JPC

JUIN 1993
NUMERO 87

Le numéro 40 F

A PROPOS DU CLUB

Le Bureau	Editorial	*1*
J. Belin	L'OmniBook 300	*2*
R. Pulluard	Libres Propos (Acte II)	*3*
	Courrier du coeur	*4*

HP28

E. Dougados	Fractions de PI	*6*
E. Dougados	Evaluation du polynôme de Newton	*6*
J.F. Garnier	Programmation System RPL sur HP28	*7*
J.F. Garnier	Points d'entrées compatibles HP48 (Acte I)	*8*

HP48

J. Belin/P. de Sacy	La HP48GX	*15*
P. Silvestre de Sacy	*Les Secrets de la HP48* tome II	*17*
J. F. Garnier	Extension des menus (Acte II)	*18*
J. F. Garnier	Programmation System RPL	*18*
G. Toublanc	Trucs et Astuces	*19*

MS-DOS

J. Belin	Visualisation d'encombrement du disque	*22*
	Le coin des codes	*31*

EDITORIAL

En moins d'un mois, Hewlett-Packard a annoncé trois nouvelles machines portables : le HP100LX (dont nous avons parlé dans le précédent *JPC*), la HP48G/GX et l'OmniBook 300 que nous vous présentons dans ce numéro. Ces trois petites merveilles prenant une avance certaine par rapport à la concurence, les équipes de développeurs de Corvallis vont-elles s'endormir sur leurs lauriers ? Ou alors (puisqu'elles ont terminé leur travail pratiquement en même temps), vont-elles s'allier sur un projet commun ? On peut toujours rêver...

Parlons de ce numéro maintenant. Suite à l'article *Libres Propos* de Christophe Vaillant, paru le mois dernier, Robert Pulluard (le très estimable éditeur de l'excellente revue 48SXtant) a jugé bon de continuer le débat en apportant ses propres réflexions. Nous ne pouvons que l'en féliciter. Rappelez vous que la rubrique *A Propos du Club* est aussi prévue pour vous permettre de vous exprimer. Maintenant que ce débat est ouvert, nous ne pouvons qu'espérer qu'il va continuer. Pour notre part, nous essayerons de ne pas intervenir dans la discussion, mais tenterons bien sûr de tenir compte de vos observations.

Ce mois-ci, la rubrique MS-DOS remplace celle du HP95. Quelle est la différence me direz-vous ? Aucune, si ce n'est une petite astuce visant à encourager les utilisateurs de HP48 à la lire ! En effet, le programme qui y est présenté a bien été écrit pour les utilisateurs de HP95 (avec une taille tenant compte de l'exiguïté de son disque interne), mais il devrait en fait être plus utilisé par les utilsateurs de HP48, qui manipulent des petits fichiers MS-DOS et doivent souvent les copier sur disquette !

Avant de terminer cet éditorial, laissez nous corriger une petite erreur parue dans le dernier numéro. En effet, nous avons mal orthographié le nom du responsable du Support Technique HP. Que Bernard Challiol, puisque c'est de lui qu'il s'agit, nous en excuse...

UN PC A LA PLACE D'UN LIVRE ?

Baptisée OmniBook 300, ce nouvel ordinateur inaugure, pour HP en tous cas, le marché des *Subnotebooks*, c'est à dire des portables dont le poids est compris entre un et deux kilos. A priori, cette machine n'entrant pas dans les activités du club, nous pourrions ne pas en parler. Cependant, certains adhérents semblant être intéressés, j'en ferais un simple résumé. Notez que n'ayant vu la machine qu'en photo, je base cet article sur des documents que m'a fourni Hewlett-Packard, complétés par des informations parues dans la presse étrangère.

Bien qu'étant largement plus grand que le HP95, le HP300 reprend exactement son design. A un tel point qu'une photo de la machine fermée nécessite une observation attentive pour savoir de quel ordinateur il s'agit !

Une fois ouvert, les différences apparaissent au premier coup d'oeil. Tout d'abord le clavier est du même type que pour les Notebooks actuellement commercialisés, sans pavé numérique séparé. D'autre part, l'écran occupe toute la hauteur du volet rabattable.

Alors que le HP95 ne contient qu'un port PCMCIA, le HP300 en contient quatre. Utilisant la version 2.0, ils peuvent accueillir des cartes de type II et III. Un des ports est occupé à l'origine par la ROM contenant les applications internes. Ceci permettra de faciliter les mises à jour. Les autres ports peuvent être utilisés pour les mémoires de masse, tels qu'un disque dur de 40 Mo (au format type III, donc occupant 2 ports).

Les applications incluses en Rom sont : MS-DOS 5.0, Windows 3.1, le traitement de texte Word pour Windows 2.0c, le tableur Excel 4.0, l'utilitaire de transfert Laplink Remote Access permettant de faciliter les échanges de fichiers entre PC, ainsi que certaines applications connues des utilisateurs du HP95 : répertoire téléphonique, agenda, calculatrice financière. Toutes ces applications sont du type XIP, c'est à dire qu'elles sont exécutées directement à partir de la ROM, sans être chargées en RAM. Ceci permet de conserver le maximum de la mémoire vive pour les données. Comme pour le HP95, une série de touches spéciales permettent d'accéder à toutes ces applications.

Mais la grande innovation de la machine concerne la souris. Au repos, elle est placée dans un logement sur le côté droit de l'ordinateur. Pour l'utiliser, il suffit de l'éjecter en appuyant sur un bouton prévu à cet effet. Elle reste alors solidaire de l'ordinateur par une courte tige rigide transmettant ses mouvements directement à l'ordinateur (un peu à la manière d'un pantographe). Il est donc possible de s'en servir debout (à condition de faire attention), sans qu'elle soit en contact direct avec une table ! La tige contient bien sûr les cablages nécassaires à la détection des appuis sur les deux boutons dont elle est équipée.

Autres caractéristiques techniques :

- Microprocesseur : 386 SXLV à 20 Mhz.
- RAM : 2 Mo extensible à 8 Mo.
- Entres/Sorties : Port série 9 broches, port parallèle 25 broches, port Infrarouge à 115 kb.
- Port spécifique pour une carte Fax/Modem, livrée en option.
- Affichage : écran 9 pouces, VGA 640×480, 16 niveaux de gris.
- Alimentation : Accus NiMH de 4.8V ou 4 piles 1.5V. Autonomie : jusqu'à 10 heures, suivant la mémoire de masse utilisée.
- Dimensions (fermé) : 26.2×16.3×3.6 cm.
- Poids : 1.27 Kg.

L'OmniBook est livré en trois versions : Sans mémoire de masse, avec une carte Flash PCMCIA de 10 Mo, ou avec un disque dur PCMCIA de 42 Mo. Les prix avoisinent les 10000 F.HT. pour la version nue, et 15000 F.HT. pour la version avec disque dur.

Jacques Belin (123)

LIBRES PROPOS
(ROUND II)

L'article *Libre Propos* de Christophe Vaillant dans JPC No.86 m'a vivement intéressé (ainsi que les réponses de Jacques Belin), et je crois que d'autres avis ou réponses sur les questions posées par Christophe peuvent alimenter une discussion particulièrement enrichissante au sein du club dont je pense que c'est l'un des rôles fondamentaux, autant que de transmettre et disséminer l'information brute à ses membres. Je vous propose donc mes propres réflexions sur certaines (pas toutes) des questions posées ou soulevées par Christophe Vaillant, en les reprenant dans l'ordre de son article.

- En ce qui concerne HEX, cette association de clubs européenne ne fonctionne actuellement qu'au niveau des échanges de journaux ou bulletins entre clubs, et encore uniquement dans la mesure où les contacts entre ces clubs sont pris au niveau individuel. Autant dire que jusqu'ici HEX ne sert pas à grand chose. J'en suis d'autant plus désolé que j'en étais l'initiateur et que mon projet prévoyait une collaboration plus étroite entre les clubs tout en respectant l'indépendance et la spécificité de chacun. Il ne m'a hélas pas été possible de mener ce projet à son terme et d'autres ont repris le flambeau... dont la flamme mériterait bien d'être ranimée. Ceci dit, il est évidemment possible de reprendre et surtout de traduire des articles parus dans les bulletins d'autres clubs participant à HEX. J'insiste sur l'aspect "traduction": d'une part parce que je ne considère pas très loyal de reprendre un article (en anglais par exemple) tel quel, aussi bien à l'égard du club dont il est originaire, qu'envers les membres de PPC-Paris qui, s'ils désirent lire ces articles dans leur langue d'origine, pourraient tout aussi bien s'adresser aux clubs en question. D'autre part parce qu'il faut bien être conscient, comme le souligne d'ailleurs Jacques Belin, que traduire un article n'est pas plus facile que d'en écrire un soi-même (le rôle des traducteurs est trop souvent sous-estimé et leur travail négligé). Quoiqu'il en soit, et je réponds ici à Jacques, bien que n'étant pas moi-même un professionnel en la matière (mais j'ai une bonne expérience), je suis tout à fait disposé à traduire occasionnellement des articles pour JPC, notamment à partir du néerlandais ou de l'allemand (à partir de l'anglais aussi, certes, mais je pense que cette langue pose beaucoup moins de problème pour trouver des traducteurs). Je terminerai sur ce point en précisant qu'il serait regrettable que JPC publie des articles en anglais ou toute autre langue que le français (quoique... un article en basque ou en breton, pourquoi pas?). Non pas que je sois anglophobe (ceux qui me connaissent pourront témoigner du contraire!) mais parce que je considère que chaque club, et pas seulement JPC ou tout autre club français, doit conserver son identité propre et que la culture qui le berce, notamment la langue, doit être respectée : c'est l'une des idées maîtresses de HEX.

- Les horaires de réunions du club ne sont sans doute pas idéaux, mais je rejoins ici entièrement Jacques Belin. Il n'est certes pas toujours facile à un membre habitant la province d'y assister, mais pour ceux qui le désirent vraiment cela devrait tout de même être possible ne serait-ce qu'occasionnellement. A vrai dire, je voudrais simplement apporter de l'eau au moulin de Jacques en précisant qu'il est possible d'assister aux réunions même lorsqu'on habite l'étranger, qui plus est un pays non limitrophe de la France : les Pays-Bas dans mon cas précis. Certes, je n'ai participé qu'à deux réunions depuis le début de l'année, mais c'est tout de même mieux que rien : à titre indicatif, il me faut 5 heures à 5 heures 30 en train pour me rendre à Paris, c'est-à-dire sensiblement la même chose que pour faire Toulon-Paris par TGV. Et il y a un petit hôtel sympa et confortable à deux pas du centre Jean Verdier : sans faire de pub pour cet hôtel, Jacques pourrait peut-être tout de même utiliser cet atout pour inciter davantage de membres qui habitent loin de Paris à venir assister aux réunions de temps à autre...

- L'avenir du club : si je partage le sentiment de Christophe sur le fait que la PASSION est nécessaire, je ne suis par contre pas d'accord avec son argumentation ni avec certains symptomes qu'il décrit. Je voudrais donc, sans préjuger qui, de lui ou de moi, a raison (sans doute un peu tous les deux), apporter mon grain de sel à cette discussion, en espérant que d'autres membres viendront y ajouter leur voix.

Les gens préfèrent consommer des programmes plutôt qu'en faire, c'est une constatation : entièrement d'accord, mais j'ai supprimé dans ma citation le *aujourd'hui* que Christophe avait placé en tête de sa phrase. Parce qu'à mon avis il n'est pas pertinent. C'était tout aussi vrai autrefois, à l'époque "héroïque", que cela l'est aujourd'hui. La différence c'est qu'autrefois (il y a dix ans, une éternité!) la micro-informatique en était à ses balbutiements : il fallait tout inventer soi-même ou presque, d'où un vaste élan d'enthousiasme et de créativité. Le "vaste" est peut-être trop fort car, si on regarde de plus près, que doit-on constater d'un point de vue "historique" ? Les clubs de cette époque étaient essentiellement animés par un tout petit groupe de PASSIONNÉS. Quand ces passionnés, pour l'une ou l'autre raison, cessèrent leurs activités, les clubs se sont effondrés:

Richard Nelson et PPC; Jean-Daniel Dodin et PPC-Toulouse... Dans quelques rares cas un autre passionné a pu prendre la relève (Wlodek Mier-Jedrzejowicz qui prit la succession de David Burch pour HPCC en Grande Bretagne). Ou alors, on retrouve toujours les mêmes passionnés, dix ans après: Gerlof Donga à la tête de Prompt aux Pays-Bas... (je ne donne ici qu'un ou deux exemples dans chaque cas, qu'on ne me reproche pas des "oublis"!) Bien sûr, il y avait, comme il y a encore aujourd'hui, un petit nuage de membres très actifs dans ces clubs. Mais la grande majorité des membres, autrefois comme maintenant, se contente de consommer.

La HP-41 avait l'avantage d'être accessible par son prix : oh que non! Elle était plus chère en francs d'époque que ne l'est la HP48SX aujourd'hui, et en francs constants cela représente plus du double de ce que coûte la 48SX! Je n'ai pas les chiffres exacts en tête (c'est vieux, tout ça!), mais la HP-41C coûtait dans les 3000 FF, et il fallait débourser plus de 4000 FF pour la 41-CX. Même à ses débuts la HP48SX n'a jamais été aussi chère!

Qui connaît les 2100 fonctions de la HP48 ? : trois personnes, Bill Wickes, Joseph Horn et Jean-Michel Ferrard... Plus sérieusement, il n'est nullement besoin de connaître toutes les fonctions de la HP48 pour pouvoir l'utiliser intelligemment et efficacement. *Qui ne se sent pas ridicule devant les programmes faits par des professionnels...* : là je ne suis plus du tout d'accord avec Christophe. A-t-il acheté ou utilisé ces fameuses cartes "professionnelles" dont il parle ? Dans bien des cas ce sont les pros qui devraient se sentir ridicules de nous proposer un "produit" souvent affligeant ! Certaines cartes qui nous viennent du nord sont franchement nulles, quant aux cartes de Sparcom, les "professionnels" par excellence, elles comportent bien des défauts et les "amateurs" peuvent et font nettement mieux. Certes, il y a quelques cartes pour HP48 qui tiennent la route, mais il y a des logiciels amateur qui surclassent ce que font les pros, y compris HP. On trouve cela en particulier (mais pas uniquement) sur les fameux "Goodies disks" de Joe Horn, disponibles pour une bouchée de pain. Un seul exemple parmi tant d'autres: les excellents utilitaires de Detlef Mueller sur le No.8 (que Christophe analyse par ailleurs dans un autre article). Non, et là je suis catégorique, il ne faut avoir aucun complexe! En tant qu'amateur il est encore tout à fait possible de contribuer fort honorablement à la construction de l'édifice: il suffit d'avoir des idées, de la persévérance et la détermination. Un détail: des trois noms cités "au hasard" au début de ce paragraphe, seul Bill Wickes est un "professionnel"; Joe Horn tout autant que Jean-Michel Ferrard sont enseignants et "amateurs". Mais passionnés !

Une dernière remarque, concernant le parallélisme avec le monde des PC: je ne crois pas qu'une telle comparaison soit appropriée. Le "monde des PC" est complètement verrouillé par les professionnels depuis qu'IBM s'en est mêlé. Et à la question "qui aujourd'hui fait un programme seul?" je répondrai (fort égocentriquement) : moi. Car les monstres pour PC que nous proposent les pros, non seulement sont aujourd'hui d'un niveau de prix qui frôle l'absurdité, mais ils atteignent des tailles démentielles qui ne se justifient souvent pas, et de plus ils ne font pratiquement jamais ce que l'on veut: il faut trop souvent se plier aux caprices d'un employé programmeur pressé de coder quelques lignes avant de rentrer chez lui, ou aux raccommodages ad hoc entre modules incompatibles conçus par des "équipes entières" indépendantes et à la limite de la compétence. Il ne s'agit évidemment pas de réécrire Wordperfect ou un compilateur C++, mais il y a de nombreuses applications (surtout personnelles) pour lesquelles un programme idoine modeste peut être conçu et écrit en quelques soirées, fournissant la solution optimale.

Pour conclure j'ajouterai simplement que la micro-informatique style PC est devenue une affaire de gros sous qui a totalement écrasé l'amateurisme (surtout au niveau de la distribution des programmes; les revues spécialisées en sont en grande partie responsables: devenues toutes, ou presque, plus nulles les unes que les autres, ce ne sont plus que des catalogues publicitaires s'adressant surtout aux consommateurs professionnels). Qu'à cela ne tienne: il nous reste les calculatrices de poche, et surtout les HP, permettant encore à tous les véritables passionnés de laisser fleurir leur créativité insubmersible. La masse des consommateurs de programmes? elle est utile et elle a son rôle à jouer. Il ne faut pas la dénigrer, ni la mépriser et surtout pas la rejeter. S'il n'y a que deux passionnés, cela suffit à faire un club.

Robert Pulluard (561)

COURRIER DU COEUR

Romuald CLOMENIL
49, rue Emile Zola
92100 BOULOGNE
Tel : (1) 46 21 49 64

Vend : HP28S : 500 F

HP28

E. Dougados	Fractions de PI	*6*
E. Dougados	Evaluation du polynôme de Newton	*6*
J.F. Garnier	Programmation System RPL sur HP28	*7*
J.F. Garnier	Points d'entrées compatibles HP48 (Acte I)	*8*

PI ET FRACTIONS

Voici une petite astuce créant la fonction →Qπ bien connue des utilisateurs de HP48.

On prend le programme →FR?, du livre *Voyage au centre de la HP28*, on le modifie et on le renomme ?→Qπ. Pensez à garder →FR? dans la mémoire de votre HP28 pour une utilisation autre qu'avec π.

Voici la marche à suivre :

1- Entrez donc le sous-programme suivant :

```
«
  → nπ
  «
    nπ π / →NUM  ?→FR π *
  »
»
'→Qπ' STO
```

2- Avec la copie de ?→FR mis dans la pile, faire EDIT et remplacer tous les →FRAC par →Qπ. Nommer le nouveau programme '?→Qπ' et le tour est joué.

Exemple d'utilisation :

Faire : π 1.2 * puis →NUM. Puis, lancer ?→Qπ. Ceci donne : '6/5*π' qui est sensiblement équivalent à 3.7699.

Edmond Dougados

BINOMES ET NEWTON

Ce programme permet le développement du binôme de Newton.

Mode d'emploi :

- Entrer les valeurs *a*, *b*, et *n* dans la pile, puis lancer le programme DEVL.

Exemple :

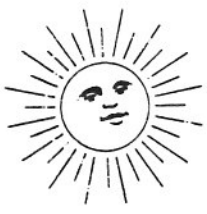
```
1:    'A'
2:    'B'
3:     2
```

suivi de DEVL donnera "$a^2+2ab+b^2$" au bout de 3 a 4 secondes. Mais on peut faire la même chose en prenant 'A' 'B' 100... Si vous avez SPEED, alors là, c'est le fin du fin ! Imaginez ce travail à la main, sans la moindre erreur !

Listing :

```
«
  1 FIX → a b n
  «
    'POLY.' 0 'bc' STO STO -1 n
    FOR t
      t EVAL 'T' STO
      IF -1 t ==
      THEN
        0 'T' STO
      END
      bc 'n-T' EVAL DUP 'bc' STO
      IF T 0 >
      THEN
        * DUP 'bc' STO
      END
      T 1 + 'ABX' STO
      IF t -1 ==
      THEN
        n /
      ELSE
        ABX ! /
      END
      IF t -1 ==
      THEN
        1 'r' STO
      ELSE
        t EVAL 2 + 'r' STO
      END
      'a^(n-r+1)' EVAL 'b^(r+1-2)' EVAL * *
      POLY. SWAP + 'POLY.' STO
    NEXT
    POLY. DUP 'POLY.' PURGE
    { T bc ABX r } PURGE
    2 ROLL DROP 2 ROLL DROP SWAP DROP
    1500 .5 BEEP
  »
»
```

Edmond Dougados



SYSTEM RPL SUR HP28S

Petit rappel historique. En 1987 apparait la HP28C, première de la nouvelle génération des calculateurs scientifique HP basée sur ce qu'on nomme communément RPL, soit Reverse Polish Lisp. La HP28C ne comportait que 2 Ko de mémoire, jugée rapidement comme dramatiquement insuffisante. Un an plus tard (1988), HP corrige le tir en annonçant la HP28S dotée de 32 Ko. Mais le plus gros défaut de la 28 reste l'absence de possibilité de sauvegarde externe, contrairement à ses prédécesseurs 41 et 71. Néanmoins, on ne tarde pas à voir apparaitre des programmes en assembleur pour cette machine, aidé en cela par la connaissance du processeur Saturn issu du HP71. Armé de la commande magique SYSEVAL, apparaissent les fonctions PEEK et dérivés, et les premières informations sur la structure interne de la machine.

En 1990, arrive la HP48SX. Bien qu'il s'agisse sans conteste d'une nouvelle machine, il n'en reste pas moins vrai qu'elle est la descendante directe de la 28, reprenant un grand nombre de fonctionnalités de son aînée. Les connaissances accumulées sur la 28 sont rapidement adaptées à la 48. Associées à la facilité de diffusion par disques PC, de nombreuses réalisations voient le jour en assembleur ou en RPL interne.

Tout change courant 1991 quand HP décide de mettre à la disposition des développeurs un ensemble de logiciels (sur PC) permettant de travailler enfin sérieusement en System RPL ! Même si la documentation des points d'entrées reste notoirement insuffisante (comparée aux I.D.S du 71 par exemple), ces outils ont marqués la fin des tentatives précédentes de pénible décryptage des ROM et le début de l'ère de l'exploration organisée et réfléchie des multiples objets internes.

Fin du rappel historique. Possédant une HP28S, j'ai comme beaucoup cherché très tôt à en explorer le contenu. Rebuté par la difficulté de l'exploration in-situ, j'ai réalisé rapidement (hum!) le transfert de la ROM sur PC, et commencé à me construire des outils d'exploration. Mais je n'ai pu aller très loin. Que faire devant une foule d'objets plus obscurs les uns que les autres, sauf quelques rares DUP, DROP et autres ROT vite identifiés ? Après l'achat de la 48 dès sa sortie (version A s'il vous plait), j'ai naturellement porté mes outils pour cette nouvelle machine, d'autant plus que le transfert de la ROM de la 48 ne pose aucun problème (merci HP), mais sans véritablement poursuivre plus avant. Je n'ai pu disposer des outils HP qu'à l'automne 1992.

A ce moment, l'interêt de l'étude de la ROM devenait évident : on peut de cette manière palier au manque de documentation des points d'entrée, il suffit d'aller jeter un coup d'oeil. Mais rapidement l'idée suivante m'est venu : la HP48 a repris un grand nombre d'objets system RPL de la 28. On retrouve effectivement de nombreuses routines identiques lorsqu'on étudie leurs ROM respectives. Pourquoi ne pas faire bénéficier la 28 des connaissances actuelles sur la 48 ?

Grâce à la comparaison des 2 ROMs, j'ai pu déterminer un certain nombre de points d'entrées de la 28 que je suis heureux de partager avec vous aujourd'hui. La liste que je vous propose comporte environ 500 adresses, dont près de la moitié correspondent il est vrai à des objets utilisateurs déjà connus. J'ai donné à ces points d'entrées le même nom que celui correspondant sur la 48, dans la mesure où les objets pointés sont identiques. Quelles sont les applications ? J'en vois deux, hormis le plaisir de la connaissance pure :

- se créér une boite à outils à base de SYSEVAL, étendant ainsi les possibilités de cette machine.
- donner accès à la programmation en system RPL, en profitant de l'acquis de la HP48. Les documents HP, associés à la liste présentée ici le permettent.

Pour illustrer la première possibilité, et pour terminer mon propos, je vous soumets un certain nombre de fonctions d'usage général qui devraient être présentes dans toute 28 bien née. Une dernière précision cependant, ces points d'entrées ne concernent que la 28S version 2BB (désolé pour les 28C). A ma connaissance, une seule version de 28S a été commercialisée, mais il serait prudent de vérifier par la commande ~~#10h~~ #0Ah SYSEVAL.

Notes sur les programmes :

- Les '@' indiquent des commentaires à ne pas saisir.
- J'ai indiqué pour chaque SYSEVAL le mnémonique correspondant.

Rappel: il n'y a généralement pas de vérification de présence ou de type d'arguments en system RPL, donc s'en assurer avant usage.

SYSRCL (#nnnnnh → objet)

Rappelle un objet interne sur la pile, à partir de son adresse (entier binaire utilisateur).

```
«
  # 71BDh SYSEVAL      @ TWO
  # 4152h SYSEVAL      @ NTHELCOMP
  DROP
»
```

FRE (--→n)

Retourne la taille de la mémoire restante en octets. Ne provoque de garbage collector.

```
«
  # 48C1h SYSEVAL       @ MEM
  # C610h SYSEVAL       @ UNCOERCE
  2 /
»
```

CRNAME ("nom" → 'nom')

Création d'un nom à partir d'une chaine. Permet de créer des noms interdits ('(*)', 'DATE+', ...).

```
«
  # 44EDh SYSEVAL       @ $>ID
»
```

B→SB (#nnnnh → <nnnnh>)

Conversion d'un binaire utilisateur en binaire système. De nombreux objets RPL attendent en arguments des binaires systèmes. Note : la HP28 affiche les binaires système sous forme de System Objet.

```
«
  # 43ADh SYSEVAL       @ HXS>#
»
```

SB→B (<nnnnh> → #nnnnh)

Conversion d'un binaire système en binaire utilisateur. De nombreux objets RPL fournissent en résultats des binaires systèmes.

```
«
  # C610h SYSEVAL       @ UNCOERCE
  B→R  R→B
»
```

COM→ (composite → obj1 obj2 ... objn n)

Décomposition d'un objet composite en ses éléments. Comme OBJ→, mais fonctionne aussi sur les programmes et les algébriques.

```
«
  # 3F41h SYSEVAL       @ INNERCOMP
  # C610h SYSEVAL       @ UNCOERCE
»
```

→PRG (obj1 obj2 ... objn n → program)

Création d'un objet programme à partir de ses éléments Associé à COM→, permet des manipulations sur les programmes : suppression des « », insertion de System Objets rappelés par SYSRCL, ...

```
«
  # C53Bh SYSEVAL       @ COERCE
  # 3EEBh SYSEVAL       @ ::N
»
```

CEVAL (liste → ?)

Evaluation d'un objet composite. Comme EVAL, mais s'applique aux listes de façon semblable aux programmes (comme sur la 48). Application : exécution de listes construites par programme.

```
«
  # C784h SYSEVAL       @ COMPEVAL
»
```

PARSE ("chaine" → objet 1

(si pas d'erreur)
→ "chaine" <nnnnnh> "chaine" 0
(si erreur))

Analyse d'une chaine et création de l'objet correspondant.

```
«
  # EAF7h SYSEVAL       @ palparse
  # C9F4h SYSEVAL       @ COERCEFLAG
»
```

Jean-François Garnier (242)

POINTS D'ENTREES HP28

Voici une liste des points d'entrées équivalents entre la HP28 et la HP48. Ils ne concernent que la HP28S version 2BB.

NDLR : Etant donné la longueur de cette liste (plus de 500 points d'entrées !) et afin d'éviter de donner un aspect "France Telecom" à JPC, nous avons décidé de la passer en deux fois. Vous en trouverez donc la fin dans le prochain numéro. D'autre part, elle peut être obtenue sous forme magnétique, triée par ordre alphabétique.

```
=DEPTH      EQU #0200E *  (primitive)
=2DUP       EQU #0206E *  (primitive)
=NDUP       EQU #0209B *  (primitive)
=DUP        EQU #0204A *  (primitive)
=SWAP       EQU #020E5 *  (primitive)
=DROP       EQU #02106 *  (primitive)
=2DROP      EQU #0211A *  (primitive)
=NDROP      EQU #02130 *  (primitive)
=ROT        EQU #02157 *  (primitive)
=OVER       EQU #02184 *  (primitive)
=PICK       EQU #021A4 *  (primitive)
```

```
=ROLL          EQU #021E7 *  (primitive)
=UNROLL        EQU #02231 *  (primitive)
=MAKEARRY      EQU #022AB *  (primitive)
=ARSIZE        EQU #023D4 *  (primitive)
=DIMLIMITS     EQU #0241B *  (primitive)
=MUL#          EQU #0281B *  (assembly)
=GETTEMP       EQU #02839 *  (assembly)
=PRLG          EQU #028FC =  (assembly)
=DOBINT        EQU #02911 =  (assembly)
=DOREAL        EQU #02933 =  (assembly)
=DOEREAL       EQU #02955 =  (assembly)
=DOCMP         EQU #02977 =  (assembly)
=DOECMP        EQU #0299D =  (assembly)
=DOCHAR        EQU #029BF =  (assembly)
=DOARRY        EQU #02A0A =  (assembly)
=DOCSTR        EQU #02A4E =  (assembly)
=DOHSTR        EQU #02A70 =  (assembly)
=DOLIST        EQU #02A96 =  (assembly)
=DORRP         EQU #02AB8 =  (assembly)
=DOSYMB        EQU #02ADA =  (assembly)
=DOCOL         EQU #02C67 =  (assembly)
=DOCODE        EQU #02C96 =  (assembly)
=DOIDNT        EQU #02D12 =  (assembly)
=DOLAM         EQU #02D37 =  (assembly)
=DOROMP        EQU #02D5C =  (assembly)
=SKIPOB        EQU #02E94 *  (assembly)
=SEMI          EQU #02F90 =  (assembly)
=ERRORSTO      EQU #036BD *  (primitive)
=ERRORCLR      EQU #036E2 *  (primitive)
=ERRSET        EQU #037E3 *  (primitive)
=ABORT         EQU #03829 *  (prog)
=ERRTRAP       EQU #0383D *  (primitive)
=ERRJMP        EQU #03856 *  (primitive)
=Errjmp        EQU #0396A *  (assembly)
=>T$           EQU #03D75 *  (primitive)
=::N           EQU #03EEB *  (prog)
={}N           EQU #03EFF *  (prog)
=INNERCOMP     EQU #03F41 *  (primitive)
=DOGARBAGE     EQU #03FE8 *  (assembly)
=NULL$         EQU #0407F *  (string)
=LEN$          EQU #040C2 *  (primitive)
=LENCOMP       EQU #04108 *  (primitive)
=NTHELCOMP     EQU #04152 *  (primitive)
=SUB$          EQU #041CF *  (primitive)
=SUBHXS        EQU #042BE *  (primitive)
=SUBCOMP       EQU #042CA *  (primitive)
=HXS>#         EQU #043AD *  (primitive)
=CHR>#         EQU #04408 *  (primitive)
=#>CHR         EQU #04439 *  (primitive)

=CHANGETYPE    EQU #04486 *  (prog)

=$>ID          EQU #044ED *  (primitive)
=GARBAGE       EQU #048A4 *  (primitive)
=MEM           EQU #048C1 *  (primitive)

=GARBAGECOL    EQU #04A94 *  (assembly)
=PUSH#         EQU #04E31 *  (assembly)
=GPMEMERR      EQU #04EA4 *  (assembly)
=POP#          EQU #04F27 *  (assembly)
=TOTEMPOB      EQU #04F3D *  (primitive)
=MOVEUP        EQU #04F9F *  (assembly)
=MOVEDOWN      EQU #04FF2 *  (assembly)
=SAVPTR        EQU #05081 *  (assembly)
=GETPTR        EQU #050B8 *  (assembly)
=ROOM          EQU #050EC *  (assembly)
=CREATETEMP    EQU #053AE *  (assembly)

='             EQU #0541F *  (primitive)
='R            EQU #05473 *  (primitive)
=EVAL          EQU #054E8 *  (primitive)
=RDROP         EQU #05511 *  (primitive)
=COLA          EQU #0552B *  (primitive)
=?SKIP         EQU #05604 *  (primitive)
=SKIP          EQU #05627 *  (primitive)
=BEGIN         EQU #05673 *  (primitive)
=AGAIN         EQU #0567C *  (primitive)
=UNTIL         EQU #05699 *  (primitive)
=INDEX@        EQU #056F2 *  (primitive)
=LOOP          EQU #05805 *  (primitive)
=#1+_ONE_DO    EQU #058AC *  (primitive)
=ABND          EQU #05973 *  (primitive)

=BIND          EQU #059B5 *  (prog)

=DOBIND        EQU #059C9 *  (primitive)
=@             EQU #05BEC *  (primitive)
=CONTEXT@      EQU #06B15 *  (primitive)
=TRUE          EQU #06BF4 *  (primitive)

=PUSHA         EQU #06BF9 *  (assembly)

=FALSE         EQU #06C33 *  (primitive)
=XOR           EQU #06C4D *  (primitive)
=NOT           EQU #06C65 *  (primitive)
=EQ            EQU #06CA1 *  (primitive)
=AND           EQU #06CB9 *  (primitive)
=OR            EQU #06CE8 *  (primitive)
=EQUAL         EQU #06D0A *  (primitive)
=#0=           EQU #06E17 *  (primitive)
=#=            EQU #06E8A *  (primitive)
=#<>           EQU #06EBF *  (primitive)
=#>            EQU #06EF4 *  (primitive)
=#+            EQU #06F2D *  (primitive)
=#1+           EQU #06F60 *  (primitive)
=#1-           EQU #06F7F *  (primitive)
=#2+           EQU #06F9E *  (primitive)
=#2*           EQU #06FE0 *  (primitive)

=POP2#         EQU #070BD *  (assembly)

=ZERO          EQU #071A9 *  (systbin)
=ONE           EQU #071B3 *  (systbin)
=TWO           EQU #071BD *  (systbin)
```

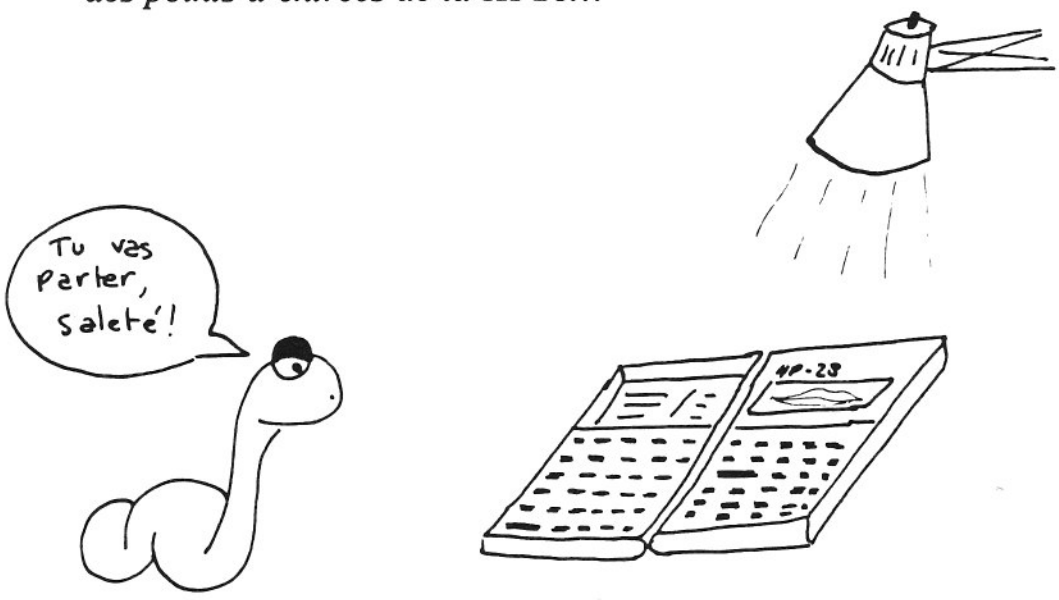
```
=THREE          EQU #071C7 *  (systbin)
=FOUR           EQU #071D1 *  (systbin)
=FIVE           EQU #071DB *  (systbin)
=SIX            EQU #071E5 *  (systbin)
=SEVEN          EQU #071EF *  (systbin)
=EIGHT          EQU #071F9 *  (systbin)
=NINE           EQU #07203 *  (systbin)
=TEN            EQU #0720D *  (systbin)
=ELEVEN         EQU #07217 *  (systbin)
=TWELVE         EQU #07221 *  (systbin)
=THIRTEEN       EQU #0722B *  (systbin)
=FOURTEEN       EQU #07235 *  (systbin)
=FIFTEEN        EQU #0723F *  (systbin)
=SIXTEEN        EQU #07249 *  (systbin)
=SEVENTEEN      EQU #07253 *  (systbin)
=EIGHTEEN       EQU #0725D *  (systbin)
=NINETEEN       EQU #07267 *  (systbin)
=TWENTY         EQU #07271 *  (systbin)
=TWENTYONE      EQU #0727B *  (systbin)
=TWENTYTWO      EQU #07285 *  (systbin)
=TWENTYTHREE    EQU #0728F *  (systbin)
=TWENTYFOUR     EQU #07299 *  (systbin)
=TWENTYFIVE     EQU #072A3 *  (systbin)
=TWENTYSIX      EQU #072AD *  (systbin)
=TWENTYSEVEN    EQU #072B7 *  (systbin)
=TWENTYEIGHT    EQU #072C1 *  (systbin)
=TWENTYNINE     EQU #072CB *  (systbin)
=THIRTY         EQU #072D5 *  (systbin)
=THIRTYONE      EQU #072DF *  (systbin)
=THIRTYTWO      EQU #072E9 *  (systbin)
=THIRTYTHREE    EQU #072F3 *  (systbin)
=THIRTYFOUR     EQU #072FD *  (systbin)
=THIRTYFIVE     EQU #07307 *  (systbin)
=THIRTYSIX      EQU #07311 *  (systbin)
=THIRTYSEVEN    EQU #0731B *  (systbin)
=THIRTYEIGHT    EQU #07325 *  (systbin)
=THIRTYNINE     EQU #0732F *  (systbin)
=FORTY          EQU #07339 *  (systbin)
=FORTYONE       EQU #07343 *  (systbin)
=FORTYTWO       EQU #0734D *  (systbin)
=FORTYTHREE     EQU #07357 *  (systbin)

=TrueTrue       EQU #07393 *  (prog)
=failed         EQU #073A7 *  (prog)
=!*trior        EQU #07429 *  (code)
=!*triand       EQU #07489 *  (prog)

=xDUP           EQU #075DC *  (user RPL)
=xDUP2          EQU #075F6 *  (user RPL)
=xSWAP          EQU #07610 *  (user RPL)
=xDROP          EQU #0762A *  (user RPL)
=xDROP2         EQU #07644 *  (user RPL)
=xROT           EQU #0765E *  (user RPL)
=xOVER          EQU #07678 *  (user RPL)
=xDEPTH         EQU #07692 *  (user RPL)
=xDROPN         EQU #076B1 *  (user RPL)
=xDUPN          EQU #076CB *  (user RPL)
=xPICK          EQU #076E5 *  (user RPL)
=xROLL          EQU #076FF *  (user RPL)
=xROLLD         EQU #07719 *  (user RPL)
=xCLEAR         EQU #07733 *  (user RPL)
=x>LIST         EQU #07752 *  (user RPL)
=xR>C           EQU #0776C *  (user RPL)
=xRE            EQU #077A4 *  (user RPL)
=xIM            EQU #077F0 *  (user RPL)
=xSUB           EQU #07832 *  (user RPL)
=xLIST>         EQU #07888 *  (user RPL)
=xC>R           EQU #078C0 *  (user RPL)
=xSIZE          EQU #078EE *  (user RPL)
=xPOS           EQU #0796C *  (user RPL)
=x>STR          EQU #079AE *  (user RPL)
=xSTR>          EQU #079C8 *  (user RPL)
=xNUM           EQU #079EC *  (user RPL)
=xCHR           EQU #07A10 *  (user RPL)
=xTYPE          EQU #07A34 *  (user RPL)
=x>ARRY         EQU #07AE4 *  (user RPL)
=XEQ>ARRAY      EQU #07B39 *  (prog)
=xARRY>         EQU #07B85 *  (user RPL)
=xRDM           EQU #07BD6 *  (user RPL)
=xCON           EQU #07C5E *  (user RPL)
=xIDN           EQU #07D4A *  (user RPL)
=xTRN           EQU #07DE1 *  (user RPL)
=xPUT           EQU #07E32 *  (user RPL)
=xPUTI          EQU #07FB9 *  (user RPL)
=xGET           EQU #0814A *  (user RPL)
=xGETI          EQU #08277 *  (user RPL)
=FORTYSIX       EQU #083EE *  (systbin)
=xSAME          EQU #087C3 *  (user RPL)
=xAND           EQU #087EC *  (user RPL)
=xOR            EQU #0886F *  (user RPL)
=xNOT           EQU #088F2 *  (user RPL)
=xXOR           EQU #08952 *  (user RPL)
=x==            EQU #089D5 *  (user RPL)
=x<>            EQU #08A53 *  (user RPL)
=x<?            EQU #08AD6 *  (user RPL)
=x>?            EQU #08B4A *  (user RPL)
=x<=            EQU #08BBE *  (user RPL)
=x>=            EQU #08C32 *  (user RPL)
=x=             EQU #08CA6 *  (user RPL)
=xNEG           EQU #08D24 *  (user RPL)
=xABS           EQU #08D70 *  (user RPL)
=xCONJ          EQU #08DBC *  (user RPL)
=xPI            EQU #08E08 *  (user RPL)
=xMAXR          EQU #08E36 *  (user RPL)
=xMINR          EQU #08E64 *  (user RPL)
=xCONSTANTe     EQU #08E92 *  (user RPL)
=xi             EQU #08EC0 *  (user RPL)
=x+             EQU #08EEE *  (user RPL)
=x-             EQU #08FD0 *  (user RPL)
=x*             EQU #09076 *  (user RPL)
=x/             EQU #09144 *  (user RPL)
=x^             EQU #091FE *  (user RPL)
=xINV           EQU #092DB *  (user RPL)
=xARG           EQU #09327 *  (user RPL)
=xP>R           EQU #0937D *  (user RPL)
=xR>P           EQU #09469 *  (user RPL)
```

```
=xSIGN      EQU #094E2 *  (user RPL)
=xSQRT      EQU #09524 *  (user RPL)
=xSQ        EQU #0958E *  (user RPL)
=xSIN       EQU #09602 *  (user RPL)
=xCOS       EQU #09644 *  (user RPL)
=xTAN       EQU #09686 *  (user RPL)
=xSINH      EQU #096C8 *  (user RPL)
=xCOSH      EQU #0970A *  (user RPL)
=xTANH      EQU #0974C *  (user RPL)
=xASIN      EQU #0978E *  (user RPL)
=xACOS      EQU #09802 *  (user RPL)
=xATAN      EQU #09876 *  (user RPL)
=xASINH     EQU #098B8 *  (user RPL)
=xACOSH     EQU #098FA *  (user RPL)
=xATANH     EQU #09969 *  (user RPL)
=xEXP       EQU #099DD *  (user RPL)
=xLN        EQU #09A1F *  (user RPL)
=xLOG       EQU #09A89 *  (user RPL)
=xALOG      EQU #09AF3 *  (user RPL)
=xLNP1      EQU #09B35 *  (user RPL)
=xEXPM      EQU #09B6D *  (user RPL)
=xFACT      EQU #09BA5 *  (user RPL)
=xIP        EQU #09BDD *  (user RPL)
=xFP        EQU #09C15 *  (user RPL)
=xFLOOR     EQU #09C4D *  (user RPL)
=xCEIL      EQU #09C85 *  (user RPL)
=xXPON      EQU #09CBD *  (user RPL)
=xMAX       EQU #09CF5 *  (user RPL)
=xMIN       EQU #09D41 *  (user RPL)
=xMOD       EQU #09D8D *  (user RPL)
=xMANT      EQU #09DD9 *  (user RPL)
=xD>R       EQU #09E11 *  (user RPL)
=xR>D       EQU #09E49 *  (user RPL)
=x>HMS      EQU #09E77 *  (user RPL)
=xHMS>      EQU #09E9B *  (user RPL)
=xHMS+      EQU #09EBF *  (user RPL)
=xHMS-      EQU #09EE3 *  (user RPL)
=xRNRM      EQU #09F07 *  (user RPL)
=xCNRM      EQU #09F2B *  (user RPL)
=xDET       EQU #09F4F *  (user RPL)
=xDOT       EQU #09F73 *  (user RPL)
=xCROSS     EQU #09F97 *  (user RPL)
=xRSD       EQU #09FBB *  (user RPL)
=x%         EQU #09FE9 *  (user RPL)
=x%T        EQU #0A035 *  (user RPL)
=x%CH       EQU #0A081 *  (user RPL)
=xRAND      EQU #0A0C3 *  (user RPL)
=xRDZ       EQU #0A0DD *  (user RPL)
=xCOMB      EQU #0A101 *  (user RPL)
=xPERM      EQU #0A125 *  (user RPL)
=xLCD>      EQU #0A149 *  (user RPL)
=x>LCD      EQU #0A163 *  (user RPL)
=xDGTIZ     EQU #0A187 *  (user RPL)
=xMENU      EQU #0A1A1 *  (user RPL)
=xRCL       EQU #0A1CF *  (user RPL)
=XEQRCL     EQU #0A1F7 *  (prog)
=xSTO       EQU #0A220 *  (user RPL)
=xPURGE     EQU #0A26C *  (user RPL)
=xMEM       EQU #0A32B *  (user RPL)
=xORDER     EQU #0A359 *  (user RPL)
=xCLUSR     EQU #0A396 *  (user RPL)
=xKILL      EQU #0A3B0 *  (user RPL)
=xABORT     EQU #0A3CA *  (user RPL)
=xERRN      EQU #0A3E4 *  (user RPL)
=xERRM      EQU #0A3FE *  (user RPL)
=xEVAL      EQU #0A418 *  (user RPL)
=xIFTE      EQU #0A450 *  (user RPL)
=xIFT       EQU #0A500 *  (user RPL)
=xSYSEVAL   EQU #0A54C *  (user RPL)
=xDISP      EQU #0A5A6 *  (user RPL)
=xRND       EQU #0A5D4 *  (user RPL)
=xBEEP      EQU #0A616 *  (user RPL)
=x>NUM      EQU #0A63A *  (user RPL)
=xLAST      EQU #0A654 *  (user RPL)
=xWAIT      EQU #0A73A *  (user RPL)
=xCLLCD     EQU #0A75E *  (user RPL)
=xKEY       EQU #0A778 *  (user RPL)
=xCLMF      EQU #0A792 *  (user RPL)
=xSF        EQU #0A7B6 *  (user RPL)
=xCF        EQU #0A7F3 *  (user RPL)
=xFS?       EQU #0A830 *  (user RPL)
=xFC?       EQU #0A86D *  (user RPL)
=xDEG       EQU #0A8AF *  (user RPL)
=xRAD       EQU #0A8D3 *  (user RPL)
=xFIX       EQU #0A8F7 *  (user RPL)
=xSCI       EQU #0A92F *  (user RPL)
=xENG       EQU #0A967 *  (user RPL)
=xSTD       EQU #0A99F *  (user RPL)
=xFS?C      EQU #0A9B9 *  (user RPL)
=xFC?C      EQU #0AA0A *  (user RPL)
=xBIN       EQU #0AA60 *  (user RPL)
=xHEX       EQU #0AA94 *  (user RPL)
=xOCT       EQU #0AAAE *  (user RPL)
=xSTWS      EQU #0AAC8 *  (user RPL)
=xRCWS      EQU #0AAEC *  (user RPL)
=xRCLF      EQU #0AB06 *  (user RPL)
=xSTOF      EQU #0AB20 *  (user RPL)
```

To be continued...

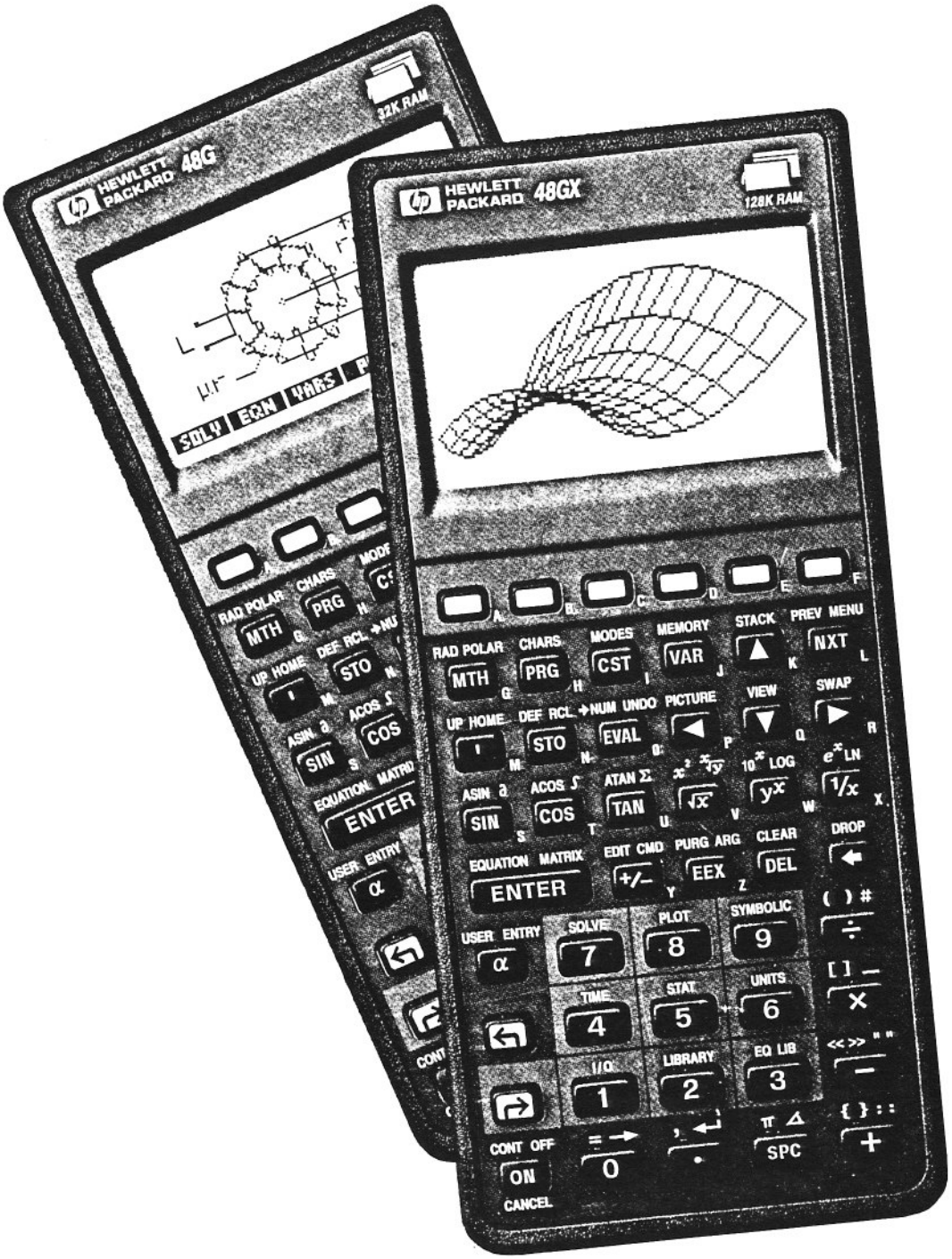
Jean François Garnier (242)

Jean-Francois Garnier à la recherche des points d'entrées de la HP28...



HP48

J. Belin/P. de Sacy	La HP48GX	*15*
P. Silvestre de Sacy	*Les Secrets de la HP48* tome II	*17*
J. F. Garnier	Extension des menus (Acte II)	*18*
J. F. Garnier	Programmation System RPL	*18*
G. Toublanc	Trucs et Astuces	*19*


HEWLETT PACKARD 48G
32K RAM
SOLV EQN VARS
HEWLETT PACKARD 48GX
128K RAM
RAD POLAR
CHARS
MODES
MEMORY
STACK
PREV MENU
MTH
PRG
CST
VAR
NXT
UP HOME
DEF RCL →NUM UNDO PICTURE
VIEW
SWAP
STO
EVAL
ASIN ∂
ACOS ∫
ATAN Σ
10^x LOG
e^x LN
SIN
COS
TAN
1/x
EQUATION MATRIX
EDIT CMD
PURG ARG
CLEAR
DROP
ENTER
+/-
EEX
DEL
USER ENTRY
SOLVE
PLOT
SYMBOLIC
TIME
STAT
UNITS
I/O
LIBRARY
EQ LIB
CONT OFF
ON
CANCEL
SPC

LES NOUVELLES HP48

Depuis son apparition en 1990, la HP48 n'avait évolué que par le biais des nouvelles versions de la Rom. Aujourd'hui, Hewlett-Packard introduit deux nouvelles versions de la machine, nommées HP48G et HP48GX (le 'G' signifiant 'Graphic') dotées de possibilités absentes de la version précédente.

Aspect

Même si la forme du boitier et l'emplacement des touches est identique à la 48SX, plusieurs éléments permettent de distinguer d'un coup d'oeil les nouvelles versions :

- Tout d'abord, la couleur du plastique utilisé pour le boitier semble avoir changé. A moins que cela soit dû aux changements de couleur du clavier, dont nous parlerons plus tard, elle donne maintenent une impression plus "métalique". Mais ce n'est que pûrement subjectif, et sera vérifié dès que nous pourrons voir les deux versions l'une à côté de l'autre.

- Un petit logo montrant un tracé d'équation est visible au dessus du coin supérieur/gauche de l'ecran. Juste au dessous de ce logo, une mention *128K RAM* (pour la GX, 32 pour la G indique la capacité mémoire de la machine.

- Un autre élément visible sur la nouvelle version est que le clavier a changé. En premier lieu, la couleur des touches et des marquages *shiftées* sont devenues 'lavande' pour le Shift gauche et vert clair pour le Shift Droit. Ensuite, une zone plus claire entoure maintenat les touches numériques et le Shift Droit. Mais la différence la plus fondementale est que de nombreuses marquages ont été déplacés, renommés, ajoutés ou supprimés. Au lieu de vous faire subir la liste complète de ces changements, nous vous encourageons à comparer les marquages de votre machine par rapport à la photo visible sur l'autre page.

- Enfin, même si il présente les mêmes caractéristiques que pour les précédentes versions, l'écran a été changé et possède maintenant un meilleur contraste.

Le Hardware

La capacité de la Rom est maintenant de 512 k. Ceci peut paraître étonnant pour les programmeurs en assembleur, car cela correspond exactement à la capacité d'adressage du microprocesseur Saturn ! En fait, les nouvelles versions utilisent une méthode de *page swaping* permettant de n'accéder à la Rom que par des blocs plus petits, dans une 'fenêtre'. C'est cette méthode qui est utilisée pour accéder à la Rom du HP95, ou plus généralement à l'EMS sur les PC.

Afin de compenser les pertes de performances dûes au Bank Switching, la fréquence du Saturn est passée à 4 Mhz. Ceci permet d'obtenir, au niveau utilisateur, un gain de vitesse de l'ordre de 40 à 50%.

Comme nous l'avons mentionné plus haut, la Ram interne de la GX est portée à 128 Ko. Celle de la G reste limitée à 32 Ko.

La GX comporte les mêmes connecteurs de cartes que la SX. Cependant, il est maintenant possible d'installer une carte ayant une capacité de 4 Mo ! Celle ci ne peut être placée que dans le deuxième port. En fait, seule la carte (de 128 ko maxi) présente dans le port 1 peut être mergée avec la mémoire principale. La carte présente dans le port 2 est accessible par blocs de 128 ko, accessible par des ports numérotés 2 à 33 (dans le cas des cartes de 4 Mo).

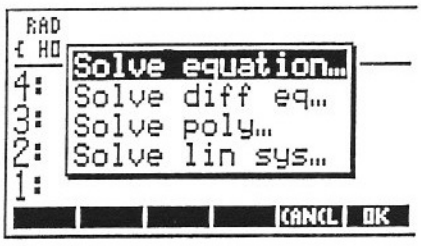
Bien sûr, les quasi-totalité des cartes d'applications de la 48SX pouront être placées dans la nouvelle version, à l'exception de certaines cartes utilisant des points d'entrées non supportés, et les cartes RAM TDS 256 et 512 ko.

Les nouvelles Fonctions

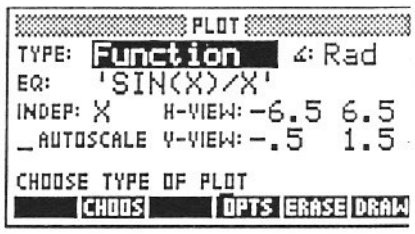
Le passage à 512 ko de la capacité de la Rom est dû en grande partie à l'inclusion de la carte *Equation Library* (aussi appelée HP Solve) regroupant quelques centaines d'equations applicables dans tous les dommaines (électricité, calculs des fluides, poutres...). Les seules omissions sont la table périodique des éléments et le jeu Tetris (inclus dans la dernière version). Il n'est pas possible de placer la carte *Equation Library* dans la GX.

Mais les grandes innovations des nouvelles versions concernent l'interface utilisateur, puisque deux nouvelles fonctionnalités ont été rajoutées :

- Des menus déroulants permettent de "naviguer" plus facilement dans l'arborescence des commandes, en conservant les options précédement sélectionnées.



- Pour les commandes recourant à de nombreux paramètres (tracés de courbes, par exemples), des masques de saisie permettent d'entrer les données de façon plus évidente que dans la pile. Eventuellement, les paramètres par défaut sont donnés.



L'utilisateur a bien sûr la possibilité de définir ses propres menus déroulants et masques de saisie.

En addition à ces aides, une fonction TEACH permet d'effectuer une mini-démonstration d'une vingtaine des fonctions les plus complexes.

En ce qui concerne les autres possibilités graphiques, il est maintenant possible d'effectuer des tracés en 3D, à la manière de DERIVE sur le HP95.

Le changement de la fréquence du microprocesseur est particulièrement visible lors des tracés en 2D. Mais cette accélération a aussi permis aux programmeurs de Corvallis d'ajouter quelques "extras" aux fonctions déjà existantes. Par exemple, lorsque l'on sélectionne une zone de tracé que l'on veut "zoomer", la zone choisie est maintenant visualisée en temps réel par un rectangle, alors qu'il n'y avait que deux croix visibles sur la série S.

De nouvelles commandes et fonctions ont été ajoutées. Parmi celles ci, nous pouvons citer en vrac : Résolution d'équations différentielles, Transformées de Fourrier, de nouvelles fonctions matricielles des commandes permettant de visualiser des portions de listes, de nouvelles fonctions matricielles (dont certaines ajoutées par James Donnelly d'après son *Donnelly's Tool Library*), une commande ANIMATE permetant d'afficher successivement (et très rapidement) une série de GROBs...

D'autre part, il est maintenant possible de regrouper les paramètres successifs d'une même fonction dans une liste. Par exemple, `2 SF 3 SF 4 SF` peut être remplacé par `{ 2 3 4 } SF`. De même `{ 1 2 3 } SQ` donne `{ 1 4 9 }` et `{ 1 2 3 } * { 4 5 6}` donne `{ 4 10 18 }`. En fait quasiment toutes les fonctions (y compris celles demandant plusieurs paramètres, telles que BEEP) peuvent accepter une liste comme argument !

Du côté des communications, le mode Xmodem est maintenant supporté (en mode binaire uniquement).

En ce qui concerne les programmes, la compatibilité ascendante est bien entendue assurée pour l'User RPL. Cependant, quelques modifications sont à noter, telles que la possibilité de déclarer certaines variables en *local*, afin de les réserver au programme et d'accélérer leur exécution. Cette déclaration s'effectue en plaçant le caractère "←" en tête du nom de la variable. D'autre part, il faudrait mettre à jour les programmes utilisant # MENU et # TMENU, car certains codes de touches auraient changé.

Pour la programmation en External ou en assembleur les points d'entrées supportés sur les S semblent conservés dans la grande majorité. La nouvelle Rom posséderait maintenant près de 4000 points d'entrées, les nouveaux étant placés dans les 256 ko supplémentaires.

Les Notices

Les nouvelles 48 sont livrées avec 3 manuels : Un manuel de prise en mains (108 pages, pour la version anglaise ?), un manuel de référence (480 p.) et un guide de traduction (en 5 langues des différents écrans et menus (80 p.).

Cependant, le manuel de référence comprenant maintenant tout ce qui concerne l'Equation Library, il ne reste quasiment plus de place pour les parties concernant la programmation. D'autre part, HP ayant constaté qu'une très faible majorité des utilisateurs programmaient, la partie décrivant la programmation en détail est maintenant livrée dans un "Advanced User's Manual" (de 722 pages), malheureusement disponible en option.

Les accessoires

HP commercialisera deux cartes Ram dès la sortie da la machine. La première est bien connue, puisqu'il s'agit de la 128 ko de la 48SX. La seconde possède une capacité de 1 Mo. Son prix n'était pas encore fixé le jour de bouclage de ce numéro. D'autre part, Sparcom commercialiserait une carte de 4 Mo, à un prix avoisinant les 1000 $.

Une nouvelle version du "Connectivity Kit" sortira avec les nouvelles versions de la machine.

Prix et disponibilité

Une bonne nouvelle, le prix des nouvelles machines est légèrement en baisse, puisque la GX sera vendue 2086 F.HT (prix catalogue) et la G : 1162 F.HT. Elles devraient être disponibles en France vers la fin du mois d'aout.

Il est à noter (une bonne nouvelle ne venant jamais seule !), que HP organise pour la sortie de la machine, et pour un temps limité, une promotion visant à offrir le Connectivity Pack avec toute machine achetée !

Pierre Silvestre de Sacy (572)
Jacques Belin (123)

LES SECRETS...
TOME II

Voici arrivé le tome 2 des "Secrets de la HP 48" de Jean Michel Ferrard. Il vient prolonger le tome 1 en poussant plus loin l'utilisation des SYSEVALs. En effet, certaines routines en ROM ne peuvent pas être utilisées selon la forme `#xxxxxh SYSEVAL` mais doivent être appelées par des méthodes différentes.

Dans ce nouveau livre sont présentés tout d'abord deux programmes pour contourner ce problème. L'un et pour la 48 l'autre pour PC (écrit en Pascal). Le gros avantage sur ce qui existe déjè est que l'on a le listing avec quelques commentaires succincts, mais qui permettent de comprendre le rôle de chaque procédure et fonction, et donc d'apporter des modifications si l'on veut "personnaliser" son programme.

Faisant suite a ces deux programmes, sont exposés de nouveaux Point d'Entrées Assembleur et Systeme-RPL, accompagnés de nombreux exemples, permettant maintenant aux néophites d'écrire de puissants programmes.

Pour vous encourager a écrire plein de programmes, on trouve a la suite des utilitaires tel que PEEK, POKE, un programme pour retrouver le "source" d'un programme un System RPL, et deux programmes pour créer et défaire des librairies, le tout avec uniquement des appels en ROM (ce que personnellement je considere comme un exploit car si j'en avait été l'auteur je n'aurait pas resisté a en écrire une partie en assembleur. Mais là est bien l'un des buts du livre : montrer qu'il est presque toujours possible de ne pas toucher à l'assembleur).

Heureusement pour moi, J.M. Ferrard a bien voulu dans la deuxième partie de son livre aborder la programmation en assembleur. Il y decrit tout d'abord les différents registres du Saturn avec leur Champs, suivis d'un cours rapide d'arithmétique non signée et signée en hexadécimal puis en Décimal Codé Binaire.

Vient ensuite une description detaillée de chaque instructions avec leurs mnémoniques, leur codage en fonction du champ concerné, et pour certains, le nombre de cycles. Puis la liste Alphabétique des mnémoniques puis des codes-machine.

Mais pour écrire des programmes en assembleur (sous forme de mnémoniques), il faut un assembleur (programme chargé de transformer le source en programme éxécutable). Pas de problème, les deux chapitres qui suivent vont vous combler : un assembleur sur 48 et un sur PC.

Pour ne pas avoir a réécrire ce qui l'a déjà été, une liste bien etoffée d'adresse de routines en ROM (donc à apeller par `GOSVBL` ou `GOVLNG`) montre encore une fois que J.M. Ferrard a voulu nous livrer tous les secrets de la ROM de la HP 48.

Pour finir, quelques programmes reprennent pour l'exemple certains des utilitaires vus dans la première partie, mais cette fois en assembleur ainsi que de nouveaux utilitaires.

Je n'ai malheureusement pas eu le temps de taper tous les programmes... Toutefois le programme `EXTERNPC` pour écrire du Systeme-RPL sur un PC en pascal m'a posé quelques problèmes lors de la compilation (j'utilise le Turbo Pascal 6) : `car(n mod 16);` me faisait une erreur de type (*n* etant un "longint" ou un "integer" et *car* etant une fonction prennant en paramètre un "byte"). Normalement le pascal devrait convertir le resultat de l'operation en "byte". Pour contouner l'erreur j'ai du créer une variable locale de type "byte" puis lui affecter le resultat de l'opération et enfin transmettre cette variable en paramètre.

Autre probleme (une erreur dans le listing cette fois), page 56, à la dernière ligne de la procédure `Convertir_Ligne` :

```
if (not stop) and (item<>'') then ajouter(result);
```

`item` est une fonction à qui il faut transmettre une chaine de caractères, or, on ne lui transmet rien... Il suffit de remplacer `item` par *ident*.

Apres avoir lu ce deuxième tome, la HP 48 n'aura plus beaucoup de secrets pour vous... Alors on peut penser que l'on ne verra jamais de Tome 3... A moins que la nouvelle HP48GX n'apporte a J.M. Ferrard assez de secrets pour un nouveau volume !

Pour ceux qui ne veulent pas taper tous les listing de ce tome, il est possible de les obtenir sur disquette, ainsi que les exécutables. Il faut s'adresser a l'éditeur.

Pierre Silvestre de Sacy (572)

MENUS, SYSEVAL ET VERSIONS DE ROM

Dans le numéro 85 de *JPC*, Pierre Sylvestre de Sacy nous expliquait dans un très intéressant article comment créer des menus spéciaux du type de ceux utilisés de façon interne par la HP48 : exécution d'un programme à chaque affichage du menu, menus avec options, menus inversés, etc ... en utilisant pour cela des objets RPL internes. J'emettrai cependant une réserve concernant l'emploi d'adresses non supportées (que Pierre Sylvestre ne prenne pas ce qui suit comme une bête critique de son travail, je trouve ses articles fort intéressants). C'est pourquoi je me permet de proposer quelques modifications. Vous trouverez des versions de SYSRCL et →PRG utilisant les points d'entrées dans un autre article dans ce numéro (*System RPL sur HP48*), et voici SMENU réécrit en utilisant COM→ décrit dans ce même article. Les autres programmes ne posent pas de problèmes.

```
SMENU
«
  [ 8 ] CTRL
  # 40788h SYSRCL                @ TakeOver
  SWAP COM→ 1 + →PRG
»
```

D'une façon plus générale, je pense qu'il serait souhaitable de n'utiliser que des points d'entrées supportés dans les articles publiés. Bien sûr, rien ne vous empêche d'utiliser d'autres adresses pour vos programmes personnels, mais pensez à ceux qui possèdent une version différente et qui souhaitent utiliser vos réalisations. La version J a déjà donné lieu à certains problèmes (avec des programmes des Goodies Disks par exemple). Le problème risque de devenir encore plus important avec l'arrivée de la 48GX, car il s'agit là d'une révision majeure de la ROM, comparée aux différentes versions de la 48SX. A l'heure où j'écris ces lignes, je n'ai pas la certitude que la 48GX conserve l'ensemble des points d'entrées (bien que cela soit vraisemblable), mais je suis sûr que bon nombre d'adresses non supportées vont être perturbées.

Ma position n'a rien de dogmatique. Si vous souhaitez utiliser une adresse non supportée pour une bonne raison (pas de point d'entrée équivalent, raccourcissement drastique du programme) allez-y, mais indiquez-le clairement en rappelant la ou les versions de 48 sur lesquels vous avez testé votre programme. Et surtout, que ces quelques réflexions ne vous empêchent pas de créer et de publier !

Jean-François Garnier (242)

SYSTEM RPL SUR HP48S

Pour que les possesseurs de HP48S(X) ne se sentent pas lésées, voici les adaptations pour cette machine (toutes versions) des SYSEVALs présentés dans l'article *System RPL sur HP28S*

SYSRCL (#nnnnnh → objet)

Rappelle un objet interne sur la pile, à partir de son adresse (entier binaire utilisateur).

```
«
  # 4003h SYSEVAL       @ TWO
  # 56B6h SYSEVAL       @ NTHELCOMP
  DROP
»
```

FRE (-- → n)

Retourne la taille de la mémoire restante en octets. Ne provoque de garbage collector.

```
«
  # 5F61h SYSEVAL       @ MEM
  # 18DBFh SYSEVAL      @ UNCOERCE
  2 /
»
```

CRNAME ("nom" → 'nom')

Création d'un nom à partir d'une chaine. Permet de créer des noms interdits ('(*)', 'DATE+', ...).

```
«
  # 5B15h SYSEVAL       @ $>ID
»
```

B→SB (#nnnnh → <nnnnh>)

Conversion d'un binaire utilisateur en binaire système. De nombreux objets RPL attendent en arguments des binaires systèmes. Note : la HP28 affiche les binaires système sous forme de System Objet.

```
«
  # 5A03h SYSEVAL      @ HXS>#
»
```

SB→B (<nnnnh> → #nnnnh)

Conversion d'un binaire système en binaire utilisateur. De nombreux objets RPL fournissent en résultats des binaires systèmes.

```
«
  # 59CCh SYSEVAL      @ HXS>#   #>HXS
»
```

COM→ (composite → obj1 obj2 ... objn n)
Décomposition d'un objet composite en ses éléments. Comme OBJ→, mais fonctionne aussi sur les programmes et les algébriques.

```
«
  # 54AFh SYSEVAL      @ INNERCOMP
  # 18DBFh SYSEVAL     @ UNCOERCE
»
```

→PRG (obj1 obj2 ... objn n → program)

Création d'un objet programme à partir de ses éléments Associé à COM→, permet des manipulations sur les programmes : suppression des « », insertion de System Objets rappelés par SYSRCL, ...

```
«
  # 18CEAh SYSEVAL     @ COERCE
  # 5445h SYSEVAL      @ ::N
»
```

CEVAL (liste → ?)

Evaluation d'un objet composite. Comme EVAL, mais s'applique aux listes de façon semblable aux programmes (comme sur la 48). Application : exécution de listes construites par programme.

```
«
  # 18EBAh SYSEVAL     @ COMPEVAL
»
```

PARSE ("chaine" → objet 1
(si pas d'erreur)
→ "chaine" <nnnnnh> "chaine'" 0
(si erreur))

Analyse d'une chaine et création de l'objet correspondant.

```
«
  # 238A4h SYSEVAL     @ palparse
  # 5380Eh SYSEVAL     @ COERCEFLAG
»
```

Jean-François Garnier (242)

TRUCS, ASTUCES ET PETITS UTILITAIRES

Si vous avez *RPL48 Toolkit* de Detlef Mueller et Raymond Hellstern et que vous désirez décompiler un objet dans votre HP48 (ROM ou RAM) en mode pas à pas (exemple : voir l'article sur UNLINE) voici un mini programme qui permet de voir les lignes de décompilation et de concaténer l'ensemble des lignes.

Mode d'emploi:

```
"" ENTER ENTER
#aaaaa   (l'adresse de départ)
RPLD
```

puis vous poursuivez de ligne en ligne avec RPLD

Lorsque vous arrêtez, les 2 premiers niveaux sont à éliminer pour récupérer l'ensemble des lignes.

RPLD

```
« SWAP DROP DCADR
OVER "                    @ newline
" + 4 ROLL SWAP + 3 ROLLD
»
```

La même procédure est à utiliser pour désassembler du code, il suffit de remplacer DCADR par DA1.

UNA

```
« SWAP DROP DA1
OVER "                    @ newline
" + 4 ROLL SWAP + 3 ROLLD
»
```

Guy Toublanc (276)


Ne vous battez plus avec votre calculateur: ACHETEZ un HP!
Make love, not war!

MS-DOS

J. Belin — Visualisation d'encombrement du disque *22*

Le coin des codes *31*

COMPTEZ VOS CLUSTERS !

Lorsque l'on veut copier le contenu d'un directory (et ses sous-répertoires s'ils existent), il est souvent intéressant de connaitre avec précision l'encombrement réel de ces fichiers, et non la taille affichée par la commande DIR. En effet, comme vous le savez (ou devriez le savoir !), les fichiers sont enregistrés dans des zones de stockage possédant une taille multiple de 512 octets appelées *clusters* (grappes). Un fichier de grande taille peut bien sûr occuper plusieurs clusters, mais dès qu'un octet occupe le début d'un d'entre eux, il sera réservé en totalité, même si il est vide à 99.9% !

Or, sur MS-DOS, la taille des clusters varie suivant les différents supports utilisés. Pour mémoire, les tailles de cluster courament utilisées sont les suivantes :

- 512 octets : Unites de disque du HP95, disquettes 3"½ 1.44 Mo et 5"¼ 1.2 Mo.
- 1024 octets : disquettes 3"½ 720 ko et 5"¼ 360 ko.
- 2048 : Partitions de disque dur < 128 Mo.
- 4096 octets : Partitions de disque dur < 256 Mo, partitions de disques secondaires.
- 8192 octets : partition de disque dur < 512 Mo...

Lorsque l'on copie un fichier, la taille réelle du fichier sera ajustée en fonction de la taille de cluster du disque de destination. Si nous avons un grand nombre de petits fichiers à copier vers un support ayant une taille de cluster différente, la taille occupée sur le disque de destination peut largement excéder les prévisions initiales. Ceci explique qu'il est parfois impossible de copier le contenu d'une disquette 1.44 Mo pleine à 20% (288 ko) dans une disquette vierge de 360 ko !

L'utilitaire présenté ce mois-ci permet de mieux apréhender l'encombrement des directories, en donnant non seulement la taille réellement utilisée par le directory source, mais aussi celle qu'elle serait dans le disque de destination.

Appelé DU (Disk Usage), sa syntaxe est la suivante :

```
du  [ -5 | -1 | -2 | -4 | -8 ]  [[disque:]path]
```

Les options '-5', '-1'... indiquent la taille de cluster du support de destination. Le caractère d'option indique le premier chiffre de la taille de cluster : '5' indique 512 octets, '1' pour 1024, '2' pour 2048...

Si aucune option n'est spécifiée, le programme effectuera les calculs en fonction de la taille de cluster du disque contenant les directories. dans tous les cas, la taille de cluster prise en compte est affichée.

Il est possible de spécifier le directory de départ après les options. Par défaut, c'est le directory courant.

Par exemple :

```
>du \lang\tc
```

affichera :

```
Cluster size =  2048

  184320 bytes in \lang\tc
 2668544 bytes in \lang\tc\BIN
  176128 bytes in \lang\tc\INCLUDE
    6144 bytes in \lang\tc\INCLUDE\SYS
 1198080 bytes in \lang\tc\LIB
  155648 bytes in \lang\tc\BGI
   57344 bytes in \lang\tc\CLASSLIB
       0 bytes in \lang\tc\CLASSLIB\EXEMPLES
  149504 bytes in \lang\tc\CLASSLIB\INCLUDE
   57344 bytes in \lang\tc\CLASSLIB\LIB
  151552 bytes in \lang\tc\CLASSLIB\SOURCE
       0 bytes in \lang\tc\EXEMPLES

Total :    4804608 bytes
```

Les encombrements sont indiquées en fonction de la taille de cluster du support.

Supposons maintenant que nous voulions copier le directory include (et son sous répertoire) sur une disquette 1.44 Mo, ou dans la mémoire du HP95. Sachant que la taille de cluster pour ces supports est de 512 octets, nous taperons la commande suivante :

```
>du -5 \lang\tc\include
```

Ce qui nous donnera :

```
Cluster size =  512

  147968 bytes in \lang\tc\include
    3072 bytes in \lang\tc\include\SYS

Total :     151040 bytes
```

Vous pouvez constater que l'encombrement des directories est inférieur à ce qui était affiché dans l'exemple précédent. Dans le cas contraire, si nous voulons copier ces fichiers dans un "gros" disque dur (avec une taille de cluster de 8192 octets) nous obtenons ceci :

```
Cluster size =  8192

 344064 bytes in \lang\tc\include
  24576 bytes in \lang\tc\include\SYS

Total :     368640 bytes
```

Soit un supplément de plus de 200 ko !

Fonctionnement du programme

Avant de décrire la structure du programme, je voudrais préciser que la version présentée aujourd'hui est issue d'une plus ancienne, écrite en langage C. Afin d'économiser la mémoire du HP95, je l'ai réécrit en assembleur. Ceci a fait passer sa taille de 7962 octets à 1532. Soit 20% de la taille originale, et avec des fonctionnalités équivalentes ! Notez aussi que je me suis efforcé de faire tenir le programme dans moins de 3 clusters de HP95, ce qui limitait sa taille à 1536 octets...

Le coeur du programme est basé sur une fonction récursive, permettant de calculer l'encombrement de chaque directory puis effectuer la même opération dans les sous-répertoires suivants :

L'algorithme de cette fonction est le suivant :

```
du(path)
  sauvegarde de l'ancienne DTA
  allocation buffer
  selection nouvelle DTA dans buffer
  taille directory = 0
  pour chaque fichier du directory
     ajouter taille du fichier à celle du directory
  afficher taille directory
  taille totale = taille totale + taille directory
  pour chaque entrée du directory
     si (attribut DIRECTORY) et (entrée # "." ou "..")
        newpath = "path\entrée"
        appeler du(newpath)
  libération buffer
  restauration ancienne DTA
```

La DTA (Direct Transfert Area) est une zone de mémoire utilisée par les fonctions de recherche de fichiers spécifiés par les *wildcards* '*' et '?'. Son adresse de départ peut être spécifiée par le programmeur, afin de faciliter l'utilisation dans les fonctions récursives.

Afin de fonctionner correctement, la fonction utilise des variables locales, qui sont principalement la DTA et un chaine de caractère contenant le chemin courant auquel on ajoute le nom des sous-répertoires à traiter (cette chaine servira de paramètre pour les appels récursifs). L'assembleur ne permettant pas, comme le langage C, de créer automatiquement les variables locales (invisibles aux niveaux suivants des appels récursifs) nous devons nous occuper nous même de cette tâche. Ceci est fait en demandant au DOS d'allouer un bloc de mémoire au début de la fonction, et de le libérer en fin d'execution. En pratique, il y aura donc un buffer alloué pour chaque niveau d'imbrication de directory, soit une dizaine au maximum.

Hormis cette fonction, le programme en utilise d'autres, qui peuvent être utilisées par d'autres programmes.

Les plus importantes concernent la manipulation des paramètres de la ligne de commandes et l'analyse des option. Etant utilisateur du langage C, j'ai choisi de recréer certaines choses existant dans ce langage, afin de simplifier le portage de mes utilitaires. La première chose à faire était donc de créer une fonction créant la variable `argc` et le tableau `*argv[]` parmettant d'accéder directement aux paramètres. Ce qui n'existe pas à l'origine dans le DOS, puisque seule la chaine séparée par des espaces est accessible.

La deuxième fonction, `getopt`, permet d'analyser les options entrées sous le format habituel sous UNIX. Il s'agit en fait d'une réécriture assez fidèle de celle que j'avais présenté (en langage C) dans le n° 76 de *JPC*. Toutefois, sur le programme présenté ce mois ci, j'ai profité d'une particularité des options de l'utilitaire afin d'en diminuer sa taille. Je vous conseille donc d'attendre le mois prochain avant d'essayer de comprendre son utilisation exacte. En effet, je devrais publier un autre utilitaire untilisant cette fonction de façon plus "normale" et complète.

Les autres fonctions sont principalement des commandes de manipulation de chaines. Etant précédées d'un en-tête, elles ne devraient pas poser trop de problèmes d'utilisation. Certaine fonctions (`strcpy` par exemple) sembleront mal écrites (car utilisant mal les possibilités du microprocesseur) pour les programmeurs expérimentés. Ceci est dû au fait qu'elles on été écrites afin de faire des manipulations entre du PSP (pointé par le registre ES) vers le segment de données du programe (pointé par DS), alors que le microprocesseur est optimisé pour faire les tranferts dans l'autre sens.

Ceci dit, il ne vous reste plus qu'à regarder l'encombrement de vos directories préférés !

Jacques Belin (123)

```
;*****************************************************************************
;*****************************************************************************
;**                                                                         **
;**                              DU.ASM                                     **
;**                  Affichage de l'encombrement des répertoires            **
;**                                                                         **
;**   Dernière modification : 28/05/93        Auteur : Jaques Belin         **
;**       Création                                                          **
;**                                                                         **
;*****************************************************************************
;*****************************************************************************
        DOSSEG
code    segment para 'CODE'
        assume cs:code,ds:code
        ORG     100h
start:  jmp     main                    ; première inst. exécutée

PATH_SIZ EQU    80                      ; taille maxi d'un chemin
DTA_SIZE EQU    48                      ; taille de la DTA

FA_ARCH  EQU    1                       ; codes d'attributs des fichiers
FA_HIDD  EQU    2
FA_SYST  EQU    4
FA_VOL   EQU    8
FA_DIR   EQU    16

        INCLUDE getopt.inc              ; routine d'analyse des options

optstr  DB      "51248",0               ; chaine des options
errflg  DB      0                       ; flag d'erreur

cl_size DW      0                       ; taille d'un cluster en octets
tot_size DW     0                       ; taille totale (en clusters)
nb_dirs DW      0                       ; nombre de directories sur disque

drive   DB      0                       ; code du disque (0=A, 1=B...)
cur_path DB     PATH_SIZ dup (?)        ; directory de base

                                        ; Messages d'erreur
bad_disk DB     "Bad drive specified",13,10  ; (pas de fin de chaine ($))
usage   DB      "usage :  du  [ -5 | -1 | -2 | -4 | -8 ]  [ [disk:] path ]"
        DB      13,10,13,10,"$"


dcl_size DB     "   Cluster size = "
dcl_si_2 DB     "xxxxx",13,10,"$"

di_size  DB     13,10
di_siz_2 DB     "xxxxxxxxxx bytes in "
di_siz_3 DB     PATH_SIZ dup (?)

total   DB      13,10,13,10,"   Total : "
total_2 DB      "xxxxxxxxxx bytes",13,10,"$"

no_path DB      ".",0
back_sl DB      "\",0
wilds   DB      "\*.*",0


main:   mov     bx,cs
        mov     ds,bx                   ; on assure DS=CS
        mov     es,bx                   ; on assure ES=CS

        mov     ah,4Ah                  ; on réjuste taille du bloc programmee
        mov     bx,4096/16
        int     21h
        mov     sp,4096                 ; on replace le pointeur de pile

        call    parse                   ; parsing dans argv[]

        xor     cl,cl
optloop: call   getopt                  ; analyse des options
        cmp     al,EOF                  ; fin d'analyse ?
        je      main_2                  ; oui, on passe à la suite du prgm
        cmp     al,"?"                  ; si erreur, on arme le flag
        je      opterr
        mov     cl,al                   ; sinon on stocke le code dans cl
        jmp     nextopt
opterr: inc     byte ptr [errflg]
nextopt: jmp    optloop                 ; on boucle tant qu'on a pas EOF
        ;-------------

main_2: cmp     byte ptr [errflg],0     ; si une erreur a été détectée...
        je      main_3
        mov     dx,OFFSET usage         ; ...on affiche le message d'erreur...
        mov     ah,09h
        int     21h
        jmp     exit                    ; ...et on sort.
        ;----------

main_3: cmp     cl,0                    ; si une option a été spécifiée...
        je      main_4
        mov     al,cl                   ; on recherche sa position dans "51248"
        mov     dx,OFFSET optstr        ;  (la position dans la chaine indique
        call    strchr                  ;  en '2^p' le nbre de secteurs/cluster
        sub     dx,OFFSET optstr
        mov     cl,dl                   ; on effectue un décalage sur ax...
        mov     ax,1                    ;  suivant la position dans la chaine
        shl     ax,cl                   ;  afin d'avoir le nombre de sect/clust
        mov     cx,512                  ; et on multiplie par 512
        mul     cx
        mov     [cl_size],ax            ; on n'a plus qu'a stocker

main_4: call    getdisk                 ; acquisition du drive courant
        mov     [drive],al              ; drive = disque courant
        mov     ax,[optind]             ; argc == optind ? (pas de path ?)
        cmp     ax,[argc]
        jne     main_5                  ; non, on récupère le path
        push    es                      ; sauvegarde segment PSP
        push    ds                      ; copie de "." dans cur_path
        pop     es
        mov     di,OFFSET no_path
        mov     si,OFFSET cur_path
        call    strcpy
        pop     es                      ; récupération segment PSP
        jmp     main_8
        ;------------

main_5: mov     bx,[optind]             ; bx = optind
        shl     bx,1
        mov     si,OFFSET argv          ; si = argv
        mov     di,ds:[si+bx]           ; di ^ argv[optind]
        mov     al,es:[di+1]            ; al = 2ème caractère du paramètre
        cmp     al,":"                  ; est-ce un descripteur de disque ?
        jne     main_7                  ; non -> on prendra le disque courant
        mov     al,es:[di]              ; al = 1er caractère du paramètre
        cmp     al,"a"
        jna     main_6
        cmp     al,"z"
        ja      main_6
        sub     al,32                   ; normalisation en majuscules
main_6: sub     al,"A"                  ; conversion en numéro de disque
        mov     [drive],al              ; nouveau disque
```

```
main_7:  mov    si,OFFSET cur_path
         call   strcpy                  ; copie argv[optind] dans cur_path
                                        ; (ES:DI déjà défini plus haut)
         mov    dx,si
         call   strlen
         mov    bx,cx
         cmp    byte ptr [si+bx-1],"\" ; dernier caractère de cur_path="\" ?
         jne    main_8                  ; non, on continue
         mov    byte ptr [si+bx-1],0    ; oui, on l'efface

main_8:  mov    ax,[cl_size]
         cmp    ax,0                    ; un cl_size a t-il été spécifié ?
         jne    main_A                  ; oui, on continue
         mov    ah,36h                  ; int 21/36h = informations disque
         mov    dl,[drive]
         inc    dl
         int    21h                     ; acquisition des infos disque
         cmp    ax,0ffffh
         jne    main_9                  ; si ax différent de FFFF, disque ok
         mov    dx,OFFSET bad_disk      ; sinon ->erreur...
         mov    ah,09h
         int    21h
         jmp    exit                    ; ...et on sort du programme.
         ;----------

main_9:  mul    cx                      ; AX=AX*CX : taille cluster en octets
         mov    [cl_size],ax            ; drive = taille cluster
main_A:  xor    dx,dx                   ; conversion taille cluster en chaine
         mov    si,OFFSET dcl_si_2
         mov    bx,5
         call   ltoa
         mov    dx,OFFSET dcl_size      ; affichage taille cluster
         mov    ah,09h
         int    21h

         push   ds                      ; on fait pointer ES:DI sur cur_path
         pop    es
         mov    di,offset cur_path
         call   d_usage                 ; APPEL A LA ROUTINE PRINCIPALE

         cmp    word ptr [nb_dirs],1    ; si 1 seul directory traité, on sort
         jna    exit

         mov    ax,[tot_size]
         mul    [cl_size]               ; DX:AX = AX*[cl_size] : taille totale
         mov    si,OFFSET total_2       ; conversion taille en chaine
         mov    bx,10
         call   ltoa
         mov    dx,OFFSET total         ; affichage taille totale
         mov    ah,09h
         int    21h

exit:    mov    ah,4ch                  ; retour au DOS
         int    21h
```

```
;*************************************************
; D_USAGE : affiche récursivement l'encombrement de
;           chaque directory
; entrées : ES:DI adresse du directory courant
;           (terminé par caractère 0)
;*************************************************
dir_size DW     (?)                    ; encombrement du directory, en clusters

d_usage  PROC   NEAR
         push   ax
         push   bx
         push   cx
         push   dx
         push   bp
         push   es
         push   di
         push   ds
         push   si

         call   getdta                  ; sauvegarde de la DTA dans la pile
         push   ax

         mov    word ptr cs:[dir_size],0 ; initialisation taille courante a 0
         inc    word ptr cs:[nb_dirs]  ; inc. nombre de directory traités

         mov    ah,48h
         mov    bx,((DTA_SIZE+PATH_SIZ)/16)+1 ; taille en paragraphes
         int    21h                     ; allocation d'un bloc de mémoire
         push   ax

         push   di                      ; sauvegarde paramètre d'entrée
         push   es
         mov    bp,sp                   ; on va se servir de BP comme indice
                                        ; pour les accès dans la pile

         ;à partir d'ici on a dans la pile :
         ; bp    =  ES niveau précédent
         ; bp+2  =  DI niveau précédent
         ; BP+4  =  segment du buffer courant
         ; BP+6  =  DTA précédente
         ;   .
         ;   .

         xor    bx,bx                   ; fait pointer nouvelle DTA sur buffer
         call   setdta

         mov    ds,ax                   ;copie path courant après DTA ds buffer
         mov    si,DTA_SIZE
         call   strcpy

         push   cs                      ; ES=CS
         pop    es                      ; ajoute "\*.*" en fin de curpath
         mov    di,OFFSET wilds
         call   strcat

                                        ; RECHERCHE DES FICHIERS DU REPERTOIRE
         mov    ah,4Eh                  ; cherche premier fichier du répertoire
         mov    dx,si
         mov    cx,FA_HIDD+FA_SYST      ; attributs de recherche
         int    21h

d_usa_1: jc     d_usa_3                 ; si carry -> recherche terminée
         mov    ax,ds:[1Ah]             ; DS:[1A,1C] = taille fichier en octets
         mov    dx,ds:[1Ch]
         div    cs:[cl_size]            ; taille du fichier en clusters
         cmp    dx,0                    ; +1 si cluster incomplet
```

```
        je      d_usa_2
        inc     ax
d_usa_2: add    cs:[dir_size],ax        ; calcul encombrement du directory
        mov     ah,4Fh                  ; cherche fichier suivant
        int     21h
        jmp     d_usa_1                 ; et on boucle
        ;-------------

d_usa_3: mov    ax,cs:[dir_size]        ; récupération encombrement directory
        add     cs:[tot_size],ax        ; on en profite pour le grand total
        mul     word ptr cs:[cl_size]   ; encombrement directory en octets
        mov     bx,10                   ; conversion taille en chaine
        push    cs
        pop     ds
        mov     si,OFFSET di_siz_2
        call    ltoa

        mov     ax,ss:[bp]              ; ES:DI = paramètre d'entrée
        mov     es,ax
        mov     di,ss:[bp+2]
        mov     si,OFFSET di_siz_3
        call    strcpy                  ; copie nom directory en fin de chaine

        mov     bx,1                    ; affichage du texte dans 'stdout'
        mov     dx,offset di_size
        call    fprintf
                                        ; RECHERCHE DANS LES SOUS REPERTOIRES
        mov     dx,ss:[bp+4]            ; on reprend cur_path comme base
        mov     ds,dx
        mov     dx,DTA_SIZE             ; cherche premier fichier
        mov     ah,4Eh
        mov     cx,FA_DIR
        int     21h

d_usa_4: jc     d_usa_9                 ; si carry -> terminé
        mov     al,ds:[15h]             ; si attribut fichier # directory...
        and     al,FA_DIR
        jz      d_usa_5
        cmp     byte ptr ds:[1Eh],"."   ; ... et nom directory = "." ou ".."
        je      d_usa_5                 ; ...on passe à l'entrée suivante

        mov     ax,ss:[bp]              ; ES:DI = paramètre d'entrée (récursif)
        mov     es,ax
        mov     di,ss:[bp+2]
        mov     si,DTA_SIZE             ; copie paramètre dans buffer
        call    strcpy

        push    cs                      ; on ajoute "\" au paramètre
        pop     es
        mov     di,OFFSET back_sl
        call    strcat

        push    ds                      ; on ajoute le nom de l'entrée
        pop     es
        mov     di,1Eh
        call    strcat

        mov     di,si                   ; ES:DI = nouveau paramètre
        call    d_usage                 ; appel récursif

d_usa_5: mov    ah,4Fh                  ; cherche fichier suivant
        int     21h
        jmp     d_usa_4                 ; et on boucle
        ;-------------

d_usa_9: pop    ax                      ; on depile les données inutiles
        pop     ax

        pop     es                      ; libération du buffer
        mov     ah,49h
        int     21h

        pop     ax                      ; reinitialisation de l'ancienne DTA
        xor     bx,bx
        call    setdta

d_usa_e: pop    si
        pop     ds
        pop     di
        pop     es
        pop     bp
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret
d_usage ENDP


;*************************************************
; STRCAT  : place une chaine à la fin d'une autre
; entrées : ES:DI à adresse de la chaine source
;           (terminée par caractère 0)
;           DS:SI à adresse de la chaine destination
;*************************************************
strcat  PROC    NEAR
        push    cx
        push    dx
        push    si

        mov     dx,si                   ; calcul longueur de chaine
        call    strlen
        add     si,cx                   ; on copie après le dernier caractère
        call    strcpy

strca_1: pop    si
        pop     dx
        pop     cx
        ret
strcat  ENDP


;*************************************************
; GETDISK : Donne le disque courant
; entrées : Aucune
; sortie  : AL = disque (0=A 1=B 2=C..)
;*************************************************
getdisk PROC    NEAR
        push    bx

        push    ax                      ; sauvegarde de AX
        mov     ah,19h                  ; lecture numéro du disque
        int     21h
        pop     bx                      ; on fait en sorte que AH ne soit pas
        mov     bl,al                   ;  modifié, par la fonction
        mov     ax,bx

        pop     bx
        ret
getdisk ENDP
```

```
;*****************************************************
; GETDTA  : Donne l'adresse de la DTA
; entrées : Aucune
; sortie  : AX:BX = segment DTA
;*****************************************************
getdta  PROC    NEAR
        push    es

        mov     ah,2Fh                  ;  lecture adresse DTA
        int     21h
        mov     ax,es

        pop     es
        ret
getdta  ENDP


;*****************************************************
; SETDTA  : Positionne l'adresse de la DTA
; entrées : AX:BX = adresse DTA
; sortie  : aucune
;*****************************************************
setdta  PROC    NEAR
        push    ax
        push    bx
        push    dx
        push    ds

        mov     ds,ax                   ; on positionn les paramètres
        mov     dx,bx
        mov     ah,1Ah                  ; selection de la DTA
        int     21h

        pop     ds
        pop     dx
        pop     bx
        pop     ax
        ret
setdta  ENDP

;*****************************************************
; FPRINTF : affiche une chaine de caractères dans
;           le flux BX
; entrées : DS:DX = Chaine
;           BX = handle
; sortie  : Aucune
;*****************************************************
fprintf PROC    NEAR
        push    ax
        push    cx

        call    strlen                  ; calcul de la longueur de la chaine
        mov     ah,40h                  ; affichage
        int     21h

        pop     cx
        pop     ax
        ret
fprintf ENDP
```

```
;*****************************************************
; LTOA : Conversion d'un entier long en chaine
;        justifiée à droite
;        Pas de code de fin de chaine ('\0')
; Entrées: DX:AX = nombre à convertir
;          BX = nombre de digits total
;          DS:SI = Adresse du début de chaine
;*****************************************************
ltoa    PROC    NEAR
        push    bx
        push    cx
        push    di
        push    si
        push    bp

        push    dx                      ; on sauve le paramètre dans la pile
        push    ax                      ;  (pour simplifier les accès)
        mov     bp,sp

        add     si,bx                   ; calcul position du dernier caractère
        dec     si                      ;  de la chaine (digit poids faible)

        mov     cx,10                   ; diviseur (base)

ltoa_1: mov     ax,[bp+2]               ; on récupère le mot de poids fort
        sub     dx,dx                   ; la division ne se fait que sur AX
        div     cx                      ; division, le reste est en DX
        mov     [bp+2],ax               ; on sauve le quotient

        mov     ax,[bp]                 ; mot de poids faible
        div     cx                      ; division DX:AX/CX, le digit est en DX
        mov     [bp],ax                 ; sauve quotient

        add     dl,30h                  ; conversion du digit en ASCII
        mov     [si],dl                 ; écriture dans chaine
        dec     si
        dec     bx

        cmp     word ptr [bp],0         ; teste si fini
        jne     ltoa_1                  ; sinon, on boucle
        cmp     word ptr [bp+2],0
        jne     ltoa_1

        mov     cx,bx                   ; on remplit les digits non utilisés
ltoa_2: mov     byte ptr [si]," "       ; par des espaces
        dec     si
        loop    ltoa_2

        pop     ax
        pop     dx
        pop     bp
        pop     si
        pop     di
        pop     cx
        pop     bx
        ret
ltoa    ENDP

Code    ends
        END     start                   ; fin du programme
```

```
;*****************************************************************************
;*****************************************************************************
;**                              PARSE.INC                                  **
;**                 Analyse et segmentation de la ligne de commande         **
;**                                                                         **
;**   Dernière modification : 28/05/93        Auteur : Jaques Belin         **
;**       Création                                                          **
;**                                                                         **
;*****************************************************************************
;*****************************************************************************
argc    DW    1                       ; nombre d'arguments
argv    DW    64 DUP (0)              ; tableau de pointeurs sur arguments
psp     DW    (?)                     ; segment du psp

;*************************************************
; PARSE   : analyse la ligne de commande et crée une
;           table de paramètres de format identique
;           à 'argc' '*argv[]' du langage C
; entrées : aucune
; retour  : argc, argv et psp initialisés
;           CX = nombre de paramètres
;           ES = segment du PSP
; Note : cette version ne gère pas argv[0] (nom du programme)
;*************************************************
parse   PROC  NEAR
        push  ax
        push  bx
        push  dx
        push  di
        push  ds
        push  si

        mov   ah,62h                  ; lecture adresse du PSP
        int   21h
        mov   [psp],bx

        ;stockage 'command' dans argv [0]

        mov   ax,[psp]                ; stockage adresse psp dans es
        mov   es,ax
        mov   bx,80h                  ; bx = ^ sur longueur ligne
        mov   di,bx
        inc   di                      ; di ^ sur 1er caractère
        xor   ch,ch
        mov   cl,es:[bx]              ; nombre de caractères dans cx
        cmp   cx,0                    ; si 0 caractères -> 0 param -> fin
        je    parse_6
        mov   dl,0                    ; inparam = 0
        mov   si,offset argv
        add   si,2                    ; position sur argv[1]

parse_1: mov  al,byte ptr es:[di]     ; caractère courant dans al
        cmp   al,22h                  ; si caractère '"'
        jne   parse_2
        xor   dl,0ffh                 ; inversion de inparam
        mov   byte ptr es:[di],0      ; on remplace par caract de fin
        jmp   parse_4                 ; on passe au caractère suivant
        ;-------------

parse_2: cmp  al," "                  ; si pas caractère espace
        jne   parse_4                 ; on passe au caractère suivant
        cmp   dl,0ffh                 ; est-on dans 1 par délimité par '"'
        je    parse_4                 ; oui, on passe au caractère suivant
        mov   byte ptr es:[di],0      ; on remplace par caract de fin
        cmp   al,es:[di+1]            ; caractère suivant = espace ?
        je    parse_4                 ; oui, on passe au caractère suivant
                                      ; le prochain caractère est le début
                                      ; d'un paramètre
        mov   bx,di
        inc   bx                      ; bx = adresse du début du param
        cmp   byte ptr es:[bx],22h    ; si caractère suivant est un '"'
        jne   parse_3
        inc   bx                      ; on saute ce caractère
parse_3: cmp  byte ptr es:[bx],0dh    ; si retour chariot
        je    parse_5
        mov   [si],bx                 ; stockage adresse dans argv[]
        inc   si
        inc   si
        inc   word ptr [argc]

parse_4: inc  di                      ; caractère suivant
        loop  parse_1                 ; on boucle...
parse_5: mov  byte ptr es:[di],0      ; on remplace 'CR' par caract de fin
        mov   cx,[argc]
        dec   cx

parse_6: pop  si
        pop   ds
        pop   di
        pop   dx
        pop   bx
        pop   ax
        ret
parse   ENDP


;*****************************************************************************
;*****************************************************************************
;**                              GETOPT.INC                                 **
;**                 Analyse des options de la ligne de commande             **
;**                                                                         **
;**   Dernière modification : 28/05/93        Auteur : Jaques Belin         **
;**       Création                                                          **
;**                                                                         **
;**   Note : Module pour programme assembleur, non prévu pour fonctionner   **
;**          en tant que fonction C                                         **
;**          Inclue les fonctions strcpy, strlen et strchr                  **
;*****************************************************************************
;*****************************************************************************
EOF     EQU   255
NULL    EQU   0ffffh

        INCLUDE parse.inc

; variables globales
optarg  DW    (?)                     ; ptr sur argument optionnel (dans psp)
optind  DW    1                       ; indice sur paramètre
SW      DB    "-"                     ; caractère d'option

;variables locales
_ptopt  DW    (?)                     ; pointeur sur option
_retour DB    "?"                     ; code retourné
_lenarg DW    (?)                     ; longueur argument
```

```
;*************************************************
; GETOPT  : analyse des options de la ligne de commande
; retour  AL : code de l'option
; Note : La fonction "parse" doit avoir été appelée précédement
;        Une variable 'optstr' contenant les options
;         doit être définie dans le programme principal
;*************************************************
getopt   PROC   NEAR
         push   bx
         push   cx
         push   dx
         push   es
         push   di
         push   ds
         push   si

         mov    word ptr [optarg],NULL

         mov    ax,[argc]               ; si fin de la ligne de commande
         cmp    ax,[optind]
         je     getop_E                 ; (getop_E : saut intérmediaire)

getop_1: mov    ax,[psp]                ; ES = segment du PSP
         mov    es,ax
         mov    si,offset argv          ; calcul de ds:si pour argv[optind]
         mov    bx,[optind]
         shl    bx,1
         mov    di,ds:[si+bx]           ; ES:DI = adresse du paramètre
         mov    al,es:[di]              ; al = premier caractère du paramètre

         cmp    al,[SW]                 ; *argv[optind] = SW ?
         jne    getop_E                 ; non, -> getop_5 via getop_E
                                        ; (pour éviter un saut trop long)
         push   ds                      ; sauvegarde ds
         push   es                      ; ds = es
         pop    ds
         mov    dx,di
         call   strlen                  ; _lenarg = longueur chaine
         pop    ds                      ; récupération ds
         mov    word ptr [_lenarg],cx
         mov    al,byte ptr es:[di+1]
         mov    [_retour],al
         cmp    al,[SW]                 ; cas option '--' ?
         jne    getop_2                 ; non, on continue
         inc    word ptr [optind]       ; optind++
getop_E: jmp    getop_5                 ; -> fin d'analyse
         ;-------------

getop_2: mov    dx,offset optstr
         call   strchr                  ; recherche option dans optstring
         cmp    dx,NULL                 ; si option non trouvée
         je     getop_6                 ; sortie avec erreur
         mov    bx,dx
         cmp    byte ptr ds:[bx+1],":" ; est ce une option à paramètre ?
         jne    getop_4                 ; non -> getop_4
         cmp    word ptr [_lenarg],2    ; longueur argument = 2 ?
         jne    getop_3                 ; non ->getop_3
         inc    word ptr [optind]       ; accès à l'argument suivant
         mov    bx,[optind]
         shl    bx,1
         mov    di,ds:[si+bx]
         mov    [optarg],di             ; acquisition adresse paramètre
         mov    cx,[optind]             ; si optind >= argc...
         cmp    cx,[argc]
         jae    getop_6                 ; ..alors erreur et sortie
         cmp    byte ptr es:[di],"-"    ; ..ou *optarg = "-"...
         je     getop_6                 ; ..alors erreur et sortie
         jmp    getop_7
         ;-------------

getop_3: add    di,2
         mov    [optarg],di             ; acquisition adresse paramètre
         jmp    getop_7
getop_4: cmp    word ptr [_lenarg],2    ; si options non groupées ("-abc")..
         je     getop_7                 ; ..on sort
                                        ; si options groupées...
         mov    bx,[optind]             ; on remplace "-abc" par "-bc"
         shl    bx,1
         inc    word ptr ds:[si+bx]     ; ..en décalant l'adresse du paramètre
         mov    byte ptr es:[di+1],"-" ; ..et en remplacant "a" par "-"
         jmp    getop_8
         ;-------------

getop_5: mov    al,EOF
         jmp    getop_9
         ;-------------

getop_6: mov    byte ptr [_retour],"?" ; retour code d'erreur
getop_7: inc    word ptr [optind]       ; on incrémente optind
getop_8: mov    al,[_retour]
getop_9: pop    si
         pop    ds
         pop    di
         pop    es
         pop    dx
         pop    cx
         pop    bx
         ret
getopt   ENDP


;*************************************************
; STRCHR  : cherche un caractère dans une chaine
; entrées : AL caractère à trouver
;           DS:DX à adresse de la chaine de référence
; retour    si trouvé : DS:DX = pointeur sur caractère
;                               dans chaine
;           si non trouvé : DX = NULL
;*************************************************
strchr   PROC   NEAR
         push   bx

         mov    bx,dx
strch_1: cmp    byte ptr ds:[bx],0      ; fin de la chaine ?
         je     strch_2                 ; oui -> erreur
         cmp    ds:[bx],al              ; caractère trouvé ?
         je     strch_3                 ; oui -> fin
         inc    bx                      ; non, on pointe sur caractère suivant
         jmp    strch_1                 ; on boucle...
         ;-------------

strch_2: mov    bx,NULL

strch_3: mov    dx,bx
         pop    bx
         ret
strchr   ENDP
```

```
;*****************************************************
; STRLEN  : calcule la longueur d'une chaine
; entrées : DS:DX adresse de la chaine
; retour  : CX = longueur
;*****************************************************
strlen   PROC   NEAR
         push   bx

         xor    cx,cx
         mov    bx,dx
strle_1: cmp    byte ptr ds:[bx],0      ; fin de la chaine ?
         je     strle_2                 ; oui -> fin de routine
         inc    cx                      ; incrémentation taille
         inc    bx                      ; on pointe sur caractère suivant
         jmp    strle_1                 ; on boucle...
         ;-------------

strle_2: pop    bx
         ret
strlen   ENDP


;*****************************************************
; STRCPY  : copie de chaine
; entrées : ES:DI à adresse de la chaine source
;           (terminée par caractère 0)
;           DS:SI à adresse de la destination
;*****************************************************
strcpy   PROC   NEAR
         push   ax
         push   di
         push   si

strcp_1: mov    al,es:[di]
         mov    ds:[si],al
         cmp    al,0
         je     strcp_2
         inc    di
         inc    si
         jmp    strcp_1
         ;-------------

strcp_2: pop    si
         pop    di
         pop    ax
         ret
strcpy   ENDP
```

LE COIN DES CODES

La compilation de certains programmes, tels que ceux écrits en assembleur, nécessitent souvent un logiciel que ne possèdent pas tous nos lecteurs. Le *Coin des Codes* permet de résoudre ce problème.

Ce programme est destiné à l'entrée des exécutables MS-DOS uniquement. Il nécessite GWBASIC.

Mode d'emploi :

1- Lancer le programme : GWBASIC MAKEDOS.BAS
2- Entrer le nom du fichier destination.
3- Entrer la taille du fichier.
4- Entrer les listes de codes puis le checksum (en prenant soin d'entrer les codes héxadécimaux en majuscules). En cas d'erreur corriger la ligne, en prenant soin de placer le curseur après le dernier caractère avant de taper sur la touche d'entrée.
5- Une fois que toutes les lignes sont entrées, sortir du GWBASIC en exécutant la commande SYSTEM. Le nouveau programme est immédiatement disponible.

Note : La taille du fichier résultant peut être supérieure d'un octet à ce qui est affiché dans le listing. Cela n'est pas un problème.

```
DU.COM                          (MS-DOS)
                              1532 octets

      0123 4567 89AB CDEF   sm

000: E965 0301 0000 0000   9E7
001: 0000 0000 0000 0000   367
002: 0000 0000 0000 0000   CE7
003: 0000 0000 0000 0000   667
004: 0000 0000 0000 0000   FE7
005: 0000 0000 0000 0000   967
006: 0000 0000 0000 0000   2E7
007: 0000 0000 0000 0000   C67
008: 0000 0000 0000 0000   5E7
009: 0000 0000 0000 0000   F67
00A: 0000 0000 0000 0000   8E7
00B: 0000 0000 0000 0000   267
00C: 0000 0000 0000 0000   BE7
00D: 0000 0000 0000 0000   567
00E: 0000 0000 0000 0000   EE7
00F: 0000 0000 0000 0000   867
010: 0000 0000 0000 0050   232
011: 5352 571E 56B4 62CD   0EC
012: 2189 1E85 01A1 8501   D1E
013: 8EC0 BB80 008B FB47   BFD
014: 32ED 268A 0F83 F900   9F5
015: 7450 B200 BE05 0183   65F
016: C602 268A 053C 2275   31A
```

Programme MAKEDOS.BAS

```
 10 INPUT "Nom du fichier : ",NOM$ : INPUT "Nombre d'octets : ",N : N=N*2
 20 OPEN NOM$ FOR OUTPUT AS #1 : CLOSE #1 : KILL NOM$
 30 OPEN "bin.tmp" FOR OUTPUT AS #1 : S=0 : P$="---- ---- ---- ----"
 40 NLINES=N\16 : LENLAST=(N MOD 16)+((N MOD 16)\5)
 50 IF (N MOD 16)=0 THEN NLINES=NLINES-1 : LENLAST=LEN(P$)
 60 FOR X=0 TO NLINES
 70   IF X=NLINES THEN P$=LEFT$(P$, LENLAST)
 80   C$=P$
 90   X2$="00"+HEX$(X) : PRINT RIGHT$(X2$,3);":";
100   Y=CSRLIN : LOCATE Y,6 : PRINT C$; : LOCATE Y,6 : INPUT "",C$ : Y=Y-1
110   LOCATE Y,27 : PRINT " sm = ---" : LOCATE Y,33 : INPUT "",D$
120   M=S
130   FOR Z=1 TO LEN(C$)
140     IF MID$(C$,Z,1)<>" " THEN M=(M+((Z-(Z\5))*ASC(MID$(C$,Z,1)))) MOD 4096
150   NEXT Z
160   D2$="00"+HEX$(M) : D2$=RIGHT$(D2$,3)
170   IF D2$<>D$ THEN PRINT "Erreur de somme" : BEEP : GOTO 90
180   FOR Z=1 TO LEN(C$) STEP 2
190     IF MID$(C$,Z,1)=" " THEN Z=Z-1 : GOTO 230
200     CH=ASC(MID$(C$,Z,1))-48 : IF CH>9 THEN CH=CH-7
210     CL=ASC(MID$(C$,Z+1,1))-48 : IF CL>9 THEN CL=CL-7
220     PRINT#1,CHR$((16*CH)+CL);
230   NEXT Z
240   S=M
250 NEXT X
260 CLOSE #1 : NAME "bin.tmp" AS NOM$ : END
```

```
017: 0A80 F2FF 26C6 0500  045
018: EB2C 903C 2075 2780  CB9
019: FAFF 7422 26C6 0500  911
01A: 263A 4501 7418 8BDF  7E8
01B: 4326 803F 2275 0143  3A6
01C: 2680 3F0D 740B 891C  1E3
01D: 4646 FF06 0301 47E2  ED4
01E: C126 C605 008B 0E03  BB8
01F: 0149 5E1F 5F5A 5B58  ACC
020: C300 0001 002D 0000  573
021: 3F00 0053 5152 0657  0F2
022: 1E56 C706 0102 FFFF  09B
023: A103 013B 0603 0274  C0A
024: 37A1 8501 8EC0 BE05  A6B
025: 018B 1E03 02D1 E38B  8B7
026: 3826 8A05 3A06 0502  491
027: 751E 1E06 1F8B D7E8  462
028: A600 1F89 0E09 0226  0E0
029: 8A45 01A2 0802 3A06  D14
02A: 0502 7507 FF06 0302  953
02B: EB5F 90BA 0C03 E871  782
02C: 0083 FAFF 7458 8BDA  7E2
02D: 807F 013A 7534 833E  54B
02E: 0902 0275 23FF 0603  1F8
02F: 028B 1E03 02D1 E38B  046
030: 3889 3E01 028B 0E03  D44
031: 023B 0E03 0173 2F26  9F6
032: 803D 2D74 29EB 2C90  880
033: 83C7 0289 3E01 02EB  65F
034: 2290 833E 0902 0274  22E
035: 1A8B 1E03 02D1 E3FF  1AD
036: 0026 C645 012D EB0F  08F
037: 90B0 FFEB 0D90 C606  F39
038: 0802 3FFF 0603 02A0  C29
039: 0802 5E1F 5F07 5A59  A76
03A: 5BC3 538B DA80 3F00  868
03B: 7407 3807 7406 43EB  5F2
03C: F4BB FFFF 8BD3 5BC3  6D0
03D: 5333 C98B DA80 3F00  4DE
03E: 7404 4143 EBF7 5BC3  40C
03F: 5057 5626 8A05 8804  07D
040: 3C00 7404 4746 EBF3  EE6
041: 5E5F 58C3 3531 3234  B0C
042: 3800 0000 0000 0000  49F
043: 0000 0000 0000 0000  E1F
044: 0000 0000 0000 0000  79F
045: 0000 0000 0000 0000  11F
046: 0000 0000 0000 0000  A9F
047: 0000 0000 0000 0000  41F
048: 0000 0000 0000 0000  D9F
049: 0000 0000 0000 0000  71F
04A: 0000 0000 0000 0000  09F
04B: 0000 0000 0000 0000  A1F
04C: 0000 0000 0000 0000  39F
04D: 0000 4261 6420 6472  EF4
04E: 6976 6520 7370 6563  ACB
04F: 6966 6965 640D 0A75  828
050: 7361 6765 203A 2020  39C
051: 6475 2020 5B20 2D35  011
052: 207C 202D 3120 7C20  C6A
053: 2D32 207C 202D 3420  88D
054: 7C20 2D38 205D 2020  488
055: 5B20 5B64 6973 6B3A  28E
056: 5D20 7061 7468 205D  F44
057: 0D0A 0D0A 2420 2020  AB8
058: 436C 7573 7465 7220  6A4
059: 7369 7A65 203D 2078  368
05A: 7878 7878 0D0A 240D  11E
05B: 0A78 7878 7878 7878  EA9
05C: 7878 7820 6279 7465  B16
05D: 7320 696E 2000 0000  5E1
05E: 0000 0000 0000 0000  F61
05F: 0000 0000 0000 0000  8E1
060: 0000 0000 0000 0000  261
061: 0000 0000 0000 0000  BE1
062: 0000 0000 0000 0000  561
063: 0000 0000 0000 0000  EE1
064: 0000 0000 0000 0000  861
065: 0000 0000 0000 0000  1E1
066: 0000 0000 0000 0000  B61
067: 0000 0000 000D 0A0D  7FF
068: 0A20 2020 546F 7461  45B
069: 6C20 3A20 7878 7878  180
06A: 7878 7878 7878 2062  DE0
06B: 7974 6573 0D0A 242E  B77
06C: 005C 005C 2A2E 2A00  8E3
06D: 8CCB 8EDB 8EC3 B44A  951
06E: BB00 01CD 21BC 0010  607
06F: E80C FD32 C9E8 8BFD  73D
070: 3CFF 740F 3C3F 7405  550
071: 8AC8 EB05 90FE 0612  312
072: 03EB EA80 3E12 0300  F60
073: 740A BA7F 03B4 09CD  EC3
074: 21E9 D900 80F9 0074  B93
075: 1B8A C1BA 0C03 E821  990
076: FE81 EA0C 038A CAB8  988
077: 0100 D3E0 B900 02F7  6E5
078: E1A3 1303 E8BE 01A2  4E6
079: 1903 A103 023B 0603  089
07A: 0175 1006 1E07 BF5F  F61
07B: 04BE 1A03 E819 FE07  E2E
07C: EB3D 908B 1E03 02D1  B8C
07D: E3BE 0501 8B38 268A  94A
07E: 4501 3C3A 7512 268A  684
07F: 053C 6176 063C 7A77  44B
080: 022C 202C 41A2 1903  08D
081: BE1A 03E8 EAFD 8BD6  1AE
082: E8D5 FD8B D980 78FF  22C
083: 5C75 04C6 40FF 00A1  FFA
084: 1303 3D00 0075 1EB4  D1E
085: 368A 1619 03FE C2CD  CFD
086: 213D FFFF 750A BA6A  D95
087: 03B4 09CD 21EB 3E90  C6B
088: F7E1 A313 0333 D2BE  AB9
089: CF03 BB05 00E8 6801  787
08A: BABD 03B4 09CD 211E  613
08B: 07BF 1A03 E824 0083  2AD
08C: 3E17 0301 7617 A115  EB7
08D: 03F7 2613 03BE 4C04  C46
08E: BB0A 00E8 4201 BA3D  A9C
08F: 04B4 09CD 21B4 4CCD  A72
090: 2100 0050 5351 5255  59F
091: 0657 1E56 E803 0150  1B6
092: 2EC7 0681 0500 002E  DBB
093: FF06 1703 B448 BB09  BA8
094: 00CD 2150 5706 8BEC  A6E
095: 33DB E8EE 008E D8BE  B75
096: 3000 E843 FD0E 07BF  B1F
097: 6304 E8B7 00B4 4E8B  9F2
098: D6B9 0600 CD21 721D  76B
099: A11A 008B 161C 002E  4AF
09A: F736 1303 83FA 0074  170
09B: 0140 2E01 0681 05B4  DC2
09C: 4FCD 21EB E12E A181  C7A
09D: 052E 0106 1503 2EF7  9ED
09E: 2613 03BB 0A00 0E1F  7E9
09F: BED9 03E8 BA00 8B46  636
0A0: 008E C08B 7E02 BEED  6FD
0A1: 03E8 ECFC BB01 00BA  66F
0A2: D703 E897 008B 5604  332
0A3: 8EDA BA30 00B4 4EB9  227
0A4: 1000 CD21 7237 A015  E99
0A5: 0024 1074 2A80 3E1E  C45
0A6: 002E 7423 8B46 008E  9B8
0A7: C08B 7E02 BE30 00E8  7AC
0A8: B6FC 0E07 BF61 04E8  64D
0A9: 2A00 1E07 BF1E 00E8  4EC
0AA: 2200 8BFE E82C FFB4  5EF
0AB: 4FCD 21EB C758 5807  404
0AC: B449 CD21 5833 DBE8  338
0AD: 3100 5E1F 5F07 5D5A  21C
0AE: 595B 58C3 5152 568B  FA3
0AF: D6E8 6CFC 03F1 E877  EDA
0B0: FC5E 5A59 C353 50B4  C96
0B1: 19CD 215B 8AD8 8BC3  C54
0B2: 5BC3 06B4 2FCD 218C  B83
0B3: C007 C350 5352 1E8E  95B
0B4: D88B D3B4 1ACD 211F  87E
0B5: 5A5B 58C3 5051 E837  5F0
0B6: FCB4 40CD 2159 58C3  3E8
0B7: 5351 5756 5552 508B  0A3
0B8: EC03 F34E B90A 008B  F0E
0B9: 4602 2BD2 F7F1 8946  D44
0BA: 028B 4600 F7F1 8946  AEC
0BB: 0080 C230 8814 4E4B  88D
0BC: 837E 0000 75E1 837E  633
0BD: 0200 75DB 8BCB C604  544
0BE: 204E E2FA 585A 5D5E  531
0BF: 5F59 5BC3            D77
```

Le Journal JPC est le bulletin de liaison entre les membres de l'Association "PPC Paris", régie par la loi de 1901. Le Club est éditeur de JPC, et son siège social est au 56, rue Jean-Jacques Rousseau, 75001 Paris.

PPC Paris est le représentant Français de **HEX** (**H**andhelds **E**uropean Clubs EXchange), la fédération des principaux clubs Européens d'utilisateurs de calculateurs HP.

La maquette de ce numéro a été préparée et réalisée par Jacques Belin et Asdin Aoufi.

Les dessins sont de Jean-Jacques Dhénin et Paul Courbis.

Directeur de la publication : Jacques Belin
Numéro ISSN : 0762 - 381X

Veuillez adresser toute correspondance à :
PPC Paris, BP 604, 75028 Paris Cedex 01.

Imprimé par Paris Copie, 4 rue Linné, 75005 Paris