Kernel Linux

(per sysadm)

Kernel Linux – distribuzione

- un po' di numeri (per i kernel della famiglia 2.6.x)
  - mole: ~ 320 Mb di sorgenti per ~ 4'500'000 LOC
  - costo dello sviluppo: ~ 4'500 anni/uomo, ~ 600'000'000 $ (fonte: http://www.dwheeler.com/essays/linux-kernel-cost.html)
- distribuito su http://www.kernel.org
  - tarball da 30 Mb contenenti sorgenti C (portabili su almeno 16 architetture ad oggi) e Assembly (architecture specific)
  - interamente (up to firmware ...) rilasciato sotto licenza GPL
  - compilabile utilizzando utility GNU (GCC, Make, Binutils)

Kernel Linux – installazione

- è un kernel *monolitico* (i.e. non microkernel), ma *modulare*
- l'installazione consta di:
  1. parte monolitica
  2. moduli
  3. symbol table
- riferimenti:
  - appunti, capp. 48—50
  - The Linux Kernel HOWTO (vecchio)
  - http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.
  - Linux Loadable Kernel Module HOWTO
  - The System.map file http://www.dirac.org/linux/system.map/

Kernel Linux – parte monolitica

- non risiede necessariamente sul filesystem, ma deve essere nota al boot loader (e.g. lilo, grub, yaboot, ...)
  - il boot loader è l'applicazione che si preoccupa di “installare” il kernel in modo che la procedura di bootstrap della macchina (tipicamente implementata nel BIOS) sia in grado di accedervi
    - è fortemente dipendente dall'architettura
    - esempi: lilo & grub (i386), yaboot (powerpc)
- quando risiede sul filesystem è tipicamente un unico file /boot/vmlinu{x,z}-<*kernel-version*> (e.g. /boot/vmlinux-2.6.8)
- viene caricata in memoria al boot e mai scaricata

Kernel Linux – moduli

- parti di kernel caricati/scaricati on demand, manualmente dall'utente o da demoni preposti (kerneld/kmod)
- devono essere accessibili a runtime, pertanto risiedono all'interno del filesystem
  - un albero di file oggetto (.o/.ko) radicato in /lib/modules/<*kernel-version*>/ (e.g. /lib/modules/2.6.8/)
- flessibilità dei moduli
  - name aliasing ed opzioni (/etc/modules.conf)
  - dipendenze inter-modulo (/lib/modules/.../modules.dep)

Kernel Linux – gestione dei moduli

- caricamento on demand
  - insmod (full path, non gestisce alias e dipendenze)
  - modprobe (nome modulo, gestisce alias e dipendenze)
- rimozione: rmmod, modprobe (gestisce dipendenze)
- ispezione: lsmod
- calcolo delle dipendenze: depmod
- informazioni su: modinfo
- manutenzione opzioni ed alias: update-modules (vecchio da module-init-tools in poi)

Kernel Linux – symbol table

- /boot/System.map-<*kernel-version*>
- contiene la lista dei simboli associata al kernel, nel formato di nm: <symbol value, symbol type, symbol name>
- viene utilizzata da alcuni programmi di sistema (e.g. klogd, ps, lsof) per la risoluzione da symbol value a symbol name
  - e.g. in caso di kernel oops (“segfault” in kernel space), il kernel dispone solamente del valore corrente dell'EIP (instruction pointer). klogd intercetta l'oops ed utilizza System.map per ottenere il symbol name corrispondente all'EIP e loggarlo via syslog
- comandi: nm

Kernel Linux – configurazione

- configurazione, compilazione ed installazione del kernel sono gestite dal Makefile distribuito assieme al kernel
- configurazione
  - target del Makefile: config, xconfig, menuconfig, oldconfig, gconfig (solo 2.6.x)
  - ogni componente del kernel può essere:
    - compilata ed inclusa nella parte monolitica
    - compilata come modulo
    - non compilata
  - vengono gestite dipendenze inter-componente
  - risiede nel file .config all'interno dell'albero dei sorgenti

Kernel Linux – compilazione & installazione

- compilazione
  - target del Makefile
    - pulizia: clean, mrproper
    - compilazione parte monolitica: bzImage
    - compilazione moduli: modules
  - risultati della compilazione
    - arch/<*architecture*>/boot/bzImage + file (.o/.ko) per i moduli
- installazione
  - su disco: fortemente dipendente dal bootloader
  - su floppy: dd if=..../bzImage of=/dev/fd0
  - moduli: target modules_install del Makefile

initrd – INITial Ram Disk (1/2)

- nel caso che parti del kernel necessarie al boot siano necessarie al boot (e.g. supporto scsi, filesystem, RAID) e' possibile utilizzare un “initrd”
- ogni initrd e' una immagine (compressa, tipicamente in formato cpio + gzip) di un filesystem minimale contenente:
  - moduli necessari al boot del sistema
  - tool per il boot del filesystem reale (post caricamento dei moduli)

initrd – INITial Ram Disk (2/2)

- procedura (semplificata) di boot con initrd:
  1. loading del kernel e del ramdisk -> boot loader
  2. mounting del ramdisk come root filesystem -> kernel
  3. esecuzione di un prog. predefinito (e.g. /init, /linuxrc) -> kernel
  4. loading di moduli necessari -> /init
  5. mounting del filesystem reale -> /init
  6. pivot_root -> /init
  7. procedura usuale di boot
- la configurazione di initrd e' bootloader-dependent
- comandi: mkinitrd (vecchio), yaird
- riferimenti: http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/initrd

UML – User Mode Linux

- UML è una macchina virtuale che implementa il kernel linux in user space su una macchina host linux
- motivazioni: sandboxing, kernel hacking, insegnamento
- attualmente supporta i kernel della famiglia 2.4.x (con patch) e 2.6.x (vanilla) su macchine host i386
- riferimenti:
  - http://user-mode-linux.sourceforge.net
    - User Mode Linux Wiki: http://uml.harlowhill.com/
    - User Mode Linux HOWTO
    - *A user-mode port of the Linux kernel*, Jeff Dike, http://user-mode-linux.sourceforge.net/als2000/index.htm