

Building a MySQL Server version on Unix

Jörg Brühe
Senior Production Engineer
Sun Microsystems, MySQL Build Team



*These slides released under the Creative Commons
Attribution-Noncommercial-Share Alike License*



Introduction

- **Subject:** Building the server on Unix
does neither cover other components (GUI tools, connectors)
nor builds on other platforms (Windows, Netware)
- **Presenter:** Jörg Brühe
member of the MySQL Build Team since 2004
Joerg.Bruehe@Sun.com
- **Mailing lists I watch:**
mysql@lists.mysql.com
internals@lists.mysql.com



Agenda

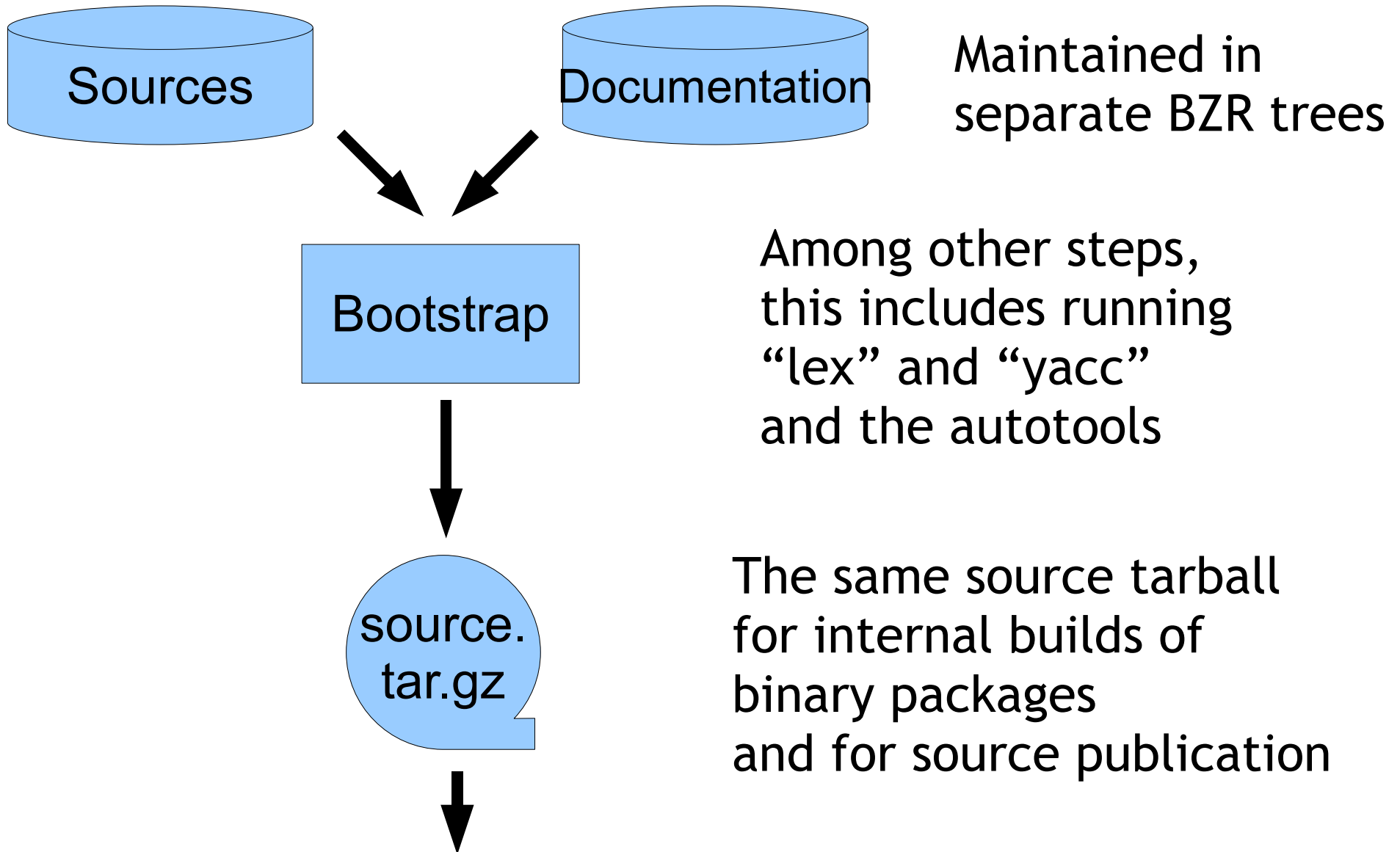
- The Big Picture
- Bootstrap: Create a Source Tarball
- Actions on the Build Platforms
- Handling of the Source Tree
- Hints for Build and Development
- Script “cmd_export_build”



The Big Picture

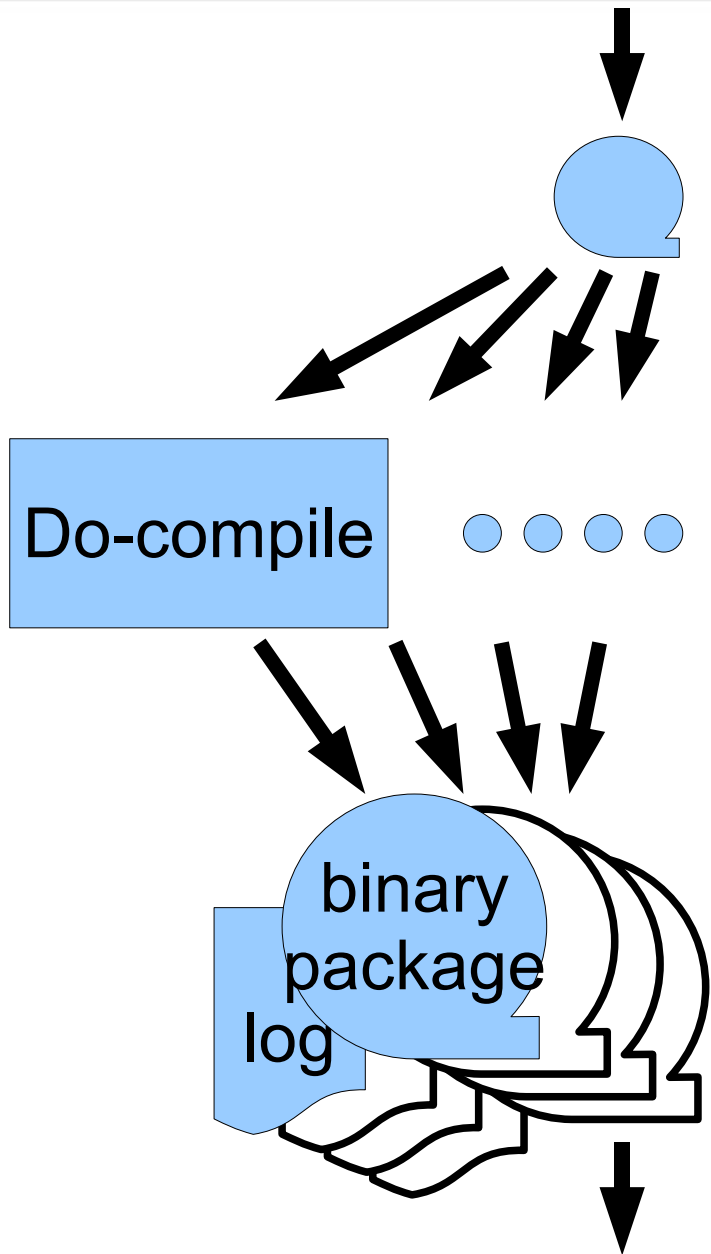


Flow of Actions (1): Create Source Tarball





Flow of Actions (2): Build and Test



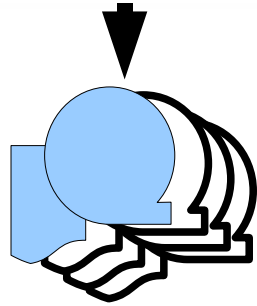
Source tarball

For all platforms in parallel:
Copy source tarball to it,
build debug server, test it,
build optimized server, test it,
create binary package

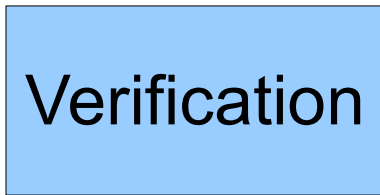
Get back all packages and logs
to the central host



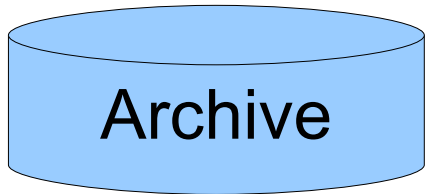
Flow of Actions (3): Verify and Publish



Packages and logs from all platforms



Check files in package, install the package, start the server, issue some SQL commands, stop, uninstall



All build files are archived



Packages are copied to mirrors world-wide



Bootstrap: Create a Source Tarball



Bootstrap(1): Collect Everything

- Get current sources of MySQL server:
bzip export
- Take version number from “configure.in”
and construct target file name
- Create “ChangeLog”
covering the set difference from last version tag
- Add files related to documentation:
manual, help tables, “INSTALL-*” texts, OS X “ReadMe”



Bootstrap(2): Combine into Source Tarball

- Run “BUILD/compile-dist”
This includes running the autotools,
then a “configure” including everything,
then “make” which will call bison (yacc) and flex (lex)
- Create source.tar.gz
by calling “make dist”
- Final rename (if needed) and repack
“cluster” uses own numbering,
zip files for Windows users



Advantages of the Source Tarball

- Several tools are not needed after Bootstrap:
 - Autotools
unless you want to modify `configure.in` or `Makefile.am`
 - Lex and yacc
unless you want to modify the parser
- Documentation is added:
`INSTALL-BINARY`, `INSTALL-SOURCE`, `INSTALL-WIN-SOURCE`,
`mysql.info`, `MacOSX/ReadMe.txt`, various man pages
- Data for help tables is added: `fill_help_tables.sql`
- ChangeLog is generated



Other Differences to a BZR Export

- Several auxiliary files are generated
- Several scripts are configured
- Error message files got extracted by language
- An initial database for installation on Windows is generated
- Some files which are irrelevant for the build don't get distributed



Actions on the Build Platforms



Role of Central Machine

- Bootstrap is run on a central machine
- All tools and definitions are available there, too lists of platforms, hosts, compiler options by platform, ...
- Build hosts allow ssh access for the build user
- Central runs a script which forks for all platforms
- Each child controls one build on one platform including copying input, tools, and results
- Multiple configurations are run in sequence



Actions on Central per Build Platform

Determine host, options, special settings

```
rsync source.tar.gz script \  
      platform:DIR/
```

```
ssh platform DIR/script
```

```
rsync platform:DIR/log logs/
```

```
rsync platform:DIR/package dist/
```

Commands run twice: first debug, then optimized



Steps on the Build Platform (1)

Check MD5 of source tarball

```
gunzip < source.tar.gz | tar xf -
```

Fix timestamps if in future if NTP doesn't work

Apply patches if any for specific platforms

Touch some files to prevent rebuild if bad timestamp

Construct “configure” flags from options

```
configure
```

Construct “make” flags from options

```
make
```



Steps on the Build Platform (2)

Run unittests

If debug build: save server for later reuse

If optimized build: get debug server back

```
scripts/make_binary_distribution
```

```
make clean
```

```
cd test-dir
```

```
gunzip < package.tar.gz | tar xf -
```

```
cd package
```

```
make test-bt # target may vary
```

Check linkage of C and C++ to libraries built



For Other Package Formats

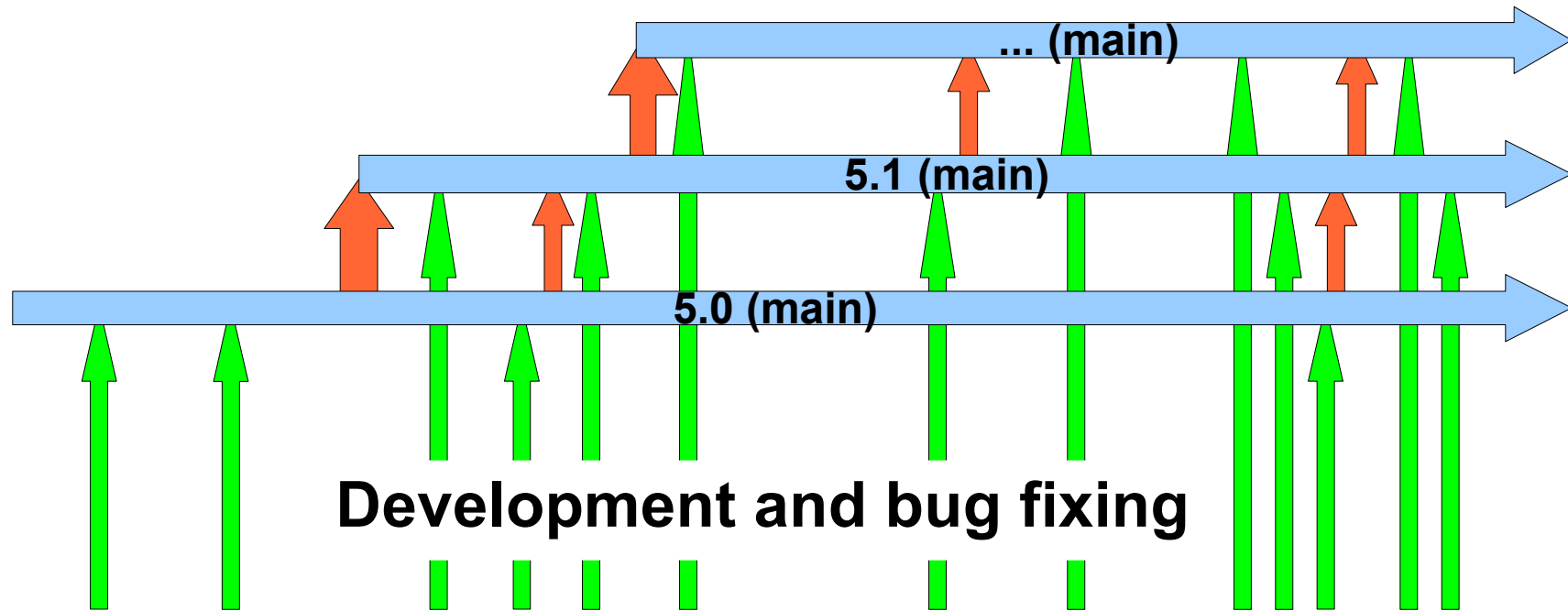
- Previous pages apply to tar.gz binary packages
- Specific formats depot, pkg, dmg are done by repacking the tar.gz contents some with different configuration or additional scripts
- RPMs are handled like tar.gz but use own scripts RPM build is fully controlled by the spec file



Handling of the Source Tree



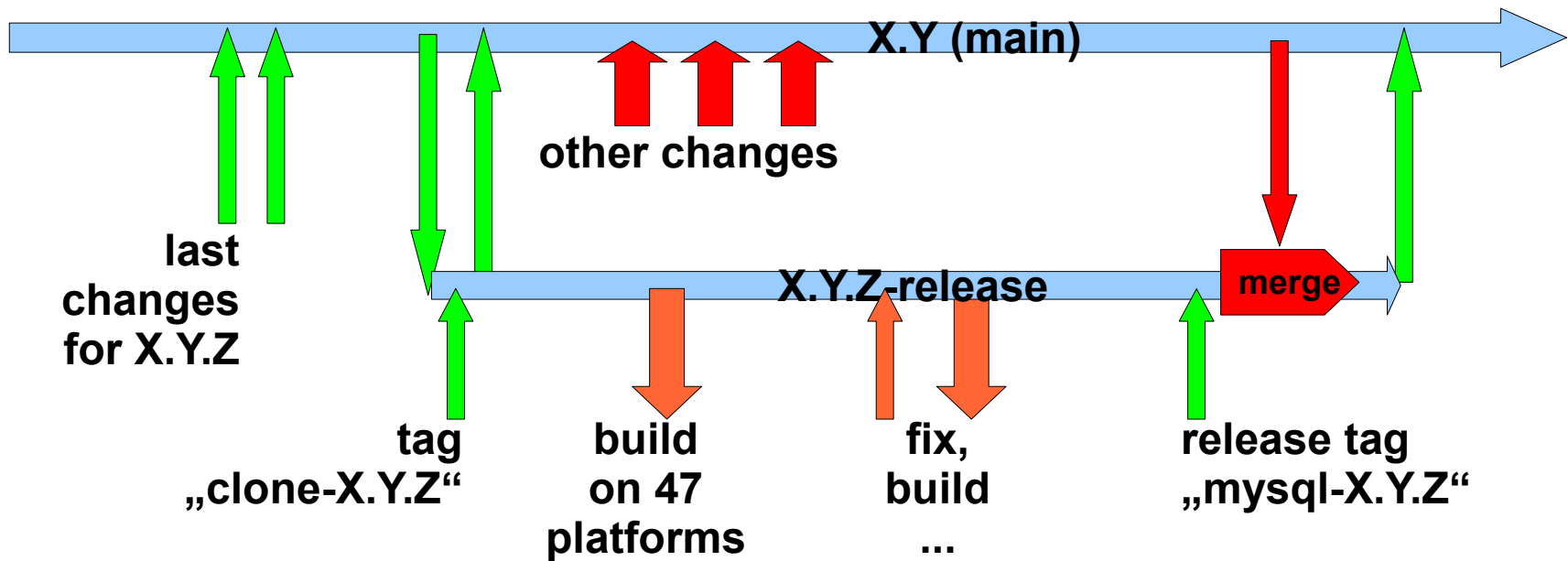
Parallel Source Trees in Development



- Newer release series is branched from older one
- Bug fixes go into oldest applicable series
- Changes are merged up into newer series



Separate Tree for Release Build



- A separate branch is used for the release build to isolate it from other changes
- Any fixes during the build are applied to that release clone and finally merged back
- Use tag “mysql-X.Y.Z” to get specific sources



Hints for Build and Development



Lessons Learnt on the Build Hosts

- Do not rely on having GNU programs
Non-Linux Unix often lacks features, especially “tar”
- Be prepared for unsynchronized time
 - “make” relies on timestamps
 - NTP may break on virtual machines
 - very recent source tarball may then fail to build
- Consider Murphy's law
With 47 build platforms, some problem will hit you:
Expect failures in any place, and write messages



Hints for Developers and Contributors

- If you add new files, check with “make dist”:
`BUILD/compile-dist`
`make dist`
`cd someplace ; tar xzvf dist.tar.gz`
`configure --foo --bar`
`make`
- If you add dependencies or introduce threads, check all combinations with/without
`--without-server`
`--disable-thread-safe-client`



Which Source to Use

You need the source tarball

- ... for documentation including on-line help
- ... or for reduced platform requirements (autotools, yacc, bison)
- ... or for some auxiliary files on non-Unix (unless you create them on Unix first)

You need sources from development (Launchpad)

- ... for access before release build



To Build from Source

```
BUILD/autorun.sh      # for BZR sources

configure             # add your options
make                  # GNU make preferred
make test-foo        # pick your choice

scripts/make_binary_distribution
# does more than "make install"
```



scripts/make_binary_distribution (tar.gz)

Hand-coded in 5.0, uses “make dist” in 5.1

Sadly, no “--help”: read the comments

- “--no-strip” to avoid stripping (5.0 only)
- “--with-ndbcluster” or not
In 5.1, rather use the CGE tree if you want cluster
- “--tmp=/absolute/path”
- “--suffix=...” adds info to the package name
- Platform recognition is automatic
Use “--platform=...” or “--machine=...” if needed



Scripts in “BUILD/”

- **BUILD/autorun.sh**
Run the autotools in the proper order, required for a build from BZR sources
- **BUILD/compile-dist**
Run the autotools, do a full configure, make, required to prepare for a “make dist”
- **BUILD/build_mccge.sh (5.1 only)**
Cluster Team tool for CGE source builds
- **All others: Not used in release builds provided “as-is”, no regular maintenance**



The InnoDB Plugin (5.1 only)

- `storage/innobase`
“builtin” version of InnoDB
- `storage/innodb_plugin` (new in 5.1.38)
“plugin” version of InnoDB, 1.0.4, enhanced scaling
- Configured by default with InnoDB
Will fail with gcc 3 and on some other platforms
Disable by “`--without-plugin-innodb_plugin`”
Ignore configure warnings!
See manual (5.1.38) for description
Replaces old InnoDB version in 5.4



Script “cmd_export_build”



Script: "cmd_export_build" (1)

This script does all steps of a server release build, starting from a local BZR tree on a developer's machine

```
1  #! /bin/sh
2  #
3  #      cmd_export_build      Mimic a release build as closely as possible
4  #
5  #      Start from a source (BZR) tree "$DIR_SRC",
6  #      export to a temporary dirextory "$DIR_EXP",
7  #      roll a source tarball, copy that to a build dir "$DIR_PLAT",
8  #      run "configure" + "make".
9  #
10 # Joerg Bruehe, MySQL Build Team, Sun Microsystems
11 # Copyright 2009 Sun Microsystems
12 #
13 # 2009-Sep-14  Added "--skip-dist" option

14 usage() {
15 echo
16 echo "$0 - Mimic a release build on a local host"
17 echo
18 echo "Usage:"
19 echo
20 echo "  $0 [Options] "
21 echo
22 echo "Options:"
23 echo
```

...



Script: "cmd_export_build" (2)

```
...
24 echo "--config-option=<option>    - passed to 'configure' (cumulative)"
25 echo "--help                - print this help text"
26 echo "--skip-dist           - start from a "make dist" done previously"
27 echo "                        (this implies --skip-export)"
28 echo "--skip-export          - start from an export done previously"
29 echo "                        (to use sources which were modified in the exported tree)"
30 echo "--tree=<bzr-tree> - use this tree as the source (default: current directory)"
31 echo
32 echo "Export and final build will be done to/in directories 'export' and 'platform'"
33 echo "having the same parent as '<bzr-tree>', any previous contents will be deleted."
34 echo "Logs will be written into sibling directory 'logs'."
35 exit 1
36 }

37 parse_arguments() {
38 CONF_OPT=""

39 for ARG do
40     case "$ARG" in
41         --config-option=*) CONF_OPT="$CONF_OPT `echo \"$ARG\" | sed -e 's;--config-
option=;;'\`" ;;
42         --help)           usage;;
43         --skip-dist)     SKIPDIST="yes" ;;
44         --skip-export)   SKIPEXP="yes" ;;
45         --tree=*)       DIR_SRC=`echo "$ARG" | sed -e "s;--tree=;;"\` ;;
46         *)
47             echo "Unknown argument '$ARG'"
48             usage
49             ;;
50     esac
51 done
52 }
```



Script: "cmd_export_build" (3)

```
...
53 #####
54 DIR_SRC=`pwd`

55 parse_arguments "$@"

56 if [ -n "$DIR_SRC" -a -d "$DIR_SRC" -a -d "$DIR_SRC/.bzip" ]
57 then :
58 else echo "'DIR_SRC' = '$DIR_SRC' is no BZR tree: ABORT"; usage
59 fi

60 DIR_PARENT=`dirname $DIR_SRC`
61 DIR_EXP="$DIR_PARENT/export"
62 DIR_PLAT="$DIR_PARENT/platform"
63 DIR_LOG="$DIR_PARENT/logs"

64 echo "About to start '$0' using these directories:"
65 echo "sources:      '$DIR_SRC'"
66 echo "export:        '$DIR_EXP'"
67 echo "platform:      '$DIR_PLAT'"
68 echo "logs:          '$DIR_LOG'"
69 if [ "$SKIPDIST" != "yes" ] ; then
70     if [ "$SKIPEXPORT" != "yes" ] ; then
71         echo "Doing a new export from the source tree."
72     else
73         echo "Re-using the previous export from the source tree."
74     fi
75 else
76     echo "Re-using the previous \"make dist\" from the source tree."
77 fi
78 echo "Configure options: '$CONF_OPT'"
79 echo "Last chance to abort (10 seconds) - hit Ctrl-C!"
80 sleep 10
...
```



Script: "cmd_export_build" (4)

```
...
81         if [ ! -d $DIR_EXP ]
82         then mkdir $DIR_EXP ; echo "Created '$DIR_EXP'" ; fi
83         if [ ! -d $DIR_PLAT ]
84         then mkdir $DIR_PLAT ; echo "Created '$DIR_PLAT'" ; fi
85         if [ ! -d $DIR_LOG ]
86         then mkdir $DIR_LOG ; echo "Created '$DIR_LOG'" ; fi

87 LOGFILE="$DIR_LOG/`basename $DIR_SRC`-`date '+%y%m%d-%H%M'`.log"
88 if [ -f $LOGFILE ]
89 then echo "Log file '$LOGFILE' exists: ABORT"; exit 1 ; fi

90 LOGPIPE="$DIR_LOG/pipe.$$"
91 if [ -e $LOGPIPE ]
92 then echo "Log pipe '$LOGPIPE' exists: ABORT"; exit 1 ; fi
93 mkfifo $LOGPIPE

94 # redirect all following output into the log
95 # Plain "exec" does not work in a shell pipe, use trick:
96 tee $LOGFILE <$LOGPIPE &
97 exec >$LOGPIPE 2>&1

98 # write summary info
99 echo "`date '+%y%m%d-%H%M%S'` Start '$0 $*'"
100 echo "sources:      '$DIR_SRC'"
101 echo "export:        '$DIR_EXP'"
102 echo "platform:      '$DIR_PLAT'"
103 echo "logs:          '$DIR_LOG'"
104 echo "configure:     '$CONF_OPT'"

105 # remove the pipe - it will be used until the process terminates
106 rm $LOGPIPE
107 # for debugging ...
108 set -x
```



Script: "cmd_export_build" (5)

```
...
109 #####
110 #
111 # Skip "Bootstrap" if desired to use the old source tarball
112 #

113 if [ "$SKIPDIST" != "yes" ] ; then

114     cd $DIR_PLAT ; rm -fr * .[a-z]*

115     #####
116     #
117     # Mimic the essential actions from "Bootstrap"
118     #

119     if [ "$SKIPEXP" != "yes" ] ; then

120         # Cleanup first
121         cd $DIR_EXP ; rm -fr * .[a-z]*

122         # Perform a "bzip export" as done by the "Bootstrap" tool.
123         cd $DIR_SRC
124         rmdir $DIR_EXP
125         bzip export $DIR_EXP
126         bzip log --long --limit=2 --show-ids > $DIR_EXP/ChangeLog
127     fi

128     # Do the initial build in the "export" directory
129     cd $DIR_EXP
130     BUILD/compile-dist

131     # Produce the source tarball
132     make dist

...
```



Script: “cmd_export_build” (6)

```
...
133         #####
134         #
135         # Mimic the essential action from "Do-compile-all"
136         #

137         cp mysql-[456]*.tar.gz $DIR_PLAT

138 fi
139 # "$SKIPDIST" != "yes"

140 #####
141 #
142 # Mimic the essential actions from "Do-compile" (on the build host)
143 #

144 if [ "$SKIPDIST" != "yes" ] ; then

145         cd $DIR_PLAT
146         # Unpack the source
147         tar xzvf mysql-[456]*.tar.gz
148         cd mysql-[456]*
149         pwd

150 else

151         cd $DIR_PLAT/mysql-[456]*
152         pwd
153         make clean
154 fi

155 ./configure $CONF_OPT
156 make
```



Thank you for your attention!

Any Questions?