Linux and parallel port IDE devices

PARIDE v1.03 (c) 1997-8 Grant Guenther <grant@torque.net>

1. Introduction

Owing to the simplicity and near universality of the parallel port interface to personal computers, many external devices such as portable hard-disk, CD-ROM, LS-120 and tape drives use the parallel port to connect to their host computer. While some devices (notably scanners) use ad-hoc methods to pass commands and data through the parallel port interface, most external devices are actually identical to an internal model, but with a parallel-port adapter chip added in. Some of the original parallel port adapters were little more than mechanisms for multiplexing a SCSI bus. (The Iomega PPA-3 adapter used in the ZIP drives is an example of this approach). Most current designs, however, take a different approach. The adapter chip reproduces a small ISA or IDE bus in the external device and the communication protocol provides operations for reading and writing device registers, as well as data block transfer functions. Sometimes, the device being addressed via the parallel cable is a standard SCSI controller like an NCR 5380. The "ditto" family of external tape drives use the ISA replicator to interface a floppy disk controller, which is then connected to a floppy-tape mechanism. The vast majority of external parallel port devices, however, are now based on standard IDE type devices, which require no intermediate controller. If one were to open up a parallel port CD-ROM drive, for instance, one would find a standard ATAPI CD-ROM drive, a power supply, and a single adapter that interconnected a standard PC parallel port cable and a standard IDE cable. It is usually possible to exchange the CD-ROM device with any other device using the IDE interface.

The document describes the support in Linux for parallel port IDE devices. It does not cover parallel port SCSI devices, "ditto" tape drives or scanners. Many different devices are supported by the parallel port IDE subsystem, including:

- MicroSolutions backpack CD-ROM
- MicroSolutions backpack PD/CD
- MicroSolutions backpack hard-drives
- MicroSolutions backpack 8000t tape drive
- SyQuest EZ-135, EZ-230 & SparQ drives
- Avatar Shark
- Imation Superdisk LS-120
- Maxell Superdisk LS-120
- FreeCom Power CD
- Hewlett-Packard 5GB and 8GB tape drives
- Hewlett-Packard 7100 and 7200 CD-RW drives

as well as most of the clone and no-name products on the market.

To support such a wide range of devices, PARIDE, the parallel port IDE subsystem, is actually structured in three parts. There is a base paride module which provides a registry and some common methods for accessing the parallel ports. The second component is a set of high-level drivers for each of the different types of supported devices:

pd	IDE disk
pcd	ATAPI CD-ROM
pf	ATAPI disk
pt	ATAPI tape
pg	ATAPI generic

(Currently, the pg driver is only used with CD-R drives).

The high-level drivers function according to the relevant standards. The third component of PARIDE is a set of low-level protocol drivers for each of the parallel port IDE adapter chips. Thanks to the interest and encouragement of Linux users from many parts of the world, support is available for almost all known adapter protocols:

aten	ATEN EH-100	(HK)
bpck	Microsolutions backpack	(US)
comm	DataStor (old-type) "commuter" adapter	(TW)
dstr	DataStor EP-2000	(TW)
epat	Shuttle EPAT	(UK)
epia	Shuttle EPIA	(UK)
fit2	FIT TD-2000	(US)
fit3	FIT TD-3000	(US)
friq	Freecom IQ cable	(DE)
frpw	Freecom Power	(DE)
kbic	KingByte KBIC-951A and KBIC-971A	(TW)
ktti	KT Technology PHd adapter	(SG)
on20	OnSpec 90c20	(US)
on26	OnSpec 90c26	(US)

2. Using the PARIDE subsystem

While configuring the Linux kernel, you may choose either to build the PARIDE drivers into your kernel, or to build them as modules.

In either case, you will need to select "Parallel port IDE device support" as well as at least one of the high-level drivers and at least one of the parallel port communication protocols. If you do not know what kind of parallel port adapter is used in your drive, you could begin by checking the file names and any text files on your DOS installation floppy. Alternatively, you can look at the markings on the adapter chip itself. That's usually sufficient to identify the correct device.

You can actually select all the protocol modules, and allow the PARIDE subsystem to try them all for you.

For the "brand-name" products listed above, here are the protocol and high-level drivers that you would use:

Manufacturer	Model	Driver	Protocol
MicroSolutions	CD-ROM	pcd	bpck
MicroSolutions	PD drive	pf	bpck
MicroSolutions	hard-drive	pd	bpck

MicroSolutions	8000t tape	pt	bpck
SyQuest	EZ, SparQ	pd	epat
Imation	Superdisk	pf	epat
Maxell	Superdisk	pf	friq
Avatar	Shark	pd	epat
FreeCom	CD-ROM	pcd	frpw
Hewlett-Packard	5GB Tape	pt	epat
Hewlett-Packard	7200e (CD)	pcd	epat
Hewlett-Packard	7200e (CD-R)	pg	epat

2.1 Configuring built-in drivers

We recommend that you get to know how the drivers work and how to configure them as loadable modules, before attempting to compile a kernel with the drivers built-in.

If you built all of your PARIDE support directly into your kernel, and you have just a single parallel port IDE device, your kernel should locate it automatically for you. If you have more than one device, you may need to give some command line options to your bootloader (eg: LILO), how to do that is beyond the scope of this document.

The high-level drivers accept a number of command line parameters, all of which are documented in the source files in linux/drivers/block/paride. By default, each driver will automatically try all parallel ports it can find, and all protocol types that have been installed, until it finds a parallel port IDE adapter. Once it finds one, the probe stops. So, if you have more than one device, you will need to tell the drivers how to identify them. This requires specifying the port address, the protocol identification number and, for some devices, the drive's chain ID. While your system is booting, a number of messages are displayed on the console. Like all such messages, they can be reviewed with the 'dmesg' command. Among those messages will be some lines like:

```
paride: bpck registered as protocol 0
paride: epat registered as protocol 1
```

The numbers will always be the same until you build a new kernel with different protocol selections. You should note these numbers as you will need them to identify the devices.

If you happen to be using a MicroSolutions backpack device, you will also need to know the unit ID number for each drive. This is usually the last two digits of the drive's serial number (but read MicroSolutions' documentation about this).

As an example, let's assume that you have a MicroSolutions PD/CD drive with unit ID number 36 connected to the parallel port at 0x378, a SyQuest EZ-135 connected to the chained port on the PD/CD drive and also an Imation Superdisk connected to port 0x278. You could give the following options on your boot command:

```
pd.drive0=0x378,1 pf.drive0=0x278,1 pf.drive1=0x378,0,36
```

In the last option, pf.drive1 configures device /dev/pf1, the 0x378

is the parallel port base address, the 0 is the protocol registration number and 36 is the chain ID.

Please note: while PARIDE will work both with and without the PARPORT parallel port sharing system that is included by the "Parallel port support" option, PARPORT must be included and enabled if you want to use chains of devices on the same parallel port.

2.2 Loading and configuring PARIDE as modules

It is much faster and simpler to get to understand the PARIDE drivers if you use them as loadable kernel modules.

Note 1: using these drivers with the "kerneld" automatic module loading system is not recommended for beginners, and is not documented here.

Note 2: if you build PARPORT support as a loadable module, PARIDE must also be built as loadable modules, and PARPORT must be loaded before the PARIDE modules.

To use PARIDE, you must begin by

```
insmod paride
```

this loads a base module which provides a registry for the protocols, among other tasks.

Then, load as many of the protocol modules as you think you might need. As you load each module, it will register the protocols that it supports, and print a log message to your kernel log file and your console. For example:

```
# insmod epat
paride: epat registered as protocol 0
# insmod kbic
paride: k951 registered as protocol 1
paride: k971 registered as protocol 2
```

Finally, you can load high-level drivers for each kind of device that you have connected. By default, each driver will autoprobe for a single device, but you can support up to four similar devices by giving their individual co-ordinates when you load the driver.

For example, if you had two no-name CD-ROM drives both using the KingByte KBIC-951A adapter, one on port 0x378 and the other on 0x3bc you could give the following command:

```
# insmod pcd drive0=0x378,1 drive1=0x3bc,1
```

For most adapters, giving a port address and protocol number is sufficient, but check the source files in linux/drivers/block/paride for more information. (Hopefully someone will write some man pages one day !).

As another example, here's what happens when PARPORT is installed, and a SyQuest EZ-135 is attached to port 0x378:

```
        # insmod paride
        paride: version 1.0 installed
        # insmod epat
        paride: epat registered as protocol 0
        # insmod pd
        pd: pd version 1.0, major 45, cluster 64, nice 0
        pda: Sharing parport1 at 0x378
        pda: epat 1.0, Shuttle EPAT chip c3 at 0x378, mode 5 (EPP-32), delay 1
        pda: SyQuest EZ135A, 262144 blocks [128M], (512/16/32), removable media
         pda: pda1
```

Note that the last line is the output from the generic partition table scanner - in this case it reports that it has found a disk with one partition.

2.3 Using a PARIDE device

Once the drivers have been loaded, you can access PARIDE devices in the same way as their traditional counterparts. You will probably need to create the device "special files". Here is a simple script that you can cut to a file and execute:

```
#!/bin/bash
#
# mkd -- a script to create the device special files for the PARIDE subsystem
#
function mkdev {
  mknod $1 $2 $3 $4 ; chmod 0660 $1 ; chown root:disk $1
}
#
function pd {
  D=$( printf \\$( printf "x%03x" $[ $1 + 97 ] ) )
  mkdev pd$D b 45 $[ $1 * 16 ]
  for P in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
  do mkdev pd$D$P b 45 $[ $1 * 16 + $P ]
  done
}
#
cd /dev
#
for u in 0 1 2 3 ; do pd $u ; done
for u in 0 1 2 3 ; do mkdev pcd$u b 46 $u ; done
for u in 0 1 2 3 ; do mkdev pf$u  b 47 $u ; done
for u in 0 1 2 3 ; do mkdev pt$u  c 96 $u ; done
for u in 0 1 2 3 ; do mkdev npt$u c 96 $[ $u + 128 ] ; done
for u in 0 1 2 3 ; do mkdev pg$u  c 97 $u ; done
#
# end of mkd
```

With the device files and drivers in place, you can access PARIDE devices like any other Linux device. For example, to mount a CD-ROM in pcd0, use:

```
        mount /dev/pcd0 /cdrom
```

If you have a fresh Avatar Shark cartridge, and the drive is pda, you might do something like:

```
                                paride.txt
        fdisk /dev/pda          -- make a new partition table with
                                   partition 1 of type 83

        mke2fs /dev/pda1        -- to build the file system

        mkdir /shark            -- make a place to mount the disk

        mount /dev/pda1 /shark
```

Devices like the Imation superdisk work in the same way, except that they do not have a partition table. For example to make a 120MB floppy that you could share with a DOS system:

```
        mkdosfs /dev/pf0
        mount /dev/pf0 /mnt
```

2.4 The pf driver

The pf driver is intended for use with parallel port ATAPI disk devices. The most common devices in this category are PD drives and LS-120 drives. Traditionally, media for these devices are not partitioned. Consequently, the pf driver does not support partitioned media. This may be changed in a future version of the driver.

2.5 Using the pt driver

The pt driver for parallel port ATAPI tape drives is a minimal driver. It does not yet support many of the standard tape ioctl operations. For best performance, a block size of 32KB should be used. You will probably want to set the parallel port delay to 0, if you can.

2.6 Using the pg driver

The pg driver can be used in conjunction with the cdrecord program to create CD-ROMs. Please get cdrecord version 1.6.1 or later from ftp://ftp.fokus.gmd.de/pub/unix/cdrecord/ . To record CD-R media your parallel port should ideally be set to EPP mode, and the "port delay" should be set to 0. With those settings it is possible to record at 2x speed without any buffer underruns. If you cannot get the driver to work in EPP mode, try to use "bidirectional" or "PS/2" mode and 1x speeds only.

3. Troubleshooting

3.1 Use EPP mode if you can

The most common problems that people report with the PARIDE drivers concern the parallel port CMOS settings. At this time, none of the PARIDE protocol modules support ECP mode, or any ECP combination modes. If you are able to do so, please set your parallel port into EPP mode using your CMOS setup procedure.

3.2 Check the port delay

Some parallel ports cannot reliably transfer data at full speed. To

offset the errors, the PARIDE protocol modules introduce a "port delay" between each access to the i/o ports. Each protocol sets a default value for this delay. In most cases, the user can override the default and set it to 0 - resulting in somewhat higher transfer rates. In some rare cases (especially with older 486 systems) the default delays are not long enough. if you experience corrupt data transfers, or unexpected failures, you may wish to increase the port delay. The delay can be programmed using the "driveN" parameters to each of the high-level drivers. Please see the notes above, or read the comments at the beginning of the driver source files in linux/drivers/block/paride.

3.3 Some drives need a printer reset

There appear to be a number of "noname" external drives on the market that do not always power up correctly. We have noticed this with some drives based on OnSpec and older Freecom adapters. In these rare cases, the adapter can often be reinitialised by issuing a "printer reset" on the parallel port. As the reset operation is potentially disruptive in multiple device environments, the PARIDE drivers will not do it automatically. You can however, force a printer reset by doing:

```
insmod lp reset=1
rmmod lp
```

If you have one of these marginal cases, you should probably build your paride drivers as modules, and arrange to do the printer reset before loading the PARIDE drivers.

3.4 Use the verbose option and dmesg if you need help

While a lot of testing has gone into these drivers to make them work as smoothly as possible, problems will arise. If you do have problems, please check all the obvious things first: does the drive work in DOS with the manufacturer's drivers ? If that doesn't yield any useful clues, then please make sure that only one drive is hooked to your system, and that either (a) PARPORT is enabled or (b) no other device driver is using your parallel port (check in /proc/ioports). Then, load the appropriate drivers (you can load several protocol modules if you want) as in:

```
# insmod paride
# insmod epat
# insmod bpck
# insmod kbic
...
# insmod pd verbose=1
```

(using the correct driver for the type of device you have, of course). The verbose=1 parameter will cause the drivers to log a trace of their activity as they attempt to locate your drive.

Use 'dmesg' to capture a log of all the PARIDE messages (any messages beginning with paride:, a protocol module's name or a driver's name) and include that with your bug report. You can submit a bug report in one of two ways. Either send it directly to the author of the PARIDE suite,

by e-mail to grant@torque.net, or join the linux-parport mailing list and post your report there.

3.5 For more information or help

You can join the linux-parport mailing list by sending a mail message to

linux-parport-request@torque.net

with the single word

subscribe

in the body of the mail message (not in the subject line). Please be sure that your mail program is correctly set up when you do this, as the list manager is a robot that will subscribe you using the reply address in your mail headers. REMOVE any anti-spam gimmicks you may have in your mail headers, when sending mail to the list server.

You might also find some useful information on the linux-parport web pages (although they are not always up to date) at

http://www.torque.net/parport/