

KERNEL LINUX

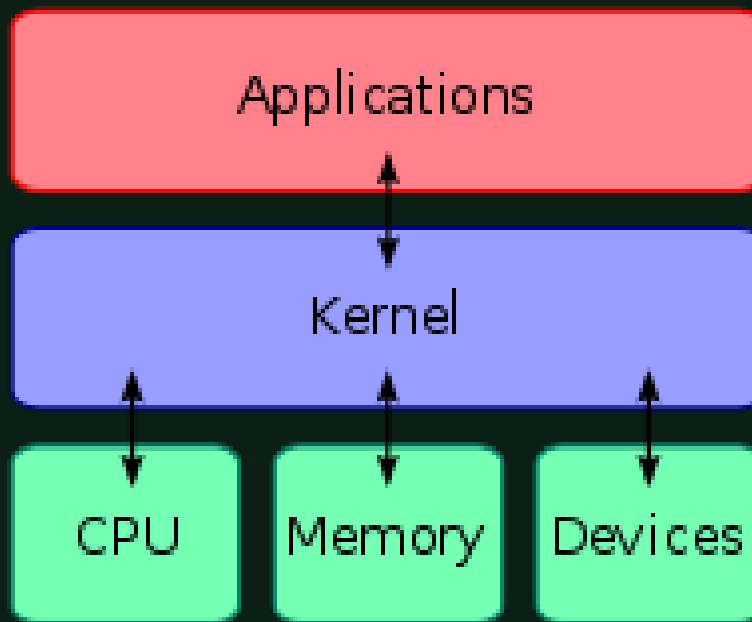
Ricompilazione ed Ottimizzazione

Relatore: Mirco Chinelli
Linux Day Torino 2009



Cos'e' un Kernel?

- Nucleo del Sistema Operativo
- Astrazione dell'hardware



Linux Day
Torino 2009

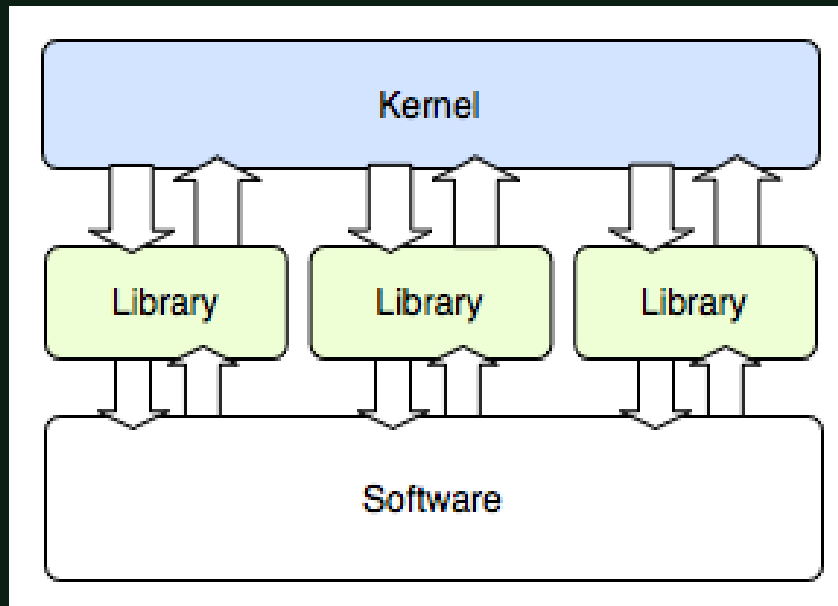
Tipi di Kernel

- Monolitico
- Microkernel
- Esokernel



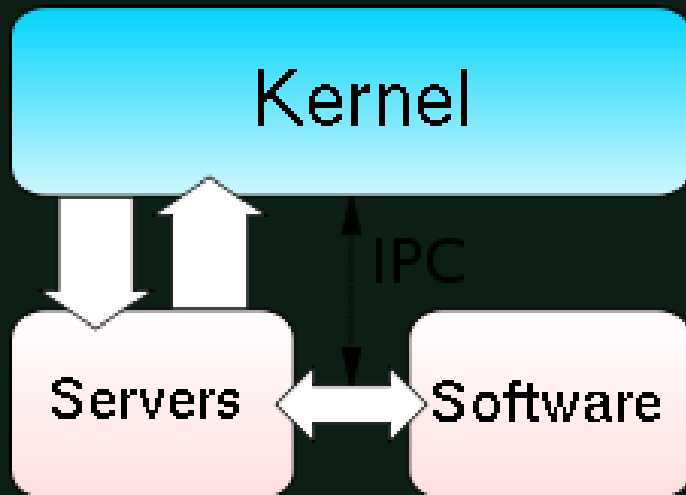
Esokernel

- Kernel minimale
- Astrazione minima dell'hardware
- Possibilita' di eseguire binari di diverso tipo



Microkernel

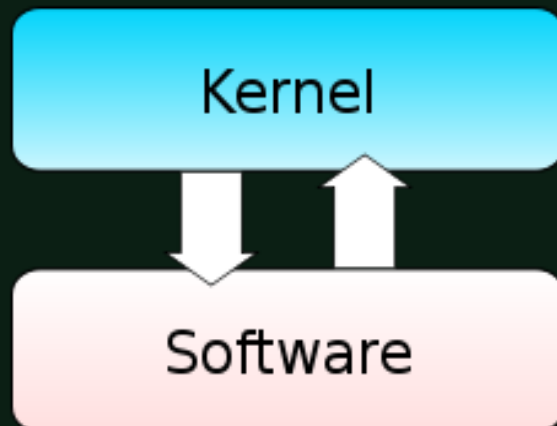
- Moduli(“pezzi di kernel”) a livello utente.
- Maggiore sicurezza
- Piu' stabilita'
- Difficile progettazione



Linux Day
Torino 2009

Monolitico

- TUTTO il kernel e' a livello kernel
- Sviluppo piu' semplice
- I bug di singoli moduli possono portare al blocco del kernel
- Interfaccia di alto livello sull'hardware



Il Kernel LINUX

- Nell'Aprile 1991, Linus Torvalds inizio' la stesura del Kernel.
- Struttura Monolitica/Modulare (criticata da Tanenbaum, Professore di Sistemi, creatore di MINIX)
- Codice sorgente di circa 12 milioni di righe



Reperire i sorgenti

- www.kernel.org
- Archivio compresso tar.gz o tar.bz2
- Patch
- ChangeLog



Estrazione sorgenti

- Creare una directory nella propria home per la compilazione
- Copiare l'archivio nella cartella
- Scompattare i sorgenti

- Caso tar.bz2

- ```
tar xvjf linux-VERSIONE.tar.bz2
```

- Caso tar.gz

- ```
tar xvzf linux-VERSIONE.tar.gz
```



Applicazione delle patch

- Bugfix
- Codice non incluso nel ramo ufficiale o stabile
- Aggiornamenti di versione, senza scaricare ogni volta i sorgenti
-
- `$bzip2 -dv patch-VERSIONE.bz2`
- `$patch -p1 < patch-VERSIONE`



Classic Way

- Come per gli altri modi di compilare, spostiamoci nella cartella dei sorgenti:
 - `$ cd linux-VERSIONE`
- CONFIGURAZIONE
- `$ make dep`
- `$ make modules`
- `$ make bzImage`
- `# make modules_install`
- `#cp -v arch/x86/boot/bzImage /boot/vmlinuz`
`-VERSION`



Debian Way

- CONFIGURAZIONE
- `$fakeroot make-kpkg --append-to-version
-nomepersonalizzato --revision=1
kernel_image kernel_headers
modules_image`
- `# dpkg -i linux-image-
[VERSIONE_DEL_KERNEL].deb`
- `# dpkg -i linux-headers-
[VERSIONE_DEL_KERNEL].deb`



Arch Way - PKGBUILD

- Salvare nella directory di compilazione(es.: ~/kernelbuid) i PKGBUILD e il kernel26.install presenti in:

http://wiki.archlinux.org/index.php/Kernel_Compilation_From_Source

- \$scp -r [kernel source location]/* kernelbuild/
- CONFIGURAZIONE
- CONTROLLIAMO i PKGBUILD e kernel26.install
- \$makepkg
- #pacman -U kernel26-VERSIONE



Configurazione

Ci sono 3 modi per configurare il kernel:

- Make xconfig (GUI basata su Qt)
- Make gconfig (GUI basata su GTK)
- Make menuconfig (via terminale)

Altre possibili opzioni

- Make defconfig
- Make silentconfig (oppure make config)

Configurazione attuale:

- /proc/config.gz



IMPORTANTE

- Per evitare problemi:
 - Controllare la corretta configurazione del bootloader:
 - /boot/grub/menu.lst
 - /etc/lilo.conf



Ottimizzazione

Obiettivi

- Adattare il kernel all'utilizzo che se ne fara'
- Adattare il kernel alla propria architettura
- Diminuire i tempi di boot



Initramfs

- E' un FS contenente l'immagine di un kernel e di tutti i moduli richiesti all'avvio

Conseguenze:

- Rallenta i tempi di boot
- E' inutile se compiliamo il kernel a dovere

Quindi...

- NO initramfs!



Cosa compilare all'interno[*] del kernel?

- Driver di periferiche non rimovibili
- FileSystem utilizzato dalla partizione di boot
- Tutto il resto non va compilato all'interno del kernel[] ma al massimo come modulo[M]



Moduli richiesti dal nostro sistema

- `#!/bin/bash`

```
# find_all_modules.sh
```

```
for i in `find /sys/ -name modalias -exec cat {} \;`; do
```

```
    /sbin/modprobe --config /dev/null --show-depends $i ;
```

```
done | rev | cut -f 1 -d '/' | rev | sort -u
```

- `$mount | grep " / "`



Ottimizzare per la propria architettura

- Processor Family
 - `$cat /proc/cpuinfo | grep "model name"`
- SMP (Simmetric multi-processing)
 - Dual/Quad Core
 - Dual Processor
- HIGHMEM64GB
PAE(Phisical Address Extensions)
 - RAM>4GB



Ottimizzare per i propri utilizzi

- Preemption: Capacita' del kernel di fermare quello che sta facendo per eseguire qualcos'altro a priorita' piu' elevata.
 - No Forced - Server
 - Voluntary - Desktop
 - Preemptile - Low Latency Desktop
- Timer Frequency Interrupts
 - 100Hz per Server
 - 1Khz per Desktop



Miglioriamo ancora un po'!

Per ottimizzare il binario:

- CFLAGS="-march=native -O2 -pipe -fomit-frame-pointer"

Per compilare piu' in fretta

- MAKEOPTS= "il doppio del numero di core/processori"
- Oppure compilare con l'opzione:
\$make -j(2*N)



Novita'

- KMS(Kernel Mode Settings) + GEM (Graphics Execution Manager)
 - Gestione dell'output e della memoria video video da parte del kernel
 - Risoluzione framebuffer nativa
 - Cambio utente veloce, piu' veloce
 - Cambio terminale veloce
 - Nessun lampeggiamento!
 - Possibilita' di BSOD!
 - Stabile per Intel, in testing su Ati



Novita'

- Devtmpfs: il ritorno di devfs
 - Gestione dei device di nuovo in kernel space
- Btrfs: ZFS per Linux
 - Copy-on-write filesystem
 - Extents (blocchi contigui)
 - Supporto migliorato a file di grosse dimensioni



Links e libri utili

- http://wiki.archlinux.org/index.php/Kernel_Compilation
- http://www.slacky.eu/wikislack/index.php?title=Compilazione_e_ricompilazione_Kernel_su_Slackware
- http://guide.debianizzati.org/index.php/Debian_Kernel_Howto
- <http://wiki.ubuntu-it.org/AmministrazioneSistema/CompilazioneKernel>
- <http://www.kernel.org/pub/linux/docs/>
- <http://www.makelinux.net/kernel/diagram>
- Linux Kernel in a nutshell – Kroah-Hartman – O'Reilly
- www.h-online.com



Buona Compilazione!!!!

Mirco Chinelli - mirco.chinelli@linux.com

