Esempio 2

- Supponiamo di dover inizializzare una matrice di interi così definita:

```
int mat[128][128];
```

- useremo quindi due indici i e j per scorrere la matrice e accedere alle sue celle, a cui assegneremo il valore 0

Esempio 2-a

- **Soluzione dell'utente Sloppy**

```
for (j=0; j<128; j++) // per ogni colonna
    for (i=0; i<128; i++)  // per ogni riga
       mat[i][j] = 0;
```

- il codice è corretto
- il problema è che le matrici sono memorizzate per riga ...
- supponiamo che le pagine siano da 128 parole, ogni riga della matrice è contenuta in una pagina diversa (128 pagine): il ciclo for esterno può arrivare a produrre 128 page fault ad ogni iterazione (128 x 128 = 16.384 p.f.)
- questo perché in generale il processo avrà disposizione un numero di frame ridotto

Esempio 2-b

- **Soluzione dell'utente Sly**

```
for (i=0; i<128; i++) // per ogni riga
    for (j=0; j<128; j++)  // per ogni colonna
        mat[i][j] = 0;
```

- il codice è corretto
- abbiamo al più 128 page fault, uno per ogni riga

Allocazione dei frame

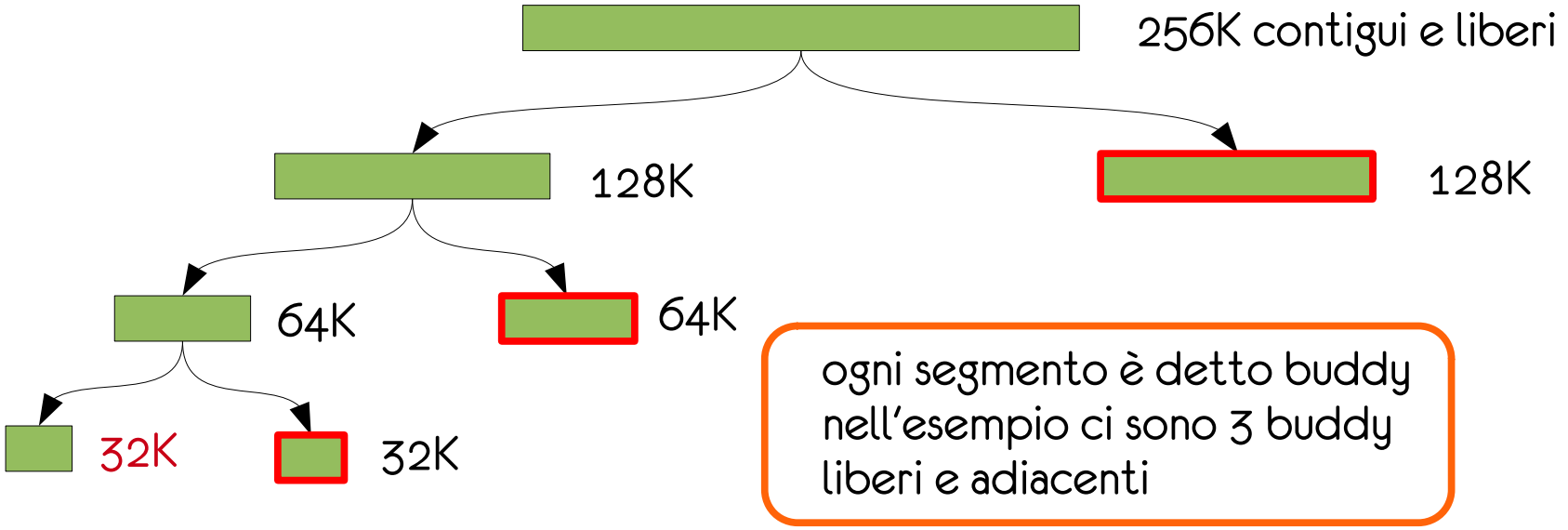
- per i processi utente
- per i processi kernel

Allocazione per il kernel

- L'allocazione di memoria per i processi kernel sottosta a meccanismi in parte dissimili da quella vista per i processi utente
- Ciò è motivato da alcune particolari caratteristiche dei processi kernel e dalla necessità di rendere molto efficiente la loro esecuzione
  - necessità di strutturare dati di dimensioni variabili, spesso molto più piccoli di una pagina -> si desidera evitare lo spreco di una pagina per lo più vuota
  - alcuni dispositivi interagiscono direttamente con la RAM, se necessitano di una porzione di memoria maggiore di una pagina, l'area richiesta deve essere sempre contigua
- **codice e dati del kernel non sono sottoposti a paginazione**

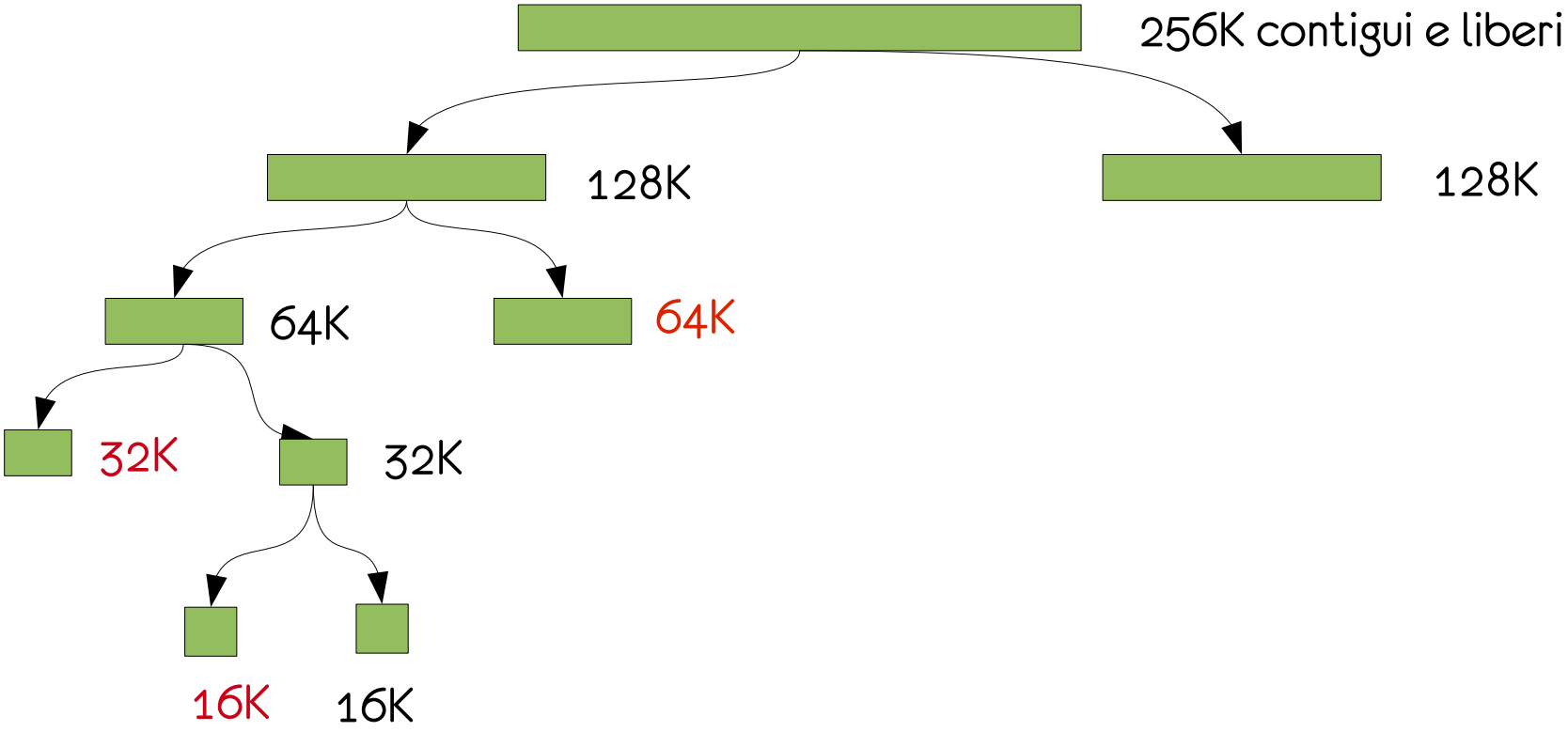
Sistema buddy

- meccanismo di allocazione della memoria, anche noto come sistema gemellare
- usa un segmento di pagine fisicamente contigue e alloca la memoria in unità di dimensioni pari a potenze di 2 (4KB, 8KB, 16KB, ecc.), arrotondando per eccesso le richieste (se chiedo 6KB, l'allocatore ne restituisce 8)
- Esempio il SO richiede 30K:



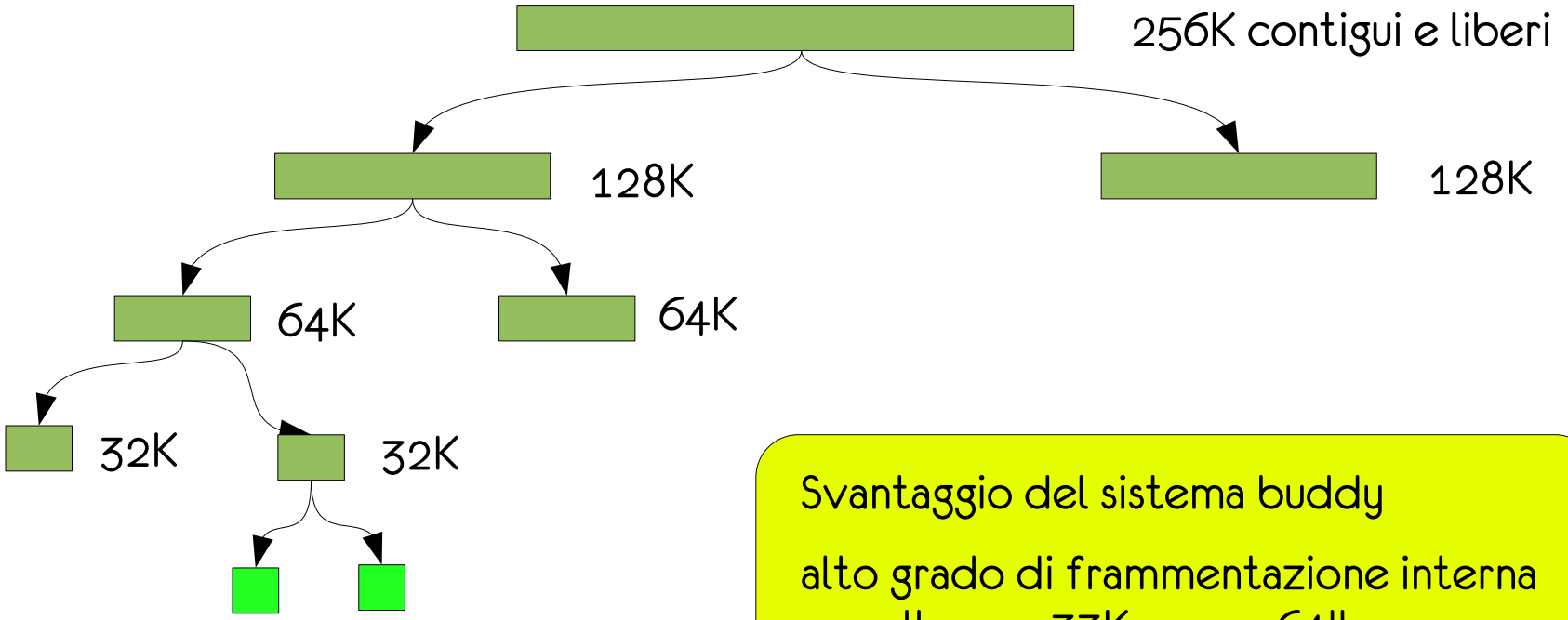
Sistema buddy

- Supponiamo che vengano richiesti 30K, 55K, 12K in sequenza



Sistema buddy

- coppie di gemelli liberi possono essere fuse in un unico buddy di dimensione doppia



Svantaggio del sistema buddy

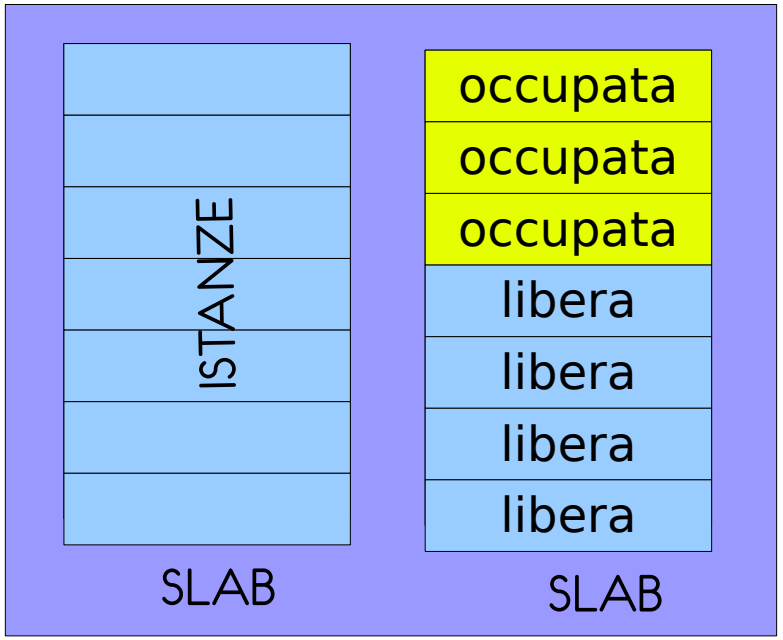
alto grado di frammentazione interna per allocare 33K ne uso 64!!

Allocazione a slab

- **slab** (lastra): sequenza di pagine fisicamente contigue
- **cache**: insieme di slab
- viene mantenuta una cache per ogni tipo di strutture dati usate dal SO, es.
  - una cache per i semafori
  - una cache per i PCB
  - una cache per i descrittori di file
  - ecc.

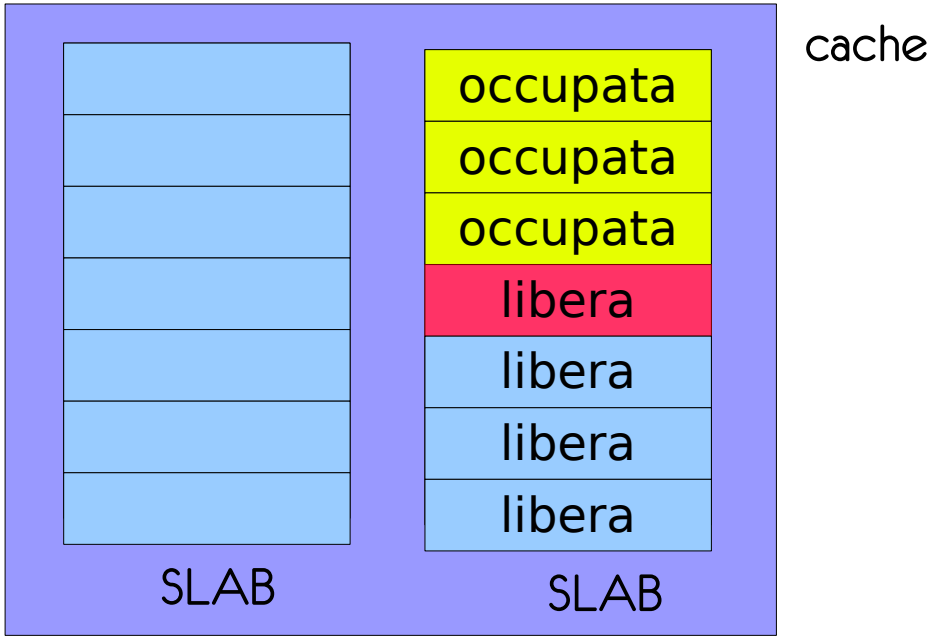
Allocazione a slab

- ogni cache contiene delle istanze del tipo di dato ad essa associato, il numero di istanze dipende dalla dimensione della cache e delle istanze stesse
  - es. posso avere 10 semafori
- ogni istanza può essere nello stato libera oppure occupata
  - es. tutti i semafori sono inizialmente liberi, cioè non utilizzati, quando un pool di processi richiede l'allocazione di un semaforo, un'istanza diventerà occupata



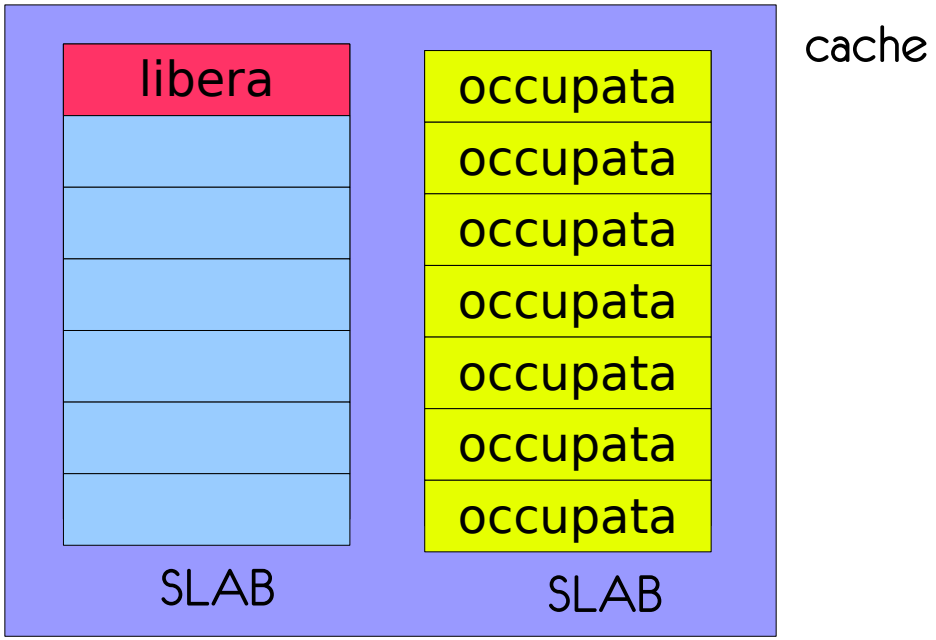
Allocazione

- quando viene richiesta una nuova istanza, il SO cerca la cache in questione
  - **<1>** all'interno di questa cerca una slab solo parzialmente occupata, e alloca un'istanza libera contenuta da quest'ultima



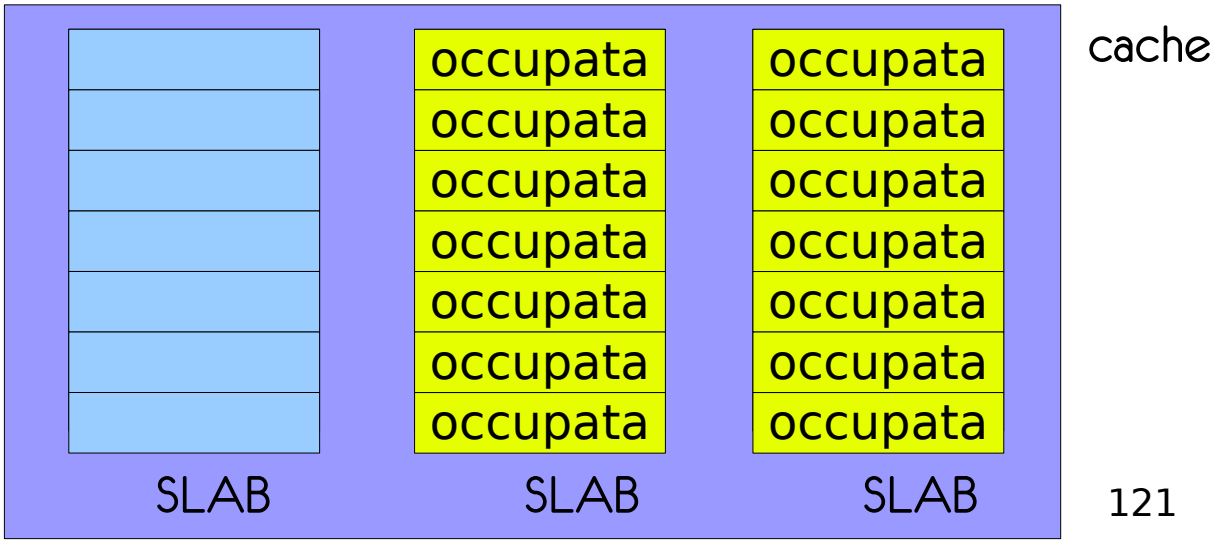
Allocazione

- quando viene richiesta una nuova istanza, il SO cerca la cache in questione
  - **<1>** all'interno di questa cerca una slab solo parzialmente occupata, e alloca un'istanza libera contenuta da quest'ultima
  - **<2>** se non ne trova, cerca una slab libera e alloca un'istanza da essa contenuta



Allocazione

- quando viene richiesta una nuova istanza, il SO cerca la cache in questione
  - **<1>** all'interno di questa cerca una slab solo parzialmente occupata, e alloca un'istanza libera contenuta da quest'ultima
  - **<2>** se non ne trova, cerca una slab libera e alloca un'istanza da essa contenuta
  - **<3>** se non ne trova crea una nuova slab a partire da un insieme di frame contigui e l'assegna alla cache poi ripete **<2>**

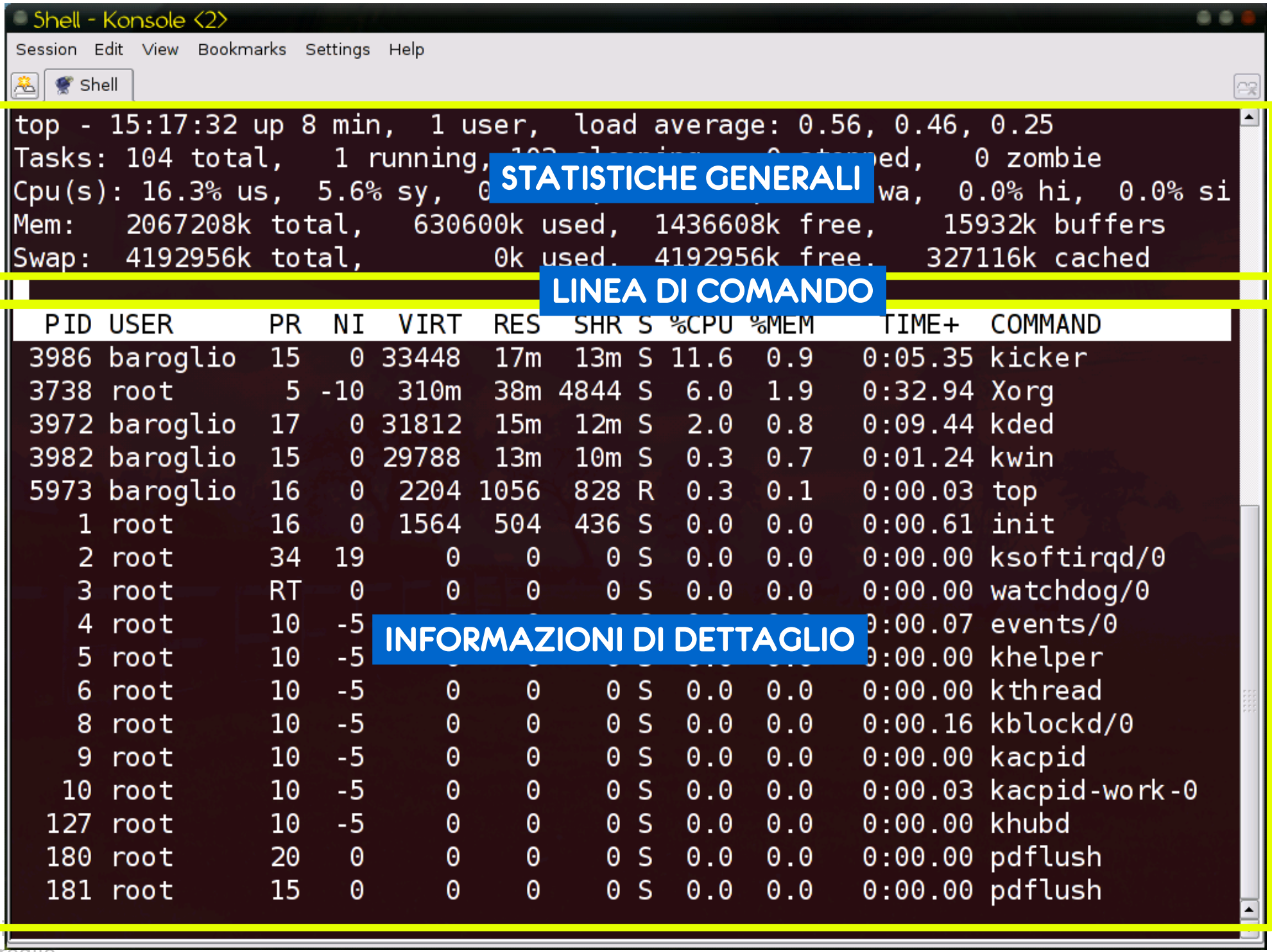


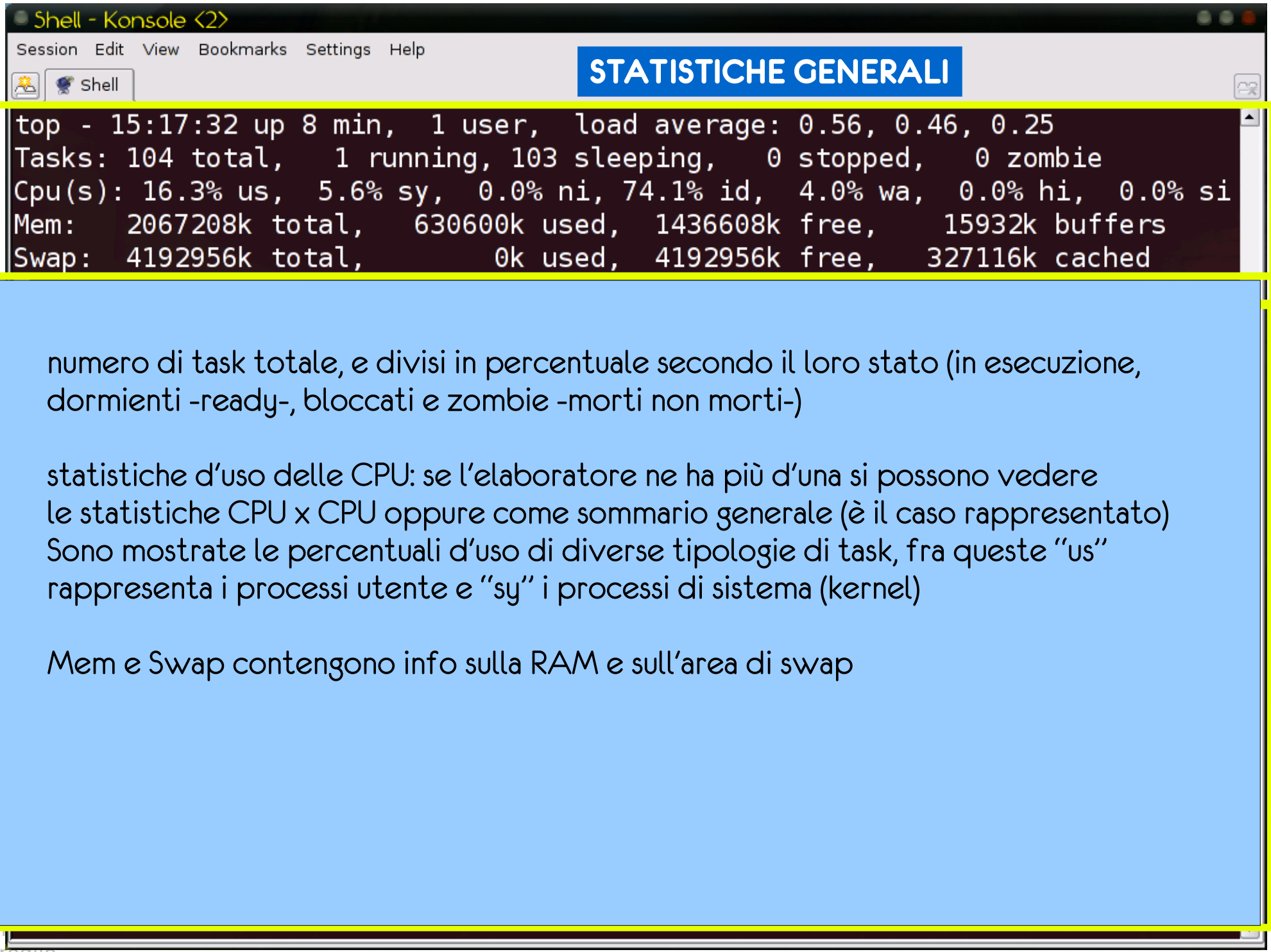
Allocazione

- quando viene richiesta una nuova istanza, il SO cerca la cache in questione
  - **<1>** all'interno di questa cerca una slab solo parzialmente occupata, e alloca un'istanza libera contenuta da quest'ultima
  - **<2>** se non ne trova, cerca una slab libera e alloca un'istanza da essa contenuta
  - **<3>** se non ne trova crea una nuova slab a partire da un insieme di frame contigui e l'assegna alla cache poi ripete **<2>**
- è una tecnica efficiente che elimina il problema della frammentazione interna perché una slab è suddivisa in spazi adatti a contenere un certo tipo di oggetti e gli oggetti sono allocati in toto o per nulla
- introdotta da Solaris, usata anche in Linux (che prima utilizzava il sistema buddy)

Top

- top è un comando presente in alcune versioni di Unix e Linux che consente di:
  - <1> vedere una gran quantità di informazioni riguardo l'uso di CPU e RAM
  - <2> eseguire comandi per la gestione dei processi
- si lancia da linea di comando
- vediamo insieme com'è strutturata l'interfaccia

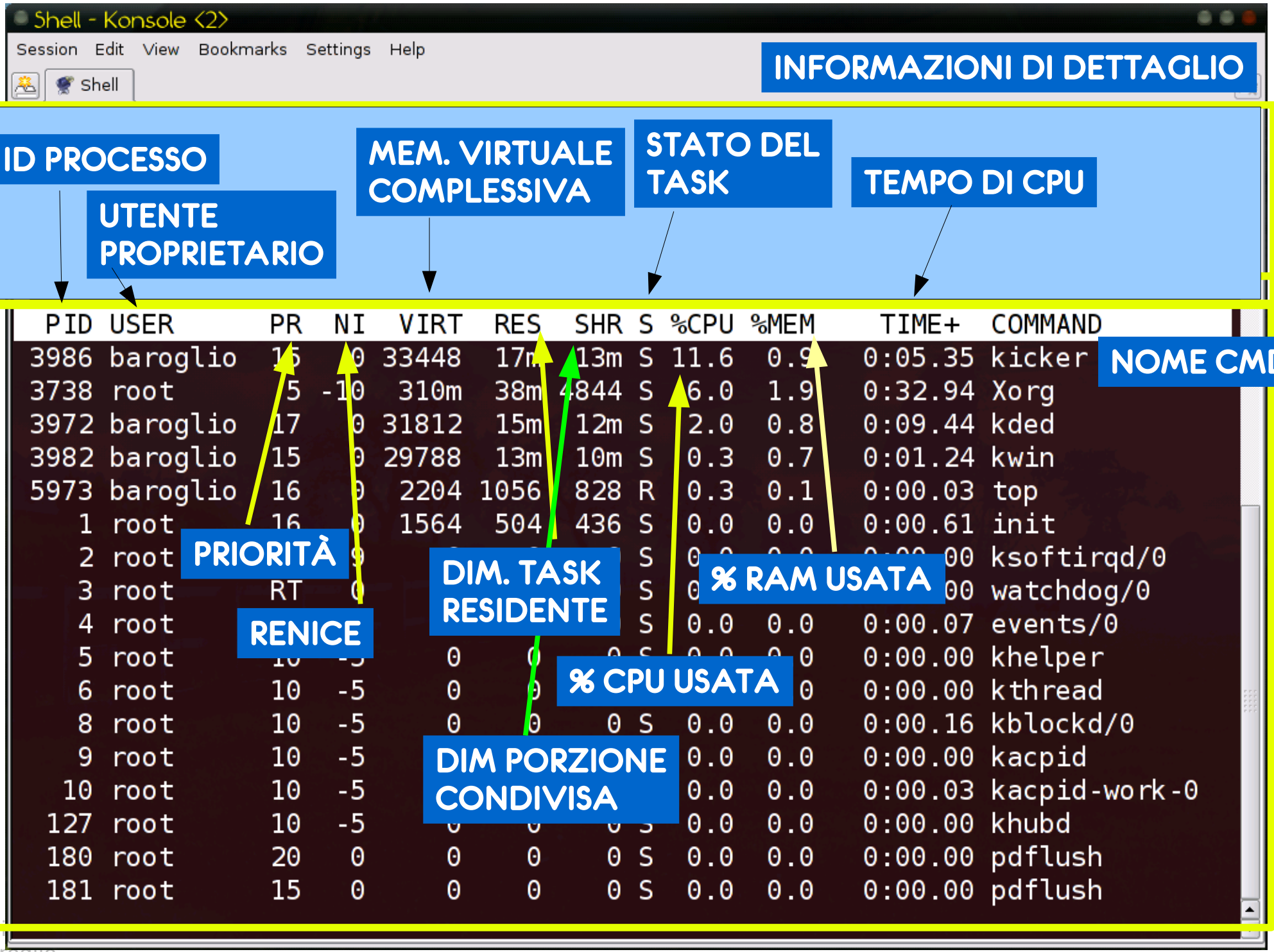

Shell - Konsole <2>
Session Edit View Bookmarks Settings Help
Shell
top - 15:17:32 up 8 min, 1 user, load average: 0.56, 0.46, 0.25
Tasks: 104 total, 1 running, 0 zombie
Cpu(s): 16.3% us, 5.6% sy, 0.0% hi, 0.0% si
Mem: 2067208k total, 630600k used, 1436608k free, 15932k buffers
Swap: 4192956k total, 0k used, 4192956k free, 327116k cached
STATISTICHE GENERALI
LINEA DI COMANDO
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3986 baroglio 15 0 33448 17m 13m S 11.6 0.9 0:05.35 kicker
3738 root 5 -10 310m 38m 4844 S 6.0 1.9 0:32.94 Xorg
3972 baroglio 17 0 31812 15m 12m S 2.0 0.8 0:09.44 kded
3982 baroglio 15 0 29788 13m 10m S 0.3 0.7 0:01.24 kwin
5973 baroglio 16 0 2204 1056 828 R 0.3 0.1 0:00.03 top
1 root 16 0 1564 504 436 S 0.0 0.0 0:00.61 init
2 root 34 19 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0
3 root RT 0 0 0 0 S 0.0 0.0 0:00.00 watchdog/0
4 root 10 -5 0:00.07 events/0
5 root 10 -5 0:00.00 khelper
INFORMAZIONI DI DETTAGLIO
6 root 10 -5 0 0 0 S 0.0 0.0 0:00.00 kthread
8 root 10 -5 0 0 0 S 0.0 0.0 0:00.16 kblockd/0
9 root 10 -5 0 0 0 S 0.0 0.0 0:00.00 kacpid
10 root 10 -5 0 0 0 S 0.0 0.0 0:00.03 kacpid-work-0
127 root 10 -5 0 0 0 S 0.0 0.0 0:00.00 khubd
180 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pdflush
181 root 15 0 0 0 0 S 0.0 0.0 0:00.00 pdflush

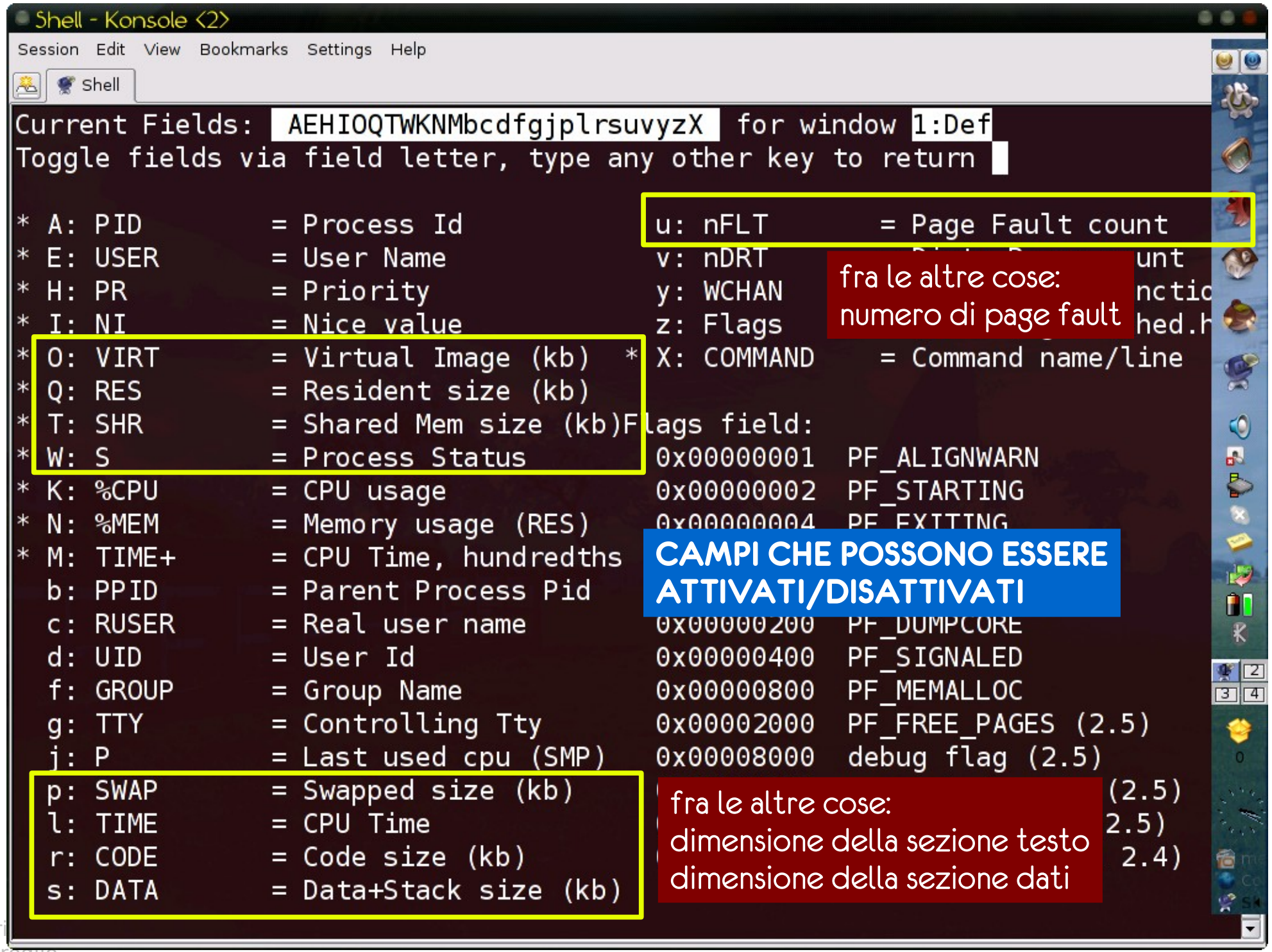


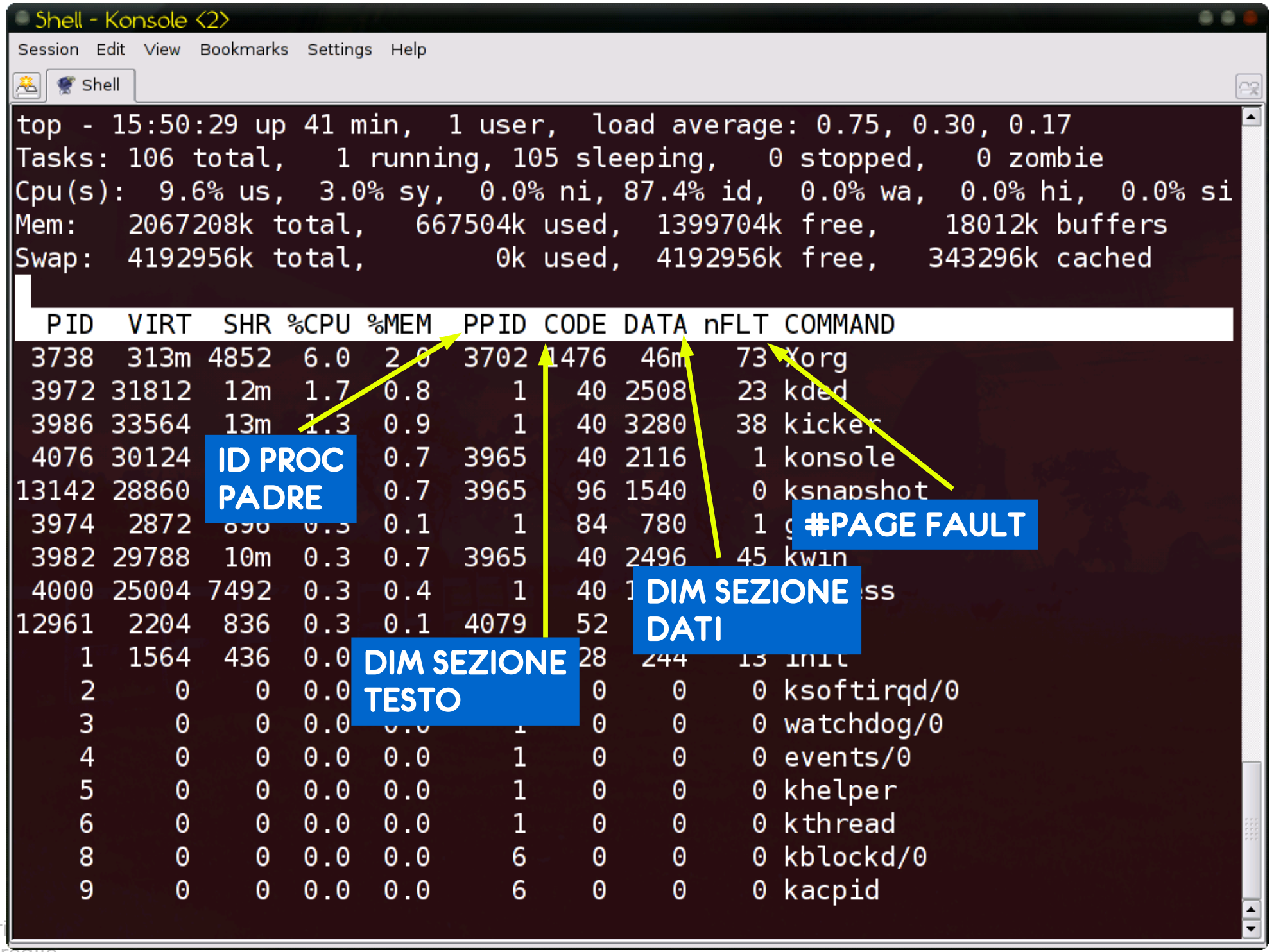
numero di task totale, e divisi in percentuale secondo il loro stato (in esecuzione, dormienti -ready-, bloccati e zombie -morti non morti-)

statistiche d'uso delle CPU: se l'elaboratore ne ha più d'una si possono vedere le statistiche CPU x CPU oppure come sommario generale (è il caso rappresentato) Sono mostrate le percentuali d'uso di diverse tipologie di task, fra queste "us" rappresenta i processi utente e "sy" i processi di sistema (kernel)

Mem e Swap contengono info sulla RAM e sull'area di swap


Shell - Konsole <2>
Session Edit View Bookmarks Settings Help
Shell
INFORMAZIONI DI DETTAGLIO
ID PROCESSO
UTENTE PROPRIETARIO
MEM. VIRTUALE COMPLESSIVA
STATO DEL TASK
TEMPO DI CPU
NOME CMD
PRIORITÀ
RENICE
DIM. TASK RESIDENTE
% RAM USATA
% CPU USATA
DIM PORZIONE CONDIVISA
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3986 baroglio 15 0 33448 17m 13m S 11.6 0.9 0:05.35 kicker
3738 root 5 -10 310m 38m 4844 S 6.0 1.9 0:32.94 Xorg
3972 baroglio 17 0 31812 15m 12m S 2.0 0.8 0:09.44 kded
3982 baroglio 15 0 29788 13m 10m S 0.3 0.7 0:01.24 kwin
5973 baroglio 16 0 2204 1056 828 R 0.3 0.1 0:00.03 top
1 root 16 0 1564 504 436 S 0.0 0.0 0:00.61 init
2 root S 00 ksoftirqd/0
3 root RT S 00 watchdog/0
4 root S 0.0 0.0 0:00.07 events/0
5 root 0 0 0 0 0:00.00 khelper
6 root 10 -5 0 0 0:00.00 kthread
8 root 10 -5 0 0 0 S 0.0 0.0 0:00.16 kblockd/0
9 root 10 -5 0.0 0.0 0:00.00 kacpid
10 root 10 -5 0.0 0.0 0:00.03 kacpid-work-0
127 root 10 -5 0.0 0.0 0:00.00 khubd
180 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pdflush
181 root 15 0 0 0 0 S 0.0 0.0 0:00.00 pdflush


Shell - Konsole <2>
Session Edit View Bookmarks Settings Help
Shell
Current Fields: AEHIOQTWKNMbcdfgjplrsuvyzX for window 1:Def
Toggle fields via field letter, type any other key to return
* A: PID = Process Id
* E: USER = User Name
* H: PR = Priority
* I: NI = Nice value
* O: VIRT = Virtual Image (kb)
* Q: RES = Resident size (kb)
* T: SHR = Shared Mem size (kb)
* W: S = Process Status
* K: %CPU = CPU usage
* N: %MEM = Memory usage (RES)
* M: TIME+ = CPU Time, hundredths
b: PPID = Parent Process Pid
c: RUSER = Real user name
d: UID = User Id
f: GROUP = Group Name
g: TTY = Controlling Tty
j: P = Last used cpu (SMP)
p: SWAP = Swapped size (kb)
l: TIME = CPU Time
r: CODE = Code size (kb)
s: DATA = Data+Stack size (kb)
u: nFLT = Page Fault count
v: nDRT
y: WCHAN
z: Flags
X: COMMAND = Command name/line
Flags field:
0x00000001 PF_ALIGNWARN
0x00000002 PF_STARTING
0x00000004 PF_EXITING
0x00000200 PF_DUMPCORE
0x00000400 PF_SIGNALED
0x00000800 PF_MEMALLOC
0x00002000 PF_FREE_PAGES (2.5)
0x00008000 debug flag (2.5)
fra le altre cose:
numero di page fault
CAMPI CHE POSSONO ESSERE ATTIVATI/DISATTIVATI
fra le altre cose:
dimensione della sezione testo
dimensione della sezione dati


Shell - Konsole <2>
Session Edit View Bookmarks Settings Help
Shell
top - 15:50:29 up 41 min, 1 user, load average: 0.75, 0.30, 0.17
Tasks: 106 total, 1 running, 105 sleeping, 0 stopped, 0 zombie
Cpu(s): 9.6% us, 3.0% sy, 0.0% ni, 87.4% id, 0.0% wa, 0.0% hi, 0.0% si
Mem: 2067208k total, 667504k used, 1399704k free, 18012k buffers
Swap: 4192956k total, 0k used, 4192956k free, 343296k cached
PID VIRT SHR %CPU %MEM PPID CODE DATA nFLT COMMAND
3738 313m 4852 6.0 2.0 3702 1476 46m 73 Xorg
3972 31812 12m 1.7 0.8 1 40 2508 23 kded
3986 33564 13m 1.3 0.9 1 40 3280 38 kicker
4076 30124 0.7 3965 40 2116 1 konsole
13142 28860 0.7 3965 96 1540 0 ksnapshot
3974 2872 0.1 1 84 780 1
3982 29788 10m 0.3 0.7 3965 40 2496 45 kwin
4000 25004 7492 0.3 0.4 1 40
12961 2204 836 0.3 0.1 4079 52
1 1564 436 0.0
2 0 0 0.0 0 0 0 ksoftirqd/0
3 0 0 0.0 0 0 0 watchdog/0
4 0 0 0.0 0.0 1 0 0 0 events/0
5 0 0 0.0 0.0 1 0 0 0 khelper
6 0 0 0.0 0.0 1 0 0 0 kthread
8 0 0 0.0 0.0 6 0 0 0 kblockd/0
9 0 0 0.0 0.0 6 0 0 0 kacpid
ID PROC PADRE
DIM SEZIONE TESTO
DIM SEZIONE DATI
#PAGE FAULT

VMSTAT

vmstat è un comando che consente di ottenere informazioni sulla memoria virtuale sugli eventi sui processori e su diversi tipi di attività della CPU

vmstat -f dice quante fork sono state eseguite a partire dall'avvio della macchina

vmstat -m mostra informazioni sulle slab

vmstat -s riporta una tabella con il conteggio di quanti eventi di diversi tipi si sono verificati + alcune info generali

Sinossi di vmstat

```
baroglio 514>> vmstat --help
usage: vmstat [-V] [-n] [delay [count]]
        -V prints version.
        -n causes the headers not to be reprinted regularly.
        -a print inactive/active page stats.
        -d prints disk statistics
        -D prints disk table
        -p prints disk partition statistics
        -s prints vm table
        -m prints slabinfo
        -S unit size
        delay is the delay between updates in seconds.
        unit size k:1000 K:1024 m:1000000 M:1048576 (default is K)
        count is the number of updates.
baroglio 515>>
```

vmstat -f

Il risultato mostrato a video è semplicissimo, es:

```
baroglio 503>> vmstat -f
        14991 forks
baroglio 504>>
```

vmstat -s

```
baroglio 516>> vmstat -s
2067208  total memory
672768  used memory
378732  active memory
200000  inactive memory
1394440  free memory
21764  buffer memory
348156  swap cache
4192956  total swap
0  used swap
4192956  free swap
71678 non-nice user cpu ticks
60 nice user cpu ticks
10587 system cpu ticks
494691 idle cpu ticks
5877 IO-wait cpu ticks
294 IRQ cpu ticks
259 softirq cpu ticks
330014 pages paged in
92188 pages paged out
0 pages swapped in
0 pages swapped out
1780101 interrupts
4393343 CPU context switches
1172671730 boot time
25266 forks
```

vmstat -m

```
baroglio 504>> vmstat -m
Cache                  Num    Total   Size   Pages
ip_conntrack_expect      0        0     92      42
ip_conntrack             7       17    232      17
wrap_mdl                 0        0     44      84
dm_tio                   0        0     16     203
dm_io                    0        0     16     203
uhci_urb_priv            1       92     40      92
ip_fib_alias             9      113     32     113
ip_fib_hash              9      113     32     113
clip_arp_cache           0        0    192      20
UNIX                   250      250    384      10
ip_mrt_cache             0        0    128      30
tcp_bind_bucket         10      203     16     203
inet_peer_cache         .....
```

Mostra le cache di RAM usate dal SO, per ogni cache sono riportate informazioni sulla sua struttura e sul suo uso

Ci sono 144 cache su questo portatile

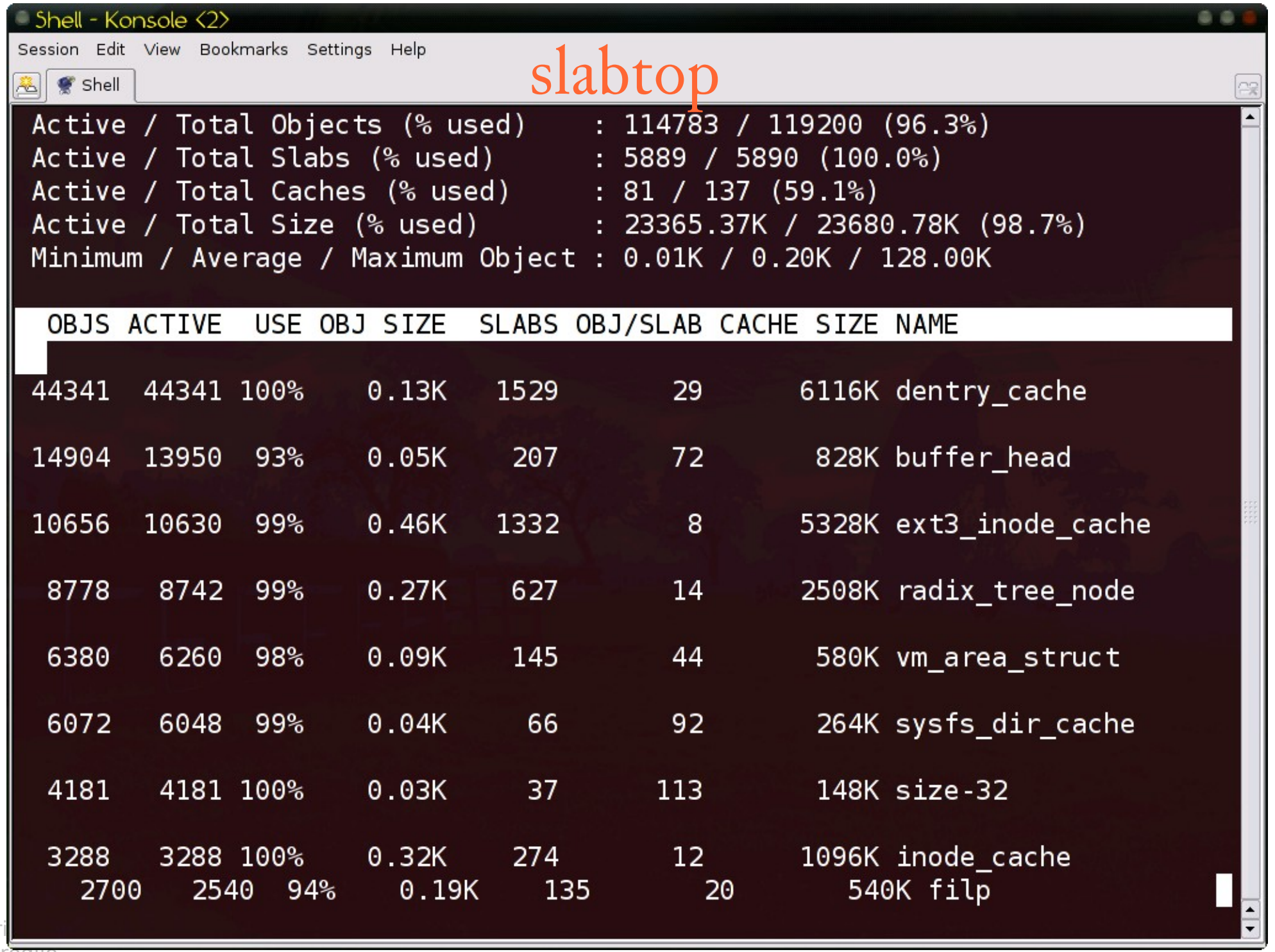
vmstat -m

```
baroglio 504>> vmstat -m
Cache                  Num    Total   Size   Pages
ip_conntrack_expect      0        0     92      42
ip_conntrack             7       17    232      17
wrap_mdl                 0        0     44      84
dm_tio
dm_io
uhci_urb_priv
ip_fib_alias
ip_fib_hash
clip_arp_cache
UNIX
ip_mrt_cache             0        0    128      30
tcp_bind_bucket         10      203     16     203
inet_peer_cache         .....
```

Descrittori dei campi

Cache	nome della cache
Num	numero di oggetti attualmente attivi
Total	numero totale di oggetti disponibili
Size	dimensione del singolo oggetto
Pages	numero di pagine

Ci sono 144 cache su questo portatile


Shell - Konsole <2>
Session Edit View Bookmarks Settings Help
Shell
slabtop
Active / Total Objects (% used) : 114783 / 119200 (96.3%)
Active / Total Slabs (% used) : 5889 / 5890 (100.0%)
Active / Total Caches (% used) : 81 / 137 (59.1%)
Active / Total Size (% used) : 23365.37K / 23680.78K (98.7%)
Minimum / Average / Maximum Object : 0.01K / 0.20K / 128.00K
OBJS ACTIVE USE OBJ SIZE SLABS OBJ/SLAB CACHE SIZE NAME
44341 44341 100% 0.13K 1529 29 6116K dentry_cache
14904 13950 93% 0.05K 207 72 828K buffer_head
10656 10630 99% 0.46K 1332 8 5328K ext3_inode_cache
8778 8742 99% 0.27K 627 14 2508K radix_tree_node
6380 6260 98% 0.09K 145 44 580K vm_area_struct
6072 6048 99% 0.04K 66 92 264K sysfs_dir_cache
4181 4181 100% 0.03K 37 113 148K size-32
3288 3288 100% 0.32K 274 12 1096K inode_cache
2700 2540 94% 0.19K 135 20 540K filp