A Report on “The Structure of the THE Multiprogramming System”

by

Edsger W. Dijkstra

Summary

The paper is a report on the activities involved in developing the THE multiprogramming system. It outlines some interesting and new approaches to the design of a multiprogramming operating system as mentioned below:

- The distinction between memory units in core and drum termed as *pages* and the corresponding information units named as *segments* is a new concept. The paging scheme described in the paper reduces memory access latency and resolves drum allocation problem, as programs now no longer need to occupy consecutive drum pages.

- Description of the entire system in terms of abstract cooperating sequential processes and use of *semaphores* for explicit synchronization among different processes make it possible to establish harmonious cooperation among the processes through logical reasoning alone, without the need of actual testing of the system. This leads to a considerable saving of time and effort in developing a system, and its logical soundness can be proved *a priori*.

- The hierarchical structure of the system with a bottom-up design approach permits gradual incorporation of new features into the system, thus making the design and debugging process manageable. An additional benefit of the layered architecture is that it helps in better understanding of the control flow and dependencies among the components of the system.

Overall, this paper presents some new and extremely interesting approaches to the art of system design.

Main Focus of the Paper

The main focus of the paper is on the principles adopted in designing the THE multiprogramming system. The paper describes the goals of the system and the design lessons learned in course of development of the system. The major objectives of the system were:

1. Reduction in turn around time for short duration programs.


3. Providing a better memory management system with more efficient utilization of memory and processor.
4. Providing an economically feasible system for applications, which require the flexibility of a
   general-purpose computer.

In order to achieve these objectives, some new design methodologies were adopted.

- The first of these is the distinction between a logical unit of information (identified by
  segments) and a physical unit of information or pages. This not only eliminates the need for a
  program to be present in consecutive drum pages but also helps in reducing memory access
  latencies, as swapped out core page is written to the available free drum page having minimum
  latency time.

- The abstraction of the system in terms of a number of cooperating processes and explicit
  synchronization among them helps in proving the soundness of the system through mathematical
  reasoning alone, and hence, eliminates ambiguity in the behavior of the system.

- The decomposition of the system in terms of different layers of processes, with the lower
  level ones servicing those at higher layers, makes the design process much simpler and
  manageable. The bottom-up approach allows the testing of the crucial lower level processes
  before the higher level ones. This results in fewer debugging necessities. The system described in
  the paper consists of five layers from level 0 to level 4. Level 5 consists of the operators. Level 0
  deals with processor allocation, interrupt handling and synchronization among different
  processes through the $P$ and $V$ operators of semaphores. Level 1 deals with memory allocation
  and level 2 with allocation of console keyboard. At level 3 we have the abstraction of the actual
  peripherals in terms of logical communication units. Level 4 consists of user programs.

The author also introduces a new class of semaphores, which he calls private semaphores. The
idea of a private semaphore is that only the sequential process with which it is associated can
execute a $P$ - operation on it while other processes can execute only a $V$ - operation on it. It can
take on only the values -1, 0 and 1.

**Strengths of the Paper**

The paper discusses a hierarchical design approach in developing large systems. New concepts
of storage and processor allocation for designing a multiprogramming operating system have
been described. It is well written.

**Weaknesses of the Paper**

There is as such no weakness of the paper except for the fact that the system described does not
have any

- provision for a common database via which users can communicate with each other.
- support for writing user programs other than in ALGOL.
New Ideas Learnt

The paper introduces a number of new ideas on storage allocation, processor allocation and hierarchical system design leading to an efficient and robust system design. Specifically it addresses the distinction between pages and segments, use of semaphores to implement synchronization among processes and a system design approach with layered architecture. Hierarchical system design facilitates clear understanding of the dependencies among the components of the system and easier testing of each component. The bottom-up approach helps in incremental development of the system - a part is first developed, tested and incorporated and then a new component is added to the system. These software engineering approaches can be adapted to designing any other type of system as well, which can result in reduced development and debugging time.

References
