Katrine Tjoelsen
6.033 Systems Engineering
Paper Critique 2 – Unix
6 March 2015

# A Unix systems critique

## INTRODUCTION

The Unix time-sharing system described in the paper by Ritchie and Thompson aligned well with the priorities of the designers – i.e. it (1) made it easy for programmers to "write, test, and run programs", (2) had a "certain elegance of design" that resulted in low hardware requirements and (3) could "maintain itself" (section VIII). These design decisions had certain tradeoffs in terms of scalability, fault-tolerance, and security, and this paper will discuss how the system performs relative to its use cases along these dimensions.

The system is the fourth version of the Unix time-sharing system and an improvement over previous versions of Unix. It is an interactive multi-user operating system that allows users to create, edit, and manage files, as well as share files between themselves. An operating system consists of many modules. The paper by Ritchie and Thompson focuses specifically on the file system and the user command interface modules of the system and how these two modules interact.

## BACKGROUND

A good system is designed to be scalable, fault-tolerant, and trustworthy according to its use cases. This section will consider specific features of the Unix system along these dimensions.

*Scalability*. The design of the Unix operating system has several features that allow it to scale. First, while it is impossible to predetermine and pre-allocate how much space a file needs, the file system instead allows files to dynamically grow as needed. When one creates a file, it is allocated an i-node entry and an initial block of 512 bytes. As a file grows, the system implements levels of indirect addressing to other parts of the hard drive (section IV). This ability to dynamically handle changes in file size results in higher disk utilization. Second, the hierarchical design of the file system makes it both fast to

search and easy to navigate for a user even with large numbers of files. That enables the system to have a large number of files with limited impact on performance and user-friendliness. Third, hard links allow multiple users to have access to the same files without copying the files and requiring large amounts of additional space (section IV).

*Fault Tolerance*. The Unix design was made with the principle that simple is better than complex and more correct (Gabriel, section 2.1), and that sometimes implies accepting incorrectness for simplicity. A notable aspect is that the file system maintains no locks visible to the user, and this means that when multiple users edit the same file concurrently, the file contents can become scrambled. The paper claims that "locks are neither necessary nor sufficient" (section 3.6), because the practical problems are in its current use case arguably small.

*Security and Trustworthiness*. Security was not a major priority for the designers of the system as is evidenced by the limited attention the Ritchie and Thompson paper pays to it. The main security features of the system are (1) to allow users to login with a name and password (section 6.7), and (2) to specify read/write/execute permissions for files and directories (section IV).

## ANALYSIS

Consider first the use cases of the system. One major use case is to allow programmers to write, test, and execute programs. The file system hierarchy allows many compact and modular files, allowing and encouraging programmers to apply modular design principles. A command or a program is also a file, so programmers can easily create programs as files of a certain type. On the other hand, treating everything as a file can have performance disadvantages. This is especially true when dynamically allocating more space for a file, because by placing portions of a file in different areas of the disk it will take longer to read and write to long sequential parts of that file. The designers made a tradeoff prioritizing generality over performance.

A second major use case is to allow multiple users to share files, and one can imagine the system being used in for example a collaborative academic laboratory setting. The i-node

file design with hard links is beneficial since it allows these multiple users to have access to the same files. However, the lack of measures to account for concurrency when multiple users edit the same file is a disadvantage for this use case of collaborative file editing. In this case, the designers made a design decision to prioritize simplicity of design over fault-tolerance.

## CONCLUSION

The authors claim that the Unix time-sharing system successfully accounts for the main priorities of the designers, and there is validity to this claim. The system is general and modular in its design and thus ready for change, simple and compact in size, and useful for programmers. On the other hand, in order to achieve this simplicity the system is in some ways less fault-tolerant, and the high level of generality can result in lower performance of Unix versus other systems that are designed for particular use cases.

Bibliography

Gabriel, Richard P. *The Rise of Worse is Better*. 1991.

Ritchie, D.M. and K. Thompson. *The UNIX Time-Sharing System*. 1974.