热插拔驱动：

以前我们写驱动的时候创建设备节点的时候用：

//mdev会根据这些信息自动创建

class\_device\_create()//创建设备节点

class\_device\_create()

class\_device\_register()

class\_device\_add(class\_dev);

kobject\_uevent()

kobject\_ueven\_env()

action\_string = action\_to\_string(action);

/\* environment index 分配环境变量\*/

envp = kzalloc(NUM\_ENVP \* sizeof (char \*), GFP\_KERNEL);

/\* environment values 分配保存环境变量的内存\*/

buffer = kmalloc(BUFFER\_SIZE, GFP\_KERNEL);

/\* event environemnt for helper process only设置环境变量的值通过命令env查看 \*/

envp[i++] = "HOME=/";

envp[i++] = "PATH=/sbin:/bin:/usr/sbin:/usr/bin";

/\* default keys \*/

scratch = buffer;

envp [i++] = scratch;

scratch += sprintf(scratch, "ACTION=%s", action\_string) + 1;

envp [i++] = scratch;

scratch += sprintf (scratch, "DEVPATH=%s", devpath) + 1;

envp [i++] = scratch;

scratch += sprintf(scratch, "SUBSYSTEM=%s", subsystem) + 1

if (uevent\_helper[0])//存在的话 调用应用程序 比如mdev

{

char \*argv [3];

//我们加上自己的测试代码

printk("Xiaohuihiu:uevent\_helper = %s\n",uevent\_helper);

for(i =0;envp[i] ;i++)

{

printk("envp[%d] = %s\n",i,envp[i]);

}

argv [0] = uevent\_helper;//”mdev”

argv [1] = (char \*)subsystem;

argv [2] = NULL;

call\_usermodehelper (argv[0], argv, envp, 0);//调用uevent\_helper;所指向的应用程序

}

1.例如我们在开发板上输入env:如下的信息这是我们的shell程序的环境变量，每个程序都有环境变量

USER=root

HOME=/

TERM=vt102

PATH=/sbin:/usr/sbin:/bin:/usr/bin

SHELL=/bin/sh

PWD=/

2. vi lib/kobject\_uevent.c +208 //我们将测试代码放在linux下编译

3．Make uImage 将生成的uiamge 拷贝成uImage\_uevent

4用新的image启动后当我们加载一个驱动后 例如 insmod Buttons.ko

Xiaohuihiu:uevent\_helper = /sbin/mdev //因为在启动脚本rcs echo /sbin/mdev > /proc/sys/kernel/hotplug 所以设置uevent\_helper为 /sbin/mdev

envp[0] = HOME=/

envp[1] = PATH=/sbin:/bin:/usr/sbin:/usr/bin

envp[2] = ACTION=add

envp[3] = DEVPATH=/class/input/input0/event0 //led驱动： DEVPATH=/devices/platform/myled

envp[4] = SUBSYSTEM=input

envp[5] = SEQNUM=732

envp[6] = MAJOR=13

envp[7] = MINOR=64 call\_usermodehelper 会调用 /sbin/mdev 所指向的应用程序。/sbin/mdev根据envp[1] envp[2] 这些环境变量创建设备节点。

medev 根据ACTION=add sys/class/input/input0/event0 调用mknod创建设备节点

根据/sys/class/input/input0/event0/dev 确定主设备号只要为c 就是设备节点

1.分析busy\_box：

int mdev\_main(int argc, char \*\*argv)

{//这块不是特别清楚只需要明白call\_usermodehelper 调用uevent\_helper 所指向的应用程序/sbin/mdev

if (argc == 2 && !strcmp(argv[1],"-s")) {/\*目前我们不是这一项因为我们有mdev -s\*/

}

else

{

action = getenv("ACTION");// ACTION 是add

env\_path = getenv("DEVPATH");

if (!strcmp(action, "remove"))

make\_device(temp, 1);

// temp= sys/class/input/input0/event0 // sys/class 下都是字符设备

// 将sys/class/input/input0/event0/dev 得到住设备号

device\_name = bb\_basename(path);//确定设备文件名类型主次设备号 例如buttons

if (mknod(device\_name, mode | type, makedev(major, minor)) && errno != EEXIST)// mknod创建设备节点

else if (!strcmp(action, "add")) {

make\_device(temp, 0);

}

}

我们的脚本程序有mdev -s

# cat /etc/init.d/rcS

mount -a

mkdir /dev/pts

mount -t devpts devpts /dev/pts

echo /sbin/mdev > /proc/sys/kernel/hotplug

mdev -s

2. mdev //可以查看帮助信息 。

最后总结也就是当我们在写驱动程序的时候。调用class\_create()。他会设置环境变量。并最终调用：

call\_usermodehelper (argv[0], argv, envp, 0);//调用uevent\_helper;所指向的应用程序。也就是mdev

mdev最后根据环境变量创建设备节点。

想接上U盘自动挂载：

当我们插入u盘时候会显示：sd 0:0:0:0: [sda] Attached SCSI removable disk

我么输入 ls -l /dev/sda\*

一般有如下的信息：

brw-rw---- 1 0 0 8, 0 Jan 1 00:00 /dev/sda//整个u盘

brw-rw---- 1 0 0 8, 0 Jan 1 00:00 /dev/sda1 //这个表示第一个分区

挂载：

mount /dev/sad1 /mnt

卸载：

umount /mnt：

想自动挂载u盘：

在我们的udev源码中：mdev.c

有如下的代码他表示有这个配置文件的话/etc/mdev.conf 就做某些事情。没有的话mknod

if (ENABLE\_FEATURE\_MDEV\_CONF) {

char \*conf, \*pos, \*end;

int line, fd;

/\* mmap the config file \*/

fd = open("/etc/mdev.conf", O\_RDONLY);

}

if (mknod(device\_name, mode | type, makedev(major, minor)) && errno != EEXIST)

在busybox下搜索；mdev.txt 有个介绍文档。

MDEV Config (/etc/mdev.conf)

the format:

<device regex> <uid>:<gid> <octal permissions> [<@|$|\*> <command>] //

device regex：正则表达式

uid: owner 设备节点的用户

gid: 组id 设备节点的组

octal permissions>以八进制显示属性。mdevc创建的设备默认的属性就是660，可以在这个文件中修改属性

command；命令 这是挂载的关键所在因为mdev在分析这个文件时发现这个设备节点（device regex）有这个命令的话就去执行

@|$|\*：

@ Run after creating the device. //创建设备节点后执行命令

$ Run before removing the device. //创建设备节点前执行命令

\* Run both after creating and before removing the device. //在创建设备节点后和删除之前执行命令

什么是正则表达式：<https://deerchao.cn/tutorials/regex/regex.htm>

一般表示正则表达式

.表示任意字符换行符除外

\*重复0次或者更多次

+重复1次或者更多次

？重复0次或者1次

[abc]表示字符里面的某一个 [1-9] 1-9中的某一个

假如我们想改变某个驱动的属性：

# insmod first\_drv.ko //加载我们写的first\_drv.ko这个驱动

# ls -l /dev/xyz

crw-rw---- 1 0 0 252, 0 Jan 1 01:14 /dev/xyz

写mdev.conf 来设置

vi /etc/mdev.conf :加上

1.xyz 0:0 777

再次去查看的话：ls –l /dev/xyz\*

crwxrwxrwx 1 0 0 252, 0 Jan 1 01:19 /dev/xyz //属性已经被改

这样很麻烦用正则表达式： xyz?[123]? 0:0 777 \* /bin/add\_rmoce\_sd.sh

假如我们dev下有leds led1 led2 led3

2可以修改为如下Leds?[123]？0:0 777 //也是同样的效果

3.假如想在创建设备节点后执行命令：

leds?[123]? 0:0 777 @ echo create /dev/$MDEV > /dev/console ///$MDEV这个环境变量表示呢个设备节点. leds?[123]? ?表示出现一次或者0

次。[123]?表示有可能出现123 的次数时0次或者一次

会有如下的打印：

Create /dev/leds

Create /dev/led1

Create /dev/led2

Create /dev/led3

我们只有xyz:所以试下：

xyz?[123]? 0:0 777

同样也试下这个：

xyz?[123]? 0:0 777 @ echo create /dev/$MDEV > /dev/console

create /dev/xyz

4. xyz?[123]? 0:0 777 \* if [$ACTION = “add”]: then echo create /dev/$MDEV > /dev/console; else echo remove /dev/$MDEV > /dev/console fi

\*表示在创建设被节点后或者卸载之前都执行命令。$MDEV环境变量是我们创建的设备节点

5.把命令写入一个脚本；例如/bin/add\_remodev\_led.sh

add\_remodev\_led.sh //$ACTION 环境变量是add 或者remove

#!/bin/sh

if [$ACTION = “add”]:

then echo create /dev/$MDEV > /dev/console;

else

echo remove /dev/$MDEV > /dev/console;

fi //结束

我自己测试：//第一次测试insmod 不能打印出# create /dev/xyz 可能是if [ $ACTION = "add" ]的格式后面详细研究

#!/bin/sh

if [ $ACTION = "add" ];

then

echo create /dev/$MDEV > /dev/console;

else

echo remove /dev/$MDEV > /dev/console;

fi

chmod +x add\_remodev\_led.sh //加上可执行权限

将mdev.conf 执行命令的地方改为脚本的位置即可

xyz?[123]? 0:0 777 \* /bin/add\_rmoveled.sh

6 u盘自动加载 变可以做出来

ls /dev/sd1或者其他

sda[1-9]+ 0:0 777 \* /bin/add\_rmoveled.sh //记得要改为sda 因为dev下的设备节点时sda记得是+号

sda[1-9]+ 0:0 777 \* if [$ACTION = “add”]: then mount /dev/$MDEV /mnt; else umount /mnt ；fi

cat /proc/mounts //可以查看挂载

就可以看到/dev/sda1 /mnt

就可以查看 ls –l 里面的内容

因为我当时做根文件系统的时候没有做mnt 目录所以挂载在ts\_dir下：

最后执行的效果：

cat /proc/mounts

rootfs / rootfs rw 0 0

/dev/root / nfs rw,vers=2,rsize=4096,wsize=4096,hard,nolock,proto=udp,timeo=11,retrans=2,sec=sys,addr=192.168.0.105 0 0

proc /proc proc rw 0 0

sysfs /sys sysfs rw 0 0

tmpfs /dev tmpfs rw 0 0

devpts /dev/pts devpts rw 0 0

/dev/sda1 /ts\_dir vfat rw,fmask=0000,dmask=0000,codepage=cp936,iocharset=cp936 0 0