LIBRARY CLEARANCE

1. (a) I give my permission for my thesis, entitled

....Concurrent...Pascal:...Implementation...and...Usage

.....

.....

to be made available to readers in the Library under the conditions determined by the Librarian.

(b) I agree to my thesis, if asked for by another institution, being sent away on temporary loan under conditions determined by the Librarian.

(c) I also agree that my thesis may be copied for Library use.

2. I do not wish my thesis, entitled

.....

.....

.....

to be made available to readers or to be sent to other institutions without my written consent within the next two years.

Signed ....MHenders.... MARTIN L HENDER

Date ....20/9/83....

Strike out the sentence or phrase which does not apply.

The Library
Massey University
Palmerston North, N.Z.

The copyright of this thesis belongs to the author. Readers must sign their name in the space below to show that they recognise this. They are asked to add their permanent address.

Name and Address Date

.....

.....

.....

.....

.....

CONCURRENT PASCAL:

IMPLEMENTATION AND USAGE

by

Martin L. Hender

A thesis presented in fulfilment

of the requirement for the degree of

Master of Science in Computer Science

at

Massey University

September 1983.

ACKNOWLEDGEMENTS

In presenting this thesis I would like to take this opportunity to express my thanks to the following people:

First and foremost to my supervisor, Ray Kemp, whose continued guidance, criticism and encouragement throughout this project has been most helpful and much appreciated.

Secondly to Dr. Steve Black for his thorough reading of this thesis and the many suggested changes. His help has definitely improved the presentation of this work.

Lastly I would like to thank the staff of the Massey University Computer Centre for their encouragement and interest in this project.

Massey University — Martin L. Hender

September 1983

Table of Contents

Chapter 0

Introduction

In the forty years that the modern electronic computer has existed it has gone through several phases. The third generation computer of today is far removed in sophistication from the early first generation machines. In just the same way as the hardware of these machines has changed or been enhanced through the years, so has the associated software on them. The early computers required the programmer to use machine language or even set switches on the machine, while today the programmer can often use one of several high level programming languages.

Whereas the early machines were used mainly for number crunching, the computer of today is asked to perform a multitude of tasks. To cater for these wide and varied applications the programming languages of today have themselves divided into several areas. Each area designed to support a certain application. There are languages to support numeric scientific problems (e.g. Fortran), data processing processing problems (e.g. Cobol) and list processing (e.g. Lisp) to name but a few. In the course of this thesis one of these applications, that of implementing operating systems, will be covered along with the corresponding languages that have been developed to support it.

Systems languages are relatively new in the software scene with them first appearing around ten years ago. This project will be concentrating on one of these languages, Concurrent Pascal [Brinch Hansen 75], a forerunner in the field. An operating system for the IBM Series/1 has been written in the language and

through this implementation an insight into the power and usefulness of the language has been gained. Brinch Hansen, the designer of Concurrent Pascal, has pointed out that "the language will no doubt have deficiencies" [Brinch Hansen 77b] and this implementation has hopefully brought these weaknesses to the surface. A discussion of facilities offered by other languages in the area is included to compliment those offered by Concurrent Pascal.

Chapter one describes the facilities of Concurrent Pascal that were designed for the systems programming field. Three such facilities are studied in detail, the class, the process and the monitor. It is suggested that the process is the main feature which separates the systems language from others. In the operating system of today several tasks commonly have to be performed simultaneously. The process or an equivalent mechanism is the way a programmer achieves this required concurrency. With several processes working independently though, there is a need in some cases to synchronize them. To support this Concurrent Pascal uses the monitor construct, although other system languages may use totally different mechanisms. The systems type languages are therefore themselves divided into sections depending upon the synchronizing mechanism they use.

In chapter two the implementation of the language Concurrent Pascal on an IBM Series/1 is described. Several problems concerned with portability were found in this part of the project. Through these difficulties several lessons have been learned in respect to producing machine independent and transportable code. It is suggested that problems will always be

encountered when moving symbolic code, or even program source code, from one computer to another. The aim must be therefore to recognise this and try to minimise these difficulties. In doing so the task of transporting the object is made far easier as less effort is required to support it on the new machine.

The implementation of a small interactive operating system (i.e. MUSCLE) is described in chapter three. This implementation has shown that Concurrent Pascal can be used to write simple general purpose operating systems. It is hoped that the implementation has been carried out in such a manner that given appropriate hardware it could easily be extended to a fully fledged general purpose system. It has however been sprinkled with difficulties which cannot entirely been blamed on the hardware or machine used. It is suggested that some of these problems are the result of shortcomings in the language. These shortcomings have been discussed and where appropriate alternatives have been suggested.

Chapter four contains some suggestions as to what properties a systems language must contain. Whereas Concurrent Pascal can be considered to be a small and very restrictive language it is suggested that this causes difficulties to the programmer. The writing of an operating system is recognised as being one of the most difficult programming tasks. The language which is used must therefore aid the programmer as much as possible. It can do this in two ways, firstly by offering a large number of facilities and secondly by relaxing restrictions on the use of certain constructs. Concurrent Pascal was found wanting in both these areas. It is suggested that the systems language must

assume that the programmer knows what he is doing and in some cases allow certain restrictions to be relaxed. Any relaxation though must be done at the specific request of the programmer. This ensures that the programmer knows that liberties are being taken and that it is obvious in the program.