```
/* -*- linux-c -*- ------------------------------------------------------- *
 *
 *   Copyright (C) 1991, 1992 Linus Torvalds
 *   Copyright 2007 rPath, Inc. - All Rights Reserved
 *   Copyright 2009 Intel Corporation; author H. Peter Anvin
 *
 *   This file is part of the Linux kernel, and is made available under
 *   the terms of the GNU General Public License version 2.
 *
 * ----------------------------------------------------------------------- */

/*
 * Main module for the real-mode kernel code
 */

#include "boot.h"

struct boot_params boot_params __attribute__((aligned(16)));

char *HEAP =  end;
char *heap end =  end;                /* Default end of heap = no heap */

/*
 * Copy the header into the boot parameter block.  Since this
 * screws up the old-style command line protocol, adjust by
 * filling in the new-style command line pointer instead.
 */

static void copy_boot_params(void)
{
          struct old cmdline {
                    u16 cl magic;
                    u16 cl_offset;
          };
          const struct old_cmdline * const oldcmd =
                    (const struct old cmdline *)OLD CL ADDRESS;

          BUILD_BUG_ON(sizeof boot_params != 4096);
          memcpy(&boot_params.hdr, &hdr, sizeof hdr);

          if (!boot_params.hdr.cmd_line_ptr &&
              oldcmd->cl magic == OLD CL MAGIC) {
                    /* Old-style command line protocol. */
                    u16 cmdline_seg;

                    /* Figure out if the command line falls in the region
                       of memory that an old kernel would have copied up
                       to 0x90000... */
                    if (oldcmd->cl_offset < boot_params.hdr.setup_move_size)
                              cmdline_seg = ds();
                    else
                              cmdline_seg = 0x9000;

                    boot params.hdr.cmd line ptr =
                              (cmdline_seg << 4) + oldcmd->cl_offset;
          }
}

/*
 * Set the keyboard repeat rate to maximum.  Unclear why this
 * is done here; this might be possible to kill off as stale code.
 */
static void keyboard set repeat(void)
{
          struct biosregs ireg;
          initregs(&ireg);
          ireg.ax = 0x0305;
          intcall(0x16, &ireg, NULL);
}

```

```
/*
 * Get Intel SpeedStep (IST) information.
 */
static void query ist(void)
{
          struct biosregs ireg, oreg;

          /* Some older BIOSes apparently crash on this call, so filter
             it from machines too old to have SpeedStep at all. */
          if (cpu.level < 6)
                    return;

          initregs(&ireg);
          ireg.ax  = 0xe980;  /* IST Support */
          ireg.edx = 0x47534943;        /* Request value */
          intcall(0x15, &ireg, &oreg);

          boot_params.ist_info.signature  = oreg.eax;
          boot params.ist info.command    = oreg.ebx;
          boot params.ist info.event      = oreg.ecx;
          boot params.ist info.perf level = oreg.edx;
}

/*
 * Tell the BIOS what CPU mode we intend to run in.
 */
static void set bios mode(void)
{
#ifdef CONFIG_X86_64
          struct biosregs ireg;

          initregs(&ireg);
          ireg.ax = 0xec00;
          ireg.bx = 2;
          intcall(0x15, &ireg, NULL);
#endif
}

static void init_heap(void)
{
          char *stack_end;

          if (boot_params.hdr.loadflags & CAN_USE_HEAP) {
                    asm("leal %P1(%%esp),%0"
                        : "=r" (stack_end) : "i" (-STACK_SIZE));

                    heap_end = (char *)
                              ((size t)boot params.hdr.heap end ptr + 0x200);
                    if (heap_end > stack_end)
                              heap_end = stack_end;
          } else {
                    /* Boot protocol 2.00 only, no heap available */
                    puts("WARNING: Ancient bootloader, some functionality "
                         "may be limited!\n");
          }
}

void main(void)
{
          /* First, copy the boot header into the "zeropage" */
          copy_boot_params();

          /* End of heap check */
          init heap();

          /* Make sure we have all the proper CPU support */
          if (validate_cpu()) {
                    puts("Unable to boot - please use a kernel appropriate "
                         "for your CPU.\n");
                    die();
          }

          /* Tell the BIOS what CPU mode we intend to run in. */
          set_bios_mode();

```

```
146             /* Detect memory layout */
147             detect_memory();
148
149             /* Set keyboard repeat rate (why?) */
150             keyboard_set_repeat();
151
152             /* Query MCA information */
153             query_mca();
154
155             /* Query Intel SpeedStep (IST) information */
156             query_ist();
157
158             /* Query APM information */
159     #if defined(CONFIG_APM) || defined(CONFIG_APM_MODULE)
160             query_apm_bios();
161     #endif
162
163             /* Query EDD information */
164     #if defined(CONFIG_EDD) || defined(CONFIG_EDD_MODULE)
165             query_edd();
166     #endif
167
168             /* Set the video mode */
169             set_video();
170
171             /* Parse command line for 'quiet' and pass it to decompressor. */
172             if (cmdline_find_option_bool("quiet"))
173                     boot_params.hdr.loadflags |= QUIET_FLAG;
174
175             /* Do the last things and invoke protected mode */
176             go_to_protected_mode();
177     }
```