GtkIMContext

GtkIMContext —入力メソッドコンテキストの基本クラス

関数

ボイド	gtk_im_context_set_client_window ()
ボイド	gtk_im_context_get_preedit_string ()
グブール	gtk_im_context_filter_keypress ()
ボイド	gtk_im_context_focus_in ()
ボイド	gtk_im_context_focus_out ()
ボイド	gtk_im_context_reset ()
ボイド	gtk_im_context_set_cursor_location ()
ボイド	gtk_im_context_set_use_preedit ()
ボイド	gtk_im_context_set_surrounding ()
グブール	gtk_im_context_get_surrounding ()
グブール	gtk_im_context_delete_surrounding ()

物性

GtkInputHints	入力ヒント	読み書き
GtkInputPurpose	入力目的	読み書き

信号

ボイド	コミット	最後に実行
グブール	削除周囲	最後に実行
ボイド	プリエディット変更	最後に実行
ボイド	プリエディットエンド	最後に実行
ボイド	プリエディットスタート	最後に実行
グブール	取り囲む	最後に実行

タイプと値

構造	GtkIMContext
構造	GtkIMContextClass
構造	GtkIMContextInfo

オブジェクト階層

```
GObject
    └──GtkIMContext
        ├──GtkIMContextSimple
        └──GtkIMMulticontext
```

含む

```
 #include <gtk / gtk.h>
#include <gtk / gtkimmodule.h>
```

説明

GtkIMContextは、GTK +入力メソッドのインターフェースを定義します。 入力メソッドは、 GtkEntryのようなGTK +テキスト入力ウィジェットによって使用され、キーイベントからUnicode文字列にマッピングされます。

デフォルトの入力方法は、 「gtk-im-module」 GtkSettingsプロパティを介してプログラムで設定できます。 または、 GTK +アプリケーションの実行で説明されているように、GTK_IM_MODULE環境変数を設定することもできます 。

GtkEntryの 「im-module」およびGtkTextViewの 「im-module」プロパティを使用して、特定のウィジェットインスタンスの入力メソッドを設定することもできます。 たとえば、特定の入力ウィジェットには、特定の入力メソッドで入力する方が簡単な特定の文字が含まれていると予想される場合があります。

インプットメソッドは、複数のキーイベントを順番に消費し、最終的に合成結果を出力する場合があります。 これは事前編集と呼ばれ、入力メソッドは中間編集状態を事前編集テキストとして表示することにより、このプロセスに関するフィードバックを提供します。 たとえば、デフォルトのGTK +入力メソッドは、CtrlキーとShiftキーを押しながら「U」に続けて16進数のコードポイントを入力することにより、任意のUnicodeコードポイントの入力を実装します。 CtrlキーとShiftキーを離すと、事前編集が終了し、文字がテキストとして挿入されます。 たとえば、Ctrl + Shift + u20ACは€記号になります。

GTK +ウィジェットでロード可能なモジュールとして使用するために、追加の入力メソッドを使用可能にすることができます。 入力メソッドモジュールは、 GtkIMContextまたはGtkIMContextSimpleのサブクラスを実装し、これらの4つの関数をエクスポートする小さな共有ライブラリです。

```
void im_module_init （ GTypeModule * module ） ;
```

この関数は、 g_type_module_register_type()をg_type_module_register_type()してインプットメソッドを実装するGtkIMContextサブクラスのGTypeを登録する必要があります。 型を動的に登録する必要があるため、 g_type_register_static()は使用できないことに注意してください。

```
void im_module_exit （ void ） ;
```

ここでは、モジュールのアンロード時に入力メソッドが必要とするクリーンアップコードを示します。

```
void im_module_list （ const GtkIMContextInfo *** contexts 、 int * n_contexts ）
{
  * contexts = info_list ;
  * n_contexts = G_N_ELEMENTS （ info_list ） ;
}
```

この関数は、モジュールが提供する入力メソッドのリストを返します。 上記の実装例は、一般的なソリューションを示しており、提供された各入力メソッドに対してGtkIMContextInfoアイテムの静的に定義された配列へのポインターを単に返します。

```
GtkIMContext * im_module_create （ const gchar * context_id ） ;
```

この関数は、 context_id識別されるGtkIMContextサブクラスの新しく作成されたインスタンスへのポインターを返す必要がありcontext_id 。 コンテキストIDは、 im_module_list()によって返されるim_module_list()配列で指定されているものと同じです。

新しいロード可能なインプットメソッドモジュールをシステムにインストールした後、GTK +アプリケーションで新しいインプットメソッドを利用できるようにするには、構成ファイルgtk.immodulesをgtk-query-immodules-3.0で再生成する必要があります。

関数

gtk_im_context_set_client_window （）

```
ボイド
gtk_im_context_set_client_window （ GtkIMContext *context 、
                                   GdkWindow *window ） ;
```

入力コンテキストのクライアントウィンドウを設定します。 これは、入力が表示されるGdkWindowです。 このウィンドウは、ステータスウィンドウを正しく配置するために使用されます。また、入力メソッドの内部目的で使用される場合もあります。

パラメーター

コンテキスト	GtkIMContext	
窓	クライアントウィンドウ。 これは、以前のクライアントウィンドウがもう存在しないことを示すためにNULLになる場合がありNULL 。	[許可なし]

gtk_im_context_get_preedit_string （）

```
ボイド
gtk_im_context_get_preedit_string （ GtkIMContext *context 、
```

```
                                gchar **str ,
                                PangoAttrList **attrs ,
                                gint *cursor_pos ) ;
```

入力コンテキストの現在の事前編集文字列、および文字列に適用する属性のリストを取得します。 この文字列は、挿入ポイントに挿入されて表示されるはずです。

パラメーター

コンテキスト	GtkIMContext	
str	取得した文字列を保存する場所。 取得した文字列はg_free()解放する必要があります。	[out] [完全転送]
attrs	取得した属性リストを保存する場所。 このリストでの作業が完了したら、pango_attr_list_unref()でリストを参照pango_attr_list_unref()する必要があります。	[out] [完全転送]
cursor_pos	プリエディット文字列内のカーソルの位置（文字）を格納する場所。	[アウト]

gtk_im_context_filter_keypress ()

```
 グブール
gtk_im_context_filter_keypress ( GtkIMContext *context ,
                                 GdkEventKey *event ) ;
```

入力メソッドがキーを押して離すイベントを内部的に処理できるようにします。 この関数がTRUE返した場合、このキーイベントに対してそれ以上の処理は行われません。

パラメーター

コンテキスト	GtkIMContext
出来事	キーイベント

返品

入力メソッドがキーイベントを処理した場合はTRUE 。

gtk_im_context_focus_in ()

```
 ボイド
gtk_im_context_focus_in ( GtkIMContext *context ) ;
```

この入力コンテキストが対応するウィジェットがフォーカスを獲得したことをインプットメソッドに通知します。 入力方法は、たとえば、表示されたフィードバックを変更して、この変更を反映させることができます。

パラメーター

コンテキスト	GtkIMContext

gtk_im_context_focus_out ()

```
 ボイド
gtk_im_context_focus_out ( GtkIMContext *context ) ;
```

この入力コンテキストが対応するウィジェットがフォーカスを失ったことをインプットメソッドに通知します。 入力メソッドは、たとえば、表示されたフィードバックを変更したり、コンテキストの状態をリセットしてこの変更を反映したりできます。

パラメーター

コンテキスト	GtkIMContext

gtk_im_context_reset ()

```
 ボイド
gtk_im_context_reset ( GtkIMContext *context ) ;
```

カーソル位置の変更などの変更が行われたことをインプットメソッドに通知します。 これにより、通常、入力メソッドは事前編集状態をクリアします。

パラメーター

コンテキスト	GtkIMContext

gtk_im_context_set_cursor_location ()

```
ボイド
gtk_im_context_set_cursor_location ( GtkIMContext *context 、
                                     const GdkRectangle *area ) ;
```

カーソル位置の変更が行われたことをインプットメソッドに通知します。 場所は、クライアントウィンドウを基準にしています。

パラメーター

コンテキスト	GtkIMContext
エリア	新しい場所

gtk_im_context_set_use_preedit ()

```
ボイド
gtk_im_context_set_use_preedit ( GtkIMContext *context 、
                                 gboolean use_preedit ) ;
```

IMコンテキストが事前編集文字列を使用してフィードバックを表示するかどうかを設定します。 use_preeditがFALSE（デフォルトはTRUE）の場合、IMコンテキストは、ルートウィンドウの子に表示するなど、フィードバックを表示するために他の方法を使用する場合があります。

パラメーター

コンテキスト	GtkIMContext
use_preedit	IMコンテキストが事前編集文字列を使用するかどうか。

gtk_im_context_set_surrounding ()

```
ボイド
gtk_im_context_set_surrounding ( GtkIMContext *context 、
                                 const gchar *text 、
                                 gint len 、
                                 gint cursor_index ) ;
```

挿入ポイントとプリエディット文字列の周囲のコンテキストを設定します。 この関数は、GtkIMContext :: retrieve_surroundingシグナルに応じて呼び出されることが期待されており、他のときに呼び出されても効果はないでしょう。

パラメーター

コンテキスト	GtkIMContext
テキスト	挿入ポイントを囲むテキスト（UTF-8）。 プリエディット文字列はtext内に含めるべきではありません。
len	textの長さ、またはtextがヌルtextで終了する場合は-1
cursor_index	text内の挿入カーソルのバイトインデックス

gtk_im_context_get_surrounding ()

```
グブール
gtk_im_context_get_surrounding ( GtkIMContext *context 、
                                 gchar **text 、
                                 gint *cursor_index ) ;
```

挿入ポイント周辺のコンテキストを取得します。 通常、入力メソッドは、既存のテキストに基づいて入力テキストを制限するためにコンテキストを必要とします。 これは、文字の一部のシーケンスのみが許可されるタイ語などの言語にとって重要です。

この関数は、入力メソッドでGtkIMContext :: retrieve_surrounding信号を発行することで実装されます。 このシグナルに応じて、ウィジェットはgtk_im_context_set_surrounding()呼び出して、利用可能な限り多くのコンテキストを段落全体まで提供する必要があります。 ウィジェットが:: retrieve_surrounding信号に応答する義務はないため、入力メソッドはコンテキストなしで機能するように準備する必要があることに注意してください。

パラメーター

コンテキスト	GtkIMContext	
テキスト	挿入ポイントの周囲のコンテキストを保持するテキストのUTF-8エンコードされた文字列を保存する場所。 関数がTRUE返す場合、この場所に格納されている結果をg_free()で解放する必要があります。	[out] [完全転送]
cursor_index	text内に挿入カーソルのバイトインデックスを格納する場所。	[アウト]

返品

周囲のテキストが提供された場合はTRUE 。 この場合、* textに保存されている結果を解放する必要があります。

---

gtk_im_context_delete_surrounding（）

```
 グブール
gtk_im_context_delete_surrounding （ GtkIMContext *context 、
                                    gint offset 、
                                    gint n_chars ） ;
```

入力コンテキストがアタッチされているウィジェットに、GtkIMContext :: delete_surroundingシグナルを発行してカーソル位置の周囲の文字を削除するように要求します。 offsetとn_charsはバイト単位ではなく文字単位であり、 GtkIMContextの他の場所とは異なることに注意してください。

この関数を使用するには、最初にgtk_im_context_get_surrounding()を呼び出して現在のコンテキストを取得し、その後すぐにこの関数を呼び出して、削除する内容を確認する必要があります。 また、シグナルが処理されたとしても、入力コンテキストは削除が要求されたすべての文字を削除しなかった可能性があるという事実を考慮する必要があります。

この関数は、新しい入力に応じて既存のテキストを置換したいインプットメソッドによって使用されます。 アプリケーションには役立ちません。

パラメーター

コンテキスト	GtkIMContext
オフセット	文字でのカーソル位置からのオフセット。 負の値は、カーソルの前から開始することを意味します。
n_chars	削除する文字数。

返品

シグナルが処理された場合はTRUE 。

タイプと値

struct GtkIMContext

```
struct GtkIMContext;
```

---

struct GtkIMContextClass

```
struct GtkIMContextClass {
 / *信号* /
 void （* preedit_start） （GtkIMContext * context） ;
 void （* preedit_end） （GtkIMContext * context） ;
 void （* preedit_changed） （GtkIMContext * context） ;
 void （* commit） （GtkIMContext * context、const gchar * str） ;
 gboolean （* retrieve_surrounding） （GtkIMContext * context） ;
 gboolean （* delete_surrounding） （GtkIMContext * context、
                                    ジントオフセット、
                                     gint n_chars） ;

 / *仮想関数* /
```

```
  void (* set_client_window)  (GtkIMContext * context、
                               GdkWindow * window) ;
  void (* get_preedit_string)  (GtkIMContext * context、
                               gchar ** str、
                               PangoAttrList ** attrs、
                               gint * cursor_pos) ;
  gboolean (* filter_keypress)  (GtkIMContext * context、
                               GdkEventKey * event) ;
  void (* focus_in)  (GtkIMContext * context) ;
  void (* focus_out)  (GtkIMContext * context) ;
  void (* reset)  (GtkIMContext * context) ;
  void (* set_cursor_location)  (GtkIMContext * context、
                               GdkRectangle * area) ;
  void (* set_use_preedit)  (GtkIMContext * context、
                               gboolean use_preedit) ;
  void (* set_surrounding)  (GtkIMContext * context、
                               const gchar * text、
                              ギン・レン、
                               gint cursor_index) ;
  gboolean (* get_surrounding)  (GtkIMContext * context、
                               gchar ** text、
                               gint * cursor_index) ;
};
```

会員

preedit_start preedit_start ()	「preedit-start」シグナルのデフォルトハンドラ。
preedit_end preedit_end ()	「preedit-end」シグナルのデフォルトハンドラ。
preedit_changed preedit_changed ()	「preedit-changed」シグナルのデフォルトハンドラー。
commit commit ()	「コミット」シグナルのデフォルトハンドラー。
retrieve_surrounding retrieve_surrounding ()	「取り囲む」信号のデフォルトハンドラ。
delete_surrounding delete_surrounding ()	「delete-surrounding」シグナルのデフォルトハンドラー。
set_client_window set_client_window ()	入力されたテキストが表示される入力ウィンドウが変更されたときにgtk_im_context_set_client_window()を介して呼び出されます。 これをオーバーライドして、たとえば入力メソッドのステータス表示を配置する目的で、現在の入力ウィンドウを追跡します。
get_preedit_string get_preedit_string ()	gtk_im_context_get_preedit_string()を介して呼び出され、現在カーソル位置に表示するために事前編集されているテキストを取得します。 複雑な文字を構成する入力メソッドや、複数の連続したキーを押すことで作成されるその他の構成は、このメソッドをオーバーライドしてフィードバックを提供する必要があります。
filter_keypress filter_keypress ()	すべてのキープレスまたはリリースイベントでgtk_im_context_filter_keypress()を介して呼び出されます。 重要なイベントからテキストへのマッピングを実装するために、すべての重要な入力メソッドがこれをオーバーライドする必要があります。 TRUEの戻り値は、入力メソッドによってイベントが消費されたことを呼び出し元に示します。 その場合、結果のテキストを入力ウィジェットに渡すために、キーシーケンスの完了時に「コミット」信号を発行する必要があります。 または、イベントがインプットメソッドによって処理されなかったことを示すためにFALSEが返されるFALSEがありFALSE 。 キーの組み込みマッピングが存在する場合、文字の生成に使用されます。
focus_in focus_in ()	入力ウィジェットがフォーカスを取得したときにgtk_im_context_focus_in()を介して呼び出されます。 現

	在のフォーカスを追跡するためにオーバーライドできます。
focus_out focus_out ()	入力ウィジェットがフォーカスを失ったときにgtk_im_context_focus_out()を介して呼び出されます。現在のフォーカスを追跡するためにオーバーライドできます。
reset reset ()	gtk_im_context_reset()を介して呼び出され、カーソル位置の変更などの変更を通知します。 事前編集を実装する入力メソッドは、リセット時に事前編集状態をクリアするためにこのメソッドをオーバーライドする必要があります。
set_cursor_location set_cursor_location ()	gtk_im_context_set_cursor_location()を介して呼び出され、クライアントウィンドウに相対的な現在のカーソル位置をインプットメソッドに通知します。 カーソル位置にポップアップウィンドウの表示を実装するためにオーバーライドできます。
set_use_preedit set_use_preedit ()	gtk_im_context_set_use_preedit()文字列の使用を制御するために、 gtk_im_context_set_use_preedit()を介して呼び出されます。 これをオーバーライドして、オフになっている場合は他の手段でフィードバックを表示します。
set_surrounding set_surrounding ()	カーソル周辺のコンテキストに関する入力メソッドのアイデアを更新するために、 「検索を囲む」というシグナルに応じてgtk_im_context_set_surrounding()経由で呼び出されます コンテキスト依存の動作を実装するインプットメソッドであっても、このメソッドをオーバーライドする必要はありません。 基本実装はgtk_im_context_get_surrounding()が機能するのに十分です。
get_surrounding get_surrounding ()	gtk_im_context_get_surrounding()を介して呼び出され、カーソル位置周辺のコンテキストを更新します。 コンテキスト依存の動作を実装するインプットメソッドであっても、このメソッドをオーバーライドする必要はありません。 基本実装は「取得-サラウンド」をget_surroundingし、その後のget_surrounding呼び出しで受信したコンテキストを記録します。

struct GtkIMContextInfo

```
struct GtkIMContextInfo {
  const gchar * context_id;
  const gchar * context_name;
  const gchar * domain;
  const gchar * domain_dirname;
  const gchar * default_locales;
};
```

ロード可能な入力メソッドに関する簿記情報。

会員

const gchar * context_id context_id ;	入力メソッドの一意の識別文字列。
const gchar * context_name context_name ;	入力メソッドの人間が読める名前。
const gchar * domain domain	dgettext()使用される翻訳ドメイン
const gchar * domain_dirname domain_dirname ;	bindtextdomain()で使用するロケールディレクトリの名前
const gchar * default_locales default_locales ;	このインプットメソッドがデフォルトであるロケールのコロン区切りのリスト。 アスタリスク「*」は、すべてのロケールのデフォルトを設定します。

物件詳細

“input-hints” プロパティ

「入力ヒント」 GtkInputHints

テキストフィールドの動作のヒント。

フラグ：読み取り/書き込み

“input-purpose” プロパティ

「入力目的」 GtkInputPurpose

テキストフィールドの目的。

フラグ：読み取り/書き込み

デフォルト値：GTK_INPUT_PURPOSE_FREE_FORM

信号の詳細

“commit” シグナル

```
 ボイド
user_function ( GtkIMContext * context、
                gchar * str、
                gpointer user_data)
```

ユーザーが完全な入力シーケンスを入力すると、:: commitシグナルが発行されます。 これは、キーを押した直後の1文字、または事前編集の最終結果です。

パラメーター

コンテキスト	信号が発信されるオブジェクト
str	ユーザーが入力した完成した文字
user_data	シグナルハンドラが接続されたときに設定されたユーザーデータ。

フラグ：最後に実行

“delete-surrounding” シグナル

```
 グブール
user_function ( GtkIMContext * context、
                ジントオフセット、
                gint n_chars、
                gpointer user_data)
```

入力メソッドがカーソルを囲むコンテキストのすべてまたは一部を削除する必要がある場合、:: delete-surroundingシグナルが発行されます。

パラメーター

コンテキスト	信号が発信されるオブジェクト
オフセット	削除するテキストのカーソル位置からの文字オフセット。 負の値は、カーソルの前の位置を示します。
n_chars	削除する文字数
user_data	シグナルハンドラが接続されたときに設定されたユーザーデータ。

返品

シグナルが処理された場合はTRUE 。

フラグ：最後に実行

“preedit-changed” 信号

```
 ボイド
```

```
user_function ( GtkIMContext * context、
                gpointer user_data)
```

:: preedit-changedシグナルは、現在入力されているプリエディットシーケンスが変更されるたびに発行されます。 また、gtk_im_context_get_preedit_string()シーケンスの最後に出力されます。この場合、 gtk_im_context_get_preedit_string()は空の文字列を返します。

パラメーター

コンテキスト	信号が発信されるオブジェクト
user_data	シグナルハンドラが接続されたときに設定されたユーザーデータ。

フラグ：最後に実行

“preedit-end” プリ “preedit-end” 信号

```
 ボイド
user_function ( GtkIMContext * context、
                gpointer user_data)
```

:: preedit-endシグナルは、事前編集シーケンスが完了またはキャンセルされると発生します。

パラメーター

コンテキスト	信号が発信されるオブジェクト
user_data	シグナルハンドラが接続されたときに設定されたユーザーデータ。

フラグ：最後に実行

“preedit-start” シグナル

```
 ボイド
user_function ( GtkIMContext * context、
                gpointer user_data)
```

:: preedit-startシグナルは、新しい事前編集シーケンスが開始されると発行されます。

パラメーター

コンテキスト	信号が発信されるオブジェクト
user_data	シグナルハンドラが接続されたときに設定されたユーザーデータ。

フラグ：最後に実行

“retrieve-surrounding” 信号

```
 グブール
user_function ( GtkIMContext * context、
                gpointer user_data)
```

入力メソッドがカーソルを囲むコンテキストを必要とする場合、:: retrieve-surroundingシグナルが発行されます。 コールバックは、gtk_im_context_set_surrounding()メソッドを呼び出して、コンテキストを囲む入力メソッドを設定する必要があります。

パラメーター

コンテキスト	信号が発信されるオブジェクト
user_data	シグナルハンドラが接続されたときに設定されたユーザーデータ。

返品

シグナルが処理された場合はTRUE 。

フラグ：最後に実行

GTK-Doc V1.30.1によって生成