Chaines de caractères en C, quelques fonctions

S2 - dept info

10 avril 2013

1 Documentation (extraits)

Fonctions définies dans `string`

1.1 `strlen` - Calculer la longueur d'une chaîne de caractères

```
size_t strlen(const char *s);
```

La fonction `strlen()` calcule la longueur de la chaîne de caractères `s`, sans compter le caractère nul « \0 » final.

La fonction `strlen()` renvoie le nombre de caractères dans la chaîne `s`

1.2 `strcpy, strncpy` - Copier une chaîne

```
char *strcpy(char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t n);
```

La fonction `strcpy()` copie la chaîne pointée par `src`, y compris le caractère nul (« \0 ») final dans la chaîne pointée par `dest`. Les deux chaînes ne doivent pas se chevaucher. La chaîne `dest` doit être assez grande pour accueillir la copie.

La fonction `strncpy()` est identique, sauf qu'au plus `n` octets de src sont copiés. Attention : s'il n'y a pas de caractère nul dans les `n` premiers octets de `src`, la chaîne résultante dans `dest` ne disposera pas de caractère nul final.

1.3 `strcmp, strncmp` - Comparaison de deux chaînes

```
int strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t n);
```

La fonction `strcmp()` compare les deux chaînes `s1` et `s2`. Elle renvoie un entier négatif, nul, ou positif, si `s1` est respectivement inférieure, égale ou supérieure à `s2`.

La fonction `strncmp()` est identique sauf qu'elle ne compare que les `n` (au plus) premiers caractères de `s1` et `s2`.

Les fonctions `strcmp()` et `strncmp()` renvoient un entier inférieur, égal ou supérieur à zéro si `s1` (ou ses `n` premiers octets) est respectivement inférieure, égale ou supérieure à `s2`.

1.4 `strcat, strncat` - Concaténer deux chaînes

```
char *strcat(char *dest, const char *src);
char *strncat(char *dest, const char *src, size_t n);
```

La fonction `strcat()` ajoute la chaîne `src` à la fin de la chaîne `dest` en écrasant le caractère nul (« \0 ») à la fin de `dest`, puis en ajoutant un nouveau caractère nul final. Les chaînes ne doivent pas se chevaucher, et la chaîne `dest` doit être assez grande pour accueillir le résultat.

La fonction `strncat()` est similaire, à la différence que :

– elle ne prend en compte que les `n` premiers caractères de `src`.

– `src` n'a pas à se terminer par un caractère nul si elle contient `n` caractères ou plus.
Comme pour `strcat()`, la chaîne résultante dans `dest` est toujours terminée par un caractère nul.
Les fonctions `strcat()` et `strncat()` renvoient un pointeur sur la chaîne résultat `dest`.

2 Exemples d'implémentation de `strcpy()`

2.1 Tableau et indice, naïf

```
char *  my_strcpy_1 (char *dest, const char *src) {
  int i;
  for (i=0; src[i] != '\0'; i++)
     dest[i] = src[i];
  dest[i] = '\0';
  return dest;
}
```

2.2 Tableau et indice, plus compact

Combinaison transfert + comparaison. La boucle s'arrête quand l'affectation a transféré le caractère nul.

```
char * my_strcpy_2 (char *dest, const char *src) {
  int i = 0;
  while ((dest[i] = src[i]) != '\0')
     i ++;
  return dest;
}
```

2.3 Tableau et indice, encore plus compact

Utilisation de la post-incrémentation de l'indice (après transfert et comparaison).

```
char * my_strcpy_3 (char *dest, const char *src) {
  int i = 0;
  while ((dest[i++] = src[i])) ;
  return dest;
}
```

2.4 Avec pointeurs

Usage de deux pointeurs pour parcourir les chaines. Post-incrémentation après transfert et comparaison.

```
char * my_strcpy_4 (char *dest, const char *src) {
  int *d = dest;
  int *s = src;
  while ((*d++ = *s++)) ;
  return dest;
}
```

3 Exercices

Re-programmez les fonctions classiques `strlen`, `strcmp`, ... Comparez le code source assembleur généré par chaque version. On l'obtient par :

```
gcc -std=C99 -O9 -S my_function_3.c
```