The `parport' code provides parallel-port support under Linux. This
includes the ability to share one port between multiple device
drivers.

You can pass parameters to the parport code to override its automatic
detection of your hardware. This is particularly useful if you want
to use IRQs, since in general these can't be autoprobed successfully.
By default IRQs are not used even if they _can_ be probed. This is
because there are a lot of people using the same IRQ for their
parallel port and a sound card or network card.

The parport code is split into two parts: generic (which deals with
port-sharing) and architecture-dependent (which deals with actually
using the port).

Parport as modules

If you load the parport code as a module, say

```
	# insmod parport
```

to load the generic parport code. You then must load the
architecture-dependent code with (for example):

```
	# insmod parport_pc io=0x3bc,0x378,0x278 irq=none,7,auto
```

to tell the parport code that you want three PC-style ports, one at
0x3bc with no IRQ, one at 0x378 using IRQ 7, and one at 0x278 with an
auto-detected IRQ. Currently, PC-style (parport_pc), Sun `bpp',
Amiga, Atari, and MFC3 hardware is supported.

PCI parallel I/O card support comes from parport_pc. Base I/O
addresses should not be specified for supported PCI cards since they
are automatically detected.

KMod

If you use kmod, you will find it useful to edit /etc/modprobe.conf.
Here is an example of the lines that need to be added:

```
	alias parport_lowlevel parport_pc
	options parport_pc io=0x378,0x278 irq=7,auto
```

KMod will then automatically load parport_pc (with the options
"io=0x378,0x278 irq=7,auto") whenever a parallel port device driver
(such as lp) is loaded.

Note that these are example lines only! You shouldn't in general need
to specify any options to parport_pc in order to be able to use a
parallel port.

Parport probe [optional]

```
-------------
```

In 2.2 kernels there was a module called parport_probe, which was used
for collecting IEEE 1284 device ID information. This has now been
enhanced and now lives with the IEEE 1284 support. When a parallel
port is detected, the devices that are connected to it are analysed,
and information is logged like this:

```
        parport0: Printer, BJC-210 (Canon)
```

The probe information is available from files in /proc/sys/dev/parport/.

Parport linked into the kernel statically

```
=========================================
```

If you compile the parport code into the kernel, then you can use
kernel boot parameters to get the same effect. Add something like the
following to your LILO command line:

```
        parport=0x3bc parport=0x378,7 parport=0x278,auto,nofifo
```

You can have many `parport=...' statements, one for each port you want
to add. Adding `parport=0' to the kernel command-line will disable
parport support entirely. Adding `parport=auto' to the kernel
command-line will make parport use any IRQ lines or DMA channels that
it auto-detects.

Files in /proc

```
==============
```

If you have configured the /proc filesystem into your kernel, you will
see a new directory entry: /proc/sys/dev/parport. In there will be a
directory entry for each parallel port for which parport is
configured. In each of those directories are a collection of files
describing that parallel port.

The /proc/sys/dev/parport directory tree looks like:

```
parport
|-- default
|   |-- spintime
|   `-- timeslice
|-- parport0
|   |-- autoprobe
|   |-- autoprobe0
|   |-- autoprobe1
|   |-- autoprobe2
|   |-- autoprobe3
|   |-- devices
|   |   |-- active
|   |   `-- lp
|   |       `-- timeslice
|   |-- base-addr
|    |-- irq
|    |-- dma
|    |-- modes
|    `-- spintime
`-- parport1
    |-- autoprobe
    |-- autoprobe0
    |-- autoprobe1
    |-- autoprobe2
    |-- autoprobe3
    |-- devices
    |   |-- active
    |   `-- ppa
    |       `-- timeslice
    |-- base-addr
    |-- irq
    |-- dma
    |-- modes
    `-- spintime


File:           Contents:

devices/active  A list of the device drivers using that port.  A "+"
                will appear by the name of the device currently using
                the port (it might not appear against any).  The
                string "none" means that there are no device drivers
                using that port.

base-addr       Parallel port's base address, or addresses if the port
                has more than one in which case they are separated
                with tabs.  These values might not have any sensible
                meaning for some ports.

irq             Parallel port's IRQ, or -1 if none is being used.

dma             Parallel port's DMA channel, or -1 if none is being
                used.

modes           Parallel port's hardware modes, comma-separated,
                meaning:

                PCSPP           PC-style SPP registers are available.
                TRISTATE        Port is bidirectional.
                COMPAT          Hardware acceleration for printers is
                                available and will be used.
                EPP             Hardware acceleration for EPP protocol
                                is available and will be used.
                ECP             Hardware acceleration for ECP protocol
                                is available and will be used.
                DMA             DMA is available and will be used.

                Note that the current implementation will only take
                advantage of COMPAT and ECP modes if it has an IRQ
                line to use.
```

```
autoprobe       Any IEEE-1284 device ID information that has been
                acquired from the (non-IEEE 1284.3) device.

autoprobe[0-3]  IEEE 1284 device ID information retrieved from
                daisy-chain devices that conform to IEEE 1284.3.

spintime        The number of microseconds to busy-loop while waiting
                for the peripheral to respond.  You might find that
                adjusting this improves performance, depending on your
                peripherals.  This is a port-wide setting, i.e. it
                applies to all devices on a particular port.

timeslice       The number of milliseconds that a device driver is
                allowed to keep a port claimed for.  This is advisory,
                and driver can ignore it if it must.

default/*       The defaults for spintime and timeslice. When a new
                port is registered, it picks up the default spintime.
                When a new device is registered, it picks up the
                default timeslice.
```

Device drivers
==============

Once the parport code is initialised, you can attach device drivers to specific ports. Normally this happens automatically; if the lp driver is loaded it will create one lp device for each port found. You can override this, though, by using parameters either when you load the lp driver:

```
        # insmod lp parport=0,2
```

or on the LILO command line:

```
        lp=parport0 lp=parport2
```

Both the above examples would inform lp that you want /dev/lp0 to be the first parallel port, and /dev/lp1 to be the _third_ parallel port, with no lp device associated with the second port (parport1). Note that this is different to the way older kernels worked; there used to be a static association between the I/O port address and the device name, so /dev/lp0 was always the port at 0x3bc. This is no longer the case - if you only have one port, it will default to being /dev/lp0, regardless of base address.

Also:

* If you selected the IEEE 1284 support at compile time, you can say `lp=auto' on the kernel command line, and lp will create devices only for those ports that seem to have printers attached.

* If you give PLIP the `timid' parameter, either with `plip=timid' on the command line, or with `insmod plip timid=1' when using modules, it will avoid any ports that seem to be in use by other devices.

* IRQ autoprobing works only for a few port types at the moment.

Reporting printer problems with parport

If you are having problems printing, please go through these steps to try to narrow down where the problem area is.

When reporting problems with parport, really you need to give all of the messages that parport_pc spits out when it initialises. There are several code paths:

o polling
o interrupt-driven, protocol in software
o interrupt-driven, protocol in hardware using PIO
o interrupt-driven, protocol in hardware using DMA

The kernel messages that parport_pc logs give an indication of which code path is being used. (They could be a lot better actually..)

For normal printer protocol, having IEEE 1284 modes enabled or not should not make a difference.

To turn off the 'protocol in hardware' code paths, disable CONFIG_PARPORT_PC_FIFO. Note that when they are enabled they are not necessarily _used_; it depends on whether the hardware is available, enabled by the BIOS, and detected by the driver.

So, to start with, disable CONFIG_PARPORT_PC_FIFO, and load parport_pc with 'irq=none'. See if printing works then. It really should, because this is the simplest code path.

If that works fine, try with 'io=0x378 irq=7' (adjust for your hardware), to make it use interrupt-driven in-software protocol.

If _that_ works fine, then one of the hardware modes isn't working right. Enable CONFIG_PARPORT_PC_FIFO (no, it isn't a module option, and yes, it should be), set the port to ECP mode in the BIOS and note the DMA channel, and try with:

```
    io=0x378 irq=7 dma=none (for PIO)
    io=0x378 irq=7 dma=3 (for DMA)
```
--
philb@gnu.org
tim@cyberelk.net