Organizzazione di Sistemi Operativi e Reti

Supplementi per il sistema FreeBSD ed integrazioni

Andrea Passarella
a.passarella@iet.unipi.it
1 aprile 2003

Nota: i presenti supplementi sono pensati per completare la dispensa sulla parte di Sistemi Operativi. In particolare, la dispensa presenta strumenti che sono normalmente disponibili in qualunque sistema UNIX. Tuttavia, nel dettaglio della spiegazione e negli esempi, viene spesso fatto riferimento a sistemi Linux. Nel seguito vengono evidenziate le differenze di uso (qualora ve ne siano) degli stessi strumenti in sistemi FreeBSD. **Non sono riportate differenze che siano banalmente ricavabili dalle pagine di manuale, come, ad esempio, differenze nei nomi di opzioni dei comandi.**

1 Capitolo 6 – Gestione dei processi

1.1 Filesystem `/proc`

I files che si trovano nel filesystem a partire da `/proc` dipendono dal particolare sistema UNIX che si sta usando. In particolare l'organizzazione dei dati nei sistemi FreeBSD e Linux è diversa, anche se il contenuto informativo è sostanzialmente lo stesso.

In particolare, la directory `/proc` di un sistema FreeBSD si presenta così:

```
$ ls /proc
0/ 115/ 169/ 2/ 90/
1/ 116/ 192/ 25/ 93/
102/ 117/ 193/ 2956/ 95/
104/ 123/ 194/ 2957/ 97/
105/ 125/ 195/ 2966/ 99/
106/ 127/ 196/ 3/ curproc@
107/ 129/ 197/ 4/
109/ 132/ 198/ 5/
114/ 135/ 199/ 6/
```

Le directories corrispondono ai processi in esecuzione (i loro nomi corrispondono ai PID dei processi). `curproc` è un link simbolico alla directory del processo attualmente in esecuzione. Le directories contengono dei files con dentro informazioni sullo stato corrente del processo (memoria occupata, linea di comando, . . .).

```
$ ls /proc/1
cmdline file@ note rlimit
ctl mem notepg status
```

In un sistema Linux il contenuto della directory `/proc` è invece qualcosa del genere:

```
$ ls /proc
1/ 2/ 3/ 330/ 359/ 4/ 408/
4710/ 5/ 577/ 589/ 598/ 6/ 613/
672/ 7/ 707/ 716/ 725/ 726/ 727/
728/ 729/ 730/ 731/ 732/ 733/ 734/
8/ 811/ 824/ 892/ 893/ 894/ 895/
896/ 897/ 900/ 915/ 920/ 924/ 928/
930/ 931/ 934/ 946/ apm bus/ cmdline
cpuinfo devices dma driver/ e820info execdomains fb
filesystems fs/ ide/ interrupts iomem ioports irq/
kcore kmsg ksyms loadavg locks mdstat meminfo
misc modules mounts net/ partitions pci self@
slabinfo stat swaps sys/ sysvipc/ tty/ uptime
version
```

La differenza principale è che Linux memorizza in questo punto del filesystem molte informazioni sullo stato complessivo del sistema che invece FreeBSD gestisce tramite strumenti diversi. Tali informazioni sono contenute nei files ordinari (es., `/proc/apm`), e nelle directories che non corrispondono ad un particolare processo (es. `/proc/net`).

Per quanto riguarda le informazioni sui singoli processi, le differenze sono minori, e le informazioni memorizzate sono sostanzialmente le stesse.

```
$ ls /proc/1
cmdline environ fd/ mem root@ statm
cwd@ exe@ maps mounts stat status
```

Si ricorda infine che le informazioni memorizzate nel filesystem `/proc` non vengono normalmente accedute direttamente, ma tramite strumenti come `top` e `ps`. Pertanto, le differenze che abbiamo qui evidenziato vengono mascherate da questi strumenti.

1.2 `top`

La differenza fondamentale tra `top` eseguito sotto Linux e `top` eseguito sotto FreeBSD riguarda la presentazione dei dati relativi allo stato complessivo del sistema. Sotto Linux un tipico output di `top` è così fatto:

```
12:03pm up 21 days, 2:59, 1 user, load average:  0.12, 0.12, 0.07
83 processes:  82 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  7.2% user, 6.3% system, 0.0% nice, 86.4% idle
Mem:  516324K av, 450484K used, 65840K free, 0K shrd, 24652K buff
Swap:  72256K av, 6824K used, 65432K free          281896K cached
```

La prima riga è di immediata comprensione (si ricorda che `load average` riporta i carichi medi della CPU nell'ultimo minuto, negli ultimi cinque minuti e nell'ultimo quarto d'ora, rispettivamente).

Nella seconda segnaliamo gli ultimi due elementi: sono il totale dei processi zombie (processi terminati con cui il processo padre non si è ancora sincronizzato) e dei processi in stato di stop (cioè che hanno ricevuto un `SIGSTOP`).

La terza riga riassume lo stato di utilizzo della CPU. Il campo `user` dà la percentuale di CPU utilizzata compelssivamente da TUTTI i processi mentre girano in modalità utente. Il campo `system` dà la percentuale di CPU utilizzata compelssivamente da TUTTI i processi mentre girano in modalità kernel (cioè quando eseguono delle syscall). Il campo `nice` è l'analogo del campo user, ma riguarda quei processi per cui sia stato modificato esplicitamente il nice level. Il campo `idle` è l'utlizzo della CPU da parte del processo dummy.

La quarta riga riassume lo stato della memoria fisica. In particolare, gli ultimi due elementi ed il campo `cached` della riga successiva indicano memoria allocata dal kernel, ed hanno il seguente significato:

- `shrd` indica quanta memoria è allocata per i meccanismi di comunicazione a memoria condivisa tra processi
- `buff` indica la memoria allocata ai buffer dei dati in I/O (ad esempio, quando si fa una lettura su disco, il sistema trasferisce automaticamente alcuni blocchi adiacenti alla locazione acceduta, e li mette in questi buffer)
- `cached` è una cache di pagine di memoria virtuale recentemente utilizzate, ma che – concettualmente – potrebbero essere swappate. Tipicamente vi finiscono temporaneamente pagine dati di processi appena terminati. In questo modo, se il processo riparte entro breve tempo, si possono riutilizzare, ottimizzando la procedura di allocazione.

La somma di queste tre voci concorre al valore del campo `used`.

Per quanto riguarda FreeBSD, l'output è il seguente:

```
last pid:  3069; load averages:  0.08, 0.02, 0.01 up 2+00:01:43 12:56:13
34 processes:  1 running, 33 sleeping
CPU states:  0.0% user, 0.0% nice, 0.4% system, 0.0% interrupt, 99.6% idle
Mem:  8360K Active, 51M Inact, 38M Wired, 6528K Cache, 22M Buf, 18M Free
```

```
Swap:  384M Total, 68K Used, 384M Free
```

Le differenze fondamentali sono:

- nello stato della CPU viene anche riportato il campo `interrupt`, che è il carico sulla CPU dovuto alla GESTIONE del meccanismo delle interruzioni
- nello stato della memoria appaiono i campi:
  - `Active`: è la memoria fisica effettivamente utilizzata da processi attivi
  - `Inactive`: è una parte di memoria fisica dove vengono temporaneamente tenute le pagine codice di processi che sono da poco terminati. Tali pagine non vengono liberate immediatamente in modo che se il processo viene rimesso subito in esecuzione non ci sia bisogno di trasferire la sua immagine dal disco fisso.
  - `Wired`: è una zona di memoria riservata dal kernel.

Per quanto riguarda la modalità interattiva, quanto detto nella dispensa è riferito ad un sistema Linux, ma vale anche per FreeBSD. Una lieve differenza esiste per il comando di renice (`r`): in FreeBSD bisogna specificare prima il nuovo nice level e poi il PID (quindi non esiste un incremento di default nel nice level).

Infine, per quanto riguarda le informazioni relative ai singoli processi, esiste una differenza per il campo CPU. FreeBSD prevede due campi:

- WCPU è il carico sulla CPU calcolato tramite una finestra temporale di ampiezza al più 1 minuto
- CPU è la percentuale di CPU di un processo, calcolata su un dato intervallo temporale (raw percentage).

Linux prevede una sola informazione, (CPU) che è il carico del processo sulla CPU durante l'utlimo intervallo di update del comando `top` (che per default vale 2 secondi).

1.3 `ps` e `nice`

Per quanto riguarda `ps`, si segnala che il significato delle opzioni riportato nella dispensa si riferisce a sistemi Linux. Su sistemi FreeBSD le opzioni `f`, `r`, `h`, `O` hanno un significato diverso, che è facilmente comprensibile dalla pagina di manuale.

Per quanto riguarda `nice`, è necessario considerare la shell che si sta usando. In particolare la C shell fornisce il comando `nice` come builtin, cioè non ha bisogno di lanciare un comando esterno ad essa. Pertanto la sintassi si ricava dalla pagina di manuale di `csh`. La Bash invece non ha tale comando builtin, quindi la sintassi si ottiene tramite `man nice`.

2 Capitolo 7 – Filesystem

2.1 Montaggio di dispositivi da parte di utenti

In sistemi FreeBSD NON è diponibile un'opzione `user` da inserire nelle righe di `/etc/fstab` per consentire ad utenti diversi da root di montare parti del VFS. È ancora possibile ottenere lo stesso effetto tramite altri strumenti che non vengono presentati nel corso.

Una possibile soluzione per gestire files su dischetti DOS da parte di utenti diversi da root è l'utilizzo delle `mtools`. Le `mtools` sono una suite di comandi per compiere operazioni su dischetti formattati MS-DOS.

La sintassi dei comandi diponibili si basa sulle seguenti note:

- è necessario indicare il drive corrispondente al dischetto (normalmente `a:`)

- il sistema memorizza la directory corrente sul dischetto; i path dei files che si specificano sono relativi a questo punto del filesystem del dischetto
- per attraversare una directory si usa la notazione UNIX (`/`) e non quella MS-DOS (\)
- per indicare tutti i files si usa la notazione UNIX (`*`) e non quella MS-DOS (`*.*`)

Riportiamo qui l'uso di alcuni dei comandi presenti nelle `mtools`. Per un elenco completo si rimanda alla pagina di manuale `man mtools` (nella parte `SEE ALSO` c'è un elenco dei comandi disponibili; ognuno è documentato su una pagina di manuale separata)

- `mdir`: permette di listare il contenuto di una directory del dischetto (`mdir a:` lista il contenuto della directory corrente; `mdir a:pippo` lista il contenuto della sottodirectory `pippo` della directory corrente)
- `mcopy`: permette di copiare files da/verso il dischetto (`mcopy a:pippo/ciao.txt .` copia il file `ciao.txt` della sottodirectory `pippo` nel punto corrente del VFS UNIX)
- `mcd`: cambia la directory corrente del dischetto (`mcd a:pippo` cambia la directory corrente nella sottodirectory `pippo` di quella attuale; `mcd a:..` cambia la directory corrente nella genitrice di quella attuale)
- `mdel`: permette di cancellare files dal dischetto (`mdel a:pippo/*.zip` elimina tutti i files che finiscono con `.zip` nella sottodirectory `pippo` della directory corrente; `mdel a:pippo/*` elimina tutti i files della sottodirectory `pippo`)
- `mdeltree`: elimina ricorsivamente una directory, cioè elimina la directory ed il suo contenuto, comprese eventuali sottodirectories (`mdeltree a:pippo/prova`)
- `mrd`: elimina una directory solo se questa è vuota, altrimenti dà un messaggio di errore (`mrd a:pippo/prova`)
- `mmd`: crea una nuova directory (`mmd a:pippo/prova`, la directory `pippo` deve già esistere)
- `minfo`: dà informazioni (settori, tracce, ...) sul dischetto presente nel drive (`minfo a:`)

Infine, per quanto riguarda l'elenco delle opzioni per il file `/etc/fstab` e per il comando `mount`, quanto presentato nella dispensa riguarda sistemi Linux. Funzionalità simili sono normalmente disponibili anche sotto FreeBSD. Si rimanda alla pagina di manuale di `fstab` e di `mount` per i dettagli.

2.2 NFS

Su FreeBSD i processi che girano sul server sono `mountd`, che gestisce le richieste di mounting provenienti dai client, e `nfsd`, che gestisce le richieste di I/O sulle parti di filesystem esportate e montate da qualche client.

Per quanto riguarda il file `/etc/exports`, è innanzitutto necessario puntualizzare alcuni vincoli presenti sui sistemi FreeBSD.

Non è possibile esportare alla stessa macchina client punti diversi dello stesso filesystem con opzioni diverse. Quindi non è possibile avere due righe di `exports` in cui le directories esportate stiano sullo stesso filesytem, e la lista delle macchine client contenga elementi comuni. Per esempio, assumendo che su `/usr` sia montata una partizione di un qualche disco fisso, NON sono lecite due righe come quelle riportate di seguito:

```
/usr      -ro              10.4.65.1
/usr/doc  -mapall=-2:-2    10.4.65.1
```

Da questo vincolo discende anche che per ogni filesystem è possibile fornire una sola riga di default (cioè con l'elenco delle macchine client vuoto).

Infine, nel path delle directories esportate non devono essere presenti link simbolici (nella seconda riga dell'esempio precedente, né `/usr` né `doc` sotto `/usr` devono essere link simbolici).

Il file `/etc/exports` sotto FreeBSD ha una sintassi leggermente diversa dallo stesso file in distribuzioni Linux. In particolare, sotto FreeBSD le opzioni vanno specificate dopo il nome delle directories esportate. Ogni opzione comincia con –, e le opzioni sono separate da uno o più spazi.

Infine la sintassi per modificare il mapping degli utenti che accedono dalle macchine client è diversa. Sotto FreeBSD, l'opzione `maproot=uid:gid_1:...:gid_n` permette di assegnare agli utenti root dei client le credenziali specificate. In sostanza, root assume lo UID indicato con `uid`, il primary group indicato con `gid_1` e fa parte dei gruppi indicati con `gid_2:...:gid_n`. Analogamente, l'opzione `mapall=uid:gid_1:...:gid_n` fa assumere a tutti gli utenti dei client le credenziali specificate.

A differenza che in FreeBSD, nei sistemi Linux non è possibile specificare più directory esportate sulla stessa riga del file `exports`: su ogni riga può comparire solamente una directory che verrà esportata. Inoltre non esiste l'opzione `alldirs`.

È però possibile esportare allo stesso utente directories diverse dello stesso filesystem con opzioni diverse: è cioè possibile avere, nel file `exports`, righe differenti con client comuni e directories dello stesso filesystem. È anche possibile specificare più di una entry di default per un singolo filesystem. Nella entry di default è comunque necessario mettere il carattere `*` al posto della lista delle macchine client. È necessario notare che in questo caso può avere senso avere due entry di default relative allo stesso filesystem; ad esempio, in questo modo è possibile esportare a tutti due directories dello stesso filesystem con opzioni differenti.

Il mapping degli utenti è equivalente, anche se realizzato con un set di opzioni distinto. In particolare:

- per default gli utenti root sono mappati sull'utente specificato tramite le opzioni `anonuid` e `anongid` nella riga corrispondente alla particolare directory esportata del file `export`. Per default, `anonuid` e `anongid` valgono -2. Gli utenti normali sono mappati secondo le loro credenziali sul client.
- l'opzione `no_root_squash` fa sì che gli utenti root siano mappati come root anche sul server
- l'opzione `all_squash` fa sì che tutti gli utenti dei client siano mappati con le credenziali specificate in `anonuid` e `anongid`.

3 Capitolo 8

3.1 Accesso al filesystem

A completamento di quanto detto nella dispensa, riportiamo in maniera semplificata le azioni eseguite dal sistema quando viene fatto un accesso al fileystem. Questa sequenza di azioni è valida su qualunque sistema UNIX.

- Se l'owner del processo che accede al filesystem ha UID 0 (cioè è root), allora viene consentito qualunque accesso.
- Altrimenti, se l'owner del processo che accede ha lo stesso UID dell'owner del file acceduto, allora si controllano i permessi del file relativi all'owner.
- Altrimenti, se uno dei gruppi dell'owner del processo che accede è il gruppo cui appartiene il file, allora si controllano i permessi del file relativi al gruppo.
- Altrimenti si controllano i permessi del file relativi al resto del mondo.

4 Capitolo 9 – Gestione degli utenti

Benchè concettualmente analoga, la gestione degli utenti in un sistema FreeBSD è realizzata in maniera leggermente diversa. Il file principale delle password è ancora `/etc/passwd`. Quando invece si usa il meccanismo del password shadowing, le versioni criptate delle password sono conservate in `/etc/master.passwd` (non esiste un file `/etc/shadow`). Il formato di una riga in `master.passwd` è il seguente:

```
username:password_cifrata:uid:gid:login_class:pw_change:expire:dati_personali:home:shell
```

Rispetto ai campi presenti in una riga di `passwd` (v. dispensa), qui abbiamo due campi aggiuntivi (`pw_change` ed `expire`) che, se diversi da 0, indicano dopo quanto tempo è necessario che la password venga cambiata e dopo quanto tempo l'account verrà disattivato, rispettivamente. Il terzo campo aggiuntivo è `login_class`, che serve a specificare una classe di login, cioè una serie di impostazioni per il login degli utenti che ne fanno parte (ad es., il valore della variabile d'ambiente `path`).

Sia su sistemi Linux che FreeBSD, per quanto riguarda la password vanno evidenziati i seguenti casi:

- `*`: poichè nessuna stringa criptata genera `*` come risultato, non è possibile fare login con quell'utente fornendo username e password (rimane invece possibile il login tramite sistemi basati su autenticazione a chiave pubblica/privata, come ad esempio `ssh`).
- campo vuoto: l'account non ha una password, quindi chiunque può fare il login con quell'utente.

Sempre in entrambi i sistemi, esiste il caso particolare in cui la shell di login è impostata a `/sbin/nologin`. In questo caso non è possibile in nessun modo fare login con quell'utente. L'utente è tipicamente usato dal sistema operativo per far girare dei server di rete con permessi inferiori a quelli di root (v. ad esempio l'utente www usato per il server Apache).

Infine, poiché FreeBSD mantiene un database ottimizzato con i dati degli utenti, la procedura per aggiungere/elimiare manualmente gli utenti descritta nella dispensa (valida per sistemi Linux) non è applicabile. In questo caso è necessario utilizzare il comando `vipw`. Questo comando apre un editor `vi` sul file `/etc/master.passwd`. Quando si salva e si esce, automaticamente viene aggiornato il file `/etc/passwd`. È poi necessario modificare a mano il file `/etc/group` e creare/eliminare la home directory del nuovo utente. Il comando `vipw` è disponibile anche sotto Linux: apre `vi` su `/etc/passwd`, e, quando si esce e si è modificato il file, permette di aprire automaticamente `/etc/shadow` per renderlo consistente.

5 Capitolo 12 – Inizializzazione del sistema

Nei sistemi UNIX, la procedura di boot del sistema prevede una prima fase in cui viene caricato il kernel e vengono controllate ed inizializzate le periferiche. Successivamente, parte il processo `init`, che si occupa di configurare la macchina ad un livello più alto di quello hardware (configurazione della rete, creazione del VFS, avvio di server di rete, ...). Sotto FreeBSD, `init` come prima cosa esegue lo script `/etc/rc`. A sua volta `rc` va a leggere il file `/etc/defaults/rc.conf`, dove trova la definizione di una serie di variabili che utilizza per configurare varie parti del sistema. Dopo legge il file `/etc/rc.conf`, che contiene un'altra serie di variabili. Tipicamente è questo il file che l'amministratore modifica per configurare la macchina, lasciando intatto `/etc/defaults/rc.conf`. Guardando dentro un file `/etc/rc` di esempio, si trovano le seguenti righe:

```
...
if [ -r /etc/defaults/rc.conf ]; then
     .  /etc/defaults/rc.conf
     source_rc_confs
elif [ -r /etc/rc.conf ]; then
```

```
        .  /etc/rc.conf
fi
...
```

Se esiste il file `/etc/defaults/rc.conf` questo viene trattato come uno script ed eseguito. L'esecuzione avviene tramite il comando `source`, o tramite l'equivalente comando `.`, come avviene nell'esempio riportato. Eseguire uno script tramite `source` fa sì che le variabili e le funzioni definite dentro allo script rimangano visibili al chiamante anche dopo la terminazione dello script stesso. In questo caso, dentro lo script `/etc/defaults/rc.conf` è definita la seguente variabile:

```
rc_conf_files="/etc/rc.conf /etc/rc.conf.local";
```

La funzione `source_rc_confs` (anch'essa definita dentro a `/etc/defaults/rc.conf`) esegue `source` con ognuno dei files definiti nella variabile `rc_conf_files`.

Ritornando alla parte evidenziata di `/etc/rc`, si nota che se non esiste un file `/etc/defaults/rc.conf`, allora il `/etc/rc` esegue `source` con il file `/etc/rc.conf`, sempre che questo esista.

Come esempio di `rc.conf`, vediamo la parte necessaria a configurare un'interfaccia di rete.

```
hostname="pippo.iet.unipi.it"
defaultrouter="131.114.15.24"
ifconfig_fxp0="inet 131.114.15.99 netmask 255.255.255.224"
```

La prima variabile serve ad impostare il nome simbolico della macchina. Questa variabile non è strettamente necessaria, in quanto un'interfaccia di rete di un computer è completamente identificata su Internet dal suo indirizzo IP. Inoltre il valore di `hostname` deve essere concorde con il nome simbolico attribuito alla macchina dal servizio DNS.

La variabile defaultrouter indica il router di default utilizzato per inviare pacchetti per un destinatario IP non presente sulla rete locale. È necessario se si vuole che la macchina comunichi su Internet e non solo su rete locale.

La variabile `ifconfig_fxp0` serve a configurare l'interfaccia di rete `fxp0`. Il nome dell'interfaccia è un acronimo che indica il tipo della scheda (casa costruttrice e modello, `fxp` in questo caso) ed il numero progressivo di schede di questo tipo presenti sul sitema (`0` sta per prima interfaccia). In questo caso all'interfaccia verrà assegnato un indirizzo IP `131.114.5.57` con una netmask `255.255.255.224` (cioè di 27 bit).

Il file `/etc/rc` si basa su queste variabili per configurare la scheda di rete `fxp0`, rendendo così il computer visibile su Internet. Meccanismi analoghi sono utilizzati per configurare al boot altre parti del sistema.