64-bit Linux Device Driver Porting

John A. Ronciak
Staff Engineer
Intel Corp.

January 21 - 25, 2002

Agenda

- IA-32 Linux Device Drivers
- Make the Device Driver 64-bit Code Clean
- Building a 64-bit Linux Device Driver
- Debugging a 64-bit Linux Device Driver
- Code Examples (Case Studies)
- Additional Information
- Summary

Current Device Drivers

- Use a current IA-32 known working driver
- Fully test the IA-32 driver and device on known working IA-32 hardware
  - Known working tools
  - Everybody understands IA-32 op-codes (assembly code)
  - More machines around for development use (more resources)

Current Device Drivers

- System Hardware Requirements:
  - IA-32 Machine
    - 550 MHz Intel® Pentium® III Processor
    - 128MB RAM
    - 1 4-GB Hard Drive

IA-32 Linux Device Drivers

Linux Environment

- Standard Linux install
  - Linux Distribution of your Choice
- Test the install for functionality
  - Need to know things are working
- Build your own Kernel
  - Test again

Standard Kernel Build

- Get a Standard Kernel (www.kernel.org)
- Get the Matching Kernel Debugger (oss.sgi.com/projects/kdb)
- Apply kdb Patch to New Kernel
- Build New Kernel
- Test New Kernel
- Test Debugger Functionality
- Test your Device Driver

Agenda

- IA-32 Linux Device Drivers
- Make the Device Driver 64-bit Code Clean
- Building a 64-bit Linux Device Driver
- Debugging a 64-bit Linux Device Driver
- Code Examples (Case Studies)
- Additional Information
- Summary

64-bit Code Clean

Source Code Cleaning

- Generic 64-bit Coding Considerations
  - Data Models
  - Data Types
  - Pointers
  - Compiler Switches
- 64-bit Driver Source Cleanup
- Regression Test “Cleaned” Driver on IA-32

Easy Porting Procedures

64-bit Code Clean

Generic IA-64 Considerations

- Data Models
  - Uniform Data Model
    - Same source runs on IA-32 and 64-bit systems
  - LP64 Data Model
    - longs and pointers are 64-bit
    - ints remain 32-bit
    - IA-32 Systems are ILP32

Easy Porting Procedures

64-bit Code Clean

Generic 64-bit Considerations

- Data Types
  - New Data Types
    - int64_t and uint64_t
    - int32_t and uint32_t
    - int16_t and uint16_t
    - int8_t and uint8_t

Easy Porting Procedures

64-bit Code Clean

Generic 64-bit Considerations

- Pointers
  - All Pointers will be system defined
    - i.e. 32-bit pointers for IA-32, 64-bit for IA-64
- Do not cast pointers to int's or long's
  - Write "clean" C code

Easy Porting Procedures

64-bit Code Clean

64-bit Driver Cleaning

- Use the new data types
  - IA-32 ulong's to uint32_t
    - long's are now 64-bit
  - ushort's to uint16_t
  - Check all hardware specific types and convert them accordingly

Easy Porting Procedures

Agenda

- IA-32 Linux Device Drivers
- Make the Device Driver 64-bit Code Clean
- Building a 64-bit Linux Device Driver
- Debugging a 64-bit Linux Device Driver
- Code Examples (Case Studies)
- Additional Information
- Summary

64-bit System

64-bit Linux Install

- Install 64-bit Linux Distribution of your choice
- Configure the System as Needed
- menuconfig Options (xconfig)
  - Processor Type must be Itanium (normally set)
  - Enable the Kernel debugger
    - Integrated into most Distributions
- Build the New Kernel

Building the 64-bit System

64-bit Linux System

- Reboot the New 64-bit Linux System
- Put the IA-32 Device Driver Source on the 64-bit System
- Compile both the Linux Kernel and Driver on the 64-bit System with the Native Tools
  - May need to Static Link into the Kernel
  - Pay Attention to Any and All Warnings

Ready for Porting Now

The New Driver

- Reboot the System with the New Linux Kernel
- Test the New Driver
  - Start with basic functionality
  - Add more and more stress to the driver

Ready for Porting Now

The 64-bit Driver

Uniform Data Model Driver

- After Validation on 64-bit Platforms
  - Regression Test the Driver on IA-32
  - Remember Uniform Data Model
  - Same Driver Must Work on Both Archs

Easy Porting Procedures

Milestones so far

- Driver Cleanly Compiles for 64-bit Linux
- Driver Source is validated on IA-32
- Driver is integrated into 64-bit Linux kernel
- Loadable Module support as needed

Ready for Porting Now

Debugging the Driver

Linux Debug Tools

- Standard Printk's still work
  - Be careful with 64-bit pointers
- Debugger Built into the Linux Kernel
  - It's able to:
    - Dump and modify memory
    - Disassemble code
    - Set breakpoints

Get an Early Start, Don't Wait!

Debugging the Driver

Kernel Debugger

- Some Debugger commands:
  - *md* and *mds* displays memory
    - kdb\kdb_md.html
  - *mm* modifies memory
    - kdb\kdb_mm.html
  - *id* disassembles code
    - kdb\kdb_bp.html
  - *rd* and *rm* diplays and modifies registers
    - kdb\kdb_rd.html

Get an Early Start, Don't Wait!

Debugging the Driver

Kernel Debugger (cont.)

- *bt* and *btp* stack backtrace and for pid
  - kdb\kdb_bt.html
- *bp(bph)*, *bl*, *bpa*, *bc*, *be*, *bd* set and manipulate breakpoints
  - kdb\kdb_bp.html
- *ss* and *ssb* single step and step to branch
  - kdb\kdb_ss.html
- *env* and *set* show and set env variables
  - kdb\kdb_env.html

Get an Early Start, Don't Wait!

Debugging the Driver

Kernel Debugger (cont.)

- *cpu* switch to new cpu
- *reboot* will immediately reboot machine
- *go* continues execution
- *ef* display exception frame

Debugging the Driver

Kernel Debugger (cont.)

- Use the Debugger
  - Saves development time
  - Use to test error/unused code paths
  - Enabled for panic backtraces
  - Use to check for correct types

Get an Early Start, Don’t Wait!

Agenda

- IA-32 Linux Device Drivers
- Make the Device Driver 64-bit Code Clean
- Building a 64-bit Linux Device Driver
- Debugging a 64-bit Linux Device Driver
- Code Examples (Case Studies)
- Additional Information
- Summary

Case Studies

Code Examples

- Structure packing
  - Compiler generates codes with naturally aligned boundaries.
- Structures use hard-coded size for padding

```
Struct Buffer {
        void   *Ptr[10];
        char     Padding[88];
        }
```

To pad data to 128-byte chunk (4*10 + 88) in IA-32

Get an Early Start, Don’t Wait!

Case Studies

Code Examples (cont.)

- Structures Padding (cont.)
  - Pointer is 8-bytes in 64-bit code! Better to use

    Char Padding[128 - (10 * sizeof(void *))]
- Performance impact
  - due to padding error

Get an Early Start, Don't Wait!

Case Studies

Code Examples (cont.)

- Don’t use Unaligned Data Accesses in the kernel
- IT IS NOT ALLOWED!!!
  - It will cause a panic!
  - Check to make sure your data is aligned!
  - It is allowed in user space but will impact performance and should still be avoided

Get an Early Start, Don’t Wait!

Case Studies

Code Examples (cont.)

```
Struct  {
            ulong  Space;
            void   *Buffer;
            uint   Offset;
            ulong  Length;
} IoBlock;
```

- **Is there any thing wrong in the example above? Will this work in 64-bit Linux?**

Get an Early Start, Don't Wait!

Case Studies

Code Examples (cont.)

```
typedef struct _cb_header_t {
    ushort     cb_status;          /* Command Block Status*/
    ushort     cb_cmd;             /* Command Block Command*/
    ulong      cb_lnk_ptr;         /* Link To Next CB*/
} cb_header_t, *pcb_header_t __attribute__ ((__packed__));
```

New IA-64 structure:

```
typedef struct _cb_header_t {
    uint16_t     cb_status;          /* Command Block Status*/
    uint16_t     cb_cmd;             /* Command Block Command*/
    uint32_t     cb_lnk_ptr;         /* Link To Next CB*/
} cb_header_t, *pcb_header_t __attribute__ ((__packed__));
```

Get an Early Start, Don’t Wait!

Case Studies

Code Examples (cont.)

From:

```
printk( "rfpd = 0x%x, rfd_status=0x%x\n", rfdp, rfd_status );
printk( "     cb_cmd = 0x%x, cb_lnk_ptr = 0x%x\n",
        rfdp->rfd_header.cb_cmd, rfdp->rfd_header.cb_lnk_ptr );
```

to:

```
printk( "rfpd = 0x%p, rfd_status=0x%x\n", rfdp, rfd_status );
printk( "     cb_cmd = 0x%x, cb_lnk_ptr = 0x%x\n",
        rfdp->rfd_header.cb_cmd,
        (uint32_t)rfdp->rfd_header.cb_lnk_ptr );
```

Get an Early Start, Don’t Wait!

Case Studies

Code Examples (cont.)

From:

```
#define E1000_READ_REG(reg)    \
    ((Adapter->MacType >= MAC_LIVENGOOD)? \
    readl(&((PE1000_REGISTERS)Adapter->HardwareVirtualAddress)->reg) : \
    readl(&((POLD_REGISTERS)Adapter->HardwareVirtualAddress)->reg))
```

To:

```
#define E1000_READ_REG(reg)    \
    ((Adapter->MacType >= MAC_LIVENGOOD)? \
    readl((unsigned long) \
            &((PE1000_REGISTERS)Adapter->HardwareVirtualAddress)->reg) : \
    readl((unsigned long) \
            &((POLD_REGISTERS)Adapter->HardwareVirtualAddress)->reg))
```

Get an Early Start, Don’t Wait!

Case Studies

Code Examples (cont.)

- Be Aware of OS I/F Changes
- In network drivers, make sure the SKB that is handed up the stack is aligned. The IP layer will panic if the data in the SKB is not aligned using the skb_reserve command. This could require a copy if your hardware can't DMA to byte alignment
- Spinlocks are not changed and operate just like IA-32. Don't change them

Get an Early Start, Don't Wait!

Agenda

- IA-32 Linux Device Drivers
- Make the Device Driver 64-bit Code Clean
- Building a 64-bit Linux Device Driver
- Debugging a 64-bit Linux Device Driver
- Code Examples (Case Studies)
- Additional Information
- Summary

Additional Information

- Useful URLs
- SDV preparation for 64-bit Linux installation
- Installing Turbo Linux

Useful URLs and References

- Intel® Itanium® Processor infomation from Intel
  - http://developer.intel.com/design/itanium/index.htm
- For kernel debugger information on (IA-32 and 64-bit Linux)
  - http://oss.sgi.com/projects/kdb/
- The Official Home Page of 64-bit Linux
  - http://www.linuxia64.org/

Useful URLs Continued

- Where is the latest kernel source?
  - http://www.kernel.org/pub/linux/kernel/ia64/
- GNUPro tools for 64-bit Linux
  - http://www.redhat.com/software/tools/gnupro/
- First Released Distibution – TurboLinux
  - ftp://ftp.turbolinux.com/pub/ia64/
  - ftp://ftp.redhat.com/pub/redhat/redhat-7.2-en/os/ia64/

Boot From CD-ROM and Installing Linux

- At EFI prompt switch to appropriate device
  - For instance FS3: (this is your CD-ROM)
    - (assuming 2 hard disks)
- Begin installation process
  - elilo linux (typical)
    - LS-120 = /dev/hda
    - CD-ROM = /dev/hdc

Installing Linux cont…

- FDISK 3 Partitions (if not using auto)
  - Partition 1, 300M (FAT 16)
  - Partition 2, 2000+M (Linux)
  - Partition 3, 1024M (Linux Swap)

  NOTE: These are typical and you can make them whatever you want. The Swap space should be set to twice the amount of memory you have in the system.

Agenda

- IA-32 Linux Device Drivers
- Make the Device Driver 64-bit Code Clean
- Building a 64-bit Linux Device Driver
- Debugging a 64-bit Linux Device Driver
- Code Examples (Case Studies)
- Additional Information
- Summary

In Summary

- Porting Tools are Available Now
- The Porting Procedures are Easy
- Ready for the Porting Process Now
- Get an Early Start, Don’t Wait