Felix[1] Preliminary Plan

DRAFT Revised: 12/23/93 5:58 PM

NOTE: This document is early thinking and changing daily. Do not distribute w/o permission from EricSt or someone higher in the chain of command.. Send comments to EricSt.

[1] The Felix project is named after the cartoon cat with the same name. The name was chosen because like the nine-lives of a cat, MS-DOS has many lives (going on life number 7 with this product). MS-DOS is a product which just won't die...

Vision

The vision of the FELIX is to use MS-DOS BU expertise to produce a product which provides additional power and comfort in Chicago for knowledgable MS-DOS users. In addition, Felix will empower simple users to perform system maintenance by automating common system activities including defragmentation and on-disk archiving.

Anchor Features

This specification describes both anchor features and wish features. Anchor features are those features which are considered key to the success of Felix. Other features are nice to have and will be considered based on resources. The anchor features for Felix are:

- Enhanced file system which uses different compression to achieve higher (20% better) compression of infrequently used files. This feature, for now called *on-disk archiving*, is integrated into the file system and is transparent to the user. Files which are stored in this format are still useable, but reading may take longer. Archival and Removal are automated based on last access date.
- Batch scheduler with support for idle time scheduling
- Windowed COMMAND.COM (integrated with Explorer) with Autocorrect, Filename expanasion, Toolbar, TT Fonts, etc.
- Windows batch language (based on VBA) and integrated via OLE 2 with COMMAND.COM.

On-Disk Archiving

Felix will enhance the file system to provide for compressing files using better compression than DoubleSpace. Early results indicate this compression will typically be 20% better than standard DoubleSpace compression. At this point, we don't know if read time for files compressed in this format will be significantly impacted. Assumming read time is slower, only infrequently used files would be stored in this format. We also don't know how much memory the retrieval code will require. At this point, it appears we will not support reading of archived data from a downlevel OS.

☞ The format and approach for storing of archived files is open. At this point, it appears we will use DoubleSpace & only support archiving on DS volumes. A separate document, Archive Doc, which descrives the design options for this has been distributed for review.

GUI means of archiving files is via the file manager part of the Explorer. We will enhance FM to have an Archive option (in same places it has Copy), and to display an alternate icon for archived files.

☞ Need to talk with Explorer guys about how to integrate.

☞ What can we do to help user permanently remove old files from the disk. Perhaps we can build a backup set to automate backup and deletion from the drive?

HIGHLY CONFIDENTIAL

CMS 00010050

MS-PCA 1117005
CONFIDENTIAL

ARCHIVE Command Reference

The command line means of archiving is with the new ARCHIVE program. The ARCHIVE program archives files which match a specified file pattern (and optionally have not been accessed in a certain number of days). Archiving of files does not change the last access date. The tool is primarily intended to run by the batch scheduler when the system is idle. Preliminary specification for the utility, ARCHIVE, is:

ARCHIVE /UNATTENDED /RETRIEVE /S /LastAccess:*number [filepattern[/S]]*

Where /UNATTENDED specified utility is being run unattended and cannot ask for input. The /LastAccess switch is used to a criteria based on last access for selecting files. *laCriteria* is a number (which specifies any file not accessed in at least that number of days; 0 days indicates that no files are to be excluded based on LastAccess date. *filepattern* is an MS-DOS path/file pattern, such as C:\apps\*.*/S. /S is a modifier for the pattern which indicates all subdirectories of that file as well. Multiple patterns can be specified. /RETRIEVE specifies that the files are to be retrieved from ARCHIVE.

ARCHIVE.INI

An ARCHIVE.INI file (or perhaps we will use Registry) will be used by archive to determine default settings. This file can be used specify specific files (including file patterns) to never archive.

☞ Need to discuss patent options with lawyers.

BATCHMGR: Scheduler with support for launching programs when the system is idle

BATCHMGR is a batch job queue. It can schedule jobs to run at specific times, at system startup/shutdown, or around specific times when the system is idle. It can also schedule recurring jobs (daily, monthly, etc.). The default BATCHMGR configuration will have recurring jobs to defragment the disk, to analyze the disk, and peform on-disk archival, and to retrieve recently used files from archival.

BATCHMGR also supports an API for BATCHMGR aware programs. The API is described below.

BATCHMGR Processing Narrative

The basic scenario is as follows:

1. BATCHMGR is started (by default it is in the startup group)
2. BATCHMGR creates a status window.
3. BATCHMGR checks its queue and determines a job is to be run.
4. A message is displayed in the status window saying the job is being executed and the time it is started. If the system is idle, the status window is forced to be on top[2]. A message is also appended to the BATCHMGR log file indicated the job is being started.
5. The job is started. If the job is a batch file or old app, it is executed with WINCOM. Else it is executed with the WIN32 exec call which returns exit codes. If the user has specified the job should be removed from the queue when it is started, the job is removed.
6. As the job proceeds, it can update the status information with the BMStatusMsg call.
7. When the job terminates, BATCHMGR adds a status message to the message window reporting when the job exited and the completion state (success, interupted, error, etc.).

---

[2] Idea is that if user returns while job is executing, they will see the status.

CMS 00010061

HIGHLY CONFIDENTIAL

MS-PCA 1117006
CONFIDENTIAL

8. Job entry is removed from the Queue if it is still in the queue.
9. A message is written to the status window indicating that the job has completed. A message is also written to the BATCHMGR log file indicating time and status of the job completion

BATCHMGR jobs can terminate in any of the following ways:

- Normal completion. This is the status for jobs which run to completion and encounter no errors.
- Interupted by user. The user can interupt jobs at anytime
- Error condition. The job encountered some error condition and exited with an error code.

☞ How are jobs interupted? Terminate button on status window? mouse movement/keyboard usage? User takes app specific action? Combination of these depending on the app? How is the return code reported back to BATCHMGR (Ralphl says the win32 exec API handles this, but we need to see if it returns a code for 16bit apps)?

BATCHMGR starts each job with the WIN32 execute API which supports a return code. By default, return codes in certain ranges will indicate success and other ranges will indicate a serious problem (the ranges can be overidden for any specific job).

Whenever a BATCHMGR job is executing, a status window is displayed. For unaware jobs, the status will list the job and the time it was started. Jobs can add their own information to the status window at anytime using the BMStatusMsg API.

BATCHMGR maintains a status window at all times. By default, this Window is hidden when the user is actively using the system and is always on top when a scheduled job is being run or when the queue has completed and the user hasn't returned to the system yet. For unaware applications, the window will show that the application is being run until the application completes[3], at which point the window will show that the application completed. BATCHMGR will provide an API for applications to report progress in the status window. By default, BATCHMGR will keep a log of which jobs were run and the completion status (completed, interupted, exited with error, etc). In addition, jobs can specify via the BATCHMGR API that additional information be added to the log.

BATCHMGR API

BATCHMGR provides an API (OLE methods that Felix batch files can call) to display messages in the status window and the log file, and to add and remove entries from the batch queue. The API are listed below

BMStatusMsg	Adds a status message to the BATCHMGR status window.
BMAddLogEntry	Adds an entry to the BATCHMGR log file.
BMVersion	Reports the version of BATCHMGR.
BMQueryQueue	Reports on jobs scheduled in the queue.
BMDeleteJob	Deletes a job from the BATCHMGR queue
BMAddJob	Adds a job to the BATCHMGR queue.

HIGHLY CONFIDENTIAL

[3] The job can be set to echo TTY output to the status window.

CMS 00010052

MS-PCA 1117007
CONFIDENTIAL

BATCHMGR Queue And Job Configuration Options

The following table lists the BATCHMGR job configuration options.

Wait until system is idle	This is a drop-down list, user can choose from # of minutes of idleness, or to wait until off-hours.
When to execute	This can be on Startup, Shutdown, or a specific date/time.
Recurring Jobs (Hourly, Daily, Weekly, Bi-Weekly, Monthly, yearly).	This will be similar to the Schedule+ UI for recurring appointments (hopefully we can use their code for the UI).
Priority	User can specify a priority for the task. Tasks with higher priority (eg 1 is higher than 2) are run first.
Run exclusively	If this is checked, this job is started when no other jobs are running and no other jobs are started until this one is finished. For example, defrag would be run this way.
Restart if interupted	This setting lets you specify whether the queue entry is removed upon initial execution of the program,or upon sucessful completion of the program. Default is the later. This way if an app gets interupted by the user returning to the system, it will automatically be restarted when the system is idle again. The setting to remove upon execution is for jobs which cannot be restarted once they're Initially started. Jobs which exit with error conditions are always removed from the queue.
Suspend Queue on Error Termination	If this job terminates with an error condition, do not process anymore jobs until the user restarts the queue.
Suspend Recurring Job on Error Termination	If this is set, no future occurrences of this job will be added to the queue until the user restarts the recurring job.
Log TTY output to file	TTY output from batch jobs can be ignored, saved to the BATCHMGR log file or saved to a separate user specified file, and/or echoed to the status window.

The following table lists BATCHMGR Queue configuration options

Work Hours	These are the hours the system is typically in use[4]
View Log File	This option is used to view the log created by BATCHMGR. It includes a history of which programs were run when along with the status of each job.
Purge BATCHMGR log file	BATCHMGR can purge information from the log file when it reaches a certain age. The log file can also be disabled entirely,

[4] Can we hook into MAPI to figure out work hours?

and data can be purged upon successful job completion.

Force Idle Now	This button causes BATCHMGR to proceed as though the system had been idle for a long time.

Jobs which are run at idle-time should be interuptable. BATCHMGR will notice when the system is not idle and will signal the idle-time job to terminate.

☞ How does BATCHMGR know system is no longer idle? How is the application interupted? Do we want to support interupting one Job and continuing another?

Default BATCHMGR Job Entries

By default, ARCHIVE is scheduled be run daily during off-hours to archive data which hasn't been used in one month. Defrag is run weekly during off-hours. Disk Analysis (structure check - not surface scan) is run daily when the system has been idle for 30 minutes. Surface scan is run monthly during off-hours.

☞ What happens when disk analysis determines there is a problem? Need to define a way for user to get notified.

☞ Need to define BATCHMGR API. Initial thinking is that we should have API to 1) query if being run by BATCHMGR, 2) register a call-back so that BATCHMGR can tell the app to terminate when the user returns, 3) a means for the app to record information in a log file, and 4) a means for the program to add entries to the batch queue.

Windowed Command.com (Enhanced WinOldApp)

Windowed command COMMAND COM is attached to the explorer. All MS-DOS commands which can be run from full-screen VM's can be run from within the window. If the command goes full-screen, it takes over the current window (unless the full-screen app is run with the start command in which case a new window is created). Much improved performance (over Win 3.1 WINOLDAP) is achieved by using a modified COMMAND.COM which knows when it is being run to support the new WINCOM and communicates directly with it instead of using INT 10 to the console.

From Wincom, you will be able to access the current selection in the explorer. Ralphl proposed we use ** to designate the current selection in the explorer (also, for command history we can safe cmd ** as opposed to trying to save a detailed list of the files which were selected at the time of the command). For example, you could say del ** to delete the current selection. You could say dir *.* to select everything, then use the mouse to deselect certain things, then use del ** to delete the remaining selections. We should have a keyboard deselect command also (perhaps *drop * C* to drop all .C files from the selection; or *drop *** to reset the selection to nothing) and a means of adding to the selection from the keyboard (perhaps *add *.asm* to add all *.asm files)

Key Features in WinCOM
- WinOldAp/COMMAND.COM integration
- Integration with Explorer
- AutoCorrect of Commands
- SmartCD (dir implies CD; autocorrect of path name)
- Filename completion
- Graft function to move entire dir branches
- Toolbar (customizable) including features for buttons
- TrueType fonts

HIGHLY CONFIDENTIAL

CMS 00010064

MS-PCA 1117009
CONFIDENTIAL

- Startup Batch File (similar to command /K: switch)
- Send mail from command line

Other features under consideration:
- Longer Path/Env Editor
- UNDO command. We could undo many of the commands including delete, copy, Rename, CD, MD.

☞ What buttons should we put on the toolbar? Ideas include: dir history, program history, command history, find file, delete, copy, move/rename, archive, print, mail, user defineable (for example, VBA procedures).

☞ Ralphl suggested we have the editor support editing the environment directly, as in edit /env, or an option on file open.

Enhanced batch language with ability to call OLE-2 object methods

Felix command.com will be an OLE object with the ability to call other OLE automation objects. This will enable batch file use of VBA commands which are exposed as OLE-2 Methods. In addition, command.com will expose its internal commands/environment as OLE objects so that OLE aware software (like VB) can use them. Felix will not include any means of creating OLE 2 objects - users will need to buy alternate software (like VB) to do that. We will provide a VBA library which exposes VB commands (as OLE automation objects) so that batch files can call them with command.com. We will also add some new commands to the library we ship so that batch files can present a dialog and get input for example.

We will also consider the following features to enhance batch processing
- Redirection of ERROR output
- Better control of free environment space

☞ Need to define the common user scenarios for this. One scenario is a startup menu. Another scenario is periodic backup of specific files.

☞ Cairo is also interested in doing this. David Stutz is the contact to keep informed about our ideas in this area

The contacts for this work are ScotP - Ole 1; Craig symmonds (Prog Mgr for VBA 1/2); Scott Wiltamuth (Prog Mgr for Ole automation).

WinCom will be an OLE Object and will expose commands to VB via OLE methods. VBA would allow you to connect to objects and use its methods/data.

Changes to COMMAND.COM

The COMMAND.COM call command will be enhanced to support calling any method on an OLE object. Since CALL is already an internal command, there is no chance of name conflict with existing programs. Since the existing call command syntax is easily supported, this approach will allow us to achieve 100% compatibility with existing batch files. According to Eric Cantel (VBA) the Type library has a means of getting at methods provided by basic objects. We'd need to have a means of connecting to this library. With this approach we can also call OLE methods which are exposed by MS applications including Word and Excel.

HIGHLY CONFIDENTIAL

CMS 00010066

MS-PCA 1117010
CONFIDENTIAL

User Scenarios

We need to think through the ways users will use this feature. We should also ship some samples. One idea is to include a batch file which backup key system and app configuration files (system.ini, AMIPRO.INI, NORMAL.DOT, etc.[5]) We could include GUI means of adding files to the list. Another idea - we could include the file find engine we did for Jaguar and have a cool GUI batch file front end for it.

We can also have a contest amongst our beta testers to come up with the coolest sample batch file.

Additional Files we need to Ship for VBA Support

We will ship a file which contains a library of VBA functions with OLE methods batch files can access We will also include a library with some generic dialog boxes for use in batch files. This library will be implemented as an OLE automation object (which can be written in VB or VC).

We could call any OLE automation objects including Excel/Word methods for example.

Other features we may consider

The following ideas are here for tracking. At this point, none of them are planned for Felix.

Improved error messages

Death of Abort, Retry, Fail - this could go a long ways with the press.

Diamond

We could ship the diamond tool for users to use when moving files between home and office.

Intellisense

(from Excel 5) Watch how you use the OS and provide hot tips (like Excel 5)

Super Copy command

Based on DCOPY2 tool developed by MS Subs.

supports all xcopy features

has the key features of replace build in

supports very complex source specifications

supports multiple destinations

[5] January 1994 issue of PC Computing, page 236, lists the critical files for some popular apps (AMI pro, Quatro, Excel, etc.)

moves files

supports file exclusion

allows nearly Error free copy processes over the Microsoft network

offers you a DOS box - in the copy process- if you need to

format new disks or things like that

optimized filling of destinations to minimize left free disk space on multiple volumes

it is Smartdrive aware

extended confirmation options

extended file attribute management

Sequential Screen Saver

Sequential Savers. This steals from WinWord 6.0's tip of the day feature. The idea is that the screen saver displays a new saver each time, until it reaches the end, then it recycles, or goes to a different set. We could offer some fun stuff (in liew of tips) with this (serial cartoon for example), and we could position it as an MIS feature (educate your users,etc.).

Maid disk cleanup utility

Blows away unnecessary files - biggest concern with this is if we kill the wrong files. It would be based on hueristics· get rid of .tmps, duplicates, last access date, duplicate functionality, old unnecessary files, etc. Could include Last access date and viewers for common data formats. Database of files with software suites.

If we do this, we should include a method for Apps to support being cleaned up.

Application Uninstall

This would be difficult to write (perhaps we could buy it). Would be a real favorite amongst the press (who are constantly installing/uninstalling apps).

File Find Engine

Could include the file find engine originally implemented for jaguar, possibly with a batch file GUI front end.

Fast Floppy driver (speeds floppy up 50%)

Chicago has a fast floppy driver which we could pull from Chicago and include in Felix.

Windows MemMaker

We have considered having a Windows memmaker, but only if Helix would do the work. At this point it doesn't look like they're interested, so it is probably not in the product. GUI (with character mode CHKSTATE.SYS) - run directly from Explorer. Use Chicago registry to see what is installed and what memory is available. Support long filenames. Cognizant of real-mode drivers (dblspace, smartdrv) which go away when Chicago transitions to protect mode. Support multi-config