Object-Oriented Programming: Review and Project Description

An Honors Project (ID 499)

by

Michael Niehaus

Thesis Director

Ball State University
Muncie, Indiana
April 30, 1990

May 5, 1990 Graduation
Introduction

What is object-oriented programming? As most people have seen, the use of this term in computer science and software engineering has grown more frequent in recent years. Large amounts of research (both in commercial and educational institutions) has been completed, with even more projects ongoing.

But why is there so much interest in object-oriented approaches, and my now? With the increased emphasis on how software is designed and implemented, research has focused on how the current methods can be improved. In past decades, structured programming was the "improved method." Now object-oriented programming (or more correctly, the object-oriented design strategy) is viewed as the next step forward [Booch 1986].

What benefits can be attributed to the new methodology? Many people point to these improvements:

- Information hiding
- Generality
- Maintainability
- Inheritance
- Reusability
- Extensibility
- Data abstraction
- Flexibility
- Simplicity
- Encapsulation

But how can a methodology lead to all of these benefits? That is what many people do not yet understand.

History

The first programming language to implement object-oriented features was Simula, developed in 1973. This language included the fundamental principles of objects, messages, and classes that are now common in object-oriented languages. But Simula did not provide the consistency and encapsulation that were also desirable [Smalltalk-80, 1981].
Flex, an improved and enhanced version of Simula, represented the next step. The first major object-oriented language drew heavily from Flex, while borrowing some ideas from LISP as well. This language was Smalltalk, developed at the Xerox Palo Alto Research Center (PARC).

Smalltalk continued to evolve, gaining more and more interest. Although it was a popular language in academic circles, its commercial success was not great. Developing powerful, efficient systems proved difficult [Rentsch 1982] because of the overhead imposed by the language (and the processing power limitations of the machines at the time).

Object-oriented languages continued to garner interest from the computer industry for the next several years. In the early 1980's, Apple Computer attempted a commercial implementation of an object-oriented system, the Lisa computer. A new language called Clascal (derived from Pascal with object-oriented extensions) was used. Although sound in concept, technical problems prevented the machine from becoming successful [Tesler 1986].

Doctor Niklaus Wirth, developer of the Pascal language, also played an important part in object-oriented language development. After a stay at Xerox PARC in 1976, Wirth began developing Modula-2. Although it did not include the class and object ideas from Smalltalk, it did add features such as information hiding and data abstraction (which later appeared in the Ada language as well) [Wirth 1984].

Modula-2, however, was not Wirth's only contribution to object-oriented programming. Apple Computer, not deterred by its Lisa experience, brought together a team to develop a new language, Object Pascal. Wirth was part of this team, who set out to extend the Pascal language. Larry Tesler, formerly of Xerox PARC, headed the group. In 1986, Apple released the language to the public, and it was since become
the prominent development language on Apple's Macintosh computer (along with the C and C++ languages).

While Apple was working on Object Pascal, Bjarne Stroustrup of AT&T's Bell Laboratories was developing a new language of his own. Called C++, this language offered object-oriented extensions to the C language. It has continued to evolve since its 1985 introduction, and currently is the focus of much interest [Mattson 1989].

In the years since, object-oriented programming languages have thrived. NeXT's new computer (and the Mach operating system kernel on which it was built) was developed using object-oriented methodologies. Commercial compiler developers have produced object-oriented language compilers as well, most of which have adopted AT&T's C++ or Apple's Object Pascal standard. (Most notable is Borland's inclusion of the Object Pascal features and extensions in the latest release of their popular MS-DOS Pascal compiler.)

Software Engineering Aspects

Traditionally, computer scientists have stressed a top-down, structured analysis of a system. For the past two decades, the following beliefs have been predominant [Semprivivo 1982]:

- It is important to identify a system's structure before attempting to define or describe the system's processes.

- One should first describe the system's structure in a very general or overview fashion and then break its structural components into smaller, more detailed components (hence the name top-down).

- Upon completing a structural taxonomy of the system, one should describe the system's processes, again employing a top-down approach.

While this methodology may work well for smaller, less complicated systems, it
begins to break down with today's more complex systems. Bertrand Meyer, a leading researcher in object-oriented design, points out the following flaws with this method [Meyer 1988]:

- The method fails to take into account the evolutionary nature of software systems.
- The very notion of a system being characterized by one function is questionable.
- Using the function as a basis often means that the data structure aspect is neglected.
- Working top-down does not promote reusability.

Meyer does not say that you cannot develop a complex system using top-down techniques. Instead, he believes that using a top-down methodology will involve tradeoffs between short-term convenience and long-term flexibility [Meyer 1989].

In place of a top-down approach, object-oriented methods and procedures suggest a bottom-up design. The goal of this methodology is to identify the naturally-occurring objects within the system. Once these have been identified, they can be developed into encapsulated units, each of which is a complete system of its own. These software objects will reflect external, real-world phenomena, making them easy to identify.

A debate of these two methodologies is beyond the scope of this paper; whole books count not completely cover all points. However, some strong points of the object-oriented approach can be exposed:

- Few interfaces. Objects should communicate with as few others as possible. This in itself will reduce coupling and complexity.
- Information hiding. Data is encapsulated inside the object itself, and the actual implementation of the object is not known. This enables changes to be made to
the objects throughout the system without requiring other consequential updates.

- Reusability. Since each object is self-contained, it is inherently more flexible, general, and hence reusable.
- Modularity. As with top-down design, modularity is the goal. The encapsulation used in object-oriented systems reinforces this goal, making it virtually a requirement.

Other benefits are sure to be found after more careful analysis of the methodology, and more experience using it.

Defining Object-Oriented Programming

What is object-oriented programming? It is a methodology that is rooted in the design process that stresses different areas than those in top-down structured design.

Bertrand Meyer gives the following motto of object-oriented design [Meyer 1989]:

Ask not what the system does:
Ask what it does it to!

He then gives a basic definition [Meyer 1989]:

Object-oriented design is a method which leads to software architectures based on the objects every system or subsystem manipulates, rather than “the” function it is meant to ensure.

While others may expand on this definition, adding other conditions and requirements, the basis remains the same.

Once an object-oriented system has been designed, it must be implemented. while some of the ideas can be implemented in standard programming languages, others require specific features and abilities to be present.

According to Geoffrey A. Pascoe, a programming language must exhibit four characteristics in order to be rightfully called an object-oriented language. These four
characteristics are [Pascoe 1986]:

1. **Information hiding.** The state of a particular software module is contained in private variables that can only be seen from within that particular module. Many modern languages support this to some degree. One noticeable exception is ISO standard Pascal, which provides no way to declare static variables within the scope of a procedure.

2. **Data Abstraction.** Using the principles of information hiding, it is possible for a programmer to define an abstract data type that consists of an internal representation plus a set of procedures used to access and manipulate the data. Two well-known programming languages that support these data abstraction mechanisms are Modula-2 and Ada.

3. **Dynamic Binding.** In a traditional procedure-oriented programming approach, programmers usually find themselves writing code to handle different cases, i.e. data types. It is not uncommon to see routines with names like PrintInt, PrintStr, PrintReal, etc. With object-oriented programming, the data types are actually part of an object. To print the data, the programmer only needs to send a print message to that object. It is the object's responsibility to worry about how the printing is accomplished. The primary goal of dynamic binding is to increase flexibility by permitting the addition of new classes of data types (objects) without having to modify existing code.

4. **Inheritance.** Objects can inherit the instance variables, class variables, and methods from a parent class. For different object that are similar in design and structure, this can eliminate the need to completely rewrite all code involved. Inheritance coupled with dynamic binding results in reusable
While many languages may exhibit some of these features, most do not completely fulfill these requirements. Partial implementation is better than none at all, however.

Some other concepts that are important in object-oriented programming are multiple inheritance and automatic storage management. While these features are not required in order to have a full-fledged object-oriented language, they are very useful and desirable.

Multiple inheritance means that a class can inherit the behavior of more than one superclass or parent. Certain LISP systems and version 2.0 of C++ support this feature.

Automatic storage management allows programmers to ignore the details of allocating and deallocation storage for an object, simplifying the production and maintenance of software and shortening the development times. Smalltalk and LISP are prime examples of languages that provide automatic storage management.

In Practice

Where are these object-oriented programming languages in use? Several major corporations and institutions, including AT&T, NeXT, and Apple, use object-oriented languages internally. These three companies are also actively involved in efforts to encourage usage of these languages outside the company.

AT&T's Bell Laboratories was the birthplace of C++, currently the most popular of object-oriented programming languages [Stroustrup 1989]. This language has since been adopted by many companies as a standard.

Apple Computers' Object Pascal is perhaps to be considered the "people's object-oriented language," since the language specifications are available without
licensing fees (since Apple's business is selling computers, not compilers).

NeXT's computer uses Objective-C, another object-oriented version of the C language. This language is so ingrained in the system that it cannot be avoided or ignored.

Other firms and industries are also active participants in this area. Some of these firms are:

Hewlett-Packard, Eastman Kodak, Symbolics, Servio Logic Development, and of course, Apple, AT&T, and NeXT.

Many leading educational institutions are also beginning to stress object-oriented methodologies:

University of Stockhold, University of Wisconsin, University of Texas, Université de Grenoble, University of California, University of Colorado, Massachusetts Institute of Technology, University of Toronto, Oregon Graduate Center, University of Massachusetts, University of Geneva, Brown University, Stanford University, University of British Columbia, etc.

In the future, even more corporations and institutions will be attracted to the object-oriented methodology as well. The prospects of reusability, lower maintenance costs, and higher flexibility will definitely attract more interest.

Class Libraries

In order to facilitate reusable code, special class libraries have been developed. These class libraries are designed to include many of the most often-used routines, those that a programmer would most likely have to include in multiple projects.

These class libraries have all the inherent advantages of object-oriented programs. If for some reason you do not like the behaviour of a certain object, you can change it by creating a subclass of that object, redefining any routines that you wish. This ability to override standard behaviour is an important part of any class library.
For example, in a sample class library you may have a file class that is used for all file input and output activities. Using the Unix standard input-output (stdio) library as a basis, you could develop a class definition like this:

```c
struct CDataFile : indirect {
    char    cFileName[256];
    FILE   *cFile;

    int     Close();
    int     Open(char *fileName, char *mode);
    int     Printf(char *format, ...);
    int     Scanf(char *format, ...);
    int     getC();
    char *  GetS(char *string, int maxSize);
    int     PutC(int charCode);
    int     PutS(char *string);
    size_t  Read(void *result, size_t size);
    size_t  Write(void *, size_t size);
    int     GetPos(fpos_t *position);
    int     Seek(long position, int posMode);
    int     SetPos(fpos_t *position);
}
```

This class definition contains all of the functions needed to access and manipulate the file. Notice that the file variable itself (cFile) is not accessed in any of the routines. This information is only part of a created object. This object keeps track of needed data such as the file handle, so the programmer need only concern himself with using the routines. Because of this, the function prototypes are also similar, allowing the programmer to focus more on function than detail.

Once such a class definition has been developed, it can be used throughout a program. In the following example code, theFile is the variable that will point to the object. Once this object has been created, it can be accessed in the same manner any other structure would be accessed.

```c
main()
{
    CFile  *theFile;
    char   string[256];
```
theFile = new(CFile); /* Create a new file object. */
theFile->Open("/mnt/usr/mithomas/mbox","r"); /* Open file */
theFile->GetS(string); /* Read a string */
/* Continue processing... */
theFile->Close(); /* Close the file */
dispose(theFile); /* Dispose the file object */
}

As you can see, this code is simplified and easy to read. With fewer parameters, programmers will not have as much difficulty remembering the prototypes of all the routines.

In order to make a collection of these objects useful, consistent naming is a necessity. The creators of class libraries usually develop their own methods to maintain this consistancy. For example, Think C (a C compiler developed by Symantec that includes object oriented extensions) uses the conventions in its class library:

1. Use uppercase and lowercase in all programs. The manner in which such should be used is as follows:
   a. All variable names should start with lowercase letters.
   b. All type definitions, class names, method names, and routine names should start with an uppercase letter.
   c. All constants (as defined) should be completely uppercase (for maximum visibility), or can also be defined as a variable (with the first letter lowercase).

2. Variables should be named logically, showing their usage.
3. All global variables should be declared with a preceding lowercase "g," for example, "gTitle."
4. All class definition names should begin with an uppercase "C," for example "COoutput."
5. All the instance variables of a class should begin with a lowercase "c," such as "cFileHandle."
6. Wherever possible, class method names should be consistent with those of other classes. For example, you would specify an "Initialization" as a method name if all of the other classes have an "Initialization" method. If other classes have an "Init" method, your classes should as well.
While some of these conventions are standard to Macintosh programming in general, others have been added by Symantec in order to keep things clear and easy to understand.

While such class libraries are useful, many times the sheer number of object classes included in them, and the number of routines in each of the classes, makes use of the library difficult. In order to simplify this, browsers have been developed. These allow programmers to view on-screen the class definitions, instance variables, and methods for every class, without having to page through manuals to find them. These browsers are usually graphically-based, allowing an accurate tree-like representation to be presented to the programmer.
The Project

As an application of these principles, I selected a moderately-sized project to work with. Rahul Dhesi's Zoo program was selected, in part due to the attempted machine-independent nature of Dr. Dhesi's original code. While the code is sufficiently independent to create versions for other environments, the Mac version required significant changes. While emulations of the standard Unix libraries are available, the performance of these is not sufficient for such an intensive program. In order to achieve sufficient levels of performance, these routines would need to be translated into equivalent Mac operating system calls.

The first task involved was the identification of the major object classes in the Zoo program. Since the program is an archival program, the first obvious class is an archive class. This class was developed as follows:

```c
struct CArchive : indirect
{
    CArcDialog       *cDialogObject;
    ListHandle       cList;
    Str63            cName;
    int              cVRefNum;
    int              cRefNum;

    void             Init();
    int              Open(Str63 whatFile, int whatRef);
    int              Close(void);
    int              Extract(int entryNum);
    /* Must be overridden in each subclass */

    int              Add(void);
    /* Must be overridden in each subclass */

    int              Delete(void);
    /* Must be overridden in each subclass */

    int              ReadHeader(void);
    /* Must be overridden in each subclass */

    int              ReadDirectory(Str63 *theName);
    /* Must be overridden in each subclass */

    Boolean         CheckEvent(EventRecord theEvent);
```

This class isolated the implementation details from the rest of the application, while still providing all of the functionality that is needed to perform any given function.

Once the basic archive class type was defined, the specific Zoo routines could be added. This was done by creating a subclass of archive class described above, overriding the needed routines (all marked above) to process the Zoo archive file. This method is general, allowing the addition of any other archive file formats in the future by just creating another subclass that defines the same routines to handle the different file format and compression methods.

These classes take care of archive-handling and input. With those out of the way, the next task is output. For this, another class was created, called COOutput. The basic function of this class is to handle the newly created files that result from the extraction of entries from the archive files. Its declaration is as follows:

```c
struct COOutput : indirect
{
    SFReply cReply;
    int cRefNum;
    int cCount;
    DialogPtr cDialog;
    int cSaveAll;

    void Init();
    int Open(Str63 *fName);
    int Write(char *data, int numChars);
    void SetEndOfFile(long eof);
    void Close();
};
```

This class is general enough to be reused with any future classes, as well. As is standard with the Macintosh, this class presented the necessary choices to the user, allowing the saving of the file in any location (a feature that can be overcome with a menu option).
The default behavior of this class is to create a file that is exactly like the original was before it was archived. In some cases, however, this is not what is wanted. In the case of text files, some conversions need to be performed. In a normal program this would be done using case selection statements. But using the object-oriented features, a new output subclass can be created. This subclass consists of one unique routine, Write, which performs the needed translations while writing out the data. All other routines remain the same, and are inherited from the parent class.

Now that the three main functions of a program (input, processing, output) have been taken care of, it is time to develop a way for the user to interact with the program. With the object-oriented design of the program, any user interface could be added. In this case, a standard Macintosh interface was added.
Although this class is complicated, the declaration for it appears simple. This is because there is little interaction between the different classes. Each class has a definite purpose, and only communicates with other classes when needed. The class declaration is as follows:

```c
struct CArcDialog : indirect
{
    DialogPtr    cDialog;
    ListHandle   cList;
    DialogPtr    cSelectDialog;

    void Create(Str63 title);
    void Add(Str63 *item);
    void GetNextSelect(int *prev);
    int Match(char *string, char *pattern);
    int Select(EventRecord theEvent);
    void CleanUp(void);
    int CheckEvent(EventRecord theEvent);
};
```

While it may seem that the interface is only a small part of the total project, much work went into this section. Consistency, ease of use, and flexibility were stressed.

This user interface combines the best of both worlds: list management and selection from the Mac environment, and wildcard selections from the Unix side.

In addition, a help class was created to give basic information to the users of the software. This help tries to answer common questions that arise when dealing with several different systems (and transferring files from one to another):

```c
struct CHelp : indirect
{
    DialogPtr    cDialog;

    void Init(void);
    void Create(void);
    void CleanUp(void);
    int CheckEvent(EventRecord theEvent);
};
```

The combination of these classes, with the addition of some support routines, creates a useful, working program. The attention to flexibility will allow the addition of
more archive formats (and corresponding archive classes) without changing the other object classes. Currently several other formats are being developed, including Arc, Tar, Zip, StuffIt, Packit, and others.

The complete code for the project follows, arranged into sections by class. The header files are also included.
MacBooz Source Code

CArchive.h

#define _H_CArchive

#define zooArchive 1024L
#define zooType 'Zoo'
#define zipArchive 1025L
#define zipType 'Zip'

#define noArcErr 0
#define invalidArcErr -1
#define ioErr -2
#define arcOpenErr -3

struct CArchive : indirect
{
    CArcDialog *cDialogObject;
    ListHandle cList;
    Str63 cName;
    int cVRefNum;
    int cRefNum;

    void Init();
    int Open(Str63 whatFile, int whatRef);
    int Close(void);
    int Extract(int entryNum);
    /* Must be overridden in each subclass */

    int Add(void);
    /* Must be overridden in each subclass */

    int Delete(void);
    /* Must be overridden in each subclass */

    int ReadHeader(void);
    /* Must be overridden in each subclass */

    int ReadDirectory(Str63 *theName);
    /* Must be overridden in each subclass */

    Boolean CheckEvent(EventRecord theEvent);
};

CArchive.c

#include "Archivist.h"

#include <Quickdraw.h>
#include <FileMgr.h>
#include <MenuMgr.h>
#include <ListMgr.h>
#include <EventMgr.h>

extern MenuHandle gMenus[3];
extern int saveAll;

void CArchive::Init(whatMenu,new,open,close)
MenuHandle whatMenu;
int new,open,close;
{
    cRefNum = 0;
    /* Will have to modify this to go to multiple files */
    DisableItem(whatMenu,close);
}

int CArchive::Open(whatFile,whatRef)
Str63 whatFile;
int whatRef;
{
    int err,i,result;
    Str63 localDirectory;
    Handle watchCursor;

    /* Local copies of object variables */
    int localFile;

    watchCursor = GetResource('CURS',4);
    if ( watchCursor != 0 )
    {
        MoveHI(watchCursor);
        HLock(watchCursor);
        SetCursor( *watchCursor);
        HUnlock(watchCursor);
    }

    /* Create a dialog object. */
    this->cDialogObject = new(CArcDialog);
    this->cDialogObject->Create(whatFile);

    /* Open file, read header and entries. */
    err=FSOpen(whatFile,whatRef,&localFile);
    if (err<0)
    {
        this->cRefNum = .0;
        this->CloseO;
        InitCursorO;
    }
}
MacBooz Source Code

```c
#include <stdio.h>

class MacBooz 
{
  MacBooz ()
  
  virtual ~MacBooz ()
  
private:
  int cRefNum;
  int cVRefNum;
  std::string cName[64];
  std::string cDialogObject;
  std::string localDirectory;

public:
  virtual int Close ()
  
  virtual int ReadHeader ()
  
  virtual int ReadDirectory ()
  
  virtual int Add ()
  
};

int CArchive::Close (void)
{
  FlInfo finderInfo;
  int i;

  /* Local Copies */

  int localRef, localVRef;
  Str63 localFile;

  if (cDialogObject != OL)
  {
    cDialogObject->CleanUp ();
    delete (cDialogObject);
    cDialogObject = OL;
  }

  for (i = 0; i < 64; i++)
  {
    cRefNum = localFile;
    for (i = 0; i < 64; i++)
      cName[i] = whatFile[i];
    cVRefNum = whatRef;

    result = this->ReadHeader ();
    if (result != 0)
    {
      cRefNum = 0;
      cClose ();
      InitCursor ();
      ZooError (1);
      return -1;
    }

    DisableItem (gMenus[1], 1);
    EnableItem (gMenus[1], 2);
    result = this->ReadDirectory (&localDirectory);
    while (result != 0)
    {
      cDialogObject->Add (&localDirectory);
      result = this->ReadDirectory (&localDirectory);
    }

    InitCursor ();
    return;
  }

  return -1;
}
```

localFile[i] = cName[i];
localRef = this->cRefNum;
localVRef = this->cVRefNum;

if (localRef == 0)
    return -1;

DisableItem(gMenus[1], 2);
EnableItem(gMenus[1], 1);
FSClose(localRef);

GetFlnfo(localFile, localVRef, &finderInfo);
finderInfo.fdType = 'Zoo';
finderInfo.fdCreator = 'Booz';
SetFlnfo(localFile, localVRef, &finderInfo);
cRefNum = 0;

return 0;

int CArchive::Extract(entryNum)
int entryNum;
{
    SysBeep(4);
}

int CArchive::Add(void)
{
    SysBeep(4);
}

int CArchive::Delete(void)
{
    SysBeep(4);
}

int CArchive::ReadDirectory(const char *theName)
char *theName;
{
    SysBeep(4);
}

int CArchive::ReadHeader(void)
{
    SysBeep(4);
}
MacBooz Source Code

Boolean CArchive::CheckEvent(theEvent)
EventRecord theEvent;
{
    int                      result,status;
    int                      current;
    Boolean                  yesNo;

    IsDialogEvent(&theEvent);

    if ( this->cDialogObject != 0L)
    {
        result = this->cDialogObject->CheckEvent(theEvent);
        switch (result)
        {
            case ARC_NULL:
                break;
            case ARC_CLOSE:
                this->CloseO;
                break;
            case ARC_EXTRACT:
                saveAll = 0;
                current = 0;
                this->cDialogObject->GetNextSelect(&current);
                while (current != -1)
                {
                    status = this->Extract(current);
                    if ( status == OPEN_CANCEL )
                        return;
                    current++;
                    this->cDialogObject->GetNextSelect(&current);
                }
                break;
        }
    }
}

Archive Class.c

#include <oops.h>
#include "CArchive.h"

#include <FileMgr.h>
#include <MenuMgr.h>
#include <ListMgr.h>

pascal void UpdateList(DialogPtr theDialog,int itemNo);
Boolean ArchiveWindow(WindowPtr whatWindow);

void Archive::Init(whatMenu,new,open,close)
MenuHandle       whatMenu;
MacBooz Source Code

```c
int cRefNum = 0;
cMenu = whatMenu;
cNew = new;
ocOpen = open;
ocClose = close;
}

int Archive::Open(whatFile,whatRef)
Str63 whatFile;
int whatRef;
[

Rect viewRect, dataBounds, theItemRect;
Point csize, theCell;
int itemType, err, i;
Handle theItemHandle;

/* Local copies of object variables */

DialogPtr localDialog;
ListHandle localList;
int localFile;
Handle localHeader;
Str63 *localDirectory;

/* Build modeless dialog window */

SetDAFont(21);
localDialog = GetNewDialog(133,0,-1);
SetWTitle((WindowPtr) localDialog, whatFile);
SelectWindow( (WindowPtr) localDialog);
SetPort(localDialog);
SetDAFont(0);

csize.v = 0;
csize.h = 199;
SetRect(&viewRect,10,10,200,200);
SetRect(&(dataBounds,0,0,1,0));

GetDItem(localDialog, 2, &ItemType, &theItemHandle, &theItemRect);
SetDItem(localDialog, 2, &ItemType, UpdateList, &theItemRect);

localList = LNew(&viewRect, &dataBounds, csize, 0, localDialog,
true, false, false, true);
SetWRefCon((WindowPtr) localDialog,(long) localList);
LActivate(true, localList);

/* Open file, read header and entries. */
```
MacBooz Source Code

```c
err=FSOpen(whatFile,whatRef,&localFile);
if (err<0) {
    return -1;
}

localHeader = this->ReadHeader();

theCell.h=0;
theCell.v=-1;

localDirectory = (Str63 *) this->ReadDirectory();

while ( localDirectory != 0L)
{
    theCell.v=LAddRow(I,theCell.v+1,localList);
    LSetCell( localDirectory,strlen(localDirectory),theCell,localList);
    localDirectory = (Str63 *) this->ReadDirectory();
}
LUpdate( (localDialog)->visRgn,localList);

/* Copy the local variables into the object's variables */

this->cDialog = localDialog;
this->cList = localList;
this->cRefNum = localFile;
this->cHeader = localHeader;
for (i=0;i<64;i++)
    this->cName[i] = whatFile[i];
this->cVRefNum = whatRef;

}

int Archive::Close(void)
{
    FInfo finderInfo;
    int i;

    /* Local Copies */

    int localRef,localVRef;
    Str63 localFile;
    ListHandle localList;
    DialogPtr localDialog;

    for (i=0;i<64;i++)
        localFile[i] = cName[i];
    localRef = cRefNum;
    localList = cList;
    localDialog = cDialog;

```
localVRef = cVRefNum;

if (cRefNum == 0)
    return -1;

LDIspos(loca1List);
DisposeDialog(localDialog);
FSClose(localRef);
GetFlnfo(localFile,localVRef,&finderInfo);
finderInfo.fdType='Zoo';
finderInfo.fdCreator='Booz';
SetFlnfo(localFile,localVRef,&finderInfo);
cRefNum = 0;

return 0;

int Archive::Extract(void)
{
    SysBeep(4);
}

int Archive::Add(void)
{
    SysBeep(4);
}

int Archive::Delete(void)
{
    SysBeep(4);
}

int Archive::Select(void)
{
    DialogPtr matchDialog;
    int itemHit;
    Handle theItem;
    Str255 theText;
    int itemType;
    Rect box;
    Str255 listText;
    int dataLen,numRows,i;
    Point theCell;

    /* Local Copies */

    ListHandle localList;
    localList = cList;

    matchDialog = GetNewDialog(134,0,-1);

MacBooz Source Code

itemHit=0;

while (itemHit!=1)
    ModalDialog(0L,&itemHit);
GetDlgItem(matchDialog,3,&itemType,&theItem,&box);
GetDlgItem(matchDialog,3,&itemType,&theItem,&box);
PtoCstr((char *)&theText);
DisposDialog(matchDialog);

theCell.h=0;
numRows=(*localList).dataBounds.bottom;
for (i=0;i<numRows;i++)
{
    dataLen=64;
    theCell.v=i;
    LGetCell(&listText,&dataLen,theCell,localList);
    if (this->Match((char *)&listText,(char *)&theText)==1)
        LSetSelect(true,theCell,localList);
}

int Archive::Match(string,panem)
char *string;
char *pattern;
{
    char *psave,*ssave; /* back up pointers for failure */
    psave = ssave = ((char *)0);
    while (1) {
        for (; *pattern == *string; pattern++,string++) /* skip first */
            if (*string == '0')
                return(1);
        if (*string != '0' && *pattern != '0') {
            pattern++; /* end of strings, succeed */
            if (*pattern == '*') {
                psave = ++pattern;
                ssave = string;
                /* have seen a star */
            } else if (ssave != (char *)0 && *ssave != '0') {
                /* if not at end */
                /* ...have seen a star */
                string = ssave + 1;
                /* skip 1 char from string */
            } else
                pattern = psave;
                /* and back up pattern */
        } else /* otherwise just fail */
    }
}
int Archive::IsArchiveEvent(theEvent) {
    EventRecord theEvent;
    
    WindowPtr whatWindow;
    int winLoc;
    int itemHit;
    Rect dragRect;
    Point localPt;
    
    /* Local Copies */
    DialogPtr localDialog;
    MenuHandle localMenu;
    int localClose;
    ListHandle localList;
    
    localDialog = cDialog;
    localClose = close;
    localMenu = cMenu;
    localList = cList;
    
    localPt = theEvent.where;
    GlobalToLocal(&localPt);
    dragRect = screenBits.bounds;
    
    if ( ! ArchiveWindow( FrontWindow() ) ) {
        DisableItem(localMenu,localClose);
        return false;
    } else {
        EnableItem(localMenu,localClose);
    }
    
    winLoc = FindWindow(theEvent.where,&whatWindow);
    if (whatWindow!=localDialog)
        return false;
    else
        return true;
    
    if (theEvent.what != mouseDown) {
        if (IsDialogEvent(&theEvent)) {
            DialogSelect(&theEvent,&localDialog,&itemHit);
            if (theEvent.what==keyDown)
                return false;
            else
                return true;
        } else
            return false;
    } else
        return true;
}
MacBooz Source Code

    return false;
}

switch (winLoc)
{
    case inDrag:
        DragWindow(localDialog, theEvent.where, &dragRect);
    case inGoAway:
        if (TrackGoAway(localDialog, theEvent.where))
            this->Close();
    case inContent:
        if (DialogSelect(&theEvent, &localDialog, &itemHit))
        {
            switch (itemHit)
            {
                case 1:
                    this->Extract();
                    break;
                case 2:
                    if (LClick(localPt, theEvent.modifiers, localList))
                        this->Extract();
                    break;
                case 3:
                    this->Select();
                    break;
            }
            break;
        default:
            return false;
        }
        return true;
}

Ptr Archive::ReadDirectory(void)
{
    SysBeep(4);
}

Handle Archive::ReadHeader(void)
{
    SysBeep(4);
}

pascal void UpdateList(theDialog, itemNo)
WindowPtr theDialog;
int itemNo;
{
    short int itemType;
    short int item;
MacBooz Source Code

Rect box;
ListHandle whatList;
GrafPtr savePort;

GetPort(&savePort);
SetPort(theDialog);
GetDItem(theDialog,itemNo,&itemType,&item,&box);
InsetRect(&box,-1,-1);
FrameRoundRect(&box,1,1);
whatList=(ListHandle) GetWRefCon(WWindowPtr) theDialog);
LUpdate( theDialog->visRgn, whatList );
SetPort(savePort);

boolean ArchiveWindow(void whatWindow)
WindowPtr whatWindow;

if (whatWindow != OL)
    return true;
else
    return false;

30 Subclass.c

#include "Archivist.h"
#include "func.h"

tern unsigned int crccode;
* gOutput;

int ZooArchive::Extract(entryNum)
int entryNum;
    Point theCell;
    Boolean found;
    int i,status;
    Str63 theName;
/* Local Copies */

int localFile;
struct dirent localDirentry;

localFile = cRefNum;

status = this->ReadHeaderO;
for (i=0;i<=entryNum;i++)
    {
MacBooz Source Code

```cpp
this->ReadDirectory(&theName);
}
localDirentry = cDirentry;

/* The following is copied from what used to be the oozext routine. */

if (localDirentry.major_ver > 1 ||
    (localDirentry.major_ver == 1 && localDirentry.minor_ver > 0))
    ZooError(6); /* Higher version needed */

/* Open an output file. Make sure it was successful. */

status = gOutput->Open(&theName);

if (status == OPEN_ERROR)
    return OPEN_ERROR;
else if (status == OPEN_CANCEL)
    return OPEN_CANCEL;

SetFPos(localFile, 1, localDirentry.offset);
crccode = 0; /* Initialize CRC before extraction */

if (localDirentry.packing_method == 0)
    status = getfile(localFile, localDirentry.size_now);
else
{
    if (localDirentry.packing_method == 1)
        status = lzd(localFile);
    else
        ZooError(6); /* Higher version needed */
}

if (localDirentry.file_crc != crccode)
    ZooError(3); /* Bad CRC */

gOutput->SetEndOfFile((long) localDirentry.org_size);
gOutput->CloseO;

return status;
}

int ZooArchive::Delete(void)
{
    SysBeep(4);
}

int ZooArchive::ReadHeader(void)
{
    int localFile;
    int status;
```
MacBooz Source Code

```c
long size;
struct zoo_header theHeader;
BYTE bytes[SIZ_ZOOH];

localFile = cRefNum;

size = (long) SIZ_ZOOH;
SetFPos(localFile, 1, 0L);
status = FSRead (localFile, &size, &bytes);
if ( status != 0 )
    return -1;
else
{
    status = b_to_zooh (&theHeader, bytes);
    if ( status != 0 )
        return -1;
    cDirentry.next = theHeader.zoo_start;
    SetFPos(localFile, 1, theHeader.zoo_start);
}
```

```c
int ZooArchive::ReadDirectory(const char *theName)
{
    int localFile;
    struct direntry direntry;

    long size;
    int status,i;
    BYTE bytes[MAXDIRSIZE];

    localFile = cRefNum;
    direntry.next = cDirentry.next;
    status = SetFPos(localFile, 1, direntry.next);

    size = (long) MAXDIRSIZE;
    status = FSRead (localFile, &size, &bytes);
    if ( (status < 0) && (size==0) )
    {
        return 0;
    }

    b_to_dir(&direntry, bytes);

    if ( (direntry.namlen == 0) || (strcmp(&direntry.lfname, ".\12")==0 ) )
        strcpy(theName,&direntry.fname);
    else
        strcpy(theName,&direntry.lfname);

cDirentry = direntry;
```
MacBooz Source Code

if (dirent.next < 0)
{
    SysBeep(4);
    PostMessage(theName);
}

if (dirent.next == 0L)
    return 0;
else
    return 1;

ZooArchive::Add()
{

}

int b_to_zooh (zoo_header, bytes)
struct zoo_header *zoo_header;
BYTE bytes[];
{
    int i;

    for (i = 0; i < SIZ_TEXT; i++)
        zoo_header->text[i] = bytes[TEXT_I + i];
    zoo_header->lo_tag = to_int(&bytes[ZTAG_I]);
    zoo_header->hi_tag = to_int(&bytes[ZTAG_I+2]);
    /* zoo_header->zoo_tag = to_long(&bytes[ZTAG_I]); */
    zoo_header->zoo_start = to_long(&bytes[ZST_I]);
    zoo_header->zoo_minus = to_long(&bytes[ZSTM_I]);
    zoo_header->major_ver = bytes[MAJV_I];
    zoo_header->minor_ver = bytes[MINV_I];

    if (zoo_header->major_ver > 2)
        return -1;
    return 0;
}

int b_to_dir(direntry, bytes)
struct direntry *direntry;
BYTE bytes[];
{
    int i;

    direntry->lo_tag = to_int(&bytes[DTAG_I]);
    direntry->hi_tag = to_int(&bytes[DTAG_I+2]);
    /* direntry->zoo_tag = to_long(&bytes[DTAG_I]); */
    direntry->type = bytes[DTYP_I];
    direntry->packing_method = bytes[PCKM_I];
    direntry->next = to_long(&bytes[NXT_I]);
MacBooz Source Code

direntry->offset = to_long(&bytes[OFS_I]);
direntry->date = to_int(&bytes[DAT_I]);
direntry->time = to_int(&bytes[TIM_I]);
direntry->file_crc = to_int(&bytes[CRC_I]);
direntry->org_size = to_long(&bytes[ORGS_I]);
direntry->size_now = to_long(&bytes[SIZNOW_I]);
direntry->major_ver = bytes[DMAJ_I];
direntry->minor_ver = bytes[DMIN_I];
direntry->deleted = bytes[DEL_I];
direntry->comment = to_long(&bytes[CMT_I]);
direntry->cmt_size = to_int(&bytes[CMTSZ_I]);
for (i = 0; i < FNM_SIZ; i++)
    direntry->fname[i] = bytes[FNAME_I + i];
direntry->namlen = (unsigned char) bytes[NAMLEN_I];
for (i=0;i<direntry->namlen;i++)
    direntry->lnfname[i] = bytes[LFNAME_I+i];
direntry->dirlen = (unsigned char) bytes[DIRLEN_I];
for (i=0;i<direntry->dirlen;i++)
    direntry->dirname[i] = bytes[DIRNAME_I+i];
return 0;
}

Zoo Utility Routines

/* adbfcrc.c */

/*
adbfcrc() accepts a buffer address and a count and adds the CRC for all bytes in the buffer to the global variable crccode using CRC-16.

CRC computation algorithm taken from source code for ARC 5.12, which in turn took it from an article by David Schwaderer in the April 1985 issue of PC Tech Journal.

I claim no copyright over the contents of this file.

-- Rahul Dhesi 1986/12/31

Simplified for the Macintosh version.

-- Michael Niehaus 1990/02/07
*/

unsigned int crccode;
unsigned int crc_table[] = {
0x0000, 0xc0c1, 0xc181, 0x0140, 0xc301, 0x03c0, 0x0280, 0xc241, 0x0c61, 
0x06c0, 0x0780, 0xc741, 0x0500, 0xc5c1, 0x0c81, 0x0440, 0xc010, 0x00c0, 
0x0d80, 0xcd41, 0x0f00, 0xcfc1, 0xce81, 0x0e40, 0x0a00, 0xcac1, 0xcb81,
MacBooz Source Code

0x0b40, 0xc901, 0x9c80, 0x0880, 0x8c41, 0xdd80, 0x18c0, 0x1980, 0x9d41,
0x1b00, 0x0d81, 0x1a40, 0x1e00, 0x0dec1, 0x0df81, 0x1f40, 0x0d01,
0x1dc0, 0x1c80, 0xd041, 0x1400, 0x94c1, 0x5d81, 0x1540, 0x9d701, 0x17c0,
0x1680, 0x9d641, 0x1d01, 0x12c0, 0x1380, 0x9d341, 0x1100, 0x9d1c1, 0x0d01,
0x1040, 0xF001, 0x30c0, 0x3180, 0xF141, 0x3300, 0xF3c1, 0xF281, 0x3240,
0x3600, 0xF6c1, 0xF781, 0x3740, 0xF501, 0x35c0, 0x3480, 0xF441, 0x3ec0,
0xFcc1, 0xFd81, 0x3d40, 0x0F001, 0x30c0, 0x3e80, 0x9e41, 0xFa01, 0x3ac0,
0x3b80, 0xFB41, 0x3900, 0xF9c1, 0xF881, 0x3840, 0x2800, 0x9e8c1, 0xe981,
0x9c40, 0x9e01, 0x2bc0, 0x2a80, 0xe041, 0xe0e1, 0x2ec0, 0x2F80, 0xeF41,
0x2d00, 0xedc1, 0xec81, 0x2c40, 0xe401, 0x24c0, 0x2580, 0xe541, 0x2700,
0x2e7c1, 0xe881, 0x2640, 0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0,
0x2080, 0xe041, 0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281,
0x6240, 0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441,
0x6c00, 0xac1c, 0xad81, 0x6d40, 0xaF01, 0x6Fc0, 0x6e80, 0xae41, 0xa0a1,
0x6e0c, 0x6b80, 0x6ab1, 0x6900, 0xa9c1, 0xa881, 0x6840, 0x7800, 0xb8c1,
0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41, 0xbe01, 0x7ec0, 0x7f80,
0xbF41, 0x7d00, 0xbd81, 0xbc81, 0x7c40, 0xb401, 0x74c0, 0x7580, 0xb541,
0x7700, 0x7b7c1, 0xb681, 0x7640, 0x7200, 0xb2c1, 0xb381, 0x7340, 0xb101,
0x71c0, 0x7080, 0xb041, 0x5000, 0x90c1, 0x9181, 0x5140, 0x9301, 0x95c0,
0x5280, 0x9241, 0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x99c1,
0x5540, 0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5F00, 0x9Fc1, 0x9e81, 0x5e40,
0x5a0c, 0x9ac1, 0x9c81, 0x5b40, 0x9901, 0x59c0, 0x5880, 0x9841, 0x8801,
0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x88a1, 0x4a40, 0x4e00, 0x8ec1,
0x8F81, 0x4F40, 0x8d01, 0x4d0c, 0x4e80, 0x8c41, 0x4400, 0x84c1, 0x8581,
0x4540, 0x8701, 0x47c0, 0x4680, 0x8641, 0x8201, 0x42c0, 0x4380, 0x8341,
0x4100, 0x81c1, 0x8081, 0x4040

int addbfrcrc(buffer, count)
char *buffer;
int count;
{
    unsigned int localcrc;
    int i;

    localcrc = crccode;
    for (i = 0; i < count; i++)
        localcrc = ((localcrc >> 8) & crctab[localcrc ^ buffer[i]]) & 0x00ff;
    crccode = localcrc;
}

#include "Archivist.h"
#include "func.h"
#include "options.h"

/* Defines */
#define INBUFSIZ (IN_BUF_SIZE - SPARE)
#define MEMERR 2
#define IOERR 1
```c
#define MAXBITS 13
#define CLEAR 256  /* clear code */
#define Z_EOF 257  /* end of file marker */
#define FIRST_FREE 258  /* first free code */
#define MAXMAX 10000  /* max code + 1, was 8192 */
#define SPARE 5

typedef struct {
    unsigned next;
    char z_ch;
} tabentry, *tab;

extern COutput *gOutput; table;

int init_dtab();
unsigned rd_dcode();
int wr_dchar();
int ad_dcode();

unsigned lzd_sp = 0;
unsigned lzd_stack[STACKSIZE + SPARE];

int push(ch)
int ch;
{
    lzd_stack[lzd_sp++] = ch;
    if (lzd_sp >= STACKSIZE) {
        ZooError(5); /* Stack overflow */
        return -1;
    }
    return 0;
}

#define pop() (lzd_stack[--lzd_sp])

char in_buf_adr[OUT_BUF_SIZE + SPARE];
char memflag = 0; /* memory allocated? flag */
unsigned cur_code;
unsigned old_code;
unsigned in_code;
unsigned free_code;
int nbits;
unsigned max_code;
char fin_char;
char k;
unsigned masks[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0x1ff, 0x3ff, 0x7ff, 0xfff, 0x1fff };
unsigned bit_offset;
int in_han, out_han;
```
MacBooz Source Code

Lempel-Ziv decompression. These routines are based on the LZD routines in Rahul Dhesi's Zoo archival program. Modifications and simplifications were made to try to fit this into a more modular, object-oriented structure.

int lzd(input_handle)
int input_handle;
{
    long size;
    int err;
    char ch;

    in_han = input_handle; /* make it avail to other fns */
    nbits = 9;
    max_code = 512;
    free_code = FIRST_FREE;
    lzd_sp = 0;
    bit_offset = 0;

    /* Grab the necessary memory here */

    table = (tab) NewPtr( (long) MAXMAX*sizeof(tabentry) + SPARE);
    if (table == OL)
        ZooError(4);
    size=INBUFSIZ;
    FSRead(in_han, &size, in_buCadr);
    if_dtab(); /* initialize table */

    loop:

    cur_code = rd_dcode();
    if (cur_code == Z_EOF)
    {
        DisposPtr(table);
        return;
    }

    if (cur_code == CLEAR)
    {
        init_dtab();
        fin_char = k = old_code = cur_code = rd_dcode();

        addbfcrc(&k,1);
        err = gOutput->Write(&k,1);
        if ( err == WRITE_ERROR )
        {
            DisposPtr(table);
            return;
        }
MacBooz Source Code

goto loop;

in_code = cur_code;
if (cur_code >= free_code) /* if code not in table (k<w>h<k<k) */
{
    cur_code = old_code;            /* previous code becomes current */
    err = push(fin_char);
}

while (cur_code > 255) /* if code, not character */
{
    err = push(table[cur_code].z_ch);    /* push suffix char */
    cur_code = table[cur_code].next;   /* <w> := <w>.code */
}

k = fin_char = cur_code;
err = push(k);
while (lzd_sp != 0)
{
    /* Following copies from wr_dcharO. the following k's were
     * ch's... */
    ch = pop();
    addbfrc(&ch,1);
    err = gOutput->Write(&ch,1);
    if (err == WRITE_ERROR)
    {
        DisposPtr(table);
        return;
    }

    /* The following code comes from ad_dcode() to replace the jump. */
    table[free_code].z_ch = k;        /* save suffix char */
    table[free_code].next = old_code; /* save prefix code */
    free_code++;
    if (free_code >= max_code)
    {
        if (nbits < MAXBITS)
        {
            nbits++;
            max_code = max_code << 1;        /* double max_code */
        }
    }

    old_code = in_code;
    goto loop;
}
```c
/* rd_dcode() reads a code from the input (compressed) file and returns its value. */

unsigned rd_dcode()
{
    long size;
    register char *ptra, *ptrb; /* miscellaneous pointers */
    unsigned word; /* first 16 bits in buffer */
    unsigned byte_offset;
    char nextch; /* next 8 bits in buffer */
    unsigned ofs_inbyte; /* offset within byte */

    ofs_inbyte = bit_offset % 8;
    byte_offset = bit_offset / 8;
    bit_offset = bit_offset + nbits;

    if (byte_offset >= INBUFSIZ - 5)
    {
        int space_left;
        bit_offset = ofs_inbyte + nbits;
        space_left = INBUFSIZ - byte_offset;
        ptrb = byte_offset + in_bucadr; /* point to char */
        ptra = in_bucadr;
    /* we now move the remaining characters down buffer beginning */
    while (space_left > 0)
    {
        *ptra++ = *ptrb++;
        space_left--;
    }
    size=byte_offset;
    FSRead(in_han,&size,ptra);
    byte_offset = 0;
}
    ptra = byte_offset + in_bucadr;
    word = (unsigned char) *ptra; ptra++;
    word = word « (unsigned char) *ptra) « 8; ptra++;
    nextch = *ptra;
    if (ofs_inbyte != 0)
    {
        /* shift nextch right by ofs_inbyte bits */
        /* and shift those bits right into word */
        word = (word >> ofs_inbyte) l ((unsigned)nextch) << (16-ofs_inbyte));
    }
    return (word & masks[nbits]);
}

int init_dtab()
```
MacBooz Source Code

{  
nbits = 9;
max_code = 512;
free_code = FIRST_FREE;
}

/* Getfile routine is for archive entries that are not compressed. */

int getfile(inpuchan, count)
int inpuchan;
long count;
{
  long           how_much;
  long           hold;
  unsigned char  buffer[512];

  while (count > 0)
  {
    if (count > 512)
      how_much = 512;
    else
      how_much = count;

    count -= how_much;
    hold=(long) how_much;
    FSRead(inpuchan, &hold, buffer);
    addbfrc (&buffer, (int) how_much);
    gOutput->Write( (char *) &buffer, how_much);
  
  }

  return (0);
}

#include "Archivist.h"
#include "options.h"
#include "func.h"

extern unsigned int crecode;
extern COutput *gOutput;

int oozext (zoo_han,direntry)
int zoo_han;
struct direntry direntry;
{
  Str63 whatFile;
  int    this_han;
  int    status,i;
MacBooz Source Code

```c
COutput *outObject;

if (direntry.major_ver > 1 ||
    (direntry.major_ver == 1 && direntry.minor_ver > 0))
    ZooError(6); /* Higher version needed */

if ((direntry.namlen == 0) || (strncpy(&direntry.lfname, "," , 2) == 0))
    strcpy(&whatFile, &direntry.fname);
else
    strcpy(&whatFile, &direntry.lfname);

status = gOutput->Open(&whatFile);

if (status == OPEN_ERROR)
    return OPEN_ERROR;
else if (status == OPEN_CANCEL)
    return OPEN_CANCEL;

SetFPos (zoo_han, 1, direntry.offset);

crccode = 0; /* Initialize CRC before extraction */

if (direntry.packing_method == 0)
    status = getfile (zoo_han, direntry.size_now);
else
{
    if (direntry.packing_method == 1)
        status = lzd (zoo_han);
    else
        ZooError(6); /* Higher version needed */
}

if (direntry.file_crc != crccode)
    ZooError(3); /* Bad CRC */

gOutput->SetEndOfFile ( (long) direntry.org_size);
gOutput->Close();
}
```

#include "Archivist.h"
#include "func.h"

#define _NewSysPtr dc.w 0xa11e+0x600

CenterDialog (theDialog)
DialogPtr theDialog;
{
    Rect theRect;

    theRect.top = screenBits.bounds.bottom/2;
```
MacBooz Source Code

theRect.left = screenBits.bounds.right/2;
theRect.top -= (theDialog->portRect.bottom-theDialog->portRect.top)/2;
theRect.left -= (theDialog->portRect.right-theDialog->portRect.left)/2;

MoveWindow(theDialog, theRect.left, theRect.top, true);

unsigned char *NewSysPtr(size)
long size;
{
    asm
    {
        move.l size, DO
        _NewSysPtr
        move.l A0, DO
    }
}

long to_long();
int to_int();

/*
 * to_long() converts four consecutive bytes, in order of increasing
 * significance, to a long integer. It is used to make Zoo independent
 * of the byte order of the system.
 */

long to_long(data)
BYTE data[];
{
    int i;
    unsigned long retval;

    retval = ((unsigned) data[2] & 0xff) |
        (((unsigned) data[3] & 0xff) << 8);
    retval <<= 16;
    retval = (((unsigned) data[0] & 0xff) |
        (((unsigned) data[1] & 0xff) << 8));
    return (long) retval;
}

/*
 * to_int() converts two consecutive bytes, in order of increasing
 * significance, to an integer, in a machine-independent manner
 */

int to_int(data)
BYTE data[];
{
    return (int) (((unsigned) data[0] & 0xff) |
        (unsigned) (data[1] & 0xff) << 8));
}
typedef unsigned long size_t;

char *strcpy(void);
char *strcat(void);
int strcmp(void);
char *strncpy(void);
char *strneat(void);
int strncmp(void);
size_t strlen(void);
char *strcpy(/* char *s1, char *s2 */)
{
    asm {
        movea.4(sp),a0          ; A0 = s1
        movea.8(sp),a1          ; A1 = s2
        move.1 a0,d0
        @1
        move.b (a1)+,(a0)+
        bne.s @1
    }
}

char *strcat(/* char *s1, char *s2 */)
{
    asm {
        movea.4(sp),a0          ; A0 = s1
        movea.8(sp),a1          ; A1 = s2
        move.1 a0,d0
        @1
        tst.b (a0)+
        bne.s @1
        subq.1 #1,a0
        @2
        move.b (a1)+,(a0)+
        bne.s @2
    }
}

int strcmp(/* char *s1, char *s2 */)
{
    asm {
        movea.4(sp),a0          ; A0 = s1
        movea.8(sp),a1          ; A1 = s2
        moveq #0,d0
        bra.s @2
        @1
        tst.b d1
        beq.s @4
        @2
        move.b (a0)+,d1
    }
}
MacBooz Source Code

```c
char *
strcpy(/* char *s1, char *s2, size_t n */)
{
    asm {
        move.l 4(sp),d0 ; D0.L = result
        movea.l d0,a0 ; A0 = s1
        movea.l 8(sp),a1 ; A1 = s2
        move.l 12(sp),d1 ; D1.L = n
        beq.s @3
        move.b (al),(a0)+
        beq.s @2
        addq.l #1,a1
        @2
        subq.l #1,d1
        bne.s @1
    }
}

char *
strncat(/* char *s1, char *s2, size_t n */)
{
    asm {
        move.l 4(sp),d0 ; D0.L = result
        movea.l d0,a0 ; A0 = s1
        movea.l 8(sp),a1 ; A1 = s2
        move.l 12(sp),d1 ; D1.L = n
        beq.s @3
        tst.b (a0)+
        bne.s @1
        subq.l #1,a0
        @2
        move.b (al)+,(a0)+
        beq.s @3
        subq.l #1,d1
        bne.s @2
        clr.b (a0)
        @3
    }
}

int
strncmp(/* char *s1, char *s2, size_t n */)
{
    asm {
        move.l 4(sp),d0 ; D0.L = result
        movea.l d0,a0 ; A0 = s1
        movea.l 8(sp),a1 ; A1 = s2
        move.l 12(sp),d1 ; D1.L = n
        beq.s @3
        move.b (al)+,(a0)+
        beq.s @3
        subq.l #1,d1
        bne.s @2
        clr.b (a0)
        @3
    }
}
```

```
MacBooz Source Code

```assembly
moveq #0,d0 ; D0.L = result
movea.l 4(sp),a0 ; A0 = s1
movea.l 8(sp),a1 ; A1 = s2
move.l 12(sp),d1 ; D1.L = n
bra.s @2
@1
tst.b d2
beq.s @4
subq.l #1,d1
@2
beq.s @4
move.b (a0)+,d2
cmp.b (a1)+,d2
beq.s @1
bhi.s @3
subq.l #2,d0
addq.l #1,d0
@3
@4 }
}

tst.b (a0)+
bne.s @1

size_t
strlen(char *s *)
{
    asm {
        moveq #-1,d0 ; D0.L = result
        movea.l 4(sp),a0 ; A0 = s
        @1
        addq.l #1,d0
        tst.b (a0)+
        bne.s @1
    }
}

Interface Code

#define _H_CArdialog

struct CArDialog : indirect
{
    DialogPtr
cDialog;
    ListHandle
cList;
    DialogPtr
cSelectDialog;

    void Create(Str63 title);
    void Add(Str63 *item);
    void GetNextSelect(int *prev);
    int Match(char *string, char *pattern);
    int Select(EventRecord theEvent);
    void CleanUp(void);
    int CheckEvent(EventRecord theEvent);
};

#include "Archivist.h"
```
MacBooz Source Code

#include <EventMgr.h>

pascal void UpdateList(DialogPtr theDialog,int itemNo);

void CArcDialog::getNextSelect(prev)

int *prev;

{ Point theCell;
  Boolean found;
  int i, err;

  /* Local Copies */
  ListHandle localList;

  localList = cList;

  theCell.h = 0;
  theCell.v = *prev;

  found = LGetSelect(true,&theCell,localList);

  if (found)
    *prev = theCell.v;
  else
    *prev = -1;

  return;
}

void CArcDialog::Create(title)

Str63 title;

{ /* Local variables needed for this routine */

  Rect viewRect, dataBounds, theItemRect;
  Point csize;
  int itemType;
  Handle theItemHandle;

  /* Local copies of object variables */
  ListHandle localList;
  DialogPtr localDialog;

  /* Build modeless dialog window */

  SetDAFont(BOOZ_FONT);
  localDialog = GetNewDialog(133, 0, -1);
  SetWTitle((WindowPtr)localDialog,title);
  SelectWindow((WindowPtr)localDialog);
MacBooz Source Code

SetPort(localDialog);
SetDAFont(0);

csize.v=0;
csize.h=199;
SetRect(&viewRect,10,10,200,200);
SetRect(&dataBounds,0,0,1,0);

GetDItem(localDialog,2,&ItemType,&theItemHandle,&theItemRect);
SetDItem(localDialog,2,itemType,UpdateList,&theItemRect);

localList=LNew(&viewRect,&dataBounds,csize,0,localDialog,
true,false,false,true);
SetWRefCon((WindowPtr)localDialog,(long)localList);
LActivate(true,localList);
LDoDraw(false,localList);

this->cList = localList;
this->cDialog = localDialog;
this->cSelectDialog = 0L;

}

void CArcDialog::Add(item)
Str63 *item;
{
    /* Local variables */

    Point theCell;

    /* Local copies of object variables */

    ListHandle localList;
    DialogPtr localDialog;

    localList = this->cList;
    localDialog = this->cDialog;

    theCell.h=0;
    theCell.v=-1;

    theCell.v=LAddRow(1,32767,localList);
    LSetCell(item,strlen(item),theCell,localList);
    LUpdate(thePort->visRgn,localList);

}

void CArcDialog::CleanUp()
{
    /* Local copies of object variables */
MacBooz Source Code

ListHandle
DialogPtr

localList = this->eList;
localDialog = this->eDialog;

LDispose(localList);
DisposDialog(localDialog);
if (cSelectDialog != 0L)
{
    localDialog = cSelectDialog;
    DisposDialog(cSelectDialog);
}

int CArcDialog::Match(string, pattern)
char *string;
char *pattern;
{
    char *psave, *ssave; /* back up pointers for failure */
    psave = ssave = ((char *) 0);
    while (1)
    {
        for (; *pattern == *string; pattern++, string++) /* skip first */
            if (*string == '\0')
                return(1); /* end of strings, succeed */
        if (*string != '\0' && *pattern == '?') {
            pattern++;
            /* '?', let it match */
            string++;
            } else if (*pattern == '*') {
                /* '*' ...*/
                psave = ++pattern;
                /* remember where we saw it */
                ssave = string;
                /* let it match 0 chars */
                } else if (ssave != ((char *) 0) && *ssave != '\0') { /* if not at end */
                    /* ...have seen a star */
                    string = ++ssave;
                    /* skip 1 char from string */
                    } else
                        pattern = psave;
                        /* and back up pattern */
                } else
                    return(0);/* otherwise just fail */
    }

int CArcDialog::Select(theEvent)
EventRecord theEvent;
{
    int itemHit;
    Handle theItem;

MacBooz Source Code

```c
Str255 theText;
int itemType;
Rect box;
Str255 listText;
int dataLen,numRows,i;
Point theCell;

/* Local Copies */

ListHandle localList;
DialogPtr localDialog;

localList = cList;
itemHit = 0;

if (FrontWindow() != cSelectDialog)
    return;

localDialog = cSelectDialog;
switch (theEvent.what)
{
    case mouseDown:
        if (DialogSelect(&theEvent,&localDialog,&itemHit))
        {
            switch (itemHit)
            {
                case 1:
                    GetItm(localDialog,4,&itemType,&theItem,&box);
                    GetIText(theItem,&theText);
                    PtoCstr((char *) &theText);
                    theCell.h = 0;
                    numRows = (**localList).dataBounds.bottom;
                    for (i=0;i<numRows;i++)
                    {
                        dataLen = 64;
                        theCell.v = i;
                        LGetCell(&listText,&dataLen, theCell,localList);
                        listText[dataLen] = 0;
                        if (this->Match((char *) &listText,(char *))
                            &theText) == 1)
                            LSetSelect(true, theCell,localList);
                    }
            break;
                case 2:
                    theCell.h = 0;
                    numRows = (**localList).dataBounds.bottom;
                    for (i=0;i<numRows;i++)
                    {
                        theCell.v = i;
                        LSetSelect(false, theCell,localList);
                    }  
```
break;

case 3:
    theCell.h = 0;
    numRows = (**localList).dataBounds.bottom;
    for (i=0;i<numRows;i++)
    {
        theCell.v = i;
        LSetSelect(true, theCell, localList);
    }
    break;
}
break;

default:
    if ( theEvent.what == keyDown )
        if ( (theEvent.message & charCodeMask) == 13 )
            return;
    if ( IsDialogEvent(&theEvent) )
        DialogSelect(&theEvent, &localDialog, &itemHit);
    break;

}

int CArcDialog::CheckEvent(theEvent)
{
    EventRecord theEvent;
    WindowPtr whatWindow;
    int winLoc;
    int itemHit;
    Rect dragRect;
    Point localPt;

    /* Local Copies */
    DialogPtr localDialog;
    MenuHandle localMenu;
    int localClose;
    ListHandle localList;

    localDialog = cDialog;
    localList = cList;

    LDoDraw(true, localList);

    localPt = theEvent.where;
    GlobalToLocal(&localPt);
    dragRect = screenBits.bounds;

    winLoc = FindWindow(theEvent.where, &whatWindow);
    if ((whatWindow!=cDialog) && (whatWindow!=cSelectDialog))
        return ARC_NULL;
MacBooz Source Code

if((theEvent.what==activateEvt)||(theEvent.what==updateEvt))
{
    DialogSelect(&theEvent,&localDialog,&itemHit);
    return ARC_NULL;
}

if(theEvent.what==mouseDown)
    switch(winLoc)
    {
        case inMenuBar:
            break;
        case inDrag:
            DragWindow(whatWindow, theEvent.where, &dragRect);
            break;
        case inGoAway:
            if(TrackGoAway(whatWindow, theEvent.where))
            {
                if(whatWindow==cDialog)
                    return ARC_CLOSE;
                else
                {
                    DisposDialog(whatWindow);
                    cSelectDialog = 0L;
                    return ARC_NULL;
                }
            }
            break;
        case inContent:
            if (FrontWindow()!=whatWindow)
            {
                SelectWindow(whatWindow);
                return ARC_NULL;
            }
            if (whatWindow==cDialog)
            {
                if (DialogSelect(&theEvent,&localDialog,&itemHit))
                {
                    switch(itemHit)
                    {
                        case 1:
                        case 2:
                            return ARC_EXTRACT;
                        if
                        (LClick(localPt, theEvent.modifiers, localList))
                            return ARC_EXTRACT;
                        break;
                        case 3:
                            if (cSelectDialog==0L)
                            {
                                SetDAFont(BOOZ_FONT);
                            }
                        break;
                    }
                }
            }
        break;
    }
MacBooz Source Code

GetNewDialog(134,0,-1);

if (eSelectDialog!=0L)
    this->Select(theEvent);
return ARC_NULL;

pascal void UpdateList(WindowPtr theDialog, int itemNo)
{
    short int itemType;
    short int item;
    Rect box;
    ListHandle whatList;
    GrafPtr savePort;

    GetPort(&savePort);
    SetPort(theDialog);
    GetDItem(theDialog,itemNo,&itemType,&item,&box);
    InsetRect(&box,-1,-1);
    FrameRoundRect(&box,1,1);
    whatList=(ListHandle) GetWRefCon((WindowPtr) theDialog);
    LUpdate( theDialog->visRgn, whatList );
    SetPort(savePort);
}

#include "Archivist.h"

#include <MenuMgr.h>

extern MenuHandle gMenus[3];
extern ZooArchive *gZoo;
extern COutput *gOutput;
MacBooz Source Code

extern CHelp

* gHelp;

DoMenu( theChoice )
long theChoice;
{
    int theMenu, theItem, count, err;
    Point where = { 100, 100 };
    Str255 prompt = "Choose an archive: ";
    SFReply reply;
    Str255 name;
    char mark;
    DialogPtr aboutBox;

    theItem = (int) LoWord( theChoice );
    theMenu = (int) HiWord( theChoice );

    switch ( theMenu )
    {
        case 130: /* Apple */
            switch ( theItem )
            {
                case 1:
                    aboutBox = GetNewDialog( 129, 0L, -1 ); /* About box */
                    CenterDialog( aboutBox );
                    ShowWindow( aboutBox );
                    ModalDialog( 0L, &theItem );
                    DisposDialog( aboutBox );
                    break;

                case 2:
                    gHelp->Create();
                    break;

                default:
                    GetItem( gMenus[0], theItem, &name );
                    OpenDeskAcc( &name );
                    break;

            }
            break;

        case 131:
            switch ( theItem )
            {
                case 1:
                    SFGetFile( where, prompt, 0L, -1, 0L, &reply );
                    if ( reply. good )
                        err = gZoo->Open( reply.fileName, reply.vRefNum );
                    /* if ( err != 0 )
                        { ZooError(1); 
                    } */
                    break;

        default:
            break;
    }
MacBooz Source Code

case 2:
gZoo->Close();
break;

case 4:
gZoo->Close();
gHelp->CleanUp();
ExitToShell();
break;

} break;

} break;

case 132:
switch (theItem)
{

case 1:
delete(gOutput);
gOutput = (COutput *) new(COutput);
Uncheck();
CheckItem(gMenus[2],1,true);
break;

case 3:
delete(gOutput);
gOutput = (CCrToLf *) new(CCrToLf);
Uncheck();
CheckItem(gMenus[2],3,true);
break;
}
break;

} HiliteMenu(0);

Uncheck()
{
    int i;
    for (i=1;i<5;i++)
        CheckItem(gMenus[2],i,false);
}

DisplayDialog()
{

    DialogPtr theDialog,whatDialog;
    long oldTime;
    EventRecord theEvent;
    int i, itemHit;

    oldTime = TickCount();
    theDialog = GetNewDialog(135,0L,-1L);
    CenterDialog( theDialog );
    ShowWindow( theDialog );
MacBooz Source Code

for (i=0;i<20;i++)
{
    GetNextEvent(everyEvent,&theEvent);
    if ( IsDialogEvent(&theEvent) )
        DialogSelect(&theEvent,&whatDialog,&itemHit);
}

while ( (!Button()) && (TickCount()-oldTime < 480) )
{
    SystemTask();
}

DisposDialog(theDialog);

#include <FileMgr.h>
#include <ToolboxUtil.h>

unsigned char *NewSysPtr(long);

typedef struct NMRec *NMRecPtr;

int FileError(err)

int err;
{
    short          strNum;
    Str255         theString;

    switch (err)
    {
    case badMDBErr:
        strNum = 1;
        break;
    case bdNamErr:
        strNum = 2;
        break;
    case dirFulErr:
        strNum = 3;
        break;
    case dskFulErr:
        strNum = 4;
        break;
    case dupFNErr:
        strNum = 5;
        break;
    case eofErr:
        strNum = 6;
        break;
    case extFSErr:
        strNum = 7;
        break;
    }
case fBsyErr:
    strNum = 8;
    break;

case fLckdErr:
    strNum = 9;
    break;

case fnfErr:
    strNum = 10;
    break;

case fnOpnErr:
    strNum = 11;
    break;

case fsRnErr:
    strNum = 12;
    break;

case gfpErr:
    strNum = 13;
    break;

case ioErr:
    strNum = 14;
    break;

case memFullErr:
    strNum = 15;
    break;

case noErr:
    return;

case noMacDskErr:
    strNum = 16;
    break;

case nsDrvErr:
    strNum = 17;
    break;

case nsvErr:
    strNum = 18;
    break;

case opWrErr:
    strNum = 19;
    break;

case paramErr:
    strNum = 20;
    break;

case permErr:
    strNum = 21;
    break;

case posErr:
    strNum = 22;
    break;

case rfNumErr:
    strNum = 23;
    break;

case tmfoErr:
MacBooz Source Code

    strNum = 24;
    break;
    case volOffLinErr:
        strNum = 25;
        break;
    case volOnLinErr:
        strNum = 26;
        break;
    case vLckdErr:
        strNum = 27;
        break;
    case wrPermErr:
        strNum = 28;
        break;
    case wPrErr:
        strNum = 29;
        break;
    default:
        strNum = 30;
        break;
}

GetIndString(theString,(short) 130,strNum);
PostMessage(theString);

ZooError(theErr)
int theErr;
{
    Str255 theString;

    GetIndString(theString,(short) 131,theErr);
    PostMessage(theString);
}

pascal MyResponse(nmReqPtr)
    NMRecPtr nmReqPtr;
{
    NMRemove(nmReqPtr);
    nmReqPtr->nmRefCon = 0L;
    DisposPtr(nmReqPtr->nmStr);
    DisposPtr(nmReqPtr);
}

PostMessage(passString)
    StringPtr passString;
{
    EventRecord theEvent;
    NMRecPtr lNote;
    StringPtr lString;
/* Allocate memory for error posting routines */

lNote = (NMRecPtr) NewSysPtr( sizeof(NMRec) );
String = (StringPtr) NewSysPtr( sizeof(Str255) );

PtoCstr( (char *) passString );
strcpy( lString, passString);
CtoPstr( (char *) lString );

lNote->qType = nmType;
lNote->nmMark = 0L;
lNote->nmSIcon = 0L;
lNote->nmSound = (Handle) -1L;
lNote->nmStr = (StringPtr) lString;
lNote->nmResp = (ProcPtr) MyResponse;
NMInstall( lNote );

SystemTaskO;
WaitNextEvent(everyEvent,&theEvent,60,thePort->visRgn);

#include <OSUtil.h>

Boolean CommInstalled(void);

void Initialize()
{
    SysEnvRec theEnv;
    int err;

    InitGraf(&thePort);
    InitFontsO;
    FlushEvents( everyEvent, 0 );
    InitWindowsO;
    InitMenusO;
    TEnInitO;
    InitDialogs(OL );
    InitCursorO;

    SysEnviron(1,&theEnv);
    if (theEnv.systemVersion < 6)
        SysBeep(4);

    /* CommToolbox Inits

    if (CommInstalled())
    {
        err=InitCRMO;
    }
MacBooz Source Code

```c
if (err!=0)
    SysBeep(4);
err=InitCTBUtility();
if (err!=0)
    SysBeep(4);
err=InitCM();
if (err!=0)
    SysBeep(4);
err=InitTM();
if (err!=0)
    SysBeep(4);
err=InitFT();
if (err!=0)
    SysBeep(4);
}

End of CommToolbox Inits */

Boolean CommInstalled() {
    long val1, val2;
    val1 = NGetTrapAddress(0x8B, OSTrap);
    val2 = NGetTrapAddress(0x9F,OSTrap);
    if (val1==val2)
        return false;
    else
        return true;
}

#include <MenuMgr.h>

extern MenuHandle gMenus[];

void InitMenus(void) {
    gMenus[0] = GetMenu(130);
    InsertMenu(gMenus[0],0);
    AddResMenu(gMenus[0], (ResType) DRVR);
    gMenus[1] = GetMenu(131);
    InsertMenu(gMenus[1],0);
    gMenus[2] = GetMenu(132);
    InsertMenu(gMenus[2],0);
    DrawMenuBar();
    if (CommInstalled()) {
        gMenus[3] = GetMenu(133);
        InsertMenu(gMenus[3],0);
    }
```
MacBooz Source Code

```c
#include "Archivist.h"
#include <FileMgr.h>
#include <EventMgr.h>
#include <StdFilePkg.h>
#include <ListMgr.h>
#include <pascal.h>

MenuHandle gMenus[4];
ZooArchive *gZoo;
COutput *gOutput;
CHelp *gHelp;

int saveAll;

void Initialize(void);
void InitMenus(void);
void DoMenu(long);
void CenterDialog(DialogPtr);
int lzd(int);

main()
{
    EventRecord myEvent;
    int message, err, theRetNum, i;
    long theChoice;
    WindowPtr theWindow;
    AppFile theStartupFile;

    Initialize();
    InitMenus();

    /* Create an archive object. */
    gZoo = new(ZooArchive);
    gZoo->Init(gMenus[1], 0, 1, 2);

    /* Create an output object */
    gOutput = new(COutput);
    gOutput->Init();
    CheckItem(gMenus[2], 1, true);

    /* Create a new help object */
    gHelp = new(CHelp);
    gHelp->Init();

    UnloadSeg(Initialize);
```
MacBooz Source Code

/* Figure out if someone wanted to open a file on startup */

CountAppFiles(&message,&i);

if (message==1)
{
    PostMessage("How do you print an archive file?");
    ExitToShell();
}

if (i>0)
{
    GetAppFiles(1,&theStartupFile);
    if (theStartupFile.fType = 'Zoo')
    {
        theRefNum = theStartupFile.vRefNum;
        err = gZoo->Open(theStartupFile.fName,theRefNum);
        ClrAppFiles(1);
    }
}
else
    DisplayDialog();

/* Main event loop */

while (1)
{
    UnloadSeg(CenterDialog);
    UnloadSeg(Izd);

    SystemTask();
    WaitNextEvent(everyEvent,&myEvent,60,thePort->visRgn);

    gZoo->CheckEvent(myEvent);
    gHelp->CheckEvent(myEvent);

    switch (myEvent.what)
    {
    case mouseDown:
        switch (FindWindow(myEvent.where,&theWindow))
        {
            case inSysWindow:
                SystemClick(&myEvent,theWindow);
                break;
            case inMenuBar:
                theChoice=MenuSelect(myEvent.where);
                DoMenu(theChoice);
                break;
        }
        break;
MacBooz Source Code

```c
#define _H_COutput
#define SUCCESSFUL 0
#define OPEN_ERROR -1
#define WRITE_ERROR -2
#define OPEN_CANCEL -3

struct COutput : indirect {
    int Reply;
    int cRefNum;
    DialogPtr cDialog;
    int cCount;
    int cSaveAll;
    void Init();
    void Open(Str63 *tName);
    void Write(char *data, int numChars);
    void SetEndOfFile(long eof);
    void Close();
};

#include "Archivist.h"
#include <pascal.h>
#include <StdFilePkg.h>
#include <EventMgr.h>

extern int saveAll;

void COutput::Init()
{
    cRefNum = 0;
}
```
pascal int SaveAllHook(item, theDialog)
int item;
DialogPtr theDialog;
{
    if (item<9)
    {
        saveAll = 0;
        return item;
    }
    if (item==9)
    {
        saveAll = 1;
        return 1;
    }
}

int COutput::Open(fName)
Str63 *fName;
{
    OSErr err;
    refNum,i,oldSaveAll;
    int where;
    Point y;
    Str255 SFReply theReply;
    DialogPtr localDialog;

    where.h=100;
    where.v=100;

    strcpy(&y,fName);
    CtoPstr( (char *) &y);

    TryAgain:

    oldSaveAll = saveAll;

    if (!saveAll)
        SFPPutFile(where,"\pSave file as: ",y,SaveAllHook,&theReply,136,0L);
    else
    {
        strcpy(&(theReply.fName),y);
        theReply.vRefNum = cReply.vRefNum;
    }

    if (theReply.good)
    {
        err=Create(theReply.fName, theReply.vRefNum, 'MWII', 'TEXT');
        if ((err!=noErr) && (saveAll==1) && (oldSaveAll==1))
        {
            saveAll=0;
        }
    }
goto TryAgain;

}  

err = FSOpen(theReply.fName, theReply.vRefNum, &refNum);
if (err != noErr)
{
    FILEError(err);
    cRefNum = 0;
    cReply = theReply;
    return OPEN_ERROR;
}
cRefNum = refNum;
cReply = theReply;

SetDAFont(BOOZ_FONT);
localDialog = GetNewDialog(131, 0L, -1);
ParamText( y, "", ",", ",");
ShowWindow((WindowPtr) localDialog);
DrawDialog(localDialog);
SetDAFont(0);
cDialog = localDialog;
return 0;

}  

else
{
    cRefNum = 0;
    cReply = theReply;
    return OPEN_CANCEL;
}

}

int COutput::Write(data, numChars)
char *data;
int numChars;
{
    int localRef, winLoc, itemHit;
    long size;
    EventRecord localEvent;
    WindowPtr theWindow;
    Rect dragRect;
    OSErr err;
    DialogPtr localDialog;

    localDialog = cDialog;
    localRef = cRefNum;
    dragRect = screenBits.bounds;

    if ( .50 == cCount++)
    {
        SystemTask0;
MacBooz Source Code

GetNextEvent(everyEvent,&localEvent);
winLoc = FindWindow(localEvent.where,&theWindow);

if (localEvent.what == mouseDown )
{
    if (winLoc == inDrag )
        if (theWindow == localDialog)
            DragWindow(theWindow,localEvent.where,&dragRect);
}
else if (localEvent.what == updateEvt )
{
    if (IsDialogEvent(&localEvent))
    {
        theWindow = (WindowPtr) localEvent.message;
        DialogSelect(&localEvent,&theWindow,&itemHit);
    }
}
cCount = 0;

size = (long) numChars;
err = FSWrite(localRef,&size,data);
if (err != noErr )
{
    FileError(err);
    return WRITE_ERROR;
}
return 0;

void COutput::SetEndOfFile(eof)
long eof;
{
    int localRef,err;
    long size;

    localRef = cRefNum;
    err = SetEOF(localRef,eof);
}

void COutput::Close()
{
    int localRef,err;
    long size;
    DialogPtr localDialog;

    localDialog = cDialog;
    localRef = cRefNum;
    DisposDialog(localDialog);
### MacBooz Source Code

```c
err = FSClose(localRef);
if (err != noErr )
    FileError(err);

#include "Archivist.h"

extern MenuHandle gMenus[3];

Convert(buf,count)
unsigned char *buf;
int count;
{
    int i;
    char mark;
    for (i=0;i<count;i++)
    {
        if ( *buf = 0xOA )
            *buf = 0x0D;
        if ( *buf = 0x00 ) /* Change Nulls to 0x0D */
            *buf = 0x0D;
        buf++;
    }
}

#include "Archivist.h"

int CCrToLf::Write(data,numChars)
char *~
int numChars;
{
    int size;
    EventRecord localEvent;
    WindowPtr theWindow;
    Rect dragRect;
    char *temp;
   OSErr err;
    DialogPtr localDialog;

    localDialog = cDialog;
    localRef = cRefNum;
    dragRect = screenBits.bounds;

    if ( 50 == cCount++)
    {
        SystemTask();
        GetNextEvent(everyEvent,&localEvent);
    }
```
MacBooz Source Code

winLoc = FindWindow(localEvent.where,&theWindow);

if (localEvent.what == mouseDown)
{
    if (winLoc == inDrag)
        if (theWindow==localDialog)
            DragWindow(theWindow,localEvent.where,&dragRect);
    else if (localEvent.what == updateEvt)
    {
        if (IsDialogEvent(&localEvent))
        {
            theWindow = (WindowPtr)localEvent.message;
            DialogSelect(&localEvent,&theWindow,&itemHit);
        }
    }

cCount = 0;
}

temp = data;
for (i=0;i<numChars;i++)
{
    if (*temp == 0xOA)
        *temp = 0xOD;
    temp++;
}

size = (long)numChars;
err = FSWrite(localRef,&size,data);
if (err!=noErr)
{
    FileError(err);
    return WRITE_ERROR;
}

return 0;

CHelp Class Routines

#define _H_CHelp

struct CHelp : indirect
{
    DialogPtr cDialog;
    void Init(void);
    void Create(void);
MacBooz Source Code

```c
void CleanUp(void);
int CheckEvent(EventRecord theEvent);

#include "Archivist.h"

void CHelp::Init(void)
{
    cDialog = 0L;
}

void CHelp::Create(void)
{
    /* Local variables needed for this routine */
    Rect viewRect, theItemRect;
    int itemType;
    Handle theItemHandle;
    Str255 theString;

    /* Local copies of object variables */
    DialogPtr localDialog;

    if (cDialog != 0L)
    {
        localDialog = cDialog;
        SelectWindow(localDialog);
        return;
    }

    /* Build modeless dialog window */
    SetDAFont(BOOZ_FONT);
    localDialog = GetNewDialog(137, 0L, -1);
    SelectWindow((WindowPtr) localDialog);
    SetPort(localDialog);
    SetDAFont(0);
    GetIndString(&theString, 132, 1);
    ParamText(&theString, "", "", "");
    SetWRefCon((WindowPtr) localDialog, 1L);
    this->cDialog = localDialog;
}

void CHelp::CleanUp(void)
{
    DialogPtr localDialog;
```
if (cDialog==0L)
    return;

localDialog = cDialog;
Disposition(localDialog);
cDialog = 0L;
}

int CHelp::CheckEvent(IEventRecord theEvent)
{
    WindowPtr whatWindow;
    int winLoc;
    int itemHit;
    Rect dragRect;
    Point localPt;
    int pictNum;
    Str255 theString;
    DialogPtr savedPort;

    /* Local Copies */
    DialogPtr localDialog;
    localDialog = cDialog;
    dragRect = screenBits.bounds;
    pictNum = GetWRefCon((WindowPtr) localDialog);

    if (cDialog==0L)
        return 0;

    winLoc = FindWindow(theEvent.where,&whatWindow);
    if (whatWindow!=eDialog)
        return 0;

    if ((theEvent.what == activateEvt) || (theEvent.what == updateEvt))
    {
        DialogSelect(&theEvent,&localDialog,&itemHit);
        return 1;
    }

    if (theEvent.what==mouseDown)
        switch (winLoc)
        {
            case inMenuBar:
                break;
            case inDrag:
                DragWindow(whatWindow,theEvent.where,&dragRect);
                break;
            case inGoAway:
                break;
        }
MacBooz Source Code

if (TrackGoAway(whatWindow, theEvent.where))
{
    this->CleanUp();
    return 1;
}
break;
case inContent:
    if (FrontWindow() != whatWindow)
    {
        SelectWindow(whatWindow);
        return ARC_NULL;
    }
    if (whatWindow == cDialog)
    {
        if (DialogSelect(&theEvent, &localDialog, &itemHit))
        {
            switch (itemHit)
            {
            case 1: /* Ignore userItem */
                return 1;
            case 2: /* Previous */
                pictNum--;
                if (pictNum < 1) pictNum = 6;
                break;
            case 3: /* Next */
                pictNum++;
                if (pictNum > 6) pictNum = 1;
                break;
            }
            GetIndString(&theString, 132, pictNum);
            ParamText(&theString, "", "", "");
            SetWRefCon((WindowPtr) localDialog, (long) pictNum);
            GetPort(&savedPort);
            SetPort(localDialog);
            EraseRect(&localDialog->portRect);
            InvalRect(&localDialog->portRect);
            SetPort(savedPort);
        }
    }
return 0;
**Bibliography**


Schmucker, Kurt J. “Object-Oriented Languages for the Macintosh,” Byte, August 1986, pp. 177-185.


