Learning from BeOS

by Alexandru Lazăr

Email: `std@cwazy.co.uk`

Web: `http:/donkey.50webs.com`

September 8, 2005

1 Introduction

The Be Operating System (BeOS in short) was an effort of Be, Inc. of developing a desktop-oriented operating system, covering Be's own platform (the BeBox) and, later, Mac and the x86 platform. BeOS was targeted towards multimedia applications, allowing for great performance in this domain. I remember running full-screen movies on my old 233 MHZ Pentium box, which was basically a sort of a dream for Linux or Windows. BeOS was ported to Mac at first, in the hope that it would encourage Apple to use it as a base for its future operating systems. NeXT was chosen though.

Despite being innovative in a few ways, and being a very clean and stable desktop platform, the Be Operating System never succeeded commercially. Despite attracting a pretty significant number of enthusiasts, It never ended up as a widely-used solution. Due to the commercial failure itself, Be, Inc. was eventually bought by Palm.

However, the Be OS didn't quite drop dead. The last version published by Be, BeOS 5, came in two versions for the x86: Personal and Professional edition. The Personal was edition was free. It would be installed just like any other application, from Windows. It would throw itself on a loopback image. To Windows, it appeared like an ordinary file, but you could actually boot it with a single click on a start menu item. The Professional edition wasn't free, but it could be installed like a full-fledged operating system. The difference wasn't necessarily huge, as long as you could still mount partitions with the Personal edition as well. After Be's vanishing, development was continued by the enthusiasts and the Personal Edition was updated a few times, bringing it closer to the present in terms of drivers and hardware support.

There are also other projects. Zeta tries to continue BeOS, while others (like Haiku) try to re-create it. Blue Eyed OS tried to port the BeOS API on top of X Window, but didn't quite succeed. At this moment, while BeOS is officially dead, enthusiasts still fiddle with it and new software is still released on BeBits, the largest repository of BeOS software.

Is there anything to be learned from Be's experience? The initial answer would probably be No. They're dead and buried and gone to Palm's heaven, but it's not really like that. BeOS itself gives a few ideas that could be implemented in other operating systems. There are obviously reasons why BeOS never got accepted as a serious platform, but, at the same time, there are things that shouldn't remain buried.

2 Learning the good things

I'd obviously start with the good parts of the BeOS. Not because I'm a fan of it, but because these are the most important when it comes to applying them in other places. There are more areas where BeOS was shining, and, despite being drawn back by areas where it didn't shine (read: it was totally black), they deserve mentioning first.

2.1 Booting

BeOS's boot process is to be remarked even for a single thing: it's quick.

By quick, I mean that on a 300-MHZ Pentium 2, it takes less than 20 seconds. Gentoo Linux boots in 20 seconds on my Pentium 4. It was rather impressive to see a desktop after 28 seconds of waiting after pressing the power button. Windows Me needed almost one minute, and I'd rather not say how much Red Hat Linux 9 needed.

How this speed is achieved – I must admit that I don't know. However, despite the fact that kernel architecture must have a good word to say, it's obviously not just that. It looks like a combination of a very fast and un-bloated kernel, well-written and optimized device drivers and a delay of starting services, in the way Windows does it. For example, mounting other filesystems except for the root filesystem seems to occur at the end of the boot phase, when the user is already seeing the desktop. Linux mounts it while the system is booting, and if there are more filesystems to mount (I have seven, including the swap), it takes a rather long time. The same applies for kernel modules.

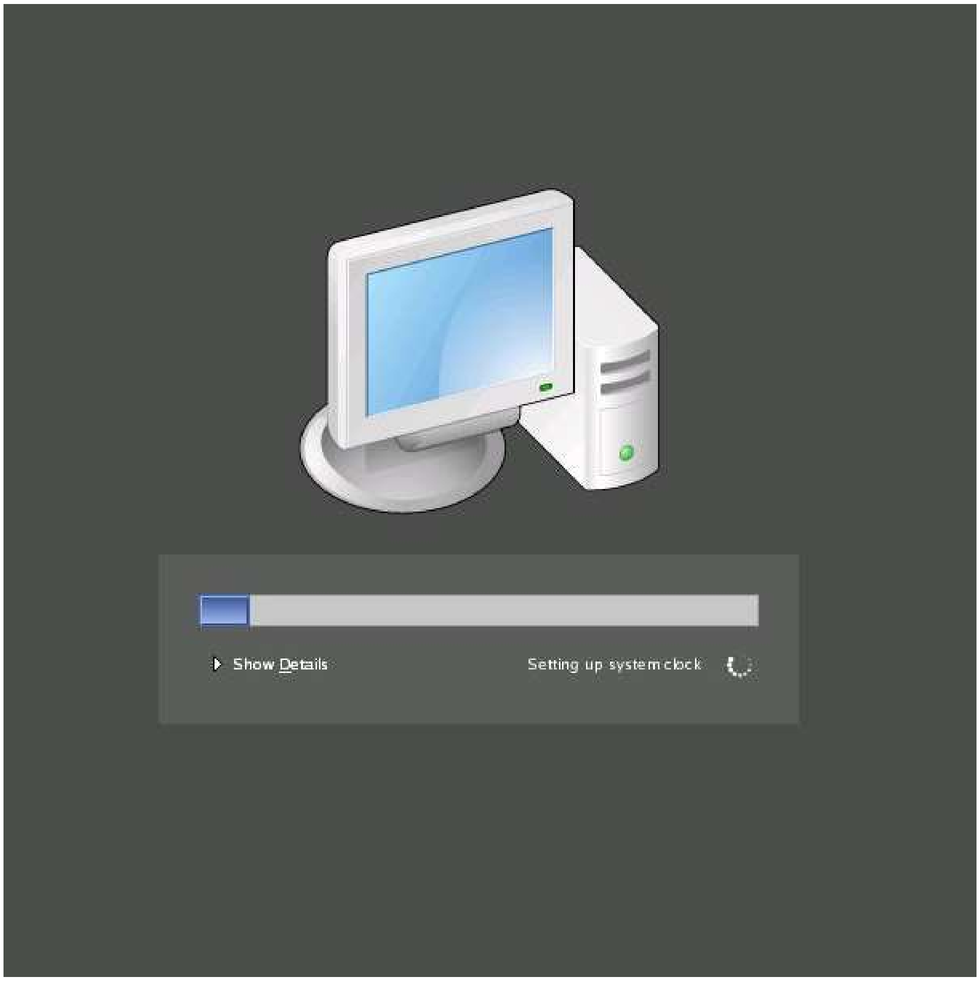
Figure 1. BeOS 5's boot screen

The boot splash screen was also well-thought. There are currently two extremes about how these look. The first one is the one Microsoft gives: a moving bar and a logo. You don't really know what's going on out there, and, in order to know, you must use special boot options. The other one is proposed by most Linux distributions: scrolling text that says everything.

Neither of these options work well for everyone. For the geeks out there, not even scrolling text is the best option. Sure, I don't mind seeing it, but to be honest, I don't care to see it too often either. After all, my computer does the same things every time.

BeOS has a different approach, somewhere in between. It doesn't tell you everything, but whenever it does something, it tells you what it does. This way, you know that it's trying to load hardware drivers or initialize the desktop. Should it encounter an error, it will display it, and, if it's fatal, it will also halt. This is a good way of showing what the system is doing in a nicely-themed way. You can configure some boot parameters from the boot screen, and that is done using a good keyboard-driven menu interface. More intuitive than editing Grub parameters I guess.

This solution is already being applied by Fedora. While I'm not really a fan of it, I must admit that their boot screen is well though. It's probably the least bloated part of the distribution itself.



Setting up system clock

Figure 2. Fedora's boot screen

I do want to make it clear that I'm not advocating for usage of boot screens. There are places where a nice boot splash just wouldn't fit. I wouldn't want one on my servers for example. But, wherever they are present, BeOS's example would be nice to follow.

Configuring boot parameters in a nicer way may not be as easy to do with Grub. Grub aims to be a more universal boot loader and the Linux kernel requires parameters to be passed immediately. BeOS, on the other hand, has its own boot loader, which is able to boot Windows at least. I heard it is more capable than that but I never had the chance to test it myself.

2.2 Configuration

In many Unix and Unix-like systems, configuration is rather difficult. Adding a new hardware driver often implies recompiling the kernel or loading a module, which is not quite intuitive for someone who doesn't know anything about what a module is and what it does in the kernel.

Control Panels are a good idea, but BeOS gives a different approach: instead of a Control Panel, it uses a Preferences menu. It's close to a Control Panel, only quicker to access.

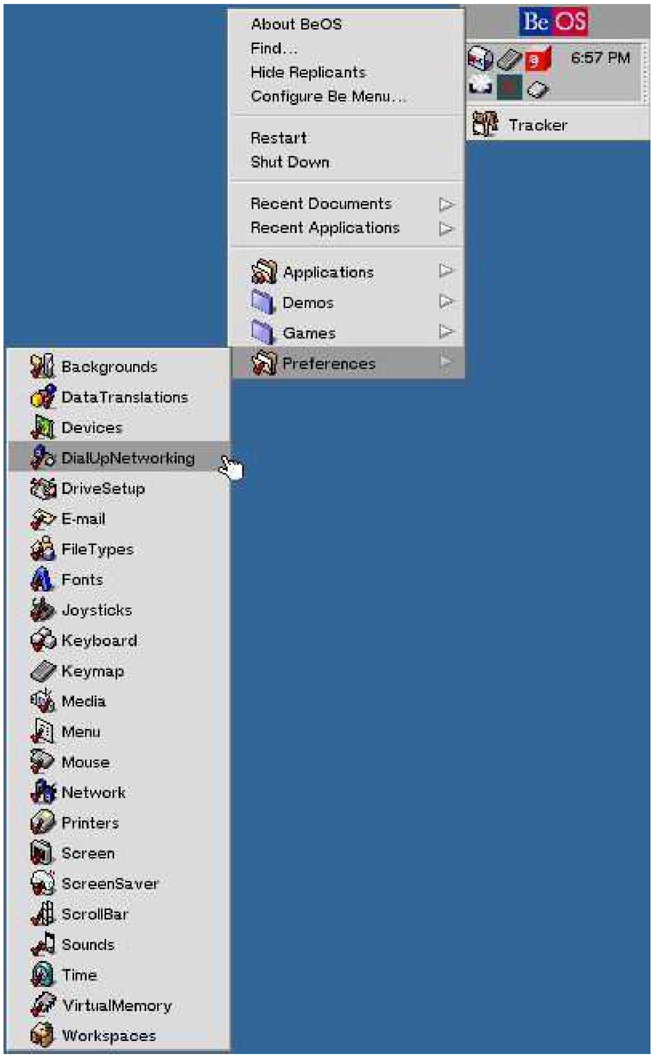


Figure 3. Preferences menu

Using this Preferences menu simply means that the user doesn't have to click Control Panel and choose everything in a new window. While not necessarily a great innovation, it is nevertheless a good improvement in terms of ergonomics over the traditional Control Panel. Gnome and Fluxbox are already using this. Windows and maybe KDE could learn a little bit.

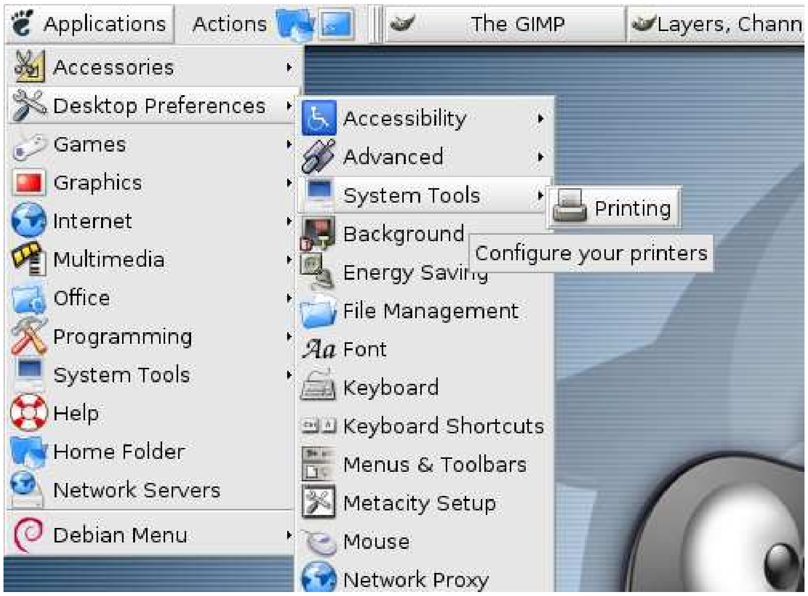
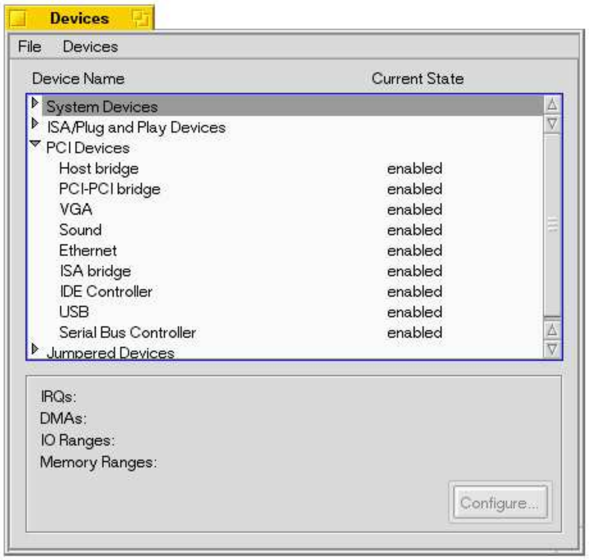


Figure 4. Gnome Desktop Preferences menu

However, what makes BeOS so much better when it comes to configuration is the Devices dialog. This menu allows for very easy access to basic configuration options for all devices. The more interesting part is that it was piece of cake for me to make BeOS detect a legacy, non-plug&play sound card I had. Configuring the IRQ and everything was very easy and it took me five minutes to have sound. On Linux, I needed about two hours and a few man pages.



Device Name	Current State
System Devices	
ISA/Plug and Play Devices	
PCI Devices	
Host bridge	enabled
PCI-PCI bridge	enabled
VGA	enabled
Sound	enabled
Ethernet	enabled
ISA bridge	enabled
IDE Controller	enabled
USB	enabled
Serial Bus Controller	enabled
Jumpered Devices	

IRQs:
DMAs:
IO Ranges:
Memory Ranges:

Figure 5. The Devices dialog

While this dialog is present, in some way or another, on many systems, the only serious implementation is on Windows, and the Add new hardware wizard ruins it all.

Configuration is general is very easy to do on the BeOS, and almost all aspects can be easily configured. One thing that appealed to me greatly is the fact that I didn't have to type my root password before making any change. Root password? Yes, that leads me to the next chapter.

2.3 BeOS is single-user

It may seem weird to you that I take this as a plus. But, well, I do, and for a simple reason: BeOS doesn't have to be multi-user.

BeOS, despite being a Unix-ish OS, is not supposed to do all the things Unix will do. It doesn't need to run XWindows for multiple clients, it doesn't need to run high-security webservers with a certain user's restricted privileges in a chroot jail and so on. It's a personal workstation and, as a personal workstation, it can do without multiple accounts.

This also saves administration hassles like maintaining multiple groups and users. The downside is that it also lets a less-experienced user to do system-wide mistakes. However, in a network environment, propper user education should make this less of a problem than it actually seems to be.

Despite seeming rather dumb to do in the era of the Internet, this approach is also taken by other operating systems. Windows is an example, but Windows is pseudo-multiuser (multiple accounts, but just one logged in at a time). MacOS does almost the same thing, but requires an administrator password to change some settings.

This also allows for a very interesting filesystem. While avoiding permissions hell, the filesystem works much like a database, making potential development of a desktop searching tool more interesting.

2.4 Desktop environment

BeOS provides a very clean desktop environment. While there are certain usability issues that I will discuss later here, there are a number of good things around.

The first thing that I'd like to bring to your attention is the great consistence of the desktop environment. You will not see hundreds of button shapes, flashy colors where it should have been grey and the like. In fact, the BeOS interface was intentionally designed so that it would be very hard to change its appearance, even with dirty hacks. Zeta did change this, but many users still prefer the original theme. And that is understandable.

BeOS boosts its own, distinguishable icon theme. You could easily recognize a BeOS icon if you saw one. They have an elegant, pixel art-like look, and despite the fact they are not antialiased, they look well.

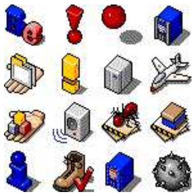


Figure 6. BeOS icons

Consistency remains dominant here as well: applications use the same icon style for their toolbars.

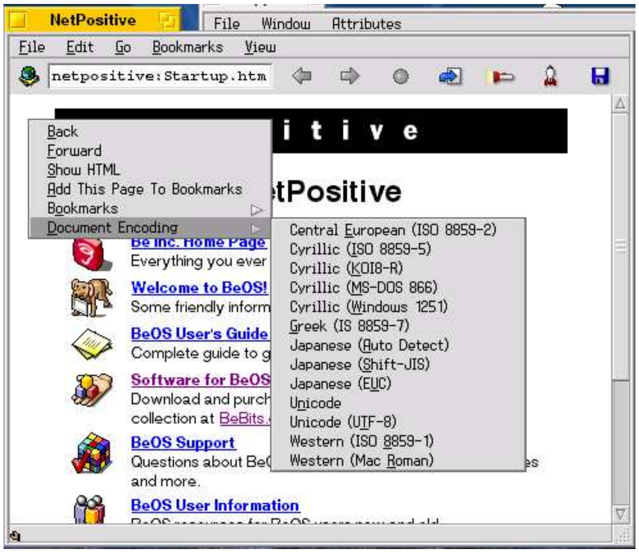


Figure 7. Applications (NetPositive here) use the same icon style for their toolbars.

In the screenshot above you can also see an interesting feature of the BeOS UI: the title bars are tab-like instead of being as long as the window. This saves a significant amount of screen space and, in terms of usability, it makes a good compromise. While the Maximize button is harder to access (or at least this is what Fitts's laws would say), the Close button remains in place, since the user is expected to press the Close button more often than Maximize. CTWM (a window manager) does that, but it's rather archaic. There are such themes for KDE, but none for Gnome or many other window managers.

Another interesting fact about the BeOS desktop is the placement of the taskbar. Well, sort of.

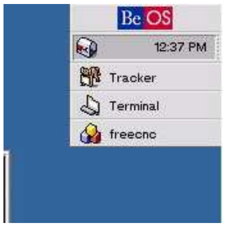


Figure 8. Task list & comp.

There is no taskbar. There is a task list, but it's not a bar. The things you see on the screen are handled by a special application called the Tracker. It's the Tracker's job to manage icons, windows and everything else. Aside from that, there is a menu in the upper-right, which works like Windows' Start Menu. Underneath it is the system tray, and then, one by one, a list of tasks. The list is not fixed size (it's not a bar); when a new item is added, it expands.

This is pretty different from the usual concept of a taskbar. However, this solves a few problems. First of all, the task list is much cleaner. Traditional taskbars are confusing because they often include a start menu, a quick launch bar, a window list and a system tray on the same line, without any clear separation between them except for the way they look. BeOS organizes these so accessing them is easier. Arranging the task list items vertically solves another problem. These items are long but very thin. While the length of the screen wouldn't allow for more than four or five of them without starting to clutter, you could arrange a few dozens of them if you put them vertically. This way, it's close to impossible to end up having a cluttered task list with truncated window names that make it impossible to know what task is really hiding under a button.

This method is not very popular, despite being more efficient than the traditional way. While most desktop environments allow the taskbar to be moved, most of them are very inefficient because the bar still remains fixed size, and wastes even more screen space.

2.5 Development

One thing BeOS has shown is that it's not entirely impossible to have a simple API for application programmers to use. Windows has often been criticized for its horrible Win32 API. As a simple example, consider the following program:

Example 1. Hello, world!

```
   //HelloView.h
   #pragma once
   #include ⟨StringView.h⟩
   class HelloView : public BStringView {
   public:
      HelloView(BRect frame, const char* name, const char* text);
   };

   // HelloWindow.h
   #pragma once
   #include ⟨Window.h⟩
   class HelloWindow : public BWindow {
   public:
   HelloWindow(BRect frame);
   virtual bool QuitRequested(); //Override
   };

   //-----------
   // HelloWindow.cpp
   #include "HelloWindow.h"
   #include "HelloView.h"
   HelloWindow::HelloWindow(BRect frame) :BWindow(frame, "Hello",
B_TITLED_WINDOW, B_NOT_RESIZABLE | B_NOT_ZOOMABLE) {
    BRect viewRect = Bounds();
    HelloView* theView = new HelloView(viewRect, "HelloView", "HelloWorld!");
    AddChild(theView);
    Show();
   }
    HelloWindow::QuitRequested() {
    be_app-> PostMessage(B_QUIT_REQUESTED);
    return true;
   }
```

This is the classic Hello, world! program in BeOS' own API. Compared to the hundred-line long program that starts and exits retiring 1 written in Win32 API, you're getting to wonder if anyone can actually program that.

This API is one of the reasons why, despite the relatively small user base, the BeOS has such a wide range of software available. Even casual users can easily get started and program, if only for the fun of it. BeOS (and subsequent projects) come with an e-copy of Programming the Be Operating System, a very well-written introduction to the BeOS API.

It is a thing that many operating systems lack. In the case of X11, the lack of a standard, unified (and decent, Athena doesn't count) widget set to ship with X from the very beginning led to a multitude of toolkits and a generalized incoherence. BeOS, on the other hand, is not dependent of any widget set, despite being able to run Gtk-based apps, for example. Having a standard, competent API for high level application programming would ensure a good start for an application base, since programmers that are not yet wizards (or don't have time to master a thick book of API reffering to everything from buttons to kernel rings) could start programming decent applications right away. It also ensures a general consistence, since instead of having 5 Gtk-based applications, 3 Qt-based, one that uses Tcl/Tk, two for WxWindows and so on, you will find that you have one or maybe two (Gaim and probably Firefox) applications that use another widget set.

BeOS also ships with a very good IDE, the BeIDE, which is a direct descendant of Metrowerks' CodeWarrior. It also ships with a number of GUI builders so that writing GUI-based applications becomes much easier.

3 Learning from the bad things

Not everything is bells and whistles in BeOS. There are some parts that require improvement and could be learnt from.

3.1 Purpose

Unfortunately, at this moment, BeOS lacks a clearly-defined purpose for development. The only seriously viable project at the moment is Zeta, but Zeta doesn't seem to have a target they're aiming at.

When BeOS was developed, it was initially thought as an operating system for multimedia, and indeed it does have good support for multiple media streams. However, subsequent efforts trying to improve media support are close to zero. This lead to some seriously brain damaged situations, like the fact that the Media Player which ships with an OS designed for multimedia doesn't know how to play DivX movies.

Without a serious (at least short-term) target, the things seem to move but without solving any issues in particular. For example, BeOS still doesn't have a decent IM client unless you want to install X and Gaim. Despite the fact that everyone acclaimed the BeOS interface, Zeta chose to make it themeable (and, knowing the BeOS API, it wasn't easy, that's for sure), instead of focusing on other things that people hated (hardware support, security issues and so on).

The lack of a certain purpose also means that the things are moving, only nobody knows why and where to. This leads to the absence of a stable base for future improvements, often translated into a serious instability. BeOS 5 wasn't a completely stable system itself, but it wasn't annoying since system crashes were rare. In the worst case, Tracker would crash and restart itself from time to time; present projects (um, Zeta) are annoying because these things happen even more often, making me suspect that the problem really resides in the memory management.

3.2 Ergonomics

There is one slight problem which I didn't mention about the Tracker. The menu is placed in the top-right corner. Since BeOS doesn't support foreign languages like Arabian, it's to be expected that everyone would read the menu from left to right.

Well, submenus unfold at the left of the parent, not the right, since it doesn't have enough space on the screen (it's already at the edge). It makes up for a good thinking about Fitts's laws: put big things in the corner, but choose the corner correctly.

Another thing BeOS lacks is a serious file manager. Unfortunately, the one that ships with it is more like a joke, often not-so-intuitive and not supporting some usual functions.

Many of the ideas shipped with Zeta should have gone through a more serious selection. Right now, Zeta 1.0 looks like more of an experiment than a rock-solid, standard desktop operating system. It is one thing that the guys at KDE understood (despite the fact that they don't develop an operating system but a desktop environment as big as one): no release is final unless it's really solid.

3.3 Interoperability

BeOS also has a few problems when it comes to interoperability. Despite not having a large user base, BeOS 5.x didn't ship with a Samba server natively. Even now, it's becoming quite hard for it to be used around a network, since getting file sharing to work has a great potential for causing headaches.

3.4 BeOS is hard to obtain

While the Personal Edition is still available (and updated) for free download, the only serious way around is probably Zeta. Unfortunately, Zeta is not available for a free download, not even a preview edition of some sort. Users who are not familiar with it won't get the chance to get familiar with it either. Despite being cheap, at only 99 Eu, Zeta is not a safe bet, and you don't get the chance to evaluate it. In this case, I must admit I really like Microsoft for letting, sorry, I mean leaking beta versions on the Internet.

3.5 Accessibility

As far as this moment, BeOS doesn't support more than a few languages. Despite being translated in a few language, using it is hard for those who don't speak a widely-used language. When compared to other operating systems like Linux or Windows, it is losing a lot of users, especially since it is cheap enough to be widely adopted in poor countries.

4 Resources

4.1 More about BeOS

http://yellowtab.com, Yellowtab's homepage. Yellowtab is the company behind Zeta.

http://haiku-os.org, homepage of the Haiku project.

http://www.iscomputeron.com/index/, IsComputerOn. More than a joke about the IsComputerOn() API function in the BeOS API (if you're wondering if it could ever return FALSE you're getting at it), it's a good source of information about BeOS.

http://www.begroovy.com/index.php, a good website also offering a few links.

4.2 BeOS hardware support

http://www.bedrivers.com/, obviously focused on, well, drivers for BeOS.

http://laptop.bug-nordic.org/, deals with BeOS for laptop computers.

4.3 BeOS UI

http://lowendmac.com/backnforth/010416.html: BeOS or NeXT, Did Apple Make the Wrong Choice?

http://lowendmac.com/backnforth/010423.html: User Interface: Mac vs. BeOS

http://en.wikipedia.org/wiki/Fittss˙Law, a good introduction to Fitts' Law

*

---

*. This document has been written using the GNU TeXmacs text editor (see www.texmacs.org).