INF2610
Noyau d’un système d’exploitation

Trousse de démarrage pour les exemples de codes

Requis

Afin de pouvoir exécuter et modifier les exemples de codes du cours, vous aurez besoin de:

- Un ordinateur Linux / Windows / MacOs
- Git (https://git-scm.com/)
- Docker (https://docs.docker.com/)

Vous devez d'abord installer ***Git*** et ***Docker*** ci-dessus avant les prochaines étapes.

Note: Si vous utilisez un ordinateur MacOs ou Linux, vous devez vous assurer d'avoir un accès superuser (pour exécuter les scripts avec la commande ***sudo****).*

Répertoire des exemples de codes

Pour télécharger le répertoire des exercices, vous pouvez vous rendre à :

https://github.com/vitonimal/ExercicesINF2610.git

et utiliser le bouton « Clone or download »

Le téléchargement peut se faire aussi à partir d'une fenêtre Terminal via la commande :

git clone https://github.com/vitonimal/ExercicesINF2610.git

➔ Un répertoire nommé ***ExercicesINF2610*** est alors crée dans le répertoire courant.

➔ ***ExercicesINF2610*** est composé d'un seul répertoire nommé ***ressources***

Répertoire des exemples

ressources : le répertoire dans lequel vous travaillerez. Il contient :

- **prepare.sh** : un script permettant de préparer Docker pour exécuter les codes (installation des librairies nécessaires).
- **run.sh** : un script permettant de lancer un code. Il prend comme premier argument le chemin vers le dossier contenant le code à exécuter **depuis le dossier codes/**. Les arguments suivants sont ceux à passer au programme à exécuter.
- **codes/** : un dossier regroupant les codes des différents chapitres. Chaque code est dans un dossier contenant le code source et un makefile qui sert à créer un exécutable ***a.out*** . Ce dernier sera exécuté par ***Docker***.

Hello World

Pour exécuter un exemple de code, il suffit de :

1. ouvrir une fenêtre Terminal et accéder via la commande ***cd*** au dossier ressources/;
2. exécuter ./prepare.sh (précédé de sudo pour les utilisateurs de Mac et Linux);
3. exécuter ./run.sh dummy (précédé de sudo pour les utilisateurs de Mac et Linux). Cela aura pour effet de compiler et d'exécuter le code se trouvant dans le dossier ***codes/dummy/;***
4. c'est tout!

Sentez-vous libre de modifier les exemples pour obtenir d'autres résultats, c'est une excellente façon d'explorer la matière!

Organisation des exemples de codes

Comme vous l'avez remarqué, le code affichant « ***hello world*** » était dans le dossier ***dummy*** qui se trouve à la racine du répertoire ***codes***. Les exemples du cours sont cependant organisés par chapitre.

Par exemple, les codes portant sur les processus et threads sont tous dans le dossier ***ProcessusEtThreads***.

Si vous voulez exécuter le code ***DeuxFils*** qui se trouve dans le répertoire ***ProcessusEtThreads***, il suffit de lancer la commande :

```
./run.sh ProcessusEtThreads/DeuxFils
```

Ajouter vos propres exemples

Il vous est possible de créer et exécuter vos propres codes en suivant la procédure suivante :

1. créer un dossier portant le nom de votre choix à l'intérieur du répertoire codes/;
2. créer vos fichiers sources à l'intérieur de ce même dossier;
3. ajouter un makefile (*boilerplate* à la page suivante) qui compile vos fichiers sources et crée un exécutable portant obligatoirement le nom **a.out;**
4. exécuter ./run.sh <NomDeVotreDossier> <ArgumentsDeVotreCodeAuBesoin>;
5. Voilà!

Boilerplate de Makefile

```
all:
    gcc -o a.out <fichier>.c
```

Note: Vous aurez à ajouter des flags et paramètres à ce makefile selon les librairies utilisées. Plusieurs exemples de makefiles plus élaborés sont fournis.

Arguments de programmes

Note: Pour les codes fournis, le chargement des variables d'environnement dans les arguments est déjà fait. Ce message s'adresse à toute personne désirant écrire son propre code.

Afin de pouvoir passer les arguments à un programme paramétré à exécuter « ***int main (int argc, char* argv[]); »,*** où ***argc*** est le compteur d'arguments et ***argv*** est le tableau des arguments), il faudrait communiquer à Docker les adresses de ***argc*** et ***argv***.

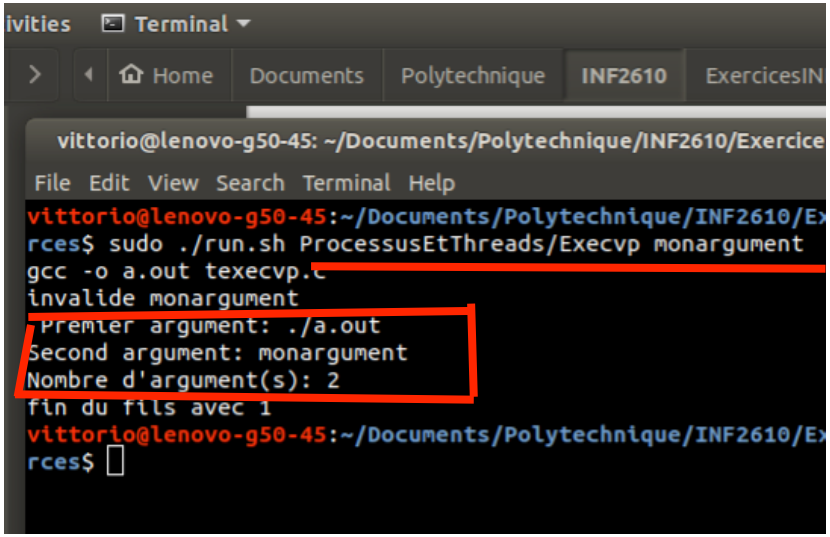
Pour ce faire, vous devez inclure, dans votre programme paramétré ,un appel à la fonction `bootEnvVars(int* pargc, char** pargv[])` en lui passant les adresses de ***argc*** et ***argv***.

Un exemple se trouve à la diapositive suivante. N'oubliez pas d'inclure le fichier env_config.c afin d'avoir accès à cette fonction.

Exemple de chargement des arguments

```
 GNU nano 2.9.3                          texecvp.c

// programme test de execvp : texecvp.c
#include "../../env_config/env_config.c"
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>
int main (int argc , char * argv[]) {
    bootEnvVars(&argc, &argv);
    printf("Premier argument: %s\n", argv[0]);
    printf("Second argument: %s\n", argv[1]);
    printf("Nombre d'argument(s): %d\n", argc);
    //...
```



Installation d’une librairie

Dans l’éventualité où vous avez écrit votre propre code, mais que celui-ci requiert une librairie qui ne se trouve pas sur l’image Docker, vous pouvez l’installer de la façon suivante :

1. ouvrez le fichier docker-base sous ressources/.docker;
2. ajoutez RUN apt-get install -y <librairie-requise>;
3. exécutez de nouveau ./prepare.sh;
4. Voilà!