LINFO1252 – Systèmes informatiques

Travail Dirigé (TD) S5 : Appels systèmes

Objectifs de ce TD : Vous comprendrez dans quels cas les appels systèmes sont utilisés.

Compétences travaillées : utilisation d'outils.

Prérequis : Puisque vous allez utiliser des appels systèmes, vous aurez besoin d'un système Linux. Aucun support ne sera apporté pour WSL. Vous aurez également besoin des programmes test, disponibles sur Moodle.

Étape 1 : Écriture sur `stdout`

Dans un premier temps, vous allez vous familiariser avec des appels systèmes simples. Commencez par regarder le programme `1-printfs.c`. Ce programme utilise la fonction `printf` (voir manpage) pour afficher des chaînes de caractères formatées sur la sortie standard.

Quel appel système vous attendez vous à rencontrer lors de l'exécution de ce code ? Combien d'instances de cet appel sont nécessaires selon vous ? Compilez le code et utilisez `strace` (voir manpage) pour analyser les appels systèmes utilisés.

```
$ gcc -o 1-printfs 1-printfs.c
$ strace ./1-printfs
```

Pour le moment, ne vous souciez pas des appels systèmes qui permettent d'initialiser le processus, vous devriez être en mesure de comprendre leur utilité d'ici la fin du quadrimestre. Quel est l'appel effectivement utilisé pour afficher les caractères ? Combien d'appels observez-vous ? Est-ce le nombre auquel vous vous attendiez ? Qu'est-ce qui pourrait expliquer ce comportement ?

Étape 2 : Allocation de mémoire sur le *heap*

Le deuxième exercice vous permettra de comprendre plus en profondeur l'allocation de mémoire dynamique utilisant malloc, et quels appels systèmes sont utilisés dans ce cas.

Commencez par comprendre le fonctionnement du programme `2-malloc.c`. Combien d'allocations effectue-t-il, et de quelle taille (en octets) ?

Ensuite, compilez le programme et examinez ses appels systèmes :

```
$ gcc -o 2-malloc 2-malloc.c
$ strace ./2-malloc
```

Concentrez-vous uniquement sur les appels utilisés pour le corps du programme.

- Quel appel système retrouvez-vous majoritairement ?
- Comptez le nombre de ces appels, en utilisant un outil de ligne de commande.
- Inspectez les arguments de ces appels, et comparez avec les paramètres des opérations `malloc` et `free` du programme en C.

Étape 3 : Écriture de données dans un fichier

Le troisième exercice se concentre les appels systèmes permettant les interactions avec des fichiers.

Commencez par comprendre le fonctionnement du programme `3-files.c`. Quels sont les appels que vous vous attendez à rencontrer ? A ce stade, vous devriez être familier avec plusieurs des appels systèmes utilisés par ce programme. Essayez de déterminer à l'avance quels sont les arguments qui seront utilisés pour chaque appels système. Quels sont les différences par rapport aux utilisations de ces appels systèmes dans les exemples précédents ?

Compilez le programme et vérifiez si vos hypothèses sont correctes :

```
$ gcc -o 3-files 3-files.c
$ strace ./3-files
```

Est-ce que vous remarquez des appels pour lesquels les arguments diffèrent de ce à quoi vous vous attendiez ? Comment expliquez-vous ces différences ?

Étape 4 : Programme interactif

Finalement, le dernier exercice met en scène un programme interactif, nommé `4-command_line.c`, qui demande à l'utilisateur d'entrer une instruction sur la ligne de commande.

Comme pour les programmes précédents, commencez par inspecter le code source et comprendre ce que le programme doit faire. Pouvez-vous déterminer à l'avance quels appels systèmes seront utilisés, et quels seront leurs arguments ?

Ensuite, compilez-le, et exécutez-le avec `strace` :

```
$ gcc -o 4-command_line 4-command_line.c
$ strace ./4-command_line
```

Quelles différences remarquez-vous par rapport aux exemples précédents ?