Stéphane Grare

MICROSOFT ®

Les Commandes Sous Windows

Pour maîtriser les commandes MS-DOS et les commandes de services Net

STEPHANE GRARE

Après avoir étudié plusieurs langages de programmation et développé de multitudes applications, cet ouvrage vous apportera une aide précieuse dans la maîtrise des différentes commandes sous MS-Dos (devenue l'invite de commande) ou les commandes de services Net. Cet ouvrage permet également de mieux comprendre les techniques utilisées par les hackers sur les réseaux grâce aux différentes commandes existantes, mais trop souvent ignorées par les plus novices.

INTRODUCTION

Vous connaissez les langages de programmations tels que le C+, C++, le Java, le HTML, les scripts tel le JavaScript ou VbScript… et bien d'autres encore ! Vous réalisez au quotidien des applications, des programmes, des sites web ou même des scripts et autres… Vous êtes administrateurs d'un réseau, géré la sécurité des différents accès aux répertoires de vos serveurs… Cet ouvrage va vous permettre de compléter vos connaissances en découvrant les bases des commandes indispensables sous Windows.

TABLE DES MATIERES

MS-Dos où l'invite de commande

MS-DOS, acronyme de Microsoft Disk Operating System, est un système d'exploitation doté d'une interface de ligne de commande utilisée sur les PC. À l'instar d'autres systèmes d'exploitation tels que OS/2, il traduit l'entrée au clavier par l'utilisateur en opération exécutable par l'ordinateur, et aussi supervise des opérations telles que les fonctions d'entrée/sortie de disque, de prise en charge vidéo, de contrôle du clavier et de nombreuses fonctions internes relatives à l'exécution de programmes et à la maintenance de fichiers.

Vous tapez les commandes MS-DOS à l'aide d'une fenêtre d'invite. Pour mettre fin à votre session MS-DOS, tapez **exit** dans la fenêtre d'invite de commandes à l'emplacement du curseur clignotant.

Le mode MS-DOS est un interpréteur de commandes dans lequel l'environnement MS-DOS est émulé en systèmes 32 bits, tels que Windows. Les programmes MS-DOS peuvent s'exécuter avec Windows et créer un program information file (PIF) qui apparaît sous la forme d'un raccourci sur le Bureau.

Ouvrez l'Invite de commandes.

Remarques

- Pour ouvrir l'Invite de commandes, cliquez sur **Démarrer**, pointez sur **Programmes**, puis sur **Accessoires**, puis cliquez sur **Invite de commandes** (ou recherche le programme cmd.exe normalement à partir de C:/Windows
- La création d'un fichier PIF pour un programme MS-DOS crée un raccourci vers le programme exécutable. Tous les paramètres enregistrés dans le fichier PIF sont contenus dans le raccourci.

Commandes du sous-système MS-DOS

Windows XP inclut des commandes 16 bits (non natives) pour le sous-système MS-DOS ainsi que pour d'autres sous-systèmes. On trouve parmi celles-ci des commandes anciennes, telles que **edlin** ou **graphics**, et des commandes spécifiques à MS-DOS, telles que **debug** ou **exe2bin**. Ces commandes 16 bits ne sont offertes que pour maintenir la compatibilité avec MS-DOS et MS OS/2 version 1.*x*.

D'autres commandes du sous-système MS-DOS, comme **share**, possèdent des fonctionnalités qui sont maintenant intégrées à Windows XP ou au sous-système MS-DOS. Les commandes sont conservées dans un souci de compatibilité avec les fichiers existants, mais n'ont pas d'effet dans la ligne de commande car la fonctionnalité est automatique.

Remarque

- Les commandes du sous-système MS-DOS 16 bits suivantes ne sont pas disponibles dans Windows XP 64-Bit Edition.

Pour plus d'informations, cliquez sur une commande : (Vous trouverez plus bas le descriptif complet)

- Append
- Debug
- Sous-commandes de Debug
- Edit
- Edlin
- Sous-commandes de Edlin
- Exe2bin
- Fastopen
- Forcedos
- Graphics
- Loadfix
- Loadhigh (lh)
- Mem
- Nlsfunc
- Setver
- Share

Pour configurer le sous-système MS-DOS, utilisez des commandes de configuration telles que **device** ou **lastdrive**. Placez ces commandes dans le fichier Config.nt du répertoire *racine_système*\System32 ou dans le fichier Config spécifié par le fichier PIF (Program Information File) d'une application. Ces commandes affectent uniquement le sous-système MS-DOS. Le sous-système MS-DOS ignore nombre de ces commandes, telles que **buffers** et **break**, car il fonctionne sans elles. Elles ne sont acceptées qu'à des fins de compatibilité.

Pour plus d'informations sur les commandes de configuration du sous-système MS-DOS, cliquez sur une commande : (vous trouverez plus bas le descriptif complet)

Append

Permet aux programmes d'ouvrir des fichiers de données se trouvant dans les dossiers spécifiés comme s'ils figuraient dans le dossier en cours. Utilisée sans paramètres, la commande **append** affiche la liste des répertoires ajoutés.

Syntaxe

append [**;**] [[*Lecteur***:**]*Chemin*[**;**...]] [**/x:**{**on**|**off**}][**/path:**{**on**|**off**}] [**/e**]

Paramètres

;
: Annule la liste des dossiers ajoutés.

[*Lecteur*:]*Chemin*
: Désigne le lecteur et le dossier à ajouter au dossier en cours. Si vous n'indiquez pas de lecteur, le lecteur en cours est le lecteur par défaut. Il est possible de spécifier plusieurs entrées [*Lecteur***:**]*Chemin* en les séparant par des points-virgules.

/x:{on|off}
: Indique si le sous-système MS-DOS doit effectuer des recherches dans les dossiers ajoutés lorsqu'il exécute des programmes. La syntaxe **/x:on** parcourt les dossiers ajoutés. La syntaxe **/x:off** ne parcourt pas les dossiers ajoutés.

/path:{on|off}
: Indique si un programme doit chercher un fichier de données dans les dossiers ajoutés lorsqu'un chemin d'accès est déjà inclus avec le nom du fichier. Le paramètre par défaut est **/path:on**.

/e
: Assigne la liste des dossiers ajoutés à une variable d'environnement nommée APPEND. Cette option de ligne de commande ne fonctionne que lors du premier emploi de la commande **append** après le démarrage du système.

/?
: Affiche de l'aide à l'invite de commandes.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Stockage de la liste des dossiers ajoutés

  L'option de ligne de commande **/e** employé avec **append** permet d'assigner à une variable d'environnement nommée APPEND la liste des dossiers ajoutés. Pour ce faire, utilisez d'abord la commande **append** avec l'option de ligne de commande **/e** seulement. Réexécutez ensuite la commande **append** et incluez les dossiers à ajouter. Il est impossible de spécifier **/e** et [*Lecteur***:**]*Chemin* sur la même ligne de commande.

- Ajout de plusieurs dossiers

  Pour ajouter plusieurs dossiers, séparez chacune des entrées par des points-virgules. Si vous réexécutez la commande **append** avec les paramètres [*Lecteur***:**]*Chemin*, le ou les dossiers spécifiés se substituent à tous ceux qui étaient désignés au moyen d'une commande **append** précédente.

- Utilisation de **dir**

Lorsque vous tapez **dir** pour afficher la liste des fichiers et des sous-répertoires d'un répertoire, le résultat de la commande **dir** ne comprend pas les noms des fichiers appartenant aux dossiers ajoutés.

- Résolution des conflits de noms de fichiers

  Si un fichier appartenant à un dossier ajouté porte le même nom qu'un fichier du dossier en cours, les programmes ouvrent le fichier du dossier en cours.

- Emploi d'**append** avec des programmes créant des fichiers

  Lorsqu'un programme ouvre un fichier stocké dans un dossier ajouté, ce fichier peut être trouvé comme s'il appartenait au dossier en cours. Si, lorsque ce programme enregistre le fichier, il en crée un autre sous le même nom, celui-ci est placé dans le dossier en cours et non pas dans le dossier ajouté. Vous pouvez utiliser la commande **append** pour les fichiers de données qui ne doivent pas être modifiés ou qui doivent l'être sans que de nouvelles copies ne soient créées. Les programmes de gestion de bases de données, par exemple, modifient souvent les fichiers de données sans créer de nouvelles copies. Les éditeurs de texte et les programmes de traitement de texte enregistrent généralement les fichiers de données modifiés en créant de nouvelles copies. Pour éviter toute confusion, il est préférable de ne pas utiliser **append** avec ces programmes.

- Utilisation de **/x:on** avec **path**

  Lorsque vous utilisez **/x:on** , vous pouvez exécuter un programme situé dans un dossier ajouté en tapant le nom du programme à l'invite de commandes. Généralement, vous utilisez la commande **path** pour spécifier des dossiers qui contiennent des programmes ; toutefois, vous n'avez pas besoin de l'utiliser pour indiquer un dossier ajouté contenant des programmes. Le sous-système MS-DOS trouve un programme dans un dossier ajouté en suivant l'ordre normal de recherche d'un programme : d'abord dans le dossier en cours, puis dans les dossiers ajoutés et enfin dans le chemin de recherche.

- Abréviation de **/x:on** en **/x**

  Vous pouvez abréger la syntaxe **/x:on** en **/x**. Pour ce faire, indiquez **/x:on** lors du premier emploi de la commande **append** après le démarrage du système. Ensuite, vous pouvez permuter les commutateurs **/x:on** et **/x:off**.

- Utilisation de **/e** avec **set**

  Vous pouvez utiliser **/e** avec la commande **set** pour afficher la liste des dossiers ajoutés. Pour plus d'informations sur les variables d'environnement et **set**, consultez Rubriques connexes.

Exemples

Pour autoriser des programmes à ouvrir des fichiers de données dans B:\Lettres et A:\Rapports, comme si ces fichiers se trouvaient dans le dossier en cours, tapez la syntaxe suivante :

append b:\lettres;a:\rapports

Pour ajouter les mêmes dossiers et conserver une copie de la liste des dossiers ajoutés dans l'environnement Windows XP (à condition qu'il s'agisse de la première exécution de la commande **append** après le démarrage du système), tapez la syntaxe suivante :

append /e

append b:\lettres;a:\rapports

Mise en forme : légende

Format	**Signification**
Italique	Informations que l'utilisateur doit fournir
Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Debug

Démarre Debug.exe, programme permettant de tester et de déboguer les fichiers exécutables MS-DOS. Utilisée sans paramètre, la commande **debug** démarre Debug.exe et affiche l'invite **debug** sous la forme d'un trait d'union (-).

Syntaxe

debug [[*Lecteur***:**][*Chemin*] *NomFichier* [*paramètres*]]

Paramètres

[*Lecteur*:][*Chemin*] *NomFichier*
Indique l'emplacement et le nom du fichier exécutable à tester.

paramètres
Indique toute information de ligne de commande requise par le fichier exécutable à tester.

?
Affiche la liste des sous-commandes de **debug**.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Utilisation des commandes du sous-système MS-DOS

  Debug est une commande du sous-système MS-DOS qui s'exécute sous WOW et NTVDM.

- Séparation des paramètres des commandes

  Vous pouvez séparer ces paramètres par des virgules ou des espaces, mais ces séparateurs ne sont requis qu'entre deux valeurs hexadécimales. Les commandes suivantes sont donc équivalentes :

  dcs:100 110

  d cs:100 110

  d,cs:100,110

- Utilisation des sous-commandes **debug**

  Vous pouvez utiliser plusieurs sous-commandes **debug**. Pour plus d'informations sur les sous-commandes **debug**, consultez Rubriques connexes.

Mise en forme : légende

Format	**Signification**
Italique	Informations que l'utilisateur doit fournir
Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Sous-commandes de Debug

Pour plus d'informations, cliquez sur une commande :

- Debug : a (assemble)
- Debug : c (compare)
- Debug : d (dump)
- Debug : e (enter)
- Debug : f (fill)
- Debug : g (go)
- Debug : h (hexadecimal)

- Debug : i (input)
- Debug : l (load)
- Debug : m (move)
- Debug : n (name)
- Debug : o (output)
- Debug : p (proceed)
- Debug : q (quit)
- Debug : r (register)
- Debug : s (search)
- Debug : t (trace)
- Debug : u (unassemble)
- Debug : w (write)
- Debug : xa (allocate expanded memory)
- Debug : xd (deallocate expanded memory)
- Debug : xm (map expanded memory pages)
- Debug : xs (display expanded memory status)
-

Debug : a (assemble)

Assemble les codes mnémoniques des 8086/8087/8088 directement dans la mémoire. Utilisée sans paramètre, la sous-commande **a** commence l'assemblage où il s'était arrêté précédemment.

Syntaxe

a [*adresse*]

Paramètres

adresse

Indique l'emplacement où vous tapez les codes mnémoniques du langage d'assemblage. Utilisez des valeurs hexadécimales pour l'*adresse* et tapez chaque valeur sans le caractère **h** à droite.

?

Affiche la liste des sous-commandes de **debug**.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Spécification d'entrées d'*adresse* valides

  Adresse est une désignation en deux parties contenant d'une part un registre de segment sous forme alphabétique ou une adresse de segment à 4 chiffres, et d'autre part une valeur de décalage. Vous pouvez omettre le registre de segment ou l'adresse de segment. CS est le segment par défaut des sous-commandes **debug** suivantes : **a**, **g**, **l**, **t**, **u** et **w**. Pour toutes les autres sous-commandes, le segment par défaut est DS. Toutes les valeurs numériques se présentent en notation hexadécimale. Vous devez inclure un signe deux-points entre le nom du segment et la valeur de décalage. Les deux adresses suivantes sont valides :

  CS:0100

  04BA:0100

- Spécification des préfixes mnémoniques

  Spécifiez un préfixe mnémonique à gauche du code opération (opcode) auquel il se réfère. La sous-commande **a** crée un code machine exécutable à partir d'instructions en langage d'assemblage. Toutes les valeurs numériques sont en notation hexadécimale et vous devez les taper sous forme de suite de 1 à 4 caractères.

- Emploi de codes mnémoniques

  Les codes mnémoniques qui ignorent les segments sont **cs:**, **ds:**, **es:** et **ss:**. Celui du retour far est **retf**. Les codes mnémoniques de manipulation de chaîne doivent indiquer explicitement la taille de la chaîne. Utilisez, par exemple, **movsw** pour déplacer des chaînes de mots (16 bits) et **movsb** pour déplacer des chaînes d'octets (8 bits).

- Assemblage de sauts et d'appels

  L'assembleur assemble automatiquement les sauts ou les appels (court, near ou far) selon le déplacement en octets vers l'adresse de destination. Pour procéder à des sauts ou à des appels différents, vous pouvez utiliser un préfixe **near** ou **far**. Par exemple :

```
-a0100:0500
0100:0500 jmp 502 ; saut court de 2 octets
0100:0502 jmp near 505 ; saut near de 3 octets
0100:0505 jmp far 50a ; saut far de 5 octets
```

Vous pouvez abréger le préfixe **near** sous la forme **ne**.

- Distinction des emplacements mémoire du type mot et du type octet

  Lorsqu'un opérande renvoie à un emplacement mémoire du type mot ou octet, vous devez spécifier le type des données au moyen du préfixe **word ptr** ou **byte ptr**, qu'il est possible d'abréger respectivement en **wo** et **by**. Par exemple :

```
dec wo [si]
neg byte ptr [128]
```

- Spécification des opérandes

  Debug.exe utilise la convention courante qui veut qu'un opérande encadré par des crochets [ ] renvoie à un emplacement mémoire. C'est la seule façon dont Debug.exe peut distinguer un opérande immédiat d'un opérande représentant un emplacement mémoire. Par exemple :

```
mov ax,21 ; charge AX avec 21h
mov ax,[21] ; charge dans AX le contenu de
;
; l'emplacement mémoire 21h
```

- Emploi de pseudo-instructions

  Deux pseudo-instructions courantes sont disponibles avec la sous-commande **a** : Le code opération **db**, qui assemble les valeurs d'octets directement dans la mémoire, et le code opération **dw**, qui assemble les mots directement en mémoire. Par exemple :

```
db 1,2,3,4,"VOICI UN EXEMPLE"
db VOICI DES GUILLEMETS : "'
db "VOICI UNE APOSTROPHE : '"
dw 1000,2000,3000,"BACH"
```

- Entrée de données dans des octets spécifiques

  Pour plus d'informations sur l'entrée de données dans des octets spécifiques à l'aide de la sous-commande **e** (enter), consultez Rubriques connexes.

- Désassemblage des octets

  Pour plus d'informations sur le désassemblage des octets à l'aide de la sous-commande **u** (unassemble), consultez Rubriques connexes.

Exemples

La sous-commande **a** prend en charge toutes les formes de commandes indirectes sur registres. Par exemple :

```
add bx,34[bp+2].[si-1]
pop [bp+di]
push [si] )
```

La sous-commande **a** prend également en charge tous les synonymes des codes opération. Par exemple :

```
loopz 100
loope 100
ja 200
jnbe 200
```

Pour les codes opération du 8087, vous devez utiliser le préfixe **wait** ou **fwait**. Par exemple :

```
fwait fadd st,st(3) ; cette ligne assemble
; un préfixe fwait
```

Mise en forme : légende

Format	Signification
Italique	Informations que l'utilisateur doit fournir
Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Debug : c (compare)

Compare deux zones de mémoire.

Syntaxe

c *adresse_plage*

Paramètres

plage
Requis. Indique les adresses de début et de fin de la première zone de mémoire à comparer ou son adresse de début et sa longueur.

adresse
Requis. Indique l'adresse de début de la seconde zone de mémoire à comparer.

?
Affiche la liste des sous-commandes de **debug**.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Spécification d'entrées de *plage* valides

  Utilisez *plage* avec une sous-commande **debug** pour spécifier une plage de mémoire. Vous pouvez choisir l'un des formats de *plage* suivants : une adresse de début et une adresse de fin ou une adresse de début et la longueur (représentée par **l**) de la plage. Les syntaxes suivantes, par exemple, spécifient toutes deux une plage de 16 octets commençant à l'adresse CS:100 :

  cs:100 10f

  cs:100 l 10

- Spécification d'entrées d'*adresse* valides

  Adresse est une désignation en deux parties contenant d'une part un registre de segment sous forme alphabétique ou une adresse de segment à 4 chiffres, et d'autre part une valeur de décalage. Vous pouvez omettre le registre de segment ou l'adresse de segment. CS est le segment par défaut des sous-commandes **debug** suivantes : **a**, **g**, **l**, **t**, **u** et **w**. Pour toutes les autres sous-commandes, le segment par défaut est DS. Toutes les valeurs numériques se présentent en notation hexadécimale. Vous devez inclure un signe deux-points entre le nom du segment et la valeur de décalage. Les deux adresses suivantes sont valides :

  CS:0100

  04BA:0100

- Si les zones mémoire spécifiées par *plage* et *adresse* sont identiques, la sous-commande **c** n'affiche rien et vous ramène à l'invite **debug**. Si elles sont différentes, **c** les affiche sous la forme suivante :

  adresse1 octet1 octet2 adresse2

Exemples

Pour comparer le bloc de mémoire allant de 100h à 10Fh à celui allant de 300h à 30Fh, tapez :

c100,10f 300

ou

c100l10 300

Ces deux commandes génèrent le résultat suivant (dans l'hypothèse où DS a pour valeur 197F) :

197F:0100 4D E4 197F:0300
197F:0101 67 99 197F:0301
197F:0102 A3 27 197F:0302
197F:0103 35 F3 197F:0303
197F:0104 97 BD 197F:0304
197F:0105 04 35 197F:0305
197F:0107 76 71 197F:0307
197F:0108 E6 11 197F:0308
197F:0109 19 2C 197F:0309
197F:010A 80 0A 197F:030A
197F:010B 36 7F 197F:030B
197F:010C BE 22 197F:030C
197F:010D 83 93 197F:030D
197F:010E 49 77 197F:030E
197F:010F 4F 8A 197F:030F

Les adresses 197F:0106 et 197F:0306 ne figurent pas dans la liste. Cela signifie que les valeurs de ces adresses sont identiques.

Mise en forme : légende

Format	**Signification**
Italique	Informations que l'utilisateur doit fournir
Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Debug : d (dump)

Affiche le contenu d'une plage d'adresses mémoire. Utilisée sans paramètre, la sous-commande **d** affiche le contenu de 128 octets, qui commence à partir de la fin de la plage d'adresses spécifiée dans la sous-commande **d** précédente.

Syntaxe

d [*plage*]

Paramètres

plage

Indique les adresses de début et de fin, ou l'adresse de début et la taille de la zone mémoire dont vous souhaitez afficher le contenu.

?

Affiche la liste des sous-commandes de **debug**.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Spécification d'entrées de *plage* valides

  Utilisez *plage* avec une sous-commande **debug** pour spécifier une plage de mémoire. Vous pouvez choisir l'un des formats de *plage* suivants : une adresse de début et une adresse de fin ou une adresse de début et la longueur (représentée par **l**) de la plage. Les syntaxes suivantes, par exemple, spécifient toutes deux une plage de 16 octets commençant à l'adresse CS:100 :

  cs:100 10f

cs:100 l 10

- Lorsque vous utilisez la sous-commande **d**, Debug.exe affiche le contenu de la mémoire en deux parties : une partie hexadécimale (chaque valeur d'octet étant représentée au format hexadécimal) et une partie ASCII (qui représente chaque valeur d'octet sous la forme d'un caractère ASCII). Dans la section ASCII de l'affichage, chaque caractère non imprimable est représenté par un point (.). Chaque ligne d'affichage indique le contenu de 16 octets, un trait d'union apparaissant entre le huitième et le neuvième octets. Chaque ligne d'affichage commence sur la ligne de démarcation d'une tranche de 16 octets.
- Pour plus d'informations sur l'affichage du contenu des registres à l'aide de la sous-commande **r** (register), consultez Rubriques connexes.

Exemples

Tapez :

dcs:100 10f

Debug.exe affiche le contenu de cette plage sous la forme suivante :

04BA:0100 54 4F 4D 00 53 41 57 59-45 52 00 00 00 00 00 00.....

Si vous tapez la sous-commande **d** sans paramètre, Debug.exe affiche le résultat sous la même forme que dans l'exemple précédent. Chaque ligne de la fenêtre Invite de commandes commence par une adresse supérieure de 16 octets à celle de la ligne précédente (ou 8 octets si vous utilisez un écran de 40 colonnes). Pour chaque sous-commande **d** suivante que vous tapez sans paramètre, Debug.exe affiche les octets immédiatement après ceux affichés précédemment.

Pour afficher le contenu de 20h octets à partir de CS:100, tapez :

dcs:100 l 20

Pour afficher le contenu de tous les octets de la plage des lignes 100h à 115h du segment CS, tapez :

dcs:100 115

Mise en forme : légende

Format	**Signification**
Italique	Informations que l'utilisateur doit fournir
Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Debug : e (enter)

Entre des données en mémoire à l'adresse spécifiée.

Syntaxe

e *adresse* [*liste*]

Paramètres

adresse
Requis. Indique le premier emplacement mémoire où vous voulez entrer les données.

liste
Représente les données que vous voulez entrer dans des octets de mémoire successifs.

?
Affiche la liste des sous-commandes de **debug**.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Spécification d'entrées d'*adresse* valides

  Adresse est une désignation en deux parties contenant d'une part un registre de segment sous forme alphabétique ou une adresse de segment à 4 chiffres, et d'autre part une valeur de décalage. Vous pouvez omettre le registre de segment ou l'adresse de segment. CS est le segment par défaut des sous-commandes **debug** suivantes : **a**, **g**, **l**, **t**, **u** et **w**. Pour toutes les autres sous-commandes, le segment par défaut est DS. Toutes les valeurs numériques se présentent en notation hexadécimale. Vous devez inclure un signe deux-points entre le nom du segment et la valeur de décalage. Les deux adresses suivantes sont valides :

  CS:0100

  04BA:0100

- Utilisation du paramètre *adresse*

  Si vous spécifiez une valeur pour l'*adresse* sans en indiquer une pour le paramètre facultatif *liste*, Debug.exe affiche l'adresse et son contenu, répète l'adresse sur la ligne suivante et attend que vous entriez des données. À ce stade, vous pouvez effectuer l'une des opérations suivantes :

  - Remplacer la valeur de l'octet. Il vous suffit pour cela de taper une nouvelle valeur à la suite de la valeur actuelle. Si la valeur tapée ne représente pas une valeur hexadécimale valide ou si elle contient plus de deux chiffres, Debug.exe ne renvoie pas en écho le caractère non valide ou superflu.
  - Avancer jusqu'à l'octet suivant. Il vous suffit pour cela d'appuyer sur la touche ESPACE. Pour changer la valeur de cet octet, tapez une nouvelle valeur à la suite de la valeur actuelle. Si vous dépassez une ligne de démarcation de 8 octets lorsque vous appuyez sur ESPACE, Debug.exe commence une nouvelle ligne d'affichage et affiche la nouvelle adresse au début de cette ligne.
  - Revenir à l'octet précédent. Pour cela, appuyez sur la touche -. Vous pouvez aussi appuyer sur cette touche à plusieurs reprises pour reculer de plus de 1 octet. Lorsque vous appuyez sur -, Debug.exe commence une nouvelle ligne et affiche l'adresse en cours ainsi que la valeur de l'octet.
  - Arrêter la sous-commande **e**. Pour cela, appuyez sur la touche ENTRÉE. Vous pouvez appuyer sur cette touche à partir de n'importe quel octet.

- Utilisation du paramètre *liste*

  Si vous spécifiez des valeurs pour le paramètre *liste*, la sous-commande **e** remplace séquentiellement les valeurs existantes des octets par les valeurs de la liste. En cas d'erreur, aucun octet n'est remplacé.

  Les valeurs de la *liste* peuvent être des valeurs d'octet hexadécimales ou des chaînes. Pour séparer les valeurs, utilisez un espace, une virgule ou un caractère de tabulation. Les chaînes doivent obligatoirement être encadrées par des apostrophes (**'***chaîne***'**) ou des guillemets (**"***chaîne***"**).

- Assemblage des codes mnémoniques

  Pour plus d'informations sur l'assemblage des codes mnémoniques à l'aide de la sous-commande **a** (assemble), consultez Rubriques connexes.

- Affichage du contenu d'une zone de mémoire

  Pour plus d'informations sur l'affichage du contenu d'une partie de la mémoire à l'aide de la sous-commande **d** (dump), consultez Rubriques connexes.

Exemples

Tapez :

ecs:100

Debug.exe affiche le contenu du premier octet sous la forme suivante :

04BA:0100 EB._

Pour remplacer cette valeur par 41, tapez **41** au niveau du point d'insertion, comme suit :

04BA:0100 EB.**41**_

Il est possible d'utiliser une seule sous-commande **e** pour taper des valeurs d'octets consécutives. Au lieu d'appuyer sur la touche ENTRÉE après avoir tapé la nouvelle valeur, appuyez sur la touche ESPACE. Debug.exe affiche la valeur suivante. Dans cet exemple, si vous appuyez trois fois sur ESPACE, Debug.exe affiche les valeurs ci-dessous :

04BA:0100 EB.41 10. 00. BC._

Pour remplacer la valeur hexadécimale BC par 42, tapez **42** au niveau du point d'insertion, comme suit :

04BA:0100 EB.41 10. 00. BC.**42**_

Pour remplacer 10 par 6F, appuyez sur la touche - à deux reprises afin de revenir à l'adresse 0101 (valeur 10). Debug.exe affiche les éléments suivants :

04BA:0100 EB.41 10. 00. BC.42-
04BA:0102 00.-
04BA:0101 10._

Pour remplacer la valeur, tapez **6F** au niveau du point d'insertion, comme suit :

04BA:0101 10.**6F**_

Pour mettre fin à la sous-commande **e** et revenir à l'invite **debug**, appuyez sur ENTRÉE.

L'exemple suivant illustre l'entrée d'une chaîne :

eds:100 "Voici l'exemple texte"

Cette chaîne remplit 24 octets à partir de DS:100.

Mise en forme : légende

Format	**Signification**
Italique	Informations que l'utilisateur doit fournir
Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Debug : f (fill)

Remplit les adresses de la zone mémoire désignée à l'aide des valeurs que vous spécifiez.

Syntaxe

f *plage liste*

Paramètres

plage
Requis. Indique les adresses de début et de fin de la zone mémoire à remplir ou son adresse de début et sa longueur.

liste
Requis. Spécifie les données à entrer.

?
Affiche la liste des sous-commandes de **debug**.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Spécification d'entrées de *plage* valides

  Utilisez *plage* avec une sous-commande **debug** pour spécifier une plage de mémoire. Vous pouvez choisir l'un des formats de *plage* suivants : une adresse de début et une adresse de fin ou une adresse de début et la longueur

(représentée par **l**) de la plage. Les syntaxes suivantes, par exemple, spécifient toutes deux une plage de 16 octets commençant à l'adresse CS:100 :

cs:100 10f

cs:100 l 10

- Spécification des données

  Les données peuvent être spécifiées sous forme hexadécimale ou ASCII. Toutes les données précédemment stockées, le cas échéant, à l'adresse spécifiée sont perdues.

- Utilisation de *liste*

  La *liste* peut être constituée de nombres hexadécimaux ou d'une chaîne encadrée par des guillemets (**"***chaîne***"**).

Exemples

Pour remplir les emplacements mémoire 04BA:100 à 04BA:1FF à l'aide de cinq valeurs spécifiques (par exemple, 42, 45, 52, 54 et 41) et répéter les cinq valeurs jusqu'à ce que Debug.exe ait rempli les 100h octets, tapez :

f04ba:100l100 42 45 52 54 41

Mise en forme : légende

Format	**Signification**
Italique	Informations que l'utilisateur doit fournir
Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Debug : g (go)

Exécute le programme qui se trouve en mémoire. Utilisée sans paramètre, la sous-commande **g** s'exécute à partir de l'adresse courante dans les registres CS:IP.

Syntaxe

g [**=***adresse*] [*points_arrêt*]

Paramètres

adresse
Spécifie l'adresse du programme actuellement en mémoire à partir de laquelle vous souhaitez exécuter celui-ci.

points_arrêt
Spécifie 1 à 10 points d'arrêt temporaires que vous pouvez définir comme faisant partie de la sous-commande **g**.

?
Affiche la liste des sous-commandes de **debug**.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Spécification d'entrées d'*adresse* valides

  Adresse est une désignation en deux parties contenant d'une part un registre de segment sous forme alphabétique ou une adresse de segment à 4 chiffres, et d'autre part une valeur de décalage. Vous pouvez omettre le registre de segment ou l'adresse de segment. CS est le segment par défaut des sous-commandes **debug** suivantes : **a**, **g**, **l**, **t**, **u** et **w**. Pour toutes les autres sous-commandes, le segment par défaut est DS. Toutes les valeurs numériques se présentent en notation hexadécimale. Vous devez inclure un signe deux-points entre le nom du segment et la valeur de décalage. Les deux adresses suivantes sont valides :

CS:0100

04BA:0100

- Utilisation du paramètre *adresse*

  Faites précéder le paramètre *adresse* par un signe égal (=) afin de distinguer *adresse* des adresses des points d'arrêt (*points_d'arrêt*).

- Spécification des points d'arrêt

  Le programme s'arrête au premier point d'arrêt rencontré, quelle que soit la position de ce dernier dans la liste *points_arrêt*. À chacun de ces points, Debug.exe remplace l'instruction d'origine par un code d'interruption.

  Lorsque le programme atteint un point d'arrêt, Debug.exe rétablit les instructions d'origine aux adresses de tous les points d'arrêt et affiche le contenu de tous les registres, l'état de tous les indicateurs et la forme décodée de la dernière instruction exécutée. Debug.exe affiche les mêmes informations que lorsque vous utilisez la sous-commande **r** (register) et spécifiez l'adresse des points d'arrêt.

  Si vous n'arrêtez pas le programme à l'un des points d'arrêt, Debug.exe ne remplace pas les codes d'interruption par les instructions d'origine.

  Vous ne pouvez définir des points d'arrêt qu'aux adresses contenant le premier octet d'un code opération 8086. Au-delà de 10 points d'arrêt, Debug.exe affiche le message suivant :

  bp erreur

- Utilisation du pointeur de pile de l'utilisateur

  Le pointeur de pile de l'utilisateur doit être valide et doit posséder 6 octets disponibles pour la sous-commande **g**. La sous-commande **g** utilise une instruction **iret** pour passer au programme à tester. Debug.exe définit le pointeur de pile de l'utilisateur et place les indicateurs, le registre des segments de code ainsi que le pointeur d'instruction dans la pile de l'utilisateur. (Si la pile de l'utilisateur n'est pas valide ou est trop petite, le système d'exploitation peut échouer.) Debug.exe place en outre un code d'interruption (0CCh) à l'adresse ou aux adresses du point d'arrêt spécifié.

- Redémarrage d'un programme

  N'essayez pas de redémarrer un programme après que le système affiche le message suivant :

  Fin normale du programme.

  Pour exécuter correctement le programme, rechargez-le au moyen des sous-commandes **n** (Name) et **l** (Load).

- Pour plus d'informations sur l'exécution d'une boucle, d'une instruction de chaîne répétée, d'une interruption au niveau du logiciel ou d'un sous-programme à l'aide de la sous-commande **p** (proceed), consultez Rubriques connexes.
- Pour plus d'informations sur l'exécution d'une instruction à l'aide de la sous-commande **t** (trace), consultez Rubriques connexes.

Exemples

Pour exécuter le programme qui se trouve en mémoire jusqu'à l'adresse 7550 du point d'arrêt dans le segment CS, tapez :

gcs:7550

Debug.exe affiche le contenu des registres et l'état des indicateurs, puis met fin à la sous-commande **g**.

Pour définir deux points d'arrêt, tapez :

gcs:7550, cs:8000

Si vous retapez la sous-commande **g** après que Debug.exe a rencontré un point d'arrêt, l'exécution commence au niveau de l'instruction qui suit ce point, et non pas à l'adresse de début.

Mise en forme : légende

Format	**Signification**
Italique	Informations que l'utilisateur doit fournir

Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Debug : h (hexadecimal)

Effectue des opérations arithmétiques hexadécimales sur deux paramètres que vous spécifiez.

Syntaxe

h *valeur1valeur2*

Paramètres

valeur1
Requis. Représente n'importe quel nombre hexadécimal compris entre 0 et FFFFh.

valeur2
Requis. Représente un second nombre hexadécimal compris entre 0 et FFFFh.

?
Affiche la liste des sous-commandes de **debug**.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Debug.exe ajoute les deux paramètres spécifiés, puis retranche le second du premier. Les résultats de ces calculs sont affichés sur une seule ligne : d'abord la somme, puis la différence.

Exemples

Tapez :

h19f 10a

Debug.exe effectue les calculs et affiche le résultat suivant :

02A9 0095

Mise en forme : légende

Format	**Signification**
Italique	Informations que l'utilisateur doit fournir
Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Debug : i (input)

Lit et affiche une seule valeur d'octet provenant du port spécifié.

Syntaxe

i *port*

Paramètres

port

Requis. Spécifie le port d'entrée à l'aide de l'adresse. Cette dernière peut être une valeur 16 bits.

?

Affiche la liste des sous-commandes de **debug**.

Remarques

- Windows XP n'utilise pas cette commande. Elle n'est acceptée que pour la compatibilité avec les fichiers de MS-DOS.
- Pour plus d'informations sur l'envoi d'un octet à un port de sortie à l'aide de la sous-commande **o** (output), consultez Rubriques connexes.

Exemples

Tapez :

i2f8

Si la valeur de l'octet sur le port est 42h, Debug.exe lit l'octet et affiche la valeur comme suit :

42

Mise en forme : légende

Format	**Signification**
Italique	Informations que l'utilisateur doit fournir
Gras	Éléments que l'utilisateur doit taper exactement tels quels
Points de suspension (...)	Paramètre pouvant être répété plusieurs fois dans une ligne de commande
Entre parenthèses ([])	Éléments facultatifs
Entre accolades ({}) ; éléments sélectionnés séparés par une barre verticale (|). Exemple : {pair|impair}	Ensemble de sélections (une seule sélection possible)
Police Courier	Code ou données du programme

Debug : l (load)

Charge un fichier ou le contenu de secteurs spécifiques du disque en mémoire. Utilisée sans paramètre, la sous-commande **l** charge le fichier désigné sur la ligne de commande **debug** en mémoire à partir de l'adresse CS:100. Debug.exe assigne aussi aux registres BX et CX le nombre d'octets chargés. En l'absence d'un nom de fichier sur la ligne de commande **debug**, le fichier chargé est le fichier le plus récemment désigné au moyen de la sous-commande **n** (name).

Syntaxe

l [*adresse*]

l [*adresse*] [*Lecteur*] [*PremierSecteur*] [*nombre*]

Paramètres

adresse

Désigne l'adresse mémoire où vous voulez charger le fichier ou le contenu des secteurs. En l'absence de ce paramètre, Debug.exe utilise l'adresse actuelle dans le registre CS.

Lecteur

Désigne le lecteur qui contient le disque dont vous voulez lire des secteurs spécifiques. Cette valeur est numérique : 0 = A, 1 = B, 2 = C, etc.

PremierSecteur

Représente, sous forme hexadécimale, le numéro du premier secteur à partir duquel vous voulez charger le contenu.

nombre

Représente, sous forme hexadécimale, le nombre de secteurs consécutifs dont vous voulez charger le contenu. Utilisez uniquement *Lecteur*, *PremierSecteur* et *nombre* si vous voulez charger le contenu de secteurs spécifiques au lieu du fichier spécifié sur la ligne de commande **debug** ou dans la sous-commande **n** (name) la plus récente.

?

Affiche la liste des sous-commandes de **debug**.