SOFTICE


By Falcon

E-Mail : falcon16@caramail.com

1 - Introduction

Les deux "grosses bêtes" de la profession sont Softice (ou Winice) et W32dasm. Il ne faut pas l'un ou l'autre, il faut l'un et l'autre. Ces outils comme leurs noms l'indiquent, débugge les prog (évidemment) c'est à dire qu'on entre dans les entrailles d'un logiciel pour faire la chasse aux bugs (faute dans un prog). Donc de là à considérer que l'apparition d'une boîte de rappel ou l'arrêt d'un prog après trente jours d'utilisation est un bug ... il n'y a qu'un pas. Quand vous modifiez un prog sous débuggers, ils simulent vos modifications du prog mais ne l'effectuent pas, au pire vous aurez droit à un beau plantage si vous vous êtes plantés mais tout doit rentrer en ordre au redémarrage du micro (enfin... normalement). Ce qui nous permet d'essayer sans aucun danger diverses possibilités pour arriver à notre but … LE CRACK !!!

Le débugging consiste à tracer un programme ligne par ligne. C'est à dire qu'en théorie on est capable de démarrer ligne de commande par ligne de commande (une à une) un programme à partir du point qu'on appellera zéro jusqu'à son chargement complet en mémoire (sachant qu'un programme de taille normale contient plusieurs milliers de lignes . . . Bon courage). Donc toujours en théorie, on est capable d'influer sur le comportement d'un programme puisque visuellement vous verrez le prog se développer au fur et à mesure de la progression dans les lignes de codes et si je continue dans mon délire, vous pouvez donc supprimer des trucs qui ne vous intéressent pas. (Quoi donc ... attendez que je réfléchisse ... par exemple ...euh ... des nags du genre "Veuillez régler votre licen ...").

Comme nos string data ref dans Wdasm qui nous aident à aller plus rapidement à l'endroit du prog, il y a aussi les API de Windows. Les API sont à Windows ce que les interruptions sont au DOS, on peut les comparer à des macros, de petits programmes standarts utilisés régulièrement par de multiples applications (ex: boîte de dialogue qui sont souvent les mêmes quelque soit le logiciel que vous utilisez; sortie de prog: exit process) Nota : je suis prêt à parier que l'utilisation des API par les éditeurs de logiciels de toutes sortes doit se traduire par une sacrée facture ... eh qui c'est qui paye au bout ...? Enfin bref, les API sont là; autant s'en servir (je parle de nous). Elles sont visibles toujours avec W32dasm et aussi Softice. Elles sont nommées par exemple Getwindowtext, Messagebox, Hmemcpy, etc ... et ce sont de bons exemples. Et enfin on arrive à l'arme suprême, le BPX (breakpoint) une des commandes présente dans W32dasm et Softice. Le BPX est pour le crackeur INDISPENSABLE. C'est grâce à lui que l'on pourra stopper un programme à un endroit précis; ce qui nous évite de tracer ligne par ligne jusqu'à un point donné puisque qu'il suffira de lancer le prog: celui s'arrêtera automatiquement sur le BPX, mais en plus on pourra aussi le stopper même quand le prog sera chargé en mémoire. Voilà en résumant pour la première étape, vous avez les "string data réf", les API les repères (il y en a d'autres) et les BPX à poser sur les repères (aux alentours) ou à des endroits stratégiques.

Si nous faisions un comparatif entre Wdasm et SoftIce :

W32dasm:
Ses avantages :
- Tout à la souris très pratique
- Ses fameuses String data références génial
- Une vue d'ensemble du prog désassemblé bien meilleure que Softice (interface Windows)

Ses inconvénients :
- Ne débugge pas les prog 16 bits (désassemble seulement)
- Plus limité dans la vision de contenues mémoires, pile, …

Softice (Winice):
Ses avantages :
- Débugge les prog 16 bits
- La possibilité de "jouer" avec les handles,
- on peut tout faire, tout voir, …

Ses inconvénients :
- Vous démarrez sous Windows mais le travail se fait sous fenêtre Dos avec manipulation clavier.
- Vous devez rebooter votre micro (ligne ajoutée dans l'autoexec.bat) si winice n'est pas chargé en mémoire.

2 - Installer SoftIce.

Vous avez dû trouver le programme d'installation pour SoftIce dans un fichier zip. Extractez le dans C:\TEMP, ou dans celui de votre choix, et lancez setup.exe. Cela débutera le processus d'installation SoftICE 3.2x.ou 4.x.

Après la saisie du serial (dans une boîte d'enregistrement), le programme va d'abord vous demander s'il existe une version de Softice sur votre poste. Si ce n'est pas le cas, cliquer sur le bouton « No Product ».

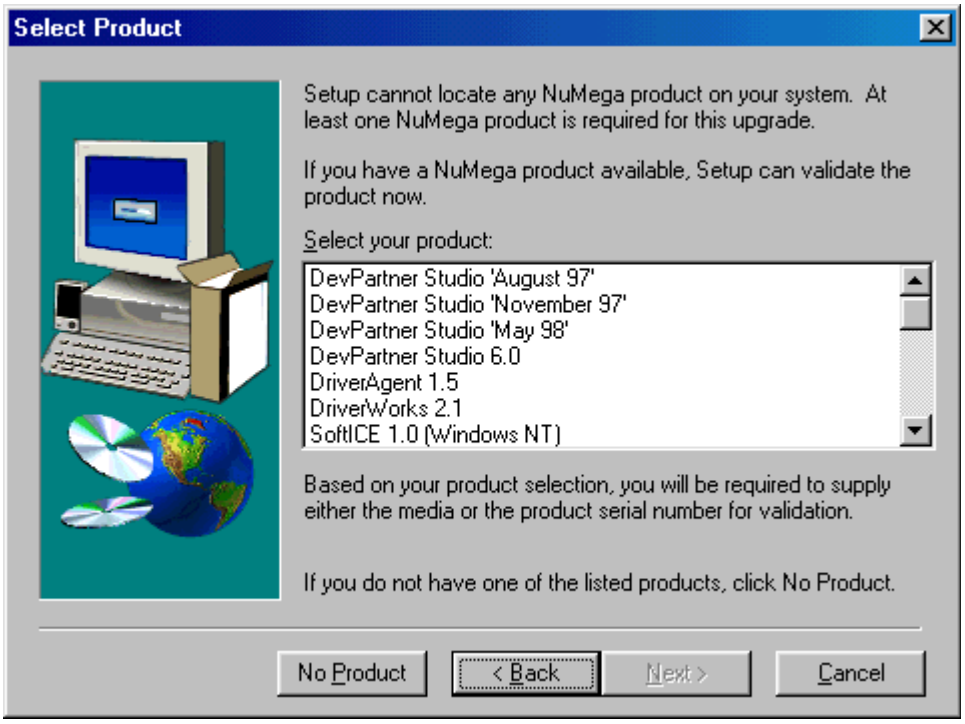


FIG-1

Après quoi il vous demandera un autre numéro de série.

Répertoire d'installation

Le setup va vous demander ensuite dans quel répertoire vous souhaitez installer SoftIce (FIG. 1). Je choisi le répertoire par défaut c:\Program Files\ pour faciliter la configuration des fichiers Config.sys et AutoConfig.bat. Si vous choisissez un autre répertoire (comme ici FIG.2 : c:\softice95), il faudra veiller à corriger le chemin d'accés au bon répertoire

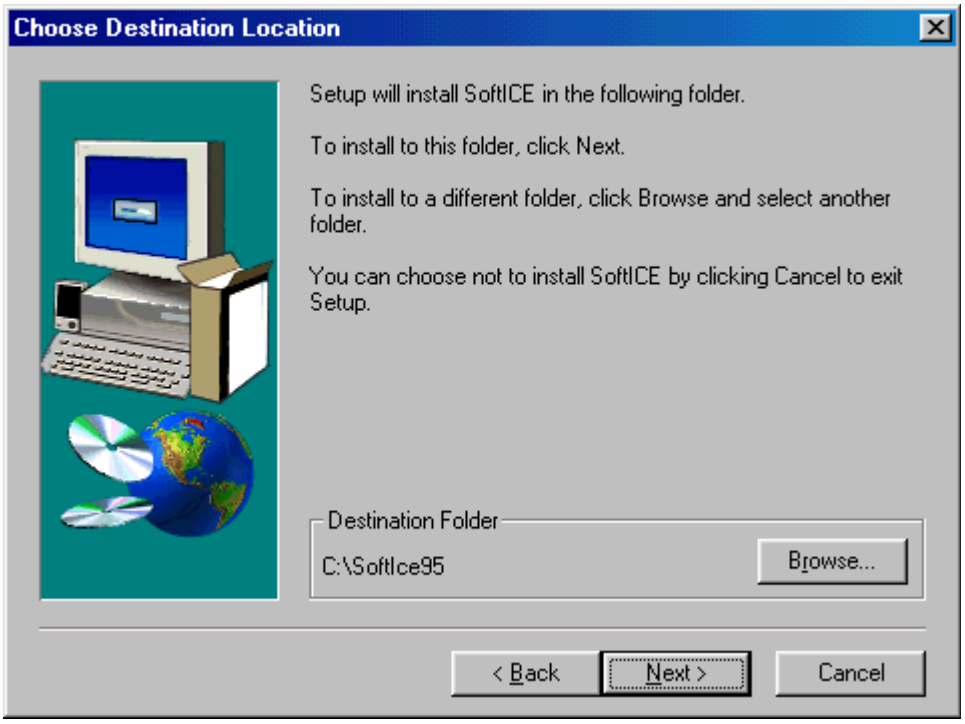


FIG-2

Video Driver Selection

Traversez tous les écrans d'initialisation (registration, License Agreement, etc.) jusqu'à ce que vous ayez sous les yeux le "Display Adapter Selection Screen" (FIG. 3). Cherchez parmi les différents drivers celui qui est adapté à votre carte, et cliquez sur le bouton "Test". Si vous ne voyez pas de message au moment de ce test, c'est que le driver choisi ne convient pas. Essayez en un autre, ou optez pour le 'Universal Video Driver' en cochant la case et en choisissant le modèle Standard VGA. C'est celui que j'utilise comme vous pouvez le voir dans la FIG.3.

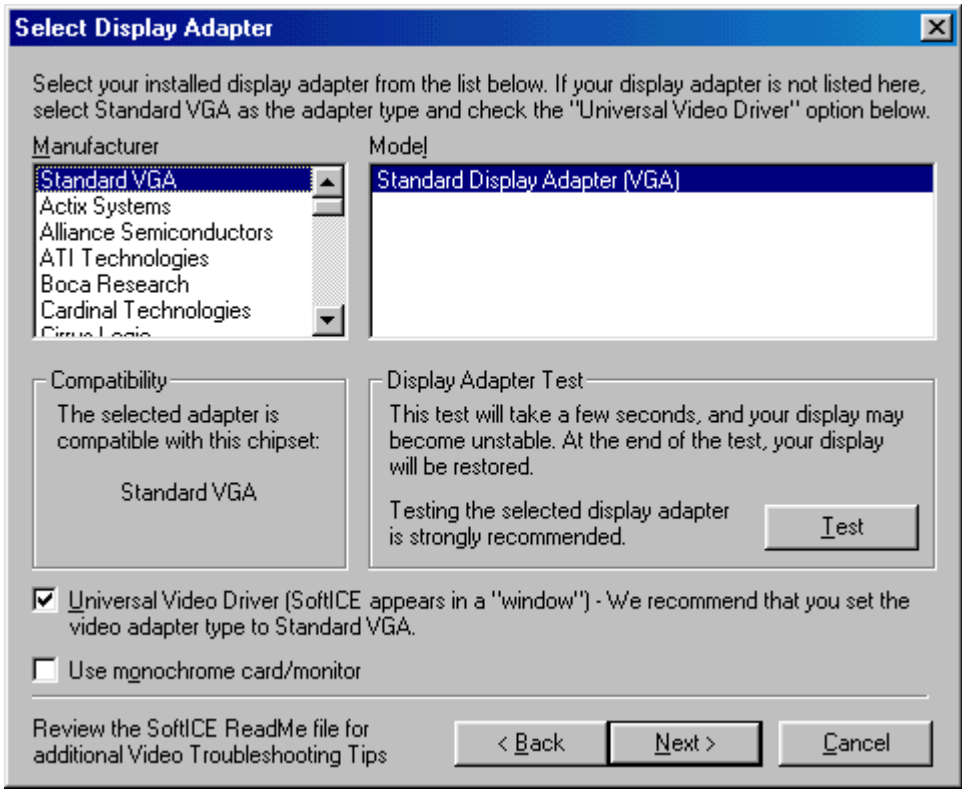


FIG-3

Sélection de la Souris

Au moment où le programme d'installation (FIG. 4) vous demandera de sélectionner un type de souris suivant le modèle que vous utilisez, vous aurez à choisir entre une PS2, ou une Sérial. La PS2 a un petit connecteur rond (mini Din), est une sérial a un connecteur trapézoïdal standard (broche à 2 rangées) du même type que celui de votre écran. Si votre souris est de ce type, assurez vous que vous choisissez bien le bon port ,COM 1 ou COM 2. Vérifiez au dos de votre boitier AT ou ATX le numéro du port sur laquelle la souris se trouve (par défaut, COM 1), et séléctionnez le. En cas de problème, vous pourrez revenir sur le choix de la souris installée par le menu Démarrer\Numéga SoftIce\Mouse setup.

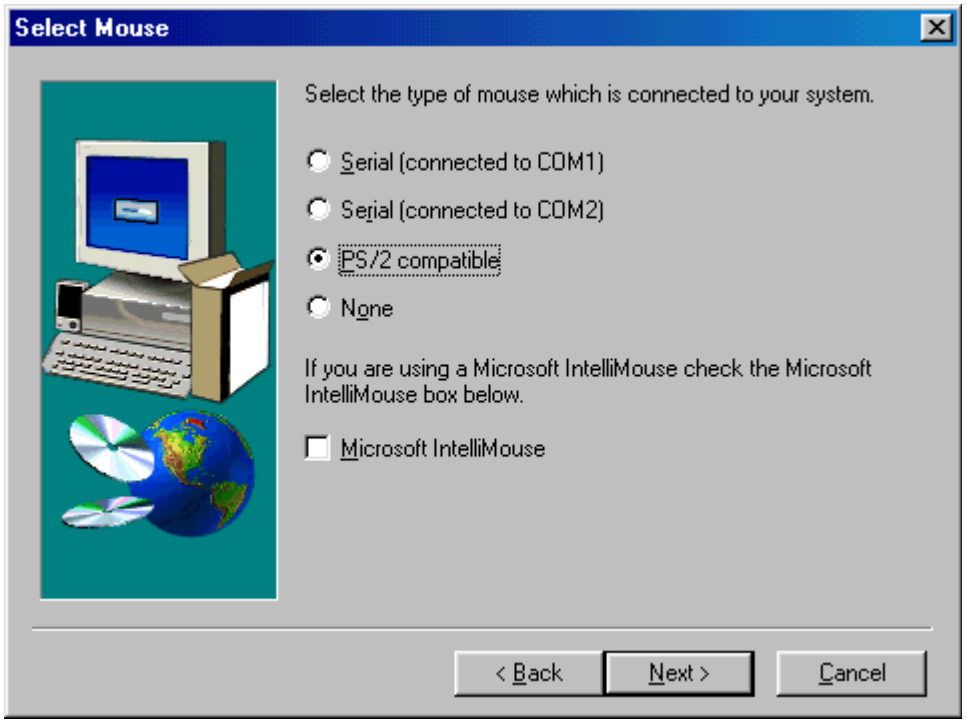


FIG-4

Configuration du Système pour SoftIce

Le dernier écran d'installation est celui de la configuration de votre OS [plus exactement du fichier AUTOEXEC.BAT] (FIG.4). Sélectionnez la dernière "Do not make any changes." (FIG.5). Les modifications qu'il envisage de faire ne sont pas les meilleures, surtout si vous avez décidé d'installer SoftIce dans un autre répertoire que celui proposé par défaut. Nous allons réaliser les changements demandés nous même.

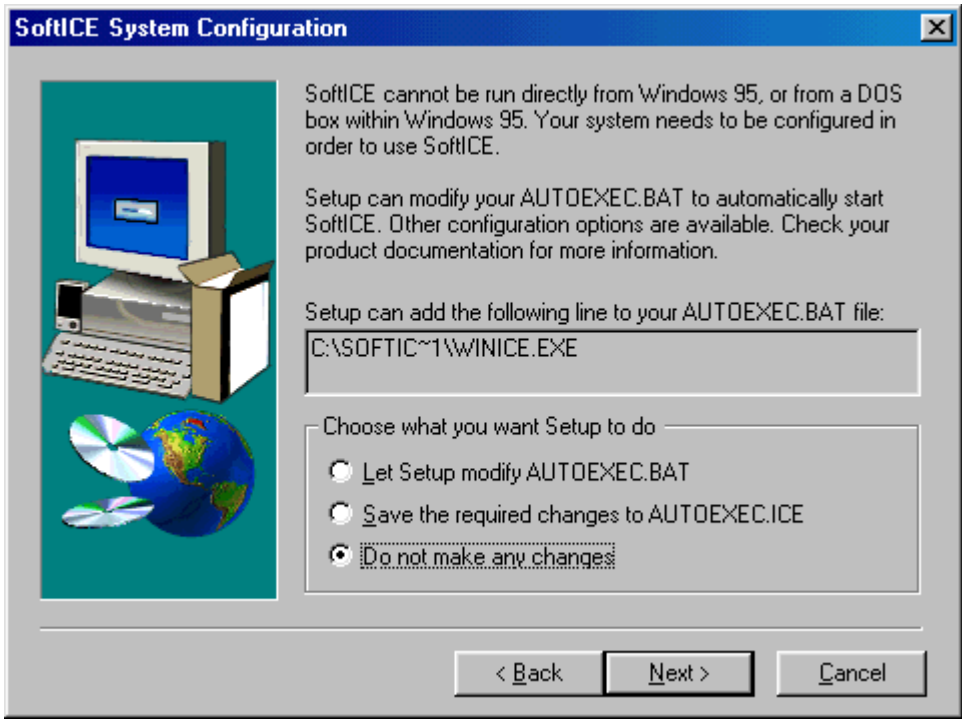


FIG-5

C'est fait

OK, vous avez sélectionné la troisième option, et vous pouvez appuyer maintenant sur le bouton 'NEXT' et finir l'installation. Appuyez sur le bouton 'FINISH' sur l'écran suivant et revenez quand le processus d'installation sera achevé.... ;-)

Ca n'était pas bien dur, n'est ce pas ?

3 - PARAMETRAGE

A – Autoexec.bat et Config.sys :

Le problème est que si vous laissez le fichier WINICE.EXE en permanence actif au démarrage de votre PC, vous risquez à tout moment, étant donné que c'est un débuggeur, de le voir réapparaître (vous signalant qu'il a détecté un bug) interrompant votre programme : ce qui peut dans certaines circonstances être génant. Et vous n'imaginez pas le nombre de prog buggés qui existe (notamment un qui s'appelle Win…) !! Il est donc fortement conseillé d'utiliser un menu au démarrage de votre machine qui vous permettra de choisir le chargement ou non de Softice.
Nous allons donc juste configurer les fichiers Config.sys et AutoExec.bat à notre convenance.
Ce setup vous permettra de décider si vous souhaitez charger SoftIce, ou non, au démarrage de votre OS. Il utilise le [menu] command dans config.sys et permet à l'utilisateur de sélectionner une option:
(Au préalable, pensez à sauvegarder une copie de vos fichiers Config.sys et autoexec.bat)

Pour cela, ajoutez dans le fichier C:\CONFIG.SYS :

```
REM ----------Begin File: C:\CONFIG.SYS--------------------------------

[menu]
menuitem=NORMAL,Mode Normal
menuitem=SOFTICE,Chargement de Softice
menudefault=NORMAL,10

[SOFTICE]
[NORMAL]
[COMMON]

REM -------------------------END CONFIG.SYS------------------------------
```

Cette configuration vous offrira 2 options (l'écran ressemblera à ceci):

```
1. Mode Normal
2. Chargement de SoftIce

Selection: 1 Time Remaining: 3
```

Le 'menudefault' sera le menu par défaut du mode normal sans softice (selection 1) au bout de 10 secondes. Le choix d'un des 'menuitem' retournera la valeur qui permettra de sauter au menu correspondant, et que %CONFIG% transmettra à votre fichier AUTOEXEC.BAT. Le respect des majuscules ou des minuscules (Case) est important à ce stade si vous ne voulez pas faire planter votre système au démarrage, aussi soyez prudent et veillez à toujours réécrire les mêmes informations partout.
Vous pouvez ajouter d'autres menuitem si vous le souhaitez, en veillant toujours à respecter la Case.
Votre "device" doit être placé dans la section [COMMON].

Passons au fichier Autoexec.bat, ajoutez les lignes suivantes :

```
REM -------------------Begin File: C:\AUTOEXEC.BAT-----------------------
GOTO %CONFIG%

:SOFTICE
C:\WINICE95\WINICE.EXE
Goto FIN
:NORMAL
echo Soft-Ice non Chargé !!!
Goto FIN
:FIN
REM -------------------------END AUTOEXEC.BAT--------------------------
```

Comme vous pouvez le voir, si nous sélectionnons l'option 1 dans le Config.sys, alors %CONFIG% pointera sur NORMAL et nous irons à GOTO NORMAL quand l'Autoexec.bat se lancera.
Si nous choisissons l'option 2, nous irons à SOFTICE, qui charge notre précieux outils.

Avec ces explications, vous finirez bien par réussir à lancer SoftIce sur votre Système. Tout ne sera pas fini pour autant, vous allez devoir configurer SoftIce lui même pour qu'il vous rende les services dont vous allez avoir besoin.

B – WINICE.DAT

Softice fait appel à un fichier de configuration dans lequel on y indique tous les paramêtres nécessaire à une utilisation optimale du debugger. Toutes ces options peuvent être entrées manuellement via la ligne de commande de Softice. Mais tant qu'à faire, les entrer une bonne fois pour toute sera bien plus pratique pour la suite de nos opérations.

Dans ce fichier, on y trouve :

- les paramêtres utiles au bon fonctionnement de Softice
- les paramêtres d'affichage de l'écran Softice
- la liste des modules à exporter (DLL) pour l'utilisation des API
- la configuration des touches macros (ne les modifiez pas, elles sont très pratiques et très souvent, dans les cours de cracks on nomme ces touches, elles sont devenues des « standards » en cracking).

Ce qu'il faut donc modifier dans ce fichier c'est la ligne INIT = "X ; " et l'activation des modules DLL.

Commençons par la ligne INIT : celle-ci est principalement utilisée pour une optimisation de l'affichage de l'écran Softice.
Par défaut nous avons INIT = "X ; "
Voici tous les paramêtres fréquemment utilisés à ajouter :

- CODE ON : permettra de voir directement le code en héxa des instructions que vous désassemblez.

- LINES # : définie la taille, en nombre (#) de lignes, qu'aura la fenêtre Dos de SoftIce.

- WC # : active la fenêtre des codes avec # nb de lignes.

- WD # : active la fenêtre des datas (pratique pour connaître le contenu d'une adresse). En tapant WD 12, SoftIce affichera 12 lignes pour cette fenêtre.

- WR : active la fenêtre des registres (très pratique pour connaître le contenu des EAX, ESI et autres EBX)

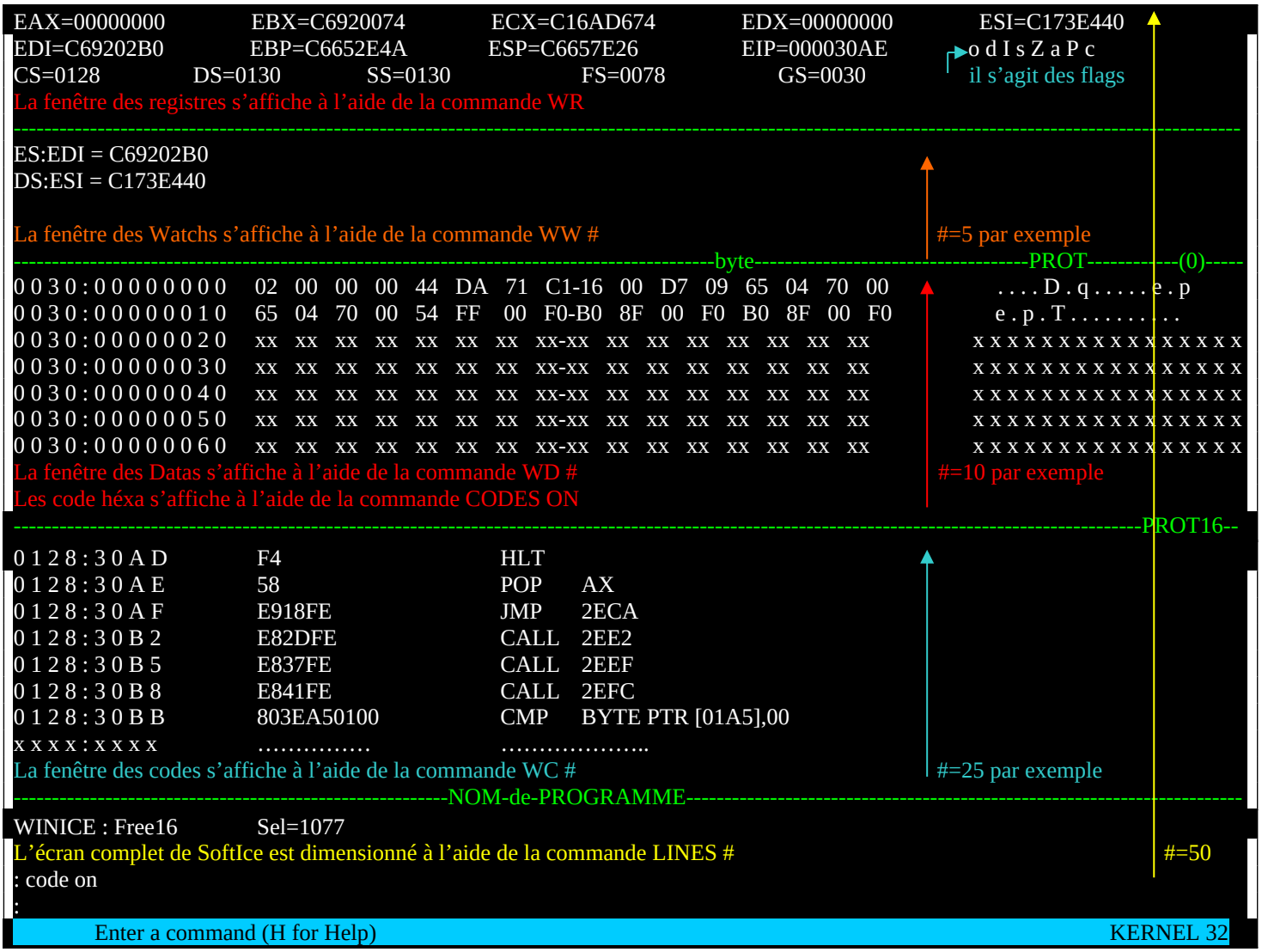
- WW # : active la fenêtre des « watchs » avec # nb de lignes. [Au fait, toutes ces commandes W x, taper seules (c'est à dire sans paramêtres) activent ou désactivent leur propre fenêtre.]

- SET FONT # : détermine la taille des caractères que SoftIce utilisera comme police. Il modifie auusi la taille de votre écran Softice car tous les paramêtres sont effectués avec un nombre de lignes et non une taille de fenêtre.

- WATCH xxx : les watchs feront apparaître le contenu des adresses spécifiées, ainsi vous aurez le loisir de tracer tranquillement avec F10 et de voir le code que vous cherchez s'afficher dans la fenêtres des Datas (enfin, parfois !).
  Exemples : WATCH ES:DI ; WATCH DS:SI ; WATCH * EAX ; WATCH * ESI

- FAULTS (ON/OFF) : Le mettre à OFF , car sinon vous aurez le droit à des breaks sur des fautes générales de protection.

Afin de personnaliser votre écran au mieux vous pouvez tapper directement toutes ces commandes via la ligne de commande de SoftIce et voir l'écran changer d'aspect en temps réel.
Il est aussi possible de redimensionner vos différents écrans avec la souris en sélectionnant la ligne qui sépare vos fenêtres (en maintenant le doigt appuyer sur le clic gauche de la souris) et en étirant vos fenêtres.
Pour comprendre tout ça, voici à quoi devrait ressembler (sans les commentaires ;)) votre écran SoftIce (sur l'exemple donné vous pouvez voir qu'il s'agit d'un programme 16 bits avec PROT16 et dans la fenêtre des codes vous voyez les adresses basées sur 16 bits et non 32 bits.)



Voici donc un exemple d'une ligne INIT (pour info toutes les commandes doivent être séparées par un "**;**" et la ligne doit se terminer obligatoirement par "**X ;** ")

```
INIT="CODE ON ;WR ;LINES 50 ;WC 25 ;WD 10 ;WW 5 ;SET FONT 2;WATCH * ES:DI;WATCH * EAX;FAULTS OFF;X; "
```

Attention, la longueur (nb de caractères) de la ligne INIT est limitée (je ne connais pas le nombe en question ;( ) Dans ce cas il vous faudra créer plusieur lignes INIT mais en faisant bien attention de mettre à la fin de chacune des lignes "X ; "

Il est aussi possible de configurer notre fichier WINICE.DAT à l'aide du logiciel Symbol Loader LOADER32.EXE.
Pour ce faire, allez dans les menus « Edit – SoftICE Initialization Settings … » et là vous aurez tout le contenu du fichier modifiable à souhait.

Description du fichier Winice.dat en détail :

```
PENTIUM=ON                      ;si vous avez un pentium laissez le à ON sinon mettez le à OFF.
NMI=ON
ECHOKEYS=OFF
NOLEDS=ON                       ;active (ON) ou désactive (OFF) le clavier numérique
NOPAGE=OFF
SIWVIDRANGE=ON
THREADP=ON
LOWERCASE=OFF                   ;affiche le code en majuscule (pour les minuscules = ON)
WDMEXPORTS=OFF
MONITOR=0

PHYSMB=64                       ;nombre de mega que vous avez en RAM => à modifier donc en fonction de votre PC
SYM=1024                        ;taille du buffer en octet alloué pour les symboles
HST=256                         ;taille du buffer en octet alloué à l'historique des commandes
TRA=8                           ;taille du buffer en octet alloué au trace
MACROS=32                       ;nombre maximum de macros
DRAWSIZE=2048                   ;taille de la mémoire Vidéo en Ko

INIT="CODE ON ;WR ;LINES 50 ;WC 25 ;WD 10 ;WW 5 ;SET FONT 2;WATCH * ES:DI;WATCH * EAX;FAULTS OFF;X; "

F1="h;"                         ;évitez de modifier ces lignes car elles sont référencées comme standard un peu partout dans les tutoriaux.
F2="^wr;"
F3="^src;"
F4="^rs;"
F5="^x;"
F6="^ec;"
F7="^here;"
F8="^t;"
F9="^bpx;"
F10="^p;"
F11="^G @SS:ESP;"
F12="^p ret;"
SF3="^format;"
CF8="^XT;"
CF9="TRACE OFF;"
CF10="^XP;"
CF11="SHOW B;"
CF12="TRACE B;"
AF1="^wr;"
AF2="^wd;"
AF3="^wc;"
AF4="^ww;"
AF5="CLS;"
AF8="^XT R;"
AF11="^dd dataaddr->0;"
AF12="^dd dataaddr->4;"
CF1="altscr off; lines 60; wc 32; wd 8;"
CF2="^wr;^wd;^wc;"

; ***** Examples of sym files that can be included if you have the SDK *****
;       Change the path to the appropriate drive and directory
;LOAD=c:\windows\system\user.exe
;LOAD=c:\windows\system\gdi.exe
;LOAD=c:\windows\system\krnl386.exe
;LOAD=c:\windows\system\mmsystem.dll
;LOAD=c:\windows\system\win386.exe
; ***** Examples of export symbols that can be included *****
;       Change the path to the appropriate drive and directory
EXP=c:\windows\system\vga.drv                   ;supprimer les ; devant les lignes EXP afin que les modules faisant appel aux API
EXP=c:\windows\system\vga.3gr                   ;soient exportés dans softice (qu'il puisse reconnaitre les API !)
…
EXP=c:\windows\system\olesvr.dll
; ***** Examples of export symbols that can be included for Windows 95 *****
;       Change the path to the appropriate drive and directory
EXP=c:\windows\system\kernel32.dll
EXP=c:\windows\system\user32.dll
…
EXP=c:\windows\system\mspwl32.dll
EXP=c:\windows\system\mpr.dll
EXP=c:\windows\system\Msvbvm50.dll              ;vous pouvez aussi ajouter vos propres exports comme par ex ces DLL utiles aux
EXP=c:\windows\system\Msvbvm60.dll              ;API Visual Basic 5 et 6. (Dans ce cas faire très attention aux maj et min.)
```

Enfin, SOFTICE est prêt à l'emploi. (après avoir redémarré votre PC biensûr !!!)

4 - LES COMMANDES

A – CLAVIER – Touches de fonctions

F1 – L'aide de softIce : plus exactement les commandes avec leur signification (pour sortir de l'aide pressez la touche ESC)

F2 – Affichage des registres en Hexa (égale à la commande WR)

F4 – Vous visualisez sous windows « l'image » ou la capture d'écran du programme (appuyez sur F4 pour retour). On ne peut rien faire dessus !

F5 – Run = démarrage normal du programme ou tout simplement pour continuer son déroulement.

F6 – Pour aller de la fenêtre Trace à la fenêtre Commandes manuelles (et vice versa)

F8 – Trace pas à pas en entrant dans les CALL

F9 – BPX, ou double clicks sur la ligne concernée (passe à la couleur bleue), similaire à la commande par ex : BPX 38CF :0015

F10 – Trace pas à pas sans entrer dans les CALL . Il n'entre pas dans les fonctions (telles les apis ; interruptions,…)

F11 – Sort de la fonction en cours (retour à windows) = CTRL-D

F12 – Permet de sortir d'un CALL et de revenir juste après l'instruction appelante du CALL, c'est en fait une macro qui place un breakpoint sur l'adresse qui a appelé le CALL. Pour info : lors d'un appel de call, le segment et l'offset de l'appel (CS :EIP) est placé sur la pile automatiquement par le processeur, le RET dépile ces adresses.

B – UTILES AU DEBUGGAGE

CTRL-D : Permet de rentrer et de sortir de Softice. (bascule entre Windows et SoftIce)

CLS : permet d'effacer le contenu de la fenêtre des commandes.

CTRL-ALT-C : permet de centrer l'écran SoftIce.

T : Trace le programme ligne par ligne. De ce fait on rentre dans les fonctions, les interruptions, les apis, … et pour cela le programme utilise l'interruption matérielle 01 qui lui permet de tracer un prog. Pas à pas.

H : identique à la touche de fonction F2 = aide

TASK : affiche tout ce qui tourne en mémoire.

D xxx : affiche le contenu d'une mémoire, d'un registre dans la fenêtre des datas si cette dernière est active.
par ex : d eax affichera la valeur contenue dans le registre EAX, d 00419533 afficher la valeur contenue à l'adresse mémoire 00419533. Vous pouvez aussi visualiser leur contenu en cliquant directement sur le registre ou l'adresse mémoire que vous voyez à l'écran avec le bouton droit de la souris → « Display ». Après le D vous pouvez ajouter le type (Byte, Word , DblWord, …)de la valeur que vous voulez afficher. Par ex : DD ESP affiche le contenu de la pile sur 32 bits (soit 8 caractères hexadécimaux).=> très utile pour voir quels sont les paramêtres contenues dans la pile.

HWND : affiche toutes les boîtes de dialogues de tous les programmes (leurs références et leurs adresses sont temporaires)

HWND nom-du-programme : affiche toutes les boîtes de dialogues du programme concerné.

EXP : affiche toutes les API qui sont prises en compte par SoftIce

EXP nom-partiel-d'une-API : affiche les API commençant par les lettres que vous avez tappé. Ex : « exp get » vous affichera toutes les api dont le nom commence par get.

E : pour modifier le contenu d'une adresse mémoire.

R : pour modifier le contenu d'un registre.

S *adresse* L *longueur* '*ce_mot_ci*': cherche en mémoire une chaîne de caractères ce_mot _ci => très utiles pour chercher le propre sérial que l'on saisie et retrouver les endroits où sont effectués les comparaisons.
Pour ce faire, tapez « S 0 L FFFFFFFF 'le_numéro_que_vous_avez_entré' ». S pour search, 0 pour le début de la recherche en mémoire, et FFFFFFFF pour la fin. Soft-Ice va vous retourner la première adresse où il aura trouvé une trace de votre code, sous cette forme:

Pattern Found at 0030:00018D1

Tapez à nouveau la lettre "s" pour que Soft-ice continue ses recherches. Notez les adresses commençants par 800, et laisser tomber celles débutants par 000C (ce sont des adresses miroirs). Nous pourrons par la suite poser des points d'arrêts sur ces zones mémoires.

? : affiche le contenu d'une adresse mémoire ou d'un registre en valeurs hexa et décimale sur la ligne de commande.
En effet, lorsque l'on tape « d eax », on voit apparaître le contenu du registre EAX dans la fenêtre des datas. Mais un registre ne contient pas toujours un contenu visible en ASCII. Par exemple la valeur décimale 123456, pourra être codée en héxadécimale, et la frappe de « d eax » ne vous donnera rien dans la fenêtre data. Dans ce cas, « ? eax », vous affichera sur la ligne des commandes les deux valeurs, héxa et décimale.

A : permet de modifier une ligne de code. Vous aurez par exemple à remplacer des JZ par des JMP, ou l'envie de supprimer purement et simplement un CALL. Dans ce cas, il va falloir modifier les codes de l'exécutable.
En appuyant sur la lettre "a", puis en validant par [entrée] SoftICE se mettra en mode assembleur et il sera alors possible de réécrire le programme. Faites cependant attention à toujours remplacer le nombre d'octets équivalents, si vous ne voulez pas faire planter le programme en beauté. Par exemple, pour supprimer un call 00432479, dont le code héxa donnera E801E70200, vous entrerez 5 fois l'instruction NOP (90) pour équilibrer les comptes.

R FL Z : lors de comparaisons, ou de tests, les branchements du type JNE, JZ, etc… sont fonction de l'état d'un drapeau (le Zéro Flag), qui suivant le résultat du test vaudra 0 ou 1.
Une technique très utile pour inverser un drapeau consiste à taper "R FL Z" en ligne de commande. De cette façon, vous transformerez (et pour cette fois seulement) un JE en JNE et inversement.

WMSG : affiche la liste des messages windows (utilisé avec la commande BMSG).

C – LES POINTS D'ARRET

(extrait du document de CyberBobJr)

Note : pour une lisibilité facile, les instructions seront en italiques, les paramètres obligatoires en gras et les paramètres optionnels entre crochets (les crochets ne doivent pas être mis quand vous tapez l'instruction !).

Ils constituent une part importante dans le cracking d'un programme, c'est quasiment le nerf de guerre, il faut aboslument les maîtriser pour (bien) contrôler le programme, voiçi quelques exemples :

1. **Bpx : Breakpoint on execution**

   Syntaxe : *Bpx* **adresse** [C=count]

   Cette commande permet l'execution du programme jusqu'à **adresse**. **Count** permet de définir le nombre d'itérations avant l'arrêt du programme à **adresse**, dans **adresse** vous pouvez définir soit une valeur explicite (ex: **bpx 014F:023DF07C**), soit un registre (ex: **bpx cs:eip**).

   Il est possible de poser un breakpoint sur une api windows, ou sur tout autre fonction, à partir du moment où elle à été déclarée dans les exports de Soft-Ice, ces fonctions se trouvent généralement dans les Dll accompagnant un programme quelconque, pour cela il suffit de poser un **bpx *nomdelafonction***
   (ex: **bpx hmemcpy** ou **bpx getdlgitemtexta**)
   Attention toutefois sur les apis windows, certaines sont contenues dans les modules **user.dll** et d'autres dans le module **user32.dll**, donc vérifier bien la syntaxe de votre bpx, au besoin vous pourrez vérifier la fonction par l'utilisation de la commande **exp *nomdelafonction*** qui vous donnera le nom du module qui l'a contient.

2. **Bpm : Breakpoint on memory**

   Syntaxe : *Bpm[B|W|D]* **adresse** [R|W|RW|X] [qualifier value] [C=count]

   Le breakpoint on memory access peut-être de différent type : B pour un accès **byte** (1 octet par défaut), W pour **word** (2 octets) et D pour **Double Word** (4 octets). L'arrêt du programme se fera lorsqu'un accès mémoire se fera à **adresse**, l'accès peut être de type **R** (Read uniquement), **W** (Write, ecriture), **RW** (Lecture-ecriture), **X** (execution).
   Nous pouvons spécifier également la valeur dans la zone mémoire qui provoquera un breakpoint en spécifiant le

paramètre **[qualifier value]** (ex: **bpm ds:eax W eq 1** provoquera un breakpoint lorsque la zone située en ds:eax sera écrite avec la valeur hexadécimale 1), la syntaxe est la suivante : **eq** pour egal à, **gt** pour greater than (plus grand que)et **lt** pour less than (plus petit que), nous pouvons également spécifier un masque de bits (ex: bpm ds:eax W eq M 1xx0 00x1). Le paramètre count est identique au **bpx**.

Je vais prendre un exemple tout simple :
Vous posez un **bpx** sur hmemcpy, pour récupérer une entrée clavier par exemple, vous tracez la fonction jusqu'a l'instruction **movsw**, vous notez l'adresse (par exemple **15d7:00000000**) et vous posez un **bpm** dessus, vous relancez le programme et ... ça marche pas ! la raison en est simple : Soft-Ice à bien posé un bpm sur cette zone mémoire, mais votre programme n'y accède pas en utilisant l'adresse 15d7:00000000, or vous savez qu'une adresse mémoire peut être référencée par un **segment:offset** différent, et là, Soft-Ice ne sait pas ... donc pour résoudre le problème vous devez obtenir l'adresse linéaire de la mémoire, en particulier si vous utilisez 15d7:00000000, vous devez faire un **page 15d7:00000000**, cela vous donnera une adresse linéaire, il ne vous restera plus qu'à poser un **bpm 0030:*adresselinéaire***, et voilà, votre breakpoint sera correctement posé.
Si vous posez un **bpm *adressememoire* X** , SoftIce vous donnera la main dès qu'une instruction est faite à cet emplacement mémoire, ça peut-être une bonne alternative au bpx si ça ne marche pas, ou si vous voulez entrez dans un module non-déclaré.

3. **Bpr : Breakpoint on memory range**

Syntaxe : *Bpr* **adresse1 adresse2** [R|W|RW|T|TW] [C=count]

Le breakpoint on memory range vous permet de spécifier toute une plage d'adresses dans laquelle Soft-Ice s'arrêtera, les paramètres **R,W,RW** sont identiques au **Bpm**, idem pour le paramètre **count**.
J'ignore à quoi serve les paramètres **T** et **TW** si quelqu'un à une idée => mail !

exemple : **Bpr ds:eax ds:ebx R C=9**
La zone mémoire située entre **ds:eax** et **ds:ebx** est sous le couvert d'un breakpoint, le programme donnera la main a Soft-Ice lorsque 9 accès aux données auront été fait.

4. **BPio : Breakpoint on i/o port**

Syntaxe : *BPio* **port** [R|W|RW] [qualifer value] [C=count]

Breakpoint sur un port d'entrée/sortie, la syntaxe **[R|W|RW]** est identique à ce que nous avons vu précedemment, idem pour **qualifer value** et **count**. Le port doit être en hexadécimal.

exemple : **BPio 378 R**
Place un breakpoint sur toute tentative de lecture sur le port 378 (Lpt1)
Un autre exemple si nous reprenons la manière de retrouver notre sérial avec la commande S :
On trouvait : Pattern Found at 0030:00018D1
Vous poserez alors un BPR 18D1 18D1+(longueur Héxa de votre numéro de série) RW. La commande BPR provoquera un break à chaque sollicitation de cette zone en mémoire. La longueur des adresses à surveiller commence à la première lettre de votre code, et fini après la dernière (en prévision d'une comparaison byte à byte, ou d'une comparaison sur un ou plusieurs bytes à des endroits précis), RW pour Read and Write, lecture ou écriture. Tapez à nouveau la lettre "s" pour que Soft-ice continue ses recherches.

5. **BPint : Breakpoint on Interruption**

Syntaxe : *BPint* **interrupt-number** [[AL|AH|AX]=value] [C=count]

Voilà revenir nos chères interruptions du Dos, les interruptions du dos sont aux apis du windows, elles me permettent de me repérer dans un programme complexe, on sait où on se trouve ...
Interrupt-number spécifie le numéro de l'interruption à "breakpointer", en paramètres éventuels vous pouvez spécifier la zone inscrite dans le registre **AL,AH ou AX**, pour ce qui ne savent pas, AX peut contenir un paramètre definissant soit une sous-interruption (cf INT 21h qui contient au moins une 60aine de fonctions) soit un paramètre précis d'une autre intérruption.
La valeur **count** est identique à ce que nous avons vu précedemment. Les BPM et les PBInt sont limités au nombre de 4 pour les processeurs Pentium et au-delà.

exemple : **BPint 13 Ah=02**

Breakpoint sur l'interruption 13 (gestion des unités de disque), sous-fonction 02 (lecture d'un secteur précis sur un disque)

6. **BMsg : Breakpoint on windows message**

Syntaxe : *BMsg* **Window-handle** [L] [begin-message [end-message]] [C=count]

Breakpoint très important qui peut très souvent nous sortir d'un mauvais pas, on va y aller calmement et doucement :
Le **Window-handle** est un numéro attribué par le système, c'est un numéro d'identifiant qui sert à repertorier un element (fenètre, boutton, boite de liste, etc...)pour visualiser l'ensemble des window-handle il faut taper la commande **hwnd**, ou **hwnd programme_executable** pour visualiser les handles attribués pour ce programme uniquement.
Les messages windows sont standarts, ils ne peuvent changer, vous les trouverez tous par la commande **wmsg**, les messages peuvent-être soit explicite (ex : **WM_LBUTTONUP**) ou numérique (ex : 0202). Ainsi aussitôt qu'un message précis sera associé à un handle de fenêtre, Soft-Ice vous redonnera la main, il peut y avoir également un message de début et un message de fin... Mais nous y reviendrons quand moi-même j'aurais pigé ce truc.
Le paramètre count est une fois de plus identique.

exemple : **Bmsg 04c8 WM_COMMAND**
Breakpoint aussitôt qu'une commande est passée à l'handle de la fenêtre (WM_COMMAND peut signifier une fermeture de l'handle, un bouton pressé, etc ...).

7. BL : Breakpoint List

Liste les points d’arrêt en cours en leur attribuant des numéro. (numéro_du_bp = 00, 01, 02, …)

8. **BC numéro_du_bp : Breakpoint Clear**

Enlève le point d’arrêt n° numéro_du_bp . Pour supprimer tous les points d’arrêts, mettez * à la place de numéro_du_bp.
Exemple : BC 02

9. **BD numéro_du_bp : Breakpoint Disable**

Désactive le point d’arrêt n° numéro_du_bp. Pour désactiver tous les points d’arrêts, mettez * à la place de numéro_du_bp.

10. **BE numéro_du_bp : Breakpoint Enable**

Active le point d’arrêt n° numéro_du_bp. Pour réactiver tous les points d’arrêts, mettez * à la place de numéro_du_bp.

D - Les expressions :

La puissance de Soft-Ice provient également de ses macro commandes, nous allons voir comment utiliser les expressions pour poser des breakpoints, notons que la syntaxe est indentique à celle du C ou du C++.

Voiçi la liste des instructions disponibles (tj pompée sur la doc de CyberBobJr ;) :

Opérateur d'indirection	Exemple
->	ebp->8 (donne le dword pointé par ebp+8)
.	eax.1C (donne le dword pointé par eax+1c)
*	*eax (donne la valeur dword pointée par eax)
@	@eax (idem .)
Opérateur mathématique	**Exemple**
/	Si tu comprends pas, retourne à l'école !!!
%	Modulo
<< ou >>	eax << 2 ou eax >> 1 (décalage de n bits)
?+	Force la valeur en décimal (exe : ?+42)
?-	Force la valeur en décimal (exe : ?-42)
Opérateurs logiques	**Exemple**
!	NOT logique
&&	AND Logique
||	OR Logique
==	Comparaison d'égalité
!=	Comparaison de différence
<	Inférieur
>	Supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal

Les fonctions : certaines fonctions sont implémentées dans Soft-Ice, et vous permettent de gagner du temps, voiçi la liste :

Fonction	Exemple/commentaire
Byte	Obtenir les bytes de poids faible (ex : ? byte (0x1234) = 0x34)
Word	Obtenir le mot de poids faible (ex : ? word (0x12345678) = 0x5678)
Dword	Idem mais pour un double mot (ex : ? Dword (0xff) = 0x000000ff)
Hibyte	Obtenir les bytes de poids forts (ex : ? hibyte (0x1234) = 0x12)
Hiword	Idem mais pour un mot (ex : ? Hiword (0x12345678) = 0x1234)
Sword	Converti un byte en mot signé (ex: ? sword (0x80) = 0xff80)
Long	Converti un byte en mot long signé (ex: ? long (0xff) = 0xffffffff)
WSTR	Affiche la chaine unicode (ex: ? WSTR(eax))
Flat	Converti une adresse relative en une adresse adresse linéaire (ex: ? flat(fs:0) = 0xffdff000)
CFL	Carry flag (ex: ? CFL=type booléen)
PFL	Parity flag
AFL	Auxiliary flag
ZFL	Zero flag
SFL	Sign flag
OFL	Overflow flag
RFL	Resume flag
TFL	Trap flag
DFL	Direction flag
IFL	Interrupt flag
NTFL	Nested Task flag
DataAddr	Retourne l'adresse du premier élément affiché dans la zone data (ex: dd @dataaddr)
CodeAddr	Retourne l'adresse de la première instruction affichée dans la zone code (ex: ? codeaddr)
Process	KPEB (Kernel process Environnement Block) du process actif
Thread	KTEB (Kernel Thread Environnement Block) du thread actif
PID	Id du process actif

TID	Id du thread actif
BPcount	Nombre de breakpoints activés (cf plus bas)
BPtotal	Nombre total de breakpoints (cf plus bas)
BPmiss	Nombre de breakpoints échoués (cf plus bas)
BPindex	Index du breakpoint actuel

Les breakpoints :
BPCount vous donne le nombre de breakpoint dont la valeur est TRUE, il faut savoir que si vous posez un breakpoint avec une condition, cette condition peut être rempli (BPCount++) ou non (BPmiss++), de ce fait vous pouvez encore avoir plus de contrôle sur le programme. BPtotal ne compte pas les réussis ou raté, il défini le nombre total de fois ou le programme va les activer ... exemple :

Vous posez un **BPX EIP IF (EAX==1)**, 2 possibilités se présente :

Le breakpoint s'active si eax==1 => BPCount++
Le breakpoint ne s'active pas car eax!=1 => BPmiss++
De toute façon BPtotal++

On peut donc être plus précis en disant : **BPX EIP IF (EAX==1) && (BPCOUNT==5)**
Soft-Ice vous donnera la main lorsque eax=1 et que cette expression sera vérifiée 5 fois.

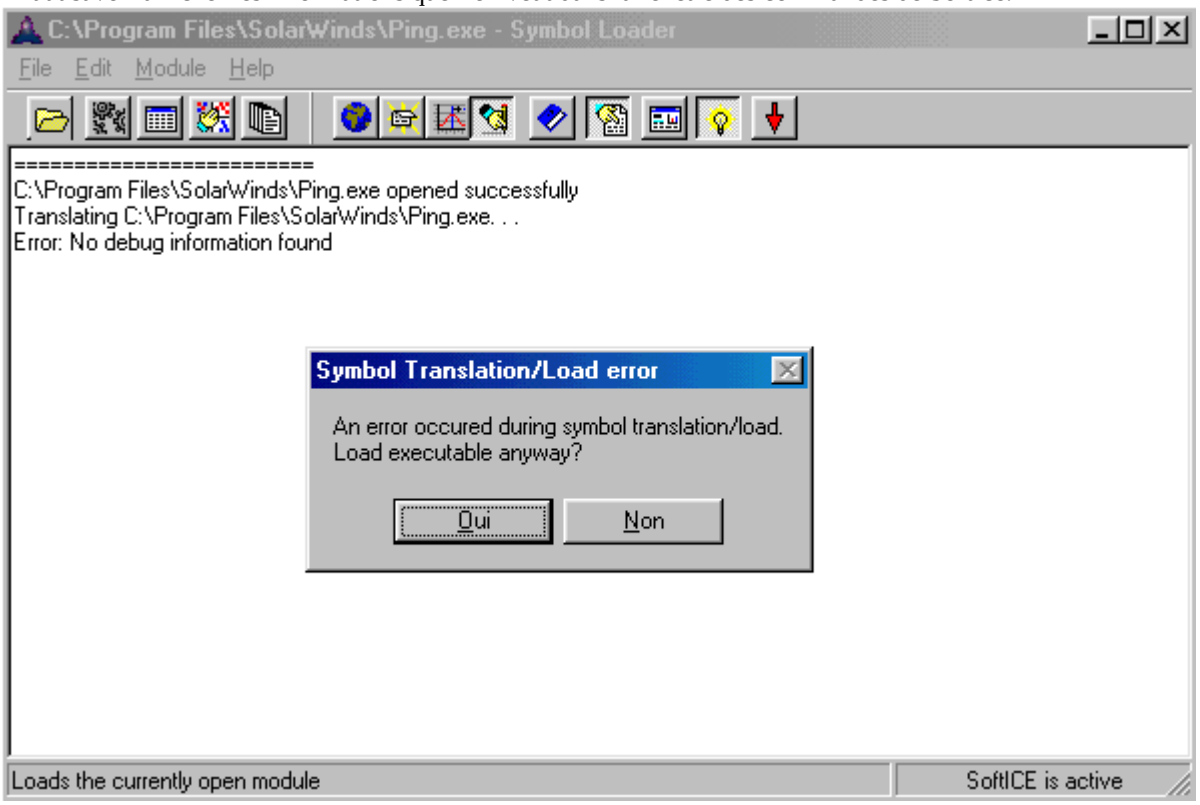
L'instruction IF :

L'instruction IF peut être utilisée de différentes manières, la première et la plus simple consiste à poser un breakpoint si et seulement si un registre contient une ou plusieurs valeurs, exemple :
BPX EIP IF (EAX==1) *// breakpoint en eip (pointeur courant) si le registre eax contient la valeur 1*
BPX EIP IF (EAX==1) || (EBX==1) *// breakpoint en eip si le registre eax=1* ***OU*** *ebx=1*
BPX EIP IF (EAX==1) && (EBX==1) *// breakpoint en eip si le registre eax=1* ***ET*** *ebx=1*

Ici la première expression est déja évaluée (comme en C) puis la seconde...

E - Sauvegarder le code dans un fichier texte :

Pour cela il faut ouvrir et charger votre programme à l'aide du Symbol Loader (LOADER32.EXE).
Il se peut qu'il y ait un message d'erreur (FIG.1), ce n'est pas grave : cliquer sur oui.
Il break donc sur le tout début du programme que vous venez de charger. L'utilisation est ensuite identique à celle de notre écran SoftICE.
Vous pouvez voir dans le Symbol Loader, un bouton qui sert à sauver le Softice History to a File. En d'autres termes, toutes les choses qui sont apparues dans la fenêtre des commandes sous Softice vont être sauvées dans un fichier. Maintenant que l'on sait cela, il faut savoir afficher les informations que l'on veut dans la fenêtre des commandes de SoftIce.



- **Afficher le code désassemblé dans la fenêtre des commandes de Softice :**

Si vous voulez afficher le code qui se trouve aux alentours de L'EIP courant, il suffit de faire :
U eip-XX L (2*XX)
Cette commande affiche le code désassemblé à partir de l'adresse eip-XX (XX étant un nombre) jusqu'à l'adresse (eip-XX) +(2*XX) soit eip+XX.
L (2*XX) veut dire que SoftIce doit désassembler 2XX bytes du code.
Autre exemple :
U 40100 L FF : ceci désassemble le code à partir de 40100h jusqu'à 401FFh (40100h + FFh).

- **Afficher de la mémoire dans la fenêtre des commandes de SoftIce :**

C'est très simple, il suffit de cacher la fenêtre des DATA à l'aide de la commande WD. Et ensuite de taper :
DB|DW|DD adresse_mémoire
Ceci va afficher soit en Byte (DB), soit en Word (DW), soit en Dword (DD) la mémoire à partir de adresse_mémoire.

- **Afficher les registres dans la fenêtre des commandes de SoftIce :**

Tapez simplement CPU et tous les états des registres vont apparaître comme par magie ;-)

5 - LES APIs

Voici donc la liste des APIs couramment utilisées lors de la pose de BPX.
Vous pourrez retrouver en détails la signification des APIs dans le document API.DOC.
Il y a des APIs pour les applis 16 bits et celles pour les applis 32 bits (j'ai donc mis un A entre ( ) pour faire la différence).
Exemple : Appli 16 bits = GetWindowText et appli 32 bits = GetWindowTextA.

APIs faisant appel à :

- **Lecture – écriture de fichier :**
  Appel générique sur la lecture et l'écriture d'un fichier, habituellement de nature binaire :
  ReadFile
  WriteFile
  Plus orienté sur la localisation de fichier :
  SetFilePointer
  GetSystemDirectory(A)
  Appel à la lecture et écriture de fichier de type *.INI :
  GetPrivateProfileString(A)
  GetPrivateProfileInt(A)
  WritePrivateProfileString (A)
  WritePrivateProfileInt(A)
  Au niveau des interruptions :
  Bpint 21 if (ah==3d)
  Bpint 2f if (ah==01)

- **Base de registres Windows :**
  Création ou suppression d'une clé dans la base de registres :
  RegCreateKey(A)
  RegDeleteKey(A)
  Lecture d'une valeur d'une clé dans la base de registres :
  RegQueryValue(A)
  RegOpenValueExa
  RegQueryValueExA
  Ouverture ou fermeture d'une clé :
  RegCloseKey(A)
  RegOpenKey(A)

- **Boîte de dialogue :**
  Saisie de données alphanumériques depuis une boîte de dialogue
  GetWindowText(A)
  GetDlgItemText(A)
  GetDlgItemInt
  Hmemcpy
  Ouverture d'une boîte dans laquelle nous avons fréquemment le message « Invalid registration » :
  MessageBox(A)
  MessageBoxExA
  MessageBeep
  ShowWindow
  D'autres possibilités sur l'apparition de texte :
  SENDMESSAGE
  WSPRINTF

- **Date et heure :**
  GetSystemTime
  GetLocalTime
  SystemTimeToFileTime
  SetTimer

- **Génération d'une fenêtre :**
  CreateWindow
  CreateWindowExA
  Bitblt (similaire à hmemcpy)

- **Appel au CD-ROM**

  GetDriveType(A) (si eax=00000005 alors CDROM présent)

  GetDriveType retourne les codes suivants :

Valeur	**Association**
0	Le lecteur ne peut être déterminé
1	Le répertoire Root n'existe pas
2	Lecteur amovible
3	Disque dur
4	Lecteur réseau
5	Lecteur CD-Rom
6	Disque Ram

  GetLogicalDrives(A)

  GetLogicalDriveStrings(A)

  Au niveau des interruptions :

  L'interruption 2f correspond à l'interruption utilisée par MSCDEX

  Bpint 2f, al=0 ah=15 Vérifie si mscdex installé et essaye de breaker sur l'accès aux fichiers.

- **Entrée numérique dans une fenêtre :**

  GetWindowWord

  GetWindowLong

- **Messages :**

  BMSG xxxx WM_GETTEXT (utile pour les mots des passes)

  BMSG xxxx WM_COMMAND (utile pour les boutons OK)