



On the Duality of Operating System Structures



Nicholas Lovett
Univ. of Central Florida
COP6614



Overview



- OS design can be placed into one of two very rough categories...
- Message-oriented system.
- Procedure-oriented system.



Derived observations

- The two models are duals of each other.
- The dual programs are logically identical.
- The performance for each is identical.

10/11/2004

Nicholas Lovett, COP 6614

3



Goals

- The considerations for choosing which model to adapt are not found in the applications the system is meant to support

BUT....

Upon the nature of the machine architecture.

10/11/2004

Nicholas Lovett, COP 6614

4



Message-oriented system

- Characterized by:
 - Small static number of processes.
 - Explicit message system for communication.

10/11/2004

Nicholas Lovett, COP 6614

5



Hallmarks for message-oriented

- Specific communication paths.
- Number of processes remain relatively static.
- Operate in relatively static context.

10/11/2004

Nicholas Lovett, COP 6614

6



Good design practices

- Synchronization implemented in message queues.
- Data structures are passed by reference.
- Processes operate upon very small number of messages at a time.
- Priorities are statically assigned.

10/11/2004

Nicholas Lovett, COP 6614

7



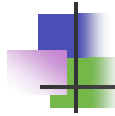
Model operations

- SendMessage[messageChannel, messageBody] returns [messageId]
- AwaitReply[messageId] returns [messageBody]
- WaitForMessages[set of messagePort] returns [messageBody, messageId, messagePort]
- SendReply[messageId, messageBody]

10/11/2004

Nicholas Lovett, COP 6614

8



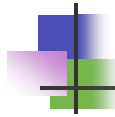
Message-Oriented resource manager

```
begin m: messageBody;
  i: messageId;
  p: portId;
  s: set of portId;
  ... --local data and state information for this process
  initialize;
  do forever;
    [m, i, p] ← WaitForMessage[s];
    case p of
      port1 => ...; --algorithm for port1
      port2 => ...
        if resourceExhausted then
          s ← s - port2;
          SendReply[i, reply];
          ...; --algorithm for port2
        ...
      portk => ...
        s ← s + port2
        ...; --algorithm for portk
    endcase;
  endloop;
end.
```

10/11/2004

Nicholas Lovett, COP 6614

9



Procedure-oriented system

- Characterized by:
 - Large, rapidly changing number of processes.
 - Process synchronization mechanisms based on shared variables. (locks, semaphores, monitors)

10/11/2004

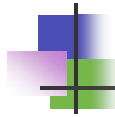
Nicholas Lovett, COP 6614

10



Hallmarks for procedure-oriented

- Global data can be protected and efficiently accessed.
- Process creation/deletion is very easy.
- Process typically has one goal, but may wander system to complete it.





Model operations

- processId <- FORK procedureName[parameterList]
- [resultList] <- JOIN processId
- Condition variables.
- WAIT conditionVariable
- SIGNAL conditionVariable



Procedure-oriented resource manager

```

ResourceManager: MONITOR =
c: CONDITION;
resourceExhausted: BOOLEAN;
...--global data and state information for this process
proc 1: ENTRY PROCEDURE[ ... ] =
...; --algorithm for proc1
proc 2: ENTRY PROCEDURE[ ... ] RETURNS[ ... ] =
BEGIN
IF resourceExhausted THEN WAIT c;
...
RETURN[results];
...;
END; --algorithm for proc2
...

proc L: ENTRY PROCEDURE[ ... ] =
BEGIN
...;
resourceExhausted ← FALSE;
SIGNAL c;
...;
END; --algorithm for proc L
endloop;
initialize;
END.

```



Duality mapping

Message-oriented system

Procedure-oriented system

Processes, CreateProcess

Monitors, NEW/START

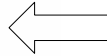
Message Channels

External Procedure identifiers

Message Ports

ENTRY procedure identifiers

SendMessage; AwaitReply
(immediate)

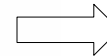


simple procedure call

SendMessage; ... AwaitReply
(delayed)

FORK; ... JOIN

SendReply



RETURN (from procedure)

main loop of standard resource manager, WaitForMessage statement, case statement

monitor lock, ENTRY attribute

arms of the case statement

ENTRY procedure declarations

selective waiting for messages

condition variables, WAIT, SIGNAL

10/11/2004

Nicholas Lovett, COP 6614

15



Message-Oriented resource manager

```

begin m: messageBody;
  i: messageId;
  p: portId;
  s: set of portId;
  ... --local data and state information for this process
initialize;
do forever;
  [m, i, p] ← WaitForMessage[s];
  case p of
    port1 => ...; --algorithm for port1
    port2 => ...
      if resourceExhausted then
        s ← s - port2;
        SendReply[i, reply];
        ...; --algorithm for port2
    ...
    portk => ...
      s ← s + port2
      ...; --algorithm for portk
  endcase;
endloop;
end.

```

Mutual exclusion
lock; one request at
a time

Different procedure
names within
monitor

10/11/2004

Nicholas Lovett, COP 6614

16



Performance Preservation

- Three main components
 - Execution time of programs themselves.
 - Computation overhead of primitive system operations.
 - Queuing/waiting times, and scheduling decisions.

10/11/2004

Nicholas Lovett, COP 6614

17



Real world rearrangement

- Obviously, it is not easy to change to structure of most OS. Expense not justified.
- Cambridge CAP Computer

10/11/2004

Nicholas Lovett, COP 6614

18



Notes to keep in mind

- The models described bear the same relationship to reality as a frictionless surface does to physics.
- Most OS can be usefully classified using these models.

10/11/2004

Nicholas Lovett, COP 6614

19



Conclusions

- One system not “better” than the other.
- Merit in both styles with respect to structure, performance, and “correctness.”
- Pick a system which will yield the set of primitives which are easier to build or better suited to constraints by hardware.

10/11/2004

Nicholas Lovett, COP 6614

20



References

- Lauer, H.C., Needham, R.M., “On the Duality of Operating Systems Structures,” in Proc. Second International Symposium on Operating Systems, IR1A, Oct. 1978, reprinted in Operating Systems Review, 13,2 April 1979, pp. 3-19.
- [On the duality of operating system structures](#)