utiles pour gérer les chaînes. La liste complète est visible avec man string. strcat peut concaténer des chaînes :

```
char fileName[100]="monHôte", suffix[10]="-data.txt"
strcat(fileName, suffix); /* Ajoute le suffixe, résultat dans fileName. */
```

Le nom d'hôte peut être lu avec fgets plutôt qu'avec fgetc :

```
fgets(fileName,100,cmdPtr); /* Lit jusqu'à EOF, fin de ligne ou 99 caractères. */
```

où 100 est la taille du tableau de caractères fileName. Enfin, la fonction gethostname de unistd.h permet aussi de le lire :

```
#include <string.h>
#include <stdio.h>
#include <unistd.h>

int main() {
  char fileName[100], suffix[10]="-data.txt";
  gethostname(fileName,100);
  strcat(fileName,suffix); /* Ajoute le suffixe au nom de fichier */
  printf("%s\n",fileName);
  return 0;
}
```

Structures

Elles ont été survolées dans le numéro 6 pour lire les infos système. Elles utilisent un bloc continu de mémoire comme les blocs en FORTRAN. Leur syntaxe est similaire aux classes C++ avec les données-membres publiques. Une struct est définie par un nom et un bloc de définition de variables qui peuvent aussi être des struct. Une structure simple :

```
  struct dataPoint {
    unsigned int timeSeconds;
    float value;
  };
```

int timeSeconds est défini en premier en mémoire, puis float value. La taille du struct en mémoire est la somme des tailles des variables. Elle doit être définie avant d'être utilisée et se trouve en principe dans le fichier d'entête, mais peut être écrite dans le même fichier avant son utilisation. Pour tester cette struct :

```
int main() {
  /* Déclare une variable de type "struct dataPoint". */
  struct dataPoints;
  /* Assigne à la structure s des valeurs initiales. */
  s.timeSeconds = 60;
  s.value = 100.0;
  /* Affiche les emplacements en mémoire et les tailles */
  printf("sizeof(s) = %ld\n", sizeof(s));
  printf("&(s.timeSeconds) = %p\n", &(s.timeSeconds));
  printf("&(s.value) = %p\n", &(s.value));
  printf("sizeof(unsigned int) = %ld\n", sizeof(unsigned int));
  printf("sizeof(float) = %ld\n", sizeof(float));
  return 0;
}
```

où le programme définit des valeurs et affiche leurs adresses pour illustrer la structure en mémoire.

Quand elles sont passées en paramètres, ce sont par défaut des copies locales à la fonction qui sont créées. Au retour de la fonction appelante, la valeur de la structure est donc inchangée. Le comportement est le même que pour une variable classique passée à une fonction :

Suite sur la page suivante...