QGIS Application - Bug report #13962
Text delimited file with carriage return line endings will not load in QGIS

2015-12-08 09:27 AM - Dewey Dunnington

Status:	Closed		
Priority:	Normal		
Assignee:			
Category:	Data Provider/Delimited Text		
Affected QGIS version:	2.12.0	**Regression?:**	No
Operating System:		**Easy fix?:**	No
Pull Request or Patch supplied:	No	**Resolution:**	end of life
Crashes QGIS or corrupts data:	No	**Copied to github as #:**	21976

Description

I am trying to load a .csv with carriage return line endings (the default for Microsoft Excel in Mac OSX). This has been broken for several iterations of QGIS and is not an issue when loading the file into other programs (e.g. R). Loading .csv files from Excel is one of the most common tasks in our lab and this is a large stumbling block when converting users to QGIS.

History

#1 - 2015-12-08 09:37 AM - Jürgen Fischer

Note: via OGR it works (ie. "Add Vector Layer" / drag & drop into qgis window)

#2 - 2015-12-08 09:46 AM - Jürgen Fischer

QGIS (Qt's TextStream) expects "\
\
" (CRLF; windows/dos style) or "\
" (LF; unixoid) as line ending. Nowadays "\
" is apparently normal on Mac - unfortunately Excel on Mac seems to differ (and still follows what was usual before OSX).

#3 - 2015-12-08 09:52 AM - Dewey Dunnington

I agree it's odd behaviour and definitely Excel's fault, but it's difficult to run tutorials having to tell people they have to open files and save them using OpenOffice in order to get their field data into QGIS. If somebody tells me where to look I'm happy to try to fix!

#4 - 2015-12-08 01:38 PM - Sebastian Dietrich

Since the corresponding Qt-Bug is set to *won't fix* you would have to work on the delimited text provider, probably by creating a subclass of QTextStream and using it at this point.

#5 - 2015-12-08 06:08 PM - Dewey Dunnington

If it were a Python issue I'd happily fix it, but I know nothing about C++ so it would be a bad idea for me to try. QTextStream::readLine() calls QTextStream::readLineInto(), which looks like this (in Qt 5.5):

```
bool QTextStream::readLineInto(QString *line, qint64 maxlen)
{
```

```
    Q_D(QTextStream);
    // keep in sync with CHECK_VALID_STREAM
    if (!d->string && !d->device) {
        qWarning("QTextStream: No device");
        if (line && !line->isNull())
            line->resize(0);
        return false;
    }

    const QChar *readPtr;
    int length;
    if (!d->scan(&readPtr, &length, int(maxlen), QTextStreamPrivate::EndOfLine)) {
        if (line && !line->isNull())
            line->resize(0);
        return false;
    }

    if (Q_LIKELY(line))
        line->setUnicode(readPtr, length);
    d->consumeLastToken();
    return true;
}
```

I don't know enough about C++ to figure out what d is or what the value of QTextStreamPrivate::EndOfLine might be, but it may be a start?

#6 - 2015-12-08 10:31 PM - Nathan Woodrow

This is how EndOfLine is used in the Qt source

```
 case EndOfLine:
 778            if (ch == QLatin1Char('\
')) {
 779                foundToken = true;
 780                delimSize = (lastChar == QLatin1Char('\
')) ? 2 : 1;
 781                consumeDelimiter = true;
 782            }
 783            lastChar = ch;
 784            break;
```

#7 - 2015-12-08 10:42 PM - Nathan Woodrow

The main issue here is that stream the file so we can't just do a normal replace like you would.

We could do something in the UI like "Corrently OSX Excel line endings" which would mean we need to read, update and write the file back, or read into memory.

Thoughts. It's annoying having to add work around but it is an annoying as hell bug given how it's OS X and Excel.

#8 - 2015-12-11 07:31 AM - Dewey Dunnington

- File textmodule.py added

The following works in PyQt4, but I did not heavily test. For tests I did see the uploaded file. Perhaps this could be converted to C++?

```
from PyQt4.QtCore import QTextStream, QString

#subclass QTextSream
class MacAwareQTextStream(QTextStream):

  def __init__(self, qstring):
    QTextStream.__init__(self, qstring)
    self.newline = None

  def readLineWithNewline(self, newline, bufferlen=128, prefix=QString(u"")):
    start = self.pos()
    buf = self.read(bufferlen)
    if buf.contains(newline):
      pos = buf.indexOf(newline)
      line = buf.left(pos)
      if not self.seek(start+pos+newline.length()):
        self.seek(line.length())
      return line.prepend(prefix)
    elif buf.length() < bufferlen:
      #is last line
      return buf.prepend(prefix)
    else:
      #try bigger buffer
      return self.readLineWithNewline(newline, bufferlen*2, buf.prepend(prefix))

  def findNewline(self, bufferlen):
    buf = self.read(bufferlen)
    if buf.contains("\
\
"):
      return QString("\
\
")
    elif buf.contains("\
"):
      return QString("\
")
    elif buf.contains("\
"):
      buf2 = self.read(1)
      if buf2 == "\
":
        return QString("\
\
")
      else:
```

```
            return QString("\
")
        elif self.atEnd(): #no new line in file
            return QString("\
")
        else:
            return self.findNewline(bufferlen*2)

    def readLine(self):
        bufferlen = 1024
        if self.newline is None:
            #need to determine newline chars
            self.newline = self.findNewline(bufferlen)
            self.seek(0)
        return self.readLineWithNewline(self.newline, bufferlen)
```

#9 - 2016-06-30 01:44 AM - Matthias Meisser

- Target version set to Version 2.14

I confirm this bug for 2.14 (Essen) and 2.2 (Valmiera).

Usually GIS users aren`t nessesarly IT experts, it would be nice, if QGIS avoids crashes by unexpected data :)

I see your point, that the bug couldn't fixed upstream at the component itself. On the other hand, it would be nice when there would be an embedded workaround.

My usecase was: http://koenigstuhl.geog.uni-heidelberg.de/~mover/rostock.zip

#10 - 2017-05-01 01:06 AM - Giovanni Manghi

- Regression? set to No

- Easy fix? set to No

#11 - 2019-03-09 03:07 PM - Giovanni Manghi

- Resolution set to end of life

- Status changed from Open to Closed

End of life notice: QGIS 2.18 LTR

Source:

http://blog.qgis.org/2019/03/09/end-of-life-notice-qgis-2-18-ltr/

QGIS 3.4 has recently become our new Long Term Release (LTR) version. This is a major step in our history – a long term release version based on the massive updates, library upgrades and improvements that we carried out in the course of the 2.x to 3x upgrade cycle.

We strongly encourage all users who are currently using QGIS 2.18 LTR as their preferred QGIS release to migrate to QGIS 3.4. This new LTR version will receive regular bugfixes for at least one year. It also includes hundreds of new functions, usability improvements, bugfixes, and other goodies. See the relevant changelogs for a good sampling of all the new features that have gone into version 3.4

Most plugins have been either migrated or incorporated into the core QGIS code base.

We strongly discourage the continued use of QGIS 2.18 LTR as it is now officially unsupported, which means we'll not provide any bug fix releases for it.

You should also note that we intend to close all bug tickets referring to the now obsolete LTR version. Original reporters will receive a notification of the ticket closure and are encouraged to check whether the issue persists in the new LTR, **in which case they should reopen the ticket**.

If you would like to better understand the QGIS release roadmap, check out our roadmap page! It outlines the schedule for upcoming releases and will help you plan your deployment of QGIS into an operational environment.

The development of QGIS 3.4 LTR has been made possible by the work of hundreds of volunteers, by the investments of companies, professionals, and administrations, and by continuous donations and financial support from many of you. We sincerely thank you all and encourage you to collaborate and support the project even more, for the long term improvement and sustainability of the QGIS project.

Files

maccsv.csv	471 Bytes	2015-12-08	Dewey Dunnington
textmodule.py	2.54 KB	2015-12-11	Dewey Dunnington