TABLE DES MATIÈRES

9. ENTRÉES ET SORTIES FORMATÉES : FONCTIONS SCANF & PRINTF

9.1. DÉFINITIONS GÉNÉRALES

Les flux d'entrées-sorties standards sont limités aux manipulations sur des caractères codés en `ASCII`. Cette limitation est très contraignante lorsque l'on cherche à formater l'affichage de valeurs numériques. C'est pour dépasser ces contraintes que l'on a introduit deux fonctions dédiées à la «traduction» entre valeurs numériques et chaînes de caractères. La fonction **`printf`** sert à formater les sorties sur le flux `stdout` et la fonction **`scanf`** sert à formater les entrées reçues sur le flux `stdin`.

Les prototypes génériques de ces deux fonctions se présentent sous la forme :

```
scanf(control, arg1, arg2, ...)
printf(control, arg1, arg2, ...)
```

- **`control`** est une chaîne de caractère qui sert à mettre en forme les arguments. Cette chaîne peut contenir des caractères «ordinaires» qui sont recopiés directement depuis ou vers le flux d'entrée-sortie standard et des formats de conversion positionnés dans la chaîne à l'aide du caractère balise `%` suivi du format choisi.
- Les arguments **`arg1, arg2, ...`**, doivent correspondre à chaque caractère `%` placé dans la chaîne **`control`**.

Le caractère `%` est une balise de positionnement suivie d'une spécification de traduction dont le prototype est : **`%[indicateur][largeur][.precision][modification]type`**.

- Tous les champs notés entre crochets sont optionnels.
- Spécifications du champ obligatoire **`type`** :

Tableau 4. Types de conversion

type	signification
%c	caractère
%s	chaîne de caractères
%d	nombre entier en décimal
%e	nombre réel sous la forme mantisse/exposant [-]m.nnnnnne[+|-]xx

type	signification
%E	nombre réel sous la forme mantisse/exposant en majuscule [-]m.nnnnnnE[+|-]xx
%f	nombre réel sous la forme [-]mmm.nnnnnn
%g	nombre réel sous la forme la plus courte entre les types %e et %f
%G	nombre réel sous la forme la plus courte entre les types %E et %f
%o	nombre entier en octal
%p	pointeur ou adresse de la valeur numérique
%u	nombre entier non signé en décimal
%x	nombre entier en hexadécimal
%X	nombre entier en hexadécimal ; lettres affichées en majuscules

- Spécifications du champ optionnel **[indicateur]** :

Tableau 5. Indicateur d'affichage

[indicateur]	signification
-	alignement à gauche pour la largeur donnée
+	affichage forcé du signe de la valeur numérique
espace	insertion d'un caractère d'espacement si la valeur numérique est positive
#	affichage précédé de 0, 0x ou 0X avec les types respectifs o, x ou X
	force l'affichage du point décimal avec les types e, E et f même si la partie décimale ne contient que des zéros
	force l'affichage du point décimal avec les types g et G sans supprimer les zéros inutiles

- Spécifications du champ optionnel **[largeur]** :

Tableau 6. Largeur de l'affichage

[largeur]	signification
nombre	nombre minimum de caractères à afficher ; ajout de caractères d'espacement si la valeur est plus «courte» que l'affichage demandé
0nombre	idem ci-dessus ; ajout de caractères 0 si la valeur est plus «courte» que l'affichage demandé

[largeur]	signification
*nombre	la largeur n'est pas spécifiée dans la chaîne **control** mais par un entier précédent l'argument à afficher

- Spécifications du champ optionnel **[precision]** :

Tableau 7. Precision de l'affichage

[precision]	signification
.nombre	pour les types d, i, o, u, x et X : nombre minimum de chiffres décimaux à afficher ; ajout de caractères d'espacement si la valeur est plus «courte» que l'affichage demandé
	pour les types e, E et f : nombre de chiffres à afficher après le point décimal
	pour les types g et G : nombre maximum de chiffres significatifs à afficher
	pour le type s : nombre maximum de caractères à afficher
	pour le type c : pas d'effet

- Spécifications du champ optionnel **[modification]** :

Tableau 8. Modification de l'affichage

[modification]	signification
h	argument traité comme un entier court (*short*)
l	argument traité comme un entier long pour les types entiers (*long*) ou comme un réel double pour les types réels (*double*)
L	argument traité comme un réel *long double*

Voici deux programmes d'illustration de sorties formatées avec la fonction **printf**. Le premier bénéficie du jeu de définitions complet sur GNU/Linux. Le second, exécuté sur un microcontrôleur MSC1210 et compilé avec SDCC, ne dispose que d'un jeu de définitions de formats plus restreint.

9.2. EXEMPLE AVEC GCC

Exemple 11. Sorties formatées avec printf sur GNU/Linux avec GCC

```
01: /* $Id: gcc_printf.c 1129 2007-05-07 15:07:52Z la
02:
```

```
03: #include <stdio.h>
04:
05: #define M_PI  3.14159265358979323846  /* pi */
06:
07: #define YEAR 2007
08:
09: int main() {
10:
11:    printf ("Caracteres : %c %c %c %c\n", 'a', 65,
12:    printf ("Entiers : %d %ld\n", YEAR, 650000);
13:    printf ("Affichage avec espaces : |%10d|\n", Y
14:    printf ("Affichage avec zeros : |%010d|\n", YE
15:    printf ("Differentes bases : %d %x %o %#x %#o
16:    printf ("Reels : |%4.2f| |%+.4e| |%E|\n", M_PI
17:    printf ("Largeur en argument : |%*d|\n", 5, 10
18:    printf ("%s\n", "Debian GNU/Linux");
19:    return 0;
20: }
```

Voici un exemple d'exécution du programme gcc_printf.c :

```
$ ./gcc_printf.out
Caracteres : a A 0 0
Entiers : 2007 650000
Affichage avec espaces : |      2007|
Affichage avec zeros : |0000002007|
Differentes bases : 100 64 144 0x64 0144
Reels : |3.14| |+6.3052e+03| |3.141593E+00|
Largeur en argument : |   10|
Debian GNU/Linux
```

9.3. EXEMPLE AVEC SDCC SUR CIBLE MSC1210

Par défaut, le support des nombres réels est désactivé dans la chaîne de développement `SDCC`. Il est nécessaire de passer par une recompilation manuelle des outils de compilation pour obtenir le résultat ci-dessous. Voir Section 13.3, « C sur MSC1210 : SDCC ».

Exemple 12. Sorties formatées avec `printf` sur un microcontrôleur MSC1210 avec SDCC

```
01: #define YEAR 2007
02:
03: void putchar(char c) {
04:    ser_putc(c);
05: }
06:
07: int main() {
08:
09:    autobaud();
10:    EA = 1;
11:
12:    printf ("Caracteres : %c %c %c %c\n\r", 'a', 6
13:    printf ("Entiers : %d %ld\n\r", YEAR, 650000);
14:    printf ("Affichage avec espaces : |%10d|\n\r",
15:    printf ("Affichage avec zeros : |%010d|\n\r",
16:    printf ("Differentes bases : %d %x %o %#x %#o
17:    printf ("Reels : |%f| |%+.4f| |%e|\n\r", M_PI,
18:    printf ("Largeur en argument : |%*d|\n\r", 5,
19:    printf ("%s\n\r", "Debian GNU/Linux");
20:    return 0;
21: }
```

Voici un exemple d'exécution du programme sdcc_printf.c :

```
MSC1210 Ver:000305F10
>M0000 ok
>M8000 ok
>E
>T
>
>Caracteres : a A 0 0
Entiers : 2007 650000
Affichage avec espaces : |      2007|
Affichage avec zeros : |0000002007|
Differentes bases : 100 64 144 #x #o
Reels : |3.141593| |+6305.1762| |E|
Largeur en argument : |*d|
Debian GNU/Linux
```

La copie d'écran ci-dessus montre que toutes les options de formatage des sorties ne sont pas supportées par la bibliothèque standard implantée dans la chaîne de développement `SDCC`. C'est une situation classique, sachant qu'un système spécialisé n'est pas fait pour réaliser une interface utilisateur.

9.4. UTILISATION DES CARACTÈRES SPÉCIAUX

Tableau 9. Caractères spéciaux

Séquence d'échappement	signification
%%	affichage du caractère '%'
\0	caractère *null* ; valeur 0, délimiteur de fin de chaîne de caractères
\a	alerte ; *beep* système
\b	*backspace* ; déplacement du curseur d'un caractère en arrière
\f	*form feed* ; saut de page
\n	*new line* ; saut de ligne
\r	*carriage return* ; retour chariot
\t	tabulation horizontale
\v	tabulation verticale
\\	affichage du caractère '\'
\'	affichage du caractère '''
\"	affichage du caractère '"'
\Onn	affichage de la valeur nn en octal
\Xnn	affichage de la valeur nn en hexadécimal

Précédent
8. Entrées et sorties standard : fonctions getchar & putchar
Sommaire
Suivant
10. Gestion d'un afficheur à cristaux liquides (LCD)

- Contact e-mail : philippe.latu (at) linux-france.org
- Page publiée le : jeudi 14 juin 2007, 14:34
- Tous les documents sont téléchargeables aux formats : (Postscript|PDF)
- Notice légale : Licence de Documentation Libre GNU (GNU Free Documentation License)
- Changelog projet
- (Vim Powered)| (Created with DocBook)| (Valid XHTML 1.0!)| (Valid CSS!)