

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-02/msg13289.html>

- *From:* Borislav Petkov <petkovbb@xxxxxxxxxxxxxxxx>
 - *Date:* Tue, 26 Feb 2008 08:49:54 +0100
-

Signed-off-by: Borislav Petkov <petkovbb@xxxxxxxx>

drivers/ide/ide-cd.c | 634 ++++++-----
1 files changed, 323 insertions(+), 311 deletions(-)

```
diff --git a/drivers/ide/ide-cd.c b/drivers/ide/ide-cd.c
index 3600648..3853eb5 100644
--- a/drivers/ide/ide-cd.c
+++ b/drivers/ide/ide-cd.c
@@ -13,8 +13,8 @@
*
```

```
* Suggestions are welcome. Patches that work are more welcome though. ;-)
* For those wishing to work on this driver, please be sure you download
- * and comply with the latest Mt. Fuji (SFF8090 version 4) and ATAPI
- * (SFF-8020i rev 2.6) standards. These documents can be obtained by
+ * and comply with the latest Mt. Fuji (SFF8090 version 4) and ATAPI
+ * (SFF-8020i rev 2.6) standards. These documents can be obtained by
* anonymous ftp from:
* ftp://fission.dt.wdc.com/pub/standards/SFF\_atapi/spec/SFF8020-r2.6/PS/8020r26.ps
* ftp://ftp.avc-pioneer.com/Mtfuji4/Spec/Fuji4r10.pdf
@@ -41,17 +41,17 @@
```

```
#include <scsi/scsi.h> /* For SCSI -> ATAPI command conversion */
```

```

-#include <asm/irq.h>
-#include <asm/io.h>
+#include <linux/irq.h>
+#include <linux/io.h>
#include <asm/byteorder.h>
-#include <asm/uaccess.h>
+#include <linux/uaccess.h>
#include <asm/unaligned.h>
```

```
#include "ide-cd.h"
```

```
static DEFINE_MUTEX(idecd_ref_mutex);
```

```

-#define to_ide_cd(obj) container_of(obj, struct cdrom_info, kref)
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
+#define to_ide_cd(obj) container_of(obj, struct cdrom_info, kref)

#define ide_cd_g(disk) \
container_of((disk)->private_data, struct cdrom_info, driver)
@@ -77,13 +77,12 @@ static void ide_cd_put(struct cdrom_info *cd)
mutex_unlock(&idecd_ref_mutex);
}

-/******
+/*
* Generic packet command support and error handling routines.
*/

-/* Mark that we've seen a media change, and invalidate our internal
- buffers. */
-static void cdrom_saw_media_change(ide_drive_t *drive)
+/* Mark that we've seen a media change, and invalidate our internal buffers. */
+static void cdrom_saw_media_change(ide_drive_t *drive)
{
struct cdrom_info *cd = drive->driver_data;

@@ -100,46 +99,45 @@ static int cdrom_log_sense(ide_drive_t *drive, struct request *rq,
return 0;

switch (sense->sense_key) {
- case NO_SENSE: case RECOVERED_ERROR:
- break;
- case NOT_READY:
- /*
- * don't care about tray state messages for
- * e.g. capacity commands or in-progress or
- * becoming ready
- */
- if (sense->asc == 0x3a || sense->asc == 0x04)
- break;
- log = 1;
- break;
- case ILLEGAL_REQUEST:
- /*
- * don't log START_STOP unit with LoEj set, since
- * we cannot reliably check if drive can auto-close
- */
- if (rq->cmd[0] == GPCMD_START_STOP_UNIT && sense->asc == 0x24)
- break;
- log = 1;
- break;
- case UNIT_ATTENTION:
- /*
- * Make good and sure we've seen this potential media
- * change. Some drives (i.e. Creative) fail to present
- * the correct sense key in the error register.
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
- */
- cdrom_saw_media_change(drive);
+ case NO_SENSE: case RECOVERED_ERROR:
+ break;
+ case NOT_READY:
+ /*
+ * don't care about tray state messages for
+ * e.g. capacity commands or in-progress or
+ * becoming ready
+ */
+ if (sense->asc == 0x3a || sense->asc == 0x04)
break;
- default:
- log = 1;
+ log = 1;
+ break;
+ case ILLEGAL_REQUEST:
+ /*
+ * don't log START_STOP unit with LoEj set, since
+ * we cannot reliably check if drive can auto-close
+ */
+ if (rq->cmd[0] == GPCMD_START_STOP_UNIT &&
+ sense->asc == 0x24)
break;
+ log = 1;
+ break;
+ case UNIT_ATTENTION:
+ /*
+ * Make good and sure we've seen this potential media
+ * change. Some drives (i.e. Creative) fail to present
+ * the correct sense key in the error register.
+ */
+ cdrom_saw_media_change(drive);
+ break;
+ default:
+ log = 1;
+ break;
}
return log;
}

-static
-void cdrom_analyze_sense_data(ide_drive_t *drive,
- struct request *failed_command,
- struct request_sense *sense)
+static void cdrom_analyze_sense_data(ide_drive_t *drive,
+ struct request *failed_command, struct request_sense *sense)
{
unsigned long sector;
unsigned long bio_sectors;
@@ -150,16 +148,17 @@ void cdrom_analyze_sense_data(ide_drive_t *drive,
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
return;

/*
- * If a read toc is executed for a CD-R or CD-RW medium where
- * the first toc has not been recorded yet, it will fail with
- * 05/24/00 (which is a confusing error)
+ * If a read toc is executed for a CD-R or CD-RW medium where the first
+ * toc has not been recorded yet, it will fail with 05/24/00 (which is a
+ * confusing error)
*/
if (failed_command && failed_command->cmd[0] == GPCMD_READ_TOC_PMA_ATIP)
if (sense->sense_key == 0x05 && sense->asc == 0x24)
return;

- if (sense->error_code == 0x70) { /* Current Error */
- switch(sense->sense_key) {
+ /* Current Error */
+ if (sense->error_code == 0x70) {
+ switch (sense->sense_key) {
case MEDIUM_ERROR:
case VOLUME_OVERFLOW:
case ILLEGAL_REQUEST:
@@ -177,25 +176,23 @@ void cdrom_analyze_sense_data(ide_drive_t *drive,
if (bio_sectors < 4)
bio_sectors = 4;
if (drive->queue->hardsect_size == 2048)
- sector <= 2; /* Device sector size is 2K */
- sector &= ~(bio_sectors - 1);
+ /* Device sector size is 2K */
+ sector <= 2;
+ sector &= ~(bio_sectors - 1);
valid = (sector - failed_command->sector) << 9;

if (valid < 0)
valid = 0;
if (sector < get_capacity(info->disk) &&
- drive->probed_capacity - sector < 4 * 75) {
+ drive->probed_capacity - sector < 4 * 75)
set_capacity(info->disk, sector);
- }
- }
- }
+ }
+ }

ide_cd_log_error(drive->name, failed_command, sense);
}

-/*
- * Initialize a ide-cd packet command request
- */
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
+/* Initialize a ide-cd packet command request */
void ide_cd_init_rq(ide_drive_t *drive, struct request *rq)
{
struct cdrom_info *cd = drive->driver_data;
@@ -219,7 +216,8 @@ static void cdrom_queue_request_sense(ide_drive_t *drive, void *sense,

rq->data = sense;
rq->cmd[0] = GPCMD_REQUEST_SENSE;
- rq->cmd[4] = rq->data_len = 18;
+ rq->cmd[4] = 18;
+ rq->data_len = 18;

rq->cmd_type = REQ_TYPE_SENSE;

@@ -229,7 +227,7 @@ static void cdrom_queue_request_sense(ide_drive_t *drive, void *sense,
(void) ide_do_drive_cmd(drive, rq, ide_preempt);
}

-static void cdrom_end_request (ide_drive_t *drive, int uptodate)
+static void cdrom_end_request(ide_drive_t *drive, int uptodate)
{
struct request *rq = HWGROUP(drive)->rq;
int nsectors = rq->hard_cur_sectors;
@@ -279,20 +277,22 @@ static void cdrom_end_request (ide_drive_t *drive, int uptodate)
ide_end_request(drive, uptodate, nsectors);
}

-static void ide_dump_status_no_sense(ide_drive_t *drive, const char *msg, u8 stat)
+static void ide_dump_status_no_sense(ide_drive_t *drive, const char *msg, u8 st)
{
- if (stat & 0x80)
+ if (st & 0x80)
return;
- ide_dump_status(drive, msg, stat);
+ ide_dump_status(drive, msg, st);
}

-/* Returns 0 if the request should be continued.
- Returns 1 if the request was ended. */
+/* Returns:
+ * 0: if the request should be continued.
+ * 1: if the request was ended.
+ */
static int cdrom_decode_status(ide_drive_t *drive, int good_stat, int *stat_ret)
{
struct request *rq = HWGROUP(drive)->rq;
int stat, err, sense_key;
-
+
/* Check for errors. */
stat = ide_read_status(drive);
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
@@ -307,14 +307,17 @@ static int cdrom_decode_status(ide_drive_t *drive, int good_stat, int *stat_ret)
sense_key = err >> 4;

if (rq == NULL) {
- printk("%s: missing rq in cdrom_decode_status\n", drive->name);
+ printk(KERN_ERR "%s: missing rq in %s\n", drive->name,
+ __func__);
return 1;
}

if (blk_sense_request(rq)) {
- /* We got an error trying to get sense info
- from the drive (probably while trying
- to recover from a former error). Just give up. */
+ /*
+ * We got an error trying to get sense info from the drive
+ * (probably while trying to recover from a former error).
+ * Just give up.
+ */

rq->cmd_flags |= REQ_FAILED;
cdrom_end_request(drive, 0);
@@ -333,26 +336,26 @@ static int cdrom_decode_status(ide_drive_t *drive, int good_stat, int *stat_ret)

/* Check for tray open. */
if (sense_key == NOT_READY) {
- cdrom_saw_media_change (drive);
+ cdrom_saw_media_change(drive);
} else if (sense_key == UNIT_ATTENTION) {
/* Check for media change. */
- cdrom_saw_media_change (drive);
+ cdrom_saw_media_change(drive);
/*printk("%s: media changed\n",drive->name);*/
return 0;
- } else if ((sense_key == ILLEGAL_REQUEST) &&
- (rq->cmd[0] == GPCMD_START_STOP_UNIT)) {
- /*
- * Don't print error message for this condition--
- * SFF8090i indicates that 5/24/00 is the correct
- * response to a request to close the tray if the
- * drive doesn't have that capability.
- * cdrom_log_sense() knows this!
- */
+ } else if ((sense_key == ILLEGAL_REQUEST) &&
+ (rq->cmd[0] == GPCMD_START_STOP_UNIT)) {
+ /*
+ * Don't print error message for this condition--
+ * SFF8090i indicates that 5/24/00 is the correct
+ * response to a request to close the tray if the
+ * drive doesn't have that capability.
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
+ * cdrom_log_sense() knows this!
+ */
} else if (!(rq->cmd_flags & REQ_QUIET)) {
/* Otherwise, print an error. */
ide_dump_status(drive, "packet command error", stat);
}
-
+
rq->cmd_flags |= REQ_FAILED;

/*
@@ -373,19 +376,22 @@ static int cdrom_decode_status(ide_drive_t *drive, int good_stat, int *stat_ret)
if (sense_key == NOT_READY) {
/* Tray open. */
if (rq_data_dir(rq) == READ) {
- cdrom_saw_media_change (drive);
+ cdrom_saw_media_change(drive);

/* Fail the request. */
- printk ("%s: tray open\n", drive->name);
+ printk(KERN_ERR "%s: tray open\n", drive->name);
do_end_request = 1;
} else {
struct cdrom_info *info = drive->driver_data;

- /* allow the drive 5 seconds to recover, some
+ /*
+ * allow the drive 5 seconds to recover, some
+ * devices will return this error while flushing
- * data from cache */
+ * data from cache
+ */
if (!rq->errors)
- info->write_timeout = jiffies + ATAPI_WAIT_WRITE_BUSY;
+ info->write_timeout = jiffies +
+ ATAPI_WAIT_WRITE_BUSY;
rq->errors = 1;
if (time_after(jiffies, info->write_timeout))
do_end_request = 1;
@@ -404,31 +410,39 @@ static int cdrom_decode_status(ide_drive_t *drive, int good_stat, int *stat_ret)
}
} else if (sense_key == UNIT_ATTENTION) {
/* Media change. */
- cdrom_saw_media_change (drive);
+ cdrom_saw_media_change(drive);

- /* Arrange to retry the request.
- But be sure to give up if we've retried
- too many times. */
+ /*
+ * Arrange to retry the request, but be sure to give up
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
+ * if we've retried too many times.
+ */
if (++rq->errors > ERROR_MAX)
do_end_request = 1;
} else if (sense_key == ILLEGAL_REQUEST ||
sense_key == DATA_PROTECT) {
- /* No point in retrying after an illegal
- request or data protect error.*/
- ide_dump_status_no_sense (drive, "command error", stat);
+ /*
+ * No point in retrying after an illegal request or data
+ * protect error.
+ */
+ ide_dump_status_no_sense(drive, "command error", stat);
do_end_request = 1;
} else if (sense_key == MEDIUM_ERROR) {
- /* No point in re-trying a zillion times on a bad
- * sector... If we got here the error is not correctable */
- ide_dump_status_no_sense (drive, "media error (bad sector)", stat);
+ /*
+ * No point in re-trying a zillion times on a bad
+ * sector... If we got here the error is not
+ * correctable
+ */
+ ide_dump_status_no_sense(drive,
+ "media error (bad sector)",
+ stat);
do_end_request = 1;
} else if (sense_key == BLANK_CHECK) {
/* Disk appears blank ?? */
- ide_dump_status_no_sense (drive, "media error (blank)", stat);
+ ide_dump_status_no_sense(drive, "media error (blank)",
+ stat);
do_end_request = 1;
} else if ((err & ~ABRT_ERR) != 0) {
- /* Go to the default handler
- for other errors. */
+ /* Go to the default handler for other errors. */
ide_error(drive, "cdrom_decode_status", stat);
return 1;
} else if ((++rq->errors > ERROR_MAX)) {
@@ -436,16 +450,17 @@ static int cdrom_decode_status(ide_drive_t *drive, int good_stat, int *stat_ret)
do_end_request = 1;
}

- /* End a request through request sense analysis when we have
- sense data. We need this in order to perform end of media
- processing */
-
+ /*
+ * End a request through request sense analysis when we have
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
+ * sense data. We need this in order to perform end of media
+ * processing.
+ */
if (do_end_request)
goto end_request;

/*
- * If we got a CHECK_CONDITION status,
- * queue a request sense command.
+ * If we got a CHECK_CONDITION status, queue a request sense
+ * command.
*/
if (stat & ERR_STAT)
cdrom_queue_request_sense(drive, NULL, NULL);
@@ -479,35 +494,37 @@ static int cdrom_timer_expiry(ide_drive_t *drive)
unsigned long wait = 0;

/*
- * Some commands are *slow* and normally take a long time to
- * complete. Usually we can use the ATAPI "disconnect" to bypass
- * this, but not all commands/drives support that. Let
- * ide_timer_expiry keep polling us for these.
+ * Some commands are *slow* and normally take a long time to complete.
+ * Usually we can use the ATAPI "disconnect" to bypass this, but not all
+ * commands/drives support that. Let ide_timer_expiry keep polling us
+ * for these.
*/
switch (rq->cmd[0]) {
- case GPCMD_BLANK:
- case GPCMD_FORMAT_UNIT:
- case GPCMD_RESERVE_RZONE_TRACK:
- case GPCMD_CLOSE_TRACK:
- case GPCMD_FLUSH_CACHE:
- wait = ATAPI_WAIT_PC;
- break;
- default:
- if (!(rq->cmd_flags & REQ_QUIET))
- printk(KERN_INFO "ide-cd: cmd 0x%x timed out\n", rq->cmd[0]);
- wait = 0;
- break;
+ case GPCMD_BLANK:
+ case GPCMD_FORMAT_UNIT:
+ case GPCMD_RESERVE_RZONE_TRACK:
+ case GPCMD_CLOSE_TRACK:
+ case GPCMD_FLUSH_CACHE:
+ wait = ATAPI_WAIT_PC;
+ break;
+ default:
+ if (!(rq->cmd_flags & REQ_QUIET))
+ printk(KERN_INFO "ide-cd: cmd 0x%x timed out\n",
+ rq->cmd[0]);
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
+ wait = 0;
+ break;
}
return wait;
}

-/* Set up the device registers for transferring a packet command on DEV,
- expecting to later transfer XFERLEN bytes. HANDLER is the routine
- which actually transfers the command to the drive. If this is a
- drq_interrupt device, this routine will arrange for HANDLER to be
- called when the interrupt from the drive arrives. Otherwise, HANDLER
- will be called immediately after the drive is prepared for the transfer. */
-
+/*
+ * Set up the device registers for transferring a packet command on DEV,
+ * expecting to later transfer XFERLEN bytes. HANDLER is the routine which
+ * actually transfers the command to the drive. If this is a drq_interrupt
+ * device, this routine will arrange for HANDLER to be called when the interrupt
+ * from the drive arrives. Otherwise, HANDLER will be called immediately after
+ * the drive is prepared for the transfer.
+ */
static ide_startstop_t cdrom_start_packet_command(ide_drive_t *drive,
int xferlen,
ide_handler_t *handler)
@@ -534,7 +551,8 @@ static ide_startstop_t cdrom_start_packet_command(ide_drive_t *drive,
drive->waiting_for_dma = 0;

/* packet command */
- ide_execute_command(drive, WIN_PACKETCMD, handler, ATAPI_WAIT_PC, cdrom_timer_expiry);
+ ide_execute_command(drive, WIN_PACKETCMD, handler,
+ ATAPI_WAIT_PC, cdrom_timer_expiry);
return ide_started;
} else {
unsigned long flags;
@@ -550,15 +568,17 @@ static ide_startstop_t cdrom_start_packet_command(ide_drive_t *drive,
}
}

-/* Send a packet command to DRIVE described by CMD_BUF and CMD_LEN.
- The device registers must have already been prepared
- by cdrom_start_packet_command.
- HANDLER is the interrupt handler to call when the command completes
- or there's data ready. */
#define ATAPI_MIN_CDB_BYTES 12
-static ide_startstop_t cdrom_transfer_packet_command (ide_drive_t *drive,
- struct request *rq,
- ide_handler_t *handler)
+
+/*
+ * Send a packet command to DRIVE described by CMD_BUF and CMD_LEN. The device
+ * registers must have already been prepared by cdrom_start_packet_command.
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
+ * HANDLER is the interrupt handler to call when the command completes or
+ * there's data ready.
+ */
+static ide_startstop_t cdrom_transfer_packet_command(ide_drive_t *drive,
+ struct request *rq,
+ ide_handler_t *handler)
+ {
+ ide_hwif_t *hwif = drive->hwif;
+ int cmd_len;
+ @@ -566,8 +586,10 @@ static ide_startstop_t cdrom_transfer_packet_command (ide_drive_t *drive,
+ ide_startstop_t startstop;

+ if (info->cd_flags & IDE_CD_FLAG_DRQ_INTERRUPT) {
+ - /* Here we should have been called after receiving an interrupt
+ - from the device. DRQ should now be set. */
+ + /*
+ + * Here we should have been called after receiving an interrupt
+ + * from the device. DRQ should now be set.
+ + */

+ /* Check for errors. */
+ if (cdrom_decode_status(drive, DRQ_STAT, NULL))
+ @@ -601,10 +623,9 @@ static ide_startstop_t cdrom_transfer_packet_command (ide_drive_t *drive,
+ return ide_started;
+ }

+ /*****
+ + */
+ * Block read functions.
+ */
+ -
+ static void ide_cd_pad_transfer(ide_drive_t *drive, xfer_func_t *xf, int len)
+ {
+ while (len > 0) {
+ @@ -629,8 +650,8 @@ static void ide_cd_drain_data(ide_drive_t *drive, int nsects)
+ * and attempt to recover if there are problems. Returns 0 if everything's
+ * ok; nonzero if the request has been terminated.
+ */
+ -static
+ -int ide_cd_check_ireason(ide_drive_t *drive, int len, int ireason, int rw)
+ +static int ide_cd_check_ireason(ide_drive_t *drive, int len, int ireason,
+ + int rw)
+ {
+ /*
+ * ireason == 0: the drive wants to receive data from us
+ @@ -644,20 +665,21 @@ int ide_cd_check_ireason(ide_drive_t *drive, int len, int ireason, int rw)

+ /* Whoops... */
+ printk(KERN_ERR "%s: %s: wrong transfer direction!\n",
+ - drive->name, __FUNCTION__);
+ + drive->name, __func__);
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
xf = rw ? hwif->atapi_output_bytes : hwif->atapi_input_bytes;
ide_cd_pad_transfer(drive, xf, len);
} else if (rw == 0 && ireason == 1) {
- /* Some drives (ASUS) seem to tell us that status
- * info is available. just get it and ignore.
+ /*
+ * Some drives (ASUS) seem to tell us that status info is
+ * available. just get it and ignore.
*/
(void)ide_read_status(drive);
return 0;
} else {
/* Drive wants a command packet, or invalid ireason... */
printk(KERN_ERR "%s: %s: bad interrupt reason 0x%02x\n",
- drive->name, __FUNCTION__, ireason);
+ drive->name, __func__, ireason);
}

cdrom_end_request(drive, 0);
@@ -676,13 +698,13 @@ static int ide_cd_check_transfer_size(ide_drive_t *drive, int len)
return 0;

printk(KERN_ERR "%s: %s: Bad transfer size %d\n",
- drive->name, __FUNCTION__, len);
+ drive->name, __func__, len);

if (cd->cd_flags & IDE_CD_FLAG_LIMIT_NFRAMES)
- printk(KERN_ERR " This drive is not supported by "
- "this version of the driver\n");
+ printk(KERN_ERR "This drive is not supported by this version of"
+ " the driver\n");
else {
- printk(KERN_ERR " Trying to limit transfer sizes\n");
+ printk(KERN_ERR "Trying to limit transfer sizes\n");
cd->cd_flags |= IDE_CD_FLAG_LIMIT_NFRAMES;
}

@@ -692,10 +714,10 @@ static int ide_cd_check_transfer_size(ide_drive_t *drive, int len)
static ide_startstop_t cdrom_newpc_intr(ide_drive_t *);

/*
- * Routine to send a read/write packet command to the drive.
- * This is usually called directly from cdrom_start_{read,write}().
- * However, for drq_interrupt devices, it is called from an interrupt
- * when the drive is ready to accept the command.
+ * Routine to send a read/write packet command to the drive. This is usually
+ * called directly from cdrom_start_{read,write}(). However, for drq_interrupt
+ * devices, it is called from an interrupt when the drive is ready to accept the
+ * command.
*/
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
static ide_startstop_t cdrom_start_rw_cont(ide_drive_t *drive)
{
@@ -721,7 +743,7 @@ static ide_startstop_t cdrom_start_rw_cont(ide_drive_t *drive)
if (rq->current_nr_sectors !=
bio_cur_sectors(rq->bio)) {
printk(KERN_ERR "%s: %s: buffer botch (%u)\n",
- drive->name, __FUNCTION__,
+ drive->name, __func__,
rq->current_nr_sectors);
cdrom_end_request(drive, 0);
return ide_stopped;
@@ -745,7 +767,7 @@ static ide_startstop_t cdrom_start_rw_cont(ide_drive_t *drive)
#define IDECD_SEEK_TIMER (5 * WAIT_MIN_SLEEP) /* 100 ms */
#define IDECD_SEEK_TIMEOUT (2 * WAIT_CMD) /* 20 sec */

-static ide_startstop_t cdrom_seek_intr (ide_drive_t *drive)
+static ide_startstop_t cdrom_seek_intr(ide_drive_t *drive)
{
struct cdrom_info *info = drive->driver_data;
int stat;
@@ -757,19 +779,13 @@ static ide_startstop_t cdrom_seek_intr (ide_drive_t *drive)
info->cd_flags |= IDE_CD_FLAG SEEKING;

if (retry && time_after(jiffies, info->start_seek + IDECD_SEEK_TIMER)) {
- if (--retry == 0) {
- /*
- * this condition is far too common, to bother
- * users about it
- */
- /* printk("%s: disabled DSC seek overlap\n", drive->name);*/
+ if (--retry == 0)
drive->dsc_overlap = 0;
- }
}
return ide_stopped;
}

-static ide_startstop_t cdrom_start_seek_continuation (ide_drive_t *drive)
+static ide_startstop_t cdrom_start_seek_continuation(ide_drive_t *drive)
{
struct request *rq = HWGROUP(drive)->rq;
sector_t frame = rq->sector;
@@ -784,41 +800,45 @@ static ide_startstop_t cdrom_start_seek_continuation (ide_drive_t *drive)
return cdrom_transfer_packet_command(drive, rq, &cdrom_seek_intr);
}

-static ide_startstop_t cdrom_start_seek (ide_drive_t *drive, unsigned int block)
+static ide_startstop_t cdrom_start_seek(ide_drive_t *drive, unsigned int block)
{
struct cdrom_info *info = drive->driver_data;
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
info->dma = 0;
info->start_seek = jiffies;
- return cdrom_start_packet_command(drive, 0, cdrom_start_seek_continuation);
+ return cdrom_start_packet_command(drive, 0,
+ cdrom_start_seek_continuation);
}

-/* Fix up a possibly partially-processed request so that we can
- start it over entirely, or even put it back on the request queue. */
-static void restore_request (struct request *rq)
+/*
+ * Fix up a possibly partially-processed request so that we can start it over
+ * entirely, or even put it back on the request queue.
+ */
+static void restore_request(struct request *rq)
{
if (rq->buffer != bio_data(rq->bio)) {
- sector_t n = (rq->buffer - (char *) bio_data(rq->bio)) / SECTOR_SIZE;
+ sector_t n = (rq->buffer - (char *) bio_data(rq->bio)) /
+ SECTOR_SIZE;

rq->buffer = bio_data(rq->bio);
rq->nr_sectors += n;
rq->sector -= n;
}
- rq->hard_cur_sectors = rq->current_nr_sectors = bio_cur_sectors(rq->bio);
+ rq->current_nr_sectors = bio_cur_sectors(rq->bio);
+ rq->hard_cur_sectors = rq->current_nr_sectors;
rq->hard_nr_sectors = rq->nr_sectors;
rq->hard_sector = rq->sector;
rq->q->prep_rq_fn(rq->q, rq);
}

-/**
+/*
+ * Execute all other packet commands.
+ */
-
-static void ide_cd_request_sense_fixup(struct request *rq)
{
-/*
- * Some of the trailing request sense fields are optional,
- * and some drives don't send them. Sigh.
+ * Some of the trailing request sense fields are optional, and some
+ * drives don't send them. Sigh.
+ */
if (rq->cmd[0] == GPCMD_REQUEST_SENSE &&
rq->data_len > 0 && rq->data_len <= 5)
@@ -846,21 +866,25 @@ int ide_cd_queue_pc(ide_drive_t *drive, struct request *rq)
error = ide_do_drive_cmd(drive, rq, ide_wait);
time = jiffies - time;
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
- /* FIXME: we should probably abort/retry or something
- * in case of failure */
+ /* FIXME: we should probably abort/retry or something in case of
+ * failure */
if (rq->cmd_flags & REQ_FAILED) {
- /* The request failed. Retry if it was due to a unit
- attention status
- (usually means media was changed). */
+ /*
+ * The request failed. Retry if it was due to a unit
+ * attention status (usually means media was changed).
+ */
struct request_sense *reqbuf = rq->sense;

if (reqbuf->sense_key == UNIT_ATTENTION)
cdrom_saw_media_change(drive);
else if (reqbuf->sense_key == NOT_READY &&
- reqbuf->asc == 4 && reqbuf->ascq != 4) {
- /* The drive is in the process of loading
- a disk. Retry, but wait a little to give
- the drive time to complete the load. */
+ reqbuf->asc == 4 &&
+ reqbuf->ascq != 4) {
+ /*
+ * The drive is in the process of loading a
+ * disk. Retry, but wait a little to give
+ * the drive time to complete the load.
+ */
ssleep(2);
} else {
/* Otherwise, don't retry. */
@@ -877,10 +901,10 @@ int ide_cd_queue_pc(ide_drive_t *drive, struct request *rq)
}

/*
- * Called from blk_end_request_callback() after the data of the request
- * is completed and before the request is completed.
- * By returning value '1', blk_end_request_callback() returns immediately
- * without completing the request.
+ * Called from blk_end_request_callback() after the data of the request is
+ * completed and before the request is completed. By returning value '1',
+ * blk_end_request_callback() returns immediately without completing the
+ * request.
*/
static int cdrom_newpc_intr_dummy_cb(struct request *rq)
{
@@ -914,9 +938,7 @@ static ide_startstop_t cdrom_newpc_intr(ide_drive_t *drive)
if (cdrom_decode_status(drive, 0, &stat))
return ide_stopped;
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
- /*
- * using dma, transfer is complete now
- */
+ /* using dma, transfer is complete now */
if (dma) {
if (dma_error)
return ide_error(drive, "dma error", stat);
@@ -927,9 +949,7 @@ static ide_startstop_t cdrom_newpc_intr(ide_drive_t *drive)
goto end_request;
}

- /*
- * ok we fall to pio :/
- */
+ /* ok we fall to pio :/ */
ireason = hwif->INB(hwif->io_ports[IDE_IREASON_OFFSET]) & 0x3;
lowcyl = hwif->INB(hwif->io_ports[IDE_BCOUNTL_OFFSET]);
highcyl = hwif->INB(hwif->io_ports[IDE_BCOUNTH_OFFSET]);
@@ -940,9 +960,7 @@ static ide_startstop_t cdrom_newpc_intr(ide_drive_t *drive)
if (thislen > len)
thislen = len;

- /*
- * If DRQ is clear, the command has completed.
- */
+ /* If DRQ is clear, the command has completed. */
if ((stat & DRQ_STAT) == 0) {
if (blk_fs_request(rq)) {
/*
@@ -953,7 +971,7 @@ static ide_startstop_t cdrom_newpc_intr(ide_drive_t *drive)
if (rq->current_nr_sectors > 0) {
printk(KERN_ERR "%s: %s: data underrun "
"%d blocks)\n",
- drive->name, __FUNCTION__,
+ drive->name, __func__,
rq->current_nr_sectors);
if (!write)
rq->cmd_flags |= REQ_FAILED;
@@ -969,9 +987,7 @@ static ide_startstop_t cdrom_newpc_intr(ide_drive_t *drive)
goto end_request;
}

- /*
- * check which way to transfer data
- */
+ /* check which way to transfer data */
if (blk_fs_request(rq) || blk_pc_request(rq)) {
if (ide_cd_check_ireason(drive, len, ireason, write))
return ide_stopped;
@@ -985,8 +1001,8 @@ static ide_startstop_t cdrom_newpc_intr(ide_drive_t *drive)
}
}
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
/*
- * First, figure out if we need to bit-bucket
- * any of the leading sectors.
+ * First, figure out if we need to bit-bucket any of the
+ * leading sectors.
*/
nskip = min_t(int, rq->current_nr_sectors
- bio_cur_sectors(rq->bio),
@@ -1010,20 +1026,16 @@ static ide_startstop_t cdrom_newpc_intr(ide_drive_t *drive)
printk(KERN_ERR "%s: %s: The drive "
"appears confused (ireason = 0x%02x). "
"Trying to recover by ending request.\n",
- drive->name, __FUNCTION__, ireason);
+ drive->name, __func__, ireason);
goto end_request;
}

- /*
- * transfer data
- */
+ /* transfer data */
while (thislen > 0) {
u8 *ptr = blk_fs_request(rq) ? NULL : rq->data;
int blen = rq->data_len;

- /*
- * bio backed?
- */
+ /* bio backed? */
if (rq->bio) {
if (blk_fs_request(rq)) {
ptr = rq->buffer;
@@ -1038,7 +1050,8 @@ static ide_startstop_t cdrom_newpc_intr(ide_drive_t *drive)
if (blk_fs_request(rq) && !write)
/*
* If the buffers are full, pipe the rest into
- * oblivion. */
+ * oblivion.
+ */
ide_cd_drain_data(drive, thislen >> 9);
else {
printk(KERN_ERR "%s: confused, missing data\n",
@@ -1086,9 +1099,7 @@ static ide_startstop_t cdrom_newpc_intr(ide_drive_t *drive)
if (write && blk_sense_request(rq))
rq->sense_len += thislen;

- /*
- * pad, if necessary
- */
+ /* pad, if necessary */
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
if (!blk_fs_request(rq) && len > 0)
ide_cd_pad_transfer(drive, xferfunc, len);

@@ -1132,9 +1143,7 @@ static ide_startstop_t cdrom_start_rw(ide_drive_t *drive, struct request *rq)
queue_hardsect_size(drive->queue) >> SECTOR_BITS;

if (write) {
- /*
- * disk has become write protected
- */
+ /* disk has become write protected */
if (cd->disk->policy) {
cdrom_end_request(drive, 0);
return ide_stopped;
@@ -1147,9 +1156,7 @@ static ide_startstop_t cdrom_start_rw(ide_drive_t *drive, struct request *rq)
restore_request(rq);
}

- /*
- * use DMA, if possible / writes *must* be hardware frame aligned
- */
+ /* use DMA, if possible / writes *must* be hardware frame aligned */
if ((rq->nr_sectors & (sectors_per_frame - 1)) ||
(rq->sector & (sectors_per_frame - 1))) {
if (write) {
@@ -1188,12 +1195,11 @@ static ide_startstop_t cdrom_do_block_pc(ide_drive_t *drive, struct request
*rq)

info->dma = 0;

- /*
- * sg request
- */
+ /* sg request */
if (rq->bio) {
int mask = drive->queue->dma_alignment;
- unsigned long addr = (unsigned long) page_address(bio_page(rq->bio));
+ unsigned long addr = (unsigned long)
+ page_address(bio_page(rq->bio));

info->dma = drive->using_dma;

@@ -1208,14 +1214,13 @@ static ide_startstop_t cdrom_do_block_pc(ide_drive_t *drive, struct request
*rq)
}

/* Start sending the command to the drive. */
- return cdrom_start_packet_command(drive, rq->data_len, cdrom_do_newpc_cont);
+ return cdrom_start_packet_command(drive, rq->data_len,
+ cdrom_do_newpc_cont);
}
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
_/*****
- * cdrom driver request routine.
- */
-static ide_startstop_t
-ide_do_rw_cdrom (ide_drive_t *drive, struct request *rq, sector_t block)
+/* cdrom driver request routine. */
+static ide_startstop_t ide_do_rw_cdrom(ide_drive_t *drive, struct request *rq,
+ sector_t block)
{
ide_startstop_t action;
struct cdrom_info *info = drive->driver_data;
@@ -1227,26 +1232,30 @@ ide_do_rw_cdrom (ide_drive_t *drive, struct request *rq, sector_t block)

if ((stat & SEEK_STAT) != SEEK_STAT) {
if (elapsed < IDECD_SEEK_TIMEOUT) {
- ide_stall_queue(drive, IDECD_SEEK_TIMER);
+ ide_stall_queue(drive,
+ IDECD_SEEK_TIMER);
return ide_stopped;
}
- printk (KERN_ERR "%s: DSC timeout\n", drive->name);
+ printk(KERN_ERR "%s: DSC timeout\n",
+ drive->name);
}
info->cd_flags &= ~IDE_CD_FLAG_SEEKING;
}
- if ((rq_data_dir(rq) == READ) && IDE_LARGE_SEEK(info->last_block, block,
IDECD_SEEK_THRESHOLD) && drive->dsc_overlap) {
+ if ((rq_data_dir(rq) == READ) &&
+ IDE_LARGE_SEEK(info->last_block, block,
+ IDECD_SEEK_THRESHOLD) &&
+ drive->dsc_overlap)
action = cdrom_start_seek(drive, block);
- } else
+ else
action = cdrom_start_rw(drive, rq);
+
info->last_block = block;
return action;
} else if (blk_sense_request(rq) || blk_pc_request(rq) ||
rq->cmd_type == REQ_TYPE_ATA_PC) {
return cdrom_do_block_pc(drive, rq);
} else if (blk_special_request(rq)) {
- /*
- * right now this can only be a reset...
- */
+ /* right now this can only be a reset... */
cdrom_end_request(drive, 1);
return ide_stopped;
}
}
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
@@ -1256,20 +1265,16 @@ ide_do_rw_cdrom (ide_drive_t *drive, struct request *rq, sector_t block)
return ide_stopped;
}

-
-
-/******
+/*
* Ioctl handling.
*
* Routines which queue packet commands take as a final argument a pointer
- * to a request_sense struct. If execution of the command results
- * in an error with a CHECK CONDITION status, this structure will be filled
- * with the results of the subsequent request sense command. The pointer
- * can also be NULL, in which case no sense information is returned.
+ * to a request_sense struct. If execution of the command results in an error
+ * with a CHECK CONDITION status, this structure will be filled with the results
+ * of the subsequent request sense command. The pointer can also be NULL, in
+ * which case no sense information is returned.
*/
-
-static
-void msf_from_bcd (struct atapi_msf *msf)
+static void msf_from_bcd(struct atapi_msf *msf)
{
msf->minute = BCD2BIN(msf->minute);
msf->second = BCD2BIN(msf->second);
@@ -1369,14 +1374,17 @@ int ide_cd_read_toc(ide_drive_t *drive, struct request_sense *sense)
/* Try to allocate space. */
toc = kmalloc(sizeof(struct atapi_toc), GFP_KERNEL);
if (toc == NULL) {
- printk (KERN_ERR "%s: No cdrom TOC buffer!\n", drive->name);
+ printk(KERN_ERR "%s: Cannot alloc cdrom TOC buffer!\n",
+ drive->name);
return -ENOMEM;
}
info->toc = toc;
}

- /* Check to see if the existing data is still valid.
- If it is, just return. */
+ /*
+ * Check to see if the existing data is still valid. If it is, just
+ * return.
+ */
(void) cdrom_check_status(drive, sense);

if (info->cd_flags & IDE_CD_FLAG_TOC_VALID)
@@ -1420,15 +1428,19 @@ int ide_cd_read_toc(ide_drive_t *drive, struct request_sense *sense)
sizeof(struct atapi_toc_entry), sense);
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
if (stat && toc->hdr.first_track > 1) {
- /* Cds with CDI tracks only don't have any TOC entries,
- despite of this the returned values are
- first_track == last_track = number of CDI tracks + 1,
- so that this case is indistinguishable from the same
- layout plus an additional audio track.
- If we get an error for the regular case, we assume
- a CDI without additional audio tracks. In this case
- the readable TOC is empty (CDI tracks are not included)
- and only holds the Leadout entry. Heiko EiÃ?feldt */
+ /*
+ * Cds with CDI tracks only don't have any TOC entries; despite
+ * of this, the returned values are first_track == last_track =
+ * number of CDI tracks + 1, so that this case is
+ * indistinguishable from the same layout plus an additional
+ * audio track.
+ *
+ * If we get an error for the regular case, we assume a CDI
+ * without additional audio tracks. In this case the readable
+ * TOC is empty (CDI tracks are not included) and only holds the
+ * Leadout entry.
+ *
+ * Heiko EiÃ?feldt */
ntracks = 0;
stat = cdrom_read_tocentry(drive, CDROM_LEADOUT, 1, 0,
(char *)&toc->hdr,
@@ -1464,9 +1476,9 @@ int ide_cd_read_toc(ide_drive_t *drive, struct request_sense *sense)
toc->ent[i].track = BCD2BIN(toc->ent[i].track);
msf_from_bcd(&toc->ent[i].addr.msf);
}
- toc->ent[i].addr.lba = msf_to_lba (toc->ent[i].addr.msf.minute,
- toc->ent[i].addr.msf.second,
- toc->ent[i].addr.msf.frame);
+ toc->ent[i].addr.lba = msf_to_lba(toc->ent[i].addr.msf.minute,
+ toc->ent[i].addr.msf.second,
+ toc->ent[i].addr.msf.frame);
}

/* Read the multisession information. */
@@ -1479,7 +1491,8 @@ int ide_cd_read_toc(ide_drive_t *drive, struct request_sense *sense)

toc->last_session_lba = be32_to_cpu(ms_tmp.ent.addr.lba);
} else {
- ms_tmp.hdr.first_track = ms_tmp.hdr.last_track = CDROM_LEADOUT;
+ ms_tmp.hdr.first_track = CDROM_LEADOUT;
+ ms_tmp.hdr.last_track = CDROM_LEADOUT;
toc->last_session_lba = msf_to_lba(0, 2, 0); /* 0m 2s 0f */
}

@@ -1490,9 +1503,9 @@ int ide_cd_read_toc(ide_drive_t *drive, struct request_sense *sense)
if (stat)
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
return stat;

- msf_from_bcd (&ms_tmp.ent.addr.msf);
+ msf_from_bcd(&ms_tmp.ent.addr.msf);
toc->last_session_lba = msf_to_lba(ms_tmp.ent.addr.msf.minute,
- ms_tmp.ent.addr.msf.second,
+ ms_tmp.ent.addr.msf.second,
ms_tmp.ent.addr.msf.frame);
}

@@ -1523,7 +1536,8 @@ int ide_cdrom_get_capabilities(ide_drive_t *drive, u8 *buf)
size -= ATAPI_CAPABILITIES_PAGE_PAD_SIZE;

init_cdrom_command(&cgc, buf, size, CGC_DATA_UNKNOWN);
- do { /* we seem to get stat=0x01,err=0x00 the first time (??) */
+ do {
+ /* we seem to get stat=0x01,err=0x00 the first time (??) */
stat = cdrom_mode_sense(cdi, &cgc, GPMODE_CAPABILITIES_PAGE, 0);
if (!stat)
break;
@@ -1574,7 +1588,7 @@ static struct cdrom_device_ops ide_cdrom_dops = {
.generic_packet = ide_cdrom_packet,
};

-static int ide_cdrom_register (ide_drive_t *drive, int nslots)
+static int ide_cdrom_register(ide_drive_t *drive, int nslots)
{
struct cdrom_info *info = drive->driver_data;
struct cdrom_device_info *devinfo = &info->devinfo;
@@ -1593,7 +1607,7 @@ static int ide_cdrom_register (ide_drive_t *drive, int nslots)
}

static
-int ide_cdrom_probe_capabilities (ide_drive_t *drive)
+int ide_cdrom_probe_capabilities(ide_drive_t *drive)
{
struct cdrom_info *cd = drive->driver_data;
struct cdrom_device_info *cdi = &cd->devinfo;
@@ -1607,7 +1621,8 @@ int ide_cdrom_probe_capabilities (ide_drive_t *drive)

if (drive->media == ide_optical) {
cdi->mask &= ~(CDC_MO_DRIVE | CDC_RAM);
- printk(KERN_ERR "%s: ATAPI magneto-optical drive\n", drive->name);
+ printk(KERN_ERR "%s: ATAPI magneto-optical drive\n",
+ drive->name);
return nslots;
}

@@ -1618,11 +1633,10 @@ int ide_cdrom_probe_capabilities (ide_drive_t *drive)
}
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
/*
- * we have to cheat a little here. the packet will eventually
- * be queued with ide_cdrom_packet(), which extracts the
- * drive from cdi->handle. Since this device hasn't been
- * registered with the Uniform layer yet, it can't do this.
- * Same goes for cdi->ops.
+ * we have to cheat a little here. the packet will eventually be queued
+ * with ide_cdrom_packet(), which extracts the drive from cdi->handle.
+ * Since this device hasn't been registered with the Uniform layer yet,
+ * it can't do this. Same goes for cdi->ops.
*/
cdi->handle = drive;
cdi->ops = &ide_cdrom_dops;
@@ -1694,7 +1708,8 @@ int ide_cdrom_probe_capabilities (ide_drive_t *drive)
#ifdef CONFIG_IDE_PROC_FS
static void ide_cdrom_add_settings(ide_drive_t *drive)
{
- ide_add_setting(drive, "dsc_overlap", SETTING_RW, TYPE_BYTE, 0, 1, 1, 1, &drive->dsc_overlap,
NULL);
+ ide_add_setting(drive, "dsc_overlap", SETTING_RW, TYPE_BYTE, 0, 1, 1, 1,
+ &drive->dsc_overlap, NULL);
}
#else
static inline void ide_cdrom_add_settings(ide_drive_t *drive) { ; }
@@ -1716,17 +1731,13 @@ static int ide_cdrom_prep_fs(struct request_queue *q, struct request *rq)
else
rq->cmd[0] = GPCMD_WRITE_10;

- /*
- * fill in lba
- */
+ /* fill in lba */
rq->cmd[2] = (block >> 24) & 0xff;
rq->cmd[3] = (block >> 16) & 0xff;
rq->cmd[4] = (block >> 8) & 0xff;
rq->cmd[5] = block & 0xff;

- /*
- * and transfer length
- */
+ /* and transfer length */
rq->cmd[7] = (blocks >> 8) & 0xff;
rq->cmd[8] = blocks & 0xff;
rq->cmd_len = 10;
@@ -1741,9 +1752,7 @@ static int ide_cdrom_prep_pc(struct request *rq)
{
u8 *c = rq->cmd;

- /*
- * Transform 6-byte read/write commands to the 10-byte version
- */
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
+ /* Transform 6-byte read/write commands to the 10-byte version */
if (c[0] == READ_6 || c[0] == WRITE_6) {
c[8] = c[4];
c[5] = c[3];
@@ -1765,7 +1774,7 @@ static int ide_cdrom_prep_pc(struct request *rq)
rq->errors = ILLEGAL_REQUEST;
return BLKPREP_KILL;
}
-
+
return BLKPREP_OK;
}

@@ -1842,8 +1851,7 @@ static unsigned int ide_cd_flags(struct hd_driveid *id)
return 0;
}

-static
-int ide_cdrom_setup (ide_drive_t *drive)
+static int ide_cdrom_setup(ide_drive_t *drive)
{
struct cdrom_info *cd = drive->driver_data;
struct cdrom_device_info *cdi = &cd->devinfo;
@@ -1872,13 +1880,12 @@ int ide_cdrom_setup (ide_drive_t *drive)
id->fw_rev[4] == '1' && id->fw_rev[6] <= '2')
cd->cd_flags |= IDE_CD_FLAG_TOCTRACKS_AS_BCD;
else if (cd->cd_flags & IDE_CD_FLAG_SANYO_3CD)
- cdi->sanyo_slot = 3; /* 3 => use CD in slot 0 */
+ /* 3 => use CD in slot 0 */
+ cdi->sanyo_slot = 3;

- nslots = ide_cdrom_probe_capabilities (drive);
+ nslots = ide_cdrom_probe_capabilities(drive);

- /*
- * set correct block size
- */
+ /* set correct block size */
blk_queue_hardsect_size(drive->queue, CD_FRAMESIZE);

if (drive->autotune == IDE_TUNE_DEFAULT ||
@@ -1886,7 +1893,8 @@ int ide_cdrom_setup (ide_drive_t *drive)
drive->dsc_overlap = (drive->next != drive);

if (ide_cdrom_register(drive, nslots) {
- printk (KERN_ERR "%s: ide_cdrom_setup failed to register device with the cdrom driver.\n",
drive->name);
+ printk(KERN_ERR "%s: %s failed to register device with the"
+ " cdrom driver.\n", drive->name, __func__);
cd->devinfo.handle = NULL;
return 1;
}
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
}
@@ -1914,8 +1922,8 @@ static void ide_cd_release(struct kref *kref)

kfree(info->toc);
if (devinfo->handle == drive && unregister_cdrom(devinfo))
- printk(KERN_ERR "%s: %s failed to unregister device from the cdrom "
- "driver.\n", __FUNCTION__, drive->name);
+ printk(KERN_ERR "%s: %s failed to unregister device from the "
+ "cdrom driver.\n", drive->name, __func__);
drive->dsc_overlap = 0;
drive->driver_data = NULL;
blk_queue_prep_rq(drive->queue, NULL);
@@ -1973,13 +1981,14 @@ static ide_driver_t ide_cdrom_driver = {
#endif
};

-static int idecd_open(struct inode * inode, struct file * file)
+static int idecd_open(struct inode *inode, struct file *file)
{
struct gendisk *disk = inode->i_bdev->bd_disk;
struct cdrom_info *info;
int rc = -ENOMEM;

- if (!(info = ide_cd_get(disk)))
+ info = ide_cd_get(disk);
+ if (!info)
return -ENXIO;

rc = cdrom_open(&info->devinfo, inode, file);
@@ -1990,12 +1999,12 @@ static int idecd_open(struct inode * inode, struct file * file)
return rc;
}

-static int idecd_release(struct inode * inode, struct file * file)
+static int idecd_release(struct inode *inode, struct file *file)
{
struct gendisk *disk = inode->i_bdev->bd_disk;
struct cdrom_info *info = ide_cd_g(disk);

- cdrom_release (&info->devinfo, file);
+ cdrom_release(&info->devinfo, file);

ide_cd_put(info);

@@ -2027,7 +2036,7 @@ static int idecd_get_spindown(struct cdrom_device_info *cdi, unsigned long arg)
struct packet_command cgc;
char buffer[16];
int stat;
- char spindown;
+ char spindown;
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
init_cdrom_command(&cgc, buffer, sizeof(buffer), CGC_DATA_UNKNOWN);

@@ -2036,12 +2045,12 @@ static int idecd_get_spindown(struct cdrom_device_info *cdi, unsigned long
arg)
return stat;

spindown = buffer[11] & 0x0f;
- if (copy_to_user((void __user *)arg, &spindown, sizeof (char)))
+ if (copy_to_user((void __user *)arg, &spindown, sizeof(char)))
return -EFAULT;
return 0;
}

-static int idecd_ioctl (struct inode *inode, struct file *file,
+static int idecd_ioctl(struct inode *inode, struct file *file,
unsigned int cmd, unsigned long arg)
{
struct block_device *bdev = inode->i_bdev;
@@ -2049,13 +2058,13 @@ static int idecd_ioctl (struct inode *inode, struct file *file,
int err;

switch (cmd) {
- case CDRROMSETSPINDOWN:
+ case CDRROMSETSPINDOWN:
return idecd_set_spindown(&info->devinfo, arg);
- case CDRROMGETSPINDOWN:
+ case CDRROMGETSPINDOWN:
return idecd_get_spindown(&info->devinfo, arg);
default:
break;
- }
+ }

err = generic_ide_ioctl(info->drive, file, bdev, cmd, arg);
if (err == -EINVAL)
@@ -2090,7 +2099,7 @@ static struct block_device_operations idecd_ops = {
};

/* options */
-static char *ignore = NULL;
+static char *ignore;

module_param(ignore, charp, 0400);
MODULE_DESCRIPTION("ATAPI CD-ROM Driver");
@@ -2110,17 +2119,20 @@ static int ide_cd_probe(ide_drive_t *drive)
/* skip drives that we were told to ignore */
if (ignore != NULL) {
if (strstr(ignore, drive->name)) {
- printk(KERN_INFO "ide-cd: ignoring drive %s\n", drive->name);
+ printk(KERN_INFO "ide-cd: ignoring drive %s\n",
+ drive->name);
}
```

[PATCH] ide-cd: fix some codestyle and most of the checkpatch.pl issues

```
goto failed;
}
}
if (drive->scsi) {
- printk(KERN_INFO "ide-cd: passing drive %s to ide-scsi emulation.\n", drive->name);
+ printk(KERN_INFO "ide-cd: passing drive %s to ide-scsi "
+ "emulation.\n", drive->name);
goto failed;
}
info = kzalloc(sizeof(struct cdrom_info), GFP_KERNEL);
if (info == NULL) {
- printk(KERN_ERR "%s: Can't allocate a cdrom structure\n", drive->name);
+ printk(KERN_ERR "%s: Can't allocate a cdrom structure\n",
+ drive->name);
goto failed;
}
}
```

--

1.5.4.1

--

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>