

# Mise en forme du texte dans un GtkTextView

par [Franck Hecht](#)

Date de publication : 06/05/2007

Dernière mise à jour :

Avec GTK+ vous avez la possibilité d'utiliser un GtkTextView pour un affichage de texte multiligne. Le texte reste par contre mis en forme de la façon la plus basique possible soit la police par défaut de votre système avec la couleur de base, noir dans la plupart des cas. Dans ce tutoriel vous allez voir comment on peut mettre en forme du texte contenu dans un GtkTextView !

- I - Introduction
- II - Petit tour sur les GtkTextTag
- III - Première approche
  - III-A - Procédure de création du GtkTextView
  - III-B - Les tags
    - III-B-1 - Texte italique
    - III-B-2 - Texte gras
    - III-B-3 - Texte souligné
    - III-B-4 - Texte barré
    - III-B-5 - Justification du texte
    - III-B-6 - Taille du texte
    - III-B-7 - Couleur du texte
  - III-C - Code source d'exemple
- IV - Seconde approche
  - IV-A - Procédure de création du GtkTextView
  - IV-B - Création des GtkTextTag
  - IV-C - Code source d'exemple
- V - Note sur la création des tags
- VI - Petit tour sur le GtkTextBuffer
  - VI-A - Comment récupérer la sélection du texte ?
  - VI-B - Comment appliquer des tags au texte sélectionné ?
  - VI-C - Comment insérer du texte avec des tags ?
- VII - Récupérer les attributs d'un texte sélectionné
- VIII - Remerciements

## I - Introduction

Mettre en forme du texte dans un GtkTextView résulte d'un certain tour de force car cela demande pas mal d'ajouts, paramétrage et donc de programmation supplémentaire. Il est vrai que dans GTK+ il n'existe pas de fonctions style `gtk_text_bold` ou autres, il vous faut donc tout faire de vous-même... ou presque.

Pour être plus juste, le texte est modifié depuis le GtkTextBuffer de votre GtkTextView et pour pouvoir faire de telles mises en forme, il vous faut donc utiliser les GtkTextTag et/ou des GtkTextTagTable si vous initialisez vous-même le GtkTextBuffer avant ou après la création de la zone de texte !

Nous allons donc voir dans ce tutoriel deux manières de créer et mettre en place les tags pour le texte. La première approche se fera directement par les fonctions simplifiées du widget GtkTextBuffer. La seconde approche va donc nous permettre de voir, une fois les notions de bases acquises, comment mettre ceci en place en créant les GtkTextTag et en les mettant dans notre GtkTextTagTable que nous insérerons dans notre GtkTextBuffer.

Ceci peut paraître un peu lourd au premier abord mais vous verrez que cela n'est pas insurmontable !



## II - Petit tour sur les GtkTextTag

Les [GtkTextTag](#) sont les éléments essentiels pour la mise en forme de texte car ce sont eux qui nous permettent de définir les différents *tags* que l'on souhaite appliquer au texte.

Le Widget [GtkTextTag](#) étant l'élément central de la mise en forme de texte, c'est par conséquent ce même widget qui détient les propriétés applicables au texte contenu dans un [GtkTextBuffer](#) ! Hé oui, comme on pouvait s'en douter (*non ?*) tout tourne également autour du widget [GtkTextBuffer](#) ce qui est normal car c'est ce widget qui stocke le texte, [GtkTextView](#) sert entre autre à l'affichage de son contenu donc nous l'utiliserons que pour cela !

Je vous invite à regarder la liste des propriétés d'un [GtkTextTag](#): [GtkTextTag - Properties](#)

Cette liste est très pratique car non seulement elle donne le nom des différentes propriétés dont il est indispensable de respecter l'identifiant mais elle permet également d'avoir les liens vers les différentes listes de constantes pour certaines propriétés. Il est donc bon de toujours garder cette liste à portée de clique !

Maintenant que nous savons que la mise en forme passe par un [GtkTextTag](#) et que nous en savons un peu plus à son sujet, passons à la partie pratique...

### III - Première approche

Cette approche est la manière simplifiée de créer et appliquer de tags à du texte. Ici nous passerons par un widget GtkTextView de base et sans même créer notre propre buffer de texte (*contrairement à la seconde approche*) ! Nous n'allons pas écrire un programme complet dans ce tutoriel (*des programmes d'exemples sont disponibles en téléchargement*) car les programmes sont quand même relativement longs.

#### III-A - Procédure de création du GtkTextView

Ici nous créons un GtkTextView tout simplement avec la fonction [gtk\\_text\\_view\\_new](#) comme vous avez sans doute l'habitude de faire, nous récupérons ensuite son GtkTextBuffer et nous le stockons:

```
GtkWidget * p_text = gtk_text_view_new ();
GtkTextBuffer * p_buf = gtk_text_view_get_buffer (GTK_TEXT_VIEW (p_text_buffer));
```

Nous partons donc d'un widget créé de cette manière ou du moins de son GtkTextBuffer.

#### III-B - Les tags

Nous voici donc dans le coeur du sujet... Ici nous allons voir comment créer des tags de toutes sortes avec la fonction simplifiée du widget GtkTextBuffer à savoir: [gtk\\_text\\_buffer\\_create\\_tag](#).

Cette fonction va créer pour nous le GtkTextTag correspondant à nos souhaits et l'insérer dans la liste des tags GtkTextTagTable de notre GtkTextBuffer sans autre action de notre part !

Il faut également savoir que des propriétés, il en existe de différents types, nous pouvons créer par exemple des tags attendant une valeur entière comme une constante ou parfois même une valeur booléenne ou des structures comme les GdkColor etc... Il convient donc de vous référer à la liste passée en lien dans le *chapitre 2* pour savoir quel type de valeur attend la propriété que vous désirez éditer dans un GtkTextTag !

Voyons donc comment nous allons créer différents tags à savoir:

- Texte italique
- Texte gras
- Texte souligné
- Texte barré
- Texte centré
- Changement de taille
- Changement de couleur

Pourquoi voir plusieurs propriétés ? Tout simplement car chacune de ces propriétés est vraiment différente et je vous montre également les plus connues et donc les plus utilisées dans des éditeurs de texte par exemple, cela vous permettra de voir un peu le fonctionnement et le raisonnement pour leur mise en place. Commençons par voir les arguments que prend la fonction [gtk\\_text\\_buffer\\_create\\_tag](#) dont voici son prototype:

```
GtkTextTag*          gtk_text_buffer_create_tag          (GtkTextBuffer *buffer,
                                                         const gchar *tag_name,
                                                         const gchar *first_property_name,
```

```
...);
```

Le premier argument n'est pas difficile, c'est le pointeur sur le GtkTextBuffer de votre GtkTextView comme celui que nous avons récupéré dans la création d'un GtkTextView dans la section précédente.

Le second argument permet de donner un nom au tag que nous créons, ceci peut être n'importe quel identifiant mais il est préférable d'en choisir un assez évocateur car c'est vous qui devrez le retenir ! Cet argument peut également être **NULL** ce qui donnera naissance à un tag anonyme. Inutile de vous dire que si vous ne gardez pas un pointeur sur un GtkTextTag anonyme, vous n'arriverez sans doute plus à y accéder. Il faut également prendre le soin de choisir un nom qui n'est pas déjà défini, aucun doublon ne peut exister.

Le troisième argument est ici le début d'une liste qui va par couple "*propriété/valeur*", cette liste doit se terminer par un **NULL**. Le nom de la propriété correspond donc au nom d'une propriété connue du widget GtkTextTag et la valeur est le style que vous désirez appliquer au texte par le biais de cette propriété.

La fonction retourne un pointeur sur le nouveau GtkTextTag créé et initialisé aux bonnes valeurs d'après vos paramètres. Le choix vous appartient de récupérer ce pointeur ou non, dans nos exemples, nous n'allons pas nous en soucier.

### III-B-1 - Texte italique

Le texte italique fait partie de la propriété *style* qui attend une constante du type *PangoStyle* dont voici la liste:

- PANGO\_STYLE\_NORMAL
- PANGO\_STYLE\_OBLIQUE
- PANGO\_STYLE\_ITALIC

Exemple:

```
gtk_text_buffer_create_tag (
    p_buf, "italic",
    "style", PANGO_STYLE_ITALIC,
    NULL
);
```

Vous pouvez juger que cela n'est pas très difficile de créer un tag mais certains demandent plus d'attention et de manipulation comme un tag de changement de taille par exemple. Ici nous avons donc créé un tag nommé "*italic*" (c'est par ce nom que nous y ferons donc référence)

### III-B-2 - Texte gras

Le texte en gras s'appuie sur le propriété *weight* (*poids*), le terme le plus approprié aurait été *thickness* (*épaisseur*) mais c'est plus long... La valeur attendue est de type **gint** comprise entre **100 et 900** ou une constante de type *PangoWeight* dont voici la liste:

- PANGO\_WEIGHT\_ULTRALIGHT
- PANGO\_WEIGHT\_LIGHT
- PANGO\_WEIGHT\_NORMAL

- PANGO\_WEIGHT\_SEMIBOLD
- PANGO\_WEIGHT\_BOLD
- PANGO\_WEIGHT\_ULTRABOLD
- PANGO\_WEIGHT\_HEAVY

Exemple:

```
gtk_text_buffer_create_tag (
    p_buf, "bold",
    "weight", PANGO_WEIGHT_BOLD,
    NULL
);
```

### III-B-3 - Texte souligné

Cette propriété est basée en fait sur son vrai terme anglais soit underline tout simplement. Elle attend une des constantes de type PangoUnderline dont voici la liste:

- PANGO\_UNDERLINE\_NONE
- PANGO\_UNDERLINE\_SINGLE
- PANGO\_UNDERLINE\_DOUBLE
- PANGO\_UNDERLINE\_LOW
- PANGO\_UNDERLINE\_ERROR

Exemple:

```
gtk_text_buffer_create_tag (
    p_buf, "underline",
    "underline", PANGO_UNDERLINE_SINGLE,
    NULL
);
```

Dans cet exemple on peut remarquer une chose assez importante et utile à savoir. En effet, nous avons nommé notre tag du même nom que sa propriété, sachez donc que c'est possible ! Pour information, la constante PANGO\_UNDERLINE\_ERROR fait un soulignement en vague comme il est utilisé dans les éditeurs de texte lorsque qu'une faute à été commise ou que le mot tapé est inconnu du dictionnaire du programme !

### III-B-4 - Texte barré

Cette propriété, tout comme la précédente, se nomme par rapport à son but mais en terme anglais donc strikethrough mais elle change par rapport aux autres car sa valeur est du type gboolean donc si vous mettez **TRUE**, votre texte sera barré et **FALSE**, non barré !

Exemple:

```
gtk_text_buffer_create_tag (
    p_buf, "strikethrough",
    "strikethrough", TRUE,
    NULL
);
```

### III-B-5 - Justification du texte

La justification du texte se fait par le biais de la propriété du même nom soit *justification* et attend une constante de type *GtkJustification* dont voici la liste:

- GTK\_JUSTIFY\_LEFT
- GTK\_JUSTIFY\_RIGHT
- GTK\_JUSTIFY\_CENTER
- GTK\_JUSTIFY\_FILL

Exemple pour un texte centré:

```
gtk_text_buffer_create_tag (
    p_buf, "center",
    "justification", GTK_JUSTIFY_CENTER,
    NULL
);
```

### III-B-6 - Taille du texte

Ceci est une propriété un peu plus spéciale car celle-ci exige l'utilisation d'un vecteur de transformation. On utilise généralement le vecteur *PANGO\_SCALE*. Cette propriété est nommée *size* et permet donc de changer la taille du texte.

Exemple d'une police à 20 points:

```
gtk_text_buffer_create_tag (
    p_buf, "size20",
    "size", 20 * PANGO_SCALE,
    NULL
);
```

### III-B-7 - Couleur du texte

Pour ce tag il existe deux moyens de le créer, on peut soit utiliser la propriété *foreground* ou *foreground-gdk*. La différence entre les deux est que la première permet de mettre en place une couleur prédéfinie dans GTK+ par son simple nom (*en anglais*) et la seconde permet de passer la couleur par le biais d'une structure de type *GdkColor*

Exemple:

```
gtk_text_buffer_create_tag (
    p_buf, "font-blue",
    "foreground", "blue",
    NULL
);
```

Voici une petite liste des noms de couleurs qui peuvent être utilisés: *white, black, gray, darkgray, red, darkred, green, darkgreen, blue, darkblue, yellow, brown, orange, purple, magenta*

Exemple avec la propriété *foreground-gdk*:

```
GdkColor color;
```

```
/*
 * Initialisation du GdkColor avec par exemple un
 * widget GtkColorSelection...
 */
...

gtk_text_buffer_create_tag (
    p_buf, "fontcolor",
    "foreground-gdk", & color,
    NULL
);
```

Comme précisé en commentaire dans cet exemple, il faut au préalable que la structure de type *GdkColor* soit initialisée correctement par exemple par le biais d'un *GtkColorSelection* ou même un *GtkColorSelectionDialog* sous peine d'avoir droit dans la plupart des cas à un crash du programme !

### III-C - Code source d'exemple

Vous pouvez [télécharger](#) le code source complet d'un programme mettant en application ce que nous venons de voir dans ce chapitre.

## IV - Seconde approche

Ceci est l'approche longue mais pas forcément la plus dure. Nous allons voir comment elle s'implémente surtout pour voir comment fonctionne la fonction `gtk_text_buffer_create_tag`. Ce n'est pas vraiment obligatoire mais nous allons également créer notre `GtkTextBuffer` ainsi qu'un `GtkTextTagTable`. La création des `GtkTextTag` se fait également plus longue car il ne faut pas moins de trois étapes pour fabriquer un tag !

### IV-A - Procédure de création du GtkTextView

La création de ce widget que nous allons voir est plus longue (*mais pas obligatoire cependant*). Nous allons créer un `GtkTextTagTable` que nous mettrons dans notre `GtkTextBuffer` lors de son initialisation que nous mettrons dans notre `GtkTextView` !

Voyons tout cela:

```
/* Creation du GtkTextTagTable: */
p_tag_table = gtk_text_tag_table_new ();

/* Creation du GtkTextBuffer: */
p_buf = gtk_text_buffer_new (p_tag_table);

/* Creation du GtkTextView: */
p_text = gtk_text_view_new_with_buffer (p_buf);
```

Tout ceci n'est pas vraiment obligatoire, on peut très bien utiliser la méthode que nous avons vu au précédent chapitre mais il vous faudra alors récupérer le `GtkTextBuffer` comme il l'a également été montré mais il faut aussi en plus récupérer un pointeur sur le `GtkTextTagTable` créé automatiquement avec la fonction [gtk\\_text\\_buffer\\_get\\_tag\\_table](#) !

### IV-B - Création des GtkTextTag

Entrons dans le vif du sujet. Ici je vais vous expliquer la seconde méthode pour créer des tags pour mettre en forme votre texte dans un `GtkTextView`. Il nous faut commencer par la création d'un `GtkTextTag` tout en lui donnant un nom, par exemple pour un tag qui permet la mise en italique du texte:

```
GtkTextTag * p_tag = gtk_text_tag_new ("italic");
```

C'est très simple effectivement ! C'est le même principe qu'auparavant, il faut donner un nom au tag, c'est ici que ça se fait. Il nous faut maintenant définir les propriétés à appliquer au texte, nous devons utiliser pour cela la fonction [g\\_object\\_set](#). Le fonctionnement est à peu près pareil, dans le premier argument on transmet un pointeur générique sur notre `GtkTextTag` puis ensuite la liste des propriétés pour le texte:

```
g_object_set (
    (gpointer) p_tag,
    "style", PANGO_STYLE_ITALIC,
    NULL
);
```

Comme d'habitude, il faut terminer la liste par **NULL** ! La dernière étape qui reste toute aussi importante, est d'ajouter le `GtkTextTag` dans notre `GtkTextTagTable` ce que nous faisons grâce à la fonction [gtk\\_text\\_tag\\_table\\_add](#):

```
gtk_text_tag_table_add (p_tag_table, p_tag);
```

Notre tag est enfin prêt à être utilisé comme avant !

## IV-C - Code source d'exemple

Vous pouvez [télécharger](#) le code source complet d'un programme mettant en application ce que nous venons de voir dans ce chapitre.

## V - Note sur la création des tags

Une petite précision qui peut avoir son importance. Il est tout à fait possible de créer un tag avec plusieurs propriétés. Ainsi, si vous désirez créer un tag mettant en forme un titre, on peut très bien faire:

```
gtk_text_buffer_create_tag (  
    p_buf, "title",  
    "weight", PANGO_WEIGHT_BOLD,  
    "underline", PANGO_UNDERLINE_SINGLE,  
    "size", 20 * PANGO_SCALE,  
    "justification", GTK_JUSTIFY_CENTER,  
    NULL  
);
```

Ce qui nous permet d'avoir d'un seul coup un tag mettant en forme un texte *centré*, *souligné* en aspect *gras* et de taille *20* ! De cette manière, on peut créer pour un logiciel de traitement de texte, différents types de formatage de texte par défaut !

## VI - Petit tour sur le GtkTextBuffer

Le widget GtkTextBuffer occupe une place importante dans l'utilisation du GtkTextView car c'est par son biais que la plupart des opérations se font comme dans notre cas, l'application de tags mais on peut faire bien plus comme récupérer la sélection du texte, mettre du texte dans le presse-papier, insérer/supprimer du texte, et bien d'autres actions.

Nous allons voir ici les opérations dont nous avons besoin pour ce tutoriel (*récupérer la sélection du texte, appliquer des tags au texte sélectionné*) mais je vous invite à consulter la page de ce widget à l'adresse suivante: <http://developer.gnome.org/doc/API/2.0/gtk/GtkTextBuffer.html>

### VI-A - Comment récupérer la sélection du texte ?

On ne récupère en fait pas le texte sélectionné mais uniquement les repères de début et de fin de la sélection donc des *GtkTextIter*. Pour ce faire, nous utilisons la fonction [gtk\\_text\\_buffer\\_get\\_selection\\_bounds](#). Cette fonction renvoie **TRUE** si du texte est sélectionné dans le GtkTextView, **FALSE** sinon. Si du texte est sélectionné, la fonction initialise alors les *GtkTextIter* dont l'adresse est passée en paramètre.

Voici un exemple d'utilisation:

```
gboolean ret = FALSE;
GtkTextIter start;
GtkTextIter end;

ret = gtk_text_buffer_get_selection_bounds (
    p_buf, & start, & end
);
```

### VI-B - Comment appliquer des tags au texte sélectionné ?

Il y a deux façons d'appliquer des tags à du texte dans un *GtkTextBuffer*. La première consiste à utiliser la fonction [gtk\\_text\\_buffer\\_apply\\_tag](#). On passe ici directement un pointeur sur le *GtkTextTag* à appliquer au texte, il faut donc posséder un pointeur sur les tags que vous utilisez.

Exemple:

```
GtkTextTag * p_tag = NULL;

/* Creation du tag... */
...

gtk_text_buffer_apply_tag (
    p_buf, p_tag,
    & start, & end
);
```

De même pour supprimer un tag sur un texte, on utilise la fonction [gtk\\_text\\_buffer\\_remove\\_tag](#)

La seconde méthode pour appliquer des tags consiste à utiliser la fonction [gtk\\_text\\_buffer\\_apply\\_tag\\_by\\_name](#). A peu de choses identique à la fonction précédente, celle-ci permet d'appliquer un tag en précisant directement son

nom, celui que vous avez choisi lors de sa création.

Exemple:

```
gtk_text_buffer_apply_tag_by_name (
    p_buf, "italic", & start, & end
);
```

La seconde méthode pour supprimer un tag sur un texte est d'utiliser la fonction [gtk\\_text\\_buffer\\_remove\\_tag\\_by\\_name](#)

## VI-C - Comment insérer du texte avec des tags ?

On a parlé jusqu'ici d'application de tag à du texte dans un GtkTextBuffer mais il est également possible d'insérer directement du texte tout en y appliquant des tags, il est même possible d'appliquer plusieurs tags en même temps et dans un unique appel. Ceci se fait avec les fonctions [gtk\\_text\\_buffer\\_insert\\_with\\_tags](#) et [gtk\\_text\\_buffer\\_insert\\_with\\_tags\\_by\\_name](#).

La première fonction permet d'ajouter les tags en passant un pointeur sur chacun des *GtkTextTag* à appliquer au texte et la seconde ne demande que le nom de ceux-ci. La liste doit être terminée par **NULL**.

Exemple:

```
gtk_text_buffer_insert_with_tags_by_name (
    p_buf, & iter,
    "Votre texte à insérer dans le GtkTextBuffer...", -1
    "italic", "bold", "underline", "center",
    NULL
);
```

Le *GtkTextIter* passé en second paramètre correspond à l'emplacement de départ de l'insertion du texte, il convient donc de l'initialiser correctement avant son utilisation. Ceci peut se faire de plusieurs manières soit avec les fonctions suivantes:

- [gtk\\_text\\_buffer\\_get\\_iter\\_at\\_line\\_offset](#)
- [gtk\\_text\\_buffer\\_get\\_iter\\_at\\_offset](#)
- [gtk\\_text\\_buffer\\_get\\_iter\\_at\\_line](#)
- [gtk\\_text\\_buffer\\_get\\_iter\\_at\\_line\\_index](#)
- [gtk\\_text\\_buffer\\_get\\_iter\\_at\\_mark](#)
- [gtk\\_text\\_buffer\\_get\\_iter\\_at\\_child\\_anchor](#)
- [gtk\\_text\\_buffer\\_get\\_start\\_iter](#)
- [gtk\\_text\\_buffer\\_get\\_end\\_iter](#)

## VII - Récupérer les attributs d'un texte sélectionné

La création et l'application de tags à du texte ne seraient pas complètes sans pouvoir récupérer les attributs d'un texte ou d'un caractère dont entre autre, le(s) *GtkTextTag* appliqué(s) ! Différentes fonctions sont à notre service, une première approche qui peut être envisagée serait de récupérer directement les attributs du texte par le biais de la fonction [gtk\\_text\\_iter\\_get\\_attributes](#).

Pour pouvoir l'utiliser, il vous faut récupérer un *GtkTextIter* valide sous peine d'avoir un message d'erreur sur votre console ou voir même pire, un crash du programme ! L'autre solution est de récupérer la liste des *GtkTextTag* affectés au texte à l'endroit précis du *GtkTextIter*. Pour cela vous avez la fonction [gtk\\_text\\_iter\\_get\\_tags](#). Cette fonction retourne une *GSLit* triée par ordre de priorité croissante (*le tag ayant la priorité la plus haute se trouve en fin de liste*).

*Vous devez vous-même supprimer cette liste une fois que vous n'en avez plus besoin !*

Des fonctions de tests vous permettent de déterminer si un tag est utilisé ou non à l'endroit précis indiqué par le *GtkTextIter* passé en premier argument:

- [gtk\\_text\\_iter\\_begins\\_tag](#)
- [gtk\\_text\\_iter\\_ends\\_tag](#)
- [gtk\\_text\\_iter\\_toggles\\_tag](#)
- [gtk\\_text\\_iter\\_has\\_tag](#)

## VIII - Remerciements

Un grand merci à [gege2061](#) pour ses suggestions et la réorganisation des codes sources et à [julp](#) pour la relecture attentive et la correction de ce tutoriel !