Les listes doublement chaînées en C

par Nicolas Joseph

Date de publication : 20 Juillet 2005

Dernière mise à jour :

Les structures de données en C. Seconde partie : les listes doublement chaînées.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

I - Introduction

Après avoir vu les listes simplement chaînées, pour cette seconde partie, nous allons nous attarder sur les listes doublement chaînées.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

II - Principe

A la différence des listes simplement chaînées, les maillons d'une liste doublement chaînée possèdent un pointeur sur l'élément qui les précèdent :

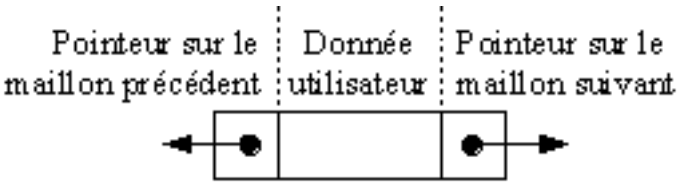

Figure 1 : représentation d'un élément d'une liste doublement chaînée.

Le fait d'avoir accès à l'élément précédent va nous permettre de simplifier les fonctions de la bibliotèque précédente :

- Il n'est plus nécessaire d'utiliser une sentinnelle pour retrouver le premier élément de la liste ce qui permet de se passer de la structure qui englobait la liste simplement chaînée
- L'insertion d'un élément peut se faire aussi bien avant qu'après celui passé en paramètre de la fonction
- Il est maintenant possible de supprimer l'élément passé en paramètre et non plus le suivant
- Il faut une fonction qui retourne le maillon précédent

A part ces quelques points, les fonctions ont un fonctionnement identique.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

III - La pratique

III-A - La structure d'un maillon

Pour obtenir la structure de base d'une liste doublement chaînée, il suffit d'ajouter un pointeur sur l'élément précédent à la structure *item_s* vue dans l'article précédent. Comme la sentinelle est devenue inutile, il n'est plus nécessaire de créer une structure d'encapsidation de la liste.

```
stypedef struct dll
{
   struct dll *prev;
   struct dll *next;
   void *data;
} dll_s;
```

III-B - Initialisation

L'initialisation de la liste est devenue inutile, cependant la fonction est conservée pour être sûr que le pointeur est mis à NULL :

```
dll_s *dll_new (void)
{
   return (NULL);
}
```

III-C - Insertion d'un élément

Le principe est identique à celui des listes simplement chaînée, il ne faut pas oublier de mettre à jour le pointeur sur l'élément précédent.

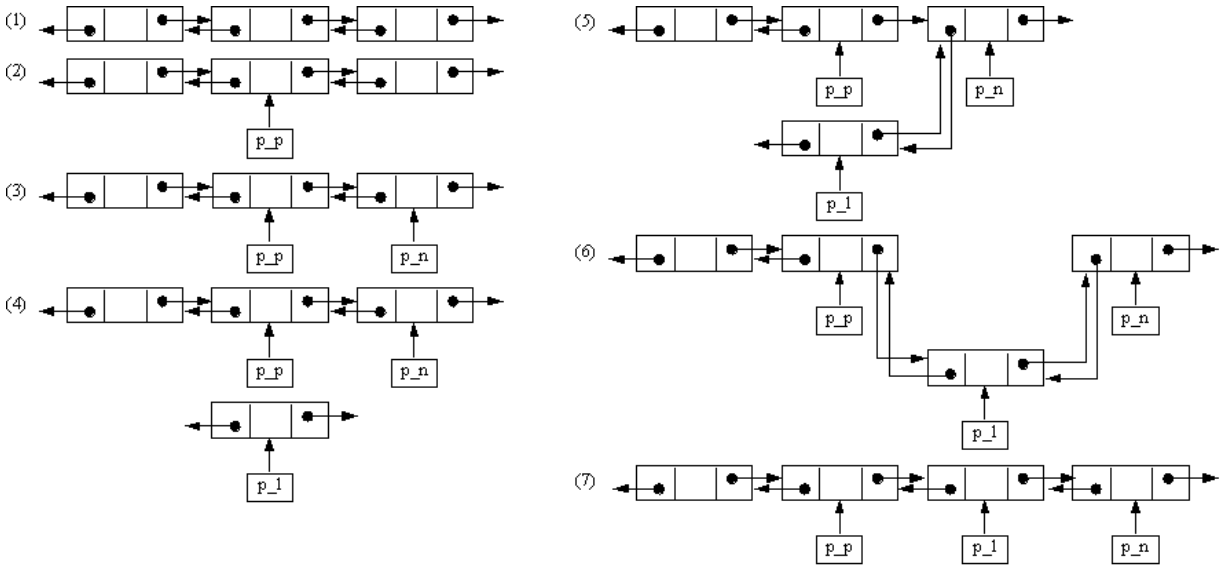

Figure 2 : représentation de l'insertion d'un élément.

1 Voici l'état de la liste avant l'appel de la fonction
2 Le pointeur *p_p* représente l'argument de la fonction, c'est donc après lui qu'il faut insérer un maillon

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

3 *p_n* pointe sur l'élément qui suit
4 Création d'un nouvel élément pointé par *p_l*
5 On commence par faire pointer le nouvel élément sur le maillon qui suit *p_p* (donc *p_n*) sans oublier de faire pointer ce dernier vers notre nouvel élément puisque maintenant la liste va dans les deux sens
6 Puis l'on modifie *p_p* pour que l'élément qui le suit soit celui nouvellement créé et réciproquement
7 Etat de la liste après l'appel de la fonction

Le code est légèrement différent du fait de l'absence de la structure d'encapsulation.

```
void dll_insert (dll_s ** pp_dll, void *data)
{
   if (pp_dll != NULL)
   {
/* (2) */
      dll_s *p_p = *pp_dll;
      dll_s *p_l = NULL;
      dll_s *p_n = NULL;

/* (3) */
      if (p_p != NULL)
         p_n = p_p->next;
/* (4) */
      p_l = malloc (sizeof (*p_l));
      if (p_l != NULL)
      {
         p_l->data = data;
/* (5) */
         p_l->next = p_n;
         if (p_n != NULL)
            p_n->prev = p_l;
/* (6) */
         p_l->prev = p_p;
         if (p_p != NULL)
            p_p->next = p_l;
         *pp_dll = p_l;
      }
      else
      {
         fprintf (stderr, "Memoire insuffisante\n");
         exit (EXIT_FAILURE);
      }
   }
}
```

III-D - Suppression d'un élément

Voici la fonction qui va subir le plus de modifications, elle va ainsi pouvoir supprimer l'élément passer en paramètre.

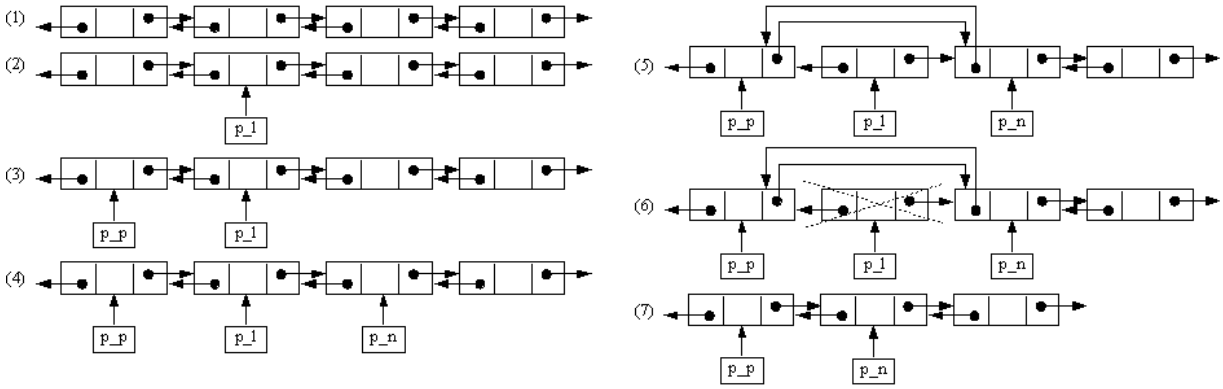

Figure 3 : représentation de la suppression d'un élément.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

1 Voici l'état de la liste avant l'appel de la fonction
2 Le pointeur *p_l* représente l'argument de la fonction et donc l'élément à supprimer
3 Sauvegarde de l'élément précédent celui à supprimer grâce au pointeur *p_p*
4 Sauvegarde de l'élément suivant celui à supprimer grâce au pointeur *p_n*
5 Modification des pointeurs de telle manière que le maillon à supprimer ne soit plus référencé
6 Libération de la mémoire
7 État de la liste après l'appel de la fonction

```
void dll_remove (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
   {
/* (2) */
      dll_s *p_l = *pp_dll;
/* (3) */
      dll_s *p_p = p_l->prev;
/* (4) */
      dll_s *p_n = p_l->next;

/* (5) */
      if (p_p != NULL)
         p_p->next = p_n;
      if (p_n != NULL)
         p_n->prev = p_p;
/* (6) */
      free (p_l);
      p_l = NULL;
      if (p_n != NULL)
         *pp_dll = p_n;
      else
         *pp_dll = p_p;
   }
}
```

III-E - Accès à l'élément précédent

Cette fonction se contente de déplacer le pointeur sur l'élément qui précédent, en faisant tout de même attention à la valeur de *pp_dll* :

```
void dll_prev (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
      *pp_dll = (*pp_dll)->prev;
}
```

III-F - Accès au premier élément

Avec l'apparition du pointeur sur l'élément précèdent, la sentinelle qui repérait le premier élément de la liste n'existe plus, ce qui a permis de simplifier certaines fonctions, cependant il faut retrouver ce premier élément de la même manière que le dernier élément.

```
void dll_first (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
   {
      while ((*pp_dll)->prev != NULL)
         dll_prev (pp_dll);
   }
}
```

III-G - Suppression de la liste

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

Comme la fonction de suppression des éléments a été modifiée, il est logique qu'il en soit de même pour la destruction de la liste.

```
void dll_delete (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
   {
      dll_first (pp_dll);
      while (*pp_dll != NULL)
         dll_remove (pp_dll);
   }
}
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

IV - Conclusion

Comme pour les listes simplement chaînées, si vous cherchez d'autres fonctions, il y a celles de la glib. Maintenant que nous avons fait le tour des listes chaînées, les deux prochains articles porteront sur des cas particuliers de liste chaînées : les piles et les files.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

V - Autres documentations

Liste doublement chaînée par CGI

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

VI - Remerciements

Merci à Anomaly pour sa relecture attentive de cet article.

Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

VII - Code source complet

Les exemples d'utilisation des listes doublements chaînées ne manquent pas puisqu'elles sont présentes dans les entrailles des systèmes d'exploitation (pour la gestion de la mémoire) jusqu'aux applications courantes (pour le tri des données par exemple). Téléchargez l'archive zippée.

listedouble.h

```
#ifndef H_LISTEDOUBLE
#define H_LISTEDOUBLE

#include <stddef.h>             /* pour size_t */

typedef struct dll dll_s;

dll_s *dll_new (void);
void dll_insert (dll_s **, void *);
void dll_remove (dll_s **);
void dll_prev (dll_s **);
void dll_next (dll_s **);
void *dll_data (dll_s *);
void dll_first (dll_s **);
void dll_last (dll_s **);
size_t dll_sizeof (dll_s *);
void dll_delete (dll_s **);

#endif /* not H_LISTEDOUBLE */
```

listedouble.c

```
#include <stdio.h>
#include <stdlib.h>
#include "listedouble.h"

struct dll
{
   struct dll *prev;
   struct dll *next;
   void *data;
};

dll_s *dll_new (void)
{
   return (NULL);
}

void dll_insert (dll_s ** pp_dll, void *data)
{
   if (pp_dll != NULL)
   {
      dll_s *p_p = *pp_dll;
      dll_s *p_l = NULL;
      dll_s *p_n = NULL;

      if (p_p != NULL)
         p_n = p_p->next;
      p_l = malloc (sizeof (*p_l));
      if (p_l != NULL)
      {
         p_l->data = data;
         p_l->next = p_n;
         if (p_n != NULL)
            p_n->prev = p_l;
         p_l->prev = p_p;
         if (p_p != NULL)
            p_p->next = p_l;
         *pp_dll = p_l;
      }
      else
      {
         fprintf (stderr, "Memoire insuffisante\n");
         exit (EXIT_FAILURE);
      }
   }
}
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

listedouble.c

```
void dll_remove (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
   {
      dll_s *p_l = *pp_dll;
      dll_s *p_p = p_l->prev;
      dll_s *p_n = p_l->next;

      if (p_p != NULL)
         p_p->next = p_n;
      if (p_n != NULL)
         p_n->prev = p_p;
      free (p_l);
      p_l = NULL;
      if (p_n != NULL)
         *pp_dll = p_n;
      else
         *pp_dll = p_p;
   }
}

void dll_next (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
      *pp_dll = (*pp_dll)->next;
}

void dll_prev (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
      *pp_dll = (*pp_dll)->prev;
}

void *dll_data (dll_s * p_dll)
{
   return ((p_dll != NULL) ? p_dll->data : NULL);
}

void dll_first (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
   {
      while ((*pp_dll)->prev != NULL)
         dll_prev (pp_dll);
   }
}

void dll_last (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
   {
      while ((*pp_dll)->next != NULL)
         dll_next (pp_dll);
   }
}

size_t dll_sizeof (dll_s * p_dll)
{
   size_t n = 0;

   if (p_dll != NULL)
   {
      dll_first (&p_dll);
      while (p_dll != NULL)
      {
         n++;
         dll_next (&p_dll);
      }
   }
   return (n);
}

void dll_delete (dll_s ** pp_dll)
{
   if (pp_dll != NULL && *pp_dll != NULL)
   {
      dll_first (pp_dll);
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/

listedouble.c
```
         while (*pp_dll != NULL)
            dll_remove (pp_dll);
      }
   }
```


Les sources présentés sur cette pages sont libres de droits, et vous pouvez les utiliser à votre convenance. Par contre la page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2006 - Nicolas Joseph. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://nicolasj.developpez.com/articles/listedouble/