import sys

import logging

from optparse import OptionParser

from flvlib import \_\_versionstr\_\_

from flvlib import tags

from flvlib import helpers

from flvlib.astypes import MalformedFLV

log = logging.getLogger('flvlib.debug-flv')

log.setLevel(logging.ERROR)

def debug\_file(filename, quiet=False, metadata=False):

try:

f = open(filename, 'rb')

except IOError, (errno, strerror):

log.error("Failed to open `%s': %s", filename, strerror)

return False

flv = tags.FLV(f)

if not quiet:

print "=== `%s' ===" % filename

try:

tag\_generator = flv.iter\_tags()

for i, tag in enumerate(tag\_generator):

if quiet:

# If we're quiet, we just want to catch errors

continue

# Print the tag information

print "#%05d %s" % (i + 1, tag)

# Print the content of onMetaData tags

if (isinstance(tag, tags.ScriptTag)

and tag.name == "onMetaData"):

helpers.pprint(tag.variable)

if metadata:

return True

except MalformedFLV, e:

message = e[0] % e[1:]

log.error("The file `%s' is not a valid FLV file: %s",

filename, message)

return False

except tags.EndOfFile:

log.error("Unexpected end of file on file `%s'", filename)

return False

f.close()

return True

def process\_options():

usage = "%prog [options] files ..."

description = ("Checks FLV files for comformance with the FLV "

"specification. Outputs a list of tags and, "

"if present, the content of the onMetaData script tag.")

version = "%%prog flvlib %s" % \_\_versionstr\_\_

parser = OptionParser(usage=usage, description=description,

version=version)

parser.add\_option("-s", "--strict", action="store\_true",

help="be strict while parsing the FLV file")

parser.add\_option("-q", "--quiet", action="store\_true",

help="do not output anything unless there are errors")

parser.add\_option("-m", "--metadata", action="store\_true",

help="exit immediately after printing an onMetaData tag")

parser.add\_option("-v", "--verbose", action="count",

default=0, dest="verbosity",

help="be more verbose, each -v increases verbosity")

options, args = parser.parse\_args(sys.argv)

if len(args) < 2:

parser.error("You have to provide at least one file path")

if options.strict:

tags.STRICT\_PARSING = True

if options.verbosity > 3:

options.verbosity = 3

level = ({0: logging.ERROR, 1: logging.WARNING,

2: logging.INFO, 3: logging.DEBUG}[options.verbosity])

logging.getLogger('flvlib').setLevel(level)

return options, args

def debug\_files():

options, args = process\_options()

clean\_run = True

for filename in args[1:]:

if not debug\_file(filename, options.quiet, options.metadata):

clean\_run = False

return clean\_run

def main():

try:

outcome = debug\_files()

except KeyboardInterrupt:

# give the right exit status, 128 + signal number

# signal.SIGINT = 2

sys.exit(128 + 2)

except EnvironmentError, (errno, strerror):

try:

print >>sys.stderr, strerror

except StandardError:

pass

sys.exit(2)

if outcome:

sys.exit(0)

else:

sys.exit(1)