Appendix A.
Error Messages

Yerk provides extensive information in its error messages. If you should write code that contains an error, yerk will provide you with clues you need to discover the source of your difficulties.

This is the general form of an error message:

```
Error Message
File Stack: a list of files.
Token=aToken  ::
```

File Stack

File Stack will have one or more file names listed only if the error occurred while you load source code into yerk. The error indication pinpoints the file that contains an error. In a nested load, the last file in the list is the file that contains the error. In this way you can go directly to the file where your error happened without guessing its location. If your error occurred while running or in interpret mode, there will naturally be no load files associated with the error, and the File Stack will be blank.

Token

Token indicates the word (usually a yerk word or selector) that the interpreter stumbled over -- the word that generated your error. In this way you will know which yerk word or selector was being processed when your error was generated.

Error Messages

The error message can take on several forms depending upon what type of error it is.

The simplest type is an error in the parsing of an input line. If you type in a word that yerk is unable to find in its dictionary, it will inform you of this. For example, if you type in the following non-Yerk word it will generate a "not found" error message:

```
0-> foo-bar
foo-bar? not found
File stack:
token=foo-bar ::
```

This indicates that the word foo-bar was not found in the yerk dictionary. Normally this will indicate only that you misspelled a word or thought something was present in the dictionary that was not. If it is a predefined yerk word that you expected to be there, look in the Glossary to confirm if it is truly there, and then compare the spelling in the Glossary with the one you typed.

Errors associated with the instantiation of a class provide different information designed to be helpful in those situations where you have misused a method. The message line will look like the following:

Msg# 104:Error from class Array :: CDE8: Error# 129: An index was out of range

This error indicates a subscript range condition, that is you were trying to access the array outside of the bounds that you originally declared (like trying to fetch a value from the "fourth" indexed cell of a three-cell array). Note that the error message tells you what class it was that generated the error message. CDE8 is the cfa -- the code field address (in hex notation) -- of the offending instance of the class. If this instance of the class has a name, you may discover that name by typing in the following code sequence (if you wish to know exactly what these yerk words do, check the Glossary):

```
$ CDE8 nfa id.
```

This puts the pfa of the object on the stack; converts it to the name field address, and prints out the name stored there. Note that this will print garbage if the object is a headerless object (that is one without a name field).

It is also possible to use this address in conjunction with the Examine Memory feature (see Part II, Chapter 1, The yerk Menu Bar) to see what is actually in the memory assigned to the object. For example, you'll be able to examine the contents of each data cell of an object's private data area. This will prove most enlightening for those difficult bugs.

Errors that occur in simple yerk words (outside of the class-object structure) produce a different message line. They tell you what yerk word the error message occurred in as well as giving you an informative error message that you may check against the list below to find some suggestions about possible causes. For example:

in ADDPARMS:: Error# 110: Too many named parameters or local variables

ADDPARMS is the yerk word that generated the error message; you may then look the error message up in this appendix for a fuller explanation and possible corrections.

There are also messages that do not necessarily indicate an error condition. In most cases, these messages indicate actions that you may want to perform under controlled circumstances; but at other times the messages alert you that you may have forgetten that a previous definition already exists in the dictionary. A simple example of this is when you redefine an object. This will generate a warning message ("object name not unique"), and the first defined object will no longer be accessible until you FORGET the more recently defined object.

Error Trapping For Programmers

All of these yerk words that we use to provide error support are available to programmers, to

help debug extensions to the language. The yerk words involved are: ClassErr", ?error, abort", msg#. All four of these words can be used only at compile time.

ClassErr" generates the error message beginning "Error from class..." Its use is defined only within a class compilation. This word takes a boolean argument off the stack: true indicates an error condition. When compiled, ClassErr" expects to find a number in its input stream (that is to say after the word). This number refers to a string-type resource in your resource file. This will be printed as the error message in the first line of the error report.

?Error generates the standard error message. It too expects a number that refers to a string-type resource in its input stream. The resource is the error message printed when an error is indicated. Again, the indication is given by passing a true boolean on the stack to the word at run time.

Abort" also expects a flag from the stack to indicate the presence of an error condition. It expects a literal text rather than a resource number, delimited by a double quote mark. This text will be printed if the presence of the error condition is indicated. All three of these error indications will terminate the running process.

Msg# expects a resource number in its input stream. This represents the message that it will print when this yerk word is executed. This is used for warning messages, does not terminate the running process, and does not expect a boolean on the stack (it always executes when encountered).

I can't open this resource file 100

Unable to open the requested resource file.

Action: Check file name. If this happens at a time other than explicitly attempting to open a resource file of your own construction, make certain that yerk.rsrc, the yerk resource file, is on your working disk.

Saved image header write failed 101

While attempting to write the header for your dictionary on a disk, a write error occurred.

Action: Likely cause of this error is insufficient disk space on your working disk.

Select indices must be sequential 102

The indices for a SELECT{ statement were not found to be sequential.

Action: If you really must have non-sequential indices, use the CASE statement.

Null object address passed to find-method 103

You passed zero as an object address for a late binding.

Action: Check the address calculation between the square brackets. If a local variable, make certain that it is initialized before use.

Dictionary write failed 105

A write statement failed when you attempted to save the dictionary.

Action: There may be insufficient room on the disk.

Unable to create save file 106

The attempt to create the save file for the dictionary failed.

Action: There may be insufficient room on the disk or a file of that name may already exist.

Unable to open save file 107

An attempt to open the requested file for saving thedictionary failed.

Action: A file of that name may already exist on the current disk. Use a different name to save the file.

Method not found in class 108

The method in question was not found anywhere in the class heirarchy chain for the object.

Action: Check the token in the error message, and compare it with methods available in the class hierarchy.

Unresolved forward reference to 109

You have declared a forward reference which has never been resolved.

Action: Give the declaration for the word using the :F compiler (see Part II, Chapter 4).

Too many named parameters or local variables 110

You have attempted to declare too many named parameters and local variables.

Action: The limit of named parameters and local variables for any one procedure is a total of six in any combination. You will have to remove the excess.

Incorrect parameter / local variable declaration 111

You have incorrectly declared the named parameters and the local variables.

Action: Verify the form of the named parameters and/or local vailables in the method indicated by the TOKEN= message. The correct form is as follows:

{ np1 np2 np3 \ lv1 lv2 lv3 -- }

Final '}' is missing 112

You have forgotten to close off the definition of your local variable or named parameters.

Action: Insert the '}' before the end of the line.

Prefix token not found 113

You used a prefix with a token that was not a named parameter, local variable, or value.

Action: Check spelling of the token in the error message. Named parameters and the like must be declared before use.

This prefix does not work with this data type 114

The prefix you have used is not defined for the data type given.

Action: If you wish to perform the operation in question, then change the data type at declaration time. For example, the token '++>' (the increment token) is not valid with instances of VECT. For a full description see on Advanced yerk Concepts (Part II, Chapter 4).

Class compilation only 115

Method definition can only occur within the context of class definition. It is undefined in other circumstances.

Action: Associate your method definition with a particular class. Make certain that they method definition in encased by :CLASS and ;CLASS.

Return stack grew too large 116

The return stack overflowed its allotted size.

Action: This should not occur in normal operation and is usually indicative of an infinite loop.

Sorry, you can't redefine instance variables 117

It is impossible to redefine instance variables.

Action: Pick a new name.

You didn't give it a name 118

You omitted the object name while instantiating a new object.

Action: Check that the instantiation statement in the code follows the form:
Classname Objectname.

That object name is not unique 119

You have just redefined an already existing object. It will no longer be accessible, until the new object is forgotten.

Action: If you won't need the earlier object for any reason, no action is necessary. It is safer, however, to FORGET the second object name and give it a new, unique name in the source code.

An object vector was just installed 120

A warning message that informs the user that object vector, a part of the nucleus, was changed. Object vectors that are executed upon startup (such as objinit) will affect the next startup of your yerk nucleus. You will want to use this when you are creating your own application to be distributed.

Action: No action needed.

I was unable to get enough heap from newptr 121

An attempt was made to allocate something on the heap when there was insufficient space.

Action: Dispose of pieces of code or data that are on the heap and no longer needed.

That is not a class name 122

The heap> statement was followed by something other than a class name.

Action: Heap> is only defined with class names. Check spelling. To get an undifferentiated block of heap space use NewPtr. Modules will be loaded automatically as they are needed. Space will be allotted for them then.

Selectors must be less than 16 characters long 123

A selector was discovered that was greater than sixteen characters.

Action: Shorten the selector name.

That is not a selector 124

An illegal selector name was declared in a method compilation.

Action: Selector names must be less than sixteen characters long and end in a colon.

I can't find that object 125

A token not in the yerk dictionary followed a selector during compilation.

Action: Check the spelling of the object. See Part II, Chapter 4, Advanced yerk Concepts, for information about forward referencing

---

That is not an object name 126

During compilation a selector was followed by a yerk word (not an object, value, named parameter, or local variable). This is an undefined operation.

Action: Check spelling. If you have a yerk word that produces a valid object address, you must use late binding: surround your yerk word with square brackets to indicate late binding.

You forgot the '[' 127

To indicate late binding, it is necessary to surround the computation that produces a valid object address by square brackets.

Action: A selector must be followed by a predefined object or a '['.

'[' should be preceded by a selector 128

Late binding is only defined in the context of selectors.

Action: Determine which method you meant to use on the object and use its selector immediately before the '['. (Note for Forth programmers: '[' has been replaced by '<[' and ']' by ']>'. Use those instead.)

An index was out of range 129

You have passed to an AT: or TO: method a value that is either less than zero or greater than the number declared as the size of the array.

Action: Check the array declaration. You may have intended it to be larger. Also check that the index value of the AT: or TO: message is within the range of the array.

You can't use indexed methods on this class 130

You have attempted to use indexed methods, such as at: or to:, on a class that contains no indexed data.

Action: Remove the offending method. Put: or Get: is probably the correct choice.

EXEC: undefined when address equals zero 131

You have attempted to EXEC: a variable that has a value of zero. This is an undefined operation.

Action: Initialize the variable with the cfa of a valid yerk word.

I can't open that file 132

An attempt was made to open a file that failed.

Action: Check to see if the file is on the disk. Make sure that it has not already been opened.

Sorry, the string must be < 32 characters long 133

Certain strings may be limited to <32 in length

Action: shorten your string

The data is too large for the object data area 134

Some classes may define a put: method to replace the data area of any object. This message indicates that the data being put is not the correct length

Action: Check your data.

String elements must be < 255 characters long 135

For class Sarray, string elements may not be >= 255 characters long.

Action: Limit the size of your string.

My list is empty 136

An attempt was made to reference to an ordered collection that is empty.

Action: Check to see why it is empty.

My list is full 137

An attempt was made to add to an ordered collection that had already reached its limits.

Action: Increase the size of the ordered collection.

I can't open this module's binary file 138

An open error occurred when attempting to open the binary modules file.

Action: Make sure the appropriate .BIN file is on the disk.

I can't create this module 139

The creation of the new binary file for this module failed.

Action: There may be insufficient room on the disk, or a file of this name may already exist on the disk and cannot be deleted.

I can't write to this module's binary file 140

A write error occurred when attempting to write to the newly created binary module file.

Action: Probable causes include insufficient space on the disk to create the binary module file.

I can't read this module's binary file 141

A read error occurred when the module loader attempted to open the module's binary file.

Action: Check to make sure that the binary file for the module that you want to use is actually on your working disk.

This module is invalid 142

The bit map for the module has been corrupted in some manner.

Action: Recompile the module.

This module has no exported names 143

You have failed to declare any names in your import statement.

Action: Without declared names the code in the module can never be executed.

You forgot to define this exported name 144

An exported name that you had declared in your import statement was not defined in your module.

Action: Compare your import statement and the code in your module. Check spelling.

Module had an error - I can't build it 145

This message indicates that the module that you were trying to construct was not built because of a serious error in the code.

Action: Correct the code error and recompile the module.

The module is empty **146**

The module just loaded attempted to compile no words.

Action: This is only a warning message.

Sorry, that is not a module definition 147

You have attempted to compile a module without having previously declared the module in an import statement.

Action: Make sure that a correct import statement occurs in the dictionary before you attempt to define a module.

You forgot to start the module with :Module 148

All modules compiled with the word Module must begin with :Module and end with ;Module to distinguish them from ordinary code compiled into the dictionary and to associate them with a particular import statement.

Action: Check spelling.

I can't find the 'Calls' file on this disk 149

You have attempted to use a Toolbox call without having the 'Calls.Tot' file on the disk you are using.

Action: Make certain that the file named 'Calls.Tot' is on your working disk.

I can't find a Toolbox call for this name 150

You have used Call with a name that is not a toolbox routine.

Action: Check the list of valid toolbox routines in the relevant Inside Macintosh chapter. Make certain that the name you have selected is on that list.

The offset from the last operation was negative 151

This indicates that a previous string operation returned an error.

Action: Check the offset value and see the string documentation from Inside Macintosh included with this manual (Appendix B).

The task to be removed was not found 152

This occurs when the program attempts to remove a task from the tasklist (using KillTask) that had not yet been entered into the task list.

Action: Check program flow. Check spelling of the task to be removed.

You must send me a new: message first 153

This message occurs when the program attempts to send some other message to a menu object, before sending a new: message.

Action: Check program flow. A new: message must precede any other uses of the menu object.

The menu text file is incomplete 154

You have not completely specified the options to be loaded into the menu in your menu loader file.

Action: You have probably not given a sufficient number of options to match the number you initialized your instance of the menu class

---

I can't open the menu text file 155

Getmtxt was called with a menu file name that does not exist on the disk.

Action: Check the spelling of the file name that you passed to getmtxt. Make certain that this file is on the disk in question. If this happens as you are firing up yerk.com, then you have forgotten to copy the file nmenu.txt to your working disk.

There isn't enough room to load this utility 156

There is insufficient heap to load the utility requested.

Action: If you have put objects or modules on the heap, you will have to release some of them before using this utility. Also it is possible to increase the amout of heap using the Install facility.

Not the address of a valid object 157

The method obj: was applied to an instance of class handle that contained the value zero.

Action: Make certain that the handle has been assigned the value of a valid object before using the obj: method.

Read must be < 65536 characters long 159

Certain types of reads using the MacTCP driver interface are confined to less than 65536 characters.

Action: Use a different type of read, or adjust your read input parameters.

Use ;M to terminate methods 163

You forgot to terminate a method with ';M'; perhaps you used a ';'.

Action: Look at all your methods, or load with +echo on.

Use "Module" loader for modules 164

You attempted to load a module using the ordinary loader.

Action: Load files that are modules (indicated because they begin with :Module) by using the word MODULE.

Drive change unsuccessful 165

An attempt was made to change the drive that a file was associated with.

Action: Check drive. Make sure a disk is in that drive and, if an external drive, that the cable is connected securely.

SetHandleSize failed 166

Not enough heap was available when you tried to set a handle size.

Action: Either you are asking for too much, or you need to increase the amount of application heap (for Yerk) using the 'Get Info' box in multifinder.

Floating Point not installed 167

You tried to do a floating point operation using Yerk.com, the integer version of Yerk, or you are using YerkFP.com but have the integer interpreter running (yerk>int) instead of the floating point interpreter (yerk>flt).

Action: Make sure you are using YerkFP.com and are running the FP interprerter (yerk>fp).

Not enough codefields 169

Indicates that you did not define as many codefields as you declared for your multiple cfa word.

Action: Check the correspondence between the number of code fields declared and number defined. They must match.

Get resource failed 170

An attempt to get a resource from the disk failed.

Action: Possible causes of this error include: resource file not open or attempting to get a non-existent resource.

I can't open the path text file 179

When Yerk started up, it could not find the path text file in the Yerk folder.

Action: Check to see that either your path file, or 'npath.txt' is located in the Yerk folder.

Object name not unique 180

You instantiated an object with the name of an existing object. This won't really hurt, but you will no longer be able to access the former object.

Action: Reload the source with +echo on.