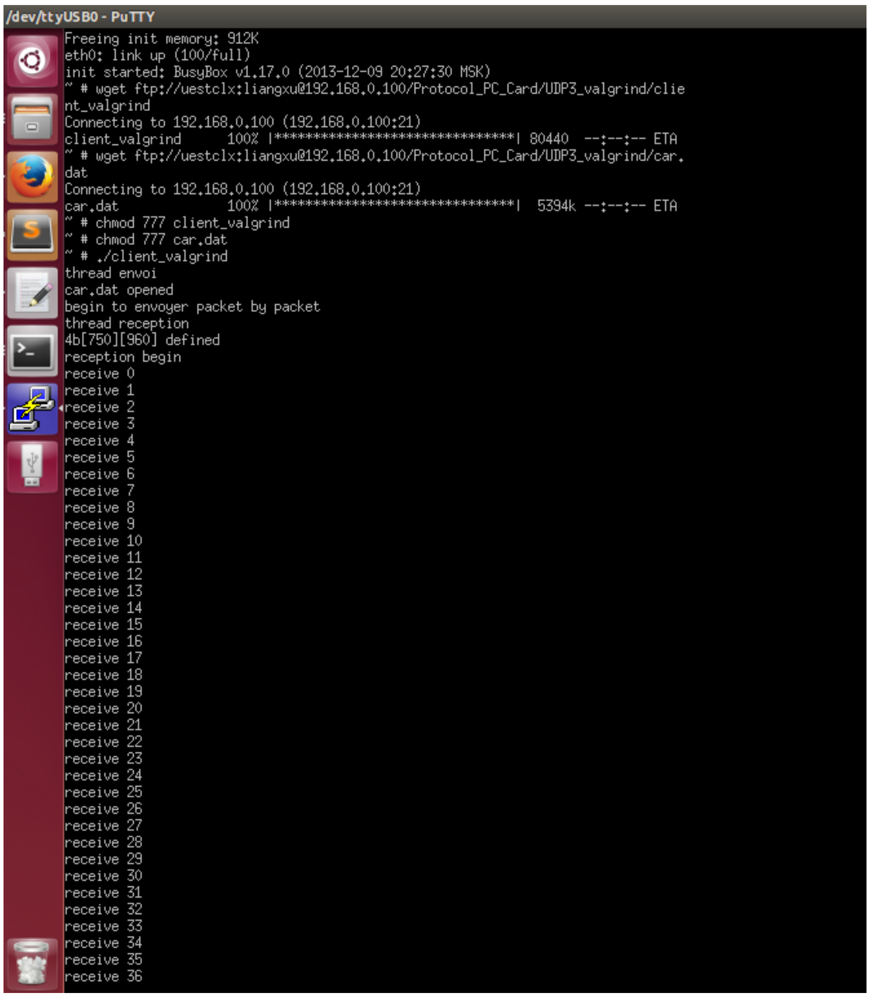
UDP3_valgrind

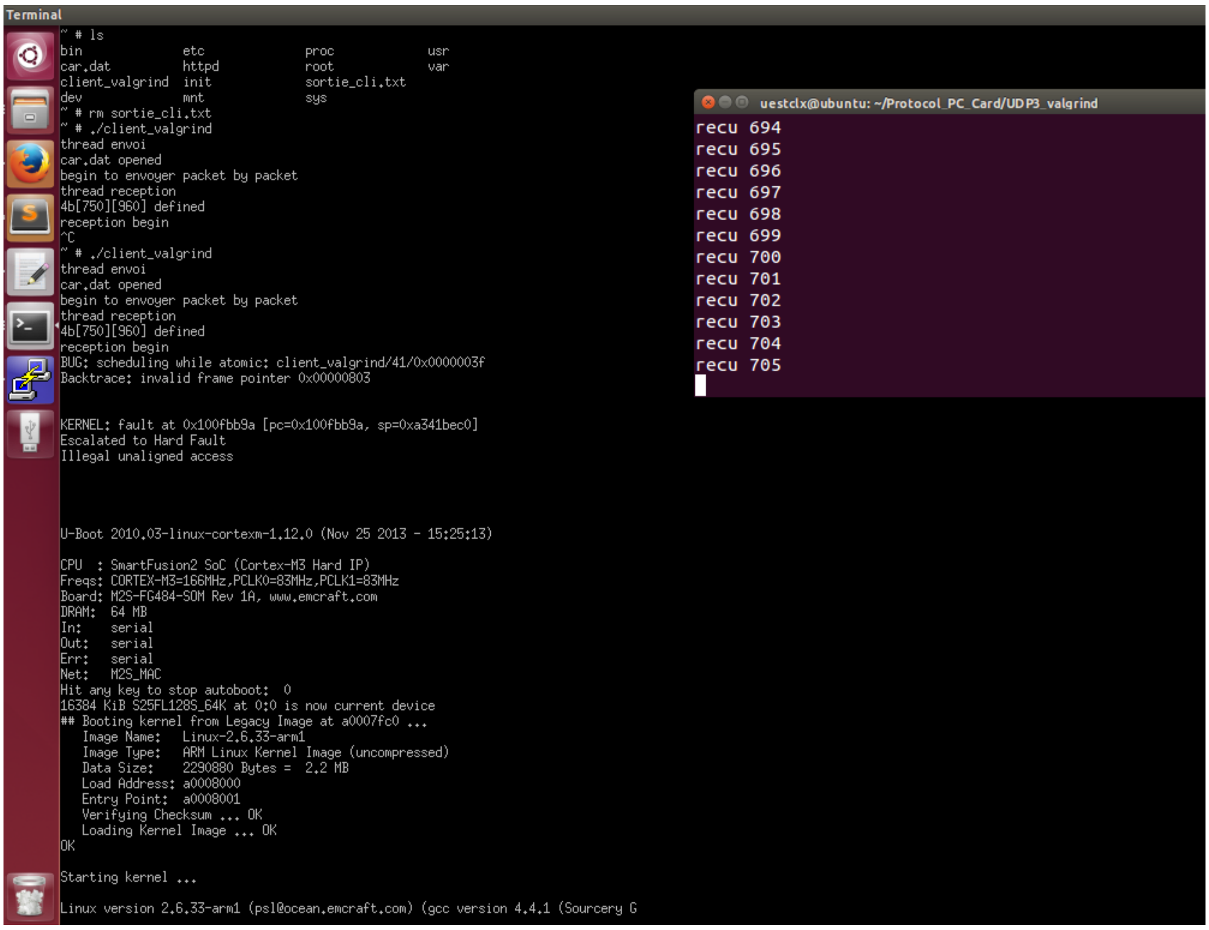
- main.cpp
  - change long to int
  - and IP address in reception and treatment.
- client.c
  - change IP address
  - add print phrases
  - add fclose(fs) and fclose(sortie2)
  - original code, stopped after the reception of the 682th packet (see screen shot), the first several values in the output file are correct. I assume that the values in the first packet will be correct, those who follows can not be garanteed.



```
/dev/ttyUSB0 - PuTTY
Freeing init memory: 912K
eth0: link up (100/full)
init started: BusyBox v1.17.0 (2013-12-09 20:27:30 MSK)
~ # wget ftp://uestclx:liangxu@192.168.0.100/Protocol_PC_Card/UDP3_valgrind/clie
nt_valgrind
Connecting to 192.168.0.100 (192.168.0.100:21)
client_valgrind      100% |********************************| 80440  --:--:-- ETA
~ # wget ftp://uestclx:liangxu@192.168.0.100/Protocol_PC_Card/UDP3_valgrind/car.
dat
Connecting to 192.168.0.100 (192.168.0.100:21)
car.dat              100% |********************************|  5394k --:--:-- ETA
~ # chmod 777 client_valgrind
~ # chmod 777 car.dat
~ # ./client_valgrind
thread envoi
car.dat opened
begin to envoyer packet by packet
thread reception
4b[750][960] defined
reception begin
receive 0
receive 1
receive 2
receive 3
receive 4
receive 5
receive 6
receive 7
receive 8
receive 9
receive 10
receive 11
receive 12
receive 13
receive 14
receive 15
receive 16
receive 17
receive 18
receive 19
receive 20
receive 21
receive 22
receive 23
receive 24
receive 25
receive 26
receive 27
receive 28
receive 29
receive 30
receive 31
receive 32
receive 33
receive 34
receive 35
receive 36
```

  - QUESTIONS
  - using ‘while’ for the reception, if there is a packet lost, the programm will never ends.
  - In the attribute of the fuction recvfrom, why using (struct sockaddr *)&serverStorage, &addr_size? In the last version, we were using NULL

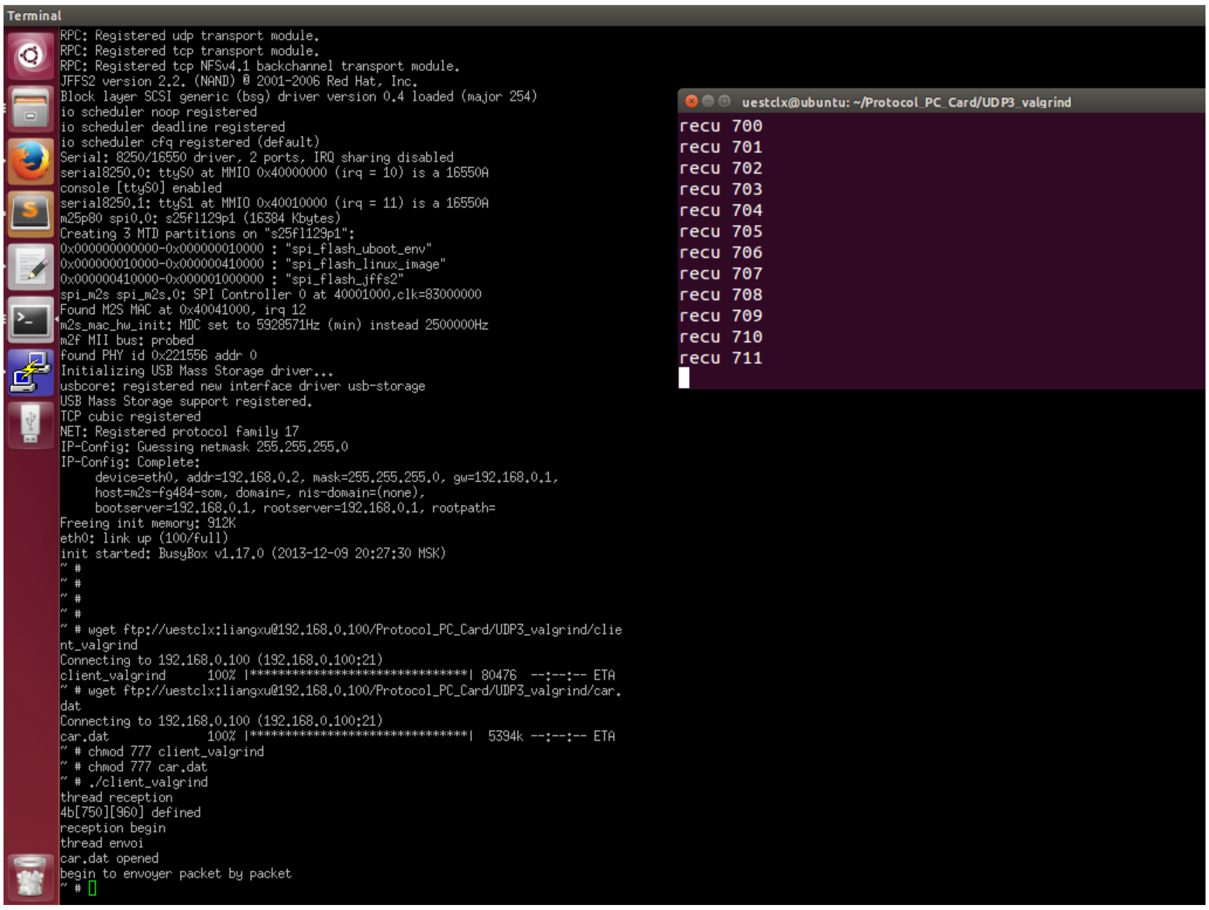
- In this reception procedure, why do we need a buffer of size [750][960]? As described in the programm we receive one frame and then write it in the file, then receive another frame, etc. In the meantime we can’t do other things, not even receive the following packet, just waiting for the programm writing some values into a file.
- And on the other hand, what if the server send another packet while we were writing the previous one into a file? There will probably be a lost of packet.
- MODIFICATIONS
- I changed the order, first receiving all the values into the buffer, then write them into a file. But always system crush.
- It happened one time, after sending the 685th packet(the server received the 685th packet and then stopped); Another time is after sending the 705th packet (the server received the 705th packet and then stopped). see the screenshot.



```
~ # ls
bin             etc             proc            usr
car.dat         httpd           root            var
client_valgrind init            sortie_cli.txt
dev             mnt             sys
~ # rm sortie_cli.txt
~ # ./client_valgrind
thread envoi
car.dat opened
begin to envoyer packet by packet
thread reception
4b[750][960] defined
reception begin
^C
~ # ./client_valgrind
thread envoi
car.dat opened
begin to envoyer packet by packet
thread reception
4b[750][960] defined
reception begin
BUG: scheduling while atomic: client_valgrind/41/0x0000003f
Backtrace: invalid frame pointer 0x00000803


KERNEL: fault at 0x100fbb9a [pc=0x100fbb9a, sp=0xa341bec0]
Escalated to Hard Fault
Illegal unaligned access




U-Boot 2010.03-linux-cortexm-1.12.0 (Nov 25 2013 - 15:25:13)

CPU  : SmartFusion2 SoC (Cortex-M3 Hard IP)
Freqs: CORTEX-M3=166MHz,PCLK0=83MHz,PCLK1=83MHz
Board: M2S-FG484-SOM Rev 1A, www.emcraft.com
DRAM:  64 MB
In:    serial
Out:   serial
Err:   serial
Net:   M2S_MAC
Hit any key to stop autoboot:  0
16384 KiB S25FL128S_64K at 0:0 is now current device
## Booting kernel from Legacy Image at a0007fc0 ...
   Image Name:   Linux-2.6.33-arm1
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2290880 Bytes =  2.2 MB
   Load Address: a0008000
   Entry Point:  a0008001
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

Linux version 2.6.33-arm1 (psl@ocean.emcraft.com) (gcc version 4.4.1 (Sourcery G
```

```
recu 694
recu 695
recu 696
recu 697
recu 698
recu 699
recu 700
recu 701
recu 702
recu 703
recu 704
recu 705
```

- And another time stopped after the server received the 711th packet. It indicates that the client didn’t finished his reception but stopped. There’s nothing in the output file.



Terminal

```
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
JFFS2 version 2.2. (NAND) © 2001-2006 Red Hat, Inc.
Block layer SCSI generic (bsg) driver version 0.4 loaded (major 254)
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
Serial: 8250/16550 driver, 2 ports, IRQ sharing disabled
serial8250.0: ttyS0 at MMIO 0x40000000 (irq = 10) is a 16550A
console [ttyS0] enabled
serial8250.1: ttyS1 at MMIO 0x40010000 (irq = 11) is a 16550A
m25p80 spi0.0: s25fl129p1 (16384 Kbytes)
Creating 3 MTD partitions on "s25fl129p1":
0x000000000000-0x000000010000 : "spi_flash_uboot_env"
0x000000010000-0x000000410000 : "spi_flash_linux_image"
0x000000410000-0x000001000000 : "spi_flash_jffs2"
spi_m2s spi_m2s.0: SPI Controller 0 at 40001000,clk=83000000
Found M2S MAC at 0x40041000, irq 12
m2s_mac_hw_init: MDC set to 5928571Hz (min) instead 2500000Hz
m2f MII bus: probed
found PHY id 0x221556 addr 0
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
TCP cubic registered
NET: Registered protocol family 17
IP-Config: Guessing netmask 255.255.255.0
IP-Config: Complete:
     device=eth0, addr=192.168.0.2, mask=255.255.255.0, gw=192.168.0.1,
     host=m2s-fg484-som, domain=, nis-domain=(none),
     bootserver=192.168.0.1, rootserver=192.168.0.1, rootpath=
Freeing init memory: 912K
eth0: link up (100/full)
init started: BusyBox v1.17.0 (2013-12-09 20:27:30 MSK)
~ #
~ #
~ #
~ #
~ # wget ftp://uestclx:liangxu@192.168.0.100/Protocol_PC_Card/UDP3_valgrind/clie
nt_valgrind
Connecting to 192.168.0.100 (192.168.0.100:21)
client_valgrind      100% |*******************************| 80476  --:--:-- ETA
~ # wget ftp://uestclx:liangxu@192.168.0.100/Protocol_PC_Card/UDP3_valgrind/car.
dat
Connecting to 192.168.0.100 (192.168.0.100:21)
car.dat              100% |*******************************|  5394k --:--:-- ETA
~ # chmod 777 client_valgrind
~ # chmod 777 car.dat
~ # ./client_valgrind
thread reception
4b[750][960] defined
reception begin
thread envoi
car.dat opened
begin to envoyer packet by packet
~ # 
```

uestclx@ubuntu: ~/Protocol_PC_Card/UDP3_valgrind

```
recu 700
recu 701
recu 702
recu 703
recu 704
recu 705
recu 706
recu 707
recu 708
recu 709
recu 710
recu 711
```

- Tried another way to compile the client.c, system crashed after the server received the 605th packet.
- I changed the amount of trame to be received to 700
- reboot the card and this time, receive and then write it into a file, it work, the programm received 700 packets and wrote them into the file.
- output values are correct (using the second way to compile the client.c)
- And then I change 700 back to 750, in either of the two compilation methode can it run to the end. I run it for three times, the system bocked once, auto-reboot once, and exit in the middle once……. it stopped around 703-705.

I will use valgrind tomorrow to see if there will be something strange. Before that, does anyone of you have some idea? If you want me to run some test to prouve something, don’t hesitate :)