Example function pointer snippet taken from the Linux Kenel
This is a very small snippet showing how function pointers can be used

Defining a function pointer

From linux/include/linux/parport.h

parport_driver is the struct where the Linux kernel has a struct
with two function pointers for attaching and detaching a device
driver to a particular lp port.

```
struct parport_driver {
        const char *name;
        void (*attach) (struct parport *);
        void (*detach) (struct parport *);
        struct parport_driver *next;
};
```

Using the function pointer

From linux/drivers/parport/share.c - Parallel port manager

parport_announce_port() -> attach_driver_chain()

parport_announce_port tells any devic drivers about a
parallel port to announce. parport_announce_port calls
attach_driver_chain to attach the device driver to this
parallel port by calling back into the device drivers
attach() method

```
/* Call attach(port) for each registered driver. */
static void attach_driver_chain(struct parport *port)
{
        struct parport_driver *drv;
        void (**attach) (struct parport *);
        int count = 0, i;


       ....
        attach = kmalloc (sizeof (void(*)(struct parport *)) * count,
                          GFP_KERNEL);
        if (!attach) {
                printk (KERN_WARNING "parport: not enough memory to attach\n");
               return;
```

```
        }

        spin_lock (&driverlist_lock);
        for (i = 0, drv = driver_chain; drv && i < count; drv = drv->next)
                attach[i++] = drv->attach;
        spin_unlock (&driverlist_lock);

        for (count = 0; count < i; count++)
                (*attach[count]) (port);

        kfree (attach);
}
```

Implementing the function that the function pointer calls

From linux/drivers/char/lp.c

lp_driver is a parport_driver type that says the driver's name is lp,
and defines two functions for attaching and detaching the lp driver.

```
static struct parport_driver lp_driver = {
        "lp",
        lp_attach,
        lp_detach,
        NULL
};
```

lp_attach is the actual function that is called by attach_driver_chain
when it calls lp's attach function which is actually lp_attach

```
static void lp_attach (struct parport *port)
{
        unsigned int i;

        switch (parport_nr[0])
        {
        case LP_PARPORT_UNSPEC:
        case LP_PARPORT_AUTO:
                if (parport_nr[0] == LP_PARPORT_AUTO &&
                    port->probe_info[0].class != PARPORT_CLASS_PRINTER)
                        return;
                if (lp_count == LP_NO) {
                        printk("lp: ignoring parallel port (max. %d)\n",LP_NO);
                        return;
                }
```

```
                if (!lp_register(lp_count, port))
                        lp_count++;
                break;

        default:
                for (i = 0; i < LP_NO; i++) {
                        if (port->number == parport_nr[i]) {
                                if (!lp_register(i, port))
                                        lp_count++;
                                break;
                        }
                }
                break;
        }
}
```