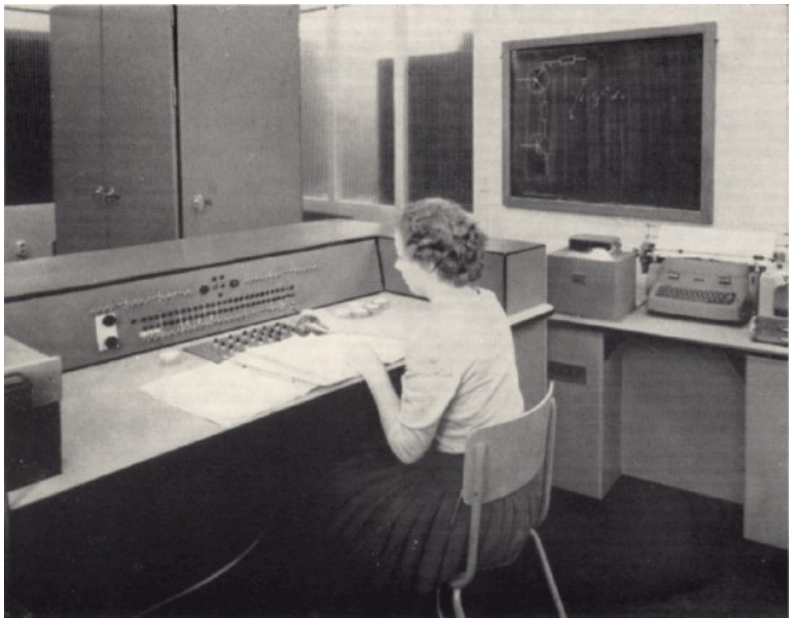
The Structure of "THE"-Multiprogramming System

Edsger W. Dijkstra

presented by Ian Elliot

“THE” hardware

- “EL X8” (Electrologica X8)
  - core memory
    - cycle 2.5µsec (400Khz)
    - 27 bits words
    - 32K (words?)



- Drum of 512K words, 1K words per track, 40ms revolution time (25 revolutions per second, up to 1600 times slower to retrieve from than core memory)

“THE” hardware

(Not much of a UI)

- Indirect addressing mechanism
- Interrupts
- Many low-capacity “channels”

“THE” goals

- Multiprogramming, not multitasking
  - Primarily batch programs
  - I/O takes a lot of time and CPU cycles are expensive
  - Run a program until it performs a slow I/O operation, then run a different program during the I/O
  - From a throughput point of view, this is attractive since an expensive context switch is only performed in the presence of a more time-consuming I/O operation instead of arbitrarily often as with multitasking

“THE” goals

- Reduction of turn-around time for short-duration programs
- Economic use of peripheral devices
- Automatic control of backing store (drum)
- “Economic feasibility to use the machine for those applications for which only the flexibility of a general purpose computer is needed, but (as a rule) not the capacity nor the processing power.” – Exactly the case multiprogramming is for.

“THE” goals

- Not intended to be a multiaccess system
- Programs share only a procedure library
- Not designed for user programs written in machine language

“THE” claims

- “At the time this was written the testing has not yet been completed, but the resulting system is guaranteed to be flawless.

  When the system is delivered we shall not live in the perpetual fear that a system derailment may still occur in an unlikely situation, such as might result from an unhappy “coincidence” of two or more critical occurrences”...

“THE” claims

- As revealed in the appendix, resolution of “critical occurrences” is solved by the use of semaphores for mutual exclusion.

- ... Using mutexes and critical sections ... guaranteed flawless?

“THE” structure

- A notion of “segments” is introduced
  - Same size as pages
  - More segment addresses than memory/drum
  - Segment identification is independent of location
- As a result, if a segment needs to be dumped onto the drum to make core pages available for other use, there is no need to return the segment to the same drum page from where it came. (Remember, 25 revolutions per second!)

“THE” structure

- Furthermore, programs need not occupy consecutive drum pages which helps mitigate the problem of allocation. (However, one must wonder how this impacts turn-around time for programs that can be written to access pages sequentially – there's no way to optimize for this case anymore.)
- Remind you of virtual memory in some ways?
  - The segment address space is shared, though?

“THE” structure

- The lack of time in sequential processes is acknowledged (only the order of steps matters)
- Input and output devices are considered sequential processes via buffering
- Blocking locks work because they only delay, not reorder, steps in a sequential process

“THE” system hierarchy

- System arranged into levels
- Level 0
  - Processor allocation
  - Clock is used to prevent processes from monopolizing the system (presumably a long timer since we're not trying to multithread)
  - Above level 0 the number of processors actually shared is irrelevant (that is, above this level you've a “virtual processor” to some extent)

“THE” system hierarchy

- Level 1
  - “Segment controller” implemented by a sequential process
  - Does the “book keeping” of segments (hence satisfying the goal of an automatic backing store)
  - At higher levels, identification of information is in terms of segments (virtual addresses of sorts)

“THE” system hierarchy

- Level 2
  - “Message interpreter” (also a sequential process)
  - Allocation of the console keyboard
  - Processes identify themselves when printing lines
  - User must identify which process to address when inputing
  - Above the segment controller since messages may be lengthy and require more core memory to buffer than is permissible
  - At higher levels it is as if processes have their own private console

“THE” system hierarchy

- Level 3
  - Sequential processes for buffering input and output streams
  - Above the message interpreter so it may output errors to the operator
- Level 4
  - User programs
- Level 5
  - Operator (not implemented)

“THE” testing

- Each level is tested before considering the next level (no specifics given)
- Being able to test a level at a time prevents an “explosion” of potential interrupt sequences needing testing

“THE” conclusion

- A long-winded paragraph expressing “levels seem like a nice idea, especially for testing.”
- “Industrial software makers react to the system with mixed feelings”
  - But as systems become more complex, the structuring becomes more important as to maintain correctness and manageability

“THE” appendix

- Outlines semaphores
  - Use of semaphores as mutexes
  - Reasoning about semaphores
  - “Proving the Harmonious Cooperation”

A curious point...

- No MMU
- How are “segments” (virtual addresses) achieved?
- Recall earlier: Not designed for user programs written in machine language
- Modified Algol compiler

“THE” acronym

The Structure of the "THE"-Multiprogramming System

Edsger W. Dijkstra
Technological University, Eindhoven, The Netherlands

A multiprogramming system is described in which all activities are divided over a number of sequential processes. These sequential processes are placed at various hierarchical levels, in each of which one or more independent abstractions have been implemented. The hierarchical structure proved to be vital for the verification of the logical soundness of the design and the correctness of its implementation.

KEY WORDS AND PHRASES: operating system, multiprogramming system, system hierarchy, system structure, real-time debugging, program verification, synchronizing primitives, cooperating sequential processes, system levels, input-output buffering, multiprogramming, processor sharing, multiprocessing
CR CATEGORIES: 4.30, 4.32

Accordingly, I shall try to go beyond just reporting what we have done and how, and I shall try to formulate as well what we have learned.

I should like to end the introduction with two short remarks on working conditions, which I make for the sake of completeness. I shall not stress these points any further.

One remark is that production speed is severely slowed down if one works with half-time people who have other obligations as well. This is at least a factor of four; probably it is worse. The people themselves lose time and energy in switching over; the group as a whole loses decision speed as discussions, when needed, have often to be postponed until all people concerned are available.

“THE” acronym

An oddly-placed hyphen

The Structure of the "THE"-Multiprogramming System

Edsger W. Dijkstra
Technological University, Eindhoven, The Netherlands

A multiprogramming system is described in which all activities are divided over a number of sequential processes. These sequential processes are placed at various hierarchical levels, in each of which one or more independent abstractions have been implemented. The hierarchical structure proved to be vital for the verification of the logical soundness of the design and the correctness of its implementation.

KEY WORDS AND PHRASES: operating system, multiprogramming system, system hierarchy, system structure, real-time debugging, program verification, synchronizing primitives, cooperating sequential processes, system levels, input-output buffering, multiprogramming, processor sharing, multiprocessing
CR CATEGORIES: 4.30, 4.32

Accordingly, I shall try to go beyond just reporting what we have done and how, and I shall try to formulate as well what we have learned.

I should like to end the introduction with two short remarks on working conditions, which I make for the sake of completeness. I shall not stress these points any further.

One remark is that production speed is severely slowed down if one works with half-time people who have other obligations as well. This is at least a factor of four; probably it is worse. The people themselves lose time and energy in switching over; the group as a whole loses decision speed as discussions, when needed, have often to be postponed until all people concerned are available.

“THE” acronym

Technological University, Eindhoven, The Netherlands

The Structure of the "THE"-Multiprogramming System

Edsger W. Dijkstra
Technological University, Eindhoven, The Netherlands

A multiprogramming system is described in which all activities are divided over a number of sequential processes. These sequential processes are placed at various hierarchical levels, in each of which one or more independent abstractions have been implemented. The hierarchical structure proved to be vital for the verification of the logical soundness of the design and the correctness of its implementation.

KEY WORDS AND PHRASES: operating system, multiprogramming system, system hierarchy, system structure, real-time debugging, program verification, synchronizing primitives, cooperating sequential processes, system levels, input-output buffering, multiprogramming, processor sharing, multiprocessing
CR CATEGORIES: 4.30, 4.32

Accordingly, I shall try to go beyond just reporting what we have done and how, and I shall try to formulate as well what we have learned.

I should like to end the introduction with two short remarks on working conditions, which I make for the sake of completeness. I shall not stress these points any further.

One remark is that production speed is severely slowed down if one works with half-time people who have other obligations as well. This is at least a factor of four; probably it is worse. The people themselves lose time and energy in switching over; the group as a whole loses decision speed as discussions, when needed, have often to be postponed until all people concerned are available.

“THE” acronym

(Technische Hogeschool Eindhoven)

Technological University, Eindhoven, The Netherlands

The Structure of the "THE"-Multiprogramming System

Edsger W. Dijkstra
Technological University, Eindhoven, The Netherlands

A multiprogramming system is described in which all activities are divided over a number of sequential processes. These sequential processes are placed at various hierarchical levels, in each of which one or more independent abstractions have been implemented. The hierarchical structure proved to be vital for the verification of the logical soundness of the design and the correctness of its implementation.

KEY WORDS AND PHRASES: operating system, multiprogramming system, system hierarchy, system structure, real-time debugging, program verification, synchronizing primitives, cooperating sequential processes, system levels, input-output buffering, multiprogramming, processor sharing, multiprocessing
CR CATEGORIES: 4.30, 4.32

Accordingly, I shall try to go beyond just reporting what we have done and how, and I shall try to formulate as well what we have learned.

I should like to end the introduction with two short remarks on working conditions, which I make for the sake of completeness. I shall not stress these points any further.

One remark is that production speed is severely slowed down if one works with half-time people who have other obligations as well. This is at least a factor of four; probably it is worse. The people themselves lose time and energy in switching over; the group as a whole loses decision speed as discussions, when needed, have often to be postponed until all people concerned are available.

- Additional information from: http://en.wikipedia.org/wiki/THE_(operating_system)
- Pictures from: http://www.science.uva.nl/faculteit/museum/X1.html (University van Amsterdam Computer Museum)