

The GNU Debugger

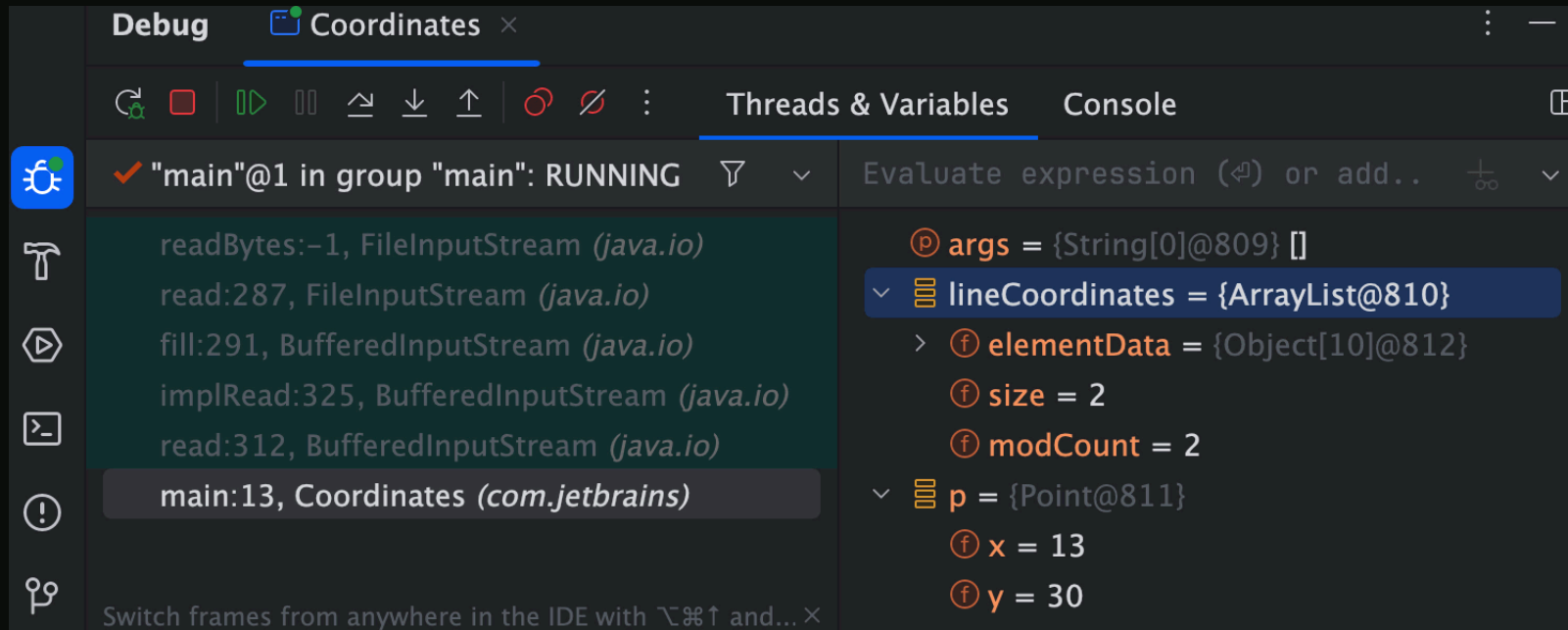
**GDB**

---

# > Ein Debugger na toll, kenn' ich schon

[1/6]

Sowas habt ihr bestimmt schon mal gesehen:



The screenshot shows the 'Debug' window of an IDE. The top bar indicates the current thread is '"main"@1 in group "main": RUNNING'. The console displays a stack trace with the following frames:

- readBytes:-1, FileInputStream (java.io)
- read:287, FileInputStream (java.io)
- fill:291, BufferedInputStream (java.io)
- implRead:325, BufferedInputStream (java.io)
- read:312, BufferedInputStream (java.io)
- main:13, Coordinates (com.jetbrains)** (highlighted)

The 'Threads & Variables' tab is active, showing the state of the current thread. The variables are:

- args = {String[0]@809} []
- lineCoordinates = {ArrayList@810} (expanded to show):
  - elementData = {Object[10]@812}
  - size = 2
  - modCount = 2
- p = {Point@811} (expanded to show):
  - x = 13
  - y = 30

Und wofür jetzt ein workshop?

GDB, ein debugger im Terminal.  
Warum würde man sich sowas antun?

```
(gdb) break 23
Breakpoint 3 at 0x555555552a2: file multi_thread.c, line 23.
(gdb) run
Starting program: /home/debx/Projects/multi_thread
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7ffff7bff6c0 (LWP 10334)]
Thread 1 here!
[New Thread 0x7ffff73fe6c0 (LWP 10335)]
Thread 2 here!
[New Thread 0x7ffff6bfd6c0 (LWP 10336)]
Thread 3 here!
[New Thread 0x7ffff63fc6c0 (LWP 10337)]
Thread 4 here!

Thread 1 "multi_thread" hit Breakpoint 3, main () at multi_thread.c:23
23      sleep(5); // Let the threads run for 5 seconds
(gdb) info threads
Id      Target Id      Frame
* 1      Thread 0x7ffff7fa6740 (LWP 10333) "multi_thread" main () at multi_thread.c:23
  2      Thread 0x7ffff7bff6c0 (LWP 10334) "multi_thread" 0x0007ffff7cdbc11 in __GI___clock_nanosleep (
clock_id=clock_id@entry=0, flags=flags@entry=0, req=req@entry=0x7ffff7bfee80, rem=rem@entry=0x0)
at ../sysdeps/unix/sysv/linux/clock_nanosleep.c:78
  3      Thread 0x7ffff73fe6c0 (LWP 10335) "multi_thread" 0x0007ffff7cdbc11 in __GI___clock_nanosleep (
clock_id=clock_id@entry=0, flags=flags@entry=0, req=req@entry=0x7ffff73fde80, rem=rem@entry=0x0)
at ../sysdeps/unix/sysv/linux/clock_nanosleep.c:78
  4      Thread 0x7ffff6bfd6c0 (LWP 10336) "multi_thread" 0x0007ffff7cdbc11 in __GI___clock_nanosleep (
clock_id=clock_id@entry=0, flags=flags@entry=0, req=req@entry=0x7ffff6bfce80, rem=rem@entry=0x0)
at ../sysdeps/unix/sysv/linux/clock_nanosleep.c:78
  5      Thread 0x7ffff63fc6c0 (LWP 10337) "multi_thread" 0x0007ffff7cdbc11 in __GI___clock_nanosleep (
clock_id=clock_id@entry=0, flags=flags@entry=0, req=req@entry=0x7ffff63fbe80, rem=rem@entry=0x0)
at ../sysdeps/unix/sysv/linux/clock_nanosleep.c:78
(gdb) █
```

*“Schau mal hier ist ein Binary. Das fragt nach nem Passwort und nur wenn du das richtige Passwort eingibst, bekommst du die flag.”*

*“Schau mal hier ist ein Binary. Das fragt nach nem Passwort und nur wenn du das richtige Passwort eingibst, bekommst du die flag.”*

Nichts leichter als das!



`strings` command

Gibt alle Strings in einer Datei aus.

`strings` command

Gibt alle Strings in einer Datei aus.

Demo `time (Bypass Me)`

## > Die Lösung: Debugger!

---

[5/6]

Einfach mit dem debugger durchsteppen. Zur Laufzeit sollte irgendwann das Passwort im Speicher liegen!

## > Die Lösung: Debugger!

---

[5/6]

Einfach mit dem debugger durchsteppen. Zur Laufzeit sollte irgendwann das Passwort im Speicher liegen!

Aber moment mal...

## > Die Lösung: Debugger!

[5/6]

Einfach mit dem debugger durchsteppen. Zur Laufzeit sollte irgendwann das Passwort im Speicher liegen!

Aber moment mal...

wir haben ja gar keinen source code!



Mit GDB geht das!

Mit GDB geht das!

Schauen wir es uns an!