La fonction `system` évalue la chaîne passée en paramètre comme si elle avait été tapée en ligne de commande. La sortie standard de la commande n'est pas capturée par le programme mais elle est par contre envoyée sur l'écran.

La sortie standard d'une commande système ou d'un programme peut être capturée grâce à un tube (pipe en anglais). Les tubes ont la même syntaxe que les fonctions sur les fichiers traditionnels, autorisant lecture, écriture et connexions bidirectionnelles. Par exemple, le contenu du répertoire courant peut être lu dans un programme en utilisant :

```
#include <stdio.h>
int main() {
  int c;
  FILE *ptr = 0; /* Crée un pointeur FILE nul */
  ptr = popen("ls ./", "r"); /* Liste le contenu du dossier et se met en écoute */
  if(!ptr) return 1; /* Renvoie échec si la commande échoue. */
  while((c=fgetc(ptr)) != EOF) { /* Lit chaque caractère. */
    printf("%c",(char)c); /* Affiche les caractères. */
  }
  pclose(ptr); /* Ferme le tube */
  return 0; /* Renvoie succès au système d'exploitation. */
}
```

Dans cet exemple, chaque nom de fichier retourné est disponible dans le programme.

Toute commande qui peut être saisie en ligne de commande peut être exécutée avec `system` ou `popen`. Au lieu de simplement appeler des commandes shell de base, il est possible d'en utiliser pour tracer des données avec gnuplot :

```
#include <stdlib.h>
#include <stdio.h>
int main(int argc, char *argv[]) {
  int x_min = 0, x_max = 4; /* Fixe l'étendue du tracé. */
  char commandStr[100], systemCmd[200];
  if(argc < 2) {
    printf("Utilisation %s <fonction>\n", argv[0]); /* Un argument requis.*/
    return 1; /* Retourne une erreur. */
  }
  /* Construit la commande en deux étapes pour montrer ce qui se passe. */
  sprintf(commandStr, "plot [x=%d:%d] %s(x)", x_min, x_max, argv[1]);

  /* Lance la commande afin que gnuplot reste ouvert. */
  sprintf(systemCmd,"echo \"%s\" | gnuplot --persist",commandStr);
  system(systemCmd); /* Demande à gnuplot de faire le tracé. */
  return 0; /* Retourne succès au système d'exploitation. */
}
```

Avant d'essayer cet exemple, il faut installer gnuplot en tapant :

```
sudo apt-get gnuplot-x11
```

Ensuite, une fois le programme compilé, lancez : `./gplot sin`. Le paramètre `--persist` permet à la fenêtre de gnuplot de rester ouverte après la fin du programme. Davantage d'informations sur le programme gnuplot sont disponibles sur : http://www.gnuplot.info/

Surveiller un système LINUX

Il existe plusieurs fonctions utiles disponibles sous Linux, mais elles ne sont pas implémentées de la même façon sur les autres systèmes d'exploitation. Par exemple, l'état de la mémoire peut être obtenu grâce à `sysinfo :`

```
#include <stdio.h>
#include <sys/sysinfo.h>
int main() {
  struct sysinfo info; /* Crée une instance sysinfo pour stocker le résultat. */
  sysinfo(&info); /* Récupère les informations sur le système */
  printf("Mémoire utilisée = %d\n",info.totalram - info.freeram);
  return 0; /* Retourne succès au système d'exploitation. */
}
```