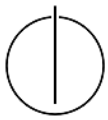
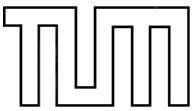


TECHNICAL UNIVERSITY
OF MUNICH
Department of Informatics



System development project:

Porting OpenBSD to the Xbox

July 17th, 2007

Markus Ritzer
ritzer@in.tum.de

Abstract

The aim of this project is to port the Unix-like operating system OpenBSD to a new platform, the Microsoft Xbox gaming console. This document describes the steps that need to be done to modify the OpenBSD source code in order to run on the console. All relevant bugs of the Xbox are mentioned. As a first step, the free BIOS replacement Cromwell is put into the EEPROM of the Xbox. Cromwell can boot the newly created OpenBSD kernel on the Xbox. After all, the user can login and work on the Xbox as on a standard PC.

Contents

Chapter 1

Introduction

In Juli 2006, Michael Steil, the founder of the Xbox Linux project suggested to port the operating system (OS) OpenBSD to the Microsoft Xbox platform. Several other operating systems are currently running on it:

- Microsoft Windows 2000 - the standard Xbox OS is a stripped-down version of the MS Windows 2000 kernel
- Linux - this was the first OS that has been ported to the Xbox
- FreeBSD - ported by Rink Springer
- NetBSD - ported by Jared McNeill and Andrew Gillham, since January 2007
- Darwin (the Mac OS foundation) - also ported by Michael Steil
- ReactOS - a free Windows NT clone

1.1 Overview over the Xbox

The Xbox is a game console produced by Microsoft. The production period was 2002 until the end of 2006, when its successor, the Xbox 360 was released. Over 24 million units have been sold world wide. It uses mostly standard PC hardware componentes, like a Pentium III processor, a nVidia nForce2 chipset and a nVidia GeForce 3MX graphics adapter. However, there a some little changes, so other i386 OSes don't run out-of-the-box and have to be modified slightly.

Figure 1.3 gives a detailed overview over the Xbox hardware. The SMBus has a PIC controller that controlles the front LED, the power and eject button. This device is always running, even if the Xbox is powered off. The video encoder produces a PAL/NTSC TV signal in (SDTV or HDTV), it replaces the RAMDAC memory of the graphics card.

The four USB connectors at the front of the Xbox are mechanically modified, but electrically compatible to standard USB (1.1) ports. So standard USB componentes can be connected to the Xbox via an adapter.

A standard 100Mbps ethernet adapter is integrated in the nForce chipset.

Figure 1.1: Xbox front view: DVD tray (with Xbox font on it), four USB 1.1 ports, power button (below) and the eject DVD button (with the LED surrounding it)

Figure 1.2: Xbox rear view: power supply connector, cooling fan, A/V connector, ethernet port

The primary IDE channel has a standard IDE eight or ten gigabyte harddrive connected, the DVD drive is slightly modified (different power supply connector and eject button handling).
A GeForce 3MX chip produces graphical output that gets encoded to a valid TV signal with the Conexant/Focus/Xcalibur (depending on the hardware version of the Xbox) video encoder on the SMBus.
A detailed hardware overview can be found at [1] and on figure 1.3.

1.2 Overview over OpenBSD

OpenBSD is a Unix-like computer operating system descended from Berkeley Software Distribution (BSD), a Unix derivative developed at the University of California, Berkeley. It was forked from NetBSD by project leader Theo de Raadt in late 1995. The project is widely known for the developers' insistence on open source code and quality documentation; uncompromising position on software licensing; and focus on security and code correctness.
It includes a number of security features absent or optional in other operating systems and has a tradition of developers auditing the source code for software bugs and security problems. The project maintains strict policies on licensing and prefers the open source BSD license and its variants — in the past this has led to a comprehensive license audit and moves to remove or replace code under

```
Xbox V1.0-1.5
|-- Intel Pentium III 733MHz, 128KB cache
|-- Flash ROM
\-- PCI bus
    |-- Programmable Interrupt Controller
    |-- System Timer
    |-- DMA Controller
    |-- PCI Bridge Device - Host Bridge
    |-- Memory Controller - SDRAM - Samsung
    |-- HUB Interface - ISA Bridge
    |-- SMBus Controller
    |   |-- PIC16LC
    |   |-- Conexant 25871/Focus 454/Xcalibur Video Encoder
    |   |-- ADM1032 System Temperature Monitor
    |   \-- Serial EEPROM
    |-- OHCI USB Controller
    |   \-- USB HUB (Controller Ports 1 & 2)
    |-- OHCI USB Controller
    |   \-- USB HUB (Controller Ports 3 & 4)
    |-- MCP-X V3 Southbridge
    |   |-- MCP Networking Adapter
    |   \-- MCP APU
    |-- AC‘97 Audio Codec Interface
    |-- IDE Controller
    |   |-- Primary IDE Channel
    |       |-- Seagate 10.2GB (8GB used)/Custom WD 8GB HDD
    |       \-- Custom Samsung/Phillips/Thompson/LG DVD Drive
    |-- PCI Bridge
    \-- AGP Host to PCI Bridge
        \-- NV2A GeForce 3MX Integrated GPU/Northbridge 233MHz
```

Figure 1.3: Xbox hardware overview graph

licenses found less acceptable.
As with most other BSD-based operating systems, the OpenBSD kernel and userland programs, such as the shell and common tools like `cat` and `ps`, are developed together in a single source repository. Third-party software is available as binary packages or may be built from source using the ports collection.
OpenBSD currently runs on 17 different hardware platforms, including the DEC Alpha, Intel i386, Hewlett-Packard PA-RISC, AMD AMD64 and Motorola 68000 processors, Apple's PowerPC machines, Sun SPARC and SPARC64-based computers, the VAX and the Sharp Zaurus. The Xbox is now the next platform it runs on. More information can be found at [2].

1.3 Requirements

An unmodified Xbox starts the proprietary Microsoft code in the ROM on poweron. This code loads the dashboard, the GUI of the Xbox. In order to run

own code at the Xbox, one has to replace that code. This can be achieved in two ways: flash the TSOP or insert a modchip. In both choices opening the Xbox and soldering is required. The result is that one can bring own code into the Xbox. In this case, Cromwell 2.41-dev is used. Cromwell is a special Xbox replacement BIOS, it gets executed immediately after poweron. Cromwell is able to boot a kernel (e.g. in ELF format) from CD/DVD, the HDD oder via network.
Another possibility is to use a so-called "soft-modded" Xbox, that means, no hardware modifications need bo done. Cromwell is loaded as an executable file in Xbox format, then it is called "Xromwell" to distinguish it. Details can be found at [3] and [4].

1.4 Approach

The porter of an operating system needs a "build-box", a machine where he can compile a kernel in order to test it. In this case, a virtual VMware machine (with OpenBSD 4.0-current installed) running on a Gentoo Linux host PC was used. Connected via SSH to the virtual machine, the porter can modify the source code, compile a kernel and copy it to the Gentoo Linux PC that is connected with an ethernet cross-cable to the Xbox. DHCP and TFTP deamons must be set up in order to deliver the kernel to the Xbox. Cromwell boots this kernel on the Xbox. In the development phase, the front LED of the Xbox and later the TV were used to find out critical sections that made the kernel crash.

1.5 Homepage

This project may be interesting to lots of people in the open source community so it makes sense to publish information about it on the web. All interested persons (especially OpenBSD developers and Xbox Linux users) can see the progress, the advantages and the problems of the project. Some people contacted the author of this document in order to help with detailed knowledge of the OpenBSD kernel. The homepage can be found at [5]. After the announcement of this project on several mailing lists, there were over two thousand visitors in the first three days.

Chapter 2

Porting procedure

It is wise to get the latest developer snapshot of the OpenBSD source code. This increases the chances that this port becomes an official one. The first thing to do is to geht the sources via CVS.

2.1 The OpenBSD boot process

In order to boot a kernel, the kernel must be put into memory at first. This is accomplished by the boot loader, Cromwell in this case. It gets the kernel image from CD/DVD, hard disk or via DHCP/TFTP. Then it places it in the memory, starting at zero. After that, the kernel gets executed from its entry point.
The first file of the kernel that gets executed is `arch/i386/i386/locore.s`, written in x86 assembly language. Its purpose is (among other things) to determine the CPU type.
After all this low-level initialization, the first C code, the function `init386()` in `arch/i386/i386/machdep.c` gets invoked. A lot of initialization is done here, and code was added to determine if the machine is a Xbox and if true how much memory it has. Some code has to be skipped (especially BIOS specific things; the Xbox doesn't have a BIOS) that would make the kernel crash.

2.2 Memory management

Another important thing done there is loading the memory. The function `uvm_page_physload()` adds the memory to a list, which is needed for memory management. It is crucial that the first few megabytes of the memory are skipped, because this is where the kernel image lies. It would be better to get an information from Cromwell what the first free page is that can be used, but for simplicity, a fixed value of five megabytes is hardcoded here. The top four megabytes of the memory are reserved for the framebuffer that gets mapped there. So the significant part looks like figure 2.1:

```
#define SKIP 0x500000
physmem = atop((xbox_memsize*1024*1024) - FB_SIZE - SKIP);
dumpmem[0].start=atop(SKIP);
dumpmem[0].end=atop((xbox_memsize*1024*1024) - FB_SIZE);
uvm_page_physload(atop(0),atop(xbox_memsize*1024*1024),
        dumpmem[0].start,dumpmem[0].end,VM_FREELIST_DEFAULT);
```

Figure 2.1: Listing: memory management

2.3 The autoconf system

After this, the boot code continues in `main()` in `kern/init_main.c`. Again, a lot of initialization is done, for example of the virtual memory. The function `cpu_configure()` gets called. OpenBSD manages all its device drivers in a tree structure. `cpu_configure()` creates a fictive "root" node, where all the device drivers of the system get attached to. Every driver has to provide a `match()` function, which determines if a driver and a hardware device fit together, and an `attach()` function, that attaches this hardware driver to the autoconf tree. A `detach()` function can remove the driver lateron. The kernel gets the names of the functions from a `cfattach` structure (see listing 2.3).

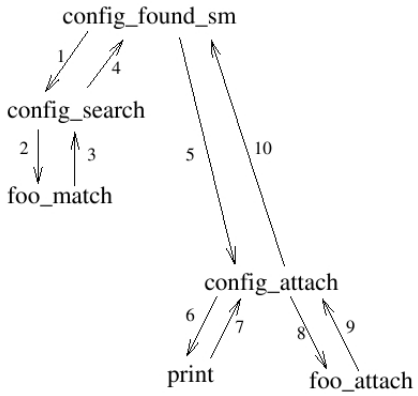


Figure 2.2: autoconf calling chain in direct configuration (e.g. on the PCI bus)

```
struct cfattach xboxfb_ca = {
    sizeof(struct xboxfb_softc),
    xboxfb_match,
    xboxfb_attach,
    NULL  /* detach */
};
```

Figure 2.3: Listing: the cfattach structure for the framebuffer driver

2.4 The framebuffer driver

Cromwell sets up a screen with 640x480 pixels and 32 bit color depth. In order to do so, the corresponding values are sent to the video encoder of the Xbox. This means an amount of

$$640 \cdot 480 \cdot \frac{32bit}{8\frac{bit}{byte}} = 1228800\ Bytes$$

of the physical memory needs to be accessed in the right way so there is output on the TV. Before this space can be used by a framebuffer driver, it needs to be mapped with `bus_space_map()`. The mapping must start at 4 MB below the total size of the main memory, usually at 60 MB. After this, the memory can be accessed and modified by standard C pointers. Listing 2.4 shows how the space is mapped and the screen is cleared. The address is ORed with `0xf0000000`, because the space is mirrored there with caching disabled.

```
#define XBOX_RAM_SIZE (xbox_memsize * 1024 * 1024)
#define XBOX_FB_SIZE  (0x400000)
#define XBOX_FB_START (0xf0000000|(XBOX_RAM_SIZE - XBOX_FB_SIZE))

bus_space_map(I386_BUS_SPACE_MEM, XBOX_FB_START, XBOX_FB_SIZE,
                                        0, &xboxfb_console_memh);
bzero((void*)xboxfb_console_memh, XBOX_FB_SIZE);
```

Figure 2.4: Listing: mapping the space for the framebuffer

2.5 Console integration

With the code above, everything is prepared to put text on the TV. For debugging, the author implemented a `putchar()` and a `putstring()` function, that was able to draw characters pixel by pixel. But code for this is already available in the `rasops` framework. So the next thing to implement was a driver for the console. OpenBSD's `main()` function initializes consoles by calling `consinit()`, which again calls `wscninit()` in `arch/i386/i386/wscons_machdep.c`. This code calls the `cnattach()` function a framebuffer driver, which does the mapping of the memory mentioned above and sets up a `struct rasops_info` that holds information about the screen size, color depth and so on (see listing 2.5). On the Xbox, `consinit()` must be called after the initialization of the virtual memory and before the call to `cpu_configure()`. Otherwise, `bus_space_map()` wouldn't be available or the memory wouldn't be ready.
The `attach()` function of the framebuffer driver sets up the `struct wsdisplay _accessops`, a structure that holds pointers to screen access functions. One of these is the function `xboxfb_alloc_screen()`, which handles the screen initialization.
The function `xboxfb_attach()` ends with the call to `config_found()`.
The console driver does not support screen switching or scrollback, as there is now virtual console framework in OpenBSD. According to Miod Vallet, it is planned to develop such a framework in the near future.

```
struct rasops_info *ri = &dc->dc_rinfo;

ri->ri_depth = 32;
ri->ri_bits = (void *)xboxfb_console_memh;
ri->ri_width = 640;
ri->ri_height = 480;
ri->ri_stride = ri->ri_width * ri->ri_depth / 8;
```

Figure 2.5: Listing: set up the `rasops_info` structure in `xboxfb_cnattach()`

2.6 Mounting the root filesystem

One of the last things done in `main()` is mounting the root filesystem. There are several possibilities to do so: prepare a hard disk with a valid filesystem on a regular OpenBSD computer, use a ramdisk, mount it from CD or via NFS. For simplicity, the first method was used. The hard disk was prepared on the "build-box" and inserted in the Xbox afterwards.
Usually, the BIOS tries to find all devices connected on the IDE bus and creates checksums of these devices. Those checksums are passed to the OpenBSD kernel, and so it knows what disk to use for the root filesystem. However, without a BIOS, the kernel has no such checksums and can't locate the disk.
The solution to this problem is to hardcode the name of the device with the root filesystem in the kernel configuration. The line looks like figure 2.6.

```
config bsd root on wd0a swap on wd0b
```

Figure 2.6: Listing: kernel configuration

With this configuration, the kernel searches the root filesystem on the device `wd0a`, and the swap filesystem on `wd0b` (which are prevalent names for OpenBSD disklabels). After mounting, the `init` process is started, and a login shell is opened. Using the credentials stored on the disk before, a user can login to the system now.

2.7 Timer frequency

The system timer inside the "8254 PIT" unit of the chipset on a PC has a base frequency of 1.193182 MHz. The number of task switches, the system clock and all timing for multimedia is usually based on the interrupts generated by the PIT unit. For some unknown reason, this base frequency is 1.125000 MHz on the Xbox, which is about 6% lower. This problem is handled in the file `arch/i386/isa/timerreg.h`.

2.8 PCI enumeration bug

There are three bugs on the Xbox PCI bus:

- The nonexistent memory controllers at 0:0.1 and 0:0.2 are completely broken: If probed, i.e. by reading out their PCI IDs, the Xbox freezes.
- There are ghost devices on bus 1, after the video controller (1:0.0). Garbage is returned on probing them.
- The same is true for the complete bus 2.

So the PCI bus intialization in `arch/i386/pci/pci_machdep.c` must skip these bus addresses.

```
#ifdef XBOX
    if (arch_i386_is_xbox) {
        if (busno == 1)
            return 1;
        else if (busno > 1)
            return 0;
    }
#endif
```

Figure 2.7: Listing: probe for only one device on PCI bus 1, skip PCI bus 2

```
#ifdef XBOX
    if (arch_i386_is_xbox) {
       int bus, dev, fn;
       pci_decompose_tag(pc, tag, &bus, &dev, &fn);
       if (bus == 0 && dev == 0 && (fn == 1 || fn == 2))
           return (pcireg_t)-1;
    }
#endif
```

Figure 2.8: Listing: don't probe the first and the second device on PCI bus 0

2.9 Network driver

The network interface card (NIC) of the Xbox is a standard nForce ethernet NIC. Usually, the driver `nfe` would be used, but it didn't work out-of-the-box. A few small changes from the NetBSD `nfe` and `icsphy` drivers have been integrated in this port in order to support networking.
There is also another problem: Cromwell has a flaw which causes the `nfe` driver to fail attaching to the NIC. This is because Cromwell leaves the NIC running; this will cause the nVidia driver to fail as the NIC does not return any sensible values and thus fails attaching. So a workaround is needed: the function `xbox_startup()` in `arch/i386/xbox/xbox.c` tells the NIC to stop whatever it's doing; this makes `nfe` attach correctly. As the NIC always resides at `0xfef00000-0xfef003ff` on an Xbox, this address is simply hardcoded. This code was taken from the FreeBSD port.

```
#define XBOX_NFORCE_NIC 0xfef00000

void
xbox_startup(void)
{
    bus_space_handle_t h;
    int rv;

    if (!arch_i386_is_xbox)
        return;

    rv = bus_space_map(I386_BUS_SPACE_MEM, XBOX_NFORCE_NIC,
                                            0x400, 0, &h);
    if (!rv) {
        bus_space_write_4(I386_BUS_SPACE_MEM,h,0x188,0);
        bus_space_unmap(I386_BUS_SPACE_MEM,h,0x400);
    }
}
```

Figure 2.9: Listing: disabling the network interface

2.10 Poweroff, reboot and the LED driver

The Xbox has a PIC16LC (PIC) device, which is connected through the SMBus. Its purpose is to control the two front buttons (power on/off and eject DVD tray), the DVD tray and the Xbox power supply. A driver for this was already available in the NetBSD port. Modifications were necessary because OpenBSD doesn't have the same framework for the integration of hardware monitoring sensors, so sensor related parts of code have been deleted. The driver implements the `match()` and `attach()` functions mentioned in section 2.3. Documentation on the PIC can be found at [6].

2.11 X-Server

The X-Server is a userspace program whose purpose is to provide a graphical user interface (GUI). It supports a driver called `wsfb` that works on an underlying framebuffer driver. However, this framebuffer driver must implement the functions `ioctl()` and `mmap()`. The function heads are in the `xboxfb` driver, but they are not implemented. So X cannot be run.

Chapter 3

Known errors and things that still could be done

There are several more things that could be done in order to make this port more complete.

- The CD/DVD drive is not found. Normally, it should work, but it isn't recognized. No problems like this are known on other OSs.
- The program `dmesg` fails to run with the message: `sysctl: KERN_MFGBUFSIZE: Device not configured`. A reason for this could be that kernel and userland are out of sync. The problem seems to be the magic number of the buffer. `0xffffffff` is passed as magic number, but `0x63061` is expected. So there are no kernel log messages.
- The framebuffer driver lacks `ioctl()` and `mmap()` functions. See section 2.11.
- The framebuffer driver lacks scrollback support. Users cannot scroll back in the console. The function header `xboxfb_scrollback()` is in the driver, but, however, it is not implemented. This could be solved when OpenBSD gains a virtual console framework.
- The framebuffer driver lacks screen switching capability. This means there is no `Alt+X` screen switching in the console. Like the item above, this could be solved when OpenBSD gains a virtual console framework.
- Eject button handling: The PIC takes care of the handling of the eject button. On the PlayStation, it was possible to trick the console into loading illegal copies by inserting an original, having the CD checked and quickly replacing it with a copy. The designers of the Xbox made this trick impossible by having an additional security chip, the SMC, which resets the system if the user presses the eject button or even tries to manually eject the tray.
  But the system does not always reset in this case. In the Xbox Dashboard, for example, it is possible to open the tray. Every time the user presses the eject button or the tray opens (for example if the drive ejects on request of software), the PIC sends an EXTSMI# interrupt to the CPU. If

the software running on the CPU handles this interrupt and sends back a message to the PIC, then the system will not be reset.
NetBSD has a interrupt handling subsystem on the SMBus which OpenBSD has not. So this bug cannot be fixed at the moment.

- Several drivers are still missing:
  - a driver for the serial eeprom (like NetBSD has one)
  - a driver for the gamepad
  - a driver to read the temperature sensors on the Xbox.
  - a filesystem driver to read and write the FATX filesystem of the Xbox.
- Kernel image size. Cromwell should tell the kernel, how big its image is. In other words: the kernel needs to know the address of the first free page of the memory, in order to set up memory management. See also subsection 2.2.
- For convenience reasons, there should be an installer to make it easier to get OpenBSD on the Xbox. A LiveCD for quick testing could be provided.
- The content of the project homepage [5] could be transferred into the Xbox-Linux homepage [7].
- This port should be tested on all hardware revisions of the Xbox.
- The framebuffer driver should set up the video mode itself and not rely on Cromwell.

Chapter 4

Conclusion

The author's opinion is that the project was succesful. There are still many things to be done, but the major parts are running. The kernel is booting, memory management is working, there's output on the TV, the root filesystem gets mounted and the user can login. On of the most important devices, the NIC, is working.
This project ends with a good starting points for others who are interested in the project and are willing to do the rest of the work. The inhibition threshold for others to complete the port should be very low, as the most annoying and time wasting problems are already solved.

Chapter 5

Thanks to

The author would like to thank all of the people that helped to do the port:

- Michael Steil, founder of the Xbox-Linux project; he had the idea for the project and lended his Xbox to the author.
- Dr. Christian Rehn, Technical University of Munich. He attended the project.
- David Gwynne, OpenBSD kernel developer; he answered lots of questions on the OpenBSD kernel.
- Rainer Giedat, OpenBSD professional, he has answered tons of questions.
- Jared McNeill, he has helped a lot and did the NetBSD port, from which some code was taken.
- Rink Springer, the same, he has ported FreeBSD.
- Tobias Schröpf, he owns the server where the homepage is stored at.
- Jason Wright, Marco Peereboom, Chris Demetriou, Matthieu Herrb, Miod Vallat: OpenBSD experts.
- Michael "Mickey" Shalayeff - OpenBSD developer; he helped with console integration.
- Ed Schouten, involved in the FreeBSD port, he had some ideas with the network driver.

Bibliography

[1] The Xbox Linux project: Xbox hardware overview. http://www.xbox-linux.org/wiki/Xbox_Hardware_Overview, viewed at 06/07/2007.

[2] The OpenBSD project homepage. http://www.openbsd.org, viewed at 06/07/2007.

[3] The Xbox Linux project: Hardware method HOWTO. http://www.xbox-linux.org/wiki/Hardware_Method_HOWTO, viewed at 06/07/2007.

[4] The Xbox Linux project: Software method HOWTO. http://www.xbox-linux.org/wiki/Software_Method_HOWTO, viewed at 06/07/2007.

[5] The OpenBSD on the Xbox project homepage. http://tobias.schroepf.de/doku/doku.php?id=xbox:porting_openbsd_to_the_xbox, viewed at 07/17/2007.

[6] The Xbox Linux project: PIC documenatation. http://www.xbox-linux.org/wiki/PIC, viewed at 06/12/2007.

[7] The Xbox Linux project homepage. http://www.xbox-linux.org, viewed at 06/07/2007.

[8] Wikipedia: OpenBSD. http://en.wikipedia.org/wiki/OpenBSD, viewed at 06/07/2007.

[9] NetBSD Driver Writing Guide. http://www.netbsd.org/docs/kernel/ddwg.html, viewed at 06/11/2007.