

Userland threads

Stephane Sezer

Threads?

Why userland  
threads?

Quick and dirty  
implementation

Questions?

# Userland threads

Stephane Sezer

July 7, 2011

- 1 Threads?
  - What defines a thread?
  - Threading model

Userland threads

Stephane Sezer

Threads?

What defines a thread?

Threading model

Why userland threads?

Quick and dirty implementation

Questions?

- 1 Threads?
  - What defines a thread?
  - Threading model

Userland threads

Stephane Sezer

Threads?

What defines a thread?

Threading model

Why userland threads?

Quick and dirty implementation

Questions?

# What defines a thread?

Userland threads

Stephane Sezer

Threads?

What defines a thread?

Threading model

Why userland threads?

Quick and dirty implementation

Questions?

Separate program counter flow of execution

Separate registers internal state

Separate stack some isolated data

Shared memory some shared data

# What defines a thread?

Userland threads

Stephane Sezer

Threads?

What defines a thread?

Threading model

Why userland threads?

Quick and dirty implementation

Questions?

Separate program counter flow of execution

Separate registers internal state

Separate stack some isolated data

Shared memory some shared data

# What defines a thread?

Separate program counter flow of execution

Separate registers internal state

Separate stack some isolated data

Shared memory some shared data

Userland threads

Stephane Sezer

Threads?

What defines a thread?

Threading model

Why userland threads?

Quick and dirty implementation

Questions?

# What defines a thread?

Userland threads

Stephane Sezer

Threads?

What defines a thread?

Threading model

Why userland threads?

Quick and dirty implementation

Questions?

Separate program counter flow of execution

Separate registers internal state

Separate stack some isolated data

Shared memory some shared data

- 1 Threads?
  - What defines a thread?
  - Threading model

Userland threads

Stephane Sezer

Threads?

What defines a thread?

Threading model

Why userland threads?

Quick and dirty implementation

Questions?



- 1:1 kernel threads
- 1:N userland threads
- N:M hybrid model

Userland threads

Stephane Sezer

Threads?

What defines a thread?

Threading model

Why userland threads?

Quick and dirty implementation

Questions?

## 2 Why userland threads?

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Questions?

# Why userland threads?

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Questions?

- Systems without kernel threads
- Lightweight threads for some virtual machines
- Because we can §

# Why userland threads?

Userland threads

Stephane Sezer

Threads?

Why userland  
threads?

Quick and dirty  
implementation

Questions?

- Systems without kernel threads
- Lightweight threads for some virtual machines
- Because we can §

# Why userland threads?

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Questions?

- Systems without kernel threads
- Lightweight threads for some virtual machines
- Because we can §

## 3 Quick and dirty implementation

- Context saving
- Preemption
- Stack allocation

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?

## 3 Quick and dirty implementation

- Context saving
- Preemption
- Stack allocation

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?

- Standard of the C library
- Allows “non-local” returns
- Can be used to implement a basic try {} catch {} system in C

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?



- Standard of the C library
- Allows “non-local” returns
- Can be used to implement a basic try {} catch {} system in C

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?

- Standard of the C library
- Allows “non-local” returns
- Can be used to implement a basic `try {} catch {}` system in C

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?

```
ENTRY(_setjmp)
    movl    4(%esp),%eax
    movl    0(%esp),%edx
    movl    %edx, 0(%eax)
    movl    %ebx, 4(%eax)
    movl    %esp, 8(%eax)
    movl    %ebp, 12(%eax)
    movl    %esi, 16(%eax)
    movl    %edi, 20(%eax)
    xorl    %eax,%eax
    ret
```

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?

```
ENTRY(_longjmp)
    movl    4(%esp),%edx
    movl    8(%esp),%eax
    movl    0(%edx),%ecx
    movl    4(%edx),%ebx
    movl    8(%edx),%esp
    movl    12(%edx),%ebp
    movl    16(%edx),%esi
    movl    20(%edx),%edi
    testl   %eax,%eax
    jnz     1f
    incl   %eax
1:    movl   %ecx,0(%esp)
    ret
```

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?

## 3 Quick and dirty implementation

- Context saving
- **Preemption**
- Stack allocation

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

**Preemption**

Stack allocation

Questions?

- `void signal(2)`
- `setitimer(2)`

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

**Preemption**

Stack allocation

Questions?

- `void signal(2)`
- `setitimer(2)`

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

**Preemption**

Stack allocation

Questions?

- When a signal is received by a process, it is queued in the kernel's internal data structures related to this process
- The next time the process is scheduled, the signal handler is called in a special context: a `sigreturn(2)` stub is written on the stack

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?



- When a signal is received by a process, it is queued in the kernel's internal data structures related to this process
- The next time the process is scheduled, the signal handler is called in a special context: a `sigreturn(2)` stub is written on the stack

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?

## 3 Quick and dirty implementation

- Context saving
- Preemption
- Stack allocation

Userland threads

Stephane Sezer

Threads?

Why userland threads?

Quick and dirty implementation

Context saving

Preemption

Stack allocation

Questions?

Here, we just need a simple `mmap(2)`

Userland threads

Stephane Sezer

Threads?

Why userland  
threads?

Quick and dirty  
implementation

Context saving

Preemption

Stack allocation

Questions?

