TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute for System Architecture, Operating Systems Group

Exercise 2: IPC

Inter-Process Communication

Recap

- Inter-Process Communication
  - Send
  - Wait (open/closed)
  - Call: Send + closed Wait
  - Reply and Wait
- Message Payload
  - Plain data : copy
  - Capabilities (memory, kernel objects) : map
- Sync / Async
  - Sync: Rendezvous, direct copy sender → receiver
  - Async: Queues, buffers, fire and forget

The Plan

- Echo server a.k.a. Log server
  - Send a string to a local echo server
  - Return the message to the caller
- Capability Delegation
  - Create your own echo server (Endpoint + Thread) and send a capability for this server to another thread
  - Use your new echo server for logging
- Client-Server
  - Connection and Session management, server registers a service, client looks it up

TECHNISCHE UNIVERSITÄT DRESDEN

Setup

- Download the source archive from `http://os.inf.tu-dresden.de/~nils/nova-nre.tar.gz`
- `tar -xfz nova-nre.tar.gz && cd cross && ./download.sh`
- Choose host platform (Ubuntu 12.04 x86_64)

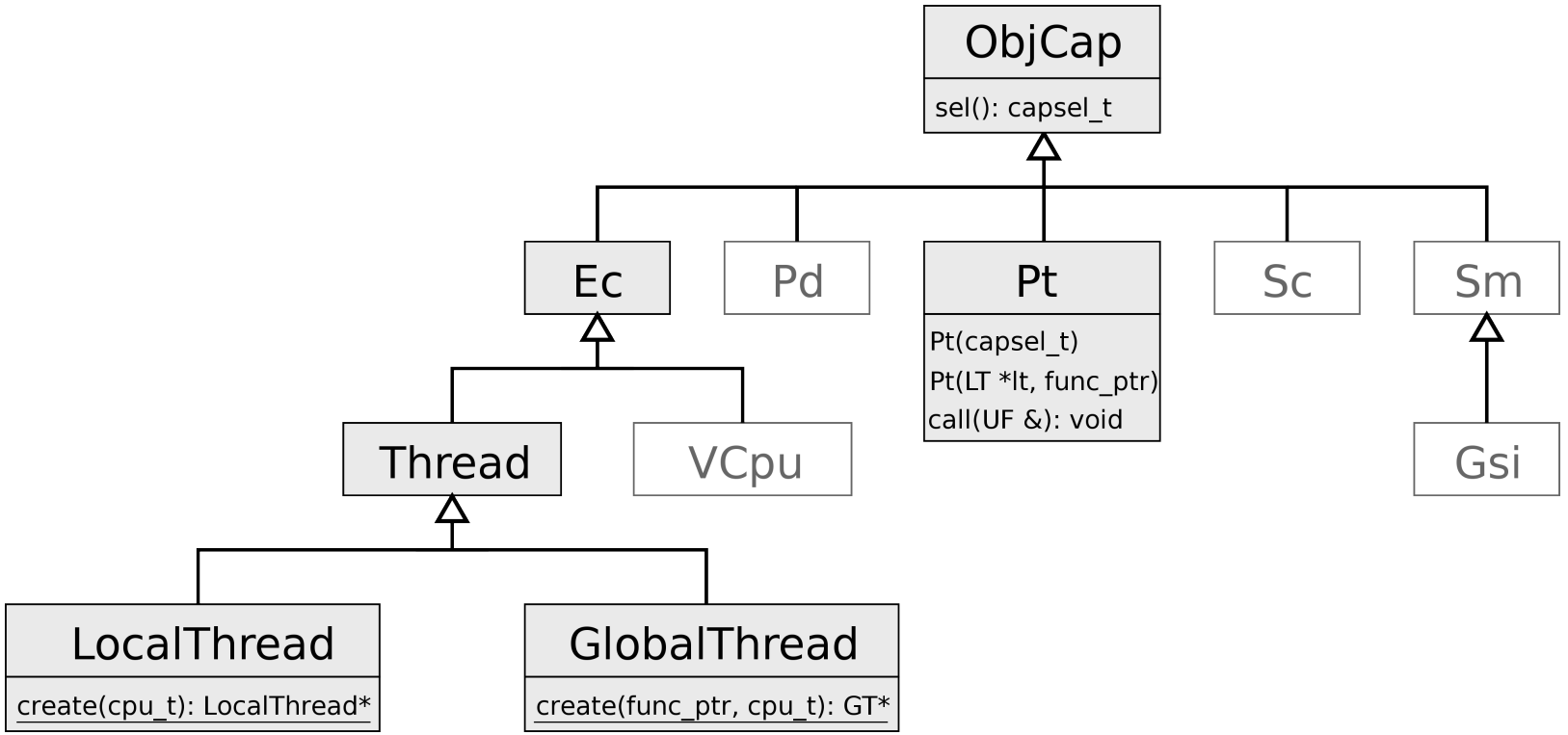
- Build via: ./b
- Run via: ./b qemu boot/echo

- <DIR>/nre/apps : echo, delegate, client-server
- <DIR>/nre/include : headers, e.g. UtcbFrame.h

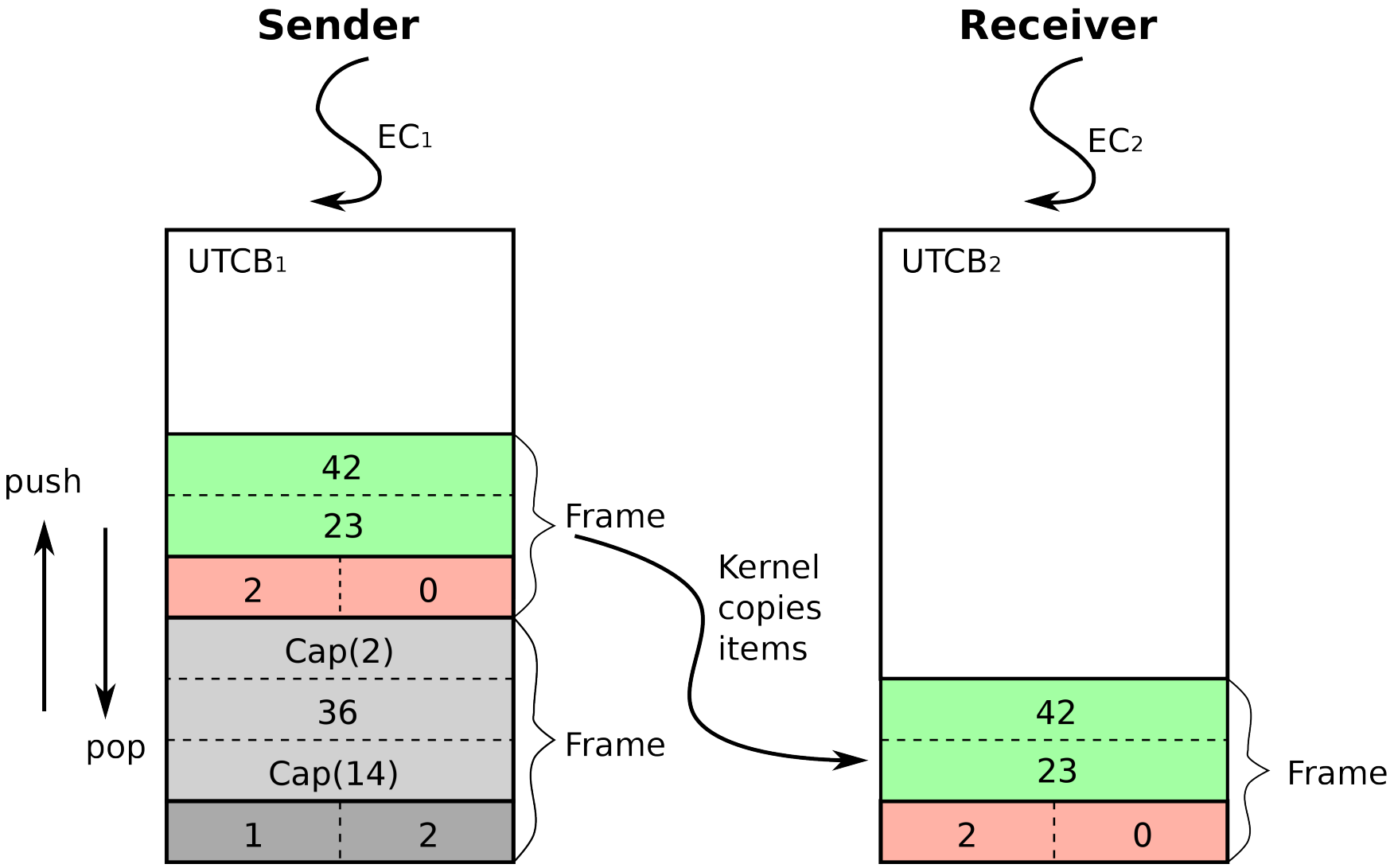
NOVA Abstractions

- Thread / Execution Context (EC)
  - Unit of execution, thread or virtual machine
- Portal (PT)
  - Communication endpoint
  - Messages (data, capabilities) are sent to portals, threads wait there to receive them
- Local Threads (services)
  - Wait/block in portals until get invoked/called
  - Return to the portal after request completion
- Global Threads
  - Run on their own

NRE : Capability Hierarchy



NOVA IPC : UTCB Frames - Concept



NOVA IPC : UTCB Frames - Usage

- Message Buffer, holds data OR capabilities
- Data accessed via stream operators:
  - read: **`UtcbFrame >> value;`**
  - write: **`UtcbFrame << result;`**
- Capabilities accessed via:
  - send: **`UtcbFrame.delegate(cap);`**
  - receive: **`UtcbFrame.delegation_window(Crd);`**
- **`UtcbFrame.finish_input()`**
  - Required after reading all arguments and before writing any result (same buffer)
- **`UTCBFrame`** constructor pushs a new frame on the UTCB, the destructor pops it again

Service “Loop”

```
for (; receive (&utcb); reply(utcb)) {
    process_message (utcb);
    generate_reply (&utcb);
}
```

- Receive: waits for incoming message, which is copied into receiver's UTCB
- Handle message and generate reply
- Reply: copy reply into caller's UTCB and await next message

1. Echo server (apps/echo)

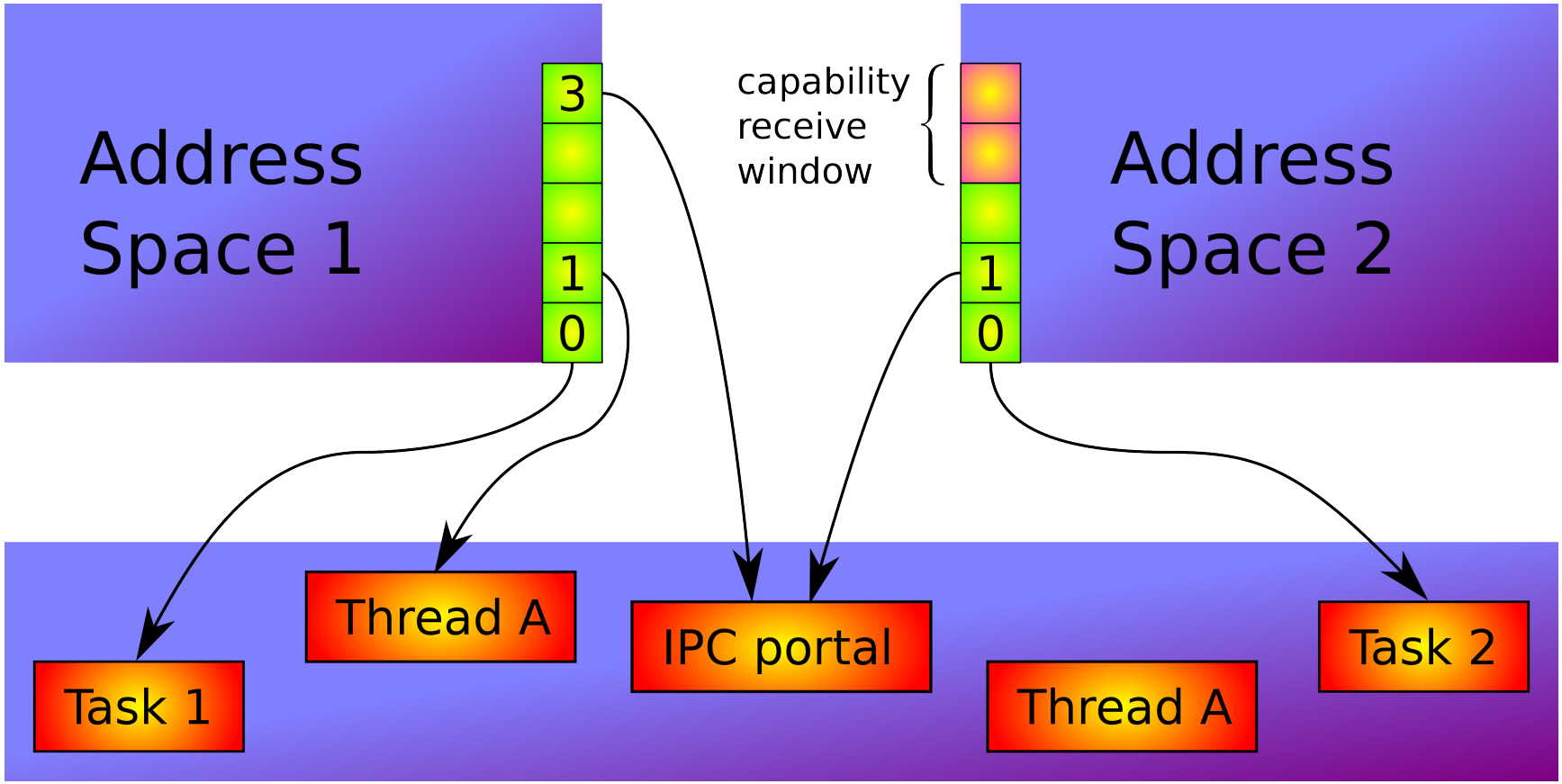
```
// create a new thread
LocalThread *lt = LocalThread::create
  (CPU::current().log_id());
// create a new portal for this thread
Pt echo(lt, portal_echo)

UtcbFrame uf;           // new UTCB frame
uf << 42;               // fill in argument
echo.call(uf);          // call server
int res;
uf >> res;              // read result
```

1. Echo server (apps/echo)

- Implement service functionality
- Read argument, send it back to the caller
- Improvement: send two numbers a and b, send back a+b / a-b / a*b / a/b ...

Capability Space



Capability Range Descriptor (Crd)

- Denote caps similar to memory pages, see flex pages (power-of-two sized, size-aligned)

- Offset and size in cap space, e.g.
  - Offset = 0, size = 1 → cap index 0
  - Offset = 0, size = 2 → cap index 0-1
  - Offset = 8, size = 4 → cap index 8-11
  - Offset = 16, size = 32 → illegal, misaligned

2. Delegation (apps/delegate)

```
// allocate/reserve num cap space indices
capsel_t CapSelSpace::get().allocate(num);

// open receive window for new caps
// see include/Desc.h for Crd details
Utcb.delegation_window (Crd crd);

// invoke portal, thereby receiving cap(s)
pt.call(utcb);
```

In A Nutshell

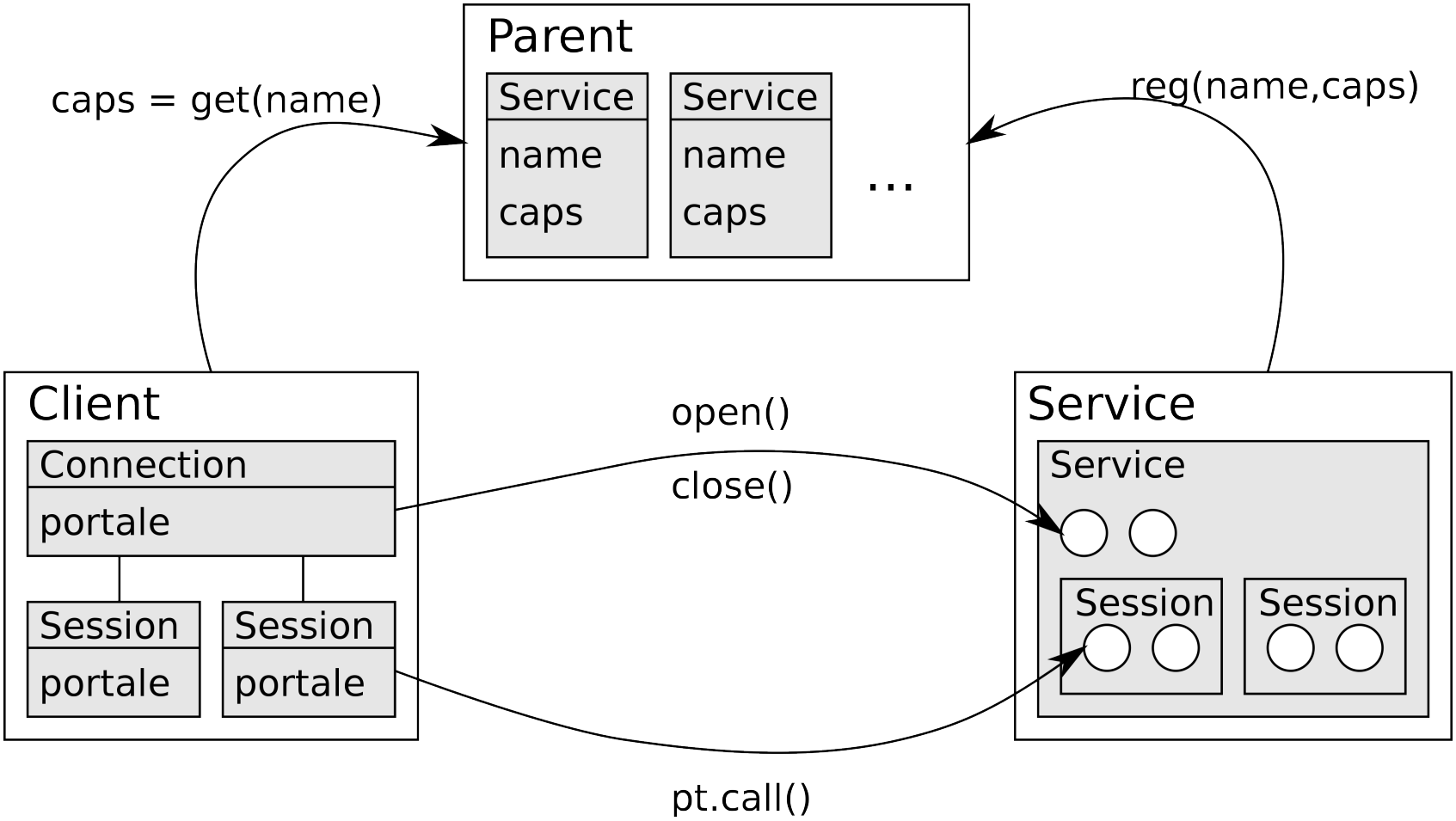
```
// create new local thread
LocalThread *t = LocalThread::create(cpuid);

// allocate/reserve num cap space indices
capsel_t CapSelSpace::get().allocate(num);

utcb >> value;       // read value from UTCB
utbc << value;       // write value into UTCB
Utcb.finish_input();       // prepare writing
// cap delegation: open receive window
Utcb.delegation_window (Crd crd);
// cap delegation: add cap to UTCB for sending
Utcb.delegate (capsel_t sel);

Pt::Pt (LocalThread*, func_ptr);
Pt::Pt (capsel_t sel);
pt.call(utcb);
```

Session Management



```
// SERVER
static void echo_func (capsel_t) {
   UtcbFrame uf;
   // … service implementation
}
int main() {
   Service *s = new Service ("echo", CPUSet(CPUSet::ALL), echo_func);
   s->start();
}

// CLIENT
int main()
{
   Connection con("echo");
   PtClientSession sess(con);
   UtcbFrame uf;
   sess.pt().call(uf);
}
```