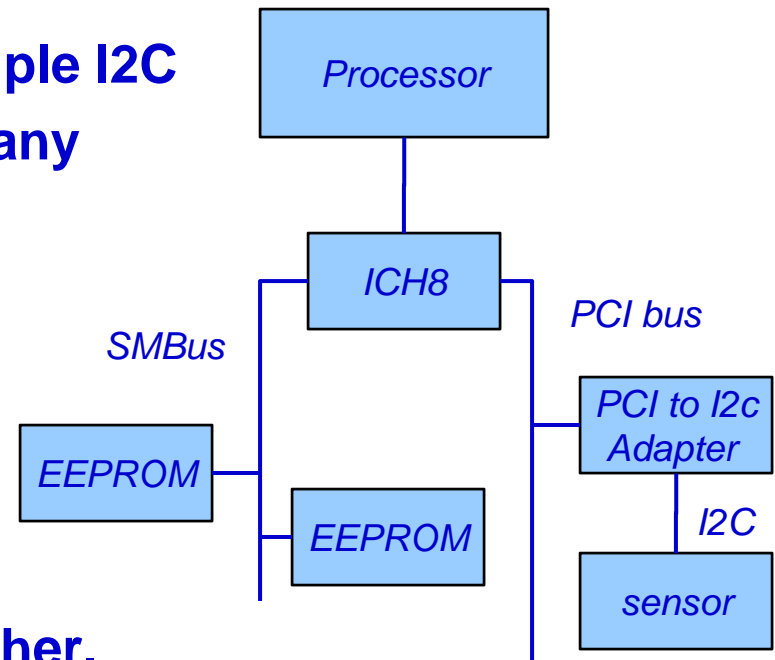# I2C Driver in Linux

Computer Science & Engineering Department
Arizona State University
Tempe, AZ 85287

Dr. Yann-Hang Lee
*yhlee@asu.edu*
(480) 727-7507

I2C and SMBus in x86

❑ **In general, a system can have multiple I2C buses via different adapters and many I2C devices**

❑ **2-wire synchronous serial buses**

   ❖ Master and slave, addressable

❑ **SMBus module in ICH8**

   ❖ *0000:00:1f.3* [0x400-0x41f]

❑ **I2C bus and the SMBus are essentially compatible with each other.**

❑ **Differences:**

   ❖ Timeout (in SMBus, reset interfaces when clock is low forlonger than 35ms))

   ❖ Maximum clock speed: 100MHz(Smbus) but I2C bus has both 400kHz and 3.4MHz versions.

   ❖ Logic level: 1: 3V in I2C and 2.1V in SMBus

   ❖ General call and alert response
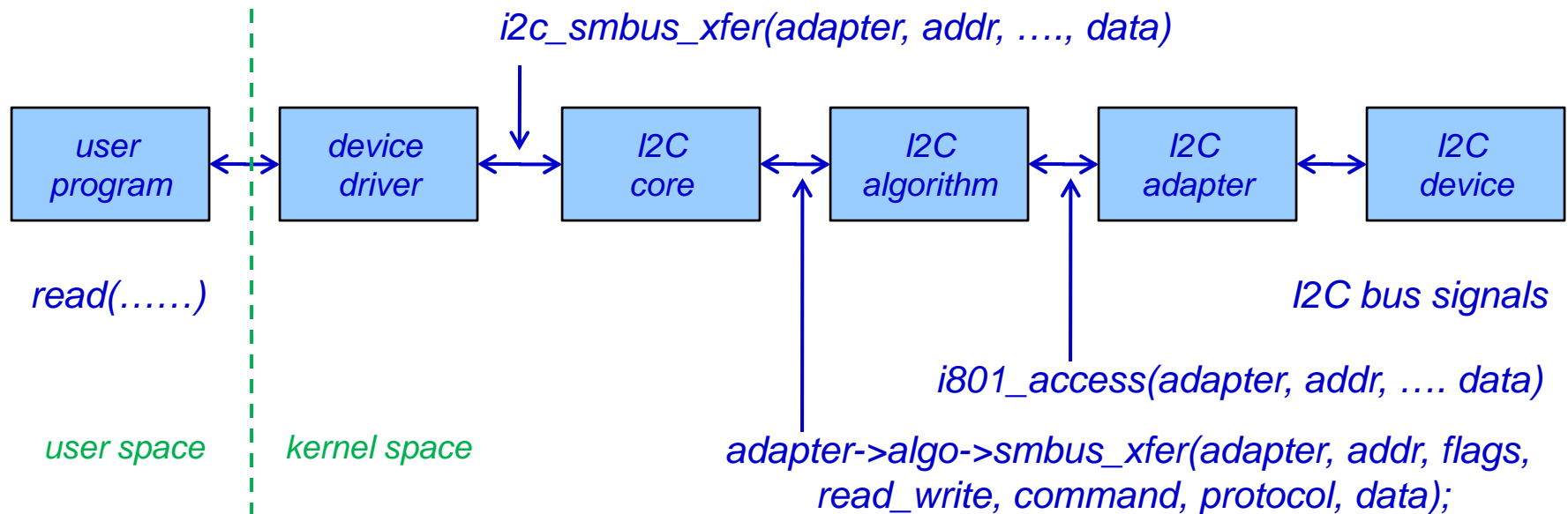
I2C Drivers in Linux

- ❑ **A driver for I2C bus**
    - ❖ adapter and algorithm drivers
    - ❖ manages I2C bus transactions
- ❑ **Drivers for I2C devices**
- ❑ **A client has the device's I2C bus address and a pointer to a driver which is attached with an adapter**
- ❑ **When a user program issues a file operation that needs an I2C transaction**
    - ❖ i2C_transfer (i2C-core.c) to invoke adap_algo_master_xfer
    - ❖ command or data is in an msg array
    - ❖ the adapter issues reads/writes to hardware I/O addresses.
- ❑ **Other operations (except I2C bus transaction) are handler by the drivers**
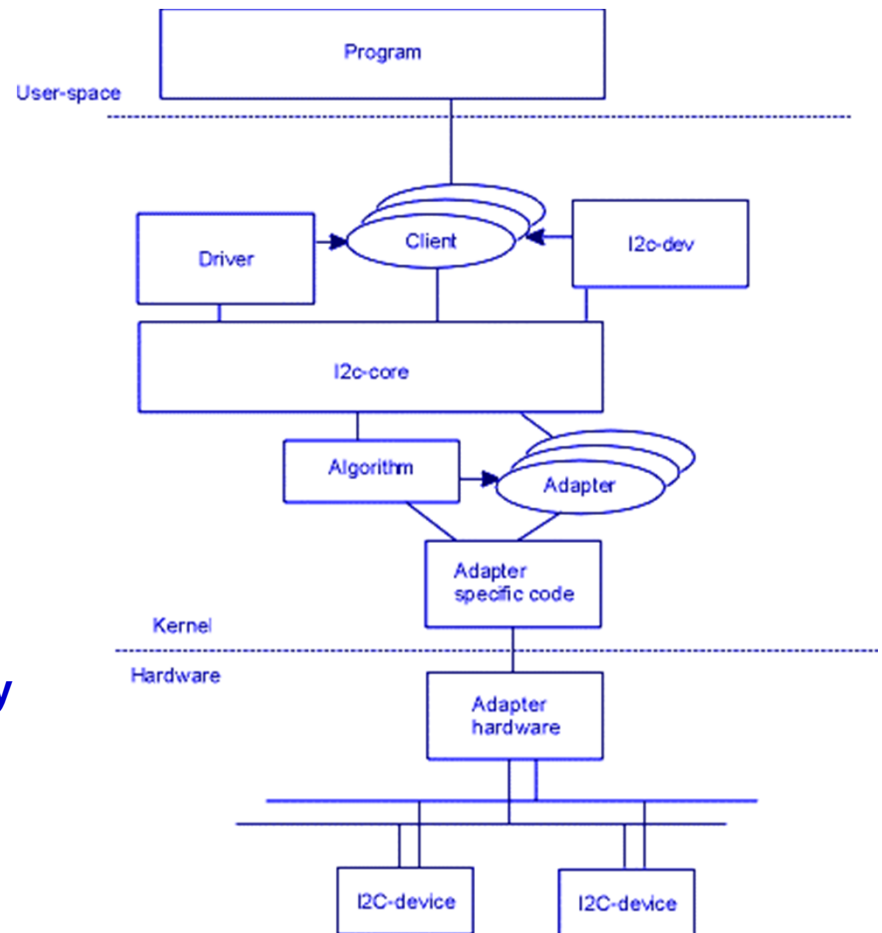
(https://i2c.wiki.kernel.org/index.php/Driver_Architecture)

Example of Accessing I2C/SMBus Devices

i2c_smbus_xfer(adapter, addr, …., data)

user program	device driver	I2C core	I2C algorithm	I2C adapter	I2C device

read(……)

I2C bus signals

i801_access(adapter, addr, …. data)

user space | kernel space

adapter->algo->smbus_xfer(adapter, addr, flags,
read_write, command, protocol, data);

```
// for each i2c device              struct i2c_driver *  driver;
struct i2c_client {                 struct device  dev;
    unsigned short  flags;           int  irq;
    unsigned short  addr;           struct list_head  list;
    char  name[I2C_NAME_SIZE];      struct completion  released;
    struct i2c_adapter *  adapter;  };
```

Example of I2C Devices

❑ **Two Wii nunchuck devices**

  ❖ one is connected to ICH8 SMBus

  ❖ one is connected to I2C adapter on PCI bus

❑ **2 instances of I2C_client**

  ❖ different I2C device names

  ❖ different adapters

  ❖ Use the same device driver

  ❖ same I2C slave address 0x52

❑ **When read from the nunchucks**

  ❖ Same I2C signals on both buses, e.g. start, addr, R/W, ack …..

  ❖ Different commands are sent to the different adapters (ICH8 SMBus module and PCI I2C adapter)

  ❖ Driver makes the same call to

   i2c_smbus_xfer of *i2c.core* and then *adapter->algo->smbus_xfer*

User Space Access to I2C Devices

❑ **Basically, a device driver to control I2C adapters**

  ❖ Send and receive raw data to and from I2C buses

❑ **An I2C device driver can process the raw data and present data according to device model**

  ❖ A nunchuck device driver measures the speed of joystick movement instead of reporting joystick position.

❑ **I2C-dev – loadable module**

  ❖ Major number: 89

  ❖ Minor number: defined for each adapter

  ❖ i2c_dev represents an i2c_adapter, an I2C or SMBus master, not a slave (i2c_client) – called */dev/i2c-0, /dev/i2c-1, /dev/i2c-2*, etc.

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

How to Use I2C-dev

❑ **Load i2c-dev module**

❑ **Create an i-node for the device**

> % mknod /dev/i2c-0 c 89 0

❑ **Include i2c-dev.h where i2c-dev interface is defined**

```
#define ADDRESS 0x38
int fd;
fd = open( "/dev/i2c-0", O_RDWR );          // open a device file
ioctl( fd, I2C_SLAVE, ADDRESS );            // set up the slave address
```

❖ Using read() and write) for an entire I2C transaction takes place (i.e. start bit, address, data, stop).

❖ Using the wrapper functions that i2c-dev.h provides.

❖ SMBus commands

$i2c\_smbus\_write\_byte\_data()$ → $i2c\_smbus\_acces$

→ $ioctl(file,I2C\_SMBUS,\&args)$

→ $S \_ Addr \_ Wr \_ [A] \_ Comm \_ [A] \_ Data \_ [A] \_ P$