– NetBSD/Soc –

Google’s Summer of Code within NetBSD

Jan Schaumann
jschauma@netbsd.org
Summer of Code, eh?

Want to write open source this summer? Want to make money? Want to do both?

Google
Summer of Code, eh?
Google summer of code: Familiar Linux
Par marc, dimanche 5 juin 2005 à 11:32 :: Geekage

On peut souvent lire ces derniers temps que tel ou tel projet va essayer de participer au Google Summer of Code [en].
(pour rappel, Google sponsorise des étudiants pour travailler sur des projets libres innovants.
Google ne demande pas que le code soit professionnel, ça reste libre.) J’ai croisé déjà deux gros projets qui vont essayer d’avoir un étudiant sur le coup: FreeBSD [en] et KDE [en]. J’ai rien contre ces deux projets, mais ils ont déjà plein de contributeurs, alors que d’autres, moins (pas de concurrence entre les projets, chacun aura son étudiant). Au hasard, je pensais à Handhelds [en] (vraiment, au hasard). Ça donnerait un coup de pouce. Par exemple:

http://marc-blog.ketaplop.net/index.php/2005/06/05/125-google-summer-of-code-familiar-linux
Summer of Code Dates

- **May 31st**: Start of application process on http://code.google.com
- **June 1st**: Last Day new Organizations will be listed on http://code.google.com
- **Interim Period**: Back and Forth with applicants on the Summer-Discuss Google Group
- **June 14th**: Final application submission deadline.
- **June 24th**: All applications approved or rejected. Cut $500 checks for initial funding.
- **Interim Period**: Give the students a helping hand and guidance.
- **August 3rd**: Google gives a preliminary progress report at OSCON
- **September 1st**: Deadline for all student work (pencils down).
- **September 30th**: All adviser feedback in.
- **October 1st**: Announce successful participants. Cut final checks send t-shirts.
Suggested Projects

- NetBSD Ports
- NetBSD Userland
- NetBSD Kernel
- Filesystems
- Networking
- pkgsrc
- Miscellaneous
Suggested Projects: Ports & Userland

- NetBSD Ports
  - Port NetBSD to SGI Octane and Origin machines
  - Support for MMU-less systems
  - Zaurus
  - IA64

- NetBSD Userland
  - WiFi browser
  - BSD licensed privacy guard
  - Wide Character Support in curses
  - BSD licensed rsync replacement
  - Dynamic NSS modules
Suggested Projects: Ports & Userland

- **NetBSD Ports**
  - Port NetBSD to SGI Octane and Origin machines
  - Support for MMU-less systems
  - Zaurus
  - IA64 (*WIP using the HP Ski simputer (see references)*)

- **NetBSD Userland**
  - WiFi browser
  - BSD licensed privacy guard
  - Wide Character Support in curses
  - BSD licensed rsync replacement
  - Dynamic NSS modules
Suggested Projects: Kernel

- NetBSD Kernel
  - Improve FFS
  - Improve Caching
  - Improve writing to FS
  - NetBSD block device driver for NAND flash chips
  - Flash translation layer
  - Compressed Cache System
  - Debug softdep on slow machines
  - Real time support
  - Bluetooth support
Suggested Projects: Filesystems

- BSD tool to create ISO filesystems
- BSD licensed XFS
- BSD licensed JFS
- BSD licensed HFS+
- Journaling for UFS
- ACLs
- Efficient Memory Filesystem
- resize_ffs
Suggested Projects: Networking & pkgsrc

- Networking
  - Teredo: Tunneling IPv6 over UDP through NATs
  - Kismet
  - NDIS network driver
  - Policy routing
  - Cleanup routing code
  - Implement IPv6 ipflow_fastforward
  - zeroconf

- pkgsrc
  - Unprivileged pkgsrc builds
  - Parallel bulk builds
Suggested Projects: Networking & pkgsrc

- Networking
  - Teredo: Tunneling IPv6 over UDP through NATs
  - Kismet
  - NDIS network driver
  - Policy routing
  - Cleanup routing code
  - Implement IPv6 ipflow_fastforward
  - zeroconf

- pkgsrc
  - Unprivileged pkgsrc builds
  - Parallel bulk builds (*WIP called “bobac”; ask jlam@NetBSD.org*)
Suggested Projects: Miscellaneous

- Miscellaneous
  - syspkgs
  - valgrind
  - NetBSD LiveCD with installer
  - CD Bootloader
  - Automate regression framework
Suggested Projects: Miscellaneous

- Miscellaneous
  - syspkgs
  - valgrind
  - NetBSD LiveCD with installer
  - CD Bootloader (*WIP* `makefs -t cd9660 in-tree`)
  - Automate regression framework
Summer of Code => Endless Summer?
Selection Process

- small team weeds out obvious rejectees
- list of remaining applications presented to developer body
- list ranked based on developer feedback
- mentors solicited from developer body
- list sorted based on developer interest + mentor availability
- developers vote for their favorite projects
- ranked list returned to Google
- Google decides the total number of awarded projects, picks top ranked applications
Accepted Projects

Out of 96 applications in total, the following projects were chosen:

- `bpg`: BSD licensed privacy guard (pgp)
- `hfs`: HFS+
- `ndis`: NDIS network driver
- `tmpfs`: Efficient memory file-system
- `userfs`: Userspace file system hooks
- `wcurses`: Wide Character Support for Curses
- `zerconf`: Zeroconfd
Green: students
Yellow: mentors
Blue: TNF organization
**hfs: HFS+**

**Why?**
- no HFS+ support currently available
- good amount of work for summer project

**Who?**
- Mentoring NetBSD developer: Bill Studenmund <wrstuden@NetBSD.org>
- Developing student: Yevgeny Binder <yevbee@comcast.net>
hfs: Results

- deliverables adjusted earlier on as the project was found slightly too ambitious
- student did not have much of a NetBSD background, so some time was spent on getting into NetBSD
- basic HFS+ filesystem completed in time
- import into NetBSD source tree: not ready yet
hfs: Results

- deliverables adjusted earlier on as the project was found slightly too ambitious
- student did not have much of a NetBSD background, so some time was spent on getting into NetBSD
- basic HFS+ filesystem completed in time
- import into NetBSD source tree: not ready yet

Success.
ndis: NDIS network driver

Why?
- driver available for FreeBSD
- allow Windows driver to run on NetBSD
- previous experience
- previous collaboration with mentor

Who?
- Mentoring NetBSD developer: Phil Nelson <phil@NetBSD.org>
- Developing student: Alan Ritter <rittera@cc.wwu.edu>
ndis: NDIS network driver

Deliverables:
- produce a driver working well enough to use at least a standard wired Ethernet card on PCI bus

Long Term Goals:
- get a working driver for one or more PCMCIA cards
- test and fix bugs on a multiprocessor system
- test and make sure it works on 64 bit systems
- merge with latest code from FreeBSD
- run as LKM
ndis: Results

- Intel EtherExpress Pro/100: works
- Broadcom wireless card: works
- mentor satisfied => we’re satisfied
- student continues work after official end of SoC
- mentor will review code before feature freeze for NetBSD 4.0
ndis: Results

- Intel EtherExpress Pro/100: works
- Broadcom wireless card: works
- mentor satisfied — we’re satisfied
- student continues work after official end of SoC
- mentor will review code before feature freeze for NetBSD 4.0

Success.
tmpfs: Efficient memory file-system

Why?
- mfs(8) is regular ffs on top of memory
- no specifically designed filesystem for temporary use available

Who?
- Mentoring NetBSD developer: Luke Mewburn <lukem@NetBSD.org>
- Mentoring NetBSD developer: Bill Studenmund <wrstud@NetBSD.org>
- Developing student: Julio M. Merino Vidal <jmmv84@gmail.com>
tmpfs: Goals

- an implementation of an efficient memory file-system
- in-depth documentation about tmpfs in detail, describing its data structures, algorithms used and the rationales that lead to the decisions taken.
- a “file-system how-to” document explaining how to write a file-system driver for NetBSD from scratch.
tmpfs: Summary

- all goals met
- rated “top-notch”
- `tmpfs(8)` already imported into NetBSD-current
- comparisons with `mfs(8)` have shown `tmpfs(8)` to be
  - more memory-efficient
  - more accurate in reporting memory usage
  - faster
- student learned enough about filesystems to already have found and fixed some serious bugs in our NFS code
- expect an article on `tmpfs(8)` on OnLamp
tmpfs: Summary

- all goals met
- rated “top-notch”
- `tmpfs(8)` already imported into NetBSD-current
- comparisons with `mfs(8)` have shown `tmpfs(8)` to be
  - more memory-efficient
  - more accurate in reporting memory usage
  - faster
- student learned enough about filesystems to already have found and fixed some serious bugs in our NFS code
- expect an article on `tmpfs(8)` on OnLamp

Success!
userfs: Userspace file system hooks

What?
- make it possible to write a file system in userspace so that an application will see no difference to the pure in-kernel file system.

Who?
- Mentoring NetBSD developer: Bill Studenmund <wrstuden@NetBSD.org>
- Developing student: Antti Kantee <antti.kantee@hut.fi>
userfs: Userspace file system hooks

- most ambitious project
- consists of three parts
  - a kernel file system shim
  - a communications protocol
  - a userland API for the file system to use
- also provide a trivial userland file system to demonstrate functionality
userfs: Userspace file system hooks

Flow-control of the “Pass-to-Userspace F-F-f-f-fileSystem”:

1. application
2. kernel (syscall, vfs ...)
3. kernel puffs
4. userspace puffs
5. fs implementation (userspace)
6. userspace puffs
7. kernel puffs
8. application
userfs: Results

- still bare-bones
- simple filesystem with some hardcoded files (which are modifyable) written
- “The framework coughs but manages to avoid complete and utter defeat.”
- code not yet imported
- all SoC goals met
userfs: Results

- still bare-bones
- simple filesystem with some hardcoded files (which are modifyable) written
- “The framework coughs but manages to avoid complete and utter defeat.”
- code not yet imported
- all SoC goals met

Success.
wcurses: Wide Character Support for Curses

Why?
- wide characters not supported in NetBSD’s curses
- limited support for internationalized character sets

Who?
- Mentoring NetBSD developer: Julian Coleman <jdc@NetBSD.org>
- Mentoring NetBSD developer: Brett Lymn <blymn@NetBSD.org>
- Developing student: Ruibiao Qiu <ruibiao@arl.wustl.edu>
wcurses: Results

- all goals met
- code will be imported into NetBSD source Real Soon Now
wcurses: Results

- all goals met
- code will be imported into NetBSD source Real Soon Now

Success.

Jan Schaumann

October 5, 2005
zeroconf: Zeroconfd

Who?

- Mentoring NetBSD developer: Christos Zoulas <christos@NetBSD.org>
- Mentoring NetBSD developer: David Young <dyoung@NetBSD.org>
- Mentoring NetBSD developer: Jason R. Thorpe <thorpej@NetBSD.org>
- Mentoring NetBSD developer: Ignatios Souvatzis <is@NetBSD.org>
- Developing student: Silvio Valenti <silvio.valenti@gmail.com>
zeroconf: Zeroconfd

Work split in two parts:

- daemon which autoconfigures an IPv4 link-local address for a network interface
- a library for multicast DNS, which is used to resolve local network host name and discover available services in network where there is no DNS server
zeroconf: Results

- zeroconfd implemented
- responderd implemented
- both working, but need more work
- import into NetBSD CVS: not yet
- too many mentors
- nevertheless: all goals met
zeroconf: Results

- zeroconfd implemented
- responderd implemented
- both working, but need more work
- import into NetBSD CVS: not yet
- too many mentors
- nevertheless: all goals met

Success.
bpg: BSD licensed privacy guard (pgp)

Why?
- no BSD licensed OpenPGP tools available
- GPL licensed `gnupg` convoluted

Who?
- Mentoring NetBSD developer: Alistair Crooks <agc@NetBSD.org>
- Mentoring NetBSD developer: Curt Sampson <cjs@NetBSD.org>
- Developing student: Manuel Freire <droggo@gmail.com>
  - previous work on `myPGP`
bpg: BSD licensed privacy guard (pgp)

BPG, the BSD Privacy Guard, is a BSD-licensed program that performs authentication and encryption using the OpenPGP standard (RFC 2440).

It provides:

- A set of libraries for signing and encrypting data, allowing the integration of OpenPGP features in other applications.
- A modular “PGP cryptography toolkit” that allow users to chose their own encryption and signing algorithms, key management structure, and so on.
- A scriptable and well thought command-line interface built over the libraries. This standalone application will be a suitable replacement for GnuPG or PGP.
bpg: main uses

The main uses supported are:

- **Data confidentiality**: the library must support different algorithms for encryption of data. Concretely, it aims to be used for symmetric and asymmetric encryption.

- **Data integrity and authentication**: via digital signatures, BPG will support providing integrity and authentication to data, as defined in OpenPGP standard.

- **Integrated key management**: BPG aims to support centralized management of all of a user’s public and private keys.
bpg: Goals

- Provide a complete implementation of the OpenPGP standard, with the only exception of possible old formats incompatibility if project needs demands it.
- Settle on the basis of a well-thought and well-designed data security framework.
- Develop command-line interface that is both:
  - powerful: it must support all the program functionality in an easily scriptable way;
  - usable: confusing user interfaces reduce security by making it harder for a user to make correct decisions.
- Design the libraries for extensibility. We'd like BPG to be a good field for developing researchs in the state-of-the-art of authentication and cryptography.
- Make BPG a good candidate to replace GnuPG usage in BSD Unices.
bgp: Architecture

The main goal of BPG is to provide applications with a toolkit for using OpenPGP facilities. For that, functionality was packed into libraries.

There are four libraries, corresponding with the four problems BPG tries to solve:

- securing data
- key management
- trust management
- algorithms
bgp: Architecture: Securing data

Library name: libbpg

Relies on other libraries:

- Key management library: PKI keys are specified in the API with user-IDs and obtained from the BPG key management library. The key management library is the responsible of key decryption if necessary.

- Algorithms library: for performing low-level encryption, hashing and compression, it uses the BPG algorithms library.

- Compression library: BPG will use libzip for compression
bgp: Architecture: Key Management

Library name: libbpgkey

The key management library is divided into a set of specialized submodules:

- **Key fetcher**: receives petitions for a key and performs the necessary operations to give it back to the user (or say why it wasn’t possible).
- **Key generator**: receives petitions for creating asymmetric or symmetric keys.
- **Key importer/exporter**: this module receives petitions from the user to take a key from a location A and insert it into location B, where A and B can be files, keyrings or key servers.
- **Key interpreter**: translates OpenPGP packets containing keys into the internal data structure for keys and vice versa.
- **Key deliverer**: with no public functions, this module performs the internal checkouts and commits of keys from and to a file, keyring or key server.
bgp: Architecture: Trust Management

Library name: libbpgtrust

The trust library handles the trust database and the trust policy. The trust database contains a list of

\[ UserID, trustlevel \]

pairs. The policy defines the rules for deriving the trust level of a given key from the trust database (i.e. OpenPGP web of trust, X.509 hierarchical trust model, ...).
bgp: Architecture: Algorithms

Library name: libbpgalgo

The initial algorithms supported will be:

- Hash functions: SHA-1.
- Symmetric algorithms: AES.
- Asymmetric algorithms: RSA, DSA.

libbpgalgo may offer with a plugins system would take the extensibility and reusability to a higher level.
bpg: Security Issues

- Memory purge
- Integrity of the keyring
- MITM attacks
- Emission captures
- Time-based attacks
- Password sniffing
bpg: Results

- very good work, mature code
- all goals set were achieved
- detailed documentation available (see references)
- student was pro-active, responsive
- result still under development on Sourceforge
- discussion on import into NetBSD source tree are ongoing
- expect a summary article in “Dr. Dobb’s Journal”
bpg: Results

- very good work, mature code
- all goals set were achieved
- detailed documentation available (see references)
- student was pro-active, responsive
- result still under development on Sourceforge
- discussion on import into NetBSD source tree are ongoing
- expect a summary article in “Dr. Dobb’s Journal”

Success!
References

General:
http://www.netbsd.org/
http://netbsd-soc.sourceforge.net/
http://www.netbsd.org/contrib/projects.html
http://www.netbsd.org/Foundation/press/soc.html
http://www.netmeister.org/netbsd/soc/

http://code.google.com/summerofcode.html
References

BPG:
http://www.sourceforge.net/projects/mypgp/
http://netbsd-soc.sourceforge.net/projects/bpg/

HFS+:
http://netbsd-soc.sourceforge.net/projects/hfs/

NDIS:
http://netbsd-soc.sourceforge.net/projects/nids

tmpfs:
http://netbsd-soc.sourceforge.net/projects/tmpfs
References

userfs:
http://netbsd-soc.sourceforge.net/projects/userfs

w curses:
http://netbsd-soc.sourceforge.net/projects/wcurses

zeroconf:
http://www.zeroconf.org/
http://netbsd-soc.sourceforge.net/projects/zeroconf
References

Other projects:

- **NetBSD/ia64:**
  - [http://www.netbsd.org/Ports/ia64/](http://www.netbsd.org/Ports/ia64/)
  - [http://mail-index.netbsd.org/port-ia64/](http://mail-index.netbsd.org/port-ia64/)