DESIGN OF SYSTEMS LANGUAGES

A thesis presented in partial fulfilment
of the requirements for the degree of
Master of Science in Computer Science
at Massey University

Christopher Allen Freyberg

March 1975

Abstract

Systems Languages have often been designed on a rather ad hoc basis. This thesis attempts to formulate and analyse design criteria in a more systematic manner. These criteria are drawn from three major sections: a survey of languages used for systems programming, a discussion of systems programs features, and a discussion of programming language effectiveness. The resulting criteria are then discussed in relation to their application to the language design. A collection of language summaries is included in the appendices.

To the many people who have helped
or encouraged me in the completion of this thesis.

I would like to acknowledge by name the work of the following:

Brian Carpenter, Bob Doran, Prof. G. Tate (Supervisors)

Lloyd Thomas (Algol 68)

Lyn Stening (typist)

Table of Contents

§1 Introduction

- §1.1 Aim
- §1.2 Some Difficulties
- §1.3 Outline

§2 A Brief Survey

- §2.1 The Various Languages
- §2.2 Algol Based Languages
- §2.3 PL/I Based Languages
- §2.4 Other High-Level Languages
- §2.5 PL360 Based Languages and Assemblers
- §2.6 Summary

§3 Systems Programs : Common Features

- §3.1 Systems Programs : Definition and Scope
- §3.2 Breakdown of Processing Features
- §3.3 Collected Features

§4 Effectiveness : Functionality, Error Immunity and Efficiency

- §4.1 Functionality
- §4.2 Error Immunity
- §4.3 Efficiency

§5 Criteria and their Application to the Language Design

- §5.1 Extensibility/Orthogonality
- §5.2 Data Related Criteria
- §5.3 Control Related Criteria
- §5.4 Miscellaneous Critera
- §5.5 Conclusions

§6 Appendices

- §6.1 Language Summaries
- §6.2 A Language Design
- §6.3 Glossary
- §6.4 References

§1 Introduction

§1.1 Aim

In the past, a commonly accepted project for a masterate in the programming languages field was to take an already designed language, suitably modified, to implement on a given machine. The student thereby derived experience in implementation problems. However the usefulness of that type of project is severely limited when the language is a systems programming language, in that most machines to which the student has access already have well developed suites of systems programs. Hence rewriting a part of any suite runs foul of intercommunication problems, and rewriting the whole would be an excessively large task even for a small machine. The project would accomplish little more than an intimate knowledge of one language and one machine.

An obvious alternative, that of designing and implementing a systems language, was discarded for similar reasons; also there is already a plethora of languages of that type. Such a 'home grown' language is, moreover, only likely to gain acceptance and be used in the immediate locality unless it happens to incorporate some startlingly new and useful technique or construct. In other words, it would be little more than an exercise.

The topic finally selected, "Design of Systems Languages", was anticipated to require two reasonably distinct subprojects:

i. a survey of existing systems languages

ii. development of design criteria based on an analysis of their features

and possibly a third subproject developing a language based on those criteria. However it soon became obvious, while surveying the existing languages, that in many cases the criteria employed by their authors were neither explicit nor extensive, so that a somewhat different approach would be required, even if the basic intention remained the same.

§1.2 Some inherent difficulties

A major problem in tackling a topic such as this is that the experience (or lack of it) of the author can lead to large distortions of outlook. He attempts to survey and criticise a group of languages with a wide range of features, when the only features he has experience of are limited to those implemented on the few machines he has worked on. Coupled to this, the machines to which he at present has access colour and in some respects bias his appraisal of anything which applies to other machines, particularly to those to which he has never had access. For example, access to a stack machine leaves him with doubts about the sanity of anyone who uses 360 type parameter passing.

Realising that these two difficulties exist fortunately provides some solutions - abstraction becomes a keyword and generality becomes an overall goal. The survey of languages thus becomes a means to an end, and can be divided into two parts:

i. a gathering of information from separate sources

ii. a criticism of what has been done or not done to date.

Following this, greater emphasis can be given to determining the paramount features and linguistic requirements of the various types of systems program, instead of relying upon other people's opinions about them. Similarly it is preferable to determine for oneself the desirable features of a systems language, particularly in the shadow of considerable disagreement in the literature over machine independence of systems languages, and even over terminology, notably 'efficiency'. These disagreements point to fields of study outside the scope of this thesis.

§1.3 Outline

This thesis therefore attempts, through a survey of existing systems languages and an examination of the characteristics of systems programs, to develop a series of criteria by which a systems programming language may be judged, and through which a new language can be constructed to make it a useful tool.

Section 2 surveys existing languages by grouping them with respect to common base languages. It relies heavily upon §6.1, which is a table of the features of the various languages. The section is summarised in §2.6.

Section 3 addresses the nature of systems programs and what language features are required to write and support them. Emphasis is placed here (as in Section 2) on the language itself and the linguistic criteria, rather than on the compiler or the methodology of construction. A summary is made in §3.3.

Section 4 is an attempt to put some order into arguments about efficiency, commentation, and the general methodology of systems programs.

It also attempts to draw up guidelines for those features the compiler must provide exclusive of the language itself.

Section 5 collects together the criteria from the above three sections and, along with criteria related to extensibility, attempts to order them into a preferential system. This system is then discussed in terms of the limitations it places on, and demands it makes on, the language structure.

The conclusions of the thesis are really contained in Section 5, and for this reason are not given a separate section heading.