Première partie

Le système d'exploitation comme machine virtuelle

Chapitre 2

Programmer avec le DOS

Nous avons vu comment programmer en langage d'assemblage, en utilisant d'abord seulement les instructions du microprocesseur puis, dans une seconde étape, en utilisant les *appels système* du BIOS, en particulier pour les entrées-sorties. Dans le cas de MS-DOS, comme pour le BIOS, les appels système sont implémentés sous forme d'interruptions logicielles. Nous allons donc décrire les interruptions de MS-DOS dans ce premier chapitre.

Il y a donc, du point de vue de la programmation, trois types de routines de service des interruptions logicielles :

- les **interruptions du BIOS**, dont les fonctions ont été conçues par les créateurs de l'IBM-PC ; les routines de service de ces interruptions se trouvent pour certaines sur une puce ROM située sur la carte mère et, pour d'autres, celles concernant certains périphériques comme le disque dur, sur une puce ROM située sur la carte de ce périphérique ;

- les **interruptions MS-DOS**, conçues par les créateurs (des différentes versions) du système d'exploitation MS-DOS ; elles sont chargées à chaque démarrage de ce système ;

- les **interruptions propriétaires**, conçues pour une ligne de matériel donné ou par l'utilisateur.

Nous avons vu les interruptions du BIOS. Nous allons voir ici, de façon presque exhaustive, les interruptions du DOS. Elles simplifient grandement la vie du programmeur.

Les interruptions de MS-DOS sont principalement des fonctions de l'interruption `INT 21h`. Il y a environ 90 fonctions, chacune étant identifiée par son **numéro de fonction**, à placer dans le registre `AH` avant d'appeler `INT 21h`.

2.1 Les entrées-sorties standard

Nous avons déjà dit plusieurs fois que les ordinateurs modernes sont orientés mots, et non nombres ; nous allons donc considérer la saisie d'une chaîne de caractères au clavier et l'affichage d'une chaîne de caractères à l'écran. On les appelle les **entrées-sorties standard**.

Profitons-en pour nous constituer une bibliothèque `stdio.lib` consacrée aux entrées-sorties standard.

2.1.1 Fonction `02h` d'affichage d'un caractère

Description de l'interruption.- On peut afficher un caractère à l'écran en utilisant la fonction `02h` de l'interruption MS-DOS `21h`, en plaçant le caractère à afficher (plus précisément son code ASCII) dans le registre `DL` avant l'appel de l'interruption.

Application.- Écrivons une procédure `putchar` permettant d'afficher un caractère, celui-ci (son code ASCII) étant placé dans le registre `AL`.

```
TITLE putchar.asm
      .model small
      .code
PUBLIC putchar
putchar proc
      push ax
      push dx
      mov dl, al
      mov ah, 02
      int 21h
      pop dx
      pop ax
      ret
putchar endp
      end
```

Assemblons ce module et plaçons-le dans le bibliothèque, après avoir créé cette dernière :

```
masm putchar.asm
lib stdio;
lib stdio +putchar
```

Désormais nous ne rappellerons plus cette étape.

Test.- Écrivons un programme `tputchar.asm` de test de cette procédure, qui affiche 'ab' à l'écran et va à la ligne.

```
TITLE tputchar.asm
      .model small
      .stack 256
      .code
EXTRN putchar:proc
debut:
      mov al, 'a'
      call putchar
```

```
mov al, 'b'
call putchar
mov al, 13 ; CR
call putchar
mov al, 10 ; LF
call putchar
mov ax,4c00h
int 21h
end debut
```

2.1.2 Fonction `01h` de saisie d'un caractère avec écho

Description de l'interruption.- La saisie d'un caractère au clavier avec écho à l'écran est l'objet de la fonction `01h` de l'interruption DOS `21h`. Son comportement exact est celui-ci.

Le BIOS maintient un tampon circulaire de 15 caractères stockant les caractères dans l'ordre dans lequel on a appuyé sur les touches. Lorsque le tampon est plein (avant qu'un programme ait utilisé ces valeurs), l'ordinateur émet un bip et les touches supplémentaires sur lesquelles on appuie sont ignorées.

La fonction `01h` prend un caractère dans ce tampon, éventuellement en attend un si le tampon est vide, affiche ce caractère en écho sur la sortie standard (le moniteur) et stocke le caractère dans le registre `AL`.

En fait `AL` contient le code ASCII du caractère s'il s'agit d'un caractère affichable et 0 sinon. Si `AL` contient 0 alors le registre `AH` contient le numéro de la fonction, par exemple <Home>, <F1> ou <PgUp> ; il faut faire appel une deuxième fois à la fonction pour obtenir le caractère.

Application.- Écrivons une procédure `getche` permettant de saisir un caractère affichable, à placer dans le registre `AL`, avec écho à l'écran.

```
TITLE getche.asm
     .model small
     .code
PUBLIC getche
getche proc
       mov   ah,01   ; fonction saisie
       int   21h     ; appel de l'interruption
       ret
getche endp
       end
```

Assemblons ce module et plaçons-le dans le bibliothèque `stdio.h` :

```
masm getche.asm
lib stdio +getche
```

Désormais nous ne rappellerons plus cette étape.

Application.- Écrivons un programme `tgetche.asm` permettant de saisir deux caractères (affichables) et de les afficher dans l'ordre inverse sur la ligne suivante, en utilisant la procédure `putchar` déjà vue.

```
TITLE tgetche.asm
      .model small
      .stack 256
      .code
EXTRN putchar:proc, getche:proc
debut:
      call getche
      mov dl, al        ; sauvegarde du premier caractere
      call getche
      mov dh, al        ; sauvegarde du second caractere
      mov al, 13 ; CR
      call putchar
      mov al, 10 ; LF
      call putchar      ; passage a la ligne
      mov al, dh        ; rappel du second caractere
      call putchar
      mov al, dl        ; rappel du premier caractere
      call putchar
      mov al, 13
      call putchar
      mov al, 10
      call putchar
      mov ax,4c00h
      int 21h
      end debut
```

2.1.3 Fonction 08h de saisie d'un caractère sans écho

Description de l'interruption.- La fonction `08h` de l'interruption DOS `21h` agit comme la fonction `01h`, sauf qu'il n'y a pas d'écho à l'écran. Ceci peut être utile, par exemple dans le cas de la saisie d'un mot de passe.

Application.- Écrivons une procédure `getch` permettant de saisir un caractère affichable, dont le code ASCII est alors placé dans le registre `AL`, sans écho à l'écran.

```
TITLE getch.asm
      .model small
      .code
PUBLIC getch
getch proc
      mov ah, 8h
      int 21h
      ret
getch endp
      end
```

Assemblons ce module et plaçons-le dans le bibliothèque `stdio.h`.

Application.- Écrivons un programme `tgetch.asm` permettant de saisir un code secret, disons de deux caractères et d'afficher une étoile à chaque fois qu'un caractère est entré (afin de visualiser la saisie). On ne fait rien ici du code entré, qui est donc perdu.

```
TITLE tgetch.asm
      .model small
      .stack 256
      .code
EXTRN putchar:proc, getch:proc
debut:
      call getch
      mov dl, al       ; sauvegarde du premier caractere
      mov al, '*'      ; affichage d'une etoile
      call putchar
      call getch
      mov dh, al       ; sauvegarde du second caractere
      mov al, '*'
      call putchar
      mov al, 13
      call putchar
      mov al, 10
      call putchar
      mov ax,4c00h
      int 21h
      end debut
```

2.1.4 Saisie d'une chaîne de caractères

2.2 L'interruption 33h de manipulation de la souris

Nous supposons, évidemment, que le lecteur est familiarisé avec l'utilisation d'une souris. Nous allons voir ici comment la programmer. Toutes les opérations concernant la souris sont des fonctions de l'interruption 33h de MS-DOS. Une fonction est définie par une valeur placée dans le registre AX (et non AH, contrairement à la plupart des autres interruptions).

2.2.1 Fonction 00h d'initialisation de la souris

L'initialisation est la première commande à exécuter avant d'utiliser la souris dans un programme ; il suffit de le faire une seule fois.

Syntaxe.- La fonction est 00h, à placer dans AX. Si l'opération ne réussit pas alors l'interruption renvoie 0000h dans AX. Sinon elle renvoie FFFFh dans AX, le nombre de boutons dans BX, place le *pointeur de souris* au centre de l'écran (mais il n'est pas visible pour l'instant) et effectue quelques petites autres choses.

2.2.2 Fonctions 01h et 02h d'affichage et de transparence du pointeur

Indicateur de pointeur.- Le pilote de la souris contient un **indicateur de pointeur** qui détermine si le pointeur doit être visible ou non. Le pointeur est affiché si l'indicateur a la valeur zéro et ne l'est pas pour les autres valeurs. Lors de l'initialisation, le pointeur prend la valeur -1 = FFh.

Affichage et transparence du pointeur.- La fonction 01h de l'interruption 33h permet d'afficher le pointeur de la souris. La fonction 02h permet de le rendre transparent, et donc non visible.

Exemple.- Écrivons un programme permettant d'initialiser la souris, de faire apparaître le pointeur, de le faire disparaître lorsqu'on appuie sur une touche et de quitter le programme lorsqu'on appuie une deuxième fois sur une touche :

```
TITLE initsou.asm
; initialise la souris
      .model small
      .stack 256
      .data
message db 'souris non presente', '$'
      .code
debut:
      mov ax, @data
      mov ds, ax

      mov ax, 0            ; initialise
      int 33h              ; la souris
      cmp ax, 0            ; souris presente ?
      je erreur
      mov ax, 1            ; affiche
      int 33h              ; le pointeur
      mov ah, 8            ; attend un
      int 21h              ; un caractere
      mov ax, 2            ; fait disparaitre
```

```
      int 33h               ; le pointeur
      mov ah, 8             ; attend
      int 21h               ; un caractere
      jmp fin

erreur:
      mov dx, offset message
      mov ah, 9h
      int 21h               ; affiche absence de souris

fin:
      mov ax, 4c00h
      int 21h
      end debut
```

Remarque.- Si vous faites exécuter ce programme dans une fenêtre DOS sous Windows 95/98, il faut passer en plein écran. En effet Windows n'utilise pas les interruptions DOS pour gérer la souris et donc ceux-ci n'ont aucun effet sur une fenêtre de Windows (l'émulation n'est pas complète).

2.2.3 Fonction 03h d'obtention de la position du pointeur

Une fois la souris initialisée, le pointeur de souris peut être déplacé grâce à la souris ou positionné grâce au programme.

Statut des boutons.- Pour utiliser la souris, il faut savoir dans quelle zone se trouve le pointeur et si l'on a appuyé sur l'un des boutons lorsqu'on est dans cette zone.

Syntaxe.- Pour obtenir la position du pointeur, on utilise la fonction `03h` de l'interruption `33h`. Elle renvoie l'abscisse (coordonnée horizontale) dans le registre `CX` et l'ordonnée (coordonnée verticale) dans le registre `DX`. Comme d'habitude l'origine du repère est le coin supérieur gauche de l'écran, les abscisses sont comptées de gauche à droite en partant de zéro et les ordonnées de haut en bas en partant de zéro.

Cette fonction place de plus dans le registre `BX` le statut des boutons de la façon suivante :

- bit 0 : statut du bouton gauche (0 levé, 1 pressé)
- bit 1 : statut du bouton droit (0 levé, 1 pressé)
- bit 2 : statut du bouton du milieu (0 levé, 1 pressé)
- bits 3 à 15 : réservés.

2.2.4 Fonction 04h de déplacement du pointeur

Syntaxe.- On peut vouloir faire déplacer le pointeur de la souris par le programme (et non en manipulant celle-ci). On utilise pour cela la fonction 04h de l'interruption 33h, après avoir placé l'abscisse dans le registre CX et l'ordonnée dans le registre DX.

Exemple.- Écrivons un programme permettant d'initialiser la souris, de la déplacer puis, lorsqu'on presse une touche, de déplacer le pointeur de la souris de 10 unités, à la fois pour l'abscisse et pour l'ordonnée.

```
TITLE movesou.asm
; deplace le pointeur de la souris
      .model small
      .stack 256
      .data
message db 'souris non presente', '$'
      .code
debut:
      mov ax, @data
      mov ds, ax

      mov ax, 0             ; initialise
      int 33h               ; la souris
      cmp ax, 0             ; souris presente ?
      je erreur
      mov ax, 1             ; affiche
      int 33h               ; le pointeur
      mov ah, 8             ; attend un
      int 21h               ; un caractere
      mov ax, 3             ; coordonnees du
      int 33h               ; pointeur
      add cx, 50
      add dx, 50
      mov ax, 4             ; deplace
      int 33h               ; le pointeur
      mov ah, 8             ; attend
      int 21h               ; un caractere
      mov ax, 2
      int 33h
      jmp fin

erreur:
      mov dx, offset message
      mov ah, 9h
      int 21h               ; affiche absence de souris

fin:
      mov ax, 4c00h
      int 21h
      end debut
```

2.3 Fonction 05h d'impression

Introduction.- De nos jours nous sommes habitués aux imprimantes laser ou couleur à jet d'encre à haute résolution, et donc graphiques. Au début des micro-ordinateurs, une imprimante permettant d'imprimer du texte caractère par caractère était déjà beaucoup plus efficace qu'une machine à écrire : on pouvait commencer par visualiser le texte et le corriger si nécessaire. Ceci est le premier intérêt pour voir comment on faisait, même si cela est émulé sur les imprimantes modernes ; mais n'est-ce pas tout l'intérêt des interruptions. De toute façon, et c'est le deuxième intérêt de notre étude, les instructions sont toujours de nos jours transmises à l'imprimante de la même manière, caractère par caractère ; la seule différence est qu'on n'envoie plus le texte proprement dit de façon brute mais un programme source (dans un **langage de description de page** tels que `Postcript` ou `PCL`).

Description de la fonction.- La fonction `05h` de l'interruption MS-DOS `21h` permet d'envoyer un caractère à l'imprimante, le registre `DL` contenant le caractère à imprimer (plus précisément son code ASCII). MS-DOS attend que l'imprimante soit prête à accepter le caractère (c'est donc une fonction bloquante) ; on peut interrompre en appuyant sur `CTRL-Break`. La sortie par défaut concerne le port pour LPT1.

Beaucoup d'imprimantes conservent les caractères dans un tampon interne jusqu'à ce qu'il soit plein ou qu'une fin de ligne (`0Dh`) ou une fin de page (`0Ch`) soit rencontrée.

Application.- Écrivons un programme imprimant 'ab' sur une page.

```
TITLE print.asm
; imprime 'ab'
      .model small
      .stack 256
      .code
debut:
      mov ah, 5h
      mov dl, 'a'
      int 21h     ; impression de 'a'
      mov dl, 'b'
      int 21h     ; impression de 'b'
      mov dl, 0Ch
      int 21h     ; fin de page
      mov ax, 4c00h
      int 21h
      end debut
```

2.4 Le système

2.4.1 Fonction 2Ah d'obtention de la date

Description.- La fonction `2Ah` de l'interruption `21h` permet d'obtenir la date. Après appel de cette fonction, les informations sont placées dans les registres suivants :

`AL` : jour de la semaine, avec 0 = dimanche,

`CX` : année, avec `07D0h` = 2000,

`DH` : mois, de `01h` à `0Ch`,

`DL` : quantième du mois, de `01h` à `1Fh`.

Exemple.- Le programme suivant, utilisant `debug`, nous permet de déterminer la date :

```
C:\>debug
-a
17A4:0100 mov ah,2A
17A4:0102 int 21
17A4:0104 nop
17A4:0105
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=17A4  IP=0100   NV UP EI PL NZ NA PO NC
17A4:0100 B42A          MOV     AH,2A
-t

AX=2A00  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=17A4  IP=0102   NV UP EI PL NZ NA PO NC
17A4:0102 CD21          INT     21
-p

AX=2A04  BX=0000  CX=07D0  DX=090E  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=17A4  IP=0104   NV UP EI PL NZ NA PO NC
17A4:0104 90            NOP
-
```

ce qui donne le jeudi 14 septembre 2000.

2.4.2 Récupération de l'heure

2.5 Liste des interruptions du DOS

Nous avons vu les interruptions du DOS suivant l'ordre logique de leur fonctions. Donnons-en une liste par numéro d'interruption puis numéro de fonction et, éventuellement de sous-fonctions.

2.5.1 Numéros d'interruption

IBM a réservé les interruptions `20h` à `3Fh` pour système d'exploitation MS-DOS.

`INT 20h` : termine le programme. On préfère cependant utiliser à la place en général de nos jours la fonction `00h`, ou mieux la fonction `4Ch`, de l'interruption `21h`.

`INT 21h` : il s'agit de l'interruption principale du système d'exploitation MS-DOS, avec de très nombreuses fonctions, que nous détaillerons ci-dessous.

`INT 22h` : lorsqu'on programme se termine il passe la main à cette interruption. Cette interruption ne doit pas être utilisée dans un programme.

`INT 23h` : appel d'une routine de MS-DOS lorsqu'on appuie sur CTRL-Break ou CTRL-C. Cette interruption ne doit pas être utilisée dans un programme.

`INT 24h` : appel d'une routine de MS-DOS lors d'une erreur critique (concernant le disque ou l'imprimante). Cette interruption ne doit pas être utilisée dans un programme.

`INT 25h` : lecture absolue sur un disque. On utilise plutôt maintenant la fonction `440Dh`, code mineur `61h`, de l'interruption `INT 21h`.

`INT 26h` : écriture absolue sur un disque. On utilise plutôt maintenant la fonction `440Dh`, code mineur `41h`, de l'interruption `INT 21h`.

`INT 27h` : termine un programme `.com` devant rester en mémoire vive. On utilise plutôt de nos jours la fonction `31h` de l'interruption `INT 21h`.

`INT 28h` : réservé.

`INT 29h` : réservé.

`INT 2Ah` : réservé.

`INT 2Bh` : réservé.

`INT 2Ch` : réservé.

`INT 2Dh` : réservé.

`INT 2Eh` : réservé.

`INT 2Fh` : concerne la communication entre programmes.

`INT 30h` : réservé.

`INT 31h` : réservé.

`INT 32h` : réservé.

`INT 33h` : concerne la souris, avec de nombreuses fonctions.

2.5.2 Les fonctions de l'interruption `21h`

Avant d'appeler l'interruption `21h`, le numéro de fonction doit être placé dans le registre `AH`.

`00h` : termine un programme. Fait la même chose que l'interruption `INT 20h`. Ensuite remplacée par la fonction `4Ch`.

`01h` : saisie d'un caractère au clavier avec écho.

`02h` : affichage d'un caractère à l'écran.

`03h` : lecture d'un caractère à partir d'un port série, placé dans le registre `AL`. Il vaut mieux lui préférer l'interruption `INT 14h` du BIOS.

`04h` : envoi d'un caractère, situé dans le registre `DL`, vers un port série. Il vaut mieux lui préférer l'interruption `INT 14h` du BIOS.

`05h` : impression d'un caractère.

`06h` : accès direct au clavier et à l'écran, sans grand intérêt.

`07h` : accès direct au clavier sans écho, sans grand intérêt.

`08h` : saisie d'un caractère au clavier sans écho.

`09h` : affichage d'une chaîne de caractères.

`0Ah` : saisie tamponnée d'un caractère.

`0Bh` : vérification du statut du clavier.

`0Ch` : vidage du tampon du clavier et attente d'une entrée nouvelle.

`0Dh` : réinitialisation du lecteur de disquette.

`0Eh` : sélection du lecteur de disque par défaut.

`0Fh` : ouverture d'un fichier spécifié par un FCB (*File Control Block*).

`10h` : fermeture d'un fichier spécifié par un FCB.

`11h` : recherche du premier appariement sur un disque. Obsolète, remplacée par la fonction `4Eh`.

`12h` : recherche de l'appariement suivant sur un disque. Obsolète, remplacée par la fonction `4Fh`.

`13h` : effacement d'un fichier spécifié par un FCB. Obsolète, remplacée par la fonction `41h`.

`14h` : lecture séquentielle d'un fichier spécifié par un FCB.

`15h` : écriture séquentielle sur un fichier spécifié par un FCB.

`16h` : création d'un fichier spécifié par un FCB.

`17h` : renommage d'un fichiers pécifié par un FCB. Obsolète, remplacée par la fonction `56h`.

`18h` : réservé.

`19h` : détermination du lecteur de disque par défaut.

`1Ah` :

2.6 Bibliographie

[IBM-82] **Disk Operating System, version 1.10**, IBM 6172220, Personal Computer, Computer Language Series, second edition, May 1982, xii + 8 + 30 + 67 + 34 + 28 + 50 + 28 + 13 + 19 + 20 + 11 + 3 + 6 + 13 p.

[L'appendice D donne la description officielle des interruptions de DOS 1.0.]

[IBM-84] **Disk Operating System, vesrion 2.10**, IBM, Personal Computer, Computer Language Series, second edition, January1984, xvi + 27 + 156 + 23 + 10 + 12 + 44 + 28 + 61 + 87 + 3 + 3 + 27 + 14 + 14 + 14 + 14 p. (571 p.).

[L'appendice donnant la description des interruptions du DOS n'existe plus. On est renvoyé au *Technical Manual Reference*.]

[Gil-97] Van Gilluwe, Frank, **The undocumented PC : A Programmer's Guide to I/O, CPUs, and Fixed Memory Areas**, second edition, Addison-Wesley, 1997, xii + 1124 p. + diskette. Tr. fr.

[BK-94] Brown, Ralf & Kyle, Jim, **PC Interrupts : A programmer's reference to BIOS, DOS, and third-party calls**, second edition, Addison-Wesley, 1994, vi + 1210 p.