Secret Debian Internals

Enrico Zini
enrico@debian.org

25 February 2007

- Source code: `bzr branch http://bugs.debian.org/debbugs-source/mainline/`
- Data on `merkel` at `/org/bugs.debian.org/spool/`
- Data rsyncable at `merkel.debian.org::bts-spool-db/`

Directory structure:

- `/` various data files (which I haven't explored)
- `archive/nn` archived bugs (*nn* is last 2 digits of bug no.)
- `db-h/nn` active bugs (*nn* is last 2 digits of bug no.)
- `user/` usertags data

Files:

- `*.log` all raw bug activity, including the archived messages
- `*.report` the mail that opened the bug
- `*.summary` some summary bug information
- `*.status` obsolete, superseded by `summary`

- LDAP query

```
ldapsearch -p 10101 -h bts2ldap.debian.net -x -b \
  dc=current,dc=bugs,dc=debian,dc=org \
  "(&(debbugsSourcePackage=$SRCPKG)(debbugsState=open))" \
  debbugsID | grep ^debbugsID | sed 's/^debbugsID: //'
```

- SOAP interface (see `http://bugs.debian.org/377520`, ask `dondelelcaro` for more info)

BTS
Example code

```
#!/usr/bin/perl -w
# Prints the e-mail of the sender of the last message for
the given bug

use Debbugs::Log;
use ...;

my $CACHEDIR = './cache';
my $MERKELPATH =
'/org/bugs.debian.org/spool/db-h/';
my $RSYNCPATH =
'merkel.debian.org::bts-spool-db/';


my $bug = shift(@ARGV);
die "'$bug' is not a bug number" if ($bug !~
/^\d+$/);

my $log = substr($bug, -2)."/".$bug.".log";
if ( -d $MERKELPATH ) {
    # We are on merkel
    $log = $MERKELPATH.$log;
} else {
    # We are elsewhere: rsync the bug log from merkel
    my $cmd = "rsync -q $RSYNCPATH$log
$CACHEDIR/";
    system($cmd) and die "Cannot fetch bug log
from merkel: $cmd failed with status $?";
    $log = "$CACHEDIR/$bug.log";
}

my $in = IO::File->new($log);
my $reader = Debbugs::Log->new($in);

my $lastrec = undef;
while (my $rec = $reader->read_record()) {
    $lastrec = $rec if $rec->{type} eq
'incoming-recv';
}

die "No incoming-recv records found" if not
defined $lastrec;
$in->close();

open (IN, "<", \$lastrec->{text});
my $h = Mail::Header->new(\*IN);
my $from = $h->get("From");
close(IN);

die "No From address in the last mail" if not
defined $from;

for my $f (Mail::Address->parse($from)) {
    print $f->address(), "\n";
}

exit 0;
```

Mole

Big index of data periodically mined from the archive, by Jeroen van Wolffelaar.

- Info: `http://wiki.debian.org/Mole`
- Source: `merkel:/org/qa.debian.org/mole/db/`
- Public source: `http://qa.debian.org/data/mole/db`

Databases I used:

- `desktopfiles`: all `.desktop` files in the archive
- `dscfiles-control`: all `debian/control` files

More databases:

`dscfiles-watch`, `lintian-`*version*,
`packages-debian-`*suite*`-bin` `packages-debian-`*suite*`-src`

Mole
Example code

```
#!/usr/bin/python

import bsddb
import re

DB = '/org/qa.debian.org/data/mole/db/dscfiles-control.moledb'

db = bsddb.btopen(DB, "r")

re_pkg = re.compile(r"^Package:\s+(\S+)\s*$", re.M)
re_tag = re.compile(r"^Tag: +([^\n]+?)(?:,|\s)*$", re.M)

for k, v in db.iteritems():
    m_pkg = re_pkg.search(v)
    if not m_pkg: continue
    m_tag = re_tag.search(v)
    if not m_tag: continue
    print "%s: %s" % (m_pkg.groups()[0], m_tag.groups()[0])
```

To access it, from any Debian machine:

```
ldapsearch -x -h db.debian.org -b dc=debian,dc=org "$@"
```

Example code:

```
# Count developers:
ldapsearch -x -h db.debian.org -b dc=debian,dc=org \
  '(&(keyfingerprint=*)(gidnumber=800))' | grep ^uid: | wc

# Stats by nationality:
ldapsearch -x -h db.debian.org -b ou=users,dc=debian,dc=org c \
  | grep ^c: | sort | uniq -c | sort -n | tail
```

Debian Developer's Packages Overview

Besides `developer.php` there is a repository with raw data at `http://qa.debian.org/data/ddpo/`.

How to read maintainer / comaintainer information:

- Location:
  `http://qa.debian.org/data/ddpo/results/ddpo_maintainers`
- `passwd`-like format, one maintainer per line.
- Comaintained packages are marked with a #:

```
;enrico@debian.org;NOID;Enrico Zini;buffy cnf dballe debtags debtags-edit
festival-it# guessnet launchtool libapt-front# libbuffy libdebtags-perl libept#
libwibble# openoffice.org-thesaurus-it polygen python-debian# tagcoll tagcoll2
tagcolledit thescoder;;;;;
```

Aggregated package descriptions

- All package descriptions of all architectures of sid and experimental:

  `http://people.debian.org/~enrico/AllPackages.gz`
- Same, but sid only:

  `http://people.debian.org/~enrico/AllPackages-nonexperimental.gz`
- In your system only: `grep-aptavail -sPackage,Description .`

Indexing and searching package descriptions

```
#!/usr/bin/python
"Create the package description index"

import xapian, re, gzip, deb822

tokenizer = re.compile("[^A-Za-z0-9_-]+")
# How we normalize tokens before indexing
stemmer = xapian.Stem("english")
def normalise(word):
    return stemmer.stem_word(word.lower())

# Index all packages
# ( wget -c
http://people.debian.org/~enrico/AllPackages.gz )
database = xapian.WritableDatabase( \
    "descindex", xapian.DB_CREATE_OR_OPEN)
input = gzip.GzipFile("AllPackages.gz")
for p in deb822.Packages.iter_paragraphs(input):
    idx = 1
    doc = xapian.Document()
    doc.set_data(p["Package"]);
    doc.add_posting(normalise(p["Package"]), idx);
    idx += 1
    for tok in tokenizer.split(p["Description"]):
        if len(tok) == 0: continue
        doc.add_posting(normalise(tok), idx);
        idx += 1
    database.add_document(doc);
database.flush()
```

```
#!/usr/bin/python
"Search the package description index"

import xapian, sys

# Open the database
database = xapian.Database("descindex")

# We need to stem search terms as well
stemmer = xapian.Stem("english")
def normalise(word):
    return stemmer.stem_word(word.lower())

# Perform the query
enquire = xapian.Enquire(database)
query = xapian.Query(xapian.Query.OP_OR, \
        map(normalise, sys.argv[1:]))
enquire.set_query(query)

# Show the matching packages
matches = enquire.get_mset(0, 30)
for match in matches:
    print "%3d%%: %s" % ( \
        match[xapian.MSET_PERCENT], \
        match[xapian.MSET_DOCUMENT].get_data())
```

Aggregated popcon frequencies

- `http://people.debian.org/~enrico/popcon-frequencies.gz`

```
#!/usr/bin/python
"Print the most representative packages in the system"

import gzip, math
freqs, local = {}, {}

# Read global frequency data
for line in gzip.GzipFile("popcon-frequencies.gz"):
    key, val = line[:-1].split(' ')
    freqs[key] = float(val)
docCount = freqs.pop('__NDOCS__')

# Read local popcon data
for line in open("/var/log/popularity-contest"):
    if line.startswith("POPULARITY"): continue
    if line.startswith("END-POPULARITY"): continue
    data = line[:-1].split(" ")
    if len(data) < 4: continue
    if data[3] == '<NOFILES>': # Empty/virtual
        local[data[2]] = 0.1
    elif len(data) == 4: # In use
        local[data[2]] = 1.
    elif data[4] == '<OLD>': # Unused
        local[data[2]] = 0.3
    elif data[4] == '<RECENT-CTIME>':
        local[data[2]] = 0.8 # Recently installed
```

```
# TFIDF package scoring function
def score(pkg):
    if not pkg in freqs: return 0
    return local[pkg] * math.log(docCount / freqs[pkg])

# Sort the package list by TFIDF score
packages = local.keys()
packages.sort(key=score, reverse=True)

# Output the sorted package list
for idx, pkg in enumerate(packages):
    print "%2d)  %s" % (idx+1, pkg)
    if idx > 30: break
```

Popcon-based suggestions

1. Submit `/var/log/popularity-contest` as a file form field called `scan` to `http://people.debian.org/ enrico/anapop`
2. Get a `text/plain` answer with a token
3. Get statistics with

   `http://people.debian.org/ enrico/anapop/stats/`*`token`*
4. Get package suggestions with

   `http://people.debian.org/ enrico/anapop/xposquery/`*`token`*

debtags data

Locally installed data sources:

- Package→tag mapping in `/var/lib/debtags/package-tags` (merges all configured tag sources)
- Facet and tag descriptions in `/var/lib/debtags/vocabulary` (merges all configured tag sources)
- Tags in the packages file: `grep-aptavail -sPackage,Tag .`

On the internet:

- `http://debtags.alioth.debian.org/tags/tags-current.gz`
- `http://debtags.alioth.debian.org/tags/vocabulary.gz`
- Other tag sources can be available (e.g. `http://www.iterating.org/tags/{tags-current,vocabulary}.gz`)

`tagcoll grep - tagcoll reverse - debtags search - debtags tagsearch - debtags dumpavail - debtags tag [add,rm,ls] - debtags smartsearch - ...`

debtags web queries

Suite of CGIs at `http://debtags.alioth.debian.org/cgi-bin/...`

- `fts/key1[/key2...]` Full text search
- `stags/key1[/key2...]` Smart tag search
- `sugg/pkg` Suggests tags for the package
- `unt[/num]` Packages with no tags, or no more than *num* tags
- `maint/email` Package list for a given maintainer
- `pkgs/tag1[/tag2...]` Packages with at least all these tags
- `pkg/pkgname` Package tags and description
- `qstats` Quick tag database statistics
- `patch` Submit an unreviewed patch

- `http://debtags.alioth.debian.org/js/debtags.js`
  Functions to easily interface to all these CGIs
- `http://debtags.alioth.debian.org/js/vocabulary.js`
  Javascript-accessible facet and tag descriptions

In need of more documentation

- What else do you know?

Conclusion

- There is no conclusion.
- Add more examples in the wiki.
- Post more examples in `debian-devel-announce`.
- Blog more examples.
- This can only be fun.