GNU Linux

Questo documento contiene una bozza di traccia delle lezioni.
Il contenuto e' stato utilizzato all'interno della lezione in cui si e' discusso di

Compile Linux Kernel 4.2.6 on Light Ubuntu 15.04

Contenuti delle lezioni :

- apt-get , aptitude
- gzcat
- uname -r
- .config
- make oldconfig
- libncurses5 libncurses5-dev
- make menuconfig
- grub

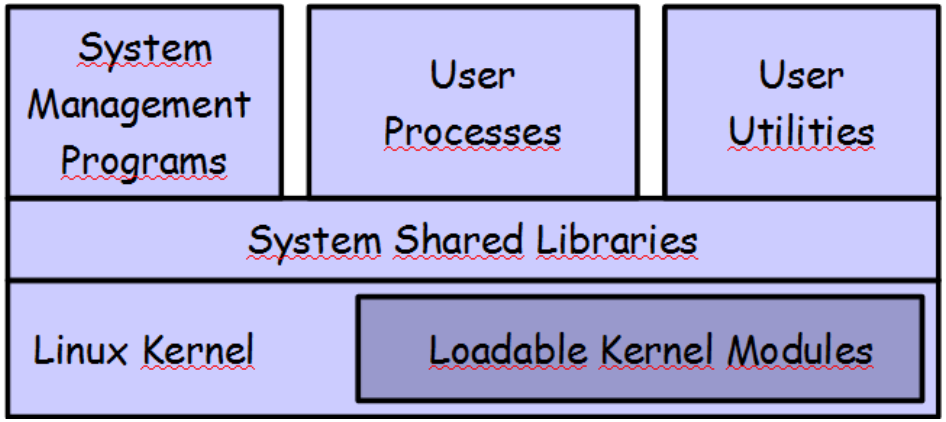
- lsmod, insmod, rmmod
- tail -f
- file system /proc
- socket netlink

- Virtual File System
- sysctl

Linux

Il Nome Vero

GNU Linux



Rapporti con altri UNIX

"compliant" con le specifiche POSIX
API fortemente basata su UNIX SVR4
molti tool e librerie derivanti da BSD

Moduli

sezioni del codice del kernel che possono essere compilati, caricati e "scaricati" in modo indipendente dal resto del kernel
un modulo del kernel può implementare tipicamente un device driver, un file system, o un protocollo di networking
l'interfaccia dei moduli permette a terze parti di scrivere o distribuire, in base ai propri termini, device driver o altro codice che non può essere distribuito sotto GPL

Tre componenti formano il supporto di linux per la gestione dei moduli:

manager dei moduli
registrazione dei driver
risoluzione dei conflitti

Linux Modules

Registrazione dei driver

permette ai moduli di informare il resto del kernel che un nuovo driver è disponibile
il kernel mantiene un tabella dinamica dei moduli e fornisce metodi per aggiungere o rimuovere un modulo dalla tabella
possibili moduli che possono essere aggiunti

- device driver
- protocolli di rete
- file system

Posso caricare moduli collocati dovunque con **insmod**

```
sudo insmod parameter_module.ko mystring="vaf"
```

Per caricare automaticamente un modulo devo installarlo nella directory

```
/lib/modules/`uname -r`
```

In tal caso puo' essere caricato anche mediante l'eseguibile **modprobe**

Dopo il caricamento, il modulo sara' visibile nella directory:

```
/sys/module/parameter_module/
```

e i suoi parametri saranno disponibili in lettura nei file contenuti nella directory:

```
/sys/module/parameter_module/parameters/
```

Da leggere per approfondire:

```
http://mirror.linux.org.au/linux-mandocs/2.6.1/index.html
```

File System in Linux

Media based

ext2 - Linux native
ufs - BSD
fat - DOS FS
vfat - win 95
hpfs - OS/2
minix - Minix
Isofs - CDROM
sysv - Sysv Unix
hfs - Macintosh
affs - Amiga Fast FS
NTFS - NT's FS
adfs - Acorn-strongarm

Network

nfs
Coda
AFS - Andrew FS
smbfs - LanManager
ncpfs - Novell

Journaling

ext3 - Linux native
ext4 - Linux native
reiserfs - Linux native

Special

procfs -/proc
sysfs - /sys
umsdos - Unix in DOS
userfs - redirector to user

Linux - Virtual File System

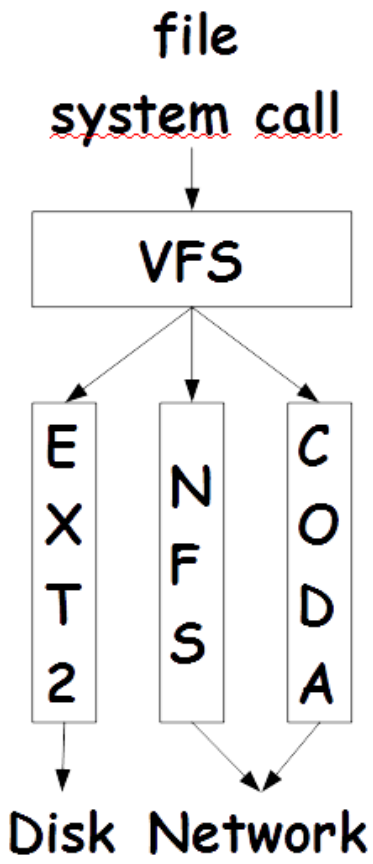
Problema
come accedere ai file system in modo uniforme?

VFS
il kernel gestisce un ulteriore livello di astrazione, detto *virtual file system*
VFS mantiene una tabella di "tipi di file system"
ogni tipo di file system è associato a una tabella di "metodi virtuali"

Quando avviene una system call:
il kernel determina il tipo di file
seleziona il metodo richiesto per una certa operazione dalla tabella corrispondente

Al caricamento di un modulo kernel per il file system:
il modulo contiene le implementazioni dei metodi virtuali
registra queste implementazioni in una struttura dati del kernel

E' simile alla programmazione object-oriented
le diverse istanze di file system corrispondono agli oggetti
l'insieme dei metodi virtuali è l'interfaccia
è anche possibile che un metodo sia "ereditato" dalla sua versione generica



Linux - Virtual File System

include/linux/fs.h:

```
struct file_operations {

        loff_t (*llseek)(struct file *, loff_t, int);
        ssize_t (*read)(struct file *, char *, size_t, loff_t *);
        ssize_t (*write)(struct file *, const char *, size_t, loff_t*);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll)(struct file*, struct poll_table_struct *);
        int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long );
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *);
        int (*release) (struct inode *, struct file *);
        ...
};
```

Linux - proc File System

Il file system proc

non memorizza dati.

il suo contenuto è calcolato "on demand" a seconda delle richieste dei processi utenti

proc deve implementare una struttura di directory, e il contenuto dei file; questo significa anche definire un inode number univoco e persistente per ogni directory e file contenuto

quando i dati vengono letti, proc raccogliere le informazioni appropriate, le formatta in forma testuale e le copia nel buffer di lettura del processo richiedente

Linux - sysctl

Command example:

where: base

ls /proc/sys/

what:

ls /proc/sys/net/ipv4/ip_forward

read:

sysctl net/ipv4/ip_forward

write:

sudo sysctl net/ipv4/ip_forward=1

read all:

sysctl -a | grep ip_forward

che fa questo?

find /sys/ -type f -perm -u=w -exec ls -alH '{}' \;