue	プラグイン設定ファイルの設定値を取得します。
PFID_GETFILENAME	PFT_GetFileName	ファイル名を取得します。
PFID_SHOWINFORMATIONMSGBOX	PFT_ShowInformationMsgBox	情報メッセージボックスを表示します。
PFID_SHOWQUESTIONMSGBOX	PFT_ShowQuestionMsgBox	質問メッセージボックスを表示します。
PFID_SHOWERRORMSGBOX	PFT_ShowErrorMsgBox	エラーメッセージボックスを表示します。
PFID_GETMODIFIEDFLG	PFT_GetModifiedFlg	更新フラグを取得します。
PFID_GETVERSION	PFT_GetVersion	主筆のバージョンを取得します。
PFID_GETEXTENDINFOCOUNT	PFT_GetExtendInfoCount	利用可能な拡張情報領域の数を取得します。
PFID_GETEXTENDINFO	PTF_GetExtendInfo	拡張情報領域の値を取得します。
PFID_SETEXTENDINFO	PTF_SetExtendInfo	拡張情報領域に値を設定します。
PFID_SAVEFILE	PFT_SaveFile	ファイルを保存します。
PFID_OPENFILE	PFT_OpenFile	ファイルを開きます。
PFID_OPENFILENEWWINDOW	PFT_OpenFileNewWindow	新しいウインドウでファイルを開きます。
PFID_ALLOCWORKMEMORY	PFT_AllocWorkMemory	新規に作業用メモリ領域を確保します。
PFID_FINDWORKMEMORY	PFT_FindWorkMemory	既存の作業用メモリ領域を検索します。
PFID_FREEWORKMEMORY	PFT_FreeWorkMemory	既存の作業用メモリ領域を開放します。

３．２．１　PFID_GETCHAR

指定した位置の文字を取得する。

```
PFT_BOOL (*PFT_GetChar)(
        unsigned long line,
        unsigned long cpos,
        wchar_t* pChar
);
```

型

PFT_GetChar

概要

line 行目 cpos 文字目の文字を*pChar に取得します。

引数

line

行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

cpos

line 行内の何番目の文字を取得するか、を指定します。指定可能な値は 0 から line 行目に存在する文字数-1 までです。

pChar

文字を取得するための変数のアドレスを指定します。

戻り値

文字の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．２　PFID_SETCHAR

指定した位置に文字を設定する。

```
PFT_BOOL (*PFT_SetChar)(
        unsigned long line,
        unsigned long cpos,
        wchar_t c
);
```

型

PFT_SetChar

概要

line 行目 cpos 文字目に文字 c を設定します。

引数

line

行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

cpos

line 行内の何番目の文字を取得するか、を指定します。指定可能な値は 0 から line 行目に存在する文字数-1 までです。

c

設定する文字を指定します。

戻り値

文字の設定に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．３　PFID_GETSTRING

指定した範囲の文字列を取得する。

```
PFT_BOOL (*PFT_GetString)(
        unsigned long SLP,
        unsigned long SCP,
        unsigned long ELP,
        unsigned long ECP,
        wchar_t* pBuf,
        unsigned long BufLength
);
```

型

PFT_GetString

概要

SLP 行目 SCP 文字目から ELP 行目 ECP-1 文字目までの文字列を pBuf が示すアドレスに取得します。

文字列の末尾には'\0'が設定されます。

もし、取得される文字列が BufLength より長かった場合には、バッファに格納することができた位置までの文字列が返されます。

引数

SLP

文字列の取得を開始する行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

SCP

SLP 行内の何番目の文字から取得するか、を指定します。指定可能な値は 0 から SLP 行目に存在する文字数-1 までです。

ELP

文字列の取得を終了する行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

ECP

ELP 行内の何番目の文字まで取得するか、を指定します。指定可能な値は 0 から SLP 行目に存在する文字数までです。 なお、。取得される文字列には ELP 行目 ECP 文字目の文字は含まれません。

pBuf

文字列を取得するためのメモリ領域のアドレスを指定します。

BufLength

pBuf に用意されたバッファの長さ (文字数) を指定します。

戻り値

文字列の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．４　PFID_REPLACE

指定した範囲の文字列を置換する。

```
PFT_BOOL (*PFT_Replace)(
        unsigned long SLP,
        unsigned long SCP,
        unsigned long ELP,
        unsigned long ECP,
        const wchar_t* pBuf
);
```

型

PFT_Replace

概要

SLP 行目 SCP 文字目から ELP 行目 ECP-1 文字目までの文字列を pBuf に指定された文字列と置換します。

pBuf に指定された、置換後の文字列の末尾には'\0'が設定されている必要があります。

引数

SLP

置換範囲の開始位置の行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

SCP

置換範囲が SLP 行内の何番目の文字から開始されるか、を指定します。指定可能な値は 0 から SLP 行目に存在する文字数-1 までです。

ELP

置換範囲の終了位置の行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

ECP

置換範囲が ELP 行内の何番目の文字で終了されるか、を指定します。指定可能な値は 0 から SLP 行目に存在する文字数までです。 なお、置換される文字列には ELP 行目 ECP 文字目の文字は含まれません。

pBuf

文字列を指定します。

戻り値

文字列の置換に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．５　PFID_GETLINECOUNT

行数を取得する。

```
PFT_BOOL (*PFT_GetLineCount)(
        unsigned long* pCnt
);
```

型

PFT_GetLineCount

概要

pCnt に指定した変数に行数を取得します。

引数

pCnt

行数を取得するための変数のアドレスを指定します。

戻り値

行数の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．６　PFID_GETCHARCOUNT

指定した行の文字数を取得する。

```
PFT_BOOL (*PFT_GetCharCount)(
        unsigned long LP,
        unsigned long* pCnt
);
```

型

PFT_GetCharCount

概要

LP 行目に含まれる文字の文字数を pCnt に指定した変数に取得します。

取得される文字数には、行の末尾に存在する改行コードも含みます。ただし、一番最後の行には改行コードは存在しません。

引数

LP

文字数を取得する行の行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

pCnt

文字数を取得するための変数のアドレスを指定します。

戻り値

文字数の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．７　PFID_GETCURSORPOSITION

現在のカーソルの位置を取得する。

```
PFT_BOOL (*PFT_GetCursorPosition)(
        unsigned long* pLP,
        unsigned long* pCP
);
```

型

PFT_GetCursorPosition

概要

現在のカーソルの位置を取得します。

引数

pLP

カーソルが存在する位置の行番号を取得するための、変数のアドレスを指定します。

pCP

カーソルが存在する位置の文字位置を取得するための、変数のアドレスを指定します。

戻り値

カーソル位置の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．８　PFID_SETCURSORPOSITION

カーソルの位置を設定します。

```
PFT_BOOL (*PFT_SetCursorPosition)(
        unsigned long LP,
        unsigned long CP
);
```

型

PFT_SetCursorPosition

概要

カーソルの位置を設定します。

引数

LP

カーソルを設定する位置の行番号を指定します。指定可能な値は 0 から総行数-1 までです。

CP

カーソルを設定する位置の文字位置を指定します。指定可能な値は 0 から LP 行目に含まれる文字数-1 までです。

戻り値

カーソル位置の設定に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．９　PFID_GETSELECTIONRANGE

現在の選択範囲を取得します。

```
PFT_BOOL (*PFT_GetSelectionRange)(
        unsigned long* pSLP,
        unsigned long* pSCP,
        unsigned long* pELP,
        unsigned long* pECP
);
```

型

PFT_GetSelectionRange

概要

現在の選択範囲を取得します。

引数

pSLP

選択範囲の開始位置の行番号を取得するための、変数のアドレスを指定します。

pSCP

選択範囲の開始位置の文字位置を取得するための、変数のアドレスを指定します。

pELP

選択範囲の終了位置の行番号を取得するための、変数のアドレスを指定します。

pECP

選択範囲の終了位置の文字位置を取得するための、変数のアドレスを指定します。 なお、(*pELP)行目(*pECP) 文字目の文字は選択範囲に含まれません。

戻り値

選択範囲の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．１０　PFID_SETSELECTIONRANGE

選択範囲を設定します。

```
PFT_BOOL (*PFT_SetSelectionRange)(
        unsigned long SLP,
        unsigned long SCP,
        unsigned long ELP,
        unsigned long ECP
);
```

型

PFT_SetSelectionRange

概要

選択範囲を設定します。

引数

pSLP

選択範囲の開始位置の行番号を指定します。

pSCP

選択範囲の開始位置の文字位置を指定します。

pELP

選択範囲の終了位置の行番号を指定します。

pECP

選択範囲の終了位置の文字位置を指定します。 なお、ELP 行目 ECP 文字目の文字は選択範囲に含まれません。

戻り値

選択範囲の設定に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

３．２．１１　PFID_GETCONFIGVALUE

プラグイン設定ファイルに記述された設定情報を取得する

```
const wchar_t* (*PFT_GetConfigValue)(
        const wchar_t* pKey
);
```

型

PFT_GetConfigValue

概要

プラグイン設定ファイルに記述された設定情報を参照します。参照可能な値は、呼び出されたプラグインについて記述しているセクションに属する値のみです。

たとえば、プラグイン設定ファイルに下記のように記述されていて、

```
[MyPlugin1]
PluginName = MyPlugin1
LibraryName = libmyplugin.so
FunctionName = foo
MenuLabel = プラグイン１
ConfigValue1 = あいうえお
[MyPlugin2]
PluginName = MyPlugin2
LibraryName = libmyplugin.so
FunctionName = foo
MenuLabel = プラグイン２
ConfigValue2 = かきくけこ
```

ユーザが「プラグイン１」を選択した時に、 キーとして”ConfigValue1”を指定してこの関数を呼び出した場合は、「あいうえお」の値が返ってきます。 また、キーに”ConfigValue2”を指定した場合にはNULLが返ってきます。 この例のように、呼び出しているライブラリと関数が同じであっても、別のセクションの値を参照することはできません。

引数

pKey

値を検索するためのキーを指定します。

戻り値

値の取得に成功した場合は、そのアドレスが返されます。そうでない場合はNULLが返されます。なお、返される値は内部で保持している値のアドレスなので、プラグイン側では解放しないでください。

互換性

第９版以降

３．２．１２　PFID_GETFILENAME

ファイル名を取得します。

```
const char* (*PFT_GetFileName)();
```

型

PFT_GetFileName

概要

現在開いているファイルのファイル名を取得します。 もしファイル名が未定ならば、長さ 0 の文字列が返されます。 NULL が返されることはありません。

戻り値

ファイル名の文字列のアドレスが返されます。この文字列は内部で保持している値なので、プラグイン側で解放しないでください。

互換性

第 9 版以降

３．２．１３　PFID_SHOWINFORMATIONMSGBOX

情報メッセージボックスを表示します。

```
void (*PFT_ShowInformationMsgBox)(
        const wchar_t* pMsg
);
```

型

PFT_ShowInformationMsgBox

概要

モーダルな情報メッセージボックスを表示させます。 ユーザが「OK」ボタンを押下するまで制御が戻りません。

引数

pMsg

表示するメッセージを指定します。

互換性

第１０版以降

３．２．１４　PFID_SHOWQUESTIONMSGBOX

質問メッセージボックスを表示します。

```
int (*PFT_ShowInformationMsgBox)(
        const wchar_t* pMsg,
        PFT_BOOL ShowCancel
);
```

型

PFT_ShowInformationMsgBox

概要

モーダルな質問メッセージボックスを表示させます。ユーザが「はい」「いいえ」「キャンセル」のいずれかのボタンを押下するまで制御が戻りません。

引数

pMsg

表示するメッセージを指定します。

ShowCancel

真を設定した場合には「キャンセル」ボタンが表示されます。

戻り値

ユーザが押下したボタンに応じて、下記の値が返されます。

ボタン	値
はい	1
いいえ	2
キャンセル	3

互換性

第１０版以降

３．２．１５　PFID_SHOWERRORMSGBOX

エラーメッセージボックスを表示する

```
void (*PFT_ShowErrorMsgBox)(
        const wchar_t* pMsg
);
```

型

PFT_ShowErrorMsgBox

概要

モーダルなエラーメッセージボックスを表示させます。 ユーザが「OK」ボタンを押下するまで制御が戻りません。

引数

pMsg

表示するメッセージを指定します。

互換性

第１０版以降

３．２．１６　PFID_GETMODIFIEDFLG

更新フラグを取得します。

```
PFT_BOOL (*PFT_GetModifiedFlg)();
```

型

PFT_GetModifiedFlg

概要

更新フラグを取得します。最後に更新されてから変更が加えられていた場合には真を返します。

戻り値

最後に保存されてから更新されていた場合には真、保存されて以降変更されていなければ偽が返されます。

互換性

第１２版以降

３．２．１７　PFID_GETVERSION

主筆のバージョンを取得します。

```
int (*PFT_GetVersion)();
```

型

PFT_GetVersion

概要

プラグインを呼び出した主筆のバージョン番号を返します。例えば、第１３版であれば13が返されます。

戻り値

バージョン番号が返されます。

互換性

第１２版以降

３．２．１８　PFID_GETEXTENDINFOCOUNT

拡張情報領域の数を取得します。

```
int (*PFT_GetExtendInfoCount)();
```

型

PFT_GetExtendInfoCount

概要

現在利用可能な拡張情報領域の数を取得します。

拡張情報領域の数はリソースファイルの extendInfoColumnCount によって設定されます。

戻り値

利用可能な拡張情報領域の数が返されます。

互換性

第１２版以降

３．２．１９　PFID_GETEXTENDINFO

拡張情報領域の値を取得します。

```
PFT_BOOL (*PTF_GetExtendInfo)(
          unsigned long LP,
          unsigned long idx,
          PFT_BOOL* pVal
);
```

型

PFT_GetExtendInfoCount

概要

拡張情報領域に設定されている値を取得します。

拡張情報領域は、各行ごとに複数個の真偽値を保持することが可能な領域です。また、拡張情報領域に真が設定されると、画面の左端にマークが表示され、ユーザは拡張情報領域のステータスを確認することができます。

引数

LP

値を取得する行の行番号を指定してください。

idx

何番目の拡張情報領域の値を取得するのか、を指定してください。指定可能な値は0からPFID_GETEXTENDINFOCOUNTで取得される値-1までです。

pVal

値を取得するための変数のアドレスを指定してください。

戻り値

処理に成功したら真が返されます。

互換性

第１２版以降

３．２．２０　PFID_SGETEXTENDINFO

拡張情報領域に値を設定します。

```
PFT_BOOL (*PTF_SetExtendInfo)(
          unsigned long LP,
          unsigned long idx,
          PFT_BOOL Val
);
```

型

PFT_SetExtendInfoCount

概要

拡張情報領域に値を設定します。

拡張情報領域は、各行ごとに複数個の真偽値を保持することが可能な領域です。また、拡張情報領域に真が設定されると、画面の左端にマークが表示され、ユーザは拡張情報領域のステータスを確認することができます。

引数

LP

値を設定する行の行番号を指定してください。

idx

何番目の拡張情報領域に値を設定するのか、を指定してください。指定可能な値は0からPFID_GETEXTENDINFOCOUNTで取得される値-1までです。

pVal

設定する値を指定してください。

戻り値

処理に成功したら真が返されます。

互換性

第１２版以降

３．２．２１　PFID_SAVEFILE

ファイルを保存します。

```
PFT_BOOL (*PFT_SaveFile)();
```

型

PFT_SaveFile

概要

ファイルを保存します。

もし、現在開いているファイルに名前がなかった場合には、「名前を付けて保存」ダイアログボックスが表示されます。

戻り値

ファイルの保存に成功した場合には真が返されます。

「名前を付けて保存」ダイアログでユーザがキャンセルした、ファイル保存時に実行するコマンドによりファイルの保存がキャンセルされた、もしくは何らかの理由でファイルの保存に失敗した場合、には偽が返されます。

互換性

第１２版以降

３．２．２２　PFID_OPENFILE

ファイルを開きます。

```
PFT_BOOL (*PFT_OpenFile)(
          const char* pFileName
);
```

型

PFT_OpenFile

概要

指定された名前のファイルを開きます。

もし pFileName に長さ 0 の文字列が指定された場合には、「ファイルを開く」ダイアログボックスを表示します。NULL を指定することはできません。

引数

pFileName

ファイル名を指定してください。

相対パスが指定された場合には、プロセスのカレントディレクトリからの相対パスが使用されます。

長さ 0 の文字列が指定された場合には、「ファイルを開く」ダイアログボックスが表示されます。

NULL が指定された場合には、関数の処理は失敗します。

戻り値

正常にファイルを開くことができた場合には真が返されます。

「ファイを開く」ダイアログでユーザがキャンセルした、ファイルオープン時に実行するコマンドによりファイルのオープンがキャンセルされた、もしくは何らかの理由でファイルのオープンに失敗した場合、には偽が返されます。

互換性

第１２版以降

３．２．２３　PFID_OPENFILENEWWINDOW

ファイルを新しいウインドウで開きます。

```
PFT_BOOL (*PFT_OpenFileNewWindow)(
            const char* pFileName
);
```

型

PFT_OpenFileNewWindow

概要

指定された名前のファイルを新しいウインドウで開きます。

もし pFileName に長さ 0 の文字列が指定された場合には、新しいウインドウが開かれ、新規のファイルが作成されます。

主筆は新しいウインドウを開くために新規にプロセスを立ち上げます。

引数

pFileName

ファイル名を指定してください。

相対パスが指定された場合には、プロセスのカレントディレクトリからの相対パスが使用されます。

長さ 0 の文字列が指定された場合には、新しいウインドウが開かれ、新規のファイルが作成されます。

NULL が指定された場合には、関数の処理は失敗します。

戻り値

新しいウインドウを開くことに成功した場合には真が返ります。

ファイルのオープンそのものに失敗した場合でも、新規のウインドウを開くことに成功してさえいれば真が返されます。新規のウインドウを開くことができなかった場合には偽が返されます。

互換性

第１２版以降

３．２．２４　PFID_ALLOCWORKMEMORY

作業用メモリを確保します。

```
void* (*PFT_AllocWorkMemory)(
          unsigned long size
          const wchar_t* pName,
          PFT_BOOL IsGlobal
);
```

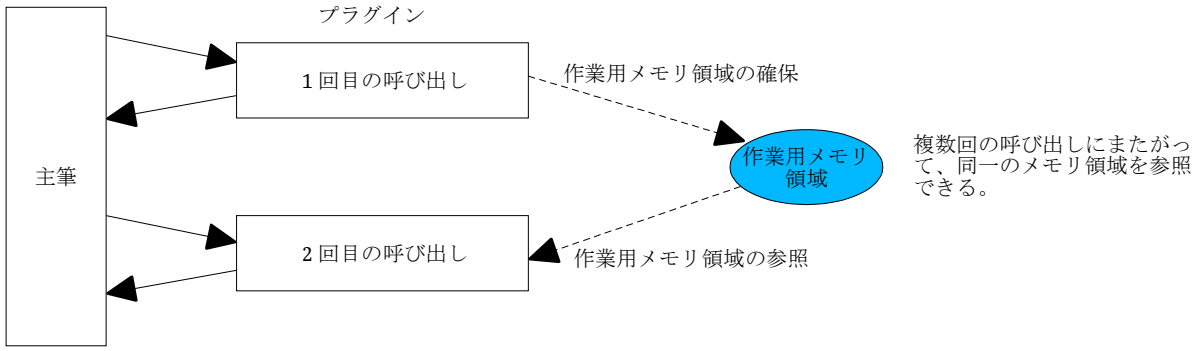
型

PFT_AllocWorkMemory

概要

作業用のメモリ領域を確保します。

このAPIで確保された作業用メモリ領域は、PFID_FINDWORKMEMORYにより名前を指定することで、メモリ領域のアドレス値を参照することができます。また、確保されたメモリ領域は、主筆のプロセスが終了されるまで継続して使用することができます。そのため、プラグインの複数回の呼び出しにわたって継続して値を保持したい場合に使用することができます。



IsGlobalに真を設定した場合は、pNameに指定した名前を用いて全プラグインで同一のメモリ領域を参照することができます。IsGlobalに偽を設定した場合は、メモリ領域を確保したプラグインでのみ参照することができます。プラグイン間での通信を行う場合に使用することができます。

このAPI関数によるメモリ領域の確保は、一回の呼び出しでメモリ領域を確保して解放する場合には、あまり適しません。そのような場合には通常のmallocを使用してください。

このAPI関数ではメモリ領域の初期化を行いません。

引数

size

確保するメモリ領域長をバイト単位で指定してください。

pName

作業用メモリ領域の名前を指定してください。

size

全プラグイン間で共通して参照可能にする場合は真を指定してください。

戻り値

確保されたメモリ領域のアドレスが返されます。

失敗した場合はNULLが返されます。

互換性

第１２版以降

３．２．２５　PFID_FINDWORKMEMORY

既存の作業用メモリ領域を参照します。

```
void* (*PFT_FindWorkMemory)(
          const wchar_t* pName,
          PFT_BOOL IsGlobal
);
```

型

PFT_FindWorkMemory

概要

PFID_ALLOCWORKMEMORY で確保された作業用メモリ領域を pName で検索し、アドレスを返します。

引数

pName

作業用メモリ領域の名前を指定してください。

戻り値

検索された作業用メモリ領域のアドレスが返されます。

指定された名前の作業用メモリ領域が存在しない場合には NULL が返されます。

互換性

第１２版以降

３．２．２６　PFID_FREEWORKMEMORY

既存の作業用メモリ領域を解放します。

```
void* (*PFT_FreeWorkMemory)(
          const wchar_t* pName,
          PFT_BOOL IsGlobal
);
```

型

PFT_FreeWorkMemory

概要

PFID_ALLOCWORKMEMORY で確保された作業用メモリ領域を pName で検索し、その作業用メモリ領域を開放します。

不要になった作業用メモリ領域はこの API 関数で解放してください。

指定された名前の作業用メモリ領域が存在しない場合には、この関数は何も処理を行いません。

引数

pName

作業用メモリ領域の名前を指定してください。

互換性

第１２版以降

４．例

下記に、簡単なプラグインの例を示します。

４．１　プログラム

プラグインとして、呼び出されたときに「Welcome to Syuhitu plugin hell world」というメッセージ表示します。

msg.c

```
#include "../TaEdit/PluginFuncID.h"
#include <wchar.h>

/* メニューが選択されたときに、この関数が実行されます。 */
void ShowMessage( PFT_GetAPIFunction GetAPIFunction )
{
            /* メッセージボックスを表示するための、API 関数を取得します */
    PFT_ShowInformationMsgBox ShowInformationMsgBox =
        PFT_ShowInformationMsgBox)GetAPIFunction( PFID_SHOWINFORMATIONMSGBOX );

    /* メッセージを表示します */
    ShowInformationMsgBox( L"Welcome to Syuhitu plugin hell world" );
}
```

４．２　コンパイル

上記「msg.c」ファイルを、下記のコマンドでコンパイルします。

```
cc -xarch=v9 -G -o msg.so msg.c
```

なお、主筆は 64 ビットでコンパイルされているため、対象アーキテクチャで v9 を指定して 64 ビットでコンパイルされるようにして下さい。そうでないと、実行時にライブラリのロードに失敗します。

４．３　組み込み

コンパイルして生成された「msg.so」ファイルを、環境変数 LD_LIBRARY_PATH が設定されているフォルダに移動して下さい。なお、主筆の標準のインストールでは、実行時にインストール先ディレクトリ内の「plugin」ディレクトリを、LD_LIBRARY_PATH に追加します。

下記のようにプラグイン設定ファイルを記述して下さい。

```
[00000]
PluginName = MyPlugin
LibraryName = msg.so
FunctionName = ShowMessage
```

セクション名と PluginName は任意です。

LibraryName は、コンパイルして生成された「msg.so」が参照できるように記述して下さい。フルパスで記述しても問題ありません。

FunctionName は、プラグインを実行するときに呼び出す関数の名前を指定して下さい。ここでは上記ソースコードより「ShowMessage」となります。

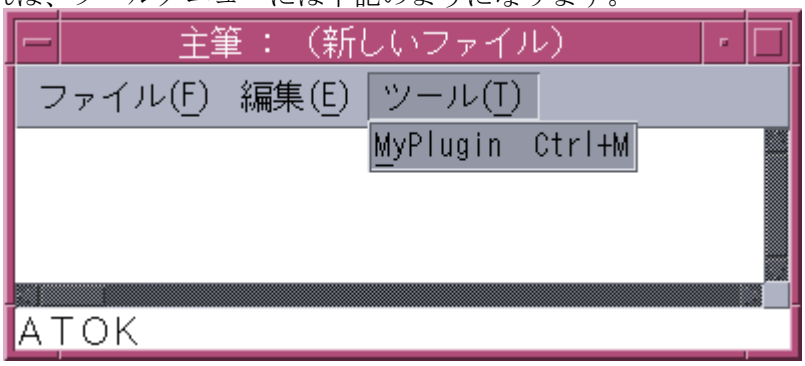
下記の文をリソースファイルに追加して下さい。

```
TaEdit*TEPI_MyPlugin.labelString : MyPlugin
TaEdit*TEPI_MyPlugin.mnemonic: M
TaEdit*TEPI_MyPlugin.accelerator: Ctrl<Key>M
TaEdit*TEPI_MyPlugin.acceleratorText: Ctrl+M
```

４．４　実行

以上を設定を終えたら主筆を起動して下さい。

問題が発生しなければ、ツールメニューには下記のようになります。



そして、「MyPlugin」メニューを選択すると、下記のように情報メッセージボックスが表示されます。