



# **Sviluppo driver e kernel per Linux embedded Esercitazioni pratiche**

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

Traduzione di Giansalvo 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 è protetto secondo la licenza GNU Free Documentation License, senza sezioni invariati. E' consentito copiare e modificare questo documento purché si mantenga la licenza. Per maggiori dettagli consultare <http://www.gnu.org/licenses/fdl.html> .

Aggiornamenti del documento 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 è 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 perché si e' fatto ricorso alla versione 2.6.

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



## Lab 1 – Sorgenti del kernel

Obiettivo: apprendere come ottenere i sorgenti del kernel e applicare patch. Acquistare familiarità con i sorgenti.

Durante questa esercitazione si imparerà a:

- Scaricare i sorgenti del kernel dal sito ufficiale
- Verificare l'autenticità dei sorgenti del kernel
- Applicare le patch al kernel
- Esplorare i sorgenti alla ricerca di file, prototipi di funzione ed altre informazioni...
- Compilare un programma GNU/Linux a partire dai sorgenti nella maniera più comune.
- Configurare, compilare e fare il boot del kernel su un PC virtuale.
- Montare e modificare l'immagine del filesystem di root aggiungendo elementi alla directory /dev.

### Setup

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

### Scaricare i sorgenti

Andare sul sito web del kernel Linux (<http://www.kernel.org/>) e identificare l'ultima versione stabile.

Giusto per essere sicuri di saperlo fare, verificare la versione del kernel in uso sulla propria macchina.

Per l'esercitazione useremo linux-2.6.7, per il quale è stata testata.

Per esercitarsi con il comando patch in seguito, si scarichino i sorgenti completi della versione 2.6.6. Si scarichi inoltre la firma digitale GPG corrispondente.

### Verifica dell'autenticità dei sorgenti

Per prima cosa si esegua il comando seguente per caricare la chiave GPG public di Linux, come indicato in <http://www.kernel.org/signature.html> (verificare che l'identificatore della chiave non sia cambiato):

```
gpg --keyserver wwwkeys.pgp.net --recv-keys 0x517D0F0E
```

Ora si verifichi l'autenticità dei sorgenti scaricati.

Una volta fatto ciò, si può scompattare l'archivio creando una directory linux-2.6.6.

### Applicare le patch

Scaricare la patch corrispondente alla versione 2.6.7. Verificarne l'autenticità.

Scorrere il file di patch con `vi` o `gvim` (se si preferisce un editor grafico), per capire che tipo di informazioni contiene. Come viene indicata la rimozione o l'aggiunta di un file?

Si applichi la patch alla directory linux-2.6.6.

Prestare attenzione al significato dei link F, V, VI, C su questa pagina

Per comodità conviene copiare l'indirizzo URL dei sorgenti dal browser web e quindi usare il comando `wget` per scaricare i sorgenti::

```
wget <url>  
wget <url>.sign
```

`wget` è capace di riprendere download interrotti

Non preoccuparsi dell'avvertimento concernente la firma digitale! La firma digitale ci permette almeno di confermare che i sorgenti sono stati firmati da kernel.org. La firma potrebbe essere falsa, per esempio qualcuno non autorizzato potrebbe essersi infiltrato nel sito, aver modificato il kernel e aver creato una firma digitale con una chiave diversa. L'istruttore potrà spiegare l'argomento più in dettaglio se necessario.

Vim supports syntax highlighting for patch files



Rinominare la directory `linux-2.6.6` in `linux-2.6.7`.

Si applichi anche la patch contenuta in `data/kernel.patch`.

Corregge un bug minore ma fastidioso nel Makefile principale. Tale bug è corretto nell'ultima versione.

## Prendere familiarità con i sorgenti

In qualità di utilizzatore del kernel Linux, si ha spesso bisogno di trovare quale file implementa una certa funzione. Per far ciò è utile aver familiarità con i sorgenti del kernel.

1. Cercare il logo Linux nei sorgenti.
2. Cercare chi è il mainainer del driver di rete 3C505.
3. Cercare la home page del team che sviluppa la porta parallela.
4. Cercare la dichiarazione della funzione `platform_device_register()`. Cercare anche il file che la implementa.

## Utilizzo di strumenti automatici

Ora che sappiamo fare le cose manualmente possiamo agli strumenti automatici.

Provare LXR (Linux Cross Reference) a <http://lxr.linux.no/>. Seguire il link "Browse the code" e cliccare sulla versione del kernel più vicina a quella usata.

Come già fatto nella sezione precedente, si cerchi in quale file è dichiarata, implementata e richiamata la funzione `platform_device_register()`.

Ovviamente se il kernel usato contiene molte parti personalizzate, oppure se non si ha una connessione Internet agevole, è possibile eseguire LXR sul proprio computer.

## Configurazione del kernel

Eeguire `make xconfig` per lanciare `qconf`, la nuova interfaccia di configurazione del kernel.

Nell'interfaccia di `qconf` si attivi l'opzione `Option -> Show Name`. A volte questo può essere utile, per esempio quando il nome del parametro è più esplicito della sua descrizione, oppure quando si seguono delle istruzioni che fanno riferimento al nome del parametro.

Si provi anche l'opzione `Option -> Show All Options`. In questo modo si possono vedere tutti i parametri per i quali le precondizioni non siano soddisfatte a causa dell'architettura della macchina oppure perché un altro parametro non è impostato correttamente. Cliccando sulla descrizione del parametro si vedono le sue precondizioni e si può capire perché non sia attivo.

Si proceda a configurare il kernel in maniera minimalistica per un PC con disco IDE e boot da console framebuffer con filesystem `ext2`. Si disattivino tutti gli altri tipi di filesystem e periferiche particolari (SCSI, USB, video, graphics, Bluetooth, IrDA, networking...).

Non si esiti a chiedere al proprio istruttore dettagli approfonditi circa una certa opzione o in caso di dubbi sull'attivazione o meno di un'opzione.

## Compilare il kernel

Si esegua:

```
make
```



## Scaricare un emulatore di PC

Il modo più facile di testare il proprio kernel senza dover riavviare continuamente è di usare un emulatore.

qemu (<http://fabrice.bellard.free.fr/qemu/>) è uno strumento veloce che può emulare vari processori (x86, ppc, arm, sparc) o anche sistemi completi (per il momento solo PC x86 e PowerMac o PREP PowerPC).

Per comodità Qemu è disponibile sul server in `/mnt/tools/bin`.

Quindi conviene aggiungere questa directory alla variabile d'ambiente PATH: `export PATH=/mnt/tools/bin:$PATH`

## Boot del kernel

Siamo ora pronti a (provare a) fare il boot del nuovo kernel. Useremo il file `data/linux_i386.img` come filesystem di root. Eseguire:

```
qemu -m 32 -kernel linux-2.6.7/arch/i386/boot/bzImage \  
-append "clock=pit root=/dev/hda" \  
-hda data/linux_i386.img -boot c
```

Se il boot non ha successo e se il messaggio di errore è abbastanza specifico, si provi a modificare opportunamente la configurazione e rigenerare il kernel. Ovviamente la procedura sarà più veloce poiché il file non coinvolti nella modifica non saranno ricompilati.

Non si esiti a mostrare i problemi all'istruttore.

Se si rimane bloccati, è sempre possibile provare il file di configurazione `data/.config` che dovrebbe essere stato verificato dall'istruttore.

Se tutto va bene si dovrebbe arrivare al messaggio seguente:

```
Warning: unable to open an initial console.
```

Questo succede perché nel filesystem root non è presente un device console, ed è semplice da correggere.

Digitare `quit` nella console qemu per uscire dall'emulatore.

## Aggiunta di un device console nel filesystem di root

Fare il login come root (`su -`), eseguire il comando seguente per montare il filesystem di root:

```
mkdir /mnt/<user>_rootfs  
cd ~knoppix/labs/linux/lab1/  
mount -o loop data/linux_i386.img /mnt/<user>_rootfs/
```

Stiamo usando una directory dipendente dall'utente come destinazione per evitare problemi dovuti all'uso della stessa macchina da parte di più studenti.

Sempre collegati come root, si crei il device mancante `dev/console`. Si può verificare il tipo e i numeri di versione maggiore e minore del file `/dev/console` sulla macchina usata per le lezioni.

Una volta terminato, si deve smontare il filesystem root:

```
cd  
umount /mnt/<user>_rootfs
```

Si riesegua il comando `qemu`. Si dovrebbe ottenere il messaggio:

`clock=pit` è usato come soluzione di ripiego in caso di problemi dovuti all'emulazione dell'orologio macchina in qemu

Quest'ultima affermazione non è sempre vera. Se le modifiche ai file di configurazione sono molte, un numero importante di file sorgenti potrebbero venire ricompilati per evitare problemi legati a cambiamenti di dipendenze

Per creare `/mnt/rootfs` entry e anche per eseguire il comando `mount` bisogna avere i permessi di root

I device di loop sono immagini di filesystem che il kernel può montare come ogni altro filesystem su un device fisico. Per esempio, lo si può usare per accedere al contenuto di un cdrom ISO senza registrare il cdrom.

Indipendentemente dall'architettura sulla quale gira Linux, i numeri di device maggiore e minore sono gli stessi.

Se non si esegue `umount` o non si usano speciali opzioni di `mount` non si può essere certi che le modifiche siano state registrate sul dispositivo fisico (in questo caso il file `.img`)

`cd` è usato per uscire dalla directory `/mnt/rootfs`. Altrimenti, il sistema non consentirebbe di eseguire `umount` perché la directory sarebbe ancora in uso.



Please press Enter to activate this console.

Press Enter and run a few commands in the virtual PC shell.

## Conclusioni

Complimenti! Adesso sai come configurare, compilare e fare il boot del kernel per un PC di base.

## Lab 2 – Cross compilazione

Obiettivo: imparare a cross-compilare un kernel per la piattaforma target.

---

Durante questa esercitazione si imparerà a:

- Cross-compilare il kernel per la piattaforma ARM, molto diffusa
- Configurare l'ambiente di cross compilazione corrispondente

### Setup

Andare in `labs/tools/lab2` nella propria directory home.

### Scaricare i sorgenti

Il nostro obiettivo è di compilare linux per la piattaforma ARM.

Scaricare l'ultima patch da <http://maxim.org.za/AT91RM9200/2.6/> .

Scaricare anche i sorgenti corrispondenti per il kernel Linux 2.6.

Applicare la patch ai sorgenti. Se la versione è ancora Linux 2.7, applicare anche la patch `../lab1/data/kernel.patch`.

### Setup dell'ambiente di cross compilazione

Per cross compilare Linux, occorre aggiungere la toolchain alla variabile d'ambiente PATH:

```
export PATH=/mnt/tools/arm-unknown-linux-gnu/gcc-3.3.4-  
glibc-2.3.2/bin:$PATH
```

### Setup del Makefile

Modificare il Makefile principale per cross compilare per la piattaforma arm usando la toolchain sucitata

### Configurazione del kernel Linux

Usare la configurazione di default di 2.6 per AT91RM9200.

Non si esiti a visualizzare i nuovi settaggi eseguendo `make xconfig!`

### Cross compilazione

Provare a compilare il proprio kernel. Si vedrà subito se ci siano problemi o meno con il setup.

Se l'esecuzione è andata a buon fine, dove è stata creata l'immagine del kernel Linux?



## Lab3 – Scrittura di un modulo

Obiettivo: imparare a scrivere e compilare un modulo

---

Durante questa esercitazione si imparerà a:

- Scrivere un modulo del kernel con molteplici funzionalità, come per esempio parametri e output su /proc.
- Accedere alle risorse interne del kernel dal proprio modulo.
- Setup dell'ambiente per compilarlo
- Copiare il modulo nella propria partizione di root.
- Aggiungere i sorgenti del proprio modulo a quelli del kernel e creare una patch al kernel a partire dai nuovi sorgenti..

### Setup

Andare in `labs/linux/lab3` nella propria home directory.

### Scrivere il modulo

Creare il file di implementazione di un modulo, `hello_version.c`, che visualizza il seguente messaggio quando viene caricato:

```
Salve Maestro. La versione di Linux usata è la <versione>.  
... e visualizza un messaggio al momento della terminazione  
(rimozione del modulo).
```

Attenzione: per ottenere informazioni sulla versione occorre usare una variabile o una funzione del kernel e non una semplice macro C. Altrimenti si otterrebbero solamente informazioni sul kernel usato per compilare il modulo.

### Compilare il modulo

Scrivere un makefile per generare il modulo all'esterno dei sorgenti del kernel. Si usi la stessa directory di build usata in `lab1`.

Compilare il makefile.

### Copiare il modulo sul filesystem di root

Copiare il modulo sul filesystem di root in `data/linux_i386.img`.

### Test del modulo

Si può usare il comando seguente (`run_qemu executable`)

```
qemu -m 32 \  
-kernel ../lab1/linux-2.6.7/arch/i386/boot/bzImage \  
-append "clock=pit root=/dev/hda rw" \  
-hda data/linux_i386.img -boot c
```

### Lista dei moduli

Si provi il comando `lsmod`. Si verifica un errore, perché?

Per risolvere il problema, si deve creare la directory `/proc` e farne il

Inizia con un modulo che stampa il messaggio di benvenuto e poi aggiungi le informazioni sulla versione.

Suggerimento: cerca i file sorgenti di Linux che contengono `version` nel loro nome e vedi cosa fanno.

Se usi `insmod` per caricare un modulo, allora è possibile copiare il modulo in qualsiasi punto del filesystem di root.

Un comando di `mount` un po' inusuale, no? Per capirlo occorre considerare il primo argomento `/proc` come una directory virtuale conosciuta al kernel Linux ma non disponibile nell'albero delle directory..



mount (non si può fare il mount senza prima aver creato la directory corrispondente):

```
mkdir /proc
mount -t proc /proc /proc
```

### Aggiungere un parametro al modulo

Si aggiunga il parametro “nome” al modulo. Il modulo stamperà “Ciao <nome>” al posto di “Salve Maestro”.

Compilare e testare il modulo verificando che il parametro venga preso in considerazione al momento del caricamento.

Prova a cercare nei sorgenti del kernel altri driver che usino la funzione `do_gettimeofday()`. Guardare altri esempi aiuta molto!

### Aggiungere informazioni sul tempo trascorso

Migliorare il modulo, in modo che quando viene rimosso, stampi il tempo trascorso dal momento del caricamento.

Per far ciò si può usare la funzione `do_gettimeofday()`.

### Aggiungere un interfaccia /proc

Aggiungere un'interfaccia in spazio utente, mettendo a disposizioni in tempo reale in `/proc/hello_version_elapsed`, informazioni sul tempo trascorso dal caricamento.

Per far ciò, si usi la funzione `create_proc_info_entry()`. Si possono trovare molti esempi del suo utilizzo un pò dappertutto nel kernel.

### Aggiungere hello\_version ai sorgenti del kernel

Spostare i sorgenti di linux 2.6.7 da `lab1/` a `lab3/`.

Aggiungere i sorgenti del modulo in una nuova directory `drivers/foo` dei sorgenti del kernel. Ovviamente occorre modificare i file di configurazione e di creazione del kernel opportunamente, in modo da poter selezionare il proprio modulo in `make xconfig` e poterlo compilare usando il comando `make`.

Compilare il nuovo modulo staticamente nel kernel e passare l'argomento “nome” alla linea di comando del kernel.

Eseguire il kernel e verificare che tutto funzioni!

### Creare una patch al kernel

Puoi essere fiero del tuo nuovo modulo! Per poterlo condividere con altri, si crei una patch che aggiunga i nuovi file al kernel 2.6.7 originale.

Testare la patch.

Complimenti per il buon lavoro!