A Friendly Programming Environment for the ALTO Mini-Computer

Uri Shani
Computer Science Department
University of Rochester
Rochester, NY, 14627

TR54

May 1980

This work was supported in part by the Alfred P. Sloan Foundation under Grant 74-12-5.

Table of Contents

Abstract

Personal minicomputers sophisticated enough to work under a disk operating system are considered in light of their convenience as a programming environment.

Programming activities are classified as editing, compiling, loading, and so forth. Each such activity is carried out by one or more programs, called (programming) tools. The files processed by those tools are referenced via names entered by the user. Names of files involved in programming activities should be chosen to reflect file type information. Like variables in a programming language, a type of file defines what activities (operations) are applicable to it.

Guided by this idea, file naming convention should be defined by the user and stored in a user profile structure. This definition is then used by the computer's command-interpreter (executive) to decide what programming tool to activate in order to carry out a requested activity on a file.

An extension to an existing executive program for a minicomputer (16 bit, 64KW with 3Mb disk and raster CRT display) that makes use of such a user profile is described in this paper. The system utilizes good human engineering and, with minimum overhead, improves significantly the usefulness of the computer as a programming vehicle.

ACKNOWLEDGMENTS:

I wish to thank Richard F. Rashid for his encouragement, and helpful suggestions during the design and implementation of the system. Keith Knox at Xerox Research Center in Webster, New York, suggested the "commented load command." Thanks are also due to Keith A. Lantz, Anil Nigam and Mark W. Kahrs for their useful comments during the preparation of this paper.

1.Ø INTRODUCTION

A very important part of a computing system is its user interface - the collection of rules and mechanisms by which a user gains access to services provided by the machine. This is the responsibility of a program usually called the executive or command-interpreter. A user communicates with the executive by use of a command language. The executive will perform file maintenance functions and the like. In particular, the executive will invoke other programs (also called subsystems). If the machine is used for programming, it will include subsystems for text editing, compiling, and loading of user written programs. These subsystems are termed programming tools.

Each of these tools operates on input file(s) and produces output file(s). Input and output files may have different properties depending on their purpose. For example, an input file to a Fortran compiler will not work for an Algol compiler. An executive, which invokes a programming tool irrespectively of the program, will allow the user to mistakenly feed the wrong file to the wrong tool. In an active programming site, where these tools are extensively used, special commands for performing programming activities should be provided by the executive so that the user will avoid such mistakes.

This need was recognized by many system designers and computer users. It led to the development of compile-class-commands in the TOPS1Ø operating system for the DEC PDP1Ø computer [1] and the development of the Rapid Program Generator - RPG [2]. In these systems, a distinguished logical part of the file name is the extension, used to decide which tool will perform a particular compile class command. A set of conventions for file extensions was set by the system designers. The SDS94Ø [3] is a time sharing system in which file types are encoded internally in the files (*) in a limited variety for use by the system.

At Rochester, we extensively use the ALTO minicomputer [4] for programming. It is a 16 bit 64KW machine with a keyboard, raster CRT display and 3M bytes removable disk storage (Fig. 1); it runs under a disk operating system (DOS) that is similar in capabilities to OS6 [5,6].

---

* A user can not tell the file's type by looking at its name.

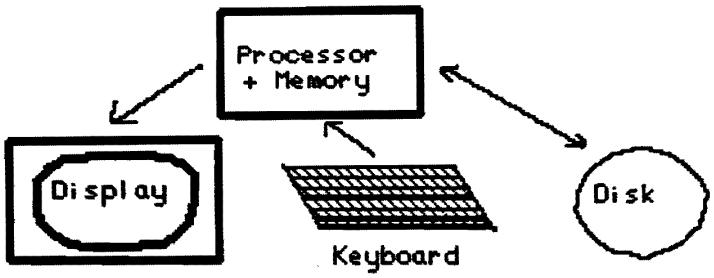

Fig. 1: ALTO machine configuration.

The executive program on the ALTO is simple and does not have any bias towards programming activities. File names do have extensions that are used by the executive when invoking a subsystem (a subsystem name is expected to have the extension "run"). I have extended the executive capabilities to include the commands Edit, Compile and Load. These commands use file extensions as file-types to decide what tool should be invoked for a particular programming activity. The notion of file type was expanded so that it is defined by the user in a user profile. In it, associations between file extensions, programming activities and programming tools are defined and can be easily changed. As our system is dynamic, when new tools are added they can immediately be incorporated into the existing programming environment. Some varieties of user profile exist in many systems (the "switch.ini" file in the TOPS10, a profile structure in TSO [7], and the "user.cm" file in the Alto), but they are not used for defining file types.

File extension does not have to be fully specified; a partially specified extension is matched against all possible extensions of existing files with the same name, in a priority order defined by the user, and the best match is chosen. If no file name is given in the command line, then the name used in the last command invocation is taken (i.e. ellipsis).

These additions to the executive do not involve any changes to existing programming tools or the executive itself. They are simple, easy to implement, and are suitable to any other personal computer of a configuration similar to that of the Alto. This extension is considered the friendly programming environment.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.