Les secrets de "printf"

Professeur Don Colton

Université Brigham Young d'Hawaï

printf est la fonction du langage C pour effectuer une impression formatée. La même fonction est également disponible en PERL. Cet article explique comment fonctionne printf et comment concevoir la spécification de formatage appropriée pour chaque occasion.

1. Origines

Au début, les programmeurs informatiques écrivaient leurs propres sous-programmes pour lire et imprimer des nombres. Ce n'est pas très difficile, en fait. Allouez simplement un tableau de caractères pour contenir le résultat, divisez le nombre par dix, conservez le reste, ajoutez-y x30 et stockez-le à la fin du tableau. Répétez le processus jusqu'à ce que tous les chiffres soient trouvés. Puis imprimez-le.
Trop facile, non ?

Mais même si c'était facile (pour Einstein), cela demandait quand même un certain effort. Et qu'en est-il de la vérification des erreurs et des nombres négatifsÿ? Ainsi, les programmeurs informatiques ont produit des bibliothèques de fonctions préenregistrées. Et c'était bon. Finalement, les plus populaires de ces fonctions ont été canonisées en tant qu'appartenance aux bibliothèques "standard". L'impression numérique était assez populaire pour mériter cet honneur sacré.

Cela signifiait que les programmeurs n'avaient pas à réinventer encore et encore le sous-programme d'impression des nombres. Cela signifiait également que les options préférées de tout le monde essayaient de faire partie de la norme.

Ainsi est né printf.

2 Impression simple

Dans le cas le plus simple, printf prend un argument : une chaîne de caractères à imprimer. Cette chaîne est composée de caractères, dont chacun est imprimé exactement tel qu'il apparaît. Donc printf("xyz"); imprimerait simplement un x, puis un ay et enfin un z. Ce n'est pas exactement une impression "formatée", mais c'est toujours la base de ce que fait printf.

2.1 Caractères naturellement spéciaux

Pour identifier le début de la chaîne, nous mettons un guillemet double ("") au début. Pour identifier la fin de la chaîne, nous mettons un autre guillemet double à la fin. Mais que se passe-t-il si nous voulons réellement imprimer un guillemet doubleÿ?
Nous ne pouvons pas exactement mettre un guillemet double au milieu de la chaîne car il serait confondu avec le marqueur de fin de chaîne. Les guillemets doubles sont un caractère spécial. Les règles normales d'impression de ce que vous voyez ne s'appliquent pas.

Différentes langues adoptent différentes approches de ce problème. Certains exigent que le caractère spécial soit saisi deux fois. C utilise une barre oblique inverse (virgule, \) comme caractère d'échappement pour modifier la signification du caractère suivant. Ainsi, pour imprimer un guillemet double, vous tapez un guillemet double antislash. Pour imprimer une barre oblique inverse, vous devez l'échapper en tapant une autre barre oblique inverse devant celle-ci. La première barre oblique inversée signifie "donne au caractère suivant sa signification alternative". La deuxième barre oblique inverse a une autre signification de "imprimer une barre oblique inverse".

Sans barre oblique inverse, les caractères spéciaux ont une signification spéciale naturelle. Avec une barre oblique inverse, ils s'impriment tels qu'ils apparaissent. Voici une liste partielle.

\	échappe le caractère suivant \\ imprime une barre oblique inverse au début ou à la fin de la chaîne \" imprime un guillemet double au début ou à la fin d'un caractère constant \' imprime un guillemet simple % commence une spécification de format \% imprime un signe de pourcentage

2.2 Caractères spéciaux en alternance

D'un autre côté, nous avons des caractères qui s'impriment normalement comme prévu, mais lorsque vous ajoutez une barre oblique inverse, ils deviennent spéciaux. Un exemple est le caractère de nouvelle ligne. Pour imprimer un n, nous tapons simplement un n. Pour imprimer une nouvelle ligne, nous tapons un \n, donc in-

invoquant le sens alternatif de n, qui est nouvelle ligne.
Voici une liste partielle.

\a alerte sonore (cloche) \b retour arrière \f saut de page \n retour à la ligne (saut de ligne) \r retour chariot \t tabulation \v tabulation verticale	

3 Spécifications des formats

La vraie puissance de printf est lorsque nous imprimons le contenu des variables. Prenons le spécificateur de format %d par exemple. Cela imprime un numéro. Ainsi, un numéro doit être fourni pour l'impression. Cela se fait en ajoutant un autre argument à l'instruction printf, comme illustré ici.

```
int âgeÿ;
âge = 25ÿ;
printf( "J'ai %d ans\n", age );
```

Dans cet exemple, printf a deux arguments. Le premier est une chaîneÿ: "J'ai %d ans\n". Le second est un entier, l'âge.

3.1 La liste des arguments

Lorsque printf traite ses arguments, il commence à imprimer les caractères qu'il trouve dans le premier argument, un par un. Lorsqu'il trouve un pourcentage, il sait qu'il a une spécification de format. Il passe à l'argument suivant et utilise sa valeur, l'affichant selon cette spécification de format. Il revient ensuite à l'impression d'un caractère à la fois (à partir du premier argument).

Vous pouvez inclure plusieurs spécifications de format dans la chaîne printf. Dans ce cas, la première spécification de format va avec le premier argument supplémentaire, la seconde va avec le second, et ainsi de suite. Voici un exemple:

```
entier x = 5, y = 10ÿ;
printf( "x est %d et y est %d\n", x, y );
```

3,2 %

Chaque spécification de format commence par un signe de pourcentage et se termine par une lettre. Les lettres sont choisies pour avoir une signification mnémotechnique. Voici une liste partielle:

%c imprime un seul caractère %d imprime un nombre décimal (base 10) %e imprime un nombre exponentiel à virgule flottante %f imprime un nombre à virgule flottante %g imprime un nombre à virgule flottante au format général %i imprime un entier dans base 10 %o imprime un nombre en octal (base 8) %s imprime une chaîne de caractères %u imprime un nombre décimal non signé (base 10) %x imprime un nombre en hexadécimal (base 16) %% imprime un signe de pourcentage (\ % fonctionne aussi)

Pour imprimer un nombre de manière simple, le spécificateur de format est simplement %d. Voici quelques exemples de cas et de résultats.

printf	produit 0
("%d",0)	
("%d",-7)	-7
("%d",1560133635)	1560133635
("%d",-2035065302)	-2035065302

Notez que de manière simple, %d, il n'y a pas de taille prédéfinie pour le résultat. printf prend simplement autant d'espace qu'il en a besoin.

3.3 L'option de largeur

Comme je l'ai mentionné ci-dessus, le simple fait d'imprimer des chiffres ne suffisait pas. Il y avait des options spéciales qui étaient souhaitées. Le plus important était probablement l'option de largeur. En disant %5d, le nombre était garanti pour occuper cinq espaces (plus si nécessaire, jamais moins).
Cela était très utile pour imprimer des tableaux car les petits et les grands nombres occupaient le même espace. Presque toutes les impressions étaient à espacement fixe à cette époque, ce qui signifie que aw et an i occupaient tous les deux la même quantité d'espace. Ceci est encore courant dans les éditeurs de texte utilisés par les programmeurs.
Pour imprimer un nombre avec une certaine largeur (minimale), disons 5 espaces de large, le spécificateur de format est %5d. Voici quelques exemples de cas et de résultats. (Nous utiliserons le symbole pour indiquer explicitement un espace.)

printf	produit 0
("%5d",0)	
("%5d",-7)	-7
("%5d",1560133635)	1560133635
("%5d",-2035065302)	-2035065302

Notez que pour les nombres plus courts, le résultat est complété par des espaces de tête. Pendant trop longtemps

nombres il n'y a pas de remplissage, et le nombre complet est imprimé.

En utilisation normale, on élargirait le champ assez pour le plus grand nombre que l'on puisse attendre. Si vos chiffres sont généralement un, deux ou trois chiffres, alors %3d est probablement suffisant. Dans une utilisation normale, on pourrait finir par imprimer un numéro c'est trop grand pour le terrain. printf prend la décision d'imprimer entièrement ces nombres, même s'ils prendre trop de place. C'est parce qu'il vaut mieux imprimer la bonne réponse et avoir l'air moche que d'imprimer la mauvaise réponse et être jolie.

3.4 Remplir l'espace supplémentaire

Lors de l'impression d'un petit nombre comme 27 dans un champ %5d, la question est alors devenue où mettre le 27 et quoi mettre dans les trois autres emplacements. Il pourrait être imprimé dans les deux premiers espaces, les deux derniers espaces, ou peut-être les deux espaces du milieu (si cela peut être déterminé). Les espaces vides pourraient être remplis avec les un caractère vide, ou peut-être des étoiles (***27 ou 27*** ou **27*), ou signes dollar ($$$27), ou signes égal (===27), ou des zéros non significatifs (comme 00027).

Ces caractères supplémentaires sont souvent appelés caractères de «ÿprotection contre les chèquesÿ» car ils sont destinés à empêcher les malfaiteurs de modifier le montant en dollars sur un chèque imprimé. Il est relativement facile de changer un espace dans autre chose. Il est plus difficile de changer une étoile, un signe dollar ou un signe égal.

printf fournit un remplissage d'espace (gauche ou droite) et zéro remplir (gauche uniquement). Si vous souhaitez une protection contre les chèques ou un centrage, vous devez prendre d'autres dispositions. Mais même sans protection contre les chèques ni centrage de l'impression a toujours une collection impressionnante (et déconcertante) d'options.

3.5 L'option Justifier

L'utilisation de nombres printf peut être justifiée à gauche (imprimé dans la partie gauche du champ) ou justifié à droite (imprimé dans la partie droite du champ). La manière la plus naturelle imprimer des nombres semble être justifié à droite avec espaces de tête. C'est ce que %5d signifie : imprimer un nombre en base 10 dans un champ de largeur 5, avec le nombre aligné à droite et rempli d'espaces.

Pour aligner le nombre à gauche, un signe moins est ajouté au spécificateur de format. Pour imprimer un numéro 5 espaces larges et justifiés à gauche (alignés à gauche), le spécificateur de format est %-5d. Voici quelques exemples de cas et résultats.

printf	produit
("%-5d",0)	0
("%-5d",-7)	-7
("%-5d",1560133635) 1560133635	
("%-5j",-2035065302) -2035065302	

Comme précédemment, pour des nombres plus courts, le résultat est rempli d'espaces. Pour les numéros plus longs n'y a pas de remplissage et le nombre n'est pas raccourci.

3.6 L'option de remplissage zéro

Pour que les choses s'alignent bien et jolies, il est courant pour écrire une date en utilisant des zéros non significatifs. Nous pouvons écrire 5 mai 2003 aux États-Unis en tant que 05/05/2003. On pourrait aussi écrivez-le comme 2003.05.05. Notez que dans les deux cas, le les zéros non significatifs ne changent pas la signification. Ils ont juste faites-le bien aligner dans les listes.

Lorsqu'un nombre est rempli de zéros, les zéros vont toujours devant, et le nombre résultant est à la fois gauche et justifié à droite. Dans ce cas, le signe moins n'a pas effet. Pour imprimer un nombre rempli de zéros de 5 cases de large le spécificateur de format est %05d. Voici quelques exemples cas et résultats.

printf	produit
("%05d",0)	00000
("%05d",-7)	-0007
("%05d",1560133635) 1560133635	
("%05d",-2035065302) -2035065302	

Les nombres plus courts sont complétés par des zéros en tête. Les nombres plus longs sont inchangés.

3.7 Amusez-vous avec les signes plus

Les nombres négatifs s'impriment toujours avec un signe moins. Les nombres positifs et zéro ne s'impriment généralement pas avec un signe, mais vous pouvez en demander un. Un plus (+) dans le le spécificateur de format fait cette demande.

Pour imprimer un numéro signé 5 cases de large le format le spécificateur est %+5d. Voici quelques exemples de cas et résultats.

printf ("%	produit
+5d",0) +0	
("%+5j",-7) -7	
("%+5j",1560133635) +1560133635	
("%+5j",-2035065302) -2035065302	

Notez que zéro est traité comme un nombre positif. Les nombres plus courts sont remplis. Les nombres plus longs sont inchangé.

Plus et moins ne sont pas liés. Les deux peuvent apparaître dans un spécificateur de format.

3.8 Le signe plus invisible

Celui-ci est un peu bizarre. C'est un plus invisible
signe. Au lieu d'imprimer un plus sur les nombres positifs
(et zéro), on imprime un espace où irait le signe.
Cela peut être utile pour imprimer des nombres justifiés à gauche
où vous voulez que les signes moins se démarquent vraiment.
Remarquez ces deux alternatives.

printf ("%	produit
+-5d",0) ("%	+0
+-5d",-7) ("%	-7
+-5d",1560133635)	+1560133635
("%+-5j",-2035065302)	-2035065302

printf ("%	produit
-5d",0) ("%	0
-5d",-7) ("%	-7
-5d",1560133635) ("%	1560133635
-5d",-2035065302)	-2035065302

Rappelez-vous d'en haut que le spécificateur de format
%-5d nous obtenons les résultats suivants (affichés à nouveau pour
comparaison plus facile).

printf	produit
("%-5d",0) 0 ("%-5d",-7)	-7
("%-5d",1560133635)	1560133635
("%-5j",-2035065302)	-2035065302

Notez que les signes plus disparaissent, mais le signe
prend toujours de la place devant le numéro.

Notez également que nous pouvons combiner plusieurs options
dans le même spécificateur de format. Dans ce cas, nous avons
combiné les options plus, moins et cinq, ou espace,
moins, et cinq, ou juste moins et cinq.

3.9 Plus, Espace et Zéro

Voici un autre exemple de combinaison de plusieurs options
en même temps.

En utilisant le spécificateur de format % 05d ou %0 5d, nous obtenons
les résultats suivants.

printf ("%	produit
05d",0) ("%	0000
05d",-7) ("%	-0007
05d",1560133635) ("%	1560133635
05d",-2035065302)	-2035065302

En utilisant le spécificateur de format %+05d ou %0+5d, nous obtenons
les résultats suivants.

printf ("%	produit
+05d",0) ("%	+0000
+05d",-7) ("%	-0007
+05d",1560133635)	+1560133635
("%+05d",-2035065302)	-2035065302

Quand on combine plus et espace en même temps
temps, l'espace fait place à une enseigne et
le plus l'utilise. C'est comme si l'espace était
même pas précisé. Le plus a priorité sur le
espace.

3.10 Résumé

Les options sont également appelées "drapeaux" et entre elles
elles peuvent apparaître dans n'importe quel ordre. Voici une partie
liste.

effet drapeau	
aucun	s'imprime normalement (justifier à droite, remplir l'espace)
-	justifier à gauche
0	remplissage par zéro en tête
+	imprimer plus sur les nombres positifs
	signe plus invisible

Après les options, le cas échéant, la largeur de champ minimale
peut être spécifié.

4 chaînes d'impression

L'option %s nous permet d'imprimer une chaîne à l'intérieur d'un
corde. Voici un exemple.

```
caractère * noteÿ;
if ( année == 11 ) grade = "junior"ÿ;
printf ( "%s est un %s\n", "Fred", note );
```

L'indicateur de justification à gauche s'applique aux chaînes, mais de
bien sûr le zéro rempli, le signe plus et le signe plus invisible
sont vides de sens.

printf	produit
("%5s","")	
("%5s","a")	un
("%5s","ab")	un B
("%5s","abcdefg")	abcdefg

printf	produit
("%-5s","")	
("%-5s","a")	un
("%-5s","ab")	un B
("%-5s","abcdefg")	abcdefg

5 virgule flottante

Les nombres à virgule flottante sont ceux comme 3.1415 qui
avoir un point décimal quelque part à l'intérieur. C'est dans
contrairement aux nombres entiers ordinaires comme 27 qui n'ont pas
virgule.

Tous les mêmes drapeaux et règles s'appliquent pour le flottement
nombres de points comme pour les nombres entiers, mais nous en avons quelques-uns
de nouvelles possibilités. Le plus important est un moyen de
spécifier le nombre de chiffres apparaissant après la virgule décimale.
Ce nombre est appelé la précision du nombre.

Pour le commerce ordinaire, les prix sont souvent mentionnés
en dollars entiers ou en dollars et cents (zéro
ou deux chiffres de précision). Pour l'essence, les prix sont
mentionnés en dollars, cents et dixièmes de cent
(trois chiffres de précision). Voici quelques exemples d'impression
de ces types de nombres. Laisser
e=2,718281828.

printf	produit
("%.0f",e)	3
("%.0f.",e)	3.
("%.1f",e)	2.7
("%.2f",e)	2,72
("%.6f",e)	2.718282
("%f",e)	2.718282
("%.7f",e)	2.7182818

Notez que si un point et un nombre sont spécifiés, le
nombre (la précision) indique combien de places
doit être affiché après la virgule décimale.

Notez que si aucun point et aucune précision ne sont spécifiés pour
%f, la valeur par défaut est %.6f (six chiffres après la virgule
indiquer).

Notez que si une précision de zéro est spécifiée, le
la virgule décimale disparaît également. Si tu veux le récupérer,
vous devez le lister séparément (après le spécificateur de format
%f).

Nous pouvons spécifier à la fois une largeur et une précision à
le même temps. Remarquez surtout que 5.2 signifie
une largeur totale de cinq, avec deux chiffres après la virgule. Il est
très courant et naturel de le penser
signifie cinq chiffres avant la virgule et
deux chiffres après, mais ce n'est pas correct. Être
prudent.

printf	produit
("%5.0f",e)	3
("%5.0f.",e)	3.
("%5.1f",e)	2.7
("%5.2f",e)	2,72
("%5.7f",e)	2.7182818

Nous pouvons également combiner la précision avec les drapeaux que nous
appris auparavant, pour spécifier la justification à gauche, menant
zéros, signes plus, etc.

printf	produit
("%5.1f",e)	2.7
("%-5.1f",e)	2.7
("%+5.1f",e)	+2,7
("%+-5.1f",e)	+2.7
("%05.1f" ,e)	002.7
("%+05.1f",e)	+02.7
("% 05.1f",e)	02.7
("%- 5.1f",e)	2.7

6 Concevoir les spécifications parfaites

Si vous concevez un spécificateur de format printf, le
la première étape est de décider quel genre de chose vous êtes
impression. S'il s'agit d'un entier, d'un flottant, d'une chaîne ou d'un
caractère, vous ferez des choix différents quant à
format de base à utiliser.

La deuxième grande question est de savoir quelle est l'étendue de votre champ
devrait être. Habituellement, ce sera la taille du plus grand
nombre que vous vous attendez à imprimer dans des circonstances
normales. Parfois, cela est contrôlé par le
quantité d'espace qui est fourni sur un pré-imprimé
forme (comme un chèque ou une facture).

Décidez ce que vous aimeriez imprimer dans diverses
circonstances. Dans cet article, nous avons souvent illustré les
résultats en utilisant un petit nombre positif, un petit nombre négatif,
un nombre surdimensionné.
nombre positif et un nombre négatif surdimensionné.
Vous devez inclure ces options ainsi que de grandes (mais
nombres non surdimensionnés). Concevez votre format pour le
le plus grand nombre auquel vous vous attendriez normalement
voir.

7 conseils pour le test

Le test printf comprend une variété de problèmes d'appariement.
Ils sont conçus pour être délicats, et les étudiants
les commentaires indiquent que si quoi que ce soit, ils sont plus
délicat que prévu.

Vous pouvez utiliser le processus d'élimination pour faire
ce test très rapide et précis. Pendant que vous regardez un
caractéristique commune dans la ligne de réponse, vous pouvez exclure
toutes ces déclarations printf qui n'ont pas ça
fonctionnalité. Très rapidement, vous pouvez réduire vos options
à un ou deux.

7.1 Fonctionnalités simples

Il est facile de voir si les nombres courts ont un début
des zéros. Si c'est le cas, il doit y avoir un zéro dans le formatage
spécification.

Il est facile de voir si les nombres positifs ont plus
panneaux. Si oui, il doit y avoir un plus dans le formatage
spécification.

7.2 Avant, entre et derrière

La prochaine chose à surveiller est l'avant, entre,
et derrière le numéro qui est imprimé. Dans une spécification de
tapis comme x%5dz, il y a un x avant
le nombre et az derrière le nombre. Le x et
z ne font pas partie de la spécification du format, mais ils
font partie du résultat imprimé. Tout le reste que
impressions est "entre".

Pour décider s'il y a quelque chose avant ou après un nombre,
regardez le nombre négatif surdimensionné. Tous les espaces
avant sont sûrement avant la spécification de format. Tous les
espaces derrière sont sûrement
derrière la spécification de format. Voici un exemple.

Si -2035065302 imprime comme -2035065302 , tu peux
assurez-vous que la chaîne printf est %... avec deux ,
espaces avant la spécification de formatage et un
espace derrière. C'est parce que toutes les positions d'impression
entre (le % et ce qui va avec) sont utilisés
par le nombre surdimensionné.

Une fois que vous avez déterminé ce qui est avant et après,
vous pouvez utiliser ces informations pour comparer avec
les choix correspondants. Cela vous donnera souvent la
répondre directement.

7.3 Le signe plus invisible

Comparez le nombre négatif surdimensionné au nombre positif
surdimensionné. Si le nombre positif a
un espace supplémentaire devant, c'est un signe plus invisible.
S'il n'y a pas d'espace supplémentaire, il n'y a pas de plus invisible
signe.

7.4 Justification à gauche

Soustrayez l'avant et l'arrière. Regardez ce qui reste.

Regardez le petit nombre négatif. Où sont les
espaces supplémentaires imprimésÿ? Si les sont devant, le nombre
est justifié à droite. S'ils sont à l'arrière, le nombre est
justifié à gauche. S'ils sont devant et derrière, vous
fait quelque chose de mal.

8Conclusion

La fonction printf est un outil puissant pour imprimer des nombres
et d'autres éléments stockés dans des variables.
Avec ce pouvoir, il y a une certaine complexité. Prise d'un coup, la
complexité rend
printf semble presque impossible à comprendre. Mais
la complexité peut être facilement dépliée en simple
caractéristiques, y compris la largeur, la précision, la signalisation,
la justification et le remplissage. En reconnaissant et en comprenant
ces fonctionnalités, printf deviendra un outil utile et
serviteur amical dans vos efforts d'impression.