Inter-Office Memorandum

To	Mesa Users	Date	July 11, 1980
From	Richard Johnsson	Location	Palo Alto
Subject	**Mesa 6.0u Change Summary**	Organization	SDD/SS/Mesa

XEROX

Filed on: [Igor]<AlphaMesa>Doc>Summary60u.bravo

This memo outlines changes made in Mesa since the last alpha update (Mesa 60.t, June 11, 1980). It contains changes from the documents contained in that update (those documents are also being updated to correctly reflect the differences between Mesa 5.0 and Mesa 6.0). As Mesa60u.press this document also includes documentation of two new Debugger tools and a list of ARs fixed since 6.0t.

Language

The language extensions to Mesa 5.0 have been modified in the following areas:

DIRECTORY clauses have an additional form for dealing with parameterization.

Uncounted zones have been modified.

The type specification StringBody[n] describes a STRING of n characters (n need not be a constant); it can be only used with NEW and SIZE.

The Mesa 6.0 Compiler Update has been updated with complete descriptions of these changes.

System

All of the system interfaces have been recompiled for 6.0u. This means that all modules must be recompiled/rebound to use the new Mesa.image. No source changes are necessary. Several new interface items and defaults were added. Additions include:

ForgotDefs: all item are now in SegmentDefs and copied into ForgotDefs for temporary compatibility. You should remove references to ForgotDefs as soon as possible.

FrameDefs.LoadConfig added.

Process/ProcessDefs.Aborted = ABORTED.

SegmentDefs.DefaultBaseN (N IN [0..3]) added.

Debugger

In addition to many bug fixes and some performance improvements, the following changes or additions have been made:

`AScii Display [address, count]` is equivalent to interpreting `LOOPHOLE[address, POINTER TO PACKED ARRAY OF CHARACTER][0!count]`.

`ATtach Condition [BreakPoint#, Condition]` changes a normal breakpoint into a conditional breakpoint. Restrictions on conditions are as in the Mesa 5.0 debugger. The interpreter evaluates conditions at attach time in the context of the breakpoint.

`ATtach Keystrokes` replaces `ATtach Expression`.

`ATtach Loadstate [imageFile]` attaches the initial loadstate of an image file rather than the current loadstate. This command is for wizards only. Don't worry if you don't understand.

`CLear Break [BreakPoint#]` clears breakpoints by number. Typing <return> instead of a number will clear the current breakpoint, i.e. the one that got you into the debugger.

`CLear Condition [BreakPoint#]` changes a conditional breakpoint into a normal breakpoint. Typing <return> acts the same as in Clear Break.

`CLear Keystrokes [BreakPoint#]` undoes `ATtach Keystrokes.` Typing <return> acts the same as in Clear Break.

`--` comments can now be entered in display stack and display process modes.

Type `REAL` is supported for output only. You cannot type in or do arithmetic on `REAL`s.

Variables declared in nested blocks are shown indented according to the nesting level in display stack mode.

"?" in a variable display now uniformly means that the value is out of range; "`. . .`" is used to indicate that there are additional fields which cannot be displayed due to lack of symbols.

Debugger User Interface

The method for invoking scrollbars remains the same but, the scrollbars themselves are twice as wide and you can "see through" them.

The keyset edit functions are now also available on the keyboard as control characters:

Ctrl-C	-- Cut
Ctrl-F	-- Paste
Ctrl-K	-- Replace/Next
Ctrl-N	-- Next
Ctrl-R	-- Replace
Ctrl-S	-- Stuff

Note that edit functions are available only if the window is *editable*.

Reporting Debugger Problems

There is a new command in "`//`" mode to aid in collecting data for reporting debugger problems:

> `Trace Stack` dumps the debugger's call stack in octal.

ARs reporting debugger problems that result in uncaught signals or entering "`//`" mode for any other reason should be accompanied by a debug log which includes the output of this command.

Distribution:
- Mesa Users
- Mesa Group
- SDSupport

10.0 File Tool

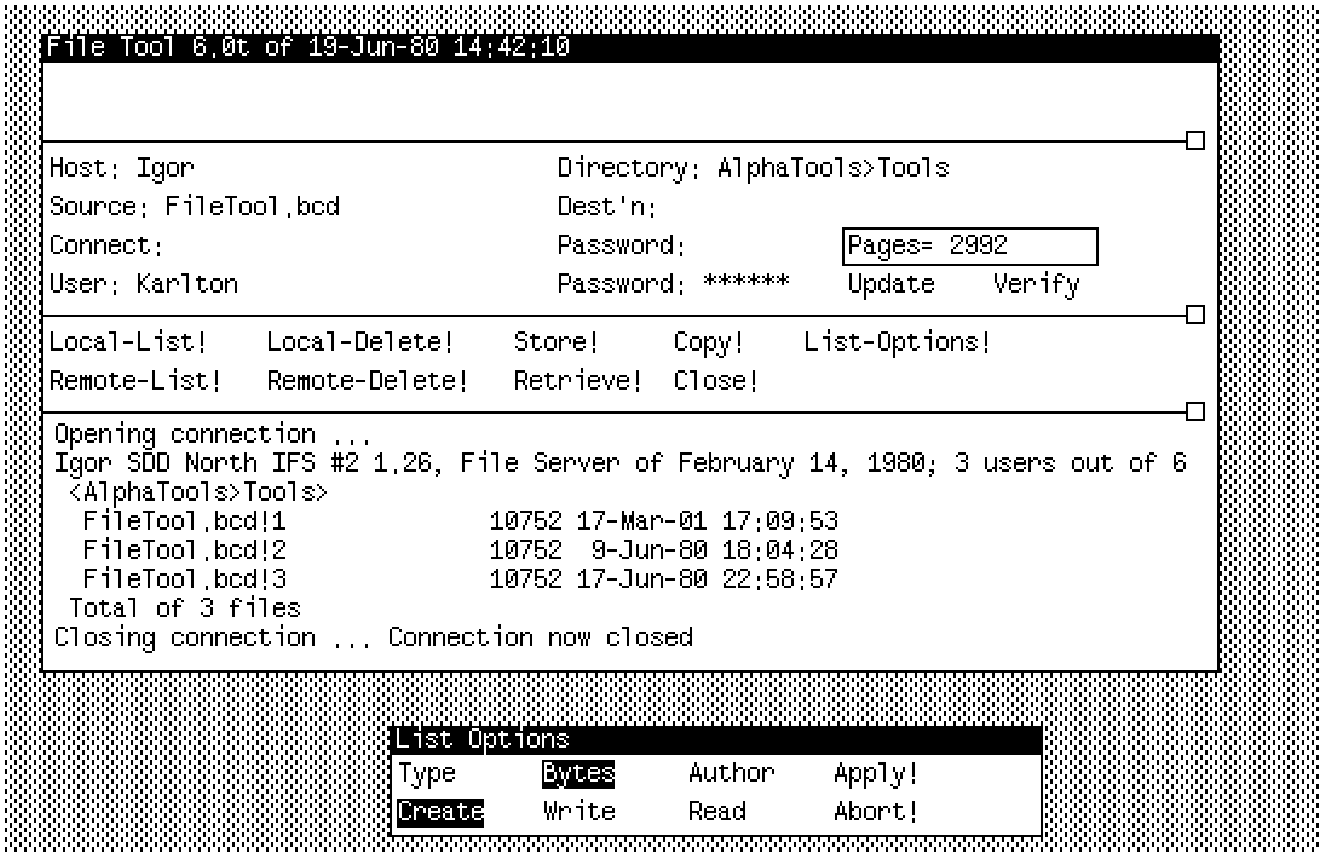
The File Tool provides a means of dealing with the files on the local disk as well as remote file systems from within the Development Environment.

10.1 The User Illusion

The File Tool employs the standard features of the Development Environment. See section 3 for further details.

10.2 Tool Appearance

Below is an illustration of a File Tool with the List Options window (explained below) also available.



```
File Tool 6.0t of 19-Jun-80 14:42:10

Host: Igor                          Directory: AlphaTools>Tools
Source: FileTool.bcd                Dest'n:
Connect:                            Password:          Pages= 2992
User: Karlton                       Password: ******   Update    Verify

Local-List!    Local-Delete!    Store!     Copy!    List-Options!
Remote-List!   Remote-Delete!   Retrieve!  Close!

Opening connection ...
Igor SDD North IFS #2 1.26, File Server of February 14, 1980; 3 users out of 6
 <AlphaTools>Tools>
  FileTool.bcd!1            10752 17-Mar-01 17:09:53
  FileTool.bcd!2            10752  9-Jun-80 18:04:28
  FileTool.bcd!3            10752 17-Jun-80 22:58:57
 Total of 3 files
Closing connection ... Connection now closed

List Options
Type      Bytes     Author    Apply!
Create    Write     Read      Abort!
```

10.3 Parameter Subwindow

In the upper form window, the parameters that will be used by the next command should be filled in by the user.

Host: the name of the host to be used for remote files and operations. If a connection is already open, any editing of this field causes it to be closed.

Directory: the default remote directory.

Source: a list of files (separated by spaces or returns) for the next command to act upon. If the first character of a file name is "@", then the file is taken to be an indirect file and its contents are used as a list of files. Indirect files may nest.

Dest'n: file name for the destination of a transfer. If this field is left blank, then the file name is the same as the source.

Connect:, Password: the secondary directory and the associated password.

Pages= number of free pages left on the disk. This item is read only.

User:, Password: the primary directory and the associated password. This field is initialized from the value of the user's last Alto Operating System login. Editing of this field is local to the File Tool and does *not* affect the user's login in the Alto Operating System.

Verify request confirmation for each file transfer. The default is false.

Update only store or retrieve the file if the source is newer than the destination (comparing creation dates). The default is false.

10.4 Command Subwindow

In the second form window, the commands are available. Some of the commands are accomplished by a background process. Those commands clear the the Command subwindow so that a second transfer cannot be invoked while one is under way. The Copy! command operates only on the local disk. It does not take a list of files to operate upon. Close! closes a remote connection (if there is one), thereby freeing resources and earning you the kudos of your peers.

It is important to remember that the commands are postfix, e.g., fill in the Host: and Source: fileds before invoking the Retrieve! command. The following commands are available:

Local-List! lists all files on the local disk corresponding to the name in Source:. *s and #s are expanded.

Local-Delete! deletes the files specified in Source: from the local disk, regardless of whether the file is in use. If it finds it impossible to delete the file due to *some debugger pointers that would be left dangling*, it does not allow you to do so (e.g., you cannot delete XDebug.image). *Beware of your own references!*

Store! transfers the filename specified in Source: from the local disk to the remote `Host`. Alto filename conventions apply to the local file.

Copy! makes a copy of a file on the local disk to the local disk. Only a single file may be copied and *s and #s are not honored.

List-Options! creates a List Options window if one does not already exist.

Remote-List! lists all files on the remote file system corresponding to the name in `Source:`. This must conform to the file naming conventions on the remote host. You may designate multiple files by the use of * expansion only to the extent that the remote server supports it (currently Maxc and IFS do, but differently).

Remote-Delete! deletes the file name specified in Source: from the remote file system. You may designate multiple files by the use of * expansion only to the extent that the remote server supports it.

Retrieve! transfers the filename specified in Source: from the remote file system to the local disk. The filename must conform to the file-naming conventions on the remote host. You may designate multiple files by the use of * expansion only to the extent that the remote server supports it.

Close! closes the currently open FTP connection.

If Verify is TRUE, then for each file that might be transferred, the following commands are displayed

Confirm! do the operation.

Deny! don't do the operation.

Stop! don't do the operation and terminate the command. This may take some time while the termination is negotiated with the server.

10.5 List Options window

The List Options window is created by the List-Options! command. The properties that will be displayed, in addition to the file name, by a Local-List! or Remote-List! are governed by the booleans in this window. After changing the options to be shown use Apply! to effect those changes. The Abort! command will restore the options to what they were before the List-Options! command was chosen. Choosing either of the commands in the List Options window will cause that window to be removed.

10.6 Exceptions

The actual transfer takes place in a background process so the user is free to issue other commands or even change the values in the parameter subwindow without affecting the command currently executing. Changing a parameter while the FileTool is waiting for Confirm! will **not** affect the name of the destination file; you should abort the transfer and reissue the command with the desired parameter already set.

Warning: If you are using the FileTool from inside the debugger, be careful not to change any files out from under the program you are debugging; the debugger makes no provisions for checking this when you delete or update a local file!

Inter-Office Memorandum

To	Mesa Users	Date	July 11, 1980
From	Jim Sandman, John Wick	Location	Palo Alto
Subject	**Integrated Mesa Environment**	Organization	SDD/SS/Mesa

XEROX

Filed on: [Iris]<Mesa>Doc>CommandCentral.bravo (and .press)

DRAFT

This memo documents a small executive called Command Central; this Tool is intended to be installed with the Debugger and can be used to invoke the Compiler, the Binder, and client programs, all of which upon completion are directed to return to Command Central rather than to the Alto Executive. The idea is that, while programming in Mesa, you enter Command Central's control only once, and you rarely have to leave it; this is made possible by the editor that is now included in the Debugger, as well as by the context switching facilities provided by Command Central.

Installation

To include Command Central in the Debugger, type the following Alto Executive command when installing, after retrieving `<Mesa>PupAndFtp.bcd`, `<Mesa>FileTool.bcd`, and `<Mesa>Utilities>CommandCentral.bcd`. (If you have more than 64K of memory, be sure to consult the Installation section of the Debugger documentation before proceeding.)

```
>XDebug PupAndFtp/l FileTool/l CommandCentral
```

While it is possible to use Command Central without also installing the **FileTool**, including it will help minimize the number of times you have to leave the Mesa environment. If you have enough memory on your machine, you might consider installing other Tools with your Debugger as well (e.g., **Chat**, **SndMsg**).

Entering Command Central

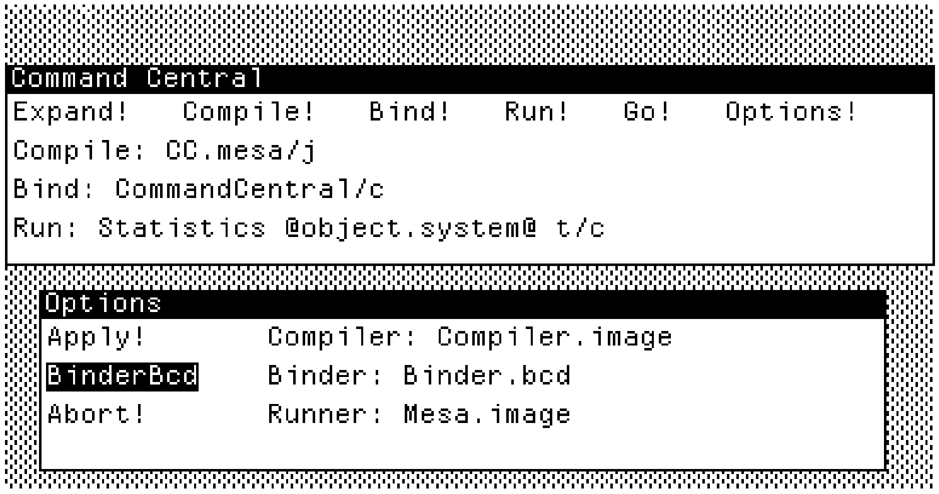
When using Command Central, the Debugger becomes the executive from which all programs are invoked. To first enter this environment, type

```
>Mesa/d
```

to the Alto Executive. You can now use the **FileTool** to retrieve the modules you wish to work on and the Tools editor to modify them. When you have finished your changes, turn your attention to the Command Central window.

Command Central Window

This window implements three fields and five commands; it also supports the standard window operations (scrolling, growing, etc.).


```
Command Central
Expand!   Compile!   Bind!   Run!   Go!   Options!
Compile: CC.mesa/j
Bind: CommandCentral/c
Run: Statistics @object.system@ t/c

Options
Apply!       Compiler: Compiler.image
BinderBcd    Binder: Binder.bcd
Abort!       Runner: Mesa.image
```

The three fields contain command lines for the Compiler, the Binder, and the System; their contents are written to `Com.cm` when the commands are invoked. (A special global switch /q is added so that control is returned to the Debugger rather than to the Alto Executive.)

The `Compile!`, `Bind!`, and `Run!` commands invoke the appropriate programs using the command lines constucted from the `Compile:`, `Bind:`, and `Run:` fields, respectively. The `Run!` command can be used to invoke `.bcd`, `.image`, and `.run` files (see below). The `Go!` command constructs a combined command line using all non-null fields and executes the appropriate programs in order.

The parameter fields also recognize command files preceeded by the traditional at-sign (e.g., `@file.cm`); the `Expand!` command will expand all such references into their contents and write the result back into the window. Indirect references are also automatically expanded when any of the other commands are invoked.

The `Options!` command produces the options window which allows you to tailor your environment by supplying the names of the compiler, binder and runner. If the binder is not a .bcd, the boolean item `BinderBcd` should be turned off. The `Apply!` command will save the new names and the `Abort!` command will restore the names to their previous state. Both commands will remove the options window.

If the Compiler or the Binder detect errors (and the pause switch is in effect), they will invoke the Debugger with an appropriate message instead of pausing. You can then load the appropriate error log into a window and step through it and your source file together. Because the file index of the error is included in each message, the `position` menu command can be used to find the source of the error quickly.

Invoking Other Programs

Any Mesa `.bcd` which expects to be loaded into `Mesa.image` and obtains its commands from the command line (`Com.cm`) can be invoked by Command Central using the `Run:` field and the `Run!` command. (As above, the global /q switch is added to the command line so that control will return to the Debugger.) Some obvious programs which you might include on your disk are **Access** and **Print**.

You can also run arbitrary `.image` and `.run` files using Command Central, but unless they have made provision to return control to the Debugger, they will exit to the Alto Executive upon completion. Use the `Mesa/d` command to reenter Command Central.

Limitations

If you use the `Compile!`, `Bind!`, `Run!`, or `Go!` commands when you are in the middle of a debugging session (at a breakpoint or an uncaught signal, for example), the state of the client will be lost. In particular, normal termination processing of the client will not take place (e.g., open files will be left dangling).

Distribution:
Mesa Users
Mesa Group
SDSupport

```
Numb  Originator     Subsystem      Subject
er

4038  Evans          Binder         Lock released once too often when zap target bcd
4502  BLyon          Binder         'IMPORT' instead of 'IMPORTS' in a CONFIG blew the binder up
4514  Knutsen        Binder         Copying code to bcd w/ "code" gives "Ref'd in diff versions"
4593  Luniewski      Binder         Output files incorrectly chosen
4658  Johnsson       Binder         Command line '; eats next char
4765  Newman         Binder         "Foo.run/r" puts "Foo.run/" in Com.cm
4766  Newman         Binder         "Foo.image/r" doesn't work
4315  Swinehart      Compiler       Variant record selection expression fatal
4501  Gobbel         Compiler       Order in text affects defaults
4537  McJones        Compiler       Zero-size field in M.D. record with explicit field positions
4547  Levin          Compiler       Bad FGT entry
4616  birrell        Compiler       Missing closing quote error message not very informative
4617  McJones        Compiler       Exported types as procedure parameters
4627  BLyon          Compiler       EXPORTED TYPES => INCORRECT TYPE errors in three or four
4632  MBrown         Compiler       LONG DESCRIPTOR FOR ARRAY OF longDescriptorRec
4637  McJones        Compiler       Missing . in subrange constructor gives ERROR in Pass 3
4670  McJones        Compiler       Wrong SIZE of M.D. record with explicit bit positions
4731  birrell        Compiler       Type mis-match on initialising to imported variable
4760  Newman         Compiler       "Foo.run/r" puts "Foo.run/" in Com.cm
4780  ayers          Compiler       Catch phrase in imported INLINE gives pass 4 error
2052  McJones        Debugger       Doc: Return Values vs Local Variables
2336  Hamilton       Debugger       Can't Comment in Display Stack Mode
2431  Schwartz       Debugger       Doc: One dash (rather than two) defines a comment.
3648  Levin          Debugger       Attach Image to use current loadstate
4160  LNelson        Debugger       Condition breaks vs subranges
4419  Murray         Debugger       ARRAY [0..1000) OF WORD
4421  Murray         Debugger       very long STRING printout
4437  Murray         Debugger       process not bound
4518  Hamilton       Debugger       Leaves Display Stack mode unasked
4564  Kayashima      Debugger       Xdebug Hangs when Move or Grow sized window
4575  mbrown         Debugger       infinite chain of uncaught SIGNAL 10601B, msg=177777B
4576  Sweet          Debugger       Nested blocks in main body
4614  Kayashima      Debugger       Stripping Bravo Trailers
4619  birrell        Debugger       source windows: length increase not displayed
4633  mbrown         Debugger       Confusion about contexts
4672  Gobbel         Debugger       Debugger bitmap goes away sometimes if Edit is used.
4680  morris         Debugger       Debugger crash
4739  Levin          Debugger       Internal debugger won't install
4743  Levin          Debugger       Lockup while repainting windows
4745  Murray         Debugger       Can't install Internal Debugger
4758  Newman         Debugger       "ATtach TtachSymbols"
4761  Newman         Debugger       Misleading error message from Display Frame
4777  Murray         Debugger       context mixup
4781  ayers          Debugger       missing source during DisplayStack
4783  ayers          Debugger       Debuggers erxtra-memory bitmap space
4789  ayers          Debugger       <alphamesa>temp>xdebug.image Intall problems
4804  Levin          Debugger       Debugger bootloading: NIL PuntInfo gives PointerFault
4824  Newman         Debugger       Interpreter doesn't know size of UNSPECIFIED
4844  Newman         Debugger       ATtach Conditon command rejected
2906  israel         Ether          Pup: FTP: Slow on Dorado
3626  Murray         Ether          FTP: sending mail
3900  Murray         Ether          FTP: UNWIND from FTPEnumerate/retrieve
4352  Karlton        Ether          FTP: FTPAltoFile.PreProcessFile does a blind ReleaseFile
4456  Birrell        Ether          Pup: FTP: Recompile packages to fix long return record bug
4499  Murray         Ether          FTP: FTPTransferFile doesn't pass through the creation date
4544  birrell        Ether          FTP: FTPRetrieve hangs until timeout on a "no" mark
4552  Birrell        Ether          Pup: ByteStream timeout
725   White          System         Pause primitive
2007  Murray         System         Image Files Poisoned With Bad FP
3061  israel         System         Nil and bounds checking
3129  Olmstead       System         Request for info
3619  Levin          System         Bootmesa hack
3661  Birrell        System         Aborting process in ReadChar leaves keyboard monitor locked
3668  Birrell        System         XMesa Nucleus: determining memory size
3751  Wyatt          System         re: MakeImage problem (see previous AR)
4108  Levin          System         More processes and GFT slots in Mesa.image
4218  Levin          System         RunConfig: Return global frame of the control module loaded
4244  Levin          System         Loader Bug
4245  Levin          System         UnNewConfig bug
4267  Levin          System         UnNewConfig bugs: FindNLinks bugs
4410  Schmidt.PA     System         GetFileTimes should use ActOnPages
4441  Murray         System         ABORTED#ProcessDefs.Aborted
4604  Newman         System         FrameDefs.[New/Run]Config should return a GlobalFrameHandle
4662  McGregor       System         Allocating Space in Specific Banks
4786  mitchell       System         Incorrectly raised InsufficientVM error
```