CODE CRC

(Cyclic Redondancy Code) (11-03-2020) R. Coquidé

code détecteur d'erreur

En informatique, les données (numériques, littérales, sonores, graphiques…) sont transformées, mémorisées, transférées, manipulées… sous forme binaire. On utilise pour cela un codage : c'est une convention qui fera correspondre par exemple tel caractère à telle configuration : suite de bits (ou chiffres binaires) ordonnés (exemples ASCII, EBCDIC, UNICODE, OCTAL, HEXADECIMAL...).

Les supports de mémorisation sont soumis au vieillissement (bande magnétique, cassette, CD, DVD, disque dur, SSD, clé USB...). Les transferts sont soumis aux perturbations parasites. Une « erreur » résultante se traduit par la transformation d'un « **1** » en « **0** » ou inversement.

Il est donc de bon usage de « détecter » si possible, l'existence d'une erreur (ou plusieurs). L'utilisateur conscient peut alors, si c'est possible et souhaitable, recommencer la lecture ou demander à l'expéditeur de ré- expédier le message (en osant espérer que cette erreur, ou une autre, ne se produira pas !).

Le plus simple est le bit de parité. Exemple :

Soit un message à transmettre : **m = 0 1 0 0 1 0 1 1**

On transmet le message codé **mc = 0 1 0 0 1 0 1 1 0**

Le dernier bit concaténé **0** (dit bit de parité) a été choisi de telle sorte que le nombre de **1** dans **mc** soit pair.

S'il se produit « **<u>une</u>** » erreur pendant le transfert ou la lecture, c'est-à-dire le changement d'un « **1** » en « **0** » ou inversement, on constatera dans tous les cas l'anomalie en examinant le bit de parité. Tout nombre impair d'erreurs sera détecté par ce procédé mais tout nombre pair d'erreurs sera indétectable. Si le bit de parité est erroné, une erreur sera détectée à tort. Ce procédé a donc ses limites.

La méthode du CRC

Elle est basée sur les propriétés de la division, en binaire, de 2 polynômes à coefficients binaires (plus précisément modulo 2).

Les coefficients de ces polynômes sont uniquement des **0** et des **1** ainsi que pour le quotient et le reste.

Exemple : Soit à diviser $\mathbf{N(x)=x^7+x^5+x^4+x^2+x}$ par $\mathbf{D(x)=x^4+1}$

$x^7 + x^5 + x^4 + x^2 + x \mid x^4+1$

$\backslash \quad \backslash \quad \backslash \; -x^3 \quad -x \mid \underline{\quad}$

$-1 \qquad x^3+x+1 = Q(x)$

$\overline{\quad\quad\quad\quad}$

$-x^3+x^2 -1 = R(x)$

On a **N(x) = D(x).Q(x)+R(x)**

Or -1 =1 (mod 2)

Donc **R(x) = x^3+x^2+1**

En informatique, ce reste est calculé avec les vecteurs booléens des coefficients en utilisant le « **ou exclusif** ». Cela se justifie par le fait que en calcul modulo 2 l'addition et la soustraction donnent le même résultat.

Le « **OU Exclusif** » noté **~:** en **J** (différent de) vérifie :

0 ~: 0 = 0
1 ~: 1 = 0
0 ~: 1 = 1
1 ~: 0 = 1

On identifie un polynôme à ses coefficients booléens.
Dans notre exemple : **N(x) → 1 0 1 1 0 1 1 0** **D(x) → 1 0 0 0 1**

```
  1 0 1 1 0 1 1 0
  1 0 0 0 1 : : :
= 0 0 1 1 1 1 1 :
      1 0 0 0 1 :
    = 0 1 1 1 0 0
        1 0 0 0 1
      = 0 1 1 0 1
```

on a **1 1 0 1 → R(x) = x^3+x^2+1**

Utilisation pour le code CRC

Soit à expédier le message **M = 1 1 1 0 0 1 1 1**
On utilise un polynôme diviseur (nommé polynôme générateur) de coefficients binaires **PG = 1 0 1 1 0** de degré **4**
On ajoute à droite **4** zéros à **M** (nombre de zéros égal au degré de **PG**)

```
  1 1 1 0 0 1 1 1 0 0 0 0      On calcule le reste de la division
  1 0 1 1 0 : : : : : : :
= 0 1 0 1 0 1 : : : : : :
    1 0 1 1 0 : : : : : :
  = 0 0 0 1 1 1 1 0 : : :
        1 0 1 1 0 : : :
    = 0 1 0 0 0 0 : :
          1 0 1 1 0 : :
      = 0 0 1 1 0 0 0
            1 0 1 1 0
          = 0 1 1 1 0
```

⇒ **CRC = 1 1 1 0 =** reste de la division

On transmet le message codé **MC = M,CRC = 1 1 1 0 0 1 1 1 ,1 1 1 0**
A la réception du message codé **MC = 1 1 1 0 0 1 1 1 1 1 1 0**

1) On calcule le **reste** de la division de **MC** par **PG = 1 0 1 1 0**

2a) **Si le reste est nul**, on supprime les 4 derniers bits de **MC** pour obtenir le message binaire **M = 1 1 1 0 0 1 1 1 .**

2b) **Si le reste est non nul**, il y a erreur(s). On peut alors soit se faire une raison ou recommencer le transfert ou la lecture en osant espérer que ce soit, alors, sans erreur.

Calcul du reste de la division de **MC = 1 1 1 0 0 1 1 1 1 1 1 0** par **PG = 1 0 1 1 0**

Exemple sans erreur

```
  1 1 1 0 0 1 1 1 1 1 1 0
  1 0 1 1 0 : : : : : : :
= 0 1 0 1 0 1 : : : : : :
    1 0 1 1 0 : : : : : :
  = 0 0 0 1 1 1 1 1 : : :
        1 0 1 1 0 : : :
    = 0 1 0 0 1 1 : :
        1 0 1 1 0 : :
      = 0 0 1 0 1 1 0
            1 0 1 1 0
          = 0 0 0 0 0
```

reste nul ⇒ pas d'erreur

Exemple avec une erreur

```
  1 1 1 0 0 1 0 1 1 1 1 0
  1 0 1 1 0 : : : : : : :
= 0 1 0 1 0 1 : : : : : :
    1 0 1 1 0 : : : : : :
  = 0 0 0 1 1 0 1 1 : : :
        1 0 1 1 0 : : :
    = 0 1 1 0 1 1 : :
        1 0 1 1 0 : :
      = 0 1 1 0 1 1 :
            1 0 1 1 0 :
          = 0 1 1 0 1 0
                1 0 1 1 0
              = 0 1 1 0 0
```

reste non nul ⇒ il y a erreur(s)

On démontre que le polynôme générateur **PG** doit être un polynôme irréductible dans l'anneau des polynômes $\mathbb{F}_2$**[x] où** $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$**.** Nous admettons ici ce résultat de la théorie des polynômes.

Cet exemple a été choisi, avec le polynôme générateur **PG = 1 0 1 1 0** de degré 4 (5 coefficients binaires) parce que les calculs sont relativement simples, à la main. Par ordinateur, on n'hésite pas à choisir des calculs beaucoup plus complexes avec des polynômes générateurs de degrés plus élevés. On démontre que plus le degré du polynôme générateur est élevé, plus la probabilité de laisser passer une erreur non détectée est faible… mais aussi plus c'est coûteux en mémoire et en temps de codage-décodage.

Selon l'utilisation que l'on veut en faire, on choisit le polynôme générateur parmi des polynômes « normalisés » dont une liste (non exhaustive) suit.

Le polynôme générateur **x+1 (norme CRC-1)** correspond au bit de parité… qui laisse passer beaucoup d'erreurs. On estime que si le polynôme $\mathbf{x^{64}+x^4+x^3+x+1}$ **(norme CRC-64-ISO)** est utilisé, plus de 99,5% des erreurs sont détectées.

QUELQUES POLYNÔMES GÉNÉRATEURS NORMALISÉS

NORME	POLYNOME GENERATEUR
CRC-1	$x+1$
CRC-3-GSM	x^3+x+1
CRC-4-UIT	x^4+x+1
CRC-4-V	x^4+x^2+1
CRC-5-CBE	x^5+x^3+1
CRC-5-UIT	$x^5+x^4+x^2+1$
CRC-5-USB	x^5+x^2+1
CRC-6-GSM	$x^6+x^5+x^3+x^2+x+1$
CRC-6-UIT	x^6+x+1
CRC-7	x^7+x^3+1
CRC-8	$x^8+x^7+x^6+x^4+x^2+1$
CRC-8-CCITT	x^8+x^2+x+1
CRC-8-AB	$x^8+x^5+x^3+x^2+x+1$
CRC-8-DALLAS	$x^8+x^5+x^4+1$
CRC-8-GSM	$x^8+x^4+x^3+x^2+1$
CRC-8-WCDMA	$x^8+x^7+x^4+x^3+x+1$
CRC-10-GSM	$x^{10}+x^9+x^5+x^4+x+1$
CRC-11	$x^{11}+x^9+x^8+x^7+x^2+1$
CRC-12-GSM	$x^{12}+x^{11}+x^3+x^2+x+1$
CRC-13	$x^{13}+x^{12}+x^{11}+x^{10}+x^7+x^6+x^5+x^4+x^2+1$
CRC-15	$x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1$
CRC-16-CCITT	$x^{16}+x^{12}+x^5+1$
CRC-16-MA2000	$x^{16}+x^{10}+x^8+x^7+x^3+1$
CRC-16-SCSIDIF	$x^{16}+x^{15}+x^{11}+x^9+x^8+x^7+x^5+x^4+x^2+x+1$
CRC-16-DNP	$x^{16}+x^{13}+x^{12}+x^{11}+x^{10}+x^8+x^6+x^5+x^2+1$
CRC-16-IBM	$x^{16}+x^{15}+x^2+1$
CRC-24	$x^{24}+x^{22}+x^{20}+x^{19}+x^{18}+x^{16}+x^{14}+x^{13}+x^{11}+x^{10}+x^8+x^7+x^6+x^3+x+1$
CRC-24-RADIX	$x^{24}+x^{23}+x^{18}+x^{17}+x^{14}+x^{11}+x^{10}+x^7+x^6+x^5+x^4+x^3+x+1$
CRC-24-WCDMA	$x^{24}+x^{23}+x^6+x^5+x+1$
CRC-30	$x^{30}+x^{29}+x^{21}+x^{20}+x^{15}+x^{13}+x^{12}+x^{11}+x^8+x^7+x^6+x^2+x+1$
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
CRC-32-C	$x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^9+x^8+x^6+1$
CRC-32-K	$x^{32}+x^{31}+x^{29}+x^{28}+x^{26}+x^{20}+x^{19}+x^{17}+x^{16}+x^{15}+x^{11}+x^{10}+x^7+x^6+x^4+x^2+x+1$
CRC-32-Q	$x^{32}+x^{31}+x^{24}+x^{22}+x^{16}+x^{14}+x^8+x^7+x^5+x^3+x+1$
CRC-64-ECMA	$x^{64}+x^{62}+x^{57}+x^{55}+x^{54}+x^{53}+x^{52}+x^{47}+x^{46}+x^{45}+x^{40}+x^{39}+x^{38}+x^{37}+x^{35}+x^{33}+x^{32}+x^{31}+x^{29}+x^{27}+x^{24}+x^{23}+x^{22}+x^{21}+x^{17}+x^{13}+x^{12}+x^{10}+x^9+x^7+x^4+x+1$
CRC-64-ISO	$x^{64}+x^4+x^3+x+1$

QUELQUES UTILISATIONS

Réseaux mobiles : CRC-3-GSM, CRC-6-GSM, CRC-8-GSM, CRC-8-WCDMA, CRC-10-GSM,CRC-12-GSM,CRC15,CRC-16-MA2000

Télécommunications : CRC-7

Canal de données radio : CRC-7

Téléphone sans fil : CRC-16-MA2000

Modem : CRC-16-CCITT

Automobile : CRC-8

Train : CRC-7

Aviation : CRC-32-Q

Bluetooth : CRC-8

ANSI : CRC-16-IBM,CRC-32

ISO : CRC-32

SATA : CRC-32

MPEG2 : CRC-32

PKZIP : CRC-32

Cartes SD : CRC-7

USB : CRC-5-USB, CRC-16-IBM

Bus : CRC-8-DALLAS, CRC-16-IBM

NB. **CodeCRC.ijs**

NB. Calculs en modulo 2 (somme=différence=ou exclusif = XOR = ~:)

NB. ***R =. Sg0 y*** : Suppression des zéros à gauche dans le vecteur booléen y

NB. (simule un registre à décalage)

```
Sg0 =: +./\#]
```

NB. ***R =. pg Pbmod pol*** : R = reste de la division du polynôme pol par le

NB. polynôme générateur pg ; on utilise les vecteurs booléens des coefficients des polynômes

```
    Pbmod =: 4 : 0"_ _
nn=.#num=.Sg0 y [ nd=.#div=.x=.Sg0 x
while. nn>:nd do. R=.Sg0 div ~: nd{.num
nn=.#num=.R,nd}.num end.
R=.(1-nd){.(nd$0),R
)
```

NB. ***R =. pg CodCRC M*** : R = code CRC du message M (binaire)

NB. avec pg (polynôme générateur)

```
    CodCRC =: 4 : 'x Pbmod y,(_1+#x)$0'"_ _
```

NB. ***MC =. pg CodageCRC M*** : MC = message codé (avant expédition)

NB. par le polynôme générateur pg

```
    CodageCRC =: [: : ],[CodCRC]
```

NB. ***R =. pg TestCRC MC*** : R = reste de la division du message codé reçu MC

NB. par le polynôme générateur pg.

NB. (pas d erreur si et seulement si R = polynôme à coefficients nuls)

```
    TestCRC =: [ Pbmod ]
```

NB. ***M =. pg DecodageCRC MC*** : M = message originel en clair si pas d erreur

NB. ou demande de rè-expédition du message codé si détection d'erreurs

```
    DecodageCRC =: ((1:-#@[)}.])`('Message codé
erroné'"_)@.([: +./ [ Pbmod ])"_ _
```

NB. ***M =. TEXT2BIN T*** : Texte T transformé en vecteur binaire (ou message) M

```
    TEXT2BIN =: 3 : ',(8$2)(#:"_ 0)a.i.y'
```

NB. *T =. BIN2TEXT M* : Message binaire transformé en caractères ASCII

```
BIN2TEXT =: 3 : '(2(#."_ 1)((8%~#y),8)$y){a.'
```

NB. *MC =. pg TEXT2MC T* : transformation du texte ASCII en message codé

```
TEXT2MC =: [CodageCRC [: TEXT2BIN]
```

NB. *T =. pg MC2TEXT MC* : transformation du message codé en texte ASCII

```
MC2TEXT=: ([:BIN2TEXT]) ::]@([DecodageCRC])
```

Exemple 1

a) Sans erreur de transmission :

pg =. 1 0 0 1 1 NB. Vecteur des coeff. du polynôme générateur

NB. De degré 4 (5 coefficients)

T =.'Bonjour chez vous' NB. Texte en clair du message

MC =. pg TEXT2MC T NB. Texte transformé en message codé

pg MC2TEXT MC NB. Transformation du message codé en texte

Bonjour chez vous NB. Récupération du texte

#MC NB. Taille du vecteur booléen MC

140 NB. Dont 4 de CRC soit 136 = 17*8 (17 octets)

b) Avec erreur(s) de transmission :

MC =. (-. 20{MC) 20 } MC NB. Erreur sur la composante n°20 de MC

pg MC2TEXT MC NB. Tentative de retrouver le texte

Message codé erroné NB. Il y a eu détection d'erreur(s)

NB. On a le choix :

NB. 1) Demander la ré-expédition du message codé MC

NB. 2) Se faire une raison et exploiter au mieux le message erroné

M0 =. _4}.MC NB. Suppression des 4 derniers bits (CRC)

NB. Nombre égal au degré du pol générateur

BIN2TEXT M0 NB. Texte en clair du message erroné

Bofjour chez vous NB. Avec un peu de chance on devinera

NB. le texte d'origine

Exemple 2 :

a) Sans erreur de transmission :

[PG =. |. 1 (0 1 2 3 5 8)}9$0

1 0 0 1 0 1 1 1 1

NB. C'est le vecteur booléen du polynôme générateur $\mathbf{x^8+x^5+x^3+x^2+x+1}$

NB. (9 composantes) défini par la norme **CRC-8-AB**

NB. Voici le texte en clair :

T=. 'Le soir tombait (BOUM !). Il tombait bien, d''ailleurs, pour remplacer le jour devenu si blafard qu''on susurrait qu''il ne passerait pas la nuit.'

MC =. PG TEXT2MC T NB. Création du message codé

NB. Transmission sans erreur(s)

PG MC2TEXT MC NB. Restitution du texte

Le soir tombait (BOUM !). Il tombait bien, d'ailleurs, pour remplacer le jour devenu si blafard qu'on susurrait qu'il ne passerait pas la nuit.

b) Avec erreur de transmission

$MC

1152

MCe =. (-. 5 20 87{MC) 5 20 87} MC

PG MC2TEXT MCe

Message codé erroné

BIN2TEXT _8}.MCe

He(soir tolbait (BOUM !). Il tombait bien, d'ailleurs, pour remplacer le jour devenu si blafard qu'on susurrait qu'il ne passerait pas la nuit.