How to run memory checks on your R package

Jisca Huisman

2023-01-13

Introduction

The problem

Your R package with compiled C or Fortran code got accepted on CRAN, but after a while you are notified about Valgrind and/or UBSAN/ASAN and/or LTO errors. You are not able to reproduce them, and so are not sure what to fix and/or whether your fix is working.

You should of course read the handbook (https://cran.r-project.org/doc/manuals/r-devel/R-exts.html#Checking-memory-access), but I've found it rather hard to understand the first 10 times I read it, as it assumes quite a lot of prior knowledge.

The solution

- A: download & install R-devel under linux [1] on your own computer, and configure it in such a way that it will incorporate/allow for these checks
- B: install Docker, and run a docker image with R that is configured to incorporate/allow for these checks [2]

The settings CRAN uses for its memory tests can be found at https://www.stats.ox.ac.uk/pub/bdr/memtests/README.txt . It describes making edits in the configure files config.site, ~/.R/Makevars , and .Renvironment (I think), but this did not seem to *do* anything in my weeks of tinkering [3]. The approach described below did result in successful reproduction of the errors reported by CRAN, on Ubuntu 22.04.

A: Install & configure R-devel

This is slightly adapted from: https://francoismichonneau.net/2014/03/how-to-do-ubsan-tests-r-package/

update /etc/apt/sources.list

```
sudo apt-get update
sudo apt-get upgrade
```

install/update needed software

```
sudo apt-get install valgrind subversion r-base-dev clang texlive-fonts-extra texlive-latex-extra tidy
```

install required R packages (add any that your pkg depends on) (this assumes you already have a copy of R installed, else do it a few steps later)

```
R -e install.packages('V8', 'qpdf', 'plyr', 'openxlsx', 'knitr', 'rmarkdown')
```

get latest R-devel (copy from url to folder ~/R-devel)

[1] Some of this may work under MacOS too.
[2] Supposedly works on all OS's
[3] Suggestions on how to make this work, or whether I misinterpreted that file, are welcome.

```
svn co https://svn.r-project.org/R/trunk ~/R-devel
```

basic configuration & install

```
cd R-devel
make clean
./configure --with-x=no --without-recommended-packages
make
sudo make install
```

You may need 'make distclean' for a proper clean up if `make` fails.

The `./configure` command is where the magic happens, see following section for some options.

And to check your package from the command line (assuming the .tar.gz is in the current directory):

```
R CMD check --as-cran mypkg_0.0.1.tar.gz
```

configure R

You can add options both before and after `./configure`.

The ones before are a subset of the options listed in /R-devel/config.site , which contains short descriptions and handy pointers for each item.

The available options after are listed when you run `./configure --help`.

For example, this is the command I used for UBSAN, which I copied and adapted from https://hub.docker.com/r/rocker/r-devel-san/dockerfile :

```
LIBnn=lib64 \
   CC="gcc -std=gnu99 -Wall -Wstrict-prototypes -fsanitize=undefined -fno-omit-frame-pointer" \
   CFLAGS="-g -O2 -mtune=native" \
   CXX="g++ -fsanitize=undefined,bounds-strict -fno-omit-frame-pointer" \
   CXXFLAGS="-g -O2 -Wall -pedantic -mtune=native" \
   CXXSTD=-std=gnu++98 \
   MAIN_LDFLAGS="-fsanitize=undefined -pthread" \
   FC="gfortran -fsanitize=undefined" \
   FCFLAGS="-g -Og -Wall -mtune=native" \
   FFLAGS="-g -O2 -mtune=native" \
   LTO='-flto' \
   ./configure --without-recommended-packages \
       --disable-openmp \
       --without-x \
       --without-blas \
       --without-lapack \
       --enable-lto=no \
```

Instead of gcc & g++, you can also use the compilers clang & clang++, if you have only C code. However if you have fortran code, it's better to use gcc (?), as gfortran is part of gcc and use the same sanitizers, whereas the sanitizers of clang and gfortran are completely different

LTO: Link Time Optimisation

Problems that are detected when LTO is turned on indicate a mismatch in the number, order, or data type of arguments passed between R and Fortran/C (via init.c).

- info: https://gcc.gnu.org/wiki/LinkTimeOptimization .
- R configure: `--enable-lto=yes`
- R configure options (might work without):

  - LIBnn="lib64" (depends on your platform, see notes in /R-devel/config.site)
  - AR="gcc-ar" (or "ar –plugin=/usr/lib/gcc/x86_64-linux-gnu/11/liblto_plugin.so")
  - NM="gcc-nm"
  - RANLIB="gcc-ranlib"
  - LTO="-flto"
- install pkg: `--configure-args=--enable-lto=check`
- check pkg: `--install-args=--configure-args=--enable-lto=check`

Valgrind

Checks for memory-related errors.

- https://kevinushey.github.io/blog/2015/04/05/debugging-with-valgrind/
- https://valgrind.org/docs/manual/quick-start.html#quick-start.intro .
- R configure: `--with-valgrind-instrumentation=2 --with-system-valgrind-headers`
- R configure options: add `-g -fno-omit-frame-pointer` to CFLAGS, FCFLAGS, CXXFLAGS
- check pkg: `--use-valgrind` (makes examples & tests run MUCH slower)
- test R script: `R -d "valgrind --tool=memcheck --leak-check=full" --vanilla < test.R`

UBSAN: undefined behaviour sanitizer

'Undefined behaviour' is e.g. array subscript out of bounds

- https://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html
- R configure: none
- R configure options:
  - add `-fsanitize=undefined` to CC, FC, CXX, and MAIN_LDFLAGS
  - add `-g -fno-omit-frame-pointer` to CFLAGS, FCFLAGS, CXXFLAGS
- check pkg: none
- test R script: none

There is a package that lets you check if you managed to set it up correctly, since when you see no errors, it either means your code is OK, or the sanitizer isn't working. In R,

```
install.packages("sanitizers")
sanitizers::stackAddressSanitize(42)
sanitizers::heapAddressSanitize(1)
```

If the functions give an error, the sanitizer works. Note that if you re-configure and re-make R, you need to re-install this package to check your new R installation.

ASAN: address sanitizer

- https://github.com/google/sanitizers/wiki/AddressSanitizer
- https://clang.llvm.org/docs/AddressSanitizer.html
- R configure: none
- R configure options: add `-fsanitize=address` to CC, CXX, and MAIN_LDFLAGS

NOTE: with `FC="gfortran -fsanitize=address"`, I did not manage to get R to compile succesfully. If you have a non-R version of your fortran code, you can check it by compiling with e.g.

```
gfortran -std=f95 -g -Og -fall-intrinsics -fsanitize=address,undefined -pedantic -Wall myprog.f90 -o myp
```

and running with e.g. (see https://github.com/google/sanitizers/wiki/AddressSanitizerFlags)

```
ASAN_OPTIONS=detect_leaks=1 ./myprog
ASAN_OPTIONS=detect_stack_use_after_return=true ./myprog
```

these flags can also be combined with R, e.g. `ASAN_OPTIONS=detect_leaks=1 R --vanilla < test.R`

B: Docker

https://knausb.github.io/2017/06/running-r-devel-ubsan-clang-in-docker/ https://rocker-project.org/images/base/r-devel.html http://dirk.eddelbuettel.com/code/sanitizers.html