



# **Strumenti di sviluppo per sistemi Linux embedded Laboratorio pratico**

*Michael Opdenacker*  
*Free Electrons*  
<http://free-electrons.com>

Traduzione italiana di G. Gusinu <http://gusinu.net>



## Corso completo

Questo documento fa parte di un corso su Linux embedded di Free Electrons.

Tutto il materiale per l'intero corso (presentazione, laboratorio pratico e altro) puo' essere consultato in linea o scaricato all'indirizzo <http://free-electrons.com/training/devtools>

## Diritti di riproduzione

© 2004, Michael Opdenacker, [michael@free-electrons.com](mailto:michael@free-electrons.com)

Questo documento e' disponibile secondo la licenza GNU Free Documentation License, senza variazioni. E' consentito copiare e modificare questo documento purché venga mantenuta la stessa licenza.

Per maggiori dettagli si veda <http://www.gnu.org/licenses/fdl.html>

Aggiornamenti del documento sono disponibili su <http://free-electrons.com/training/devtools>

Correzioni, suggerimenti e collaborazioni sono ben accetti!

## Aggiornamenti del documento

Salvo specificazione contraria tutte le modifiche sono dell'autore Michael Opdenacker.

28 sett. 2004. Prima edizione ufficiale.

20-24 sett. 2004. Prima presentazione per [Atmel](#), Rousset (Francia).

## Prima di iniziare

Istruzioni per il formatore e gli studenti senza supervisione.

L'ultima sessione di questa formazione e' stata condotta usando Knoppix GNU/Linux 3.6 (2004-08-16), con l'opzione di boot `linux26`. Si noti che i binari precompilati di `qemu` non funzionano con Linux 2.4, ecco perche' si e' fatto ricorso alla versione 2.6.

Scaricare la directory dei file esercizio da [http://free-electrons.com/labs/embedded\\_linux\\_labs.tar.bz2](http://free-electrons.com/labs/embedded_linux_labs.tar.bz2).  
Estrarre gli archivi nella directory home dello studente.

Scaricare la directory degli strumenti da [http://free-electrons.com/labs/embedded\\_linux\\_tools.tar.bz2](http://free-electrons.com/labs/embedded_linux_tools.tar.bz2).  
Estrarre l'archivio in `/mnt/tools`. Tale directory ha diritti di scrittura sotto Knoppix. Si puo' anche farne un link su una partizione montata sull'hard disk.

Attenzione! All'interno dell'archivio vi e' una toolchain che non puo' essere rilocata. E' obbligatorio estrarre l'archivio nella directory indicata (che puo' anche essere un link), altrimenti e' necessario ricompilare la toolchain con un prefisso diverso da quello standard.



## Lab 1 – Toolchain per cross compilazione

Obiettivo: apprendere come generare la propria toolchain per la cross compilazione

Durante queste lezioni pratiche si imparerà a:

- configurare gli strumenti crosstool
- eseguire i crosstool per generare la propria toolchain

### Setup

Andare nella directory `$HOME/labs/tools/lab1`

### Scaricare i crosstool

Scaricare l'ultima versione di sviluppo dei crosstool da <http://kegel.com/crosstool/>.

Estrarre l'archivio nella directory corrente.

### Setup di wget

I crosstool sono degli script che si occupano di scaricare i sorgenti delle applicazioni necessarie attraverso il comando `wget`.

Nel caso un proxy sia installato può rendersi necessario configurare `wget` perché possa attraversarlo, esattamente come farebbe un navigatore web.

In tal caso si crei il file `“.wgetrc”` nella propria directory home seguendo l'esempio seguente:

```
ftp_proxy = 124.125.126.127:5678
http_proxy = 124.125.126.127:5678
```

Si chiedi al formatore quali settaggi siano indicati per la rete usata.

### Setup dei crosstool

I crosstool vengono eseguiti attraverso uno degli script del tipo `demo-<arch>.sh`

Nel nostro esempio verrà generata una toolchain per ARM. Si edita dunque il file `“demo-arm.sh”`:

- verificare che `TARBALLS_DIR` sia impostata su `$HOME/downloads`
- settare `RESULT_TOP` con la directory `tools` nella directory home
- si scelga l'accoppiata `gcc-3.3.4 / glibc-2.3.2`

### Setup della toolchain

Ora si può modificare il file `“.dat”` al fine di usare i file header di Linux 2.6.7.

### Generare la toolchain

Verificare che la variabile d'ambiente `LD_LIBRARY_PATH` non sia impostata. Altrimenti eseguire: `unset LD_LIBRARY_PATH`

Si esegua `“sh demo-arm.sh”` tenendo d'occhio i messaggi. Al termine della procedura si esamina la struttura di directory creata a partire dalla directory `tools`.

In alternativa al file `“.wgetrc”` si possono usare le variabili d'ambiente `http_proxy` e `ftp_proxy`. Il vantaggio è che, essendo tali variabili standard per tutto il sistema, i settaggi saranno riconosciuti da altre applicazioni.

Normalmente i settaggi per i proxy `http` e `ftp` sono i medesimi.

Le toolchain non sono rilocabili: una volta generate non è possibile spostarle in un'altra directory.

In un caso reale si sceglie la “vera” destinazione finale!

In particolare si veda la directory `/mnt/tools/arm-unknown-linux-gnu/gcc-3.4.1-glibc-2.3.3/arm-unknown-linux-gnu/lib` contenente la libreria `glibc` compilata per la piattaforma ARM.

Tale procedura richiede normalmente diverse ore! È consigliabile non attendere passivamente che sia terminata, ma proseguire con l'esercizio pratico successivo.



Infine, non si dimentichi di cancellare le directory con i sorgenti della toolchain che altrimenti occuperebbero inutilmente molto spazio disco della propria directory home.

## Lab 2 – Cross compilazione di busybox

Obiettivo: apprendere come configurare, generare, testare e installare busybox per il dispositivo target.

Durante queste lezioni pratiche si imparerà a:

- conoscere capacità e opzioni di busybox
- configurare le opzioni di busybox
- configurare busybox per la cross compilazione
- compilare e testare busybox.

### Setup

Andare nella directory `$HOME/labs/tools/lab2/`.

### Scaricare e configurare busybox

Scaricare l'ultima versione di busybox su <http://busybox.net/>.  
Estrarre l'archivio e entrare nella directory dei sorgenti.

Configurare busybox scegliendo quali funzionalità includere.

```
make config
```

E' consigliabile leggere tutte le opzioni per avere un'idea di tutte le possibilità di busybox.

Per conoscere che taglia può avere il file eseguibile di busybox, si scelga il link statico (non usa libreria condivise). Si scelgano solo le funzioni per far partire e usare un sistema di base. All'occorrenza e' possibile accettare le opzioni di default.

Quando richiesto si scelga la cross compilazione di busybox.

Dare il seguente prefisso al cross compilatore:

```
/mnt/tools/arm-unknown-linux-gnu/gcc-3.3.4-glibc-  
2.3.2/bin/arm-unknown-linux-gnu-
```

Finalmente si procede a cross compilare busybox:

```
make  
make install
```

Ispezionare il contenuto della directory `_install`. Si tratta di una struttura completa di filesystem root che può essere usato per initrd o in piccoli sistemi dedicati.

### Testare busybox

Usando l'emulatore qemu e' addirittura possibile testare busybox prima ancora di installare sulla macchina target! E' sufficiente dare:

```
/mnt/tools/bin/qemu-arm -L /mnt/tools/arm-unknown-linux-gnu/gcc-3.3.4-  
glibc-2.3.2/arm-unknown-linux-gnu ./busybox ls
```

Quando la toolchain (lab1) e' pronta, e' possibile testarla. Basta ricompilare busybox con la toolchain appena generata e rieseguire la procedura di test. Eseguire `make clean` per assicurarsi che i sorgenti di busybox siano ricompilati.

Si noti l'analogia con la configurazione del kernel linux: le impostazioni di configurazione sono registrate in un file `“.config”` con sintassi Makefile.

Questo non e' un "prefisso" nel senso usato da "autoconf" (script di configurazione). In questo caso il "prefisso" corrisponde al path completo del file eseguibile del cross compilatore, fatta eccezione per la stringa "gcc" finale.

Si noti come tutti gli eseguibili siano in realta' dei link all'unico file eseguibile busybox. Quando invocato busybox verifica con quale nome sia stato invocato: `ls`, `cat`, `gzip`, etc.

In questo caso usiamo il modo "user" dell'emulatore qemu, che si fa carico di emulare il solo processore target e non l'intero sistema. Quando qemu e' eseguito in spazio utente ("userland") le funzioni di sistema ("system calls") sono tradotte in chiamate a funzioni di sistema native (Linux i386 per noi). Ecco perche' viene passata la glibc compilata per ARM come parametro.