NetBSD

at Chaos Communication Camp 2003

Stefan Schumacher, <stefan@net-tex.de>, PGP Key <0xB3FBAE33>

http://www.uni-magdeburg.de/steschum/

NetBSD? Huh?

- successor of 4.4BSD

NetBSD? Huh?

- successor of 4.4BSD
- UNIX-like Operatingsystem

NetBSD? Huh?

- successor of 4.4BSD
- UNIX-like Operatingsystem
- Open Source

NetBSD? Huh?

- successor of 4.4BSD
- UNIX-like Operatingsystem
- Open Source
- liberal BSD Licence

NetBSD? Huh?

- successor of 4.4BSD
- UNIX-like Operatingsystem
- Open Source
- liberal BSD Licence
- main goals: portability, interoperability, security

NetBSD? Huh?

- successor of 4.4BSD
- UNIX-like Operatingsystem
- Open Source
- liberal BSD Licence
- main goals: portability, interoperability, security
- most portable OS

Open Source

- complete sourcecode available

Open Source

- complete sourcecode available
- older versions available

Open Source

- complete sourcecode available
- older versions available
- several branches are maintained (stable, -current)

Open Source

- complete sourcecode available
- older versions available
- several branches are maintained (stable, -current)
- updates available as source or binaries

BSD Licence

- no warranty & mention my name

BSD Licence

- no warranty & mention my name
- changes don't have to be published

BSD Licence

- no warranty & mention my name
- changes don't have to be published
- code can be proprietarized

BSD Licence

- no warranty & mention my name
- changes don't have to be published
- code can be proprietarized
- commercial products basing on NetBSD available

Organization

- $\approx$ 220 Official developers

Organization

- $\approx$ 220 Official developers
- Management via Board and Core

Organization

- $\approx$ 220 Official developers
- Management via Board and Core
- Port maintainers
- Security Officers
- Release Engineers (releng)
- Internal System Administrator
- Website / Doc Maintainers

The most portable OS

13 CPU architectures

alpha arm hppa i386 m68010 m68k mipseb mipsel

ns32k powerpc sh3eb sh3el sh5 sparc sparc64 vax x86_64

The most portable OS

13 CPU architectures

alpha arm hppa i386 m68010 m68k mipseb mipsel

ns32k powerpc sh3eb sh3el sh5 sparc sparc64 vax x86_64

60 Hardwareplatforms

alpha acorn26 acorn32 cats evbarm hpcarm netwinder shark hp700 i386 sun2 amiga atari cesfic hp300

luna68k mac68k mvme68k news68k next68 ksun3 x68k evbmips mipsco newsmips sbmips sgimips algor arc

cobalt evbmips hpcmips playstation2 pmax sbmips pc532 amigappc bebox evbppc macppc mvmeppc ofppc

pmppc prep sandpoint evbsh3 mmeye dreamcast evbsh3 hpcsh evbsh5 sparc sparc64 vax x86_64

portability

- flexible word size

portability

- flexible word size
- flexible endianess

portability

- flexible word size
- flexible endianess
- high abstraction level

portability

- flexible word size
- flexible endianess
- high abstraction level
- high compatibility

The most portable OS

- easy to port

The most portable OS

- easy to port
- usual open source support available (WWW, ML, Usenet, IRC/silc)

The most portable OS

- easy to port
- usual open source support available (WWW, ML, Usenet, IRC/silc)
- commercial support (e.g. Wasabi)

security

- usual open source advantages

security

- usual open source advantages
- security "out-of-the-box"

security

- usual open source advantages
- security "out-of-the-box"
- security software available

security

- usual open source advantages
- security "out-of-the-box"
- security software available
- cgd(4) - cryptographic pseudodevice

security

- usual open source advantages
- security "out-of-the-box"
- security software available
- cgd(4) - cryptographic pseudodevice
- verified executable

security

- usual open source advantages
- security "out-of-the-box"
- security software available
- cgd(4) - cryptographic pseudodevice
- verified executable
- chroot(8), systrace(1)

security

- usual open source advantages
- security "out-of-the-box"
- security software available
- cgd(4) - cryptographic pseudodevice
- verified executable
- chroot(8), systrace(1)
- kernel security levels, file flags

security

- usual open source advantages
- security "out-of-the-box"
- security software available
- cgd(4) - cryptographic pseudodevice
- verified executable
- chroot(8), systrace(1)
- kernel security levels, file flags
- ipf, pf as LKM

buzzword bingo

- complete IPv6 ready

buzzword bingo

- complete IPv6 ready
- SMP (alpha, i386, VAX, sparc, macppc, amd64)

buzzword bingo

- complete IPv6 ready
- SMP (alpha, i386, VAX, sparc, macppc, amd64)
- crosscompilable

buzzword bingo

- complete IPv6 ready
- SMP (alpha, i386, VAX, sparc, macppc, amd64)
- crosscompilable
- multimedia support (mplayer, xmms, fxtv ...)

buzzword bingo

- complete IPv6 ready
- SMP (alpha, i386, VAX, sparc, macppc, amd64)
- crosscompilable
- multimedia support (mplayer, xmms, fxtv ...)
- desktop ready (LaTeX, OOo, Staroffice ...)

buzzword bingo

- complete IPv6 ready
- SMP (alpha, i386, VAX, sparc, macppc, amd64)
- crosscompilable
- multimedia support (mplayer, xmms, fxtv ...)
- desktop ready (LaTeX, OOo, Staroffice ...)
- WLAN support (e.g. BSD airtools)

buzzword bingo

- complete IPv6 ready
- SMP (alpha, i386, VAX, sparc, macppc, amd64)
- crosscompilable
- multimedia support (mplayer, xmms, fxtv ...)
- desktop ready (LaTeX, OOo, Staroffice ...)
- WLAN support (e.g. BSD airtools)
- slim base installation

building a -current kernel (1/2)

- get kernelsources (FTP/CVS)

building a -current kernel (1/2)

- get kernelsources (FTP/CVS)
- edit configfile (examples/templates)

```
cd /usr/src/sys/arch/alpha/conf/ &&
cp GENERIC ARWEN && vi ARWEN
```

building a -current kernel (1/2)

- get kernelsources (FTP/CVS)
- edit configfile (examples/templates)

```
cd /usr/src/sys/arch/alpha/conf/ &&
cp GENERIC ARWEN && vi ARWEN
```

- build toolchain

```
/usr/src/build.sh tools
```

building a -current kernel (1/2)

- get kernelsources (FTP/CVS)
- edit configfile (examples/templates)

```
cd /usr/src/sys/arch/alpha/conf/ &&
cp GENERIC ARWEN && vi ARWEN
```

- build toolchain

```
/usr/src/build.sh tools
```

- build kernel

```
/usr/src/build.sh kernel=ARWEN
```

building a -current kernel (2/2)

- drink some coffee & listen to SLAYER

building a -current kernel (2/2)

- drink some coffee & listen to SLAYER
- copy new kernel to `/netbsd`

building a -current kernel (2/2)

- drink some coffee & listen to SLAYER
- copy new kernel to `/netbsd`
- reboot into new kernel

building a -current kernel (2/2)

- drink some coffee & listen to SLAYER
- copy new kernel to `/netbsd`
- reboot into new kernel
- build and install distribution

building a -current kernel (2/2)

- drink some coffee & listen to SLAYER
- copy new kernel to `/netbsd`
- reboot into new kernel
- build and install distribution

```
/usr/src/build.sh distribution &&
/usr/src/build.sh install=/
```

building a -current kernel (2/2)

- drink some coffee & listen to SLAYER
- copy new kernel to `/netbsd`
- reboot into new kernel
- build and install distribution

```
/usr/src/build.sh distribution &&
/usr/src/build.sh install=/
```

- restart services (or reboot)

building a -current kernel (2/2)

- drink some coffee & listen to SLAYER
- copy new kernel to `/netbsd`
- reboot into new kernel
- build and install distribution

  ```
  /usr/src/build.sh distribution &&
  /usr/src/build.sh install=/
  ```
- restart services (or reboot)
- done

crosscompiling (1/2)

- system is complete crosscompilable

crosscompiling (1/2)

- system is complete crosscompilable
- esp. for older/slower systems and developers

crosscompiling (1/2)

- system is complete crosscompilable
- esp. for older/slower systems and developers
- build crossarchitecture toolchain

```
./build.sh -m macppc -T /usr/cross tools
```

crosscompiling (1/2)

- system is complete crosscompilable
- esp. for older/slower systems and developers
- build crossarchitecture toolchain

```
./build.sh -m macppc -T /usr/cross tools
```

- crosscompile the kernel

```
./build.sh kernel=AREDHEL_AR-FENIEL
```

crosscompiling (1/2)

- system is complete crosscompilable
- esp. for older/slower systems and developers
- build crossarchitecture toolchain

```
./build.sh -m macppc -T /usr/cross tools
```

- crosscompile the kernel

```
./build.sh kernel=AREDHEL_AR-FENIEL
```

crosscompiling (2/2)

- build distribution

```
./build.sh -D $DESTDIR -d
```

crosscompiling (2/2)

- build distribution

  ```
  ./build.sh -D $DESTDIR -d
  ```

- cp Distribution to target

crosscompiling (2/2)

- build distribution

  ```
  ./build.sh -D $DESTDIR -d
  ```

- cp Distribution to target
- install distribution on target

  ```
  ./build.sh install=/
  ```

crosscompiling (2/2)

- build distribution

  ```
  ./build.sh -D $DESTDIR -d
  ```
- cp Distribution to target
- install distribution on target

  ```
  ./build.sh install=/
  ```
- done

pkgsrc

- www.pkgsrc.org

pkgsrc

- www.pkgsrc.org
- framework for building third-party software

pkgsrc

- www.pkgsrc.org
- framework for building third-party software
- ≈4000 packages

pkgsrc

- www.pkgsrc.org
- framework for building third-party software
- ≈4000 packages
- pkgsrc itself is portable

pkgsrc

- www.pkgsrc.org
- framework for building third-party software
- ≈4000 packages
- pkgsrc itself is portable
- /etc/mk.conf

installing software via pkgsrc

- install pkgsrc (download & extract or cvs)

installing software via pkgsrc

- install pkgsrc (download & extract or cvs)
- move to the specific directory

```
cd /usr/pkgsrc/print/teTeX
```

installing software via pkgsrc

- install pkgsrc (download & extract or cvs)
- move to the specific directory

```
cd /usr/pkgsrc/print/teTeX
```

- install it

```
make install && make clean
```

installing software via pkgsrc

- install pkgsrc (download & extract or cvs)
- move to the specific directory

  ```
  cd /usr/pkgsrc/print/teTeX
  ```

- install it

  ```
  make install && make clean
  ```

- enjoy LaTeX

pkgsrc pkgtools

- pkg_comp builds pkg in chrooted tree

pkgsrc pkgtools

- pkg_comp builds pkg in chrooted tree
- pkgdepgraph visual representation of packages

pkgsrc pkgtools

- pkg_comp builds pkg in chrooted tree
- pkgdepgraph visual representation of packages
- pkg_tarup generates binary package

pkgsrc pkgtools

- pkg_comp builds pkg in chrooted tree
- pkgdepgraph visual representation of packages
- pkg_tarup generates binary package
- check up for security issues (cron-able)

```
download-vulnerability-list && audit-packages
```

pkgsrc pkgtools

- pkg_comp builds pkg in chrooted tree
- pkgdepgraph visual representation of packages
- pkg_tarup generates binary package
- check up for security issues (cron-able)

  `download-vulnerability-list && audit-packages`
- ...

pkgsrc update

- easy maintenance of packages

pkgsrc update

- easy maintenance of packages
- update pkgsrc via CVS

```
cvs update -dP
```

pkgsrc update

- easy maintenance of packages
- update pkgsrc via CVS

  ```
  cvs update -dP
  ```

- update obsolete packages

  ```
  pkg_chk -u
  ```

Where to get?

- FTP Server at Camp: ftp://NetBSD.serveftp.org/
- CCC/Camp03/BSDVillage/NetBSD/

Where to get?

- FTP Server at Camp: ftp://NetBSD.serveftp.org/
- CCC/Camp03/BSDVillage/NetBSD/
- ftp://ftp.netbsd.org/pub/NetBSD/ (please use mirror)

Where to get?

- FTP Server at Camp: ftp://NetBSD.serveftp.org/
- CCC/Camp03/BSDVillage/NetBSD/
- ftp://ftp.netbsd.org/pub/NetBSD/ (please use mirror)
- CVS

```
export CVSRSH=ssh
export CVSROOT=anoncvs@anoncvs.netbsd.org:/cvsroot
cvs checkout src
```

- several vendors (ixSoft, Lehmanns, Wasabi, freeX ...)