ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός Ενσωματωμένων Συστημάτων

9ο Εξάμηνο, 2024-2025

6η Εργαστηριακή Άσκηση

αναφορά της φοιτήτριας:

Λάζου Μαρία-Αργυρώ (el20129)

Ομάδα:**35**

Εγκατάσταση custom cross-compiler building toolchain crosstool-ng

Ακολουθήθηκαν τα βήματα που περιγράφονται στην εκφώνηση μέσα σε vm που τρέχει Ubuntu 22.04.5 LTS και τα προβλήματα που ανέκυψαν ήταν ελάχιστα και αφορούσαν στην εγκατάσταση κάποιων missing dependencies (makinfo, help2man, libtool) κατά το configuration step. Για την επίλυσή τους απλώς τα εγακτέστησα μέσω του apt package. Συγκεκριμένα:

```
checking for makeinfo... no
configure: error: missing required tool: makeinfo
mlazoy@mlazoy-vm:~/crosstool-ng$ sudo apt install texinfo
```

```
checking for help2man... no
configure: error: missing required tool: help2man
mlazoy@mlazoy-vm:~/crosstool-ng$ sudo apt install help2man
```

```
checking for GNU libtool >= 2.4... no
configure: error: Required tool not found: libtool
mlazoy@mlazoy-vm:~/crosstool-ng$ sudo apt install libtool-bin
```

Έπειτα η εντολή *./configure –prefix=${HOME}/crosstool* επιτυγχάνει, οπότε μπροώ να προσχωρήσω στο building step του cross-compiler για την αρχιτεκτινική **arm-cortexa9_neon-linux-gnueabihf**. Κατά την διάρκεια των επόμενων βημάτων δεν παρουσιάστηκαν προβλήματα.

Άσκηση 1

1. Η βασική διαφορά των 2 αρχιτεκτονικών **debian_wheezy_armel** και **debian_wheezy_armhf** έγκυται στον τρόπου υπολογισμού πράξεων κινητής υποδιαστολής. Η πρώτη διαχειρίζεται τα floating point operations μέσω software emulation τεχνικές, ενώ η δεύτερη τις εκτελεί απευθείας πάνω στο hardware και προορίζεται για επεξεργαστές που διαθέτουν κατάλληλα units (FPUs). Εξού και τα ακρωνύμια **el** (Embedded Linux) που εκφράζει την πλειοψηφία μηχανημάτων soft float που χρησιμοποιήθηκαν στο παρελθόν για ενσωματωμένες εφαρμογές και υιοθετήθηκε για ιστορικούς λόγους και **hf** (Hard Float) που αναφέρεται σε σύγχρονους επεξαργαστές με FPUs (π.χ ARM Cortex-A8 ή Cortex-A9). Η el είναι συμβατή με επεξεργαστές των οικογενειών ARMv5, ARMv5, ARMv6 ενώ η hf απαιτεί επεξεργαστή γενίας ARMv7. Η επίδοση των τελευταίων είναι σαφώς καλύτερη.

2. Χρησιμοποιούμε την αρχιτεκτονική **arm-cortexa9_neon-linux-gnueabihf** για λόγους compatibility με το qemu image που κατεβάσαμε. Συγκεκριμένα Debian Wheezy ARMHF είναι configured με τα ακόλουθα χαρακτηριστικά:

- ARMv7-A (που περιλαμβάνει τον Cortex A9 core)
- Neon intrinsics για υποστήριξη SIMD εντολών (ως περαιτέρω optimizations)
- VFPv3-D16 units για hard-float υπολογισμούς

Και τα 3 ταυτίζονται με επιλεγμένη αρχιτεκτονική ένα προς ένα όπως δηλώνουν και τα επιμέρους πεδία του ονόματός της.

Σε περίπτωση επιλογής διαφορετικής αρχιτεκτινικής δεν θα υπήρχε ταύτιση με το ISA (π.χ. διαφορετικό μήκος εντολών, μη υποστήριξη FLOPs) οπότε θα παίρναμε σφάλματα: *Illegal insrtuction* ή *Exec format error.*

3. Δεν απαιτήθηκε κάποια χειροκίνητη αλλαγή στην βιβλιοθήκη που χρησιμοποιήσε στο configuration script ο cross compiler αφού η gcc ήταν by default επιλεγμένη. Επιβεβαιώνω το παραπάνω μετά την εγκατάσταση με την ακόλουθη εντολή:

```
mlazoy@mlazoy-vm:~$  ldd  -v  x-tools/arm-cortexa9_neon-linux-gnueabihf/bin/arm-cortexa9_neon-linux-
gnueabihf-gcc
  linux-vdso.so.1 (0x00007fff67d78000)
  libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007d9ae8400000)
  /lib64/ld-linux-x86-64.so.2 (0x00007d9ae8799000)

  Version information:
  x-tools/arm-cortexa9_neon-linux-gnueabihf/bin/arm-cortexa9_neon-linux-gnueabihf-gcc:
    ld-linux-x86-64.so.2 (GLIBC_2.3) => /lib64/ld-linux-x86-64.so.2
    libc.so.6 (GLIBC_2.3) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.15) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.7) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.14) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.32) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.33) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.4) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.34) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.11) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.2.5) => /lib/x86_64-linux-gnu/libc.so.6
    libc.so.6 (GLIBC_2.3.4) => /lib/x86_64-linux-gnu/libc.so.6
  /lib/x86_64-linux-gnu/libc.so.6:
    ld-linux-x86-64.so.2 (GLIBC_2.2.5) => /lib64/ld-linux-x86-64.so.2
    ld-linux-x86-64.so.2 (GLIBC_2.3) => /lib64/ld-linux-x86-64.so.2
    ld-linux-x86-64.so.2 (GLIBC_PRIVATE) => /lib64/ld-linux-x86-64.so.2
```

Είναι η πιο safe επιλογή καθώς είναι συμβατή με Debian - μάλιστα προτείνεται από την ίδια για cross-compiling - και υποστηρίζει headers για ARMv7 αρχιτεκτονικές.

4. Τρέχω την ακόλουθη εντολή για την μεταγλώττιση του phods.c με τον cross compiler που χτίσαμε:

```
mlazoy@mlazoy-vm:~$ ~/x-tools/arm-cortexa9_neon-linux-gnueabihf/bin/arm-cortexa9_neon-linux-gnueabihf-
gcc -O0 -Wall -o phods_crosstool.out /home/mlazoy/Downloads/Lab6/phods.c
```

Παρατηρώ το είδος του εκτελέσιμου που παρήχθη με την ετνολή:

```
mlazoy@mlazoy-vm:~$ file phods_crosstool.out
phods.out: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /
lib/ld-linux-armhf.so.3, for GNU/Linux 3.2.0, with debug_info, not stripped
```

Από το readelf μπορούμε να δούμε περισσότερες λεπτομέρειες για την αρχιτεκτονική που προορίζεται, το endian, τα section headers (.text, .data, etc), τα program headers, το symbol table (globals, functions, etc), relocation entries και debugging πληροφορίες:

```
mlazoy@mlazoy-vm:~$ readelf -h -A phods_crosstool.out
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x104e0
  Start of program headers:          52 (bytes into file)
  Start of section headers:          13648 (bytes into file)
  Flags:                             0x5000400, Version5 EABI, hard-float ABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         9
  Size of section headers:           40 (bytes)
  Number of section headers:         36
  Section header string table index: 35
Attribute Section: aeabi
File Attributes
  Tag_CPU_name: "7-A"
  Tag_CPU_arch: v7
  Tag_CPU_arch_profile: Application
  Tag_ARM_ISA_use: Yes
  Tag_THUMB_ISA_use: Thumb-2
  Tag_FP_arch: VFPv3
  Tag_Advanced_SIMD_arch: NEONv1
  Tag_ABI_PCS_wchar_t: 4
  Tag_ABI_FP_rounding: Needed
  Tag_ABI_FP_denormal: Needed
  Tag_ABI_FP_exceptions: Needed
  Tag_ABI_FP_number_model: IEEE 754
  Tag_ABI_align_needed: 8-byte
  Tag_ABI_align_preserved: 8-byte, except leaf SP
  Tag_ABI_enum_size: int
  Tag_ABI_VFP_args: VFP registers
  Tag_CPU_unaligned_access: v6
  Tag_MPextension_use: Allowed
  Tag_Virtualization_use: TrustZone
```

Πρόκειται, λοιπόν, για ένα 32-bit ELF file,little-endian με υποστήριξη στο EABI. Επιπλέον διαθέτει THUMB-2 και NEONv1 εντολές και χρησιμοποιεί dynamic linking με την GLIBC 3.2.0.

5. Για να μεταγλωττίσω το ίδιο αρχείο phpds.c με τον linaro compiler εκτελώ την εντολή:

```
mlazoy@mlazoy-vm:~$   linaro/gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux/bin/arm-linux-gnueabihf-
gcc -O0 -Wall -std=gnu11 -o phods_linaro.out /home/mlazoy/Downloads/Lab6/phods.c
```

η οποία σκάει με το ακόλουθο error οπότε κατεβάζω την ζητούμενη shared library που λείπει:

```
z.so.1: cannot open shared object file: No such file or directory
mlazoy@mlazoy-vm:~$ sudo apt install lib32z1-dev
```

και επαναλαμβάνω το compilation step που τώρα επιτυγχάνει.

Συγκρίνοντας το μέγεθος των 2 παραγόμενων εκτέλεσιμων, παρατηρώ ότι το phods_linaro.out είναι περίπου το μισό (8878 bytes) σε σχέση με το phods_crosstool (15088 bytes).

```
mlazoy@mlazoy-vm:~$ ls -al | grep phods
-rwxrwxr-x  1 mlazoy mlazoy  15088 Ιαν  11 12:57 phods_crosstool.out
-rwxrwxr-x  1 mlazoy mlazoy   8878 Ιαν  11 13:18 phods_linaro.out
```

Από το αντίστοιχο readelf βλέπω:

```
mlazoy@mlazoy-vm:~$ readelf -h -A phods_linaro.out
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x8479
  Start of program headers:          52 (bytes into file)
  Start of section headers:          4264 (bytes into file)
  Flags:                             0x5000402, Version5 EABI, hard-float ABI, <unknown>
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         8
  Size of section headers:           40 (bytes)
  Number of section headers:         31
  Section header string table index: 28
Attribute Section: aeabi
File Attributes
  Tag_CPU_name: "7-A"
  Tag_CPU_arch: v7
  Tag_CPU_arch_profile: Application
  Tag_ARM_ISA_use: Yes
  Tag_THUMB_ISA_use: Thumb-2
  Tag_FP_arch: VFPv3-D16
  Tag_ABI_PCS_wchar_t: 4
  Tag_ABI_FP_rounding: Needed
  Tag_ABI_FP_denormal: Needed
  Tag_ABI_FP_exceptions: Needed
  Tag_ABI_FP_number_model: IEEE 754
  Tag_ABI_align_needed: 8-byte
  Tag_ABI_align_preserved: 8-byte, except leaf SP
  Tag_ABI_enum_size: int
  Tag_ABI_HardFP_use: Deprecated
  Tag_ABI_VFP_args: VFP registers
  Tag_CPU_unaligned_access: v6
```

Παρατηρώ ότι τα περισσότερα χαρακτηριστικά είναι ίδια, όμως το linaro ΔΕΝ χρησιμοποιεί NEONv1 που εξηγεί την διαφορά στο μέγεθος των 2 ELFs αφού εισάγει έξτρα ελέγχους για πραγματοποιήση SIMD εντολών.

6. Η εκτέλεση του binary από τον cross-compiler αποτυγχάνει με το ακόλουθο error:

```
root@debian-armhf:/home/user# ./phods_crosstool.out
./phods_crosstool.out: /lib/arm-linux-gnueabihf/libc.so.6: version `GLIBC_2.34' not found (required
by ./phods_crosstool.out)
```

Αντίθετα η εκτέλεση του binary από το linaro toolchain είναι επιτυχής.

```
root@debian-armhf:/home/user# ./phods/phods_linaro.out
Previous frame doesn't exist.
```

7. Μεταγλώττίζοντας ξανά το phods.c με τα 2 εργαλεία και το έξτρα flag -static παρατηρώ τα μεγέθη των νέων παραγόμενων αρχείων συγκριτικά με τα προηγούμενα:

```
mlazoy@mlazoy-vm:~$ ls -lh | grep phods
-rwxrwxr-x  1 mlazoy mlazoy  15K Ιαν  11 12:57 phods_crosstool.out
-rwxrwxr-x  1 mlazoy mlazoy 2,5M Ιαν  11 14:11 phods_crosstool_static.out
-rwxrwxr-x  1 mlazoy mlazoy 8,7K Ιαν  11 14:12 phods_linaro.out
-rwxrwxr-x  1 mlazoy mlazoy 497K Ιαν  11 14:12 phods_linaro_static.out
```

Η διαφορά είναι αισθητή όπως αναμενόταν διότι το -static option εξαναγκάζει τον compiler να παράγει κώδικα για ολόκληρη την βιβλιοθήκη μέσα στο εκτέλισμο και δεν γίνεται δυναμικό linking με την υπάρχουσα shared μορφή της.

Εαν δοκιμάσω να τρέξω το νέο εκτελέσιμο από τον crosstool στο target δεν εμφανίζει κανένα πλέον σφάλμα για εύρεση της GLIBC:

```
root@debian-armhf:/home/user# ./phods/phods_crosstool_static.out
Previous frame doesn't exist.
```

8. Στο υποθετικό σενάριο δημιουργίας μιας δικής μας custom συνάρτησης στην libc (θεώρησα ότι η αλλαγή κάποιου headerfile της glibc καθώς και το Makefile ώστε να συμπεριλάβει την mlab_foo() μάλλον δεν είναι τόσο καλή ιδέα...) αναμένω τα ακόλουθα μετά το compilation του με τον cross compiler του βήματος 3 και flags -Wall -O0 -o my_foo.out :

A. Σε κάθε περίπτωση το ./my_foo.out στον host αποτυγχάνει αφού δεν ταιριάζει στην αρχιτεκτονική που έχω.

B. Στο target μηχάνημα θα αποτύχει διότι έχει παραχθεί με δυναμικό linking της glibc και η shared εκδοχή που εντοπίζει on runtime στο target δεν περιλαμβάνει την mlab_foo() σε κανένα headerfile.

C. Στην περίπτωση του στατικού linking επειδή γίνεται παραγωγή κώδικα για ολόκληρη την glibc μαζί με το .c αρχείο στο ίδιο εκτελέσιμο, το ./my_foo.out θα επιτύχει γιατί δεν έχει εξαρτήσεις από κανένα shard library και η mlab_foo() έχει γίνει inline.

Άσκηση 2

Αρχικά αντιγράφω ολόκληρο το ./boot directory στο host ώστε να μπορώ μετά το τέλος της εγκατάστασης του νέου πυρήνα να συγκρίνω τα περιεχόμενά τους.

```
mlazoy@mlazoy-vm:~$ scp -P 22223 -r root@localhost:/boot ./
root@localhost's password:
vmlinuz-3.2.0-4-vexpress                    100% 1989KB 694.8KB/s   00:02
System.map-3.2.0-4-vexpress                 100% 1100KB 491.0KB/s   00:02
vmlinuz                                     100% 1989KB   2.7MB/s   00:00
initrd.img                                  100% 1928KB   2.7MB/s   00:00
config-3.2.0-4-vexpress                     100%   77KB 492.2KB/s   00:00
initrd.img-3.2.0-4-vexpress                 100% 1928KB   2.7MB/s   00:00
```

Αντιγράφω τον νέο πηρύνα στο host για να γίνει locally το extraction:

```
mlazoy@mlazoy-vm:~$ scp -P 22223 -r root@localhost:/usr/src/linux-source-3.16.tar.xz ./
root@localhost's password:
linux-source-3.16.tar.xz                    100%   78MB   1.0MB/s   01:17
mlazoy@mlazoy-vm:~$ tar -xJf linux-source-3.16.tar.xz
```

Κατά την εισγαωγή του config δημιουργήθηκε το σφάλμα που περιγράφει η εκφώνηση με τον lexer οπότε δηλώνω τη αντίστοιχη μεταβλητή ως extern και επαναλαμβάνω:

```
mlazoy@mlazoy-vm:~/linux-source-3.16/scripts/dtc$ sed -i 's/YYLTYPE yylloc;/extern YYLTYPE yylloc;/g'
dtc-lexer.lex.c_shipped
mlazoy@mlazoy-vm:~/linux-source-3.16/scripts/dtc$ cat dtc-lexer.lex.c_shipped  | grep YYLTYPE
extern YYLTYPE yylloc;
```

Ακολούθησαν διάφορα σφάλματα που σχετίζονται με την δήλωση των .section flags. Χρειάστηκε να αντικατασταθούν οι ακόλουθες γραμμές *.section .start,#alloc,#execute* με *.section,"ax"* στα αρχεία:

1. *arch/arm/boot/compressed/head.S*
2. *arch/arm/boot/compressed/big-endian.S*
3. *arch/arm/boot/bootp/init.S*
4. *arch/arm/mm/proc-v7.S*

και την *.section .start,#alloc* με *.section .start,"a"* στα αρχεία :

1. *arch/arm/boot/compressed/piggy.lzma.S*
2. *arch/arm/boot/compressed/piggy.gzip.S*

Τελικά τα 3 αρχεία παράχθηκαν με επιτυχία και παρακάτω βλέπουμε κάποιες πληροφορίες για το καθένα:

```
mlazoy@mlazoy-vm:~/linux-source-3.16$ dpkg -I ../linux-headers-3.16.84_3.16.84-6_armhf.deb
 new Debian package, version 2.0.
 size 6753290 bytes: control archive=199564 bytes.
     449 bytes,    14 lines      control
  954326 bytes, 10078 lines      md5sums
 Package: linux-headers-3.16.84
 Source: linux-upstream
 Version: 3.16.84-6
 Architecture: armhf
 Maintainer: Anonymous <root@mlazoy-vm>
 Installed-Size: 42367
 Provides: linux-headers, linux-headers-2.6
 Section: kernel
 Priority: optional
 Homepage: http://www.kernel.org/
 Description: Linux kernel headers for 3.16.84 on armhf
  This package provides kernel header files for 3.16.84 on armhf
  .
  This is useful for people who need to build external modules
```

```
mlazoy@mlazoy-vm:~/linux-source-3.16$ dpkg -I ../linux-libc-dev_3.16.84-6_armhf.deb
 new Debian package, version 2.0.
 size 770966 bytes: control archive=18508 bytes.
     462 bytes,    13 lines      control
   50703 bytes,   773 lines      md5sums
 Package: linux-libc-dev
 Source: linux-upstream
 Version: 3.16.84-6
 Architecture: armhf
 Maintainer: Anonymous <root@mlazoy-vm>
 Installed-Size: 3743
 Provides: linux-kernel-headers
 Section: devel
 Priority: optional
 Homepage: http://www.kernel.org/
 Description: Linux support headers for userspace development
  This package provides userspaces headers from the Linux kernel.  These headers
  are used by the installed headers for GNU glibc and other system libraries.
```

```
mlazoy@mlazoy-vm:~/linux-source-3.16$ dpkg -I ../linux-image-3.16.84_3.16.84-6_armhf.deb
 new Debian package, version 2.0.
 size 12572118 bytes: control archive=28880 bytes.
     463 bytes,    14 lines      control
   98433 bytes,  1121 lines      md5sums
     285 bytes,    12 lines   *  postinst             #!/bin/sh
     281 bytes,    12 lines   *  postrm               #!/bin/sh
     283 bytes,    12 lines   *  preinst              #!/bin/sh
     279 bytes,    12 lines   *  prerm                #!/bin/sh
 Package: linux-image-3.16.84
 Source: linux-upstream
 Version: 3.16.84-6
 Architecture: armhf
 Maintainer: Anonymous <root@mlazoy-vm>
 Installed-Size: 40357
 Suggests: linux-firmware-image-3.16.84
 Provides: linux-image, linux-image-2.6, linux-modules-3.16.84
 Section: kernel
 Priority: optional
 Homepage: http://www.kernel.org/
 Description: Linux kernel, version 3.16.84
  This package contains the Linux kernel, modules and corresponding other
  files, version: 3.16.84.
```

Δυστυχώς το compression αυτών δεν υποστηρίζεται όταν τα ανεβάζουμε στο qemu και παίνρουμε αυτό το σφάλμα:

```
root@debian-armhf:~# dpkg -i /bootable/linux-headers-3.16.84_3.16.84-6_armhf.deb
dpkg-deb: error: archive '/bootable/linux-headers-3.16.84_3.16.84-6_armhf.deb' contains not understood
data member control.tar.zst, giving up
dpkg: error processing /bootable/linux-headers-3.16.84_3.16.84-6_armhf.deb (--install):
 subprocess dpkg-deb --control returned error exit status 2
Errors were encountered while processing:
 /bootable/linux-headers-3.16.84_3.16.84-6_armhf.deb
```

Οι επιλογές είναι 2, είτε να αναβαθμίσω το dpkg package στον armhf που είναι πολύ παλιό και δεν θα παίξει οπότε πρέπει κάνω χρήση των ακόλουθων πακέτων: *aptitude, ztsd, bzip2* είτε να επαναλάβω το compilation του kernel αλλάζοντας το compression των παραγόμενων αρχείων από το αρχείο *scripts/package/builddeb* μετατρέποντας την γραμμή *dpkg –build* σε *dpkg-deb -Zgzip –build*. Ακολυθώ την 2η καθώς έχει λιγότερο manual compression-decompression ανάμεσα σε διαφορετικά formats. Τελικά αποσυμπιέζω τα αρχεία του kernel:

```
root@debian-armhf:~# dpkg -i /bootable2/linux-headers-3.16.84_3.16.84-7_armhf.deb
Selecting previously unselected package linux-headers-3.16.84.
(Reading database ... 32610 files and directories currently installed.)
Unpacking linux-headers-3.16.84 (from .../linux-headers-3.16.84_3.16.84-7_armhf.deb) ...
Setting up linux-headers-3.16.84 (3.16.84-7) ...
root@debian-armhf:~# dpkg -i /bootable2/linux-image-3.16.84_3.16.84-7_armhf.deb
Selecting previously unselected package linux-image-3.16.84.
(Reading database ... 45841 files and directories currently installed.)
Unpacking linux-image-3.16.84 (from .../linux-image-3.16.84_3.16.84-7_armhf.deb) ...
Setting up linux-image-3.16.84 (3.16.84-7) ...
update-initramfs: Generating /boot/initrd.img-3.16.84
root@debian-armhf:~# dpkg -i /bootable2/linux-libc-dev_3.16.84-7_armhf.deb
(Reading database ... 47211 files and directories currently installed.)
Preparing to replace linux-libc-dev:armhf 3.16.84-1 (using .../linux-libc-dev_3.16.84-7_armhf.deb) ...
Unpacking replacement linux-libc-dev ...
Setting up linux-libc-dev (3.16.84-7) ...
root@debian-armhf:~#
```

Για να εκκινήσει το μηχάνημα με τον νέο πηρύνα χρειάζεται να τα περάσουμε ως ορίσματα στο script που σηκώνει το qemu ως ακολούθως:

```
mlazoy@mlazoy-vm:~/QEMU$ cat run_qemu_armhf.sh
sudo qemu-system-arm \
-M vexpress-a9 \
-kernel vmlinuz-3.16.84 \
-initrd initrd.img-3.16.84 \
-drive if=sd,file=debian_wheezy_armhf_standard.qcow2 \
-append "root=/dev/mmcblk0p2" \
-net nic \
-net user,hostfwd=tcp:127.0.0.1:22223-:22
```

Συνεπώς πρέπει να τα κατεβάσουμε από το /boot directory του guest στον host.

Ερωτήματα

1. Πριν την εκκίνηση με τον νέο πηρύνα εκτελώ:

```
root@debian-armhf:/home/user# uname -a
Linux debian-armhf 3.2.0-4-vexpress #1 SMP Debian 3.2.51-1 armv7l GNU/Linux
```

Μετά την εκκίνηση η ίδια εντολή παράγει διαφορετικό αποτέλεσμα:

```
root@debian-armhf:~# uname -a
Linux debian-armhf 3.16.84 #7 SMP Sun Jan 12 11:29:09 EET 2025 armv7l GNU/Linux
```

Παρατηρώ ότι έχει λάβει το όνομα του νέου πυρήνα.

2. Για την προσθήκη ενός δικού μου custom system call ακολούθησα τα παρακάτω βήματα

- Δημιούργησα εκ νέου ένα directory με όνομα **greet/** μέσα στο *linux-source-3.16.84* το οποίο αποτελείται από το *greet.c* και το *Makefile* που φαίνονται παρακάτω:

greet.c

```c
#include <linux/kernel.h>

asmlinkage long sys_greet(void)
{
  int team_no = 35;
  printk("Greeting from kernel and team %d\n", team_no);
  return 0;
}
```

```
mlazoy@mlazoy-vm:~/linux-source-3.16$ cat greet/Makefile
obj-y := greet.o
```

- Προσθέτω στο ***arch/arm/kernel/call.S*** την ακόλουθη γραμμή μετά το definition του τελευταίου system call:

```
                CALL(sys_ni_syscall)            /* seccomp */
                CALL(sys_ni_syscall)            /* getrandom */
/* 385 */       CALL(sys_memfd_create)
                CALL(sys_greet_35)              /*custom syscall */
#ifndef syscalls_counted
.equ syscalls_padding, ((NR_syscalls + 3) & ~3) - NR_syscalls
#define syscalls_counted
#endif
```

Τώρα το system call μου έχει εισαχθεί στο system call table του συστήματος.

- Προσθέτω στο αρχείο ***arch/arm/include/uapi/asm/unistd.h*** στην τελευταία γραμμή τον αριθμό του custom system call (+ 386 από την βάση εφόσον ο υπάρχοντας τελευταίος ήταν 385 όπως φαίνεται στο παραπάνω στιγμιότυπο):

```
#define __NR_greet                  (__NR_SYSCALL_BASE+386)
```

- Προσθέτω στο αρχείο ***include/linux/syscalls.h*** την επικεφαλίδα της συνάρτησης που καλεί το custom system call μου:

```
asmlinkage long sys_greet(void);
```

- Τέλος προσθέτω το νέο directory *greet/* στους κανόνες για το core-y στο Makefile στο root directory του linux-source:

makefile.txt

```
core-y          += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ greet/
```

Με τον τρόπο αυτό ο compiler γνωρίζει που βρίσκεται το νεό system call και εξασφαλίζει ότι θα κάνει make το εν λόγω directory.

Με την ολοκλήρωση των παραπάνω βημάτων, επαναλαμβλανω το compilation με τον ίδιο cross compiler και τα ίδια configurations με προηγουμένων και έπειτα αποσυμπιέζω τα νέα .deb αρχεία στο guest. Για να λειτουργήσει το σύστημα με τις επιθυμητές αλλαγές απαιτείται **reboot**.

3. Για τον έλεγχο της επιθυμητής λειτουργίας του custom system call χρησιμοποιήθηκε ο ακόλουθος κώδικας:

test_greet.c

```c
#include <sys/syscall.h>
#include <unistd.h>
#include <stdio.h>

#define SYS_GREET 386

int main() {
    printf("Calling custom system call sys_greet...\n");
    int result = syscall(SYS_GREET);
    printf("sys_greet return status: %d\n", result);
    return 0;
}
```

ο οποίος παράγει το αποτέλεσμα :

```
root@debian-armhf:/home/user# gcc -Wall -std=gnu11 -o greet.o test_greet.c
root@debian-armhf:/home/user# ./greet.o
Calling custom system call sys_greet...
sys_greet return status: 0
root@debian-armhf:/home/user# dmesg | tail
[   85.753633] RPC: Registered named UNIX socket transport module.
[   85.758023] RPC: Registered udp transport module.
[   85.765045] RPC: Registered tcp transport module.
[   85.768947] RPC: Registered tcp NFSv4.1 backchannel transport module.
[   85.930012] FS-Cache: Loaded
[   86.215643] FS-Cache: Netfs 'nfs' registered for caching
[   86.608906] Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
[  242.211177] Greeting from kernel and team 35
[  352.542977] Greeting from kernel and team 35
[  409.429371] Greeting from kernel and team 35
root@debian-armhf:/home/user#
```

Τα απαιτούμενα αρχεία για τους σκοπούς της άσκησης παρατίθενται σε ξεχωριστούς φακέλους μέσα στο zip.