Ecco una brevissima, incompleta e -purtroppo- poco precisa guida su udev. Ero nella necessita' di creare una regola che all'inserimento di una chiavetta usb, eseguisse il mount ed un rsync. Son riuscito nell'impresa, ma onestamente non ho capito molto di udev :-(
Chi vuole aggiungere qualcosa di sua fonte e' il benvenuto.
Ecco una parziale traduzione

udev - Linux gestione dinamica dei devices

DESCRIZIONE

udev e' il punto di congiunzione tra gli eventi dei devie e l'ambiente software di sistema. Esso amministra i permessi dei device
e puo' creare symlink aggiuntivi, oppure rinominare le schede di rete. Il kernel assegna nomi-device a volte imprevedibili,
basati sull'ordine col quale vengono riconosciuti. E' sicuramente piu' comodo utilizzare nomi-device significativi in sostituzione
di cio' che assegna il kernel
il demone udevd riceve gli eventi-devices direttamente dal kernel, ed esegue delle azioni in base a delle regole, locate in
/etc/udev/rules.d , oppure in /lib/udev/rules.d . In caso di conflitto le regole locare sotto /etc/udev/rules.d hanno la
priorita'.

CONFIGURAZIONE

i files di configurazione di udev son locati in /etc/udev ed in /lib/udev . Le linee vuote e le linee che iniziano con " # "
vengono ignorate

Files di configurazione:

udev si aspetta di trovare la configurazione principale in /etc/udev/udev.conf. Essa consiste in un gruppo di regole e di
variabili che consentono di sovrascrivere i nomi predefiniti dei dispositivi. Le seguenti variabili possono assumere i
seguenti valori:

udev_root
Indica dove inserire i nodi nel filesystem. Il valore predefinito e' /dev .

udev_log
The logging priority. Valid values are the numerical syslog priorities or their textual representations: err, info and debug.

Rules files
Le regole sono lette dai files posizionati della directory di default delle regole /lib/udev/rules.d, le regole personalizzate
son lette da /etc/udev/rules.d e le regole temporanee son lette da /run/udev/rules.d . Tutti i files son processati in ordine
alfabetico, indipendentemente da quale directory siano contenute. Le regole presenti in /etc/udev/rules.d han la preceedenza in

caso di conflitto. Cio' puo' essere utile per ignorare le regole predefinite (per esempio). I files delle regole devono
obbligatoriamente avere l'estensione .rules . Altre estensioni sono ignorate.
All'interno dei files delle regole possiamo trovare due tipi di direttive: confronti ed assegnazioni. Se tutti i confronti presenti
nella linea sono soddisfatti ecco che la direttiva assegnazione viene eseguita. Tramite le regole e' possibile rinominare le schede
di rete, creare link ai dispositivi, oppure lanciare specifici comandi. Gli operatori validi all'interno del file delle regole
sono:

==
compara un'uguaglianza.

!=
compara una differenza .

=
assegna un valore .

+=
Aggiungere il valore di una chiave che contiene un elenco di voci. (traduzione letterale di google)

:=
Assign a value to a key finally; disallow any later changes, which may be used to prevent changes by any later rules.

Le seguenti direttive servono per effettuare una comparazione sulla periferica, controllandone le caratteristiche al
fine di ottenere l'identificazione della periferica e di applicare la regola.

ACTION
confronta il nome dell'azione-evento . puo' essere "add" o "remove"

DEVPATH
confronta il path del device cui l'evento si riferisce

KERNEL
confronta il nome che il kernel assegna al device cui l'evento si riferisce

NAME
assegna il nome all'interfaccia di rete (e solo a lei, se ho capito bene)

SYMLINK
crea un link ad un determinato device. In pratica /dev/sdb1 puo' essere (anche) chiamata " DISCO_2 " , O con
altro nome.

SUBSYSTEM
confronta il subsystem in riferimento ad un evento/device

DRIVER
confronta il nome del driver, e l'evento del device. Utilizzabile solo per i dispositivi che obbligatoriamente utilizzano
un determinato ed unico driver.

ATTR{filename}
confronta i valori degli attributi sys riferiti all'evento/device. Gli spazi bianchi vengono ignorati.

KERNELS
cerca il path del device e controlla un'eventuale corrispondenza

SUBSYSTEMS
cerca il path del device e controlla un'eventuale corrispondenza del subsystem

DRIVERS
cerca il path del device e controlla un'eventuale corrispondenza del driver utilizzato.

ATTRS{filename}
confronta il devpath di un determinato device riferito ai valori del sysfs . Qualora vengano specificati diversi ATTRS
tutti devono corrispondere a quanto specificato. Non deve contenere spazi bianchi.

ENV{key}
confronta il valore della proprieta' del device

TAG
Match against a device tag.

TEST{octal mode mask}
controlla l'esistenza di un file, la notazione ottale puo' essere specificata se occorre.

PROGRAM
esegue un programma. Il valore di ritorno e' " vero " se il programma viene eseguito con successo.

RESULT
Trova (individua) la stringa restituita dall'esecuzione di un programma precedente. La stessa (sringa) puo' essere
utilizzata nella medesima regole oppure in regole successive.

La maggior parte dei campi e' in stile bash, sono supportate le seguenti caratteristiche:

*
corrisponde a nessuna oppure ad un numero imprecisato (qualunque) di caratteri.

?

Corrisponde ad un solo carattere (qualunque)

[]
corrisponde ad una qualsiasi occorrenza, i caratteri contenuti all'interno delle parentesi quadre indicano un inervallo.
Per esempio la sequenza " tty[SR] puo' indicare ttyS oppure ttyR , mentre sd[c-e] puo' indicare sdc , sdd , sde .
il punto esclamativo " ! " , inverte il significato, esattamente come nella bash.

I seguenti nomi possono assumere i seguenti significati:

NAME
Il nome col quale un'interfaccia di rete viene rinominata. Il kernel assegna un nome al dispositivo di rete prima dell'intervento
di udev, tramite la direttiva NAME (all'interno di una regola) ecco che possiamo cambiare tale nome.

SYMLINK
Crean un link ad un determinato dispositivo (identificato dalla regola opportuna), senza modificare il nome primario
del dispositivo stesso. E' possibile specificare piu' di un SYMLINK ed assegnare quindi piu' di un link al medesimo
dispositivo.

OWNER, GROUP, MODE
Specifica le "permission" (i permessi di accesso/utilizzo) di un determinato device.

ATTR{key}
    The value that should be written to a sysfs attribute of the event device.

ENV{key}
    Set a device property value. Property names with a leading ´.´ are not stored in the database or exported to external tool or
    events.

TAG
    Attach a tag to a device. This is used to filter events for users of libudev´s monitor functionality, or to enumerate a group
    of tagged devices. The implementation can only work efficiently if only a few tags are attached to a device. It is only meant
    to be used in contexts with specific device filter requirements, and not as a general-purpose flag. Excessive use might result
    in inefficient event handling.

RUN
Esegue un determinato programma (script, eseguibile) nel momento in cui una determinata regola viene soddisfatta.
Qualora l'esecuzione di tale programma (script, eseguibile) risulti onerosa, ecco che le altre regole udev potrebbero
subire dei ritardi nella loro esecuzione. Occorre specificare il path assoluto del programma (script, eseguibile) , in assenza
di questo ecco che viene ritenuto valido il path /lib/udev/

LABEL
Identifica una zona di codice ove e' possibile "saltare" (= dirigere il corse della regola in esecuzione)

GOTO
Salta (= si dirige) nella zona di codice identificata con LABEL

IMPORT{type}
    Import a set of variables as device properties, depending on type:

    program
        Execute an external program specified as the assigned value and import its output, which must be in environment key format.

Path specification, command/argument separation, and quoting work like in RUN.

file
Import a text file specified as the assigned value, which must be in environment key format.

db
Import a single property specified as the assigned value from the current device database. This works only if the database is already populated by an earlier event.

cmdline
Import a single property from the kernel commandline. For simple flags the value of the property will be set to ´1´.

parent
Import the stored keys from the parent device by reading the database entry of the parent device. The value assigned to IMPORT{parent} is used as a filter of key names to import (with the same shell-style pattern matching used for comparisons).

If no option is given, udev will choose between program and file based on the executable bit of the file permissions.

WAIT_FOR
Wait for a file to become available or until a 10 seconds timeout expires. The path is relative to the sysfs device, i. e. if no path is specified this waits for an attribute to appear.

OPTIONS
Rule and device options:

link_priority=value
Specify the priority of the created symlinks. Devices with higher priorities overwrite existing symlinks of other devices. The default is 0.

event_timeout=
Number of seconds an event will wait for operations to finish, before it will terminate itself.

string_escape=none|replace
Usually control and other possibly unsafe characters are replaced in strings used for device naming. The mode of replacement can be specified with this option.

static_node=
Apply the permissions specified in this rule to a static device node with the specified name. Static device nodes might be provided by kernel modules, or copied from /lib/udev/devices. These nodes might not have a corresponding kernel device at the time udevd is started, and allow to trigger automatic kernel module on-demand loading.

watch
Watch the device node with inotify, when closed after being opened for

```
        writing, a change uevent will be synthesised.

    nowatch
        Disable the watching of a device node with inotify.

The NAME, SYMLINK, PROGRAM, OWNER, GROUP, MODE and RUN fields support simple
printf-like string substitutions. The RUN format chars
gets applied after all rules have been processed, right before the program is
executed. It allows the use of device properties set
by earlier matching rules. For all other fields, substitutions are applied while the
individual rule is being processed. The
available substitutions are:

$kernel, %k
    il nome che il kernel assegna al device

$number, %n
il numero che il kernel assegna al devie. "dev/sda3" ha come numero del kernel " 3 "

$devpath, %p
Il devpath del device

$id, %b
    The name of the device matched while searching the devpath upwards for
    SUBSYSTEMS, KERNELS, DRIVERS and ATTRS.

$driver
    The driver name of the device matched while searching the devpath upwards for
    SUBSYSTEMS, KERNELS, DRIVERS and ATTRS.

$attr{file}, %s{file}
    The value of a sysfs attribute found at the device, where all keys of the rule
    have matched. If the matching device does not
    have such an attribute, and a previous KERNELS, SUBSYSTEMS, DRIVERS, or ATTRS
    test selected a parent device, use the attribute
    from that parent device. If the attribute is a symlink, the last element of the
    symlink target is returned as the value.

$env{key}, %E{key}
    A device property value.

$major, %M
    The kernel major number for the device.

$minor, %m
    The kernel minor number for the device.

$result, %c
    The string returned by the external program requested with PROGRAM. A single part
    of the string, separated by a space character
    may be selected by specifying the part number as an attribute: %c{N}. If the
    number is followed by the ´+´ char this part plus
    all remaining parts of the result string are substituted: %c{N+}

$parent, %P
    The node name of the parent device.
```

```
       $name
           The current name of the device node. If not changed by a rule, it is the name of
           the kernel device.

       $links
           The current list of symlinks, separated by a space character. The value is only
           set if an earlier rule assigned a value, or
           during a remove events.

       $root, %r
           The udev_root value.

       $sys, %S
           The sysfs mount point.

       $tempnode, %N
           The name of a created temporary device node to provide access to the device from
           a external program before the real node is
           created.

       %%
           The ´%´ character itself.

       $$
           The ´$´ character itself.

AUTHOR
       Written by Greg Kroah-Hartman greg@kroah.com and Kay Sievers kay.sievers@vrfy.org.
       With much help from Dan Stekloff and many
       others.

SEE ALSO
       udevd(8), udevadm(8)


UDEVADM(8)                          udevadm                          UDEVADM(8)



NAME
    udevadm - strumento di gestione di udev

SYNOPSIS
       udevadm [--debug] [--version] [--help]

       udevadm info options

       udevadm trigger [options]

       udevadm settle [options]

       udevadm control command

       udevadm monitor [options]

       udevadm test [options] devpath
```

```
DESCRIPTION
    udevadm si aspetta un comando e le opzioni di tale comando. Esso controlla
    il comportamento di runtime di udev, eventi/richieste del kernel, gestisce
    l'evento e fornisce un debug semplice.

OPTIONS
       --debug
    stampa l'output del debug sullo standard error.

       --version
           mostra il numero di  versione

       --help
           mostra un messaggio di aiuto

   udevadm info options
    interroga il database udev in riferimento al dispositivo indicato,
    allo scopo di ottenere informazioni utili alla scrittura delle
    regole  [1]


       --query=type
           Query the database for specified type of device data. It needs the
           --path or --name to identify the specified device. Valid queries
           are: name, symlink, path, property, all.

       --path=devpath
           The devpath of the device to query.

       --name=file
           The name of the device node or a symlink to query

       --root
           The udev root directory: /dev. If used in conjunction with a name
           or symlink query, the query returns the absolute path including the
           root directory.

       --run
           The udev runtime directory: /run/udev.

       --attribute-walk
           Print all sysfs properties of the specified device that can be used
           in udev rules to match the specified device. It prints all devices
           along the chain, up to the root of sysfs that can be used in udev
           rules.

       --export
           Print output as key/value pairs. Values are enclosed in single
           quotes.

       --export-prefix=name
           Add a prefix to the key name of exported values.

       --device-id-of-file=file
           Print major/minor numbers of the underlying device, where the file
           lives on.
```

```
      --export-db
          Export the content of the udev database.

      --version
          Print version.

      --help
          Print help text.

  udevadm trigger [options]
   chiede al kernel gli eventi (le risposte?) dei dispositivi,
   lo scopo e' di raccogliere informazioni per usi successivi.



      --verbose
          Print the list of devices which will be triggered.

      --dry-run
          Do not actually trigger the event.

      --type=type
          Trigger a specific type of devices. Valid types are: devices,
          subsystems. The default value is devices.

      --action=action
          Type of event to be triggered. The default value is change.

      --subsystem-match=subsystem
          Trigger events for devices which belong to a matching subsystem.
          This option can be specified multiple times and supports shell
          style pattern matching.

      --subsystem-nomatch=subsystem
          Do not trigger events for devices which belong to a matching
          subsystem. This option can be specified multiple times and supports
          shell style pattern matching.

      --attr-match=attribute=value
          Trigger events for devices with a matching sysfs attribute. If a
          value is specified along with the attribute name, the content of
          the attribute is matched against the given value using shell style
          pattern matching. If no value is specified, the existence of the
          sysfs attribute is checked. This option can be specified multiple
          times.

      --attr-nomatch=attribute=value
          Do not trigger events for devices with a matching sysfs attribute.
          If a value is specified along with the attribute name, the content
          of the attribute is matched against the given value using shell
          style pattern matching. If no value is specified, the existence of
          the sysfs attribute is checked. This option can be specified
          multiple times.

      --property-match=property=value
          Trigger events for devices with a matching property value. This
          option can be specified multiple times and supports shell style
          pattern matching.
```

```
    --tag-match=property
        Trigger events for devices with a matching tag. This option can be
        specified multiple times.

    --sysname-match=name
        Trigger events for devices with a matching sys device name. This
        option can be specified multiple times and supports shell style
        pattern matching.

udevadm settle [options]
     osserva (controlla) la coda degli eventi di udev, e termina (esce) se
 tutti gli eventi son terminati

    --timeout=seconds
        Maximum number of seconds to wait for the event queue to become
        empty. The default value is 180 seconds. A value of 0 will check if
        the queue is empty and always return immediately.

    --seq-start=seqnum
        Wait only for events after the given sequence number.

    --seq-end=seqnum
        Wait only for events before the given sequence number.

    --exit-if-exists=file
        Stop waiting if file exists.

    --quiet
        Do not print any output, like the remaining queue entries when
        reaching the timeout.

    --help
        Print help text.

udevadm control command
    Modify the internal state of the running udev daemon.

    --log-priority=value
        Set the internal log level of udevd. Valid values are the numerical
        syslog priorities or their textual representations: err, info and
        debug.

    --stop-exec-queue
        Signal udevd to stop executing new events. Incoming events will be
        queued.

    --start-exec-queue
        Signal udevd to enable the execution of events.

    --reload-rules
        Signal udevd to reload the rules files. The udev daemon detects
        changes automatically, this option is usually not needed. Reloading
        rules does not apply any changes to already existing devices.

    --property=KEY=value
        Set a global property for all events.
```

```
       --children-max=value
           Set the maximum number of events, udevd will handle at the same
           time.

       --help
           Print help text.

   udevadm monitor [options]
       Listens to the kernel uevents and events sent out by a udev rule and
       prints the devpath of the event to the console. It can be used to
       analyze the event timing, by comparing the timestamps of the kernel
       uevent and the udev event.

       --kernel
           Print the kernel uevents.

       --udev
           Print the udev event after the rule processing.

       --property
           Also print the properties of the event.

       --subsystem-match=string[/string]
           Filter events by subsystem[/devtype]. Only udev events with a
           matching subsystem value will pass.

       --tag-match=string
           Filter events by property. Only udev events with a given tag
           attached will pass.

       --help
           Print help text.

   udevadm test [options] devpath
       Simulate a udev event run for the given device, and print debug output.

       --action=string
           The action string.

       --subsystem=string
           The subsystem string.

       --help
           Print help text.

AUTHOR
       Written by Kay Sievers kay.sievers@vrfy.org.

SEE ALSO
       udev(7) udevd(8)



udev                                 03/30/2011                          UDEVADM(8)
```

[1]

ipotizziamo di consultare le caratteristiche di /dev/sda , ecco alcune sintassi possibili:
[operiamo su una distribuzione fedora core 15 virtuale]

```
udevadm info --query=all --name=sda

udevadm info --query=all --attribute-walk --name=sda

udevadm info -a -n /dev/sda
```

Ecco il file /etc/udev/udev.con originale della distribuzione slackware 14

```
    # udev.conf
    # The main config file for udev
    #
    # This file can be used to override some of udev's default values for where it
    # looks for files, and where it places device nodes.

    # udev_root - where in the filesystem to place the device nodes
    udev_root="/dev/"

    # udev_log - The initial syslog(3) priority: "err", "info", "debug" or
    # its numerical equivalent.  For runtime debugging, the daemons
    # internal state can be changed with:  udevadm control log_priority=<value>
    udev_log="err"
```

Come notiamo c'e' la diretiva udev_root="/dev/" la quale indica la posizione ove verranno posizionati i files-dispositivo.
NOTA BENE !! nella distribuzione slackware 14 ad ogni inserimento di regole nuove (o modifica di regole esistenti) occorre
riavviare il servizio tramite la sintassi

```
    /etc/rc.d/rc.udev force-restart
```

Senza questa precauzione le modifiche non vengono considerate. Pare che il riavvio del servizio non sia necessario
su fedora core 15. Perlomeno, dalle mie prove e' apparso cosi'.
Ora arriva la parte "pratica", la molla che mi ha portato a scrivere queste righe. La necessita' era di riconoscere
al volo una chiavetta usb da 2 gb di marca kingston , montarla correttamente in /mnt/2gb ed avviare uno script di rsync.
Mi limito a riportare le regole udev che si incaricano del riconoscimento della chiavetta. Operiamo su una fedora core 15
virtuale e su una slackware 14 virtuale. La chiavetta viene riconosciuta come /dev/sdb . Lo

capiamo osservando il file
/var/log/messages . Inoltre in /dev/ vengono aggiunti due devices: /dev/sdb e /dev/sdb1 .
Ora analizziamo la chiavetta
e cerchiamo di scrivere la regola. Primo passo e' ottenere le informazioni specifiche della chiavetta:

```
udevadm info -a -n /dev/sdb


Udevadm info starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.

  looking at device
  '/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/block/sdb':
    KERNEL=="sdb"
    SUBSYSTEM=="block"
    DRIVER==""
    ATTR{range}=="16"
    ATTR{ext_range}=="256"
    ATTR{removable}=="1"
    ATTR{ro}=="0"
    ATTR{size}=="3966976"
    ATTR{alignment_offset}=="0"
    ATTR{discard_alignment}=="0"
    ATTR{capability}=="51"
    ATTR{stat}==""     341        6     2776     3408        0        0        0
    0        0     3405     3405"
    ATTR{inflight}==""       0        0"
    ATTR{events}=="media_change"
    ATTR{events_async}==""
    ATTR{events_poll_msecs}=="-1"

  looking at parent device
  '/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0':
    KERNELS=="3:0:0:0"
    SUBSYSTEMS=="scsi"
    DRIVERS=="sd"
    ATTRS{device_blocked}=="0"
    ATTRS{type}=="0"
    ATTRS{scsi_level}=="0"
    ATTRS{vendor}=="Kingston"
    ATTRS{model}=="DataTraveler 2.0"
    ATTRS{rev}=="PMAP"
    ATTRS{state}=="running"
    ATTRS{timeout}=="30"
    ATTRS{iocounterbits}=="32"
    ATTRS{iorequest_cnt}=="0x19a"
    ATTRS{iodone_cnt}=="0x19a"
    ATTRS{ioerr_cnt}=="0x1"
    ATTRS{evt_media_change}=="0"
    ATTRS{dh_state}=="detached"
    ATTRS{queue_depth}=="1"
    ATTRS{queue_type}=="none"
    ATTRS{max_sectors}=="240"
```

```
  looking at parent device
  '/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0':
    KERNELS=="target3:0:0"
    SUBSYSTEMS=="scsi"
    DRIVERS==""

  looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3':
    KERNELS=="host3"
    SUBSYSTEMS=="scsi"
    DRIVERS==""

  looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0':
    KERNELS=="1-1:1.0"
    SUBSYSTEMS=="usb"
    DRIVERS=="usb-storage"
    ATTRS{bInterfaceNumber}=="00"
    ATTRS{bAlternateSetting}==" 0"
    ATTRS{bNumEndpoints}=="02"
    ATTRS{bInterfaceClass}=="08"
    ATTRS{bInterfaceSubClass}=="06"
    ATTRS{bInterfaceProtocol}=="50"
    ATTRS{supports_autosuspend}=="1"

  looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1/1-1':
    KERNELS=="1-1"
    SUBSYSTEMS=="usb"
    DRIVERS=="usb"
    ATTRS{configuration}==""
    ATTRS{bNumInterfaces}==" 1"
    ATTRS{bConfigurationValue}=="1"
    ATTRS{bmAttributes}=="80"
    ATTRS{bMaxPower}=="200mA"
    ATTRS{urbnum}=="1223"
    ATTRS{idVendor}=="0930"
    ATTRS{idProduct}=="6545"
    ATTRS{bcdDevice}=="0100"
    ATTRS{bDeviceClass}=="00"
    ATTRS{bDeviceSubClass}=="00"
    ATTRS{bDeviceProtocol}=="00"
    ATTRS{bNumConfigurations}=="1"
    ATTRS{bMaxPacketSize0}=="64"
    ATTRS{speed}=="480"
    ATTRS{busnum}=="1"
    ATTRS{devnum}=="2"
    ATTRS{devpath}=="1"
    ATTRS{version}==" 2.00"
    ATTRS{maxchild}=="0"
    ATTRS{quirks}=="0x0"
    ATTRS{avoid_reset_quirk}=="0"
    ATTRS{authorized}=="1"
    ATTRS{manufacturer}=="Kingston"
    ATTRS{product}=="DataTraveler 2.0"
    ATTRS{serial}=="001D0F0CAAB55B8909141484"

  looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1':
    KERNELS=="usb1"
```

```
    SUBSYSTEMS=="usb"
    DRIVERS=="usb"
    ATTRS{configuration}==""
    ATTRS{bNumInterfaces}==" 1"
    ATTRS{bConfigurationValue}=="1"
    ATTRS{bmAttributes}=="e0"
    ATTRS{bMaxPower}=="  0mA"
    ATTRS{urbnum}=="44"
    ATTRS{idVendor}=="1d6b"
    ATTRS{idProduct}=="0002"
    ATTRS{bcdDevice}=="0206"
    ATTRS{bDeviceClass}=="09"
    ATTRS{bDeviceSubClass}=="00"
    ATTRS{bDeviceProtocol}=="00"
    ATTRS{bNumConfigurations}=="1"
    ATTRS{bMaxPacketSize0}=="64"
    ATTRS{speed}=="480"
    ATTRS{busnum}=="1"
    ATTRS{devnum}=="1"
    ATTRS{devpath}=="0"
    ATTRS{version}==" 2.00"
    ATTRS{maxchild}=="8"
    ATTRS{quirks}=="0x0"
    ATTRS{avoid_reset_quirk}=="0"
    ATTRS{authorized}=="1"
    ATTRS{manufacturer}=="Linux 2.6.38.6-26.rc1.fc15.i686.PAE ehci_hcd"
    ATTRS{product}=="EHCI Host Controller"
    ATTRS{serial}=="0000:00:0b.0"
    ATTRS{authorized_default}=="1"

  looking at parent device '/devices/pci0000:00/0000:00:0b.0':
    KERNELS=="0000:00:0b.0"
    SUBSYSTEMS=="pci"
    DRIVERS=="ehci_hcd"
    ATTRS{vendor}=="0x8086"
    ATTRS{device}=="0x265c"
    ATTRS{subsystem_vendor}=="0x0000"
    ATTRS{subsystem_device}=="0x0000"
    ATTRS{class}=="0x0c0320"
    ATTRS{irq}=="10"
    ATTRS{local_cpus}=="ffffffff,ffffffff"
    ATTRS{local_cpulist}=="0-63"
    ATTRS{dma_mask_bits}=="32"
    ATTRS{consistent_dma_mask_bits}=="32"
    ATTRS{enable}=="1"
    ATTRS{broken_parity_status}=="0"
    ATTRS{msi_bus}==""
    ATTRS{companion}==""

  looking at parent device '/devices/pci0000:00':
    KERNELS=="pci0000:00"
    SUBSYSTEMS==""
    DRIVERS==""
```

Ora abbiamo gli elementi sufficienti per scrivere la nostra regola:

```
ACTION=="add", KERNEL=="sd[b-z][0-9]", ATTRS{vendor}=="Kingston",
ATTRS{model}=="DataTraveler 2.0", SYMLINK+="kingston"
```

Tramite questa regola la chiavetta "kingston" modello DataTravel 2.0 viene riconosciuta in modo univoco , e viene generato
il link /dev/kingston che si riferisce a /dev/sdb1 , in questo caso la chiavetta. La regola si applica SOLO ED UNIVOCAMENTE
alla chiavetta in oggetto, oppure ad una chiavetta uguale identica, medesimo modello e medesima marca. Potremmo ottenere
un risultato simile con la regola seguente:

```
ACTION=="add", KERNEL=="sdb1", SYMLINK+="chiavetta"
```

con questa regola il sistema udev crea il link /dev/chiavetta all'atto del riconoscimento DI UN QUALUNQUE DISPOSITIVO
riconosciuto come /dev/sdb1 . Quindi un hard disk esterno, una chiavetta o simili. Attenzione, la regola si applica
solo a /dev/sdb1 . Se in seguito utilizziamo un ulteriore hard disk esterno che prende come device /dev/sdc1 , ecco che
nessuna regola viene applicata.
Ora vogliamo che allinserimento della chiavetta "kingston" si generi il link /dev/kingston e che venga eseguito uno script
locato in /bin/script . Ecco la regola

```
ACTION=="add", KERNEL=="sd[b-z][0-9]", ATTRS{vendor}=="Kingston",
ATTRS{model}=="DataTraveler 2.0", SYMLINK+="kingston", RUN+="/bin/script"
```

Il funzionamento della regola e' stato verificato su slackware 14 e fedora core 15. Ovviamente lo script /bin/script eseguira'
tutto quello che ci interessa.
E' possibile personalizzare la regola e renderla migliore in molte maniere, soprattutto leggendo la parte (del presente documento)
riguardante le varibili cui udev fa riferimento alla periferica. Es , %p , %n , %k .
Inoltre ricordate che su slackware (la prima, l'unica e la migliore), ad ogni modifica delle regole occorre informare il servizio oppure
riavviarlo. /etc/rc.d/rc.udev reload , oppure /etc/rc.d/rc.udev force-restart .
Io ho ottenuto quanto mi prefiggevo, quindi la guida termina qui. Chi vuole aggiungere del suo e' il benvenuto.