UV LO22 - Méthodes de Test et de Vérification du Logiciel

Mohamed Sallak – Walter Schön
Maître de Conférences - HDR
Université de Technologie de Compiègne

Analyse statique automatique

Statique automatique (1)

- **L'idée de base est d'analyser un code source .**

- **Il n'y a pas d'exécution du code.**
- **Il n'y a pas de modification ou d'altération du code.**

- **L'analyse statique automatique est plus liée à la phase de conception du logiciel.**

- **Elle est réalisée à l'aide d'outils informatiques analysant le code source ou générant du code compilé :**
  - analyseurs plus ou moins évolués ou
  - compilateurs.
  - Elles ne remplacent pas les inspections et autres revues mais viennent en complément.

Statique automatique (2)

- **L'analyse statique automatique est plus liée à la phase de conception du logiciel.**

- **Elle est réalisée à l'aide d'outils informatiques analysant le code source ou générant du code compilé :**
  - analyseurs plus ou moins évolués ou
  - compilateurs.

Statique automatique (4)

- **Elles ne remplacent pas les inspections et autres revues mais viennent en complément.**

Static analysis checks

Fault class	Static analysis check
Data faults	Variables used before initialisation Variables declared but never used Variables assigned twice but never used between assignments Possible array bound violations Undeclared variables
Control faults	Unreachable code Unconditional branches into loops
Input/output faults	Variables output twice with no intervening assignment
Interface faults	Parameter type mismatches Parameter number mismatches Non-usage of the results of functions Uncalled functions and procedures
Storage management faults	Unassigned pointers Pointer arithmetic

Comparateur (1)

- Un comparateur de fichier permet d'établir les différences qui existent entre deux fichiers.
- Cet outil joue un rôle important lors de la mise au point d'un système car cette activité nécessite l'examen d'un volume important de sorties de test.
- Il est donc souhaitable de pouvoir l'automatiser.

- Exemple : la commande UNIX « diff »

Comparateur (2)

- **Quelques utilisations:**
  - La comparaison de programme permet d'isoler les évolutions d'une version à l'autre.
    - ✓ Cela permet d'isoler les partie qui ont changer
    - ✓ Cela permet de juger si les évolutions sont conforme à l'évolution.
  - La comparaison des fichiers de résultats de tests permet de vérifier que des résultats de test sont toujours identiques.

Lint et autre (1)

- « Lint » est un outil Unix qui permet d'analyser un code C afin d'identifier les erreurs latentes liées aux typages, aux appels de fonctions, …
- Cet outil est nécessaire car le langage C est un langage faiblement typé et les compilateurs C sont souvent très laxiste.

- Il ne s'agit plus seulement d'un outils mais d'une famille.

Exemple (1)

```
main(void)
{
    int i;
    for(i = 0; i < 10l; i++)
    {
        printf("%d\n",i);
     }
}
```

Programme correct et compilé sans erreur.

Quel est le dernier chiffre affiché ?

Exemple (1bis)

```
main(void)
{
    int i;
    for(i = 0; i < 10l; i++)
    {
        printf("%d\n",i);
     }
}
```

Et avec ce changement de fonte de caractère
Quel est le dernier chiffre affiché ?

Pour information 10l caractérise un grand entier en C

Exemple (2)

```
void fill (char *)
char *copy( char * );

#define N 100

void f()
{
    char buffer[N];
    char *p;
    char *q;
    p = (char *) malloc( N );
    p++;
    fill( buffer );
    fill( p );
    q = copy( p );
    strcpy( q, buffer );
    free( buffer );
    free( p );
}
```

Morceau de code « correct » ?

Exemple (2bis)

Splint 3.1.1 --- 23 Apr 2004

f.c: (in function f)
f.c:22:12: Passed storage buffer not completely defined (*buffer is undefined): fill (buffer)
Storage derivable from a parameter, return value or global is not defined.

f.c:23:12: Possibly null storage p passed as non-null param: fill (p)
A possibly null pointer is passed as a parameter corresponding to a formal parameter.

f.c:20:10: Storage p may become null
f.c:23:12: Passed storage p not completely defined (*p is undefined): fill (p)

f.c:20:6: Storage *p allocated

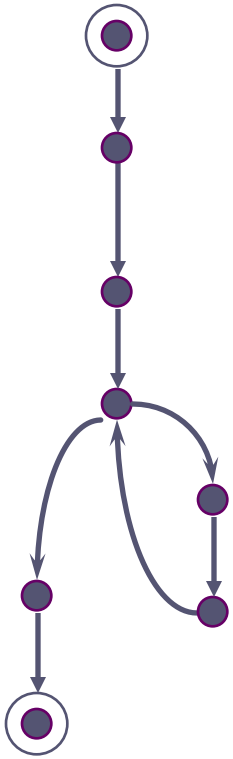
f.c:28:2: Fresh storage q not released before return
A memory leak has been detected. Storage allocated locally is not released before the last reference to it is lost.

f.c:24:6: Fresh storage q created

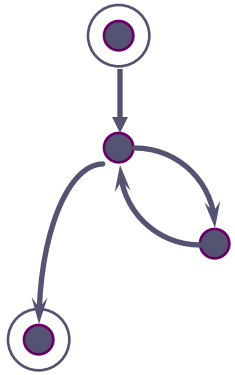
Exemple (1)

```
void main()
{
    int x = 0;
    int y = 1;
    while (y < 10)
    {
        y = 2 * y;
        x = x + 1;
    }
    printf ("%d",x);
    printf ("%d",y);
}
```

Exemple (2)



V= 8 -8 +2



V= 4-4+2