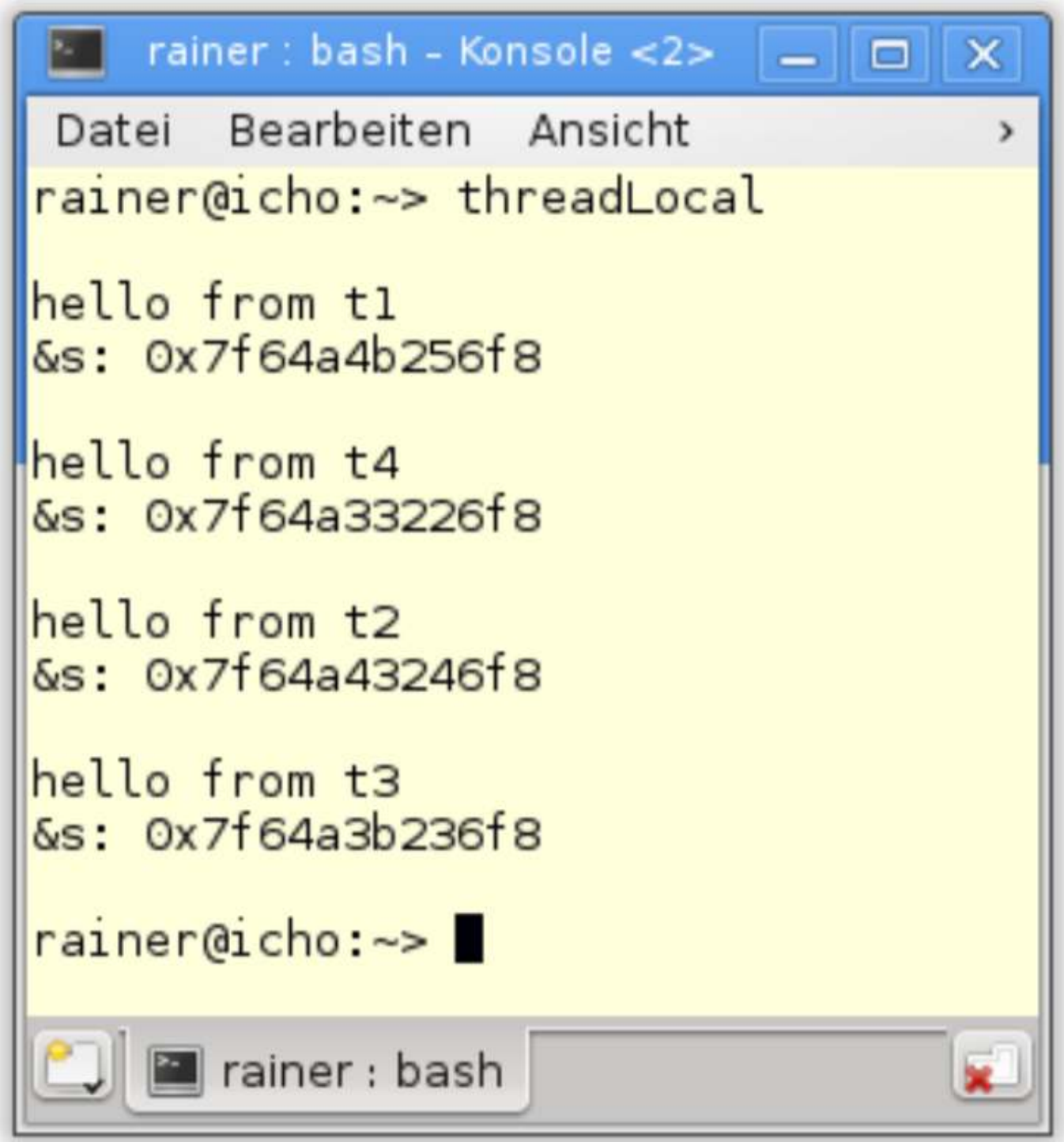
Thread Local Data

In this lesson, we will learn about data that is bound to the lifecycle of a thread.

By using the keyword `thread_local`, you have thread local data also known as **thread local storage**. Each thread has its copy of the data. Thread-local data behaves like static variables. They are created at their first usage, and their lifetime is bound to the lifetime of the thread.

```
...
std::mutex coutMutex;
thread_local std::string s("hello from ");
void addThreadLocal(std::string const& s2){ s+= s2;
  std::lock_guard<std::mutex> guard(coutMutex);
  std::cout << s << std::endl;
  std::cout << "&s: " << &s << std::endl;
  std::cout << std::endl;
}
std::thread t1(addThreadLocal, "t1"); std::thread t2(addThreadLocal, "t2"); std::thread t3(ad
```

Each thread has its copy of the `thread_local` string. Therefore, each string modifies its string independently, and each string has its unique address:


rainer : bash - Konsole <2>
Datei Bearbeiten Ansicht
rainer@icho:~> threadLocal
hello from t1
&s: 0x7f64a4b256f8
hello from t4
&s: 0x7f64a33226f8
hello from t2
&s: 0x7f64a43246f8
hello from t3
&s: 0x7f64a3b236f8
rainer@icho:~>
rainer : bash