Java Byte IPC: Part 1 – Deriving NIO from I/O

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)

Rox Tutorial

The Rox Java NIO Tutorial

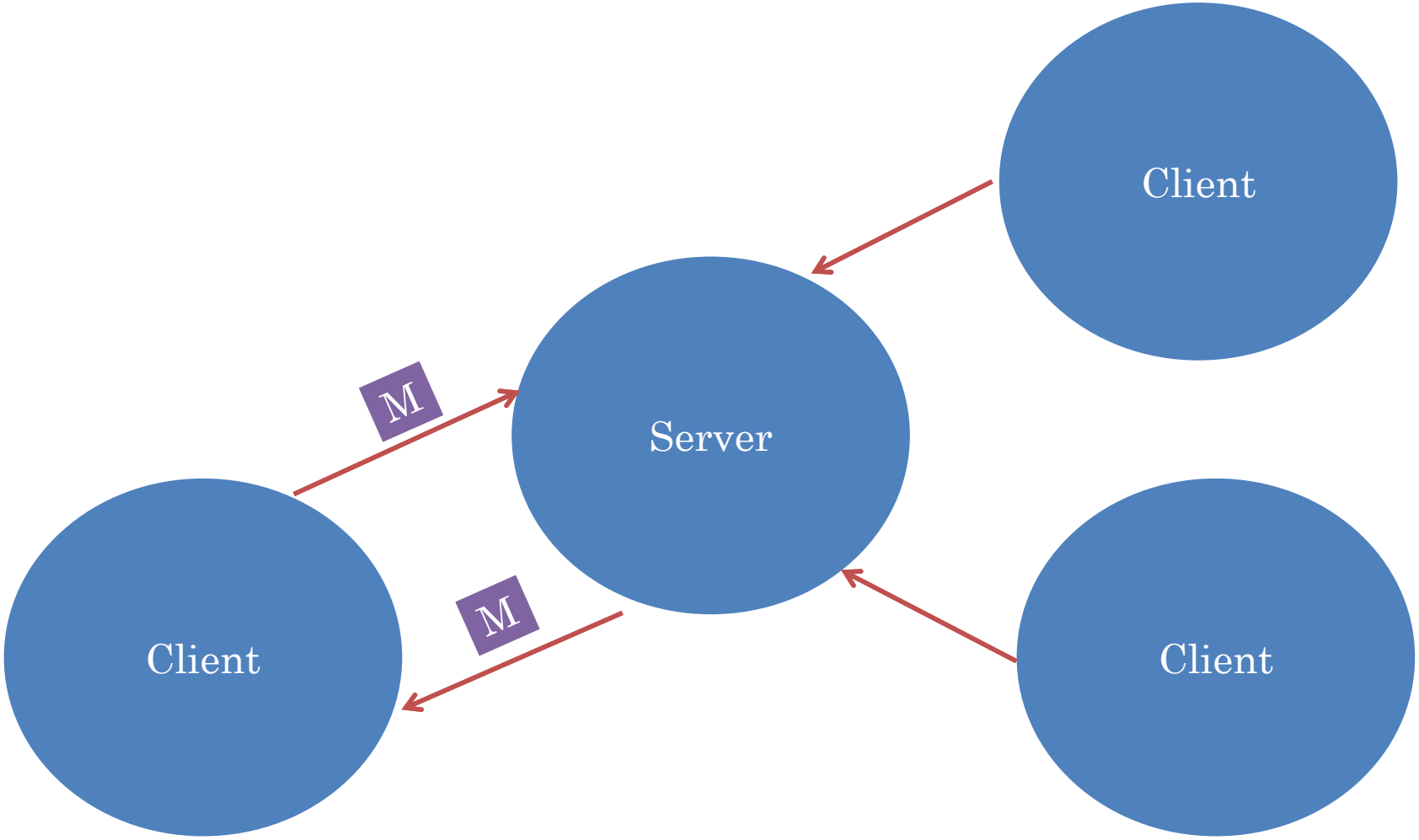
Contents

Introduction

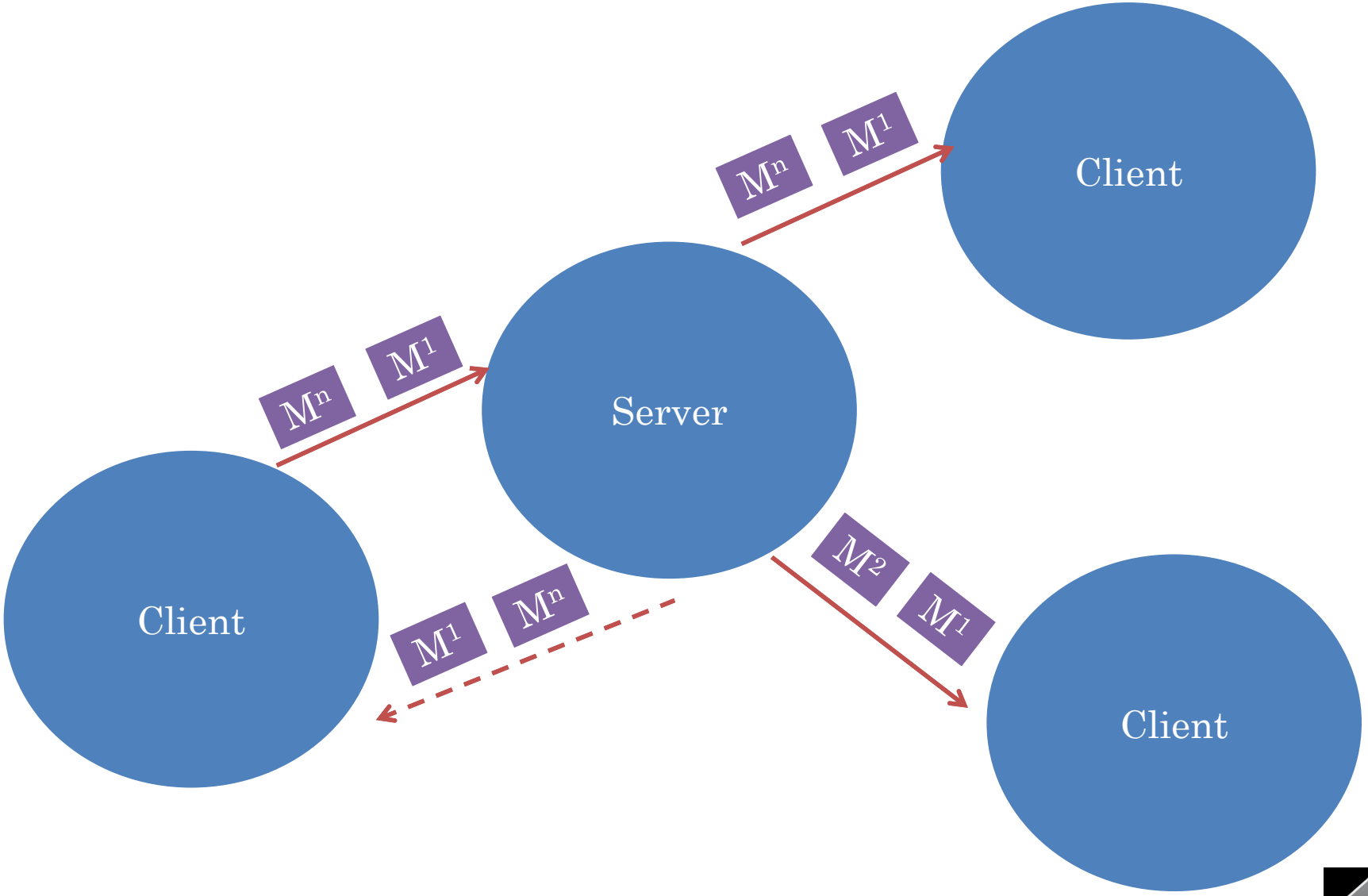
This tutorial is intended to collect together my own experiences using the Java NIO libraries and the dozens of hints, tips, suggestions and caveats that litter the Internet. When I wrote Rox all of the useful information existed as just that: hints, tips, suggestions and caveats on a handful of forums. This tutorial actually only covers using NIO for asynchronous networking (non-blocking sockets), and not the NIO libraries in all their glory. When I use the term NIO in this tutorial I'm taking liberties and only talking about the non-blocking IO part of the API.

If you've spent any time looking at the JavaDoc documentation for the NIO libraries then you know they're not the most transparent docs floating around. And if you've spent any time trying to write code based on the NIO libraries and you use more

Rox Echo Application



Assignment Requirements



Rox Tutorial: Pros and Cons

The Rox Java NIO Tutorial

Contents

Introduction

This tutorial is intended to collect together my own experiences using the Java NI
suggestions and caveats that litter the Internet. When I wrote Rox all of the useful tips,
suggestions and caveats on a handful of forums. This tutorial actually only covers g (non-
blocking sockets), and not the NIO libraries in all their glory. When I use the term nd only
talking about the non-blocking IO part of the API.

If you've spent any time looking at the JavaDoc documentation for the NIO librarie
transparent docs floating around. And if you've spent any time trying to write code e more

Explains NIO and its limitations

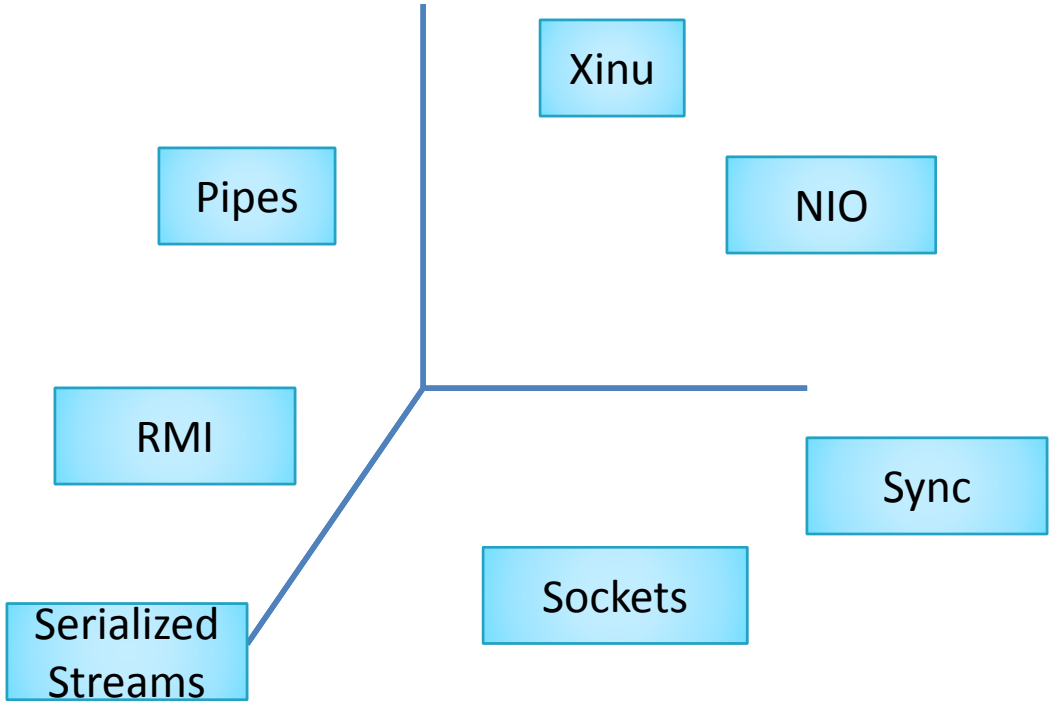
Has an implicit application architecture (design pattern)

Some students have trouble understanding what is to be done

Design pattern was not explicit

Abstraction vs. Implicit Design Pattern in Web Material

Points vs. Design space



Focus on point

Later design space

NIO Discussion from First Principles

I/O?

I/O → Inter Process Communication (IPC)?

Blocking → Non- Blocking?

I/O (API)

Console

put(char)

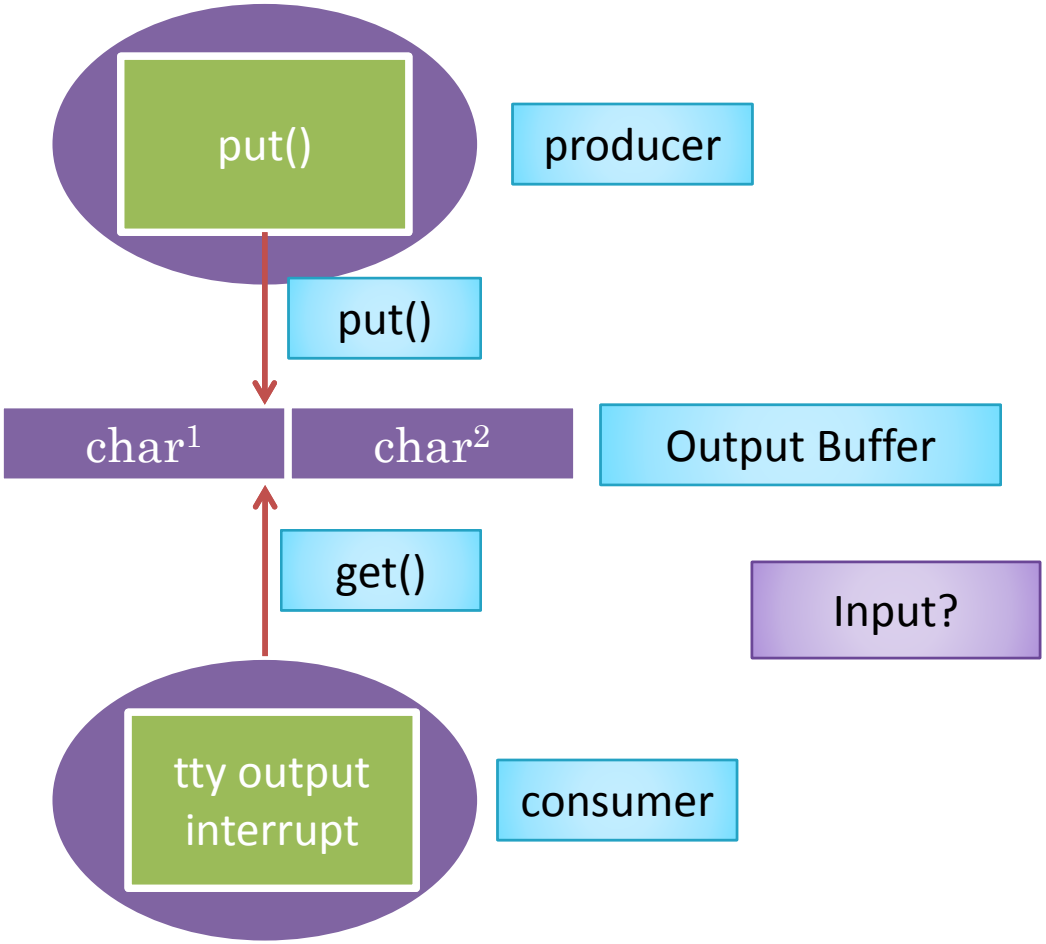
char get()

Implementation and Blocking Semantics?

Device driver?

Application and device events concurrently and at different rates

TTY Output Driver: Producer, consumer



Upper half executes in application thread

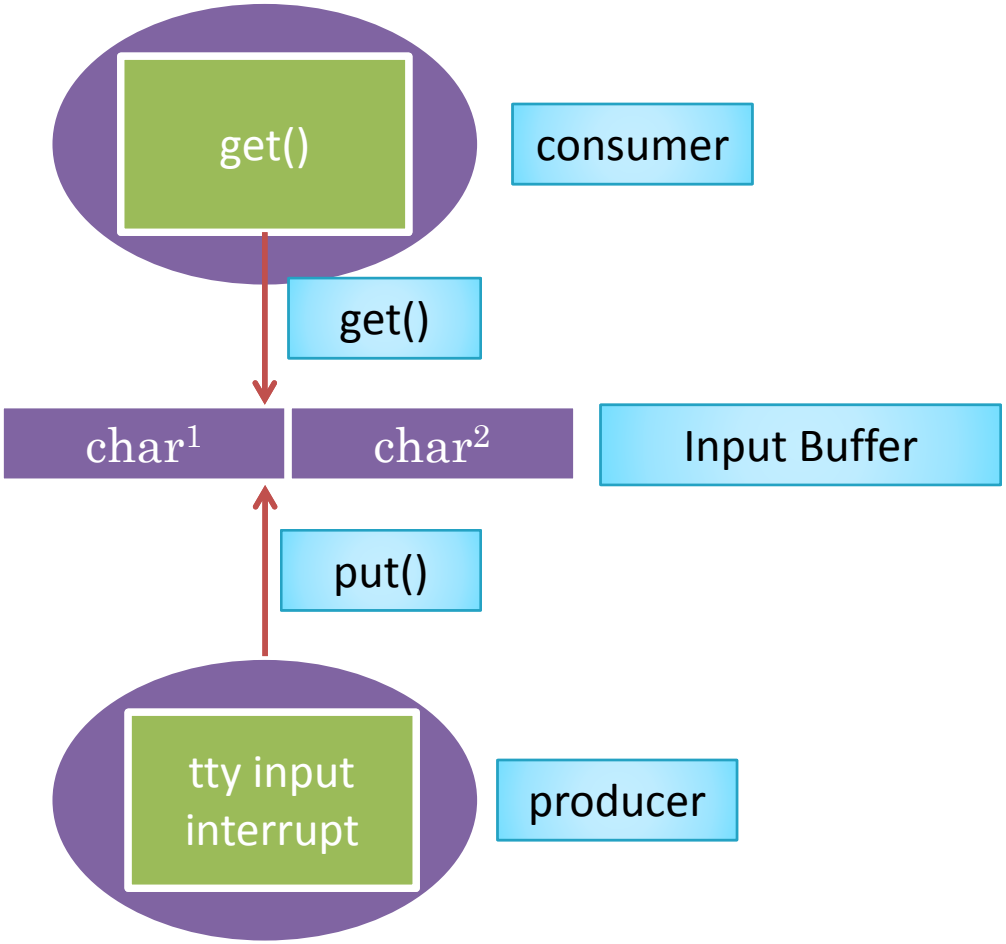
Enqueues printed characters

Lower half executes in interrupting thread

Consumes next char and writes it to device register

Device interrupts after output register written/ cleared, and after interrupt posting enabled

TTY Input Driver: Producer, Consumer



Upper half executes in application thread

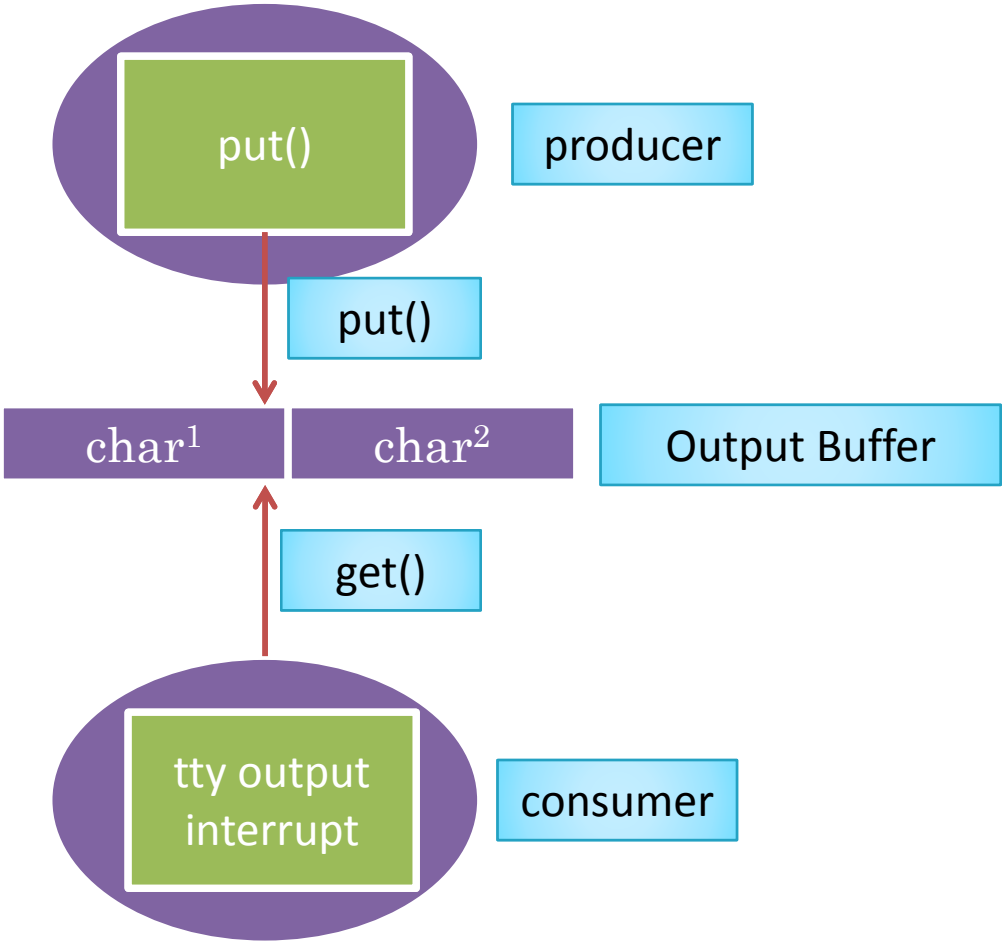
Consumes characters, blocks if buffer empty

Lower half executes in interrupting thread

Reads device register and puts in read buffer

Device interrupts each time a new character written to register

TTY Output Driver: Blocking

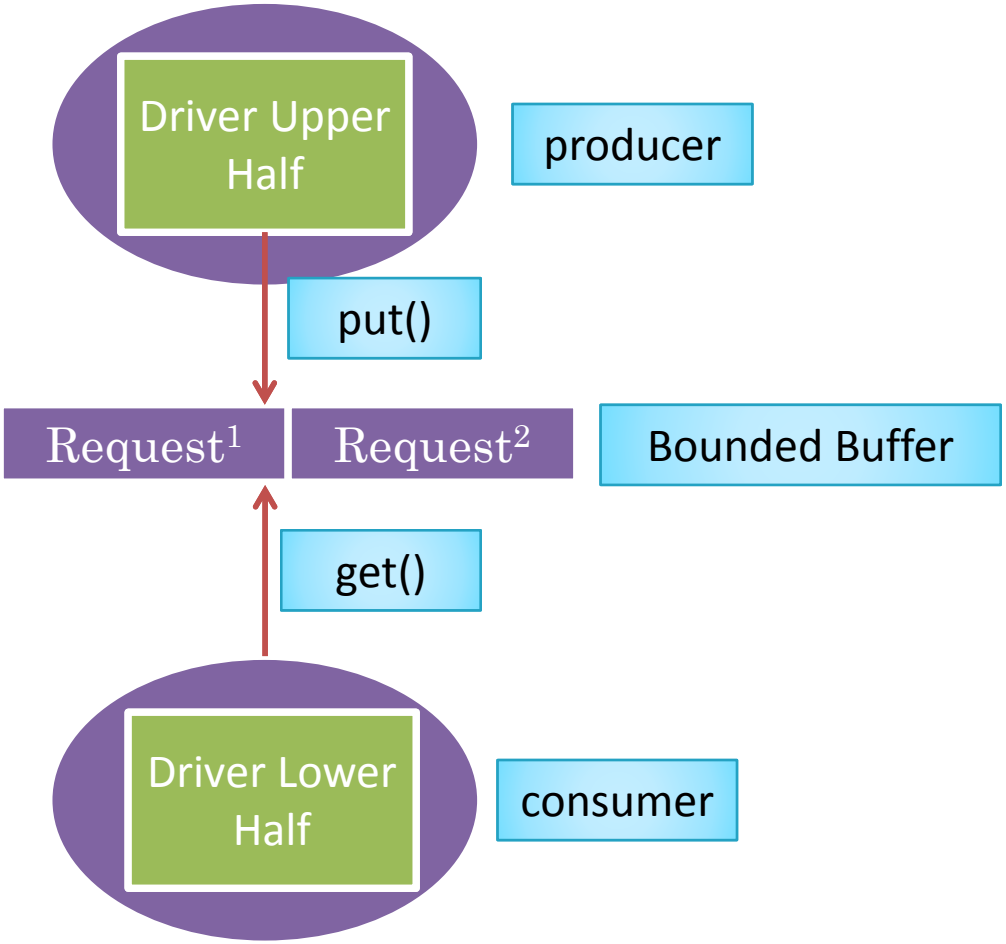


Print blocks if output buffer full

Output buffer in kernel space

Lower-half blocking?

Lower-Half Blocking



Interrupt routines cannot block

Device registers might get full

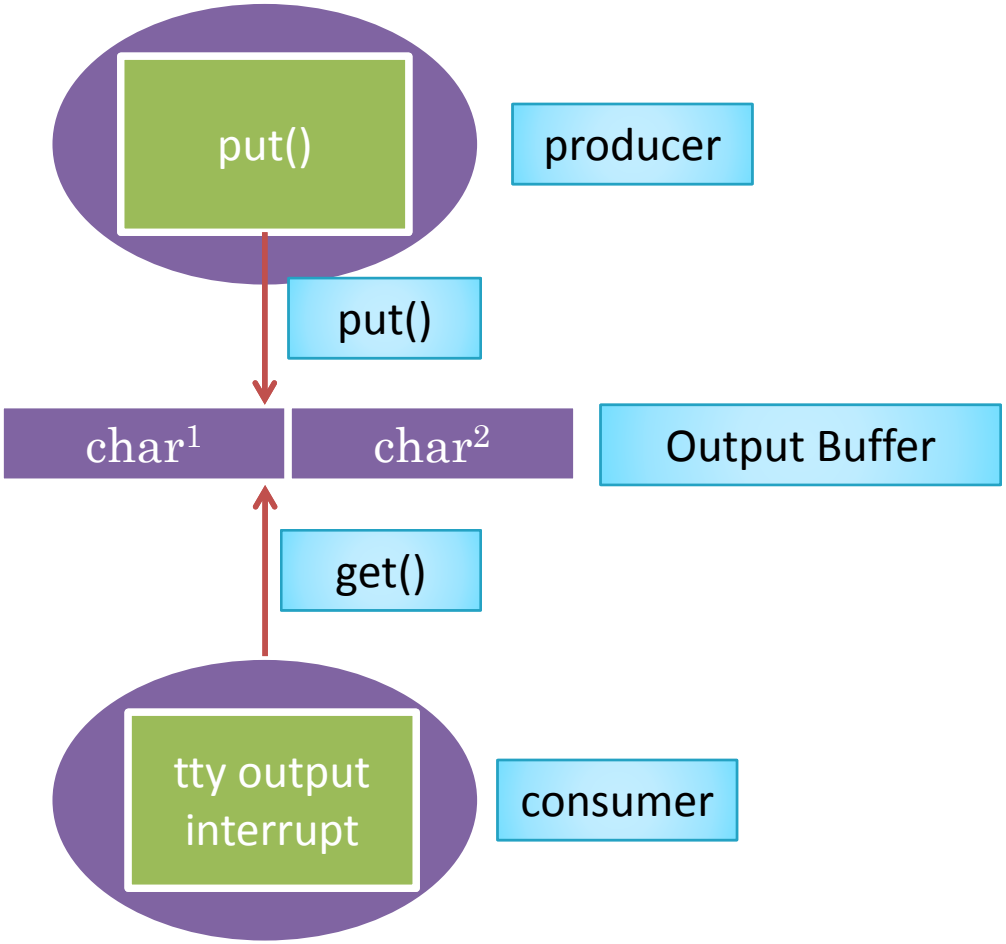
If tty interrupt routine blocks, input device register can be overwritten as user enters new character

Often interrupt-service disabled while they are executing to ensure atomicity

While bounded buffer being manipulated, clock interrupt servicing and context switching should be disabled

Blocked routine can starve or delay threads

TTY Output Driver Cannot Block



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

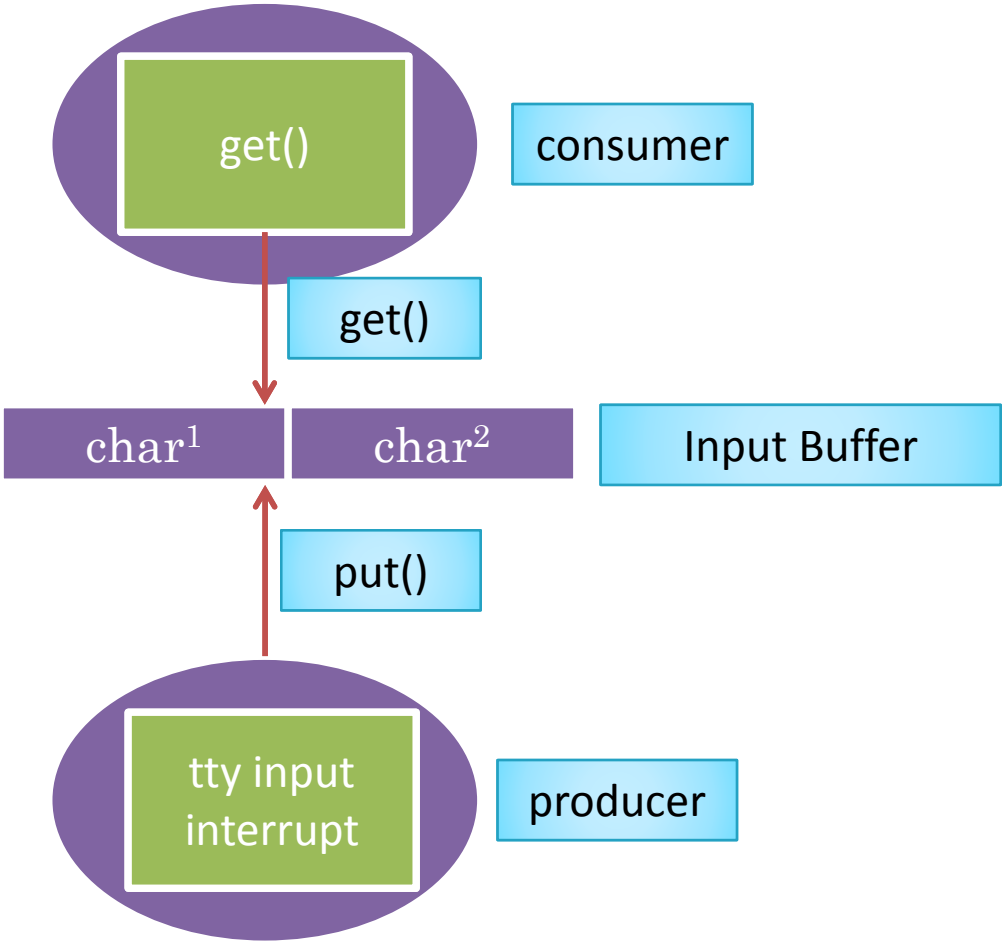
What triggers its call when a new character is entered, as it cannot wait for non empty buffer?

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 called

TTY Input driver Cannot Block



Tty input routine cannot block if input buffer full

Echoes a bell character to user to not type any more

Half Synchronized Bounded Buffer

Output: Producer synchronized: producer waits but not consumer

Input: Consumer synchronized: consumer waits but not produced

Not synchronizing

add(<ElementType> e)

<ElementType>get()

int size()

Java ArrayBlockingQueue <ElementType>

Synchronizing

put(<ElementType> e)

<ElementType>take()

I/O: API

Console	put(char)
	char get()

Put produces writes characters into a bounded buffer consumed by the console device

Get consumes characters from a bounded buffer produced by the console device

Console has two bounded buffers (combining Display and Keyboard devices)

NIO Discussion from First Principles

I/O? ✓

I/O → Inter Process Communication (IPC)?

Blocking → Non- Blocking?