Linux Hotplugging

http://linux-hotplug.sourceforge.net/

Greg Kroah-Hartman

greg@kroah.com http://www.kroah.com/linux/

Kernel modules now "just work"

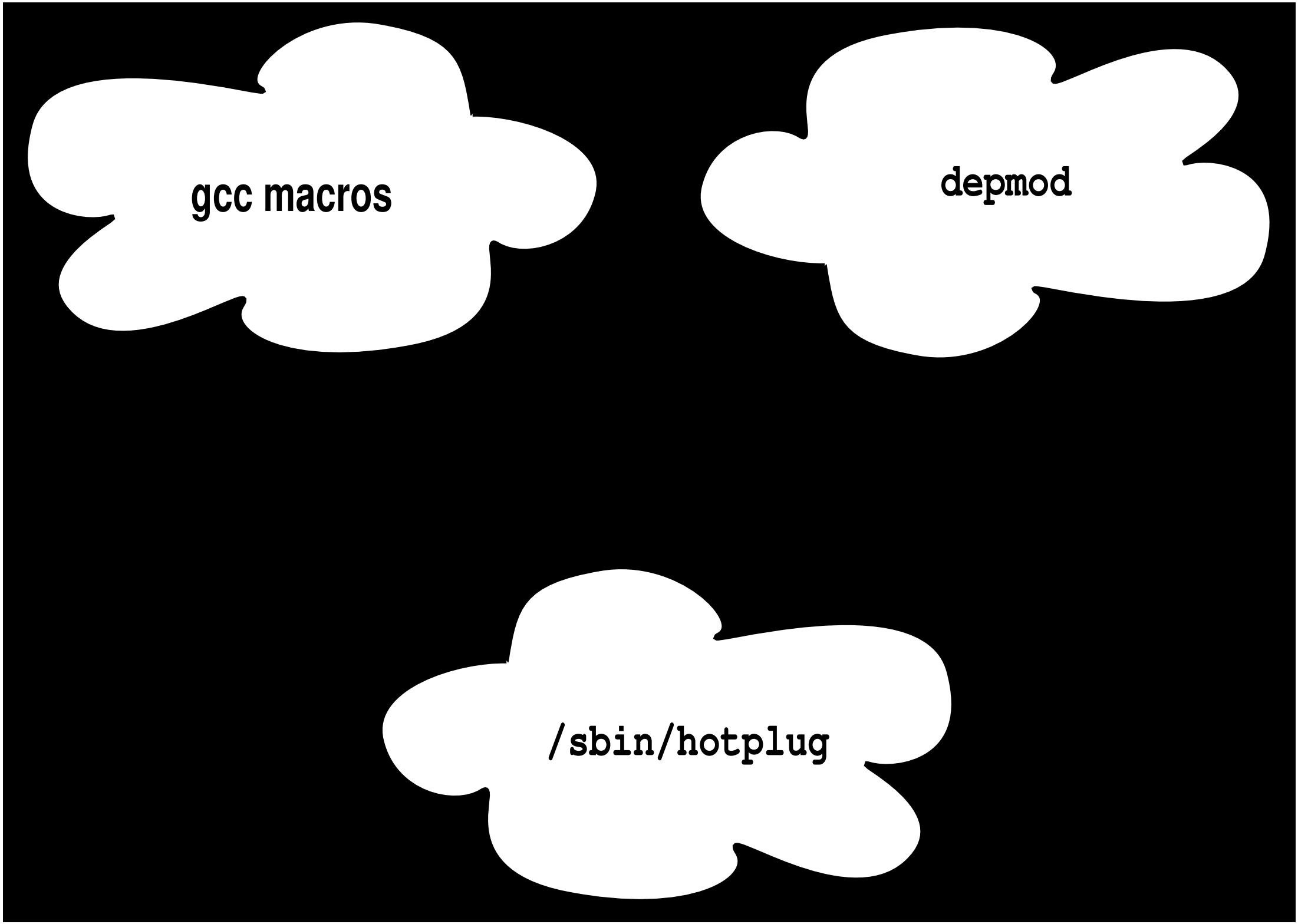
- No more manual configuration
- Automatic module loading on demand

Been done before

- /etc/pcmcia/config
- modules.conf
- Both maintained by hand

Let the computer do it itself

- Devices tell the computer what they are
  - PCI ids
  - USB ids
  - PCMCIA ids
- Drivers know what hardware they support
- Kernel knows what is going on


gcc macros
depmod
/sbin/hotplug

Let the kernel do it

/sbin/hotplug

- CONFIG_HOTPLUG
- /proc/sys/kernel/hotplug
- PCI and USB add or remove
- Patches exist for CPU and Docking stations
- Network register or unregister

PCI event environment

argv[0] = hotplug_path
argv[1] = pci
argv[2] = 0

HOME = /
PATH = /sbin:/bin:/usr/sbin:/usr/bin
PCI_CLASS = class_code
PCI_ID = vendor:device
PCI_SUBSYS_ID = subsystem_vendor:subsystem_device
PCI_SLOT_NAME = slot_name
ACTION = action

action=(add|remove)

red is actual values
green is replaced with runtime information

USB event environment

argv[0] = hotplug_path
argv[1] = usb
argv[2] = 0

HOME = /
PATH = /sbin:/bin:/usr/sbin:/usr/bin
PRODUCT = idVendor/idProduct/bcdDevice
TYPE = device_class/device_subclass/device_protocol
ACTION = action

if device_class == 0 then
INTERFACE = class/subclass/protocol

action=(add|remove)

red is actual values
green is replaced with runtime information

Network event environment

argv[0] = hotplug_path
argv[1] = net
argv[2] = 0

HOME = /
PATH = /sbin:/bin:/usr/sbin:/usr/bin
INTERFACE = interface
ACTION = action

action=(register|unregister)

red is actual values
green is replaced with runtime information

CPU event environment

argv[0] = hotplug_path
argv[1] = cpu
argv[2] = 0

HOME = /
PATH = /sbin:/bin:/usr/sbin:/usr/bin
CPU = cpu_number
ACTION = action

action=(add|remove)

red is actual values
green is replaced with runtime information

Docking station event environment

argv[0] = hotplug_path
argv[1] = dock
argv[2] = 0

HOME = /
PATH = /sbin:/bin:/usr/sbin:/usr/bin
ACTION = action
DOCK = location_id/serial/capabilities

action=(add|remove)

red is actual values
green is replaced with runtime information

Small is good

```
#!/bin/sh
if [ "$1" = "usb" ]; then
    if [ "$INTERFACE" = "3/1/2" ]; then
        if [ "$ACTION" = "add" ]; then
            modprobe usbmouse
        else
            rmmod usbmouse
        fi
    fi
fi
```

Automatically sync your visor

```
#!/bin/sh
USER=gregkh
if [ "$1" = "usb" ]; then
    if [ "$PRODUCT" = "82d/100/0" ]; then
        if [ "$ACTION" = "add" ]; then
            modprobe visor
            su $USER - -c "/usr/bin/coldsync"
        else
            rmmod visor
        fi
    fi
fi
```

Rename your network connections

```
#!/bin/sh
if [ "$1" = "network" ]; then
    if [ "$ACTION" = "register" ]; then
        nameif -r $INTERFACE -c /etc/mactab
    fi
fi
```

thanks to Sukadev Bhattiprolu

Lots of devices

```
#!/bin/sh
if [ "$1" = "usb" ]; then
    if [ "$ACTION" = "add" ]; then
        PROGRAM="modprobe"
    else
        PROGRAM="rmmod"
    fi
    if [ "$PRODUCT" = "6cd/103/0" ]; then
        $PROGRAM keyspan_pda
    fi
    if [ "$INTERFACE" = "3/1/1" ]; then
        $PROGRAM usbkbd
    fi
    if [ "$PRODUCT" = "841/1/0" ]; then
        $PROGRAM rio500
    fi
fi
```

Here we go again

- Manually editing files.
- New drivers get added every week.
- Huge headache

PCI drivers for one device

drivers/net/sunhme.c

```
struct pci_device_id
  happymeal_pci_ids[] __initdata = {
    {
      vendor:    PCI_VENDOR_ID_SUN,
      device:    PCI_DEVICE_ID_SUN_HAPPYMEAL,
      subvendor: PCI_ANY_ID,
      subdevice: PCI_ANY_ID,
    },
    { }   /* Terminating entry */
};

MODULE_DEVICE_TABLE (pci, happymeal_pci_ids);
```

PCI driver for lots of devices

drivers/sound/i810_audio.c

```
static struct pci_device_id
  i810_pci_tbl [] __initdata = {
    { PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_82801,
      PCI_ANY_ID, PCI_ANY_ID, 0, 0, ICH82801AA},
    { PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_82901,
      PCI_ANY_ID, PCI_ANY_ID, 0, 0, ICH82901AB},
    { PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_440MX,
      PCI_ANY_ID, PCI_ANY_ID, 0, 0, INTEL440MX},
    { PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_ICH2,
      PCI_ANY_ID, PCI_ANY_ID, 0, 0, INTELICH2},
    { PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_ICH3,
      PCI_ANY_ID, PCI_ANY_ID, 0, 0, INTELICH3},
    {0,}
};

MODULE_DEVICE_TABLE (pci, i810_pci_tbl);
```

PCI driver for a class of devices

drivers/hotplug/cpqphp.c

```
static const struct pci_device_id
  __devinitdata hpcd_pci_tbl[] = { {

    /* handle any PCI Hotplug controller */
    class:      ((PCI_CLASS_SYSTEM_PCI_HOTPLUG << 8) | 0x00),
    class_mask: ~0,

    /* no matter who makes it */
    vendor:     PCI_ANY_ID,
    device:     PCI_ANY_ID,
    subvendor:  PCI_ANY_ID,
    subdevice:  PCI_ANY_ID,
    }, { /* end: all zeroes */ }
};

MODULE_DEVICE_TABLE (pci, hpcd_pci_tbl);
```

USB copied PCI

drivers/usb/mdc800.c

```
static struct usb_device_id
  mdc800_table [] = {
    { USB_DEVICE(MDC800_VENDOR_ID, MDC800_PRODUCT_ID) },
    { } /* Terminating entry */
};

MODULE_DEVICE_TABLE (usb, mdc800_table);
```

USB driver for many devices

drivers/usb/ov511.c

```
static __devinitdata struct usb_device_id
  device_table [] = {
    { USB_DEVICE(0x05a9, 0x0511) },  /* OV511 */
    { USB_DEVICE(0x05a9, 0xA511) },  /* OV511+ */
    { USB_DEVICE(0x0813, 0x0002) },  /* Intel Play Me2Cam */
    { } /* Terminating entry */
};

MODULE_DEVICE_TABLE (usb, device_table);
```

USB has class too

drivers/usb/printer.c

```
static struct usb_device_id usblp_ids [] = {
   { USB_INTERFACE_INFO(USB_CLASS_PRINTER, 1, 1) },
   { USB_INTERFACE_INFO(USB_CLASS_PRINTER, 1, 2) },
   { USB_INTERFACE_INFO(USB_CLASS_PRINTER, 1, 3) },
   { }    /* Terminating entry */
};

MODULE_DEVICE_TABLE (usb, usblp_ids);
```

USB has complicated classes

drivers/usb/audio.c

```
static struct usb_device_id usb_audio_ids [] = {
  {
    match_flags: (USB_DEVICE_ID_MATCH_INT_CLASS |
                  USB_DEVICE_ID_MATCH_INT_SUBCLASS),
    bInterfaceClass:    USB_CLASS_AUDIO,
    bInterfaceSubClass: 1,
  },
  { }    /* Terminating entry */
};

MODULE_DEVICE_TABLE (usb, usb_audio_ids);
```

Abuse the preprocessor

include/module.h

```
#define MODULE_GENERIC_TABLE(gtype,name)
   static const unsigned long
      __module_##gtype##_size
      __attribute__ ((unused)) = sizeof(struct gtype##_id);
   static const struct gtype##_id
      *__module_##gtype##_table
       __attribute__ ((unused)) = name

#define MODULE_DEVICE_TABLE(type,name)
   MODULE_GENERIC_TABLE(type##_device,name)
```

Results of the preprocessor

```
$ nm drivers/usb/printer.o | grep module_usb
0000048c r __module_usb_device_size
000000d8 d __module_usb_device_table
```

```
$ nm drivers/sound/i810_audio.o | grep module_pci
00000178 r __module_pci_device_size
000000c8 d __module_pci_device_table
```

The magic of modutils

- depmod works wonders

/lib/modules/<kernel_version>/modules.pcimap

/lib/modules/<kernel_version>/modules.usbmap

After depmod is finished with pci

modules.pcimap

```
# pci module  vendor     device     subvendor  subdevice  class      class_mask driver_data
hotplug_pci   0xffffffff 0xffffffff 0xffffffff 0xffffffff 0x00080400 0xffffffff 0x00000000
i810_audio    0x00008086 0x00002415 0xffffffff 0xffffffff 0x00000000 0x00000000 0x00000000
i810_audio    0x00008086 0x00002425 0xffffffff 0xffffffff 0x00000000 0x00000000 0x00000001
i810_audio    0x00008086 0x00007195 0xffffffff 0xffffffff 0x00000000 0x00000000 0x00000002
i810_audio    0x00008086 0x00002445 0xffffffff 0xffffffff 0x00000000 0x00000000 0x00000003
i810_audio    0x00008086 0x00002485 0xffffffff 0xffffffff 0x00000000 0x00000000 0x00000004
```

After depmod is finished with usb

modules.usbmap

```
# usb module  match_flags idVendor idProduct bcdDevice_lo bcdDevice_hi bDeviceClass
    bDeviceSubClass bDeviceProtocol bInterfaceClass bInterfaceSubClass bInterfaceProtocol
      driver_info

audio         0x0180       0x0000     0x0000     0x0000       0x0000       0x00
      0x00            0x00            0x01            0x01            0x00
        0x00000000

mdc800        0x0003       0x055f     0xa800     0x0000       0x0000       0x00
      0x00            0x00            0x00            0x00            0x00
        0x00000000

printer       0x0380       0x0000     0x0000     0x0000       0x0000       0x00
      0x00            0x00            0x07            0x01            0x01
        0x00000000

printer       0x0380       0x0000     0x0000     0x0000       0x0000       0x00
      0x00            0x00            0x07            0x01            0x02
        0x00000000

printer       0x0380       0x0000     0x0000     0x0000       0x0000       0x00
      0x00            0x00            0x07            0x01            0x03
        0x00000000
```

Take advantage of automation

http://linux-hotplug.sourceforge.net

the future

- hotplug kernel patches
  - SCSI
  - better PCMCIA support
  - IDE
  - PCI and cPCI
  - CPU
  - Docking station (-ac kernel now)
- enhanced script support
- better distro support
- obsoleted by ACPI?

Linux Hotplugging

http://linux-hotplug.sourceforge.net/

Greg Kroah-Hartman

greg@kroah.com http://www.kroah.com/linux/