TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Link Mechanism in XALT

XALT : LD Wrapper

PRESENTED BY:

Amit Ruhela

Texas Advanced Computing Center,

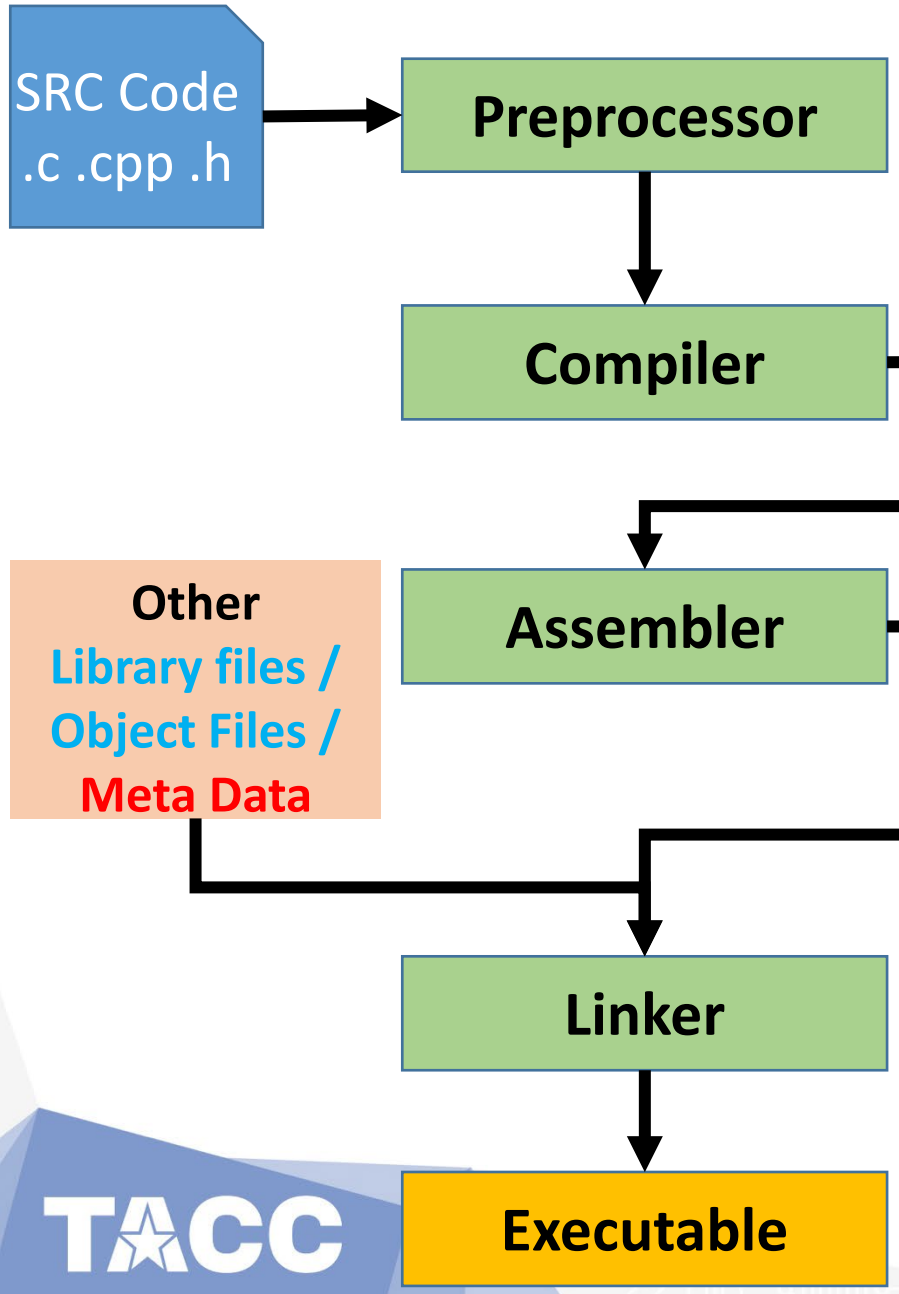
Austin Texas

User Group Meeting, August 18, 2022

Agenda

- Compilation Process
- XALT - General
- Watermarks
- Function Tracking
- Reading Watermarks
- XALT Linking Process
- Code Walkthrough - XALT Link Wrapper

Compilation Process



(Preprocessing) **$ cpp hello.c > hello.i or**
$ gcc -E hello.c > hello.i
Processes include files and macros e.g., #include, #define etc.

(Compilation) **$ gcc -S hello.i -o hello.s**

(Assembly) **$ as hello.s -o hello.o**

(Linking) **$ gcc hello.o -o hello**
GCC calls "ld" to link various object codes to an executable

(Execution) **$ ./hello** (Load and Execute)

XALT - Extended Automatic Library Tracking

A tool to allow a sites to track user executables and library usage on a cluster

- Tracks both compile-time and runtime information of applications to better manage our systems

Provides information about

- Compilers including the top-level compiler in the process tree
- Libraries
- External Subroutines from optional software provided by modules
- Build environments
- Job level information

Automatic and transparent collection of above information is performed by a wrapper shell routine that hijack system linker (ld) at link time.

XALT

- Requirements
  - Site installers make sure that XALT's *ld*, *ld.gold* and *x86_64-linux-gnu-ld* wrapper scripts should be found before the compilers or system's *ld* program

    prepend_path("COMPILER_PATH","/opt/apps/xalt/xalt/bin")
    prepend_path{"PATH","/opt/apps/xalt/xalt/bin",priority="100"}
    prepend_path("LD_PRELOAD","/opt/apps/xalt/xalt/lib64/libxalt_init.so") – For Runtime only

- Adds watermark to every executable it builds
  - Unload XALT module when building gcc or g++

Watermarks

Records extra information (also called markers) in compiled files, often called binaries or ELF objects

- ❑ Series of ELF notes in a special section.
- ❑ Notes are concatenated together when binaries are linked.
- ❑ ELF notes doesn't get stripped when debug information is removed.

Uses :

- Captures the system’s & user’s information, build timestamp, loaded modules, etc.
- Connects the Link Records to the Run Records
- Deter un-authorized resharing of sensitive information.

Read Watermarks

$ readelf --notes --wide <file>

```
staff.frontera(1004)$ readelf --notes --wide a.out

Displaying notes found in: .note.gnu.build-id
  Owner                Data size       Description
  GNU                  0x00000014      NT_GNU_BUILD_ID (unique build ID bitstring)        Build ID: 0b672ff33e1f5f12a4dda14899c5d671e
5cb85e0

Displaying notes found in: .note.ABI-tag
  Owner                Data size       Description
  GNU                  0x00000010      NT_GNU_ABI_TAG (ABI version tag)           OS: Linux, ABI: 2.6.32

Displaying notes found in: .note.xalt.info
  Owner                Data size       Description
  XALT                 0x00000465      Unknown note type: (0x746c6158)    description data: 02 58 41 4c 54 5f 4c 69 6e 6b 5f 49 6e 66
6f 00 3c 58 41 4c 54 5f 56 65 72 73 69 6f 6e 3e 25 25 32 2e 31 30 2e 33 34 25 25 00 3c 42 75 69 6c 64 5f 68 6f 73 74 3e 25 25 73 74 61
66 66 2e 66 72 6f 6e 74 65 72 61 2e 74 61 63 63 2e 75 74 65 78 61 73 2e 65 64 75 25 25 00 3c 42 75 69 6c 64 5f 53 79 73 68 6f 73 74 3e
25 25 66 72 6f 6e 74 65 72 61 25 25 00 3c 42 75 69 6c 64 5f 63 6f 6d 70 69 6c 65 72 3e 25 25 6d 70 69 69 63 63 28 69 63 63 29 25 25 00
3c 42 75 69 6c 64 5f 63 6f 6d 70 69 6c 65 72 50 61 74 68 3e 25 25 69 63 63 3a 2f 6f 70 74 2f 69 6e 74 65 6c 2f 63 6f 6d 70 69 6c 65 72
73 5f 61 6e 64 5f 6c 69 62 72 61 72 69 65 73 5f 32 30 32 30 2e 31 2e 32 31 37 2f 6c 69 6e 75 78 2f 62 69 6e 2f 69 6e 74 65 6c 36 34 2f
69 63 63 25 25 00 3c 42 75 69 6c 64 5f 4f 53 3e 25 25 4c 69 6e 75 78 5f 25 5f 25 5f 33 2e 31 30 2e 30 2d 31 31 36 30 2e 34 35 2e 31 2e
```

Read Watermarks

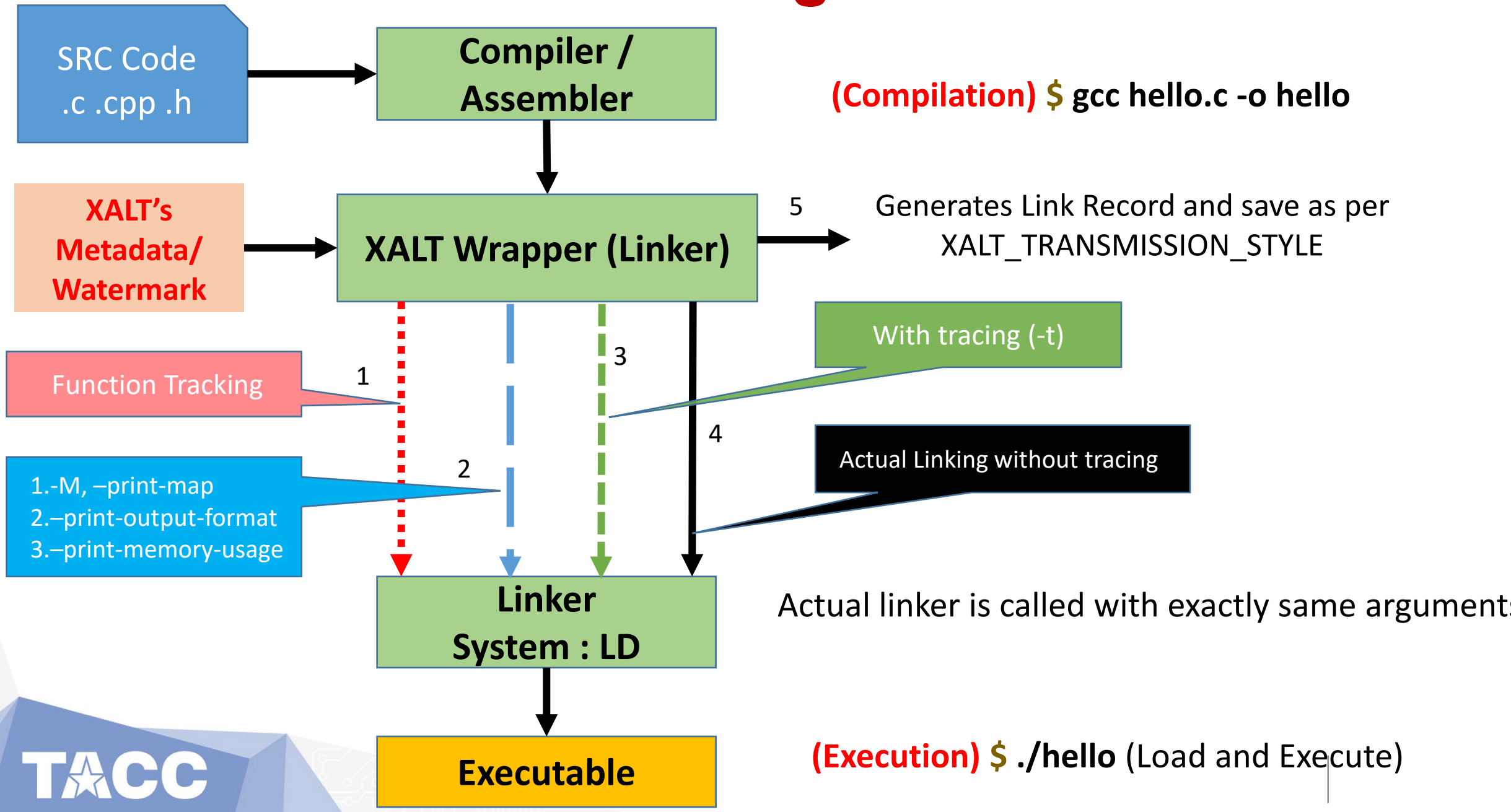
$ xalt_extract_record <file>

```
staff.frontera(1067)$ xalt_extract_record hello
****************************************
XALT Watermark: hello
****************************************
Build_CWD               /home1/05231/aruhela/examples
Build_Epoch             1659651079.8173
Build_LMFILES           /opt/apps/modulefiles/intel/19.1.1.lua:/opt/apps/intel19/modulefiles/impi/19.0.9.lua:/opt/apps/modulefiles/git
/2.24.1.lua:/opt/apps/modulefiles/autotools/1.2.lua:/opt/apps/intel19/impi19_0/modulefiles/python3/3.7.0.lua:/opt/apps/modulefiles/cmak
e/3.20.3.lua:/opt/apps/modulefiles/pmix/3.1.4.lua:/opt/apps/modulefiles/hwloc/1.11.12.lua:/opt/apps/modulefiles/xalt/2.10.34.lua:/opt/a
pps/modulefiles/TACC.lua
Build_LOADEDMODULES     intel/19.1.1:impi/19.0.9:git/2.24.1:autotools/1.2:python3/3.7.0:cmake/3.20.3:pmix/3.1.4:hwloc/1.11.12:xalt/2.1
0.34:TACC
Build_OS                Linux 3.10.0-1160.45.1.el7.x86_64
Build_Syshost           frontera
Build_UUID              b5344e03-77db-487a-840e-f69f92bab0ae
Build_User              aruhela
Build_Year              2022
Build_compiler          gcc
Build_compilerPath      /opt/apps/gcc/8.3.0/bin/gcc
Build_date              Thu Aug 04 17:11:19 2022
Build_host              staff.frontera.tacc.utexas.edu
XALT_Version            2.10.34
```

```
staff.frontera(1070)$ xalt_extract_record /bin/ls

No XALT Watermark
```

XALT Linking Process



XALT – Function Tracking

Goal : What external functions and subroutines are used by an application?

- Function tracking is performed at Link Time and not Run Time
- Done by dual linking process
  1. First linking done by system linker (ld)
  2. Second linking done by Xalt wrapper that does fail linking by omitting object files and libraries that are found in the reverse map (rmap)
     - Rmap is JSON-formatted file that captures association between libraries, their path and module information, and names of functions/subroutines in them.

LD Wrapper Workflow

1. Precursor
   a) unset LD_PRELOAD
   b) XALT initializations
   c) Find Executable and LD
   d) Options Parsing
2. XALT Setup
   a) Helper scripts identification
   b) Extract Hostname
   c) Generate Universal Unique ID (UUID) (Its not a user-id).
   d) Create a work directory in /tmp for XALT intermediaries
3. Watermark
   a) Get Compiler and Linker Information
   b) Generate Watermark
4. Linking
   a) Link first time if function tracking is requested by the site
   b) Link second-time if –print-map is specified by the user
   c) Link with tracing enabled (Print the names of the input files as ld processes them)
   d) Link last time if tracing fails
5. Generate Link Record
   a) Get realpath of executable
   b) Calculate checksum (SHA1)
   c) Generate the link record and save as per XALT_TRANSMISSION_STYLE (file/syslog/database)

Code Walkthrough

Linker Wrapper :

https://github.com/xalt/xalt/blob/master/sh_src/ld.in

e.g., XALT_TRACING=link mpicc hello.c -o hello

Watermark :

https://github.com/xalt/xalt/blob/master/src/linker/xalt_generate_watermark.C

Next Meeting

. Sept 15th at 10:00 AM CST (15:00 UTC)

Thanks for Listening

Links

1. XALT Documentation : https://xalt.readthedocs.io/en/latest/index.html#
2. XALT LD: https://github.com/xalt/xalt/blob/master/sh_src/ld.in
3. Bash https://pubs.opengroup.org/onlinepubs/9699919799/utilities/V3_chap02.html#tag_18_06_02
4. Shared vs Static Libraries : https://www.linuxtopia.org/online_books/an_introduction_to_gcc/gccintro_25.html