Programmation C avancée

Concepts & Outils
pour le développement

maj 01/2023

tracer les accès mémoires

- valgrind [vælgrɪnd] est un outil qui intègre de nombreux tests pour l'exécution de programmes.
- Par défaut, c'est l'outil **memcheck** qui est utilisé:
  - Permet de détecter des erreurs d'exécution qui ne sont pas relevées par le système comme le dépassement d'un tableau par exemple.
  - L'exécution du programme par valgrind est contrôlée pas à pas, ce qui ralenti beaucoup le programme. Il y a également une surconsommation mémoire importante.

memcheck

- memcheck propose de nombreuses options pour contrôler son execution, cela peut permettre d'exécuter valgrind sur des programmes « lourds »

--leak-check=<no|summary|yes|full> [default: summary]
--leak-resolution=<low|med|high> [default: high]
--show-leak-kinds=<set> [default: definite,possible]
--errors-for-leak-kinds=<set> [default: definite,possible]
--leak-check-heuristics=<set> [default: all]
--leak-check-heuristics=stdstring,length64,newarray,multipleinheritance.
--show-reachable=<yes|no> , --show-possibly-lost=<yes|no>
--xtree-leak=<no|yes> [no]
--xtree-leak-file=<filename> [default: xtleak.kcg.%p]
--undef-value-errors=<yes|no> [default: yes]
--track-origins=<yes|no> [default: no]
....

memcheck

- memcheck propose de nombreuses options pour contrôler son execution, cela peut permettre d'exécuter valgrind sur des programmes « lourds »

--partial-loads-ok=<yes|no> [default: yes]
--expensive-definedness-checks=<no|auto|yes> [default: auto]
--keep-stacktraces=alloc|free|alloc-and-free|alloc-then-free|none [default: alloc-and-free]
--freelist-vol=<number> [default: 20000000]
--freelist-big-blocks=<number> [default: 1000000]
--workaround-gcc296-bugs=<yes|no> [default: no]
--ignore-range-below-sp
--ignore-range-below-sp=<number>-<number>
--ignore-range-below-sp=8192-8189. Only one range may be specified.
--show-mismatched-frees=<yes|no> [default: yes]
--ignore-ranges=0xPP-0xQQ[,0xRR-0xSS]
--malloc-fill=<hexnumber>
--free-fill=<hexnumber>

memcheck

- les erreurs détectés par memcheck sont:
  - Invalid read of size 4
  - Conditional jump or move depends on uninitialised value(s)
  - Invalid free()
  - Source and destination overlap in memcpy(0xbffff294, 0xbffff280, 21)
  - fuites mémoires...

valgrind

- Pour pouvoir fonctionner efficacement avec valgrind, le programme doit avoir été compilé avec -g et -O0
- ensuite, on lance le programme dans valgrind :

  valgrind ./a.out arg1 arg2 ...
- valgrind va afficher les erreurs mémoires au fur et à mesure de leurs apparitions
- Il est possible de demander à valgrind de lancer gdb à chaque erreur :

  valgrind --vgdb-error=1 ./a.out

exemple :

```
allali@hebus:/tmp$ valgrind --vgdb-error=1 ./a.out
==9539== Memcheck, a memory error detector
==9539== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==9539== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==9539== Command: ./a.out
==9539==
==9539==
==9539== TO DEBUG THIS PROCESS USING GDB: start GDB like this
==9539==   /path/to/gdb ./a.out
==9539== and then give GDB the following command
==9539==   target remote | /usr/lib/valgrind/../../bin/vgdb --pid=9539
==9539== --pid is optional if only one valgrind process is running
==9539==
==9539== Invalid write of size 4
==9539==    at 0x400570: main (s.c:7)
==9539==  Address 0x51fc050 is 0 bytes after a block of size 16 alloc'd
==9539==    at 0x4C2ABA0: malloc (in /usr/lib/valgrind/vgpreload_memcheck.so)
==9539==    by 0x40054E: main (s.c:4)
==9539==
==9539== (action on error) vgdb me ...
```

Toute erreur signalée par valgrind mérite une analyse et, à de très rares exceptions, doit être corrigée !

```
allali@hebus:/tmp$ gdb a.out
...
Reading symbols from a.out...done.
(gdb) target remote | /usr/lib/valgrind/../../bin/vgdb --pid=9539
…
Loaded symbols for /lib64/ld-linux-x86-64.so.2
0x0000000000400570 in main (argc=1, argv=0xffeffff98) at s.c:7
7           t[i]=0;
(gdb) p t
$1 = (int *) 0x51fc040
(gdb) p i
$2 = 4
```

```
1 #include<stdlib.h>

3 int main(int argc, char **argv){
4   int *t=malloc(sizeof(int)*4);
5   int i;
6   for(i=0;i<6;++i)
7     t[i]=0;
8    return 0;
9 }
```

valgrind: lire le message

```
int main(){
  int *p=malloc(sizeof(int)*3);
  int i;
  for(i=0;i<5;++i)
    p[i]=0;
}
```

```
==253883== Invalid write of size 4
==253883==    at 0x109180: main (a.c:8)
==253883==  Address 0x4ab304c is 0 bytes after a block of size 12 alloc'd
==253883==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-
linux.so)
==253883==    by 0x10915E: main (a.c:5)
```

valgrind: lire le message

```
int main(){
  int *p=malloc(sizeof(int)*3);
  int i,s=0;
  for(i=0;i<5;++i)
    s+=p[i];
}
```

```
==254030== Invalid read of size 4
==254030==    at 0x109187: main (a.c:8)
==254030==  Address 0x4ab304c is 0 bytes after a block of size 12 alloc'd
==254030==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux
==254030==    by 0x10915E: main (a.c:5)
==254030==
```

valgrind: lire le message

```
int main(){
  int *p=malloc(sizeof(int)*3);
  int i,s=0;
  if (p[i]==0){
    s=1;
  }
}
```

```
==254181== Use of uninitialised value of size 8
==254181==    at 0x10917E: main (a.c:7)
==254181==
==254181== Conditional jump or move depends on uninitialised value(s)
==254181==    at 0x109182: main (a.c:7)
==254181==
```

valgrind: limite avec la pile

- Valgrind détecte bien les erreurs liées à la mémoire sur le tas mais moins bien lorsque la mémoire concernée est dans la pile:
- ce code ne génère pas d'erreur alors qu'on utilise de la mémoire non initialisée!

```
int f(){
  int p[4];
  return p[2];
}

int main(){
  int i=0;
  i+=f();
  i+=f();

}
```

- Si l'on utilise i plus tard, alors une erreur peut apparaître.

valgrind : fuite mémoire

#QDLE#Q#ABCD*#45#

- Une fuite mémoire correspond à de la mémoire encore réservée à la fin de votre programme
- Celle-ci peut être ?
  - A. dans la pile et le tas
  - B. dans la pile, le tas et le segment de données
  - C. uniquement dans la pile
  - D. uniquement dans le tas

valgrind : fuite mémoire

- A la fin de l'exécution, valgrind liste les blocs non libérés et les classe selon :
  - que plus rien ne pointe sur ce segment (definitively lost)
  - qu'un autre segment de mémoire pointe encore dessus (indirectly lost)
  - encore accessible par une variable statique (still reachable)

valgrind : definitively lost

```
allali@hebus:/tmp$ valgrind --leak-check=full ./a.out
==9900== Memcheck, a memory error detector
==9900== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==9900== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==9900== Command: ./a.out
==9900==
==9900==
==9900== HEAP SUMMARY:
==9900==     in use at exit: 1 bytes in 1 blocks
==9900==   total heap usage: 1 allocs, 0 frees, 1 bytes allocated
==9900==
==9900== 1 bytes in 1 blocks are definitely lost in loss record 1 of 1
==9900==    at 0x4C2ABA0: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==9900==    by 0x400543: main (s.c:4)
==9900==
==9900== LEAK SUMMARY:
==9900==    definitely lost: 1 bytes in 1 blocks
==9900==    indirectly lost: 0 bytes in 0 blocks
==9900==      possibly lost: 0 bytes in 0 blocks
==9900==    still reachable: 0 bytes in 0 blocks
==9900==         suppressed: 0 bytes in 0 blocks
==9900==
==9900== For counts of detected and suppressed errors, rerun with: -v
==9900== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

```
#include<stdlib.h>

int main(){
  malloc(sizeof(char));
}
```

valgrind : indirectly lost

```
allali@hebus:/tmp$ valgrind --leak-check=full --show-reachable=yes ./a.out
==9973== Memcheck, a memory error detector
==9973== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==9973== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==9973== Command: ./a.out
==9973==
==9973==
==9973== HEAP SUMMARY:
==9973==     in use at exit: 9 bytes in 2 blocks
==9973==   total heap usage: 2 allocs, 0 frees, 9 bytes allocated
==9973==
==9973== 1 bytes in 1 blocks are indirectly lost in loss record 1 of 2
==9973==    at 0x4C2ABA0: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==9973==    by 0x400555: main (s.c:5)
==9973==
==9973== 9 (8 direct, 1 indirect) bytes in 1 blocks are definitely lost in loss record 2 of 2
==9973==    at 0x4C2ABA0: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==9973==    by 0x400547: main (s.c:4)
==9973==
==9973== LEAK SUMMARY:
==9973==    definitely lost: 8 bytes in 1 blocks
==9973==    indirectly lost: 1 bytes in 1 blocks
==9973==      possibly lost: 0 bytes in 0 blocks
==9973==    still reachable: 0 bytes in 0 blocks
==9973==         suppressed: 0 bytes in 0 blocks
==9973==
==9973== For counts of detected and suppressed errors, rerun with: -v
==9973== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

```
#include<stdlib.h>

int main(){
  char **p=malloc(sizeof(char *));
  p[0]=malloc(sizeof(char ));
}
```

valgrind : still reachable

```
allali@hebus:/tmp$ valgrind --leak-check=full --show-reachable=yes ./a.out
==9996== Memcheck, a memory error detector
==9996== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==9996== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==9996== Command: ./a.out
==9996==
==9996==
==9996== HEAP SUMMARY:
==9996==     in use at exit: 1 bytes in 1 blocks
==9996==   total heap usage: 1 allocs, 0 frees, 1 bytes allocated
==9996==
==9996== 1 bytes in 1 blocks are still reachable in loss record 1 of 1
==9996==    at 0x4C2ABA0: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==9996==    by 0x400543: main (s.c:5)
==9996==
==9996== LEAK SUMMARY:
==9996==    definitely lost: 0 bytes in 0 blocks
==9996==    indirectly lost: 0 bytes in 0 blocks
==9996==      possibly lost: 0 bytes in 0 blocks
==9996==    still reachable: 1 bytes in 1 blocks
==9996==         suppressed: 0 bytes in 0 blocks
==9996==
==9996== For counts of detected and suppressed errors, rerun with: -v
==9996== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
#include<stdlib.h>

char *p;
int main(){
  p=malloc(sizeof(char ));
}
```

valgrind

#QDLE#Q#AB*CD#30#

```
int main(){
    int **p=malloc(sizeof(*p)*3) ;
    p[2]=malloc(sizeof(**p)*4) ;
    free(p) ;
}
```

- Sur cet exemple, quel est le type de la fuite ?

A. direct B. indirect C. still D. pas de fuite

valgrind

- options :
  - --leak-check=full --show-reachable=yes
  - --malloc-fill=<hexnumber>
  - --free-fill=<hexnumber>
  - …
- Attention : un programme n'ayant pas de problème avec valgrind n'est pas nécessairement un programme sans bug !

SSL BuG in Debian

- May 2006, bug report (debian maintainer):

The problems are the following 2 pieces of code in crypto/rand/md_rand.c:

247:

```
        MD_Update(&m,buf,j);
```

467:

```
#ifndef PURIFY
        MD_Update(&m,buf,j); /* purify complains */
#endif
```

What it's doing is adding uninitialised numbers to the pool to create random numbers.

Valgrind: ”Use of uninitialised value of size ...”

SSL BuG in Debian

- patch:

The problems are the following 2 pieces of code in crypto/rand/md_rand.c:

246:

```
        /**Don't add uninitialised datas
         * MD_update(&m, buf, j);
        */
```

467:

```
#ifndef PURIFY
        /* MD_Update(&m,buf,j); /* purify complains */ */
#endif
```

What it's doing is adding uninitialised numbers to the pool to create random numbers.

=> Le générateur de nombre aléatoire de ssl ne dépend plus que du pid du process (32768 valeurs possibles) et devient donc prédictible!

Moralité

- Identifier un bug est une chose, le corriger en est une autre

- Réfléchir à deux fois avant de mettre quelque chose en production!

Convention de codage

- Une convention de codage est un document qui liste les règles d'écriture de code source pour un projet / une entreprise :
  - nommage des fonctions, variables, macro…
  - nommage et organisation de fichiers
  - langue pour le code et les commentaires
  - formatage spécifique (boucles, tests…)
  - techniques de programmation
- Une convention est un document vivant qui doit être mis à jour si nécessaire.

convention de codage : exemple

- https://www.kernel.org/doc/Documentation/CodingStyle
  - indentation
  - taille max de lignes
  - positionnement des accolades et parenthèses
  - espaces
  - nommage : C is a Spartan language, and so should your naming be….
  - typedef
  - fonctions
  - sortie de fonctions
  - commentaires
  - … .

Convention de codage & clang-format

- L'outil *clang-format* permet de reformater automatiquement du code source selon une convention.
- Il contient déjà plusieurs conventions de codage : LLVM, Google, Chromium, Mozilla, WebKit
- Il est possible de définir sa convention via un fichier .clang-format
- Ne permet pas d'implémenter **toutes** les règles

Documentation

- La documentation est primordiale pour un projet sur le long terme
- Plusieurs niveaux de doc :
  - très haut niveau : présentation large, vue d'ensemble, éléments d'architecture
  - modules : aspects fonctionnels, périmètre
  - fonction : description des paramètres, valeurs de retours, spécifications
  - code : astuces mises en œuvre, point d'algorithmique non trivial, justification de choix.

Documentation

- Le maintient d'une documentation peut être un travail long et il arrive souvent qu'il y ai divergence entre la documentation et le code.
- Pour cela, on rapproche la documentation du code en l'incluant dans celui-ci
- Utilisation des commentaires et de générateurs de documentation
- Ne permet pas de faire toute la documentation (en particulier, la doc de haut niveau, manuel d'utilisation...).

doxygen

- doxygen permet de générer la documentation au format html/pdf/latex… à partir des commentaires dans le code source.
- doxygen peut également intégrer de la documentation au format Markdown
- Le résultat est une documentation séparée des sources mais synchronisée avec celles-ci.

doxygen

- doxygen utilise des commentaires suivant certaines règles d'écriture :

```
//! power function
/*! The pow() function returns the value of x raised to the power of y.
* \param x a real in double format
* \param y a real in double format
* \return x raised to power of y or NaN if wrong arguments
*/
double pow(double x, double y) ;
```

- doxygen gère d'autres formats (javadoc par exemple).

doxygen

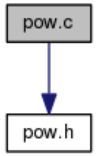
- A partir des commentaires structurés des sources, doxygen va générer une documentation dans différents formats :
  - Html
  - Latex
  - Page de manuel (man pages)
  - ...

My Project

pow.c File Reference

```
#include "pow.h"
```

Include dependency graph for pow.c:



Go to the source code of this file.

Functions

double **pow** (double x, double y)

Function Documentation

```
double pow ( double  x,
             double  y
           )
```

power function The **pow()** function returns the value of x raised to the power of y.

Parameters

x a real in double format

y a real in double format

Returns

x raised to power of y or NaN if wrong arguments

Definition at line **5** of file **pow.c**.

doxygen et README.md

- Il est souhaitable pour un projet d'avoir un fichier qui donne des informations générales de haut niveau :
  - auteurs
  - objectif de l'application
  - pré-requis
  - installation
  - utilisation
  - …
- Une technique consiste à donner ces éléments dans un fichier README.md à la racine du projet. Ce fichier pourra être intégré à la documentation par doxygen :

```
My great project        {#mainpage}
==============

About
--------
```

My Project

Main Page | Files

My Project
- My great project
- Files

My great project

Author

Julien Allali

About

example for PG106 course.

Commentaires : qq règles

- Les commentaires doivent être **utiles**
- Pour une fonction :
  - ce que fait la fonction (spécification)
  - éventuellement, comment elle le fait (algo, complexité, coût mémoire…)
  - **domaine de valeur des paramètres**
  - cas d'erreurs
- Les commentaires dans le code doivent servir à suivre la logique de celui-ci, par ex :
  - // set default value into the matrix
  - …
  - // fill the matrix according to the formula : M[i][j]=min(M[i-1][j],M[i][j-1])
  - …
  - // backtrace to compute the alignment
  - …

Commentaires : qq règles

- Pour les modules, penser à ajouter une description générale de ce que fait le module, avec un code d'exemple d'utilisation.
- Au début des fichiers d'implémentation, un entête spécifie :
  - les auteurs,
  - la licence, (copyright si rien)
  - une liste datée des modifications

commentaires

#QDLE#Q#A*BC#30#

```
int my_function(int arg){
     // set a counter to 0
     int counter=0 ;
     … ..
     … ..
     if (x==0){
          // because the string is empty in
          //this case
     }
     …
     …
     for(i=0;i<counter-1;++i){
          // parse the char of the
          // string avoiding the \0
          … .
     }
     … .
```

- sur cet exemple, quel commentaire est sans intérêt ?
  - A
  - B
  - C

gestion de sources

- Lorsque l'on interagit avec d'autres développeurs, il est indispensable de pouvoir communiquer des propositions de modifications (ajout de fonctionnalité, correctif de bug, amélioration des perfs…) :

gestion de sources

#QDLE#S#ABC#25#

- Lorsque l'on interagit avec d'autres développeurs, il est indispensable de pouvoir communiquer des propositions de modifications (ajout de fonctionnalité, correctif de bug, amélioration des perfs…) :

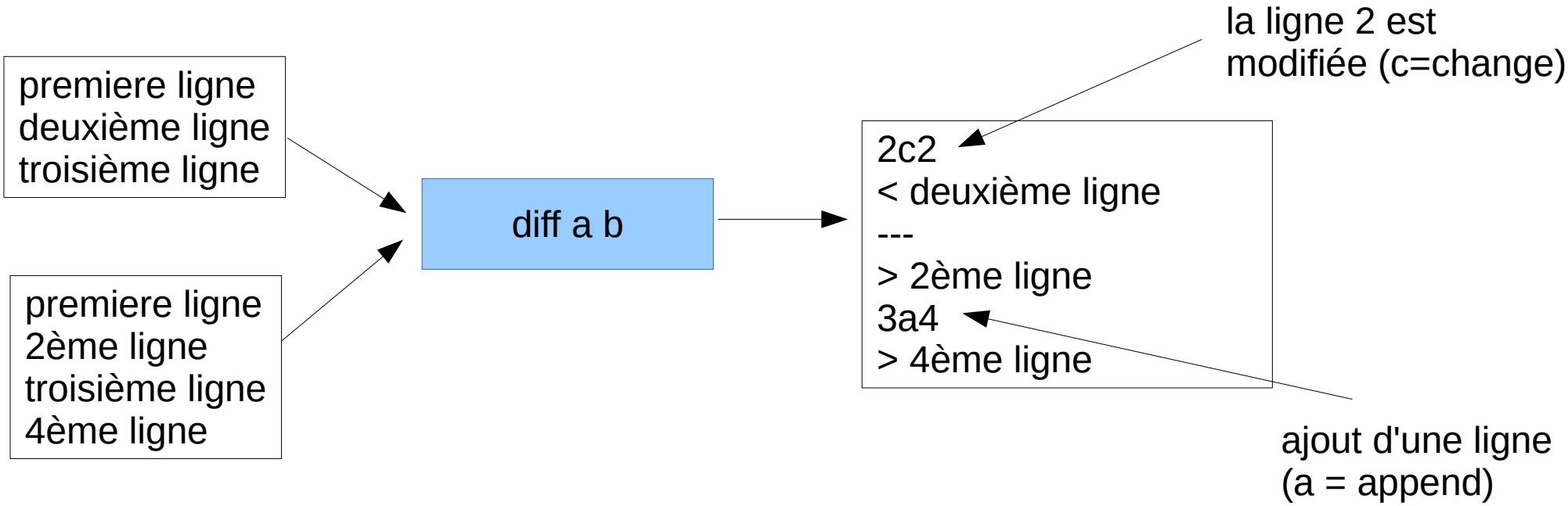
A) j'envoie tout le code en indiquant que c'est une nouvelle version.

B) j'écris un email détaillé des modifications à effectuer.

C) autre approche...

diff

- diff est un outil d'analyse de texte qui compare deux fichiers entre eux et produit le nombre minimum d'éditions de lignes à faire sur le premier fichier pour obtenir le second :



diff : side by side

- on peut afficher les deux fichiers cote à cote :

```
diff -y a b
```

```
premiere ligne                                  premiere ligne
deuxième ligne                                | 2ème ligne
troisième ligne                                 troisième ligne
                                              >     4ème ligne
```

- diff permet la comparaison récursive de deux arborescences.

diff : recursif

- Je souhaite modifier le code source d'un projet :
  1. je fais un copie de sauvegarde des sources d'origine
  2. j'effectue mes modifications
  3. à tout moment, je visualise mes modifications avec diff : `diff -r projet projet_new`

diff : recursif

- Je souhaite modifier le code source d'un projet :
  1. je fais un copie de sauvegarde des sources d'origine
  2. j'effectue mes modifications
  3. à tout moment, je visualise mes modifications avec diff : `diff -r projet projet_new`

```
diff -r GenTaskLib/TaskParser.cpp GenTaskLib_new/TaskParser.cpp
15c15
<         throw gtl::InvalidArgumentException("json parser error: expected a dictionnary at each task");
---
>         throw gtl::InvalidArgumentException("json parser error: expected a dictionnary for each task");
diff -r GenTaskLib/TaskParser.hpp GenTaskLib_new/TaskParser.hpp
36a37,38
>   /*! TaskParser: build a task from a json description.
>    */
```

Ajout d'un commentaire

diff : algorithme

- Le programme diff repose sur un problème classique d'algorithmique du texte : La plus longue sous-séquence commune

  MOTELS ARE NOT HELL!

  MIROIR TU ES LA!

- Sur cet exemple « MO T » est une sous séquence commune.

diff : algorithme

#QDLE#Q#ABC*D#60#

- Le programme diff repose sur un problème classique d'algorithmique du texte : La plus longue sous-séquence commune

  MOTELS ARE NOT HELL!

  MIROIR TU ES LA!

- Taille de la plus longue sous séquence commune ? (les espaces comptes).

  A. 6 B. 8 C. 9 D. 10

diff : algorithme

- Le programme diff repose sur un problème classique d'algorithmique du texte : La plus longue sous-séquence commune

  MOTELS_ARE_NOT_HELL!

  MIROIR_TU_ES_LA!

diff : algorithm

- Chaque fichier est découpé en lignes
- Les lignes sont comparées entre elles
- Puis l'ensemble des lignes sont comparées entre elles (LCS où chaque « symbole » représente une ligne).

diff et communication

- Ainsi, si l'on souhaite communiquer une modification, il suffit d'envoyer le résultat d'un diff récursif : `diff -rupN original new > patch`
- On appel ce fichier un patch.
- Le destinataire peut lire ce fichier et comprendre vos modifications
- Il peut également appliquer ces modifications en local grâce au programme patch

- *les options upN sont nécessaire au fonctionnement de patch, elles ajoutent des informations de contexte (u), de fonction (p), d'ajout de fichier (N)*

patch

- le programme patch permet d'appliquer les modifications identifiées par diff.
- Ainsi sur l'exemple précédent je peux faire :

```
$ cp -R GenTaskLib GenTaskLibMod
$ cd GenTaskLibMod
$ patch < ../patch
patching file TaskParser.cpp
patching file TaskParser.hpp
$ cd .. ; diff -r GenTaskLib GenTaskLibMod
```

```
diff -r GenTaskLib/TaskParser.cpp GenTaskLibMod/TaskParser.cpp
15c15
<         throw gtl::InvalidArgumentException("json parser error: expected a dictionnary at each task");
---
>         throw gtl::InvalidArgumentException("json parser error: expected a dictionnary for each task");
diff -r GenTaskLib/TaskParser.hpp GenTaskLibMod/TaskParser.hpp
36a37,38
>     /*! TaskParser: build a task from a json description
```

diff & patch

- Dans le monde open source, diff et patch sont extrêmement utilisés