Printer

Printer est un logiciel de génération de programmes. Il propose un format technique permettant la réalisation de code source dans n’importe quel langage.

Printer lit un fichier en entrée et exécute ce fichier pour obtenir un résultat sous la forme d’un texte. Le fichier en entrée est structuré de manière à pouvoir être facile à construire et réaliser.

Ce document présente le format du fichier d’entrée et explique la compilation de ce fichier pour en obtenir le texte final.

# La configuration

Il s’agit d’un couple nom/valeur utilisable à n’importe quel endroit du fichier d’entrée. Le texte peut comporter des retours chariots.

Le fichier d’entrée contient la liste des couples nom/valeur et la valeur est entourée par deux guillemets. Si le contenu contient un ou plusieurs guillemets, ils sont remplacés par l’entité « **:dbquot;** » et tout caractère « : » est remplacé par l’entité « **:dbdot;** ».

# Les variables

Les variables sont exploitées dans la génération du texte ; elles permettent de personnaliser le contenu à l’exécution du fichier d’entrée.

Elles sont de deux types :

1. Une variable simple, un couple nom/valeur.
2. Une variable qui inclut un fichier.   
   Cette possibilité permet de décrire le codage descriptif d’une boucle et du corps de celle-ci.

Il est possible d’augmenter l’indentation ; dans ce cas, l’exécution construit le fichier de sortie en indentant chaque nouvelle ligne.

# Le texte final

Le texte contenu dans le fichier d’entrée est directement recopié dans le fichier de sortie. Le texte peut comporter des noms de variables qui sont alors entre crochets.

# Inclusion

Le nom du fichier inclus peut être nommé par la configuration. Les variables appliquées au fichier inclus peuvent être redéfinies dans le fichier d’entrée.

Par exemple, je peux déclarer une boucle for

var indented for = ‘for’ [

var start = ‘int i’

var limit = ‘i < 10’

var increment = ‘i++’

var body = ‘# body’

]

Une autre possibilité est celle de créer des séquences continues.

var list = ‘list’ [

var element = ‘un’

var separator = ‘,’

var next = ‘list’ [

var element = ‘deux’

var separator = ‘,’

var next = ‘list’ [

var element = ‘trois’

var separator = ‘’

var next = ‘’

]

]

]

Je peux également définir une instruction complète.

var statement = ‘st’ [

var content = ‘content’

var delimiter = ‘ ;’

]