

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-01/msg11045.html>

- *From:* Borislav Petkov <petkovbb@xxxxxxxx>
 - *Date:* Sun, 27 Jan 2008 10:48:20 +0100
-

From: Borislav Petkov <bbpetkov@xxxxxxxx>

Which is, in particular:

- reorganize #defines at the top
- whitespace cleanup
- remove the "!= NULL" chunk from the pointer validity tests since the short version "if (ptr)" is more intuitive
- fix comments
- remove unused "length" arg from idetape_create_read_buffer_cmd()
- shorten some vars in order to fit lines into 80 columns.
- include proper headers
- finally, bump driver version

thus decreasing checkpatch errors to 0.

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

drivers/ide/ide-tape.c | 1525 ++++++-----
1 files changed, 758 insertions(+), 767 deletions(-)

diff --git a/drivers/ide/ide-tape.c b/drivers/ide/ide-tape.c

index 424879d..f51712c 100644

--- a/drivers/ide/ide-tape.c

+++ b/drivers/ide/ide-tape.c

@@ -15,7 +15,7 @@

* Documentation/ide/ChangeLog.ide-tape.1995-2002

*/

--- #define IDETAPE_VERSION "1.19"

+++ #define IDETAPE_VERSION "1.20"

#include <linux/module.h>

#include <linux/types.h>

@@ -39,34 +39,12 @@

#include <scsi/scsi.h>

#include <asm/byteorder.h>

--- #include <asm/irq.h>

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```

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

-/****** Tunable parameters *****/
-
-
-/*
- * Pipelined mode parameters.
- *
- * We try to use the minimum number of stages which is enough to
- * keep the tape constantly streaming. To accomplish that, we implement
- * a feedback loop around the maximum number of stages:
- *
- * We start from MIN maximum stages (we will not even use MIN stages
- * if we don't need them), increment it by RATE*(MAX-MIN)
- * whenever we sense that the pipeline is empty, until we reach
- * the optimum value or until we reach MAX.
- *
- * Setting the following parameter to 0 is illegal: the pipelined mode
- * cannot be disabled (idetape_calculate_speeds() divides by
- * tape->max_stages.)
- */
-#define IDETAPE_MIN_PIPELINE_STAGES 1
-#define IDETAPE_MAX_PIPELINE_STAGES 400
-#define IDETAPE_INCREASE_STAGES_RATE 20

enum {
DBG_ERR = (1 << 0),
@@ -90,17 +68,34 @@ enum {
#endif

/*
- * After each failed packet command we issue a request sense command
- * and retry the packet command IDETAPE_MAX_PC_RETRIES times.
+ * Tunable parameters:
+ *
+ * Pipelined mode parameters.
+ *
+ * We try to use the minimum number of stages which is enough to keep the tape
+ * constantly streaming. To accomplish that, we implement a feedback loop around
+ * the maximum number of stages:
+ *
+ * We start from MIN maximum stages (we will not even use MIN stages if we don't
+ * need them), increment it by RATE*(MAX-MIN) whenever we sense that the
+ * pipeline is empty, until we reach the optimum value or until we reach MAX.
+ */

```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
- * Setting IDETAPE_MAX_PC_RETRIES to 0 will disable retries.
+ * Setting the following parameter to 0 is illegal: the pipelined mode cannot be
+ * disabled (idetape_calculate_speeds() divides by tape->max_stages.)
+ */
+#define IDETAPE_MIN_PPL_STAGES 1
+#define IDETAPE_MAX_PPL_STAGES 400
+#define IDETAPE_INCREASE_STAGES_RATE 20
+
+/*
+ * After each failed packet command we issue a request sense command and retry
+ * the packet command IDETAPE_MAX_PC_RETRIES times. Set it to 0 to disable.
+ */
#define IDETAPE_MAX_PC_RETRIES 3

/*
- * With each packet command, we allocate a buffer of
- * IDETAPE_PC_BUFFER_SIZE bytes. This is used for several packet
- * commands (Not for READ/WRITE commands).
+ * With each packet command, we allocate a buffer of IDETAPE_PC_BUFFER_SIZE
+ * bytes. (Used for several packet commands but not for READ/WRITE commands).
+ */
#define IDETAPE_PC_BUFFER_SIZE 256

@@ -112,48 +107,40 @@ enum {
#define IDETAPE_WAIT_CMD (900*HZ)

/*
- * The following parameter is used to select the point in the internal
- * tape fifo in which we will start to refill the buffer. Decreasing
- * the following parameter will improve the system's latency and
- * interactive response, while using a high value might improve system
- * throughput.
+ * The following parameter is used to select the point in the internal tape fifo
+ * in which we will start to refill the buffer. Decreasing the following
+ * parameter will improve the system's latency and interactive response, while
+ * using a high value might improve system throughput.
+ */
-#define IDETAPE_FIFO_THRESHOLD 2
+#define IDETAPE_FIFO_THRESHOLD 2

/*
- * DSC polling parameters.
- *
- * Polling for DSC (a single bit in the status register) is a very
- * important function in ide-tape. There are two cases in which we
- * poll for DSC:
+ * DSC polling parameters.
+ *
+ * 1. Before a read/write packet command, to ensure that we
+ * can transfer data from/to the tape's data buffers, without
+ * causing an actual media access. In case the tape is not
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
- * ready yet, we take out our request from the device
- * request queue, so that ide.c will service requests from
- * the other device on the same interface meanwhile.
+ * Polling for DSC (a single bit in the status register) is a very important
+ * function in ide-tape. There are two cases in which we poll for DSC:
*
- * 2. After the successful initialization of a "media access
- * packet command", which is a command which can take a long
- * time to complete (it can be several seconds or even an hour).
+ * 1. Before a read/write packet command, to ensure that we can transfer data
+ * from/to the tape's data buffers, without causing an actual media access.
+ * In case the tape is not ready yet, we take out our request from the device
+ * request queue, so that ide.c could service requests from the other device
+ * on the same interface in the meantime.
*
- * Again, we postpone our request in the middle to free the bus
- * for the other device. The polling frequency here should be
- * lower than the read/write frequency since those media access
- * commands are slow. We start from a "fast" frequency –
- * IDETAPE_DSC_MA_FAST (one second), and if we don't receive DSC
- * after IDETAPE_DSC_MA_THRESHOLD (5 minutes), we switch it to a
- * lower frequency – IDETAPE_DSC_MA_SLOW (1 minute).
+ * 2. After the successful initialization of a "media access packet command",
+ * which is a command that can take a long time to complete (the interval can
+ * range from several seconds to even an hour).
*
- * We also set a timeout for the timer, in case something goes wrong.
- * The timeout should be longer then the maximum execution time of a
- * tape operation.
- */
-
-/*
- * DSC timings.
+ * Again, we postpone our request in the middle to free the bus for the other
+ * device. The polling frequency here should be lower than the read/write
+ * frequency since those media access commands are slow. We start from a "fast"
+ * frequency – IDETAPE_DSC_MA_FAST (one second), and if we don't receive DSC
+ * after IDETAPE_DSC_MA_THRESHOLD (5 minutes), we switch it to a lower frequency
+ * – IDETAPE_DSC_MA_SLOW (1 minute). We also set a timeout for the timer, in
+ * case something goes wrong. The timeout should be longer then the maximum
+ * execution time of a tape operation.
*/
+
+/* DSC timings. */
#define IDETAPE_DSC_RW_MIN 5*HZ/100 /* 50 msec */
#define IDETAPE_DSC_RW_MAX 40*HZ/100 /* 400 msec */
#define IDETAPE_DSC_RW_TIMEOUT 2*60*HZ /* 2 minutes */
@@ -162,17 +149,12 @@ enum {
#define IDETAPE_DSC_MA_SLOW 30*HZ /* 30 seconds */
#define IDETAPE_DSC_MA_TIMEOUT 2*60*60*HZ /* 2 hours */
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
_/***** End of tunable parameters *****/
+/* End of tunable parameters */

-/*
- * Read/Write error simulation
- */
+/* Read/Write error simulation */
#define SIMULATE_ERRORS 0

-/*
- * For general magnetic tape device compatibility.
- */
-
+/* For general magnetic tape device compatibility. */
/* tape directions */
typedef enum {
  idetape_dir_none,
  @@ -223,9 +205,7 @@ enum {
  PC_FL_WRITING = (1 << 5),
};

-/*
- * A pipeline stage.
- */
+/* A pipeline stage. */
typedef struct idetape_stage_s {
  struct request rq; /* The corresponding request */
  struct idetape_bh *bh; /* The data buffers */
  @@ -320,7 +300,7 @@ typedef struct ide_tape_obj {
  /* active data request */
  struct request *act_data_rq;
  /* Data buffer size chosen based on the tape's recommendation */
  - int stage_size;
  + int stage_sz;
  idetape_stage_t *merge_stage;
  int merge_stage_sz;
  struct idetape_bh *bh;
  @@ -418,7 +398,7 @@ typedef struct ide_tape_obj {
  /* restart speed control req */
  int rs_speed_ctl_rq;

  - u32 debug_level;
  + u32 debug_level;
  } idetape_tape_t;

static DEFINE_MUTEX(idetape_ref_mutex);
@@ -451,9 +431,7 @@ static void ide_tape_put(struct ide_tape_obj *tape)
mutex_unlock(&idetape_ref_mutex);
}

-/*
```

```
- * Tape door status
- */
+/* Tape door status */
#define DOOR_UNLOCKED 0
#define DOOR_LOCKED 1
#define DOOR_EXPLICITLY_LOCKED 2
@@ -479,32 +457,24 @@ enum {
IDETAPE_FL_MEDIUM_PRESENT = (1 << 8),
};

-/*
- * Some defines for the READ BUFFER command
- */
+/* define for the READ BUFFER command */
#define IDETAPE_RETRIEVE_FAULTY_BLOCK 6

-/*
- * Some defines for the SPACE command
- */
+/* defines for the SPACE command */
#define IDETAPE_SPACE_OVER_FILEMARK 1
#define IDETAPE_SPACE_TO_EOD 3

-/*
- * Some defines for the LOAD UNLOAD command
- */
+/* defines for the LOAD UNLOAD command */
#define IDETAPE_LU_LOAD_MASK 1
#define IDETAPE_LU_RETENSION_MASK 2
#define IDETAPE_LU_EOT_MASK 4

/*
- * Special requests for our block device strategy routine.
+ * Special requests for our block device strategy routine.
*
- * In order to service a character device command, we add special
- * requests to the tail of our block device request queue and wait
- * for their completion.
+ * In order to service a character device command, we add special requests to
+ * the tail of our block device request queue and wait for their completion.
*/
-
enum {
REQ_IDETAPE_PC1 = (1 << 0), /* packet command (first stage) */
REQ_IDETAPE_PC2 = (1 << 1), /* packet command (second stage) */
@@ -513,19 +483,16 @@ enum {
REQ_IDETAPE_READ_BUFFER = (1 << 4),
};

-/*
- * Error codes which are returned in rq->errors to the higher part
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
- * of the driver.
- */
+/* Error codes returned in rq->errors to the higher part of the driver. */
#define IDETAPE_ERROR_GENERAL 101
#define IDETAPE_ERROR_FILEMARK 102
#define IDETAPE_ERROR_EOD 103

/*
- * The following is used to format the general configuration word of
- * the ATAPI IDENTIFY DEVICE command.
+ * Used to format the general configuration word of the ATAPI IDENTIFY DEVICE
+ * command.
*/
-struct idetape_id_gcw {
+struct idetape_id_gcw {
unsigned packet_size :2; /* Packet Size */
unsigned reserved234 :3; /* Reserved */
unsigned drq_type :2; /* Command packet DRQ type */
@@ -540,10 +507,10 @@ struct idetape_id_gcw {
#define IDETAPE_CAPABILITIES_PAGE 0x2a

/*
- * The variables below are used for the character device interface.
- * Additional state variables are defined in our ide_drive_t structure.
+ * The variables below are used for the character device interface. Additional
+ * state variables are defined in our ide_drive_t structure.
*/
-static struct ide_tape_obj * idetape_devs[MAX_HWIFS * MAX_DRIVES];
+static struct ide_tape_obj *idetape_devs[MAX_HWIFS * MAX_DRIVES];

#define ide_tape_f(file) ((file)->private_data)

@@ -559,37 +526,36 @@ static struct ide_tape_obj *ide_tape_chrdev_get(unsigned int i)
return tape;
}

-/*
- * Function declarations
- *
- */
-static int idetape_chrdev_release (struct inode *inode, struct file *filp);
-static void idetape_write_release (ide_drive_t *drive, unsigned int minor);
+static int idetape_chrdev_release(struct inode *inode, struct file *filp);
+static void idetape_write_release(ide_drive_t *drive, unsigned int minor);

/*
* Too bad. The drive wants to send us data which we are not ready to accept.
* Just throw it away.
*/
-static void idetape_discard_data (ide_drive_t *drive, unsigned int bcount)
+static void idetape_discard_data(ide_drive_t *drive, unsigned int bcount)
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```

{
while (bcount--)
(void) HWIF(drive)->INB(IDE_DATA_REG);
}

-static void idetape_input_buffers (ide_drive_t *drive, idetape_pc_t *pc, unsigned int bcount)
+static void idetape_input_buffers(ide_drive_t *drive, idetape_pc_t *pc,
+ uint bcount)
{
struct idetape_bh *bh = pc->bh;
int count;

while (bcount) {
if (bh == NULL) {
- printk(KERN_ERR "ide-tape: bh == NULL in "
- "idetape_input_buffers\n");
+ printk(KERN_ERR "ide-tape: bh == NULL in %s\n",
+ __func__);
idetape_discard_data(drive, bcount);
return;
}
- count = min((unsigned int)(bh->b_size - atomic_read(&bh->b_count)), bcount);
- HWIF(drive)->atapi_input_bytes(drive, bh->b_data + atomic_read(&bh->b_count), count);
+ count = min((uint)(bh->b_size - atomic_read(&bh->b_count)),
+ bcount);
+ HWIF(drive)->atapi_input_bytes(drive,
+ bh->b_data + atomic_read(&bh->b_count), count);
bcount -= count;
atomic_add(count, &bh->b_count);
if (atomic_read(&bh->b_count) == bh->b_size) {
@@ -601,24 +567,26 @@ static void idetape_input_buffers (ide_drive_t *drive, idetape_pc_t *pc, unsigne
pc->bh = bh;
}

-static void idetape_output_buffers (ide_drive_t *drive, idetape_pc_t *pc, unsigned int bcount)
+static void idetape_output_buffers(ide_drive_t *drive, idetape_pc_t *pc,
+ uint bcount)
{
struct idetape_bh *bh = pc->bh;
int count;

while (bcount) {
if (bh == NULL) {
- printk(KERN_ERR "ide-tape: bh == NULL in "
- "idetape_output_buffers\n");
+ printk(KERN_ERR "ide-tape: bh == NULL in %s\n",
+ __func__);
return;
}
- count = min((unsigned int)pc->b_count, (unsigned int)bcount);
+ count = min((uint)pc->b_count, (uint)bcount);
}

```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```

HWIF(drive)->atapi_output_bytes(drive, pc->b_data, count);
bcount -= count;
pc->b_data += count;
pc->b_count -= count;
if (!pc->b_count) {
- pc->bh = bh = bh->b_reqnext;
+ bh = bh->b_reqnext;
+ pc->bh = bh->b_reqnext;
if (bh) {
pc->b_data = bh->b_data;
pc->b_count = atomic_read(&bh->b_count);
@@ -627,7 +595,7 @@ static void idetape_output_buffers (ide_drive_t *drive, idetape_pc_t *pc, unsign
}
}

-static void idetape_update_buffers (idetape_pc_t *pc)
+static void idetape_update_buffers(idetape_pc_t *pc)
{
struct idetape_bh *bh = pc->bh;
int count;
@@ -637,11 +605,11 @@ static void idetape_update_buffers (idetape_pc_t *pc)
return;
while (bcount) {
if (bh == NULL) {
- printk(KERN_ERR "ide-tape: bh == NULL in "
- "idetape_update_buffers\n");
+ printk(KERN_ERR "ide-tape: bh == NULL in %s\n",
+ __func__);
return;
}
- count = min((unsigned int)bh->b_size, (unsigned int)bcount);
+ count = min((uint)bh->b_size, (uint)bcount);
atomic_set(&bh->b_count, count);
if (atomic_read(&bh->b_count) == bh->b_size)
bh = bh->b_reqnext;
@@ -658,7 +626,6 @@ static idetape_pc_t *ide_tape_alloc_pc(void)
if (!pc)
printk(KERN_ERR "ide-tape: %s: memory allocation error.",
__func__);
-
return pc;
}

@@ -674,14 +641,10 @@ static struct request *ide_tape_alloc_rq(void)
if (!rq)
printk(KERN_ERR "ide-tape: %s: memory allocation error.",
__func__);
-
return rq;
}

```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```

-/*
- * idetape_init_pc initializes a packet command.
- */
-static void idetape_init_pc (idetape_pc_t *pc)
+static void idetape_init_pc(idetape_pc_t *pc)
{
memset(pc->c, 0, 12);
pc->retries = 0;
@@ -777,17 +740,14 @@ static void idetape_activate_next_stage(ide_drive_t *drive)
tape->next_stage = stage->next;
}

-/*
- * idetape_kfree_stage calls kfree to completely free a stage, along with
- * its related buffers.
- */
-static void __idetape_kfree_stage (idetape_stage_t *stage)
+/* kfree a stage completely, along with its related buffers. */
+static void __idetape_kfree_stage(idetape_stage_t *stage)
{
struct idetape_bh *prev_bh, *bh = stage->bh;
int size;

- while (bh != NULL) {
- if (bh->b_data != NULL) {
+ while (bh) {
+ if (bh->b_data) {
size = (int) bh->b_size;
while (size > 0) {
free_page((unsigned long) bh->b_data);
@@ -802,16 +762,16 @@ static void __idetape_kfree_stage (idetape_stage_t *stage)
kfree(stage);
}

-static void idetape_kfree_stage (idetape_tape_t *tape, idetape_stage_t *stage)
+static void idetape_kfree_stage(idetape_tape_t *tape, idetape_stage_t *stage)
{
__idetape_kfree_stage(stage);
}

-/*
- * idetape_remove_stage_head removes tape->first_stage from the pipeline.
- * The caller should avoid race conditions.
+ * remove tape->first_stage from the pipeline. The caller should avoid race
+ * conditions.
*/
-static void idetape_remove_stage_head (ide_drive_t *drive)
+static void idetape_remove_stage_head(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;
idetape_stage_t *stage;

```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
@@ -833,10 +793,12 @@ static void idetape_remove_stage_head (ide_drive_t *drive)
tape->nr_stages--;
if (tape->first_stage == NULL) {
tape->last_stage = NULL;
- if (tape->next_stage != NULL)
- printk(KERN_ERR "ide-tape: bug: tape->next_stage != NULL\n");
+ if (tape->next_stage)
+ printk(KERN_ERR
+ "ide-tape: bug: tape->next_stage != NULL\n");
if (tape->nr_stages)
- printk(KERN_ERR "ide-tape: bug: nr_stages should be 0 now\n");
+ printk(KERN_ERR
+ "ide-tape: bug: nr_stages should be 0 now\n");
}
}

@@ -867,8 +829,8 @@ static void idetape_abort_pipeline(ide_drive_t *drive,
}

/*
- * idetape_end_request is used to finish servicing a request, and to
- * insert a pending pipeline request into the main device queue.
+ * finish servicing a request and insert a pending pipeline request into
+ * the main device queue.
*/
static int idetape_end_request(ide_drive_t *drive, int uptodate, int nr_sects)
{
@@ -882,10 +844,11 @@ static int idetape_end_request(ide_drive_t *drive, int uptodate, int nr_sects)
debug_log(DBG_PROCS, "Enter %s\n", __func__);

switch (uptodate) {
- case 0: error = IDETAPE_ERROR_GENERAL; break;
- case 1: error = 0; break;
- default: error = uptodate;
+ case 0: error = IDETAPE_ERROR_GENERAL; break;
+ case 1: error = 0; break;
+ default: error = uptodate;
}
+
rq->errors = error;
if (error)
tape->failed_pc = NULL;
@@ -908,7 +871,8 @@ static int idetape_end_request(ide_drive_t *drive, int uptodate, int nr_sects)
if (error) {
tape->flags |= IDETAPE_FL_PIPELINE_ERR;
if (error == IDETAPE_ERROR_EOD)
- idetape_abort_pipeline(drive, active_stage);
+ idetape_abort_pipeline(drive,
+ active_stage);
}
} else if (rq->cmd[0] & REQ_IDETAPE_READ) {
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
if (error == IDETAPE_ERROR_EOD) {
@@ -916,12 +880,10 @@ static int idetape_end_request(ide_drive_t *drive, int uptodate, int nr_sects)
idetape_abort_pipeline(drive, active_stage);
}
}
- if (tape->next_stage != NULL) {
+ if (tape->next_stage) {
idetape_activate_next_stage(drive);

- /*
- * Insert the next request into the request queue.
- */
+ /* Insert the next request into the request queue. */
(void) ide_do_drive_cmd(drive, tape->act_data_rq,
ide_end);
} else if (!error) {
@@ -955,7 +917,7 @@ static int idetape_end_request(ide_drive_t *drive, int uptodate, int nr_sects)
return 0;
}

-static ide_startstop_t idetape_request_sense_callback (ide_drive_t *drive)
+static ide_startstop_t idetape_request_sense_callback(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;

@@ -965,15 +927,16 @@ static ide_startstop_t idetape_request_sense_callback (ide_drive_t *drive)
idetape_analyze_error(drive, tape->pc->buffer);
idetape_end_request(drive, 1, 0);
} else {
- printk(KERN_ERR "ide-tape: Error in REQUEST SENSE itself - Aborting request!\n");
+ printk(KERN_ERR "ide-tape: Error in REQUEST SENSE itself"
+ " - Aborting request!\n");
idetape_end_request(drive, 0, 0);
}
return ide_stopped;
}

-static void idetape_create_request_sense_cmd (idetape_pc_t *pc)
+static void idetape_create_request_sense_cmd(idetape_pc_t *pc)
{
- idetape_init_pc(pc);
+ idetape_init_pc(pc);
pc->c[0] = REQUEST_SENSE;
pc->c[4] = 20;
pc->rq_xfer = 20;
@@ -988,21 +951,21 @@ static void idetape_init_rq(struct request *rq, u8 cmd)
}

/*
- * idetape_queue_pc_head generates a new packet command request in front
- * of the request queue, before the current request, so that it will be
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```

- * processed immediately, on the next pass through the driver.
+ * generate a new packet command request in front of the request queue, before
+ * the current request, so that it will be processed immediately, on the next
+ * pass through the driver.
*
- * idetape_queue_pc_head is called from the request handling part of
- * the driver (the "bottom" part). Safe storage for the request should
- * be allocated with ide_tape_alloc_pc and ide_tape_alloc_rq
- * before calling idetape_queue_pc_head.
+ * idetape_queue_pc_head() is called from the request handling part of the
+ * driver (the "bottom" part). Safe storage for the request should be allocated
+ * with ide_tape_alloc_pc and ide_tape_alloc_rq before calling the function
+ * below.
*
- * The higher level of the driver – The ioctl handler and the character
- * device handling functions should queue request to the lower level part
- * and wait for their completion using idetape_queue_pc_tail or
- * idetape_queue_rw_tail.
+ * The higher level of the driver – The ioctl handler and the character device
+ * handling functions should queue request to the lower level part and wait for
+ * their completion using idetape_queue_pc_tail() or idetape_queue_rw_tail().
*/
-static void idetape_queue_pc_head (ide_drive_t *drive, idetape_pc_t *pc, struct request *rq)
+static void idetape_queue_pc_head(ide_drive_t *drive, idetape_pc_t *pc,
+ struct request *rq)
{
struct ide_tape_obj *tape = drive->driver_data;

@@ -1041,11 +1004,10 @@ static ide_startstop_t idetape_retry_pc(ide_drive_t *drive)
}

/*
- * idetape_postpone_request postpones the current request so that
- * ide.c will be able to service requests from another device on
- * the same hwgroup while we are polling for DSC.
+ * postpone the current request so that ide.c will be able to service requests
+ * from another device on the same hwgroup while we are polling for DSC.
*/
-static void idetape_postpone_request (ide_drive_t *drive)
+static void idetape_postpone_request(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;

@@ -1074,7 +1036,7 @@ static ide_startstop_t idetape_pc_intr(ide_drive_t *drive)
idetape_io_buf *iobuf;
unsigned int temp;
#if SIMULATE_ERRORS
- static int error_sim_count = 0;
+ static int error_sim_count;
#endif
u16 bcount;
```

```
u8 stat, ireason;
@@ -1235,46 +1197,39 @@ static ide_startstop_t idetape_pc_intr(ide_drive_t *drive)
}

/*
- * Packet Command Interface
- *
- * The current Packet Command is available in tape->pc, and will not
- * change until we finish handling it. Each packet command is associated
- * with a callback function that will be called when the command is
- * finished.
+ * Packet Command Interface
+ *
+ * The handling will be done in three stages:
+ * The current Packet Command is available in tape->pc, and will not change
+ * until we finish handling it. Each packet command is associated with a
+ * callback function that will be called when the command is finished.
+ *
- * 1. idetape_issue_pc will send the packet command to the
- * drive, and will set the interrupt handler to idetape_pc_intr.
+ * The handling will be done in three stages:
+ *
- * 2. On each interrupt, idetape_pc_intr will be called. This step
- * will be repeated until the device signals us that no more
- * interrupts will be issued.
+ * 1. idetape_issue_pc() will send the packet command to the drive, and will set
+ * the interrupt handler to idetape_pc_intr().
+ *
- * 3. ATAPI Tape media access commands have immediate status with a
- * delayed process. In case of a successful initiation of a
- * media access packet command, the DSC bit will be set when the
- * actual execution of the command is finished.
- * Since the tape drive will not issue an interrupt, we have to
- * poll for this event. In this case, we define the request as
- * "low priority request" by setting rq_status to
- * IDETAPE_RQ_POSTPONED, set a timer to poll for DSC and exit
- * the driver.
+ * 2. On each interrupt, idetape_pc_intr() will be called. This step will be
+ * repeated until the device signals us that no more interrupts will be issued.
+ *
- * ide.c will then give higher priority to requests which
- * originate from the other device, until will change rq_status
- * to RQ_ACTIVE.
+ * 3. ATAPI Tape media access commands have immediate status with a delayed
+ * process. In case of a successful initiation of a media access packet command,
+ * the DSC bit will be set when the actual execution of the command is finished.
+ * Since the tape drive will not issue an interrupt, we have to poll for this
+ * event. In this case, we define the request as "low priority request" by
+ * setting rq_status to IDETAPE_RQ_POSTPONED, set a timer to poll for DSC and
+ * exit the driver.
+ * ide.c will then give higher priority to requests which originate from the
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
+ * other device, until we change rq_status to RQ_ACTIVE.
*
- * 4. When the packet command is finished, it will be checked for errors.
+ * 4. When the packet command is finished, it will be checked for errors.
*
- * 5. In case an error was found, we queue a request sense packet
- * command in front of the request queue and retry the operation
- * up to IDETAPE_MAX_PC_RETRIES times.
- *
- * 6. In case no error was found, or we decided to give up and not
- * to retry again, the callback function will be called and then
- * we will handle the next request.
+ * 5. In case an error was found, we queue a request sense packet command in
+ * front of the request queue and retry the operation up to
+ * IDETAPE_MAX_PC_RETRIES times.
*
+ * 6. In case no error was found, or we decided to give up and not to retry
+ * again, the callback function will be called and then we will handle the next
+ * request.
*/
static ide_startstop_t idetape_transfer_pc(ide_drive_t *drive)
{
@@ -1285,8 +1240,9 @@ static ide_startstop_t idetape_transfer_pc(ide_drive_t *drive)
ide_startstop_t startstop;
u8 ireason;

- if (ide_wait_stat(&startstop, drive, DRQ_STAT, BUSY_STAT, WAIT_READY)) {
- printk(KERN_ERR "ide-tape: Strange, packet command initiated yet DRQ isn't asserted\n");
+ if (ide_wait_stat(&startstop, drive, DRQ_STAT, BUSY_STAT, WAIT_READY)) {
+ printk(KERN_ERR "ide-tape: Strange, packet command initiated"
+ " yet DRQ isn't asserted\n");
return startstop;
}
ireason = hwif->INB(IDE_IREASON_REG);
@@ -1329,7 +1285,7 @@ static ide_startstop_t idetape_issue_pc(ide_drive_t *drive, idetape_pc_t *pc)
if (tape->pc->c[0] == REQUEST_SENSE &&
pc->c[0] == REQUEST_SENSE) {
printk(KERN_ERR "ide-tape: possible ide-tape.c bug - "
- "Two request sense in serial were issued\n");
+ "two request sense in serial were issued\n");
}

if (tape->failed_pc == NULL && pc->c[0] != REQUEST_SENSE)
@@ -1384,7 +1340,8 @@ static ide_startstop_t idetape_issue_pc(ide_drive_t *drive, idetape_pc_t *pc)
if (dma_ok) /* Will begin DMA later */
pc->flags |= PC_FL_DMA_IN_PROGRESS;
if (tape->flags & IDETAPE_FL_DRQ_INTERRUPT) {
- ide_set_handler(drive, &idetape_transfer_pc, IDETAPE_WAIT_CMD, NULL);
+ ide_set_handler(drive, &idetape_transfer_pc, IDETAPE_WAIT_CMD,
+ NULL);
hwif->OUTB(WIN_PACKETCMD, IDE_COMMAND_REG);
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
return ide_started;
} else {
@@ -1393,10 +1350,8 @@ static ide_startstop_t idetape_issue_pc(ide_drive_t *drive, idetape_pc_t *pc)
}
}

-/*
- * General packet command callback function.
- */
-static ide_startstop_t idetape_pc_callback (ide_drive_t *drive)
+/* General packet command callback function. */
+static ide_startstop_t idetape_pc_callback(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;

@@ -1406,25 +1361,25 @@ static ide_startstop_t idetape_pc_callback (ide_drive_t *drive)
return ide_stopped;
}

-/*
- * A mode sense command is used to "sense" tape parameters.
- */
-static void idetape_create_mode_sense_cmd (idetape_pc_t *pc, u8 page_code)
+/* A mode sense command is used to "sense" tape parameters. */
+static void idetape_create_mode_sense_cmd(idetape_pc_t *pc, u8 page_code)
{
idetape_init_pc(pc);
pc->c[0] = MODE_SENSE;
if (page_code != IDETAPE_BLOCK_DESCRIPTOR)
- pc->c[1] = 8; /* DBD = 1 - Don't return block descriptors */
+ /* DBD = 1 - Don't return block descriptors */
+ pc->c[1] = 8;
pc->c[2] = page_code;
/*
* Changed pc->c[3] to 0 (255 will at best return unused info).
*
- * For SCSI this byte is defined as subpage instead of high byte
- * of length and some IDE drives seem to interpret it this way
- * and return an error when 255 is used.
+ * For SCSI this byte is defined as subpage instead of high byte of
+ * length and some IDE drives seem to interpret it this way and return
+ * an error when 255 is used.
*/
pc->c[3] = 0;
- pc->c[4] = 255; /* (We will just discard data in that case) */
+ /* We will just discard data in that case */
+ pc->c[4] = 255;
if (page_code == IDETAPE_BLOCK_DESCRIPTOR)
pc->rq_xfer = 12;
else if (page_code == IDETAPE_CAPABILITIES_PAGE)
@@ -1453,7 +1408,7 @@ static void idetape_calculate_speeds(ide_drive_t *drive)
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
(tape->pipeline_head - tape->ctl_prev_pipe_h) * 32 *
HZ / (jiffies - tape->ctl_prev_hptime);

- if (tape->nr_pending_stages < tape->max_stages /*- 1 */) {
+ if (tape->nr_pending_stages < tape->max_stages/*- 1 */) {
/* -1 for read mode error recovery */
if (time_after(jiffies, tape->unctl_prev_hptime + 10 * HZ)) {
tape->unctl_pipe_hptime = jiffies;
@@ -1487,7 +1442,7 @@ static void idetape_calculate_speeds(ide_drive_t *drive)
tape->max_ins_speed = max(tape->max_ins_speed, 500);
}

-static ide_startstop_t idetape_media_access_finished (ide_drive_t *drive)
+static ide_startstop_t idetape_media_access_finished(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;
idetape_pc_t *pc = tape->pc;
@@ -1513,7 +1468,7 @@ static ide_startstop_t idetape_media_access_finished (ide_drive_t *drive)
return pc->callback(drive);
}

-static ide_startstop_t idetape_rw_callback (ide_drive_t *drive)
+static ide_startstop_t idetape_rw_callback(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;
struct request *rq = HWGROUP(drive)->rq;
@@ -1549,23 +1504,25 @@ static ide_startstop_t idetape_rw_callback (ide_drive_t *drive)
return ide_stopped;
}

-static void idetape_create_read_cmd(idetape_tape_t *tape, idetape_pc_t *pc, unsigned int length, struct
idetape_bh *bh)
+static void idetape_create_read_cmd(idetape_tape_t *tape, idetape_pc_t *pc,
+ uint len, struct idetape_bh *bh)
{
idetape_init_pc(pc);
pc->c[0] = READ_6;
- put_unaligned(cpu_to_be32(length), (unsigned int *) &pc->c[1]);
+ put_unaligned(cpu_to_be32(len), (uint *) &pc->c[1]);
pc->c[1] = 1;
pc->callback = &idetape_rw_callback;
pc->bh = bh;
atomic_set(&bh->b_count, 0);
pc->buffer = NULL;
- pc->buf_size = length * tape->blk_sz;
+ pc->buf_size = len * tape->blk_sz;
pc->rq_xfer = pc->buf_size;
- if (pc->rq_xfer == tape->stage_size)
+ if (pc->rq_xfer == tape->stage_sz)
pc->flags |= PC_FL_DMA_RECOMMENDED;
}
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
-static void idetape_create_read_buffer_cmd(idetape_tape_t *tape, idetape_pc_t *pc, unsigned int length,
struct idetape_bh *bh)
+static void idetape_create_read_buffer_cmd(idetape_tape_t *tape,
+ idetape_pc_t *pc, struct idetape_bh *bh)
{
int size = 32768;
struct idetape_bh *p = bh;
@@ -1583,14 +1540,16 @@ static void idetape_create_read_buffer_cmd(idetape_tape_t *tape, idetape_pc_t
*p
atomic_set(&p->b_count, 0);
p = p->b_reqnext;
}
- pc->rq_xfer = pc->buf_size = size;
+ pc->rq_xfer = size;
+ pc->buf_size = size;
}

-static void idetape_create_write_cmd(idetape_tape_t *tape, idetape_pc_t *pc, unsigned int length, struct
idetape_bh *bh)
+static void idetape_create_write_cmd(idetape_tape_t *tape, idetape_pc_t *pc,
+ uint len, struct idetape_bh *bh)
{
idetape_init_pc(pc);
pc->c[0] = WRITE_6;
- put_unaligned(cpu_to_be32(length), (unsigned int *) &pc->c[1]);
+ put_unaligned(cpu_to_be32(len), (uint *) &pc->c[1]);
pc->c[1] = 1;
pc->callback = &idetape_rw_callback;
pc->flags |= PC_FL_WRITING;
@@ -1598,14 +1557,13 @@ static void idetape_create_write_cmd(idetape_tape_t *tape, idetape_pc_t *pc,
uns
pc->b_data = bh->b_data;
pc->b_count = atomic_read(&bh->b_count);
pc->buffer = NULL;
- pc->rq_xfer = pc->buf_size = length * tape->blk_sz;
- if (pc->rq_xfer == tape->stage_size)
+ pc->rq_xfer = len * tape->blk_sz;
+ pc->buf_size = len * tape->blk_sz;
+ if (pc->rq_xfer == tape->stage_size)
pc->flags |= PC_FL_DMA_RECOMMENDED;
}

-/*
- * idetape_do_request is our request handling function.
- */
+/* our request handler */
static ide_startstop_t idetape_do_request(ide_drive_t *drive,
struct request *rq, sector_t block)
{
@@ -1619,23 +1577,18 @@ static ide_startstop_t idetape_do_request(ide_drive_t *drive,
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```

rq->sector, rq->nr_sectors, rq->current_nr_sectors);

if (!blk_special_request(rq)) {
- /*
- * We do not support buffer cache originated requests.
- */
+ /* We do not support buffer cache originated requests. */
printk(KERN_NOTICE "ide-tape: %s: Unsupported request in "
"request queue (%d)\n", drive->name, rq->cmd_type);
ide_end_request(drive, 0, 0);
return ide_stopped;
}

- /*
- * Retry a failed packet command
- */
- if (tape->failed_pc != NULL &&
- tape->pc->c[0] == REQUEST_SENSE) {
+ /* Retry a failed packet command */
+ if (tape->failed_pc && tape->pc->c[0] == REQUEST_SENSE)
return idetape_issue_pc(drive, tape->failed_pc);
- }
- if (postponed_rq != NULL)
+
+ if (postponed_rq)
if (rq != postponed_rq) {
printk(KERN_ERR "ide-tape: ide-tape.c bug - "
"Two DSC requests were queued\n");
@@ -1678,7 +1631,8 @@ static ide_startstop_t idetape_do_request(ide_drive_t *drive,
} else {
return ide_do_reset(drive);
}
- } else if (time_after(jiffies, tape->dsc_polling_start + IDETAPE_DSC_MA_THRESHOLD))
+ } else if (time_after(jiffies, tape->dsc_polling_start +
+ IDETAPE_DSC_MA_THRESHOLD))
tape->dsc_poll_freq = IDETAPE_DSC_MA_SLOW;
idetape_postpone_request(drive);
return ide_stopped;
@@ -1711,7 +1665,7 @@ static ide_startstop_t idetape_do_request(ide_drive_t *drive,
if (!pc)
goto err;

- idetape_create_read_buffer_cmd(tape, pc, rq->current_nr_sectors,
+ idetape_create_read_buffer_cmd(tape, pc,
(struct idetape_bh *)rq->special);
goto out;
}
@@ -1735,10 +1689,8 @@ err:
return ide_stopped;
}

```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```

-/*
- * Pipeline related functions
- */
-static inline int idetape_pipeline_active (idetape_tape_t *tape)
+/* Pipeline related functions */
+static inline int idetape_pipeline_active(idetape_tape_t *tape)
{
int rc1, rc2;

@@ -1748,25 +1700,26 @@ static inline int idetape_pipeline_active (idetape_tape_t *tape)
}

/*
- * idetape_kmalloc_stage uses __get_free_page to allocate a pipeline
- * stage, along with all the necessary small buffers which together make
- * a buffer of size tape->stage_size (or a bit more). We attempt to
- * combine sequential pages as much as possible.
+ * use __get_free_page() to allocate a pipeline stage, along with all the
+ * necessary small buffers which together make a buffer of size tape->stage_sz
+ * (or a bit more). We attempt to combine sequential pages as much as possible.
+ *
- * Returns a pointer to the new allocated stage, or NULL if we
- * can't (or don't want to) allocate a stage.
+ * we return a pointer to the new allocated stage, or NULL if we can't (or don't
+ * want to) allocate a stage.
+ *
- * Pipeline stages are optional and are used to increase performance.
- * If we can't allocate them, we'll manage without them.
+ * pipeline stages are optional and are used to increase performance. If we
+ * can't allocate them, we'll manage without them.
+ */
-static idetape_stage_t *__idetape_kmalloc_stage (idetape_tape_t *tape, int full, int clear)
+static idetape_stage_t *__idetape_kmalloc_stage(idetape_tape_t *tape, int full,
+ int clear)
{
idetape_stage_t *stage;
struct idetape_bh *prev_bh, *bh;
int pages = tape->pages_per_stage;
char *b_data = NULL;

- if ((stage = kmalloc(sizeof (idetape_stage_t),GFP_KERNEL)) == NULL)
+ stage = kmalloc(sizeof(idetape_stage_t), GFP_KERNEL);
+ if (!stage)
return NULL;
stage->next = NULL;

@@ -1774,16 +1727,21 @@ static idetape_stage_t *__idetape_kmalloc_stage (idetape_tape_t *tape, int full,
if (bh == NULL)
goto abort;
bh->b_reqnext = NULL;
- if ((bh->b_data = (char *) __get_free_page (GFP_KERNEL)) == NULL)

```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
+
+ bh->b_data = (char *) __get_free_page(GFP_KERNEL);
+ if (!bh->b_data)
goto abort;
+
if (clear)
memset(bh->b_data, 0, PAGE_SIZE);
bh->b_size = PAGE_SIZE;
atomic_set(&bh->b_count, full ? bh->b_size : 0);

while (--pages) {
- if ((b_data = (char *) __get_free_page (GFP_KERNEL)) == NULL)
+ b_data = (char *) __get_free_page(GFP_KERNEL);
+ if (!b_data)
goto abort;
+
if (clear)
memset(b_data, 0, PAGE_SIZE);
if (bh->b_data == b_data + PAGE_SIZE) {
@@ -1800,8 +1758,9 @@ static idetape_stage_t *__idetape_kmalloc_stage (idetape_tape_t *tape, int full,
continue;
}
prev_bh = bh;
- if ((bh = kmalloc(sizeof(struct idetape_bh), GFP_KERNEL)) == NULL) {
- free_page((unsigned long) b_data);
+ bh = kmalloc(sizeof(struct idetape_bh), GFP_KERNEL);
+ if (!bh) {
+ free_page((ulong) b_data);
goto abort;
}
bh->b_reqnext = NULL;
@@ -1819,7 +1778,7 @@ abort:
return NULL;
}

-static idetape_stage_t *idetape_kmalloc_stage (idetape_tape_t *tape)
+static idetape_stage_t *idetape_kmalloc_stage(idetape_tape_t *tape)
{
idetape_stage_t *cache_stage = tape->cache_stage;

@@ -1827,14 +1786,15 @@ static idetape_stage_t *idetape_kmalloc_stage (idetape_tape_t *tape)

if (tape->nr_stages >= tape->max_stages)
return NULL;
- if (cache_stage != NULL) {
+ if (cache_stage) {
tape->cache_stage = NULL;
return cache_stage;
}
return __idetape_kmalloc_stage(tape, 0, 0);
}
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
-static int idetape_copy_stage_from_user (idetape_tape_t *tape, idetape_stage_t *stage, const char __user
*buf, int n)
+static int idetape_copy_stage_from_user(idetape_tape_t *tape,
+ idetape_stage_t *stage, const char __user *buf, int n)
{
struct idetape_bh *bh = tape->bh;
int count;
@@ -1842,13 +1802,16 @@ static int idetape_copy_stage_from_user (idetape_tape_t *tape, idetape_stage_t
*

while (n) {
if (bh == NULL) {
- printk(KERN_ERR "ide-tape: bh == NULL in "
- "idetape_copy_stage_from_user\n");
+ printk(KERN_ERR "ide-tape: bh == NULL in %s\n",
+ __func__);
return 1;
}
- count = min((unsigned int)(bh->b_size - atomic_read(&bh->b_count)), (unsigned int)n);
- if (copy_from_user(bh->b_data + atomic_read(&bh->b_count), buf, count))
+ count = min((uint)(bh->b_size - atomic_read(&bh->b_count)),
+ (uint)n);
+ if (copy_from_user(bh->b_data + atomic_read(&bh->b_count), buf,
+ count))
ret = 1;
+
n -= count;
atomic_add(count, &bh->b_count);
buf += count;
@@ -1862,7 +1825,8 @@ static int idetape_copy_stage_from_user (idetape_tape_t *tape, idetape_stage_t *
return ret;
}

-static int idetape_copy_stage_to_user (idetape_tape_t *tape, char __user *buf, idetape_stage_t *stage, int n)
+static int idetape_copy_stage_to_user(idetape_tape_t *tape, char __user *buf,
+ idetape_stage_t *stage, int n)
{
struct idetape_bh *bh = tape->bh;
int count;
@@ -1870,8 +1834,8 @@ static int idetape_copy_stage_to_user (idetape_tape_t *tape, char __user *buf, i

while (n) {
if (bh == NULL) {
- printk(KERN_ERR "ide-tape: bh == NULL in "
- "idetape_copy_stage_to_user\n");
+ printk(KERN_ERR "ide-tape: bh == NULL in %s\n",
+ __func__);
return 1;
}
count = min(tape->b_count, n);
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
@@ -1882,7 +1846,8 @@ static int idetape_copy_stage_to_user (idetape_tape_t *tape, char __user *buf, i
tape->b_count -= count;
buf += count;
if (!tape->b_count) {
- tape->bh = bh = bh->b_reqnext;
+ tape->bh = bh->b_reqnext;
+ bh = bh->b_reqnext;
if (bh) {
tape->b_data = bh->b_data;
tape->b_count = atomic_read(&bh->b_count);
@@ -1892,10 +1857,10 @@ static int idetape_copy_stage_to_user (idetape_tape_t *tape, char __user *buf, i
return ret;
}

-static void idetape_init_merge_stage (idetape_tape_t *tape)
+static void idetape_init_merge_stage(idetape_tape_t *tape)
{
struct idetape_bh *bh = tape->merge_stage->bh;
-
+
tape->bh = bh;
if (tape->chrdev_dir == idetape_dir_write)
atomic_set(&bh->b_count, 0);
@@ -1905,7 +1870,7 @@ static void idetape_init_merge_stage (idetape_tape_t *tape)
}
}

-static void idetape_switch_buffers (idetape_tape_t *tape, idetape_stage_t *stage)
+static void idetape_switch_buffers(idetape_tape_t *tape, idetape_stage_t *stage)
{
struct idetape_bh *tmp;

@@ -1915,10 +1880,8 @@ static void idetape_switch_buffers (idetape_tape_t *tape, idetape_stage_t *stage
idetape_init_merge_stage(tape);
}

-/*
- * idetape_add_stage_tail adds a new stage at the end of the pipeline.
- */
-static void idetape_add_stage_tail (ide_drive_t *drive, idetape_stage_t *stage)
+/* add a new stage at the end of the pipeline. */
+static void idetape_add_stage_tail(ide_drive_t *drive, idetape_stage_t *stage)
{
idetape_tape_t *tape = drive->driver_data;
unsigned long flags;
@@ -1927,10 +1890,11 @@ static void idetape_add_stage_tail (ide_drive_t *drive, idetape_stage_t *stage)

spin_lock_irqsave(&tape->que_lock, flags);
stage->next = NULL;
- if (tape->last_stage != NULL)
- tape->last_stage->next=stage;
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
+ if (tape->last_stage)
+ tape->last_stage->next = stage;
else
- tape->first_stage = tape->next_stage=stage;
+ tape->first_stage = stage;
+ tape->next_stage = stage;
tape->last_stage = stage;
if (tape->next_stage == NULL)
tape->next_stage = tape->last_stage;
@@ -1940,19 +1904,18 @@ static void idetape_add_stage_tail (ide_drive_t *drive, idetape_stage_t *stage)
}

/*
- * idetape_wait_for_request installs a completion in a pending request
- * and sleeps until it is serviced.
- *
- * The caller should ensure that the request will not be serviced
- * before we install the completion (usually by disabling interrupts).
+ * install a completion in a pending request and sleep until it is serviced. The
+ * caller should ensure that the request will not be serviced before we install
+ * the completion (usually by disabling interrupts).
*/
-static void idetape_wait_for_request (ide_drive_t *drive, struct request *rq)
+static void idetape_wait_for_request(ide_drive_t *drive, struct request *rq)
{
DECLARE_COMPLETION_ONSTACK(wait);
idetape_tape_t *tape = drive->driver_data;

if (rq == NULL || !blk_special_request(rq)) {
- printk (KERN_ERR "ide-tape: bug: Trying to sleep on non-valid request\n");
+ printk(KERN_ERR "ide-tape: bug: Trying to sleep on non-valid"
+ " request\n");
return;
}
rq->end_io_data = &wait;
@@ -2000,14 +1963,11 @@ static ide_startstop_t idetape_read_position_callback(ide_drive_t *drive)
}

/*
- * idetape_create_write_filemark_cmd will:
- *
- * 1. Write a filemark if write_filemark=1.
- * 2. Flush the device buffers without writing a filemark
- * if write_filemark=0.
- *
+ * Write a filemark if write_filemark=1. Then, flush the device buffers without
+ * writing a filemark if write_filemark=0. *
*/
-static void idetape_create_write_filemark_cmd (ide_drive_t *drive, idetape_pc_t *pc, int write_filemark)
+static void idetape_create_write_filemark_cmd(ide_drive_t *drive,
+ idetape_pc_t *pc, int write_filemark)
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
{
idetape_init_pc(pc);
pc->c[0] = WRITE_FILEMARKS;
@@ -2024,26 +1984,25 @@ static void idetape_create_test_unit_ready_cmd(idetape_pc_t *pc)
}

/*
- * idetape_queue_pc_tail is based on the following functions:
+ * The function below is based on the following functions:
*
- * ide_do_drive_cmd from ide.c
- * cdrom_queue_request and cdrom_queue_packet_command from ide-cd.c
+ * ide_do_drive_cmd() from ide.c
+ * cdrom_queue_request() and cdrom_queue_packet_command() from ide-cd.c
*
- * We add a special packet command request to the tail of the request
- * queue, and wait for it to be serviced.
+ * We add a special packet command request to the tail of the request queue, and
+ * wait for it to be serviced.
*
- * This is not to be called from within the request handling part
- * of the driver ! We allocate here data in the stack, and it is valid
- * until the request is finished. This is not the case for the bottom
- * part of the driver, where we are always leaving the functions to wait
- * for an interrupt or a timer event.
+ * This is not to be called from within the request handling part of the driver!
+ * We allocate here data on the stack, and it is valid until the request is
+ * finished. This is not the case for the bottom part of the driver, where we
+ * are always leaving the functions to wait for an interrupt or a timer event.
*
- * From the bottom part of the driver, we should allocate safe memory
- * using ide_tape_alloc_pc and ide_tape_alloc_rq, and add
- * the request to the request list without waiting for it to be serviced !
- * In that case, we usually use idetape_queue_pc_head.
+ * From the bottom part of the driver, we should allocate safe memory using
+ * ide_tape_alloc_pc() and ide_tape_alloc_rq() and add the request to the
+ * request list without waiting for it to be serviced! In that case, we usually
+ * use idetape_queue_pc_head().
*/
-static int __idetape_queue_pc_tail (ide_drive_t *drive, idetape_pc_t *pc)
+static int __idetape_queue_pc_tail(ide_drive_t *drive, idetape_pc_t *pc)
{
struct ide_tape_obj *tape = drive->driver_data;
struct request rq;
@@ -2054,7 +2013,8 @@ static int __idetape_queue_pc_tail (ide_drive_t *drive, idetape_pc_t *pc)
return ide_do_drive_cmd(drive, &rq, ide_wait);
}

-static void idetape_create_load_unload_cmd (ide_drive_t *drive, idetape_pc_t *pc,int cmd)
+static void idetape_create_load_unload_cmd(ide_drive_t *drive, idetape_pc_t *pc,
+ int cmd)
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
{
idetape_init_pc(pc);
pc->c[0] = START_STOP;
@@ -2063,15 +2023,13 @@ static void idetape_create_load_unload_cmd (ide_drive_t *drive, idetape_pc_t
*pc
pc->callback = &idetape_pc_callback;
}

-static int idetape_wait_ready(ide_drive_t *drive, unsigned long timeout)
+/* Wait for the tape to become ready */
+static int idetape_wait_ready(ide_drive_t *drive, ulong timeout)
{
idetape_tape_t *tape = drive->driver_data;
idetape_pc_t pc;
int load_attempted = 0;

- /*
- * Wait for the tape to become ready
- */
tape->flags |= IDETAPE_FL_MEDIUM_PRESENT;
timeout += jiffies;
while (time_before(jiffies, timeout)) {
@@ -2079,10 +2037,12 @@ static int idetape_wait_ready(ide_drive_t *drive, unsigned long timeout)
if (!__idetape_queue_pc_tail(drive, &pc))
return 0;
if ((tape->sense_key == 2 && tape->asc == 4 && tape->ascq == 2)
- || (tape->asc == 0x3A)) { /* no media */
+ || (tape->asc == 0x3A)) {
+ /* no media */
if (load_attempted)
return -ENOMEDIUM;
- idetape_create_load_unload_cmd(drive, &pc, IDETAPE_LU_LOAD_MASK);
+ idetape_create_load_unload_cmd(drive, &pc,
+ IDETAPE_LU_LOAD_MASK);
__idetape_queue_pc_tail(drive, &pc);
load_attempted = 1;
/* not about to be ready */
@@ -2094,24 +2054,25 @@ static int idetape_wait_ready(ide_drive_t *drive, unsigned long timeout)
return -EIO;
}

-static int idetape_queue_pc_tail (ide_drive_t *drive, idetape_pc_t *pc)
+static int idetape_queue_pc_tail(ide_drive_t *drive, idetape_pc_t *pc)
{
return __idetape_queue_pc_tail(drive, pc);
}

-static int idetape_flush_tape_buffers (ide_drive_t *drive)
+static int idetape_flush_tape_buffers(ide_drive_t *drive)
{
idetape_pc_t pc;
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
int rc;

idetape_create_write_filemark_cmd(drive, &pc, 0);
- if ((rc = idetape_queue_pc_tail(drive, &pc)))
+ rc = idetape_queue_pc_tail(drive, &pc);
+ if (rc)
return rc;
idetape_wait_ready(drive, 60 * 5 * HZ);
return 0;
}

-static void idetape_create_read_position_cmd (idetape_pc_t *pc)
+static void idetape_create_read_position_cmd(idetape_pc_t *pc)
{
idetape_init_pc(pc);
pc->c[0] = READ_POSITION;
@@ -2119,7 +2080,7 @@ static void idetape_create_read_position_cmd (idetape_pc_t *pc)
pc->callback = &idetape_read_position_callback;
}

-static int idetape_read_position (ide_drive_t *drive)
+static int idetape_read_position(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;
idetape_pc_t pc;
@@ -2134,18 +2095,20 @@ static int idetape_read_position (ide_drive_t *drive)
return position;
}

-static void idetape_create_locate_cmd (ide_drive_t *drive, idetape_pc_t *pc, unsigned int block, u8
partition, int skip)
+static void idetape_create_locate_cmd(ide_drive_t *drive, idetape_pc_t *pc,
+ uint block, u8 partition, int skip)
{
idetape_init_pc(pc);
pc->c[0] = POSITION_TO_ELEMENT;
pc->c[1] = 2;
- put_unaligned(cpu_to_be32(block), (unsigned int *) &pc->c[3]);
+ put_unaligned(cpu_to_be32(block), (uint *) &pc->c[3]);
pc->c[8] = partition;
pc->flags |= PC_FL_WAIT_FOR_DSC;
pc->callback = &idetape_pc_callback;
}

-static int idetape_create_prevent_cmd (ide_drive_t *drive, idetape_pc_t *pc, int prevent)
+static int idetape_create_prevent_cmd(ide_drive_t *drive, idetape_pc_t *pc,
+ int prevent)
{
idetape_tape_t *tape = drive->driver_data;

@@ -2160,7 +2123,7 @@ static int idetape_create_prevent_cmd (ide_drive_t *drive, idetape_pc_t *pc, int
```

```

return 1;
}

-static int __idetape_discard_read_pipeline (ide_drive_t *drive)
+static int __idetape_discard_read_pipeline(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;
unsigned long flags;
@@ -2173,10 +2136,11 @@ static int __idetape_discard_read_pipeline (ide_drive_t *drive)
cnt = tape->merge_stage_sz / tape->blk_sz;
if (tape->flags & IDETAPE_FL_FILEMARK) {
tape->flags &= ~IDETAPE_FL_FILEMARK;
- ++cnt; /* Filemarks count as 1 sector */
+ /* Filemarks count as 1 sector */
+ ++cnt;
}
tape->merge_stage_sz = 0;
- if (tape->merge_stage != NULL) {
+ if (tape->merge_stage) {
__idetape_kfree_stage(tape->merge_stage);
tape->merge_stage = NULL;
}
@@ -2195,10 +2159,10 @@ static int __idetape_discard_read_pipeline (ide_drive_t *drive)
idetape_wait_for_request(drive, tape->act_data_rq);
spin_unlock_irqrestore(&tape->que_lock, flags);

- while (tape->first_stage != NULL) {
+ while (tape->first_stage) {
struct request *rq_ptr = &tape->first_stage->rq;

- cnt += rq_ptr->nr_sectors - rq_ptr->current_nr_sectors;
+ cnt += rq_ptr->nr_sectors - rq_ptr->current_nr_sectors;
if (rq_ptr->errors == IDETAPE_ERROR_FILEMARK)
++cnt;
idetape_remove_stage_head(drive);
@@ -2209,15 +2173,14 @@ static int __idetape_discard_read_pipeline (ide_drive_t *drive)
}

/*
- * idetape_position_tape positions the tape to the requested block
- * using the LOCATE packet command. A READ POSITION command is then
- * issued to check where we are positioned.
+ * position the tape to the requested block using the LOCATE packet command. A
+ * READ POSITION command is then issued to check where we are positioned.
*
- * Like all higher level operations, we queue the commands at the tail
- * of the request queue and wait for their completion.
- *
+ * Like all higher level operations, we queue the commands at the tail of the
+ * request queue and wait for their completion.
*/

```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
-static int idetape_position_tape (ide_drive_t *drive, unsigned int block, u8 partition, int skip)
+static int idetape_position_tape(ide_drive_t *drive, uint block, u8 partition,
+ int skip)
{
idetape_tape_t *tape = drive->driver_data;
int retval;
@@ -2235,28 +2198,30 @@ static int idetape_position_tape (ide_drive_t *drive, unsigned int block, u8 par
return (idetape_queue_pc_tail(drive, &pc));
}

-static void idetape_discard_read_pipeline (ide_drive_t *drive, int restore_position)
+static void idetape_discard_read_pipeline(ide_drive_t *drive, int restore_pos)
{
idetape_tape_t *tape = drive->driver_data;
int cnt;
int seek, position;

cnt = __idetape_discard_read_pipeline(drive);
- if (restore_position) {
+ if (restore_pos) {
position = idetape_read_position(drive);
seek = position > cnt ? position - cnt : 0;
if (idetape_position_tape(drive, seek, 0, 0)) {
- printk(KERN_INFO "ide-tape: %s: position_tape failed in discard_pipeline()\n", tape->name);
+ printk(KERN_INFO "ide-tape: %s: position_tape failed in"
+ " %s\n", tape->name, __func__);
return;
}
}
}

/*
- * idetape_queue_rw_tail generates a read/write request for the block
- * device interface and wait for it to be serviced.
+ * generate a read/write request for the block device interface and wait for it
+ * to be serviced.
*/
-static int idetape_queue_rw_tail(ide_drive_t *drive, int cmd, int blocks, struct idetape_bh *bh)
+static int idetape_queue_rw_tail(ide_drive_t *drive, int cmd, int blocks,
+ struct idetape_bh *bh)
{
idetape_tape_t *tape = drive->driver_data;
struct request rq;
@@ -2273,7 +2238,8 @@ static int idetape_queue_rw_tail(ide_drive_t *drive, int cmd, int blocks, struct
rq.rq_disk = tape->disk;
rq.special = (void *)bh;
rq.sector = tape->first_frm_pos;
- rq.nr_sectors = rq.current_nr_sectors = blocks;
+ rq.nr_sectors = blocks;
+ rq.current_nr_sectors = blocks;
(void) ide_do_drive_cmd(drive, &rq, ide_wait);
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
if ((cmd & (REQ_IDETAPE_READ | REQ_IDETAPE_WRITE)) == 0)
@@ -2286,10 +2252,7 @@ static int idetape_queue_rw_tail(ide_drive_t *drive, int cmd, int blocks, struct
return (tape->blk_sz * (blocks-rq.current_nr_sectors));
}

-/*
- * idetape_ins_ppl_into_queue is used to start servicing the
- * pipeline stages, starting from tape->next_stage.
- */
+/* start servicing the pipeline stages, starting from tape->next_stage. */
static void idetape_ins_ppl_into_queue(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;
@@ -2303,15 +2266,16 @@ static void idetape_ins_ppl_into_queue(ide_drive_t *drive)
}
}

-static void idetape_create_inquiry_cmd (idetape_pc_t *pc)
+static void idetape_create_inquiry_cmd(idetape_pc_t *pc)
{
idetape_init_pc(pc);
pc->c[0] = INQUIRY;
- pc->c[4] = pc->rq_xfer = 254;
+ pc->c[4] = 254;
+ pc->rq_xfer = 254;
pc->callback = &idetape_pc_callback;
}

-static void idetape_create_rewind_cmd (ide_drive_t *drive, idetape_pc_t *pc)
+static void idetape_create_rewind_cmd(ide_drive_t *drive, idetape_pc_t *pc)
{
idetape_init_pc(pc);
pc->c[0] = REZERO_UNIT;
@@ -2319,7 +2283,7 @@ static void idetape_create_rewind_cmd (ide_drive_t *drive, idetape_pc_t *pc)
pc->callback = &idetape_pc_callback;
}

-static void idetape_create_erase_cmd (idetape_pc_t *pc)
+static void idetape_create_erase_cmd(idetape_pc_t *pc)
{
idetape_init_pc(pc);
pc->c[0] = ERASE;
@@ -2328,17 +2292,17 @@ static void idetape_create_erase_cmd (idetape_pc_t *pc)
pc->callback = &idetape_pc_callback;
}

-static void idetape_create_space_cmd (idetape_pc_t *pc,int count, u8 cmd)
+static void idetape_create_space_cmd(idetape_pc_t *pc, int count, u8 cmd)
{
idetape_init_pc(pc);
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
pc->c[0] = SPACE;
- put_unaligned(cpu_to_be32(count), (unsigned int *) &pc->c[1]);
+ put_unaligned(cpu_to_be32(count), (uint *) &pc->c[1]);
pc->c[1] = cmd;
pc->flags |= PC_FL_WAIT_FOR_DSC;
pc->callback = &idetape_pc_callback;
}

-static void idetape_wait_first_stage (ide_drive_t *drive)
+static void idetape_wait_first_stage(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;
unsigned long flags;
@@ -2352,17 +2316,16 @@ static void idetape_wait_first_stage (ide_drive_t *drive)
}

/*
- * idetape_add_chrdev_write_request tries to add a character device
- * originated write request to our pipeline. In case w+ gcw.protocol);
else if (gcw.device_type != 1)
- printk(KERN_ERR "ide-tape: Device type is not set to tape\n");
+ printk(KERN_ERR "ide-tape: Device type %d is not set to tape\n",
+ gcw.device_type);
else if (!gcw.removable)
printk(KERN_ERR "ide-tape: The removable flag is not set\n");
else if (gcw.packet_size != 0) {
- printk(KERN_ERR "ide-tape: Packet size is not 12 bytes long\n");
+ printk(KERN_ERR "ide-tape: Packet size of %d is not 12 bytes"
+ " long\n", gcw.packet_size);
if (gcw.packet_size == 1)
- printk(KERN_ERR "ide-tape: Sorry, padding to 16 bytes is still not supported\n");
+ printk(KERN_ERR "ide-tape: Sorry, padding to 16 bytes"
+ " is still not supported\n");
} else
return 1;
return 0;
@@ -3463,7 +3439,7 @@ static void idetape_get_inquiry_results(ide_drive_t *drive)
* Ask the tape about its various parameters. In particular, we will adjust our
* data transfer buffer size to the recommended value as returned by the tape.
*/
-static void idetape_get_mode_sense_results (ide_drive_t *drive)
+static void idetape_get_mode_sense_results(ide_drive_t *drive)
{
idetape_tape_t *tape = drive->driver_data;
idetape_pc_t pc;
@@ -3511,51 +3487,61 @@ static void idetape_get_mode_sense_results (ide_drive_t *drive)
}

#ifdef CONFIG_IDE_PROC_FS
-static void idetape_add_settings (ide_drive_t *drive)
+static void idetape_add_settings(ide_drive_t *drive)
```

[PATCH 32/32] ide-tape: cleanup the remaining codestyle issues

```
{
idetape_tape_t *tape = drive->driver_data;

-/*
- * drive setting name read/write data type min max mul_factor div_factor data pointer set function
- */
ide_add_setting(drive, "buffer", SETTING_READ, TYPE_SHORT, 0, 0xffff,
1, 2, (u16 *)&tape->caps[16], NULL);
- ide_add_setting(drive, "pipeline_min", SETTING_RW, TYPE_INT, 1, 0xffff, tape->stage_size / 1024, 1,
&tape->min_pipeline, NULL);
- ide_add_setting(drive, "pipeline", SETTING_RW, TYPE_INT, 1, 0xffff, tape->stage_size / 1024, 1,
&tape->max_stages, NULL);
- ide_add_setting(drive, "pipeline_max", SETTING_RW, TYPE_INT, 1, 0xffff, tape->stage_size / 1024, 1,
&tape->max_pipeline, NULL);
- ide_add_setting(drive, "pipeline_used", SETTING_READ, TYPE_INT, 0, 0xffff, tape->stage_size / 1024,
1, &tape->nr_stages, NULL);
- ide_add_setting(drive, "pipeline_pending", SETTING_READ, TYPE_INT, 0, 0xffff, tape->stage_size /
1024, 1, &tape->nr_pending_stages, NULL);
+ ide_add_setting(drive, "pipeline_min", SETTING_RW, TYPE_INT, 1, 0xffff,
+ tape->stage_sz / 1024, 1, &tape->min_pipeline, NULL);
+ ide_add_setting(drive, "pipeline", SETTING_RW, TYPE_INT, 1, 0xffff,
+ tape->stage_sz / 1024, 1, &tape->max_stages, NULL);
+ ide_add_setting(drive, "pipeline_max", SETTING_RW, TYPE_INT, 1, 0xffff,
+ tape->stage_sz / 1024, 1, &tape->max_pipeline, NULL);
+
+ ide_add_setting(drive, "pipeline_used", SETTING_READ, TYPE_INT, 0,
+ 0xffff, tape->stage_sz / 1024, 1, &tape->nr_stages,
+ NULL);
+
+ ide_add_setting(drive, "pipeline_pending", SETTING_READ, TYPE_INT, 0,
+ 0xffff, tape->stage_sz / 1024, 1,
+ &tape->nr_pending_stages, NULL);
+
ide_add_setting(drive, "speed", SETTING_READ, TYPE_SHORT, 0, 0xffff,
1, 1, (u16 *)&tape->caps[14], NULL);
- ide_add_setting(drive, "stage", SETTING_READ, TYPE_INT, 0, 0xffff, 1, 1024, &tape->stage_size,
NULL);
+ ide_add_setting(drive, "stage", SETTING_READ, TYPE_INT, 0, 0xffff,
+ 1, 1024, &tape->stage_sz, NULL);
ide_add_setting(drive, "tdsc", SETTING_RW, TYPE
```