

Desenvolvimento Aberto

```
4  #include "utilities.h"
5
6  Game game;
7
8  int main(int argc, char **argv) {
9
10     game = Game();
11
12     game.initialize();
13 }
```

Dicas de Debug

Igor dos Santos Montagner (igorsm1@insper.edu.br)

Minha primeira contribuição de código

Fix crash when creating new window from tab drop #1100

[Edit](#)[Open with ▾](#)[Merged](#)[jeremypw](#) merged 1 commit into [elementary:master](#) from [igordsm:fix-new-window-from-tab](#) 20 days ago[Conversation](#) 1[Commits](#) 1[Checks](#) 4[Files changed](#) 1

+1 -1

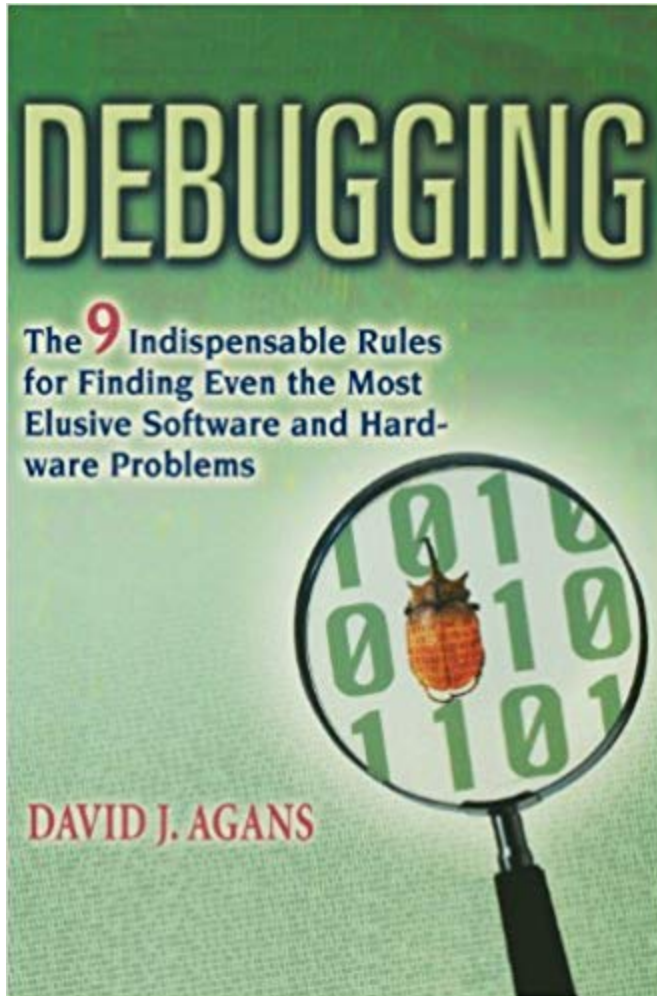
[Changes from all commits ▾](#) [File filter ▾](#) [Conversations ▾](#) [Jump to ▾](#) [⚙️ ▾](#)

0 / 1 files viewed

[Review changes ▾](#)[2](#) [src/Widgets/DocumentView.vala](#) [📄](#)☐ Viewed

```
@@ -288,7 +288,7 @@ public class Scratch.Widgets.DocumentView : Granite.Widgets.DynamicNotebook {
288 288     }
289 289
290 290     private bool find_unique_path (File f1, File f2, out string? path1, out string? path2) {
291 -     if (f1 == f2) {
291 +     if (f1.equal (f2)) {
292 292         path1 = null;
293 293         path2 = null;
294 294         return false;
```

Dicas para debugar (qualquer coisa)



9 regras de debug

1. UNDERSTAND THE SYSTEM
2. MAKE IT FAIL
3. QUIT THINKING AND LOOK
4. DIVIDE AND CONQUER
5. CHANGE ONE THING AT A TIME
6. KEEP AN AUDIT TRAIL
7. CHECK THE PLUG
8. GET A FRESH VIEW
9. IF YOU DIDN'T FIX IT, IT AIN'T FIXED

1. Understand the system

Nada acontece se não conseguirmos

1. Baixar a versão de desenvolvimento
2. Instalar todas as dependências
3. Compilar nossa própria versão
4. Rodar testes na versão do `master`

1. Understand the system

Ferramentas importantes:

1. [Virtual environments](#): venv, virtualenv, pipenv, conda
2. [setuptools](#) - *development mode*
3. Debugging
 - *pdb*
 - função `breakpoint()` ([tutorial](#))

1. Understand the system

Conheça suas ferramentas

- Debugging visual linha a linha
- Stacktrace
- Busca em projeto inteiro
- Ir para definição de função

Debugar usando `print` é perda de tempo

2. Make it fail

Reproduza seu bug

- Crie um exemplo mínimo que reproduza o bug desejado ou que usaria a feature que quer implementar
 - entrada de exemplo e saída esperada vs saída obtida
- Possivelmente isto já foi descrito na *issue* escolhida.
- Anote elementos que te ajudem a encontrar onde isto ocorre no código (funções usadas, elementos de interface gráfica, mensagens de erro.)

3. Quit thinking and look

Encontre onde está o problema e leia o código com atenção

1. Encontre no código onde o bug pode estar
2. Comece geral (em qual arquivo está a funcionalidade?) e vá restringindo (em qual função o bug "explode"?)
3. Ferramentas de debug são essenciais. Veja o item 1.
 - Não atenderei ninguém que esteja debugando só com `print`

3. Quit thinking and look

Buscando por nomes de arquivos

Comando `find` ([man page](#))

Exemplo: procurar por arquivos cujo nome é aceito por uma certa expressão regular começando no diretório atual.

```
$ find -iname "regexp" .
```

3. Quit thinking and look

Buscando no conteúdo dos arquivos

Comando `grep` ([man page](#))

```
$ grep [OPTIONS] PATTERN FILES
```

- `PATTERN` : expressão regular
- `FILES` : lista de diretórios ou arquivos

3. Quit thinking and look - grep

Exemplo 1: buscar todos arquivos nas pasta atual (.) e subpastas com o texto "dialog" ignorando maiúsculas/minúsculas.

```
$ grep -r -i dialog .
```

Exemplo 2: Listas todos os arquivos *.cpp* que fazem algum include

```
$ grep -r --include "*.cpp" "#include" .
```

Sua IDE/editor devem ter algo parecido. Procure e use.

4. Divide and Conquer

Crie um plano de ação

1. Por que o bug ocorre?
 - Está relacionado a qual função?
 - Qual variável tem o valor errado?
2. O que deve ser mudado para que pare de ocorrer?

Pré-requisito: debugar visualmente usando alguma IDE / editor

5. Change one thing at a time

Um bom PR muda o mínimo possível

- use várias branches se quiser testar ideias diferentes
- `git commit` é grátis. Quando chegar na versão final é só juntar tudo e mandar.
- `git rebase` para atualizar seu branch com o `master upstream` caso necessário.

6. Keep an audit trail

Registre suas descobertas

- Log de como você encontrou onde mexer
- Log de todos arquivos de interesse e seus usos
- Log de todas as pesquisas feitas
- Log de todas as modificações feitas
- `git commit` é útil para registrar testes também

Não se esqueça: `git diff` é seu melhor amigo

7. Check the plug

Nunca se esqueça de testar a soluções mais simples primeiro

8. Get a fresh view

Empacou?

É por isso que vocês trabalham em grupo no primeiro bug.

Empacou mesmo?

Para isto serve seu grupo.

Empacou mesmo?!?!?!?

Me chame!

9. If you didn't fix it, it ain't fixed



Hora de programar

Tentem aplicar estas ideias aos seus problemas

Hacktober 20 fest 21

Envie 4 PRs para projetos participantes e ganhe uma camiseta

Atividade extra: hacktoberfest



Objetivo: Enviar 3 PRs para projetos externos no mês de outubro.

```
"metadata": {"date": "2021-10-31"}
```

Desenvolvimento Aberto

```
4  #include "utilities.h"
5
6  Game game;
7
8  int main(int argc, char **argv) {
9
10     game = Game();
11
12     game.initialize();
13 }
```

Dicas de debug

Igor dos Santos Montagner (igorsm1@insper.edu.br)