AWK

Awk è utilizzato per selezionare porzioni di testo e/o sostituirle. In realtà è un vero e proprio linguaggio di programmazione.

La sua sintassi si basa su una serie di coppie *pattern {action statement}* inserite tra apici singoli ***'pattern {action statement}'***.

Awk legge un file, una riga per volta verifica la corrispondenza con il pattern. Se tale corrispondenza è soddisfatta allora awk esegue l'azione tra parentesi graffe.

E' possibile scrivere solo la parte pattern o solo la parte action:

- *solo pattern*: esegue la action di default che è la stampa.
- *solo action*: la corrispondenza è sempre soddisfatta e quindi esegue la action in tutti i casi.

Esempi di Pattern --> BEGIN, END, generiche espressioni regolari tra slash (/ / - attenzione se si vuole scrivere lo slash bisogna metterlo dopo un backslash \/), operazioni booleane tra espressioni regolari (And /.../ && /.../, Or || e Not !)

Esempi di Action --> {print} stampa la linea ed è il default

{print $0} stampa tutta la riga appena letta

{print $0 "ciao"} stampa la riga appendendo in coda ad ognuna 'ciao'

Awk legge le righe del file che gli diamo in input e le separa in una serie di campi basando la suddivisione su di un certo separatore di campo. Esiste un separatore di campo standard e anche un separatore di record standard, ma è possibile configurarne di personalizzati. Dopo la suddivisone awk inserisce i campi in $1, $2, $3 ecc. $0 indica l'intero record.

Esempi base:

awk '{print $1}' stampa il primo campo di tutte le righe.

awk '$2=="pippo" ' stampa tutte le righe che hanno come secondo campo pippo

awk '$2=="pippo" {print $3}' stampa il terzo campo delle righe che hanno come secondo pippo

Esempi pratici a partire dall'output del comando ps aux

nimloth@nimloth-mac:~$ ***ps aux***

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2948  1852 ?        Ss   16:27   0:01 /sbin/init
root       155  0.0  0.0      0     0 ?        S<   16:27   0:00 [kseriod]
root      2154  0.0  0.0      0     0 ?        S<   16:27   0:00 [ata_aux]
daemon    5435  0.0  0.0   1964   432 ?        Ss   16:27   0:00 /usr/sbin/atd
root      5449  0.0  0.0   2332   904 ?        Ss   16:27   0:00 /usr/sbin/cron
root      5477  0.0  0.1  10152  2420 ?        Ss   16:27   0:00 /usr/sbin/apache2 -k start
nimloth   5601  0.0  0.3  28080  7628 ?        Ssl  16:27   0:00 x-session-manager
nimloth   5654  0.3  1.1  41336 22568 ?        S    16:28   0:07 gnome-panel --sm-client-id default1
nimloth   5750  0.0  0.3  19692  7032 ?        S    16:28   0:00 vino-session --sm-client-id default5
nimloth   6593  0.0  0.1   5748  3112 pts/0    Ss   16:43   0:00 bash
nimloth   6651  0.0  0.0   2624  1004 pts/0    R+   16:59   0:00 ps aux
```

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

Ps aux restituisce: l'utente che ha lanciato il processo, il pid del processo, la precentuale di CPU e di meomria utilizzata dal processo, la Virtual SiZe (memoria virtuale assegnata al processo), la Resident Set Size (memoria effettivamente usata dal processo), lo stato in cui si trova, quando è partito, quanto tempo di Cpu ha utilizzato e quale comando lo ha fatto partire.

Vogliamo vedere solo i comandi relativi ad un certo utente, quindi filtrare il primo campo:

nimloth@nimloth-mac:~$ ***ps aux | awk '$1=="nimloth"'***

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
nimloth   6682  0.0  0.0   1756   528 ?        S    13:15   0:00 /bin/sh /usr/bin/firefox
nimloth   6775  1.0  1.0  76336 21432 ?        Sl   13:25   0:00 gnome-terminal
nimloth   6778  0.0  0.0   2668   744 ?        S    13:25   0:00 gnome-pty-helper
nimloth   6779  0.1  0.1   5756  3116 pts/0    Ss   13:25   0:00 bash
nimloth   6797  0.0  0.0   2624  1004 pts/0    R+   13:26   0:00 ps aux
nimloth   6798  0.0  0.0   1920   544 pts/0    S+   13:26   0:00 awk $1=="nimloth"
nimloth   5598  0.0  0.0  13416  1832 ?        SL   16:27   0:00 /usr/bin/gnome-keyring-daemon -d
nimloth   5601  0.0  0.3  28080  7628 ?        Ssl  16:27   0:00 x-session-manager
```

Se invece vogliamo escludere un certo utente, filtriamo il primo campo con un NOT:

nimloth@nimloth-mac:~$ ***ps aux | awk '$1!="nimloth"'***

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2948  1852 ?        Ss   16:27   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S<   16:27   0:00 [kthreadd]
root        11  0.0  0.0      0     0 ?        S<   16:27   0:00 [khelper]
root        31  0.0  0.0      0     0 ?        S<   16:27   0:00 [kblockd/0]
root      4507  0.0  0.0   1696   520 tty3     Ss+  16:27   0:00 /sbin/getty 38400 tty3
root      4508  0.0  0.0   1692   512 tty1     Ss+  16:27   0:00 /sbin/getty 38400 tty1
root      4509  0.0  0.0   1696   520 tty6     Ss+  16:27   0:00 /sbin/getty 38400 tty6
syslog    4842  0.0  0.0   1916   704 ?        Ss   16:27   0:00 /sbin/syslogd -u syslog
klog      4896  0.0  0.0   2504  1396 ?        Ss   16:27   0:00 /sbin/klogd -P /var/run/klogd/kmsg
avahi     5255  0.0  0.0   2732   456 ?        Ss   16:27   0:00 avahi-daemon: chroot helper
daemon    5435  0.0  0.0   1964   432 ?        Ss   16:27   0:00 /usr/sbin/atd
root      5488  0.0  0.0  10152  1676 ?        S    16:27   0:00 /usr/sbin/apache2 -k start
```

Possiamo fare la stessa cosa anche con una sintassi più completa e che usi l'if:

nimloth@nimloth-mac:~$ ***ps aux | awk '{if($1=="nimloth")print $0}'***

```
nimloth   5598  0.0  0.0  13416  1832 ?        SL   16:27   0:00 /usr/bin/gnome-keyring-daemon -d
nimloth   5601  0.0  0.3  28080  7628 ?        Ssl  16:27   0:00 x-session-manager
nimloth   5750  0.0  0.3  19692  7032 ?        S    16:28   0:00 vino-session --sm-client-id default5
nimloth   5751  0.0  0.3  19384  7344 ?        S    16:28   0:00 bluetooth-applet
nimloth   5754  0.0  0.5  25128 12244 ?        S    16:28   0:00 update-notifier
nimloth   5762  0.0  0.8  30508 16800 ?        S    16:28   0:00 python /usr/bin/fusion-icon
nimloth   6593  0.0  0.1   5748  3116 pts/0    Ss   16:43   0:00 bash
nimloth   6613  0.5  0.9  34836 18484 ?        S    16:44   0:06 gedit
nimloth   6658  0.0  0.0   2624  1000 pts/0    R+   17:01   0:00 ps aux
nimloth   6659  0.0  0.0   1920   548 pts/0    S+   17:01   0:00 awk {if($1=="nimloth")print $0}
```

Possiamo usare anche delle variabili e le operazioni matematiche all'interno di Awk.

Possiamo definire una variabile stringa e chiedere ad awk di stamparla:

nimloth@nimloth-mac:~$ ***awk -v x="ciao" 'BEGIN {print x}'***
ciao

Oppure fargli eseguire operazioni con i numeri:

nimloth@nimloth-mac:~$ ***ps aux | awk 'END {print "5"+"6"}'***

11

Awk considera quello che scriviamo come delle stringhe ma è in grado di sommare il loro valore.

Nella lista dei processi RSS è il, sesto campo, quindi:

nimloth@nimloth-mac:~$ ***ps aux | awk '{print $6}'***

RSS

1852

0

1232

1676

22568

30080

536

10196

16284

476

23948

Creiamo una variabile s in cui sommiamo tutti gli RSS e poi la stampiamo:

nimloth@nimloth-mac:~$ ***ps aux | awk '{s=s+$6} END {print s}'***

479172

RSS considera anche la memoria condivisa (ad esempio quella delle librerie condivise) e anche quella viene sommata. In realtà ha convertito anche la stringa "RSS" in un numero, ma non cambia il risultato perché vale zero.

Possiamo specificare se lo vogliamo l'inizializzazione della variabile s, ma in questo caso awk deve sapere che questa operazione deve essere fatta solo all'inizio. Usiamo BEGIN:

nimloth@nimloth-mac:~$ ***ps aux | awk 'BEGIN {s=0}{s=s+$6} END{print s}'***

638532

Se non usassimo BEGIN awk inizializzerebbe s a 0 ogni volta che passa da un record all'altro.

nimloth@nimloth-mac:~$ ***ps aux | awk '{s=0}{s=s+$6} END{print s}'***

500

E otterremmo solo l'RSS dell'ultimo record.

Variabili di Awk

Awk ha delle variabili proprie predefinite, per esempio NR (Number of Record) o NF (Number of Fields).

Filtriamo l'output impedendo che stampi la prima riga:

nimloth@nimloth-mac:~$ ***ps aux | awk 'NR!=1'***

root 1 0.0 0.0 2948 1852 ? Ss 16:27 0:01 /sbin/init

root 2 0.0 0.0 0 0 ? S< 16:27 0:00 [kthreadd]

root 3 0.0 0.0 0 0 ? S< 16:27 0:00 [migration/0]

root 4 0.0 0.0 0 0 ? SN 16:27 0:00 [ksoftirqd/0]

nimloth 6906 0.0 0.0 1756 528 ? S 16:19 0:00 /bin/sh /usr/bin/firefox

nimloth 6972 0.0 0.1 5712 3112 pts/0 Ss 10:22 0:00 bash

nimloth 7014 4.1 2.6 69384 53420 ? Sl 10:30 1:54 wish8.5 /usr/bin/amsn

nimloth 5750 0.0 0.3 19692 7032 ? S 16:28 0:00 vino-session --sm-client-id default5

nimloth 6700 0.1 0.1 5748 3128 pts/1 Ss+ 17:09 0:00 bash

nimloth 6719 0.0 0.0 2624 1004 pts/0 R+ 17:10 0:00 ps aux

nimloth 6720 0.0 0.0 1920 544 pts/0 R+ 17:10 0:00 awk NR!=1

Allora possiamo rifare la somma di prima evitando che sommi anche RSS:

nimloth@nimloth-mac:~$ ***ps aux | awk 'NR!=1' | awk '{s=s+$6} END {print s}'***

485956

NF è il numero di campi quindi se per esempio vogliamo l'ultimo campo senza dover conoscerne a priori il numero possiamo farlo così:

nimloth@nimloth-mac:~$ ***ps aux | awk '{print $NF}'***

COMMAND

/sbin/init

[kthreadd]

/usr/sbin/gdm

bash

aux

$NF}

FS e RS sono rispettivamente il Field Separator e il Record Separator. Il separatore di campo di default è lo spazio vuoto, mentre quello di record è l'a capo (\n). Possiamo per dire ad awk di utilizzare un altro separatore. Per farlo impostiamo la variabile di awk FS ed utilizziamo l'argomento -v che serve appunto per assegnare un valore ad una variabile di awk.

Altri esempi pratici a partire dalla stampa del file /etc/passwd:

nimloth@nimloth-mac:~$ ***cat /etc/passwd***

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

dhcp:x:100:101::/nonexistent:/bin/false

avahi:x:106:114:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false

nimloth:x:1000:1000:Iriel Nimloth,,,:/home/nimloth:/bin/bash

Vogliamo selezionare gli utenti che usano una certa shell, che è l'ultimo campo, e stampare il nome di questi utenti, ma in questo file i campi sono separati da : non da spazi vuoti quindi settimao anche il FS:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '$NF=="/bin/sh" {print $1}'***

daemon

bin

irc

gnats

nobody

Conto quanti sono usando una variabile che incremento ad ogni passaggio in cui la corrispondenza $NF=="/bin/sh" è verificata.

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '$NF=="/bin/sh" {print $1; a=a +1}END {print a}'***

daemon

bin

irc

gnats

nobody

16

Vediamone una variante:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk 'BEGIN {FS=":"} $NF=="/bin/sh" {print $1; a=a+1}END {print a}'***

daemon

bin

irc

gnats

nobody

16

In questo caso però non potrei prendere il valore da assegnare ad FS da una variabile d'ambiente perché sta all'interno degli apici .

Stampiamo le shell:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{print $7}'***

/bin/bash

/bin/sh

/bin/sync

/bin/sh

/bin/sh

/bin/false

/bin/false

/bin/bash

Le ordiniamo, eliminiamo i duplicati e li contiamo:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{print $7}' | sort | uniq -c***

```
      3
      2 /bin/bash
      9 /bin/false
     16 /bin/sh
      1 /bin/sync
```

Eliminiamo le righe che hanno campo vuoto per la shell, utilizzando ^$ che combina i campi vuoti:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{print $7}' | grep -v '^$' | sort | uniq -c***

```
      2 /bin/bash
      9 /bin/false
     16 /bin/sh
      1 /bin/sync
```

Infine ordiniamo numericamente con valori decrescenti:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{print $7}' | grep -v '^$' | sort | uniq -c | sort -rn***

```
     16 /bin/sh
      9 /bin/false
      2 /bin/bash
      1 /bin/sync
```

Vogliamo stampare le due colonne in maniera invertita:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{print $7}' | grep -v '^$' | sort | uniq -c | sort -rn | awk '{print $2 $1}'***

```
/bin/sh16
/bin/false9
/bin/bash2
/bin/sync1
```

Mettiamo uno spazio per renderlo più leggibile.

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{print $7}' | grep -v '^$' | sort | uniq -c | sort -rn | awk '{print $2 " " $1}'***

/bin/sh 16

/bin/false 9

/bin/bash 2

/bin/sync 1

Si può notare che la concatenazione di stringhe in awk si fa semplicemente mettendo in coda le stringhe.

Proviamo a separare con un tabulatore:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{print $7}' | grep -v '^$' | sort | uniq -c | sort -rn | awk '{print $2 "\t" $1}'***

/bin/sh 16

/bin/false 9

/bin/bash 2

/bin/sync 1

Purtroppo il tabulatore però non funziona con meno di 8 caratteri.

In awk è possibile utilizare la printf come in c:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{print $7}' | grep -v '^$' | sort | uniq -c | sort -rn | awk '{printf "%15s %5s\n", $2, $1}'***

/bin/sh 16

/bin/false 9

/bin/bash 2

/bin/sync 1

<u>*Strutture di Controllo*</u>

Nei comandi awk possiamo inserire anche le classiche strutture di controllo per i cicli, come ad esempio il for. Vediamone un applicazione:

Abbiamo detto che awk suddivide ogni riga che legge in $1, $2, $3 ecc. Possiamo iterare su questi $i. Nell'esempio seguente per ogni $i stampiamo il relativo campo tra parentesi angolari.

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{ for(i=1; i<=NF; i++){printf "<%s> ", $i } }'***

<root> <x> <0> <0> <root> </root> </bin/bash> <daemon> <x> <1> <1> <daemon> </usr/sbin> </bin/sh> <bin> <x> <2> <2> <bin> </bin> </bin/sh> <sys> <x> <3> <3> <sys> </dev> </bin/sh> <sync> <x> <4> <65534> <sync> </bin> </bin/sync> <games> <x> <5> <60> <games> </usr/games> </bin/sh> <x> <100> <101> <> </nonexistent> </bin/false> <syslog> <x> <101> <102> <> </home/syslog> </bin/false> <klog> <x> <102> <103> <> </home/klog> </bin/false> <messagebus> <x> <103> <109> <> </var/run/dbus> </bin/false> <hplip> <x> <104> <7> <HPLIP system user,,,> </bin/false> <gdm> <x> <108> <118> <Gnome Display Manager> </var/lib/gdm> </bin/false> <nimloth> <x> <1000> <1000> <Iriel Nimloth,,,> </home/nimloth> </bin/bash>

Aggiungiamo un a capo per mantenere suddivise le righe:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{ for(i=1; i<=NF; i++){printf "<%s> ", $i } print "" }'***

<root> <x> <0> <0> <root> </root> </bin/bash>

<daemon> <x> <1> <1> <daemon> </usr/sbin> </bin/sh>

<bin> <x> <2> <2> <bin> </bin> </bin/sh>

<uucp> <x> <10> <10> <uucp> </var/spool/uucp> </bin/sh>

<avahi> <x> <106> <114> <Avahi mDNS daemon,,,> </var/run/avahi-daemon> </bin/false>

<haldaemon> <x> <107> <116> <Hardware abstraction layer,,,> </home/haldaemon> </bin/false>

<u>Sostituzione di Stringhe</u>

Abbiamo visto che con tr possiamo sostituire caratteri. Con awk possiamo sostituire intere stringhe o sottostringhe. Le parole chiave da utilizzare sono sub per sostituire la prima occorrenza della stringa e gsub per sostituire tutte le occorrenze.

La sintassi è:

- *gsub(/A/,"B"), dove al posto di A va un' espressione regolare che matchi la stringa da sostituire e al posto di B va la stringa da inserire al posto di A. In questo modo la modifica avviene in tutta la riga.*

- *gsub(/A/,"B", $i), specificando il campo da sostituire la sostituzione avverrà solo in quel punto.*

Stampiamo il file passwd modificando tutti gli slash in trattini. (Attenzione al fatto che lo slash è un carattere speciale e dobbiamo scriverlo \/).

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{gsub(/\//,"-");print $0 }'***

root:x:0:0:root:-root:-bin-bash

daemon:x:1:1:daemon:-usr-sbin:-bin-sh

bin:x:2:2:bin:-bin:-bin-sh

sys:x:3:3:sys:-dev:-bin-sh

sync:x:4:65534:sync:-bin:-bin-sync

games:x:5:60:games:-usr-games:-bin-sh

Applichiamo la sostituzione solo all'ultimo campo:

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' '{gsub(/\//,"-",$7);print $0 }'***

root x 0 0 root /root -bin-bash

daemon x 1 1 daemon /usr/sbin -bin-sh

man x 6 12 man /var/cache/man -bin-sh

lp x 7 7 lp /var/spool/lpd -bin-sh

mail x 8 8 mail /var/mail -bin-sh

news x 9 9 news /var/spool/news -bin-sh

In questo caso vediamo che non ci sono più i due punti. Dove ha trovato il Field Separator in input e lo ha sostituito in output con l'OFS (Outpu FS) di default che è lo spazio. Chiamiamo quindi lo stesso comando specificando come OFS i due punti.

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' -v OFS=':' '{gsub(/\//,"-",$7);print $0 }'***

root:x:0:0:root:/root:-bin-bash

daemon:x:1:1:daemon:/usr/sbin:-bin-sh

bin:x:2:2:bin:/bin:-bin-sh

sys:x:3:3:sys:/dev:-bin-sh

sync:x:4:65534:sync:/bin:-bin-sync

games:x:5:60:games:/usr/games:-bin-sh

man:x:6:12:man:/var/cache/man:-bin-sh

lp:x:7:7:lp:/var/spool/lpd:-bin-sh

mail:x:8:8:mail:/var/mail:-bin-sh

Le sostituzioni viste finora erano di un singolo carattere, vediamo vere e proprie stringhe.

Sostituiamo la stringa /bin/sh con la stringa pippo

nimloth@nimloth-mac:~$ ***cat /etc/passwd***

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

sys:x:3:3:sys:/dev:/bin/sh

games:x:5:60:games:/usr/games:/bin/sh

man:x:6:12:man:/var/cache/man:/bin/sh

nimloth@nimloth-mac:~$ ***cat /etc/passwd | awk -v FS=':' -v OFS=':' '$7=="/bin/sh" {$7="pippo"} {print $0 }'***

daemon:x:1:1:daemon:/usr/sbin:pippo

bin:x:2:2:bin:/bin:pippo

sys:x:3:3:sys:/dev:pippo

games:x:5:60:games:/usr/games:pippo

man:x:6:12:man:/var/cache/man:pippo

RS: Record Separator

Il Record Separator di default in awk è l'a capo. Abbiamo bisogno di un output che abbia righe molto lunghe...Stampiamo l'elenco dei pacchetti di un certo repository:

nimloth@nimloth-mac:~$ ***less /var/lib/apt/lists/security.ubuntu.com_ubuntu_dists_gutsy-security_universe_binary-i386_Packages***

Package: apache2-mpm-itk

Priority: extra

Section: universe/net

Installed-Size: 428

Maintainer: Steinar H. Gunderson <sesse@debian.org>

Architecture: i386

Version: 2.2.3-04-3build4.1

Provides: apache2-mpm, apache2, httpd, httpd-cgi

Depends: apache2.2-common (= 2.2.4-3ubuntu0.1), libapr1, libaprutil1, libc6 (>= 2.6-1), libdb4.4, libexpat1 (>= 1.95.8), libldap2 (>= 2.1.17-1), libpcre3 (>= 4.5), libpq5, libsqlite3-0 (>= 3.4.2), libuuid1 [...]

La riga che indica le dipendenze non entra interamente all'interno della larghezza dello schermo, e come quella molte altre. Questo fenomeno abbassa di molto la leggibilità dell'output e la distinzione a occhio delle diverse voci. Utilizziamo quindi il RS per selezionare ad esempio solo il nome del pacchetto:

nimloth@nimloth-mac:~$ ***cat /var/lib/apt/lists/security.ubuntu.com_ubuntu_dists_gutsy-security_universe_binary-i386_Packages | awk -v FS="\n" -v RS="" '{print$1}'***

Package: apache2-mpm-itk

Package: audacity

Package: bcp

Package: cacti

Package: clamav

[...]

Package: yarssr

Package: zabbix-agent

Package: zabbix-frontend-php

Package: zabbix-server-mysql

Package: zabbix-server-pgsql

Vogliamo solo il nome del pacchetto e non la dicitura Package: quindi applichiamo un ulteriore filtro in cascata che stampa solo il secondo campo :

nimloth@nimloth-mac:~$ ***cat /var/lib/apt/lists/security.ubuntu.com_ubuntu_dists_gutsy-security_universe_binary-i386_Packages | awk -v FS="\n" -v RS="" '{print$1}' | awk '{print $2}'***

apache2-mpm-itk

audacity

bcp

cacti

clamav

[...]

yarssr

zabbix-agent

zabbix-frontend-php

zabbix-server-mysql

zabbix-server-pgsql

Abbiamo visto in precendenza come sostituire delle stringhe attraverso la direttiva gsub di awk. Possiamo quindi scrivere la seguente variante del comando appena eseguito in modo da renderlo indipendente dalla posizione del campo Package sostituendo la stringa con uno spazion vuoto.

Questo approccio in effetti è più corretto.

nimloth@nimloth-mac:~$ ***cat /var/lib/apt/lists/security.ubuntu.com_ubuntu_dists_gutsy-security_universe_binary-i386_Packages | awk -v FS="\n" -v RS="" '{print$1}' | awk '{gsub(/Package: /,""); print $0}'***

apache2-mpm-itk

audacity

bcp

cacti

clamav

[...]

yarssr

zabbix-agent

zabbix-frontend-php

zabbix-server-mysql

zabbix-server-pgsql