

# The Structure of the “THE” - Multiprogramming System

“THE” (Technische Hogeschool Eindhoven)

Edsger W. Dijkstra 1968

Presented by: Payal Agrawal

# Who is Dijkstra?

- Remember him from algorithms
- Dutch professor of Mathematics
- Designed and implemented one of the first modern operating systems for a multi-process computer using a layered scheme

History Revisited!!

# Outline

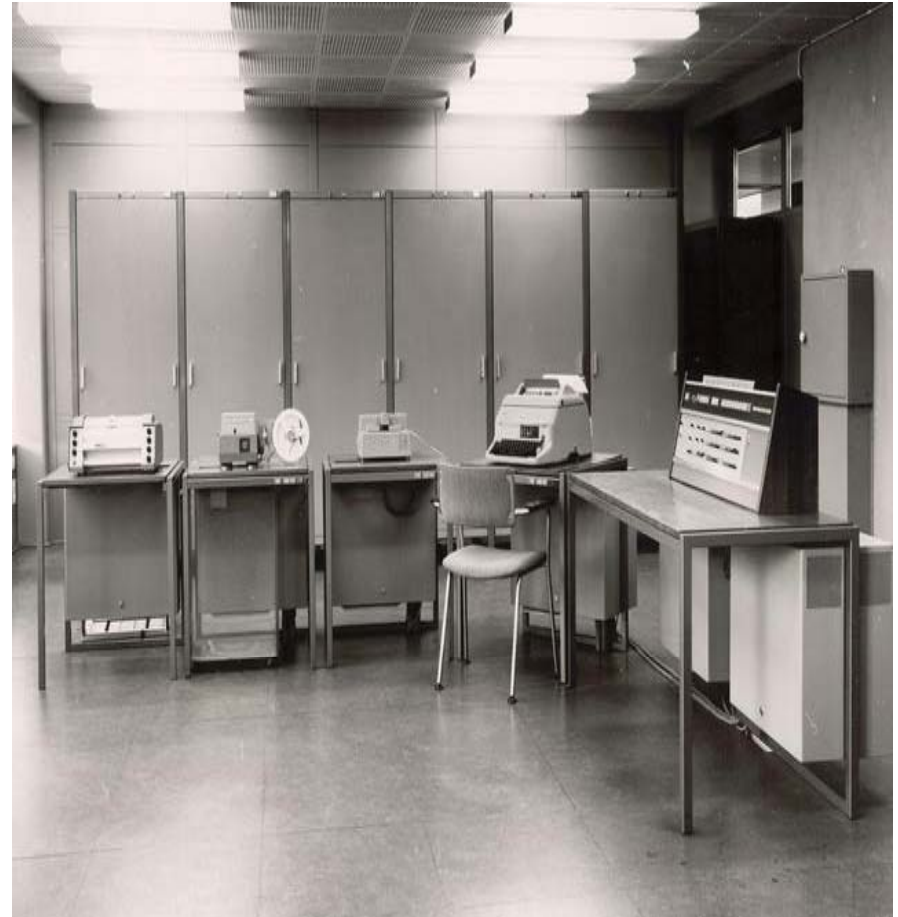
- Goal of project
- Platform (Tool)
- System Structure
- System Hierarchy (Level 0-5)
- Software Eng. Concepts.
- Contributions

# Goal of Project

- Process smoothly a **continuous flow of user programs**
- Specific objectives
  - Reduction of turn-around time for programs of short duration
  - Economic use of peripheral devices
  - Efficient use of Memory and CPU
  - The ability to run large flexible user programs that can communicate with each other
  - Testable
  - **Not designed as a multi-user operating system**

# Tool

- Dutch Electrológica **EL X8** computer
  - 32K core memory vs. 1-2 Gb
  - 512K words drum vs. 100 Gb
  - Memory cycle time 2.5  $\mu$ s vs. 5 – 70 ns
  - Address size 27bits vs. 64bits
  - Low capacity channels supporting peripherals
    - (3 paper tape readers and punches, 1 printer, 1 plotter, and 2 teleprinter)



[Source: electrologica.nl]

# Terms used differently

- Core memory ↔ Main Memory
- Drum ↔ Secondary storage – Disk
- Segments ↔ Virtual pages
- Whole system is “Society of Sequential processes” not sequential execution

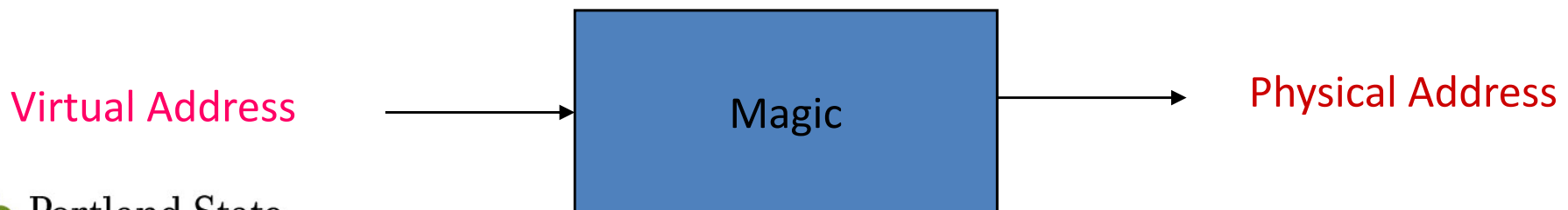
# System Structure

- Storage allocation
- Processor allocation
- System Hierarchy

# Storage Allocation

- Von Neumann machine (stored-program computer): Information in drum address.
- Distinction between memory units “pages” and information units “segment”
- Pages
  - Core Pages (page frames)
  - Drum Pages (disk blocks)
- Segment (virtual pages) have “virtual address” that are mapped to the page “physical address”
- Virtual space >> physical space
- A page returned from core to drum can reside anywhere in the drum
- A program should not be saved in consecutive drum pages

## Virtual Memory !!



# Process Allocation

- Society of sequential processes
  - Process per program
  - Process per peripheral
  - Processes for segment controller and message interpreter
- Delaying of process execution will have no harmful effects to the internal logic of the process
- Numerous processes executing independently of number of actual processors, as long as processors can switch to the various processes
- Mutual synchronization of parallel sequential processes is implemented via "**semaphores.**"

**virtualization of the CPU!!**

# Semaphores

- Thread not invented by that time.
- Integer variable initialized to 0 or 1
- Two atomic operations, P: "passeren" (to let pass) and V: "vrygeven" (to give up)
- P(Semaphore S) (Wait)
  - $S.val = S.val - 1;$
  - while  $S.val < 0$  add calling process to S.list and block ;
  - `/* sleep! */`
- V(Semaphore S) (Signal)
  - $S.val = S.val + 1$
  - If  $S.val \leq 0$
  - { remove a process Q from S.list;
  - wakeup (Q);}

# Semaphores For Mutual Exclusion

*mutex = 1;*

*P(mutex);*

*critical section*

*V(mutex);*

- Mutual Exclusion: One process at a time in critical section
- P and V: “indivisible action” (atomic actions) on “mutex”
- Mutex is globally accessible
- Protection of critical sections

# Private Semaphores (Conditional Synchronization)

*mutex = 1;*

*privateSemaphore = 0;*

*P(mutex);*

*modify global state variables, inspect condition*

*V(privateSemaphore);*

*V(mutex);*

*P(privateSemaphore);*

- Private Semaphores used for Condition synchronization:
  - **wait** until condition holds before proceeding
  - **signal** when condition holds so others may proceed
- Each sequential process has associated with it a number of private semaphores, initialized to 0.
  - Max value equals 1,
  - Min value equals -1.
- Private semaphore can be released globally but only locked privately

# Proving Harmonious Co-operation

Homing Position: The neutral point in which all cyclic processes are when the system is at rest.

- Process waiting for V operation on private semaphore (signal) is in homing position
- Process leaves homing position, then performs task
- Process returns to homing position
- All processes will eventually be in homing position (no deadlock)

# System Hierarchy

- Layered Operating system.
- Lower layers provide abstraction of resources
- Higher layers access lower levels for resources
- Access always proceeds from top down
- Each layer can be tested and verified independently
- Six Levels (layers): Level 0 – 5

# Level 0 – CPU Virtualization

- Contains:
  - Processor Allocation
  - Real-time Clock Interrupt
  - Priority scheduling (CPU Scheduler)
  - Semaphores: to implement synchronization
- Provides: Processor Virtualization
- Abstraction: Each process appears to have own processor

<b>Processor Allocation</b>
EL X8

# Level 1 – Memory Virtualization

- Contains:

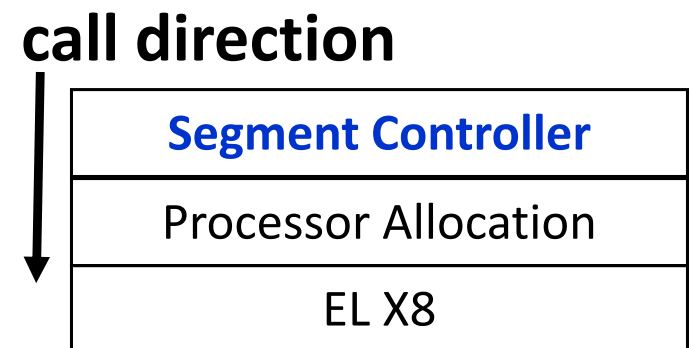
  - Segment Controller

  - Unique identifiers for each segment of memory

  - At all higher levels identification of information takes place in terms of segments

- Provides: Memory Virtualization

- Abstraction: Each process has a segment of memory to use



# Level 2 – Console Virtualization

- Contains:

  - Message Interpreter

  - All processes share one physical console using mutual synchronization

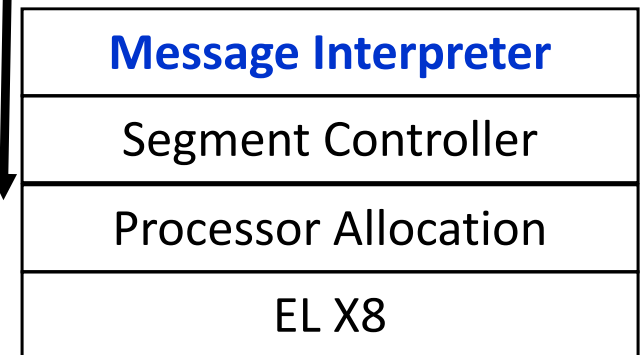
  - Messages passed from operators to processes

  - Handles user input and processes output (Virtual console)

- Provides: Console Virtualization

- Abstraction: Each process has own individual console

call direction

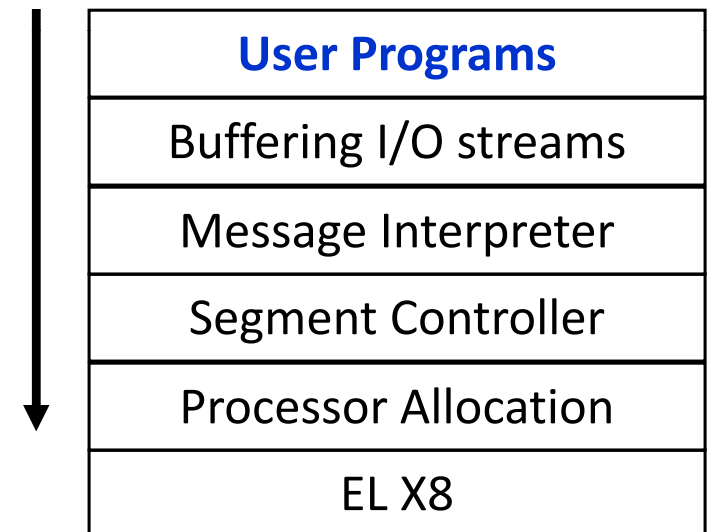


# Level 3 – 5

- Level 3 – I/O Virtualization
  - Contains the Sequential Processes associated with buffering of input streams and unbuffering of output streams.
  - Manages all I/O between the devices attached to the computer
  - Abstraction: Devices wrapped by buffers and abstracted to logical communication units.

**call direction**

- Level 4 – User Programs
  - Consists of the independent – user programs
- Level 5 – The actual user!
  - *“not implemented by us”*



# Experience

- Conception
- Construction
- Verification
- ✓ Proof of correctness of the system

# Conception, Construction

- **Conception**
  - All the concepts are born
  - It took a long time!
  - Learnt that the society of mutually synchronized processes in time behavior can satisfy all the requirements
- **Construction**
  - Done in rather traditional manner
  - Change of specifications has been rare

# Verification (Testing)

- Done in stages
- Bottom layer thoroughly tested before implementing layer above
- Prevent bugs instead of debugging
- Allows full system testing in manageable time
- $5 + 5 + 5 + 5 + 5 + 5$  vs.  $5 * 5 * 5 * 5 * 5 * 5$
- “... testing had not yet been completed, but the resulting system is guaranteed to be flawless.”

# Software Engineering Observations

- Production speed severely slows down when working with part–time people
- People lose time and energy in switching over and the group loses decision speed.
  - Similar to context switching in OS
- This type of work is very difficult, and that every effort to do it with other than the best people is doomed to either failure or moderate success at enormous expense.

# Mistakes made

- Paying too much attention to eliminating what was not the real bottle neck
- Trying for “Perfect Installation”
- Not much thought given during design phase
- Late Debugging: lesson learnt that prevention is better then cure

# Contributions

- Layered Structure of OS
- Concurrent Programming (semaphores)
- Memory segments (virtual addresses): The “THE” system introduced the first forms of software-based memory segmentation freeing programmers from being forced to use actual physical locations on the drum memory
- Proof of correctness of the system: A refined multiprogramming system whose implementation removes exhaustive testing

# References

- Reused material by

Jin Li CS 533, Spring 2006

Jimmy Pierce CS 533, Spring 2005

Navya Jammula CS 533, Winter 2008

- Additional information from:

[http://en.wikipedia.org/wiki/THE \(operating system\)](http://en.wikipedia.org/wiki/THE_(operating_system))

*qstream.org/~krasic/cs508-*

*2006/summaries/paper12/THE.ppt*

**THANK YOU!!**