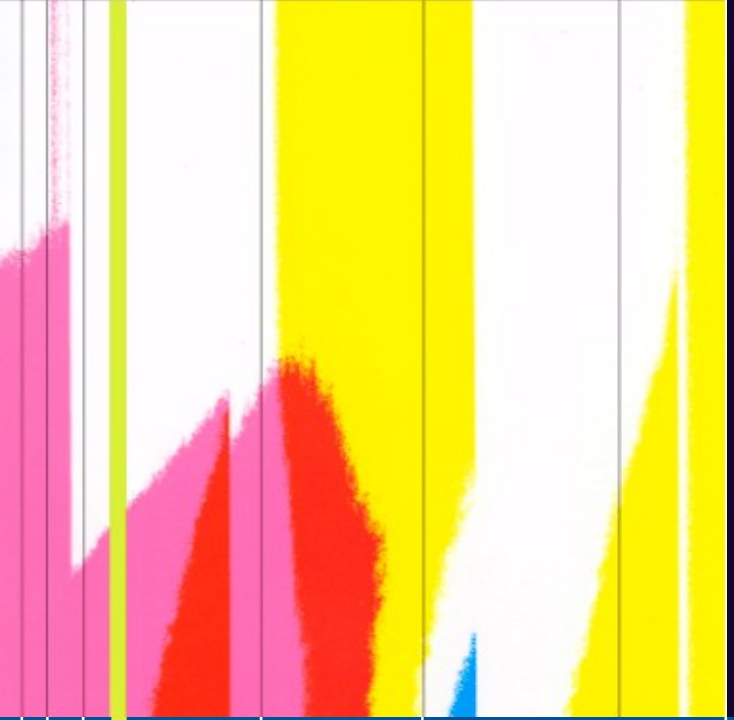

Debogage :
Méthodes...

Rué François - 28 / 01 / 10

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
BORDEAUX - SUD-OUEST

Présentation

- Pourquoi ?
- Comment ?
- Mise en Application.
- En parallèle.
- Pistes à suivre ?

[Pourquoi]
À l'origine du bogue.

- **BOGUE** = défaut de logique

⇒ calcul ou traitement incorrect à l'origine de dysfonctionnements et/ou de pannes.

[Pourquoi]
À l'origine du bogue.

- **BOGUE** = défaut de logique

⇒ calcul ou traitement incorrect à l'origine de dysfonctionnements et/ou de pannes.

Gravité du dysfonctionnement :

- Bénigne (défauts d'affichage mineurs...)
- Majeure (explosion d'ariane 5)

BOGUE = ERREUR DE PROGRAMMATION

[Pourquoi]
Les Classiques.

Il existe une panoplie de bogues classiques :

- Comportement du code.
- Comportement des librairies.
- Comportement des outils de compilation.
- Comportement de l'OS ou du programme utilisé.
- Panne Matériel.
- ...

[Pourquoi]

Ou comment l'éviter...

Les bonnes pratiques de programmation … brièvement :

- Commenter son code.
- Inclure des sauts de ligne pour faciliter la lecture.
- Indenter les boucles et les blocs.
- Choisir des noms de variables et fichiers qui correspondent à ce qu'ils doivent représenter.
- Utiliser le contrôle de version …

Liens :

- GNU Coding Standards : http://www.gnu.org/prep/standards.html
- Sun's Java Code Conventions : http://java.sun.com/docs/codeconv

[Comment]
Principes Généraux.

Chaque cas est unique. Cependant, des principes généraux peuvent être décrits ainsi :

1. Reconnaître l'existence d'un bogue.
2. Isoler la source du bogue.
3. Identifier la cause du bogue.
4. Déterminer un correctif pour ce bogue.
5. Appliquer le correctif et tester.

[Comment]
En règle générale.

Personne ne sait aujourd'hui créer de logiciel sans défaut ! La validation de Windows 2000 avait fait appel à 600 000 bêta testeurs, il restait pourtant au lancement de sa commercialisation 63 000 problèmes potentiels dans le code, dont 28 000 sont réels. Un **logiciel commercial** *type contient entre* **20 et 30 bogues pour mille lignes** !

[Comment]

1. Reconnaître le bogue.

Éclaircir pourquoi il s'agit d'un bogue :

- Est – ce une erreur de conception ?
- Est – ce une mauvaise compréhension de son utilisation ?
- Problèmes intermédiaires non traités par le programme ?

Identifier les symptômes du bogue :

- Observer les symptômes du problème.
- Sous quelles conditions le problème est détecté ?
- Qu'est ce qui fonctionne ... si le bogue n'apparaît pas ?

Documenter le problème : Bugs file, Bug Tracking System ...

[Comment]

2. Isoler la source du bogue.

Reproduire le Bogue.

L'idée est d'identifier quelle portion du code produit l'erreur.

Une fois le bogue trouvé  déterminer la cause.

Source du symptôme ≠ Source du problème

[Comment]
3. Identifier la cause.

- Cette étape est la **plus difficile** du débogage.
- Une bonne compréhension du système est indispensable afin d'identifier les causes du bogue :
  - Quel est le système ?
    - Le code source.
    - Les outils de « construction » et leur environnement.
    - Le support d'exécution
  - Lire les manuels :
    - Les librairies standards.
    - Documentation développeur.

[Comment]
3. Identifier la cause.

– Exemple :

- ***Le Code :***

- `Char *str=g_strconcat(«ja»,«va»,«naise»);`

- ***Le Manuel :***

- « Concatenates all of the given strings into one long string. The returned string should be freed when no longer needed.

  Warning :

  The variable argument list must end with NULL. If you forget the NULL, g_strconcat() will start appending random memory junk to your string. »

[Comment]
3. Identifier la cause.

Dichotomie dans le programme :

- **Diviser … encore et encore :**
  - Point de départ : le programme démarre.
  - Point d'arrivée : le programme produit une erreur.
  - L'erreur survient elle avant ou après la moitié du programme ?
- **Raffiner** les critères définissant une erreur dans le programme.

...

Demander de l'aide lorsqu'on est coincé :

- Expliquer objectivement ce qui ne marche pas peut déclencher les idées
  - Ne pas présenter de conclusions (pour ne pas perturber la personne qui prend le temps de vous écouter....).
  - Accepter les remarques naïves.
- Avoir un point de vue nouveau sur votre code peut vous ouvrir les yeux.

...

[Comment]
4. Déterminer un correctif.

L'échelle de correctif peut aller de :

• **Simple.** • Ex:description originelle mal implémentée. → Réécrire une implémentation correcte.	• **Difficile.**	• **Impossible.** • Ex:défaut majeur dans le design imprégnant une grande partie du système → Nécessite une réécriture totale de l'application.

Corriger un bogue peut révéler / produire d'autres bogues.

- **CORRIGER UNE CHOSE A LA FOIS.**
- **REVENIR SUR LES CHANGEMENTS QUI N'ONT PAS D'EFFET.**

[Comment]
5. Appliquer le correctif.... et le tester.

2 objectifs au test :

- Le correctif aboutit à corriger le bogue initial.
- Pas d'effet de bord.

Penser au test de régression :

Une série de test pour évaluer le système :

- Vérifier l'exécution du système après des changements significatifs ou même de simple correctifs.
- Nouvelle fonction(s) = Test(s) Supplémentaire(s)

[Mise En Application]
Revue d'outils

- Prenons un code .... en fortran
- Sur un problème de produit vecteur / matrices creuses[1]. [*SPMXV*]

[1] : www.euroben.nl

[Mise En Application] Dysfonctionnements courants.

- Segmentation Fault :

  La plus fréquement rencontrée … A la compilation, pas d'erreur ....
- Mais aussi : fuite mémoire, dépassement tampon, pile etc...
- Compiler avec **l'option -g** et laisser faire **gdb**, **ddd, kgdb**… *ou à défaut utiliser les options de compilation type -fbacktrace (fortran...) pour générer des messages à l'exécution lorsque l'erreur est produite à l'exécution.*
- Parlons de gdb ... L'option -g produit de l'information de debugage dans le format natif de l'OS.... ensuite il n'y a quasiment plus qu'à laisser faire …
- .... et de valgrind.

[Mise En Application] Dysfonctionnements courants.

- Gdb et valgrind sont des alliés importants dans la recherche de bogues.... mais d'autres outils plus puissants comme ddt peuvent également apporter quelques avantages.
  - Exemple 1
  - Exemple 2

[En parallèle]
What else ?

- **Pour Allinea :**

  « Both serial and parallel debuggers are useful. Serial debuggers are what most programmers are used to (e.g., gdb), while parallel debuggers can attach to all the individual processes in an MPI job simultaneously, treating the MPI application as a single entity. This can be an extremely powerful abstraction, allowing the user to control every aspect of the MPI job, manually replicate race conditions, etc. »

- **Pour Sgi :**

  « Debugging a multiprocessed program is much more difficult than debugging a single-processor program. »

- **Plus sûrement ...**

[En parallèle]
Spécificités.

- Le parallélisme introduit surtout de la communication et de l'écriture de données.
- Au niveau du calcul rien ne diffère *vraiment* d'un code séquentiel …
- Le vrai problème … c'est que ça communique ... que c'est distribué … et que la mémoire est partagée … ou pas … parfois c'est même sur des grilles …

  Example : race condition

[En parallèle]
Des méthodes différentes ?

Conseils d'Allinea, comparés aux principes généraux :

«

I. Compare all the processes in the program to see which differences they exhibit. -Reconnaître l'existence d'un bogue (1)-

II. Group the processes by the results – for example into “broken” and “normal” groups. -Isoler la source du bogue (2)-

III. Analyze examples from each group in detail using existing, well-developed tools for examining single processes. -Isoler la source du bogue (2)-

IV. Understand the cause of the differences in the groups, and thus the cause of the problem. -Identifier la cause du bogue (3)-

»

[Pistes à suivre]
Peut – être ...

- Les conseils étaient :
  - Corriger une chose à la fois
    - Supposons qu'on travaille en équipe ....
    - Dans ces conditions que faire ? Ou le petit traité du « débogage parallèle » [1]...
  - Revenir sur les changements qui n'ont pas d'effet
    - Cela n'a pas de raisons d'être revu.
- Les tests :
  - Travailler en équipe sur des problèmes « orthogonaux » .

[1] : Debugging in Parallel *James A. Jones, James F. Bowring, Mary Jean Harrold*

[International Symposium on Software Testing and Analysis] *Proceedings of the 2007 international symposium on Software testing and analysis*

[Fin]

- *Questions ?*