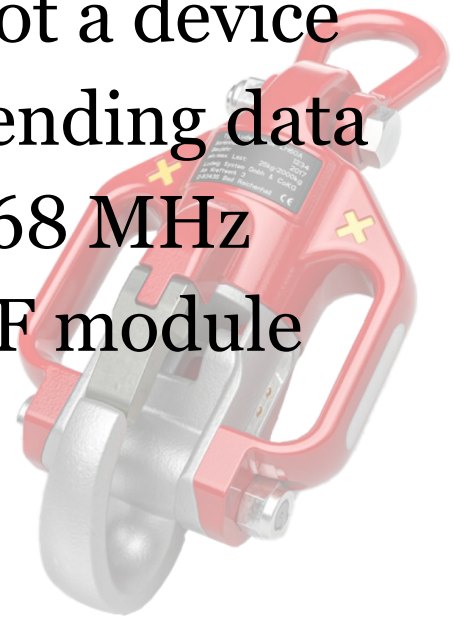
How system calls made my day

A short story

- Hooking up a dongle
- .NET on Windows
- C on Unix/MacOS
- Calling the system
- Daring a rewrite
- Terminal woes
- USB serial ports
- .NET on MacOS
- Reading the code
- All the way down
- Finding a bug
- Getting a fix

Hooking up a dongle

- Got a device
- Sending data
- 868 MHz
- RF module

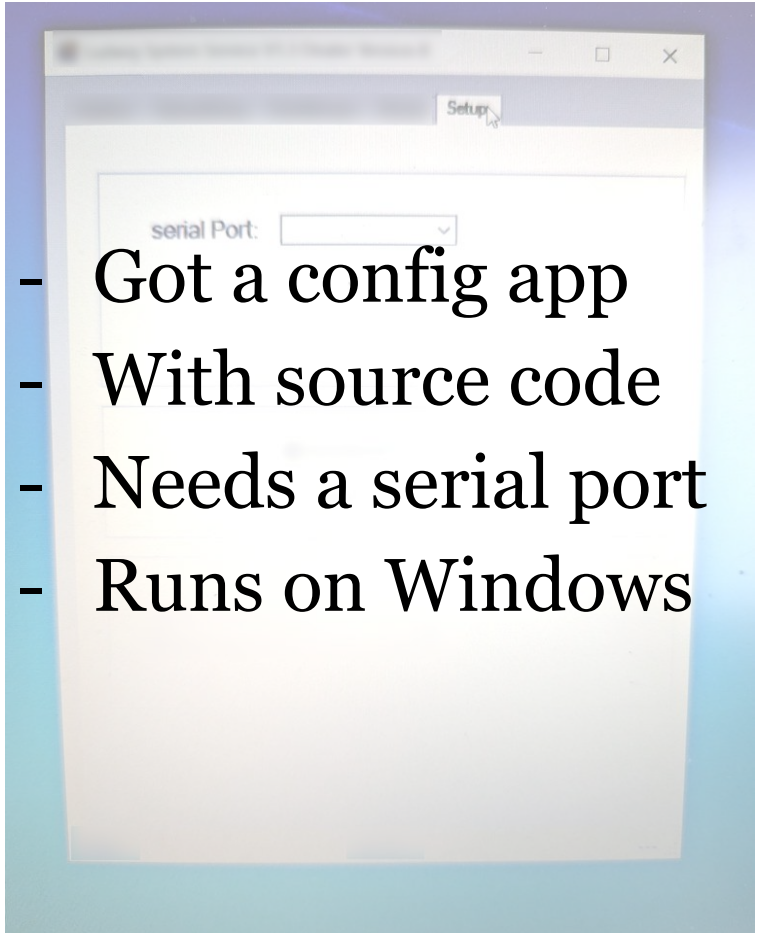


- Got a dongle
- Receiving it
- ISM band
- Via USB

.NET on Windows

- C#
- MSIL
- .NET SDK
- .NET Runtime

It's been a while.

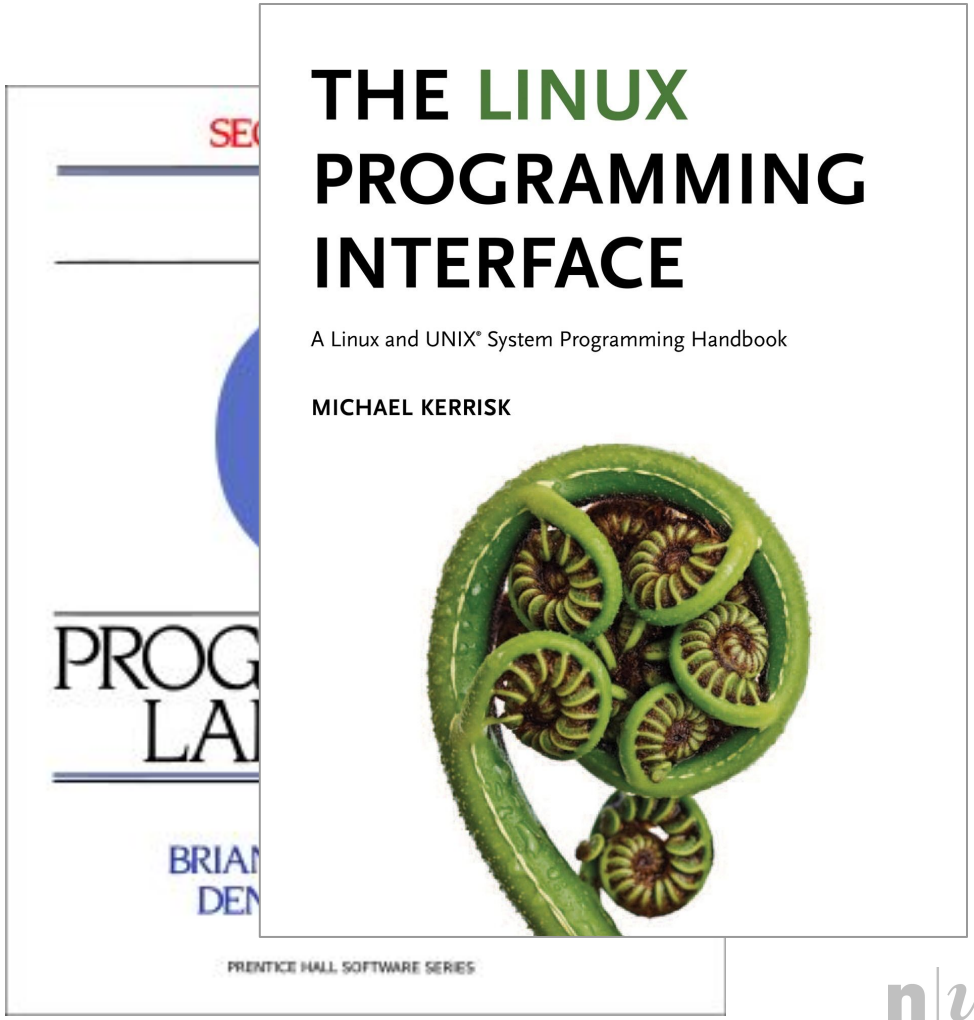


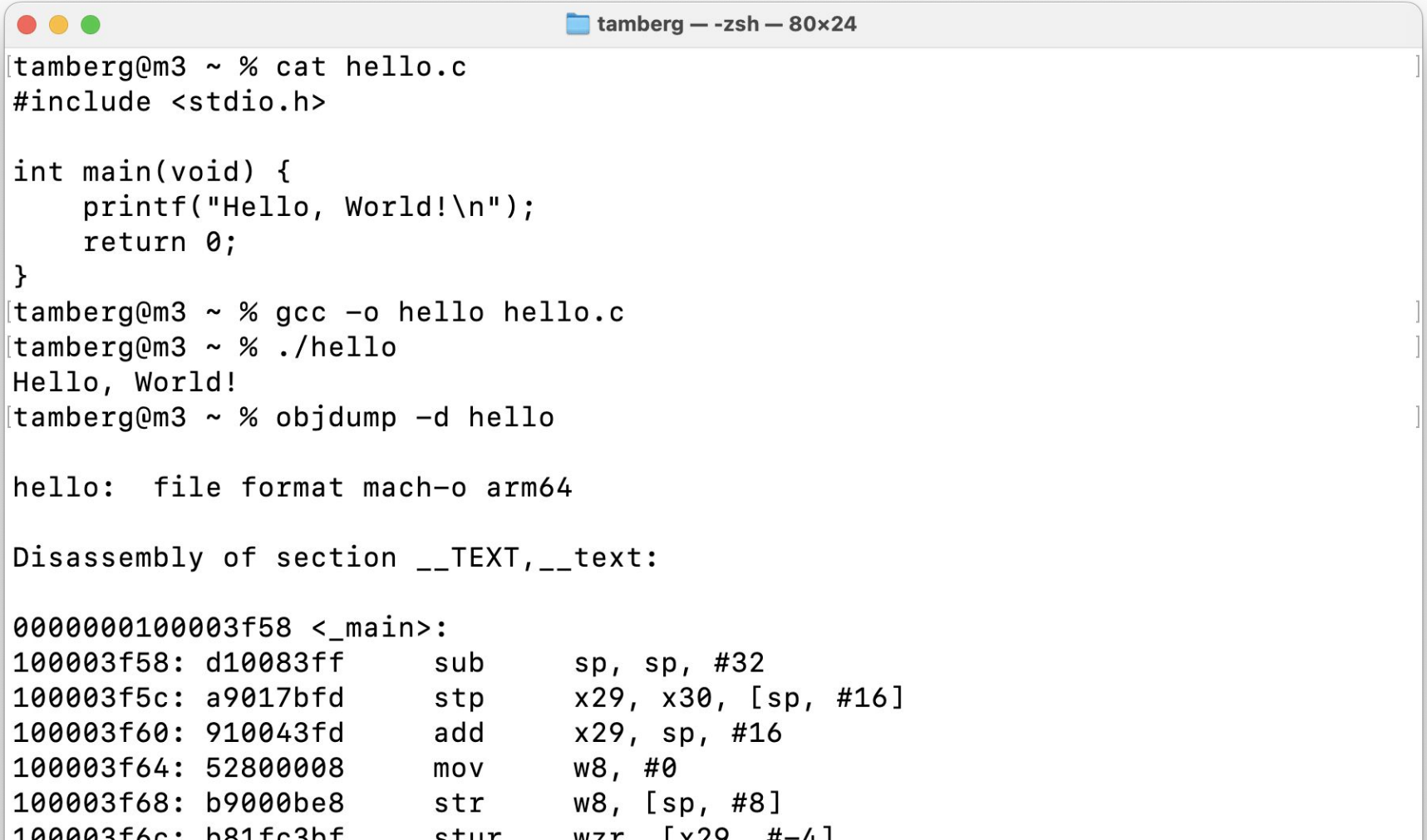
- Got a config app
- With source code
- Needs a serial port
- Runs on Windows

C on Unix/MacOS

- Programming in C
- Teaching syspr

- On Linux
- Or Unix?
- Or MacOS!




tamberg — -zsh — 80×24
tamberg@m3 ~ % cat hello.c
#include <stdio.h>
int main(void) {
printf("Hello, World!\n");
return 0;
}
tamberg@m3 ~ % gcc -o hello hello.c
tamberg@m3 ~ % ./hello
Hello, World!
tamberg@m3 ~ % objdump -d hello
hello: file format mach-o arm64
Disassembly of section __TEXT,__text:
0000000100003f58 <_main>:
100003f58: d10083ff sub sp, sp, #32
100003f5c: a9017bfd stp x29, x30, [sp, #16]
100003f60: 910043fd add x29, sp, #16
100003f64: 52800008 mov w8, #0
100003f68: b9000be8 str w8, [sp, #8]

github.com/jart

jart

Type / to search

Overview | Repositories 223 | Projects | Packages | Stars 237 | Sponsoring

Justine Tunney

jart

Follow | Sponsor

9.9k followers · **123** following

Followed by **mat-lo** and 1 more

jtunney@gmail.com

https://justine.lol

Pinned

Mozilla-Ocho/llamafile Public
Distribute and run LLMs with a single file.
C++ 19.5k 985

cosmopolitan Public
build-once run-anywhere c library
C 18k 618

sectorlisp Public
Bootstrapping LISP in a Boot Sector
C 1.3k 57

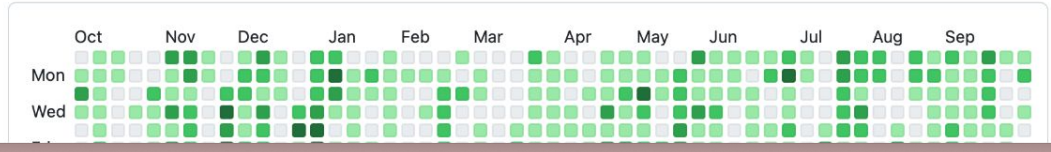
blink Public
tiniest x86-64-linux emulator
C 6.9k 218

bestline Public
ANSI Standard X3.64 Teletypewriter Command Session Library
C 448 29

landlock-make Public
Sandboxing for GNU Make has never been easier
C++ 224 3

1,467 contributions in the last year

2024 | 2023 | 2022



README ISC license

Cosmopolitan

Cosmopolitan Libc makes C a build-once run-anywhere language, like Java, except it doesn't need an interpreter or virtual machine. Instead, it reconfigures stock GCC and Clang to output a POSIX-approved polyglot format that runs natively on Linux + Mac + Windows + FreeBSD + OpenBSD + NetBSD + BIOS with the best possible performance and the tiniest footprint imaginable.

Background

For an introduction to this project, please read the actually portable executable blog post and cosmopolitan libc website. We also have API documentation.

αcτμally pδrτablε εxεcμτablε

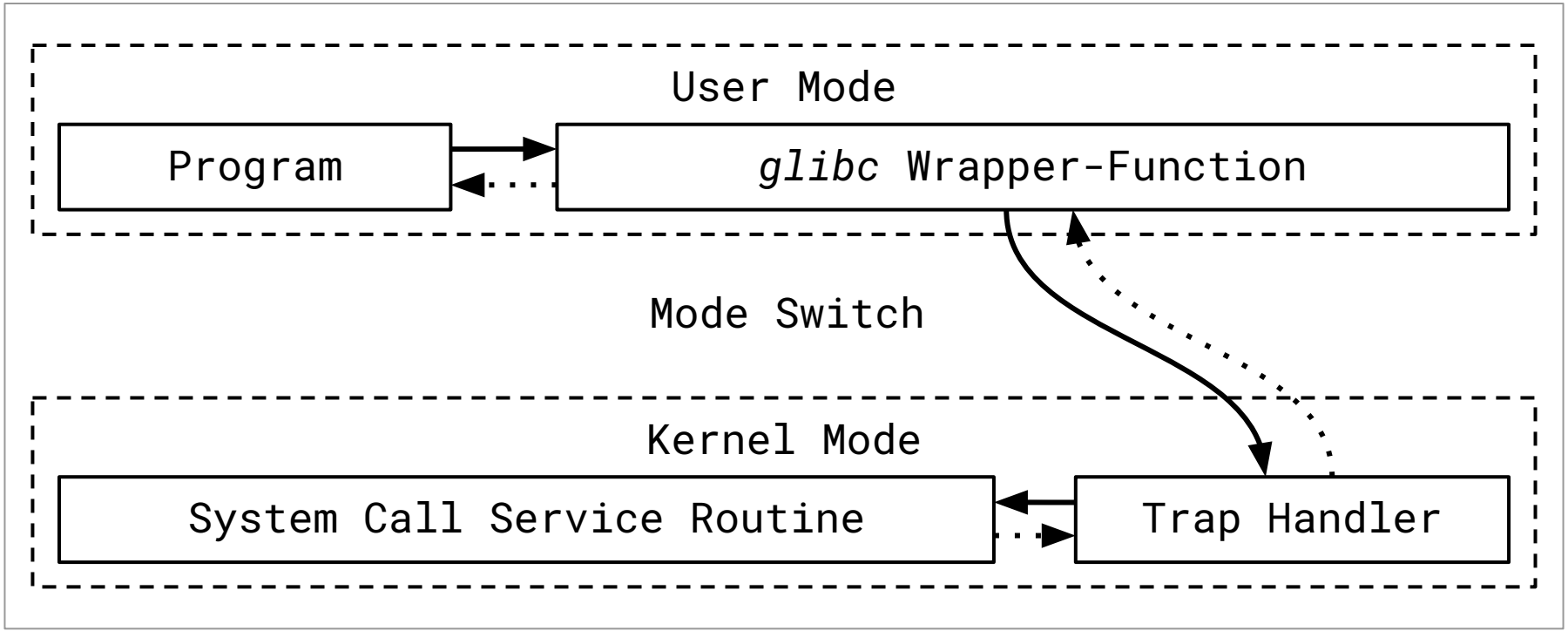
24 aug 2020 @ justine's web page

One day, while studying old code, I found out that it's possible to encode Windows Portable Executable files as a UNIX Sixth Edition shell script, due to the fact that the Thompson Shell didn't use a shebang line. Once I realized it's possible to create a synthesis of the binary formats being used by Unix, Windows, and MacOS, I couldn't resist the temptation of making it a reality, since it means that high-performance native code can be almost as pain-free as web apps. Here's how it works:

```
MZqFpD='
BIOS BOOT SECTOR'
exec 7<> $(command -v $0)
printf '\177ELF...LINKER-ENCODED-FREEBSD-HEADER' >&7
exec "$0" "$@"
exec qemu-x86_64 "$0" "$@"
exit 1
REAL MODE...
ELF SEGMENTS...
OPENBSD NOTE...
NETBSD NOTE...
MACHO HEADERS...
CODE AND DATA...
ZIP DIRECTORY...
```

I started a project called Cosmopolitan which implements the αcτμally pδrτablε εxεcμτablε format. I chose the name because I like the idea of having the freedom to write software without restrictions that transcends traditional boundaries. My goal has been helping C become a build-once run-anywhere language, suitable for greenfield development, while avoiding any assumptions that would prevent software from being shared between tech

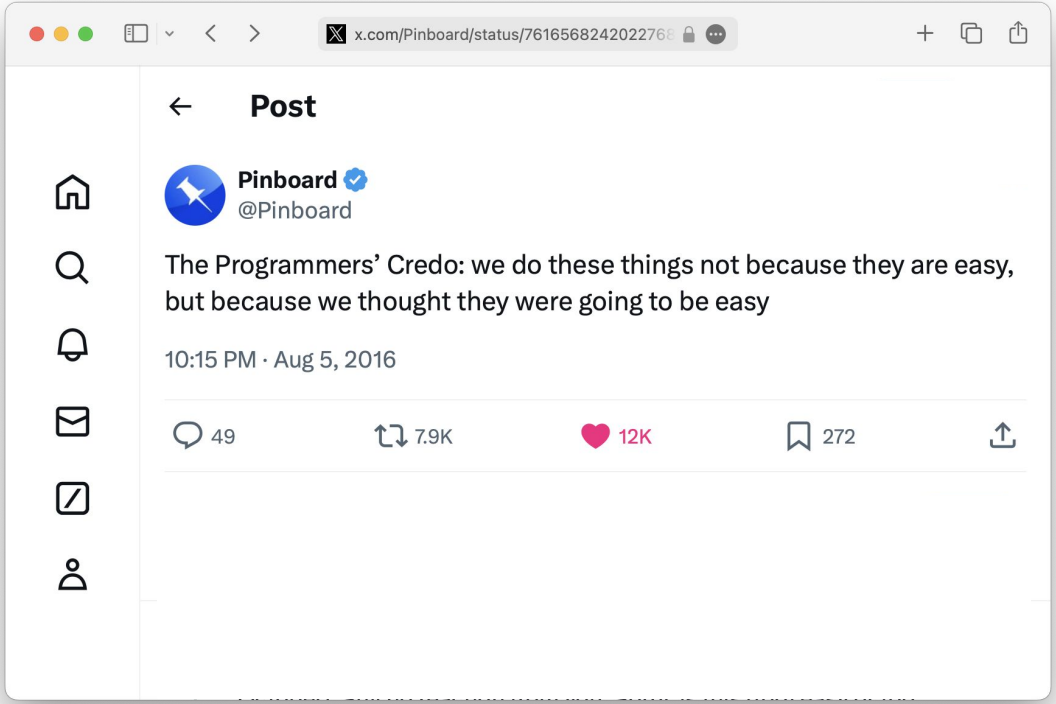
Calling the system



Daring a rewrite

- Can't be hard
- Open a "file"
- Parse bytes

- *open()*
- *read()*



OPEN(2) System Calls Manual

NAME

open, **openat** — open or create a file for reading or writing

SYNOPSIS

```
#include <fcntl.h>

int
open(const char *path, int oflag, ...);

int
openat(int fd, const char *path, int oflag, ...);
```

DESCRIPTION

The file name specified by path is opened for reading and/or writing, as specified by the argument oflag; the file descriptor is returned to the calling process.

The oflag argument may indicate that the file is to be created if it does not exist (by specifying the O_CREAT flag). In this case, **open**() and **openat**() require an additional argument mode_t mode; the file is created with mode mode as described in chmod(2) and modified by the process'

READ(2) System Calls Manual

NAME

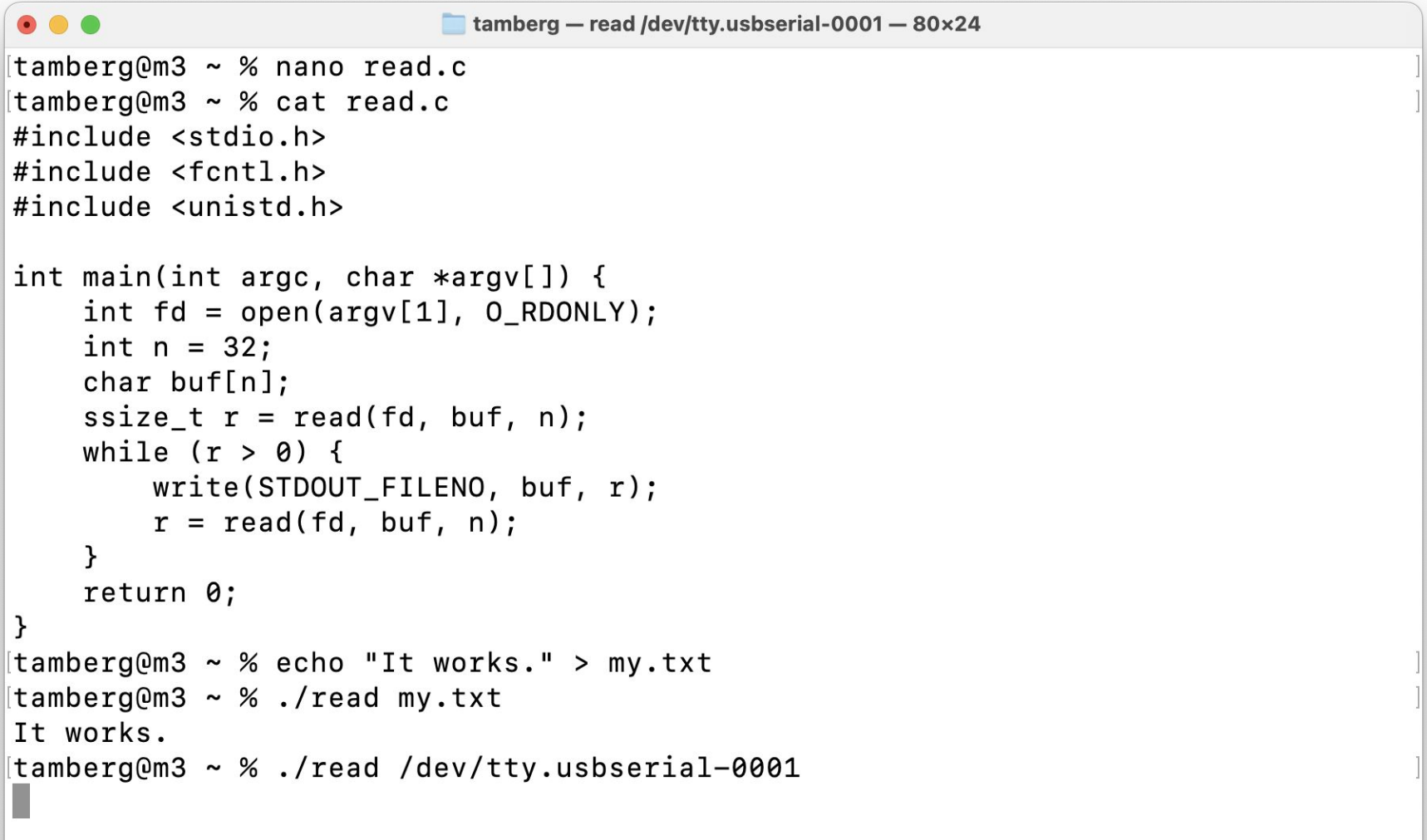
pread, **read**, **preadv**, **readv** — read input

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>

ssize_t
pread(int d, void *buf, size_t nbyte, off_t offset);

ssize_t
read(int fildes, void *buf, size_t nbyte);

ssize_t
preadv(int d, const struct iovec *iov, int iovcnt, off_t offset);

ssize_t
```

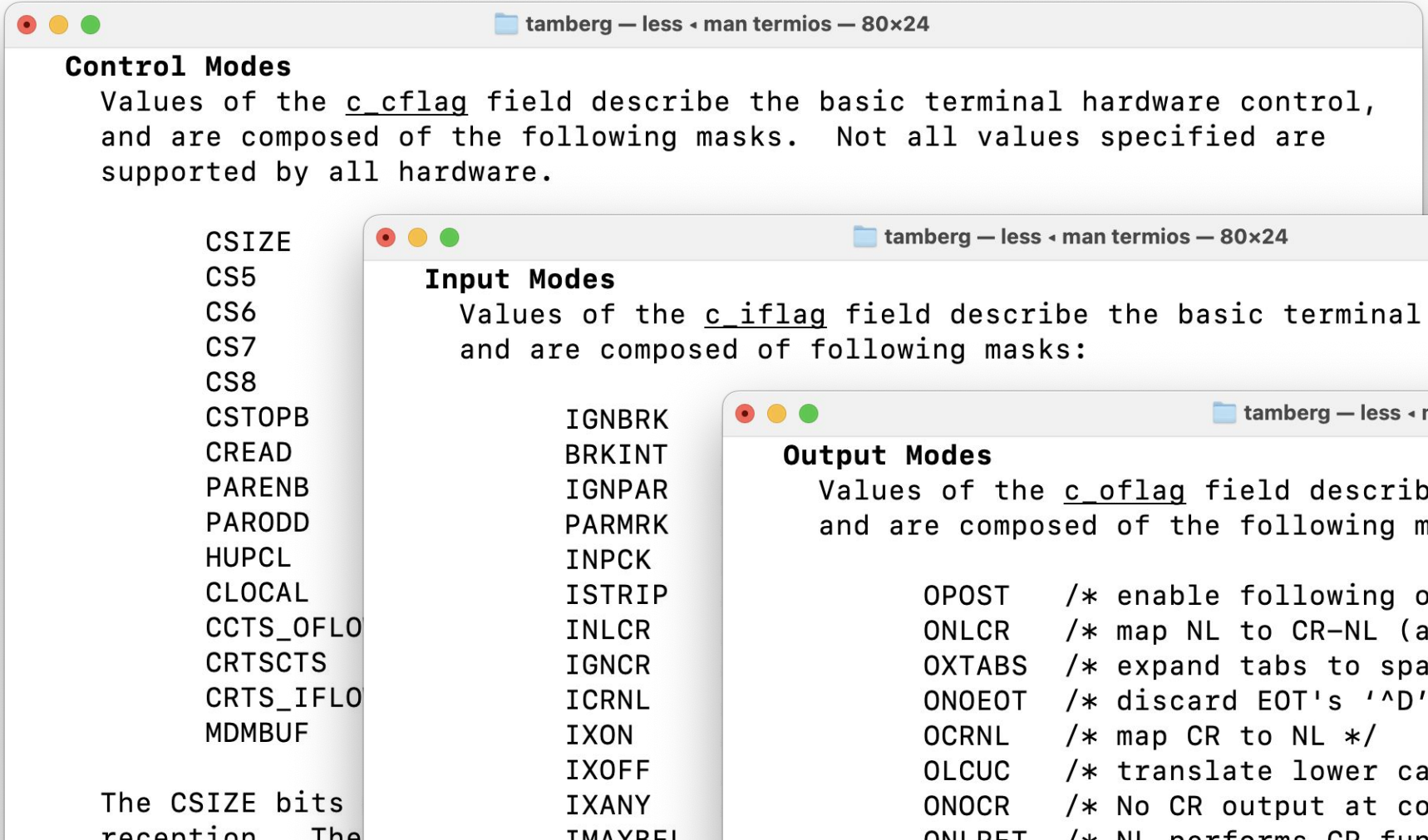

tamberg — read /dev/tty.usbserial-0001 — 80×24
tamberg@m3 ~ % nano read.c
tamberg@m3 ~ % cat read.c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
int fd = open(argv[1], O_RDONLY);
int n = 32;
char buf[n];
ssize_t r = read(fd, buf, n);
while (r > 0) {
write(STDOUT_FILENO, buf, r);
r = read(fd, buf, n);
}
return 0;
}
tamberg@m3 ~ % echo "It works." > my.txt
tamberg@m3 ~ % ./read my.txt
It works.
tamberg@m3 ~ % ./read /dev/tty.usbserial-0001

Terminal woes

- *struct termios t*
- *tcgetattr(fd, &t)*
- *t.c_iflag &= ~(...)**
- *tcsetattr(fd, ..., &t)*

- Terminal modes
- Cooked/cbreak/raw
- Non-/canonical
- MIN, TIME
- Baud rate

*But which flags?


tamberg — less ◂ man termios — 80×24
Control Modes
Values of the c_cflag field describe the basic terminal hardware control,
and are composed of the following masks. Not all values specified are
supported by all hardware.
CSIZE
CS5
CS6
CS7
CS8
CSTOPB
CREAD
PARENB
PARODD
HUPCL
CLOCAL
CCTS_OFLO
CRTSCTS
CRTS_IFLO
MDMBUF
The CSIZE bits
tamberg — less ◂ man termios — 80×24
Input Modes
Values of the c_iflag field describe the basic terminal inpu
and are composed of following masks:
IGNBRK
BRKINT
IGNPAR
PARMRK
INPCK
ISTRIP
INLCR
IGNCR
ICRNL
IXON
IXOFF
IXANY
tamberg — less ◂ man term
Output Modes
Values of the c_oflag field describe the
and are composed of the following masks:
OPOST /* enable following output
ONLCR /* map NL to CR-NL (ala CR
OXTABS /* expand tabs to spaces *
ONOEOT /* discard EOT's '^D' on o
OCRNL /* map CR to NL */
OLCUC /* translate lower case to
ONOCR /* No CR output at column

Local Modes

Values of the c_lflag field describe the control of various functions, and are composed of the following masks.

```
ECHOKE      /* visual erase for line kill */
ECHOE       /* visually erase chars */
ECHO        /* enable echoing */
ECHONL      /* echo NL even if ECHO is off */
ECHOPRT     /* visual erase mode for hardcopy */
ECHOCTL     /* echo control chars as ^(Char) */
ISIG        /* enable signals INTR, QUIT, [D]SUSP */
ICANON      /* canonicalize input lines */
ALTWERASE   /* use alternate WERASE algorithm */
IEXTEN      /* enable DISCARD and LNEXT */
EXTPROC     /* external processing */
TOSTOP      /* stop background jobs from output */
FLUSHO      /* output being flushed (state) */
NOKERNINFO  /* no kernel output from VSTATUS */
PENDIN      /* XXX retype pending input (state) */
NOFLSH      /* don't flush after interrupt */
```

If ECHO is set, input characters are echoed back to the terminal. If ECHO is not set, input characters are not echoed

Noncanonical Mode Input Processing

In noncanonical mode input processing, input bytes are not assembled into lines, and erase and kill processing does not occur. The values of the MIN and TIME members of the c_cc array are used to determine how to process the bytes received.

MIN represents the minimum number of bytes that should be received when the read function successfully returns. TIME is a timer of 0.1 second granularity that is used to time out bursty and short term data transmissions. If MIN is greater than { MAX_INPUT}, the response to the request is undefined. The four possible values for MIN and TIME and their interactions are described below.

Case A: MIN > 0, TIME > 0

In this case TIME serves as an inter-byte timer and is activated after the first byte is received. Since it is an inter-byte timer, it is reset after a byte is received. The interaction between MIN and TIME is as follows: as soon as one byte is received, the inter-byte timer is started. If MIN bytes are received before the inter-byte timer expires (remember that the timer is reset upon receipt of each byte), the read is satisfied. If the timer expires before MIN bytes are received, the characters received to that point are returned to the user. Note that if

GETTING AND SETTING THE BAUD RATE

The input and output baud rates are found in the termios structure. The unsigned integer speed_t is typedef'd in the include file ⟨termios.h⟩. The value of the integer corresponds directly to the baud rate being represented; however, the following symbolic values are defined:

```
#define B300    300
#define B600    600
#define B1200   1200
#define B1800   1800
#define B2400   2400
#define B4800   4800
#define B9600   9600
#define B19200  19200
#define B38400  38400
#ifndef _POSIX_C_SOURCE
#endif  /*_POSIX_C_SOURCE */
```

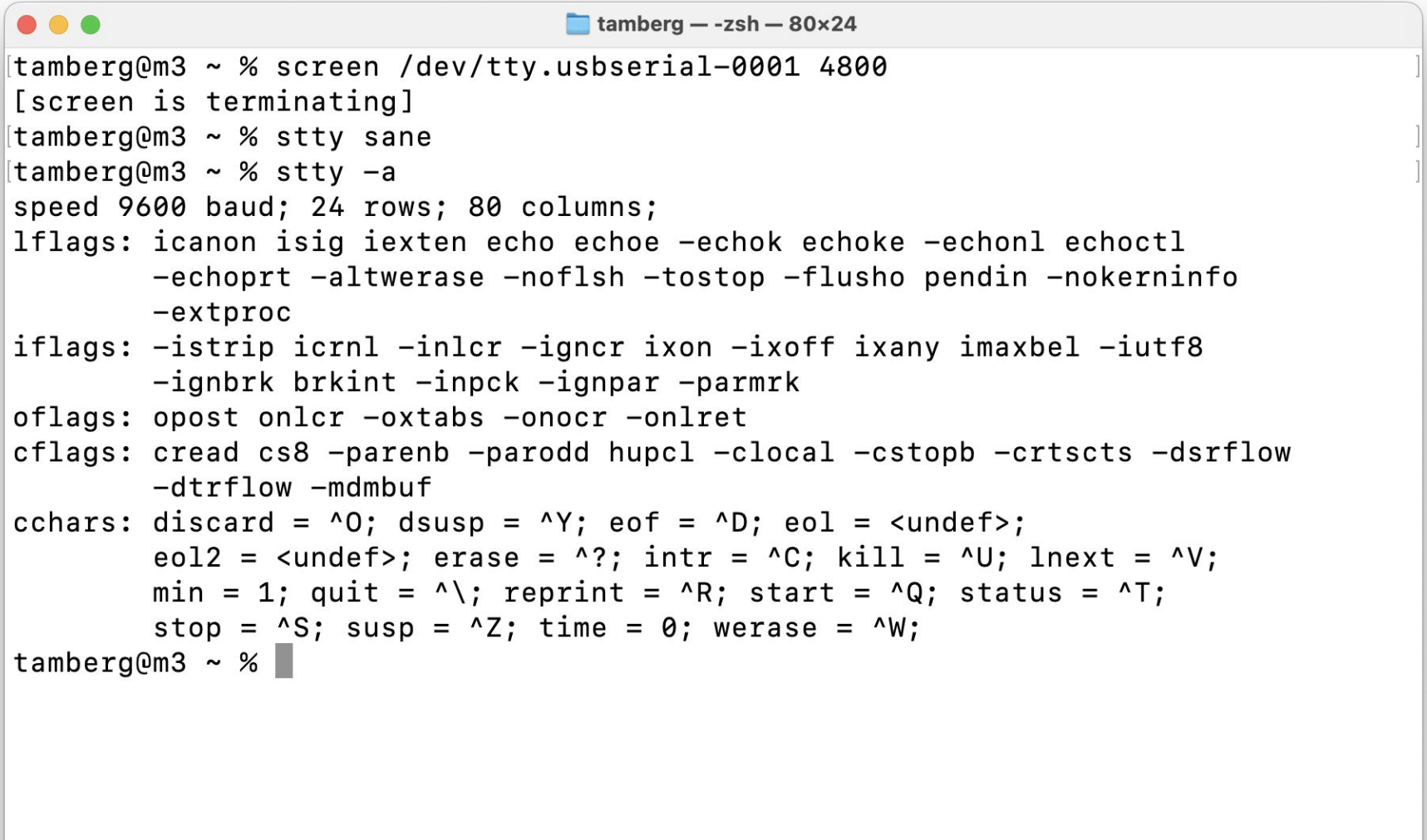
The **cfgetispeed**() function returns the input baud rate in the termios structure referenced by termios_p.

The **cfsetispeed**() function sets the input baud rate in the termios

USB serial ports

- *COM3* on Windows
- *PuTTY* on Windows
- To read and write
- E.g. AT commands*
- */dev/tty.u<TAB>*
- *screen* on MacOS
- *CTRL-a-k y* to quit
- *stty -a* to see config**

*But no ASCII here; **no *stty* on Windows.

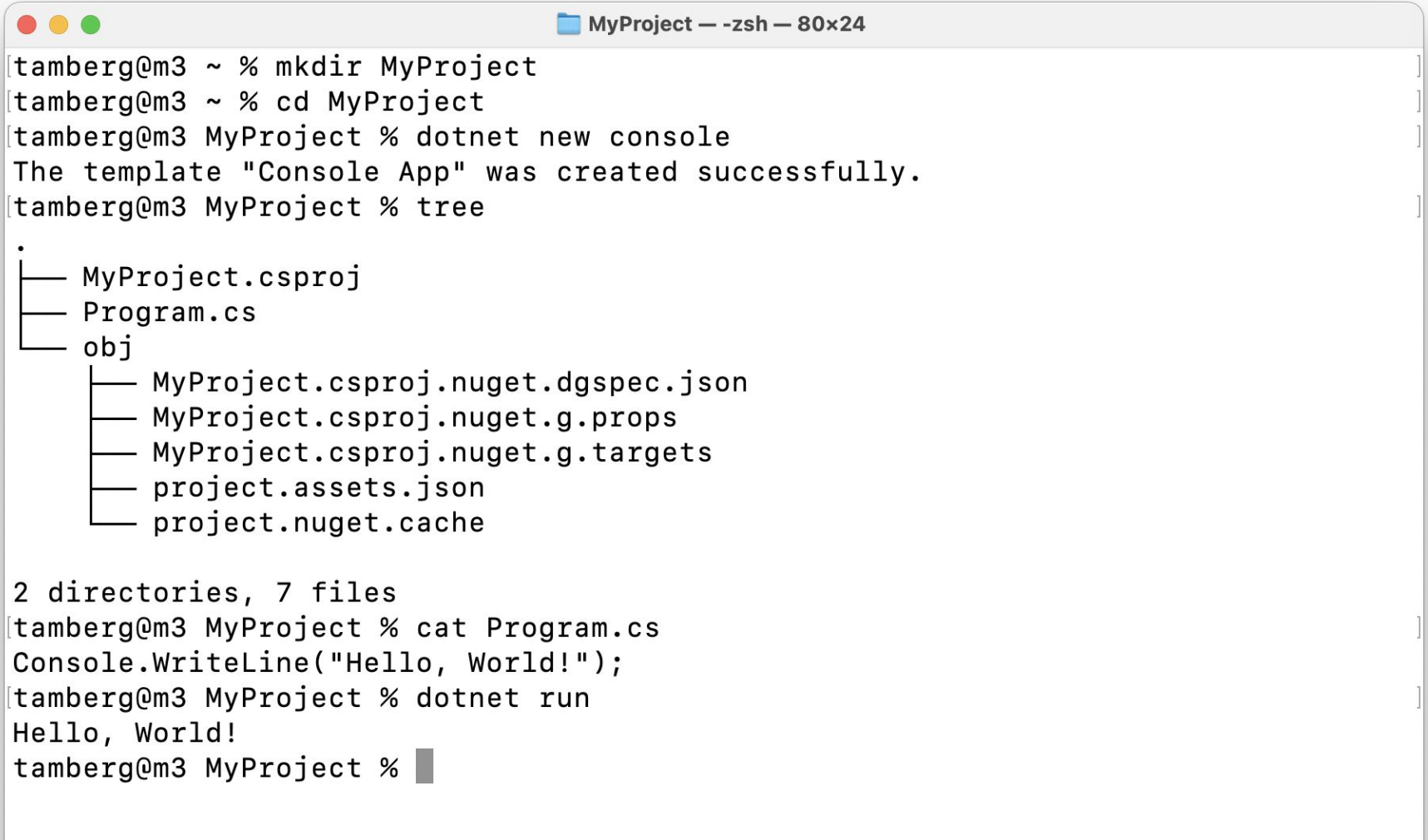

tamberg — -zsh — 80×24
tamberg@m3 ~ % screen /dev/tty.usbserial-0001 4800
[screen is terminating]
tamberg@m3 ~ % stty sane
tamberg@m3 ~ % stty -a
speed 9600 baud; 24 rows; 80 columns;
lflags: icanon isig iexten echo echoe -echok echoke -echonl echoctl
-echoprt -altwerase -noflsh -tostop -flusho pendin -nokerninfo
-extproc
iflags: -istrip icrnl -inlcr -igncr ixon -ixoff ixany imaxbel -iutf8
-ignbrk brkint -inpck -ignpar -parmrk
oflags: opost onlcr -oxtabs -onocr -onlret
cflags: cread cs8 -parenb -parodd hupcl -clocal -cstopb -crtscts -dsrflow
-dtrflow -mdmbuf
cchars: discard = ^O; dsusp = ^Y; eof = ^D; eol = <undef>;
eol2 = <undef>; erase = ^?; intr = ^C; kill = ^U; lnext = ^V;
min = 1; quit = ^\; reprint = ^R; start = ^Q; status = ^T;
stop = ^S; susp = ^Z; time = 0; werase = ^W;
tamberg@m3 ~ %

.NET on MacOS

- .NET Core
- Fork of Mono
- .NET, since 5.0
- *dotnet run*

- *System.IO.Ports*
- *new SerialPort()*
- *System.Diagnostics*
- *Process.Start()*
- Call *stty -a* in proc*

*But "resource busy".


MyProject — -zsh — 80×24
tamberg@m3 ~ % mkdir MyProject
tamberg@m3 ~ % cd MyProject
tamberg@m3 MyProject % dotnet new console
The template "Console App" was created successfully.
tamberg@m3 MyProject % tree
.
├── MyProject.csproj
├── Program.cs
└── obj
    ├── MyProject.csproj.nuget.dgspec.json
    ├── MyProject.csproj.nuget.g.props
    ├── MyProject.csproj.nuget.g.targets
    ├── project.assets.json
    └── project.nuget.cache
2 directories, 7 files
tamberg@m3 MyProject % cat Program.cs
Console.WriteLine("Hello, World!");
tamberg@m3 MyProject % dotnet run
Hello, World!
tamberg@m3 MyProject %

MyProject — -zsh — 80×24

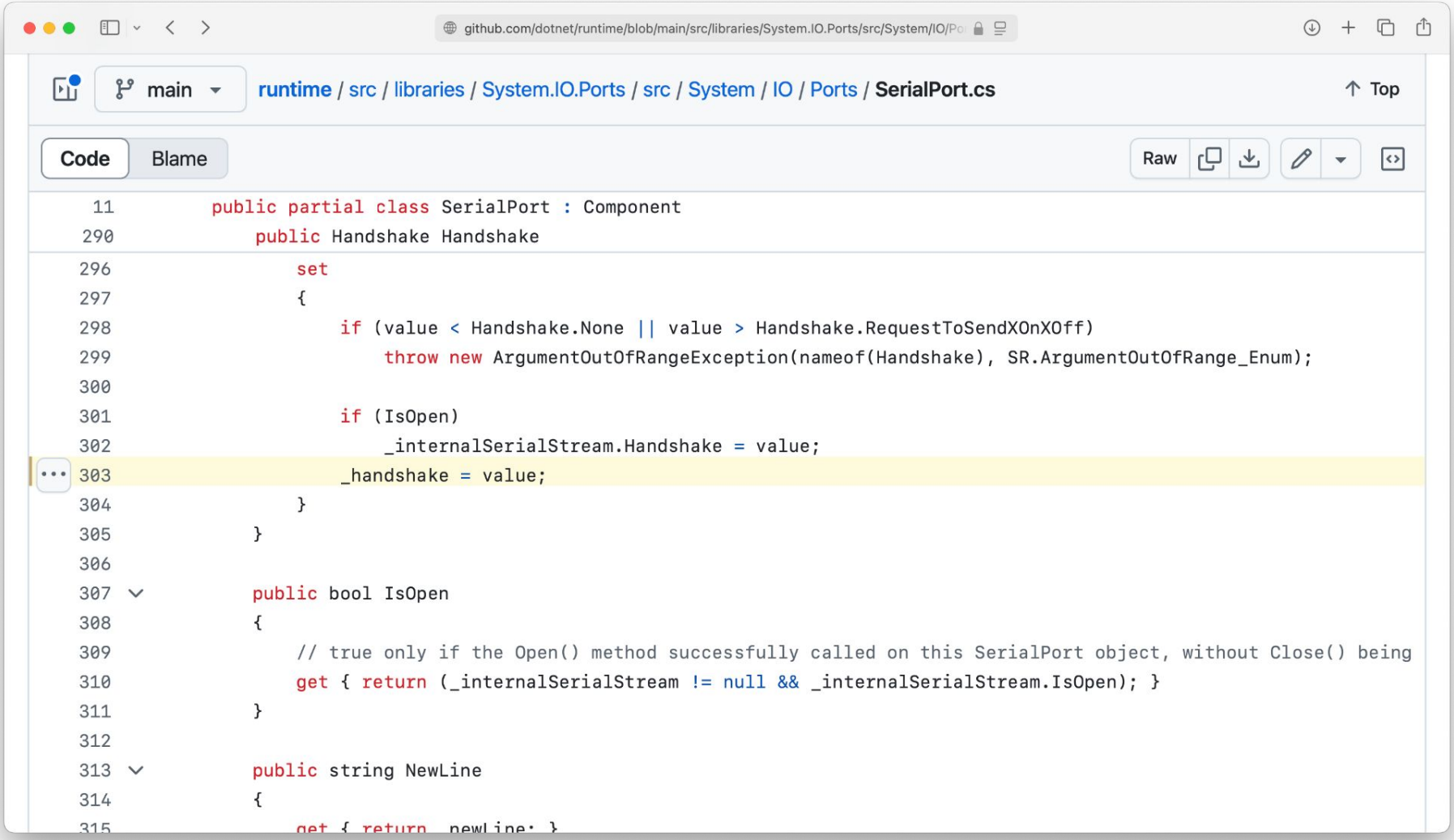
```
tamberg@m3 MyProject % cat Program.cs
using System;
using System.Diagnostics;
using System.IO.Ports;

class Program {
    static void Main () {
        int i = 0;
        string[] names = SerialPort.GetPortNames();
        foreach(string name in names) {
            Console.WriteLine(i + ": " + name);
            i++;
        }
        int n = Int32.Parse(Console.ReadLine());
        SerialPort port = new SerialPort(names[n]);
        port.Handshake = Handshake.None; // and others
        port.BaudRate = 4800;
        port.Open();
        Process proc = Process.Start("stty", "-a -f " + names[n]);
        proc.WaitForExit();
    }
}
```

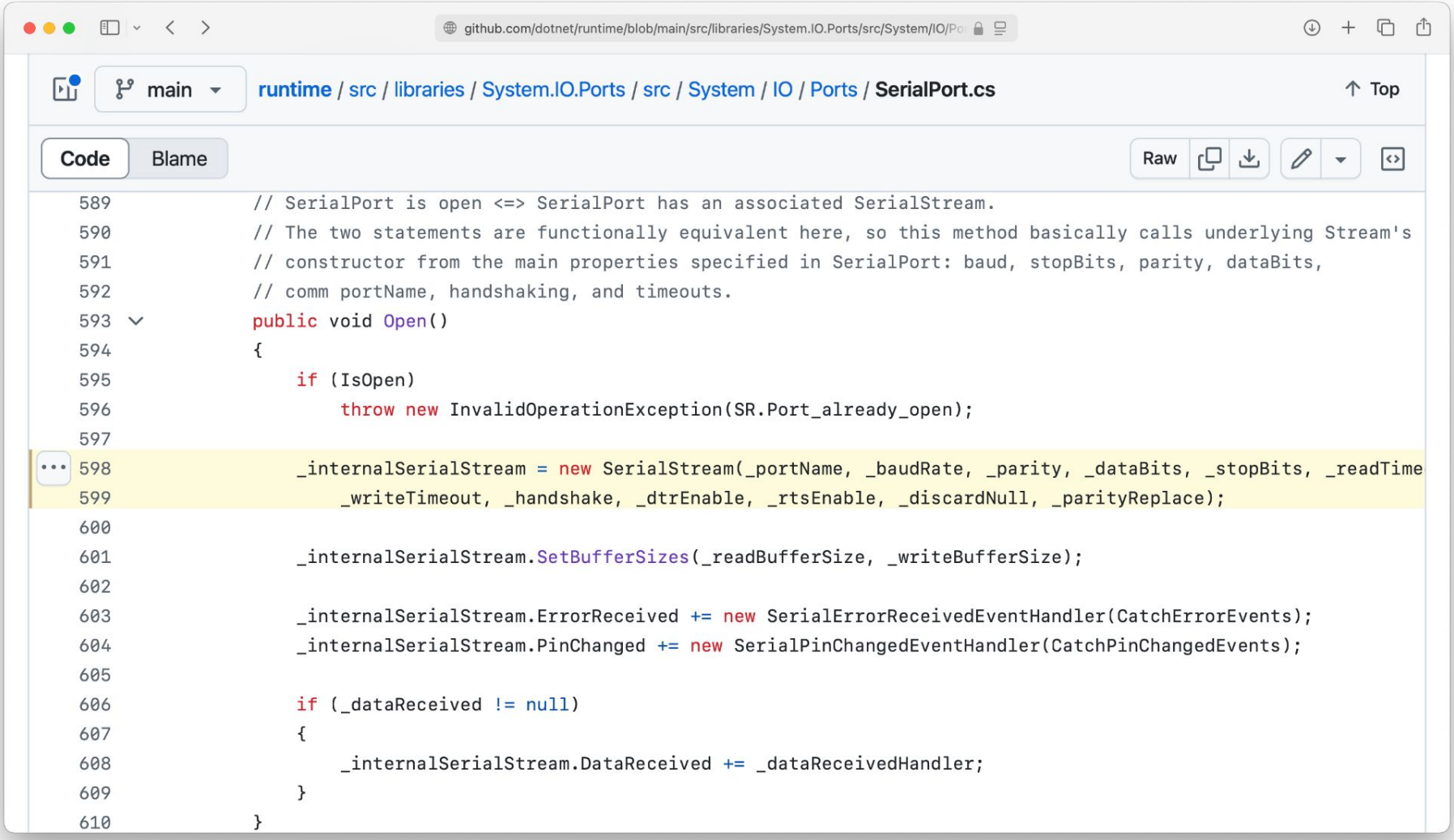
Reading the code

- github.com/dotnet
- .NET Foundation
- Any OSI license

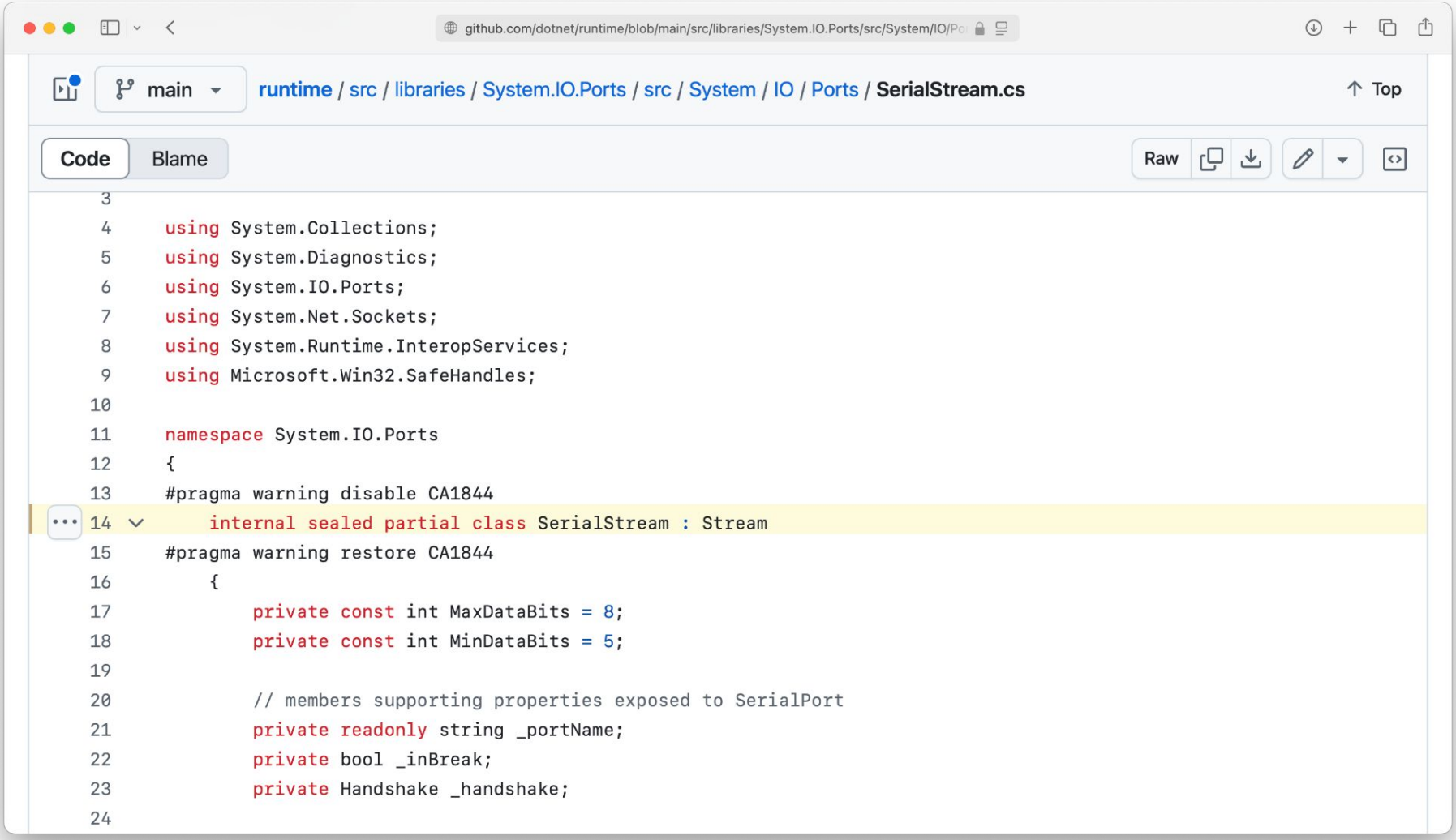
- E.g. the runtime
- MIT License

- Of course, M$ is M$
- isdotnetopen.com
- It's complicated


github.com/dotnet/runtime/blob/main/src/libraries/System.IO.Ports/src/System/IO/Po
main
runtime / src / libraries / System.IO.Ports / src / System / IO / Ports / SerialPort.cs
Top
Code
Blame
Raw
11 public partial class SerialPort : Component
290 public Handshake Handshake
296 set
297 {
298 if (value < Handshake.None || value > Handshake.RequestToSendXOnXOff)
299 throw new ArgumentOutOfRangeException(nameof(Handshake), SR.ArgumentOutOfRange_Enum);
300
301 if (IsOpen)
302 _internalSerialStream.Handshake = value;
303 _handshake = value;
304 }
305 }
306
307 public bool IsOpen
308 {
309 // true only if the Open() method successfully called on this SerialPort object, without Close() being
310 get { return (_internalSerialStream != null && _internalSerialStream.IsOpen); }
311 }
312
313 public string NewLine
314 {
315 get { return _newLine; }




```
589         // SerialPort is open <=> SerialPort has an associated SerialStream.
590         // The two statements are functionally equivalent here, so this method basically calls underlying Stream's
591         // constructor from the main properties specified in SerialPort: baud, stopBits, parity, dataBits,
592         // comm portName, handshaking, and timeouts.
593         public void Open()
594         {
595             if (IsOpen)
596                 throw new InvalidOperationException(SR.Port_already_open);
597
598             _internalSerialStream = new SerialStream(_portName, _baudRate, _parity, _dataBits, _stopBits, _readTime
599                 _writeTimeout, _handshake, _dtrEnable, _rtsEnable, _discardNull, _parityReplace);
600
601             _internalSerialStream.SetBufferSizes(_readBufferSize, _writeBufferSize);
602
603             _internalSerialStream.ErrorReceived += new SerialErrorReceivedEventHandler(CatchErrorEvents);
604             _internalSerialStream.PinChanged += new SerialPinChangedEventHandler(CatchPinChangedEvents);
605
606             if (_dataReceived != null)
607             {
608                 _internalSerialStream.DataReceived += _dataReceivedHandler;
609             }
610         }
```

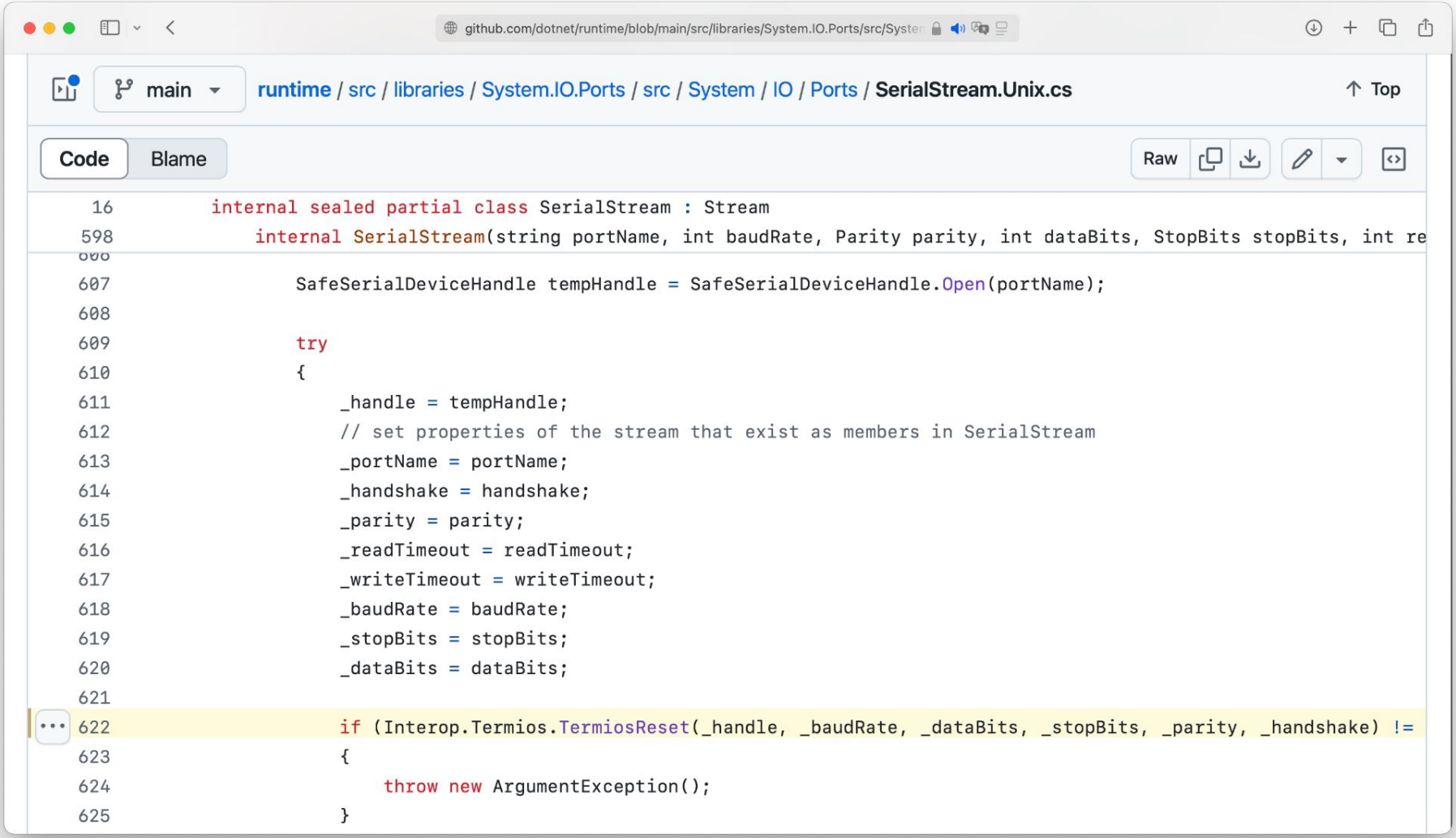

github.com/dotnet/runtime/blob/main/src/libraries/System.IO.Ports/src/System/IO/Po
main
runtime / src / libraries / System.IO.Ports / src / System / IO / Ports / SerialStream.cs
Top
Code
Blame
Raw
3
4 using System.Collections;
5 using System.Diagnostics;
6 using System.IO.Ports;
7 using System.Net.Sockets;
8 using System.Runtime.InteropServices;
9 using Microsoft.Win32.SafeHandles;
10
11 namespace System.IO.Ports
12 {
13 #pragma warning disable CA1844
14 internal sealed partial class SerialStream : Stream
15 #pragma warning restore CA1844
16 {
17 private const int MaxDataBits = 8;
18 private const int MinDataBits = 5;
19
20 // members supporting properties exposed to SerialPort
21 private readonly string _portName;
22 private bool _inBreak;
23 private Handshake _handshake;
24

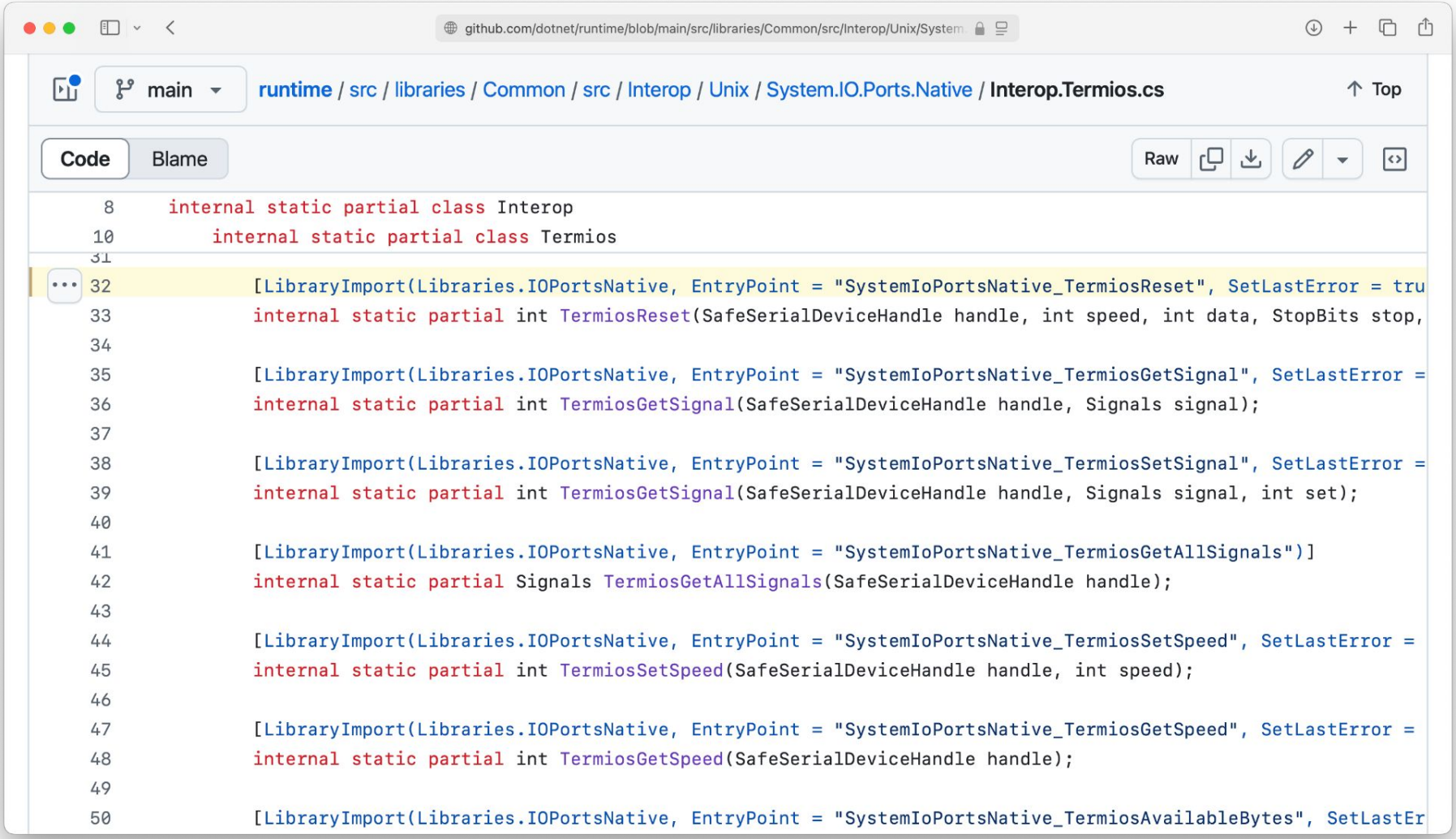
All the way down

- C#, as any language
- Uses system calls
- On Unix systems
- Like MacOS

- .NET libraries call C
- Using native interop

- *SerialStream.Unix.cs*
- *Interop.Termios.cs*
- *pal_termios.c*


github.com/dotnet/runtime/blob/main/src/libraries/System.IO.Ports/src/Syste
main
runtime / src / libraries / System.IO.Ports / src / System / IO / Ports / SerialStream.Unix.cs
Top
Code
Blame
Raw
16 internal sealed partial class SerialStream : Stream
598 internal SerialStream(string portName, int baudRate, Parity parity, int dataBits, StopBits stopBits, int re
607 SafeSerialDeviceHandle tempHandle = SafeSerialDeviceHandle.Open(portName);
608
609 try
610 {
611 _handle = tempHandle;
612 // set properties of the stream that exist as members in SerialStream
613 _portName = portName;
614 _handshake = handshake;
615 _parity = parity;
616 _readTimeout = readTimeout;
617 _writeTimeout = writeTimeout;
618 _baudRate = baudRate;
619 _stopBits = stopBits;
620 _dataBits = dataBits;
621
622 if (Interop.Termios.TermiosReset(_handle, _baudRate, _dataBits, _stopBits, _parity, _handshake) !=
623 {
624 throw new ArgumentException();
625 }


github.com/dotnet/runtime/blob/main/src/libraries/Common/src/Interop/Unix/System
main
runtime / src / libraries / Common / src / Interop / Unix / System.IO.Ports.Native / Interop.Termios.cs
Top
Code
Blame
Raw
internal static partial class Interop
internal static partial class Termios
[LibraryImport(Libraries.IOPortsNative, EntryPoint = "SystemIoPortsNative_TermiosReset", SetLastError = tru
internal static partial int TermiosReset(SafeSerialDeviceHandle handle, int speed, int data, StopBits stop,
[LibraryImport(Libraries.IOPortsNative, EntryPoint = "SystemIoPortsNative_TermiosGetSignal", SetLastError =
internal static partial int TermiosGetSignal(SafeSerialDeviceHandle handle, Signals signal);
[LibraryImport(Libraries.IOPortsNative, EntryPoint = "SystemIoPortsNative_TermiosSetSignal", SetLastError =
internal static partial int TermiosGetSignal(SafeSerialDeviceHandle handle, Signals signal, int set);
[LibraryImport(Libraries.IOPortsNative, EntryPoint = "SystemIoPortsNative_TermiosGetAllSignals")]
internal static partial Signals TermiosGetAllSignals(SafeSerialDeviceHandle handle);
[LibraryImport(Libraries.IOPortsNative, EntryPoint = "SystemIoPortsNative_TermiosSetSpeed", SetLastError =
internal static partial int TermiosSetSpeed(SafeSerialDeviceHandle handle, int speed);
[LibraryImport(Libraries.IOPortsNative, EntryPoint = "SystemIoPortsNative_TermiosGetSpeed", SetLastError =
internal static partial int TermiosGetSpeed(SafeSerialDeviceHandle handle);
[LibraryImport(Libraries.IOPortsNative, EntryPoint = "SystemIoPortsNative_TermiosAvailableBytes", SetLastEr

```
430
431 int32_t SystemIoPortsNative_TermiosReset(intptr_t handle, int32_t speed, int32_t dataBits, int32_t stopBits, int32_
432 {
433     int fd = ToFileDescriptor(handle);
434     struct termios term;
435     speed_t brate;
436     int ret = 0;
437
438     if (tcgetattr(fd, &term) < 0)
439     {
440         return  -1;
441     }
442
443 #if HAVE_CFMAKERAW
444     cfmakeraw(&term);
445 #else
446     term.c_iflag &= ~(IMAXBEL|IGNBRK|BRKINT|PARMRK|ISTRIP|INLCR|IGNCR|ICRNL|IXON);
447     term.c_oflag &= ~OPOST;
448     term.c_lflag &= ~(ECHO|ECHONL|ICANON|ISIG|IEXTEN);
449     term.c_cflag &= ~(CSIZE|PARENB);
450     term.c_cflag |= CS8;
451 #endif
```

Finding a bug

- *System.IO.Ports*
  .Native
  .pal_termios.c

- *term.c_iflag &= ...*
- *term.c_iflag = // (!)*
  IGNBRK

- Might overwrite bits
- Filed issue #108404

System.IO.Ports.Native termios c_iflag settings ignored? #108404

Edit New issue

Open **tamberg** opened this issue yesterday · 1 comment · May be fixed by #108409

tamberg commented yesterday • edited

Description

Hi, from reading the System.IO.Ports.Native code it seems that the termios c_iflag flags set in line [0] are overwritten by the assignment (!) in line [1], before tcsetattr() is called in [2]. This might lead to termios flags being ignored? Kind regards, Thomas

```
term.c_iflag &= ~(IMAXBEL|IGNBRK|BRKINT|PARMRK|ISTRIP|INLCR|IGNCR|ICRNL|IXON); // see [0]
...
term.c_iflag = IGNBRK; // see [1]
...
... tcsetattr(fd, TCSANOW, &term) ... // see [2]
```

[0] https://github.com/dotnet/runtime/blob/main/src/native/libs/System.IO.Ports.Native/pal_termios.c#L446
[1] https://github.com/dotnet/runtime/blob/main/src/native/libs/System.IO.Ports.Native/pal_termios.c#L455
[2] https://github.com/dotnet/runtime/blob/main/src/native/libs/System.IO.Ports.Native/pal_termios.c#L540

Assignees

No one assigned

Labels

area-System.IO.Ports in-pr untriaged

Projects

None yet

Milestone

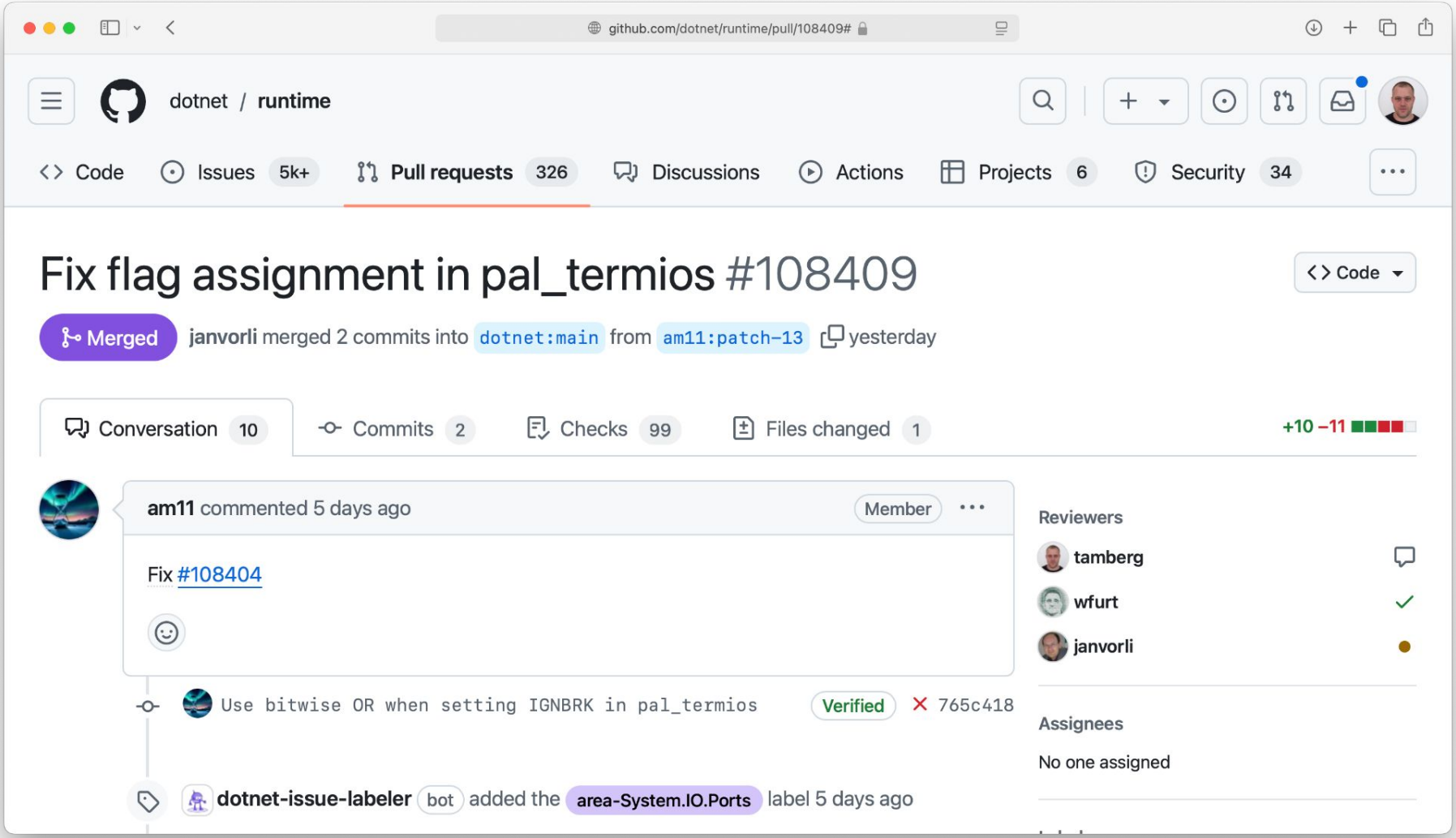
10.0.0

Development

Getting a fix

- *dotnet-issue-labeler*
- *dotnet-policy-service*
- *am11*'s pull request
- *build-analysis* rebuilt
- *jeffhandley* triaged
- *wfurt* approved
- *janvorli* merged

- Removed one line
- .NET Runtime
- 10.0.0, oh


github.com/dotnet/runtime/pull/108409#
dotnet / runtime
Code
Issues 5k+
Pull requests 326
Discussions
Actions
Projects 6
Security 34
Fix flag assignment in pal_termios #108409
Code
Merged
janvorli merged 2 commits into dotnet:main from am11:patch-13 yesterday
Conversation 10
Commits 2
Checks 99
Files changed 1
+10 −11
am11 commented 5 days ago
Member
Fix #108404
Use bitwise OR when setting IGNBRK in pal_termios
Verified
765c418
dotnet-issue-labeler bot added the area-System.IO.Ports label 5 days ago
Reviewers
tamberg
wfurt
janvorli
Assignees
No one assigned

Merged

Fix flag assignment in pal_termios #108409

janvorli merged 2 commits into `dotnet:main` from `am11:patch-13` yesterday

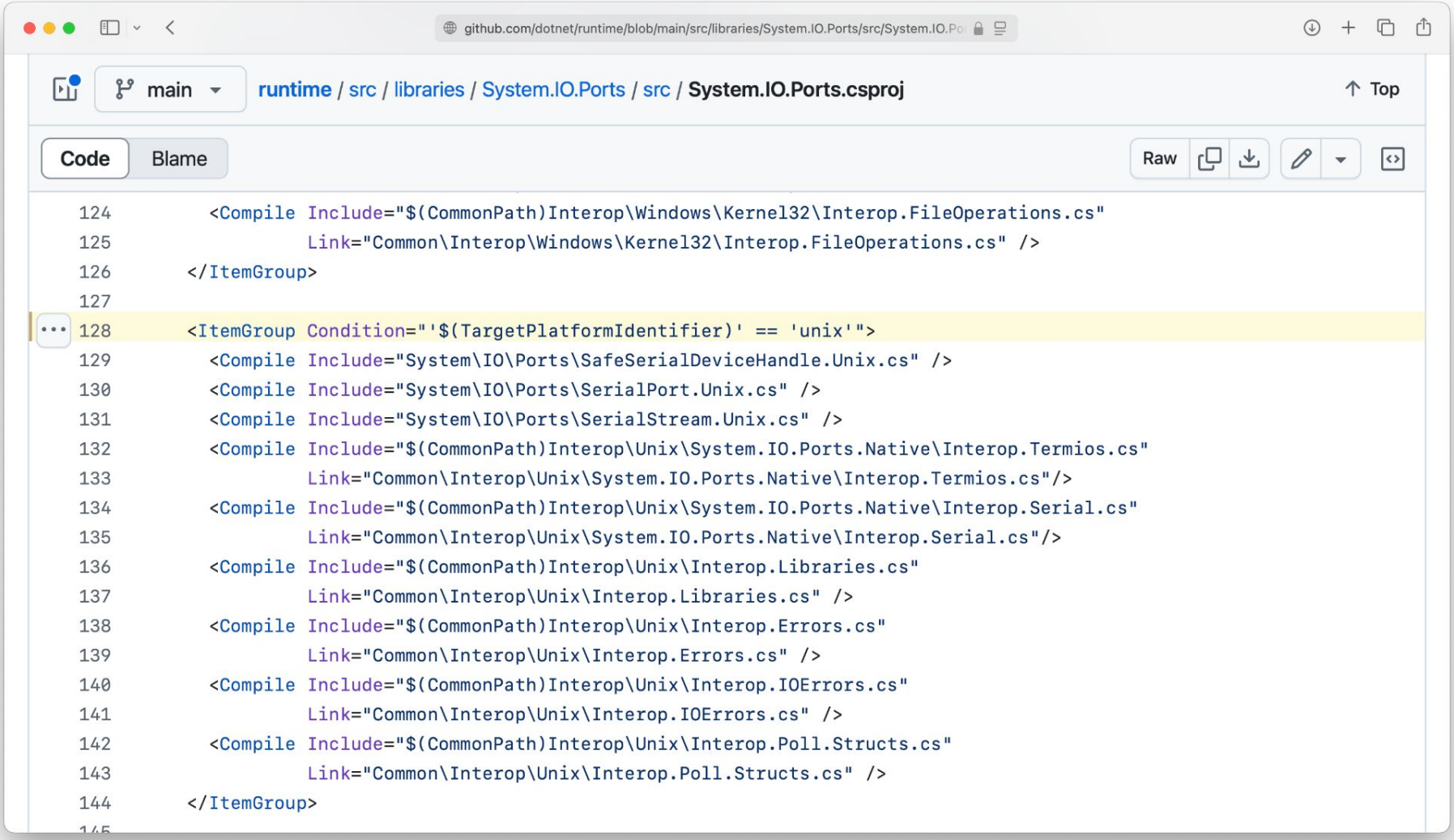
janvorli merged commit `a7f96cb` into `dotnet:main` yesterday

Hide details

97 of 99 checks passed

- ✓ runtime (Build Libraries Test Run checked coreclr osx x64 Deb... Details
- ✓ runtime (Build Libraries Test Run release coreclr linux x64 Deb... Details
- ✓ runtime (Build Libraries Test Run release coreclr linux_musl x6... Details
- ✓ runtime (Build Libraries Test Run release coreclr osx x64 Debu... Details
- ✓ runtime (Build Libraries Test Run release coreclr windows x64 ... Details
- ✓ runtime (Build Libraries Test Run release coreclr windows x86 ... Details
- ✓ runtime (Build android-arm Release AllSubsets_Mono) Build an... Details
- ✓ runtime (Build android-arm64 Release AllSubsets_Mono) Build... Details

am11 deleted the `patch-13` branch yesterday




```
124       <Compile Include="$(CommonPath)Interop\Windows\Kernel32\Interop.FileOperations.cs"
125                Link="Common\Interop\Windows\Kernel32\Interop.FileOperations.cs" />
126     </ItemGroup>
127
128     <ItemGroup Condition="'$(TargetPlatformIdentifier)' == 'unix'">
129       <Compile Include="System\IO\Ports\SafeSerialDeviceHandle.Unix.cs" />
130       <Compile Include="System\IO\Ports\SerialPort.Unix.cs" />
131       <Compile Include="System\IO\Ports\SerialStream.Unix.cs" />
132       <Compile Include="$(CommonPath)Interop\Unix\System.IO.Ports.Native\Interop.Termios.cs"
133                Link="Common\Interop\Unix\System.IO.Ports.Native\Interop.Termios.cs"/>
134       <Compile Include="$(CommonPath)Interop\Unix\System.IO.Ports.Native\Interop.Serial.cs"
135                Link="Common\Interop\Unix\System.IO.Ports.Native\Interop.Serial.cs"/>
136       <Compile Include="$(CommonPath)Interop\Unix\Interop.Libraries.cs"
137                Link="Common\Interop\Unix\Interop.Libraries.cs" />
138       <Compile Include="$(CommonPath)Interop\Unix\Interop.Errors.cs"
139                Link="Common\Interop\Unix\Interop.Errors.cs" />
140       <Compile Include="$(CommonPath)Interop\Unix\Interop.IOErrors.cs"
141                Link="Common\Interop\Unix\Interop.IOErrors.cs" />
142       <Compile Include="$(CommonPath)Interop\Unix\Interop.Poll.Structs.cs"
143                Link="Common\Interop\Unix\Interop.Poll.Structs.cs" />
144     </ItemGroup>
```

Build System

When building dotnet/runtime, we use the "TargetOS" property to control what target platform we are building for. The valid values for this property are windows (which is the default value from MSBuild when running on Windows), linux and osx.

Project Files

Whenever possible, a single .csproj should be used per assembly, spanning all target platforms, e.g. System.Console.csproj includes conditional entries for when targeting Windows vs when targeting Linux. A property can be passed to dotnet build to control which flavor is built, e.g. `dotnet build /p:TargetOS=osx System.Console.csproj`.

Constants

- Wherever possible, constants should be defined as "const". Only if the data type doesn't support this (e.g. IntPtr) should they instead be static readonly fields.
- Related constants should be grouped under a partial, static, internal type, e.g. for error codes they'd be grouped under an Errors type:

github.com/dotnet/runtime/blob/main/eng/build.sh#L307

main

runtime / eng / build.sh

Top

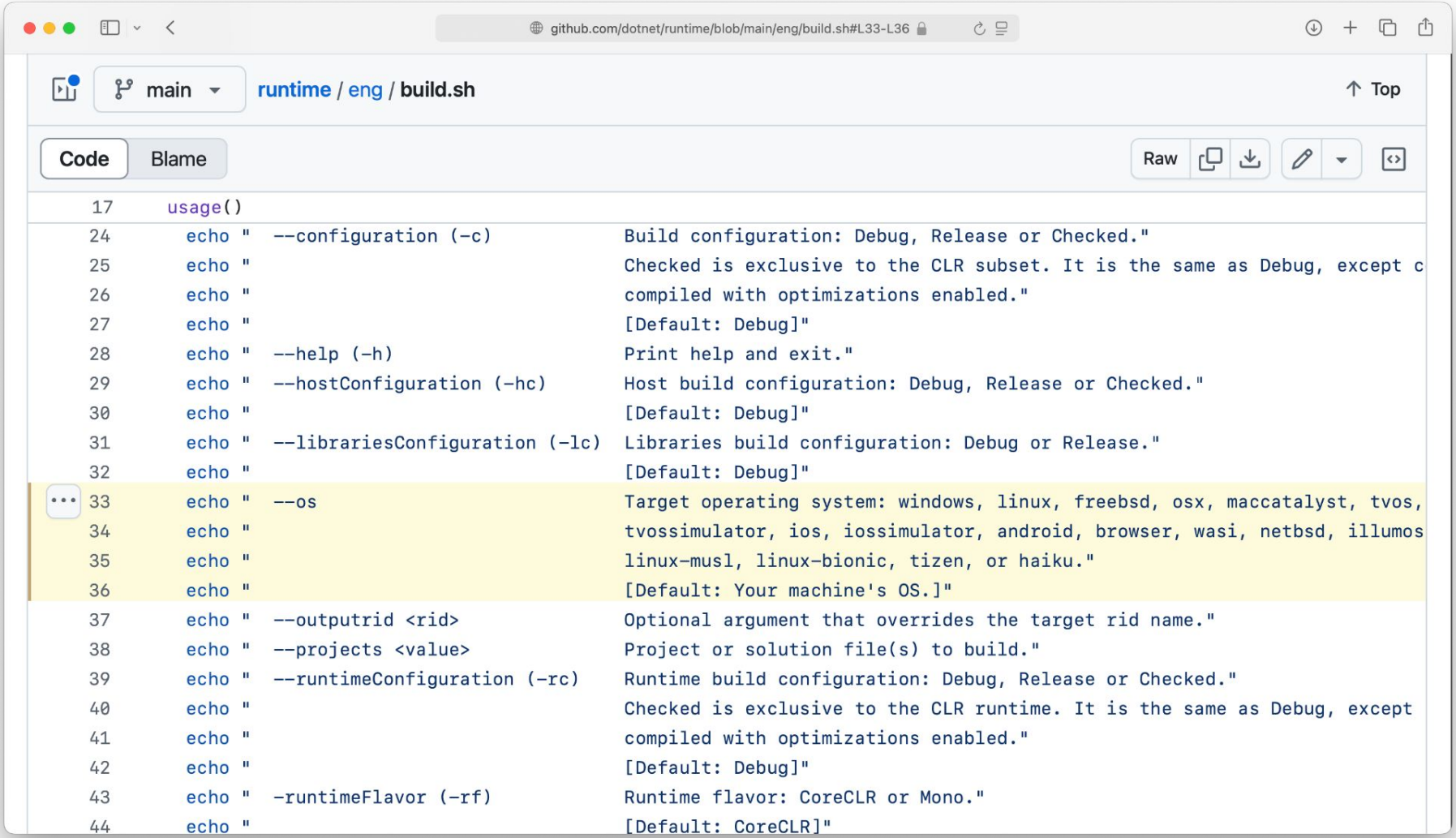
Code | Blame

Raw

```
297             __PortableTargetOS=linux-musl
298             ;;
299           haiku)
300             os="haiku" ;;
301           *)
302             echo "Unsupported target OS '$2'."
303             echo "Try 'build.sh --help' for values supported by '--os'."
304             exit 1
305             ;;
306         esac
307         arguments="$arguments /p:TargetOS=$os"
308         shift 2
309         ;;
310
311       -allconfigurations)
312         arguments="$arguments /p:BuildAllConfigurations=true"
313         shift 1
314         ;;
315
316       -testscope)
317         if [ -z ${2+x} ]; then
```

tamberg — -zsh — 80×24

```
tamberg@m3 ~ % grep -Rnw 'runtime' -e 'TargetOS='
```




```
17  usage()
24      echo "  --configuration (-c)           Build configuration: Debug, Release or Checked."
25      echo "                                 Checked is exclusive to the CLR subset. It is the same as Debug, except c
26      echo "                                 compiled with optimizations enabled."
27      echo "                                 [Default: Debug]"
28      echo "  --help (-h)                    Print help and exit."
29      echo "  --hostConfiguration (-hc)      Host build configuration: Debug, Release or Checked."
30      echo "                                 [Default: Debug]"
31      echo "  --librariesConfiguration (-lc) Libraries build configuration: Debug or Release."
32      echo "                                 [Default: Debug]"
33      echo "  --os                           Target operating system: windows, linux, freebsd, osx, maccatalyst, tvos,
34      echo "                                 tvossimulator, ios, iossimulator, android, browser, wasi, netbsd, illumos
35      echo "                                 linux-musl, linux-bionic, tizen, or haiku."
36      echo "                                 [Default: Your machine's OS.]"
37      echo "  --outputrid <rid>              Optional argument that overrides the target rid name."
38      echo "  --projects <value>             Project or solution file(s) to build."
39      echo "  --runtimeConfiguration (-rc)   Runtime build configuration: Debug, Release or Checked."
40      echo "                                 Checked is exclusive to the CLR runtime. It is the same as Debug, except
41      echo "                                 compiled with optimizations enabled."
42      echo "                                 [Default: Debug]"
43      echo "  -runtimeFlavor (-rf)           Runtime flavor: CoreCLR or Mono."
44      echo "                                 [Default: CoreCLR]"
```

Thank you

- ❤️ Programming
- ❤️ Open source
- ❤️ System calls

Not done, but happy.