

DKUUG

Offers
exiting
training tutorials

12th and 13th
September

Prior to EuroBSDCon **2007**

DKUUG is proud to offer the following training tutorials the 12th and 13th September 2007. The tutorials will be held in connection with the EuroBSDcon conference at the 14th and 15th September 2007.

Mr. Marshall Kirk McKusick (USA) gives 2 one-day training tutorials about The FreeBSD Kernel (day 1) and File systems and networking (day 2). Peter N. M. Hansteen (Norway) gives a 1/2 day training tutorial about Packet Filter (PF), Brooks Davis (USA) gives 1/2 day training tutorial about his experience with building and maintaining of large cluster systems. Marko Zec (Croatia) gives 1/2 day training tutorial about the Virtualized network stack (Vimages) in FreeBSD-CURRENT.

*Kirk McKusick Peter Hansteen
Brooks Davis Marko Zec*

Everybody can participate in these tutorials independent of participation in the EuroBSDCon conference. The price is inclusive coffee and lunch. (lunch is also included for 1/2 day tutorials).

The expected prices are not yet final, but will be at this level. The final prices will be announced at the conference booking site.

1/2 day	2/2 day	3/2 day	4/2 day
DKK 1125 (~EUR 150)	DKR 1500 (~EUR 200)	DKR 2500 (~EUR 333)	DKR 3000 (~EUR 400)

Speaker	Duration	Titel
Kirk McKusick	1 day 12th Sep. 8.00-17.00	An Introduction to the FreeBSD Open-Source Operating System: Kernel Functions
Kirk McKusick	1 day 13th Sep. 8.00-17.00	An Introduction to the FreeBSD Open-Source Operating System: Filesystems and Networking
Peter N. M. Hansteen	1/2 day 12th Sep. 8.00-12.00	Firewalling with PF (Packet Filter)
Marco Zec	1/2 day 12th Sep. 13.00-17.00	Virtualized network stack in FreeBSD -CURRENT
Brooks Davis	1/2 day 13th Sep. 8.00-12.00	Building Clusters With FreeBSD
Legoland Tour	Full day 16th Sep. 7.00- 24.00	Bus Tour to Legoland in Billund, Coffe, Entrance, Dinner.

Booking is done at the EuroBSDCon conference website at <http://2007.eurobsdcon.org> website.

Bus tour to Legoland in Billund Denmark

At Sunday 16th September 2007 we will arrange a Bus Tour to Legoland in Billund. We will go by bus to Billund, and will be approx 7 hours in the park.

Time	Event
7.00	Depart from CPH-Hostel
11.00	Arrival at Legoland Billund.
18.00	Depart from Legoland.
19.00 (approx)	Dinner. (details to be announced at website)
21.00 (approx)	Bus depart to Copenhagen.
24.00 (approx)	Bus is back at CPH-Hostel.

The price for the Bus tour is DKK 835,- (EUR 112 or USD 151) and includes Transport, morning coffee with bread, entrance to Legoland, and dinner at home tour. Participants must pay separate for lunch in Legoland (of your choice) and beverages at the dinner (wine/bear/water of your choice).

Please Observe

We reserve the right to cancel tutorials and the bus tour if too few has signed up 3 weeks before. If you are interested in taking any of these tutorials or joining the bus tour to Legoland you should sign up as early as possible.

Kirk McKusick

two 1 day tutorials (8 hours each).

In general: The 2 1-day courses can be taken as either a full 2 day course or you can pick the part of the FreeBSD system you are particular interested in. There is no overlapping information between the 2 days. An Introduction to the FreeBSD Open-Source Operating System. Dr. Marshall Kirk McKusick Author and Consultant.

Day 1: Kernel Functions

Title: An Introduction to the FreeBSD Open-Source Operating System: Kernel Functions

An Introduction to the FreeBSD Open-Source Operating System: Kernel Functions Dr. Marshall Kirk McKusick Author and Consultant

What is FreeBSD?

FreeBSD, like Linux, is an open-source UNIX-like operating system that is widely used to support the core infrastructure of many companies worldwide. Because it can be built with a small footprint, it is also seeing increased use in embedded applications. The licensing terms of FreeBSD do not require the distribution of changes and enhancements to the system. The licensing terms of Linux require that all changes and enhancements to the kernel be made available in source form at minimal cost. Thus, companies that need to control the distribution of their intellectual property increasing are building their products using FreeBSD.

Who Should Take this Course?

This course is of direct use to the professionals who work with FreeBSD systems. Individuals involved in technical and sales support can learn the capabilities and limitations of the system; system administrators without direct experience with the FreeBSD kernel can learn how to maintain, tune, and configure the system; applications developers can learn how to effectively and efficiently interface with the system; and systems programmers can learn how to extend, enhance, and interface to the system.

This course provides a broad overview of how the FreeBSD kernel implements its basic services. It will be most useful to those who need to learn how these services are provided. Students who will benefit from this course include operating-system implementors, system programmers, UNIX application developers, administrators, and curious users. This course is directed to users who have had at least a year of experience using a UNIX-like system. Knowledge of the C programming

language is helpful, but not essential. They should have an understanding of fundamental algorithms (searching, sorting, and hashing) and data structures (lists, queues, and arrays).

Description

This course will provide a firm background in the FreeBSD kernel. The course will cover basic kernel services, process structure, the FreeBSD jail facility for hosting virtual machines, scheduling, signal handling, and virtual and physical memory management. The kernel I/O structure will be described showing how I/O is multiplexed, and how special devices are handled. The presentations will emphasize code organization, data structure navigation, and algorithms. It will not cover the machine specific parts of the system such as device drivers.

Morning

- Kernel Overview
- Process structure
- Locking
- Communications
- Process groups and sessions
- Jails
- Scheduling
- Signals and timers
- Virtual memory management

Afternoon

- Kernel I/O structure
- I/O data structures
- Disk management
- Multiplexing I/O
- Autoconfiguration strategy
- Configuration of a device driver

Course Text

Marshall Kirk McKusick and George V. Neville-Neil, *The Design and Implementation of the FreeBSD Operating System*, Addison-Wesley Publishing Company, Reading, Massachusetts, 2005, 720 pages.

Day 2: Filesystems and Networking

Title: An Introduction to the FreeBSD Open-Source Operating System: Filesystems and Networking

Abstract

An Introduction to the FreeBSD Open-Source Operating System: Filesystems and Networking
Dr. Marshall Kirk McKusick Author and Consultant

What is FreeBSD?

FreeBSD, like Linux, is an open-source UNIX-like operating system that is widely used to support the core infrastructure of many companies worldwide. Because it can be built with a small footprint, it is also seeing increased use in embedded applications. The licensing terms of FreeBSD do not require the distribution of changes and enhancements to the system. The licensing terms of Linux require that all changes and enhancements to the kernel be made available in source form at minimal cost. Thus, companies that need to control the distribution of their intellectual property increasing are building their products using FreeBSD.

Who Should Take this Course?

This course is of direct use to the professionals who work with FreeBSD systems. Individuals involved in technical and sales support can learn the capabilities and limitations of the system; system administrators without direct experience with the FreeBSD kernel can learn how to maintain, tune, and configure the system; applications developers can learn how to effectively and efficiently interface with the system; and systems programmers can learn how to extend, enhance, and interface to the system.

This course provides a broad overview of how the FreeBSD kernel implements its basic services. It will be most useful to those who need to learn how these services are provided. Students who will benefit from this course include operating-system implementors, system programmers, UNIX application developers, administrators, and curious users. This course is directed to users who have had at least a year of experience using a UNIX-like system. Knowledge of the C programming language is helpful, but not essential. They should have an understanding of fundamental algorithms (searching, sorting, and hashing) and data structures (lists, queues, and arrays).

Description

This course will provide a firm background in the FreeBSD kernel. The course will begin with a description of how the filesystem buffers are managed. The implementation of the filesystem and its capabilities including soft updates and snapshots will be described. The filesystem interface will then be generalized to show how to support multiple filesystem types. The course will also cover the FreeBSD socket-based network architecture, layering and implementation. The socket communications primitives and internal layering will be discussed, with emphasis on the interfaces between the layers. A discussion of routing issues will be included. The presentations will emphasize code organization, data structure navigation, and algorithms. It will not cover the machine specific parts of the system such as device drivers.

Morning

- Filesystem Overview
- Filesystem organization
- Block I/O system (buffer cache)
- Filesystem implementation
- Soft Updates
- Snapshots
- Support for multiple filesystems

Afternoon

- Networking Implementation
- System layers and interfaces
- Internet protocols (TCP/IP)
- Data structures (mbufs and control blocks)
- Routing issues

Course Text

Marshall Kirk McKusick and George V. Neville-Neil, *The Design and Implementation of the FreeBSD Operating System*, Addison-Wesley Publishing Company, Reading, Massachusetts, 2005, 720 pages.

Biography (Kirk Mc Kusick)

About the speaker

Dr. Marshall Kirk McKusick writes books and articles, consults, and teaches classes on UNIX- and BSD-related subjects. For the past ten years he has been a developer and commiter to the FreeBSD Project. His particular areas of interest are the virtual-memory system and the filesystem. While at the University of California at Berkeley, he implemented the 4.2BSD fast file system, and was the Research Computer Scientist at the Berkeley Computer Systems Research Group (CSRG) overseeing the development and release of 4.3BSD and 4.4BSD. He earned his undergraduate degree in Electrical Engineering from Cornell University, and did his graduate work at the University of California at Berkeley, where he received Masters degrees in Computer Science and Business Administration, and a doctoral degree in Computer Science. He is a past president of the Usenix Association, is on the editorial board of ACM's Queue magazine, and is a member of ACM and IEEE.



Peter N. M. Hansteen

length: 1/2 day (4 hours)

Title:

Firewalling with PF

Abstract

This tutorial is for aspiring or seasoned network professionals with at least a basic knowledge of networking in general and TCP/IP particular. The aim is to teach tools and techniques to make sure your network works the way it's supposed to, keeping you in charge. Central to the toolbox is the OpenBSD PF packet filter. Whether you are a greybeard looking for ways to optimize your setups or a greenhorn just starting out, this session will give you valuable pointers to how you build the network you need. The session will also offer some insight in the significant changes to be introduced to PF in the upcoming OpenBSD 4.2 version.

Biography (Peter Hansteen)

About the speaker

Peter N. M. Hansteen is a consultant, writer and sysadmin based in Bergen, Norway. He has been tinkering with computers since the mid 1980s, mainly while working to document how the systems work and why they don't, in English as well as his native Norwegian. In 1991 he co-founded Datadokumentasjon AS, a documentation and localization company where he is still chairman and senior consultant. Peter rediscovered Unixes about the time 386BSD appeared. After a few years on Linux, which included participation in the RFC1149 implementation (2001), he eventually migrated all important bits to FreeBSD and OpenBSD. A long time freenix advocate, he is a member of the BLUG (Bergen (BSD and) Linux User Group) core group and current vice president of NUUG (the Norwegian Unix User Group). During recent years a frequent lecturer and tutor with emphasis on FreeBSD and OpenBSD topics, he is now working on a book on building the network you need using PF and other free tools.



Brooks Davis

length: 1/2 day (4 hours)

Title:

Building Clusters With FreeBSD

Abstract:

Since late 2000 we have developed and maintained a general purpose technical and scientific computing cluster running the FreeBSD operating system. In that time we have grown from a cluster of 8 dual Intel Pentium III systems to our current mix of 64 dual Intel Xeon and 289 dual AMD Opteron systems. This paper looks back on the system architecture as documented in our BSDCon 2003 paper “Building a High-performance Computing Cluster Using FreeBSD” and our changes since that time. After a brief overview of the current cluster we revisit the architectural decisions in that paper and reflect on their long term success. We then discuss lessons learned in the process. Finally, we conclude with thoughts on future cluster expansion and designs.

Our cluster, Fellowship (for “The Fellowship of the Ring”), consists of 233 dual CPU nodes. 64 of these nodes use Intel Xeon CPUs and 169 use AMD Opterons (32 of which are dual core). Additionally, another 120 dual-core Opterons are on order and will be installed shortly. This will yield a total of 1010 CPU cores. These nodes are connected to each other and an array of core servers via a gigabit Ethernet switch. The majority of these servers run FreeBSD 6 as do the nodes. The nodes are booted over the network using the Intel PXE framework. They have internal disks, but those disks are used solely for local, temporary storage and are automatically configured during boot. To control access to the cluster we run Sun Grid Engine (SGE), an open source batch queuing system. Users submit their job scripts which may consist of independent single system jobs or parallel jobs using MPI (the Message Passing Interface), PVM (the Parallel Virtual Machine), or Grid Mathematica. System operations are monitored using the Ganglia cluster monitor, the Nagios system and service monitor, and SGE’s internal data collection capabilities. We currently have over 100 users and see nearly 100% utilization during the day.

While designing and implementing Fellowship, we made a large number of architectural decisions. Some such as the choice of network booting using the FreeBSD diskless infrastructure have continually paid dividends, in this case by dramatically simplifying maintenance. Others such as using custom chassis in 2-post racks have worked well for us, but aren’t for everyone. Similarly, the use of serial consoles was a good idea

in the abstract, but has been a feature we have pulled back from due to high costs and limited utility most of the time. Finally, some decisions like allowing users direct system access during early development and attempting to encourage voluntary migration to SGE were definite mistakes, in this case due to user inertia. In this section we discuss the decisions we needed to make, options we considered then or would consider today, our initial decision, any deviations from that decision, and how the decision played out. The goal of this section is to evaluate our architectural decisions and in the process give readers the tools they need to begin designing their own clusters.

In the process of operating Fellowship, we have had a number of lessons driven home to us. None of them are truly shocking, but they were not thing we were expecting. For example, relatively uncommon problems can be major issues in a cluster. In one case, we were forced to disable BIOS access via serial console on some machines because they hung at boot around 1 out of 30 times. That meant we thought we had corrected the problem during testing by reducing baud rate, but in fact we had just made it less common. In production, several machine hung at boot every time we rebooted the cluster. This forced us to disable the feature. The need to perform tasks on all nodes has really driven home the power of the Unix tool model. For example the following command was used to automatically add each host the correct per CPU type host group:

```
grep r[01][01789] hostlist | fping -a | \
xargs -i node -n1 rsh node 'qconf -aattr hostgroup hostlist
hostname \
@cpu_apteron_cpuid | grep "name string" | sed -e "s/[^0-9]*//"'
```

This and other similar examples highlight the value of knowing the powerful set of tools at your fingers as an administrator. Another key lesson is the difficulty of remembering that our users are experts in one or more domains, but not usually Computer Science or related fields despite the often large and complex applications they have written. This is helpful to keep in mind when debugging as it generally means the users don't know where to start. It is also important to keep in mind when discussing complex issues like scheduling because it is often the case that users have a mental model of computation that does not match reality. For example we have found that the belief that a job that starts sooner--no matter how contended the system is--will finish sooner when there are a number of cases where this is clearly untrue.

Thus far we have followed a model of continued expansion and refresh of Fellowship's hardware and software rather than wholesale replacement. This has had the advantage of allowing us to achieve a larger size than would have otherwise been possible given our capital budget. One downside is that some decisions such as node form factor and network interconnect are hard to change incrementally. With an eye toward an eventual second cluster for more capabilities and improved redundancy we have kept an eye on both technology trends that are easy to apply to Fellowship and ones that would only apply to a clean slate. We are in the process of designing a second redundant cluster to be installed at another location some time this fiscal year. In many respects it will be similar to Fellowship. The main expected differences are a faster network, probably 10 Gigabit Myrinet and using clustered storage such as that provided by Panasas and Isilon to completely eliminate node disks, our single largest source of hardware problems.

Biography (Brooks Davis)

About the speaker

Brooks Davis is a Senior Member of Technical Staff in the High Performance Computing Section of the Computer Systems Research Department at the Aerospace Corporation. He has been a FreeBSD user since 1994 and a FreeBSD committer since 2001. He earned a Bachelors Degree in Computer Science from Harvey Mudd College in 1998. His computing interests include high performance computing, networking, security, mobility, and, of course, finding ways to use FreeBSD in all these areas. When not computing, he enjoys reading, brewing, cooking, and pounding on red-hot iron in his garage blacksmith shop.



Marco Zec

Length: 1/2 day (4 hours) tutorial

Title:

Virtualized network stack in FreeBSD -CURRENT

Abstract

Network stack virtualization allows for complete networking independence between jails on a FreeBSD system, including providing each jail with its own virtual network interface set, routing tables, firewalls, rate limiting, IPSEC configuration and more. Compared to full hardware or paravirtualization platforms such as VMWare, Xen or UML, the combination of FreeBSD jails and virtualized stacks incurs significantly lower virtualization overhead while efficiently utilizing the available hardware resources, allowing for great scalability and levels of performance virtually indistinguishable from an unmodified OS kernel.

In this workshop we will show how to configure a virtualized FreeBSD system for typical applications such as virtual hosting, network emulation, or monitoring multiple independent VPNs. We will also explain how to manage the new networking facilities in the OS from C programs, and briefly discuss the reasoning behind the design of the new kernel-level virtualization APIs.

The network stack virtualization project is supported through a sponsorship of the FreeBSD Foundation and Stichting NLNet, and is run as a joint technology development agreement between the FreeBSD Foundation, NLNet, and the University of Zagreb, with the ultimate goal of including the virtualization features in a new release of the FreeBSD operating system.

Biography

About the speaker

Marko Zec is a research assistant at the University of Zagreb. In a past life he used to work for IBM and several local Cisco system integrators in Croatia as a “networking specialist”. The projects he recently worked or is working on include XORP, IMUNES, and network stack virtualization for FreeBSD -CURRENT.

Picture: attachment:marko.jpg





The sixth European BSD Conference

12th - 15th September 2007 in Copenhagen - Denmark

*And you tell me
it doesn't have
to reboot?*



Buy your ticket at
<http://www.euroBSDCon.org/>