# OSLab6 文档

5140379064 陈俊

一、Time_tick 相关

这部分实现很简单，在 trap.c 中关于时钟中断的处理函数中加入 time_tick 函数，用来进行 tick 的计数。之后，添加系统调用，在系统调用中调用 time_msec，就可以实现所需的功能。

二、PCI attach

首先，在 pci.c 的 pci_attach_vendor 中，添加 vendor、device 和 attach 函数。Vendor 和 device 在文档中 5.2 可获得。

在 e1000.c 中，实现 attach 函数。最开始，调用 pci_fun_enable 函数，激活 e1000 device。之后，使用 mmap，将设备的地址映射到内存地址中，我选择了 KSTACKTOP 以及 KERNBASE 中间相距 PTSIZE 的空闲虚拟地址，用来对 e1000 设备进行映射。

之后，就是对 Transmit 和 Receive 相关的寄存器的初始化。首先，构造两个数组：tx_table 以及 rx_table，用来充当（TX ring 和 RX ring）。同时，初始化这两个 table，将 tx_table 的 status 位置成 DD，将 rx_table 中的地址位都指向一个足够大数组的物理地址。

在 tx 相关寄存器中，将 TDBAL/TDBAH、TDLEN、TDH/TDT、TCTL 以及 TIPG 都置上合适的值，完成 tx 的初始化。在 rx 相关寄存器中，将 RAL/RAH、RDBAL/RDBAH、RDH/RDT、RCTL，都按照文档上的指示，填上合适的值，完成 rx 的初始化。

三、Transmit 相关

在 transmit_packet 中，将 td 的地址作为参数传入。这个 td 是在 sys_transmit_packet 中根据传入的 data 和 len 构造好的。在 transmit_packet 函数里，根据当前的 tdt，从 tx_table 中找到对应的 td，将新传入的 td 赋给这个 td，并修改 cmd 中的值，以及将 tdt 置为后一个 tdt。

在 sys_transmit_packet 中，构建一个 tx_desc 的对象。将 data 的虚拟地址转成物理地址，放入该对象中，并将 len 也放入对象中。

最外层的是 output.c，它接收 nsipcbuf，获得 data 的地址和长度，并调用 system call，不断循环调用 sys_transmit_packet，用来不断发送包。

这样，就完成了整个 transmit 的过程。

特别地，当 tx ring 满的时候，transmit_packet 会返回 -1，这时候，在 sys_transmit_packet 会对 transmit_packet 返回值进行判断，本来是不断循环，遇到返回值为 0 则 break 跳出；若为-1，则继续循环，直至 tx ring 空闲。

四、Receive 相关

我在先前也提到，在初始化 rx_table（RX ring）的同时，我也初始化了一个 rx_buf，用来当作每个 rx_desc 的地址。在初始化 rx_table 的同时，将 rx_table 中的 buffer_addr 设为 rx_buf 中对应数组的地址。

在 receive_packet 中，将(*rdt + 1) % RXDESC_LENGTH 作为读取的对象，从 rx_table 中读取相应的 rd。同时，清空该 rd 的 DD 位和 EOP 位，告诉 e1000 这块区域可以重新使用，并更新 rdt。

在 sys_receive_packet 中，构建一个 rx_desc 的对象，并将其指针作为参数传给 receive_packet 函数，获得 rd。将 rd 中的地址通过 KADDR 转为虚拟地址，再将内容 memset 到 buf 的地址。

最外层的是 input.c 中的函数。在 intput.c 中，调用 sys_receive_packet，获得 buf 的

地址和长度。通过 memmove，将 data 的内容 copy 到指定的地址，并通过 ipc_send 发送给 core network 进程。

这样，就完成了整个 receive 过程。

特别地，当 rx_ring 为空的时候，会返回-1，sys_receive_packet 也会返回-1。在 input.c 中，如果返回值小于 0，则在循环中调用 sys_yield，否则不符合条件，跳出循环，这样就实现了一直等待接收的效果。

五、Web server 相关

这部分主要实现了 send_data 和 send_file。在 send_data 中，首先，通过 fstat，获得文件的相关信息（比如文件的长度）。然后，调用 readn 函数，从 fd 中读取内容到 buf 中，之后，调用 write 函数，往 req->socket 中写 buf，达到写 data 的目的。

在 send_file 中，之后的 send_header、send_size、send_data 等逻辑已经实现了。我主要做的就是打开 fd，通过 fstat 获得文件的信息（len），并判断是否是一个文件夹，并对这些情况进行 send_error 处理。最开始，通过 open，以只读方式打开 req->url，获得 fd。之后，调用 fstat，获得 len，并通过 st_isdir 来判断其是否为文件夹。

这样，整个 send_file 的逻辑就实现了。

当 server 打开时，在网页中输入 http://localhost:26002/index.html 后，就能看到

This file came from JOS.

Cheesy web page!

这说明 web server 是成功的。

六、Challenge

Challenge 中，我选择了第一个 Challenge，从 EEPROM 中获取 MAC 地址。其实现和测试详见 answers-lab6.txt，在此不再赘述。