

# Pascal

## SYNOPSIS

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

**PASCAL LANGUAGE INTRODUCTION** There are several implementations of the Pascal language available on the various UNIX systems at Cal Poly. In this document we will be addressing the compiler which falls into the "standard" Pascal category as supplied on the IBM RISC/6000 Systems.

Hardware Platform	Command Name	Compiler Vendor
RISC/6000 AIX 3	<i>xlp</i>	IBM XLP Extended Pascal Compiler

**NAMING CONVENTIONS** Pascal source programs must follow standard UNIX naming conventions. The Pascal source file must end with ". pas".

**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 PASCAL SOURCE PROGRAM** The general syntax of the XL Pascal compiler call is

```
% xlp [-c] [-g] [-Ipathname] [-o executable_name] [-O] [-p] [-pg]
      [-U] sourcefile [sourcefile ... sourcefile]CR>
```

where the options are defined as

- c Compile the source only, do not call the linker (*ld*).
- 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.
- o *name* Name the executable file *name* instead of *a.out*.
- O Optimize code generation.
- p Generate simple profiling support code.
- pg Generate BSD profiling support code.
- U Preserves upper and lower case letters in the source code.

*sourcefile [sourcefile ... sourcefile]*

One or more source filenames.

Additional parameters may be found by viewing the man page by entering

```
% man xlp<CR>
```

#### EXECUTE A PASCAL 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". *xlp* 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. pas", "myprog2. pas", and "myprog3. pas") are all located in the current working directory. To compile them all, then produce a single object file, would be accomplished by executing the *xlp* command. This is done by typing

```
% xlp myprog1. pas myprog2. pas myprog3. pas<CR>
```

This can also be accomplished in multiple steps by entering

```
% xlp -c myprog1. pas<CR>
```

```
% xlp -c myprog2. pas<CR>
```

```
% xlp myprog3. pas myprog1. o myprog2. o<CR>
```

This compiles the three separate source files. They are then linked to the standard Pascal library and produce an object module named "a. out".

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

```
% xlp myprog. pas -l curs -l curses<CR>
```

Note that only the portion "curs" and "curses" are used from the library file name. When this is done as part of the *xlp* 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 each of the Pascal compilers on most of the campus UNIX systems. Please refer to the section entitled "Debugging Tools" for more information on using these interactive debuggers.

#### COMMAND SUMMARY

```
% xlp filename<CR>
```

Compile the Pascal source file specified by *filename* and place the object module in "a. out".

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

```
% xlp -c myprog1. pas<CR>
```

```
% xlp -c myprog2. pas<CR>
```

```
% xlp myprog3. pas myprog1. o myprog2. o<CR>
```

`% dbx a.out<CR>`

Run a program under the interactive debugger on the current site where the *a.out* file is from the Pascal compiler.

`% xlp myprog.pas -lsubs<CR>`

Compile the routine defined in "*myprog.pas*" and link it to the additional library "*libsubs.a*".

**DOCUMENT CODE: UNIX-41301A**

**DATE REVISED: August 25, 1994**

## **NOTES**