
218951 Tue Jul 1 08:15:46 2008

new/tools/ioemu/vl.c

Enable pty-based serial console emulation in ioemu.

Signed-off-by: John Levon <john.levon@sun.com>

```

1 /*
2  * QEMU System Emulator
3  *
4  * Copyright (c) 2003-2007 Fabrice Bellard
5  *
6  * Permission is hereby granted, free of charge, to any person obtaining a copy
7  * of this software and associated documentation files (the "Software"), to deal
8  * in the Software without restriction, including without limitation the rights
9  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10 * copies of the Software, and to permit persons to whom the Software is
11 * furnished to do so, subject to the following conditions:
12 *
13 * The above copyright notice and this permission notice shall be included in
14 * all copies or substantial portions of the Software.
15 *
16 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
19 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22 * THE SOFTWARE.
23 */
24 #include "vl.h"

26 #include <unistd.h>
27 #include <fcntl.h>
28 #include <signal.h>
29 #include <time.h>
30 #include <errno.h>
31 #include <sys/time.h>
32 #include <zlib.h>

34 #ifndef _WIN32
35 #include <sys/times.h>
36 #include <sys/wait.h>
37 #include <termios.h>
38 #include <sys/poll.h>
39 #include <sys/mman.h>
40 #include <sys/ioctl.h>
41 #include <sys/socket.h>
42 #include <netinet/in.h>
43 #include <arpa/inet.h>
44 #include <dirent.h>
45 #include <netdb.h>
46 #ifdef _BSD
47 #include <sys/stat.h>
48 #ifndef __APPLE__
49 #include <libutil.h>
50 #endif
51 #else
52 #ifdef __sun__
53 #include <libdlpi.h>
54 #include <sys/ethernet.h>
55 #include <stropts.h>
56 #include <sys/bufmod.h>
57 #include <assert.h>
58 #else
59 #include <linux/if.h>

```

```

60 #include <linux/if_tun.h>
61 #include <pty.h>
62 #include <malloc.h>
63 #include <linux/rtc.h>
64 #include <linux/ppdev.h>
65 #endif
66 #endif
67 #endif

69 #if defined(CONFIG_SLIRP)
70 #include "libslirp.h"
71 #endif

73 #ifdef _WIN32
74 #include <malloc.h>
75 #include <sys/timeb.h>
76 #include <windows.h>
77 #define getopt_long_only getopt_long
78 #define memalign(align, size) malloc(size)
79 #endif

81 #include "qemu_socket.h"

83 #ifdef CONFIG_SDL
84 #ifdef __APPLE__
85 #include <SDL/SDL.h>
86 #endif
87 #endif /* CONFIG_SDL */

89 #ifdef CONFIG_COCOA
90 #undef main
91 #define main qemu_main
92 #endif /* CONFIG_COCOA */

94 #include "disas.h"

96 #include "exec-all.h"

98 #include <xen/hvm/params.h>
99 #define DEFAULT_NETWORK_SCRIPT "/etc/xen/qemu-ifup"
100 #define DEFAULT_BRIDGE "xenbr0"
101 #ifdef __sun__
102 #define SMBD_COMMAND "/usr/sfw/sbin/smbd"
103 #else
104 #define SMBD_COMMAND "/usr/sbin/smbd"
105 #endif

107 // #define DEBUG_UNUSED_IOPORT
108 // #define DEBUG_IOPORT

110 #define PHYS_RAM_MAX_SIZE (2047 * 1024 * 1024)

112 #ifdef TARGET_PPC
113 #define DEFAULT_RAM_SIZE 144
114 #else
115 #define DEFAULT_RAM_SIZE 128
116 #endif
117 /* in ms */
118 #define GUI_REFRESH_INTERVAL 30

120 /* Max number of USB devices that can be specified on the commandline. */
121 #define MAX_USB_CMDLINE 8

123 /* XXX: use a two level table to limit memory usage */
124 #define MAX_IOPORTS 65536

```

```

126 const char *bios_dir = CONFIG_QEMU_SHAREDIR;
127 char phys_ram_file[1024];
128 void *ioport_opaque[MAX_IOPORTS];
129 IOPortReadFunc *ioport_read_table[3][MAX_IOPORTS];
130 IOPortWriteFunc *ioport_write_table[3][MAX_IOPORTS];
131 /* Note: bs_table[MAX_DISKS] is a dummy block driver if none available
132    to store the VM snapshots */
133 BlockDriverState *bs_table[MAX_DISKS + MAX_SCSI_DISKS + 1], *fd_table[MAX_FD];
134 /* point to the block driver where the snapshots are managed */
135 BlockDriverState *bs_snapshots;
136 int vga_ram_size;
137 int bios_size;
138 static DisplayState display_state;
139 int nographic;
140 int vncviewer;
141 int vncunused;
142 struct sockaddr_in vnclisten_addr;
143 const char* keyboard_layout = NULL;
144 int64_t ticks_per_sec;
145 char *boot_device = NULL;
146 uint64_t ram_size;
147 int pit_min_timer_count = 0;
148 int nb_nics;
149 NICInfo nd_table[MAX_NICS];
150 QEMUTimer *gui_timer;
151 int vm_running;
152 int rtc_utc = 1;
153 int cirrus_vga_enabled = 1;
154 #ifdef TARGET_SPARC
155 int graphic_width = 1024;
156 int graphic_height = 768;
157 #else
158 int graphic_width = 800;
159 int graphic_height = 600;
160 #endif
161 int graphic_depth = 15;
162 int full_screen = 0;
163 int no_quit = 0;
164 CharDriverState *serial_hds[MAX_SERIAL_PORTS];
165 CharDriverState *parallel_hds[MAX_PARALLEL_PORTS];
166 #ifdef TARGET_I386
167 int win2k_install_hack = 0;
168 #endif
169 int usb_enabled = 0;
170 static VLANState *first_vlan;
171 int smp_cpus = 1;
172 const char *vnc_display;
173 #if defined(TARGET_SPARC)
174 #define MAX_CPUS 16
175 #elif defined(TARGET_I386)
176 #define MAX_CPUS 255
177 #else
178 #define MAX_CPUS 1
179 #endif
180 int acpi_enabled = 0;
181 int fd_bootchk = 1;
182 int no_reboot = 0;
183 int daemonize = 0;
184 const char *option_rom[MAX_OPTION_ROMS];
185 int nb_option_roms;
186 int semihosting_enabled = 0;
187 int autostart = 1;

189 extern int vcpus;

191 int xc_handle;

```

```

193 time_t timeoffset = 0;

195 char domain_name[64] = "xVM-HVM-no-name";
196 extern int domid;

198 char vncpasswd[64];
199 unsigned char challenge[AUTHCHALLENGESIZE];

201 /*****
202  * x86 ISA bus support */

204 target_phys_addr_t isa_mem_base = 0;
205 PicState2 *isa_pic;

207 uint32_t default_ioport_readb(void *opaque, uint32_t address)
208 {
209 #ifdef DEBUG_UNUSED_IOPORT
210     fprintf(stderr, "inb: port=0x%04x\n", address);
211 #endif
212     return 0xff;
213 }

    unchanged_portion_omitted

1826 #ifdef __sun
1827 static int openpty(int *amaster, int *aslave, char *name,
1828                  struct termios *termp, struct winsize *winp)
1829 {
1830     const char *slave;
1831     int mfd = -1, sfd = -1;

1833     *amaster = *aslave = -1;

1835     mfd = open("/dev/ptmx", O_RDWR | O_NOCTTY);
1836     if (mfd < 0)
1837         goto err;

1839     if (grantpt(mfd) == -1 || unlockpt(mfd) == -1)
1840         goto err;

1842     if ((slave = ptsname(mfd)) == NULL)
1843         goto err;

1845     if ((sfd = open(slave, O_RDONLY | O_NOCTTY)) == -1)
1846         goto err;

1848     if (ioctl(sfd, I_PUSH, "ptem") == -1 ||
1849         ioctl(sfd, I_PUSH, "ldterm") == -1)
1850         goto err;

1852     if (amaster)
1853         *amaster = mfd;
1854     if (aslave)
1855         *aslave = sfd;
1856     if (winp)
1857         ioctl(sfd, TIOCSWINSZ, winp);

1859     if (termp)
1860         tcsetattr(sfd, TCSANOW, termp);

1862     assert(name == NULL);

1864     return 0;

1866 err:
1867     if (sfd != -1)

```

```

1868         close(sfd);
1869     close(mfd);
1870     return -1;
1871 }

1873 void cfmakeraw (struct termios *termios_p)
1874 {
1875     termios_p->c_iflag &=
1876         ~(IGNBRK|BRKINT|PARMRK|ISTRIP|INLCR|IGNCR|ICRNL|IXON);
1877     termios_p->c_oflag &= ~OPOST;
1878     termios_p->c_lflag &= ~(ECHO|ECHONL|ICANON|ISIG|IEXTEN);
1879     termios_p->c_cflag &= ~(CSIZE|PARENB);
1880     termios_p->c_cflag |= CS8;

1882     termios_p->c_cc[VMIN] = 0;
1883     termios_p->c_cc[VTIME] = 0;
1884 }

1886 #endif /* __sun */

1888 #if defined(__linux__) || defined(__sun__)
1825 #if defined(__linux__)
1889 static CharDriverState *qemu_chr_open_pty(void)
1890 {
1891     struct termios tty;
1892     int master_fd, slave_fd;
1893
1831     /* Not satisfying */
1894     if (openpty(&master_fd, &slave_fd, NULL, NULL, NULL) < 0) {
1895         return NULL;
1896     }
1897
1898     /* Set raw attributes on the pty. */
1899     tcgetattr(slave_fd, &tty);
1900     cfmakeraw(&tty);
1901     tcsetattr(slave_fd, TCSANOW, &tty);
1838     tcsetattr(slave_fd, TCSAFLUSH, &tty);
1902
1903     fprintf(stderr, "char device redirected to %s\n", ptsname(master_fd));

1905     return qemu_chr_open_fd(master_fd, master_fd);
1906 }
1907 #else /* defined(__linux__) || defined(__sun__) */
1908 static CharDriverState *qemu_chr_open_pty(void)
1909 {
1910     return NULL;
1911 }
1912 #endif /* defined(__linux__) || defined(__sun__) */

1914 #ifndef __linux__

1916 static void tty_serial_init(int fd, int speed,
1917                             int parity, int data_bits, int stop_bits)
1918 {
1919     struct termios tty;
1920     speed_t spd;

1922 #if 0
1923     printf("tty_serial_init: speed=%d parity=%c data=%d stop=%d\n",
1924           speed, parity, data_bits, stop_bits);
1925 #endif
1926     tcgetattr (fd, &tty);

1928     switch(speed) {
1929     case 50:
1930         spd = B50;

```

```

1931         break;
1932     case 75:
1933         spd = B75;
1934         break;
1935     case 300:
1936         spd = B300;
1937         break;
1938     case 600:
1939         spd = B600;
1940         break;
1941     case 1200:
1942         spd = B1200;
1943         break;
1944     case 2400:
1945         spd = B2400;
1946         break;
1947     case 4800:
1948         spd = B4800;
1949         break;
1950     case 9600:
1951         spd = B9600;
1952         break;
1953     case 19200:
1954         spd = B19200;
1955         break;
1956     case 38400:
1957         spd = B38400;
1958         break;
1959     case 57600:
1960         spd = B57600;
1961         break;
1962     default:
1963     case 115200:
1964         spd = B115200;
1965         break;
1966     }

1968     cfsetispeed(&tty, spd);
1969     cfsetospeed(&tty, spd);

1971     tty.c_iflag &= ~(IGNBRK|BRKINT|PARMRK|ISTRIP
1972                   |INLCR|IGNCR|ICRNL|IXON);
1973     tty.c_oflag &= ~OPOST; /* no output mangling of raw serial stream */
1974     tty.c_lflag &= ~(ECHO|ECHONL|ICANON|IEXTEN|ISIG);
1975     tty.c_cflag &= ~(CSIZE|PARENB|PARODD|CRTSCTS|CSTOPB);
1976     switch(data_bits) {
1977     default:
1978     case 8:
1979         tty.c_cflag |= CS8;
1980         break;
1981     case 7:
1982         tty.c_cflag |= CS7;
1983         break;
1984     case 6:
1985         tty.c_cflag |= CS6;
1986         break;
1987     case 5:
1988         tty.c_cflag |= CS5;
1989         break;
1990     }
1991     switch(parity) {
1992     default:
1993     case 'N':
1994         break;
1995     case 'E':
1996         tty.c_cflag |= PARENB;

```

```
1997         break;
1998     case 'O':
1999         tty.c_cflag |= PARENB | PARODD;
2000         break;
2001     }
2002     if (stop_bits == 2)
2003         tty.c_cflag |= CSTOPB;
2004
2005     tcsetattr (fd, TCSANOW, &tty);
2006 }
```

unchanged_portion_omitted_

```
2116 #endif /* __linux__ */
2045 #else
2046 static CharDriverState *qemu_chr_open_pty(void)
2047 {
2048     return NULL;
2049 }
2050 #endif
```

```
2118 #endif /* !defined(_WIN32) */
```

```
2120 #ifdef _WIN32
2121 typedef struct {
2122     CharDriverState *chr;
2123     int max_size;
2124     HANDLE hcom, hrecv, hsend;
2125     OVERLAPPED orecv, osend;
2126     BOOL fpipe;
2127     DWORD len;
2128 } WinCharState;
```

unchanged_portion_omitted_