ISSN 0254-2757

QI QUAESTIONES INFORMATICAE

Volume 4 Number 3 October 1986

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Africa en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

QUÆSTIONES INFORMATICÆ

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Africa en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Subscriptions

Annual subscriptions are as follows:

	SA	US	UK
Individuals	R10	$ 7	£ 5
Institutions	R15	$14	£10

Circulation and Production

Mr C.S.M. Mueller
Department of Computer Science
University of the Witwatersrand
1 Jan Smuts Avenue
Johannesburg, 2001

Quæstiones Informaticæ is prepared by the Computer Science Department of the University of the Witwatersrand and printed by Printed Matter, for the Computer Society of South Africa and the South African Institute of Computer Scientists.

WHAT'S WRONG WITH CP/M?

S.P. Byron-Moore
Department of Computing Science
University of Zimbabwe
Harare
Zimbabwe

Trends in operating systems are examined in the light of advances in computer hardware. Consideration is given to the desirability of a good, up-to-date single-user operating system. The specification and implementation of such a system is discussed.

1. INTRODUCTION

1.1 Software

At the present time, most discussions about operating systems include mention of UNIX, a multi-user operating system developed by Bell Laboratories in the early 1970's. The aim of its designers was to "create a computing environment ... where they themselves could comfortably and effectively pursue their own work - programming research" [1]. Due to the management policies of Bell Laboratories, UNIX has only recently become commercially available, althougth it has been widely used by Bell and some universities since 1971. This system is being proposed by many authors and salesmen as the answer to everyone's dreams - "the operating system of the future", for both commercial and research usage.

1.2 Hardware

In the 16 years since UNIX was first conceived, advances in computer hardware have changed the face of the computer industry. Microcomputers have become common and due to their low price, many businesses, as well as researchers and hobbyists, have invested in small and often single-user systems. The introduction of microcomputers opened up new sales markets and 'price wars' raged as manufacturers competed for this lucrative trade. Consequently prices for hardware such as random-access memory and fast access, high capacity storage devices such as Winchester disc drives have been lowered considerably. Increasing miniaturisation of components and improved technology have allowed computer firms to produce portable (or at least transportable) computers, which have rapidly gained an expanding market.

1.3 The Future

With the reduction in prices of hardware for microcomputers, many researchers and business users are now able to have dedicated microcomputers sitting on their desks, which satisfy most of their computing needs. This is a rapidly expanding market - "Future Computing", a US research group, predicts that portable computer shipments alone will exceed 1 million units in 1988 [2]. More and more microcomputers are being used as stand-alone machines, with the capability of communicating with larger computers via networks. Many users have become accustomed to having a microcomputer for their own personal use, without having to share its facilities with other users. Given the current and anticipated increase in the number of personal computers, many for single-user applications, it seems ironical that UNIX, a large multi-user operating system, born in the age of mainframe and minicomputer dominance, should be coming to prominence.

2. SINGLE-USER COMPUTER SYSTEMS

2.1 Advantages

The main advantages of single-user operating systems compared to multi-user operating systems are:

- small size
- rapid response time
- versatility

A single user operating system does not have to include the security and accounting features which are necessary in a multi-user system, consequently the operating system is of a smaller size. This means that operating system loading time is reduced, execution time is lessened and more memory space is left free for the user.

System versatility is an important advantage of a single-user system. The user of a dedicated microcomputer can tailor his system to his own requirements and preferences. Like the owner of a bicycle, who adjusts his machine for his personal use, the user of a single-user system can adapt this system for efficiency and ease of use. For example, the user may devote part of the machine's memory space for usage as a disc emulator [4]. He may rename operating system commands to provide the type of operating system interface he prefers [5]. He may set up a menu-driven interface to the operating system. Such personalisation would be banned or seriously frowned upon in a multi-user operating system. Here several users may have differing requirements, so a user cannot be permitted to adapt the system for his own needs. Furthermore , users of a multi-user system, very reasonably, expect consistency from the system they are using. Thus this type of system cannot be easily adapted.

When a computer is to be utilised in a single-user environment, a single-user operating system is highly desirable since it can be more efficient than a multi-user operating system and is far more easily adaptable to its user's tastes and needs.

2.2 Current Operating Systems

Most single-user operating systems currently available have their deficiencies. Perhaps the most well known, CP/M 2.2 [6] is almost as old as UNIX. Its speed of disc access is painfully slow and its directory structure does not accomodate high capacity discs. CP/M Plus [13] supports bank switching to exploit extra memory and facilitates multi-record reads to speed disc access, but it still has a single-level directory structure and is fairly difficult to implement. MS-DOS 2.0 has a hierarchical directory structure and multiple sector buffers to speed disc access but has a strange set of system calls, because it is a compromise between CP/M and XENIX. It provides only a print spooler and not full concurrency.

Most operating system development work seems to be focused on multi-user systems. UNIX, for example, has many nice features [9], which are desirable in an operating system. Although UNIX is now being put on to microcomputers, albeit in a reduced form, it is still a multi-user system, not appropriate to small single-user systems. Digital Research has recently released Concurrent PC-DOS, a multi-user, multi-tasking operating system for 16-bit microcomputers, but this occupies 156KB of memory and is fairly slow [7].

3. PROPOSED SYSTEM

3.1 General

Given that there is now, and will be in the future, an increasing number of single user computer systems, we must consider what facilities are required in a modern single-user operating system.

3.2 Structured Directory System

Owners of single-user machines have traditionally had limited disc storage available and have been content to use a single level directory structure like that of CP/M 2.2. However, with the price of mass storage devices dropping, users are tending to keep many more filenames on a particular device. Since no user wants to see a hundred or more files displayed on the screen when he requests a directory listing, the operating system needs to support a suitably structured directory system in order to provide for fast and efficient location and retrieval of files.

A popular method for structuring the directory allows the user to set up a hierarchical file system. This system has great advantages in a multi-user operating system since it facilitates easy separation of different user's files. However, a hierarchical system can be rather confusing to new or inexperienced users, who have some difficulty in navigating the tree structure.

An alternative method is to allow the user to divide a physical disc into a number of logical discs, each with its own directory. This is conceptually easier for a non-professional computerist to understand, but is not suitable for a user who wishes to store a number of long files. There is also a tendency for space to be wasted, due to the fragmentation of the disc. Should our proposed operating system include both of the above methods, and allow the user to set up his directories as he prefers or is there some other structure which is ideally suited to a single-user environment?

3.3 Faster Disc Access

Slow speed of disc access is one of the major disadvantages of current operating systems. Many systems read small blocks of information from disc in order to conserve memory space. CP/M 2.2, for example, reads/writes one 128 byte record at a time. Many users have increased their disc access speed under CP/M by adding code to their BIOS (the user configurable part of CP/M) to perform multi-record reads/writes. CP/M Plus [14] includes a new operating system function to do just this. As memory space is becoming less crucial, a modern operating system should be able to transfer large amounts of data, in one go. Many modern floppy disc controllers, for example, allow an entire track read/write. The operating system should also be able to cope with the transfer of this amount of information.

With the increasing availability of memory, cache buffering is also a useful feature. One of the good points of MS-DOS, is the provision of a user-specifiable number of buffers for disc cacheing [15].

A disc track is normally divided into a number of sectors. The disc controller can only read/write a complete sector. Due to the speed of rotation of the disc, after the controller carries out a sector read/write, several more sectors may pass the disc head before the controller is ready to carry out more I/O. For this reason, many operating systems provide for a "skew" factor for disc I/O. CP/M, for example, has a standard skew of 6 sectors - this means that after reading sector one, sector seven will be read, followed by sector 13, etc.. The size of the skew factor depends on the capabilities of the hardware and the operating system overhead. Normally, when a skew factor is used, adjacent sectors on disc have contiguous numbering. With CP/M for example, a file of length 3 sectors, may be stored in physical sectors 2,8 and 14. The operating system has to map the logical sectors making up the file to the physical sectors on disc. This transformation can be avoided by renumbering the sectors on the disc so that the controller reads sectors numbered sequentially (i.e 1,2,3,4,...), although sector n-1 and sector n will not be physically adjacent on the disc. By renumbering in this way, operating system overhead is reduced and the transfer of the discs to another computer system is made more straightforward, since there is no need to know with what skew factor the disc was recorded.

3.4 Improved User Interface

The majority of operating systems now in use were written at a time when most computer users were professionals and there was little emphasis on user-friendliness. Ritchie, for example, states "Both input and output of UNIX programs tend to be very terse. This can be very

disconcerting, especially to the beginner" [10]. Menu systems are being written as add-ons to many operating systems, for example NCR's menu driven interface to UNIX [3]. In the department of computing science at the University of Zimbabwe, we have a menu-driven interface to CP/M, which has been in use for three years for research work and for eighteen months for teaching purposes. We have found it extremly useful as it removes the burden of remembering a command set, reduces the amount of typing required and facilitates rapid familiarisation with the system. All modern operating systems should provide support for a menu-system as well as providing a command entry mode for experts who wish to avoid the menu-system or carry out special tasks. There should also be an operating system utility to enable the user to tailor his menus, to take full advantage of the single-user system's versatility.

3.5 I/O Redirection

This is one of the strong points of UNIX - it allows the user to change easily the source of input and destination of output of a process. This means that program writing can be made more general, without concern for the final I/O devices. It is certainly a desirable feature in a modern single-user operating system and is not too much of a problem to implement.

3.6 Pipelines

Pipes are another UNIX feature . They allow a process to communicate with another process without having to set up and manage one or more temporary files. To implement pipes, the operating system should include a protected buffer for the transfer of data and a scanning routine to insert/remove data from this buffer. Pipes are invaluable if concurrency is allowed.

3.7 Concurrency

The frustration of waiting for one job to finish before getting on with the next job is common to most users of single-tasking systems. Concurrency is a desirable feature, but not if it noticeably degrades system performance. Awalt [7], for example, claims a 29 percent performance penalty when running two tasks concurrently (instead of sequentially), under Concurrent PC-DOS.

By forcing the user, not the operating system, to specify the priority and/or weighting of any processes that he runs concurrently, it seems likely that we can minimise the visibility of any performance degradation. Possibly we should also limit the number of concurrent processes to two, to avoid serious degradation.

Providing for concurrency inevitably imposes an added burden on the operating system, which must protect one process from another and perform complex memory management functions. A concurrent operating system will be much larger and more complicated than a single-tasking system. However, this should not prove to be major problem with the continuing reduction in the price of memory. We will probably see more and more users moving to a banked memory system, which would allow transient parts of the operating system to be rapidly moved to main memory.

Is it worth paying the penalty of supporting concurrency? A user may only want to run concurrent processes on an occasional basis. He may never want concurrency or he may require its use frequently. It seems that the best solution is to provide a separate operating system module to handle concurrency. The user must specifically invoke this module if it is required, otherwise the operating system acts as a single-tasking system.

3.8 Virtual Memory Handling

This is a very old idea, first used on mainframes in the early 1960's. The user sees no limitation on the memory space available to him and if a program is too large for memory the operating system is responsible for rolling-in and rolling-out code or data as it is required. From

the programmer's point of view this removes unnecessary limits on program size. It reduces problems regarding the number of concurrent processes that memory can hold at one time. But it does have its cost - a larger operating system size. Is virtual memory a desirable feature of a single-user operating system for microcomputers now that more memory is becoming available or does its processing overhead outweigh its advantages?

3.9 Other Considerations

The above discussion lists only the major design considerations for a new single-user operating system. There are many other lesser features that are worth considering for inclusion. For example, is it worth while providing an on-line help facility to explain the usage of operating system commands; a type ahead buffer for the terminal input and a command line editor to facilitate rapid command entry. Split-screen viewing is useful when a user wishes to perform a subsidiary task. Discussion with a number of users would help to identify which additional features are required.

No program should crash, but this is particularly important for an operating system. There must be good error trapping and reporting facilities.

4. IMPLEMENTATION

4.1 Alternatives

It seems desirable that our proposed single-user operating system should be compatible with CP/M 2.2, a well established and widely used single-user system, because:

- a large number of varied utilities have been developed for use under CP/M 2.2, many of them in the public domain [11, 12].
- many other CP/M utilities are available for moderate costs.
- there are many users of CP/M compatible operating systems(e.g. ConIX [18], RP/M [16], MRS/OS [17], C/NIX [19]), as well as users of Digital Research CP/M. The source code of CP/M 2.2, or a CP/M look-alike such as RP/M, MRS/OS can be obtained.

To maintain compatibility with CP/M 2.2 our alternatives are

- to enhance CP/M 2.2
- to write a new operating system compatible with CP/M 2.2

4.2 Enhancements to CP/M 2.2

Many of the enhancements, discussed in section 3, can be made to CP/M 2.2 without a great amount of difficulty. However, this technique is rather like building a house and than adding numerous extensions - the result is never as good as building the whole house in one operation. It seems preferable to carefully design a new efficient operating system.

4.3 New operating system.

Since the source code of CP/M is available it is possible to write a new operating system, which retains compatibility with CP/M by using the same system calls.

This new system should be written in a modular form, to make it portable to computers having different processors and make it suitable for use on machines with either small or large addressable memory. Advantages of this modular form include

- the straight forward selection of a subset of the operating system's facilities, depending on the size and capabilities of the computer system and the needs and preferences of that

system's user. For example, the type ahead buffer mentioned in section 3.8, can only be implemented in a machine with a hardware interrupt capability.
- the easy amendment of a module for a particular user's software requirements and hardware configuration.
- the operating system may easily be separated into transient and resident portions for a banked memory system.

Ideally, the new operating system should be written in a middle-level language. Assembly code is unsuitable since it is processor dependent and we require our system to be easily transportable. High-level language programs, although transportable, tend to produce unnecessarily large code files, so a high level language should not be used. The writers of UNIX attempted to avoid this problem by creating the language "C". Unfortunately UNIX is still not as easily transportable as would be liked [8]. We propose that the new operating system should be written in a macro language so that only individual small macros have to be rewritten to implement the system on a machine with a different processor.

5. CONCLUSION

There is an increasing need for a good single-user operating system. Although there are a number of sound multi-user operating systems available, most single-user systems are relatively old and outdated. A modern single-user system is needed to fill this gap in the market. In section 3, we discussed some of the main design considerations for this new system. It is important that the system be transportable to different processors, with small or large amounts of memory. Ideally it should be available in the public domain.

REFERENCES

1. M D McIlroy, E N Pinson, B A Tague, UNIX Time-Sharing System: Foreword, pp 1899-1904. *The Bell System Technical Journal.*, Vol 57 No 6 part 2 July-Aug 1978.
2. O Tucker, Joining a market on the move. pp6-8, *Computing The Magazine*, Dec 13 1984.
3. Tower System, Administration manual., *NCR*, Vols 1 and 2 Release 1.0 July 1983.
4. E Nisley, Spinning your own VDISK., pp 100-109, *PC Tech Journal* , Vol 3 No 3 Mar 1985.
5. D Fielder, the UNIX tutorial, pp 257-278, *Byte magazine*, Vol 8 No 9 Sept 1984.
6. CP/M 2.2 manual., Digital Research Inc 1979.
7. D Awalt, Concurrent PC-DOS, pp 45-54, *PC Tech Journal* , Vol 3 No 3 Mar 1985.
8. M Tilson, Moving UNIX to new machines, pp 266-276, *Byte magazine*, Vol 8 No 10 Oct 1983.
9. D M Ritchie K Thompson, The UNIX Time-Sharing System, pp 1905-1929, *The Bell System Technical Journal.*, Vol 57 No 6 Part 2 July-Aug 1978.
10. D M Ritchie, UNIX Time-Sharing System: A Retrospective, pp 1947-1969, *The Bell System Technical Journal.*, Vol 57 No 6 Part 2 July-Aug 1978.
11. SIG/M (Special Interest Group for Microcomputers, Amateur Computer Group of New Jersey), Inc. Box 97 Iselin NJ 08830. USA.
12. CPMUG (CP/M User's Group) 1651 3rd Avenue NY10028. USA.
13. B R Ratoff, Implementing the Advanced Features of CP/M Plus, pp 26-29, *Microsystems*, Feb 1983
14. B R Ratoff, Implementing the Advanced Features of CP/M Plus: Part2, pp 70-72, *Microsystems*, Apr 1983.
15. W G Wong, MS/DOS : An Overview Part 1, pp 47-52, *Microsystems* , Mar 1984.
16. RP/M. Micro Methods. Box G Warrenton OR 97146. USA.
17. MRS/OS. OCCO inc. 16 Bowman lane Westboro, MA 01581, USA.
18. ConIX. Computer Helper Industries Inc. P.O. Box 680 Parkchester Station NY 10462. USA.
    D Lunsford, Software review ConIX, pp 83-86, *Computer language* , Jun 1985.
20. C/NIX. The Software Toolworks 15233 Ventura Boulevard Suite 1118 Sherman Oaks California 91403. USA.

NOTES FOR CONTRIBUTORS

The purpose of the journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review artilces and exploratory articles of general interest to readers of the journal. The preferred languages of the journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to:

Prof. G. Wiechers
INFOPLAN
Private Bag 3002
Monument Park 0106
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. Manuscripts produced using the Apple Macintosh will be welcomed. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name and affiliation and address. Each paper must be accompanied by an abstract less than 200 words which will be printed at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Tables and figures should not be included in the text, although tables and figures should be referred to in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Figures should also be supplied on separate sheets, and each should be clearly identified on the back in pencil and the authors name and figure number. Original line drawings (not photocopies) should be submitted and should include all the relevant details. Drawings etc., should be submitted and should include all relevant details. Photographs as illustrations should be avoided if possible. If this cannot be avoided, glossy bromide prints are required.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters; between the letter O and zero; between the letter I, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetic order of the author's name, and cited in the text in square brackets. Journal references should be arranged thus:

1. Ashcroft E. and Manna Z., The Translation of 'GOTO' Programs to 'WHILE' programs., *Proceedings of IFIP Congress 71*, North-Holland, Amsterdam, 250-255, 1972.
2. Bohm C. and Jacopini G., Flow Diagrams, Turing Machines and Languages with only Two Formation Rules., *Comm. ACM*, **9**, 366-371, 1966.
3. Ginsburg S., Mathematical Theory of Context-free Languages, McGraw Hill, NewYork, 1966.

Proofs and reprints

Proofs will be sent to the author to ensure that the papers have been correctly typeset and *not* for the addition of new material or major amendment to the texts. Excessive alterations may be disallowed. Corrected proofs must be returned to the production manager within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Only orginal papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.