Instructor: Prasun Dewan (FB 150, dewan@unc.edu)
NIO Discussion from First Principles (End of Previous Part)

- I/O? ✓
- I/O → Inter Process Communication (IPC)?
- Blocking → Non-Blocking?
IPC AS I/O?

Console
- put(char )
- char get()

Process
- Process
- Process
- Process
Do all processes, \( p_1 \ldots p_n \) communicating with a process, \( P \), share a common bounded buffer?

Does not allow private communication of credit card and other information.
PRIVATE CHANNELS

Channel creation and connection?
Each process has a single well defined standard input and output.

Multiple, dynamic read/write streams in non distributed application?
NIO Discussion from First Principles (Review)

I/O?

I/O as Inter Process Communication (IPC)?

Non-Blocking?
**TTY Output Driver Cannot Block (review)**

After outputting the last queued character, the interrupt routine will find output buffer empty.

What triggers its call when a new character is entered, as it cannot wait for non empty buffer?

On finding empty output buffer, interrupt routine disables interrupt posting.

Upper half enables interrupt posting when it writes (to an empty buffer).

Enabling disabled interrupt posting causes an output “interrupt” to be posted and interrupts routine to be called.

After outputting the last queued character, the interrupt routine will find output buffer empty.

What triggers its call when a new character is entered, as it cannot wait for non empty buffer?

On finding empty output buffer, interrupt routine disables interrupt posting.

Upper half enables interrupt posting when it writes (to an empty buffer).

Enabling disabled interrupt posting causes an output “interrupt” to be posted and interrupts routine to be called.
TTY Input driver Cannot Block (Review)

- `get()`: consumer
- `put()`: producer
- `char^1` and `char^2`: Input Buffer
- Tty input routine cannot block if input buffer full
- Echoes a bell character to user to not type any more
NIO Discussion from First Principles (Review)

I/O?

I/O as Inter Process Communication (IPC)?

Non-Blocking?
IPC as I/O? (Review)

- Console
  - put(char)
  - char get()

- Process
- Process
- Process
IPC as I/O? (Review)

Do all processes, $p^1..p^n$ communicating with a process, $P$, share a common bounded buffer?

Does not allow private communication of credit card and other information.
PRIVATE CHANNELS (REVIEW)

Channel/Port

put(char)
char get()

Channel/Port
put(char)
char get()

Channel creation and connection?

Process

Process

Channel/Port
put(char)
char get()

Channel/Port
put(char)
char get()
I/O Console Creation and Naming? (Review)

Each process has a single well defined standard input and output

Multiple, dynamic read/write streams in non distributed application?
A particular process can talk to multiple File Stream objects

Need appropriate OS access rights to open file
How to control access when processes do not belong to one operating system?

Both processes have to agree to connect their channel objects.
Does each process have to know about the other?

Does not allow server to communicate with unknown clients.
Client/Server Distinction

One process (client) knows about other process (server)
**Process ID Awareness?**

Does a client process name use the server process id?

Process is local to OS and changes each time the server starts.
PORT ID BINDING AND AWARENESS

Client knows host, port number of server channel

A la room number for lectures
PORT NUMBER PER CLIENT?

Does the server have a separate port number for each client?

Cannot support an arbitrary number of clients and implies client awareness.
**Single Port for Multiple Clients?**

How to create separate individual channel from a single port number?
How to create separate individual channel from a single port number?
OPENING THE SAME FILE MULTIPLE TIMES

Separate streams object created on each open from a File factory object

But they write to the same file
Server Factory, Listening, Unbound End-Point Channel/Port

- Server Factory Selector
- static ServerChannelFactory open(portNum)
- Second client?
- Channel accept()
- put(char )
- char get()
- connect (host, port)
- Channel/Port

Client

Server

Diagram showing the interaction between a client and a server using a server channel factory for listening on an unbound endpoint.
**LOOPING ACCEPTS**

Server accepts in a loop, with each accept returning a new connected channel.
PORT-PROCESS MAPPING?

Single port per server?

Example of server with multiple ports?
Why multiple ports?
Grader per Class/Section/Group

- Graders provide modularity and load balancing
- Dispatching Server has separate ports (with different visibility) for clients and graders
- Grader is client or server?
- (Master) Dispatching server is fixed point, so (slave) graders are clients but provide the service!
**Blocking Semantics**

- **connect (host, port)**
  - Block until server accepts connection to server socket

- **Channel accept()**
  - Block until next client tries to contact the server socket

- **put(char)**
  - Block until char in system (bounded) buffer

- **char get()**
  - Block until char received

- **Communicate multiple characters?**
Library Support for Arrays

Library.write(chanel, char[])

channel.put(char)

Full Write?

Can build libraries for arrays

Kernel call for each put!

Switch to kernel mode (Context switch)

Switch to another thread (Process switch)
**System-Supported Byte-Array Communication**

- `connect (host, port)`
  - Block until server accepts connection to server socket
- `Channel accept()`
  - Block until next client tries to contact the server socket
- `write(byte[])`
  - How much to block for?
  - How does application know how much was written and read?
- `read(byte[])`
  - Block until at least one byte received
- Increases concurrency
**Java ByteBuffer** Bytebuffer with Pointers

- **mark**: Position of next element to be read or written
- **position**: Storage for contents
- **limit**: Size of (available) contents
- **capacity**: Position of next element to be read or written

Read may not yield expected bytes, write may not empty all bytes

May use the same buffer for multiple serial operations or batch operations, need to mark position of first unconsumed byte

*mark <= position <= limit <= capacity*
**System-Supported Byte-Array Communication (Review)**

- `connect (host, port)`
  - Block until server accepts connection to server socket
- `Channel accept()`
  - Block until next client tries to contact the server socket
- `write(byte[])`
  - Block until at least one byte in system buffer
- `read(byte[])`
  - Block until at least one byte received
- Increases concurrency
- How much to block for?
Sent/Received ByteBuffer Relation

- **connect** (host, port)
  - Block until server accepts connection to server socket

- **Channel accept()**
  - Block until next client tries to contact the server socket

- **write(ByteBuffer)**
  - Block until at least one byte in system buffer

- **ByteBuffer written equals corresponding ByteBuffer read?**

- **read(ByteBuffer)**
  - Block until at least one byte received

- **Messages Scattered or Gathered**
Application must gather if necessary.
**MESSAGE GATHERING (BULK TRANSFER)**

Application must scatter if necessary.
**BYTE ARRAY INFORMATION**

- How does application know how much was written and read?
- Return values
- How to reuse read or even write buffer
- Byte array with pointers
**Java ByteBuffer** ByteBuffer with Pointers

- **mark**: Position of next element to be read or written
- **position**: Position of next element to be read or written
- **limit**: Size of (available) contents
- **capacity**: Storage for contents

Read may not yield expected bytes, write may not empty all bytes

May use the same buffer for multiple serial operations or batch operations, need to mark position of first unconsumed byte

**Mark <= Position <= Limit <= Capacity**
Channel Objects

Server accepts in a loop, with each accept returning a new connected channel.
**Java API: ServerSocket, Socket Channels, Factories**

- **Selectable Channel**
  - ServerSocket socket()
  - SocketChannel accept()
  - boolean connect()

- **Socket Channel**
  - Socket socket()
  - int read(ByteBuffer)
  - int write(ByteBuffer)

- **HAS-A**
  - ServerSocket
    - Socket accept()
  - ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();

- **HAS-A**
  - Socket
    - InputStream getInputStream()
    - OutputStream getOutputStream()
  - SocketChannel socketChannel = SocketChannel.open();
NIO Discussion from First Principles

I/O?

I/O → Inter Process Communication (IPC)?

Blocking → Non-Blocking?