Informatique embarquée 9/16

J.-M Friedt

FEMTO-ST/département temps-fréquence

jmfriedt@femto-st.fr

transparents à `jmfriedt.free.fr`

Machine virtuelle GNU/Linux à
`http://jmfriedt.sequanux.org/stretch.ova`
(VirtualBox 5.0.24)

14 mars 2017

Introduction

- Pourquoi un système d'exploitation (OS) sur circuit embarqué ?
- Impact (mémoire, CPU)
- Méthode de travail (les programmes développés sur PC tournent sur le système embarqué)
- gcc : environnement unifié de travail

Au dela du C : un système d'exploitation

- ajout d'une couche d'abstraction supplémentaire (assembleur - C - noyau)
- Pourquoi un système d'exploitation : un scheduler, un gestionnaire de mémoire (multitâche), une couche d'abstraction supposée cacher le matériel au programmeur, gestion des **systèmes de fichier** (> rawrite), **communication** (IP, TCP ...), console pour l'utilisateur.
- Par contre une contrainte additionnelle : maîtriser un nouvel ensemble de protocoles et méthodes de programmation ...
- ... afin de gérer le partage de ressources entre plusieurs programmes utilisateurs.

Généralités sur les systèmes d'exploitation ?

- Qu'est-ce que GNU/Linux : système d'exploitation, clone d'unix, compatible posix, **multiplateforme**.
- **Linux** est un noyau autour duquel fonctionnent des outils libres (**GNU**).
- Diverses bibliothèques C disponibles, avec différents impacts mémoire : glibc, uClibc, newlib ... et différentes fonctionnalités.
- Une distribution n'est qu'un emballage pour ces outils.
- uClinux pour les systèmes sans MMU [1], Linux pour les systèmes avec ; OpenWRT pour les routeurs.
- autres OS propriétaires : WinCE, LynxOS, QNX, vxWorks
- autres OS libres : *BSD (Free, Net, Open), Plan9, Inferno, Hurd

1. *Memory Management Unit*, gestionnaire de mémoire chargé des contrôles d'accès et de la conversion de mémoire virtuelle en mémoire physique

Méthodes de développement

9. `output/host/usr/bin/` pour la toolchain (host = PC – y faire pointer `$PATH`)

OS sur plateforme cible :

- Développement : le code est **validé sur PC** puis transféré sur **plateforme embarquée** (OS commun aux deux platformes).
- Respect des appels système POSIX ⇒ code exploitable sur toute plateforme *sous réserve* de séparer accès matériel et *endianness*
- pour deux systèmes avec **gestionnaire de mémoire**, le code PC est directement exploitable
- en l'absence de gestionnaire de mémoire, il manque quelques fonctionnalités qu'il faut éviter (`fork`, lourdeur de `malloc`)
- NFS pour rapidement tester les nouvelles images.

```
mount -o nolock 192.168.1.2:/home /tmp/nfs
```

Rappel : bibliothèques

Une implémentation de `libc` fournissant les appels systèmes de Linux.

Sur Olinuxino A13-micro :

```
# ldd gpio_sleep
        libc.so.0 => /lib/libc.so.0 (0xb6f56000)
        ld-uClibc.so.0 => /lib/ld-uClibc.so.0 (0xb6fac000)
```

- Ici, uClibc fournit ces fonctionnalités.
- En cas d'oubli de ces bibliothèques dynamiques, le message d'erreur cryptique

  ```
  # ./gpio_sleep
  -sh: ./gpio_sleep: not found
  ```

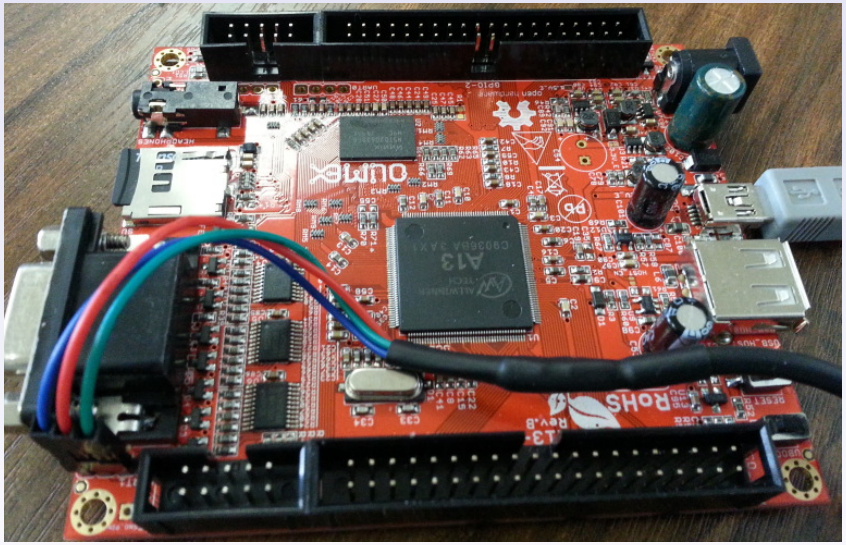
  ici dû à l'utilisation du mauvais compilateur (absence des bibliothèques dynamiques)

  ```
  # ldd gpio_sleep
  checking sub-depends for 'not found'
          libc.so.6 => not found (0x00000000)
          /lib/ld-linux.so.3 => /lib/ld-linux.so.3 (0x00000000)
  # ls -l /lib | grep ld-l
  #
  ```

Environnement de développement

Environnement de développement complexe car doit fournir des outils cohérents pour

- compiler le noyau Linux
- compiler les bibliothèques nécessaires aux outils en espace utilisateur
- compiler les exécutables ("paquets")
- compiler le *bootloader* et fichiers de configuration de la plateforme



Olinuxino A13-micro : système d'exploitation et bootloader sur carte SD

Objectifs

Un environnement cohérent de développement pour générer

1. la toolchain de cross-compilation
2. le bootloader (uboot : initialisation CPU + chargement noyau)
3. le noyau du système d'exploitation (Linux)
4. le rootfs (programmes en espace utilisateur)

qui se trouveront en divers emplacements de la carte SD.

La très grande majorité des dispositifs embaqués sont non-x86 [2] ⇒ cross-compilation des programmes de l'hôte vers la cible

2. http://iqjar.com/jar/an-overview-and-comparison-of-todays-single-board-micro-computers/

Environnements de développement

1. Buildroot
2. Openembedded
3. Yocto

Pour Olinuxino A13-micro :
`https://github.com/trabucayre/buildroot`

6 GB d'espace disque pour une images de 200+ MB dans
`output/images/a13_olinuxino.sdimg`

Voir dans configs les plateformes supportées :
`a13_olinuxino_micro_defconfig` donc

`make a13_olinuxino_micro_defconfig && make` [3]

3. `http://jmfriedt.free.fr/A13_v2.pdf`

Buildroot

`make menuconfig` pour configurer buildroot (outils espace utilisateur)
`make linux-menuconfig` pour configurer le noyau (support USB p.ex – le noyau se trouve dans `output/build/linux*`)
Organisation de l'arborescence :

1. `configs/*defconfig` : configuration de buildroot
2. `board/a13*/*defconfig` : configuration du noyau Linux
3. `output` contient tous les résultats de la compilation
4. `output/host/usr/bin/` pour la toolchain (host = PC)
5. `output/target/` contient les fichiers pour la cible ARM
6. `output/target/lib` contient les bibliothèques dynamiques du système embarqué
7. `output/build/linux-*` : source du noyau Linux
8. `output/build/linux-*/arch/arm/boot` : noyau Linux compilé
9. `output/images/` : image à flasher sur support non-volatil (`dd`)

Gestion de paquets : ajout dans packages [4]

4. `http://www.linuxembedded.fr/2011/03/ajouter-un-package-dans-buildroot-en-5-minutes/`

Mise en pratique

Découverte de la carte Olinuxino A13-micro [5]
Accès au matériel depuis l'espace utilisateur [6]

- configuration réseau TCP/IP (`ifconfig`, `route`)
- `cat < /dev/ttyUSB0`
- depuis le shell (`/sys/class/gpio`) [7] : `Device Drivers` → `GPIO Support`
- ⇒ html/web serveur/CGI

L'OS n'est pas toujours bien

- un OS doit booter : prend du temps et donc de l'énergie
- un OS nécessite de maîtriser des API
- un OS nécessite de la mémoire et de la puissance de calcul

⇒ bien peser les avantages (bibliothèques, contributions externes, stabilité des outils) et les contres avant de faire un choix.

5. `https://www.olimex.com/Products/OLinuXino/A13/A13-OLinuXino-MICRO/open-source-hardware`
6. `http://jmfriedt.free.fr/a13_userspace.pdf`
7. `https://www.kernel.org/doc/Documentation/gpio/sysfs.txt`