Miscellaneous

COLLABORATORS

	TITLE : Miscellaneous	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Miscellaneous

1.1 Chapter 8 - Miscellaneous

```
                Previous Chapter:
 7. Advanced AmigaDOS Routines
-------------------------------------------------------------


               CHAPTER 8 - MISCELLANEOUS


                 Introduction

                 The Global Error Code

                 How to Execute Shell Commands from a C Program

                 Examples
```

1.2 Introduction

```
INTRODUCTION

In this chapter I will describe some very useful AmigaDOS
functions which did not fit in the other chapters. Although
some knowledge about AmigaDOS is required to understand this
chapter you do not have to read all previous chapters before
this one. The  introduction chapter
```

1.3 The Global Error Code

```
THE GLOBAL ERROR CODE

When a dos function fails it will usually store some
information about the error, an error code, in a special global
variable. You can get a copy of this error code with the help
```

```
of the  IoErr()

There are three good reasons why you should call the IoErr()
function when you encounter a dos error:

  1. First you need to inform the user and it is then of course
     good to know what really have happened so you can tell the
     user to take appropriate action.

  2. The more your program knows about the error the easier it
     is for your program to solve it.

  3. Some times you need to check if it really was an error or
     not when a dos function failed. For example, in the
     previous chapter you read about how to list all objects in
     a directory. When no more objects can be found the
     function,  ExNext()
     "real" error since there are of course a limited number of
     objects in a directory, and that dos function should
     "fail" when all objects have been listed. It is in these
     situations you have to check what the actual error was -
     if there were not any more objects in the directory it is
     OK, but if it was something else which caused the function
     to fail there was a "real" error.

See example 1 for more information:

   Read!    Run!    Edit!


Some dos functions,  FRead() , were badly written
and do not "clear" the error code before they are executed. It
is therefore impossible to know if the value in the global
error variable is valid after you have executed these
functions - it might be an old value in the global error
variable.

To solve this problem you must therefore "clear" the global
error variable yourself (set the variable to 0) before you
execute these badly written functions.

To alter the global error variable simply call the  SetIoErr()
function.

Although your own program also can use this global error
variable with the help of SetIoErr() and IoErr(), it is not
recommended. It is better to use your own variables if needed.
```

1.4 How to Execute Shell Commands from a C Program

```
HOW TO EXECUTE "SHELL" COMMANDS FROM A C PROGRAM

You can execute "Shell" commands from C with the help of the
 Execute()
type of programs (not only the standard commands like "dir",
```

"list", "type" etc...) which you can access from the "Shell".

See example 2 for more information:

   Read!   Run!   Edit!

1.5 Examples

EXAMPLES

Example 1: Read!   Run!   Edit!
  This example demonstrates how to use the IoErr()
  It contains a simple but rather useful function which will
  try to give the user some more information about the last
  dos error. Good for debugging!

Example 2: Read!   Run!   Edit!
  This example demonstrates how to execute "Shell" commands
  from a C program with the help of the Execute()