目次

tshark の導入

apt-get を使ったインストール

tshark を使いたいだけであれば、お手軽な apt-get がお奨めです。

```
root@plum:/usr/local/src# apt-get update
root@plum:/usr/local/src# apt-get install -y tshark
root@plum:/usr/local/src# thsark -v
```

TShark 1.10.6 (v1.10.6 from master-1.10) がインストールされたことがわかります。

ソースコードからのインストール

ただ apt-get だけでは対応できない場合 (PPA が存在しない場合やどうしても最新版を使いたい場合など) ソースコードを自力でコンパイルしてインストール必要があります。 そこで、その方法を tshark を例にやってみましょう。

まずは、最新の tshark を落としてきて展開します。

```
root@plum:/usr/local/src# wget
https://1.as.dl.wireshark.org/src/wireshark-1.10.8.tar.bz2
root@plum:/usr/local/src# tar jxf wireshark-1.10.8.tar.bz2
root@plum:/usr/local/src# cd wireshark-1.10.8/
```

次にconfigureのオプションを選択します。とりあえず、ヘルプをみてそれっぽいものを探しましょう。

```
root@plum:/usr/local/src/wireshark-1.10.8# ./configure --help
```

必要そうなのはこの辺。

```
Optional Features:
  --disable-option-checking  ignore unrecognized --enable/--with options
  --disable-FEATURE       do not include FEATURE (same as --enable-
FEATURE=no)
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --enable-static[=PKGS]  build static libraries [default=no]
  --disable-dependency-tracking  speeds up one-time build
  --enable-dependency-tracking   do not reject slow dependency extractors
  --enable-shared[=PKGS]  build shared libraries [default=yes]
  --enable-fast-install[=PKGS]
                          optimize for fast installation [default=yes]
  --disable-libtool-lock  avoid locking (might break parallel builds)
  --enable-osx-deploy-target
                          choose an OS X deployment target [default=major
                          release on which you're building]
```

```
  --disable-largefile     omit support for large files
  --enable-extra-gcc-checks
                          do additional -W checks in GCC [default=no]
  --enable-warnings-as-errors
                          treat warnings as errors (only for GCC or clang)
                          [default=no]
  --enable-silent-rules          less verbose build output (undo: `make
V=1')
  --disable-silent-rules         verbose build output (undo: `make V=0')
  --enable-usr-local      look for headers and libs in /usr/local tree
                          [default=yes]
  --enable-wireshark      build GTK+-based Wireshark [default=yes, if GTK+
                          available]
  --enable-packet-editor  add support for packet editor in Wireshark
                          [default=no]
  --enable-profile-build  build profile-ready binaries [default=no]
  --disable-gtktest       do not try to compile and run a test GTK+ program
  --disable-glibtest      do not try to compile and run a test GLIB program
  --enable-tshark         build TShark [default=yes]
  --enable-editcap        build editcap [default=yes]
  --enable-capinfos       build capinfos [default=yes]
  --enable-mergecap       build mergecap [default=yes]
  --enable-reordercap     build reordercap [default=yes]
  --enable-text2pcap      build text2pcap [default=yes]
  --enable-dftest         build dftest [default=yes]
  --enable-randpkt        build randpkt [default=yes]
  --enable-airpcap        use AirPcap in Wireshark [default=yes]
  --enable-dumpcap        build dumpcap [default=yes]
  --enable-rawshark       build rawshark [default=yes]
  --enable-pcap-ng-default
                          use the pcap-ng file format by default instead of
                          pcap [default=yes]
  --enable-ipv6           use IPv6 name resolution, if available
[default=yes]
  --enable-setcap-install install dumpcap with cap_net_admin and cap_net_raw
                          [default=no]
  --enable-setuid-install install dumpcap as setuid [default=no]

Optional Packages:
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --with-pic[=PKGS]       try to use only PIC/non-PIC objects [default=use
                          both]
  --with-gnu-ld           assume the C compiler uses GNU ld [default=no]
  --with-sysroot=DIR Search for dependent libraries within DIR
                        (or the compiler's sysroot if not specified).
  --with-gnutls=[yes/no]  use GnuTLS library [default=yes]
  --with-gcrypt=[yes/no]  use gcrypt library [default=yes]
  --with-libgcrypt-prefix=PFX
                          prefix where LIBGCRYPT is installed (optional)
  --with-qt=[yes/no]      use Qt instead of GTK+ [default=no]
```

```
  --with-libnl[=VERSION]  use libnl (force version VERSION, if supplied)
                          [default: yes, if available]
  --with-gtk3=[yes/no]    use GTK+ 3.0 instead of 2.0 [default=no]
  --with-libsmi=[DIR]     use libsmi MIB/PIB library [default=yes],
optionally
                          specify the prefix for libsmi
  --with-osx-integration  use OS X integration functions [default=yes, if
                          available]
  --with-pcap[=DIR]       use libpcap for packet capturing [default=yes]
  --with-pcap-remote      use libpcap remote capturing (requires libpcap)
  --with-zlib[=DIR]       use zlib (located in directory DIR, if supplied)
for
                          gzip compression and decompression [default=yes,
if
                          available]
  --with-lua[=DIR]        use liblua (located in directory DIR, if supplied)
                          for the Lua scripting plugin [default=yes, if
                          available]
  --with-portaudio[=DIR]  use libportaudio (located in directory DIR, if
                          supplied) for the rtp_player [default=yes, if
                          available]
  --with-dumpcap-group=GROUP
                          restrict dumpcap to GROUP
  --with-libcap[=DIR]     use libcap (located in directory DIR, if supplied)
                          for POSIX.1e capabilities management [default=yes,
                          if present]
  --with-ssl[=DIR]        use SSL crypto library (located in directory DIR,
if
                          supplied) [default=no]
  --with-krb5[=DIR]       use Kerberos library (located in directory DIR, if
                          supplied) to use in Kerberos dissection
                          [default=yes]
  --with-c-ares[=DIR]     use c-ares (located in directory DIR, if supplied)
-
                          supersedes --with-adns [default=yes, if present]
  --with-adns[=DIR]       use GNU ADNS (located in directory DIR, if
supplied)
                          [default=yes, if present]
  --with-geoip[=DIR]      use GeoIP (located in directory DIR, if supplied)
                          [default=yes, if present]
  --with-plugins[=DIR]    support plugins (installed in DIR, if supplied)
                          [default=yes, if possible]
```

ここから最低限必要**そうなものを**選択**します**。

```
root@plum:/usr/local/src/wireshark-1.10.8# time ./configure --disable-
wireshark  --enable-tshark \
> --enable-ipv6 --disable-gtktest --disable-glibtest \
> --disable-editcap --disable-capinfos \
> --disable-mergecap --disable-reordercap \
> --disable-text2pcap --disable-dftest \
```

```
> --disable-randpkt --disable-airpcap \
> --disable-dumpcap --disable-rawshark \
> --enable-pcap-ng-default \
> --without-lua --without-ssl
```

今回は tshark だけ欲しいのでこんな感じにします。 これを実行します。

```
checking build system type... armv7l-unknown-linux-gnueabihf
checking host system type... armv7l-unknown-linux-gnueabihf

...

checking for perl... /usr/bin/perl
checking for python... no
checking for bison... no
checking for byacc... no
checking for yacc... no
configure: error: I couldn't find yacc (or bison or ...); make sure it's
installed and in your path
```

error がでて途中で止まってしまいました。 ざっと読むと yacc がないって言っているようです yacc を入れましょう。

yacc もソースコードを取ってきてインストールすればいいのですが ここは簡単に apt-get を使って入れます。

```
root@plum:/usr/local/src/wireshark-1.10.8# apt-get install -y yacc
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package yacc
```

yacc なんていうパッケージはないと言われてしまいました。 そこで apt-cache search を使ってどれに含まれているか探してみましょう。

```
root@plum:/usr/local/src/wireshark-1.10.8# apt-cache search yacc
bison - YACC-compatible parser generator
byacc-j - Berkeley YACC parser generator extended to generate Java code
cup - LALR parser generator for Java(tm)
erlang-parsetools - Erlang/OTP parsing tools
exuberant-ctags - build tag file indexes of source code definitions
gob2 - GTK+ Object Builder
jflex - lexical analyzer generator for Java
libbison-dev - YACC-compatible parser generator - development library
libparse-recdescent-perl - Perl module to create and use recursive-descent
parsers
libparse-yapp-perl - Perl module for creating fully reentrant LALR parser OO
Perl modules
python-ply - Lex and Yacc implementation for Python2
python-ply-doc - Lex and Yacc implementation for Python (documentation)
python-pyparsing - Python parsing module
```

```
python-pyparsing-doc - Python parsing module, documentation package
python3-ply - Lex and Yacc implementation for Python3
python3-pyparsing - Python parsing module, Python3 package
9base - Plan 9 userland tools
btyacc - Backtracking parser generator based on byacc
byacc - public domain Berkeley LALR Yacc parser generator
cscope - interactively examine a C program source
cutils - C source code utilities
fp-utils - Free Pascal - utilities dependency package
fp-utils-2.6.2 - Free Pascal - utilities
global - Source code search and browse tools
happy - Parser generator for Haskell
jikespg - Jikes Parser Generator
kimwitu - Compiler development tool, complementary to lex and yacc
kimwitu++ - A (syntax-)tree-handling tool (term processor)
kimwitu-doc - documentation for compiler development tool Kimwitu
lemon - LALR(1) Parser Generator for C or C++
libghc-highlighting-kate-dev - syntax highlighting library based on Kate
syntax descriptions
libghc-highlighting-kate-doc - library documentation for highlighting-kate;
documentation
libghc-highlighting-kate-prof - highlighting-kate library with profiling
enabled; profiling libraries
menhir - Parser generator for OCaml
mono-jay - LALR(1) parser generator oriented to Java/CLI
pccts - The Purdue Compiler Construction Tool Set (PCCTS).
peg - recursive-descent parser generators for C
perl-byacc - Berkeley LALR parser generator, Perl version
python-lesscpy - LessCss Compiler for Python 2.x
python-parsley - pattern-matching language based on OMeta and Python
python3-lesscpy - LessCss Compiler for Python 3.x
racc - Ruby LALR parser generator
sloccount - programs for counting physical source lines of code (SLOC)
styx - combined parser/scanner generator for C/C++
```

さっきの configure **のエラーでも** yacc (or bison or ...)**と書いてあったし** 一番最初**にある** bison **をインストールしましょう。**

```
root@plum:/usr/local/src/wireshark-1.10.8# apt-get install -y bison
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libbison-dev libsigsegv2 m4
Suggested packages:
  bison-doc
The following NEW packages will be installed:
  bison libbison-dev libsigsegv2 m4
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
2 not fully installed or removed.
Need to get 756 kB of archives.
```

```
After this operation, 2004 kB of additional disk space will be used.
Get:1 http://ports.ubuntu.com/ubuntu-ports/ trusty/main libsigsegv2 armhf
2.10-2 [14.0 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports/ trusty/main m4 armhf
1.4.17-2ubuntu1 [178 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports/ trusty/main libbison-dev armhf
2:3.0.2.dfsg-2 [337 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports/ trusty/main bison armhf
2:3.0.2.dfsg-2 [227 kB]
Fetched 756 kB in 4s (178 kB/s)
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LANG = "ja_JP.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_MESSAGES to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
Selecting previously unselected package libsigsegv2:armhf.
(Reading database ... 24673 files and directories currently installed.)
Preparing to unpack .../libsigsegv2_2.10-2_armhf.deb ...
Unpacking libsigsegv2:armhf (2.10-2) ...
Selecting previously unselected package m4.
Preparing to unpack .../m4_1.4.17-2ubuntu1_armhf.deb ...
Unpacking m4 (1.4.17-2ubuntu1) ...
Selecting previously unselected package libbison-dev:armhf.
Preparing to unpack .../libbison-dev_2%3a3.0.2.dfsg-2_armhf.deb ...
Unpacking libbison-dev:armhf (2:3.0.2.dfsg-2) ...
Selecting previously unselected package bison.
Preparing to unpack .../bison_2%3a3.0.2.dfsg-2_armhf.deb ...
Unpacking bison (2:3.0.2.dfsg-2) ...
Setting up postgresql-client-9.3 (9.3.4-1) ...
update-alternatives: using /usr/share/postgresql/9.3/man/man1/psql.1.gz to
provide /usr/share/man/man1/psql.1.gz (psql.1.gz) in auto mode
update-alternatives: error: error creating symbolic link
`/usr/share/man/man7/DROP_LANGUAGE.7.gz.dpkg-tmp': No such file or directory
dpkg: error processing package postgresql-client-9.3 (--configure):
 subprocess installed post-installation script returned error exit status 2
dpkg: dependency problems prevent configuration of postgresql-9.3:
 postgresql-9.3 depends on postgresql-client-9.3; however:
  Package postgresql-client-9.3 is not configured yet.

dpkg: error processing package postgresql-9.3 (--configure):
 dependency problems - leaving unconfigured
Setting up libsigsegv2:armhf (2.10-2) ...
Setting up m4 (1.4.17-2ubuntu1) ...
Setting up libbison-dev:armhf (2:3.0.2.dfsg-2) ...
Setting up bison (2:3.0.2.dfsg-2) ...
update-alternatives: using /usr/bin/bison.yacc to provide /usr/bin/yacc
```

```
(yacc) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/yacc.1.gz
because associated file /usr/share/man/man1/bison.yacc.1.gz (of link group
yacc) doesn't exist
Processing triggers for libc-bin (2.19-0ubuntu6) ...
Errors were encountered while processing:
 postgresql-client-9.3
 postgresql-9.3
localepurge: Disk space freed in /usr/share/locale: 0 KiB
localepurge: Disk space freed in /usr/share/man: 0 KiB

Total disk space freed by localepurge: 0 KiB

E: Sub-process /usr/bin/dpkg returned an error code (1)
```

これで bison がインストールできました。 再び configure を実行してみましょう。

```
checking for perl... /usr/bin/perl
checking for python... no
checking for bison... bison -y
checking for bison... /usr/bin/bison
checking for flex... no
checking for lex... no
checking for flex... no
configure: error: I couldn't find flex; make sure it's installed and in your
path
```

無事 bison を入れたのでクリア・・・と思いきや今度は flex がないと言っているようです。 同様に apt-get を使って flex をインストールします。

```
root@plum:/usr/local/src/wireshark-1.10.8# apt-get install -y flex
```

インストール完了 configure を実行。

```
checking for bison... /usr/bin/bison
checking for flex... flex
checking lex output file root... lex.yy
checking lex library... -lfl

...

checking for GNU sed as first sed in PATH... yes
checking if profile builds must be generated... no
checking for pkg-config... no
checking for GLIB - version >= 2.14.0... no
*** A new enough version of pkg-config was not found.
*** See http://www.freedesktop.org/software/pkgconfig/
configure: error: GLib 2.14.0 or later distribution not found.
```

flex の部分はクリアしましたが 今度は Glib のバージョンが古いと文句を言われているようです tshark

に GUI はないのでいらないはずなんですが・・・ ともかくこれも apt-get でインストールします。

```
root@plum:/usr/local/src/wireshark-1.10.8# apt-get install -y libglib2.0-dev
```

何度目かの configure

```
checking if profile builds must be generated... no
checking for pkg-config... /usr/bin/pkg-config
checking for GLIB - version >= 2.14.0... yes (version 2.40.0)
checking for uic... no

...

checking pcap.h usability... no
checking pcap.h presence... no
checking for pcap.h... no
configure: error: Header file pcap.h not found; if you installed libpcap
from source, did you also do "make install-incl", and if you installed a
binary package of libpcap, is there also a developer's package of libpcap,
and did you also install that package?
```

Glib はクリアしました。 今度は、libpcap がないと言っています。

```
root@plum:/usr/local/src/wireshark-1.10.8# apt-get install -y libpcap0.8-dev
```

もういい加減にパスさせてくれても・・・と思いつつ configure を実行。

```
checking for broken pcap-config... no
checking pcap.h usability... yes
checking pcap.h presence... yes
checking for pcap.h... yes
checking for pcap_open_dead... yes
checking for pcap_freecode... yes
checking whether pcap_breakloop is present... yes

...

The Wireshark package has been configured with the following options.
                     Build wireshark : no
                        Build tshark : yes
                      Build capinfos : no
                       Build editcap : no
                       Build dumpcap : no
                      Build mergecap : no
                    Build reordercap : no
                      Build text2pcap : no
                        Build randpkt : no
                         Build dftest : no
                      Build rawshark : no
```

```
   Save files as pcap-ng by default : yes
  Install dumpcap with capabilities : no
             Install dumpcap setuid : no
                  Use dumpcap group : (none)
                        Use plugins : yes
                    Use Lua library : no
                 Use Python binding : no
                   Build rtp_player : no
             Build profile binaries : no
                   Use pcap library : yes
                   Use zlib library : yes
               Use kerberos library : no
                 Use c-ares library : no
               Use GNU ADNS library : no
                Use SMI MIB library : no
             Use GNU crypto library : no
             Use SSL crypto library : no
          Use IPv6 name resolution : yes
                   Use gnutls library : no
     Use POSIX capabilities library : no
                  Use GeoIP library : no
                     Use nl library : no
```

libpcap をクリアしようやくパスしたようです。 次はこれをコンパイルしましょう。 (事前に gcc などは入れておいてください)

```
root@plum:/usr/local/src/wireshark-1.10.8# time make
```

ようやくコンパイル終了。

```
make[2]: Leaving directory `/usr/local/src/wireshark-1.10.8/doc'
make[1]: Leaving directory `/usr/local/src/wireshark-1.10.8'

real    146m58.819s
user    114m6.972s
sys     8m15.467s
```

コンパイルには結構時間がかかります。

さて動くか確認してみましょう。

```
root@plum:/usr/local/src/wireshark-1.10.8# ./tshark -v
```

TShark 1.10.8 (Git Rev Unknown from unknown) と表示されるので最新版が完成しているもよう。 ちょっとパケットが取れるかも確認してみましょう。

```
root@plum:/usr/local/src/wireshark-1.10.8# ./tshark -i eth0 arp -c 3
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
  1   0.000000 00:01:8e:e4:4f:2f -> ff:ff:ff:ff:ff:ff ARP 60 Who has
192.168.130.20?  Tell 192.168.130.71
```

```
1   2   0.945578 d0:67:e5:1a:7b:ac -> ff:ff:ff:ff:ff:ff ARP 60 Who has
192.168.130.106?  Tell 192.168.130.104
  3   0.994780 00:01:8e:e4:4f:2f -> ff:ff:ff:ff:ff:ff ARP 60 Who has
192.168.130.20?  Tell 192.168.130.71
3
```

危険だから root で実行するなと怒られてはいるものの無事パケットは取れているもよう。 目的は達成できたので後はこれをインストールして完了です。