barkhausen institut

Microkernel Construction

Exercise 4: IPC

Nils Asmussen

2026-06-04

Roadmap

- Create and map UTCBs
- Create portals
- Call portals
- Reply from portals

- Hands-on
  - Core of `call` syscall
  - Core of `reply` syscall
  - IPC between two threads in userspace

Get the Code

```shell
$ git clone https://github.com/Nils-TUD/MKC
$ git checkout exercise4
```

```shell
# build it
$ make
```

```shell
# run it
$ make run
```

UTCB Allocation

- Kernel: `kern/{include,src}/ec.{h,cc}`
  - EC allocates UTCB on creation
  - Kernel maps it to user-defined virtual address
  - UTCB address is used as an identifier for threads (until we have capabilities)
  - Main thread does **not** have a UTCB
- User: `user/src/user.cc`
  - Creates two ECs for sender and receiver
  - Use `UTCB_{SENDER,RECEIVER}`

Portals

- Kernel: `kern/{include,src}/pt.{h,cc}`
  - Properties:
    1. `rip`: instruction pointer
    2. `id`: identifier (until we have capabilities)
    3. `recv`: receiving EC
  - `create_pt` creates new portal object
    - UTCB address specifies receiving EC
    - Portals are managed in singly-linked list
- User: `user/src/user.cc`
  - `create_pt` binding
  - Dummy `portal` function

IPC Basics

- ECs have a state
  1. `READY`: can run or is running
  2. `WAITING`: waiting on portal call
  3. `BLOCKED`: blocked in portal call
- ECs with `rip = 0` are considered local threads
  - Initially blocked instead of ready
- Simple scheduler
  - Picks the next ready EC
- Skeletons of:
  1. `sys_call`
  2. `sys_reply`

Task 1: Implement `call` System Call

- User part:
  - Create portal via `sys_create_pt`
  - Implement `sender`
    1. Put something in UTCB (e.g., 2 integers)
    2. Perform `sys_call`
    3. You can use `sys_dump` to print 2 values
- Kernel part:
  - `sys_call` gets portal id
  - Check if the receiving thread is waiting
  - Prepare resume of current thread
  - Prepare resume of receiving thread
  - What if receiving thread is not waiting?

Task 2: Implement `reply` System call

- User part:
  - Perform some operation with data in UTCB
  - Call `sys_reply`
- Kernel part:
  - Put local thread (`current`) back to `WAITING`
  - Copy message
  - Switch back to caller

- More to play with:
  1. What if the receiver is not ready?
  2. Create multiple senders
  3. How can the receiver distinguish senders?