JAVA BYTE IPC: PART 6-SUMMARY

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)
### Summary: I/O vs IPC

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O is a producer-consumer problem involving I/O devices and local threads</td>
<td></td>
</tr>
<tr>
<td></td>
<td>I/O-based IPC a producer-consumer problem involving threads in different processes</td>
</tr>
<tr>
<td></td>
<td>A process should be able to create independent buffers for different with the processes with which it communicates</td>
</tr>
<tr>
<td></td>
<td>A process should be able to control whether it communicates with a process started by a different user or on a different OS</td>
</tr>
<tr>
<td></td>
<td>Should keep the awareness a process has about other processes low</td>
</tr>
<tr>
<td></td>
<td>A client knows about a server but not vice versa</td>
</tr>
<tr>
<td></td>
<td>Each service provided by a server represented by a port number known to its clients, which types the connection the clients make</td>
</tr>
</tbody>
</table>
To establish connection of type T, with a server on host H offering a, a client creates a regular port object and then invoke the connect call with H and port number for T, which sends a message to a server listener-port.

To accept a connections of type T from clients, the server creates a listener-port object associated with the port number for T.

To accept a the connect message from a specific client regular port, the server invokes an accept call on the listener-port, which returns a new regular port connected to the client regular port.
SUMMARY: READS AND WRITES

Send data may be scattered or gathered so the reverse has to occur at the other end.
Summary: Blocking vs Non Blocking

Accept, connect, read, write operations can block or not.

Blocking calls result in thread proliferation.

Non blocking calls can return status values indicating if they executed.

To prevent polling it should be possible to wait for multiple channel operations to be executable and to know which ones are executable when unblocked and to change this set dynamically.
Provides a single service-independent thread for waiting and setting waiting parameters

Services queues of command objects specifying application-specific functionality.

Initiate methods of command objects called when they are dequeued and execute methods when associated operations executable.

Predefined command objects make buffering, scatter/gather and threading decisions about inform listeners.

Predefined command objects designed to reduce copying and thread switches.
Command Objects (From Part That Did not Get Recorded)

How to create a single blocking, selecting, thread and make sure only it changes interest operations and checks ready operations to reduce concurrency conflicts and increase code resuability, while allowing application-specific interest in operations and application-specific post processing of operations?

Generic selecting thread processes application-provides operation-specific command objects enqueued by other application-specific threads executes initiate() methods of dequeued command objects and execute() methods of command objects associated with ready operations.
**LISTENERS (FROM PART THAT DID NOT GET RECORDED)**

A command object associated with an operation must do certain tasks – execute the operation when it is ready, register interest in that operation – how to prevent application programmers from performing these tasks while allowing application-specific post operation processing?

Provide predefined command objects (selected by factories) that allow application-specific customization through constructor parameters and post-operation listeners.