Linux on zSeries

Exploiting udev in SLES9 and RHEL4

Christian Borntraeger
IBM Boeblingen

Agenda

- **introduction**
- **device access in Linux**
- **device attach – and then?**
  - hotplug in Linux
- **how udev works**
- **configuring udev**

How devices are accessed in Linux

- **Linux adopts UNIX philosophy**
  - (almost) everything is a file
  - several file types: directory, link, device node, pipe....
  - device are accessed via device nodes
- **device nodes behave like normal file**
  - reside on a file system
  - file operations like open, read, write, seek are possible
  - if you write to the device node the kernel writes to the device
  - same with reading
  - e.g. you could do an offline backup using the device node:

```
# dd if=/dev/dasdf of=/home/backup/dasdf.img
```

How devices are accessed in Linux

- **special properties**
  - two numbers: major and minor number
  - device type: block device or character device

```
#ls -l /dev/dasda
brw-------  1 root root  94, 0   2005-01-05 17:50 /dev/dasda
```

- **the kernel cares only about the type and numbers and ignores the name of the device node**
- **most applications only care about the name of the device node**
- **this relationship can be freely configured by the administrator**

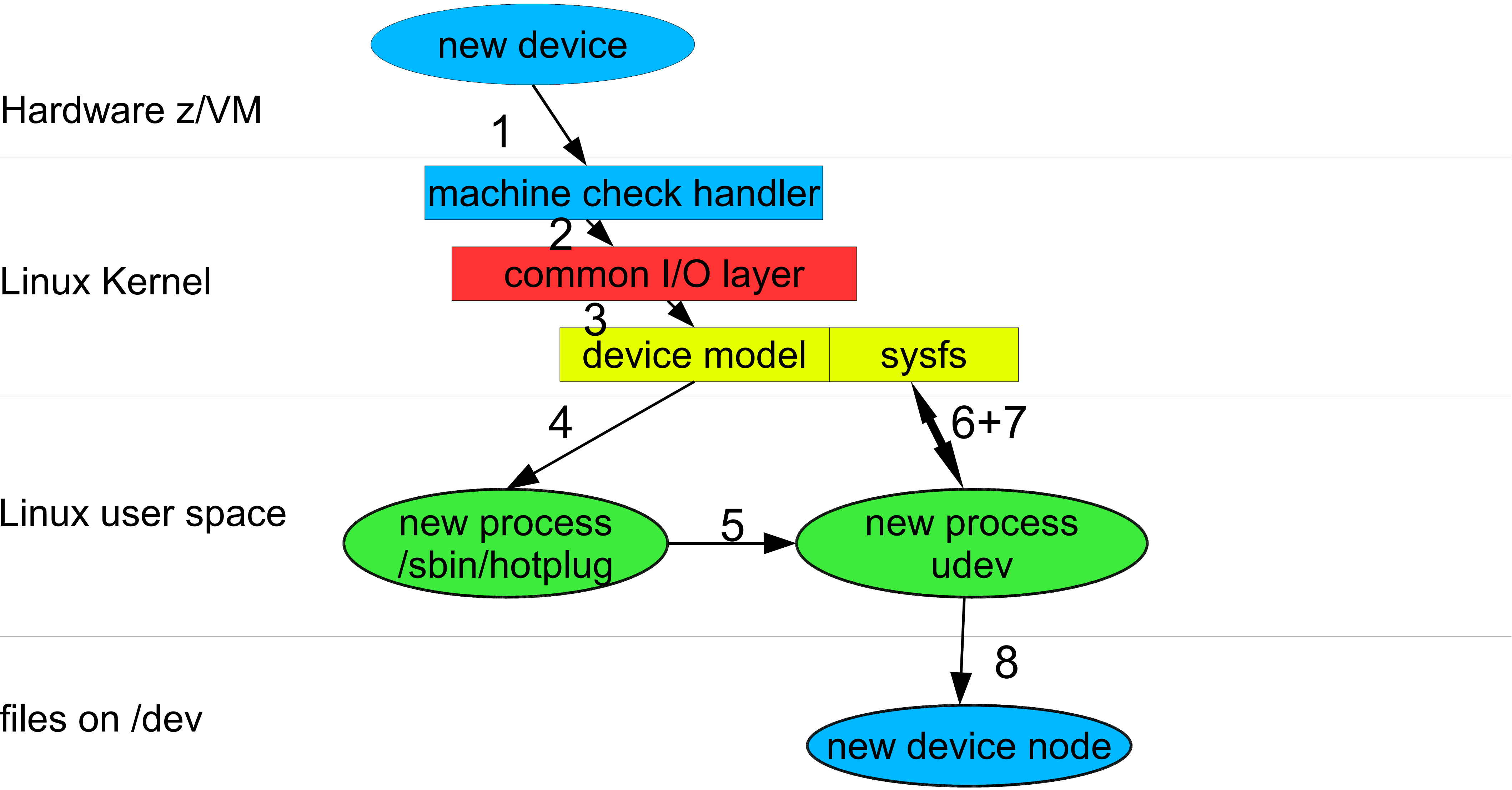
Creation of device nodes

- **manual invocation of mknod**
  - /proc/devices gives you the major number of every device type
- **done by the distributor or the installation program**
  - all distributions prepare a set of device nodes on their filesystem images
- **devfs (deprecated by kernel community)**
- **udev**
  - newest approach for automated device node creation
  - based on sysfs and hotplug

Introduction into hotplug – attaching new devices (1)

- **What happens if a new device arrives?**
  - the hardware or z/VM are creating a machine check
  - the Linux machine check handler handles the machine check
  - the Linux common I/O layer queries the channel subsystem
  - the new devices is registered in the Linux device infrastructure
    - a new sysfs entry appears
    - a hotplug event is created

Introduction into hotplug – attaching new devices (2)



How does udev works -general

- **the kernel calls /sbin/hotplug with parameters**
- **/sbin/hotplug multiplexes events and calls udev**
- **parameters are saved in environment variables**
- **/sys/block/dasdb/dev contains major and minor number**
- **/etc/udev/udev.conf specifies**
  - rules file: howto name
  - permissions file: access rights

```
DEVPATH=/block/dasdb
PATH=/sbin:/bin:/usr/sbin:/usr/bin
ACTION=add
PWD=/
SHLVL=1
HOME=/
SEQNUM=201
```

- **udev examines the configuration files and creates the device node appropriately**

how does udev works – newer version

- **newer version of udev also execute /etc/dev.d/**
  - scripts specific for different devices
  - additonal variable DEVNAME giving the name of the device node
  - scripts with .dev extension in lexical order in these directories
    - /etc/dev.d/$(DEVNAME)/*.dev,
    - /etc/dev.d/$(SUBSYSTEM)/*.dev
    - /etc/dev.d/default/*.dev
- **udev can also rename network interfaces (in theory)**
  - not supported by SLES or RHEL
  - network scripts have to be adopted
- **newer desktop distributions use udevsend as hotplug multiplexer**

Introduction into hotplug – attaching new devices (3)

- **lets have a look at SUSE**
  - config options for kernel parameter line
    - NOHOTPLUG=udev-only
    - NOHOTPLUG=[ccw|scsi|any other subsystem..]
  - /etc/hotplug/ is a directory containing several agents
    - /sbin/hotplug <system> <parm> -> /etc/hotplug/<system>.agent parm is called
      - ccw.agent, scsi.agent, tty.agent and so on
  - udev is called as /etc/hotplug/generic_udev.agent for all devices
  - Debugging: create a folder /events: everything is logged into this folder

Introduction into hotplug – attaching new devices (4)

- **what about RHEL4?**
  - /etc/hotplug.d/default/ contains 4 scripts
    - 05-wait_for_sysfs.hotplug
    - 10-udev.hotplug
    - 20-hal.hotplug
    - default.hotplug
  - there are additional tool called udevd and udevsend
    - udevsend submits the taks to udevd
    - udevd queues all events according to the sequence number and calls udev for each event

Current Status

- **udev is part of SuSE SLES9, RedHat RHEL4 and Debian**
- **different versions in different distributions**
- **only a minimal configuration**
- **coldplugged devices (already present during boot) are not handled by udev on SLES9**
  - static device nodes are used instead
  - manual invocation: udevstart or /etc/init.d/boot.udev start
- **device nodes are named after kernel names**
  - **DEVPATH=/block/dasdb**
- **big infrastructure for a small bonus**

So, what is also possible?

- **define your own access rights for dynamically attached devices**
- **get persistent names for your devices**
- **create symbolic links to have several names for a device**
- **use volume ID, device number or other characteristic hardware information to name your device**
- **Lots of other ideas...**

The udev config files

- **central config file is `/etc/udev/udev.conf`**
  - define general options
    - `udev_root` where should udev create device nodes, e.g. *“/dev/”*
    - `dev_db` where to create udevs data base , e.g. *"/dev/.udev.tdb"*
    - `default_mode` standard permissions of files, e.g. *“0600"*
    - `default_owner` standard user id of files , e.g. *"root"*
    - `default_group` standard group id of files, e.g. *"root"*
    - `udev_log` if set to *“yes”*, udev will log its activity into syslog
  - define the location of other config files
    - `udev_rules` rules for udev, e.g. *“/etc/udev/udev.rules"*
    - `udev_permissions` permissions for udev, e.g *“etc/udev/udev.permissions"*

udev.permissions

- **sets the permissions of device nodes**
  - override udev.conf for matching device nodes
  - permissions in UNIX style

USER			GROUP			OTHERS		
R	W	E	R	W	E	R	W	E

- UserGroupOthers X Read(4) Write(2) Execute (1)
- octal coding: just add Read, Write and Execute for each user spec
- example: dasd/0190/*:root:users:640
  - read and write access for *root*, read access for all users in group *users*

- **Do not play. Think about your rules. You are dealing with security**
  - e.g. if the disk is mounted during boot, only root needs access

udev rules

- **one rule per line: `key[,key...][,NAME] [,SYMLINK]`**
- **`key=`**

– `BUS`	every device on this bus
– `KERNEL`	every device matching this kernel name
– `PROGRAM`	execute this program, pass parameters
– `RESULT`	query the return value of the program
– `ID`	match the id of the device within the bus
– `SYSFS{x}`	match the content of the sysfs file

- **you can specify a name, a symlink or both**
- **of no name is specified, the kernel name is used**
- **“NAME=” makes this rule the last one**

udev rules

- **some parameters for NAME, SYMLINK and PROGRAM**

– %n	the "kernel number", e.g. “dasda1” has “1”
– %k	the "kernel name" for the device, e.g. dasda
– %p	the devpath for the device. (not in SLES9)
– %M	the kernel major number for the device
– %m	the kernel minor number for the device
– %b	the bus id for the device
– %c ,%c{N}	the string/substring returned by the external program
– %s{filename}	the content of a sysfs attribute
– %%	the % character itself

udev rules

- **some usage examples:**
  - BUS=”scsi”, SYMLINK=”scsi/%k”
  - KERNEL="dcssblk*", SYMLINK="dcssblk/%b"
  - BUS=”ccw”, PROGRAM=”/sbin/magictool”, SYMLINK=”%c”
  - BUS=”ccw”, PROGRAM=”/sbin/vendor-abc --check”, RESULT=”supported”, SYMLINK=”abc%n”
  - ID="0.0.0191", KERNEL=”dasd*[a-z]”, SYMLINK="cmshome"
  - BUS=”scsi”, SYSFS{model}=”2105*”, SYMLINK=”ESS800-%k”

Setting Up SuSE SLES9 – activate on boot

- **distinction between coldplug/hotplug**
- **coldplug brings up statically set up devices (your mindisk or network adapter)**
- **hotplug is for devices which appear while running**
- **udev is only used for hotplugged devices.....**
- **......but it can work for available devices as well**

```
# /etc/init.d/boot.udev start
creating device nodes
```

- **to do this every boot:**

```
# chkconfig boot.udev on
```

Setting Up SuSE SLES9 – special rules

- **SuSE has several early rules**

```
BUS="scsi", PROGRAM="/sbin/udev.get_persistent_device_name.sh", NAME="%k" SYMLINK="%c{1+}"
BUS="usb", PROGRAM="/sbin/udev.get_persistent_device_name.sh", NAME="%k" SYMLINK="%c{1+}"
BUS="ide", PROGRAM="/sbin/udev.get_persistent_device_name.sh", NAME="%k" SYMLINK="%c{1+}"
BUS="ccw", PROGRAM="/sbin/udev.get_persistent_device_name.sh", NAME="%k" SYMLINK="%c{1+}"
```

- **after a matching rule with NAME= udev stops**
- **to apply your rules**
  – put your rules at the beginning
  – do not use NAME, only use SYMLINK
- **save your rule file and watch for package update**

Setting up RedHat RHEL4

- **redhat ships with udev version 0.50**
- **rules are applied to coldplugged devices as well**
  – during boot the script /sbin/start_udev is called
- **you can put your rules in an separate file to avoid trouble during updates, e.g. `/etc/udev/rules.d/51-my.rules`**

Ideas for dasd devices

- **some persistent device names already exist:**
  - SUSE SLES9 has already rules for persistent device names
    - Volume ID and device number
    - /dev/disk/by-id/<VOLUME_ID>
    - /dev/disk/by-path/ccw-<BUS_ID>
    - activated by boot.udev script
  - Redhat RHEL4
    - device number
    - /dev/dasd/<BUS_ID>/disc and /dev/dasd/<BUS_ID>/part[1-3]
      - e.g /dev/dasd/0.0.0150/disc
-

Ideas for dasd

- to use the volume id in redhat you need a program

```
#!/bin/bash
MINOR=$2
let PARENT=MINOR-MINOR%4
TEMPDIR=`mktemp -d /tmp/dasd.XXXXXX`
if [ $? != 0 ] ; then
exit 1
fi
mknod $TEMPDIR/dasd-$1-$PARENT b $1 $PARENT
RETURN=`dasdview -j -f $TEMPDIR/dasd-$1-$PARENT`
rm -f $TEMPDIR/dasd-$1-$PARENT
rmdir $TEMPDIR
echo $RETURN
```

- and rules to use it

```
KERNEL="dasd*[a-z]", PROGRAM="/sbin/getdasd.sh %M %m", SYMLINK="dasd/%c/disc"
KERNEL="dasd*[0-9]", PROGRAM="/sbin/getdasd.sh %M %m", SYMLINK="dasd/%c/part%n"
```

Ideas for dcss block devices

- **segment name**
  - device nodes named after the DCSS
  - KERNEL="dcssblk*", SYMLINK="dcssblk/%b"

other ideas

- encode the WWPN or volume ID of an FCP/SCSI device
- create link to 3270 or 3215 console device nodes depending on the used console
- tell me!

Outlook

- **udev got additional modifiers**

– %N	the name of a created temporary device node
– %P	The node name of the parent device.
– %e	adds a number if the device node already exists

- **better integration in newer distributions (SLES10, RHEL5)**
- **better integration in a hardware abstraction layer**

Question and Discussion

- **Now**
- **After this session**
- **Any time during WAVV**
- **Email:**
  - cborntra@de.ibm.com
- **Thank you for your attention**