

csH-reference

COLLABORATORS

	TITLE : csH-reference		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		July 26, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ssh-reference	1
1.1	Documentation du C-Shell	1
1.2	Utilisateurs francophones	1
1.3	abortline	2
1.4	action	2
1.5	addbuffers	3
1.6	addpart	3
1.7	alias	3
1.8	ascii	4
1.9	assign	4
1.10	basename	5
1.11	cat	5
1.12	cd	5
1.13	chgrp	6
1.14	chmod	6
1.15	chown	7
1.16	class	7
1.17	close	8
1.18	cls	8
1.19	copy	8
1.20	cp	9
1.21	date	9
1.22	dec	9
1.23	delete	10
1.24	dir	10
1.25	diskchange	12
1.26	echo	12
1.27	else	12
1.28	endif	12
1.29	error	13

1.30	exec	13
1.31	fault	13
1.32	filenote	13
1.33	flist	14
1.34	ftlower	14
1.35	ftupper	14
1.36	foreach	14
1.37	forever	15
1.38	forline	15
1.39	fornum	15
1.40	getenv	16
1.41	goto	16
1.42	head	16
1.43	help	17
1.44	history	17
1.45	howmany	17
1.46	htype	18
1.47	if	18
1.48	inc	19
1.49	info	19
1.50	input	20
1.51	join	20
1.52	keymap	21
1.53	label	21
1.54	linecnt	21
1.55	ln	21
1.56	local	22
1.57	ls	22
1.58	makelink	22
1.59	man	22
1.60	md	23
1.61	mem	23
1.62	menu	23
1.63	mkdir	24
1.64	mv	24
1.65	open	24
1.66	path	25
1.67	pri	25
1.68	protect	26

1.69 ps	26
1.70 pwd	27
1.71 qsort	27
1.72 quit	27
1.73 rback	28
1.74 readfile	28
1.75 rehash	28
1.76 relabel	29
1.77 rename	30
1.78 resident	30
1.79 return	30
1.80 rm	31
1.81 rpn	31
1.82 run	31
1.83 rxrec	32
1.84 rxsend	32
1.85 search	33
1.86 set	34
1.87 setenv	34
1.88 sleep	34
1.89 split	34
1.90 stack	35
1.91 strhead	35
1.92 strings	35
1.93 strleft	36
1.94 strlen	36
1.95 strmid	36
1.96 strright	36
1.97 strtail	37
1.98 source	37
1.99 tackon	38
1.100tail	38
1.101tee	38
1.102touch	38
1.103truncate	39
1.104type	39
1.105unalias	39
1.106uniq	39
1.107unset	40

1.108usage	40
1.109version	40
1.110waitforport	40
1.111whereis	40
1.112window	41
1.113writefile	41
1.114commandes	41
1.115_abbrev	42
1.116_bground	42
1.117_clinumber	43
1.118_clipri	43
1.119_cquote	43
1.120_cwd	43
1.121_debug	43
1.122_dirformat	44
1.123_every	44
1.124_except	44
1.125_failat	44
1.126_hilite	44
1.127_history	45
1.128_insert	45
1.129_ioerr	45
1.130_kick2x	45
1.131_kick3x	45
1.132_lasterr	45
1.133_lcd	46
1.134_man	46
1.135_maxerr	46
1.136_minrows	46
1.137_nobreak	46
1.138_nomatch	47
1.139_noreq	47
1.140_passed	47
1.141_path	47
1.142_prghash	47
1.143_prompt	48
1.144_pipe	48
1.145_qcd	48
1.146_rback	49

1.147_rxpath	49
1.148_scroll	49
1.149_terminal	49
1.150_titlebar	49
1.151_verbose	50
1.152_version	50
1.153_variables	50
1.154_abbrev	50
1.155_abs	50
1.156_age	50
1.157_age_mins	51
1.158_appsuff	51
1.159_arg	51
1.160_ask	51
1.161_availmem	51
1.162_basename	51
1.163_center	51
1.164_checkpoint	52
1.165_clinum	52
1.166_complete	52
1.167_concat	52
1.168_confirm	52
1.169_console	52
1.170_dectohex	52
1.171_delword	53
1.172_delwords	53
1.173_dirname	53
1.174_dirs	53
1.175_dirstr	53
1.176_drive	53
1.177_drives	53
1.178_exists	54
1.179_fileblks	54
1.180_filedate	54
1.181_fileinfo	54
1.182_filelen	54
1.183_filenote	54
1.184_fileprot	54
1.185_filereq	55

1.186@files	55
1.187@filesize	55
1.188@flines	55
1.189@freebytes	55
1.190@freeblks	55
1.191@freestore	55
1.192@getenv	56
1.193@getclass	56
1.194@hextodec	56
1.195@howmany	56
1.196@index	56
1.197@info	56
1.198@intersect	56
1.199@ioerr	57
1.200@lookfor	57
1.201@lower	57
1.202@match	57
1.203@max	57
1.204@megs	57
1.205@member	57
1.206@min	58
1.207@mix	58
1.208@mounted	58
1.209@nameext	58
1.210@nameroot	58
1.211@opt	58
1.212@pathname	58
1.213@pickargs	59
1.214@pickopts	59
1.215@rnd	59
1.216@rpn	59
1.217@scrheight	59
1.218@scrwidth	59
1.219@sortargs	59
1.220@sortnum	60
1.221@split	60
1.222@strcmp	60
1.223@stricmp	60
1.224@strhead	60

1.225 @strleft	60
1.226 @strmid	60
1.227 @strright	61
1.228 @strtail	61
1.229 @subfile	61
1.230 @subwords	61
1.231 @tackon	61
1.232 @trim	61
1.233 @unique	61
1.234 @union	62
1.235 @upper	62
1.236 @volume	62
1.237 @wincols	62
1.238 @winheight	62
1.239 @winleft	62
1.240 @winrows	62
1.241 @wintop	63
1.242 @winwidth	63
1.243 @without	63
1.244 @word	63
1.245 @words	63
1.246 functions	63

Chapter 1

csh-reference

1.1 Documentation du C-Shell

REFERENCE C-SHELL 5.20+ , Copyright 1991-1993 aux auteurs.
Traduite par le BUGSS .

Commandes
Variables
Fonctions

1.2 Utilisateurs francophones

UTILISATEURS FRANCOPHONES

```
*****
*
*          TRADUCTION FRANCAISE PAR :          *
*
*          OBP ELECTRONIXS                      *
*          ~~~~~                                *
*          &                                    *
*
*          Gaël MARTINEZ                       *
*          ~~~~~                                *
*
*          POUR LE COMPTE DU:                   *
*
*          B.   U.   G.   S.   S.              *
*          ~~~~~                                *
*****
```

TOUS MES REMERCIEMENTS A MARCEL DURUFLE ET FREDERIC DELHOUME POUR LEUR AIDE,
ET A URBAN DOMINIK MUELLER POUR SA GENTILLESSE ET SON ACCORD A CETTE
TRADUCTION.

NOTE : Cette traduction se veut la plus fidèle que possible de l'original;
cependant, pour obtenir la documentation originale en anglais, contactez le

B.U.G.S.S.

POUR RECEVOIR UN CATALOGUE DES DISQUETTES DU DP DU BUGSS, ENVOYEZ-NOUS UNE ENVELOPPE TIMBREE A VOTRE ADRESSE.

Pour tous renseignements, questions, propositions, etc...

SI VOUS AVEZ TRADUIT DES DOCS DE PROGRAMMES DOMAINE PUBLIC ETRANGERS ET SOUHAITEZ LES VOIR DISTRIBUES,

Ou Si vous avez écrit un logiciel Domaine Public (ou Shareware) et souhaitez le voir distribué,

contactez le BUGSS auprès de :

BUGSS
25b, rue du prof. Lande
33380 BIGANOS FRANCE
56-82-70-01

Envoyez vos questions à rullier@platon.emi.u-bordeaux.fr
ou 2:324/8.1 2:324/8.3

© 1992 OBP ELECTRONIXS pour la traduction française de le version 5.19!
© 1993-94 MGC Productions pour le passage de la doc au format AMIGAGUIDE
et la mise à jour de la version 5.31 et +.

Supportez les auteurs de programmes Domaine Public ou Shareware et faites de l'Amiga la machine de vos rêves.

Et rappelez vous : SEUL L'AMIGA PEUT LE FAIRE !!!

1.3 abortline

ABORTLINE

Usage : abortline

Exemple : echo a;abort;echo b

Results : a

Oblige le reste de la ligne à être enlevé. Créée pour servir en conjonction avec la gestion d'exceptions.

1.4 action

ACTION

Usage : action [-a] nomaction fichier [arguments]

Exemple : action extr csh516.lzh csh.doc

Envoie une action à un fichier. Voir le chapitre XIV CLASSES

Options:

-a (abort) retourne 0 si ça a échoué et 1 si ça a marché. Sinon, les codes d'erreurs normaux (10 ou 11) sont retournés

1.5 addbuffers

ADDBUFFERS

Usage : addbuffers drive

addbuffers drive buffers [drive buffers ...]

Exemple : addbuffers df0: 24

Comme Addbuffer de AmigaDOS, associe de nouveaux buffers d' E/S au disk. Chaque buffer prend 512 octets de mémoire, mémoire CHIP si c'est un lecteur de disquettes. Les buffers peuvent être négatifs pour enlever des buffers d'un lecteur).

Pour voir le nombre actuel de buffers tapez "addbuffers drive" ou "addbuffers drive 0".

1.6 addpart

ADDPART (ou TACKON)

Equivalent à tackon.

1.7 alias

ALIAS

Usage : alias [name [command string]]

Usage : alias [nom [chaîne de la commande]]

Exemple : alias vt "echo Lance VT100;run sys:outils/vt100"

Définit un nom qui représente toute une chaîne. Vous pouvez faire un alias d'un jeu de commandes sur un seul mot si vous les mettez entre guillemets. En tapant vt, tout simplement, la ligne de commande ci dessus serait exécutée.

Les Alias pourraient s'appeler mutuellement, mais la récursivité directe est prohibée, ainsi ce qui suit marche: alias ls "ls -s"

Pour empêcher le remplacement par un alias, entrez: \ls

En tapant "alias nom", vous obtiendrez l'alias pour ce nom, alors qu'avec "alias" vous obtiendrez une liste de tous les alias.

ARGUMENT PASSES A UN ALIAS:

Usage : alias nom "%var[%var...] [chaîne de commande]"

alias nom "*var[%var...] [chaîne de commande]"

```
Exemple : alias xx "%q%w echo Salut $q, t'as l'air $w
          xx Steve en forme
Results : Salut Steve, t'as l'air en forme
```

La seconde forme de la commande alias permet de passer des arguments à n'importe quel endroit dans la chaîne de commande via l'utilisation d'un nom de variable. Pour passer des arguments à la fin de la chaîne de commande, cette méthode n'est en fait pas nécessaire. Ces variables sont locales, aussi elles ne détruisent pas une autre variable avec le même nom.

Si vous spécifiez des arguments multiples, on assigne à chaque argument un mot, et le dernier argument recevra le reste de la ligne de commande.

Utiliser un '*' au lieu du premier % empêche l'expansion des motifs:

```
alias zoo *a zoo $a
Pour étendre les motifs après que vous les ayez eus, utilisez
exec set a $a
```

ALIAS IMPLICITE:

```
Usage      : {commande;commande}
            {%var commande;commande} arg arg
Exemple : {%tmp echo $tmp $tmp} bonjour --> bonjour bonjour
```

Les accolades définissent des alias temporaires. Ils peuvent être redirigés comme un tout, peuvent avoir des arguments et des variables locales. Ils comptent comme un argument, même si il y a des blancs à l'intérieur (comme avec les guillemets), et peuvent être entremêlés. Des définitions d'alias complexes peuvent utiliser les accolades au lieu des guillemets, bien que ceci ajoute un surplus de travail. L'accolade fermante est optionnelle si elle est à la fin de la ligne. Exemple:

```
alias assertion {e "Etes vous sûr? ";input -s sur
```

1.8 ascii

ASCII

```
Usage      : ascii
            ascii chaîne
```

Si appelé sans arguments, ascii affiche une table ASCII complète. Si on lui donne une chaîne, montre chaque caractère en ASCII.

Options:

```
-h montre les nombres en hexadécimal
-o montre les nombres en octal
```

1.9 assign

ASSIGN

Usage : assign

assign logique

assign [-adlnp] logique1 physique1 [logique2 physique2 ...]

La première forme montre tous les assigns.

La seconde forme détruit un assign.

La troisième forme assigne logique1 à physique1 et ainsi de suite.

Options:

Options:

-a ajoute un nouveau chemin à un assign existant

-d crée un assign différé (lien-retard)

(identique à l'ancienne option -l)

-p crée un chemin (non-lié) assigné

(identique à l'ancienne option -n)

Pour une définition de différé/non-lié, référez vous à votre manuel AmigaDOS.

1.10 basename

BASENAME

Usage : basename var chemin [chemin ...]

Exemple : basename x df0:c/Dir # met "Dir" dans x

Met les noms de base des chemins dans la variable spécifiée.

1.11 cat

CAT (ou TYPE)

Usage : cat [-n][fichier fichier....]

Exemple : cat toto.txt

Affiche à l'écran les fichiers spécifiés. Si aucun fichier n'est indiqué, STDIN est utilisé (note: ^\ est EOF). CAT est fait pour afficher seulement les fichiers texte. L'option -n numérote toutes les lignes de l'affichage.

1.12 cd

CD

Usage : cd [chemin]

cd -g unité1 [unité2 [unité3 ...]]

Change votre répertoire de travail courant. Vous pouvez indiquer '..'

pour revenir au répertoire au-dessus (c'est une fonction spécifique de CD, qui ne marche pas avec les spécifications de chemin normales).

Dans la plupart des cas vous n'aurez plus besoin d'utiliser la commande CD. Tapez juste le répertoire désiré au prompt (très pratique en conjonction avec la complétion des noms de fichiers). Taper un ~ seul sur la ligne de commande fait un cd sur le répertoire courant précédent.

Il y a deux situations dans lesquelles vous en aurez encore besoin:

Entrer 'cd *tem' fera un cd dans le premier nom qui corresponde.

La seconde forme génère une liste (un fichier ASCII) de tous les répertoires dans les unités données. Elle sera stockée dans le fichier donné dans \$_qcd (par défaut: 'csh:csh-qcd'). Notez que ce fichier ASCII ne sera pas concaténé mais réécrit. Une fois que vous avez généré ce fichier, vous pouvez faire un cd vers n'importe quel répertoire sur votre disque dur même s'il n'est pas dans votre répertoire courant.

Si vous avez deux répertoires avec le même nom et que vous utilisez l'un plus que l'autre, déplacez le plus important au début de votre fichier qcd. Vous pourriez aussi trier le fichier.

C'est légal de ne taper qu'une abbréviation du répertoire dans lequel vous souhaitez faire un cd. Aucune astérisque n'est nécessaire. Si vous arrivez dans le mauvais répertoire, cd vers le même répertoire à nouveau (mieux vaut le faire avec curseur haut + RETURN). Vous allez cycler à travers tous les répertoires qui correspondaient à l'abréviation donnée. L'autre possibilité est de spécifier le nom complet du répertoire parent: cd devs/keym

Vous pourriez aussi ajouter des unités et des assigns, ainsi si 'Pagestream:' est une ligne dans le fichier qcd, un cd vers 'page' réussit.

CD sans aucun argument affiche le chemin du répertoire où vous êtes actuellement.

1.13 chgrp

CHGRP

Usage : chgrp group fichier1 ... fichierN

Exemple : chgrp 42 monfichier

règle l'id-groupe (0-65535) ou le nom-groupe des fichiers spécifiés. (pour l'instant, name-to-id mapping non implémenté)

1.14 chmod

CHMOD

Usage : protect [u|g|o|a][+|-|=][flags] fichier1 ... fichierN

Exemple : protect +rwe monfichier

Règle les bits de protection AMIGADOS du ou des fichiers indiqués. Ces bits d'attributs sont h, s, p, a, r, w, e, d. x est équivalent à e.

Propriétaire:

- u Règle le bit spécifié pour User (aka Owner)
- g Règle le bit spécifié pour Group (aka Owner)
- o Règle le bit spécifié pour Other (pas User, pas Group)
- a Tout, alias pour "ugo" (User/Group/Other)

Ne pas spécifier de propriétaire est équivalent à 'u'.

Modes:

- + Active les bits de protection spécifiés, laisse les autres
- Enlève les bits spécifiés, laisse tous les autres
- = Active les bits de protection, enlève les autres

Ne pas spécifier de mode est équivalent à '='.

Le bit d'archive est inactif par défaut.

Note: Cette commande est équivalente à "protect" excepté que les arguments pour le(s) nom(s) du (des) fichier(s) et les bits sont inversés.

1.15 chown

CHOWN

Usage : chgrp owner fichier1 ... fichierN

Exemple : chgrp 42 monfichier

règle l'id-owner(propriétaire) (0-65535) ou le nom-groupe des fichiers spécifiés. (pour l'instant, name-to-id mapping non implémenté)

1.16 class

CLASS

Usage : [-n] nom {type=param} ["actions" {action=commande}]

Exemple : class zoo offs=20,dca7c4fd ext=.zoo actions view="zoo l"

Définit une nouvelle classe de fichiers et les actions à faire dessus dans différents cas, ou montre les anciennes définitions si on ne lui donne pas d'arguments.

Cf section XIV: OBJETS

Options:

- n (new) oublie les anciennes définitions

1.17 close

CLOSE

Usage : close [numéroofichier]

Ferme le fichier indiqué ouvert par open. Sans numéro de fichier, ferme tous les fichiers ouverts. Voir open et flist pour plus d'info.

1.18 cls

CLS

Usage : cls

Ceci est un alias. Il efface juste l'écran, mais marche aussi sur un terminal (echo ^L ne le fait pas).

1.19 copy

COPY (ou CP)

Usage : copy [-udfpmo] fichier fichier
 ou : copy [-udfpmo] fichier1 fichier2...fichierN dir
 ou : copy [-rudfpo] dirl...dirN fichier1...fichierN dir

Options :

- r récursif, copie aussi tous les sous-répertoires.
- u met à jour, si une version plus récente existe en dest, ne fait pas la copie.
- f rafraîchir, si le fichier n'existe pas en dest ou est plus récent, ne pas copier.
- q supprime les messages 'not newer' et 'not there' dans -u et -f
- d ne met pas dans la date du fichier destination celle du fichier source.
- p ne met pas les bits de protection du fichier source dans la destination.
- m efface l'original. ne fonctionne pas avec -r
- o réécrit sur les protégés en écriture/effacement, lit les protégés en lecture
- a n'efface pas le bit d'archive

Exemple: copy -r df0: df1:

Copie les fichiers ou répertoires. Lors de la copie de répertoires, l'option -r doit être mise pour copier aussi les sous-répertoires. Autrement, seuls les fichiers du haut du répertoire source seront copiés.

Tous les fichiers sont affichés alors qu'ils sont copiés et les répertoires affichés lorsqu'ils sont créés. Cette sortie peut être supprimée en redirigeant vers nil: ex. copy -r >nil: df0: df1:

Copy s'arrête après le fichier en cours si on tape Control-C.

Copy met par défaut comme date du fichier destination celle du fichier source. Pour éviter ceci utiliser le switch -d .

De façon similaire, il met les bits de protection (flags) comme ceux du fichier source et tous les commentaires sont copiés.

Une autre option pratique est -u (update) qui n'effectue pas la copie d'un fichier déjà existant dans le répertoire destination si le fichier destination est plus récent ou égal au fichier source. C'est très pratique lorsqu'on programme, disons en ram:, ex: 'copy *.c ram:', et quand c'est fait vous tapez 'copy -u ram: df1:' et seuls les modules que vous avez modifiés seront recopiés.

La commande Copy crée maintenant le répertoire destination s'il n'existe pas lorsqu'il est spécifié par 'copy [-r] répertoire répertoire'. Si vous mettez 'copy fichier fichier fichier rep', alors 'rep' doit déjà exister .

1.20 cp

CP (ou COPY)

Equivalent à copy.

1.21 date

DATE

Usage : date [-bsr] [new date and/or time]

Exemple : date Wednesday # this refers to NEXT wed, of course

Options:

- b affiche date/heure de l'horloge sur batterie si elle existe.
- s stocke l'heure courante
- r affiche l'heure relative à la dernière stockée en secondes et centièmes

Utilisé pour lire ou mettre la date/heure système à jour. Toutes les options standard peuvent être utilisées (yesterday, tomorrow, monday, etc.).

Les zéros au devant ne sont pas nécessaires.

Sans paramètres affichera JJ-MMM-AA HH:MM:SS.

1.22 dec

DEC

Usage : dec nomvar [valeur]

Exemple : dec abc

Décrémente l'équivalent numérique de la variable de la valeur spécifiée (défaut: 1) et place le résultat en chaîne-ASCII dans cette variable.

1.23 delete

DELETE (ou RM)

Usage : delete [-fpqrv] fichier fichier fichier...

Exemple : delete toto.txt test.c

Efface (delete) les fichiers indiqués. Delete retourne toujours le code d'erreur 0. Vous pouvez enlever des répertoires vides.

Options:

- r effacera récursivement les répertoires non vides.
- p (ou f); efface les fichiers protégés contre l'effacement.
- v alterne la sortie verbose. Utile si 'delete' est 'aliasée'.
- q (quitte), delete avorte si le fichier à éliminer n'existe pas ou ne peut être effacé. Ceci n' affecte pas les motifs non-correspondant.

Si vous spécifiez aucun motif d'effacement les fichiers seront listés comme effacés. Ceci peut être éliminé en redirigeant vers nil:

1.24 dir

DIR (ou LS)

Usage : dir [-abcdfhiklnogstuv] [-z [lformat]] [chemin chemin ...]

Exemple : dir -ts df0:

dir -lz "%7s %-16n %m" *.c

Options :

- d affiche seulement les répertoires
- f affiche seulement les fichiers
- h affiche seulement les fichiers qui ne commencent pas par un point, ne finissent pas par '.info' ou n'ont pas le bit h mis. Ajoute un bit 'i' aux flags qui signale si un fichier .info correspondant existe.
- s affichage court en multi(4) colonne.
- c ne change pas la couleur pour les sous-répertoires.
- q affichage réduit: n'affiche pas la longueur en blocs.
- o affiche les commentaires sur les fichiers.
- n affiche seulement les noms.
- p affiche les noms de fichiers complets et supprime les noms de répertoires

- a affiche l'age de chaque fichier dans le format
jours heures:minutes
- i identifie chaque fichier, affiche le type au lieu de la date.
Cf chapitre XIV CLASSES
- v (viewdir) additionne récursivement la longueur des fichiers dans
un répertoire.
- l classe les fichiers par longueur, le plus long en premier.
- t classe les fichiers par la date, le plus récent d'abord.
- k trie les fichiers par leur classe
- b trie les fichiers à l'envers.
- g affiche les répertoires au début
- e affiche les répertoires à la fin
- u on doit lui donner exactement deux répertoires. Affiche les
fichiers seulement dans le premier répertoire, les fichiers dans
les deux, et les fichiers dans le second.
- z format customisé

Affiche un répertoire des fichiers indiqués. L'affichage par défaut montre les date, protection, taille en blocks, taille en octets et l'espace total occupé. Les flags de protection incluent les nouveaux flags 1.2/1.3 (voir à protect), plus un flag 'c' qui indique si le fichier a un commentaire. Les fichiers sont rangés dans l'ordre alphabétique, sans tenir compte des majuscules, et les répertoires sont en couleur rouge (sauf si vous mettez -c). Dir prend en compte la largeur de votre fenêtre.

La chaîne lformat est utilisée pour créer votre propre format de répertoire. A la place de la commande lformat vous pouvez régler la variable "_dirformat" (et éliminer l'argument de lformat). Cela remplace tout le reste et peut contenir les codes suivants:

%a age	%p nom avec chemin
%b taille en blocs	%q nom avec slash
%c flag c (commentaires)	%r taille en anglais
%d date	%s taille
%e flag i (.info)	%+ flag i comme '+' ou ' '
%f flags hspawed	%t heure
%i flag d (dIr)	%u taille en Ko
%k class	%v taille du répertoire en anglais
%l LF dans les commentaires	%w taille du répertoire en Ko
%m multi colonnes	%x date traduite
%n nom	
%o filenote (commentaire)	

%I information sur les liens (S: lien soft, H: lien hard, P: pipe, -: autre)
 %L nom du fichier original si il existe un lien (empty autrement)
 %N nom + nom original (du lien)
 %F bits protection (flags "rwed") pour group/other
 %U user-id
 %G group-id

Entre le '%' et la lettre d'identification, il pourrait y avoir un champ de largeur optionnel. Si le nombre est précédé par un '-', le contenu du champ sera ajusté à gauche. Si à côté d'un point, le contenu sera raccourci pour convenir à la largeur du champ s'il est trop large.

Si la chaîne de format contient un %m, cshell essaiera d'imprimer

plus d'une entrée par ligne. La largeur en colonnes est la largeur du champ de l'entrée %m. S'il est omis, on prend la largeur du premier fichier. Si un fichier est plus long, il utilisera deux colonnes.

Si vous préférez l'ancien style d'affichage de cette commande (5.19 et avant) ajoutez la ligne suivante dans votre fichier .cshrc:

```
set _dirformat "    %-24n %c%f %7s %4b %d %t"
```

1.25 diskchange

DISKCHANGE

Usage : diskchange drive

Comme DiskChange de AmigaDOS. Multiples noms de lecteurs acceptés.

1.26 echo

ECHO

Usage : echo [-en] chaîne
Exemple : echo salut tout le monde
Resultat : salut tout le monde

Affiche la chaîne donnée. Sans l'option -n un retour chariot est ajouté. Si -e est mis, affiche sur stderr.

1.27 else

ELSE

Usage : else ; commande
Usage : if -f toto.c ; else ; echo "Pas ici" ; endif

La clause else, doit suivre un test par IF.

1.28 endif

ENDIF

Usage : endif

La fin de la commande if.

Note: si vous revenez d'un fichier script avec des IF non terminés, et que le dernier IF était faux, le prompt sera remplacé par un

sousigné '_' et aucune commande ne sera exécutée tant que 'endif' ne sera pas tapé.

1.29 error

ERROR

Usage : error n

Génère un code de retour n.

1.30 exec

EXEC

Usage : exec [-i] commande [args]

Exemple : set lignecmd "dir ram:"

exec \$lignecmd # ne marcherait pas sans exec

Options:

-i retourne le code 0.

Exécute la commande indiquée; exec commande est équivalent à commande sauf que vous pouvez utiliser des variables pour spécifier le nom de la commande.

Notez que la ligne de commande est interprétée deux fois! Exemples:

set a dir ram;; exec \$a # juste

set a mkdir; exec \$a "Mon répertoire" # faux! crée deux répertoires

Exec retourne le code de retour de la commande exécutée à moins que l'option -i (ignore) ne soit mise, auquel cas 0 est toujours retourné.

1.31 fault

FAULT

Usage : fault erreur1 .. erreurN

Exemple : fault 205 212

Comme le 'fault' AmigaDOS, imprime les messages d'erreurs spécifiés.

1.32 filenote

FILENOTE

Usage : FILENOTE fichier1 .. fichierN note

FILENOTE -s fichier1...fichierN

Options:

-s (seconde forme); affiche les notes de fichiers des fichiers spécifiés.

Dans la première forme, met un commentaire AmigaDOS au fichier spécifié. La seconde forme affiche les commentaires des fichiers.

1.33 flist

FLIST

Usage : flist

Liste les numéros de fichiers ouverts par open.
Voir open et close pour plus d'info.

1.34 fltlower

FLTLOWER

Usage : fltlower

Exemple : dir | fltlower

Ou : fltlower <readme

C'est une commande filtre; i.e. elle lit dans stdin et écrit dans stdout. La manière la plus naturelle de l'utiliser est un pipe, ou elle peut être redirigée.

Son propos est de convertir tous les caractères alphabétiques en minuscules.

1.35 fltupper

FLTUPPER

Identique à fltlower, mais elle convertie en majuscules.

1.36 foreach

FOREACH

Usage : foreach [-v] varnom (chaînes) commande

Exemple : foreach i (a b c d) "echo -n \$i;echo \" ha\""

Resultat: a ha

b ha

c ha

d ha

Options:

-v affiche les arguments à chaque fois que la commande est exécutée.

'chaînes' est réparti en arguments. Chaque argument est placé à son tour dans une variable locale 'varnom' et 'commande' est exécuté. Placez les commandes entre guillemets.

Foreach est très pratique pour interpréter des arguments passés dans un alias.

ex.

```
foreach i ( *.pic ) viewilbm $i
en supposant que a.pic et b.pic sont dans le répertoire actuel, les
commandes suivantes seront exécutées:
viewilbm a.pic
viewilbm b.pic
```

Toutes les commandes 'for...' peuvent être interrompues avec CTRL-D ou CTRL-E.

1.37 forever

FOREVER

Usage : forever commande
ou : forever "commande;commande;commande..."

Les commandes indiquées sont exécutées encore et encore sans fin.

L'exécution est arrêtée si vous tapez ^C ou ^D, ou si les commandes retournent un code d'erreur.

1.38 forline

FORLINE

Usage : forline var nomfichier commande
ou : forline var nomfichier "commande;commande..."
Exemple : forline i RAM:temp "echo line \$_linenum=\$i"

Pour chaque ligne ASCII du fichier, les commandes indiquées sont exécutées et var pointe sur le contenu de la ligne.

Vous pouvez tester la variable '_linenum' pour trouver le numéro de la ligne actuellement lue.

Si STDIN (en majuscules) est spécifié comme fichier d'entrée, les lignes sont lues sur l'entrée standard.

1.39 fornum

FORNUM

Usage : fornum [-v] var n1 n2 commande
ou : fornum [-v] -s var n1 n2 pas commande

Exemple : fornum -v x 1 10 echo \$x
ou : fornum -s x 10 1 -1 echo \$x # compte à rebours

Exécute les commandes pour toutes les valeurs numériques de x entre n1 et n2. Si plus d'une commande est indiquée, ou si elle est redirigée, mettez les commandes entre guillemets.

Options:

- v (verbose) imprime les nombres progressivement.
- s vous permet de spécifier un pas; si c'est négatif, on compte à rebours.

1.40 getenv

GETENV

Usage : getenv [varshell] varenv

Prend la valeur d'une variable ENV: et la stocke dans la variable shell 'varshell'. Si varshell est omise, la valeur de la variable ENV: est affichée à l'écran.

Cette commande est obsolète puisque les variables ENV: peuvent être retrouvées par \$varenv n'importe où dans la ligne de commande.

1.41 goto

GOTO

Usage : goto étiquette

Exemple :

```
label debut
echo "Au début"
dir ram:
goto debut
```

Va au nom de l'étiquette (label) indiquée. Vous ne pouvez utiliser cette commande que dans un fichier script. Les étiquettes peuvent être avant ou après la position courante. Il est autorisé de sauter au dehors des if.

1.42 head

HEAD

Usage : head nomfichier [num]
Exemple : head readme 20

Affiche les "num" premières lignes de "nomfichier". Si num n'est pas spécifié, on prend 10 par défaut. Si le nom du fichier n'est pas spécifié l'entrée standard (stdin) est utilisée à la place.

1.43 help

HELP

Usage : help [-f]
Exemple : help
Options :
-f liste aussi les Fonctions

Affiche simplement toutes les commandes disponibles. Les commandes sont affichées dans l'ordre de recherche. Donc si vous donnez un nom partiel, la 1ère commande qui correspond à ce nom dans la liste sera exécutée. Généralement, vous devrez spécifier suffisamment du nom d'une commande pour que ce soit complètement unique.

1.44 history

HISTORY

Usage : history [-nr] [chaîne_partielle]
Exemple : history
Options :
-n ne numérote pas les lignes
-r lit l'historique depuis stdin

Affiche l'historique des actions passées dans le Shell. La taille de la liste est contrôlée par la variable _history. Si vous spécifiez une chaîne partielle, seules les actions correspondant à cette chaîne seront affichées.

1.45 howmany

HOWMANY

Usage : howmany

Cette commande vous dit combien de Shells fonctionnent dans votre système.

1.46 htype

HTYPE

Usage : htype [-r] [fichier1 .. fichiern]

Options:

-r affiche tous les fichiers d'un répertoire.

Affiche les fichiers nommés en hex et ASCII, comme la commande système 'Type fichier opt h'. Très intéressant pour les fichiers binaires.

Si il n'y a pas de nom de fichiers spécifiés, l'entrée standard est utilisée, alors vous pouvez utiliser htype comme destination pour un pipe.

1.47 if

IF

Usage : if [-n] argument conditionnel argument [then]
 ou : if [-n] argument
 ou : if [-n] -f fichier ou -e fichier
 ou : if [-n] -d fichier/répertoire
 ou : if [-n] -m
 ou : if [-n] -t fichier fichier1 .. fichierN
 ou : if [-n] -r rpnextpression
 ou : if [-n] -v nomvar
 ou : if [-n] -o car arg ... arg

Options:

-n (NON) inverse le résultat.
 -d teste le type de l'objet spécifié: si c'est un répertoire, alors TRUE; si c'est un fichier (ou qu'il n'existe pas) alors FALSE.
 -f (ou -e) vérifie l'existence du fichier spécifié.
 -m teste si de la mémoire FAST est présente.
 -o teste l'option 'car' dans le reste des arguments.
 -r évalue une expression RPN donnée (voir RPN pour plus d'informations). si la valeur sur le sommet de la pile est 0, alors FAUX, sinon VRAI.
 -t compare la date et l'heure du premier fichier avec tous les autres; si le premier est plus jeune que TOUS les autres, alors FALSE, aussi non TRUE. Si un fichier n'existe pas, il est considéré comme plus ancien.
 -v teste si une variable donnée est définie.

Rend les instructions suivantes, et jusqu'au prochain endif conditionnelles. Le 'then' est optionnel. La clause if doit être suivie par un point-virgule si les instructions suivent sur la même ligne.

Si (if) un seul argument est quelque chose pour un autre argument. Les clauses conditionnelles suivantes sont permises:

<, >, =, ! et combinaisons. Ainsi != est non-égal, >= plus grand ou égal , etc...

Si les arguments sont non-numériques, ils sont testés comme chaînes.

D'habitude c'est soit une constante soit une variable (\$nomvar).

La 2eme forme de IF traite de l'existence de l'argument.
Si l'argument est une chaîne "", alors FAUX, sinon VRAI.

La 3eme forme de IF avec l'option -f traite de l'existence du fichier indiqué. -e est la même chose que -f

L'option -d teste le type de l'object indiqué: si c'est un répertoire, alors VRAI; si c'est un fichier (ou inexistant) alors FAUX.

L'option -m teste si la mémoire FAST est présente.
Exemple (qui doit être mis dans un fichier login.sh):
if -m; resident -d lc1 lc2 blink; endif

Utiliser -t compare la date et l'heure du 1er fichier avec toutes les autres; si le 1er est plus jeune que TOUS les autres, alors FAUX, sinon VRAI. Si un fichier n'existe pas, il est considéré comme étant plus vieux.
Cette option est très pratique pour faire des makefiles sans aucune utilisation de MAKE.

Exemple:
if -t test.o test.asm test.i ; asm -o test.o test.asm ; endif

L'option -o teste l'option 'car' dans le reste des arguments.

Exemple: if -o r -rp ram:comm1.c signifiera TRUE.

Quand vous utilisez la commande 'IF' de façon interactive, et si vous entrez des commandes après un 'IF' donnant FAUX, le prompt sera remplacé par un souligné '_' indiquant que toutes les commandes seront ignorées jusqu'à ce qu'un 'ELSE' ou un 'ENDIF' soit rencontré.

1.48 inc

INC

Usage : inc variable [valeur]
Exemple : inc abc 5

Incrémente l'équivalent numérique de la variable de la valeur indiquée (par défaut: 1) et replace le résultat chaîne-ASCII dans cette variable.

1.49 info

INFO

Usage : info [-pt] [chemin1 chemin2 ... cheminN]

Options:

- p affiche seulement les lecteurs avec des disques lisibles (présents)
- t affiche les types et octets utilisés par les disques/fs au lieu des tailles en blocs.

Si elle est appelée sans arguments, info affiche des statistiques sur toutes les unités de type disque dans le système. Si un ou plusieurs chemins sont indiqués, seule l'information concernant ces unités sera affichée.

Note: CShell arrondi correctement toutes les valeurs affichées alors que le Info de Commodore ne le fait pas. Les valeurs peuvent légèrement être différentes.

1.50 input

INPUT

Usage : input [-sr] var var ... var

Exemple : input abc

Options :

- s la ligne entière est lue comme un seul mot, en incluant les espaces.
- r met la console sur un mode caractère par caractère avant de lire (i.e. n'attend pas que RETURN soit appuyé). Utilisez ceci avec précaution.

Envoie l'entrée sur STDIN (ou une redirection, ou un pipe) vers une variable. La ligne suivante de la saisie est séparée en mots (à moins qu'elle ne soit entre guillemets) et placée dans la variable.

1.51 join

JOIN

Usage : join [-r] fichier1..fichierN fichierdest

Exemple : join part1 part2 part3 total

Options :

- r écrit dessus tout fichierdest existant.

Join (concatène) les fichiers indiqués pour obtenir fichierdest. Si le fichier de destination existe déjà, un message d'erreur est envoyé et l'opération est stoppée, à moins que vous ne mettiez l'option -r (remplacer).

1.52 keymap

KEYMAP

Usage : keymap [number {touche=fonction}]

Exemple : keymap 0 1030=4 1032=12

Définit un clavier pour l'édition de la ligne de commande de ssh.
Cf chapitre XV.

1.53 label

LABEL

Usage : label nom

Crée une étiquette du programme à cet endroit. Utilisé dans les fichiers scripts, permet de faire un GOTO vers l'étiquette.

1.54 linecnt

LINECNT

Un autre filtre. Compte le nombre de lignes sur son entrée standard et écrit sur stdout.

1.55 ln

LN (ou MAKELINK)

Usage : ln [-s] nomfichier [nom du lien]

Exemple : ln stuff/données nouveaunom

Option :

-s crée un lien soft (par défaut lien hard)

ln crée une entrée répertoire appelée un lien, vers un fichier ou un répertoire. Un grand nombre de liens peuvent être assigné à un fichier.

nomfichier est le nom du fichier/répertoire original.

nomlien est le nouveau nom à associer avec le fichier ou nomfichier

Si nomlien est omis, le dernier constituant du nomfichier est utilisé comme nom du lien.

Un lien hard (par défaut) est une entrée répertoire comme celle faite lors de la création d'un fichier. Les liens hard peuvent uniquement être créés avec des fichiers existants. Les liens hard ne

peuvent être faits avec des fichiers systèmes (partitions de disques, fichiers systèmes montés).

Pour effacer un fichier, tous ses liens hard doivent être enlevés, y compris le nom par lequel il a été originellement créé; enlever le dernier lien hard libèrera le noeud associé avec le fichier.

Un lien symbolique, créé avec l'option `-s`, est une entrée répertoire spéciale qui pointe sur un autre nom de fichier. Les liens symboliques peuvent couvrir les fichiers systèmes et pointer sur des répertoires. En effet, vous pouvez créer un lien symbolique qui pointe sur un fichier qui actuellement est absent des fichiers du système; effacer le fichier sur lequel pointe le lien symbolique ne l'affecte et ne l'altère pas lui-même.

NOTE: les liens symboliques (aussi connus comme "liens softs") NE SONT PAS ACTUELLEMENT SUPPORTES par l'AmigaOS. NE LES UTILISEZ PAS !!

1.56 local

LOCAL

Usage: `local [var...var]`

Crée une ou plusieurs variables locales. Ces variables disparaissent à la fin de leur alias ou de leur fichier script, et ne peuvent pas être utilisées de l'intérieur d'autres alias ou fichiers scripts. Sans arguments, affiche toutes les variables globales et leurs valeurs.

1.57 ls

LS (ou DIR)

Equivalent à `dir`.

1.58 makelink

MAKELINK

Equivalent à `ln`.

1.59 man

MAN

Usage : `man commande(s)`

Exemple : `man mkdir`

Prend des infos sur une commande shell, ou d'autres mots clés. Ceci inclut toutes les `_variables` spéciales, plus divers mots-clés: COMPLETION, PIPES, EDITION, OPTIONS et plus.

Voyez la liste spéciale d'alias du manuel pour avoir une liste de tous les mots clés supportés par man.

Vous devez faire pointer `_man` sur le chemin de vos fichiers .doc:

```
set _man dh1:docs/aliases.doc dh1:docs/csh.doc
```

Pour créer vos propres fichiers.doc, précédez tous vos mots-clés par quatre blancs. 'man' affichera alors les lignes jusqu'à ce que le premier caractère d'une ligne soit alphanumérique ou qu'on ait quatre espaces en premier.

1.60 md

MD (ou MKDIR)

Equivalent à mkdir.

1.61 mem

MEM

Usage : mem [-cfqsl]

Options:

- c affiche seulement la mémoire chip disponible
- f affiche seulement la mémoire fast disponible
- q sort juste un nombre sans les titres
- s enregistre la mémoire libre actuellement
- r affiche la mémoire utilisée par rapport au dernier enregistrement
- l libère toute la mémoire non utilisée

1.62 menu

MENU

Usage : menu [-mn] [titre objet...objet]

Exemple : menu Shell JrComm,,j Rename,"rename ",r quit

Options:

- n efface tous les menus existants.
- m utilise une fonte monospace.

Ajoute un menu à la fenêtre de console courante. Jusqu'à 31 menus de 63 objets chacun (incluant le titre) peuvent être installés.

Si l'objet est juste une chaîne, cette chaîne sera dans l'entrée du

menu. Quand vous le sélectionnez, il sera mis au prompt et exécuté.

S'il y a une virgule et après cette virgule une seconde chaîne, ce sera la commande qui sera insérée au prompt. Cette fois, vous aurez à ajouter le ^M par vous-même si vous voulez que la commande soit exécutée.

S'il y a une seconde virgule , la lettre après cette virgule sera le raccourci clavier de cet objet de menu. (Il sera un jour sensible aux majuscules, utilisez des minuscules).

Si pour une raison quelconque votre menu courant est endommagé, entrez juste une commande 'menu' vide.

Quand le premier menu est installé vous pouvez utiliser l'option -m pour choisir une fonte monospace (Fonte du système par défaut) au lieu de la fonte Intuition par défaut (qui peut être une fonte proportionnelle). C'est utile pour les menus utilisateurs-formatés (comme dans le script exemple "menu.sh").

1.63 mkdir

MKDIR (ou MD)

Usage : mkdir [-p] nom nom nom...

Exemple : mkdir df0:stuff

Options:

-p créera tous les répertoires du chemin si nécessaire.

Crée les répertoires spécifiés.

Si "nom" fini par un / il sera enlevé.

mkdir supporte maintenant l'option -p. mkdir -p suivie par un nom de chemin entier créera tous les répertoires nécessaires pour créer le chemin. Par exemple "mkdir -p ram:foo/bar/tst/a" créera ram:foo/bar, ram:foo/bar/tst, et ram:foo/bar/tst/a en une seule fois.

De plus, cela ne renverra pas de codes d'erreurs pour des répertoires qu'il ne pourrait créer.

1.64 mv

MV (ou RENAME)

Equivalent à rename.

1.65 open

OPEN

Usage : open nomfichier modéfichier numérofichier

Exemple : open RAM:data w 1

Ceci vous permet d'ouvrir un fichier, rediriger vers ce fichier comme autant de commandes que vous voulez, puis le fermer. nomfichier est n'importe quel nom de fichier AmigaDOS valide, modéfichier est soit "r" pour read (lecture) ou "w" pour write (écriture), numérofichier est un nombre entre 1 et 10. Pour rediriger un programme vers ou depuis un fichier ouvert, utilisez comme nom de fichier de redirection un point suivi du numéro de fichier. Voici un exemple complet :

```
open RAM:data w 1
echo -n 2+2= >.1
rpn 2 2 + . CR >.1
close 1
type RAM:data # affichera 2+2=4
```

Voir aussi close, flist.

1.66 path

PATH

Usage : path [-gr] [dir...dir]

Sans argument, liste les chemins (path) AmigaDOS. Sinon ajoute au path les répertoires donnés, en empêchant les entrées doubles.

Options:

- r Réinitialise le path
- g modifications Globales de chemins; les opérations (ajout, élimination) s'appliquent à tous les processus CLI à la place du seul CLI courant.

Note:

Ce n'est pas parfaitement "légal" de modifier la liste de chemins des autres processus. L'ajout d'entrées (option -g) fonctionne correctement dans la plupart des cas. Mais le retrait d'entrées (option -gr ensemble) peut crasher le système parce que CSH ne sait rien sur la gestion mémoire des autres processus. (il ne sait pas comment ils ont alloué la mémoire pour la liste des chemins d'entrées).

Alors utilisez l'option -g avec précaution (évitez la avec -r).

Si ça marche c'est parfait ! Sinon, vous avez perdu ;-) Il n'y a pas d'efficacité 100 % pour les modifications de chemin globales.

1.67 pri

PRI

Usage : pri numero-cli pri

Exemple : pri 3 5 # met la priorité du cli n\textdegree{}3 à 5

Change la priorité de la tâche indiquée (utilisez la commande PS pour déterminer le numero-cli). Si vous indiquez 0 comme numero-cli vous changez la priorité de la tâche actuelle (celle exécutant le shell).

1.68 protect

PROTECT

Usage : protect fichier1 ... fichierN [u|g|o|a] [+|-|=][flags]

Exemple : protect monfichier u+rwe

Met les flags de protection AMIGADOS pour le fichier indiqué. Les flags utilisés sont h, s, p, a, r, w, e, d. (x est le même que e).

Propriétaire:

- u Met les bits spécifiés pour User
- g Met les bits spécifiés pour Groupe
- o Met les bits spécifiés pour Other(Autres) (ni User, ni Groupe)
- a tout, alias pour "ugo" (User/Groupe/Other)

Spécifier aucun propriétaire est équivalent à 'u'.

Modes:

- + Met les bits spécifiés, laisse les autres
- Annule les bits spécifiés, laisse les autres
- = Met les bits spécifiés, en effaçant tous les autres

Si vous ne spécifiez pas le mode, on prend '=' par défaut.

Le bit archive est effacé par défaut.

Note: Cette commande est équivalente à "chmod" excepté que les arguments pour le(s) nom(s) de fichier et le(s) flag(s) sont inversés.

1.69 ps

PS

Usage : ps [-les] [nomcommande...nomcommande]

Options:

- l affiche les noms de chemins complets des commandes
- e exclut les noms de commandes donnés dans la liste
- s ne montre pas la taille des piles et le type, utilise l'ancien format d'affichage.

Indique l'état des processus CLI. ex:

Proc	Command Name	CLI Type	Pri.	Address	Directory
* 1	ssh	Initial CLI	0	97b0	Stuff:shell
2	clock	Background	-10	2101a8	Workdisk:
3	emacs	Background	0	212f58	Stuff:shell
4	VT100	Background	0	227328	Workdisk:

Address est l'adresse de la tâche, directory est le répertoire actuel de la tâche. Par défaut, seuls les Noms de Base de la commande sont affichés. Votre propre CLI sera marqué par un astérisque dans la première colonne.

La taille de la pile est la taille réelle de la pile de commande. Ce n'est pas la taille qu'un programme prend lors de son lancement. Utilisez plutôt "status" si vous voulez la taille de la Pile par Défaut. (Soyez prudents: "Status" ne montrera pas la pile utilisée par un programme!).

Cli type sont deux lettres. La première est soit "f" (foreground, 'premier plan') ou "b" (background, 'arrière plan'). La seconde est une suivante :

- i: invalide
- a: ajouté
- r: tournant / prêt à tourner
- w: en attente
- e: exception
- d: enlevé

1.70 pwd

PWD

Usage : pwd

Reconstruit `_cwd` à reculons à partir de votre répertoire actuel.

1.71 qsort

QSORT

Usage : qsort [-cr] <in >out

Options :

- c case-sensitive (sensible aux majuscules)
- r fait trier en sens inverse.

Trie rapidement depuis l'entrée standard vers la sortie standard.

1.72 quit

QUIT

Usage : quit

Quitte le Shell pour revenir au CLI.

1.73 rback

RBACK

Usage : rback command

Démarre un nouveau processus exécutant la commande indiquée, mais ne peut faire d'entrée/sortie. Equivalent à 'run command >NIL: <NIL:'. Au lieu d'utiliser rback, vous pouvez ajouter un '&' à la fin de la ligne de commande.

Note: rback ne peut pas lancer les commandes encapsulées. Vous devez lancer un sous-shell:

```
rback ssh -c "copy ram:temp prt;;rm
ram:temp
```

1.74 readfile

READFILE

Usage : readfile nomvar [nomfichier]

Lit complètement un fichier ASCII et l'assigne à une variable. Chaque ligne devient un mot dans la chaîne résultante. Des blancs intercalés ne posent aucun problème. Si le nom de fichier est omis, stdin est utilisé. Voir aussi 'writefile', @subfile et @flines

1.75 rehash

REHASH

Usage : rehash [-cglos]

Options :

- c efface la hash list locale des programmes
- g efface la hash list des programmes globale
- l charge la hash list de programmes globale dans le buffer local
- o sort(output) la hash list de programmes locale
- s sauve la hash list locale de programmes vers le disque

Scanne le chemin de recherche du DOS entier (voir aussi 'path') et construit un hash list des programmes. Ceci peut être utilisé pour la

complétion des noms de programmes de la ligne de commande (par défaut: ESC-p ESC-P). Et alors les commandes de CShell ne scanneront plus les chemins DOS pour les commandes mais la hash list de programmes à chaque commande tapée mais ira scanner la liste en mémoire d'où une limitation des accès disques et donc une accélération significative.

Bien sur dès que vous ajouterez (ou effacerez) un répertoire ou un programme dans les chemins courants vous devrez reconstruire la hash list.

Chaque invocation de CShell a son propre buffer local pour gérer sa hash list de programmes. Utilisez l'option -s pour sauver cette hash list locale vers le disque (csh:csh-prgs). Avec l'option -l la liste est lue en mémoire. Le premier CShell lit la liste et en mets une copie dans un buffer global alors la prochaine fois qu'un CShell voudra lire la hash list, il le fera dans ce buffer et non sur le disque.

La liste globale reste en mémoire -- même si vous quittez tous les CShell. Utilisez l'option -g pour libérer cette liste globale (si vous êtes court en mémoire) mais ceci n'affectera pas les listes locales des CShell tournant déjà. L'option -c efface la liste locale.

Si vous lancez cette commande sans options la liste scannée n'est pas seulement mise dans le buffer local mais aussi dans le buffer global.

Il n'est pas nécessaire d'effacer la liste locale/globale avant d'en lire ou d'en construire une nouvelle. Ce sera fait automatiquement.

La variable `$_prghash` (par défaut: 'csh:csh-prgs') contient le nom du fichier d'où la hash list sera lue ou sauvée.

Les programmes de la hash list sont insensibles aux majuscules et peuvent être abrégés. Ceci peut être changé en modifiant la variable `"_abbrev"`.

Usage recommandé:

Premièrement lancez "rehash" de votre shell pour construire la hash list de programmes. Sauvez la avec "rehash -s". Maintenant incluez "rehash -l" dans "s:.cshrc" pour lire cette liste à chaque lancement de CShell.

1.76 relabel

RELABEL

Usage : relabel drive nom

Exemple : relabel DH0: Picard

Change le nom de volume du disque dans le drive donné en nom spécifié. Les noms initiaux sont créés initialement quand vous formater le disque.

Si vous avez un système avec un seul lecteur de disque, soyez sûres de

spécifier le disque par son nom de volume au lieu du nom du drive.

1.77 rename

RENAME (ou MV)

Usage : rename [-fv] depuis vers
ou : rename [-fv] depuis depuis depuis ... depuis versdir

Options :
-f ne s'interrompt pas lors d'erreurs
-v mode verbose (affiche les fichiers renommés)

Permet de renommer un fichier, ou de le déplacer à l'intérieur d'un disque. Permet aussi de déplacer 1 fichier ou plus dans un simple répertoire. Le bit archive du (des) fichier(s) sera effacé.

1.78 resident

RESIDENT

Usage : resident [-dr] [fichiers]
Exemple : resident lc1 lc2 blink # les charge comme résidents
resident -d lc1 lc2 blink # ne charge que quand nécessaire.
resident -r lc1 lc2 blink # les enlève
resident # liste des programmes residents

Options :
-d lecture différée;
-r enlève le fichier de la liste des résidents

Ceci est le resident DOS. Les commandes sont cherchées par Shell dans la liste des resident AVANT de chercher dans une quelconque unité externe. Seuls les programmes PURE peuvent tourner comme residents, voir les docs DOS pour plus d'infos. L'Option -d est très pratique: vous pouvez dire dans votre fichier de lancement: resident -d fichier..fichier; les programmes ne seront pas chargés immédiatement, mais seulement quand vous essayerez de les charger. De cette manière, vous ne perdrez pas de temps, ni de mémoire si vous ne vous servez pas de ces programmes.
L'ancienne option -a n'a plus d'effet.

1.79 return

RETURN

Usage : return [n]
Exemple : return 10

Sort d'un fichier script, ou sort d'un shell avec un code de sortie optionnel.

1.80 rm

RM (ou DELETE)

Equivalent à delete.

1.81 rpN

RPN

Usage : rpN expression

Exemple : rpN 3 7 * # imprime la valeur 21

Evalue une expression RPN, utilisant des valeurs 32-bits. Dans les versions précédentes du Shell, RPN contenait aussi des fonctions de chaînes, mais maintenant ces chaînes sont gérées par des commandes spécifiques, elles ne sont plus utilisées. A la fin de l'évaluation, RPN imprime les valeurs sur la pile, ainsi vous pouvez dire par exemple "rpN \$x 2 * | input x" pour doubler la valeur de la variable x. Les fonctions implémentées sont:

+ - * /	Signification usuelle; / signifie la division entière
%	Opérateur modulo ex. "rpN 7 3 %" répond 1
& ~	Opérateurs binaires: et, ou, non
> < ==	Tests pour plus-grand-que, plus-petit-que, egal. Pour tester >= (ou <=), prenez < ! (ou > !)
!	Opérateur logique Non
DUP	Duplique la valeur sur le haut de la pile.
DROP	Met la valeur sur le haut de la pile
SWAP	Echange 2 valeurs sur le haut de la pile

pour éviter toute confusion avec les redirections > et < doivent être entre guillemets. par exemple:

```
3 2 ">" # Imprime 1
```

1.82 run

RUN

Usage : run prgm args

Exemple : run emacs test.c

Lance un nouveau processus qui execute la commande indiquée. Cette commande n'est pas entièrement fiable: utilisez-la à vos propres risques. Voir aussi rback.

1.83 rxrec

RXREC

Usage : rxrec [nomport]

Crée un port compatible-AREXX du nom indiqué (par défaut ce sera "rex_csh"), puis met le Shell en veilleuse, en attente de messages.

ATTENTION: Le seul moyen de sortir de cet état est d'envoyer au port le message "bye".

Exemple:

Ouvrir 2 Shells dans 2 CLI séparés. Pour le 1er, tapez:

```
rxrec
```

Maintenant le 1er Shell ne répond plus au clavier; à la place, il attend des messages sur un port appelé "rex_csh". Maintenant, depuis l'autre, tapez:

```
rxsend rex_csh "dir df0:"
```

Vous verrez le catalogue de df0: dans le 1er Shell. Expérimentez comme vous le désirez, puis tapez:

```
rxsend rex_csh bye
```

Et tout redeviendra normal.

1.84 rxsend

RXSEND

Usage : rxsend [-lr] nomport commande...commande

Options :

- r règle la variable _result sur la valeur de résultat de la commande AREXX.
- l envoie la ligne bien portante comme *une* commande.

Envoie une commande à tout programme ayant un port compatible-AREXX. Sachez que chaque mot est envoyé comme une simple commande!

Vous ne devez rien charger pour utiliser ces commandes (ou rxrec): Tout ce dont vous avez besoin est d'un programme avec le port correct.

Un exemple est CygnusEdProfessional: il y a, par ex., une commande pour le réveiller, charger le programme test.c et sauter à la ligne 20:

```
rxsend rex_ced cedtofront "open test.c" "jmp to line 20"
# rex_ced est le nom du port AREXX pour CygnusEd
```

L'option `-r` met la variable `_result` sur la chaîne du résultat de la commande AREXX.

L'option `-l` envoie la ligne entière comme **une** commande.

Voir le manuel de votre application pour les détails, pour le nom des commandes et du port.

1.85 search

SEARCH

Usage : `search [-abcefnoqrvw] fichier...fichier chaîne`

Recherche une chaîne dans les fichiers indiqués. Seules les lignes contenant les chaînes indiquées sont affichées.

Si le nom de fichier est STDIN (en capitales), l'entrée standard est utilisée, ainsi vous pouvez utiliser `search` comme destination pour un pipe.

Exemple:

```
strings monprog * | search STDIN .library
```

Liste toutes les bibliothèques utilisées dans "monprog".

Search est très rapide si l'une des options `-w`, `-e`, si STDIN était spécifié et si le fichier se trouve en mémoire.

Options:

- `-a` (arrêt) stoppe la recherche dès que le motif a été trouvé une fois
- `-b` (binaire) affiche seulement les offsets en octets au lieu des lignes. S'il est combiné avec `-n`, affiche les nombres seuls.
- `-c` (capitales) Allume la sensibilité aux majuscules.
- `-e` (exclut) liste les lignes ne contenant pas le motif
- `-f` (fichiers) cause simplement l'affichage des noms des fichiers dans lesquels le motif a été trouvé.
- `-l` (left) le motif doit être au début de la ligne (c'est plus rapide que d'utiliser un motif d'expansion)
- `-n` (nombre) éteint la numérotation des lignes
- `-o` (only) trouve seulement les mots entiers
- `-q` (quiet) supprime l'impression des noms de fichiers
- `-r` (récursif) si vous spécifiez un répertoire, tous les fichiers dans ce répertoire sont passés en revue récursivement
- `-v` (verbose) montre chaque nom de fichier sur une simple ligne. C'est activé automatiquement si la recherche est redirigée.
- `-w` (wild) repérage de motifs de complétion. cf notes plus bas

Notes sur l'utilisation de motifs de complétion (wild cards);

- Utilise les complétions standard du Shell.
- Tous les motifs standard DOS sont autorisés `*` `?` `[]` `()` `|` `~` `'` `#`
- La Ligne ENTIERE doit correspondre à la chaîne, pas seulement une sous-chaîne.
- La chaîne DOIT être mise entre guillemets pour éviter la complétion

du motif

Exemples:

```
search -cr df0:include ACCESS
```

Trouve toutes les occurrences d'ACCESS (en majuscules) dans tous les fichiers contenus dans le répertoire include.

```
search -w shell.h "'#define*"
```

Liste seulement les lignes du fichier commençant par (et ne contenant pas seulement) #define. Notez l'usage de ' pour pouvoir utiliser le symbole spécial #.

1.86 set

SET

Usage : set [nom] [=] [chaîne]

Exemple : set abc bonjour

Set sans arguments liste toutes les variables actuellement définies. Set avec un argument liste le contenu de cette variable particulière. Indiquer nom et chaîne, initialise la variable 'nom' avec le contenu 'chaîne'.

Voir aussi la section sur les _variables spéciales.

1.87 setenv

SETENV

Usage : setenv varenv valeur

Met une valeur à la variable ENV: donnée. La valeur doit être mise entre guillemets si elle contient des espaces. Pour obtenir la valeur d'une variable ENV:, utilisez juste \$envvar n'importe où dans la ligne de commande.

1.88 sleep

SLEEP

Usage : sleep délai

Exemple : sleep 10

Dort pendant 'délai' secondes, ou jusqu'à ce que ^C soit tapé.

1.89 split

SPLIT

Usage : split srcvar dstvar...dstvar

Assigne un mot de srcvar a chaque dstvar, le reste de srcvar à la dernière dstvar.

Note: vous entrez les NOMS des variables et pas les variables.

1.90 stack

STACK

Usage : stack [nombre]

Exemple : stack [-s] 8000

Options :

-s affiche la taille seulement (nombre pure, pas de texte).

Change la pile par default (stack) pour ce CLI. Sans arguments, il l'affiche.

1.91 strhead

STRHEAD

Usage : strhead nomvar..séparateur chaîne

Exemple : strhead x . toto.bas # met "toto" dans x

Enlève tout ce qu'il y a après et y compris le caractère séparateur dans 'chaîne' et place le restant dans la variable 'nomvar'.

1.92 strings

STRINGS

Usage : strings [-bnrv] [fichier1..fichierN] [longmin]

Exemple : strings [-bnrv] c:dir c:list shell 7

Options :

- r si vous spécifiez un répertoire, tous les fichiers dans ce répertoire seront fouillés.
- n affiche le nom du fichier courant devant chaque chaîne
- b montre chaque chaîne entourée de caractères '|', ce pour montrer les espaces précédants ou suivants
- v sortie verbose avant chaque fichier (nom fichier, minlength)

Imprime les chaînes de longueur>=minlength contenues dans les fichiers

indiqués (habituellement binaires). Par défaut 4.

Vous ne pouvez utiliser un nom de fichier qui représente un nombre comme dernier argument. S'il n'y a pas de nom de fichiers spécifié l'entrée standard est utilisée, alors vous pouvez utiliser les chaînes comme destination pour un pipe.

1.93 strleft

STRLEFT

Usage : strleft nomvar chaîne n

Exemple : strleft x LongueChaîne 5 # met "Longu" dans x

Place les n caractères les plus à gauche de la chaîne dans la variable nomvar.

1.94 strlen

STRLEN

Usage : strlen nomvar chaîne

Exemple : strlen x Bonjour # met x à "7"

Met la longueur de la chaîne dans variable varnom.

1.95 strmid

STRMID

Usage : strmid nomvar chaîne n1 [n2]

Exemple : strmid x LongueChaîne 5 3 # mettra x à "Str"

Place n2 caractères de chaîne, à partir de n1, dans la variable nomvar. En omettant n2, vous aurez tous les caractères de n1 à la fin de la chaîne.

1.96 strright

STRRIGHT

Usage : strright nomvar chaîne n

Exemple : strright x LongueChaîne 5 # met x à "haîne"

Place les n caractères les plus à droite de la chaîne dans la variable nomvar.

1.97 strtail

STRTAIL

Usage : strtail nomvar séparateur chaîne

Exemple : strtail x . toto.bas # met "bas" dans x

Enlève tout ce qu'il y a avant (et y compris) le caractère séparateur dans 'chaîne' et place le restant dans la variable 'nomvar'.

1.98 source

SOURCE

Usage : source fichier [arguments]

Exemple : source monmake.sh all

Result : execute le fichier 'mymake.sh' en lui passant la variable
_passed = 'all'

Execute les commandes dans un fichier. Vous pouvez écrire des programmes SHELL dans un fichier et les exécuter alors avec cette commande. Les fichiers sourcés ont en plus l'avantage que vous pouvez avoir des boucles dans vos fichiers de commandes (cf GOTO et LABEL). Vous pouvez passer des arguments à votre fichier SCRIPT en indiquant les arguments après le nom du fichier. Les arguments sont passés via la variable _passed (comme une simple chaîne, un groupe de mots). Cf la variable _failat pour l'interruption des scripts.

De lignes longues peuvent être coupées par un anti-slash (\) à la fin de la 1ere partie. Une simple ligne doit être plus courte que 512 octets, mais la ligne concaténée peut être aussi longue que vous le voulez. Il n'y a pas de limite sur la longueur de la ligne concaténée.

L'action 'source' est accomplie automatiquement en ajoutant .sh au nom du fichier (pas besoin de lui mettre le bit-s) et en l'exécutant comme vous le feriez avec un programme en C :

```
----- fichier hello.sh -----
foreach i ( $_passed ) "echo yo $i"
-----
```

```
$ hello a b c
yo a
yo b
yo c
```

Si le dernier caractère de la ligne dans un fichier script est '{', toutes les lignes suivantes seront ajoutées à la ligne courante et séparées par des ';' jusqu'à ce que le dernier caractère d'une ligne soit '}'. Ces blocs peuvent être imbriqués. Vous pouvez utiliser des commentaires et des chaînes non terminées à l'intérieur.

```
----- fichier login.sh -----
```

```
alias complexe {
    echo -n "cet alias
    echo " marche!"
}
```

```
-----

$ login
$ complex
cet alias marche!
```

1.99 tackon

TACKON (ou ADDPART)

```
Usage : tackon var nomchemin nomfichier
Exemple : tackon x df0:c Dir # met "df0:c/Dir" dans x
ou : tackon x df0: Preferences # met "df0:Preferences" dans x
```

Ajoute correctement un nom de fichier à celui d'un chemin, et met le résultat dans la variable spécifiée.

1.100 tail

TAIL

```
Usage : tail nomfichier [num]
Exemple : tail readme 20
```

Affiche les "num" dernières lignes de "nomfichier". Si num n'est pas spécifié, 10 est pris par défaut. Si nomfichier n'est pas spécifié, l'entrée standard (stdin) est utilisé à la place.

1.101 tee

TEE

```
Usage : tee [fichier]
Exemple : cc test.c | tee >error.list
```

Copie l'entrée standard stdin sur stdout et le fichier donné. Si le fichier est omis, sterr est utilisé.

1.102 touch

TOUCH

Usage : touch fichier1 .. fichierN

Met la date de création des fichiers indiqués aux date et heure actuelles et réinitialise le bit archive.

Si un fichier n'existe pas, touch en créera un vide pour vous.

1.103 truncate

TRUNCATE

Usage : truncate [n]

Exemple : alias | qsort | truncate

Un filtre qui tronque la largeur de l'entrée standard au nombre spécifié, essayant de compter les tabs et les séquences d'échape. Si le nombre est omis, la fenêtre courante est utilisée.

1.104 type

TYPE (ou CAT)

Equivalent à CAT.

1.105 unalias

UNALIAS

Usage : unalias nom .. nom

Exemple : unalias vt

Détruit les alias..

1.106 uniq

UNIQ

Usage : uniq

C'est un filtre qui enlève les lignes consécutives en double dans un fichier. Il est très utile sur les fichiers triés.

1.107 unset

UNSET

Usage : unset nom .. nom

Exemple : unset abc

Supprime une ou plusieurs variables. Les détruit entièrement.

1.108 usage

USAGE

Usage : usage [commande...commande]

Si appelée dans arguments, usage vous donne une courte information sur les caractères spéciaux utilisés. Sinon, usage vous montre l'utilisation des commandes données. Appeler une commande avec '?' comme seul argument vous donnera aussi son usage.

1.109 version

VERSION

Usage : version

Montre le nom de la version actuelle, et des auteurs.

1.110 waitforport

WAITFORPORT

Usage : waitforport nomport [secondes]

Exemple : waitforport rexx_ced 5

Attend qu'un port apparaisse. Le temps par défaut est 10 secondes.

1.111 whereis

WHEREIS

Usage : whereis [-r] nomfichier [unité1...unitéN]

Options:

-r regarde tous les lecteurs.

Si un nom de fichier est juste donné, whereis fouille tous les sous

répertoires du répertoire courant pour trouver ce fichier. Un astérisque '*' est concaténé au fichier. Les motifs de complétion sont autorisés pour le fichier (aucun astérisque ne sera alors ajouté), mais pas les noms de chemins. Si des arguments additionnels sont donnés, whereis fouille seulement ces chemins, pas le répertoire courant.

1.112 window

WINDOW

Usage : window [-fblsaqw] [dimensions]

Options :

- f (front=devant) Fenêtre devant
- b (back=derrière) Fenêtre derrière
- l (large) Fenêtre en taille maximale
- s (small=petite) Fenêtre en taille minimale
- a (activée)
- q (questionne) Liste les fenêtres et les écrans ouverts
- w (width=largeur) Ignore les largeurs de fenêtre pour l'option "-q"

Diverses opérations sur la fenêtre CLI. Si les dimensions sont indiquées, elles doivent être de la forme x y largeur hauteur, avec un espace entre les valeurs.

La commande "window -l" peut être très pratique sur les machines PAL pour avoir une pleine fenêtre PAL à partir de votre séquence login, ou si vous utilisez le WorkBench en overscan .

L'option -q donne, pour chaque fenêtre et écran actuellement ouverts, le titre, le bord gauche, le bord haut, largeur, hauteur, (profondeur).

1.113 writefile

WRITEFILE

Usage: writefile nomvar

Ecrit un groupe de mots sur la sortie standard, un mot par ligne.

Notez que le nom de la variable (nomvar) doit être fourni, et non la valeur (\$nomvar).

1.114 commandes

abortline	action	addbuffers	addpart	alias
ascii	assign	basename	cat	cd
chgrp	chmod	chown	class	close
cls	copy	cp	date	dec
delete	dir	diskchange	echo	else
endif	error	exec	fault	filenote

flist	ftllower	fltupper	foreach	forever
forline	fornum	getenv	goto	head
help	history	howmany	htype	if
inc	info	input	join	keymap
label	linecnt	ln	local	ls
makelink	man	md	mem	menu
mkdir	mv	open	path	pri
protect	ps	pwd	qsort	quit
rback	readfile	rehash	relabel	rename
resident	return	rm	rpn	run
rxrec	rxsend	search	set	setenv
sleep	split	stack	strhead	strings
strleft	strlen	strmid	strright	strtail
source	tackon	tail	tee	touch
truncate	type	unalias	uniq	unset
usage	version	waitforport	whereis	window
writefile				

1.115 _abbrev

_abbrev

contient un nombre qui vous permet de sélectionner les différents modes de complétion des commandes:

- 0 commandes internes et commandes bufferisées avec "rehash" ne peut plus être abrégées (pareil a "unset _abbrev"
- 1 commandes internes peuvent être abrégées
- 2 commandes bufferisées avec "rehash" peuvent être abrégées, la première (partielle) correspondance de la liste est prise
- 4 commandes bufferisées avec "rehash" peuvent être abrégées, si la correspondance de commande correspond a une commande bufférisée complète alors elle est prise, ou alors la première partie correspondante est utilisée

Les numéros peuvent être ajoutés pour combiner les modes

La principale différence entre '2' et '4' est que '2' ne reconnaîtra pas une correspondance complète plus tard dans la liste. Par exemple, vous tapez "ed" et vous avez (dans cet ordre) "EdPlayer" et "Ed" dans votre liste, alors "Ed" ne pourra jamais être appelé (execepté avec le chemin entier). Alors '4' recherche d'abord pour une correspondance entière -- et s'il ne trouve rien (et alors seulement) il recherche les commandes abrégées.

Ceci rend obsolète l'utilisation de '2' et '4' ensemble.

Variable réglée sur '5' (1+4) par défaut.

1.116 _bground

`_bground`

Vraie si le shell a été lancé depuis une entrée non-interactive.

1.117 `_clnumber`

`_clnumber`

Contient le numéro (1-20) du CLI courant.

1.118 `_clipri`

`_clipri`

Priorité de la tâche où est éditée la ligne de commande.
(affecte aussi la complétion du nom du fichier)

1.119 `_cquote`

`_cquote`

Si réglé sur une valeur, les quotes sont gérées comme sous le Shell de Commodore, alors elles sont parsables par `ReadArgs()` (fonction appelée dans la `dos.library`). Le Shell Commodore traite les quotes dans une chaîne "telle qu'elle". Seules les quotes des extrémités entourent les autres caractères spéciaux (comme les espaces). En contraste avec les shells UNIX ou les quotations marquent toujours les autres caractères -sans tenir compte de leur position. Le style UNIX est réglé par défaut.

1.120 `_cwd`

`_cwd`

Contient une chaîne représentant le répertoire dans lequel nous nous trouvons depuis la racine. Le SHELL peut être trompé sur ce qui concerne son répertoire courant si des programmes externes changent le répertoire. Utilisez `PWD` pour reconstruire dans ce cas la variable `_cwd`.

1.121 `_debug`

`_debug`

Mode Debug... utilisez-le si vous l'osez. doit être mis à une valeur

1.122 `_dirformat`

`_dirformat`

Contient une chaîne formatée qui (si la variable est active) substitue la chaîne de format de la commande "dir" (option -z, regardez-y). Utilisée pour garder les alias courts et capable de spécifier les autres options après le format option (-z) sans utiliser @pickargs/@pickopts.

1.123 `_every`

`_every`

Contient le nom d'une commande qui doit être exécutée chaque fois juste avant que le prompt soit imprimé. N'utilisez pas ceci pour afficher le prompt.

1.124 `_except`

`_except`

voir EXCEPTION

1.125 `_failat`

`_failat`

Si une commande se termine avec un code de retour supérieur à ceci, le fichier script s'arrête. La valeur par défaut est 20.

1.126 `_hilite`

`_hilite`

Contient les attributs de fontes utilisés pour l'allumage des caractères (highlighting). Une lettre pour un attribut:

- b pour gras (bold)
- i pour italique
- u pour souligné (underlined)
- r pour inverse vidéo (reverse)
- c3 pour la couleur de devant no3
- c3,2 pour la couleur de devant no3 et celle du fond no2

Toutes les combinaisons sont autorisées. `_hilite` prend par défaut "c7", en mode terminal sur "r".

1.127 `_history`

`_history`

Cette variable est utilisée pour contenir une valeur numérique, et spécifie combien votre historique peut contenir. Mettez le à 0 pour annuler l'historique par exemple si vous testez des programmes contre les pertes de mémoire. La valeur par défaut est de 50.

1.128 `_insert`

`_insert`

Met le mode insertion/refrappe par défaut pour l'édition de la ligne de commande. ESC-i le bascule, mais après <RET> la valeur par défaut est rétablie comme indiqué par cette variable. Par défaut `_insert` est à 1, la mettre à zéro mettra la reffrappe par défaut.

1.129 `_ioerr`

`_ioerr`

Contient le code d'erreur secondaire de la dernière commande. Sera changé après toute commande externe et après une commande interne ayant échoué. Voir `@ioerr()`

1.130 `_kick2x`

`_kick2x`

Vraie si la `dos.library V37+` a pu être ouverte (ce qui signifie que le `kickstart 2.0` est présent)

1.131 `_kick3x`

`_kick3x`

Vraie si la `dos.library V39+` a pu être ouverte (ce qui signifie que le `kickstart 3.0` est présent)

1.132 `_lasterr`

`_lasterr`

Retourne le code de retour de la dernière commande exécutée. Ceci comprend les commandes internes aussi bien que les commandes externes, aussi pour utiliser ces variables devez-vous le faire IMMEDIATEMENT après la commande en question

1.133 `_lcd`

`_lcd`

Contient le nom du dernier répertoire. L'alias encapsulé 'dswap' fait un cd sur ce répertoire. S'il est rappelé à nouveau, vous êtes à nouveau là où vous étiez.

1.134 `_man`

`_man`

Le nom et le chemin d'accès à vos fichiers .doc. Par défaut sur 'csh:csh.doc'

1.135 `_maxerr`

`_maxerr`

La plus mauvaise (la plus haute) valeur de retour en date. Pour l'utiliser, vous la mettez habituellement sur '0', puis faites un jeu de commandes, et la contrôlez.

1.136 `_minrows`

`_minrows`

Donne le nombre minimum de colonnes qu'une fenêtre doit avoir pour autoriser le défilement rapide. Par défaut sur 34.

1.137 `_nobreak`

`_nobreak`

S'il est mis sur une valeur, désactive CTRL-C.

1.138 `_nomatch`

`_nomatch`

S'il est mis sur une valeur, il contrôle les motifs pour voir si quelque chose est identique (et avorte l'exécution de la commande si les motifs ne correspondent pas).

1.139 `_noreq`

`_noreq`

S'il est mis à une valeur, désactive les requesters système ("Please insert volume ..."). Allumé en mode vt200.

1.140 `_passed`

`_passed`

Cette variable contient les arguments passés quand vous exécutez un fichier par SOURCE ou exécutez un fichier .sh. Par exemple:

```
test a b c d
```

```
----- fichier test.sh -----
echo $_passed
foreach i ( $_passed ) "echo YO $i"
-----
```

1.141 `_path`

`_path`

Dit à CShell où aller chercher les fichiers exécutables. Le répertoire courant et le chemin AmigaDOS seront testés en premier. Le slash en fin de nom pour les répertoires n'est plus nécessaire. Le path entier sera testé d'abord pour la <commande>, puis pour <commande>.sh (exécution automatique des scripts shell). Exemple:

```
set _path ram:c,ram:,sys:system,dhl:tools,df0:c
```

(Ce chemin (path) a l'avantage que ces répertoires n'ont pas même besoin d'exister, que vous pouvez accéder aux unités (le path AmigaDOS connaît seulement les unités sous Kick 1.3) et qu'aucune recherche sur les disques n'apparaît au lancement)

1.142 `_prghash`

`_prghash`

Le nom du fichier d'où la hashlist (commande 'rehash') sera lue ou sauvee.

1.143 `_prompt`

`_prompt`

Cette variable peut maintenant contenir les caractères de contrôle suivants:

- `%c` pour un changement de couleur. Cela met le prompt en couleurs. Voir `_hilite`
- `%e` pour le temps écoulé. Le temps que la dernière commande a pris pour s'exécuter.
- `%m` pour la mémoire. Cela montre la mémoire courante en Ko
- `%t` pour l'heure. Cela montre l'heure courante au format HH:MM:SS
- `%d` pour la date. Ceci montre la date courante dans le format JJ-MMM-AA
- `%p` pour le chemin. Ceci insère le chemin courant.
- `%V` pour volume. Ceci insère le volume courant.
- `%n` pour le numéro. Ceci insère le numéro de process courant.
- `%v` pour la version. Ceci montre le numéro de version de CShell
- `%h` pour l'historique. Ceci affiche le numéro courant dans l'historique
- `%f` pour l'espace libre. Ceci affiche la place libre sur le lecteur courant.
- `%r` pour la priorité. Insère la priorité de la tâche courante
- `%s` pour les shells ouverts. Insère le résultat de 'howmany'
- `%U` pour user. Affiche l'utilisateur courant (seulement avec le package "MultiUser")
- `%x` pour les code de retours des commandes externes. Affiche le dernier code d'erreur

La valeur par défaut du prompt est actuellement `"%c%p> "`

La commande `if` mettra le prompt sur un `'_ '` si les commandes sont désactivées pendant l'attente d'une commande `'endif'` ou `'else'` (mode interactif seulement).

1.144 `_pipe`

`_pipe`

Le répertoire dans lequel les fichiers temporaires sont stockés.
Par défaut: `'T:'`

1.145 `_qcd`

`_qcd`

Contient le nom du fichier où tous les répertoires de votre disque dur sont stockés. S'il n'est pas mis, désactive le cd rapide.

1.146 `_rback`

`_rback`

Est le nom de la commande à être placée au début de la ligne de commande lorsque `'&'` lui a été ajouté. Par défaut c'est `'rback'`. Ca ne peut pas encore être une commande de plusieurs mots.

1.147 `_rxpath`

`_rxpath`

La même chose que pour `_path`, mais ceci représente là où CShell cherche les fichiers `.rexx`. Par défaut dans REXX:

1.148 `_scroll`

`_scroll`

contient le nombre de lignes à défiler en même temps quand on utilise le défilement rapide. Si elle n'est pas mise ou `<=1`, le défilement rapide est inactif. Par défaut c'est 3.

1.149 `_terminal`

`_terminal`

Indique si le shell a été lancé ou non en mode terminal.

1.150 `_titlebar`

`_titlebar`

Les mêmes caractères de contrôle que pour le `_prompt` peuvent être aussi utilisés pour `_titlebar`. La seule différence est que `%c` est ignoré. La barre titre est remise à jour chaque fois avant que le prompt n'apparaisse.

1.151 **_verbose**

`_verbose`

Si mis sur 's', allume le mode verbose pour les fichiers scripts (chaque commande sera affichée avant d'être exécutée). Si mis sur 'a', affiche tous les pas intermédiaires dans la substitution des alias. 'h' mettra en rouge la sortie de debugage. Toute combinaison autorisée: `set _verbose sah`

1.152 **_version**

`_version`

Contient le numéro de version du shell, par ex. 510.

1.153 **variables**

<code>_abbrev</code>	<code>_bground</code>	<code>_clnumber</code>	<code>_clipri</code>	<code>_cquote</code>
<code>_cwd</code>	<code>_debug</code>	<code>_dirformat</code>	<code>_every</code>	<code>_except</code>
<code>_failat</code>	<code>_hilite</code>	<code>_history</code>	<code>_insert</code>	<code>_ioerr</code>
<code>_kick2x</code>	<code>_kick3x</code>	<code>_lasterr</code>	<code>_lcd</code>	<code>_man</code>
<code>_maxerr</code>	<code>_minrows</code>	<code>_nobreak</code>	<code>_nomatch</code>	<code>_noreq</code>
<code>_passed</code>	<code>_path</code>	<code>_prghash</code>	<code>_prompt</code>	<code>_pipe</code>
<code>_qcd</code>	<code>_rback</code>	<code>_rxpath</code>	<code>_scroll</code>	<code>_terminal</code>
<code>_titlebar</code>	<code>_verbose</code>	<code>_version</code>		

1.154 **@abbrev**

`@abbrev(str1 str2 [len])`

vraie si les <len> premiers caractères de str1 sont une abréviation. de str2

1.155 **@abs**

`@abs(num)`

retourne la valeur absolue de <num>.

1.156 **@age**

`@age(fichier)`

l'age de ce fichier en jours, null-string si le fichier n'est pas trouvé.

1.157 @age_mins

```
@age_mins()
```

l'age de ce fichier en minutes, null-string si fichier non trouvé.

1.158 @appsuff

```
@appsuff( nom suffixe )
```

ajoute un suffixe (.ZOO) à une chaîne s'il n'y est pas déjà.

1.159 @arg

```
@arg( arg ... arg )
```

voir @pickargs()

1.160 @ask

```
@ask( titre objet ... objet )
```

demande de confirmation pour chaque objet et les retourne confirmés (très similaire à @confirm(), mais par défaut est négatif).

1.161 @availmem

```
@availmem( [type] )
```

retourne la mémoire 'chip', 'fast' libre ou sinon toute la mémoire.

1.162 @basename

```
@basename( chemin ... chemin )
```

retourne la partie nom de fichier des chemins.

1.163 @center

```
@center( mot longueur )
```

retourne une chaîne de longueur <longueur> avec <mot> centré à l'intérieur.

1.164 @checkpoint

```
@checkpoint ( nomport )
```

indique si le port donné existe.

1.165 @clinum

```
@clinum( nomproc )
```

retourne le numéro du cli identifié par un nom ou un numéro.

1.166 @complete

```
@complete( abbrev mot ... mot )
```

retourne le premier mot dont abbrev soit une abréviation.

1.167 @concat

```
@concat( word word ... word )
```

concatène tous les mots dans une chaîne séparés par des blancs, voir @split.

1.168 @confirm

```
@confirm( titre objet ... objet )
```

demande confirmation de chaque objet et retourne ceux qui ont été confirmés (très similaire à @ask(), mais par défaut est positif).

1.169 @console

```
@console( STDIN|STDOUT )
```

dit si stdin ou stdout sont interactives (non redirigées).

1.170 @dectohex

```
@dectohex( numéro )
```

retourne une chaîne représentant <nombre> en hexa.

1.171 @delword

```
@delword( mot mot ... mot n )
```

retourne une chaîne avec le n-ième mot effacé.

1.172 @delwords

```
@delwords( mot mot ... mot n m )
```

efface les m mots à partir du n-ième.

1.173 @dirname

```
@dirname( chemin )
```

enlève le nom de base du chemin, retourne juste le répertoire.

1.174 @dirs

```
@dirs( nom nom nom nom )
```

retourne les répertoires parmi les noms de fichiers donnés, voir @files

1.175 @dirst

```
@dirst( lformat fichier )
```

retourne toutes les informations (taille, date, commentaires) sur un fichier.

1.176 @drive

```
@drive( chemin )
```

affiche le nom de lecteur (unité) associé à <chemin>.

1.177 @drives

```
@drives( )
```

affiche tous les lecteurs disponibles.

1.178 @exists

```
@exists( fichier )
```

dit si un fichier existe ou non.

1.179 @fileblks

```
@fileblks( fichier fichier ... fichier )
```

retourne le # de blocs utilisés par les fichiers y compris les blocs de répertoires.

1.180 @filedate

```
@filedate( fichier )
```

retourne une chaîne représentant la date du fichier donné.

1.181 @fileinfo

```
@fileinfo
```

Equivalent à @dirstr.

1.182 @filelen

```
@filelen( fichier fichier ... fichier )
```

compte le nombre total d'octets des fichiers donnés.

1.183 @filenote

```
@filenote( fichier )
```

retourne le commentaire du fichier donné.

1.184 @fileprot

```
@fileprot( fichier )
```

retourne une chaîne comme ---arwed.

1.185 @filereq

```
@filereq( titre chemin&motif nomfichier )
```

affiche le requester de fichiers ASL et retourne le nom de fichier.
sélectionné

1.186 @files

```
@files( fichier fichier ... fichier )
```

vous donne les fichiers parmi ces noms, pas les répertoires, cf @dirs.

1.187 @filesize

```
@filesize
```

Equivalent à @filelen.

1.188 @flines

```
@flines( nomvar )
```

compte le nombre de lignes dans un fichier de lecture (plus rapide que @words).

1.189 @freebytes

```
@freebytes( chemin )
```

le nombre d'octets libres sur le chemin donné.

1.190 @freeblks

```
@freeblks( chemin )
```

le nombre de blocks libres sur le chemin donné.

1.191 @freestore

```
@freestore( chemin )
```

la quantité de place libre sur un chemin, donnée en K, M et G.

1.192 @getenv

```
@getenv( nomvar )
```

retourne la valeur de la variable env: nommée.

1.193 @getclass

```
@getclass( fichier )
```

retourne la classe (type) du fichier. Voir chapitre XIV.

1.194 @hextodec

```
@hextodec( hex-nombre )
```

retourne une chaîne représentant <hex-nombre> en décimales.

1.195 @howmany

```
@howmany( )
```

indique le nombre de shells qui tournent.

1.196 @index

```
@index( chaîne motif )
```

retourne l'emplacement du motif dans la chaîne (commençant à 1), 0 si ne le trouve pas.

1.197 @info

```
@info( chemin )
```

la ligne correspondante de la commande 'info', chaque entrée étant un mot.

1.198 @intersect

```
@intersect( mot1 mot2 mot3 , mot4 mot5 mot6 )
```

retourne tous les mots qui sont dans les deux listes. voir @union, @member.

1.199 @ioerr

```
@ioerr( num )
```

retourne la chaîne d'erreur correspondant à num.

1.200 @lookfor

```
@lookfor( fichier chemins )
```

cherche un fichier dans le répertoire courant et les chemins. Voir \$_path.

1.201 @lower

```
@lower
```

met en minuscules ses arguments. cf @upper.

1.202 @match

```
@match( mot ... mot "motif" )
```

retourne les mots dans la liste qui conviennent au motif DOS.

1.203 @max

```
@max( num num ... num )
```

calcule le maximum des nombres donnés.

1.204 @megs

```
@megs( nombre )
```

exprime un nombre en K, M et G (-octets), arrondi correctement.

1.205 @member

```
@member( mot1 mot mot ... mot )
```

vous dit si mot1 est parmi la liste des mots restants.

1.206 @min

```
@min( num num ... num )
```

calcule le minimum des nombres donnés.

1.207 @mix

```
@mix( arg1 ... argn )
```

mélange aléatoirement ses arguments.

1.208 @mounted

```
@mounted( unité )
```

retourne un booléen indiquant si l'unité spécifiée est montée.

1.209 @nameext

```
@nameext( nomfichier )
```

retourne tout ce qu'il y a après le dernier point de <nomfichier>.

1.210 @nameroot

```
@nameroot( nomfichier )
```

retourne tout ce qu'il y a avant le dernier point de <nomfichier>.

1.211 @opt

```
@opt( arg ... arg )
```

voir @pickopts().

1.212 @pathname

```
@pathname( chemin )
```

obsolete. utilisez @dirname.

1.213 @pickargs

```
@pickargs( arg ... arg )
```

prend parmi ses arguments ceux qui ne commencent pas par un '-'.

1.214 @pickopts

```
@pickopts( arg ... arg )
```

prend parmi ses arguments ceux qui commencent par un '-'.

1.215 @rnd

```
@rnd( seed )
```

retourne un nombre aléatoire de 32 bits (seed par défaut 1).
'seed' est optionel et peut être utilisé pour choisir une nouvelle valeur de seed pour @rnd(), si vous utilisez seed=0 alors CSH prends l'heure courante comme seed.

1.216 @rpn

```
@rpn( expression )
```

calcule l'expression rpn. Voir la commande rpn.

1.217 @scrheight

```
@scrheight( )
```

affiche la hauteur courante de l'écran dans lequel le shell fonctionne.

1.218 @scrwidth

```
@scrwidth( )
```

affiche la largeur courante de l'écran dans lequel le shell fonctionne.

1.219 @sortargs

```
@sortargs( nom ... nom )
```

trie ses arguments par ordre alphabétique

1.220 @sortnum

```
@sortnum( nombre ... nombre )
```

trie ses arguments numériquement

1.221 @split

```
@split( chaîne )
```

fait de chaque partie de @string séparée par un espace un mot, voir @concat.

1.222 @strcmp

```
@strcmp( nom nom )
```

retourne -1, 0 ou 1 suivant la comparaison alphabétique (sensible aux capitales).

1.223 @stricmp

```
@stricmp( nom nom )
```

retourne -1, 0 ou 1 suivant la comparaison alphabétique (non-sensible aux capitales).

1.224 @strhead

```
@strhead( séparateur chaîne )
```

voir la commande strhead.

1.225 @strleft

```
@strleft( chaîne numéro )
```

voir la commande strleft.

1.226 @strmid

```
@strmid( chaîne n1 n2 )
```

voir la commande strmid.

1.227 @strright

@strright(chaîne n)

voir la commande strright.

1.228 @strtail

@strtail(séparateur chaîne)

voir la commande strtail.

1.229 @subfile

@subfile(nomvar n m)

comme @subwords, mais agit sur un fichier de lecture et est plus rapide.

1.230 @subwords

@subwords(mot ... mot n m)

retourne les m prochains mots de la liste donnée à partir du n-ième.

1.231 @tackon

@tackon(chemin fichier)

voir la commande tackon.

1.232 @trim

@trim(mot mot mot)

enlève tous les espaces au début ou à la fin des mots.

1.233 @unique

@unique(mot ... mot)

trie les arguments et rend chacun d'eux unique.

1.234 @union

```
@union( nom ... nom , nom ... nom )
```

retourne tous les noms étant dans l'une des deux listes. Voir @intersect, @member.

1.235 @upper

```
@upper( mot ... mot )
```

met le mot donné en majuscules, voir @lower.

1.236 @volume

```
@volume( chemin )
```

retourne le nom de volume de ce chemin ou "".

1.237 @wincols

```
@wincols( )
```

retourne le nombre de colonnes dans la fenetre de shell courante.

1.238 @winheight

```
@winheight( )
```

affiche la hauteur de votre fenêtre en pixels.

1.239 @winleft

```
@winleft( )
```

retourne le bord gauche de votre fenêtre.

1.240 @winrows

```
@winrows( )
```

retourne le nombre de lignes dans la fenêtre de shell courante.

1.241 @wintop

```
@wintop( )
```

retourne le bord haut de votre fenêtre.

1.242 @winwidth

```
@winwidth( )
```

affiche la largeur de votre fenêtre en pixels.

1.243 @without

```
@without( nom ... nom , nom ... nom )
```

retourne tous les noms de la liste 1 qui ne sont pas dans la liste 2.

1.244 @word

```
@word( nom ... nom n )
```

extraît le n-ième mot de la liste.

1.245 @words

```
@words( nom ... nom )
```

retourne le nombre de mots dans la liste.

1.246 functions

@abbrev	@abs	@age	@age_mins	@appsuff
@arg	@ask	@availmem	@basename	@center
@checkport	@clinum	@complete	@concat	@confirm
@console	@dectohex	@delword	@delwords	@dirname
@dirs	@dirstr	@drive	@drives	@exists
@fileblks	@filedate	@fileinfo	@filelen	@filenote
@fileprot	@filereq	@files	@filesize	@flines
@freebytes	@freeblks	@freestore	@getenv	@getclass
@hextodec	@howmany	@index	@info	@intersect
@ioerr	@lookfor	@lower	@match	@max
@megs	@member	@min	@mix	@mounted
@nameext	@nameroot	@opt	@pathname	@pickargs

@pickopts	@rnd	@rpn	@scrheight	@scrwidth
@sortargs	@sortnum	@split	@strcmp	@stricmp
@strhead	@strleft	@strmid	@strright	@strtail
@subfile	@subwords	@tackon	@trim	@unique
@union	@upper	@volume	@wincols	@winheight
@winleft	@winrows	@wintop	@winwidth	@without
@word	@words			