Préambule

Les sources (C/C++) des fichiers sont disponibles sur froggy dans le /scratch.

1. Connectez vous sur froggy avec ssh (-X pour récupérer l'environnement X) et mettez à jour l'environnement de développement (source /applis/site/env.bash ..., voir consignes)

2. Récupérez puis désarchivez les sources du TP : /scratch/PROJECTS/prformation-ced-2017/TPdebug.tar

Vous trouverez dans src les différents exemples de code sources qui devront être utilisés pour chaque partie du TP.

1 Premiers pas avec un debogueur - Recherche du contenu d'une variable

Lorsqu'on a une erreur dans un code, examiner le contenu des variables pendant l'exécution, est souvent la première méthode à utiliser.
Dans cette partie, il s'agira de comprendre la mise en oeuvre d'un débogueur, d'utiliser les commandes de base et de visualiser le contenu de variables pour déterminer l'erreur.
Quelques commandes de base de gdb, à tester : **help, info, run, break, step, next, dis, undisp,...**

Description:

Utilisez le source **variablePrinting.c**, programme simple qui :

- Alloue deux tableaux **array1** et **array2**,
- initialise **array1**,
- copie **array1** dans **array2**,
- passe **array2** à une fonction qui calcule le carré de chaque élément et stocke le résultat dans **array2**,
- calcule la différence entre les deux tableaux et l'imprime.

1. Créez l'exécutable en utilisant l'un des compilateurs GNU ou INTEL. Vous devrez mettre à jour l'environnement nécessaire à l'utilisation de ces compilateurs (commande module).

2. Exécutez le programme et observez le résultat, notamment la différence entre array1 et array2 qui ne devrait pas être nulle.

3. Corrigez les différents problèmes à l'aide du debogueur.

Méthodologie proposée : pour identifier l'erreur, on recompile le code en utilisant l'option **-g** et on l'analyse en utilisant **gdb**.
Imprimez le contenu de certaines variables durant l'exécution en utilisant les commandes **disp, undisp**.
"Visitez" (via gdb) la fonction **squareArray()** et observez que l'erreur ne vient pas de cette fonction.

2 Erreur d'indice dans un tableau

Lorsqu'on déclare la taille d'un tableau, on définit de façon implicite l'intervalle de variation des indices de ce tableau. Une erreur courante est de référencer l'élément d'un tableau en utilisant un indice en dehors de l'intervalle de variation défini. La définition implicite de cet intervalle de variation n'est pas la même suivant les langages ce qui génère beaucoup d'erreurs lorsqu'une application utilise plusieurs langages.

Dans ce TP, il s'agira de mettre en évidence une erreur d'indice, d'en comprendre les causes et de fournir une correction de cette erreur.

Description

Utilisez le source **arrayIndex.c**, programme simple qui remplit un tableau d'entiers (tock) dont la taille est donnée par le paramètre N. Ce tableau est ensuite ajouté à un autre tableau d'entiers (evensum ou oddsum, suivant la parité des indices).

Créez un programme avec le compilateurs GNU puis un autre avec INTEL. Exécutez le et observez les résultats dans chaque cas. Corrigez, à l'aide de gdb, les éventuelles erreurs.

3 Entrées-sorties

Les entrées sorties d'un code sont à l'origine de nombreux bugs, les plus courants étant les entrées/sorties formatées. Il est facile d'oublier de tenir compte des espaces dans des fichiers textes, de faire une erreur en définissant la précision d'un réel etc. On peut aussi essayer de lire plus de données qu'il n'y en a sur un enregistrement ou dans le fichier; certains compilateurs interprètent la fin d'un fichier comme une erreur, d'autres non.

Actuellement de nombreuses erreurs simples sont diagnostiquées de façon claire et automatique au runtime (i.e. à l'exécution). Toutefois, si ce n'est pas le cas, l'utilisation d'un débogueur est un peu plus compliquée que pour des problèmes d'assignation de variables parce que le processus réel de lecture ou d'écriture de données d'un fichier est géré par des appels système pour lequel il n'y a pas de code source à suivre.

Dans cette partie du TP, il s'agira de diagnostiquer des erreurs de lecture dans un fichier texte en utilisant à la fois les messages d'erreurs à l'exécution et un débogeur (gdb, DDT).

On pourra utiliser les commandes de base de gdb **help, info, run, break, step, next, dis, unzips,...** ou de façon plus simple l'interface graphique de DDT:

Description

Utilisez le source **ioBadCode.f90** et le fichier de données **sectionsCT.txt**. Le programme :

- Lit les données en format ASCII dans un fichier,
- effectue quelques manipulations sur ces données,
- écrit les résultats dans un fichier binaire,
- relit le fichier binaire et écrit quelques données dans un fichier texte.

1. Créer l'exécutable en utilisant l'un des compilateurs GNU ou INTEL. Exécuter le programme et observer les messages d'erreurs.

2. Utiliser un débogueur (gdb ou DDT) pour localiser les erreurs et les corriger.

Pour utiliser **DDT**, il est nécessaire de charger le module associé (allinea-ddt/xxx).

4 Passage de paramètres

Lorsque un sous-programme est appelé, la liste des paramètres dans le programme appelant doit correspondre à la liste des paramètres formels du sous-programme. Les erreurs les plus courantes sont:

- le nombre de paramètres dans l'appel du sous-programme ne correspond pas au nombre de paramètres formels.
- le type entre paramètres effectifs et paramètres formels ne correspond pas,
- les paramètres peuvent être passés par valeur ou par adresse, source d'erreur fréquente. (Particulièrement lorsqu'un programme C appelle un sous-programme fortran, les variables doivent être passées par adresse plutôt que par valeur).

Les compilateurs réagissent différemment à ces erreurs, certains affichent un warning, d'autres non. Pour les programmes C qui utilisent le prototypage et les programmes fortran90 qui déclarent les routines par des blocs interface, le compilateur est capable de vérifier la compatibilité entre le nombre des arguments et leur type dans les différents appels. Ce n'est pas le cas pour les programmes C qui n'utilisent pas le prototype et les programmes fortran qui ne déclarent pas un bloc interface pour chaque routine utilisée (fortran77), l'éditeur de liens peut créer un exécutable sans détecter l'erreur. Au runtime l'erreur se traduit soit par des résultats faux soit par des message du type "segmentation fault". dans ce cas, l'utilisation d'un débogueur peut s'avérer utile.

Dans ce TP, il s'agira de diagnostiquer des erreurs de passage de paramètres à l'appel d'une routine. On utilisera gdb ou de façon plus simple l'interface graphique de DDT. On observera la capacité du compilateur à détecter les erreurs.

Description

Utilisez les sources miss.c ou miss.f90.

Utilisez gdb et/ou DDT pour détecter et corriger les différentes erreurs, liées aux déclaration et au types des arguments de fonctions.

Si vous travaillez en C ou C++, i.e. avec miss.c, vous aurez besoin d'une fonction définie en fortran, dans foo.

Pour compiler du code C/C++ (dans main.c) qui appelle une fonction fortran définie dans fichier.f90, il faut utiliser la séquence suivante:

```
gfortran -c fichier.f90
# --> création d'un fichier.o, .mod ...
gcc ou g++ -lgfortran fichier.o main.c

# avec intel:
ifort -c fichier.f90
# --> création d'un fichier.o, .mod ...
icc ou icpc -lifcore fichier.o main.c
```

Inter-opérabilité C/Fortran, voir par exemple: *https://software.intel.com/en-us/node/678422*

Rappel: pour utiliser **DDT**, il sera nécessaire de charger le module associé (allinea-ddt/xxx).