DebugApp : Débogage d'une application avec Project Builder

Version française

http://trad.applescript.free.fr/

Préambule

Ce tutoriel n'est absolument pas une traduction officielle de la Société Apple.

Ce tutoriel est un essai de traduction française de "DebugApp:Debugging an Application With Project Builder", documentation technique Apple.

Dans l'espoir que cette version française comblera l'attente de tous les utilisateurs francophones, je vous souhaite une bonne lecture.

Un utilisateur Mac francophone

Merci à Daniel et Laurent pour leur aide et leurs conseils indispensables.

DebugApp : Débogage d’une application avec Project Builder

Ce tutoriel montre comment déboguer un projet avec Project Builder. Dans ce tutoriel, vous allez construire un projet de débogage, régler des points d’arrêt, exécuter pas à pas le code et examiner les données. Vous allez aussi apprendre comment entrer des commandes dans GDB, l’outil de commandes en ligne, utilisé par Project Builder comme debugger.

Ce tutoriel suppose quelques connaissances en programmation Mac OS X et d’avoir déjà utilisé un autre logiciel de débogage.

Ce tutoriel comporte les parties suivantes :

1. “Construire une application débogable” (page 3)
2. “Régler un point d’arrêt” (page 4)
3. “Lancer le débogage d’une application” (page 5)
4. “Exécution du code pas à pas” (page 7)
5. “Examen des données” (page 9)
6. “Utilisation des commandes en ligne de GDB” (page 10)
7. “Arrêt de Debugger” (page 13)

Construire une application débogable

Dans cette section, vous allez créer un nouveau projet et activer l’option pour la production des informations de débogage.

1. Créez un nouveau projet.

   Choisissez File > New Project. Sélectionnez Carbon Application, et cliquez sur Next. Saisissez DebugApp pour le nom du projet, choisissez un emplacement pour

l'enregistrement, puis cliquez sur Finish. Project Builder crée alors un nouveau projet et ouvre sa fenêtre. Le projet contient déjà des fichiers modèles que vous pouvez compiler et exécuter tel quel.

2. Cliquez sur l'onglet Targets situé sur le côté droit de la liste des fichiers, sélectionnez DebugApp, puis cliquez sur l'onglet Build situé juste en haut de la fenêtre affichant le code. En cliquant sur ce dernier onglet, les réglages des options de construction devraient apparaître.

3. Vérifiez que l'option "Generate debugging symbols" est activée. Si vous ne pouvez pas visualiser cette option, cliquez sur le triangle situé à côté des réglages du compileur.

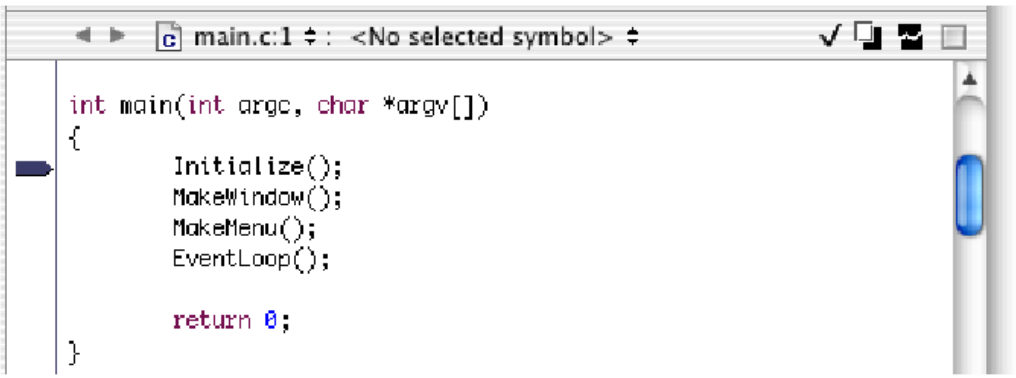
   À présent, lorsque vous construirez l'application, Project Builder y inclura les informations de débogage.

4. Cliquez sur le bouton Build (le marteau) situé dans la barre d'outil de la fenêtre du projet.

Régler un point d'arrêt

Vous pouvez régler les points d'arrêt directement dans Project Builder, même avant le lancement de Debugger. Project Builder sauvegarde les points d'arrêt lors de la fermeture du projet, de cette façon, vous les retrouvez à la prochaine ouverture.

Cliquez sur l'onglet Files, ouvrez `main.c`, faites défiler la fenêtre jusqu'à ce que vous voyez la fonction `main`. Cliquez dans la marge à côté de l'instruction `Initialize`, une flèche apparaît au niveau de cette instruction.

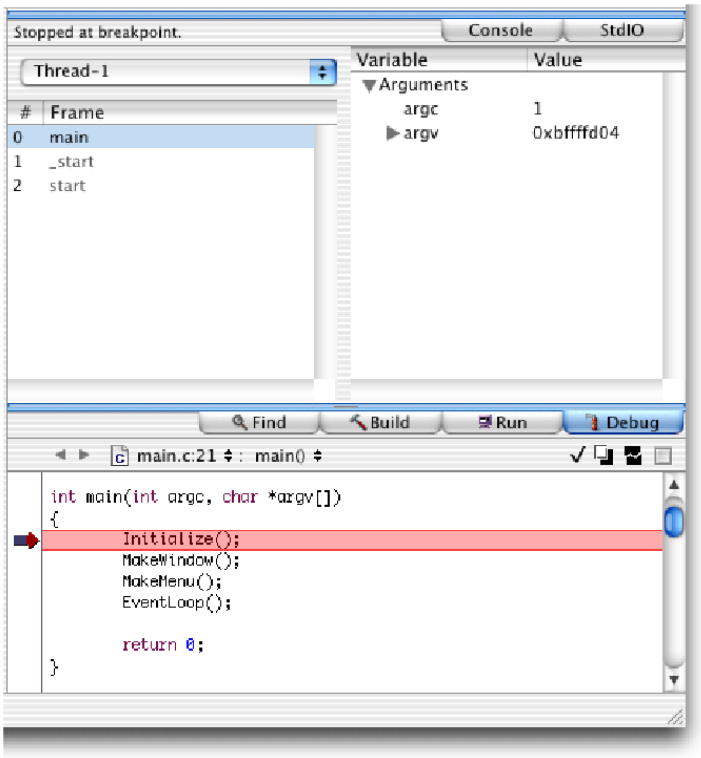


En cliquant sur l'onglet Breakpoints, vous pourrez visualiser la liste des points d'arrêt. Depuis ce panneau, vous pouvez visualiser le code source pointé par un point d'arrêt, activer, désactiver ou supprimer le ou les points d'arrêt.

Lancer le débogage d'une application

Cliquez sur le bouton "Build et Debug" (une bombe aérosol devant un marteau) dans la barre d'outil.

Le panneau de débogage apparaît dans la partie haute de la fenêtre du projet. Project Builder lance l'exécution de l'application sous le panneau Debug et s'arrête au point d'arrêt. L'instruction désignée par le point d'arrêt est affichée dans l'éditeur de code, elle est surlignée et une flèche rouge apparaît dans la marge gauche.



Les boutons qui contrôlent les opérations de Debugger sont situés dans la barre d'outil à droite. Vous les utiliserez dans "Exécution du code pas à pas" (page 7).

Pause suspend l'application et affiche l'instruction en cours d'exécution dans l'éditeur.

Continue reprend l'exécution d'une application suspendue.

Step Over exécute l'instruction suivante à l'intérieur de la même fonction.

Step Into exécute la première instruction de la fonction suivante si son code est disponible. Step Into passe à la fonction suivante seulement si elle existe, dans le cas contraire, il ne se passe rien.

Step Out exécute toutes les instructions suivantes de la fonction en cours, puis revient à la fonction qui a appelé cette fonction et se positionne, dans la fonction appelante, sur l'instruction suivant immédiatement celle de l'appel.

Pour activer GDB et pouvoir saisir des commandes en ligne, il vous suffit de cliquer sur l'onglet Console situé en haut du panneau Debug. Dans l'interface de Project Builder, vous gèrerez plus de tâches, mais pour des tâches plus pointues, vous utiliserez GDB. GDB sera utilisé dans "Utilisation des commandes en ligne de GDB" (page 10).

En dessous du panneau Debug, vous trouverez deux listes de fichiers. Ces listes indiquent, respectivement à droite et à gauche, la fonction et l'instruction en cours. Vous les utiliserez dans "Examen des données" (page 9).

- Le menu déroulant Threads affiche tous les threads de l'application. Dans ce tutoriel, un seul thread exécute le code source de l'application, les autres sont des threads systèmes qui gèrent les communications inter-applications et le Debugger. Sélectionner un thread provoque l'affichage de sa chaîne d'appel dans la liste des frames.

- La liste des frames affiche la chaîne d'appel du thread sélectionné. Cliquer sur un frame provoque, l'affichage de ses variables dans la liste des variables, et une flèche rouge apparaît, dans l'éditeur de code, à côté de son instruction généralement exécutée.

- La liste des variables affiche toutes les variables locales et globales visibles dans le frame sélectionné.

Exécution du code pas à pas

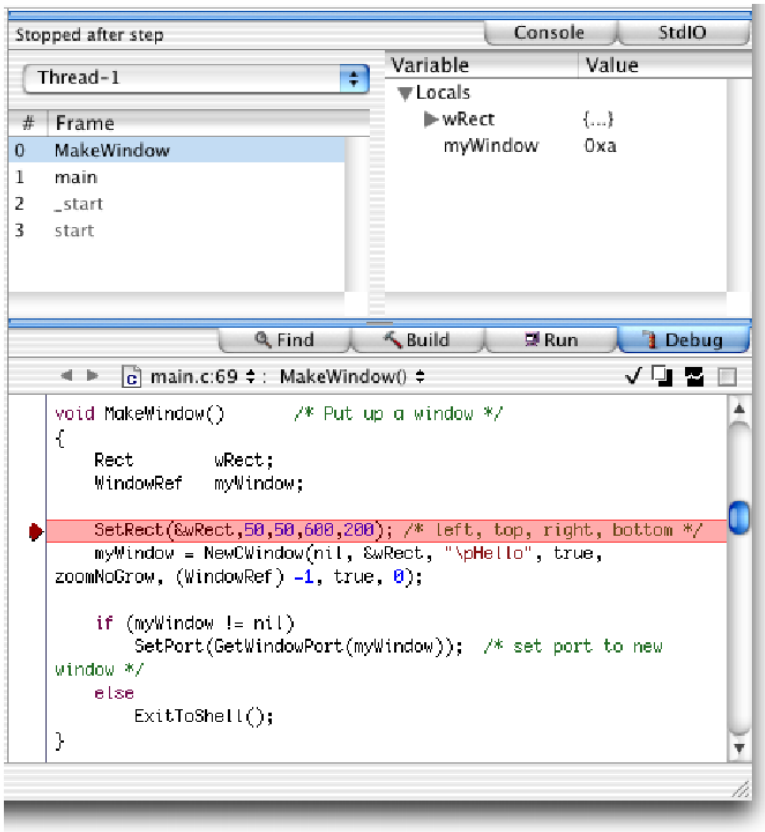
À présent, vous allez utiliser les boutons Step Over et Step Into pour regarder l'application s'exécuter.

1. Appliquez Step Over à la fonction `Initialize`.

Project Builder exécute la fonction `Initialize` et s'arrête à l'instruction suivante `MakeWindow`.

2. Cliquez à présent sur le bouton Step Into.

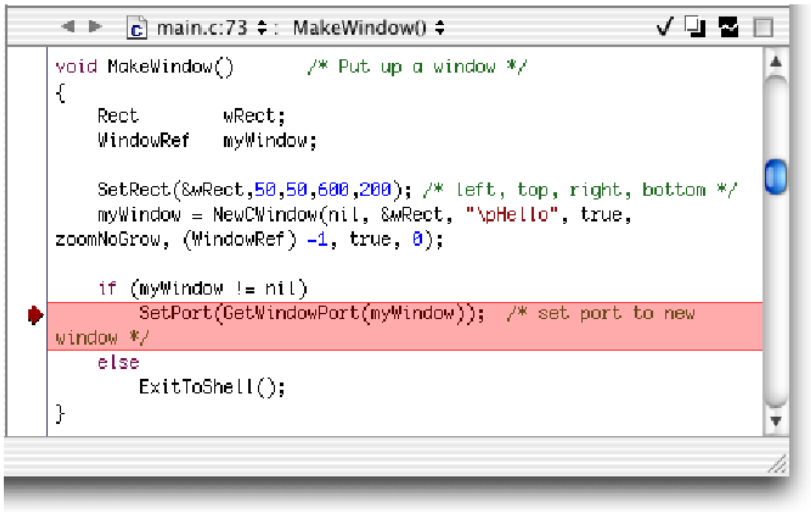
Project Builder affiche alors la première instruction de `MakeWindow` dans l'éditeur de code, et ses variables dans la liste des variables.



3. Regardons le sauter à l'instruction `if`.

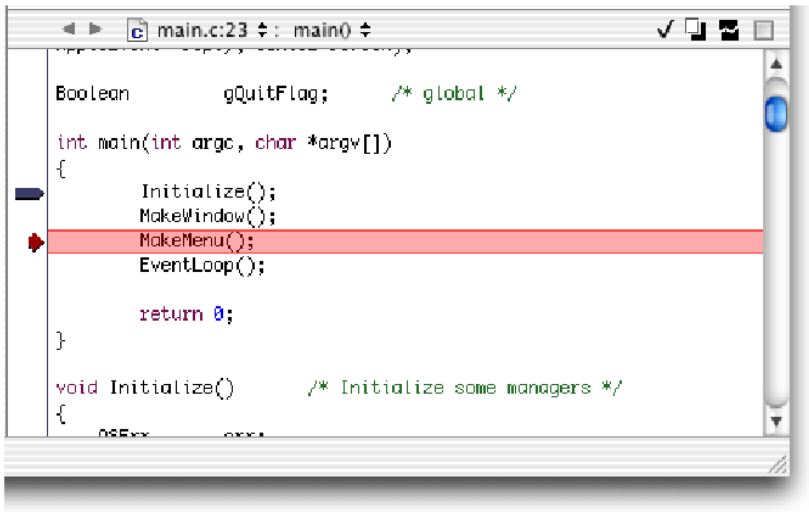
Appuyez plusieurs fois, soit sur Step Into, soit sur Step Over, jusqu'à ce que la flèche rouge soit à côté de l'instruction `if`. Notez que lorsque vous exécutez une fonction dont vous n'avez pas le code source (comme une fonction Carbon), les boutons Step Into et Step Over ont la même fonction.

Appuyez de nouveau sur Step Into ou Step Over. Project Builder évalue alors l'instruction `if` et va à l'instruction `SetPort`.



4. Regardons le revenir à la fonction `main`.

Appuyez sur Step Out. La flèche rouge retourne à la fonction `main` et se positionne à côté de l'instruction `MakeMenu`.



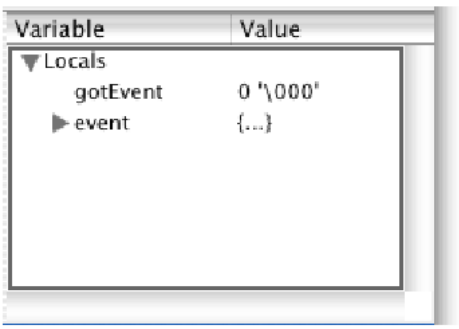
Examen des données

Dans cette section, vous allez examiner les données de l'application avec Project Builder.

1. Appliquons Step Into à `EventLoop`.

   Appliquez Step Over à `MakeMenu` puis Step Into à `EventLoop`. Vous devriez obtenir le résultat suivant :

   À présent, regardons la liste des variables :

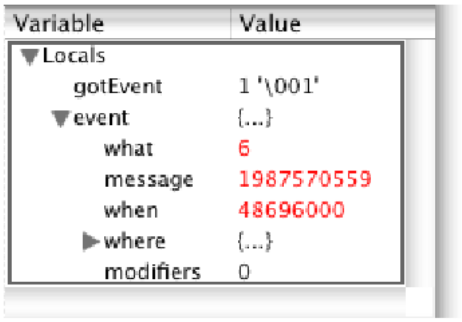


   Elle n'a qu'une section, Locals, laquelle liste toutes les variables locales de la fonction `EventLoop` : la variable booléenne `gotEvent` et la structure `event`. Si cette fonction avait eu des arguments, ils y aurait une section Arguments.

2. Appliquons Step Over à l'instruction `WaitNextEvent` et examinons l'enregistrement `event`.

   Appliquez Step Over à `WaitNextEvent`. Puis cliquez sur le triangle à côté de `event` pour

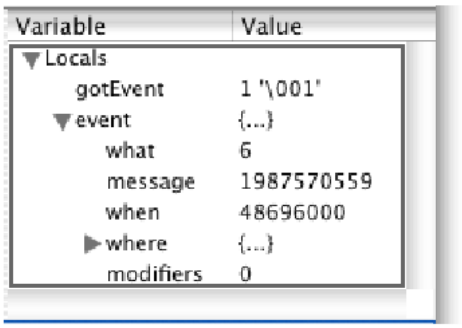
voir son contenu :



Les valeurs sont toutes en rouge, indiquant qu'elles ont été modifiées depuis que la dernière instruction a tourné. Si vous le voulez, vous pouvez aussi développer la structure `where` de la même manière.

3. Appliquez Step Over à l'instruction `if`.

Notez que les variables sont en noir, car cette fois, les valeurs sont restées inchangées.



4. Reprenons l'exécution de l'application.

Appuyez sur Continue.

Utilisation des commandes en ligne de GDB

Project Builder débogue votre programme en envoyant des commandes à GDB, un débogueur GNU. Project Builder fournit une interface graphique qui gère les caractéristiques basiques de GDB, mais pour des travaux plus complexes, vous pouvez saisir directement des commandes dans GDB.

Dans ce chapitre, vous allez utiliser GDB pour régler un point d'arrêt conditionnel, Project Builder ne s'arrêtera à ce point que si la condition spécifiée est vraie.

Notez que lorsque vous fermerez le projet, Project Builder sauvegardera vos points d'arrêt mais pas les conditions associées. Project Builder n'enregistrera rien de ce qui a été saisi directement dans GDB.

1. Suspendons l'exécution du programme.

   Appuyez sur le bouton Pause.

   Project Builder suspend le programme et s'arrête sur n'importe quelle instruction en cours d'exécution. Ce sera généralement sur l'instruction `WaitNextEvent` dans `EventLoop`, mais si ce n'est pas le cas, ce n'est pas très important.

2. Création d'un point d'arrêt pour que le programme soit suspendu lorsque vous appuyez sur une touche du clavier.

   Faites défiler la fenêtre jusqu'à la fonction `DoEvent`, puis identifiez l'instruction `case keyDown`. L'instruction à partir de laquelle, l'appui sur une touche est géré. Après, c'est une instruction qui règle la variable `key`. Comme nous voulons savoir quelle touche a été pressée, nous allons placer le point d'arrêt après cette instruction, au niveau de `if (event->modifiers & cmdKey)`.

3. Continuons l'exécution du programme.

   appuyez sur le bouton Continue.

4. Appuyons sur une touche dans l'application.

   Lancez l'application DebugApp en double-cliquant sur son icone dans le dock. L'application affiche alors sa fenêtre Hello. Appuyez sur n'importe quelle lettre du clavier.

   Project Builder passe alors à l'avant-plan, suspend l'application et affiche l'instruction sur laquelle pointe le point d'arrêt.

5. Trouvons le numéro du point d'arrêt.

   Tout en haut du panneau Debug, cliquez sur l'onglet Console.

   La console de Debugger, qui permet de saisir les commandes de GDB, s'affiche.

   Dans la console, saisissez `info br` pour afficher les informations concernant les points d'arrêt du programme. Vous devriez obtenir ce qui suit :

```
(gdb) info br
Number    Type        disp  Enb   Address     What
1         breakpoint  keep  y     0x00003254 in main at main.c:21
breakpoint already hit 1 time

2         breakpoint  keep  y     0x000037c4 in DoEvent at main.c:161
breakpoint already hit 1 time
```

   Notez que le numéro du point d'arrêt dans `DoEvent`, en tout cas ici, est 2.

6. Créons à présent un point d'arrêt conditionnel.

   Saisissez `cond <num> (key=='a')`, où `<num>` est le numéro du point d'arrêt. Par exemple :

```
(gdb) cond 2 (key=='a')
(gdb)
```

   Maintenant, Project Builder mettra sur pause à ce point d'arrêt uniquement si vous appuyez

sur la touche a. Vous pouvez le vérifier en saisissant `info br` de nouveu. Par exemple :

```
(gdb) info br
Number    Type        disp  Enb   Address     What
1         breakpoint  keep  y     0x00003254 in main at main.c:21
breakpoint already hit 1 time

2         breakpoint  keep  y     0x000037c4 in DoEvent at main.c:161
stop only if key == 97 'a'
breakpoint already hit 1 time
```

7. Reprenons l'exécution du programme.

   Appuyez sur le bouton Continue, puis cliquez sur l'icone de DebugApp dans le dock.

8. Appuyez sur q.

   L'exécution de l'application n'est pas suspendue.

9. Appuyez sur a.

   Project Builder passe à l'avant-plan, suspend l'application et affiche l'instruction sur laquelle pointe le point d'arrêt.

Arrêt de Debugger

Comme Mac OS X a une mémoire protégée, vous n'avez pas besoin de vous inquiéter sur les conséquences de quitter l'application directement depuis Debugger. En appuyant juste sur le bouton Stop, situé en haut à gauche du panneau Debug, Project Builder force l'application à quitter.

Voilà, vous avez fini !

À plus, pour de prochaines aventures ! comme dirait Daniel. :-))))