TP02

1. Étudiez rapidement le code source qui vous est fourni pour en comprendre la structure. Donnez un court résumé de ce que vous avez compris de l'organisation de ce code source.

Chaque algorithme de tri est défini dans un fichier .c séparé. Chaque fichier redéfini la fonction sort pour appeler sa propre implémentation.

2. Lancez la commande make dans l'archive que vous avez récupérée. Examinez à l'aide de la commande ldd avec le programme `tri_bubble-basicExe.exe` généré pour savoir s'il utilise des bibliothèques ou non. Si oui, quelles sont les bibliothèques utilisées ?

- linux-vdso.so.1
- libc.so.6
- ld-linux-x86-64.so.2

3. Comment rendre votre programme complètement indépendant des bibliothèques qu'il utilise, même par défaut ?

```
STATIC_EXE=$(EXE:.exe=-staticExe.exe)

tri_%-staticExe.exe: main.o %.o timer.o utils.o unused.o
    $(CC) -static -o $@ $^
```

```
*** 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): 5718
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): 5407
Sorted array:335 383 386 421 492 649 777 793 886 915

*** Fin des tests
```

```
user@host:~$ ldd tri_bubble-staticExe.exe
        not a dynamic executable
```

```
user@host:~$ ll *.exe
-rwxr-xr-x 1 user user  24K  2 févr. 10:31 tri_bubble-basicExe.exe
-rwxr-xr-x 1 user user 791K  2 févr. 10:34 tri_bubble-staticExe.exe
```

4. Comment créer une bibliothèque statique qui sera incluse dans votre exécutable ?
   Décommentez les règles dans la section Exercice 4 du Makefile et complétez la pour obtenir une bibliothèque statique qui sera incluse à votre programme.

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

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

libTri_%-staticLib.a: %.o unused.o
    ar -r $@ $^
    ranlib $@
```

```
*** 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): 5770
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): 5464
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): 5654
Sorted array:335 383 386 421 492 649 777 793 886 915

*** Fin des tests
```

```
user@host:~$ ll *.exe
-rwxr-xr-x 1 user user  24K  2 févr. 10:31 tri_bubble-basicExe.exe
-rwxr-xr-x 1 user user 791K  2 févr. 10:34 tri_bubble-staticExe.exe
-rwxr-xr-x 1 user user  23K  2 févr. 10:50 tri_bubble-staticLib.exe
```

```
user@host:~$ size *.exe
   text    data     bss     dec     hex filename
   4845     668      12    5525    1595 tri_bubble-basicExe.exe
 691425   23132    6400  720957   b003d tri_bubble-staticExe.exe
   4413     668      12    5093    13e5 tri_bubble-staticLib.exe
```

```
user@host:~$ nm tri_bubble-basicExe.exe | grep foo
000000000000184d T foo
```

```
user@host:~$ nm tri_bubble-staticLib.exe | grep foo
```

5. Comment créer une bibliothèque partagée ?

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

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

libTri_%-dynamicLib.so: %.o unused.o
    $(CC) -shared -o $@ $^
```

6. Comme le Makefile est a été bien réalisé et maintenant que vous pouvez créer les exécutables statiques, les bibliothèques statiques et dynamiques et les exécutables qui les utilisent, nous allons modifier le Makefile pour utiliser d'autres algorithmes de tri. Supprimez le premier commentaire utilisé dans la variable EXE et relancez la compilation.

```
user@host:~$ LD_LIBRARY_PATH="$LD_LIBRARY_PATH":. 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): 5635
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_insertion-basicExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 4226
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_merge-basicExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 8047
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_quick-basicExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 5457
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): 5439
Sorted array:335 383 386 421 492 649 777 793 886 915
```

```
Lancement du programme: tri_insertion-staticExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 4133
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_merge-staticExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 10151
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_quick-staticExe.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 5181
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): 5557
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_insertion-staticLib.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 4184
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_merge-staticLib.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 7978
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_quick-staticLib.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 6791
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_bubble-dynamicLib.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 6901
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_insertion-dynamicLib.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
```

```
Time taken for sorting (nanoseconds): 5596
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_merge-dynamicLib.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 9297
Sorted array:335 383 386 421 492 649 777 793 886 915

Lancement du programme: tri_quick-dynamicLib.exe
Array to sort:383 886 777 915 793 335 386 492 649 421
Time taken for sorting (nanoseconds): 6834
Sorted array:335 383 386 421 492 649 777 793 886 915

*** Fin des tests
```

7. Pour éviter de générer autant de programmes qu'il n'y a d'algorithmes de tri, nous allons produire un seul et unique programme qui chargera dynamiquement et explicitement la bibliothèque de tri à utiliser (bibliothèque dynamique que nous avons construire lors de l'exercice précédent).

`main.c`

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
+#include <string.h>

#include "sort.h"
#include "utils.h"
#include "timer.h"
+#include "load_library.h"

...

static int Size_Array = SIZE;
static int Verbose = FALSE;
+static char library[512] = "";

...

int main(int argc, char *argv[])
{
    /* Décodage des arguments de la ligne de commande */
```

```
    Scan_Args(argc, argv);

+    load_library(library);

    ...
}

static void Scan_Args(int argc, char *argv[])
{
    for (int i = 1; i < argc; i++) {
        if (argv[i][0] == '-') {
            switch (argv[i][1]) {
            case 'h':
                Usage(argv[0]);
                exit(1);
            case 's':
                Size_Array = atoi(argv[++i]);
                break;
+           case 'l':
+               strcat(library, argv[++i]);
+               break;
            case 'v':
                Verbose = TRUE;
                break;
            }
        }
    }
}

...
```

`load_library.h`

```
void load_library(char *library_name);
```

`load_library.c`

```
#include <stdlib.h>
#include <stdio.h>
#include <dlfcn.h>

#include "load_library.h"
```

```
void (*sortfn)(int list[], int size);

void sort(int list[], int size){
    (*sortfn)(list, size);
}

void load_library(char *library_name){
    void *handle;
    char *error;
    handle = dlopen(library_name, RTLD_LAZY);
    if (!handle){
        fprintf(stderr, "%s\n", dlerror());
        exit(1);
    }

    dlerror();

    sortfn = (void (*)(int*, int)) dlsym(handle, "sort");

    error = dlerror();
    if (error != NULL){
        fprintf(stderr, "%s\n", error);
        exit(1);
    }

    dlclose(handle);
}
```

`Makefile`

```
DYNAMIC_LOAD=tri.exe

tri.exe: main_dynload.o load_library.o timer.o utils.o
    $(CC) -rdynamic -o $@ $^ -ldl
```

8. Et si vous deviez ajouter un nouvel algorithme de tri, comme le Shell sort. Comme feriez-vous ? Auriez-vous besoin de recompiler le programme tri.exe ?

La bibliothèque étant chargée à l'exécution en fonction de l'entrée utilisateur, il n'y a aucunement besoin de recompiler.

9. Au lieu de passer le type d'algorithme de tri utilisé en paramètre de votre exécutable, modifiez votre main pour charger successivement les différentes bibliothèques implémentant les algorithmes de tri.

10. Quelles sont les bibliothèques dynamiques utilisées par les programmes `python3` ou `java` qui se trouvent dans `/usr/bin/` ? Que pouvez-vous dire sur les bibliothèques utilisées ? Que pouvez-vous en conclure ?

```
user@host:~$ ldd /usr/bin/java
    linux-vdso.so.1 (0x00007fff98f84000)
    libjli.so => not found
    libc.so.6 => /usr/bin/../lib/libc.so.6 (0x00007f751f4d1000)
    /lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2
(0x00007f751f6ce000)
```

```
user@host:~$ ldd /usr/bin/python
    linux-vdso.so.1 (0x00007ffee1fdb000)
    libpython3.10.so.1.0 => /usr/lib/libpython3.10.so.1.0
(0x00007f6a438b7000)
    libc.so.6 => /usr/lib/libc.so.6 (0x00007f6a436eb000)
    libpthread.so.0 => /usr/lib/libpthread.so.0 (0x00007f6a436ca000)
    libdl.so.2 => /usr/lib/libdl.so.2 (0x00007f6a436c3000)
    libutil.so.1 => /usr/lib/libutil.so.1 (0x00007f6a436be000)
    libm.so.6 => /usr/lib/libm.so.6 (0x00007f6a4357a000)
    /lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2
(0x00007f6a43ca3000)
```

Assez peu de bibliothèques (dont `libdl`)

$\Longrightarrow$ potentiellement beaucoup de bibliothèques chargées à l'exécution