Manuel d'utilisation
Ns-DebugMem

Table des matières

RESUME

NS-DebugMem permet de rechercher les problèmes liés à l'utilisation de la mémoire dans les projets NS-DK, NatStar et NatWeb.
Les problèmes mémoires sont détectés indépendamment de leur impact sur l'exécution. Il devient possible de corriger simplement des problèmes de stabilité.

INTRODUCTION

NS-DebugMem offre un ensemble de fonctionnalités permettant d'investiguer les problèmes liés à l'utilisation de la mémoire dans les projets NS-DK, NatStar ou NatWeb.

Cet ensemble de fichiers permet cinq modes d'investigation différents :

- Mode 'DEBUG': exécution d'une application et génération d'un fichier de trace signalant les allocations n'ayant pas fait l'objet d'une désallocation ou les corruptions de mémoire.
- Mode 'CHECK' : vérification de la validité des instructions MOV et FILL du NCL sur la mémoire allouée
- Mode 'SPY' : visualisation en temps réel des quantités de mémoire allouées via une fenêtre spécialisée (Moniteur).
- Mode 'TRACE avec Recompilation' : Exécution d'une application et génération d'un fichier de trace signalant les allocations n'ayant pas fait l'objet d'une désallocation ou les corruptions de mémoire avec des informations permettant de retrouver aisément les lignes de code NCL correspondantes...
- Mode 'API' : consultation depuis un programme NCL des diverses quantités de mémoire allouées a l'aide de l'API GETMEMORYINFO%, contrôle d'intégrité des blocs de mémoire alloués a l'aide de l'API CHECKMEMORY.

FICHIER DE TRACE

A la sortie du programme, un fichier trace <Nom du programme>.LOG est créé dans le répertoire précisé par la variable d'environnement NS-TRACE (ou NS_TRACE sous UNIX). Il contient toutes les informations de trace.

> La mémoire nommée ou partagée n'est pas tracée.

ACTIVATION DU MODE DEBUGMEM SANS RECOMPILATION

DEBUGMEM : Mode trace des non-désallocations

Exécuter les programmes avec la variable d'environnement NSDBGMEM avec la valeur DEBUGMEM.
Exemple :

```
local POINTER l_hseg%

new 1000, l_hseg%, "ETIQUETTE"

ST_ADRESSE = l_hseg%
FILL l_hseg%, 1000, 0
fill l_hseg%, 3, 65

setdata l_hseg%
```

Le fichier de trace contient la liste des segments alloués non libérés.
Exemple de fichier .LOG généré :

```
Program file mame : tstmemlb.exe
program start Time : 15:19:46

Application memory leak
Unfreed blocks at application ending
-----------------------------------
Size= 44 value=10064D72
Size= 20000 value=1005FF0E
Size= 33 value=1005FEAA
Size= 11 value=1005FE5E
```

DEBUGMEMALL : Mode trace des allocations / non-désallocations

Exécuter les programmes avec la variable d'environnement NSDBGMEM avec la valeur DEBUGMEMALL.
Exemple :

```
local POINTER l_hseg%

new 1000, l_hseg%, "ETIQUETTE"

ST_ADRESSE = l_hseg%
FILL l_hseg%, 1000, 0
fill l_hseg%, 3, 65

setdata l_hseg%
```

Le fichier de trace contient la liste de tous les segments alloués avec leur numéro d'ordre ainsi que la liste des segments alloués non libérés.
Exemple de fichier .LOG généré :

```
Program file mame : tstmemlb.exe
program start Time : 15:44:25

Allocated Blocks
================
Number = 2229 Size= 11 value=1005FE5E
Number = 2230 Size= 22 value=1005FEAA
Number = 2231 Size= 20000 value=1005FF02
Number = 2232 Size= 44 value=10064D66
Number = 2233 Size= 33 value=10064DD6

Application memory leak
Unfreed blocks at application ending
-----------------------------------
Number= 2232 Size= 44 value=10064D66
Number= 2231 Size= 20000 value=1005FF02
Number= 2230 Size= 22 value=1005FEAA
```

Mode de vérification systématique des instructions FILL et MOV

Ce mode permet de tester la validité des MOV, FILL et de la plupart des traitements de strings sur la mémoire allouée et d'avertir l'utilisateur en cas de détection de problème.

Utilisation

Positionner la variable d'environnement NSDBGMEM aux valeurs suivantes qui peuvent être mixées, séparées par un point-virgule.

- CHECKFILL
- CHECKMOV
- CHECKSTRINGS
- CHECKALL
- ADVERTUSER
- ONERRORTRUNCLEN
- ONERRORDOBADACTION
- ONERRORABORT

CHECKFILL : Contrôle des débordements avec la fonction FILL sur mémoire allouée

Exécuter les programmes avec la variable d'environnement NSDBGMEM avec la valeur CHECKFILL.
Le fichier de trace contient la liste des segments sur lesquels le programme a tenté d'effectuer un FILL avec débordement.
Exemple de fichier .LOG généré :

```
Command : wFill
Error : length invalid, required=50, max allowed=44
Action : Action done
```

CHECKMOV : Contrôle des débordements avec la fonction MOV sur mémoire allouée

Exécuter les programmes avec la variable d'environnement NSDBGMEM avec la valeur CHECKMOV.
Le fichier de trace contient la liste des segments sur lesquels le programme a tenté d'effectuer un MOV avec débordement.
Exemple de fichier .LOG généré :

```
Command : wMov
Error src: length invalid, required=50, max allowed=44
Error dst: length invalid, required=50, max allowed=33
Action : Action done
```

CHECKSTRINGS : Contrôle des débordements avec les fonctions WLSTRCPY et WSETTEXT sur mémoire allouée

Cette option ralentit les programmes.

Exécuter les programmes avec la variable d'environnement NSDBGMEM avec la valeur CHECKSTRING.
Le fichier de trace contient la liste des segments sur lesquels le programme a tenté d'effectuer des LSTRCPY et WSETTEXT sur mémoire allouée
Exemple de fichier .LOG généré :

```
Command : wLStrCpy
Error src: OK
Error dst: length invalid, required=21, max allowed=12
Action : Action done
```

CHECKALL : toutes options CHECK précédentes

Cette option cumule les options CHECKFILL, CHECKMOV et CHECKSTRING.

ADVERTUSER : Affichage d'un message en cas de problème

Exécuter les programmes avec la variable d'environnement NSDBGMEM avec la valeur ADVERTUSER.
Si détection d'un problème CHECK, une boite de message s'affiche proposant trois options :

1. 'Trunc' à la bonne taille.
2. Exécution de la commande telle quelle.
3. Non exécution de la commande.

Choisir l'option désirée pour continuer le programme.

ONERRORTRUNCLEN, ONERRORDOBADACTION, ONERRORABORT

Ces trois options sont utilisées dans le cas où on n'a pas spécifié la commande ADVERTUSER.
Si un problème est détecté, le mode de fonctionnement choisi sera appliqué.

LE MONITEUR MEMOIRE SPYMEM

Ce moniteur permet d'avoir en permanence à l'écran la quantité de mémoire consommée par le programme et de suivre en temps réel son évolution.

Utilisation

Ce mode est automatiquement activé par le positionnement de la variable d'environnement suivante :

```
set NSDBGMEM=SPYMEM
```

A l'exécution du programme, si la DLL NSxxSPYM.DLL est accessible, la fenêtre de SPY apparaîtra.
Activer la fenêtre du moniteur mémoire, la redimensionner si désiré, la positionner dans un coin de l'écran.
Pour qu'elle reste toujours visible, activer l'option "always on top" et régler la fréquence de rafraîchissement désirée.
Pour supprimer l'affichage de la barre de titre, activer l'option "Hide title". Pour la faire réapparaître, double-cliquer sur la fenêtre.
Faire fonctionner l'application normalement.

Fonctionnalités

- Une barre de statut indiquant la quantité de mémoire allouée par l'utilisateur et par l'application peut être affichée.
- On peut sélectionner l'affichage de la mémoire utilisateur ou de la mémoire de l'application.
- Fonction pause.
- Boîte de dialogue affichant des informations supplémentaires et des statistiques sur l'utilisation de la mémoire.
- Changement d'échelle automatique.
- Nom du programme testé dans la barre de titre.
- Rafraîchissement de la fenêtre par simple double-clic.

Légende

En jaune, s’affiche la quantité de mémoire allouée par la commande NEW des application NCL (application elle-même ou custom controls).
En vert, s’affiche la quantité de mémoire allouée par le run-time NS-DK, NatStar ou NatWeb.

MODE MEMTRACE (AVEC RECOMPILATION)

Ce mode permet d'avoir les mêmes informations que le mode DEBUGMEM mais en outre ce mode permet de connaître exactement la ligne NCL où a eu lieu le problème.

Ce mode permet également de connaître la liste des tentatives de libération de blocs mémoire avec un pointeur invalide. Pour chacune de ces désallocations elle permet en outre de situer l'emplacement de la commande DISPOSE dans le fichier source NCL.

Utilisation

Recompiler les projets avec l'option de compilation /DNSMEMTRACE. Une information apparaît dans la boite d'information de NS-DESIGN.

Positionner la variable d'environnement NSDBGMEM aux valeurs vues précédemment.

Le fichier de trace

Dans le fichier de trace apparaissent les noms des sources incriminés ainsi que les numéros de ligne. Il faut ensuite manuellement extrapoler l'emplacement dans le source NCL à partir de l'emplacement dans le C généré (voir annexe A).
Exemple de fichier généré :

```
NSDK:19:12:43 NSGENTRACE-funct { TST_WLSTRCAT1$ in
TOTOR.EXE/TESTCHAR.NCL(31)
NSDK:19:12:43 STRING TRUNCATION - File
'C:\BUGS_NAT\BUGS_NATSTAR\TESTV5\WIN32\testchar.c' Line 85 ( Size 20
Maxsize 15 in wStrCatS )
NSDK:19:12:43 NSGENTRACE-funct } TST_WLSTRCAT1$ in
TOTOR.EXE/TESTCHAR.NCL(34) (CLOCK=0000.875)
.
Program file mame : G:\prj\nsdk\GEN\INNT112W\DLL\TSTMEMTR.EXE
program start Time : 10:55:14

Command : wFill
FileName : d:\PRJ\MEMORY\TSTMEMTR\WIN32\MAIN.c line:266
Error : length invalid, required=50, max allowed=44
Pointer : Allocated in file d:\PRJ\TSTMEMTR\WIN32\MAIN.c line:367
Action : Action done

Command : wMemDispose
FileName : d:\PRJ\MEMORY\TSTMEMTR\WIN32\MAIN.c line:321
Error : invalid pointer : FEFEFEh


Application memory leak
Unfreed blocks at application ending
-----------------------------------
d:\PRJ\MEMORY\TSTMEMTR\WIN32\MAIN.c line=367 Size= 44 value=100609EA
d:\PRJ\MEMORY\TSTMEMTR\WIN32\MAIN.c line=358 Size= 33 value=10060AA2
d:\PRJ\MEMORY\TSTMEMTR\WIN32\MAIN.c line=340 Size= 11 value=1006093E
```

Dans cet exemple on peut voir que :

- Un pointeur a été alloué en ligne 367 du fichier MAIN.C, il a été fait une tentative de FILL avec débordement en ligne 266 du fichier MAIN.C.
- Une instruction NCL DISPOSE invalide a été faite depuis le fichier MAIN.C en ligne 321. Le pointeur qui a été passé à cette instruction valait FE:FEFE en hexadécimal et ne pointait pas sur une zone mémoire valide.

Trois blocs mémoire n'ont pas été désalloués à la sortie du programme, avec l'emplacement de leurs allocations.

> La mémoire nommée ou partagée n'est pas tracée.

CONSULTATION DEPUIS UN PROGRAMME NCL DES DIVERSES QUANTITES DE MEMOIRE ALLOUEES

API d'interrogation GETMEMORYINFOS%

Cette nouvelle API permet depuis un programme NS-DK (NatStar ou NatWeb) de connaître les diverses quantités de mémoire allouées.

Utilisation

Dans un projet NCL à tester, inclure le fichier de ressource NCL NSDBGMEM.

```
function GETMEMORYINFOS% (int Kind (2)) return int (4) external
'NS**MISC.GETMEMORYINFOS'
```

Cette fonction doit être appelée avec l'un des types de mémoire suivant :

Type	Description
USER_MEMORY%	Renvoie la quantité de mémoire allouée par la commande NEW des applications NCL.
MAXUSER_MEMORY%	Renvoie la valeur maximum de la quantité de mémoire utilisateur allouée.
APPLI_MEMORY%	Renvoie la quantité de mémoire allouée par l'application (programme utilisateur + Run-Time NS-DK ou NatStar)
ALLOCATED_MEMORY%	Renvoie la quantité de mémoire que l'application s'est allouée vis à vis du système.
MAXALLOC_MEMORY%	Renvoie la valeur maximum de la quantité de mémoire allouée par l'application vis à vis du système.
ALLOC_COUNT%	Renvoie le nombre d'allocations (NEW) faites dans l'application.
DESALLOC_COUNT%	Renvoie le nombre de désallocations (DISPOSE) faites dans l'application.

La valeur de retour est exprimée en octets.

Cette fonction peut être utilisée pour afficher la quantité de mémoire allouée dans l'application ou pour automatiser les contrôles de perte de mémoire.

API de vérification de l'intégrité de la mémoire

Instruction CHECKMEMORY external 'NS**MISC.CHECKMEMORY'

L’appel à cette instruction lance la vérification de l’intégrité de la mémoire.

Instruction SETMEMORYCHECKMODE int CheckMode (4)

Cette instruction positionne le mode d'analyse de l'intégrité mémoire à vrai ou faux.

Instruction SETMEMORYCHECKMODE

Lorsque cette instruction est positionnée à TRUE%, toutes les fonctions d'allocation ou de désallocation mémoire effectuent une analyse de l'intégrité de la mémoire avant d'être exécutées.

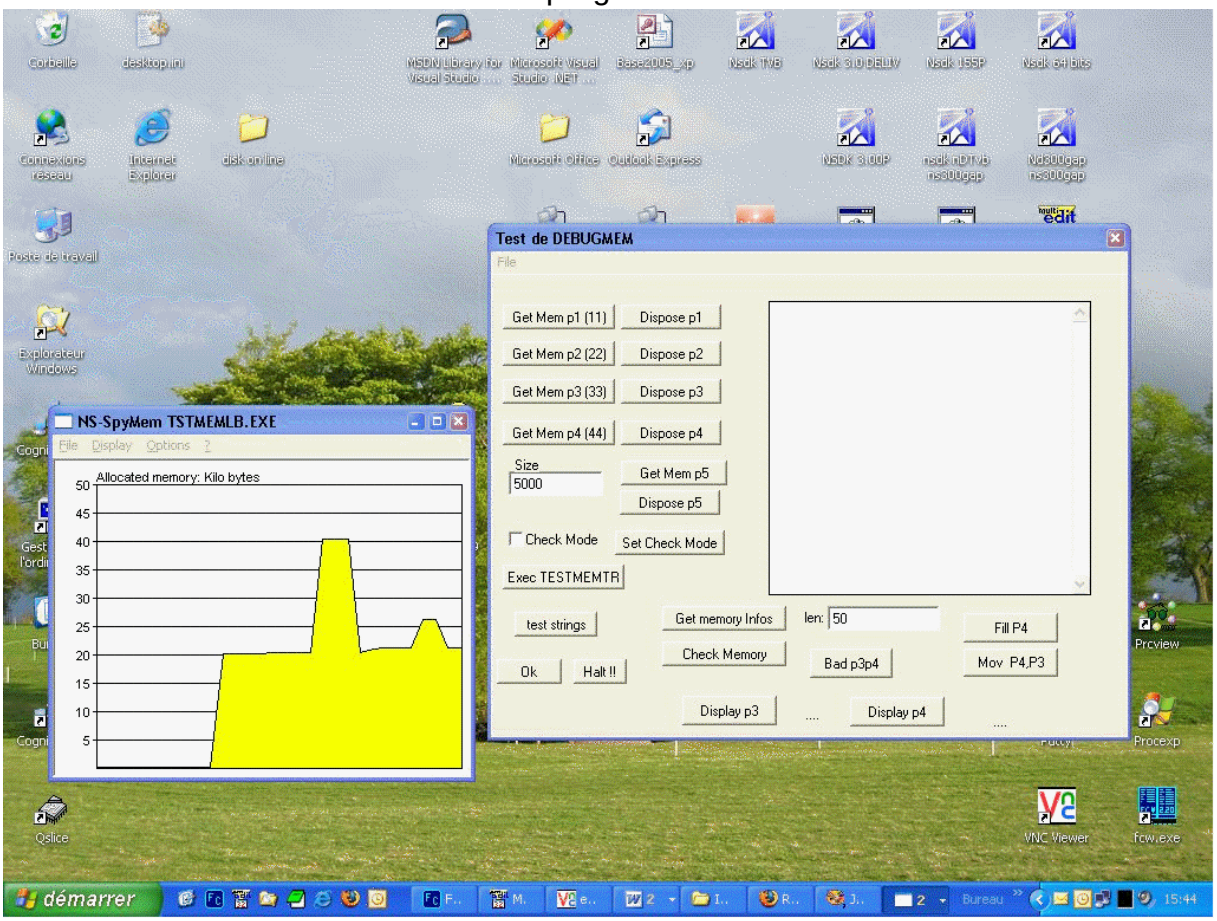
Si un écrasement mémoire est détecté, un message d'erreur apparaît et un fichier de trace NSDMEM.NSD est écrit sur le disque dans le répertoire pointé par les variables d'environnement NS TMP, TMP ou TEMP

Lorsque cette instruction est positionnée à FALSE%, elle désactive l'analyse automatique de l'intégrité de la mémoire.

Valeur par défaut : FALSE%

EXEMPLE D'AFFICHAGE DE LA FENETRE SPYMEM

Suivi de consommation mémoire d'un programme :



NS-TRUNCTRACE

NS-TruncTrace offre un ensemble de fonctionnalités permettant d'investiguer les problèmes liés aux troncatures de chaînes lors de l'exécution d'un programme généré avec NS-DK, NatStar ou NatWeb.

Il fonctionne avec deux modes d'investigation :

- Mode sans recompilation : lorsqu'une troncature de chaîne est détectée, affichage d'une fenêtre de warning, et/ou génération d'un message d'information dans le fichier de trace spécifié par la variable d'environnement NS-TRACE.
- Mode avec recompilation : lorsqu'une troncature de chaîne est détectée, génération dans le fichier de trace d'un message d'information précisant la localisation de la troncature (fichier/numéro de ligne).

Mode sans recompilation

Trace des troncatures de chaînes

Exécuter le programme avec la variable d'environnement NSTRUNCSTR positionnée à la valeur TRACE.
Les messages de troncature sont stockés dans le fichier de trace spécifié par la variable d'environnement NS-TRACE.
Exemple de trace générée :

```
STRING TRUNCATION - Detected on a wStrCatArray command - Required size : 21
- Available size : 15
```

Affichage d'une boîte de message en cas de problème

Exécuter le programme avec la variable d'environnement NSTRUNCSTR à la valeur ADVERTUSER.
Lors de la détection d'un problème une fenêtre du type suivant s'affiche :

Combinaison de la trace et de l'affichage

Il est possible de combiner les deux comportements :

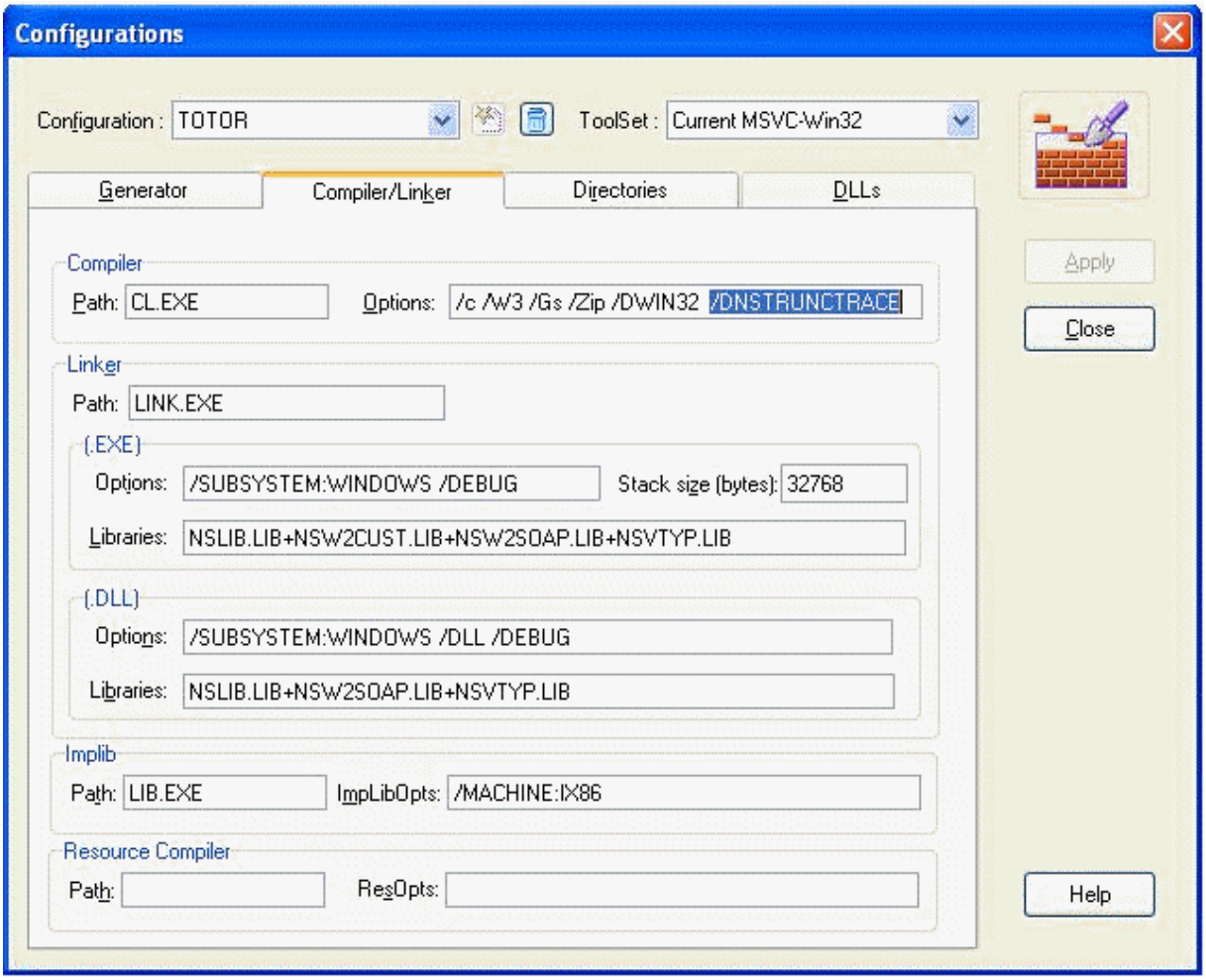
```
SET NSTRUNCSTR=TRACE;ADVERTUSER
```

Mode avec recompilation

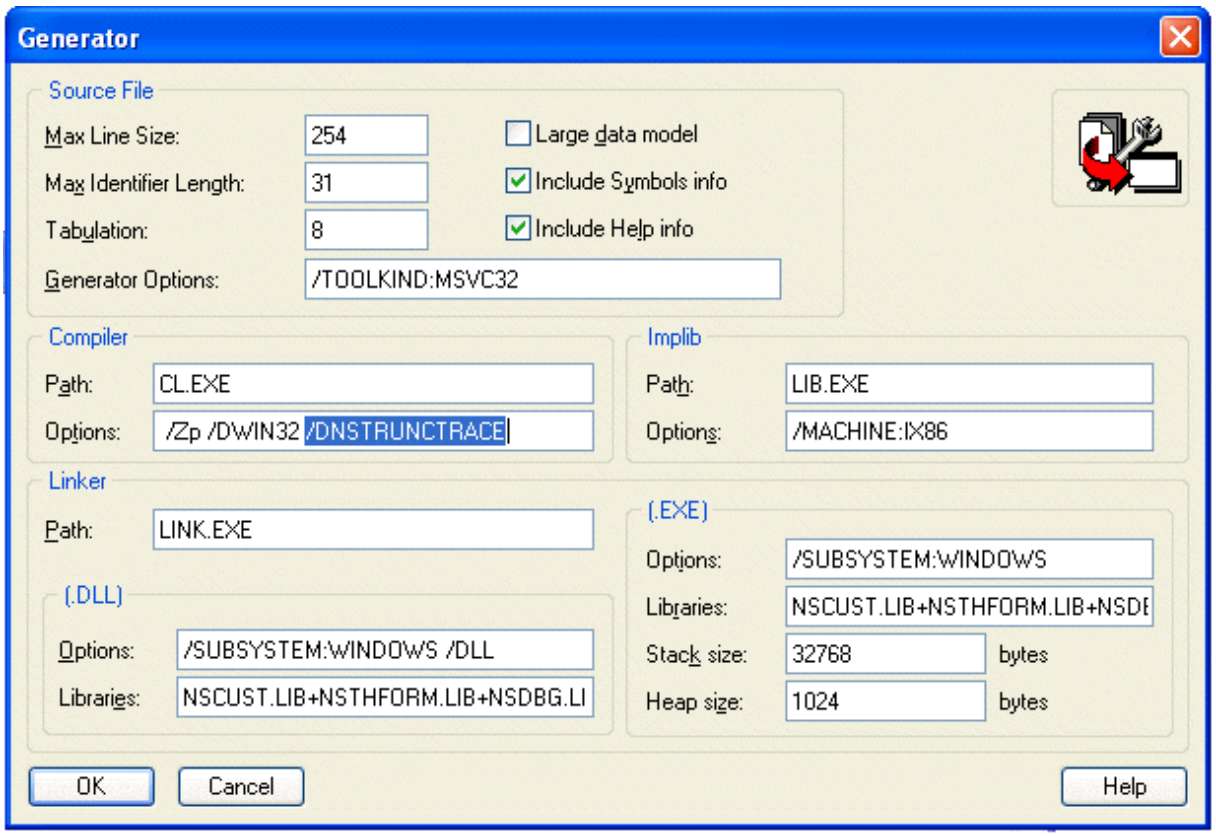
Ce mode permet d'avoir les mêmes informations que le mode précédent, mais il permet en outre de connaître exactement la ligne de code où a eu lieu le problème.

Utilisation

1. Dans NsDesign, ouvrez la boîte de dialogue Configurations (menu Options\Build Configurations), à l'onglet Compiler/Linker, il faut rajouter l'option de compilation /DNSTRUNCTRACE



2. Dans NSADE, sélectionnez le menu Build/Set Configuration puis le bouton Gen de la boîte qui apparait, ajouter l'option de compilation /DNSTRUNCTRACE



3. Regénérer le programme.
4. Lancer le programme avec la variable d'environnement NS-TRACE spécifiant le fichier de sortie.

Description des messages

On obtient dans le fichier des informations du style :

```
NSDK:19:13:30 STRING TRUNCATION - File <<Nom_fichier>> Line
<<Numero_ligne>> ( Size <<taille_demandee>> Maxsize <<taille_disponible>>
in <<Nom_interne_fonction>> )
```

Exemple :

```
STRING TRUNCATION - File 'C:\BUGS_NAT\BUGS_NATSTAR\TESTV5\WIN32\testchar.c'
Line 85 ( Size 20 Maxsize 15 in wStrCatS )
```

Dans cet exemple, on peut voir qu'une chaîne a été tronquée en ligne 85 du fichier testchar.c lors d'une concaténation (wStrCatS), car la taille nécessaire au résultat est 20 caractères alors que seuls 15 sont disponibles.

Pour retrouver plus facilement le code NCL correspondant, il est conseillé de générer avec l'option «Runtime Trace ». On aura une trace spécifiant l'instruction NCL appelante, du genre :

```
NSDK:19:12:43 NSGENTRACE-funct { TST_WLSTRCAT1$ in
TOTOR.EXE/TESTCHAR.NCL(31)
NSDK:19:12:43 STRING TRUNCATION - File
'C:\BUGS_NAT\BUGS_NATSTAR\TESTV5\WIN32\testchar.c' Line 85 ( Size 20
Maxsize 15 in wStrCatS )
NSDK:19:12:43 NSGENTRACE-funct } TST_WLSTRCAT1$ in
TOTOR.EXE/TESTCHAR.NCL(34) (CLOCK=0000.875)
```

ANNEXE : MARCHE A SUIVRE POUR RETROUVER L'ORIGINE EN NCL D'UNE LIGNE DE C GENERE

Les fichiers de trace indiquent les erreurs sous la forme suivante :

```
D:\nsdk\memory\tstmemtr\win32\main.c line=824 size=11
```

Dans ce cas, la ressource incriminée a pour nom : "MAIN".
Trouver dans le projet NS-DK (NatStar ou NatWeb), la ressource de nom MAIN. Deux cas sont possibles :

- La ressource est de type fenêtre (.SCR),
- La ressource est de type librairie (.NCL).

Cas N° 1, la ressource est de type SCR

Exemple :
D'après le fichier de trace généré, il apparaît deux erreurs aux lignes suivantes :

```
d:\nsdk\memory\tstmemtr\win32\main.c line=824 size=11

d:\nsdk\memory\tstmemtr\win32\main.c line=836 size=33

switch (ID) {
case 0:
switch (msg) {
case NS_MSG_INIT: {
Ligne 824 -> iM1= (NS_LONG)wMemNew(0,11,"");
return 0;
}
case NS_MSG_TERMINATE: {
return 0;
}
}
break;
case ID_MAIN_PB_GETMEM3:
switch (msg) {
case NS_MSG_EXECUTED:
if (wDlgCheck1(wnd,ID_MAIN_PB_GETMEM3,100,120)) {
Ligne 836 -> iM3= (NS_LONG)wMemNew(0,33,"");
}
return 0;
}
break;
```

Erreur de la ligne 824

Dans ce cas, en remontant le source, on lit, 2 lignes plus haut, "case NS_MSG_INIT". Cela signifie que le code exécuté l'est dans un événement INIT

En remontant encore 2 lignes, on lit "case 0" , (case dépendant du « switch (ID)) la valeur 0 signifie que l'événement est exécuté dans une fenêtre et non dans un contrôle.

La 1ère erreur s'est donc produite dans l'événement INIT de la fenêtre MAIN.

Erreur de la ligne 836

Dans ce cas, en remontant le source, on lit, 2 lignes plus haut, "case NS_MSG_EXECUTED". Cela signifie que le code exécuté l'est dans un événement NCL EXECUTED.

En remontant encore 2 lignes, on lit "case ID_MAIN_PB_GETMEM3" il faut décomposer l'identificateur "ID_MAIN_PB_GETMEM3".

- ID signifie Identificateur.
- MAIN est le nom de la ressource (fenêtre).
- PB_GETMEM3 est le nom du contrôle.

L'erreur s'est donc produite dans l'événement EXECUTED du contrôle PB_GETMEM3 de la fenêtre MAIN.

Cas N° 2, la ressource est de type NCL

Exemple :
D'après le fichier de trace généré, il apparaît deux erreurs aux lignes suivantes.

```
d:\cvsprj\nsdk\memory\tstmemtr\win32\testmem.c Line 20 Size=150

d:\cvsprj\nsdk\memory\tstmemtr\win32\testmem.c Line 29 Size=80

/* */
/* Library TESTMEM */
/* */
NS_FN(void) riTESTMEM(void)
{
iP= (NS_LONG)wMemNew(0,150,"");  <- Ligne 20
}

/* */
/* Instruction TESTMEMORY */
/* */
NS_FN(void) TESTMEMORY(void)
{
NS_LONG iP2;
iP2= (NS_LONG)wMemNew(0,80,"");  <- Ligne 29
}
```

Dans le 1er cas, la fonction a le même nom que la librairie et est préfixée par « ri ». Cela signifie que l'erreur s'est produite pendant l'initialisation de la librairie.
Dans le 2ème cas, on peut voir directement que l'erreur s'est produite dans l'instruction TESTMEMORY.

Cas N° 3, le nom de fichier C est le même que celui de la ressource

Les fichiers de trace indiquent la position des erreurs dans le code C généré.
Exemple :

```
STRING TRUNCATION - File 'C:\BUGS_NAT\BUGS_NATSTAR\TESTV5\WIN32\testchar.c'
Line 85 ( Size 20 Maxsize 15 in wStrCatS )
```

Le nom de fichier C est le même que celui de la ressource. Dans ce cas, la ressource incriminée a donc pour nom « TESTCHAR »
Le code C généré est :

```
/* */
/* Function szTST_WLSTRCAT1 */
/* */
NS_FN(NS_PCHAR) szTST_WLSTRCAT1(NS_PCHAR sz__RETURN)
{
static const NS_CHAR NS_szCurId[] = "TST_WLSTRCAT1$";
static const NS_GenTrace NS_traceEnter={1,NS_traceEnterFunc,NS_szCurMod,
NS_szCurRsc,NS_szEmpty,NS_szCurId,NS_NULL};
static const NS_GenTrace NS_traceLeave={1,NS_traceLeaveFunc,NS_szCurMod,
NS_szCurRsc,NS_szEmpty,NS_szCurId,NS_NULL};
NS_CHAR szVARIABLE[16];
wNSGenTrace(&NS_traceEnter, 31, NS_NULL);
wLStrCpy(szVARIABLE,"1234567890",16);
wStrCatS(szVARIABLE,16,szVARIABLE,"ABCDEFGHIJ"); // LIGNE 85
{
wLStrCpy(sz__RETURN,szVARIABLE,21);
wNSGenTrace(&NS_traceLeave, 34, NS_NULL);
goto Return;
}
Return:
return sz__RETURN;
}}
```

A partir du nom de la fonction en C, on peut voir que l'erreur s'est produite dans la function NCL TST_WLSTRCAT1$(le préfixe sz en C, correspond au suffixe $ en NCL).

```
FUNCTION Tst_wLstrcat1$ RETURN CSTRING (20)
 LOCAL CSTRING VARIABLE$ (15)

 VARIABLE$ = "1234567890"
 VARIABLE$ = VARIABLE$ && "ABCDEFGHIJ" ; Troncature

 RETURN VARIABLE$
ENDFUNCTION
```

INDEX