

test1: (J'ai pas fait ce test, puisque)

On a déjà essayé d'envoyer le fichier car.dat en basant sur UDP2, et ça a bien marché.

On peut envoyer 750*960 échantillons depuis la carte, et les valeurs reçues par le PC sont correctes.

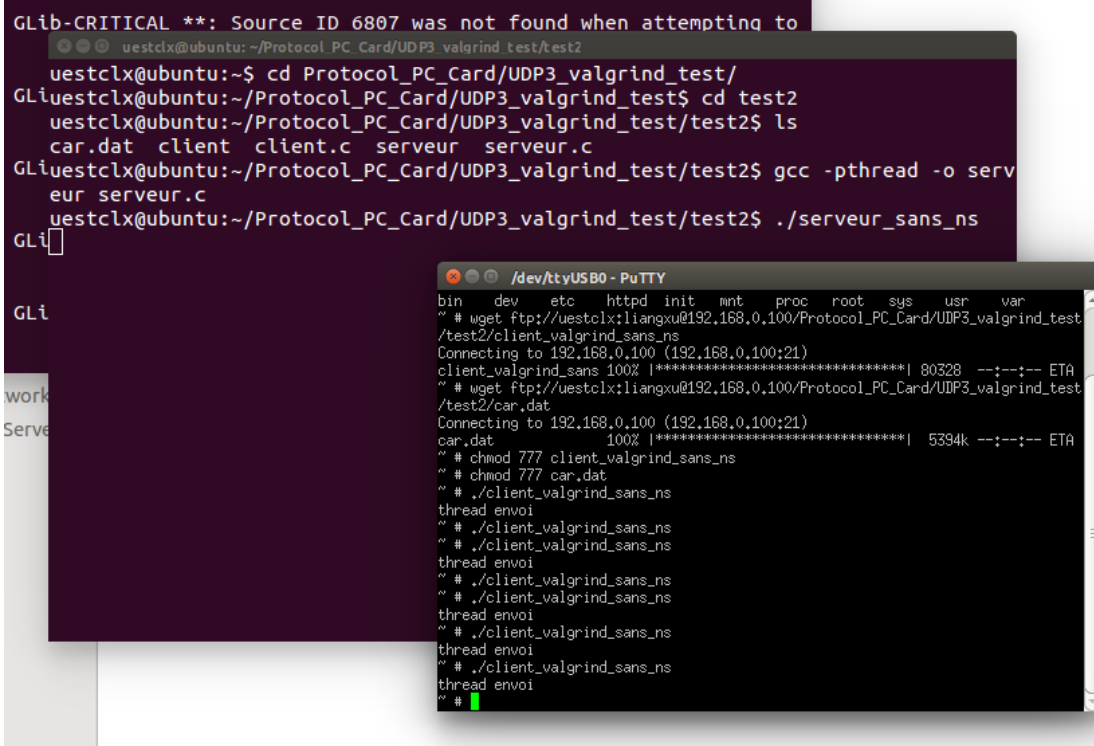
Dans ce cas-là, la carte lit 960 échantillons et puis les envoie, et lit autre 960 échantillons; le buffer d'envoi n'a qu'une dimension.

test2:

J'ai changé l'adresse IP dans le fichier serveur.c, également 'long'-->'int' du buffer;

Aussi les adresses IP dans le fichier client.c;

Je l'ai compilé en utilisant le toolchain; Le programme ne s'exécute pas sur la carte. même plusieurs essais (voir le screenshot)



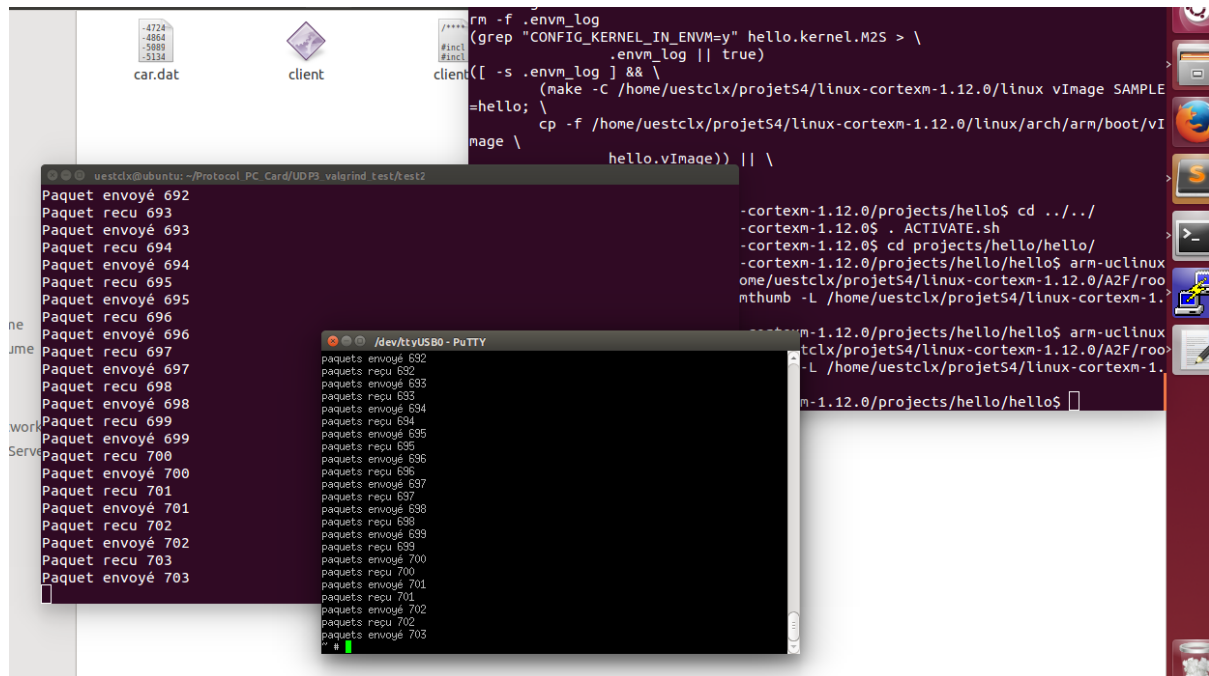
The image contains two overlapping terminal window screenshots. The background window shows a series of commands in a terminal on an Ubuntu system, including directory navigation, file listing, compilation with gcc, and execution of a server program. The foreground window, titled '/dev/ttyUSB0 - PuTTY', shows the output of the server program, including file transfers via wget and wget, and a loop of sending data via threads.

```
Glib-CRITICAL **: Source ID 6807 was not found when attempting to
uestclx@ubuntu: ~/Protocol_PC_Card/UDP3_valgrind_test/test2
uestclx@ubuntu:~$ cd Protocol_PC_Card/UDP3_valgrind_test/
GLiuestclx@ubuntu:~/Protocol_PC_Card/UDP3_valgrind_test$ cd test2
uestclx@ubuntu:~/Protocol_PC_Card/UDP3_valgrind_test/test2$ ls
car.dat client client.c serveur serveur.c
GLiuestclx@ubuntu:~/Protocol_PC_Card/UDP3_valgrind_test/test2$ gcc -pthread -o serveur serveur.c
uestclx@ubuntu:~/Protocol_PC_Card/UDP3_valgrind_test/test2$ ./serveur_sans_ns
GLi

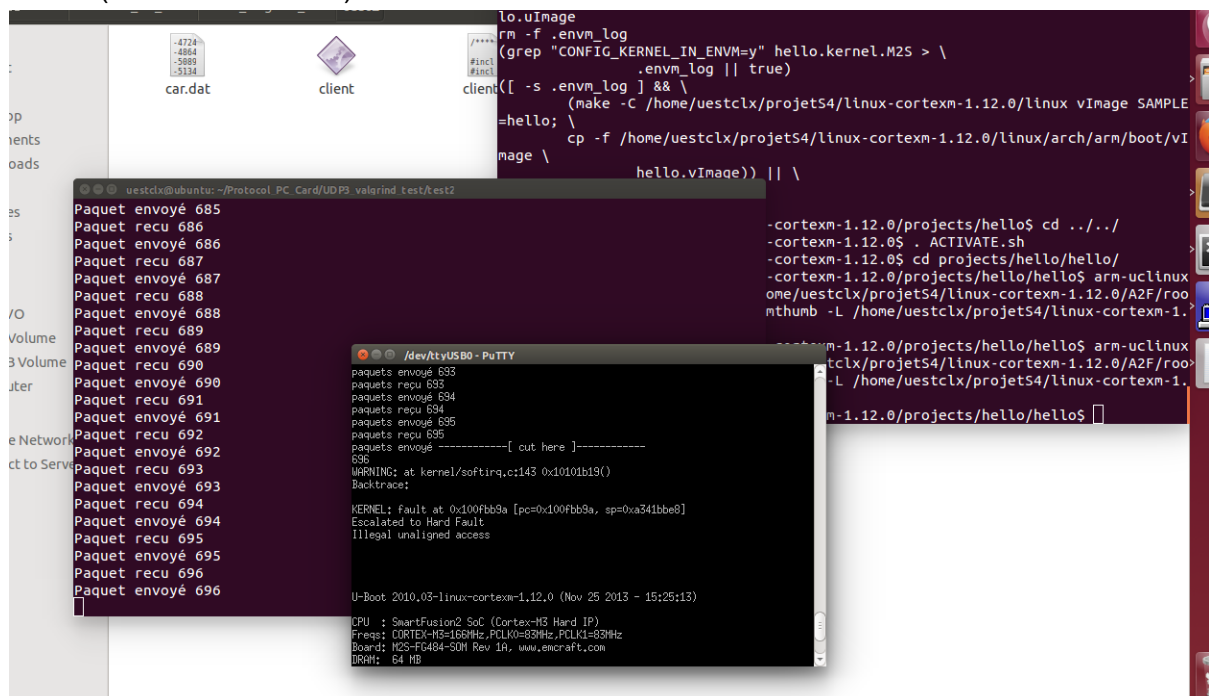
GLi
work
Serve

/dev/ttyUSB0 - PuTTY
bin dev etc httpd init mnt proc root sys usr var
" # wget ftp://uestclx:liangxu@192.168.0.100/Protocol_PC_Card/UDP3_valgrind_test
/test2/client_valgrind_sans_ns
Connecting to 192.168.0.100 (192.168.0.100:21)
client_valgrind_sans 100% |*****| 80328 --:--:-- ETA
" # wget ftp://uestclx:liangxu@192.168.0.100/Protocol_PC_Card/UDP3_valgrind_test
/test2/car.dat
Connecting to 192.168.0.100 (192.168.0.100:21)
car.dat 100% |*****| 5394k --:--:-- ETA
" # chmod 777 client_valgrind_sans_ns
" # chmod 777 car.dat
" # ./client_valgrind_sans_ns
thread envoi
" # ./client_valgrind_sans_ns
" # ./client_valgrind_sans_ns
thread envoi
" # ./client_valgrind_sans_ns
" # ./client_valgrind_sans_ns
thread envoi
" # ./client_valgrind_sans_ns
thread envoi
" #
" #
```

Je l'ai compilé en utilisant une autre commande, le programme s'exécute, mais s'arrête au 703 ème paquet, et sortie du programme (voir le screenshot)



Je lance le programme encore une fois, il s'arrête au 695 ème paquet et la carte reboot.(voir le screenshot)



Je retransmets le fichier et exécute le client encore une fois, cette fois-ci, le programme s'arrête au 703 ème paquet, relance, encore au 703ème paquet. Je le relance 3 autres fois, et il s'arrête toujours au 703ème paquet.

Et dans ce cas-là, sur la carte, le programme ne crée pas le fichier 'sortie_cli.txt'.

test2_opt:
changer adresses IP

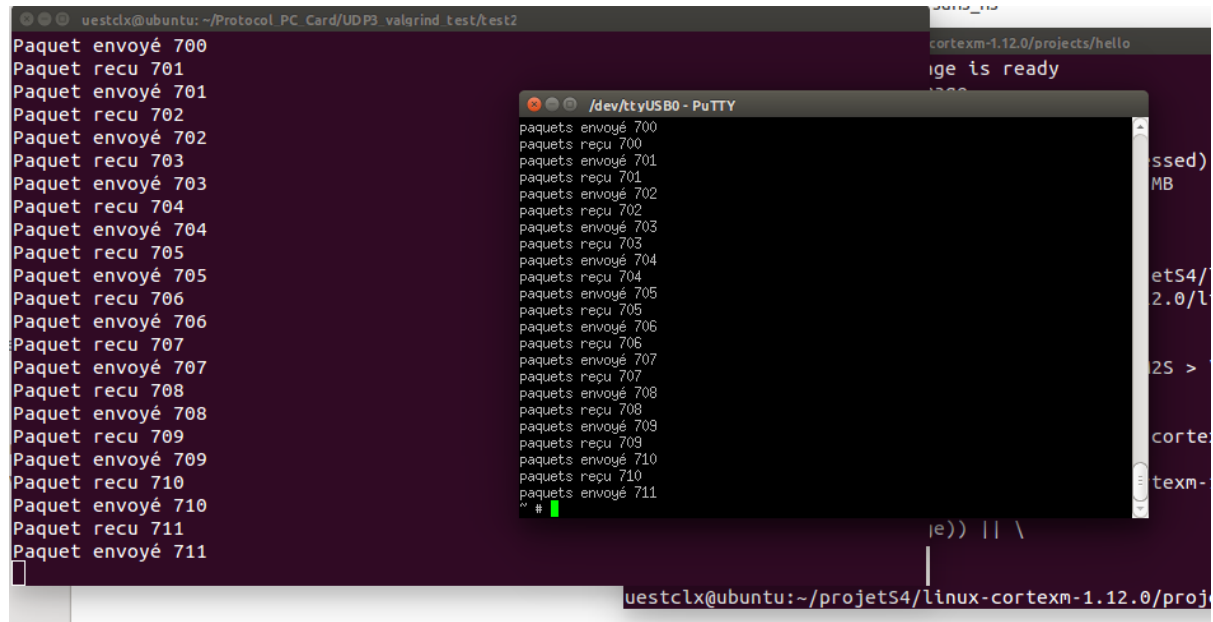
Je cherche l'optimisation de compilation, je trouve que dans le Makefile fichier, on a utilisé -Os pour compiler l'exécutable sur la carte. J'ai changé l'attribut '-Os' dans la ligne 'CFLAGS' à -O2;

Re-compiler et le programme s'arrête au 711 ème paquet sur la carte.

Puis j'ai ajouté l'attribut '-O2' à la ligne 'LDFLAGS';

Reboot la carte;

Re-compiler et le programme s'arrête au 711 ème paquet sur la carte.



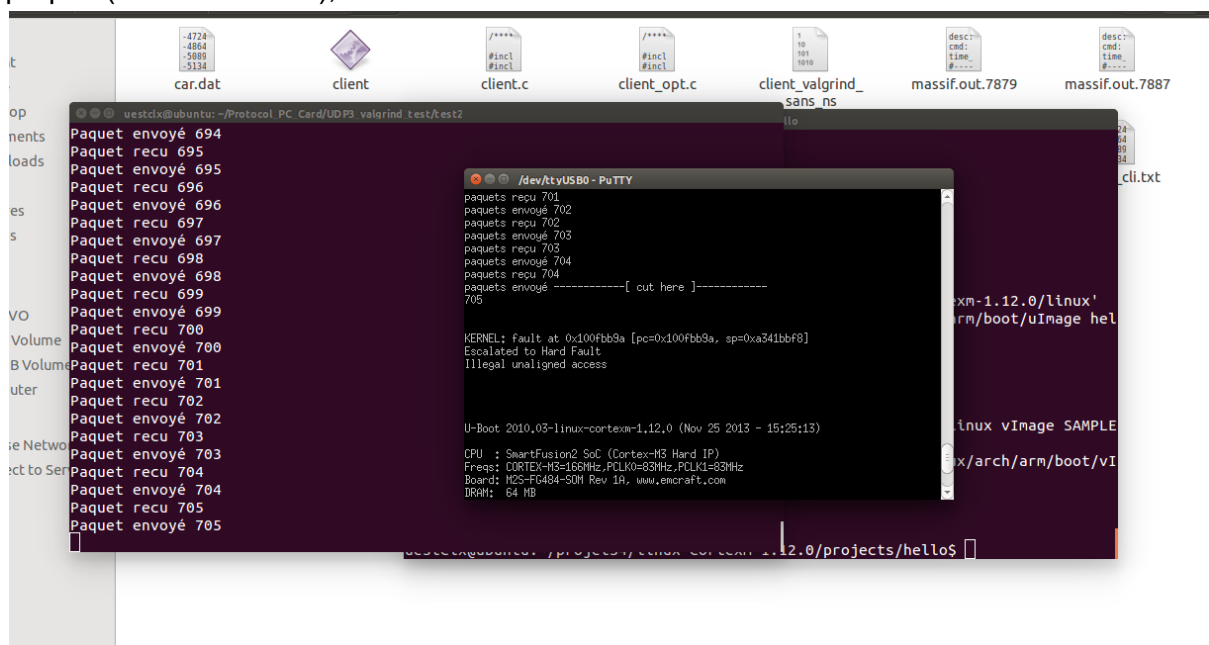
```
uestclx@ubuntu: ~/Protocol_PC_Card/UDP3_valgrind_test/test2
Paquet envoyé 700
Paquet reçu 701
Paquet envoyé 701
Paquet reçu 702
Paquet envoyé 702
Paquet reçu 703
Paquet envoyé 703
Paquet reçu 704
Paquet envoyé 704
Paquet reçu 705
Paquet envoyé 705
Paquet reçu 706
Paquet envoyé 706
Paquet reçu 707
Paquet envoyé 707
Paquet reçu 708
Paquet envoyé 708
Paquet reçu 709
Paquet envoyé 709
Paquet reçu 710
Paquet envoyé 710
Paquet reçu 711
Paquet envoyé 711
Paquet reçu 711

uestclx@ubuntu: ~/projetS4/linux-cortexm-1.12.0/proj
```

Puis J'ai changé le niveau d'optimisation à -O1

La carte crash et reboot après la réception de 698 ème paquet.

Relance le programme, la carte crash et reboot après la réception de 705 ème paquet.(voir screenshot);



```
uestclx@ubuntu: ~/Protocol_PC_Card/UDP3_valgrind_test/test2
Paquet envoyé 694
Paquet reçu 695
Paquet envoyé 695
Paquet reçu 696
Paquet envoyé 696
Paquet reçu 697
Paquet envoyé 697
Paquet reçu 698
Paquet envoyé 698
Paquet reçu 699
Paquet envoyé 699
Paquet reçu 700
Paquet envoyé 700
Paquet reçu 701
Paquet envoyé 701
Paquet reçu 702
Paquet envoyé 702
Paquet reçu 703
Paquet envoyé 703
Paquet reçu 704
Paquet envoyé 704
Paquet reçu 705
Paquet envoyé 705

paquets reçus 701
paquets envoyés 702
paquets reçus 702
paquets envoyés 703
paquets reçus 703
paquets envoyés 704
paquets reçus 704
paquets envoyés 705
paquets reçus 705

[ cut here ]-----
KERNEL: Fault at 0x100fbb9a [pc=0x100fbb9a, sp=0xa341bbf8]
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: M25-FG484-SOM Rev 1A, www.enecraft.com
DRAM: 64 MB
```

Donc je pense que l'optimisation -O2 est plus stable, et plus optimisée.

Je recompile le programme encore une fois avec -O2, mais malheureusement, la carte crash...(fuck!)

Reboot la carte, et retransférer tous les fichiers, lance le programme, il s'arrête au 711ème paquet.

Je veux savoir le problème vient de la réception ou bien l'envoi. Donc je ne lance pas le serveur, ainsi que le client ne reçoit pas les paquets. De cette manière, on peut savoir si le client peut bien envoyer tous les 750 paquets.

Je constate que le programme peut envoyer tous les 750 packets sans problème.

Je pense qu'on peut faire un test, le serveur envoie 750 paquets au client, et voir s'il peut bien tout recevoir.