La boîte à outils libres pour l'embarqué

Patrice Kadionik, Maître de Conférences à l'ENSEIRB (kadionik@enseirb.fr)

Décembre 2005

Introduction

Cet article se propose de faire la revue des principaux outils libres adaptés à l'embarqué.

Ils sont répertoriés suivant les différentes étapes de conception et de fonctionnement d'un système embarqué. Les principales caractéristiques de ces outils seront détaillées sans toutefois préciser leur mise en oeuvre. Certains le seront indirectement au travers des différents articles de ce Hors Série.

La boîte à outils libres pour l'embarqué. Une classification

Quand on désire mettre en oeuvre des logiciels libres autour du noyau Linux pour l'embarqué, il existe un certain nombre de contraintes importantes imposées par le matériel :

- La mémoire étant une ressource à optimiser pour réduire le coût du système embarqué (on n'a pas à disposition des centaines de Mo comme avec un PC classique), il convient d'avoir des outils occupant peu de mémoire. On dit qu'ils sont à faible empreinte mémoire.
- Ils doivent éventuellement pouvoir s'exécuter directement en mémoire ROM (FLASH ou EEPROM) sans recopie préalable en RAM pour optimiser l'utilisation de la mémoire RAM. On dit qu'ils supportent le XIP (*eXecute in Place*).
- Ils doivent fournir les mêmes fonctionnalités que leurs homologues pour Linux « classique ».
- Ils doivent être optimisés en temps d'exécution.

Ces outils peuvent être classifiés suivant les différentes étapes de conception et de fonctionnement du système embarqué. On s'intéressera donc à l'aspect logiciel et non matériel, par exemple aux outils libres pour concevoir le matériel (conception du PCB, conception numérique...).

Cette classification se base ainsi sur le déroulement de vie d'un système sous Linux embarqué :

- La compilation croisée des sources des outils libres.
- Les bibliothèques *libc* à faible empreinte mémoire.
- Le debug des outils libres et des applications.
- Le noyau Linux pour l'embarqué : liste des différents portages disponibles. Il est possible suivant les portages/distributions, de générer aussi le système de fichiers *root* (*root File System*) en même temps. Si cela, n'est pas proposé, des outils libres comme *BusyBox* se proposent de le faire.
- Lancement du noyau : le *bootloader* Linux remplit ce rôle.
- Les outils libres.

Les outils libres peuvent être rangés à leur tour dans différentes catégories :

- Shells et commandes Linux de base.
- Editeurs de texte.
- Outils réseaux et connectivité IP.
- Interfaces graphiques.
- Emulateurs de processeurs.
- Autres.

Outils libres : mise en oeuvre

La mise en place d'un environnement de développement croisé est nécessaire. On trouve deux éléments indispensables :

- le PC de développement sous Linux. C'est la machine hôte ou *host*. Un compilateur C/C++ généralement est installé. C'est un compilateur croisé capable de générer du code exécutable par le système embarqué à développer.
- Le système embarqué à développer. C'est la machine cible ou *target*. Il exécutera le noyau Linux, les utilitaires et les applications spécifiques déployées.

La machine hôte et la machine cible sont reliées au minimum par l'intermédiaire d'une liaison série pour le téléchargement, la programmation et le debug de la cible. La liaison série sert classiquement comme sous Linux de port par défaut où sont envoyées les traces du noyau Linux embarqué.

Malheureusement, la liaison série, même programmée à 115,2 kb/s, est trop lente pour servir comme moyen de téléchargement de fichiers de plusieurs Mo. Cela peut prendre de quelques minutes à quelques dizaines de minutes.

On préfère donc utiliser une liaison Ethernet bien plus rapide : le téléchargement durera quelques dizaines de secondes. Cela implique d'avoir une connexion Ethernet présente sur la carte cible. Le moniteur/*bootloader* de la carte cible doit pouvoir supporter le téléchargement de fichiers par Ethernet. Pour cela, il intègre un client TFTP (*Trivial FTP*) fonctionnant au dessus du protocole de transport UDP qui est facilement « romable » (adaptable sur mémoire morte ou flash) du fait d'être un protocole de transport non connecté et sans état.

La cible doit bien sûr pouvoir embarquer Linux. Pour cela, il faut que :

- Le processeur soit au moins un processeur 32 bits avec MMU (Linux) ou sans MMU (µClinux).
- Que la cible ait quelques Mo de mémoire RAM et ROM (FLASH pour la facilité de reprogrammation).

Dès lors, tout est possible pour utiliser des outils libres...

Outils libres : compilateurs croisés

A tout seigneur, tout honneur. Sans le compilateur GNU *gcc*, rien ne serait possible et Linux ne serait pas ! Merci RMS !

Etant fortement configurable et modulaire, il est possible à partir de ses sources de générer un

compilateur croisé pour pratiquement tous les processeurs utilisés dans l'embarqué. Les langages intéressants pour l'embarqué sont principalement le langage C et dans une moindre mesure, le langage C++. Pierre (Ficheux) a présenté dans le numéro 74 de juillet/août 2005 de Linux Magazine comment générer un tel compilateur croisé. On pourra soit :

- Générer *from scratch* le compilateur croisé à la main ou à l'aide de l'outil *crosstool*.
- Utiliser une chaîne de compilation croisée déjà générée et précompilée, ce qui est préférable.

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
gcc	http://www.gnu.org/software/gcc/	Compilateur C, C++, Objective-C, Fortran...
crosstool	http://www.kegel.com/crosstool/	Générateur automatique d'un compilateur C, C++ et la bibliothèque *glibc* pour processeurs alpha, ARM, x86, IA64, MIPS, PowerPC, SH4...
Buildroot	http://buildroot.uclibc.org/	Générateur automatique d'un compilateur C, C++ et la bibliothèque *µClibc* pour processeurs ARM, MIPS, PowerPC ainsi que le *root File System*
Scratchbox	http://www.scratchbox.org/	Générateur automatique d'un compilateur C, C++ et la bibliothèque *glibc* ou *µClibc* pour processeurs ARM, x86, PowerPC, MIPS
ELDK	ftp://mirror.switch.ch/mirror/eldk/eldk/	Kit *Embedded Linux Development Kit* de la société DENX. Contient les compilateurs croisés précompilés pour processeurs PowerPC, ARM, MIPS Documentation : http://www.denx.de/wiki/DULG/Manual
	http://www.codesourcery.com/gnu_toolchains/arm/	Compilateur croisé précompilé pour processeurs ARM
	ftp://ftp.handhelds.org/projects/toolchain/	Compilateur croisé précompilé pour processeurs ARM
	http://www.linux-mips.org/wiki/MIPS_SDE_Installation	Compilateur croisé précompilé pour processeurs MIPS
	http://www.uclinux.org/pub/uClinux/m68k-elf-tools/	Compilateur croisé précompilé pour processeurs m68k, ColdFire et ARM
	http://www.uclinux.org/pub/uClinux/ports/h8/	Compilateur croisé précompilé pour processeurs H8

Outils libres : bibliothèques libc

La bibliothèque *libc* permet d'avoir accès aux appels systèmes standards d'E/S de Linux (*stdio.h*). Il convient d'avoir une *libc* à faible empreinte mémoire. La *libc* standard est celle du projet GNU

dénommée *glibc*. Malheureusement, elle n'est pas adaptée à l'embarqué car elle n'est pas à faible empreinte mémoire.

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
glibc	http://www.gnu.org/software/libc/	*libc* standard GNU. Peu adaptée à l'embarqué
µClibc	http://www.uclibc.org/	*libc* pour l'embarqué
newlib	http://sources.redhat.com/newlib/	*libc* pour l'embarqué
dietlibc	http://www.fefe.de/dietlibc/	*libc* pour l'embarqué

Outils libres : debug

Ces outils permettent de debugger le noyau Linux et les applications embarquées. L'outil debugger roi est le GNU *gdb*.

Il est bien sûr possible, si l'on a les outils adéquats, d'utiliser un émulateur ICE (*In Circuit Emulator*) qui reste un outil performant mais cher. Si l'on possède par exemple une interface de debug matérielle ICD (*In Circuit Debugger*) comme le BDM (*Background Debug Module*), il est possible de la coupler à *gdb*.

Dans le cas du debug d'une application, 2 possibilités s'offrent au développeur :

- Utiliser le port série de liaison entre l'hôte et la cible, couplé à *gdb*.
- Utiliser le réseau Ethernet. Dans ce cas, une application serveur *gdbserver* est lancée sur la cible pour debugger l'application. On utilisera alors l'application *gdb* comme application cliente pour se connecter ensuite à l'application *gdbserver* de la cible.

L'outil *gdb*, bien que puissant, s'utilise en mode lignes de commande très austères. Il est possible de le coupler à une interface graphique servant de *front end* pour pouvoir l'utiliser de façon plus conviviale.

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
gdb	http://www.gnu.org/software/gdb/gdb.html	Debugger GNU. Une version « cross compilée » est généralement fournie avec les chaînes de compilation croisée précompilées
DDD	http://www.gnu.org/software/ddd/	*Data Display Debugger*. *Front end* à *gdb*

Outils libres : noyau Linux pour l'embarqué

Le noyau Linux standard est initialement conçu pour les processeurs de la famille x86. La version courante du noyau (2.6) contient néanmoins le portage pour les processeurs suivants : alpha, ARM, H8300, IA64, m68k, m68knommu, MIPS, PowerPC, S390, SH, SH64, SPARC, SPARC64, V850,

x86_64...

Mais l'on doit souvent patcher les sources du noyau Linux officiel pour tenir compte des particularités spécifiques à un type de processeur dans une famille donnée.

Il vaut mieux alors s'orienter vers des noyaux patchés et adaptés à son type de processeur et prêts à l'emploi.

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
Noyau Linux	http://www.linux.org/	Noyau Linux standard
Noyau µClinux	http://www.uclinux.org/pub/uClinux/uClinux-2.0.x/ http://www.linux-m68k.org/	Noyau µClinux pour processeurs m68k
Noyau µClinux	http://www.uclinux.org/ports/coldfire/	Noyau µClinux pour processeurs ColdFire
Noyau µClinux	http://www.uclinux.org/pub/uClinux/ports/arm7tdmi/	Noyau µClinux pour processeurs ARM7TDMI
Noyau µClinux	http://blackfin.uclinux.org/	Noyau µClinux pour processeurs Blackfin
Noyau µClinux	http://www.uclinux.org/pub/uClinux/ports/h8/	Noyau µClinux pour processeurs H8
Noyau Linux	http://www.denx.de/en/Software/WebHome	Noyau Linux pour processeurs ARM, PowerPC et MIPS (ELDK)
Noyau Linux	http://www.arm.linux.org.uk/	Noyau Linux pour processeurs ARM
Noyau Linux	http://www.alphalinux.org/	Noyau Linux pour processeurs alpha
Noyau Linux	http://tsukuba.m17n.org/linux-sh/	Noyau Linux pour processeurs superH
Noyau Linux	http://www.linux-mips.org/	Noyau Linux pour processeurs MIPS
Noyau µClinux	http://www.niosforum.com	Noyau µClinux pour processeurs *softcore* NIOS II. S'inscrire auparavant
Noyau µClinux	http://www.itee.uq.edu.au/~jwilliams/mblaze-uclinux/	Noyau µClinux pour processeurs *softcore* Microblaze

Outils libres : bootloaders

Le *bootloader* est avant tout un moniteur ROM qui permet d'initialiser la carte cible et d'avoir accès à des commandes de debug de base :

- Lecture/écriture mémoire.
- Assemblage/désassemblage en ligne.
- Téléchargement de fichiers images (fichiers au format binaire, Motorola Srecord ou Intel) par la liaison série ou par l'interface réseau Ethernet de la carte cible.
- Lancement de programme et debug avec points d'arrêt et mode pas à pas.

- Programmation de mémoire ROM de type FLASH (avec par exemple l'image d'un noyau Linux embarqué et son *root File System*).

La fonctionnalité la plus intéressante est celle de *bootloader* qui permet de lancer un noyau Linux et de lui spécifier son *root File System* installé généralement en mémoire FLASH. Ce dernier peut être aussi éventuellement monté par NFS (*Network File System*).

Le *bootloader* Linux le plus universel et le plus intéressant est sans aucun doute *u-boot* (*Universal boot*).

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
u-boot	http://sourceforge.net/projects/u-boot	*Universal Boot. Bootloader* Linux pour processeurs PowerPC, ARM, x86, MIPS, Xscale, ColdFire 52x2,PXA... et processeurs *softcore* MicroBlaze, NIOS I et NIOS II
colilo	http://www.reasonability.net/uclinux/colilo/	*Bootloader* Linux pour processeurs ColdFire
ppcboot	http://sourceforge.net/projects/ppcboot/	*Bootloader* Linux pour processeurs PowerPC. Intégré dans u-boot
ARMboot	http://sourceforge.net/projects/armboot	*Bootloader* Linux pour processeurs ARM
YAMON	http://www.mips.com/sitemap/content/Products/SoftwareTools/content_html/yamon	*Bootloader* Linux pour processeurs MIPS
MRB	http://www.cybertec.com.au/mrb.html	*My Right Boot*. *Bootloader* Linux pour processeurs ColdFire
RedBoot	http://ecos.sourceware.org/redboot/	*Bootloader* Linux pour processeurs ARM, CalmRISC, FR-V, H8, IA32, 68k, Matsushita AM3x, MIPS, NEC V8xx, PowerPC, SPARC, SuperH. S'utilise avec eCos
OpenBIOS	http://www.openbios.info/	BIOS pour carte mère à base de processeurs x86, Alpha et AMD64
FreeBIOS	http://freebios.sourceforge.net/	BIOS pour carte mère à base de processeurs x86
PXELinux	http://syslinux.zytor.com/pxe.php	*Bootloader* Linux par le réseau Ethernet conforme à la norme PXE pour carte mère de PC à base de processeurs x86
EtherBoot	http://etherboot.sourceforge.net/	*Bootloader* Linux par le réseau Ethernet pour carte mère de PC à base de processeurs x86

Outils libres : shells et commandes Linux de base

L'interpréteur de commandes Linux ou *shell* est indispensable pour une interaction et la configuration de la carte cible. Il doit être à faible empreinte mémoire. Le véritable couteau suisse pour Linux embarqué est *BusyBox*.

L'outil *BusyBox* :

- Est un outil libre à faible empreinte mémoire.
- Fournit un shell (*ash, lash*).
- Fournit toutes les commandes Linux de base via un seul exécutable et des liens symboliques.
- Peut générer un système de fichiers *root* avec des scripts simples d'initialisation.
- Se compile avec la *glibc* ou *µClibc*.
- Intègre un serveur web, un client/serveur DHCP, des outils réseaux et fournit les commandes de gestion de modules Linux...

La liste des commandes et utilitaires fournis est impressionnante (http://busybox.net/downloads/BusyBox.html) :

`addgroup, adduser, adjtimex, ar, arping, ash, awk, basename, bunzip2, busybox, bzcat, cal, cat, chgrp, chmod, chown, chroot, chvt, clear, cmp, cp, cpio, crond, crontab, cut, date, dc, dd, deallocvt, delgroup, deluser, devfsd, df, dirname, dmesg, dos2unix, dpkg, dpkg-deb, du, dumpkmap, dumpleases, echo, egrep, env, expr, false, fbset, fdflush, fdformat, fdisk, fgrep, find, fold, free, freeramdisk, fsck.minix, ftpget, ftpput, getopt, getty, grep, gunzip, gzip, halt, hdparm, head, hexdump, hostid, hostname, httpd, hush, hwclock, id, ifconfig, ifdown, ifup, inetd, init, insmod, install, ip, ipaddr, ipcalc, ipcrm, ipcs, iplink, iproute, iptunnel, kill, killall, klogd, lash, last, length, linuxrc, ln, loadfont, loadkmap, logger, login, logname, logread, losetup, ls, lsmod, makedevs, md5sum, mesg, mkdir, mkfifo, mkfs.minix, mknod, mkswap, mktemp, modprobe, more, mount, msh, mt, mv, nameif, nc, netstat, nslookup, od, openvt, passwd, patch, pidof, ping, ping6, pipe_progress, pivot_root, poweroff, printf, ps, pwd, rdate, readlink, realpath, reboot, renice, reset, rm, rmdir, rmmod, route, rpm, rpm2cpio, run-parts, rx, sed, seq, setkeycodes, sha1sum, sleep, sort, start-stop-daemon, strings, stty, su, sulogin, swapoff, swapon, sync, sysctl, syslogd, tail, tar, tee, telnet, telnetd, test, tftp, time, top, touch, tr, traceroute, true, tty, udhcpc, udhcpd, umount, uname, uncompress, uniq, unix2dos, unzip, uptime, usleep, uudecode, uuencode, vconfig, vi, vlock, watch, watchdog, wc, wget, which, who, whoami, xargs, yes, zcat`

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
BusyBox	http://busybox.net/	Véritable couteau suisse ! *BusyBox* est incontournable aujourd'hui
ash	http://www.uclibc.org/http://packages.debian.org/unstable/shells/ash.html	*Shell* (intégré dans *BusyBox*)

Outils libres : éditeurs de texte

L'éditeur de texte intégré dans le système cible est très pratique pour pouvoir faire des modifications en cours de fonctionnement sur les fichiers de configuration.

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
vi	http://busybox.net/	vi est dans *BusyBox*
ae	http://www.nyangau.fsnet.co.uk/	*Andy's Editor*. Editeur
e3	http://www.sax.de/~adlibit/	Editeur compatible *vi*, *emacs* et *Wordstar*
elvis	http://elvis.the-little-red-haired-girl.org/	Editeur compatible *vi*
nano	http://www.nano-editor.org/	Editeur
jove	http://packages.debian.org/unstable/editors/jove	Editeur compatible *emacs*

Outils libres : outils réseaux et connectivité IP

Les outils réseaux et de connectivité IP concernent toutes les applications mettant en oeuvre les protocoles IP. Cela comprend principalement tous les serveurs web embarqués qui servent à piloter le système embarqué à distance.

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
boa	http://www.boa.org/	Serveur web. Le serveur web pour l'embarqué. Support des scripts CGI boa+php pour µClinux disponible ici : http://www.menie.org/georges/uClinux/boa-php.html
thttpd	http://www.acme.com/software/thttpd/	*tiny/turbo/throttling HTTP server*. Serveur web
busybox	http://busybox.net/	*BusyBox* contient un serveur web
lightTPD	http://lighttpd.net/	Serveur web. Support des scripts CGI
dillo	http://www.dillo.org/	Navigator web. Mode graphique
minimo	http://www.mozilla.org/projects/minimo/	Navigator web. Mode graphique
iproute	http://packages.debian.org/unstable/net/iproute.html	Outil pour la configuration réseau

Outil	Adresse Internet	Description
ntpclient	http://doolittle.faludi.com/ntpclient/	Client NTP
dropbear	http://www.ucc.gu.uwa.edu.au/~Ematt/dropbear/dropbear.html	Client et serveur SSH. Peut être compilé avec µClibc
tinylogin	http://tinylogin.busybox.net/	Outil pour gérer le contrôle d'accès au système embarqué : login, passwd, getty...
udhcp	http://udhcp.busybox.net/	Client et serveur DHCP (intégré dans *BusyBox*)

Outils libres : interfaces graphiques

Certains systèmes embarqués peuvent intégrer un environnement graphique pour avoir des IHM graphiques (*Graphical User Interface*). Si le circuit graphique intégré dans le système embarqué le supporte, il est possible d'utiliser le mode *Frame Buffer* disponible par défaut avec le noyau Linux. Il est aussi possible de mettre en place des environnements graphiques du type X11.

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
Frame Buffer	http://www.linux.org	Intégré au noyau Linux
DirectFB	http://www.directfb.org/	Environnement graphique tournant au-dessus du *Frame Buffer*
Nano-X	http://www.microwindows.org/	Environnement graphique tournant au-dessus du *Frame Buffer*, X Window, SVGAlib. Il existe 2 API (*Application Programming Interface*) : - Nano-X : API compatible Xlib - Win32 : API compatible Windows Win32
Minigui	http://www.minigui.org/	Environnement graphique tournant au-dessus du *Frame Buffer*
FLTK	http://fltk.org/	Environnement graphique tournant au-dessus de X11/Xlib avec OpenGL
SDL	http://www.libsdl.org/index.php	Environnement graphique tournant au-dessus du *Frame Buffer*
OPIE	http://opie.handhelds.org/	*Open Palmtop Integrated Environment.* Environnement graphique tournant au-dessus du *Frame Buffer*
GPE	http://gpe.handhelds.org/	*GPE Palmtop Environment*. Environnement graphique tournant au-dessus du X11

Outils libres : émulateurs

Les émulateurs permettent sur la machine hôte d'émuler le processeur de la machine cible. Il est même possible aussi d'émuler le système Linux de la machine cible dans sa globalité sur la machine hôte !

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
qemu	http://qemu.org/	Emulateur pour processeurs x86, ARM, SPARC et PowerPC. Peut émuler un système Linux pour processeurs x86, x86_64 et PowerPC
UML	http://user-mode-linux.sourceforge.net/	Emulateur d'un système Linux pour processeurs x86
Coldfire Emulator	http://www.slicer.ca/coldfire/	Emulateur pour processeurs ColdFire. Peut émuler un système µClinux
SkyEye	http://skyeye.sourceforge.net/index.shtml	Emulateur pour processeurs ARM. Peut émuler un système Linux ou µClinux
Softgun	http://softgun.sourceforge.net/	Emulateur pour processeurs ARM. Peut émuler un système Linux
SWARM	http://www.cl.cam.ac.uk/~mwd24/phd/swarm.html	Emulateur pour processeurs ARM. Peut émuler un système µClinux

Outils libres : autres

Les outils suivants sont importants dans l'environnement Linux embarqué.

Le tableau suivant présente les principaux outils libres de cette rubrique :

Outil	Adresse Internet	Description
MTD	http://www.linux-mtd.infradead.org/	*Memory Technology Device*. Bibliothèque pour gérer la mémoire FLASH. Intégré au noyau Linux
JFFS2	http://sources.redhat.com/jffs2/	*Journalling Flash File System, version 2*. Système de fichiers pour mémoire FLASH
LTP	http://ltp.sourceforge.net/	*Linux Test Project*. Suite de tests d'un système Linux

Où trouver d'autres outils libres ?

Il est possible de trouver des outils libres sous licence GPL sur différents sites les hébergeant. Ils ne sont pas forcément adaptés à l'embarqué mais doivent pouvoir être portés sur la carte cible si l'on possède le compilateur croisé adéquat.

Le tableau suivant présente les principaux sites hébergeant des outils libres :

Site	Adresse Internet
GNU	http://www.gnu.org/software/
FRESHMEAT	http://freshmeat.net/
SOURCEFORGE	http://sourceforge.net/

Enfin, il reste les sites consacrés à un processeur cible donné qui contiennent de nombreuses ressources.

Le tableau suivant présente les principaux sites sur les processeurs cibles :

Processeur	Adresse Internet
m68k, ColdFire	http://www.linux-m68k.org/ http://www.uclinux.org/
ARM	http://www.arm.linux.org.uk/ http://www.at91.com/
PowerPC	http://penguinppc.org/embedded/
MIPS	http://www.linux-mips.org/
SH	http://linuxsh.sourceforge.net/ http://tsukuba.m17n.org/linux-sh/
SPARC	http://www.ultralinux.org/
ALPHA	http://www.alphalinux.org/

Conclusion

Nous avons pu voir les principaux outils libres disponibles pour l'embarqué. Il en existent bien sûr d'autres et consulter les sites comme *sourceforge.net* et *freshmeat.net* au préalable est un réflexe à acquérir.

De même, s'abonner à un *newsgroup* s'intéressant à l'outil libre choisi est plus qu'obligatoire car l'on retrouve généralement dans les archives du *newsgroup* la réponse à son problème.

Une chose est néanmoins certaine : dans l'embarqué, il est inutile de vouloir réinventer la roue !

Bibliographie

- La page outils pour Linux embarqué de l'auteur. http://www.enseirb.fr/~kadionik/embedded/embeddedlinux.html#OUTILS
- Free Software tools for embedded systems. M. Opdenacker, Free Electrons. http://free-electrons.com/training/devtools/en