UNIVERSITÉ
de Cergy-Pontoise
Département des Sciences Informatiques

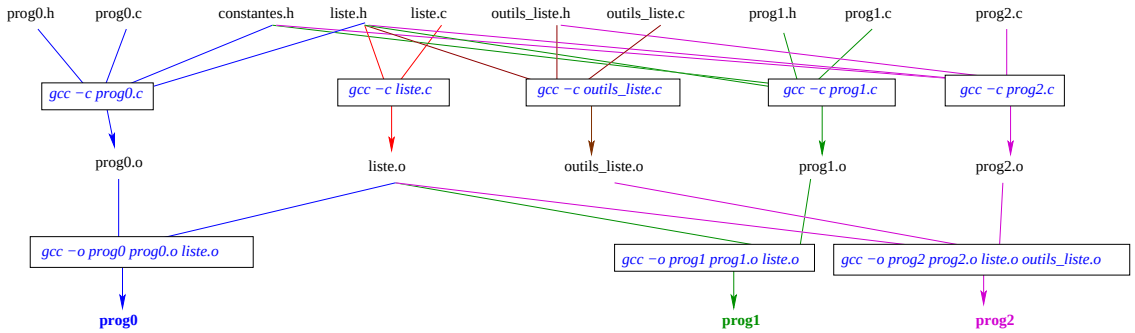
Licence 2 I/MI
2008–2009
Programmation Système
T.T. Dang Ngoc
dntt@u-cergy.fr

TD 5 - Compilation et débogage (correction)

I Compilation

Rappel

1. Pretraitement : `.c > .i (code C traité)`
   `Traites les #include, #define`
   `cpp truc.c > truc.i`
   ou
   `gcc -E truc.c -o truc.i`
2. Compilation : `.i > .s (code assembleur)` analyse syntaxique, typage, generation du code assembleur
   `/usr/lib/gcc/i486-linux-gnu/4.3/cc1 truc.i > truc.s 2> /dev/null`
   ou
   `gcc -S truc.i -o truc.s`
   `gcc -S truc.c -o truc.s`
3. Assemblage : `.s > .o (fichier objet)`
   `as truc.s > truc.o`
   ou
   `gcc -c truc.s -o truc.o`
   `gcc -c truc.c -o truc.o`
4. Edition de liens : `.o > executable` Ajout des bibliothèques C
   `ld -o executable truc.o xxxxx.o-lc -e main`
   `-lc : ajout bib C`
   `-e main : point d'entrée`
   ou
   `gcc truc.o -o truc`




```
#
# 2006/04/04 dntt : Makefile très basique
#

all : prog0 prog1 prog2

liste.o : liste.h liste.c
gcc -c -g liste.c

outils_liste.o : outils_liste.h outils_liste.c liste.h
gcc -c -g outils_liste.c

prog0.o : constantes.h prog0.h prog0.c liste.h
gcc -c -g prog0.c

prog1.o : constantes.h prog1.h prog1.c liste.h
gcc -c -g prog1.c

prog2.o : constantes.h prog2.c liste.h outils_liste.h
gcc -c -g prog2.c

prog0 : prog0.o liste.o
gcc prog0.o liste.o -o prog0

prog1 : prog1.o liste.o
gcc prog1.o liste.o -o prog1

prog2 : prog2.o liste.o outils_liste.o
gcc prog2.o liste.o outils_liste.o -o prog2

clean :
rm -f liste.o outils_liste.o prog0.o prog1.o prog2.o prog0 prog1 prog2
```

II Débogage

prog0

lancer prog0 :

```
$ ./prog0
```

```
Contenu de la liste : [four - three - two - one - ]
$
```

OK, pas de problème

prog0

lancer prog1 :

```
$ ./prog1
Contenu de la liste : [four - three - two - one - ]
Position de one : 3
Position de four : 0
Memory fault
$
```

Apparement, il y a une erreur mémoire.

1. ulimit -c 1000 pour activer la création des coredump
2. Compiler prog1 avec l'option -g
3. Lancer le programme, un fichier core est généré
4. Utiliser gdb ou ddd pour l'analyse post-mortem :
   - Avec gdb, utiliser la commande bt pour remonter la pile d'exécution
     ```
     $ gdb prog1 core
      Core was generated by `./prog1'.
     Program terminated with signal 11, Segmentation fault.
     Reading symbols from /lib/libc.so.6...done.
     Loaded symbols for /lib/libc.so.6
     Reading symbols from /lib/ld-linux.so.2...done.
     Loaded symbols for /lib/ld-linux.so.2
     #0  0x080484a8 in rechercher (l=0x8049928, nom=0x804875d "trois") at liste.c:40
     40                      if (!strcmp (courant->nom, nom)) {
     (gdb) bt
     #0  0x080484a8 in rechercher (l=0x8049928, nom=0x804875d "trois") at liste.c:40
     #1  0x080485b7 in main () at prog1.c:17
     (gdb) quit
     $
     ```
   - avec ddd, lancer : `ddd prog1 core` Puis sélectionner `Status -> Backtrace`. On voit aussi que liste.c ligne 40 appelée par prog1.c :17 est incriminée.
5. En relisant bien le code à la ligne liste.c : 40 on se dit qu'effectivement, si le nom cherché n'est pas dans la liste, on sort de la liste et on pointe n'importe où. Il faut donc remplacer
   ```
   while (1)
   par
   while (courant != NULL)
   ```

prog2

1. Compiler prog2 avec l'option -g

2. Lancer le programme, le programme tourne indéfiniment : il y a un problème.
3. Avec gdb ou dd suivre pas à pas le déroulement du programme
4. On voit à la ligne 20 du fichier `outils_liste.c`, qu'il faut remplacer le `courant = l->suivant` par `courant = courant->suivant`

Makefile avec option de debugage

```
DEBUG=-g

all : prog0 prog1 prog2

liste.o : liste.h liste.c
gcc ${DEBUG} -c liste.c

outils_liste.o : outils_liste.h outils_liste.c liste.h
gcc -c ${DEBUG} outils_liste.c

prog0.o : constantes.h prog0.h prog0.c liste.h
gcc -c ${DEBUG} prog0.c

prog1.o : constantes.h prog1.h prog1.c liste.h
gcc -c ${DEBUG} prog1.c

prog2.o : constantes.h prog2.c liste.h outils_liste.h
gcc -c ${DEBUG} prog2.c

prog0 : prog0.o liste.o
gcc prog0.o liste.o -o prog0

prog1 : prog1.o liste.o
gcc prog1.o liste.o -o prog1

prog2 : prog2.o liste.o outils_liste.o
gcc prog2.o liste.o outils_liste.o -o prog2

clean :
rm -f liste.o outils_liste.o prog0.o prog1.o prog2.o prog0 prog1 prog2
```

III Bibliothèques

statique

```
$ gcc -c liste.c outils_liste.c
$ ar cr libliste.a liste.o outils_liste.o
# creation de l'index
$ ranlib libliste.a
$ gcc -static -o prog0 prog0.o -L. -lliste # -IRepinclude
$ ./prog0


# Montrer liste objet :
```

```
$ ar t libliste.a
liste.o
outils_liste.o
#Montrer symboles :
$ nm -s libliste.a
Archive index:
afficher in liste.o
ajouter in liste.o
rechercher in liste.o
supprimer in outils_liste.o

liste.o:
00000000 T afficher
0000004c T ajouter
         U malloc
         U printf
         U puts
00000095 T rechercher
         U strcmp

outils_liste.o:
         U perror
00000000 T supprimer
```

Makefile avec bibliothèque statique :

```
DEBUG='-g'

all : prog0 prog1 prog2

liste.o : liste.h liste.c
gcc -c ${DEBUG} liste.c

outils_liste.o : outils_liste.h outils_liste.c
gcc -c ${DEBUG} outils_liste.c

libliste.a : liste.o outils_liste.o
ar cr libliste.a liste.o outils_liste.o
ranlib libliste.a

prog0.o : constantes.h prog0.h prog0.c liste.h
gcc -c ${DEBUG} prog0.c

prog1.o : constantes.h prog1.h prog1.c liste.h
gcc -c ${DEBUG} prog1.c

prog2.o : constantes.h prog2.c liste.h outils_liste.h
gcc -c ${DEBUG} prog2.c

prog0 : prog0.o libliste.a
gcc -static -o prog0 prog0.o -L. -lliste

prog1 : prog1.o libliste.a
gcc -static -o prog1 prog1.o -L. -lliste

prog2 : prog2.o libliste.a
```

```
gcc -static -o prog2 prog2.o -L. -lliste

clean :
rm -f liste.o outils_liste.o prog0.o prog1.o prog2.o \
prog0 prog1 prog2 libliste.a
```

dynamique

```
$ gcc -c liste.c outils_liste.c
$ gcc -shared -o libliste.so liste.o outils_liste.o

$ gcc  -o prog0  prog0.o -L. -lliste
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PWD:
$ ./prog0
```

Ou, sans avoir besoin de positionner la variable d'environnement `LD_LIBRARY_PATH` :

```
$ gcc  -o prog0  prog0.o -L. -lliste -Wl,-rpath,${PWD}
$ ./prog0
```

(ne pas bouger libliste.so de repertoire après !)

Voir dependances :

```
$ ldd prog0
linux-gate.so.1 =>  (0xb7f6e000)
libliste.so => not found
libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb7dff000)
/lib/ld-linux.so.2 (0xb7f6f000)
```

Makefile avec bibliothèque dynamique :

```
DEBUG='-g'

all : prog0 prog1 prog2

liste.o : liste.h liste.c
gcc -c ${DEBUG} liste.c

outils_liste.o : outils_liste.h outils_liste.c
gcc -c ${DEBUG} outils_liste.c

libliste.so : liste.o outils_liste.o
gcc -shared -o  libliste.so liste.o outils_liste.o

prog0.o : constantes.h prog0.h prog0.c liste.h
gcc -c ${DEBUG} prog0.c

prog1.o : constantes.h prog1.h prog1.c liste.h
```

```
gcc -c ${DEBUG} prog1.c

prog2.o : constantes.h prog2.c liste.h outils_liste.h
gcc -c ${DEBUG} prog2.c

prog0 : prog0.o libliste.so
gcc -o prog0 prog0.o -L. -lliste -Wl,-rpath,${PWD}

prog1 : prog1.o libliste.so
gcc -o prog1 prog1.o -L. -lliste -Wl,-rpath,${PWD}

prog2 : prog2.o libliste.so
gcc -o prog2 prog2.o -L. -lliste -Wl,-rpath,${PWD}

clean :
rm -f liste.o outils_liste.o prog0.o prog1.o prog2.o \
prog0 prog1 prog2 libliste.so
```