rbuffer.h, un buffer tournant

Bernard Tatin

2013/2017

Voici un premier essai de *literate programming*, concept inventé par D. Knuth il y a plus de trente ans. À partir de ce seul fichier on génère la documentation et le code. Ici, je reprend du vieux code, cela m'oblige, même s'il est simple, à le repenser et donc, espérons le, à l'améliorer. Même si je passe beaucoup de temps sur la présentation...

Contents

1 rbuffer

C'est un buffer tournant le plus simple possible, capable de gérer des lignes délimitées par *LF* (`'\n'`) mais *CR* (`'\r'`) n'est pas pris en compte, plus exactement, il est rejetté.

1.1 premières définitions

Pour limiter les calculs, le code..., la taille du buffer est une puissance de 2 d'où la définition du nombre de bits qui ouvre le bal :

2a ⟨*intro-bits* 2a⟩≡
```
#define _RBUFFER_BITS    8
```
This definition is continued in chunk 2.
This code is used in chunk 5a.
Defines:
  _RBUFFER_BITS, used in chunk 2b.

La taille du buffer sera donc :

2b ⟨*intro-bits* 2a⟩+≡
```
#define RBUFFER_SIZE     (1 << _RBUFFER_BITS)
```
This code is used in chunk 5a.
Defines:
  RBUFFER_SIZE, used in chunks 2–4.
Uses _RBUFFER_BITS 2a.

Et le masque permettant un rapidide *modulo* arithmétique avec un **and** binaire :

2c ⟨*intro-bits* 2a⟩+≡
```
#define RBUFFER_MASK     (RBUFFER_SIZE - 1)
```
This code is used in chunk 5a.
Defines:
  RBUFFER_MASK, used in chunk 4b.
Uses RBUFFER_SIZE 2b.

1.2 la structure

Note: tous les membres de la structure sont définis comme `volatile`. C'est important dans un système embarqué avec des interruptions pouvant manipuler le buffer. Sans `volatile`, une optimisation trop agressive pourrait placer une des valeurs entières dans un registre. En cas d'interruption modifiant cette valeur, le registre, lui, ne bougera pas et des caractères pourraient se perdre.

3a ⟨*tsrbuffer* 3a⟩≡
```
/**
 * @struct TSrbuffer
 * La structure gérant le buffer tournant.
 */
typedef struct {
    volatile int in;
    volatile int out;
    volatile int line_count;
    volatile char buffer[RBUFFER_SIZE];
} TSrbuffer;
```

Root chunk (not used in this document).
Defines:
`TSrbuffer`, used in chunk 4b.
Uses `RBUFFER_SIZE` 2b.

1.2.1 les champs

1.2.2 remarques diverses

On pourrait définir un `VOLATILE` en fonction de l'architecture du type :

3b ⟨*define-volatile* 3b⟩≡
```
#if defined(__with_irqs)
  #define VOLATILE volatile
#else
  #define VOLATILE
#endif
```

This code is used in chunk 5a.

Ce qui donnerait au final :

4a ⟨*tsrbuffer-final* 4a⟩≡
```
/**
 * @struct TSrbuffer
 * La structure gérant le buffer tournant.
 */
typedef struct {
    VOLATILE int in;
    VOLATILE int out;
    VOLATILE int line_count;
    VOLATILE char buffer[RBUFFER_SIZE];
} TSrbuffer;
```
This code is used in chunk 5a.
Defines:
TSrbuffer, used in chunk 4b.
Uses **RBUFFER_SIZE** 2b.

1.3 le fonctionnement

1.3.1 ajout d'un caractère

Le fonctionnement est le suivant pour l'ajout d'un caractère :

- si le caractère est '\r', on ne fait rien,
- on place le caractère dans le buffer à la position **in**,
- on incrémente **in**,
- si on atteint la limite du buffer, on positionne **in** à 0,
- si le caractère est '\n', on incrémente **line_count**.

4b ⟨*add-char* 4b⟩≡
```
static INLINE void rbf_add_char(TSrbuffer *rb, const char c) {
    if (c != '\r') {
        rb->buffer[rb->in++] = c;
        rb->in &= RBUFFER_MASK;
        if (c == '\n') {
            rb->line_count++;
        }
    }
}
```
This code is used in chunk 5a.
Defines:
rbf_add_char, never used.
Uses **RBUFFER_MASK** 2c and **TSrbuffer** 3a 4a 4a.

1.4 le code final

5a ⟨* 5a⟩≡
⟨*intro-bits* 2a⟩
⟨*define-volatile* 3b⟩
⟨*tsrbuffer-final* 4a⟩
⟨*add-char* 4b⟩
Root chunk (not used in this document).

2 annexes

2.1 la ligne de commande

Pour obtenir le fichier LaTeX et le code source, voici ce qu'il faut faire depuis un terminal :

5b ⟨*command-line* 5b⟩≡
```
# fichier LaTeX
noweave -delay -autodefs c -index rbuffer.nw > rbuffer.tex
# fichier PDF
pdflatex rbuffer.tex && \
  pdflatex rbuffer.tex && \
  pdflatex rbuffer.tex
# le code source
notangle rbuffer.nw > rbuffer.h
```
Root chunk (not used in this document).

L'option `-autodefs c` permet à `noweave` de déterminer lui-même les éléments du langage C. Sans cette option, dans le cadre de ce fichier, les définitions de `intro-bits` ne seraient pas visibles.

3 tables et index

3.1 table des extraits de code

3.2 index