FAUCHER Vinh

TD1 :

1) On commence par décoder les arguments donnés par la ligne de commande.

Puis on crée un tableau d'int que l'on remplit de nombres aléatoires entre 0 et 999.

Si la variable Verbose passe à TRUE, on affiche le tableau à trier.

Puis on trie le tableau en rangeant les int par ordre croissant et on calcule en même temps le temps mis pour ranger le tableau.

Si la variable Verbose vaut TRUE, on affiche le temps mis et le tableau trié.

Avec l'argument -h, on affiche les informations d'utilisation, c'est-à-dire la liste des options possibles pour l'argument du programme.

Avec l'argument -v, on passe la variable Verbose à TRUE, ce qui donne un affichage du tableau avant et après tri, ainsi que le temps d'exécution de l'algorithme.

Avec l'argument -s on rentre la taille du tableau que l'on souhaite.

2) Avec la commande ldd, on peut voir :

```
user@PNS-VirtualBox:~/Documents/ProgSys/td02$ ldd tri_bubble-basicExe.ex
e
        linux-vdso.so.1 (0x00007ffe14fbe000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f667422b000
)
        /lib64/ld-linux-x86-64.so.2 (0x00007f6674435000)
```

Donc des bibliothèques sont utilisées.

3) – Les résultats sont bien identiques, mais le 2eme va plus vite :

```
user@PNS-VirtualBox:~/Documents/ProgSys/td02$ make test
*** Début des tests des programmes générés
Lancement du programme: tri_bubble-basicExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 1084
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_bubble-staticExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 943
Sorted array:335 383 386 421 492 649 777 793 886 915

*** Fin des tests
```

- Le programme static n'utilise aucune bibliothèque

```
user@PNS-VirtualBox:~/Documents/ProgSys/td02$ ldd tri_bubble-staticExe.e
xe
        n'est pas un exécutable dynamique
```

- La version static est plus lourde, car elle doit posséder elle-même les bibliothèques nécessaires, contrairement à la basic qui va importer des bibliothèques déjà existantes.

4)

```
####
## Exercice 4:
STATIC_LIB=$(EXE:.exe=-staticLib.exe)

tri_%-staticLib.exe: main.o timer.o utils.o libTri_%-sta
    $(CC) -o $@ $^

libTri_%-staticLib.a: %.o unused.o
#   TODO: écrire la règle permettant de créer une bibilc
    ar -r $@ $^
    ranlib $@
```

- Les résultats sont bien identiques :

```
Lancement du programme: tri_bubble-basicExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 817
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_bubble-staticExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 742
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_bubble-staticLib.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 767
Sorted array:335 383 386 421 492 649 777 793 886 915
```

- Les tailles de basic et staticLib sont quasi identiques, en revanche le staticExe est beaucoup plus lourd.

```
-rwxrwxr-x 1 user user  25248 févr.  2 09:32 tri_bubble-basicExe.exe
-rwxrwxr-x 1 user user 884976 févr.  2 10:00 tri_bubble-staticExe.exe
-rwxrwxr-x 1 user user  24304 févr.  2 10:23 tri_bubble-staticLib.exe
```

- Avec la commande size on voit que la taille prise par les variables ainsi que les instructions sont bien plus grandes dans le programme statique. Il n'y a en revanche pas de différence dans ce cas entre le programme normal et celui avec la bibliothèque statique.

```
user@PNS-VirtualBox:~/Documents/ProgSys/td02$ size tri_bubble-basicExe.e
xe
   text    data     bss     dec     hex filename
   5157     684      12    5853    16dd tri_bubble-basicExe.exe
user@PNS-VirtualBox:~/Documents/ProgSys/td02$ size tri_bubble-staticExe.
exe
   text    data     bss     dec     hex filename
 766820   20964    6080  793864   c1d08 tri_bubble-staticExe.exe
user@PNS-VirtualBox:~/Documents/ProgSys/td02$ size tri_bubble-staticLib.
exe
   text    data     bss     dec     hex filename
   4709     684      12    5405    151d tri_bubble-staticLib.exe
```

- Dans le programme normal on peut voir la présence de foo et bar, ce qui n'est pas le cas pour celui avec la bibliothèque statique.

5)

```
####
## Exercice 5:
DYNAMIC_LIB=$(EXE:.exe=-dynamicLib.exe)

tri_%-dynamicLib.exe: main.o timer.o utils.o libTri_%-dy
    $(CC) -o $@ $^

libTri_%-dynamicLib.so: %.o unused.o
#   TODO: écrire la règle permettant de créer une biblic
    $(CC) -shared -Wl,-soname,$@ -o $@ $^
```

6) Tous les résultats sont bien identiques

```
user@PNS-VirtualBox:~/Documents/ProgSys/td02$ export LD_LIBRARY_PATH=.:$
LD_LIBRARY_PATH
```

7) L'exécutable a environ le même poids que les exécutables précédents hormis celui avec bibliothèque statique, on peut donc économise en faisant un exécutable "commun".

```
user@PNS-VirtualBox:~/Documents/ProgSys/td02$ size tri.exe
   text    data     bss     dec     hex filename
   5757     724      24    6505    1969 tri.exe
```

Il ne dépend d'aucune bibliothèque spécifique de tri, elles vont agir comme des "bibliothèques partagées", le programme n'a pas besoin de les posséder lui-même.

```
user@PNS-VirtualBox:~/Documents/ProgSys/td02$ ldd tri.exe
        linux-vdso.so.1 (0x00007ffea55df000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f85f181b
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f85f162900
        /lib64/ld-linux-x86-64.so.2 (0x00007f85f1839000)
```

8) Il nous suffirait d'écrire notre nouveau programme shell_sort.c par exemple et nous n'aurions pas besoin de recompiler notre tri.exe car il ne dépend d'aucune bibliothèque spécifique de tri.

10)pas de Java ?

```
user@PNS-VirtualBox:/usr/bin$ ldd python3
        linux-vdso.so.1 (0x00007ffc3c4fa000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f4bb7baa000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f4bb7b8
7000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f4bb7b81000)
        libutil.so.1 => /lib/x86_64-linux-gnu/libutil.so.1 (0x00007f4bb7b7c000)
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f4bb7a2d000)
        libexpat.so.1 => /lib/x86_64-linux-gnu/libexpat.so.1 (0x00007f4bb79ff000
)
        libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f4bb79e1000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f4bb7daf000)
```