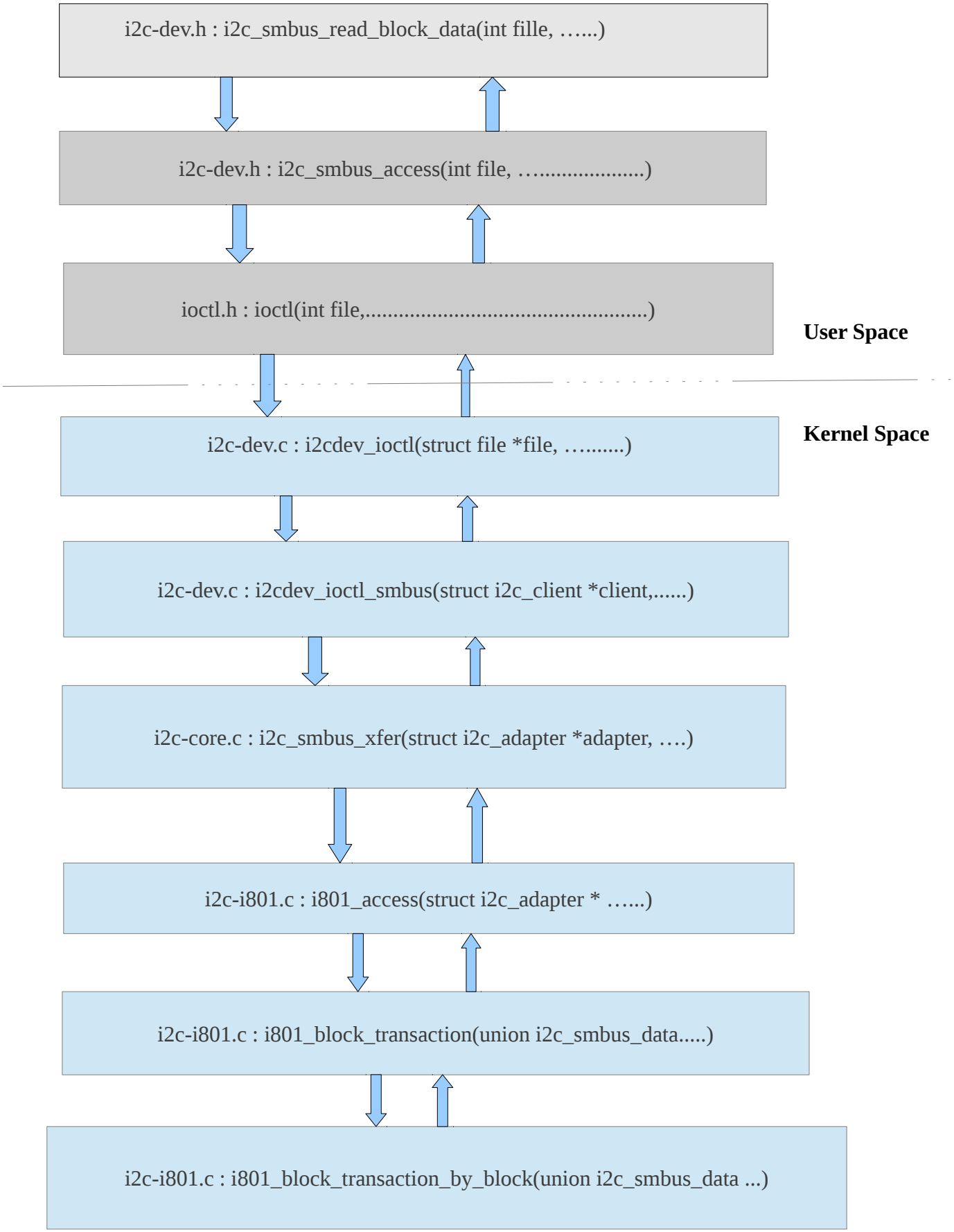
Call sequence of *i2c_smbus_read_block_data*



i2c_smbus_read_block_data() : This function is defined in i2c-dev.h, which is a user interface for i2c transactions. This method in turn calls the ioctl() system call after parsing the data provided as arguments by the user application.

i2c_smbus_access() : This function is defined in i2c-dev.h, this method is a common routine for all i2c smbus calls. It parses all the data passed by the user space and invokes ioctl system call.

Ioctl() : This is a system call which switches the context of the user application to kernel space, It modifies the permission of the user application.

i2cdev_ioctl() : This method is a file operation defined by the i2c-dev module, All ioctl() calls made from the user space on to the i2c adapters will be mapped to this method. This method parses the data passed from the user space and routes the call to other methods based on the parameters passed by the user.

i2cdev_ioctl_smbus() : All i2c transactions from the user space with a set I2C_SMBUS flag will be routed to this method. After validating the data and encapsulating the data the call is routed to *i2c_smbus_xfer()***.**

i2c_smbus_xfer() : This method is defined by i2c-core module, it is an interface to perform read and write transactions onto the i2c-bus. This method is very important because this method decides as to which adapter the call has to be routed to and what algorithm that particular adapter would use to transfer data.

i801_access() : The i2c - SMBus adapter declares i801_access as its transfer algorithm. Depending upon the input parameters, the call is routed to specific functions defined in the i801 driver. In case of block data transfer, the call is routed to ***i****801_block_transaction()* .

i801_block_transaction() : Depending upon the data passed by i801_access(), the configuration register is set, Block size to be read is decided (32 bytes max block size) and in case of SMBus transfer i801_block_transaction_by_block() is invoked.

Creating i2c_dev in i2c-dev module

- The i2c_dev structure represents the i2c device to the which the driver which defines it corresponds to.

- The i2c_dev structure has 3 members, which can be shown below:

```
struct i2c_dev
{
        struct list_head list;
        struct i2c_adapter *adap;
        struct device *dev;
};
```

- This structure is created and initialized as part of i2c_attach_adapter() routine.

- The i2c_attach_adapter() routine is registered as a callback routine with the i2c-core through the i2c_for_each_dev() method, during module initialization.

- The i2c_attach_adapter() routine is called for every device present on a particular bus type

- The bus.c module has a list of all devices present on a particular bus type, this module iterates over the list and invokes the callback functions registered by each client.

```
while ((dev = next_device(&i)) && !error)
        error = fn(dev, data);
klist_iter_exit(&i);
return error;
```

fn(dev, data) is the call back routine which is i2c_attach_adapter();

Waiting mechanism used in i2c-i801 driver

- Since i2c bus runs with a much slower clock rate it is mandatory that a call to i2c_i801 transfer functions must wait for the completion response from a bus transfer.

- The i801 adapter driver introduces a waiting mechanism using the methods *i801_wait_intr()* method.

- ***i801_wait_intr():*** waits till the BUSY flag is cleared or INTR flag or error flag is set. The status of these flags are checked intermittantly after a sleep of a fraction of a second till MAX_TIMEOUT.

- Here is a small code snippet from ***i801_wait_intr()*** which does the wait**.**

```
do {
        usleep range(250, 500);
        status = inb_p(SMBHSTSTS(priv));
} while (((status & SMBHSTSTS_HOST_BUSY) ||
        !(status & (STATUS_ERROR_FLAGS | SMBHSTSTS_INTR))) &&
        (timeout++ < MAX_RETRIES));
```