Contents

List of Figures

List of Tables

Listings

0.1 QA Issue: glibc-locale

After placing the line

```
DEPENDS_append = " libgcc"
```

in the kernel recipe, the following “QA Issue” began occurring:

```
ERROR: glibc-locale-2.32-r0 do_package: QA Issue: glibc-locale: Files/directories were insta
  /usr/lib/locale
Please set FILES such that these items are packaged. Alternatively if they are unneeded, avo
glibc-locale: 1 installed and not shipped files. [installed-vs-shipped]
ERROR: glibc-locale-2.32-r0 do_package: Fatal QA errors found, failing task.
ERROR: Logfile of failure stored in: /edg/build-csky-toolchain-5.12-locale-fix/tmp/work/csky
ERROR: Task (/mnt/hdd/EDG/dachuan-research/gs500/poky/meta/recipes-core/glibc/glibc-locale_2
```

A new .bbappend was added to the meta-csky layer to handle this with the single line

```
FILES_${PN} = "${localedir}"
```

however this did not solve the QA issue.

A suspicious test on the variable PACKAGE_NO_GCONV in the do_install() function in glibc-locale.inc was discovered. The variable PACKAGE_NO_GCONV is initialized to ”0” in libc-package.bbclass. After modifying this to ”1” the QA issue was resolved. Note

that this is just a temporary fix; the real fix should override the variable in the meta-csky layer.

In the process of discussing this on #yocto, khem mentioned that we should also include this patch:

```
diff --git a/meta/classes/libc-package.bbclass b/meta/classes/libc-package.bbclass
index de3b4250c7..a6c3447e54 100644
--- a/meta/classes/libc-package.bbclass
+++ b/meta/classes/libc-package.bbclass
@@ -259,6 +259,7 @@ python package_do_split_gconvs () {
                 "mipsisa64r6el":" --uint32-align=4 --little-endian ", \
                 "riscv64": " --uint32-align=4 --little-endian ", \
                 "riscv32": " --uint32-align=4 --little-endian ", \
+                "csky": " --uint32-align=4 --little-endian ", \
                 "i586":    " --uint32-align=4 --little-endian ", \
                 "i686":    " --uint32-align=4 --little-endian ", \
                 "x86_64":  " --uint32-align=4 --little-endian "  }
```

1. After making all these changes and rebuilding `bitbake -c populate_sdk_ext core-image-minimal` from scratch, the only error resulting is the libgcc.a error. Rebuilding a second time also resulted in just the one libgcc.a error.
2. Commenting out the FILES line in the glibc-locale_2.32.bbappend file and rebuilding from scratch also resulted in just the one libgcc.a error. Rebuilding a second time also result in just the one libgcc.a error.
3. Reverting PACKAGE_NO_GCONV back to 0 (with FILES still commented out in glibc-locale_2.32.bbappend) caused the glibc-locale QA issue to return.

0.1.1 Final root-cause determination

For the following discussion refer to the documentation in the Configuration, Compilation, and Staging and Package Splitting sections in the Overview and Concepts manual.

In a nutshell, a QA error/issue means that there were files or directories in the staging area that were not covered by the FILES variable.

The root cause of the QA issue is that, for the default value of the PACKAGE_NO_CONV variable, the directory /usr/lib/locale was created in the staging area as part of the do_install task but not specified in the FILES variable.

The glibc do_compile stage generates two folders into the /usr/lib folder, gconv and locale:

```
/edg/build-csky-toolchain-5.12-xorg-5/tmp/work/csky-poky-linux/glibc-locale/2.32-r0/image/us
```

```
total used in directory 24 available 2408295240
drwxr-xr-x 4 randyy randyy  4096 Jul  6 14:18 .
drwxr-xr-x 5 randyy randyy  4096 Jul  6 14:18 ..
drwxr-xr-x 2 randyy randyy 12288 Jul  6 14:18 gconv
drwxr-xr-x 2 randyy randyy  4096 Jul  6 14:18 locale
```

The gconv folder contains many files. The locale folder is empty.

Line three of the following glibc-locale.inc function creates the locale directory in the staging area (`localedir` has the value `/usr/lib/locale` according to bitbake.conf)). Line 5 copies the gconv folder and all files within it to the staging area.

```
1  do_install() {
2          copy_locale_files ${bindir} 0755
3          copy_locale_files ${localedir} 0644
4          if [ ${PACKAGE_NO_GCONV} -eq 0 ]; then
5                  copy_locale_files ${libdir}/gconv 0755
6                  copy_locale_files ${datadir}/i18n 0644
7          else
8                  # Remove the libdir if it is empty when gconv is not copied
9                  find ${D}${libdir} -type d -empty -delete
10         fi
11         copy_locale_files ${datadir}/locale 0644
12         install -m 0644 ${LOCALETREESRC}/SUPPORTED ${WORKDIR}/SUPPORTED
13 }
```

Listing 1: do_install() from glibc-locale.inc

Note that the QA error can also be resolved in the following ways:

1. The variable `PACKAGE_NO_GCONV` can be defined to be non-zero. However, this causes the files in the gconv folder not to be copied into the staging area.
2. A command can be added to the glibc-locale.inc function do_install function to remove the locale folder:

```
1  do_install() {
2          copy_locale_files ${bindir} 0755
3          copy_locale_files ${localedir} 0644
4          if [ ${PACKAGE_NO_GCONV} -eq 0 ]; then
5                  copy_locale_files ${libdir}/gconv 0755
6                  copy_locale_files ${datadir}/i18n 0644
7                  find ${D}${libdir}/locale -type d -empty -delete
8          else
9                  # Remove the libdir if it is empty when gconv is not
                       copied
10                 find ${D}${libdir} -type d -empty -delete
11         fi
12         copy_locale_files ${datadir}/locale 0644
13         install -m 0644 ${LOCALETREESRC}/SUPPORTED ${WORKDIR}/SUPPORTED
14 }
```

Listing 2: Modified do_install() (not recommended)

However this is not recommended as it may break other image builds.

The fix was to append the locale directory under `/usr/lib` to FILES in the glibc .bbappend file:

```
FILES_${PN} += "${libdir}/locale"
```

More info from #yocto:

```
11:48 <yates> packaging question: will a QA issue be flagged if
              empty subdirectories in the source exist that were
              not created in the destination ($(D))?
11:49 <yates> by inference i have determined the answer to this
              question is "yes" but I want to double-check
11:53 <smurray> yates: packaging only looks in ${D}, so I'm pretty
                sure the answer is no
11:54 <yates> smurray: it must be looking at source
              directories/files too in order to do this QA issue
              check, right?
11:55 <smurray> yates: no?
11:55 <yates> http://paste.ubuntu.com/p/yXMj22P4Hr/
11:55 <yates> how could that possibly be true? what is it comparing
              to to determine there is an issue?
11:56 <qschulz> yates: the check is done between what's in ${D} and
                what's in FILES_*
11:56 <smurray> yates: that's telling you the directory exists in
                ${D}, but no FILES_*
11:56 <qschulz> if after everything's been packaged, there are
                still files, you get a QA Issue
11:56 <smurray> qschulz: sniped me ;)
11:57 <yates> ah, ok.
11:57 <rburton> the assumption is that everything that appeared in
                do_install should be in a package
11:57 <rburton> so if something gets installed but not packaged,
                you get that warning
11:58 <qschulz> and IIRC, empty directories still need to be
                packaged. So if your target always create some
                directory but not always put things in it, you
                might need to add /some/dir/ in addition to
                /some/dir/my-file in FILES_*
11:58 <qschulz> your makefile/cmake/meson/whatever target*
11:58 <rburton> correct, empty directories are still things that
                need to be packaged
11:59 <rburton> either package it, or don't install it
11:59 <yates> rburton: ok you seriously confused me with "if
              something gets installed but not packaged". I thought
              installing IS packaging. Or at least the first step,
```

```
              i.e., copying stuff to a staging area, which then
              subsequently gets packaged
12:00 <rburton> yeah we went through this last week
12:00 <rburton> do_install puts stuff into a staging area,
                do_package splits it up into packages
12:01 <rburton> packaging is just turning the output of do_install
                into packages by doing what FILES_* says
12:01 *** yumasi QUIT Remote host closed the connection
12:01 *** yumasi JOIN
12:03 <yates> ok let me regroup/rethink this then. thanks for the
              (re)clarification.
12:04 <rburton> https://docs.yoctoproject.org/overview-manual/concepts.html#configuration-co
12:04 <rburton> 4.3.5.3 is do_install, 4.3.5.4 is do_package
12:05 <yates> +1
```

0.1.2 Updated Root Cause

Although the previous root cause was correct, a rebuild from scratch revealed it did not solve the entire problem. There were several other subsequent issues and/or erroneous settings which remained. The following occurred along that journey:

1. Instrumented functions in poky/meta/classes/package.bbclass (see appendix **??**) to print out debugging information:
   (a) line 203, printed out `oldfiles` which is FILES variable for a pkg.
   (b) line 1382, printed out pkg
   (c) line 1399, printed out pkg.file
2. The BBFILES in /pathpoky/meta-csky/conf/layer.conf had an extraneous entry:

   ```
   BBFILES += "${LAYERDIR}/recipes-*/*/*.bb \
               ${LAYERDIR}/classes/*/*.bbclass \
               ${LAYERDIR}/recipes-*/*/*.bbappend"
   ```

   should be

   ```
   BBFILES += "${LAYERDIR}/recipes-*/*/*.bb \
               ${LAYERDIR}/recipes-*/*/*.bbappend"
   ```

3. Found that instead of debugging the glibc-locale recipe problems by building the toolchain/image (bitbake -c populate_sdk_ext core-image-minimal) it is instead only necessary to build the recipe: bitbake glibc-locale. This is faster.
4. Searched for FILES being processed by using the -e option to bitbake via `bitbake glibc-locale -e | grep FILES > /files1.txt` and found there were no FILES_glibc-locale outputs.

5. I was able to see all packages processed by the do_package log for the glibc-locale recipe from the instrumented functions in package.bbclass. Found there was no package glibc-locale being processed but instead many different packages were generated, e.g., glibc-gconvs.

   I concluded that the problem is that I was given wrong information (in this case), both by folks on the #yocto irc channel and by the error message issued in the bitbake command. The correct FILES to issue is **NOT** FILES_${PN} nor FILES_glibc-locale, but the PN used must be one of the packages processed by the glibc-locale recipe. When I tried FILES_glibc-gconvs, I started seeing the /usr/lib/locale folder in the FILES output via -e.

6. Once it was verified that /usr/lib/locale was being supplied correctly by the FILES_ variable, I began to get this problem:

```
ERROR: glibc-locale-2.32-r0 do_package_qa: Error executing a python function in exec_py

The stack trace of python calls that resulted in this exception/failure was:
File: 'exec_python_func() autogenerated', lineno: 2, function: <module>
     0001:
 *** 0002:do_package_qa(d)
     0003:
File: '/mnt/hdd/EDG/dachuan-research/gs500/poky/meta/classes/insane.bbclass', lineno: 1
     1095:            package_qa_handle_error("pkgname",
     1096:                    "%s doesn't match the [a-z0-9.+-]+ regex" % package, d)
     1097:
     1098:        warn_checks, error_checks = parse_test_matrix("QAPATHTEST")
 *** 1099:        package_qa_walk(warn_checks, error_checks, package, d)
     1100:
     1101:        warn_checks, error_checks = parse_test_matrix("QAPKGTEST")
     1102:        package_qa_package(warn_checks, error_checks, package, d)
     1103:
File: '/mnt/hdd/EDG/dachuan-research/gs500/poky/meta/classes/insane.bbclass', lineno: 7
     0719:                    elf = None
     0720:            for func in warnfuncs:
     0721:                func(path, package, d, elf, warnings)
     0722:            for func in errorfuncs:
 *** 0723:                func(path, package, d, elf, errors)
     0724:
     0725:    for w in warnings:
     0726:        package_qa_handle_error(w, warnings[w], d)
     0727:    for e in errors:
File: '/mnt/hdd/EDG/dachuan-research/gs500/poky/meta/classes/insane.bbclass', lineno: 3
     0356:        return
     0357:
     0358:    #if this will throw an exception, then fix the dict above
     0359:    (machine, osabi, abiversion, littleendian, bits) \
```

```
 *** 0360:        = oe.elf.machine_dict(d)[target_os][target_arch]
     0361:
     0362:    # Check the architecture and endiannes of the binary
     0363:    is_32 = (("virtual/kernel" in provides) or bb.data.inherits_class("module
     0364:             (target_os == "linux-gnux32" or target_os == "linux-muslx32" or \
Exception: KeyError: 'csky'

ERROR: Logfile of failure stored in: /edg/build-csky-toolchain-5.12-xorg-15/tmp/work/cs
ERROR: Task (/mnt/hdd/EDG/dachuan-research/gs500/poky/meta/recipes-core/glibc/glibc-loc
NOTE: Tasks Summary: Attempted 471 tasks of which 470 didn't need to be rerun and 1 fai

Summary: 1 task failed:
  /mnt/hdd/EDG/dachuan-research/gs500/poky/meta/recipes-core/glibc/glibc-locale_2.32.bb
Summary: There was 1 ERROR message shown, returning a non-zero exit code.
```

This is being caused by the csky architecture not being included in the core oe QA subsystem dictionary `machine_dict` located in the `poky/meta/lib/oe/elf.py` file.

Found from the end of that file that the csky architecture can be added via `PACKAGEQA_EXTRA_MACHDEFFUNCS`, for example as shown here.

Randy Yates
Apex Semiconductors (USA) Company Limited
984-368-8148 (cell, work)
919-412-8994 (cell, personal)