Programming Semantics for Multiprogrammed Computations

- Paper by
  - Jack B. Dennis and Earl C. Van Horn

- Presented By
  - Sharon Rajan

Overview

- Terminology
- Concepts
- Parallel Programming Semantics
- Protection
- Exceptional Conditions
- Protected Entry Points
- Names and Directories
- Conclusion
Introduction

- MCS - Multiprogrammed Computer System

MCS Properties

- Support for Multiple Users - Computation process are concurrent operation for more than one user.
- Sharing of Resources
- Demand for computing resources - vary by computation
- Reference to Common Information (common subroutines)
- Should meet the changing requirements.

Terminology

- Word
  - Smallest unit of stored information
- Segment
  - Ordered set of words
  - References are done by a word name, w=[i,a]
- C-list
  - List of capabilities (Access List)
- Computations
  - Set of processes that are working harmoniously on the same problem.
Concepts

- Protection of Segments
  - Done by the use of segment capabilities using access indicators
- Access Indicators
  - X - executable as procedure
  - R - readable as data, non executable
  - XR - executable as procedure & readable as data
  - RW - readable and write able as data
  - XRW - exec as procedure, read and write as data

Concepts ...

- Ownership Indicators
  - O - owned
  - N - not owned
- Sphere of Protection
  - Defined by the list of capabilities
  - Capabilities would be frequently changed (add/delete) during execution of computation defining its sphere of protection.
- Principal
  - User of the MCS
Sphere of Protection

Fig. 1. A computation

Concepts ...

- Supervisor
  - HW and SW elements that implement basic computer system functions
  - all computations are performed around this

- Process List
  - Data structure within the Supervisor having entries of every process
Parallel Programming Semantics

- fork $w$
  - initializes a new process at instruction $w$
- quit
  - Remove a process. The state word is discarded
- join $t$, $w$
  - Conway’s join instruction
  - If counter at word $t$ becomes zero, word at $w$ is executed

Parallel Programming Example

```plaintext
    Boolean $w$;  real $S$;  integer $t$;
    private integer $i$;
    $i := n$;
    for $i := 1$ step 1 until $n$ do
        fork $e$;
        quit;
    e:  begin  private real $X$;
        substance:  $X := A[i] \times B[i]$;
        lock $w$;
        $S := S + X$;
        unlock $w$;
        join $t$, $r$;
        quit;
    end;
    r:  end;
```
Parallel Programming Semantics...

- private x
  - x exists only for the executing process
- lock w
  - The lock bit is set to 1
- unlock w
  - The lock bit is set to 0
- The lock and unlock are used to denote a critical section

---

Input/Output

- Classes of Communication
  - Programmed I/O
    - This process waits until the data is transferred from I/O to memory and vice versa
    - Suited for small amount of data
  - DMA
    - The process is suspended until all the operations between the memory and the I/O device is completed
    - Suited for large data transfer
- Semantics
  - execute i/o function i
Parallelism

- Motivation
  - Freedom to allocate system resources with great efficiency
  - Express the frequently occurring operations of computations from the same code.

Protection

- Inferior Sphere of Protection
  - Process operating within another sphere of protection

- Semantic
  - $i := \text{create sphere } w$
    - Create an inferior sphere
  - $i := \text{grant } X j,k$
    - Grant permission to execute to $j$, $k$
  - $\text{start } i,w$
    - Start process in the inferior sphere
Creating an Inferior sphere

Exceptional Conditions

- Problems occurred in the inferior sphere
  - Fault
    - Hardware Malfunction or Memory parity error
  - Resource excess
    - allocation of resources exceeding the allotment
  - Addressing snag
    - Process generated a valid address, data not in memory.
  - Sphere Violation
    - Process refers to a capability not present in the C-list.
Exceptional Conditions...

- Other exceptional Conditions- Inferior Sphere
  - Halt Instruction
    - Terminate this process and notify superior
  - Breakpoint Instruction
    - same as halt instruction
  - Undefined Instruction
    - execute an undefined operational code
  - Arithmetic contingencies
    - called for action by superior procedure

How is an exception handled?

- A new process is created in the superior.
- Suspended process capability created to have access to the state word of inferior.
- Meta-Instructions used
  - fetch status i,w
  - set status i,w
  - continue I
  - stop k
  - examine i,j,w
  - ungrant i,j
Protected Entry Points

- Procedures need to be shared.
  - Protection of local objects.
- Protected Entry Points
  - Modification or change in C-list must accompany transfer of control to a procedure.
  - Restrictive use of a procedure.
  - Process calling a restrictive procedure requires an additional entry in the C-list, called as the “entry” capability.
  - Created by the owner

Meta-Instructions Used

- \( h = \text{Create entry } w,n \)
- \( \text{enter } j,r,k \)

[Diagram of C-List and Sphere of Caller and Procedure Capability]
Entry and Exit from a procedure

Naming

- Identify the objects
- Sharing of Retained Objects
  - Sharing procedures and data segments
- Desiderata for Names
  - Names of objects should be unambiguous
  - Name of an object can never be changed
    - As this impacts all the objects that embedded this name
    - Retained objects are directly referred.
Directory

- Directory
  - Set of items, each being an association between a name component and a capability
  - One root directory for every principal.
  - Directory capability for the root directory.

Directory Meta-Instructions

- J=acquire{accessindicator}l,<namecmpt>
  - Search directory for namecmpt and enters the capability to the C-List

- release l
  - remove the object capability from the C-List

- delete l
  - Delete the object
**Directory Meta-Instructions**

- **Remove I,<namecmpnt>**
  - Remove the objectname from directory.
- **I=create segment{access list}**
  - Creates a new segment and adds the capability to the C-List of the creating computation.
- **I=create directory**
- **place{P,F}I,<cmpntname>,j**
  - Associate name with the object.
- **I=link <principal name>**
  - Capability to access the principal's directory

---

**Linking directories**

[Diagram showing linking of directories]
Conclusion

- Semantics defined with meta-instructions for parallel programming in multi programming systems.
- Semantics relate to parallel processing, protection, sharing, debugging

References

- Programming Semantics for Multiprogrammed Computations
  - Jack B Dennis & Earl C. Van Horn - March 1966
- Fano, R.M. The MAC system: The computer utility approach
  - IEEE spectrum2 - Jan 1965
Thank You