Construire un texte source

Quel que soit le langage utilisé, il est important de structurer le code source d'un programme. Plus le texte sera agréable à l'oeil dans sa structuration, plus il sera compréhensible instantanément, donc moins il y aura d'erreur.

1) La ligne :

La ligne d'un texte source est composée de caractères alphabétiques, numériques, symboliques ainsi que des caractères spéciaux invisibles.
L'explosion de l'informatique à la fin du XX^{ième} siècle a provoqué une multitude de codages d'un texte des plus complexes au plus simples. Malheureusement, même dans les plus simples il n'y a pas concordance exacte sur le codage ASCII.
Entre parenthèses, c'est inimaginable, depuis le temps que les technologies les plus pointues, les processeurs dépassants le GHz, qu'il n'y ait pas un codage unique pour un retour à la ligne ou pour une lettre accentuée.
Ceci posé, nous allons donc oeuvrer dans cette tour de Babel, sans être trop ambitieux. L'objectif premier est d'obtenir une impression du texte source qui soit fidèle au texte à l'écran. Pour cela, le bannissement des tabulations semble indispensable (il y a autant d'éditeur de texte que de représentation de la tabulation). Nous la remplaceront par le nombre d'espaces voulu. Certains éditeurs faciliteront la tâche, indentation automatique avec des espaces, remplacement automatique des tabulations avec des espaces; d'autres imposeront systématiquement une tabulation...
Si la police de caractère est modifiable, l'indentation ressortira mieux avec une police à approche fixe (police Monaco par exemple).
Le rendu du texte à l'imprimante dépend aussi du nombre de caractères maximum imprimés dans chaque lignes par l'imprimante. Néanmoins, 80 caractères maximum par ligne semble être un nombre couramment utilisé (peut être une survivance du format des antiques terminaux VT). Ce maximum peut paraître faible, il ne doit pas être considéré comme un mur infranchissable mais comme un repère au delà duquel le source risque d'être moins lisible.

2) En-têtes :

En premier lieu les en-têtes de fichiers ou d'unités de programme contiennent d'une part les informations décrivant la personne ou l'entité gérant le texte et d'autre part la description du contenu du texte.
C'est ainsi que nous retrouveront le copyright, le nom de l'auteur ou de l'entité (équipe, société), le rôle du source dans l'application, l'historique des évolutions avec la date de dernière mise à jour.
La forme de l'en-tête sera bien évidemment identique pour tous les textes sources sous forme de commentaires adaptés au langage de programmation employé.
Nous en profiteront pour insérer une ligne comportant le nombre maximum de caractères par ligne que l'on s'est fixé précédemment, comme ceci les dépassements seront repérés en un coup d'oeil.

Exemple :

```
(*------------------------------------------------------------------*)
(*
-- NOM DU CSU (principal)           : attrib.p
-- AUTEUR DU CSU                    : P. Pignard
-- VERSION DU CSU                   : 1.0b
-- DATE DE LA DERNIERE MISE A JOUR  : 28 décembre 2000
-- ROLE DU CSU                      : Algorithme d'attribution.
--
--
-- FONCTIONS EXPORTEES DU CSU       :
--
--
-- FONCTIONS LOCALES DU CSU         :
--
--
-- NOTES                            :
--
--
*)
(*------------------------------------------------------------------*)
```

Note : un CSU (Computer Software Unit) est le composant de base d'un programme. Il a la forme de programmes principaux, d'unités, de spécifications, de corps, de définitions... suivant le langage pratiqué.

3) Commentaires

Un commentaire doit être réservé pour exprimer une information qui ne peut être exprimée dans le code ou pour expliquer pourquoi telle ou telle règle n'a pas pu être appliquée. Dans la mesure du possible le texte source doit être compréhensible par lui même.

4) Indentation

L'indentation est la représentation "graphique" de la structuration du langage employé. Elle est donc spécifique à chaque langage et aussi à la compréhension de cette structuration par le programmeur. L'important est de déterminer un style propre et de s'y tenir.

5) Les aides au formatage du texte

Les reformateurs de code source, en anglais "beautifier" ou "pretty printer", existent sans être vraiment populaires. Pourtant, leur utilisation systématique permettra d'améliorer la lisibilité et maintenabilité du code. Peut-être la raison de cette impopularité est que l'on touche ici au style de codage de chacun de nous. Le style est propre à chacun des programmeurs, l'uniformiser automatiquement est ressenti comme perdre cette touche personnelle.
Seul exemple connu à ce jour comme réellement d'un emploi systématique est le code source du compilateur Ada GNAT qui provoque un arrêt de compilation si le style requis n'est pas reconnu.
De contraignant d'un premier abord, cette règle évite une dissonance des codes sources en licence libre pouvant être manipulés par toute la communauté des programmeurs.
Même si l'on travaille seul sur le texte d'un code source, l'uniformisation de la syntaxe apporte deux avantages décisifs. Dans un premier temps, son utilisation permet une liberté plus ou moins grande à la saisie du code qui sera effacée après reformatage. Donc moins de perte de temps au respect de la forme un peu fastidieuse à la saisie ou lors de la modification du code.
Deuxième temps, le suivi des modifications est simplifiée car les différences apparaissant ne sont réellement que des différences de codage et ne sont pas mélangées avec des différences de présentations.
Cela est vraiment appréciable lors d'un travail à plusieurs sur un texte, là aussi les différents styles de codage sont gommés par le reformateur.
Définir les valeurs des paramètres du formatage est délicat, on en revient

au style de chacun qui quelque soit le paramétrage ne coïncidera jamais exactement.
Je recommande donc d'adopter le paramétrage conduisant au style le plus proche du sien et de s'habituer à ce nouveau style. Les avantages gommeront les inconvénients des premières heures.
De nombreux utilitaires existent. En majorité chacun est dédié à un langage particulier.
J'utilise "Tidy" pour le code HTML, "Jalopy" pour Java et "gnatpp" pour Ada. (Voir sur Blady en page liens)
Pour le code Ada, "gnatpp" produit par défaut la syntaxe des sources de GNAT. Les valeurs par défaut sont :

- indentation avec retrait de 3 caractères
- continuation de ligne avec retrait de 2 caractères
- longueur maximale des lignes fixée à 79 caractères
- mots clés du langage en minuscule
- attributs et pragmas avec premier caractère en majuscule

Pour le code C, "indent" est présent par défaut sur Mac OS X.

Conclusion :

Le code source d'un logiciel est un ensemble de fichiers écrit par une ou plusieurs personnes. Le texte doit être suffisamment clair pour pouvoir être facilement relu par une personne ne l'ayant pas écrit. La forme est ainsi aussi importante que les instructions inscrites. Un texte source avec une bonne mise en forme contiendra moins d'erreurs car elles auront été plus facilement décelées.

Pascal Pignard Avril 2001, Septembre 2006.