TOS 1.4

système d'exploitation

ATARI ST

version 1989/1990

BUREAU

Modifications faites :

1 **Toutes les fenêtres d'arrière plan sont mises à jour** aprés une copie, un déplacement, un effacement de fichier, copie et formatage de disquettes.

2 **Informations sur fichier et sur dossier utilisent maintenant le même dialogue.** Le champ du Nom permet de renommer le fichier ou le dossier, mais les attributs fichier sont valables seulement pour les fichiers.

3 **Copie Disquette et Formatage Disquette sont combinés maintenant dans une même boîte de dialogue.** N'importe quelle opération choisie du Bureau est selectionnée dans cette nouvelle boîte quand elle paraît. Par exemple, si le disque A: est glissé vers le disque B:, la boîte Copie/Formatage paraît avec l'option "Copie" selectionnée. En cliquant la touche "Formatage", l'utilisateur choisit de formater une disquette.

4 **Le dialogue "Formatage Disquette" prend par défaut le choix "Fin"** si on appuie sur <Return>.

5 **Le Bureau peut formater maintenant des disquettes avec des secteurs de Boot compatibles MS-DOS**

6 **Un fichier peut être "Déplacé" aussi bien que copié.** Si la source et la destination sont sur le même support ou partition, le fichier est déplacé en le renommant par GEMDOS. Autrement, le fichier est copié et le fichier initial est effacé. Pour utiliser cette fonctionalité: selectionner les fichiers, maintenir la touche <Control>, et glisser les fichiers vers leur destination. Une boîte "Fichier(s) déplacé(s)" paraît.

7 **Les opérations Copie, Efface, et Déplace peuvent être interrompues avec <Undo>.** En appuyant sur la touche <Undo> le système affiche une boîte de dialogue demandant à l'utilisateur s'il veut annuler.

8 **La boîte de dialogue Copie, Efface, et Déplace garde toujours en vue le dossier destination et le nom du fichier** durant l'operation.

9 **Les programmes GEM peuvent auto-démarrer à partir du disque.** Ceci est fait par l'intermédiaire du dialogue "Installer une application" où l'utilisateur peut choisir l'état "Auto" démarrage, puis "Sauvegarde Bureau". L'application à auto-démarrage peut être désinstallée dans ce dialogue. L'application à auto-démarrage peut être un "GEM" ou un "TOS", mais non pas un "TTP" (TOS prend des paramètres.) Le fichier "DESKTOP.INF" resultant contient la ligne :
#Z <type><chemin application><application>, où <type> contient 01 pour un programme graphique (GEM) ou 00 pour un programme non graphique (TOS).

10 **"Installer une unité de disque" prend "INSTALLER" par défaut**, et quand vous installez un disque qui existe dèja, il met à jour le nom de l'icône actuelle.

11 **"Installer une application" a un item "ENLEVER"**, et "INSTALLER" est prise par défaut maintenant. Ce dialogue est aussi utilisé pour choisir l'état Normal ou Auto-démarrage. (Voir "les programmes GEM à auto-démarrage")

12 **Quand on installe une application, le Bureau passe maintenant le chemin d'accés complet** du document cliqué. Dans le fichier DESKTOP.INF, il sauvegarde le chemin d'accés complet de l'application installée.

13 **"Définir les préférences" détermine si le système doit confirmer les conflits de noms.** Si "Confirmation demandée pour : Effacement" est "Non" les fichiers seront effacés sans aucun message. S'il arrive un conflit de nom de *dossier* un dialogue s'établit dans tous les cas.

14 **Dans le cas d'un "conflit de nom durant copie", l'utilisateur a trois choix** : *COPIER* fonctionne en copiant le fichier et en écrasant celui de la destination), *SUIVANT* ne fait rien avec ce fichier particulier et continue l'opération de copie, et *ANNULER* annule l'opération entière de copie.

15 **Le Bureau montre le maximum de fichiers possibles dans chaque fenêtre.** La seule limite est l'étendue mémoire du système. Ceci enlève l'ancienne allocation statique de fichiers limitée à 400.

16 **"Informations" permet maintenant de renommer un dossier.**

17 **La fonction Voir/Imprimer fichiers texte a été complétement réécrite**

Les possibilités dans "Voir" sont :

a – Appuyant sur <Espace> au milieu d'une page fait que –Suite– compte 24 lignes à partir de ce point au lieu d'attendre le message –Suite– puis appuyer sur <Espace>.

b – d, D et ^D oblige –Suite– à venir au bout d'une page à partir de ce moment ; <Return> l'oblige à venir aprés juste une ligne

c – q, Q et ^C arrêtent la sortie immédiament.

d – L'auto-dépassement en largeur de ligne n'est pas modifié ; si le fichier tient en largeur, vous l'obtenez tel qu'il est.

Les possibilités dans "Imprimer" sont :

a – Le clavier est consulté tout les 16 caractères.

b – q, Q, ^C et <Undo> causent tous l'arrêt de l'imprimante.

c – Imprimer fait son propre évaluation du Tab. Il comprend TAB, BS, CR et les caractères de contrôle.

18 **Quand les fichiers sont copiés, le pointeur change maintenant en une "abeille"** même si le Bureau est réglé pour copier sans confirmation.

19 **Quand une erreur survient en copiant des fichiers , le pointeur devient maintenant une "abeille"** si on clique sur "REESSAYER".

20 **Abandonner et Réessayer fonctionnent maintenant comme prévu quand une erreur survient lors du formatage d'une disquette.**

21 **Quand on copie des disquettes entre A: et B:** on aura un message d'erreur s'il manque une disquette dans l'un des lecteurs. "Abandonner" ramène l'utilisateur maintenant au dialogue "Copie Disquette".

22 **Les copies de disquettes sur un système avec un seul lecteur demandent le minimum d'échange de disquettes possible.** Toute la mémoire disponible est utlisée comme un tampon de copie.

23 **Si une erreur survient quand un lecteur est "Ouvert", il n'y a plus de fenêtre vide restant sur l'écran.**

24 **Le message des droits d'auteurs donne maintenant 1986, 1987, 1988 et 1989.** 1988 et 1989 ont été ajoutées.

25 **Quand on se remet d'un plantage d'application, wind_update(FALSE) est remis** avant d'entrer dans le corps event_multi qui attend l'interaction de l'utilisateur. Ceci est géré par la nouvelle fonction 'wind_new'. (Regarder la documentation de "wind_new").

26 **Si on essaie d'obtenir le repertoire d'un lecteur sans disquette,** 'Abandonner' annule maintenant l'opération et retourne au Bureau.

27 **Plusieurs dialogues sont maintenant plus bref.** Ceci concerne aussi les dialogues d'erreurs comme par exemple celui qui affiche "Votre Péripherique de sortie ne reçoit pas de données..." et plusieurs autres.

28 **Le séparateur de Date est maintenant "/" dans "Informations" sur fichier ou sur dossier.** Avant, ils n'étaient pas les mêmes. L'un utilisait "/" l'autre "-".

AES

Modifications faites :

1 **Le selecteur de fichier (FS) a été modifié.** Les améliorations sont :

a – Une application peut maintenant envoyer un "titre" au sélecteur de fichier. (Voir "fset_exinput()")
b – FS fournit 16 étiquettes lecteur pour une sélection plus facile.
c – FS agit différement maintenant vis à vis de <RETURN> lors de l'édition de texte : aprés avoir édité un chemin d'accés, appuyer sur <RETURN> fait accepter le chemin et réaffiche FS. Aprés avoir édité un nom de fichier, FS disparaît.
d – FS reprend sa position initiale après "Voir" un fichier.
e – L'allocation statique limitée à 100 fichiers par FS a été enlevée.
f – FS peut traiter maintenant de longs chemins d'accés.
g – FS traite maintenant correctement de multiples "ANNULER/OK".
h – FS conserve les valeurs courantes de l'adresse du buffer DTA, le 'clipping' des rectangles, et les répertoires par défaut.
i – FS est doté de nouvelles bibliothèques.
j – Le nouveau FS a la même taille que l'ancien.

2 **Si une disquette est enlevée, lors de l'appel de FS**, le système traite correctement "ANNULER" dans le dialogue erreur resultant.

3 **L'appel de "appl_init()" donne un numero de version de 0130 dans global(0).**

4 **L'exécution d'un programme à partir du shell de AES entraîne par défaut le répertoire qui contient ce fichier.**

5 **L'appel de wind_get() avec un paramétre de champ de WF_SCREEN donne l'adresse et la longueur** du buffer Menu/Alerte de l'AES.

6 **La bascule entre Vrai et Faux sur la barre du menu ne détruit plus le sémaphore.**

7 **La texture–point dans la barre de déplacement d'une fenêtre n'est plus perturbée** par un réaffichage comme aprés la disparition d'un dialogue.

8 **L'AES traite maintenant les champs modifiables suivis par des caractères non modifiables** dans les boîtes de dialogue.

Documentation des Appels AES

Ce qui suit est une documentation supplémentaire au manuel existant de l'AES et amène une clarification aux informations existantes.

Bibliothéque Routines Objets

INT_OUT(1) = ob_ednewidx (non utilisée)

ob_edreturn = objc_edit(ob_edtree, ob_edobject, ob_edchar, ob_edidx, ob_edkind) ;

où ob_edtree est l'adresse de l'arbre objet contenant l'objet et le texte à éditer.

7.3.3 FORM_ALERT

Objet : Afficher une alerte.

Cette section décrit la séquence complète des routines internes à FORM_ALERT.

Paramètres : control(0)=52
control(1)=1
control(2)=1
control(3)=1
control(4)=0

int_in(0)=fo_adefbttn

int_out(0)=fo_aexbttn

addr_in(0)=fo_astring

fo_adefbttn la touche de sortie par défaut de la forme (paragraphe 7.1.3.1)
0 – pas de touche de sortie par défaut
1 – première touche de sortie
2 – Deuxième touche de sortie
3 – Troisième touche de sortie

fo_aexbttn un nombre identifiant la touche de sortie sélectionnée par l'utilisateur
1 – première touche de sortie dans la chaîne
2 – deuxième touche de sortie dans la chaîne
3 – troisième touche de sortie dans la chaîne

fo_astring l'adresse de la chaîne contenant l'alerte.
* Aucune ligne d'alerte ne doit dépasser 30 caractères.

Exemple d'appel à partir du langage C :

```
fo_aexbttn=form_alert(fo_adefbttn,fo_astring) ;
```

FORM_ERROR

Objet : Afficher une boîte d'erreur.

Paramétres : control(0)=53
control(1)=1
control(2)=1
control(3)=0
control(4)=0

int_in(0)=fo_enum

int_out(0)=fo_eexbttn

fo_enum	Les codes des erreurs GEMDOS dans le format PC DOS (à partir de 1)
fo_eexbttn	un code identifiant la touche de sortie selectionnée par l'utilisateur 1 – première touche de sortie dans la chaîne 2 – deuxième touche de sortie dans la chaîne 3 – troisième touche de sortie dans la chaîne

Exemple d'appel à partir du langage C :

```
fo_eexbttn=form_error(fo_enum) ;
```

Messages d'erreur :

fo_enum	2,3,18	Cette application ne peut pas trouver le dossier ou le fichier auquel vous voulez accéder
fo_enum	4	Cette application n'a plus d'espace disponible pour ouvrir un autre document. Pour dégager de l'espace, fermez tout document dont vous n'avez pas besoin.
fo_enum	5	Un objet ayant ce nom existe déjà sur le répertoire, ou bien cet objet est spécifié du type Lire-seulement.
fo_enum	8,10,11	Il n'y a pas suffisamment de mémoire pour l'application que vous venez de lancer.
fo_enum	15	Le lecteur que vous avez spécifié n'existe pas.

13.3.2 SHEL_WRITE

Objet : Informer GEM AES qu'il faut lancer une autre application à la fin de celle en cours et, dans ce cas, laquelle.

Paramétres : control(0)=121
control(1)=3
control(2)=1
control(3)=2
control(4)=0

int_in(0)=sh_wdoex
int_in(1)=sh_wisgr
int_in(2)=sh_wiscr

int_out(0)=sh_wreturn

addr_in(0)=sh_wpcmd
addr_in(1)=sh_wptail

Description :

sh_wdoex — Une instruction codée servant à sortir du système ou lancer une autre application quand l'utilisateur quitte l'application courante.
0 – Sort et revient au BUREAU de GEM
1 – Lancer une autre application

sh_wisgr — un code indiquant si l'application suivante est une application graphique ou non.
0 – Une application non grahique
1 – Une application graphique

sh_wiscr — actuellement ignoré par l'AES.

sh_wreturn — un message de retour codé.
0 – il y a une erreur
n – il n'y a pas une erreur

sh_wpcmd — l'adresse du nouveau fichier de commande à exécuter.

ah_wptail — l'adresse de la ligne de commande du programme suivant.
Note : Le premier octet du tampon de la ligne de commande contient la longueur (en octets) de cette ligne. La ligne de commande elle même commence au second octet du tampon.

Exemple d'appel à partir du langage C :

```
sh_wreturn=shel_write(sh_wdoex, sh_wisgr, sh_wiscr,
                      sh_wpcmd, sh_wptail) ;
```

13.3.5 SHEL_GET

Objet : Permet à l'application de lire des données du tampon interne du shell de AES (DESKTOP.INF).

La taille des données devant être sauvegardées dans ce tampon ne doit pas dépasser les 1024 octets.

Paramétres : control(0)=122
control(1)=1
control(2)=1
control(3)=1
control(4)=0

int_in(0)=sh_glen

addr_in(0)=sh_gbuff

int out(0)=sh greturn

sh_greturn un message de retour codé.
0 – il y a une erreur
n (entier positif) – aucune erreur

sh_glen la taille du tampon (Max 1024.)

sh_gbuff l'adresse du tampon.

Exemple d'appel à partir du langage C :

```
sh_greturn=shel_get(sh_gbuff, sh_glen) ;
```

13.3.6 SHEL_PUT

Objet : Permet à l'application de sauvegarder des données dans le tampon interne du shell de AES (DESKTOP.INF).

Note : Actuellement, le BUREAU de AES utilise ce tampon pour garder les données de desktop.inf . Toute utilisation de ce tampon peut détruire les données qu'il garde.

La taille des données devant être sauvegardées dans ce tampon ne doit pas dépasser les 1024 octets.

Paramétres : control(0)=123
control(1)=1
control(2)=1
control(3)=1
control(4)=0

int_in(0)=sh_plen

addr_in(0)=sh_pbuff

int_out(0)=sh_preturn

sh_preturn un message de retour codé.
0 – il y a une erreur
n (entier positif) – aucune erreur

sh_plen la taille du tampon (Max 1024.)

sh_pbuff l'adresse du tampon.

Exemple d'appel à partir du langage C :

```
sh_preturn=shel_put(sh_pbuff, sh_plen) ;
```

Il y a deux nouvelles routines AES, FSEL_EXINPUT et WIND_NEW. Elles sont documentées ci dessous :

FSEL_EXINPUT

objet : Cette fonction a la même fonctionalité que FSEL_INPUT à l'exception qu'elle accepte un paramétre entrée supplémentaire appelé fs_label pour afficher une chaîne de 30 caractères en tête de la boîte du selecteur de fichier. Cette chaîne va remplacer la chaîne originale du selecteur. Le but de cette fonctionalité est de permettre au programme de donner plus d'informations, par exemple Lecture ou Ecriture pendant que l'utilisateur est en train de selectionner un fichier.

Paramétres :
control(0)=91
control(1)=0
control(2)=2
control(3)=3
control(4)=0

in_out(0)=fs_ireturn
in_out(1)=fs_iexbutton

addr_in(0)=fs_iinpath
addr_in(1)=fs_innsel
addr_in(2)=fs_label

sh_ireturn — un message de retour codé.
0 – il y a une erreur
n (entier positif) – aucune erreur

fs_iexbutton — un code identifiant la touche de sortie sélectionnée par l'utilisateur
0 – Annuler
1 – Confirmer

fs_iinpath — L'adresse du tampon qui garde les spécifications du répertoire initial affiché dans le FS. Ce tampon garde aussi celles du répertoire qui était affiché dans le FS quand on a sélectionné Confirmer ou Annuler.

fs_innsel — L'adresse du tampon qui garde la sélection initiale affichée dans le FS. Ce tampon va garder aussi la sélection qui était dans le FS quand on a sélectionné Confirmer ou annuler

Exemple d'appel à partir du langage C :

```
fs_ireturn=fsel_exinput(fs_innpath, fs_innsel,
                        &fs_iexbutton, fs_label) ;
```

WIND_NEW

objet : Ferme et efface toutes les fenêtres, remet à zero la fonction wind_update(), vide tous les buffers fenêtres et ramène l'appartenance de la souris au système.

Paramétres : control(0)=109
control(1)=0
control(2)=0
control(3)=0
control(4)=0

Exemple d'appel à partir du langage C :

```
wind_new();
```

VDI

Modifications faites :

1 **Ptsin a été étendu pour permettre 512 vertices (couples).** Ceci était vrai (mais non documenté) depuis les ROMs du 22/4/87 (Mega).

2 **vqt_extent fonctionne bien maintenant quand la rotation est de 270 degrés**

3 **vqt_mouse a été modifié ;** pour être plus fiable.

GEMDOS [BIOS/XBIOS]

Modifications faites :

1 **L'erreur connue sous le nom "Erreur du 40-dossier" est corrigée.** Il existe toujours des limites sur la profondeur des dossier et le cumul de profondeur en fichiers ouverts, mais ces limites sont difficilement atteintes et on peut même les étendre par "FOLDRXXX.PRG" qui est partout disponible. Aussi, un dossier ne prend de la place que s'il est "activé" et non pas s'il est juste affiché. (Voir "Informations sur l'OS Pool".)

2 **Les restrictions sur Malloc ont été largement allégées.** (Exemple : la limite des 20 blocks/Traitement). Malloc utilise toujours le même "OS Pool" comme les dossiers, donc il doit toujours être utilisé comme réserve. Utiliser Malloc pour des blocs volumineux, et utiliser un autre gestionnaire de mémoire (par exemple la fonction malloc() de la bibliothéque d'un compilateur C) pour une gestion ordinaire de la mémoire.

3 **Un OS Pool surchargé a maintenant un comportement prédictible et sans risque.** La machine va bloquer avec un message indiquant que le pool est surchargé, et que il faut utiliser FOLDRXXX.PRG pour l'étendre. Elle ne donne plus des résultats erronés ; ni ne cause de dommage au disque.

4 **La routine de recherche dans la FAT pour disques et disquettes est plus rapide.** Cette différence de vitesse est remarquable quand utilise "Informations...", ou quand on crée un fichier sur un disque dur surchargé.

5 **Le 'buffering' des secteurs dans GEMDOS a été amélioré,** et l'utilisateur peut rajouter d'autres 'buffers' pour améliorer les performances du système. L'ancien 'Buffering' n'était pas efficace et (dans un des cas) presque faux.

6 **Frename peut maintenant renommer un dossier ;** il ne peut pas pourtant, déplacer le dossier ailleurs dans le répertoire comme on peut le faire pour un fichier.

7 **GEMDOS évite maintenant la duplication de fichiers ;** il ne crée plus jamais deux fichiers avec le même nom.

8 **Ddelete suivant immédiatement un Dcreate fonctionne bien maintenant.** Avant, ceci pouvait ne pas marcher mais un second Ddelete fonctionnait bien.

9 **Dcreate (mkdir) maintenant trouve et traite les erreurs.** Dcreate se remet correctement des echecs en créant un répertoire (comme les erreurs d'écriture et les erreurs de Disque-Plein), et ne crée pas de sous-répertoires incomplets.

10 **Fread et Fwrite avec un paramètre de longueur zero** ne bloque plus.

11 **Le bit "Archive" dans un attribut (0x20) est correctement maintenu** : il est appliqué quand un fichier est créé et quand on écrit dans un fichier. Les programmes Archive/Mise à jour peuvent consulter ce bit pour voir si ce fichier doit être sauvegardé, et peuvent annuler ce bit quand ils ont sauvegardé le fichier. (Voir "Informations sur l'octet d'attribut").

12 **Fattrib s'assure de la légalité de ce qui doit être fait ;** par exemple, un essai de modifier le bit de répertoire d'un fichier ou d'un sous-répertoire est interdit.

13 **Les entrées de "." et ".." dans les sous-répertoires sont correctement datées.**

14 **Fsettime et Fsetdate ajustent la date et l'heure du BIOS .** L'inverse est aussi valable : la modification de la date et de l'heure du BIOS affecte aussi celles de GEMDOS. C'est le même comportement que dans les (Mega) ROMs du 22/4/87, là où ceci était nécessaire pour le circuit d'horloge temps réel.

15 **Fdatime n'inverse plus les octets d'entrées de l'utilisateur quand on écrit une nouvelle date/heure.**

16 **Fdatime donne EIHNDL pour des 'handles' non valables et des 'handles' qui se réfèrent à des périphériques alphanumériques** (qui n'ont ni date ni heure).

17 **Cconws est plus rapide** qu'auparavant quand stdout est redirigé.

18 **Rediriger vers l'imprimante (handle 3, "PRN:") fonctionne correctement.** Avant, GEMDOS appelait la routine du BIOS de l'état de l'entrée sur l'imprimante ce qui n'est pas implémenté

19 **Le traitement console de ^S et ^C est efficace.** ^C durant n'importe quelle fonction d'E/S (Cconin, Cconout, Cnecin, Cconws, Cconrs) cause l'arrêt de la tache courante.

20 **Cconrs a été amélioré :** Il supporte les touches qui ont le bit de poids fort de leur code ASCII à un, et ne renvoie plus les données d'entrées à la console quand l'entrée standard vient d'un fichier.

21 **Les fonctions E/S caractères (y compris 'crawio' et 'Cconout') ont un fonctionnement anticipé quand elles sont redirigées.** En particulier, l'état-entrée et l'état-sortie fonctionnent.

22 **Les tampons 'type-ahead' de la console sont implémentés correctement,** et ne grossissent plus indéfiniment dans la mémoire appropriée.

23 **La 'reprise' du clavier a été améliorée :** si vous appuyez sur une touche ('#' par exemple), et vous la maintenez, vous obtenez plusieurs #. Si maintenant vous appuyez sur '$' sans relacher le # vous obtenez un $, puis plusieurs $. Les $ n'arrêtent pas de sortir - même si entre temps vous relachez la touche # -, que si vous relachez la touche $ elle même.

24 **L'OS Pool a été réduit à la taille qu'il avait le 20/11/85.** Ceci permettra de fonctionner à des programmes qui ont fonctionné avec les ROMs du 20/11/85 dans les 520ST mais n'ont pas fonctionné avec les ROMs du Mega.

25 **Reset est disponible à partir du clavier.** Appuyez sur Control et Alternate, puis sur "Delete" (en dessous de Backspace). Ceci fera la même chose que d'appuyer sur le bouton reset. Un reset à Froid est possible avec les touches Control-Alt-Shift (de droite)- Delete. Il pourra être éliminé par un gestionnaire de clavier défini par l'utilisateur.

26 **Le démarrage d'un programme est maintenant aussi rapide que les ROMs du Mega ;** plus rapide que les ROMs du 20/11/85.

27 **Les disquettes sont testées sur leur possibilité d'auto-démarrage à chaque démarrage Chaud ou Froid,** même si un auto démarrage est fourni par un disque dur. Avant ceci était fait juste au cas d'un démarrage à Froid.

28 **La fermeture d'un 'Handle' standard (0-5) le remet à sa définition BIOS par défaut ;** avant ceci était indéfini. La méthode appropriée d'utilisation d'un 'handle' standard est de lui faire un 'Fdup' pour avoir une copie, puis un 'Fforce' à ce que l'on veut, puis 'Fforce' de la copie quand c'est fini pour le remettre à son état initial. Fermer fonctionne maintenant, jusqu'à au moins le remettre à un état connu.

29 **Les disquettes sont formatées pour être compatibles avec l'IBM PC.** La routine Flopfmt du XBIOS a été changée pour introduire plus d'identification au début de chaque piste. Ceci est pour augmenter la compatibilité avec certains controleurs de lecteurs 3.5", ce qui demande plus de remplissage entre l'impulsion d'indexe et le premier secteur sur le disque. Aussi, la routine 'protobt' du BIOS peut créer des secteurs de démarrage au format MS-DOS. Les programmes qui n'utilisent pas ces routines pour formater le disque ne seront pas affectés.

30 **Pexec traite les cas exceptionnels correctement.** Pexec ne donne plus de bombes ni ne laisse de fichiers ouverts, et il libère la mémoire correctement. Il peut aussi traiter avec des fichiers ayant plus de 32K d'informations de 'relocation'.

31 **Rsconf(-2,-1,-1,-1,-1,-1) donne maintenant le dernier taux de transmission établi par Rsconf.** Si le premier paramètre de Rsconf est -2, tout le reste est ignoré.

32 **La structure de la partie privée du DTA (utilisée pour Fsfirst Fsnext) a changé.** Les applications qui s'appuient sur sa structure (réservé, non documenté) peuvent ne pas fonctionner.

33 **GEMDOS reconnaît mieux maintenant les 'media change'** : avant, il les perdait parfois.

INFORMATIONS SUR LES NOUVELLES VARIABLES SYSTEME

En commençant par les ROMs (Mega) du 22/4/87, il y a trois nouvelles variables que vous pouvez consulter : leurs pointeurs sont dans le bloc d'en tête du système. Un pointeur à ce bloc se trouve à l'adresse $4f2 (_sysbase).

A un offset $20 à partir de l'adresse _sysbase on trouve un pointeur à la variable "_root." _root est un pointeur qui contient l'origine du ospool, la mémoire interne utilisée par GEMDOS. Ce pointeur est utilisé par le programme FOLDR100.PRG. Vous pouvez toujours étendre le 'pool' de l'OS de la même manière qu'avant, mais vous devez savoir que l'OS va prendre le 'pool' que vous avez ajouté pour l'utiliser d'une manière *DIFFERENTE* de celle utilisée auparavant. La règle est : une fois que vous ajoutez de la mémoire à l'OS, *NE JAMAIS Y TOUCHER* après.

A un offset de $24 à partir de _sysbase il existe un pointeur à la variable kbshift, un Mot qui garde les bits d'état des Shifts du clavier. Celui ci est mis à jour au niveau des interruptions.

A un offset de $28 à partir de _sysbase on trouve un pointeur à la variable _run, Mot double qui garde l'ID du processus (l'adresse de la page de base) que GEMDOS est en train d'exécuter.

Toutes ces variables sont à lire seulement. Des effets imprévisibles peuvent se produire si on essaie de les modifier. Kbshift est la plus intéressante, parce qu'elle vous permet de tester rapidement une éventuelle touche 'shiftée'.
En effet, la combinaison suivante de routines peut être utilisée pour voir si les deux touches Shift sont appuyées, ce qui pourra servir pour arrêter un programme.

```
int *p_kbshift ;

init()
{
        long _sysbase=*((long *)0x4f2) ;

        p_kbshift=*(long *)(_sysbase+0x24) ;
}
#define kbtest() {if ((*(p_kbshift)& 0x03) == 0x03) abort() ; }
```

Après l'appel de init(), l'utilisation de la macro kbtest() à l'intérieur de la boucle principale de votre programme est un moyen rapide pour tester si les deux touches Shift sont appuyées – signalant que l'utilisateur veut Annuler. Ceci est beaucoup plus rapide que d'utiliser la routine Kbshift() du BIOS, et donne la même taille de code.

INFORMATIONS SUR LE POOL DE L'OS

Il existe des limites internes dans GEMDOS que les programmeurs et les utilisateurs doivent comprendre. Dans un sens général, vous devez savoir que ces limites concernent la profondeur limite de votre structure hiérarchique (sous-répertoires), et le nombre de fichiers ouverts simultanément. Dans la majorité des cas, les utilisateurs ne pourront pas atteindre ces limites.

Les limites se mettent en jeu quand vous avez plusieurs fichiers ouverts en même temps, ces fichiers étant localisés profondément dans différents sous-répertoires. Quoique des programmes appelant Malloc une des fonctions du système d'exploitation (allocation de mémoire) peuvent changer ces limites, plusieurs appels à cette fonction réduisent l'espace disponible prévu pour repérer ces fichiers ouverts et les sous-répertoires les contenant.

Techniquement parlant, les limites sont comme suit : Il y a 80 blocs dans l'OS Pool du système ; deux blocs sont utilisés par dossier "actif". Un dossier est "actif" s'il est le répertoire principal du périphérique le supportant, ou s'il contient des fichiers ouverts, ou s'il est le répertoire principal à quelqu'un, ou s'il a un "fils" "actif" (sous-répertoire). En fait c'est une définition récursive. Rappelez vous que chaque processus a un 'répertoire actuel' sur chaque périphérique logique, mais aussi rappelez vous qu'un dossier occupe seulement deux blocs même s'il est supposé être "actif" pour plusieurs processus.

Ajoutons qu'un fichier ouvert occupe un bloc, et qu'une tranche de mémoire (réservée ou libre) occupe un quart de bloc dans l'OS Pool.

L'amélioration par rapport aux précédentes ROMs est le suivant : l'ancienne définition de "actif" était "affiché". Demander une liste des fichiers dans un répertoire résulte dans le fait que tout les dossiers y figurant occuperont des blocs dans le Pool. Ajoutons que les blocs dans le "pool" n'étaient jamais libérés. Aussi bien qu'une fois que des parties du "pool" ont été utilisées pour gérer des tranches de mémoire Malloc, ces parties ne sont plus utilisables pour gérer des dossiers et vice versa. Toutes ces restrictions ont été enlevées.

Il est quand même toujours possible d'épuiser le Pool du OS. Le programme FOLDR100.PRG a été publié faisant partie de la distribution HDX (Utilitaires Disque Dur). Il ajoute de la mémoire à l'OS Pool et il fonctionne toujours avec le nouveau Pool. En plaçant ce programme dans le dossier AUTO il va ajouter 200 blocs à l'OS Pool, ce qui fait place à 100 dossiers supplémentaires (rappelez vous que seuls les dossiers actifs occupent de l'espace) ou à 800 tranches de mémoire ou toute autre combinaison.

Le nom FOLDR100.PRG peut être changé : les trois chiffres dans le nom indiquent le nombre de dossiers que vous voulez ajouter (deux blocs par dossier). FOLDR050.PRG ajoutera donc 100 blocs, alors que FOLDR200.PRG pourra en ajouter 400. Peu importe l'endroit duquel le programme est lancé, il se localise dans le dossier \AUTO\ du périphérique de

démarrage pour déterminer le nombre de blocs à ajouter.

Il est à préciser que ce programme n'est pas normalement nécessaire. Vous allez épuiser le Pool seulement si vous avez un nombre exceptionnel de fichiers ouverts et de profonds dossiers parce qu'il est géré maintenant d'une manière plus efficace.

En cas d'épuisement de Pool, le message suivant s'affichera sur votre écran :

*** OUT OF INTERNAL MEMORY ***
*** USE FOLDR100.PRG TO GET MORE

*** SYSTEM HALTED ***

(Ce message est en Anglais quelque soit le pays où vous êtes.)

Il est regrettable qu'on ne puisse rien faire lors d'un tel message sinon d'appuyer le bouton Reset ou faire un Reset clavier (avec CTRL-ALT-DELETE). Rappelez vous de ce que vous étiez en train de faire quand ceci s'est passé : étiez-vous en train de faire un répertoire qui a 50 niveaux de profondeur ? étiez-vous en train d'ouvrir le dixième fichier dans le dixième sous-répertoire ? Si vous voulez vraiment faire ce que vous étiez en train de faire, utilisez FOLDR100.PRG (ou bien augmentez le '100' si vous l'utilisez déjà).

Note : la routine Malloc du système ne paniquera jamais : elle retournera juste un zero comme résultat, indiquant qu'elle ne peut pas satisfaire la demande. Dans ce cas, malgré tout, vous saurez que votre programme a atteint les limites, car cela signifie qu'il ne reste plus même un quart de bloc pour gérer la mémoire. A ce point un programme bien conçu va détecter le manque de mémoire et terminer, libérant sufisamment de blocs pouvant servir pour d'autres programmes.

INFORMATIONS SUR LES BITS D'ATTRIBUT

Ce n'est pas comment il devait être ; c'est comment il est.

Bit d'attribut	Nom	Commentaires
0x01	Lire-Seulement	Interdit d'effacer, et d'ouvrir en écriture
0x02	Caché	Voir ci-dessous
0x04	Système	Voir ci-dessous
0x08	Etiquette-Volume	Exclusif (aucun autre bit ne doit être mis à un)
0x10	Sous-répertoire	Exclusif (aucun autre bit ne doit être mis à un)
0x20	Archive	Fichier nouveau ou a été modifié (ne marche pas avec l'ancien système, marche avec le nouveau : litéralement, "Fichier nouveau ou écrit dedans.")
0x40	RESERVE	
0X80	RESERVE	

L'attribut fichier 0x08 est exclusif : aucun autre bit ne doit être à un. Même chose avec 0x10. Ces restrictions ne sont pas protégées par le courant système d'exploitation mais ils le seront dans le prochain. Des fichiers ayant des attributs illégaux ne sont pas garantis pour leur bon fonctionement.

Ci-dessous une liste des régles appropriées aux attributs dans les recherches Fsfirst/Fsnext :

1 - Si attribut d'entrée == 0x08, inclure SEULEMENT si (attribut fichier == 0x08).
2 - Si attribut d'entrée &0x21 (archive ou Lire/Seulement), ou attribut fichier == 0, inclure.
3- Si ((attribut fichier) & (attribut d'entrée)) !=0, inclure.

Donc, si "ai" est l'attribut d'entrée et "fa" l'attribut fichier, correspondance si l'expression suivante est "VRAI" :

```
( (  !fa && (ia !=8)) || ((ia | 0x21) & fa ) )
      *          *            *          *
      A          B            C          D
```

(A) && (B) signifie pour fa aucun correspondant de recherche à l'exception de 0x08. (C) & (D) signifie un bit correspondant entre fa et ia, mais aussi que fa avec 'archive' ou 'lire seulement' vrais va trouver indépendamment un correspondant.

Ceci signifie que pour qu'un 'fichier caché' soit caché il ne peut pas être 'lire/seulement' ou 'archive'. Même chose pour un fichier système. Un fichier qui est en même temps 'caché' et 'sytème' (mais rien d'autre) va paraître quand l'un des deux bits est à un dans l'attribut d'entrée.

La 3ème règle signifie que les sous-répertoires sont inclus quand (entrée & 0x10) != 0. Même chose pour les étiquettes des volumes et (entrée & 0x08) != 0.

Vous pouvez créer avec Fcreate des fichiers avec toutes combinaisons des bits Archive, L/S, CACHE, et SYSTEME. Vous ne pouvez pas utiliser Fattrib pour changer en une combinaison illégale, et vous ne pouvez pas utiliser Fattrib, Frename, Fopen ou Fdelete sur des étiquette ou des sous-répertoires.

INFORMATIONS SUR Pexec()

Les paramètres corrects de Pexec sont (et ils étaient déjà) :

```
long    errcode=Pexec(0,prgfile,cmdline,envptr)  /* Charge & Exécute */
long    basepage=Pexec(3,prgfile,cmdline,envptr) /* Charge & n'exécute
                                                    pas */
long    errcode=Pexec(4,0L,basepage,0L)          /* Exécute seulement*/
long    basepage=Pexec(5,0L,cmdline,envptr)   /* créer juste basepage */

char    *prgfile; /* le fichier à charger */
char    *cmdline; /* ligne de commande ; longueur dans le 1° octet  */
char    *envptr;  /* 0L ou pointe à double-nul-termine environnement */
```

Si l'argument de envptr est 0L, le 'fils' hérite une copie de votre environnement. Voir les Informations sur la ligne d'environnement pour de plus amples explications.

L'argument de la 'ligne de commande' n'est pas une chaîne "C" normale. C'est un tampon caractères où le premier octet représente la taille du texte, et ce texte est aussi terminé par 'Nul'. Pexec copie ce tampon jusqu'au premier octet 'Nul' *OU* jusqu'à 126 caractères dans la nouvelle basepage. Les 'parsers' d'arguments utilisent l'octet de taille pour savoir où s'arrêter.

Pexec en mode 0 est le seul qui fonctionne efficacement. Les autres ont des problèmes avec l'appartenance de la mémoire et des fichiers et tout ce qui s'y rapporte ; Les utiliser seulement avec une attention particulière.

La ligne d'environnement passée à Pexec est définie comme une série de chaînes terminées par 'Nul'. Le format suggéré est le même que celui de MS-DOS et UNIX (UNIX est une marque déposée de AT & T Bell Labs.) :

```
NAME1=valeur1\0
NAME2=valeur2\0
...
NAMEn=valeurn\0\0
```

Ce qui est formel, cependant, c'est la terminaison double 'Nul' : la chaîne d'environnement est copiée jusqu'au premier double 'Nul'

Le BUREAU commence les programmes avec une chaîne d'environnement un peu différente :

```
PATH=0A:\00
```

C'est une faute, et elle ne doit pas être considérée comme le format d'environnement 'officiel'.

OBLIGER UN CHANGEMENT DE MEDIA

```
*-------------------------------------------------------------*
*                                                             *
*   mediach : cause un changement de media sur un périphériqueique *
*                                                             *
*   Utilisation :                                             *
*        errcode=mediach(devno);  /* donne 1 en cas d'erreur */ *
*        int errcode,devno;                                   *
*                                                             *
*                                                             *
*   Cette procédure cause un changement de media en installant un *
*   nouveau gestionnaire pour les vecteurs mediach, rwabs et getbpb *
*   sur le périphérique devno, le gestionnaire de mediach donne en *
*   réponse 'changé définitivement', et le gestionnaire de rwabs *
*   donne E_CHNG, jusqu'à ce que le nouveau gestionnaire de getbpb *
*   soit appelé. Le nouveau gestionnaire de getbpb désinstalle les *
*   nouveaux gestionnaires.                                   *
*                                                             *
*   Aprés avoir installé les nouveaux gestionnaires, cette procédure *
*   fait une opération disque quelconque (ex. ouvrir un fichier) ce *
*   qui fait que GEMDOS examine l'état du changement de média *
*   du lecteur :                                              *
*   ceci va amener les nouveaux gestionnaires de rwabs, mediach *
*   et getbpb à faire leur travail.                           *
*                                                             *
*   RESULTAT : 0 pour aucune erreur, 1 pour erreur (GEMDOS n'a *
*              jamais appelé getbpb ; dans le cas échéant.)   *
*                                                             *
*                                                             *
*-------------------------------------------------------------*
```

```
        .globl  _mediach
_mediach:
        move.w  4(sp),d0
        move.w  d0,mydev
        add.b   #'A',d0
        move.b  d0,fspec            ; spécifier les spec du
                                    ; lecteur pour première
                                    ; recherche
loop:
        clr.l   _(sp)               ; entre en mode super,
                                    ; laisse l'ancien ssp
        move.w  #$20,-(sp)          ; et le code de la fonction
                                    ; 'super' sur la pile
        trap    #1
        addq    #6,sp
        move.l  d0,-(sp)
        move.w  #$20,-(sp)

        move.l  $472,oldgetbpb
        move.l  $47e,oldmediach
        move.l  $476,oldrwabs
        move.l  #newgetbpb,$472
        move.l  #newmediach,$47e
```

```
            move.l    #newrwabs,$476
            ; Fopen ouvre un fichier sur ce lecteur

            move.w    #0,-(sp)
            move.l    #fspec,-(sp)
            move.w    #$3d,-(sp)
            trap      #1
            addq      #8,sp

            ; Fclose ferme le 'handle' qu'on vient d'obtenir

            tst.l     d0
            bmi.s     noclose

            move.w    d0,-(sp)
            move.w    #$3e,-(sp)
            trap      #1
            addq      #4,sp

noclose:
            moveq     #0,d7
            cmp.l     #newgetbpb,$472     ; encore installé ?
            bne.s     done                ;

            moveq     #1,d7          ; enlève et donne VRAI
            move.l    oldgetbpb,$472
            move.l    oldmediach,$47e
            move.l    oldrwabs,$476

done:       trap      #1                  ; retourne en mode
                                          ; utilisateur
            addq      #$6,sp              ; (utilise ce qui reste sur
                                          ; la pile en haut)
            move.l    d7,d0
            rts

*-------------------------------------------------------------*
*                                                             *
*   Nouveau getbpb : si c'est notre périphérique, désinstalle les   *
*   vecteurs ; Dans tous les cas appelle l'ancien vecteur getbpb    *
*   (pour vraiment l'obtenir)                                 *
*                                                             *
*-------------------------------------------------------------*
newgetbpb:
            move.w    mydev,d0
            cmp.w     4(sp),d0
            bne.s     dooldg
            move.l    oldgetbpb,$472 ; c'est le mien enlève les
                                     ; nouveaux vecteurs
            move.l    oldmediach,$47e
            move.l    oldrwabs,$476
dooldg:     move.l    oldgetbpb,a0   ; continuer ici toujours
            jmp       (a0)           ; appelle l'ancien
```

```
*-------------------------------------------------------------*
*                                                             *
*       Nouveau mediach: si c'est notre périphérique, donne 2 *
*                        sinon appelle l'ancien               *
*                                                             *
*-------------------------------------------------------------*

newmediach:
        move.w  mydev,d0
        cmp.w   4(sp),d0
        bne.s   dooldm
        moveq.l #2,d0           ; c'est le mien : donne 2
        rts                     ; (définitivement changé)

dooldm: move.l  oldmediach,a0   ; pas la mien : apelle l'ancien
        jmp     (a0)            ; vecteur

*-------------------------------------------------------------*
*                                                             *
*       newrwabs : donne E_CHG (-14) si c'est mon périphérique*
*                                                             *
*-------------------------------------------------------------*

newrwabs:
        move.w  mydev,d0
        cmp.w   $e(sp),d0
        bne.s   dooldr
        moveq.l #-14,d0
        rts

dooldr: move.l  oldmediach,a0
        jmp     (a0)

.data
fspec:  dc.b    "X:\\X",0       ; fichier à chercher (n'importe)

.bss
mydev:      ds.w    1
oldgetbpb:  ds.l    1
oldmediach: ds.l    1
oldrwabs:   ds.l    1

*-------------------------------------------------------------*
*                                                             *
*       Fin de mediach                                        *
*                                                             *
*-------------------------------------------------------------*
```

DIVERSES DOCUMENTATIONS

Les paramétres de Fdatime sont mal documentés : Le bon usage est :

```
Fdatime(timeptr,handle,wflag)
int *timeptr;   /* ptr à 2 ints */
int handle;     /* handle pour lire/écrire */
int wflag;      /* 1 pour écrire de timeptr au fichier
                   0 pour lire */
```

La méthode appropriée pour déterminer isatty est comme suit :

```
int isatty(handle)
int handle;
{
        long oldoffset;
        long rc;

        oldoffset=Fseek(0L,handle,1);
        rc=Fseek(1L,handle,0);
        Fseek(oldoffset,handle,0);
        if (rc==1) return 0;
}
```

Megamax C implémente isatty en utilisant Fdatime

Les codes des erreurs de GEMDOS sont mal documentés : ENMFIL est –49, et non pas –47, et ENSAME est –48.

Rsconf était documenté comme 'nul' mais il donne actuellement les anciennes valeurs des registres ucr, rsr, tsr, scr. Ils sont arrangés par octets suivant cet ordre (ordre du poids fort au poids faible) dans la valeur de retour 'LONG'. Ceci était toujours vrai mais n'a été ainsi documenté que maintenant.

AHDI ajoute des périphériques au système en commençant par le lecteur C , sans prendre en considération _drvbits et les lecteurs existants. Aussi, les ROMs (Mega) du 22/4/87 mettent à zero _drvbits lors d'un démarrage Chaud, contrairement aux ROMs du 20/11/85 de façon à ce que les RAMdisques éternels qui utilisaient ceci comme leur journal ne fonctionnaient plus pour cette raison (entre autre probablement).

Faire un Bconout vers l'imprimante donne 0 pour un echec et !0 pour une réussite. Ceci est utilisé pour le message d'erreur du Bureau "Votre périphérique de sortie ne répond pas" ; et il n'est pas traité par l'unité de traitement des erreurs critiques. Ceci avait été vrai pour toutes les ROMs. (GEMDOS donnait juste la valeur résiduelle dans D0, donc en utilisant la fonction Cprnout() du GEMDOS on a le même résultat. Les GEMDOS plus récents donnent explicitement l'état à partir de la routine du BIOS de Cprnout.)

La routine Super() est mal documentée : Super(1L) interroge le mode superviseur. Elle donne –1L si vous êtes en mode superviseur, et 0L dans le cas de mode utilisateur. [La documentation originale disait que Super(–1L) donnait 1L si on est en mode superviseur. En réalité,

Super(-1L) donne une erreur d'adresse.]
Super(0L) donne la dernière valeur du pointeur de pile superviseur et le processeur est mis en mode superviseur. Le paramétre 0L signifie "Utilise ma dernière pile utilisateur comme pile superviseur." Il y a normalement suffisamment de place sur la pile utilisateur à la fois pour vous et pour tous les interruptions qui pourraient parvenir.

La routine Super() est destinée à fonctionner comme suit :

```
extern long trap1();
#define Super(x) trap1(0x20,x)

super_sample()
{
    long    oldssp;

    /* passage en mode supeviseur */
    oldssp=Super(0L);

    /* Exécute en mode superviseur ; ssp est l'ancien usp.
       n'appelez pas AES! */

    /* revenir au mode utilisateur */
    Super(oldssp);
}
```

Il pourra y avoir quelques ambiguités en l'utilisant d'une autre manière : Faites attention. Appeler AES explosera partout parce que AES utilise usp pour sauver vos registres (même s'il est appelé du mode superviseur).

La documentaion XBIOS de la routine Setscreen énonce que la modification du point de base physique de l'écran ne prendra effet qu'à la suivante commande VBLANK. La réalité est qu'il le prend en considération immédiatement.

La variable système screenpt se comporte de telle manière que lorsqu'elle est nulle rien ne se passe, alors que quand elle est non nulle elle est introduite dans le système comme étant le point de base physique de l'écran au moment de VBLANK. (Etant donné que la routine VBLANK ne la remet pas à zero, elle continue à introduire la même valeur..pas de grand problème, mais une perte de temps.)

Si vous combinez ces deux idées pour changer le point de base physique de l'écran, vous trouverez qu'elles ne fonctionnent pas ensemble. Si vous en utilisez exclusivement une, elle va fonctionner comme elle l'a toujours fait.

Ne faites jamais Frename ou Fdelete a un fichier ouvert. Si vous faites Fdelete un fichier que quelqu'un d'autre a ouvert, ceci vous sera interdit. Si vous faites Fdelete sur un fichier que *Vous* avez ouvert vous même, GEMDOS va fermer le fichier puis l'effacer. *Malheureusement*, il y a un 'bug' : GEMDOS va fermer le fichier sans fermer son handle. Ce 'handle' va rester dans l'OS Pool, occupant une place pour toujours. Frename est pire : il n'essaie même pas de fermer le fichier ou d'interdire l'accés.

Le numéro de version est le meilleur moyen de verifier la version des ROMs. Il est au deuxième mot dans la ROM (ce qui est disons à un offset de $2 du début de l'en tête de l'OS et qui est pointé par _sysbase (au $4f2)).

Les ROMs originelle du 20/11/85 ont le numéro de version $0100. Les (Mega) ROMs du 22/4/87 ont le numéro de version $0102. Les dernières ROMs (TOS 192) ont le numéro de version $0104.

Vous pouvez verifier le numéro de version de GEMDOS spécifiquement en appelant sa routine Sversion, et vous pouvez consulter la date dans l'en tête de l'OS, mais le numéro de version est le meilleur pari parce qu'il est le même à travers tous les pays alors que la date ne l'est pas parfois.

Notes :

- Le registre A2 est employé maintenant dans presque toutes les fonctions du TOS. Sauvegardez le à chaque fois que vous devez l'utiliser

- Dans le nom d'un dossier ne jamais mettre les signes ? ou * car cela entraîne la création d'un dossier impossibile à détruire, à copier ou à de demander des informations dessus. Aussi, en mettant un * le système le transforme en points d'interrogations. En bref le dossier est permanent.

- Mêmes conséquences si par l'intermédiaire d'un certain logiciel vous créez un dossier et que vous lui mettez un espace entre deux lettres.

- Mêmes conséquences aussi pour le signe ':' en fin de nom de dossier sauf que l'Information fonctionne.

- Dans l'option 'Voir un fichier' du Bureau, si la ligne fait plus de 80 caractères le système n'affiche plus l'excédent à la ligne suivante.