



Udev - zarządzanie plikami urządzeń

Piotr Wolny

Pliki urządzeń (ang. *device nodes*), umieszczone w katalogu `/dev`, są jedną z najbardziej charakterystycznych wizytówek systemów uniksowych, w tym oczywiście Linuksa. Dzięki nim możemy w wygodny sposób zorientować się, jaki sprzęt jest dostępny w systemie, łatwo ustawimy dowolne prawa dostępu do poszczególnych urządzeń, czy - w sposób wręcz intuicyjny - wykonamy rozmaite operacje, np. przekierujemy wyjście jakiejś komendy do "czarnej dziury" `/dev/null` albo skopiujemy zawartość całego dysku przy pomocy komendy `dd`.

Katalog `/dev` - oczywiście nieobecny w MS Windows czy DOS - sprawiał tradycyjnie najwięcej problemów początkującym użytkownikom. Jeszcze do niedawna całkiem spory odsetek próśb o pomoc, które na listach dyskusyjnych zgłaszali początkujący użytkownicy Linuksa, dotyczyło odpowiednich uprawnień do plików w `/dev`. Wyszukiwarka *Google* nalicza dziesiątki tysięcy stron, które np. zawierają komendę `chmod 666 /dev/. ...`, będącą w większości przypadków najprostszym i najszybszym sposobem na radzenie sobie z uprawnieniami dostępu do urządzeń. Dzisiaj jednak większość z tych dziesiątek tysięcy stron zawiera nieaktualne informacje, gdyż we współczesnych dystrybucjach Linuksa pliki urządzeń są zarządzane przez oprogramowanie *Udev*. W tym artykule omówię, jak wykonać najprostsze operacje na plikach urządzeń, jeśli nasza dystrybucja używa *udev*, oraz jak korzystać z nowych możliwości, które daje to oprogramowanie.

Dlaczego Udev?

Tradycyjny katalog `/dev` był rozwiązaniem z powodzeniem stosowanym przez bardzo wiele lat. Ostatnio stało się jasne, że jego użyteczność powoli dobiega końca. Jednym z powodów takiego stanu rzeczy stało się wyposażanie naszych komputerów w coraz więcej rozmaitych urządzeń, które są wspierane w Linuksie. W katalogu `/dev` musiał znajdować się plik dla każdego możliwego urządzenia, niezależnie od tego, czy go posiadamy, czy też nie. Co prawda, jeden taki plik zajmuje niewiele miejsca na dysku, ale ilość rozmaitych urządzeń zaczęła powodować, że straty miejsca stały się znaczące. O ile jeszcze w przypadku jąder z serii 2.4.x i starszych dystrybucji, w katalogu `/dev` i podkatalogach było około 5 tysięcy pozycji, to później liczba ta zaczęła gwałtownie rosnąć, przekraczając szybko 20 tysięcy.

Kolejnym "gwoździem do trumny" dla tradycyjnego katalogu `/dev` stały się urządzenia podłączane w czasie pracy systemu, np.



Rysunek 1. Modyfikacje konfiguracji *Udev* mogą być szczególnie pomocne dla posiadaczy takich urządzeń USB, jak uniwersalne czytniki kart

.LINUX+ Live DVD

Po uruchomieniu Linux+ Live DVD można poznać bliżej możliwości i działanie Udev.

Na płycie DVD

Na płycie DVD znajduje się najnowsze jądro Linux.

pamięci USB. Przy każdym kolejnym podłączeniu urządzenie takie może otrzymać inny identyfikator, choćby w przypadku, gdy mamy więcej urządzeń tego typu i podłączamy je na przemian. Wtedy tradycyjne, statyczne pliki urządzeń w `/dev` okazują się niewystarczające. Nie można przecież stworzyć ich nieskończoną ilość, a jakiegokolwiek ograniczenie będzie zawsze komuś przeszkadzało.

Jasne stało się, że aby system mógł się nadal rozwijać i dobrze współpracować z urządzeniami wymiennymi, pliki urządzeń w `/dev` muszą być tworzone dynamicznie, jedynie wtedy, gdy są potrzebne. Nad takim rozwiązaniem prace rozpoczęto bardzo dawno. Ich pierwszym efektem był tzw. *devfs* (*Device File System*). Wywołał on przede wszystkim lata sporów i kontrowersji, aby ostatecznie zostać zastąpionym przez omawiany w tym artykule *Udev*.

Tego drugiego wyróżnia przede wszystkim prostota, elastyczność, wsteczna zgodność ze standardami oraz wiele innych cech, ułatwiających pracę twórcom dystrybucji Linuksa. Jako zwykli użytkownicy docenimy przede wszystkim wygodny sposób administracji prawami dostępu czy możliwość definiowania własnych nazw dla posiadanych urządzeń. Nie bez znaczenia będzie również fakt, że ilość pozycji w katalogu `/dev` i podkatalogach spada do kilkuset - jeśli nie liczyć konsol (urządzeń `tty*` oraz `pty*`), to w moim rozbudowanym systemie jest ich nieco ponad 100.

Uprawnienia urządzeń w udev

Jak wspomniałem wcześniej, praktycznie wszystkie współczesne dystrybucje korzystają z *Udev*.

Możemy to zobaczyć przy pomocy komendy `cat /proc/mounts` lub `df -h`. Ta druga na moim systemie pokazuje m.in.:

```
tmpfs 10M 2,9M 7,2M 29% /dev
```

Widzimy, że w katalogu `/dev` jest zamontowany system plików *tmpfs*, korzystający z 10 MB pamięci RAM. Oznacza to w praktyce, że zawiera on co prawda normalnie wyglądające pliki urządzeń, ale jakiegokolwiek ich modyfikacje - np. zmiana uprawnień - choć są możliwe, nie zostają zapamiętane. Po kolejnym starcie systemu, wszystko wraca do stanu wyjściowego.

W jaki sposób możemy wprowadzić modyfikacje uprawnień? Zaczynamy od znalezienia właściwego pliku konfiguracyjnego

Udev. W dystrybucjach *Debian Sarge* oraz *Ubuntu* będzie to plik `/etc/udev/permissions.rules`, natomiast w *Fedora Core 4* musimy zmodyfikować znacznie bardziej obszerny główny plik konfiguracyjny, czyli `/etc/udev/rules.d/50-udev.nds`, podobnie zresztą w *Mandriva Linux 2006*, z tym, że tam jest to `pk/etc/udev/rules.d/50-mdk.rules`.

Gdy ulubionym edytorem otworzyliśmy właściwy plik, musimy poszukać linii, która opisuje nasze urządzenie. Na ogół będzie to całkiem proste. Przykładowo, gdy chcemy zmienić uprawnienia dostępu do urządzenia `/dev/ttyS0`, w *Fedora Core 4* musimy znaleźć linię

```
KERNEL=="ttyS*", ←
GROUP="uucp", MODE="0660"
```

Jak nietrudno się domyśleć, opisuje ona wszystkie porty szeregowo (urządzenia, których nazwa zaczyna się od `/dev/ttyS`). Zgodnie z jej zapisami, urządzenia te będą należały do grupy *uucp*, która będzie miała do nich pełny dostęp. Możemy oczywiście albo zmienić grupę systemową, albo uprawnienia, np. na `MODE="0666"`, aby wybrani lub wszyscy użytkownicy systemu mieli dostęp do urządzeń na porcie `/dev/ttyS0`. Zdają sobie oczywiście sprawę, że najlepszym rozwiązaniem byłoby dopisanie właściwego użytkownika do grupy systemowej *uucp*, ale przecież nikt nie zabroni nam innego podejścia do uprawnień, tym bardziej, że w tym artykule poznajemy sposoby konfiguracji *Udev*.

Zamiast zmieniać uprawnienia oraz grupę systemową, do której należy plik urządzenia, możemy również zmienić jego właściciela, z domyślnego *root* na jakiegokolwiek innego. Wystarczy w tym celu dopisać w tej samej linii, po przecinku: `OWNER="ktos_łam"`.

Po dokonaniu zmian w pliku konfiguracyjnym *Udev* powinniśmy zrestartować go, poleceniem `/etc/init.d/udev restart`.

Czasem zmiana uprawnień nie będzie taka banalna. Przykładowo, w *Mandriva Linux 2006* linia opisująca nasz port wygląda następująco:

```
KERNEL=="ttyS[0-9]*", SYMLINK+="tts/ ←
%n", GROUP="uucp", OPTIONS="last_rule"
```

Jak widzimy, brak tutaj opcji `MODE`, ale nic nie stoi na przeszkodzie, aby ją dopisać, oddzielając przecinkiem od pozostałych opcji. Jeszcze "dziwniejszy" na pierw-

szy rzut oka wydaje się *Debian Sarge* czy *Ubuntu*, bo w nich nie ma nic na temat `ttyS*`, w zamian za to w `/etc/udev/permissions.rules` znajdujemy linię:

```
SUBSYSTEM=="tty", ←
GROUP="dialout"
```

Oczywiście, na podobnej zasadzie możemy zmienić podaną tu grupę systemową lub dopisać opcję `MODE`. Dłaczego ta linia rozpoczyna się od `SUBSYSTEM`, a nie od `KERNEL`, jak w poprzednich przykładach, postaram się wyjaśnić w dalszej części, przy opisie tworzenia własnych zasad *Udev*.

W powyższych przykładach zmienialiśmy uprawnienia dla całych grup urządzeń, modyfikując jedynie istniejące linie. Możemy również dopisać jakąś nową zasadę dla pojedynczego urządzenia. Jeśli, np. w *Debian Sarge*, chcemy zmienić uprawnienia tylko i wyłącznie dla `/dev/ttyS1`, możemy dopisać do `/etc/udev/permissions.rules` linię:

```
KERNEL=="ttyS1", GROUP="piotr", ←
MODE="0600"
```

Ważne jest, aby linia ta znalazła się poniżej wspomnianej wcześniej definicji uprawnień dla wszystkich urządzeń `tty`, gdyż w przeciwnym razie uprawnienia nie zmienią się. Alternatywnie możemy dołożyć na jej końcu, po przecinku, opcję `OPTIONS+="last_rule"` - wtedy wszystkie kolejne zapisy odnośnie tego urządzenia będą ignorowane.

Na podobnej zasadzie możemy zmieniać uprawnienia do innych urządzeń. Jedyna trudność, na którą możemy się natknąć, to nazwy podsystemów jądra, które nie wszystkim kojarzą się z urządzeniami - przykładowo, `SUBSYSTEM=block` odpowiada za wszystkie urządzenia blokowe, w tym twarde dyski, dyskiety czy pamięci USB.

Jeśli nie zdefiniujemy uprawnień do jakiegoś urządzenia, użyte zostaną domyślne wartości, na stałe wkomponowane w *Udev* - na ogół będzie to użytkownik i grupa *root* oraz uprawnienia `0600`.

Brakujące urządzenia

Istotą *udev* jest tworzenie plików urządzeń w momencie, gdy sprzęt jest podłączony i używany. Co jednak robić, gdy potrzebnego nam urządzenia brakuje w `/dev`? Przykładowo, gdy podłączyliśmy

Listing 1. Przykłady własnych wpisów w konfiguracji *udev*

```
BUS=="scsi", KERNEL=="sd?1", ←
SYSFS{vendor}=="USB 2.0 ", ←
SYMLINK="wolnyflash"
BUS=="scsi", KERNEL=="sd?1", ←
SYSFS{vendor}=="dc/sdcl", ←
SYMLINK="wolnymp3player"
BUS=="scsi", KERNEL=="sd?1", ←
SYSFS{vendor}=="FUJIFILM", ←
SYMLINK="wolnycamera"
KERNEL=="hdb", SYMLINK= ←
"cdwriter dvd nagrywarka"
KERNEL=="null", NAME="%k", ←
SYMLINK="czarna-dziura", ←
MODE="0666"
KERNEL=="mouse[0-9]", ←
SYMLINK="myszki/%n"
KERNEL=="eth*", SYSFS{address}= ←
"00:13:8f:0a:e6:68", ←
NAME="eth5"
BUS=="scsi", SYSFS{vendor}= ←
"Generic", SYSFS{model}= ←
"USB SD Reader", NAME{all_ ←
partitions}="czytnik/sd"
BUS=="scsi", SYSFS{vendor}= ←
"Generic", SYSFS{model}= ←
"USB CF Reader", NAME{all_ ←
partitions}="czytnik/cf"
BUS=="scsi", SYSFS{vendor}= ←
"Generic", SYSFS{model}= ←
"USB SM Reader", NAME{all_ ←
partitions}="czytnik/sm"
BUS=="scsi", ←
SYSFS{vendor}="Generic ", ←
SYSFS{model}=="USB MS Reader", ←
NAME{all_partitions}="czytnik/ms"
```

odbiornik *Bluetooth*, a w systemie nadal brak urządzenia */dev/rfcomm0*? Przede wszystkim nie powinniśmy zwać winy na *Udeo*. On utworzy dla nas plik urządzenia, w momencie, gdy jądro doda nowy sprzęt. To drugie dzieje się na ogół w momencie załadowania jakiegoś modułu jądra, więc większość problemów z brakującymi urządzeniami w */dev* wynika właśnie z braku jakiegoś modułu. Co prawda, w przypadku większości urządzeń, Linux powinien samodzielnie łączyć potrzebny moduł, ale czasem wciąż musimy posłużyć się komendą *modprobe*, np. *modprobe rfcomm*. Poszukując winnego braku automatycznego ładowania modułów powinniśmy raczej skupić swoją uwagę na *Hotplug* (więcej informacji na jego temat można znaleźć w *Linux+ z lipca 2005 r.*).

Powyższe nie oznacza jednak wcale, że nie istnieją urządzenia niekompatybilne z *Udev*. Co prawda, urządzeń tych jest coraz mniej i nawet trudno podać jakieś konkretne przykłady, ale wciąż istnieje pewne prawdopodobieństwo, np. przy próbach uruchomienia *winmodemów*, że mimo załadowania odpowiedniego modułu, plik urządzenia w */dev* nie pojawi się.

Problem tego typu można rozwiązać na kilka sposobów. Dla niektórych użytkowników najprostsze będzie wydanie komendy *mknod* z odpowiednimi parametrami (numerami *minor* i *major* brakującego urządzenia) oraz dopisanie jej do któregoś ze skryptów, uruchamianych przy starcie systemu. W części dystrybucji możemy zamiast tego dopisać odpowiednią linię do pliku */etc/udev/links.conf* - wystarczy otworzyć ten plik, aby po zawartych tam przykładach zorientować się co do formatu wpisów. Spowoduje to również tworzenie pliku urządzenia przy każdym starcie systemu. Ostatnim sposobem jest utworzenie pliku urządzenia (np. przy pomocy *mknod*), ale nie w katalogu */dev*, a w katalogu */etc/udev/devices* - zawartość tego ostatniego w kilku dystrybucjach jest automatycznie kopiowana do */dev* przy starcie systemu.

Tworzymy własne zasady

Wyobraźmy sobie sytuację, że mamy dwie myszki USB albo dwie drukarki USB, czy też jakieś urządzenia *usb-storage*, np. aparat fotograficzny i odtwarzacz MP3. Urządzenia te bez problemu działają w Linuksie, np. pierwsza drukarka jest dostępna na urządzeniu */dev/usb/lp0*, a druga na */dev/usb/lp1*. Podobnie w przypadku myszek mamy odpowiednie urządzenia, np. */dev/input/mouse0* oraz *mouse1* (albo *event0* oraz *event1*), a w przypadku urządzeń *usb-storage*, najczęściej będą występować nazwy */dev/sda* oraz */dev/sdb1*.

Takie nazwy urządzeń podajemy z reguły w konfiguracji serwera wydruku, środowiska graficznego czy wpisujemy do */etc/fstab*.

Każdy jednak, kto ma więcej niż jedno urządzenie tego samego typu, doskonale wie, że problemy rozpoczynają się w momencie, gdy np. uruchomimy komputer bez podłączonej pierwszej myszki, drukarki itp. Bardzo łatwo doprowadzić do sytuacji, w której urządzenia mają zamienione identyfikatory, czyli np. druga myszka stanie się pierwszą, albo nasz odtwarzacz MP3 zostanie zamontowany w miejscu przeznaczonym dla aparatu fotograficznego. W pew-

nych sytuacjach może nam to nie przeszkadzać, ale często bywa to poważny problem. Przykładowo, ja używam dwóch myszek USB, z których jedna posiada bardzo wiele przycisków. W sytuacji, gdy ta myszka "wskakuje" w miejsce standardowej, trzyprzyciskowej, tracę dostęp do dodatkowych przycisków i rolek. Stałe miejsce montowania aparatu cyfrowego również bywa bardzo wygodne, np. jeśli określimy go w programie *Digikam*. Dla posiadaczy drukarek USB porządek w nazwach jest już niezbędny, bo w przypadku urządzeń wymagających zupełnie innych sterowników, po ich zamianie wydruk nie będzie w ogóle możliwy.

Właśnie dzięki *Udev* dostajemy możliwość "sztywnego" określenia nazw urządzeń (ang. *persistent naming*). Dzięki temu nasz aparat fotograficzny, zamiast */dev/sda*, może nazywać się nawet */dev/fajny_aparaciki* i nazwa ta będzie mu już na stałe przypisana, niezależnie od tego, ile urządzeń tego samego rodzaju podłączymy do systemu. Własną nazwę urządzenia możemy podać w */etc/fstab*, albo gdziekolwiek indziej, gdzie jest potrzebna.

Skonfigurowanie własnych nazw nie jest trudne, gdyż wystarczy w tym celu kilka kroków. Pamiętajmy, że nie pomoże nam to w sytuacji, gdy naszego urządzenia brakuje, albo nie działa ono w sposób prawidłowy! Definiowanie własnych zasad *Udev* ma wpływ tylko na nazwy, pod którymi działające już urządzenia występują w systemie.

Miejsce na wpisy

Jeśli wcześniej złożyliśmy się przyjrzeć wpisom w domyślnej konfiguracji *Udev*, zapewne zwróciliśmy uwagę, że każdy z nich składa się jak gdyby z dwóch członów - po lewej stronie linii mamy opis jakiegoś pojedynczego urządzenia lub grupy urządzeń, a po prawej informację na temat jego nazwy oraz uprawnień dostępu. Jednymi i drugimi zajmujemy się za chwilę, gdyż teraz musimy znaleźć odpowiednie miejsce, w którym dopiszemy nasze definicje urządzeń - ich kolejność ma tutaj znaczenie.

Najprostszym rozwiązaniem będzie umieszczenie naszych definicji tak, aby były odczytane przez system na końcu, gdyż wtedy nie zostaną "nadpisane" przez jakieś inne, występujące już w systemie. Jeśli jednak zastosujemy podejście odwrotne i wpisujemy je na początek, najtrudniej będzie nam coś zepsuć. Wybór pozostawiam każdemu Czytelnikowi.

Tak czy inaczej, nasze pomysły proponuję dopisać do pliku `/etc/udev/rules.d/udev.rules` (w *Debianie* czy *Ubuntu*), albo do któregoś z plików w katalogu `/etc/udev/rules.d/` z jak najwyższym lub najniższym numerem początkowym - pliki te są analizowane w kolejności alfabetycznej.

Dla lepszej przejrzystości naszej konfiguracji dobrym rozwiązaniem może być również dopisanie naszych rozwiązań obok oryginalnych wpisów, odnoszących się do tych samych urządzeń, które chcemy zmodyfikować. Wybór jest w zasadzie dowolny, pamiętajmy tylko o kolejności, w jakiej definicje są analizowane.

Oczywiście, w wyjątkowych sytuacjach możemy znaleźć w systemie wpisy z opcją `OPTIONS+="last_rule"`, która oznacza, że *udev* więcej nie będzie zajmował się danym urządzeniem (lub urządzeniami). Możemy również sami stosować tę opcję.

Stare wersje *udeo* zajmowały się danym urządzeniem tak długo, aż nie znalazła się w końcu linia, zawierająca dyrektywę `NAME`. W obecnie stosowanych dystrybucjach to ograniczenie już nie występuje, więc nie musimy się nim przejmować.

Identyfikacja urządzenia

Teraz czeka nas najtrudniejszy moment, czyli decyzja, na podstawie czego rozróżnimy nasze urządzenia. Są one opisane w systemie przy pomocy wielu rozmaitych parametrów, takich jak magistrala (PCI, USB), identyfikator w tej magistrali, dla wielu urządzeń również identyfikatory producenta i produktu, a nawet jego numer seryjny. Generalnie, definicje te można podzielić na dwie grupy: ogólne: `BUS`, `KERNEL`, `SUBSYSTEM`, `DRIVER` (patrz ramka *Dobre rozpoznanie*), oraz oparte na plikach `/sys`. Brzmi może tajemniczo, ale gdy przeanalizujemy swoje pierwsze urządzenie, okaże się bardzo proste.

Aby obejrzeć dokładny opis jakiegoś urządzenia, wystarczy jedynie znać aktualne położenie pliku urządzenia. Przykładowo, dla myszki USB będzie to (mię-

dzy innymi) `/dev/input/mouseO`. Zakładając więc, że mamy działającą *udev* oraz działamy z konta *roota*, możemy wydać następujące polecenie: `udevinfo -a -p $(udevinfo -q path -n /dev/input/mouseO)`. Wypisuje ono całkiem sporo danych i to z dwóch podkatalogów w `/sys`. Nam zależy na takich danych, które umożliwią całkiem jednoznaczne rozróżnienie tej konkretnej myszki. Dla porównania zobaczymy więc jej "siostrę", czyli drugą myszkę USB w systemie (dla utrudnienia, wyprodukowaną przez tę samą firmę). Oczywiście, w tym celu wydajemy polecenie `udevinfo -a -p $(udevinfo -q path -n /dev/input/mouseI)` i Szukamy różnic w stosunku do poprzedniego wyniku. Wyniki będą bardzo różne, w zależności od wersji jądra, wersji *udev* czy dystrybucji. Na moim *Debianie Sarge* z jądrem 2.6.13.2 i *udev* w wersji 056 urządzenia te różniły się (oczywiście poza adresami w magistrali i nazwą w `/dev/input/`) jedynie atrybutem `SYSFS{modalias}`. W większości przypadków łatwo będzie jednak znaleźć wiele innych parametrów, dzięki którym możliwe będzie rozróżnienie urządzeń. Gdy porównałem swoją pamięć USB oraz odtwarzacz MP3 (dwa urządzenia *usb-storage*), zauważyłem, że co prawda tylko przy jednym z nich występował atrybut `SYSFS{model}`, ale do ich rozróżnienia świetnie nadawały się inne atrybuty: w przypadku pierwszego urządzenia miałem `SYSFS{uendor}="dc/sdcl"`, `SYSFS{size}="503378"`, a w drugim `SYSFS{uendorj}="USB 2.0"` i `SYSFS{size}="254961"`. Możemy użyć każdego z atrybutów, który oczywiście nie zmieni się po ponownym uruchomieniu systemu, czy przełączeniu urządzeń do innych gniazdek USB, oraz pozwala jednoznacznie odróżnić nasze "gadżety". Oczywiście, gdyby obydwie pamięci miały tę samą pojemność, atrybut `SYSFS{size}` byłby bezużyteczny.

Jak zapewne bardziej zaawansowani użytkownicy Linuksa zauważyli, w powyższych przykładach komenda `udevinfo` uruchamiana jest dwukrotnie. Najpierw przy

Udev ze źródeł?

Ze strony <http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html> możemy ściągnąć źródła najnowszej wersji *Udev*. Przydadzą się nam one jednak wyłącznie w sytuacji, gdy wersja obecna w naszej dystrybucji nie działa dobrze, np. zawieszają się lub zgłaszają błąd przy próbie odczytu atrybutów urządzenia. Program ten (przynajmniej w wersji 075, którą testowałem) kompiluje się bez żadnych problemów. Wystarczy rozpakować źródła (`tar jx fvp udev-075.tar.bz2`), przejść do utworzonego katalogu i wydać polecenie `make`. Przy pomocy `make install` instalujemy nową wersję.

pomocy zawartej w nawiasach komendy `udevinfo -q path -n /dev/urządzenie` ustalamy lokalizację potrzebnych nam informacji w katalogu `/sys`. Wynik działania tej komendy jest wysyłany jako argument dla `udevinfo -a -p [miejsce_ w_/sys/]`. Najnowsze wersje *udev*, np. ta zawarta w *Mandriva 2006*, mają nieco zmienione użycie komendy `udevinfo` i podany powyżej przykład z nawiasem nie zadziała. Zamiast tego musimy ręcznie znaleźć w katalogu `/sys` informacje o naszym urządzeniu. Robimy to np. poleceniem: `find /sys | grep mouseO`. Oczywiście, zamiast `mouseO` możemy szukać dowolnego innego urządzenia. W efekcie otrzymujemy:

```
/sys/class/input/mouseO
/sys/class/input/mouseO/device
/sys/class/input/mouseO/dev
```

Jako źródło potrzebnych nam informacji traktujemy pierwszą pozycję, czyli katalog o nazwie identycznej jak urządzenie. W kolejnym kroku wydajemy polecenie: `udevinfo -a -p /sys/class/input/mouseO`.

Mając jakieś charakterystyczne wyróżniki, przystępujemy do pisania opisu urządzenia. Najlepiej zawsze posługiwać się definicjami już istniejącymi w którychś pliku konfiguracyjnym. Ja, chcąc uporządkować sytuację z myszkami USB, wpisałem następujące dwie linie:

```
BUS=="usb", SYSFS(modalias)=="usb: <*-v046DpC501d0910dc00dsc00dp00ic03isc^01ip02", SYMLINK="input/szara-myszka"
BUS=="usb", SYSFS(modalias)=="usb: <*-v046DpC50Ed2500dc00dsc00dp00ic03isc->^01ip02", SYMLINK="input/czarna-myszka"
```

Urządzenie	Typ	Rozmiar	Punkt montowania	Wolne	Zajęte %	Wykres
/dev/sdd1	vfat	124,2 MB	/media/flash	122,5 MB	1,4%	[Progress bar]
/dev/czytnik/sm1	vfat	15,6 MB	/media/sd	15,6 MB	0,1%	[Progress bar]
/dev/md1	ext3	18,3 GB	/home	2,1 GB	88,8%	[Progress bar]
/dev/sdb3	ext3	189,9 MB	/boot	185,9 MB	12,4%	[Progress bar]
/dev/nda2	ext3	25,7 GB	/mnt/media	3,0 GB	88,4%	[Progress bar]
/dev/sda4	ext3	36,5 GB	/mnt/sd	23,0 GB	37,0%	[Progress bar]
/dev/md0	ext3	36,7 GB	/	7,0 GB	80,9%	[Progress bar]
/dev/sdb9	jfs	37,0 GB	/mnt/dane	33,4 GB	9,8%	[Progress bar]
/dev/sdb7	ext2	8,3 GB	/mnt/mandrake	5,9 GB	29,4%	[Progress bar]
/dev/sdb5	ext3	9,0 GB	/mnt/fedora	5,5 GB	38,8%	[Progress bar]
/dev/sdb6	ext3	9,2 GB	/mnt/ubuntu	7,8 GB	14,9%	[Progress bar]
/dev/sdb8	ext3	9,2 GB	/mnt/test	8,7 GB	5,4%	[Progress bar]

Rysunek 2. Własne nazwy urządzeń mogą pomóc w orientacji w systemie

Z braku innych atrybutów, którymi można było je rozróżnić, użyłem *SYSFS(modalias)*. Dodatkowo, dopisałem *BUS=="usb"*, aby system nie musiał sprawdzać absolutnie każdego urządzenia w systemie pod kątem tego atrybutu. Zwróćmy uwagę na podwójne znaki "=" - w najnowszych wersjach *Udev* są one niezbędne. Użyty dalej parametr *SYMLINK* nakazuje tworzenie wyłącznie dowiązania symbolicznego, ale o nim później. W efekcie, po zrestartowaniu *Udev*, w katalogu */dev/input* pojawiły się dwa nowe obiekty:

```
czarna-myszka -> ts0
szara-myszka -> ts1
```

Oczywiście, absolutnie nie o to mi chodziło, gdyż chciałem uzyskać dowiązanie symboliczne do urządzeń *mouseO* i *mouseI*. Myszkę USB to dosyć nietypowe urządzenie, gdyż pojawiają się jako */dev/input/event#*, */dev/input/mouse#* oraz dodatkowo */dev/input/ts#* (urządzenie *generic touchscreen*, u mnie niepotrzebne). Musiałem zmodyfikować mój wpis, aby odnosił się wyłącznie do urządzeń typu *mouse*:

```
BUS=="usb", KERNEL=="mouse?", <-
SYSFS{modalias}=="usb:v046DpC501d09*-
10dc00dsc00dp00ic03isc01ip02", +-'
SYMLINK="input/szara-myszka"
BUS=="usb", KERNEL=="mouse[0-9]", *-
SYSFS{modalias}=="usb:v04 6DpC50Ed25-<-
00dc00dsc00dp00ic03isc01ip02", *-
SYMLINK="input/czarna-myszka"
```

Dołożyłem jedynie atrybut *KERNEL*, przy pomocy którego określiłem, że chodzi mi o urządzenie, którym jądro normalnie nadaje nazwę *mouse#*. Przy okazji widać, że możemy używać wyrażeń regularnych - znak zapytania zastępuje dowolny pojedynczy znak, a cyfry w nawiasach kwadratowych - przedział liczbowy. Możemy również używać gwiazdki, podobnie jak np. w *BASH*.

Na identycznej zasadzie da się wyróżnić wszelkie inne urządzenia. Wystarczy spokojnie przeanalizować wynik działania polecenia *udevinfo*.

Aby jednak wszystko nie było "za łatwe", *Udev* ma jedno bardzo istotne ograniczenie co do możliwości identyfikacji urządzeń. Mianowicie, nie można wstawiać do jednej linii z opisem urządzenia danych z dwóch różnych katalogów w */sys*. Spójrzmy jeszcze raz na wynik polecenia: *udevinfo -a -p \$(udevinfo -q path -n /dev/input/mouseO)* - od razu widzimy

dwa oddzielone od siebie katalogi (nazwa pierwszego zaczyna się od */sys/class/input/*, a drugiego */sys/devices/pciO...*). Jeśli w naszej konfiguracji chcemy użyć atrybutu *SUBSYSTEM* z pierwszego z tych katalogów, nie możemy już użyć niczego z drugiego katalogu, np. *BUS*.

Z moich testów wynika, że *Udev* jest wciąż daleki od doskonałości. Dla blisko 10 różnych urządzeń USB, które testowałem w niemal każdej dystrybucji *udevinfo* pokazuje inne dane. Oczywiście, częściowo może to być wina poszczególnych jąder, czy łątek, które stosują autorzy dystrybucji. Niemniej, trzeba być przygotowanym na sytuację, w której *udevinfo* pokaże bardzo mało danych o jakimś urządzeniu np. tylko kilka linii z jednego katalogu. Może też wyświetlić niepotrzebne nam informacje o innych urządzeniach, np. opis kontrolera USB przy zapytaniu *ofksh disk*. Trzeba więc uważnie i ze zrozumieniem czytać opisy sprzętu, aby sukcesywnie identyfikować poszczególne urządzenia.

Polecenie dla udev

Udev stworzono po to, aby urządzenia w */dev* mogły mieć całkiem dowolne nazwy. Dzięki niemu za metodologię nazewnictwa urządzeń odpowiadają twórcy poszczególnych dystrybucji (albo końcowi użytkownicy), a nie programiści jądra systemu.

Jeśli przy pomocy opisanych wcześniej reguł jesteśmy w stanie zidentyfikować jakieś urządzenie, możemy dalej - w tej samej linii - użyć słowa kluczowego *NAME*, aby nadać urządzeniu własną nazwę:

```
KERNEL=="null", NAME="czarna-<-
dziura", MODE="0666"
```

Przy pomocy *KERNEL=="null"* jednoznacznie identyfikujemy urządzenie, które posiada tradycyjną (zapisaną wciąż w jądrze) nazwę *nuli* (oczywiście, znamy je jako */dev/null*). Takim wpisem jak powyższy nadajemy temu urządzeniu swoją własną nazwę */dev/czarna-dziura*. To oczywiście zadziała, ale trzeba sobie jeszcze zadać pytanie, po co nam taka modyfikacja? Na pewno wymagałaby ona sporego nakładu pracy przy modyfikacji wszystkich skryptów w systemie, które odwołują się do tradycyjnej nazwy */dev/null*. Mogłaby również spowodować spore problemy z kompatybilnością naszego systemu, np. przy instalacji nowych programów.

Z podobnymi problemami mielibyśmy do czynienia przy zmianie domyślnej nazwy niemal każdego z często wykorzystywanych w systemie urządzeń. Jak więc pogodzić chęć posiadania własnej nazwy z zachowaniem kompatybilności systemu? Oczywiście, najlepiej używać dowiązań symbolicznych. Zachowamy tradycyjną nazwę np. */dev/null*, a będziemy mogli używać również własnej, np. */dev/czarna-dziura*. Możemy to zrobić przy pomocy wpisu:

```
KERNEL=="null", NAME="k", <-
SYMLINK=<-
"czarna-dziura", MODE="0666"
```

Dobre rozpoznanie

Gdy analizujemy oryginalną konfigurację *Udev*, zawartą w naszej dystrybucji, zauważymy zapewne, że pojedyncze wpisy dotyczą tam na ogół większej grupy urządzeń (np. wszystkie dyski wymienne) albo urządzeń logicznych (np. */dev/null*). Jednak przy pisaniu naszych własnych zasad nazewnictwa urządzeń będzie nam najczęściej zależało na jak najbardziej indywidualnym podejściu do każdego urządzenia. Musimy więc - analizując dane podawane przez *udevinfo* - szukać tych atrybutów, przy pomocy których można z całkowitą pewnością jednoznacznie zidentyfikować urządzenie.

Z urządzeniami USB na ogół (choć nie zawsze) jest najłatwiej. Wystarczy użyć *BUS=="usb"* oraz właściwych *SYSFS{vendor}* i *SYSFS{model}*. Jeśli są one dostępne, to problem załatwiony. Niektóre urządzenia (np. drukarki firmy *Epson*) zgłaszają nawet numer seryjny - *SYSFS{serial}* - co jeszcze bardziej upraszcza nam pisanie regułek. Czasem nie będzie tak łatwo i trzeba będzie posłużyć się innymi atrybutami, jak np. *SYSFS{modalias}*, czy *SYSFS{size}*.

Pozostałe urządzenia w systemie często mają dużo więcej opisów niż można by się spodziewać. Przykładowo, twarde dyski posiadają *SYSFS{model}*, a jeśli chcemy coś kombinować z urządzeniami PCI, mamy również *vendor* i *model*.

Zawsze, gdy to możliwe, najlepiej jest korzystać z najprostszych słów kluczowych, czyli: *BUS* - np. *ide*, *scsi*, *usb*, *pci*, oraz *KERNEL*, czyli oryginalna nazwa urządzenia, zapisana w jądrze. Pozostałe z podstawowych atrybutów (*SUBSYSTEM*, *DRIVER*, *ID*) są już używane bardzo rzadko.

Jak widzimy, jako parametr dla NAME pojawiło się `%k`. Taki sam parametr jest bardzo często stosowany w oryginalnych plikach konfiguracyjnych `udev`, we wszystkich dystrybucjach. Oznacza on, że jako nazwa urządzenia użyta zostanie oryginalna nazwa, zapisana w jądrze. Zamiast samego `%k` czasami używa się jakiegoś dodatkowego katalogu. Przykładowo, `input/%k` spowoduje, że dla danego urządzenia użyta będzie oryginalna nazwa, ale plik urządzenia będzie umieszczony w podkatalogu `input`. Tak jest np. w przypadku `/dev/input/mouse0`. Poza `%k` w plikach konfiguracyjnych można stosować jeszcze kilka innych zmiennych. Opis ich wszystkich można znaleźć w man `udev`. W praktyce stosuje się wyłącznie dwie - właśnie `%k` oraz `%n`. Ten drugi oznacza numer urządzenia, nadawany przez jądro. Przykładowo, wpis:

```
KERNEL=="mouse[0-9] ", SYMLINK="myszki/%n"
```

powinien powodować, że myszki w systemie będą dostępne pod `/dev/myszki/0`, `/dev/myszki/1` itd.

W praktyce SYMLINK jest najlepszą opcją, aby dostosować system do własnych preferencji, bez obawy o zepsucie czegoś. Możemy nawet stosować wiele dowiązań symbolicznych do jednego urządzenia, np. przy pomocy:

```
KERNEL=="hdb", SYMLINK="cdwriter dvd cdrom nagrywarka"
```

uzyskamy aż cztery dowiązania symboliczne, które są albo łatwiejsze do zapamiętania przy wklepywaniu z linii komend, ale też są domyślnymi nazwami urządzeń dla takich programów, jak np. `mplayer` czy `vcdimager`.

Wreszcie przykład z urządzeniami `usb-storage`, które posiadamy:

```
BUS=="scsi", KERNEL=="sd?l", SYSFS{vendor}=="USB 2 . 0", *-•  
    SYMLINK="wolnyflash"  
BUS=="scsi", KERNEL=="sd?l", SYSFS{vendor}==" dc/dcl", «•  
    SYMLINK="wolnyp3player"  
BUS=="scsi", KERNEL=="sd?l", SYSFS{vendor}=="FUJIFILM", •*-•  
    SYMLINK="wolnycamera"
```

Jak widzimy w tym przypadku, właściwą magistralą jest `scsi`, a do rozróżnienia moich urządzeń wystarczył identyfikator producenta (`SYSFS{vendor}`), gdyż każde z urządzeń go posiadało (nawet jeśli zawiera on mało informacyjne określenie, jak "USB 2.0 "). Nazwy dowiązań symbolicznych zostały użyte u mnie również w `/etc/fstab`. Ponieważ nie korzystam z automatycznego montowania, przykładowy wpis w `/etc/fstab` wygląda u mnie następująco:

```
/dev/wolnyflash /niedia/flash* - ^  
    vfat,noauto,users,icharset= •*— •  
    iso8859-2,codepage=852, *—'  
    umask=0 0 0
```

Oczywiście, wciąż nikt nie zabrania nam używania dowolnych nazw urządzeń na naszym własnym komputerze. Zanim zdecydujemy się zmieniać domyślne nazewnictwo, przeszukajmy wszystkie pliki konfiguracyjne w poszukiwaniu opisów interesujących nas urządzeń. Dopiero wtedy, gdy doskonale rozumiemy

zachowanie systemu, próbujemy własnych rozwiązań. Tym bardziej, że współczesne dystrybucje posiadają dodatkowe skrypty, które np. skanują każde podłączone urządzenia blokowe, aby otrzymać listę partycji, albo umieszczają ikonkę na pulpicie. Różne tego typu udogodnienia możemy stracić, jeśli samodzielnie zaczniemy modyfikować konfigurację bez jej dogłębnego poznania.

W przypadku problemów z naszymi wpisami, możemy sprawdzić, czy w `/etc/udev/udev.conf` znajduje się wpis `udev_log="yes"`. W takim przypadku informacje na temat tworzonych i usuwanych urządzeń powinny być za każdym razem zapisywane do logów systemowych. Ponadto, istnieje sposób na przetestowanie naszej konfiguracji, np. przy pomocy `udevtest /sys/block/sdc` dostaniemy informacje na temat działań, które będą podjęte przez *Udev* odnośnie podanego urządzenia - w tym przypadku dysku `/dev/sdc`.

Nietypowe urządzenia

Urządzeniami, które wymagają nieco nietypowego traktowania, są popularne, uniwersalne czytniki kart pamięci, podłączone przez USB. Po podłączeniu takiego urządzenia w Linuksie natychmiast tworzone są np. cztery nowe urządzenia *scsi* (najczęściej `sda`, `sdb`, `sdc` i `sdd` lub kolejne - każde do obsługi innego rodzaju kart). Dzieje się tak, pomimo faktu, że wszystkie cztery złącza kart pamięci w urządzeniu są puste. Nawet jeśli czytnik potrafi obsługiwać tylko jedną kartę równocześnie, zawsze tworzone są np. cztery urządzenia. To jednak tylko część problemu. O ile bowiem system jest dobrze poinformowany o podłączeniu czytnika i reaguje na ten fakt (tworząc `sda`, `sdb` itd), to już samo włożenie karty pamięci nie wywołuje żadnego zdarzenia systemowego. Co to oznacza w praktyce? Otóż po podłączeniu czytnika kart i włożeniu karty mamy co prawda urządzenie np. `sdb`, ale brakuje nam `sdbl`, czyli partycji, którą można by zamontować w systemie.

Co prawda, wystarczy polecenie np. `fdisk -i /dev/sdb`, aby system "dowiedział się" o włożonej karcie w drugim złączu i utworzył potrzebny plik urządzenia `sdbl`, ale nie jest to ani elegancka, ani wygodna metoda.

Istnieje jednak specjalna opcja w *udev*, która pozwala na utworzenie plików urządzeń dla wszystkich teoretycznie możliwych partycji na danym urządzeniu. Nie powoduje ona wcale, że tablica partycji jest skanowana - po prostu zawsze mamy

15 *device nodes* dla możliwych w tym miejscu partycji, choć z reguły wystarczy nam tylko jedna.

Zacznijmy od początku. Przy pomocy `udevinfo` o sprawdzamy wszystkie urządzenia, które powstają po podłączeniu czytnika kart. Po znalezieniu odpowiednich atrybutów, przy pomocy których można je rozróżnić, tworzymy wpisy w konfiguracji *udeo*. Moje wyglądają następująco:

```
BUS=="scsi", SYSFS{vendor}==←
"Generic", SYSFS{model}==←
"USB SD Reader", NAME{all_←
partitions}="czytnik/sd"
BUS=="scsi", SYSFS{vendor}==←
"Generic ", SYSFS{model}==←
"USB CF Reader", NAME{all_←
partitions}="czytnik/cf"
BUS=="scsi", SYSFS{vendor}==←
"Generic", SYSFS{model}==←
"USB SM Reader", NAME{all_←
partitions}="czytnik/sm"
BUS=="scsi", SYSFS{vendor}==←
"Generic", SYSFS{model}==←
"USB MS Reader", NAME{all_←
partitions}="czytnik/ms"
```

Jak widzimy, "magiczną" funkcję pełni opq'a `{all_partitions}`, dodana zaraz po NAME. W katalogu `/dev/czytnik/` powinno pojawić się po 16 pozycji dla każdego typu karty (np. `sda` oraz `sda1` do `sda15`).

Co ciekawe, *udev* używać można również do zmiany nazwy interfejsów sieciowych (*eth0*, *eth1*, itp.). Nie zawsze wydaje mi się to dobrym pomysłem, gdyż do tych nazw odnosimy się w wielu plikach, opisujących konfigurację sieci (np. w *Debianie Sarge* mamy `/etc/network/interfaces`, który wymaga poprawienia po takiej operacji), ale nie zaszkodzi spróbować. Wystarczy w tym celu wydać komendę, np. `udevinfo -a -p /sys/class/net/eth0/`, aby sprawdzić wszystkie atrybuty naszego interfejsu sieciowego, podobnie jak robiliśmy to z innymi urządzeniami. W tworzonym wpisie do konfiguracji *udev* najlepiej jest wykorzystać niepowtarzalny adres MAC, zawarty w `SYSFS{address}`. Przykładowo, możemy dopisać taką linię:

```
KERNEL="eth*", SYSFS{address}==←
"ff:a3:sf:da:d6:68", NAME="eth5"
```

Jak łatwo się domyśleć, dzięki temu karta sieciowa o tym numerze MAC będzie miała zawsze przypisany interfejs *eth5*

(zamiast *eth#* można w zasadzie używać dowolnych nazw).

Przy zmianie nazwy interfejsów sieciowych konieczne jest nie tylko zrestartowanie *udev*, ale również wyłączenie sieci oraz usunięcie z jądra i ponowne załadowanie modułu, odpowiedzialnego za kartę sieciową. Gdy nie bardzo wiemy, jak to zrobić, najprościej będzie przeładować cały system. Pamiętajmy jednak, że jeśli w naszej dystrybucji "ręcznie" konfigurowaliśmy interfejsy sieciowe, podając ich nazwy, najprawdopodobniej niezbędne będzie poprawienie tych wpisów.

Istnieje cały szereg urządzeń, zwłaszcza USB, dla których - w momencie pisania tego artykułu - nie tworzy się żadnych wpisów w konfiguracji *udev*, bowiem nie używają one w ogóle plików urządzeń w `/dev`. Najpopularniejszymi przedstawicielami takiego rodzaju urządzeń są aparaty cyfrowe, obsługiwane przez oprogramowanie *GPhoto* (a więc te niekompatybilne z *usb-storage*), oraz skanery USB, obsługiwane przez SANE. Jedne i drugie do działania wymagają *libusb*, a konfigurację uprawnień dostępu realizuje się wyłącznie za pomocą *hotplug*. W przyszłości prawdopodobnie również te urządzenia będą bardziej konfigurowalne, także przy użyciu *Udev*.

Zakończenie

Twórcy nowoczesnych dystrybucji dokładają sporych starań, aby wiedza na temat nazw urządzeń czy sposobów ich montowania nie była użytkownikowi potrzebna. Ikonka pojawiająca się na biurku - zdaniem niektórych - jest rozwiązaniem dobrym dla wszystkich i zawsze. Osobiście nie zgadzam się z takim podejściem, nie tylko dlatego, że rzadko widuję swoje puste biurko i jak dla mnie równie dobrze ikonka mogłaby się pojawiać na wygaszaczu ekranu. Po prostu lubię samodzielnie decydować, co i gdzie umieszczone jest w moim systemie i w jaki sposób działa. Do podobnego podejścia namawiam również innych. ☺

W Internecie:

- Strona domowa programu *Udev*: <http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html>
- Opis tworzenia konfiguracji *Udev*: http://www.reactivated.net/writing_udev_rules.html