Objective C

SYNOPSIS

This document describes the implementations of the Objective C Language running under various implementations of the UNIX operating system. Topics include creation, compilation, and execution of an Objective C program. Also covered in a separate section is a discussion of user-created Objective C object libraries. It is assumed that the reader has a working knowledge of Objective C and a UNIX text editor such as vi.

OBJECTIVE C LANGUAGE INTRODUCTION

There are many implementations of the C language available on the various UNIX systems at Cal Poly. In this document we will be addressing the compilers which fall into the Objective C category as opposed to "standard" C or C++ which are addressed in their own respective sections.

The compilers involved boil down to the GNU Project C compiler named "*gcc*". The following table describes the platform, the name of the compiler, and any special considerations.

Hardware Platform	Command Name	Compiler Vendor
NeXT	*CC* or *cc++*	GNU ANSI-C Compiler with Objective-C and C++ extensions
Other UNIX systems	*gcc*	GNU ANSI-C Compiler with Objective-C

NAMING CONVENTIONS

Objective C source programs must follow standard UNIX naming conventions. The Objective C source file must end with ".c" or ".m". Please refer to the man page for the specific compiler parameters and possible extensions.

CREATE A SOURCE FILE

Use any available text editor, such as vi, to create your source file. (See the Information Systems User Guide "vi Editor" which is included in the "Introduction to UNIX" User Guide bundle from El Corral Bookstore.) You may also bring the source in from another system using such facilities as Kermit, FTP, or xmodem.

COMPILE THE C SOURCE PROGRAM

The Objective C compiler has many flags. In the following example, replace the word "*cc_cmd*" with the name of the compiler from the table on page 1. The general syntax of the Objective C compilers is

```
% cc_cmd [-c] [-Dname[=value]] [-E] [-g] [-Ipathname] [-lkey]
    [-Lpathname] [-o executable_name] [-O] [-ObjC] [-ObjC++]
    [-Uname] [-w] [-Wall] sourcefile
    [sourcefile ... sourcefile]<CR>
```

where the options are defined as

-c	Compile the source only, do not call the linker (*ld*).
-Dname[=value]	Define *name* as in a #define directive. If *=value* is not specified, *1* is assumed.

-E	Preprocess, but do not compile; output to stdout (standard output).
-g	Produce debug information for use with debuggers such as "*dbx*".
-Ipathname	Search the directory specified by "*pathname*" for include files which do not start with an absolute path.
-lkey	Selects the library lib*key*.a to be searched for unresolved references by *ld*.
-Lpathname	Search the directory specified by "*pathname*" for the libraries specified by *-lkey*.
-o name	Name the executable file *name* instead of a.out.
-O	Optimize code generation.
-ObjC	Indicate that the source file contains Objective C and should be compiled with the Objective C compiler. Not necessary if the filename has an extension of ".m".
-ObjC++	Allow Objective-C or C++ in source (NeXT only, see note below).
-Uname	Undefine name as in #undef directive.
-w	Suppress informational, language-level, and warning messages.
-Wall	Show all warnings.
sourcefile [sourcefile ... sourcefile]	One or more source filenames.

Additional parameters may be found by viewing the man page for the compiler on its system by entering

```
% man cc_cmd<CR>
```

where "*cc_cmd*" is the name of the compiler command you wish to use.

NOTE: Use of the *-ObjC++* requires that Objective-C and C includes be bracketed with the extern directive, for example

```
extern "C" {
#include <stio.h>
}
extern "Objective-C" {
#include <objc/object.h>
}
```

See "*Developers Release Notes*" for more details in the "*Digital Librarian*" on the NeXTs.

EXECUTE AN OBJECTIVE C EXECUTABLE

To execute your compiled program, you simply type the name of the object module which was produced from the compile step. The default is "a.out". The Objective C compiler does have the capability of producing an object module with a different name, but for the purposes of this example, we will assume that the object module name is "a.out". To execute it you enter

```
% a.out<CR>
```

This will cause the program to be loaded and begin execution.

COMPILING IN MULTIPLE STEPS

Let us assume in this example we have a program that is composed of three different source files. These source files ("myprog1.c", "myprog2.c", and "myprog3.c") are all located in the current working directory. To compile them all, then produce a single object file, would be accomplished by executing the Objective-C compiler call *cc_cmd* from the table on page 1 or the *ld* commands. This is done by typing

```
% cc_cmd -ObjC myprog1.c myprog2.c myprog3.c<CR>
```

This can also be accomplished in multiple steps by entering

```
% cc_cmd -ObjC -c myprog1.c<CR>
% cc_cmd -ObjC -c myprog2.c<CR>
% cc_cmd -ObjC -c myprog3.c<CR>
% ld myprog1.o myprog2.o myprog3.o -lc<CR>
```

or

```
% cc_cmd -ObjC -c myprog1.c<CR>
% cc_cmd -ObjC -c myprog2.c<CR>
% cc_cmd -ObjC myprog3.c myprog1.o myprog2.o<CR>
```

This compiles the three separate source files. They are then linked to the standard C library and produce an object module named "a.out". **NOTE:** When "*ld*" is used to link the modules together, the C library must be specified and placed in order. For example:

```
% ld myprog3.o myprog1.o myprog2.o -lc<CR>
```

To compile and link the routine "myprog.c" to additional system subroutine library files "libcurs.a" and "libcurses.a", you would type

```
% cc_cmd -ObjC myprog.c -lcurs -lcurses<CR>
```

Note that only the portion "*curs*" and "*curses*" are used from the library file name. When this is done as part of the "*ld*" step or as part of the *cc_cmd* command, the additional subroutine libraries must be indicated as the last items on the command line.

HOW TO USE THE INTERACTIVE DEBUGGER

There is an interactive debugger available for the Objective C compiler. Please refer to the section entitled "Debugging Tools" for more information on using these interactive debuggers.

COMMAND SUMMARY

NOTE: In each case substitute "*cc_cmd*" with the proper command name from the table on page 1.

`% cc_cmd -ObjC filename<CR>` Compile the Objective C source file specified by *filename* and place the object module on "a.out".

Compile the routines defined in "*myprog1.c*", "*myprog2.c*", and "*myprog3.c*". Once they are all compiled, link them into the object module "*a.out*".

```
% cc_cmd -ObjC -c myprog1.c<CR>
% cc_cmd -ObjC -c myprog2.c<CR>
% cc_cmd -ObjC -c myprog3.c<CR>
% ld myprog1.o myprog2.o myprog3.o -lc<CR>
```

`% gdb a.out<CR>` Run a program under the interactive debugger on the current site where the a.out file is from the C compiler.

`% cc_cmd -ObjC myprog.c -lsubs<CR>`
Compile the routine defined in "myprog.c" and link it to the additional library "libsubs.a".

NOTES