Ritchie74

Important Things

1. This paper introduces a system that established the state of the art of the time, beating its contemporary systems in various aspects. It costs very little for an interactive operating system. It has intricately designed hardware that is highly efficient. The software is for the first time written in C language which gives rise to many functional improvements and clarity.
2. The file system is an epitome of the interactive quality of a system. It categorizes files into three main types and introduces concise symbols used for navigating throughout the system, setting various access rights for different groups of users.
3. Surprising fact that there are no pre-defined objectives for developing a system like UNIX. The system was not a product with strict specifications but something that reflects the actual goals of programmers. All the functionalities reflect practical needs of programmers in having an interactive, hospitable, and self-maintaining computer environment.

Deficiency

There is no comparison to previous systems on how were things used to be done. This makes it difficult for readers to appreciate the novelty and the frequently claimed superiority of UNIX.

Future

This paper points out some important preferences during system design. Programmers are naturally inclined to making the programs better. If a system provides programmers with effective interface that allows modifications to be done promptly and reliably, it will be able to ‘maintain itself’. Such systems are able to evolve swiftly over time and are thus desirable.

Lampson83

Important Things

1. Top thing is defining interface and it should follow a minimalistic mindset. Do not generalize and try to create a powerful and mighty program, since the generalization we assume at the beginning are usually not accurate, and focusing on small, individual tasks makes implementation faster and directly addresses objectives.
2. Making implementations work is an intrinsically iterative process. The paper suggests that even extending previous functionalities of a system requires a whole new level of redesigning. It proposes the idea of keeping multiple prototypes and getting prepared to forego the worse part. This fundamentally challenges the idea of updating, which we usually perform by changing only parts of the original system.
3. Speed is secondary in comparison to reliability. Speed and fault-tolerance are two separate qualities and shall not be closely linked during the design of a program.

Deficiency

This paper, as the author suggests, contains a lot of details and anecdotal comments that do not form a clearly structured whole. It does not address conflict among hints, which leaves the reader confused about how to take all the advice into account and weighing their importance during specific use cases.

Future

This paper provides useful advice for developers of large systems. It addresses some common situations developers encounter at different stages of the development. Thinking about why something helps making a system and where it helps is a universal approach to systems design.