CASE STUDY

BALLY WULFF

BY MATEUSZ PIOTROWSKI

BALLY WULFF Games & Entertainment GmbH is a prominent German company in the entertainment electronics segment that develops, produces and sells cash gaming machines. With its headquarters located in Berlin, BALLY WULFF operates not only in Germany, but also in Spain, and currently employs around 300 people.

Since the early 2000s, FreeBSD has been the platform of choice for BALLY WULFF products. Thanks to its invaluable stability and consistency, the system engineering team is able to meet the ever-changing market demand whether that is for small, disk-space footprint, higher security measures, or better graphics. Oftentimes, the team contributes code and documentation patches back to the community. In addition, many BALLY WULFF employees have served as FreeBSD committers.

BALLY WULFF is well known for its development process happening entirely in-house. The final products are the collective work of various BALLY WULFF teams. Collaboration among product designers, hardware engineers and game developers is the name of the game. Everything from the design of the machines, hardware integration and game development, to the final production and assembly of the machines takes place in-house. FreeBSD is at the heart of it all not only as the operating system in the final gaming machine, but also in the development workstations and production appliances.

This short case study provides a deeper insight into BALLY WULFF system engineering team's goals and explains how FreeBSD helps to achieve them.

FreeBSD is at the heart of it all not only as the operating system in the final gaming machine, but also in the development workstations and production appliances.

Goal 1: Limiting the Disk Footprint of the OS

A BALLY WULFF gaming machine can be thought of as a huge video game console. What may come as a surprise in the era of IoT is that it is not connected to the Internet, but instead ships with all the software, games and their assets preinstalled. Once it leaves the production site, the only way to change the software on a machine is via a manual update procedure that involves physical storage media like USB sticks. Although no longer a pressing issue, the older generations of gaming machines posed an interesting challenge for the system engineers. Disk quotas for each team had to be carefully balanced so that every game and administrative tool received its fair share of a disk. However, the disks always turned out to be just a little bit too small to fit all the desired data. As a result, the operating system

had to be stripped of all unnecessary bits. FreeBSD, like other, well-designed software projects, provides a great number of build options capable of excluding everything but essentials from being compiled.

Unfortunately, the standard build options were not enough. It turned out that in order to achieve the desired disk footprint, the system engineering team had to gain control of a greater granularity over the build process. Luckily, the FreeBSD build system has been engineered, maintained, and constantly improved with customizability and stability in mind. It is by design that downstream consumers (and appliance vendors in particular) are able to look under the hood of the build system, modify it as needed, and expect only a minimal maintenance overhead caused by the local changes to the source tree.

BALLY WULFF has maintained an internal patch set for the FreeBSD build system to exclude non-essential files from final OS images. This appliance-specific patch set has naturally fit into the build infrastructure and does not feel like an external add-on bolted on to an existing environment. At the same time, it has not caused a significant maintenance burden to the system engineering team. As a result, the disk footprint of the OS has been limited to a minimum, leaving more disk space for games—a real value to customers.

Goal 2: Shipping Modified Packages

The FreeBSD Ports Collection has proven to be an invaluable asset to BALLY WULFF over the years. It offers a standardized and expandable way of customizing and adding additional software to the OS. In fact, it is so straightforward that creating customized packages is one of the first things new FreeBSD users learn about.

FreeBSD ports developers make sure that the ports framework evolves steadily and stays backward-compatible for many years. So even though the FreeBSD Project has already switched to poudriere within its packaging infrastructure, users like BALLY WULFF can still migrate at their convenience. Ultimately, FreeBSD is all about stability with no unpleasant surprises. As a result, it is easy for the system engineering team at BALLY WULFF to keep up with the changes and plan ahead.

BALLY WULFF maintains—internally—a fork of the FreeBSD Ports Collection with a handful of additional company-specific ports and patches for the existing ports. Not only is backporting of the latest versions of ports incredibly easy, but maintaining a custom version of an existing port is also simple and painless. Another advantage of the FreeBSD Ports Collection is that the distribution of packages via an internal package repository is effortless and well-supported.

It is easy for the system engineering team at BALLY WULFF to keep up with the changes and plan ahead.

The FreeBSD Project is constantly adding new improvements to ease the process of extending private collections of ports that benefit BALLY WULFF directly. The latest example is poudriere, which streamlines package building, testing, and publishing processes. Another important feature is an overlay support for ports trees, which is currently being tested by the system engineering team at BALLY WULFF. There is a high chance that it will alleviate the need to keep an internal fork of the ports tree, further reducing maintenance overhead.

Goal 3: Customizing System Startup

Typically, general-purpose operating systems feature a program to control the system start-up. It usually configures the newly booted system, for example, by mounting disks and starting essential system services like networking.

In the case of a BALLY WULFF gaming machine, the system start-up procedure is quite different from a typical desktop. The standard FreeBSD start-up procedure is configurable enough to cover most use cases for servers and desktops, but in the case of a gaming machine, it made sense to replace the standard rc(8) mechanism completely. Luckily, there is no black magic involved in replacing the standard rc(8) framework with a custom one. Actually, replacing the /etc/rc file is enough to get started. As a result, the system engineering team at BALLY WULFF maintains a dedicated system start-up script that prepares the OS environment for the games to launch.

At BALLY WULFF, the customized rc(8) framework has been in use for many releases and it continues to work flawlessly. This is certainly a benefit of FreeBSD's steady development practices and modularization of the base system—it is absolutely reasonable to customize a part of FreeBSD and expect the rest of the system to work just fine. It definitely gives the developers peace of mind and lets them focus on developing what is important rather than constantly catching up with backwards-incompatible, upstream changes.

Goal 4: Supporting Custom Update Procedures

It should come as no surprise that BALLY WULFF gaming machines require regular updates. Platform updates occur when there is a need to squash an annoying bug or add an important business functionality to an already-released and operating machine. Much more often, however, the machines are updated with new games. The update process of the gaming machines is one of the most important and rigorously tested procedures in the company. Thorough testing and QA checks guarantee that the updates applied to the machines already operating in the market are not going to cause any unnecessary downtime. The update process must allow for unsupervised and automatic installation of new software. Rendering the machine inoperable in the course of an update is out of the question.

Due to the architecture of the gaming machine's operating system, it would be unnecessarily complicated to update the system with freebsd-update(8) and pkg(8). Thankfully, FreeBSD's simplicity allows for implementing a completely custom update procedure.

Goal 5: Running the Same OS in Both Production and Development

One of the golden rules of software engineering is that development should happen in an environment identical or at least closely resembling the production environment. Developers do not have to debug their code twice when working in a unified environment.

BALLY WULFF game developers use FreeBSD workstations to test games before trying them out on the actual gaming machines. Amazingly, FreeBSD powers the gaming machines and the workstations equally well.

It is worth noting that the game development department is many times larger than the FreeBSD team at BALLY WULFF. Nevertheless, maintaining an internal distribution of FreeBSD tailored specifically to the game developers' needs is very doable. The same FreeBSD-based OS runs on both a gaming machine and on a development workstation, the difference being mainly the list of installed packages. This is a great benefit of FreeBSD being a general-purpose OS.

Goal 6: Staying Close to the Community

Being close to the community allows BALLY WULFF to both participate in the development of FreeBSD and to stay in contact with FreeBSD developers. For example, BALLY WULFF aims to keep the amount of local FreeBSD patches to a minimum. The maintenance burden of non-essential patches is simply not supportable. Not only is upstreaming patches a very sound business decision due to additional testing, but it is also a great way to give back to the FreeBSD community. Most of the time, however, the FreeBSD patches developed internally at BALLY WULFF are too vendor-specific and not suitable for inclusion in the FreeBSD source trees. Nevertheless, the company makes sure to contribute in other ways as well. BALLY WULFF developers regularly participate in FreeBSD Developer Summits and open-source conferences like FOSDEM and EuroBSDcon. In 2019, BALLY WULFF hosted a DevSummit organized at the company's headquarters in Berlin.

Summary

FreeBSD has been a great OS for BALLY WULFF thanks to its remarkable build system, which has been developed and maintained in a way that allows for effective adaptation of the system to specialized appliances. In the past, a major benefit of FreeBSD to BALLY WULFF was the small, yet functional, base system that could be stripped down even further by utilizing existing build(7) knobs or by introducing vendor-specific changes to precisely control what is included in the final OS image. The internal patches to FreeBSD base and ports build systems fit naturally into the consistent Makefile-based infrastructure. All those features allowed the BALLY WULFF system engineering team to minimize the size of the OS, which, in turn, left more space for games and their assets as the BALLY WULFF developers continued to push hardware and software limits.

Now that disk space is not as precious as it once was, the need for a special patch set reducing the final size of the OS is gone. The focus and energy at BALLY WULFF are directed toward other aspects of system development. It is no longer necessary to heavily modify FreeBSD sources to optimize for small disk footprint. The transition to building the OS from the unmodified FreeBSD sources is underway. So far, it has been painless and beneficial as it significantly simplified the build infrastructure. This is a result of a herculean effort by the FreeBSD community to maintain backward compatibility wherever feasible. Every major change to the FreeBSD system is implemented with downstream consumers workflows in mind.

The computing world has evolved quickly and the team's focus is no longer on making the system footprint as small as possible. The target now is to increase the robustness of the system and to keep maintenance costs low. Ultimately, the goal of the BALLY WULFF system engineering team is to provide game developers with a performant and stable gaming platform.

MATEUSZ PIOTROWSKI is a FreeBSD ports and documentation committer based in Berlin. He enjoys troubleshooting bugs, scripting automation, and designing robust software systems (always thoroughly documenting everything along the way). Recently, his interests have drifted toward tracing and performance engineering. When he is not hacking on the supposedly deterministic circuitry of modern software, he is exploring the ever-changing dynamics within society and culture.