POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S5) **Programmation Structurée 2011/2012**

http://laure.gonnord.org/pro/teaching/

TP7 - Pointeurs simples, traçage de programme avec ddd

La partie Linux

Tester toutes les commandes dans un terminal situé à la racine de votre compte.

Pour chercher les gros fichiers de notre compte (et éventuellement faire le ménage), on utilise (par exemple) :

```
find . -type f -size +10000000c -printf '%s %p\n'
```

(trouve tout les fichiers à partir de . de taille >10Mo et imprime quoi ?)

Ensuite, on récupère ce résultat et on demande de trier ce résultat (avec un "pipe", ie barre verticale)

```
find . -type f -size +10000000c -printf '%s %p\n' | sort -rnk1,1
```

(à quoi correspondent les arguments de sort ? man sort !)

Et enfin, on n'affiche que les 10 plus gros :

```
find . -type f -size +10000000c -printf '%s %p\n' | sort -rnk1,1 | head -10
```

►Retenir les commandes *find*, *sort*, et *head*, ainsi que le "pipe".

1 Tests de pointeurs

Exercice 1 *Écrire le programme suivant :*

```
int main()
{
   int a = 10;
   int b = 50;
   int* p;

    p=&a ;
    b=*p ;
    *p=42;

    return 0;
}
```

1. *Afficher après chaque instruction la valeur entière (printf avec %d) de : a,b, *p, et des adresses &a et p (printf avec %p). Vérifier que tout se passe comme prévu.*
2. *Vérifier qu'une compilation avec -Wall+ détecte les non initialisations de pointeurs.*

Exercice 2 *Écrire les deux exemples (transparents 28 et 29) du cours, les compiler et observer les erreurs à la compilation et à l'exécution.*

2 Pointeurs comme paramètres modifiables

On écrira un programme C COMPLET par exercice, et on testera, *evidemment*.

Exercice 3 *(source : http://diwww.epfl.ch/w3lsp/teaching/coursC/exercices/pointeurs.html Écrire un programme déclarant une variable entière v et l'initialisant. Déclarer un pointeur pv de type correspondant à cette variable et le faire pointer sur v. Déclarer un autre pointeur ppv de type approprié pour pouvoir pointer sur le pointeur pv. Faire pointer ppv sur pv. Modifier la valeur de v indirectement en utilisant le pointeur ppv, puis vérifier en imprimant v.*

Exercice 4 *Écrire une procédure* `logint` *qui prend en paramètre deux entiers n et p, et retourne la plus grande puissance q de p dans n. Elle calcule en outre le coefficient multiplicateur d et le reste r tels que* $n = d \times p^q + r$ *avec* $r < p^q$ *et* $d < p$. *r et d seront passés par adresse. Il est interdit d'utiliser la fonction* `log`. *On testera à l'aide des égalités* $27 = 1 \times 2^4 + 11$, $98 = 9 \times 10^1 + 8$.

3 Utilisation de gdb/ddd pour tracer des programmes

Gdb est un débugger : c'est un outil d'inspection de programmes écrits en C, C++, ... Il est beaucoup utilisé lors du développement de programmes, afin de trouver les bugs. Ici nous allons prendre contact avec gdb via une interface graphique (laide, rudimentaire, mais utile), nommée *ddd*.

Exercice 5 *Reprendre le programme de l'exercice 1, le copier en* `exo-ddd.c`.

1. *Compiler ce programme avec l'option -g de gcc (obligatoire pour utiliser gdb ou ddd) :*

   ```
   gcc -o exoddd exo-ddd.c -g
   ```

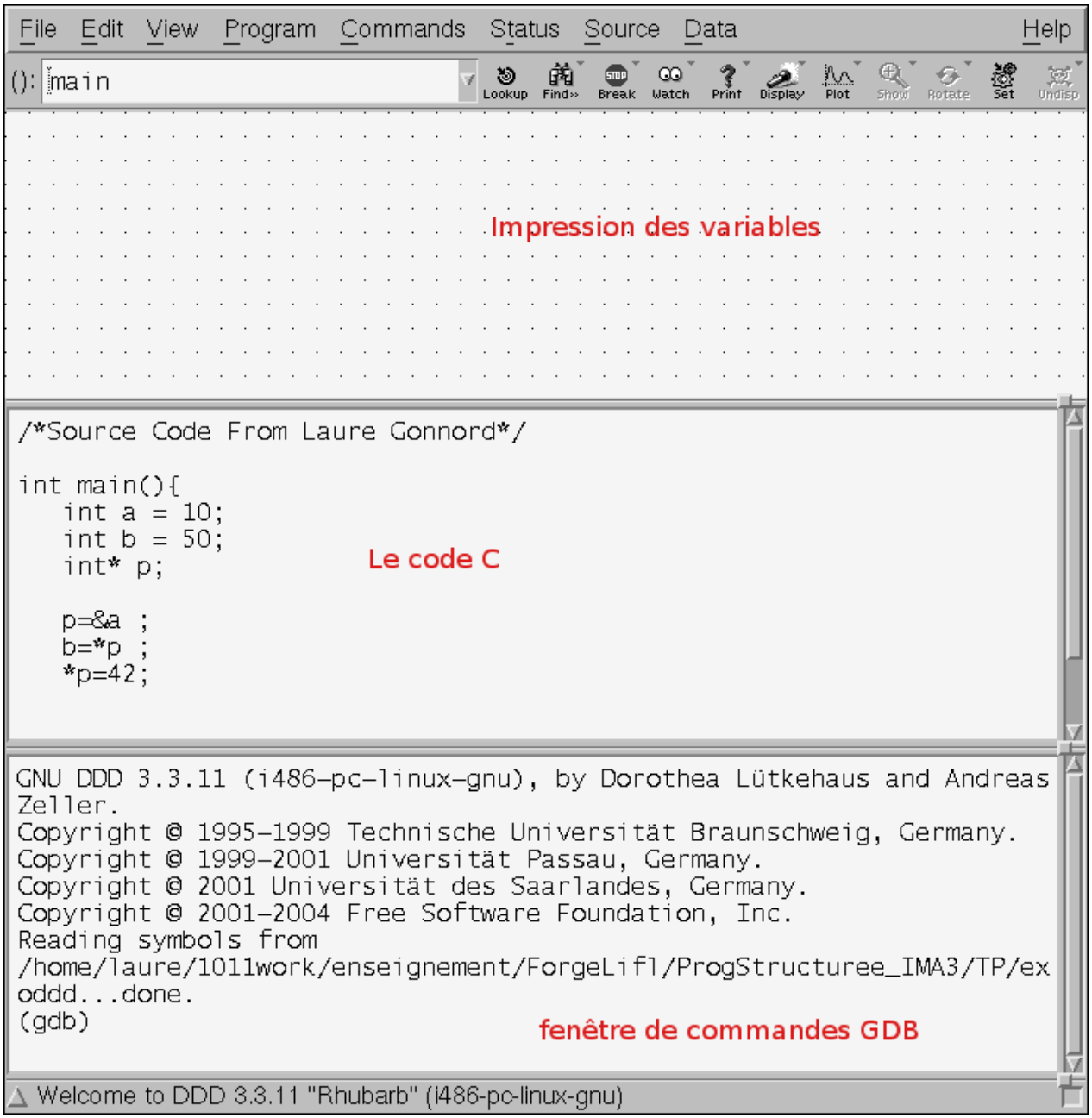
2. *Vérifier qu'il s'exécute toujours.*
3. *Lancer ddd, en remarquant qu'on lance sur le binaire :*

   ```
   ddd exoddd &
   ```

4. *Observer l'environnement en se reportant à la feuille annexe.*
5. *Mettre des points d'arrêts aux endroits suivants : ligne après les déclarations, puis après chaque instruction.*
6. *Lancer ddd : run dans la fenêtre du bas.*
7. *Afficher (display) au premier point d'arrêt les variables a, b, p,* $*p$.
8. *Pour passer d'un point d'arrêt à un autre, taper next (ou n) dans la fenêtre de commandes gdb. Observer la fenêtre du haut lors de cette commande.*

Exercice 6 *Utiliser ddd pour observer le comportement des variables de vos deux programmes de la section 2.*

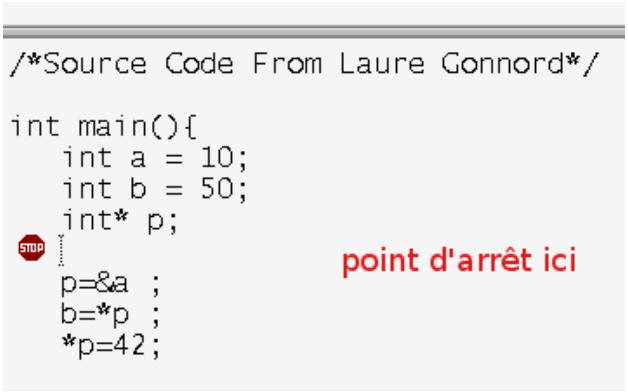
Fenêtre Initiale GDB Après lancement de ddd sur le binaire, on obtient une fenêtre :



Exécution du code Dans la fenêtre de commandes gdb :

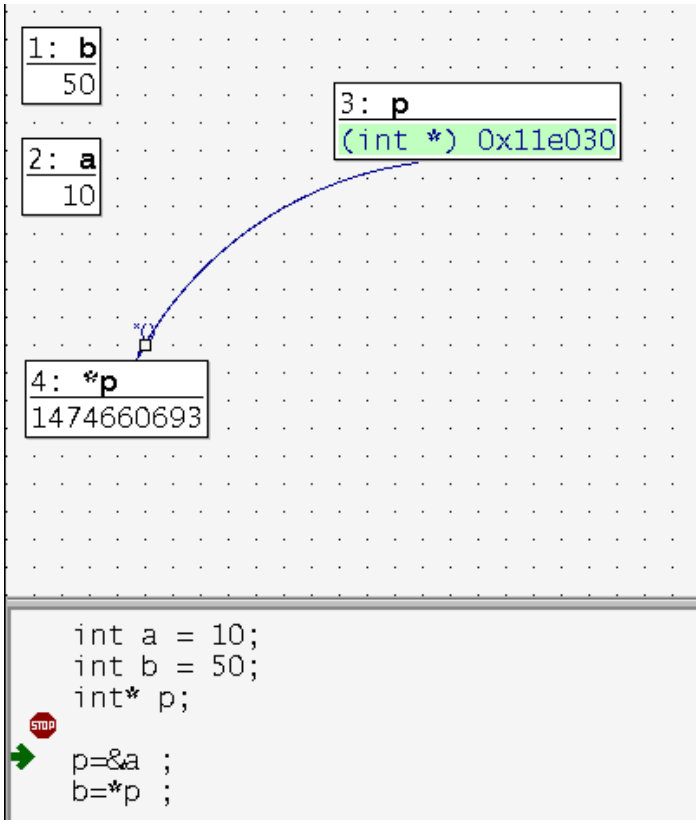
```
(gdb) run (+entrée)
```

BreakPoints Pour stopper l'exécution à un endroit précis du code, on met un point d'arrêt : clic dans le code à l'endroit en question, puis clic droit, set breakpoint.



Si il y a des breakpoints, l'exécution s'arrête au premier rencontré. On est alors capable d'afficher les variables.

Display En cours d'exécution, si on est arrêté au milieu du programme, on peut afficher les variables disponibles à cet instant, à l'aide de display. **attention, pas de clic droit dans la fenêtre du haut !** Pour le point d'arrêt montré plus haut, on peut afficher a, b, p et $*p$ (click droit sur la variable dans la fenêtre de code, et display). On obtient alors des jolis dessins dans la fenêtre du haut.



Remarque : si on double-clic sur une adresse dans la fenêtre du haut, le pointeur se "déroule".