ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Ενσωματωμένα Συστήματα

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ANDROID-DEVKIT No:02

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http:// arch.icte.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα του Πανεπιστημίου Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

1. Σκοπός της άσκησης

Μεταγλώττιση, και έλεγχος του android πυρήνα με προσομοιωτή.

2. Παραδοτέα

(A) 2 ερωτήσεις
(C) 4 ασκήσεις

- **Παραδοτέο C1:** Screenshot στο οποίο να φαίνεται η επιτυχής δημιουργία του καταλόγου.
- **Παραδοτέο C2:** Screenshot στο οποίο να φαίνεται ο επιτυχής συγχρονισμός.
- **Παραδοτέο C3:** Screenshot στο οποίο να φαίνεται η επιτυχής εγκατάσταση των παραπάνω πακέτων.
- **Παραδοτέο C4:** Screenshot στο οποίο να φαίνεται η επιτυχής ολοκλήρωση των λήψεων.

Σημείωση:
Τα εργαστήρια απαιτούν την ύπαρξη λειτουργικού συστήματος Linux. Θα πρέπει να γίνει κανονική εγκατάσταση του λειτουργικού και όχι εκτέλεση μέσω εικονικής μηχανής καθώς κάτι τέτοιο δεν υποστηρίζεται. Η παρούσα εργασία έχει υλοποιηθεί σε Ubuntu 10.4 64bit, το οποίο και προτείνεται.

Το υλικό του εργαστηρίου αναπτύχθηκε σε συνεργασία με τον κ. Τάσο Φετινίδη.

Χρησιμοποιήθηκε υλικό από http:// free-electrons.com .

3. Περιγραφή της εργαστηριακής άσκησης

3.1 Πρώτη εκτέλεση

Σε αυτή την ενότητα:

- ✓ θα διαμορφώσετε το σύστημα στο οποίο θα χτίσετε το Android.
- ✓ θα μεταγλωττίσετε το πρώτο σας σύστημα αρχείων Android.

3.2 Εγκατάσταση

Παραμείνετε στον κατάλογο **/home/<user>/felabs/android/aosp/android**

3.3 Δημιουργία περιβάλλοντος

Τώρα που η εντολή repo sync έχει ολοκληρωθεί μπορούμε να μεταγλωττίσουμε ένα σύστημα αρχείων για το λειτουργικό Android με τον εξομοιωτή. Αυτή η διαδικασία συμπεριλαμβάνει και την δημιουργία του ίδιου του εξομοιωτή.

Αρχικά δίνουμε:

```
source build/envsetup.sh
```

Αυτό το αρχείο περιέχει αρκετές χρήσιμες συναρτήσεις φλοιού και ψευδώνυμα (alias) όπως το croot για να μεταβείτε στον αρχικό κατάλογο του πηγαίου κώδικα Android και το lunch για να διαλέξετε τις επιλογές δημιουργίας.

Ελέγξτε την μεταβλητή περιβάλλοντος PATH:

```
echo $PATH
```

Θα δείτε ότι το αρχείο build/envsetup.sh δεν έχει τροποποιήσει την μεταβλητή PATH. Αυτό θα γίνει την ώρα της δημιουργίας. Το ζητούμενο αποτέλεσμα για τον εξομοιωτή είναι γενικό, και θέλουμε να έχουμε μια μηχανική κατασκευή. Για να το επιτύχουμε, δίνουμε:

```
lunch generic-eng
```

3.4 Μεταγλώττιση του αρχείου συστήματος root

Το περιβάλλον κατασκευής θα χρησιμοποιήσει την κατάλληλη ρύθμιση για τη δημιουργία αυτού του στόχου. Πριν από την εκτέλεση της make, ελέγξτε πρώτα τον αριθμό των πυρήνων του επεξεργαστή στο σταθμό εργασίας σας, όπως φαίνεται από το λειτουργικό σύστημα *(η τεχνολογία hyperthreading θα μπορούσε να κάνει το λειτουργικό σας σύστημα να βλέπει 4 πυρήνες αντί για 2, για παράδειγμα)*.

```
cat /proc/cpuinfo
```

Μπορείτε να χρησιμοποιήσετε την εντολή make -j (το j προκύπτει από το jobs για να ενημερώσετε την make να τρέξει πολλές εργασίες μεταγλώττισης παράλληλα εκμεταλλευόμενη τους πολλαπλούς πυρήνες και να διασφαλίσει ότι τα I/O και οι πυρήνες του επεξεργαστή είναι πάντα σε 100% χρήση. Αυτό θα μειώσει σημαντικά τον χρόνο δημιουργίας. Για παράδειγμα, αν τα Linux βλέπουν 4 πυρήνες, θα έχουμε καλή απόδοση, καθορίζοντας τον διπλάσιο αριθμό παράλληλων θέσεων εργασίας:

```
make -j 8
```

Παραδοτέο C1: Screenshot στο οποίο να φαίνεται η επιτυχής μεταγλώττιση.

Αν εμφανιστούν προβλήματα με λάθος έκδοση της Java τότε μπορείτε να κατεβάσετε την σωστή έκδοση.

```
sudo add-apt-repository "deb http://archive.canonical.com/lucid partner"
sudo apt-get update
sudo apt-get install sun-java6-jdk
```

Ακόμα, σε σπάνιες περιπτώσεις, κάποιες εγκαταστάσεις μπορεί να αποτύχουν όταν εκτελούνται με πολλαπλές θέσεις εργασίας παράλληλα. Εάν συμβεί κάτι τέτοιο, μπορείτε να εκτελέσετε την εντολή make χωρίς όρισμα μετά την αποτυχία, και αυτό είναι πιθανό να επιλύσει το θέμα.

3.5 Έλεγχος της δομής Android

Κοιτάξτε ξανά στη μεταβλητή περιβάλλοντος PATH.

Μπορείτε να δείτε ότι περιέχει:

- ✓ τον κατάλογο **$HOME/felabs/android/aosp/android/out/host/linux-x86/bin/** Εδώ είναι τα νέα βοηθήματα που έχουν μόλις μεταγλωττιστεί.

- ✓ Έτοιμα εκτελέσιμα και ιδιαίτερα το ARM cross-compiling toolchain το οποίο ήταν ήδη παρόν *(present in the repositories fetched by repo.)*

Κοιτάξτε τα περιεχόμενα του καταλόγου **out/target/product/generic** . Εδώ είναι οι εικόνες του συστήματος που δημιουργήθηκαν. Εκτελέστε την εντολή

```
emulator
```

Θα τρέξει τον εξομοιωτή με αυτές τις εικόνες. Περιμένετε λίγα λεπτά μέχρι να φορτώσει ο εξομοιωτής το σύστημα και μετά ελέγξτε τον αριθμό της εγκατάστασης στις ρυθμίσεις του Android.

Παραδοτέο C2: Screenshot στο οποίο να φαίνεται η επιτυχής εκτέλεση του εξομοιωτή και ο αντίστοιχος αριθμός εγκατάστασης (id).

3.6 Μεταγλώττιση και φόρτωση του πυρήνα Android

Μεταβείτε στον κατάλογο **/home/<user>/felabs/android/kernel** .

3.7 Μεταγλώττιση πυρήνα για τον εξομοιωτή Android

Ο εξομοιωτής Android χρησιμοποιεί το QEMU για να δημιουργήσει ένα εικονικό ARM9 SoC *(System on chip)* που ονομάζεται Goldfish.

Στον απλό πηγαίο κώδικα του Android που λάβαμε, ο πηγαίος κώδικα του πυρήνα δεν είχε συμπεριληφθεί. Έτσι, το πρώτο πράγμα που έχουμε να κάνουμε είναι να κατεβάσουμε τους πηγαίους του πυρήνα

```
git clone https://android.googlesource.com/kernel/goldfish.git kernel
```

Μεταβείτε στον κατάλογο του πυρήνα. Εδώ είναι το μέρος στο οποίο ο φάκελος του πηγαίου κώδικα αντιγράφτηκε. Χρησιμοποιώντας την σημαία branch -a, βρείτε την πιο πρόσφατη σταθερή έκδοση του πυρήνα που υποστηρίζει την πλατφόρμα Goldfish.

```
git branch -a
```

Την ώρα που γράφονται αυτές οι γραμμές, οι εκδόσεις Linux 3.0 και 3.4 δεν φαίνεται να λειτουργούν με τον εξομοιωτή και έτσι θα πρέπει να επιλέξετε την έκδοση 2.629 του πυρήνα.

Μεταβείτε σε αυτή την έκδοση ακολούθως:

```
git checkout <branch>
```

Τώρα που έχουμε τους πηγαίους του πυρήνα, χρειαζόμαστε το toolchain για να μεταγλωττίσουμε τον πυρήνα. Ευτυχώς, το Android παρέχει έναν.

Αρχικά, προσθέστε το toolchain από το προηγούμενο εργαστήριο στη μεταβλητή περιβάλλοντος PATH.

```
PATH=$HOME/felabs/android/aosp/android/prebuilt/linux-x86/toolchain/arm-
eabi-4.4.3/bin:$PATH
```

Τώρα, διαμορφώστε τον πυρήνα για την επιθυμητή πλατφόρμα

```
ARCH=arm make goldfish_defconfig
```

Ακόμα, διαμορφώστε τον πυρήνα χρησιμοποιώντας ένα εργαλείο της επιλογής σας για να προσθέσετε ένα πρόθεμα της επιλογής για την έκδοση του πυρήνα και μεταγλωττίστε επιτέλους τον πυρήνα!

```
ARCH=arm CROSS_COMPILE=arm-eabi- make -j 4
```

Κάποια λεπτά αργότερα, θα έχετε την εικόνα του πυρήνα στον κατάλογο

arch/arm/boot

3.8 Φορτώστε τον νέο πυρήνα

Για να εκτελέσετε τον προσομοιωτή ξανά, αυτή τη φορά με τη νέα εικόνα του πυρήνα, θα πρέπει να επαναφέρετε το περιβάλλον που είχατε αφού δημιουργήσατε το Android από τους πηγαίους. Αυτό απαιτείται κάθε φορά που κάνετε επανεκκίνηση τον σταθμό εργασίας σας και κάθε φορά που δουλεύετε με νέο τερματικό.

Πηγαίνετε στον κατάλογο $HOME/felabs/android/aosp/android/

Αυτό που έχετε να κάνετε είναι να τρέξετε την εντολή source και την εντολή lunch για να προσδιορίσετε το αντικείμενο με το οποίο δουλεύετε.

```
source build/envsetup.sh
lunch generic-eng
```

Τώρα, ελέγξτε ότι μπορείτε ακόμα να τρέχετε τον εξομοιωτή όπως κάνατε στο προηγούμενο εργαστήριο.

Για να τρέξετε τον εξομοιωτή με τον δικό σας πυρήνα μπορείτε να χρησιμοποιήσετε την σημαία -kernel :

```
emulator -kernel OME/felabs/android/kernel/kernel/arch/arm/boot/zImage
```

Ξανά, ελέγξτε την έκδοση του πυρήνα στις ρυθμίσεις του Android.

Σιγουρευτείτε ότι ο πυρήνας δημιουργήθηκε αυτή την ώρα.

3.9 Δημιουργήστε ένα patchset από τον Android πυρήνα

Για να συγκρίνετε τις πηγές του πυρήνα του Android με τους επίσημους πυρήνες, θα πρέπει να διαθέτουμε και τους τελευταίους. Αυτό θα μας επιτρέψει να δούμε ποιοι διαφέρουν. Το Git είναι ιδανικό για αυτό και προσφέρει όλα όσα χρειαζόμαστε.

Αρχικά, προσθέστε την σταθερή αποθήκη του Greg Kroah-Hartmann στο αποθετήριο μας:

```
git remote add stable git://git.kernel.org/pub/scm/linux/kernel/git/stable/
linux-stable.git
```

Τώρα, πρέπει να κατεβάσουμε τις αλλαγές από αυτό το απομακρυσμένο αποθετήριο:

```
git fetch stable
```

Λίγο αργότερα, θα πρέπει να έχετε όλες τις αλλαγές στο δικό μας αποθετήριο, με δυνατότητα περιήγησης εκτός σύνδεσης!

Τώρα, ανοίξτε το Makefile με τον επεξεργαστή της αρεσκείας σας και δείτε την έκδοση του πυρήνα που είναι σε αυτό.

Τώρα που την γνωρίζουμε, μπορούμε να δημιουργήσουμε το σύνολο των δεσμεύσεων που διαφέρουν από τον linux πυρήνα δίνοντας:

```
git log v<kernel-version>..HEAD
```

Εξαγάγετε το αντίστοιχο σύνολο patches:

```
git format-patch v<kernel-version>..HEAD
```

Σε αυτό το σύνολο των patches, βρείτε αυτό που προσθέτει τα wakelocks.

Συγχαρητήρια! Τώρα έχετε όλα τα patches για να εφαρμόσετε σε έκδοση του πυρήνα vanilla για να αποκτήσετε τον πλήρη έλεγχο του πυρήνα του Android.

Σε αυτό το εργαστήριο, δείξαμε χρήσιμες εντολές του Git, αλλά δεν μπήκαμε σε λεπτομέρειες, καθώς το project δεν αφορά το Git.

Μην διστάσετε να ζητήσετε από τον εκπαιδευτή σας περισσότερες λεπτομέρειες σχετικά με Git.

Ερώτηση A1: Στις παραπάνω εντολές, χρησιμοποιήθηκε το πρόγραμμα git. Τι κάνει αυτό το πρόγραμμα; Μπορεί να αντικατασταθεί με κάποιο άλλο;

Παραδοτέο C3: Screenshot στο οποίο να φαίνεται η επιτυχής δημιουργία του patchset στο αποθετήριο μας.

4. Υποστηρίζοντας μια νέα πλακέτα

Μετά από αυτό το εργαστήριο:

- ✓ Θα μάθετε να δημιουργείτε το λειτουργικό Android σε καινούργιο hardware
- ✓ Θα φορτώνετε το λειτουργικό Android σε πραγματικό hardware
- ✓ Θα επιλύετε απλά προβλήματα που σας παρουσιάζονται στο Android
- ✓ Θα δημιουργείτε έναν κατάλογο εργασίας

4.1 Εγκατάσταση

Μεταβείτε στον κατάλογο **/home/<user>/felabs/android/linaro**

Εγκαταστήστε τα πακέτα:

- ✓ bzr
- ✓ python-argparse
- ✓ python-parted και
- ✓ python-yaml

4.2 Κατέβασμα του πηγαίου κώδικα

Το Linaro δουλεύει εδώ και κάποιο καιρό με μια βελτιστοποιημένη διανομή του Android, που διατίθεται σε τουλάχιστον ένα τσιπ από τους πιο σημαντικούς προμηθευτές ARM επεξεργαστών.

Σε κάθε έκδοση, το Linaro διατηρεί επίσης διανομή Android που αντιστοιχεί στην πιο πρόσφατη, διαθέσιμη για κατέβασμα, έκδοση Linux.

Μια από τις πλατφόρμες που υποστηρίζονται από το Linaro του Android είναι η BeagleBoard η οποία είναι αρκετά παρόμοια με το DevKit8000 που χρησιμοποιούμε. Ειδικότερα, οι δυο πλακέτες έχουν το ίδιο OMAP3530 SystemOn-Chip. Ακόμα, στις τελευταίες εκδόσεις του πυρήνα, ο πυρήνας που χρειαζόταν και για τις δύο πλακέτες έχει ενοποιηθεί έτσι ώστε να μπορούμε να φορτώσουμε τον ίδιο πυρήνα και στις δύο πλακέτες. Για αυτό τον λόγο, θα χρησιμοποιήσουμε αυτή την διανομή ως ένα σημείο εκκίνησης για το υπόλοιπο του εργαστηρίου.

Ερώτηση Α2: Η αναπτυξιακή πλακέτα beagleboard τι χαρακτηριστικά υλικού έχει;

Πρώτα, πρέπει να κατεβάσουμε τον πηγαίο κώδικα. Θα χρησιμοποιήσουμε την έκδοση 11.11 που βασίζεται στο Android 2.3 Gingerbread και στη διανομή Linux 3.1.1.

```
mkdir src
cd src
repo init -u git://git.free-electrons.com/android/linaro/platform/manifest.git
-b linaro-android-11.11-release
```

Αφιερώστε λίγο χρόνο στο αρχείο **repo/manifest.xml** και συγκρίνετε το με το αρχείο στον κατάλογο **$HOME/felabs/android/aosp/android/**

Μπορείτε να δείτε ότι το Linaro έχει πάρει τον πλήρη έλεγχο από τις δυνατότητες του αρχείου manifest, χρησιμοποιώντας πολλαπλούς git εξυπηρετητές και όχι έναν και αντικαθιστώντας κάποια από τα στοιχεία *(όπως για παράδειγμα το Dalvik και Bionic)* με δικές τους βελτιωμένες εκδόσεις.

Τώρα, ας ασχοληθούμε με την απαιτητική δουλειά της λήψης:

```
repo sync -j4
```

Παραδοτέο C4: Screenshot στο οποίο να φαίνεται η επιτυχής ολοκλήρωση των λήψεων.

Ξανά, ακόμα και με μια γρήγορη σύνδεση στο Ίντερνετ, η εντολή συγχρονισμού repo sync μπορεί να χρειαστεί πάνω από μια ώρα.

Ο καθηγητής σας θα σας κρατήσει απασχολημένους με άλλα πράγματα όσο διαρκούν τα downloads.

Την περίοδο που γράφτηκε αυτό το εργαστήριο, το Linaro δεν υποστηρίζει το λειτουργικό Android 4.0 Ice Cream Sandwich ή νεότερες εκδόσεις της πλακέτας BeagleBoard. Έτσι, αποφασίσαμε να ασχοληθούμε με την πιο σταθερή έκδοση του hardware που διαθέτουμε.

Επίσης, στο ενδιάμεσο, θα πρέπει να κατεβάσετε και να αποσυμπιέσετε την αντίστοιχη Linaro toolchain.

```
cd ..
wget --trust-server-names http://goo.gl/Wn4dM
tar jxf android-toolchain-eabi-linaro-4.6-2011.11-4-2011-11-15_12-22-49-linux-x86.tar.bz
```