Techniques de programmation - TP n° 5 - Bibliothèques en langage C

Samson Pierre <samson.pierre@univ-pau.fr>

18/10/2021

Ce TP a pour objectif de vous familiariser avec les bibliothèques en langage C. Pour chaque exercice, sauf indication contraire, aucune fonction n'est autorisée et les mêmes options de compilation que celles du cours doivent être utilisées.

Exercice n° 1

Créez un programme `pythagoras.c` qui lit les paramètres passés en ligne de commande. Le premier paramètre est la longueur du côté d'un triangle rectangle différent de l'hypoténuse. Le deuxième paramètre est la longueur de l'autre côté de ce triangle rectangle différent de l'hypoténuse. Affichez un message d'erreur dans le flux d'erreur standard et retournez le code d'erreur 1 si le nombre de paramètres est différent de deux ou si les paramètres ne peuvent pas être convertis.

Créez une fonction `pythagoras` qui prend en paramètres la longueur du côté d'un triangle rectangle différent de l'hypoténuse `a`, la longueur de l'autre côté de ce triangle rectangle différent de l'hypoténuse `b` et qui retourne la longueur du troisième côté de ce triangle (l'hypoténuse). Voici un rappel du théorème de Pythagore : dans un triangle rectangle, le carré de la longueur de l'hypoténuse est égal à la somme des carrés des longueurs des deux autres côtés.

Affichez la longueur de l'hypoténuse. Compilez votre programme à l'aide d'un `Makefile` et liez votre programme à la bibliothèque `libm`. Remarquez que si vous compilez votre programme avec l'option `-static`, l'édition des liens ne se fera plus avec les bibliothèques partagées (fichiers `*.so`) mais avec les bibliothèques statiques (fichiers `*.a`). Ainsi, la taille du fichier généré est plus importante, la commande `ldd` ne fonctionne plus, mais cela garantit que votre programme se lance toujours avec la même version de bibliothèque que celle utilisée lors de la compilation et retire le besoin d'installer la bibliothèque sur la machine qui lance le programme. Notez toutefois que certaines machines ne supportent que les bibliothèques partagées ou que les bibliothèques statiques.

Voici le résultat attendu :

```
$ make
gcc -std=c89 -pedantic -Wall -Werror -g -o pythagoras.out pythagoras.c  -lm
$ ./pythagoras.out; echo $?
invalid number of arguments
1
$ ./pythagoras.out a; echo $?
invalid number of arguments
1
$ ./pythagoras.out a b; echo $?
invalid number for the first argument
1
$ ./pythagoras.out 3 b; echo $?
invalid number for the second argument
1
$ ./pythagoras.out 3 4; echo $?
hypotenuse = 5.000000
0
$ ldd pythagoras.out
        linux-vdso.so.1 (0x00007ffdf7f1d000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f292da2b000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f292d68a000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f292dd2f000)
$ file pythagoras.out
pythagoras.out: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, not stripped
$ du -h pythagoras.out
12K     pythagoras.out
$ gcc -std=c89 -pedantic -Wall -Werror -g -static -o pythagoras.out pythagoras.c -lm
$ ldd pythagoras.out
        not a dynamic executable
$ file pythagoras.out
pythagoras.out: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, not
stripped
$ du -h pythagoras.out
1.1M    pythagoras.out
```

```
$ 
```

Voici les fonctions autorisées pour cet exercice :

```
int fprintf(FILE *stream, const char *format, ...);
double pow(double x, double y);
int printf(const char *format, ...);
double sqrt(double x);
double strtod(const char *nptr, char **endptr);
```

Exercice n° 2

Créez un programme `quadratic-equation.c` qui lit les paramètres passés en ligne de commande. Le premier paramètre est le premier coefficient d'une équation du second degré (a). Le deuxième paramètre est le deuxième coefficient de cette équation du second degré (b). Le troisième paramètre est le troisième coefficient de cette équation du second degré (c). Affichez un message d'erreur dans le flux d'erreur standard et retournez le code d'erreur 1 si le nombre de paramètres est différent de trois ou si les paramètres ne peuvent pas être convertis.

Créez une fonction `quadratic_equation` qui prend en paramètres le premier coefficient d'une équation du second degré `a`, le deuxième coefficient de cette équation du second degré `b`, le troisième coefficient de cette équation du second degré `c`, un pointeur vers le nombre de solutions `nsol`, un pointeur vers la première solution `sol1`, un pointeur vers la deuxième solution `sol2` et qui ne retourne rien. Voici un rappel du calcul des solutions d'une équation du second degré : une équation $ax^2 + bx + c = 0$, avec discriminant $\Delta = b^2 - 4ac$, a 0 solution si $\Delta < 0$, a 1 solution si $\Delta = 0$ qui est $\frac{-b}{2a}$ et a 2 solutions si $\Delta > 0$ qui sont $\frac{-b-\sqrt{\Delta}}{2a}$ et $\frac{-b+\sqrt{\Delta}}{2a}$.

Affichez le nombre de solutions et les solutions. Compilez votre programme à l'aide d'un `Makefile` et liez votre programme à la bibliothèque `libm`.

Voici le résultat attendu :

```
$ make
gcc -std=c89 -pedantic -Wall -Werror -g -o quadratic-equation.out quadratic-equation.c  -lm
$ ./quadratic-equation.out; echo $?
invalid number of arguments
1
$ ./quadratic-equation.out a; echo $?
invalid number of arguments
1
$ ./quadratic-equation.out a b; echo $?
invalid number of arguments
1
$ ./quadratic-equation.out a b c; echo $?
invalid number for the first argument
1
$ ./quadratic-equation.out 1 b c; echo $?
invalid number for the second argument
1
$ ./quadratic-equation.out 1 1 c; echo $?
invalid number for the third argument
1
$ ./quadratic-equation.out 1 1 -2; echo $?
2 solutions: -2.000000 and 1.000000
0
$ ./quadratic-equation.out 4 4 1; echo $?
1 solution: -0.500000
0
$ ./quadratic-equation.out 1 1 1; echo $?
0 solutions
0
$ 
```

Voici les fonctions autorisées pour cet exercice :

```
int fprintf(FILE *stream, const char *format, ...);
double pow(double x, double y);
int printf(const char *format, ...);
double sqrt(double x);
double strtod(const char *nptr, char **endptr);
```

Exercice n° 3

Créez un programme `libxml2.c` qui lit un paramètre passé en ligne de commande. Ce paramètre est le chemin d'un fichier. Affichez un message d'erreur dans le flux d'erreur standard et retournez le code d'erreur 1 si le nombre de para-

mètres passés en ligne de commande est différent de un.

Utilisez la bibliothèque `libxml2` (dont la documentation est disponible à l'adresse `http://xmlsoft.org/`) pour analyser un fichier XML et en extraire des informations pour les afficher. Affichez un message d'erreur dans le flux d'erreur standard et retournez le code d'erreur 1 si le fichier ne peut pas être analysé (exemple : s'il n'existe pas). Ignorez tous les noeuds ou attributs qui ne décrivent pas une bibliothèque ou un livre. Compilez votre programme à l'aide d'un `Makefile` et liez votre programme à la bibliothèque `libxml2`. Remarquez que les commandes `xml2-config` et `pkg-config` permettent d'obtenir les options à ajouter aux variables `CFLAGS`, `LDFLAGS` et `LDLIBS` de votre `Makefile`. Lancez votre programme avec la commande `valgrind` afin de vérifier si la mémoire allouée dynamiquement a correctement été libérée.

Voici le résultat attendu :

```
$ cat test.xml
<something>hello world</something>
$ cat library.xml
<library name="Sciences Library">
        <days>
                <day>Monday</day>
                <day>Tuesday</day>
                <day>Wednesday</day>
                <day>Thursday</day>
        </days>
        <books>
                <book name="The C Programming Language">
                        <authors>
                                <author>Brian W. Kernighan</author>
                                <author>Dennis M. Ritchie</author>
                        </authors>
                        <publisher>Prentice Hall</publisher>
                        <year>1998</year>
                        <isbn>9780131103627</isbn>
                </book>
                <book name="C: The Complete Reference">
                        <authors>
                                <author>Herbert Schildt</author>
                        </authors>
                        <publisher>McGraw-Hill Education</publisher>
                        <year>2000</year>
                        <isbn>9780072121247</isbn>
                </book>
        </books>
</library>
$ make
gcc -std=c89 -pedantic -Wall -Werror -g `pkg-config libxml-2.0 --cflags` -o libxml2.out libxml2.c
`pkg-config libxml-2.0 --libs-only-L` `pkg-config libxml-2.0 --libs-only-l`
$ ./libxml2.out; echo $?
invalid number of arguments
1
$ ./libxml2.out file.xml; echo $?
I/O warning : failed to load external entity "file.xml"
unable to parse the document
1
$ ./libxml2.out test.xml; echo $?
0
$ ./libxml2.out library.xml; echo $?
library:
        name: Sciences Library
        days:
                day: Monday
                day: Tuesday
                day: Wednesday
                day: Thursday
        books:
                book:
                        authors:
                                author: Brian W. Kernighan
                                author: Dennis M. Ritchie
                        publisher: Prentice Hall
                        year: 1998
                        isbn: 9780131103627
                book:
                        authors:
                                author: Herbert Schildt
                        publisher: McGraw-Hill Education
                        year: 2000
                        isbn: 9780072121247
0
```

```
$ 
```

Voici les fonctions autorisées pour cet exercice :

```
int fprintf(FILE *stream, const char *format, ...);
int printf(const char *format, ...);
void xmlCleanupParser(void);
xmlNodePtr xmlDocGetRootElement(const xmlDoc *doc);
void xmlFreeDoc(xmlDocPtr cur);
xmlChar *xmlGetProp(const xmlNode *node, const xmlChar *name);
xmlChar *xmlNodeListGetString(xmlDocPtr doc, const xmlNode *list, int inLine);
xmlDocPtr xmlParseFile(const char *filename);
int xmlStrcmp(const xmlChar *str1, const xmlChar *str2);
```