Linaro Task Report
Porting software to Morello

Lorenzo Carletti

April 2021

1 Overview

Morello is an ARM project which focuses on new ways of designing CPU architecture that will make processors more robust in future, and able to deter some types of security breaches by using capabilities (perms) inside pointers. It uses a Capability Hardware Enhanced RISC Instructions (CHERI) architecture. Arm has developed a prototype architecture that adapts the hardware concepts of CHERI.

Linaro's task was to port tools originally developed for Linux to a CHERI Pure-Capabilities compatible Android environment, in order to gauge how difficult of a process that is, and to identify what possible problems could come up during that.

2 Setup

At the link HERE, one can find an easy-to-follow guide that prepares the basic building and running environment.

3 Choosing a porting target

Once the setup procedure has been completed, the programmer has to decide which software they want to port.
Android is a very bare bones version of Linux, so many features and functionalities aren't supported.
As an example, trying to port software that makes use of terminfo may prove difficult because of that.

With all that said, the chosen program was curl (https://curl.se/), a command-line tool for getting or sending data including files using URL syntax. curl supports HTTPS and performs SSL certificate verification by default when a secure protocol is specified such as HTTPS.

4 Porting software to Android

When making software run using CHERI Pure-Capabilities under Morello's Android, a first necessary step is making the code run under Android regularly.

If one wants to compile code for Android, they have to write a Soong build file (*.bp). This type of file contains a list of the source files and various flags which will be used during compilation.

If the chosen software has not been ported to Android before, then the person who wants to do the porting will need to manually create the list of source files and flags by looking at the Linux version's Makefile.
If the software has been already ported to Android, then it could either have a Soong build file (in which case it's not required to change it), or an Android makefile (*.mk).
Although using an Android makefile can work, it is still recommended to translate that file to the Soong build file format.
That is the case because it can be problematic if a project that uses an Android makefile needs to use a library compiled using a Soong build file.
Luckily, a somewhat automated tool named androidmk exists, and it can help translating *.mk files to the *.bp file format.

Once the software has its own Soong build file, it's time to check if it runs properly under Android.
This can either be done by adding the *.bp file to
"device/arm/morello/morello_fvp_nano.mk", recompiling Morello and running the FVP, or by following the "Android tests suites and workloads - build and run" guide available HERE.

If the software runs without issues, one can finally try to make it run using CHERI Pure-Capabilities.

curl already had an Android port which uses Soong build files, so getting it to run was not an issue.

5 Porting to CHERI Pure-Capabilities

In order to make the software use CHERI Pure-Capabilities, one must add to the Soong build file two very important lines: *compile_multilib: "c64"* and *static_executable: true.*
At the time of writing, Morello doesn't support dynamic linkage for CHERI Pure-Capabilities software, so it's important to compile the porting target statically.

After that, it's time to try to compile and run the software. The build system will likely report various errors and by fixing them the software's port can be achieved. This varies on a case-by-case basis. From here on out the report will focus on my experience trying to port curl.

6 Porting curl

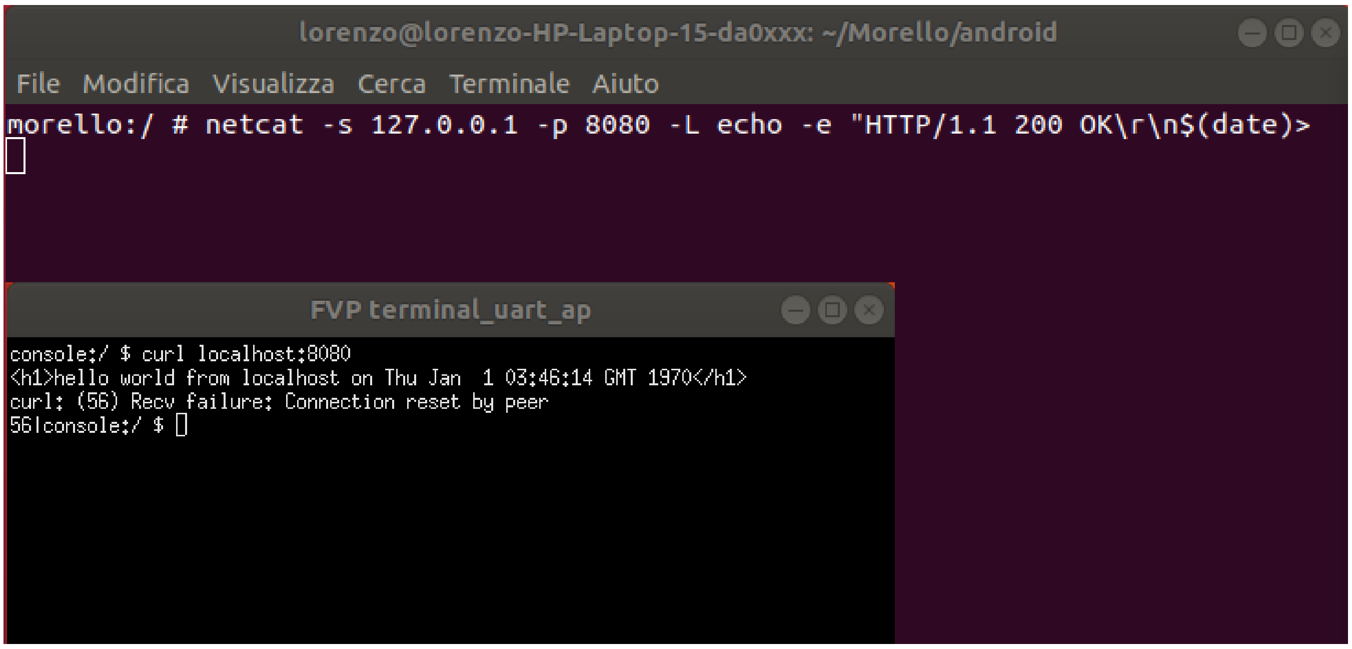
curl is a project that compiles both a tool and a library, which the tool uses. When porting something to CHERI Pure-Capabilities, it's important to realize that all the libraries used by the software must be ported as well.
This means that if there are issues within said libraries, one must fix those too.

While compiling, I was able to easily spot an issue with some of curl's code which wasn't really sure what the size of *long* was. Adding building rules for "libc64" fixed the problem.

Some other issues were brought up from the building system because of BoringSSL, Google's OpenSSL implementation. To be specific, it was trying to compile machine-specific ASM code which didn't work on Morello, and it also had some alignment issues within one struct. Fixing these was easy.

The biggest problem came after the compiling process succeded. The software was properly compiled, but upon trying to run it, it would instantly close and print to stderr "Bus Error".
Using strace, it was possible to see the program was being killed by a BUS_ADRALN SIGBUS, which meant there was an alignment issue.
Sadly, strace didn't offer any more help, and *printf* didn't work either, so the only way to understand what was causing the crash was to put an early

return inside the code, and to slowly advance through it by moving said *return*.
After a while, I was able to finally understand that the issue came from a memory access aligned to 8 inside a BoringSSL's allocated area. Changing that memory access so it's aligned to 16 fixed the problem, and curl finally started working.


lorenzo@lorenzo-HP-Laptop-15-da0xxx: ~/Morello/android
File Modifica Visualizza Cerca Terminale Aiuto
FVP terminal_uart_ap

```
morello:/ # netcat -s 127.0.0.1 -p 8080 -L echo -e "HTTP/1.1 200 OK\r\n$(date)>
```

```
console:/ $ curl localhost:8080
<h1>hello world from localhost on Thu Jan  1 03:46:14 GMT 1970</h1>
curl: (56) Recv failure: Connection reset by peer
56|console:/ $
```

7 Conclusions

The porting process could seem simple at a first glance, however one must be aware of the possible issues that may arise during it.
The most likely cause of problems seems to be alignment of data, followed by unspecified compiling rules for certain flags.

It was an interesting experience and trying to figure out how to work and fix stuff on this new platform was very stimulating.

Many thanks to Joakim Bech, Paolo Valente and Linaro for the opportunity.