About Dyson

Dyson is a general-purpose operating system, a Debian derivative using the illumos kernel, libc, and SMF init system (formerly OpenSolaris). See FAQ.

It is not a successor of any existing or existed distributions based on illumos or OpenSolaris. Dyson is constructed from scratch to be like Debian as much as possible. Namely, most of Debian packages can be built on Dyson without any changes, and arch-independent packages (arch all in Debian terms) can be installed directly as is.

Project goals:

1. General-purpose operating system with OpenSolaris kernel (for desktops and servers)
2. Port GNU GLibc
3. Official Debian port
4. ~~Hijack illumos mainstream.~~

Get Dyson

Currently Dyson only runs on **x86_64** processors (it is *fully 64-bit*).
There is a LiveCD ISO image with a simple installer: https://dl.osdyson.org/iso/2019-11-15/

For downloading and using Dyson consider using the mirrors.

VCS repositories are available for anonymous read-only access:

- https://cgit.osdyson.org - Git repositories. Almost all development happens here.

Debian patch tracker: https://patches.osdyson.org/

If the APT repository cannot be verified due to missed public key (*The following signatures couldn't be verified because the public key is not available: NO_PUBKEY*),
get current singing key located at https://apt.osdyson.org/dyson.asc and use "apt-key" to add the key manually.
This key is usually automatically installed with the "dyson-archive-keyring" package.

It is possible to use Vagrant to set up a VirtualBox machine. See https://cgit.osdyson.org/vagrant/dyson.git.

Mailing lists

https://lists.osdyson.org/listinfo/dyson-devel - development of Dyson, technical topics, packaging, porting Debian packages, porting software to illumos.

Get involved

Try it. Report issues and wishes. Feel free to email to dyson-devel@osdyson.org.

Port Debian packages and various software to Dyson. Send patches to upstream and/or Debian package maintainers. Ideally any Debian package should need no more than a simple rebuild. See Porting guide.

See https://www.osdyson.org/issues to find out if you can help with an advise or coding.

Things to be done:

- ~~get X stack, X server should work;~~.
- ~~get Java stack;~~
- ~~haskell;~~
- ~~mono C#;~~
- ~~Free Pascal;~~
- Rust;
- bootstrap and port nixpkgs ;
- get zones;
- package Kernel Virtual Machine (kvm) and Qemu;

- cross-compiling Debian (Linux) to Dyson;
- port libusb #158;
- ~~full support for SMF, including restart on upgrade;~~
- get GNU-toolchain-friendly dtrace;
- switch to GRUB2 and be more friendly (= co-installable) with other systems;
- reading and writing linux filesystems (ext2/3/4, FUSE);
- ~~LibreOffice;~~
- Firefox;
- port Glibc;
- port Debian Installer.

How to

Prevent X11 from starting

Lightdm is the default desktop manager. It is started by the svc:/system/graphical-login:lightdm SMF service. Disable it to disable X11 autostart.

Switch to a virtual terminal

Make sure vtdaemon is installed and svc:/system/vtdaemon:default is enabled. Enable the svc:/system/console-login:vt2 SMF service for virtual terminal 2. Use CTRL+ALT+F2 to switch to VT 2.

Dyson mirrors

Russia

Thanks to Yandex.

- APT repository: http://mirror.yandex.ru/mirrors/apt.osdyson.org/apt/
- Downloads: http://mirror.yandex.ru/mirrors/apt.osdyson.org/dl/

The Netherlands

Thanks to NLUUG and Mike Hulsman.
Synchronised every 2 hours on the even hours.

- APT repository: http://ftp.nluug.nl/os/illumos/dyson/apt/ (also ftp and rsync)
- Downloads: http://ftp.nluug.nl/os/illumos/dyson/ftp/ (ftp and rsync)

Denmark

Thanks to http://dotsrc.org.
Synchronised every 4 hours.

- APT repository: https://mirrors.dotsrc.org/dyson/apt/ (also ftp and rsync)
- Downloads: https://mirrors.dotsrc.org/dyson/ftp/ (also ftp and rsync)

Germany

This is the primary site:

- APT repository: http://apt.osdyson.org
- Downloads: http://dl.osdyson.org

Creating new mirrors

Mirrors can be created with rsync:

- APT repository (38 GiB and growing):

```
rsync -a --delete --force --progress rsync://ftp.nluug.nl/dyson/apt/ /path/to/dyson/apt/mirror
/
# or
rsync -a --delete --force --progress rsync://apt.osdyson.org/apt/ /path/to/dyson/apt/mirror/
```

- Downloads (1.3 GiB):

```
rsync -a --delete --force --progress rsync://ftp.nluug.nl/dyson/ftp/ /path/to/dyson/ftp/mirror
/
# or
rsync -a --delete --force --progress rsync://dl.osdyson.org/dl/ /path/to/dyson/dl/mirror/
```

Cross-compiling

Assume Debian (Linux) to Dyson.

APT repository

Dyson's APT repository includes pre-compiled packages for Debian (Linux / amd64). Add this line to /etc/apt/sources.list:

```
deb [ arch=amd64 ] https://apt.osdyson.org unstable main
```

Building binutils

Both normal and cross binutils should be installed on Debian (Linux) system. Thus original (not patched) version of binutils is overwritten by the Dyson's binutils. This is pretty safe, because Dyson's patches only affect Solaris targets, also Dyson's binutils are usually newer that Debian's.

1. Get binutils source package from https://apt.osdyson.org/pool/main/b/binutils/ (you need version patched for Dyson), e. g.:

```
$ dget -u https://apt.osdyson.org/pool/main/b/binutils/binutils_2.28-5%2Bdyson1.dsc
```

2. Build normal binutils:

```
$ cd binutils-2.28
$ ./debian/rules stamps/control
$ dpkg-buildpackage --build=any
```

3. Build cross binutils. According to debian/README.cross it's quite simple:

```
$ cd binutils-2.28
$ TARGET=x86_64-pc-solaris2.11 ./debian/rules stamps/control
$ TARGET=x86_64-pc-solaris2.11 dpkg-buildpackage --build=any
```

4. Result:

```
...
find debian/binutils-x86-64-pc-solaris2.11 -depth -newermt 'Sat, 13 May 2017 13:33:58 +0300' -
print0 | \
    xargs -0r touch --no-dereference --date='Sat, 13 May 2017 13:33:58 +0300'
dpkg --build debian/binutils-x86-64-pc-solaris2.11 ..
dpkg-deb: building package 'binutils-x86-64-pc-solaris2.11' in '../binutils-x86-64-pc-solaris2
.11_2.28-5+dyson1_amd64.deb'.

$ dpkg -c ../binutils-x86-64-pc-solaris2.11_2.28-5+dyson1_amd64.deb
drwxr-xr-x root/root         0 2017-05-13 13:33 ./
drwxr-xr-x root/root         0 2017-05-13 13:33 ./usr/
drwxr-xr-x root/root         0 2017-05-13 13:33 ./usr/bin/
-rwxr-xr-x root/root     28176 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-addr2line
-rwxr-xr-x root/root     56856 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-ar
-rwxr-xr-x root/root    369928 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-as
-rwxr-xr-x root/root     23776 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-c++filt
-rwxr-xr-x root/root   3064344 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-dwp
-rwxr-xr-x root/root     28160 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-elfedit
-rwxr-xr-x root/root     93144 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-gprof
-rwxr-xr-x root/root   1185712 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-ld
hrwxr-xr-x root/root         0 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-ld.bfd link to
 ./usr/bin/x86_64-pc-solaris2.11-ld
-rwxr-xr-x root/root   5206392 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-ld.gold
-rwxr-xr-x root/root     41192 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-nm
-rwxr-xr-x root/root    222448 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-objcopy
-rwxr-xr-x root/root    353144 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-objdump
```

```
-rwxr-xr-x root/root     56856 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-ranlib
-rwxr-xr-x root/root    494112 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-readelf
-rwxr-xr-x root/root     28024 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-size
-rwxr-xr-x root/root     28144 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-strings
-rwxr-xr-x root/root    222456 2017-05-13 13:33 ./usr/bin/x86_64-pc-solaris2.11-strip
drwxr-xr-x root/root         0 2017-05-13 13:33 ./usr/lib/
drwxr-xr-x root/root         0 2017-05-13 13:33 ./usr/lib/x86_64-linux-gnu/
-rw-r--r-- root/root   1234664 2017-05-13 13:33 ./usr/lib/x86_64-linux-gnu/libbfd-2.28-solaris
-amd64.so
-rw-r--r-- root/root   1661968 2017-05-13 13:33 ./usr/lib/x86_64-linux-gnu/libopcodes-2.28-sol
aris-amd64.so
...
```

Building GCC

To be done

Dyson Installer

Requirements

To install Dyson you need:

- 64-bit PC (amd64 aka x86_64),
- CD-ROM drive,
- 1.5 GiB of RAM,
- 6 GiB of free space on a MBR partitioned disk,
- network access to one of the Dyson mirrors (maybe your own).

You can also use virtual machines such as VirtualBox or QEMU.

Installing on QEMU

This is tested on Debian amd64 with QEMU version 1.1.2.

```
$ qemu-img create -f qcow2 dyson.qcow2 10G
$ qemu-system-x86_64 -hda dyson.qcow2 -m 1500 -cdrom dyson-netinst-2013-05-05-1506.iso
```

Getting the installer

Download the lastest ISO image of install CD from http://ftp.osdyson.org/iso/.

Network access from the install CD

When booting from CD the startup script tries to bring networking up via DHCP.

If no DHCP servers available you have to configure network manually. To configure network manually switch to the console (press the F2 key), and use commands dladm, ipadm, route, also remember to update the /etc/resolv.conf file with vim or nano. See appropriate man pages, e. i. man ipadm, for detail.

Example:

1. Print available network interfaces:

   ```
   # dladm show-phys
   LINK         MEDIA                STATE      SPEED  DUPLEX    DEVICE
   e1000g0      Ethernet             unknown    1000   full      e1000g0
   ```

2. Bring the "e1000g0" interface up (aka plumb):

   ```
   # ipadm create-if e1000g0
   ```

   A few seconds later, if you type dladm show-phys again, you will see that the state of the interface is changed from "unknown" to "up" or "down" (cable connected or not). If the state is "down" there is no reason to go with this interface until you connect the cable.
3. Assign an address to the interface. You may want to try DHCP again:

   ```
   # ipadm create-addr -T dhcp e1000g0/dhcp
   ```

   or give it a static address, e. g.:

   ```
   # ipadm create-addr -T static -a 192.168.2.45/24 e1000g0/static
   ```

4. If using static IP, set up the default gateway (e. g. 192.168.2.1), otherwise dhcpagent will do it for you:

```
route -n add default -gateway 192.168.2.1
```

You can view routing table with netstat:

```
netstat -nr
```

5. Setup name servers in /etc/resolv.conf

Getting closed source drivers

Dyson does not include closed source drivers, such as mpt (see https://www.illumos.org/issues/3).
There are instruction by Jason Upton on how to get closed source drivers and use them during installation:
http://lists.osdyson.org/pipermail/dyson-devel/2013-April/000041.html

Post-install configuration

Getting updates

Since the installer is a network installer it uses "testing" repository to install base system. "testing" repository is known to work. After installation you may want to switch to "unstable".

Dual boot with GRUB2 on Debian

Given that Dyson is installed on the primary partition N of the first drive, add following lines to /etc/grub.d/40_custom and run grub-mkconfig -o /boot/grub/grub.cfg to updated GRUB2 menu.

```
menuentry "Dyson" {
set root=(hd0,N) # change N to the partition number (1 to 4)
chainloader +1
}
```

With this configuration Debian's GRUB2 will show the "Dyson" item in its menu and will run Dyson's GRUB if you choose that item ("chain loading").

Obviously, Dyson's GRUB should be installed into a partition, not into MBR (Debian's GRUB2 holds MBR).

Installation process

This section describes an optimistic installation process on a clean virtual machine.

LiveCD GRUB menu:

00-GRUB.png

LiveCD is booting:

01-booting.png

The installer's start screen:

02-welcome.png

The console. You can switch to the console (press F2) and execute any commands as root. To switch back to the installer press F1:

03-console.png

First of all, the installer searches for hard disks and ZFS pools. The installer can install Dyson to existing ZFS pool, or can create a new one.

04-hdd.png

For new ZFS pool you need a disk with one Solaris partition:

05-no-part.png

cfdisk from util-linux is used for editing partitions :

06-cfdisk.png

Solaris partition has id = 0xbf:

07-bf.png

After creating a Solaris partirion remember to choose "Write" in cfdisk menu, then - "Quit":
08-cfdisk-solaris.png

Now you are ready to install Dyson:
09-ready.png

The installer will format Solaris partition (with a single slice) and create ZFS root pool and minimal ZFS filesystems (including swap, /home and /):
10-format.png

Choosing how big the installed system will be:
10-profile.png

Then you will be asked to choose a Dyson mirror:
11-mirrors.png

Next, installation of a base system begins (using debootstrap):
12-installing.png

If you choose basic or desktop profile, additional packages will be installed (via apt-get):
12-apt-get.png

Then - enter a hostname (aka nodename):
15-hostname.png

And finally, enter root password:
16-root.png

If installing more than just a minimal system, you will be asked to create a regular user:
16-create-user.png

Then the installer will create boot archive:
17-bootarchive.png

And populate initial SMF repository:
18-smf.png

Dyson uses GRUB 1 for booting (aka GRUB legacy). It can be installed on master boot record (MBR) or on a partitions (which you've choosen for root ZFS pool). Note that is that partition is a logical partition, GRUB will be forcedly installed on MBR. If you are installing on an existing ZFS pool (e. g. created by OpenIndiana), you may want to just update GRUB menu, or completely skip GRUB configuration. In any case an example configuration is saved in /boot/grub/menu.lst on root filesystem.

19-grubinstall.png

Then, you are done:

20-done.png

Eject CD and reboot:

21-installedgrub.png

That was just the beginning...

22-login.png

Files

00-GRUB.png	22.4 KB	2013-05-05	Igor Pashev
01-booting.png	29.1 KB	2013-05-05	Igor Pashev
02-welcome.png	28.7 KB	2013-05-05	Igor Pashev
03-console.png	29.3 KB	2013-05-05	Igor Pashev
04-hdd.png	22.3 KB	2013-05-05	Igor Pashev
06-cfdisk.png	24.3 KB	2013-05-05	Igor Pashev
05-no-part.png	25.1 KB	2013-05-05	Igor Pashev
07-bf.png	30.5 KB	2013-05-05	Igor Pashev
08-cfdisk-solaris.png	25 KB	2013-05-05	Igor Pashev
09-ready.png	26.9 KB	2013-05-05	Igor Pashev

10-format.png	20.8 KB	2013-05-05	Igor Pashev
12-installing.png	21 KB	2013-05-05	Igor Pashev
15-hostname.png	21.5 KB	2013-05-05	Igor Pashev
16-root.png	22.2 KB	2013-05-05	Igor Pashev
17-bootarchive.png	20.7 KB	2013-05-05	Igor Pashev
18-smf.png	21.4 KB	2013-05-05	Igor Pashev
19-grubinstall.png	24.1 KB	2013-05-05	Igor Pashev
20-done.png	21.7 KB	2013-05-05	Igor Pashev
21-installedgrub.png	22.2 KB	2013-05-05	Igor Pashev
22-login.png	29.1 KB	2013-05-05	Igor Pashev
12-apt-get.png	20.2 KB	2013-09-15	Igor Pashev
16-create-user.png	23.6 KB	2013-09-15	Igor Pashev
10-profile.png	23.1 KB	2013-09-15	Igor Pashev
11-mirrors.png	23.6 KB	2013-09-15	Igor Pashev

C library

For a while Dyson uses illumos libc, **not** GNU libc.

More libraries to link

Some functions, expected to be in libc, really are in libsocket, libnsl and libresolv. Also Dyson uses GNU libiconv.
Workaround:

```
$ cat /usr/lib/x86_64-illumos/libc.so
INPUT(libc.so.1 AS_NEEDED(-lsocket -lnsl -lresolv -liconv))
```

Note, that this breaks sunld (illumos link editor). The same trick is used in Debian for libncurses.

Less libraries to link

librt, libphtread, libdl are dummy ("filters"). There is not need to link with these libraries, all functions expected to be in these libraries are in libc or ld.so.1 (not worry either).

Extended attributes

There are specific functions to work with extended attributes of files. These functions involve illumos library libnvpair. Functions known from Linux are provided by libattr (see #59).

See particular recipes.

Filesystem layout

/bin is a symbolic link to /usr/bin

illumos and Solaris have been using it for a long time. GNU/Linux is going to use it too. But now in Debian /bin and /usr/bin are separated. So additional modifications are required for some "low-level" packages like "sed", "ed", all shells, etc.

Debian multiarch

- x86_64-illumos for Dyson on x86_64 (64-bit) processors
- i386-illumos for Dyson on x86 (32-bit) processors (this distribution does not exist)

So Dyson on amd64 has directories /lib/x86_64-illumos and /usr/lib/x86_64-illumos.

Manpages layout

Some packages (e. g. "sudo") detect "SunOS" and put manpages in different directories, and dh_install cannot find some manpages after that.
Debian man page layout (aka GNU-style) is prefered.

Toolchain

Default compiler is GCC 9

The first version of GCC supporting x86_64-pc-solaris2.11 hosts is 4.7

GNU linker (ld) from GNU binutils is the default linker

Some packages fondly expect that if uname returns "SunOS", the linker is the SunOS linker.
It is wrong, see for example:

1. https://fedorahosted.org/augeas/ticket/289,
2. https://bugzilla.gnome.org/show_bug.cgi?id=685626,
3. http://bugs.debian.org/689656.

kbuild used to pass the "-i" to the linker; for sun ld this means "Ignore the LD_LIBRARY_PATH", for GNU ld - "Incremental link".

Init system is SMF

Logrotate rules need attention

Some rules for logrotate run init-scripts for sysvinit, e. g. Apache and Lighttpd.
See http://cgit.osdyson.org/dh-smf.git

Particular recipes

Missing d_type of struct dirent

Synopsis:

```
inputoutput.cpp:82:27: error: ‘struct dirent’ has no member named ‘d_type’; did you mean ‘d_name’?
                  if (dirp->d_type == DT_REG)
                             ^~~~~~
inputoutput.cpp:82:37: error: ‘DT_REG’ was not declared in this scope
                  if (dirp->d_type == DT_REG)
                                      ^~~~~~
```

Solution is to use dirfd and fstatat, as shown below.
Note that decent programs should use stat/fstat/fstatat anyway, because not checking d_type == DT_UNKNOWN *is a bug*!

```
--- opencv-2.4.9.1+dfsg1.orig/modules/contrib/src/inputoutput.cpp
+++ opencv-2.4.9.1+dfsg1/modules/contrib/src/inputoutput.cpp
@@ -7,6 +7,7 @@
     #include <tchar.h>
 #else
     #include <dirent.h>
+    #include <sys/stat.h>
 #endif

 namespace cv
@@ -72,14 +73,18 @@ namespace cv
             (void)addPath;
             DIR *dp;
             struct dirent *dirp;
+            int dfd;
+            struct stat st;
             if((dp = opendir(path.c_str())) == NULL)
             {
                 return list;
             }
+            dfd = dirfd(dp);

             while ((dirp = readdir(dp)) != NULL)
             {
-                if (dirp->d_type == DT_REG)
+                if ((0 == fstatat(dfd, dirp->d_name, &st, 0)) &&
+                    S_ISREG(st.st_mode))
                 {
                     if (exten.compare("*") == 0)
                         list.push_back(static_cast<std::string>(dirp->d_name));
```

struct ifreq

If struct ifreq causes problems (e. g. missing ifr_ifindex) consider using struct lifreq, see for example http://bugs.python.org/issue21287.

Too many arguments to function ‘getpwuid_r’, etc.

Compile with CPPFLAGS="-D_POSIX_C_SOURCE=199506L" or "-D_POSIX_PTHREAD_SEMANTICS -D_REENTRANT"

Missing strverscmp()

Use libiberty-dev package: link with -liberty, include libiberty/libiberty.h. The only known package with such a problem is tree (actually it includes strverscmp.c).

Debian symbols file

glibc defines FILE as _IO_FILE, illumos libc - as __FILE.
So C++ functions differ, and dh_makeshlibs can fail:

```
dpkg-gensymbols: warning: some new symbols appeared in the symbols file: see diff output below
dpkg-gensymbols: warning: some symbols or patterns disappeared in the symbols file: see diff output below
dpkg-gensymbols: warning: debian/libqt5webkit5/DEBIAN/symbols doesn't match completely debian/libqt5webkit5.sy
mbols
```

```
...
dh_makeshlibs: failing due to earlier errors
```

Fix:

```
@@ -2182,7 +2182,8 @@ libQt5WebKit.so.5 libqt5webkit5 #MINVER#
  _ZN3JSC7JSProxy9setTargetERNS_2VMEPNS_14JSGlobalObjectE@Base 5.2.0
  _ZN3JSC7JSScope13objectAtScopeEPS0_@Base 5.0.2
  _ZN3JSC7JSValue13isValidCalleeEv@Base 5.0.2
- _ZN3JSC7Options14dumpAllOptionsEP8_IO_FILE@Base 5.0.2
+ (arch=!illumos-amd64)_ZN3JSC7Options14dumpAllOptionsEP8_IO_FILE@Base 5.0.2
+ (arch=illumos-amd64)_ZN3JSC7Options14dumpAllOptionsEP6__FILE@Base 5.2.1+dfsg-6~dyson1
  _ZN3JSC7Options9s_optionsE@Base 5.0.2
  _ZN3JSC7Options9setOptionEPKc@Base 5.0.2
  _ZN3JSC7Profile10restoreAllEv@Base 5.0.2
```

Getting stack boundaries

On Linux we usually use pthread_getattr_np(), on Dyson we have thr_stksegment().

Be careful: if stack is "unlimited", ss_size will hold RLIM_INFINITY = -3 (see sys/resource.h) which definitely must not be interpreted as stack size! In this case lower stack boundary should be set to 0xFFFF800000000000L.

Example from Mono:

```
#elif (defined(__sun__) || defined(__illumos_kernel__))
#include <thread.h>
#include <errno.h>
    {
        int rc;
        stack_t s;

        do {
            rc = thr_stksegment(&s);
        } while (rc != 0 && errno == EAGAIN);

        info->stack_end = s.ss_sp;

        // Is stack "unlimited" ?
        if (s.ss_size == RLIM_INFINITY)
        {
            // XXX Lower stack boundary
            info->stack_start_limit = (void*)0xFFFF800000000000L;
        }
        else
        {
            info->stack_start_limit = (void*)((intptr_t)s.ss_sp - (intptr_t)s.ss_size);
        }
    }
#else
```

P. S. pthread_attr_get_np() is provided by libc since 5.10.18.git.2a44663-4 (2017-11-22).

Getting path of executable

Use const char * getexecname() from stdlib.h or

```
readlink("/proc/self/path/a.out", buf, bufsize);
```

http://stackoverflow.com/questions/933850/how-to-find-the-location-of-the-executable-in-c

Conflicting declarations of abs() in C++ sources

Synopsis:

```
In file included from ../../src/base/dict.h:40:0,
                 from freetype.cpp:152:
/usr/include/c++/4.8/cstdlib: In function ‘long long int std::abs(long long int)’:
/usr/include/c++/4.8/cstdlib:174:20: error: declaration of C function ‘long long int std::abs(long long int)’
conflicts with
   abs(long long __x) { return __builtin_llabs (__x); }
```

```
                    ^
In file included from /usr/include/stdlib.h:38:0,
                 from ../../include/Inventor/SoType.h:37,
                 from ../../include/Inventor/errors/SoError.h:39,
                 from ../../include/Inventor/errors/SoDebugError.h:36,
                 from ../../src/coindefs.h:95,
                 from freetype.cpp:44:
/usr/include/iso/stdlib_iso.h:122:12: error: previous declaration ‘int std::abs(int)’ here
 extern int abs(int);
            ^
```

Make sure C++ sources include cstdlib, not stdlib.h.

Missing IXDR_GET_LONG or similar *_LONG RPC macros

These macros are deprecated and removed from 64-bit Solaris or illumos. Glibc provides them via *_INT32 macros. Workaround:

```
#include <rpc/xdr.h>

// ...

#ifndef IXDR_GET_LONG
#define IXDR_GET_LONG(buf) ((long)IXDR_GET_U_INT32(buf))
#endif

#ifndef IXDR_PUT_LONG
#define IXDR_PUT_LONG(buf, v) ((long)IXDR_PUT_INT32(buf, (long)(v)))
#endif

#ifndef IXDR_GET_U_LONG
#define IXDR_GET_U_LONG(buf) ((u_long)IXDR_GET_LONG(buf))
#endif

#ifndef IXDR_PUT_U_LONG
#define IXDR_PUT_U_LONG(buf, v) IXDR_PUT_LONG(buf, (long)(v))
#endif
```

Boost threads unavailable on this platform

See /usr/include/boost/config/platform/solaris.hpp: -D_PTHREADS must be defined, use g++ -pthread

Rebuild script

Synopsis

```
# ./rebuild pandoc
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Build-Depends dependency for pandoc cannot be satisfied because candidate version of package libghc-aeson-d
ev can't satisfy version requirements

# ./rebuild libghc-aeson-dev
```

Purpose

Non-interactively update a package from Debian source repository.

Setup

APT repositories

```
# cat /etc/apt/sources.list
deb http://apt.osdyson.org unstable main
deb-src http://ftp.de.debian.org/debian sid main
```

Signing result

Add your GPG key to GPG agent to bypass passphrase

Dput

/etc/dput.cf:

```
[dyson]
login= pashev
fqdn= apt.osdyson.org
method= scp
incoming= /srv/apt/incoming
allow_dcut= 1
post_upload_command = ssh apt.osdyson.org /srv/apt/script/import-new-packages.sh && sudo apt-get update
```

Source code

```
#!/bin/sh

set -u
set -e
set -x

pkg="$1"
vertries=7

sudo apt-get -y build-dep "$pkg"
curver=`(apt-cache show $pkg || echo 'Version: 0~0') | awk '/Version:/ {print $2}'`

tmpdir=`mktemp -d "/var/tmp/pkg-${pkg}-XXXXXXX"`

cd "$tmpdir"
apt-get source "$pkg"
pkgdir=`find * -maxdepth 0 -type d | head -n 1`
echo "*** building in $pkgdir ***"
cd "$pkgdir"

dch --l '+dyson' 'Package for Dyson'
bump=0
while true; do
    newver=`dpkg-parsechangelog | awk '/Version:/ {print $2}'`
    if dpkg --compare-versions "$newver" gt "$curver"; then
        break
    else
```

```
        if [ $bump = 0 ]; then
            echo "There is already package version $newver in Dyson. Bump version? [y/N]"
            read bump
        fi
        case $bump in
        y|Y|yes|Yes)
            dch -i ''
            vertries=`expr $vertries - 1 || true`
            if [ $vertries = 0 ]; then
                echo "Version bump failed!"
                exit 1
            fi
            ;;
        *)
            echo "Aborted" >&2
            exit 1
            ;;
        esac
    fi
done

EDITOR=true dch -r --no-force-save-on-release
dpkg-buildpackage -sa
dput dyson `realpath ../*.changes`

cd
rm -fr -- "$tmpdir"

exit 0
```

Sustaining ported Debian packages

If Debian packages needs more attention than a simple patch to upstream sources, they need to be ported, and changes should be tracked in Git. After experimenting some time the best approach has been found.

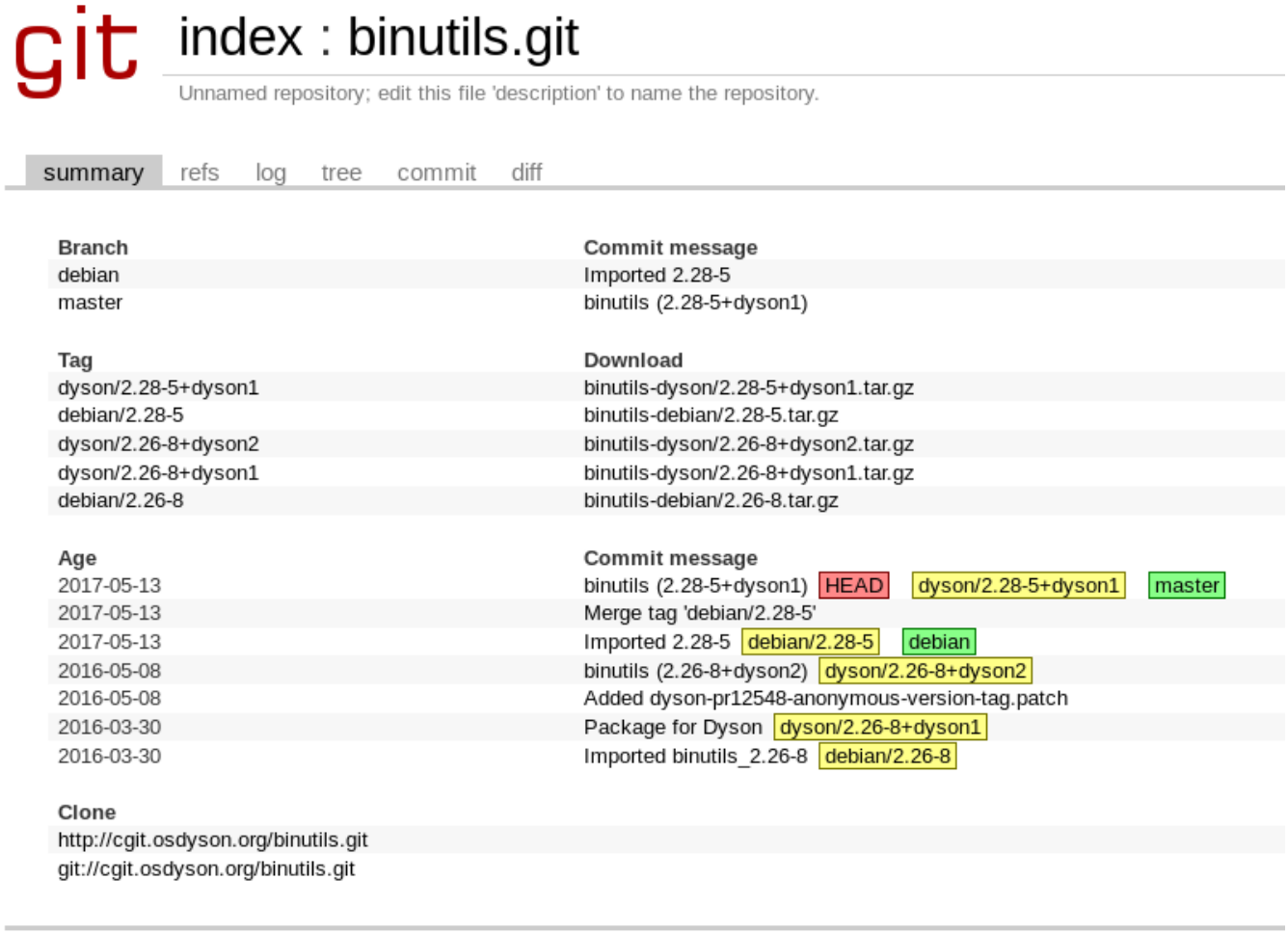
What's wrong with Debian VCS repositories

1. Some Debian packages aren't tracked in Git at all. Subversion, Bazaar are still quite popular. This affects essential and the most hard to port and maintain packages like Binutils, GCC, OpenJDK, etc.
2. Debian repositories can stale, that is new packages are uploaded to APT repositories, but VCS repositories aren't updated. It is especially painful when a new version is required to fulfill build dependencies.
3. Most of Debian's Git repositories are bloated with upstream sources, pristine tarballs, branches, etc. We don't need them all. It all needs a lot of storage.

Dyson's approach

As some of Debian packages do, we should track only the ./debian directory without upstream sources or tarballs. After all, we take them from Debian. We completely ignore Debian's VCS be it Git, Subversion or Bazaar. Instead
we start a git repository by importing the ./debian directory of some Debian version, and then we make and track our changes in the master branch. Vanilla Debian's version is tracked in the debian branch. When a new version
lands in Debian and need to be ported, we import it into the debian branch first (with no conflicts, etc.) and then merge the debian branch into the master branch, resolve conflicts, update Dyson specific parts etc.

See for example Binutils



Pitfalls

The .git directory is not friendly to dpkg format version 1.0. It can be moved away before building packages or dpkg-source can be told to ignore it.

Files

binutils.png	67.7 KB	2017-06-09	Igor Pashev