



# **Tutorial: Detecting Memory Vulnerabilities in the Components of System Code using PROMPT**

Tuba Yavuz and Ken (Yihang) Bai  
[tuba@ece.ufl.edu](mailto:tuba@ece.ufl.edu), [baiyihang@ufl.edu](mailto:baiyihang@ufl.edu)

University of Florida



#IEEESecDev



<https://secdev.ieee.org/2020>

# Tutorial Outline

- Part 0: Important information about the tool
- Part I: Background on Symbolic Execution and KLEE
- Part II: PROMPT for API Model Guided Symbolic Execution
- Part III: Real-World Case Studies
- Acknowledgements
- Q & A

# Important information

- PROMPT github page:
  - <https://github.com/sysrel/PROMPT>
  - Manual:  
[https://github.com/sysrel/PROMPT/blob/master/docs/PROMPT\\_Manual.txt](https://github.com/sysrel/PROMPT/blob/master/docs/PROMPT_Manual.txt)
  - PROSE template:
    - [https://github.com/sysrel/PROMPT/blob/master/docs/PROSE\\_template.txt](https://github.com/sysrel/PROMPT/blob/master/docs/PROSE_template.txt)
  - Tutorial examples:
    - [https://github.com/sysrel/PROMPT/tree/master/PROMPT\\_examples](https://github.com/sysrel/PROMPT/tree/master/PROMPT_examples)
  - Device driver and BlueZ case studies
    - [https://github.com/sysrel/PROMPT/tree/master/JASE\\_benchmarks](https://github.com/sysrel/PROMPT/tree/master/JASE_benchmarks)
  - Heartbleed
    - <https://github.com/sysrel/PROMPT/tree/master/caseStudies/openssl/heartbleed>

# Important information

- PROMPT virtual machine image for VirtualBox (Ubuntu 16.04, 64 bit)
  - [https://drive.google.com/file/d/1bjLJ\\_HVCWBuQQC\\_6BNER3oiwEkXJGIHR/view?usp=sharing](https://drive.google.com/file/d/1bjLJ_HVCWBuQQC_6BNER3oiwEkXJGIHR/view?usp=sharing)
  - Username: prompt passwd: prompt
  - VirtualBox can be downloaded from <https://www.virtualbox.org/wiki/Downloads>
- PROMPT source tree can be found at /home/prompt/PROMPT
- PROMPT has been built under /home/prompt/prompt\_build\_dir
- Set the PROMPT environment variable to the path of the executable
  - \$ export PROMPT=/home/prompt/prompt\_build\_dir/bin/klee
- Each tutorial example and real-world case study directory has a run.sh script
  - Follow the instructions on the slides to execute PROMPT using run.sh

# PART I: Background on Symbolic Execution and KLEE

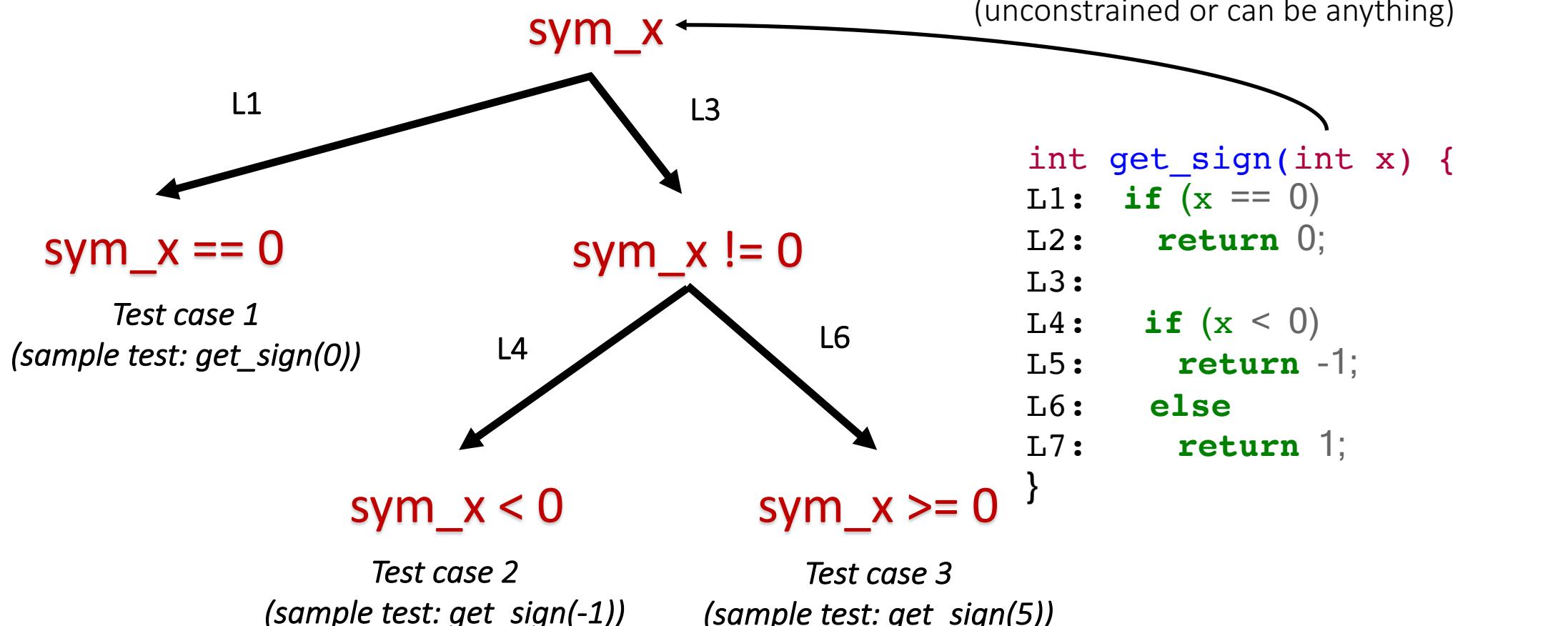
# Outline for PART I

- What is symbolic execution?
- A simple example
- The KLEE symbolic execution engine
- Challenges of symbolic execution

# What is Symbolic Execution?

- Symbolic execution is a program analysis technique that has been introduced as a testing technique by James King:
  - James C. King. Symbolic Execution and Program Testing. Communications of the ACM, 19:7, 1976.
- Symbolic execution interprets a program by propagating symbolic inputs according to the program logic
- There are two major approaches to symbolic execution
  - Concolic execution
  - Execution Generation Based Testing (EGT)
    - KLEE follows the EGT approach

# A simple example



100% Instruction/Code  
Coverage

100% Branch  
Coverage

# KLEE Symbolic Execution Engine

- Very mature tool
  - Cristian Cadar, Daniel Dunbar, Dawson Engler. KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs. OSDI 2008.
  - Authors received ACM SIGOPS Hall of Fame Award due to impact of the paper
- Open source: <https://github.com/klee/>
  - Documentation: <http://klee.github.io/>
- Based on the LLVM IR
  - Code needs to be compiled with the clang compiler to generate the LLVM bitcode
- Follows the EGT approach
- Works on closed programs
  - Needs a test driver for proper initialization of data

# A test driver for KLEE

```
#include "klee/klee.h"

int main() {
    int a;
    klee_make_symbolic(&a, sizeof(a), "a");
    return get_sign(a);
}
```

```
int get_sign(int x) {
L1: if (x == 0)
L2:     return 0;
L3:
L4: if (x < 0)
L5:     return -1;
L6: else
L7:     return 1;
}
```

```
$ clang -I $KLEE_INCLUDE_PATH -emit-llvm -c -g get_sign.c
```

```
$ KLEE --write-kqueries get_sign.bc
```

```
$ ls klee-last/test*
```

```
klee-last/test000001.kquery klee-last/test000002.ktest
```

```
klee-last/test000001.ktest klee-last/test000003.kquery
```

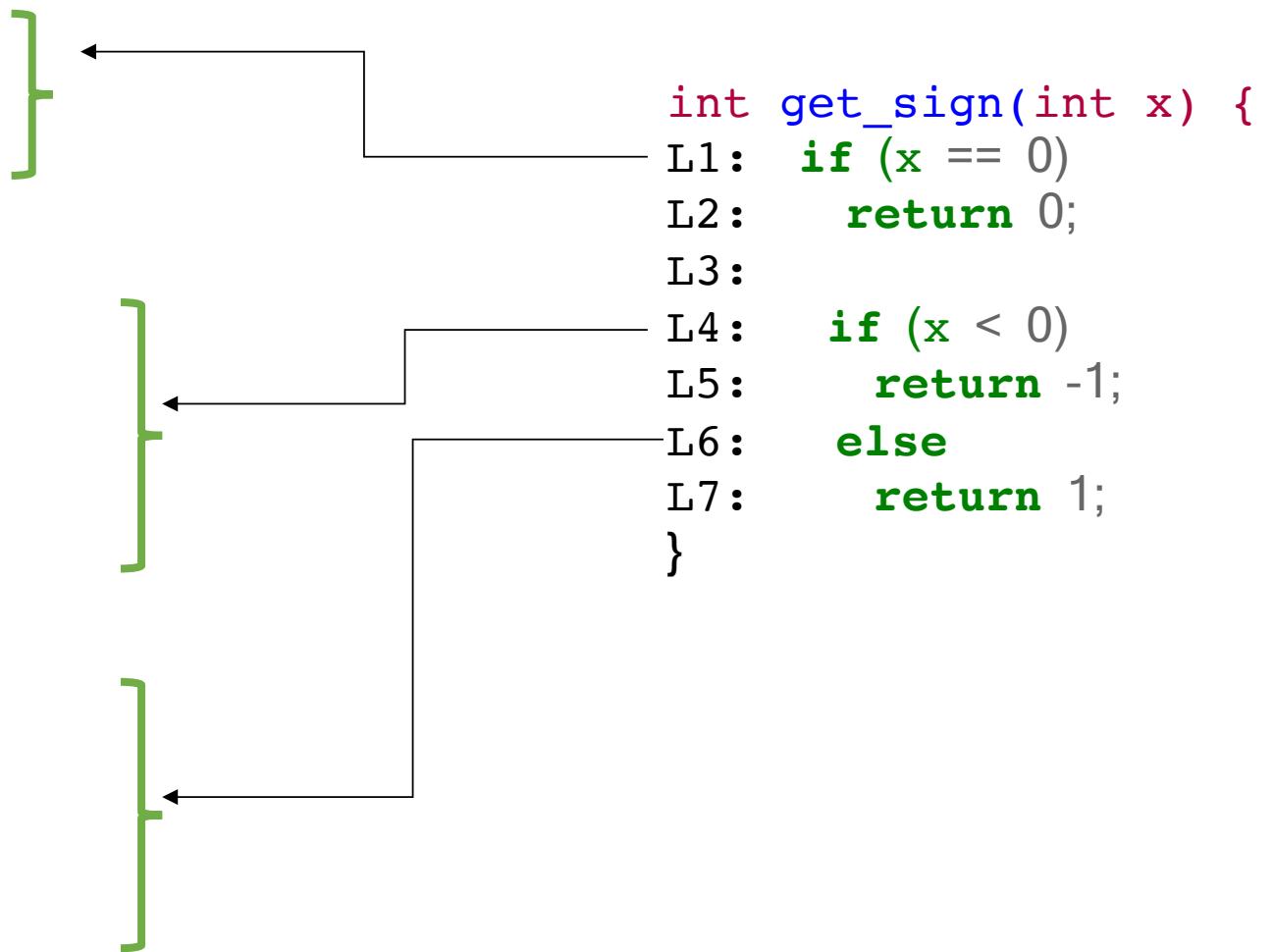
```
klee-last/test000002.kquery klee-last/test000003.ktest
```

# Analyzing get\_sign with KLEE

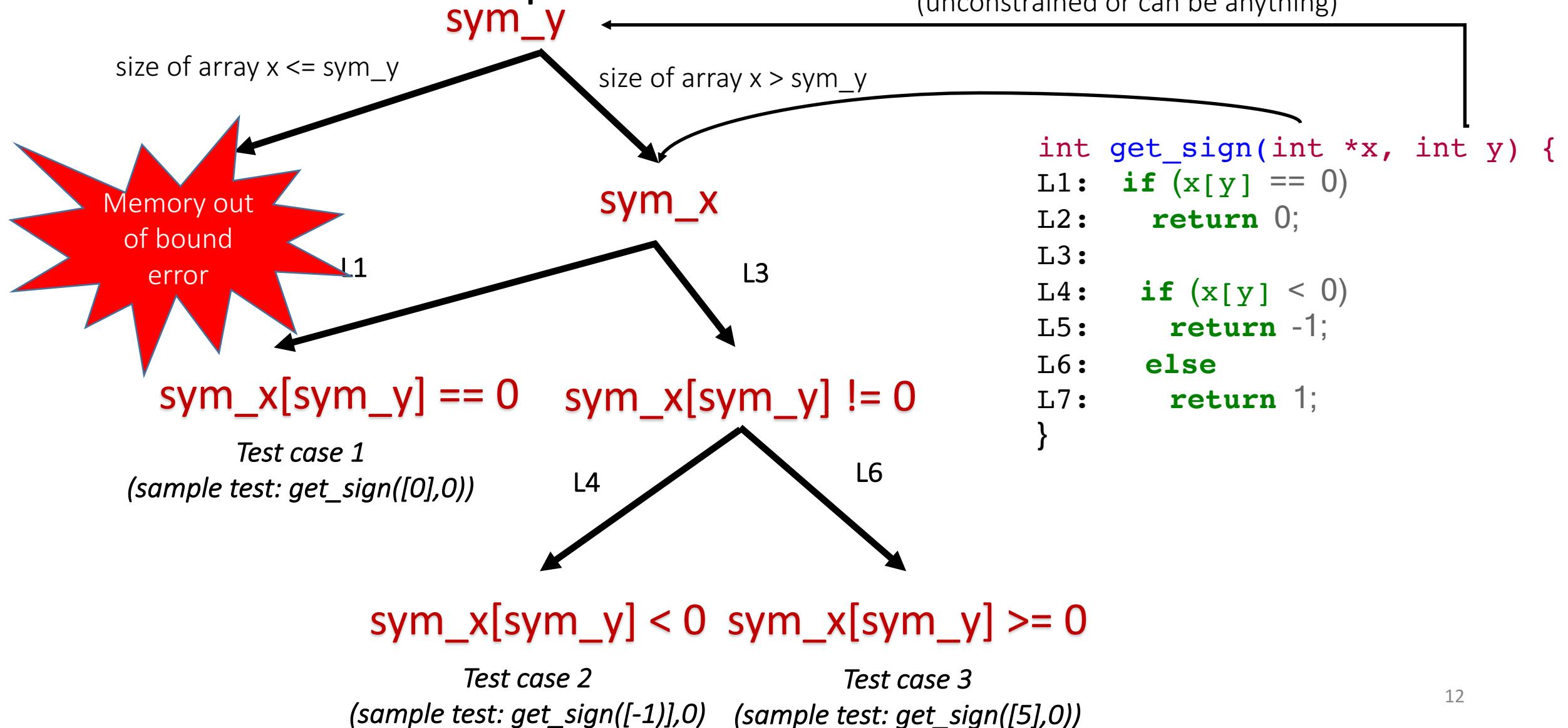
```
$ more klee-last/test000001.kquery
array a[4] : w32 -> w8 = symbolic
(query [(Eq 0
          (ReadLSB w32 0 a))]
      false)
```

```
$ more klee-last/test000003.kquery
array a[4] : w32 -> w8 = symbolic
(query [(Eq false
          (Eq 0
              N0:(ReadLSB w32 0 a)))
          (Slt N0 0))]
      false)
```

```
$ more klee-last/test000002.kquery
array a[4] : w32 -> w8 = symbolic
(query [(Eq false
          (Eq 0
              N0:(ReadLSB w32 0 a)))
          (Eq false (Slt N0 0)))]
      false)
```



# Another example



# Memory Bugs

- Thanks to the precise memory model used in its dynamic symbolic execution approach, KLEE can detect a variety of memory vulnerabilities/bugs
- NULL pointer dereferencing
- Memory out of bounds reads/writes
- Use-after-free
- Double-free
- Memory leaks

# Wrap-up for PART I

- Symbolic execution has emerged as a white-box automated testing technique by quickly became an important technique for
  - Detecting memory and other types of vulnerabilities
  - Reverse engineering and model extraction
- One of the biggest challenges in symbolic execution is the path explosion problem
  - Prevents finding deep bugs
- Another challenge is environment initialization
  - Difficult to perform for complex systems

# PART II: PROMPT for API Model Guided Symbolic Execution

# Outline for Part II

- Motivation
- PROMPT Overview
- API Modeling with PROSE
- Hands-on Activity: Tutorial Examples

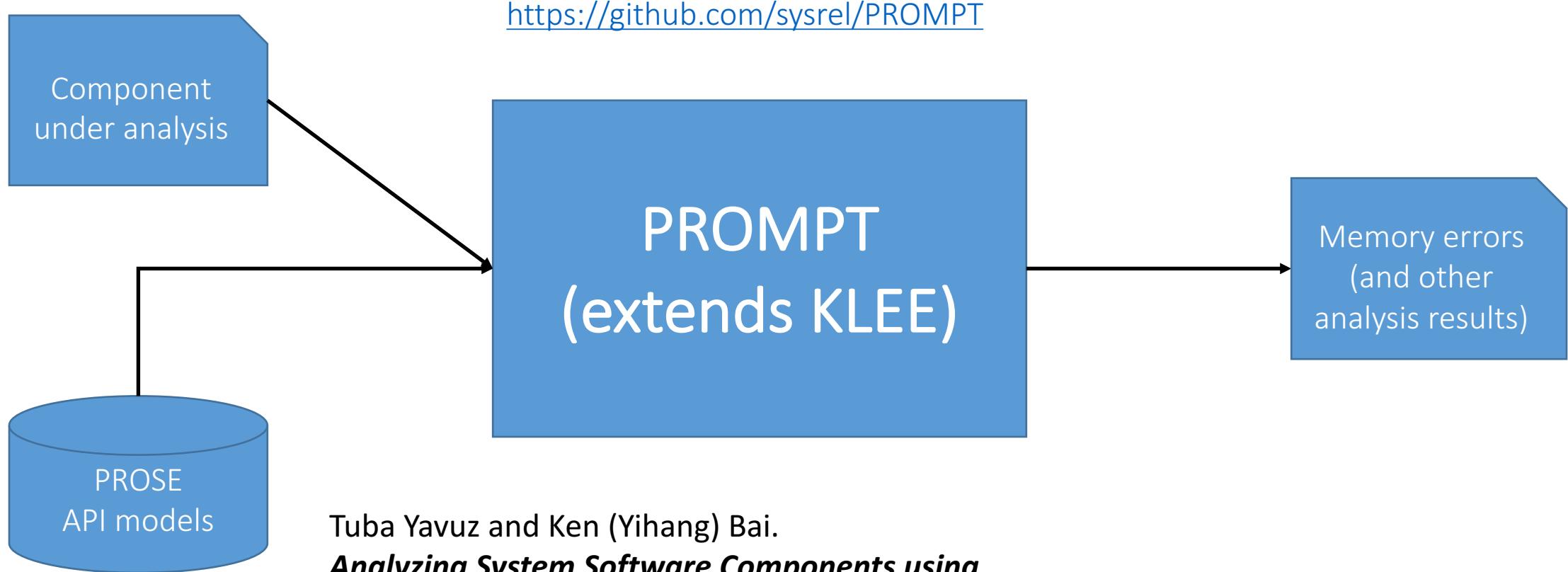
# Motivation

- The modeling of the Application Programming Interface (API) of complex system code is key to the scalable and precise analysis
- Techniques that automatically learn the behavior of APIs are not mature yet
  - Automated synthesis: the state space is often huge, guided by some syntax and/or input/output samples
  - Machine learning: still a big semantic gap between program representations and those used by the learning algorithms

# Motivation

- Under-constrained symbolic execution as implemented in UC-KLEE can support component-level analysis
  - David A. Ramos and Dawson Engler. Under-Constrained Symbolic Execution: Correctness Checking for Real Code. USENIX Security 2015.
  - UC-KLEE is not open-source
  - The goal of modeling is to filter out false positives
- **PROMPT approach for component-level analysis for system code**
  - API modeling and API model guided symbolic execution
  - Better control for scalability and precision

# Solution: PROMPT & PROSE



<https://github.com/sysrel/PROMPT>

Tuba Yavuz and Ken (Yihang) Bai.  
***Analyzing System Software Components using  
API Model Guided Symbolic Execution.***

To appear in Automated Software Engineering (27:6).  
DOI: 10.1007/s10515-020-00276-5

# Analyzing get\_sign with PROMPT

*Test driver*

```
#include "klee/klee.h"

int main() {
    int a;
    klee_make_symbolic(&a,
                       sizeof(a), "a");
    return get_sign(a);
}
```

*The model (model.txt)*

```
global settings:  
data models:  
function models:  
lifecycle model:  
    entry-point get_sign
```

*component under analysis*

```
int get_sign(int x) {
L1: if (x == 0)
L2:     return 0;
L3:
L4: if (x < 0)
L5:     return -1;
L6: else
L7:     return 1;
}
```

```
$ PROMPT –prose-api-model=model.txt –write-kqueries get_sign.bc
```

```
$ ls klee-last/test*
```

```
klee-last/test000001.kquery klee-last/test000002.ktest
```

```
klee-last/test000001.ktest klee-last/test000003.kquery
```

```
klee-last/test000002.kquery klee-last/test000003.ktest
```

# Analyzing get\_sign with PROMPT

```
$ more klee-last/test000001.kquery
```

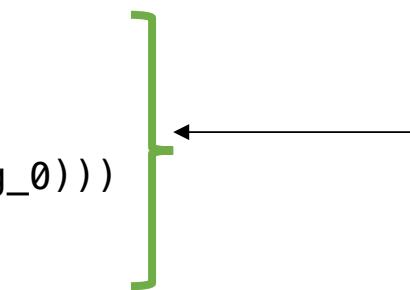
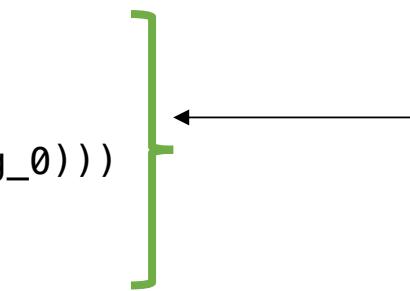
```
array get_sign_0_arg_0[4] : w32 -> w8 = symbolic  
(query [(Eq 0  
         (ReadLSB w32 0 get_sign_0_arg_0))]  
      false)
```

```
$ more klee-last/test000003.kquery
```

```
array get_sign_0_arg_0[4] : w32 -> w8 = symbolic  
(query [(Eq false  
         (Eq 0  
             N0:(ReadLSB w32 0 get_sign_0_arg_0)))  
         (Slt N0 0))]  
      false)
```

```
$ more klee-last/test000002.kquery
```

```
array get_sign_0_arg_0[4] : w32 -> w8 = symbolic  
(query [(Eq false  
         (Eq 0  
             N0:(ReadLSB w32 0 get_sign_0_arg_0)))  
         (Eq false (Slt N0 0))]  
      false)
```



*component under analysis*

```
int get_sign(int x) {  
L1: if (x == 0)  
    return 0;  
L3:  
L4: if (x < 0)  
    return -1;  
L6: else  
L7:     return 1;  
}
```

# PROMPT: API Model Guided Symbolic Execution

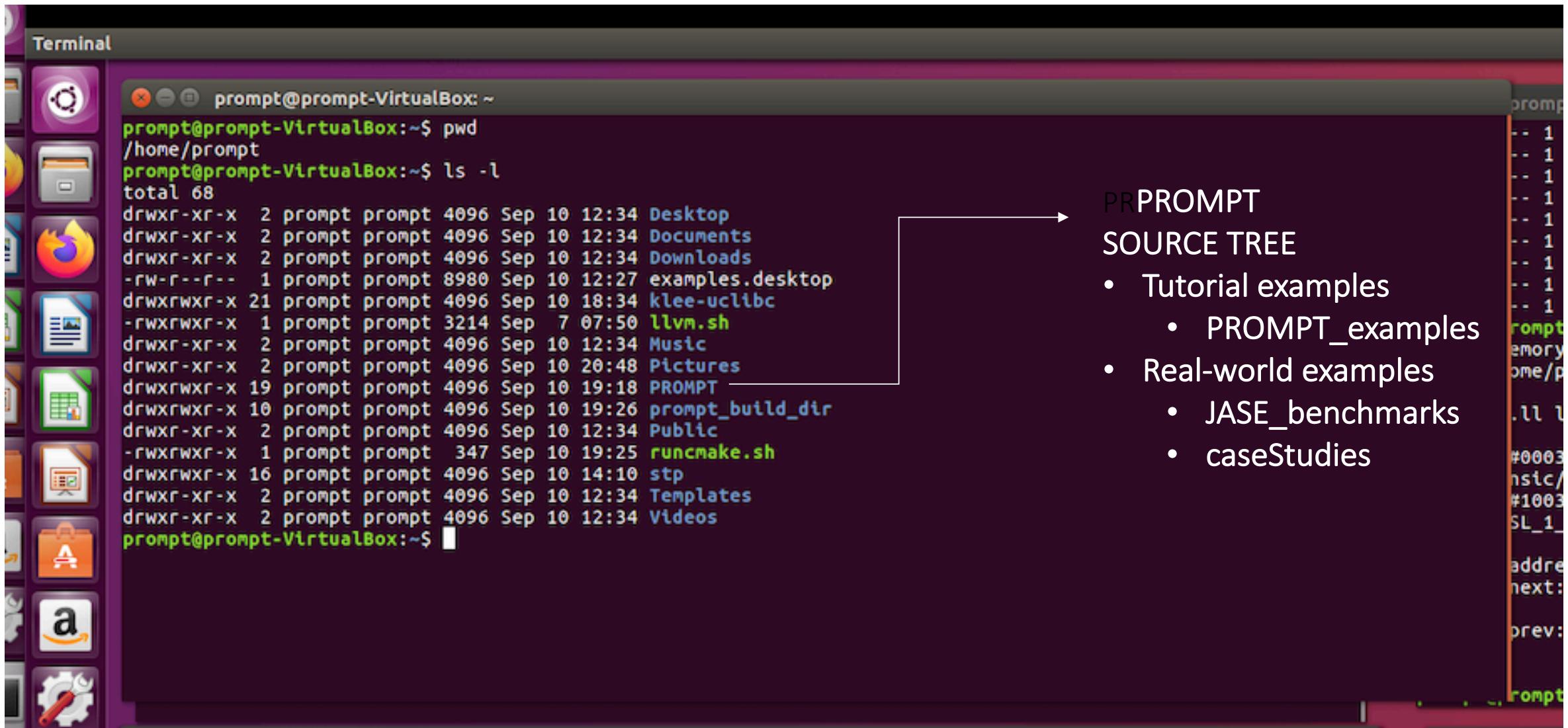
- Performs under-constrained symbolic execution
  - No need for a test driver
- Performs lazily initialization according to the specified API model
- Extends the KLEE symbolic execution engine to incorporate API modeling rules during handling of various instructions
  - Memory access instructions
  - Function calls
  - Return instructions
- Has been applied to real-world system code
  - Linux device drivers, cryptographic libraries, BlueZ

# API Modeling with PROSE

- The manual for PROSE language can be found at
  - [https://github.com/sysrel/PROMPT/blob/master/docs/PROMPT\\_Manual.txt](https://github.com/sysrel/PROMPT/blob/master/docs/PROMPT_Manual.txt)
- A template PROSE model can be found at
  - [https://github.com/sysrel/PROMPT/blob/master/docs/PROSE\\_template.txt](https://github.com/sysrel/PROMPT/blob/master/docs/PROSE_template.txt)
- Four types of specifications
  - Global settings
  - Data modeling
  - Function modeling
  - Life-cycle rules
- Tutorial examples
  - [https://github.com/sysrel/PROMPT/tree/master/PROMPT\\_examples](https://github.com/sysrel/PROMPT/tree/master/PROMPT_examples)

# Hands-on Activity: Tutorial Examples

# Using the virtual machine image

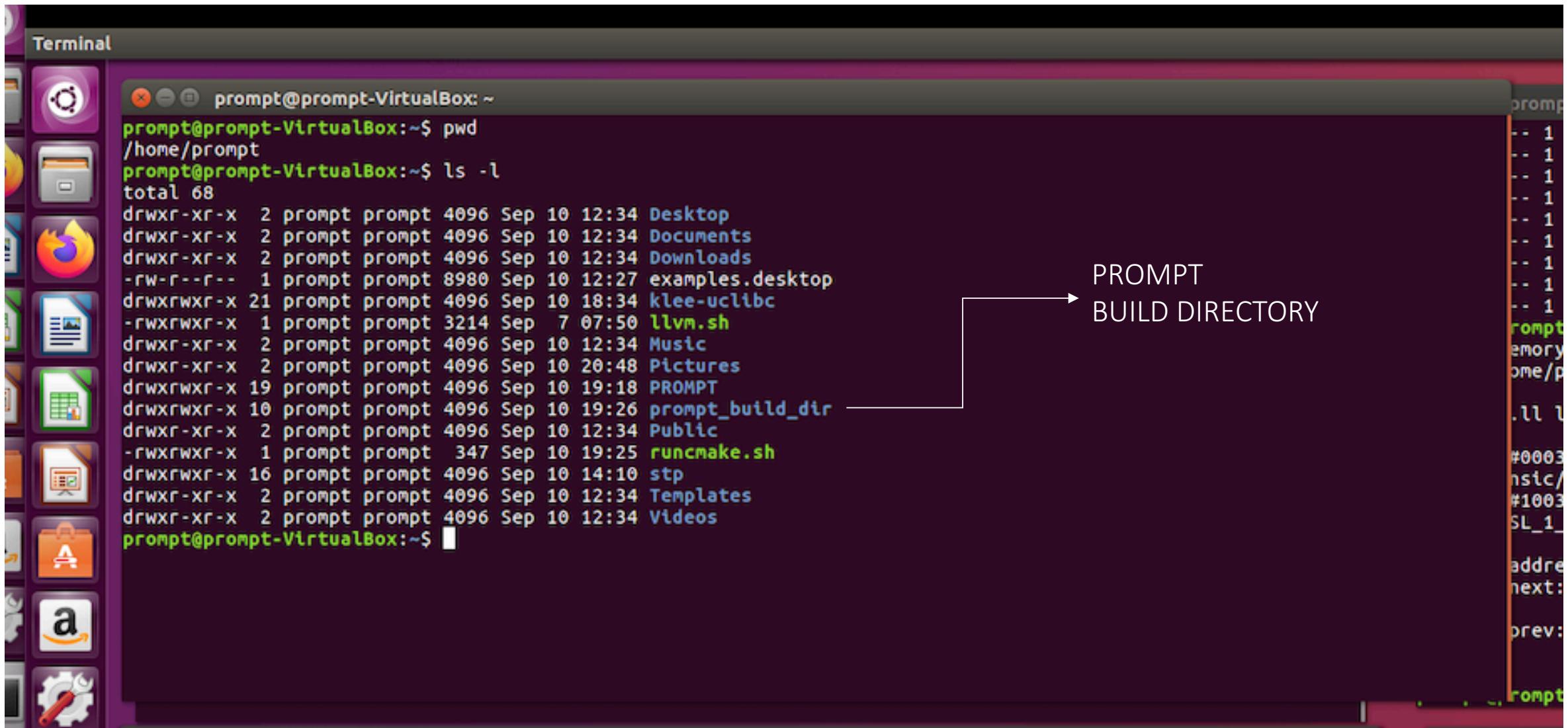


```
Terminal
prompt@prompt-VirtualBox:~$ pwd
/home/prompt
prompt@prompt-VirtualBox:~$ ls -l
total 68
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Desktop
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Documents
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Downloads
-rw-r--r--  1 prompt  prompt  8980 Sep 10 12:27 examples.desktop
drwxrwxr-x 21 prompt  prompt  4096 Sep 10 18:34 klee-uclibc
-rwxrwxr-x  1 prompt  prompt  3214 Sep  7 07:50 llvm.sh
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Music
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 20:48 Pictures
drwxrwxr-x 19 prompt  prompt  4096 Sep 10 19:18 PROMPT
drwxrwxr-x 10 prompt  prompt  4096 Sep 10 19:26 prompt_build_dir
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Public
-rwxrwxr-x  1 prompt  prompt   347 Sep 10 19:25 runcmake.sh
drwxrwxr-x 16 prompt  prompt  4096 Sep 10 14:10 stp
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Templates
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Videos
prompt@prompt-VirtualBox:~$
```

PRPROMPT  
SOURCE TREE

- Tutorial examples
  - PROMPT\_examples
- Real-world examples
  - JASE\_benchmarks
  - caseStudies

# Using the virtual machine image

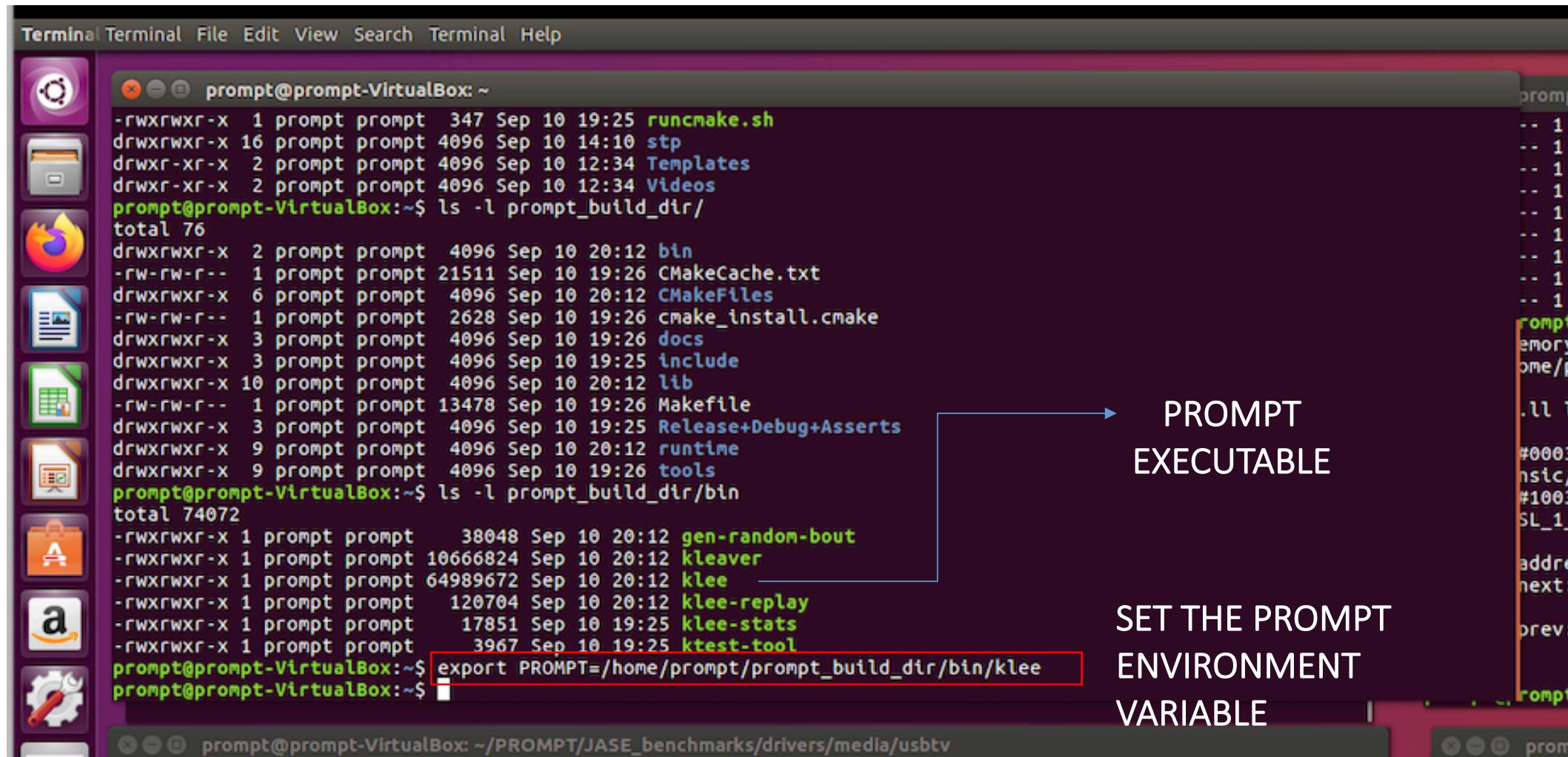


Terminal

```
prompt@prompt-VirtualBox:~$ pwd
/home/prompt
prompt@prompt-VirtualBox:~$ ls -l
total 68
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Desktop
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Documents
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Downloads
-rw-r--r--  1 prompt  prompt  8980 Sep 10 12:27 examples.desktop
drwxrwxr-x 21 prompt  prompt  4096 Sep 10 18:34 klee-uclibc
-rwxrwxr-x  1 prompt  prompt  3214 Sep  7 07:50 llvm.sh
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Music
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 20:48 Pictures
drwxrwxr-x 19 prompt  prompt  4096 Sep 10 19:18 PROMPT
drwxrwxr-x 10 prompt  prompt  4096 Sep 10 19:26 prompt_build_dir
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Public
-rwxrwxr-x  1 prompt  prompt   347 Sep 10 19:25 runcmake.sh
drwxrwxr-x 16 prompt  prompt  4096 Sep 10 14:10 stp
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Templates
drwxr-xr-x  2 prompt  prompt  4096 Sep 10 12:34 Videos
prompt@prompt-VirtualBox:~$
```

PROMPT  
BUILD DIRECTORY

# Getting ready for running the examples



The screenshot shows a terminal window with a purple title bar containing the text "Terminal". The window displays a file listing and an environment variable export command.

```
prompt@prompt-VirtualBox:~$ ls -l prompt_build_dir/
total 76
drwxrwxr-x  2 prompt  prompt  4096 Sep 10 20:12 bin
-rw-rw-r--  1 prompt  prompt 21511 Sep 10 19:26 CMakeCache.txt
drwxrwxr-x  6 prompt  prompt  4096 Sep 10 20:12 CMakeFiles
-rw-rw-r--  1 prompt  prompt 2628  Sep 10 19:26 cmake_install.cmake
drwxrwxr-x  3 prompt  prompt  4096 Sep 10 19:26 docs
drwxrwxr-x  3 prompt  prompt  4096 Sep 10 19:25 include
drwxrwxr-x 10 prompt  prompt  4096 Sep 10 20:12 lib
-rw-rw-r--  1 prompt  prompt 13478 Sep 10 19:26 Makefile
drwxrwxr-x  3 prompt  prompt  4096 Sep 10 19:25 Release+Debug+Asserts
drwxrwxr-x  9 prompt  prompt  4096 Sep 10 20:12 runtime
drwxrwxr-x  9 prompt  prompt  4096 Sep 10 19:26 tools
prompt@prompt-VirtualBox:~$ ls -l prompt_build_dir/bin
total 74072
-rwxrwxr-x  1 prompt  prompt   38048 Sep 10 20:12 gen-random-bout
-rwxrwxr-x  1 prompt  prompt 10666824 Sep 10 20:12 kleaver
-rwxrwxr-x  1 prompt  prompt 64989672 Sep 10 20:12 klee
-rwxrwxr-x  1 prompt  prompt  120704 Sep 10 20:12 klee-replay
-rwxrwxr-x  1 prompt  prompt  17851 Sep 10 19:25 klee-stats
-rwxrwxr-x  1 prompt  prompt   3967 Sep 10 19:25 ktest-tool
prompt@prompt-VirtualBox:~$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee
prompt@prompt-VirtualBox:~$
```

A red box highlights the command `export PROMPT=/home/prompt/prompt_build_dir/bin/klee`. A callout arrow points from this command to the text "PROMPT EXECUTABLE". Another callout arrow points from the same command to the text "SET THE PROMPT ENVIRONMENT VARIABLE".

At the bottom of the terminal window, the path `~/PROMPT/JASE_benchmarks/drivers/media/usbtv` is visible.

# Format for our examples

## PROSE API MODEL

```
global settings: ...
data models: ...
function models: ...
lifecycle model:
    entry-point ...
```

## COMPONENT UNDER ANALYSIS

SOME C CODE

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee
$ cd PROMPT_examples
$ cd PATH_TO_THE_EXAMPLE
$ ./run.sh foo.bc 2>&1 | tee o.txt
$ ls -l klee-last/
```

## PROMPT RESULTS:

# of paths: ?  
What to expect (error/no error) on certain line numbers  
All test cases will be written to some test\* file under the klee-last directory  
Those that end with the .err suffix are the error paths

# Modeling with PROSE

- Global settings
- Data modeling
- Function modeling
- Lifecycle modeling

# Specifying the Size of a Pointer Argument

## PROSE API MODEL

```
global settings:  
    array size 5;  
data models:  
function models:  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
void foo(int *a) {  
L1:    a[4] = 11;  
L2:    a[5] = 10;  
}
```

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd global_settings/arraySize/  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

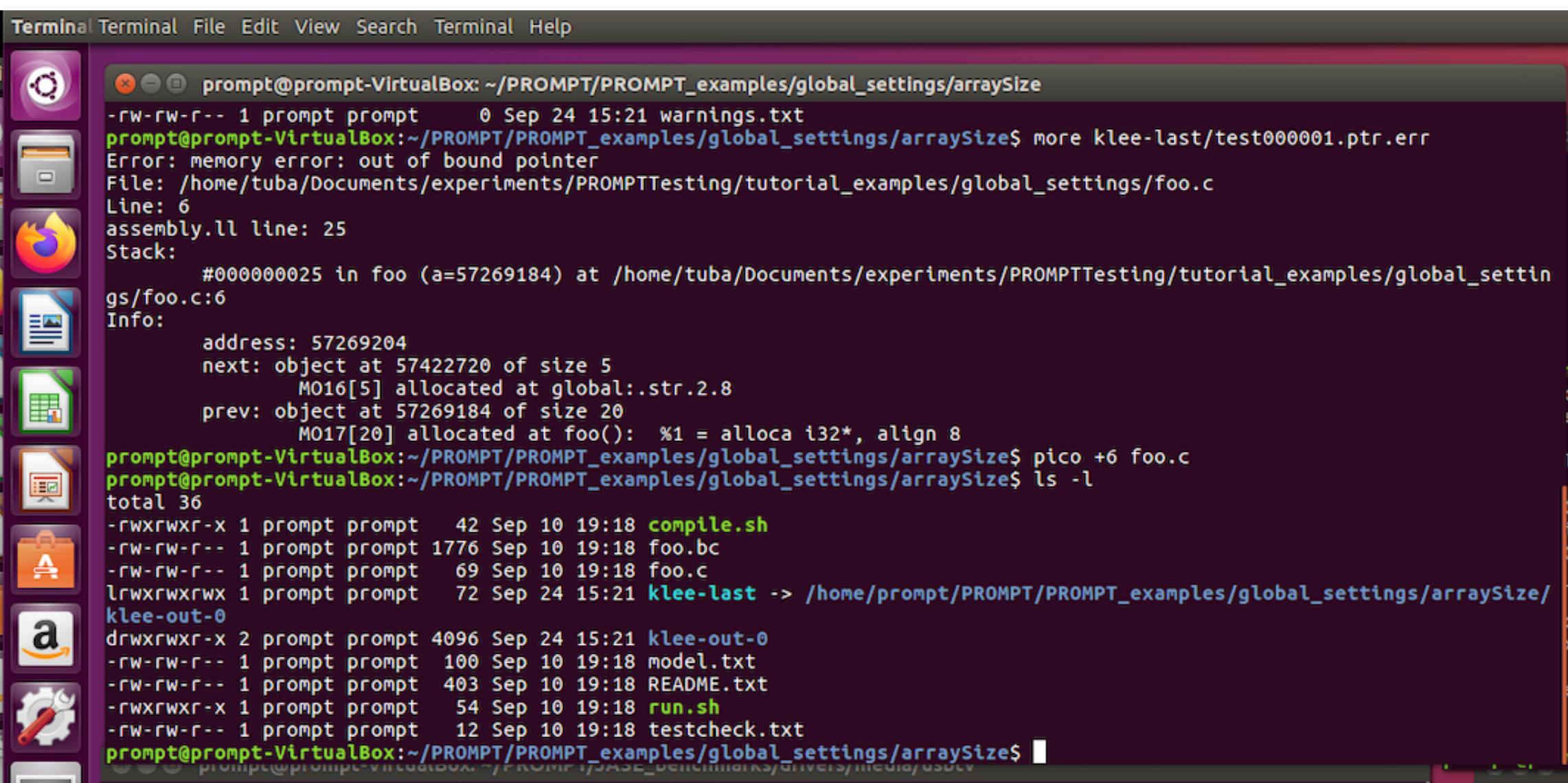
## PROMPT RESULTS:

```
# of paths: 1  
No memory error at line L1  
Memory error at line L2
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize$ pwd
/home/prompt/PROMPT/PROMPT_examples/global_settings/arraySize
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize$ ./run.sh foo.bc 2>/dev/null 1>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize$ ls -l klee-last/
total 128
-rw-rw-r-- 1 prompt prompt 76722 Sep 24 15:21 assembly.ll
-rw-rw-r-- 1 prompt prompt    636 Sep 24 15:21 info
-rw-rw-r-- 1 prompt prompt    222 Sep 24 15:21 messages.txt
-rw-rw-r-- 1 prompt prompt 26642 Sep 24 15:21 run.istats
-rw-rw-r-- 1 prompt prompt    567 Sep 24 15:21 run.stats
-rw-rw-r-- 1 prompt prompt     17 Sep 24 15:21 test000001.kquery
-rw-rw-r-- 1 prompt prompt     74 Sep 24 15:21 test000001.ktest
-rw-rw-r-- 1 prompt prompt   492 Sep 24 15:21 test000001.ptr.err
-rw-rw-r-- 1 prompt prompt      0 Sep 24 15:21 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize$ more klee-last/test000001.ptr.err
Error: memory error: out of bound pointer
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/foo.c
Line: 6
assembly.ll line: 25
Stack:
#000000025 in foo (a=57269184) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/foo.c:6
Info:
address: 57269204
next: object at 57422720 of size 5
M016[5] allocated at global:.str.2.8
prev: object at 57269184 of size 20
M017[20] allocated at foo(): %1 = alloca i32*, align 8
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize$
```

# Let's try it on the VM!



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and contains the following text:

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/global_settings/arraySize
-rw-rw-r-- 1 prompt prompt 0 Sep 24 15:21 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize$ more klee-last/test000001.ptr.err
Error: memory error: out of bound pointer
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/foo.c
Line: 6
assembly.ll line: 25
Stack:
#000000025 in foo (a=57269184) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/foo.c:6
Info:
address: 57269204
next: object at 57422720 of size 5
M016[5] allocated at global::str.2.8
prev: object at 57269184 of size 20
M017[20] allocated at foo(): %1 = alloca i32*, align 8
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize$ pico +6 foo.c
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize$ ls -l
total 36
-rwxrwxr-x 1 prompt prompt 42 Sep 10 19:18 compile.sh
-rw-rw-r-- 1 prompt prompt 1776 Sep 10 19:18 foo.bc
-rw-rw-r-- 1 prompt prompt 69 Sep 10 19:18 foo.c
lrwxrwxrwx 1 prompt prompt 72 Sep 24 15:21 klee-last -> /home/prompt/PROMPT/PROMPT_examples/global_settings/arraySize/
klee-out-0
drwxrwxr-x 2 prompt prompt 4096 Sep 24 15:21 klee-out-0
-rw-rw-r-- 1 prompt prompt 100 Sep 10 19:18 model.txt
-rw-rw-r-- 1 prompt prompt 403 Sep 10 19:18 README.txt
-rwxrwxr-x 1 prompt prompt 54 Sep 10 19:18 run.sh
-rw-rw-r-- 1 prompt prompt 12 Sep 10 19:18 testcheck.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/arraySize$
```

# Specifying initialization of function pointers to NULL

## PROSE API MODEL

```
global settings:  
    init funcptrs to null on;  
data models:  
function models:  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
struct A {  
    int (*bar)();  
    int a;  
};
```

```
typedef struct A A_t;  
  
void foo(A_t *a1) {  
L1: if (a1->bar)  
L2: assert(0);  
L3: int b = a1->a;  
}
```

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd global_settings/initFuncPtrsToNull  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 1  
No assertion error at line L2
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/global_settings/initFuncPtrsToNull
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/initFuncPtrsToNull$ pwd
/home/prompt/PROMPT/PROMPT_examples/global_settings/initFuncPtrsToNull
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/initFuncPtrsToNull$ ./run.sh foo.bc 2>/dev/null 1>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/initFuncPtrsToNull$ ls -l klee-last/
total 128
-rw-rw-r-- 1 prompt prompt 79116 Sep 24 15:31 assembly.ll
-rw-rw-r-- 1 prompt prompt 636 Sep 24 15:31 info
-rw-rw-r-- 1 prompt prompt 31 Sep 24 15:31 messages.txt
-rw-rw-r-- 1 prompt prompt 26885 Sep 24 15:31 run.istats
-rw-rw-r-- 1 prompt prompt 569 Sep 24 15:31 run.stats
-rw-rw-r-- 1 prompt prompt 17 Sep 24 15:31 test000001.kquery
-rw-rw-r-- 1 prompt prompt 374 Sep 24 15:31 test000001.ktest
-rw-rw-r-- 1 prompt prompt 0 Sep 24 15:31 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/initFuncPtrsToNull$ ls -l
total 36
-rwxrwxr-x 1 prompt prompt 42 Sep 10 19:18 compile.sh
-rw-rw-r-- 1 prompt prompt 2636 Sep 10 19:18 foo.bc
-rw-rw-r-- 1 prompt prompt 156 Sep 10 19:18 foo.c
lrwxrwxrwx 1 prompt prompt 81 Sep 24 15:31 klee-last -> /home/prompt/PROMPT/PROMPT_examples/global_settings/initFuncPtrsToNull/klee-out-0
drwxrwxr-x 2 prompt prompt 4096 Sep 24 15:31 klee-out-0
-rw-rw-r-- 1 prompt prompt 142 Sep 10 19:18 model.txt
-rw-rw-r-- 1 prompt prompt 574 Sep 10 19:18 README.txt
-rwxrwxr-x 1 prompt prompt 54 Sep 10 19:18 run.sh
-rw-rw-r-- 1 prompt prompt 15 Sep 10 19:18 testcheck.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/initFuncPtrsToNull$
```

# Specifying modeling of functions with inline assembly

## PROSE API MODEL

```
global settings:  
    model funcs with asm on  
but_use_original_with_pattern bar  
        except bar1;  
            symbolize inline asm on;  
data models:  
function models:  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
int foo1() {__asm__(...);return 0;}  
  
int bar() {__asm__(...);return 0;}  
  
int bar1() {__asm__(...);return 1;}
```

```
void foo(int *a) {  
L1:    int y = foo1();  
L2:    if (y)  
L3:        assert(0);  
L4:    int z = bar();  
L5:    if (z)  
L6:        assert(0);  
L7:    int w = bar1();  
L8:    if (w != 1)  
L9:        assert(0); }
```

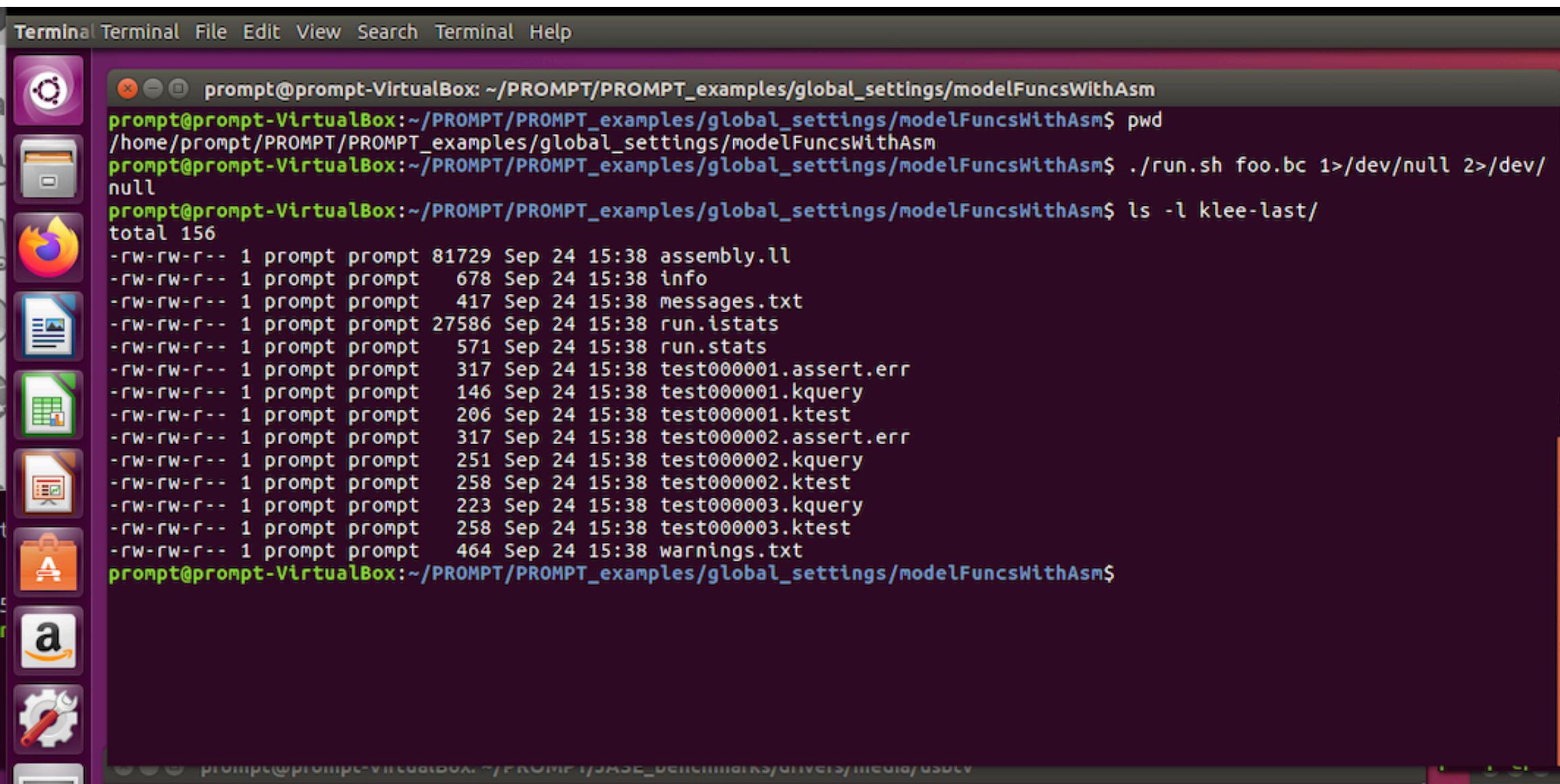
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd global_settings/modelFuncsWithAsm  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 3  
Assertion error at line L3  
No assertion error at line L6  
Assertion error at line L9
```

# Let's try it on the VM!

A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window has a dark background and light-colored text. It displays a command-line session where the user is navigating through a directory structure and running a script to produce assembly output. The desktop interface includes a dock with various application icons like Dash, Home, File Explorer, and a browser.

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm$ pwd
/home/prompt/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm$ ls -l klee-last/
total 156
-rw-rw-r-- 1 prompt prompt 81729 Sep 24 15:38 assembly.ll
-rw-rw-r-- 1 prompt prompt    678 Sep 24 15:38 info
-rw-rw-r-- 1 prompt prompt    417 Sep 24 15:38 messages.txt
-rw-rw-r-- 1 prompt prompt 27586 Sep 24 15:38 run.istats
-rw-rw-r-- 1 prompt prompt    571 Sep 24 15:38 run.stats
-rw-rw-r-- 1 prompt prompt   317 Sep 24 15:38 test000001.assert.err
-rw-rw-r-- 1 prompt prompt   146 Sep 24 15:38 test000001.kquery
-rw-rw-r-- 1 prompt prompt   206 Sep 24 15:38 test000001.ktest
-rw-rw-r-- 1 prompt prompt   317 Sep 24 15:38 test000002.assert.err
-rw-rw-r-- 1 prompt prompt   251 Sep 24 15:38 test000002.kquery
-rw-rw-r-- 1 prompt prompt   258 Sep 24 15:38 test000002.ktest
-rw-rw-r-- 1 prompt prompt   223 Sep 24 15:38 test000003.kquery
-rw-rw-r-- 1 prompt prompt   258 Sep 24 15:38 test000003.ktest
-rw-rw-r-- 1 prompt prompt   464 Sep 24 15:38 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm$
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm
-rw-rw-r-- 1 prompt prompt 571 Sep 24 15:38 run.stats
-rw-rw-r-- 1 prompt prompt 317 Sep 24 15:38 test000001.assert.err
-rw-rw-r-- 1 prompt prompt 146 Sep 24 15:38 test000001.kquery
-rw-rw-r-- 1 prompt prompt 206 Sep 24 15:38 test000001.ktest
-rw-rw-r-- 1 prompt prompt 317 Sep 24 15:38 test000002.assert.err
-rw-rw-r-- 1 prompt prompt 251 Sep 24 15:38 test000002.kquery
-rw-rw-r-- 1 prompt prompt 258 Sep 24 15:38 test000002.ktest
-rw-rw-r-- 1 prompt prompt 223 Sep 24 15:38 test000003.kquery
-rw-rw-r-- 1 prompt prompt 258 Sep 24 15:38 test000003.ktest
-rw-rw-r-- 1 prompt prompt 464 Sep 24 15:38 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm$ more klee-last/test000001.assert.err
Error: ASSERTION FAIL: 0
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/modelFuncsWithAsm/foo.c
Line: 36
assembly.ll line: 51
Stack:
#000000051 in foo (a=54731760) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/modelFuncsWithAsm/foo.c:36
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm$ more klee-last/test000002.assert.err
Error: ASSERTION FAIL: 0
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/modelFuncsWithAsm/foo.c
Line: 42
assembly.ll line: 73
Stack:
#000000073 in foo (a=54731760) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/modelFuncsWithAsm/foo.c:42
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/modelFuncsWithAsm$
```

# Specifying NULL return option for modeled functions

## PROSE API MODEL

```
global settings:  
    null return on;  
data models:  
function models:  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
int *bar();  
  
void foo(int *a) {  
L1: int *y = bar();  
L2: int z = *y;  
}
```

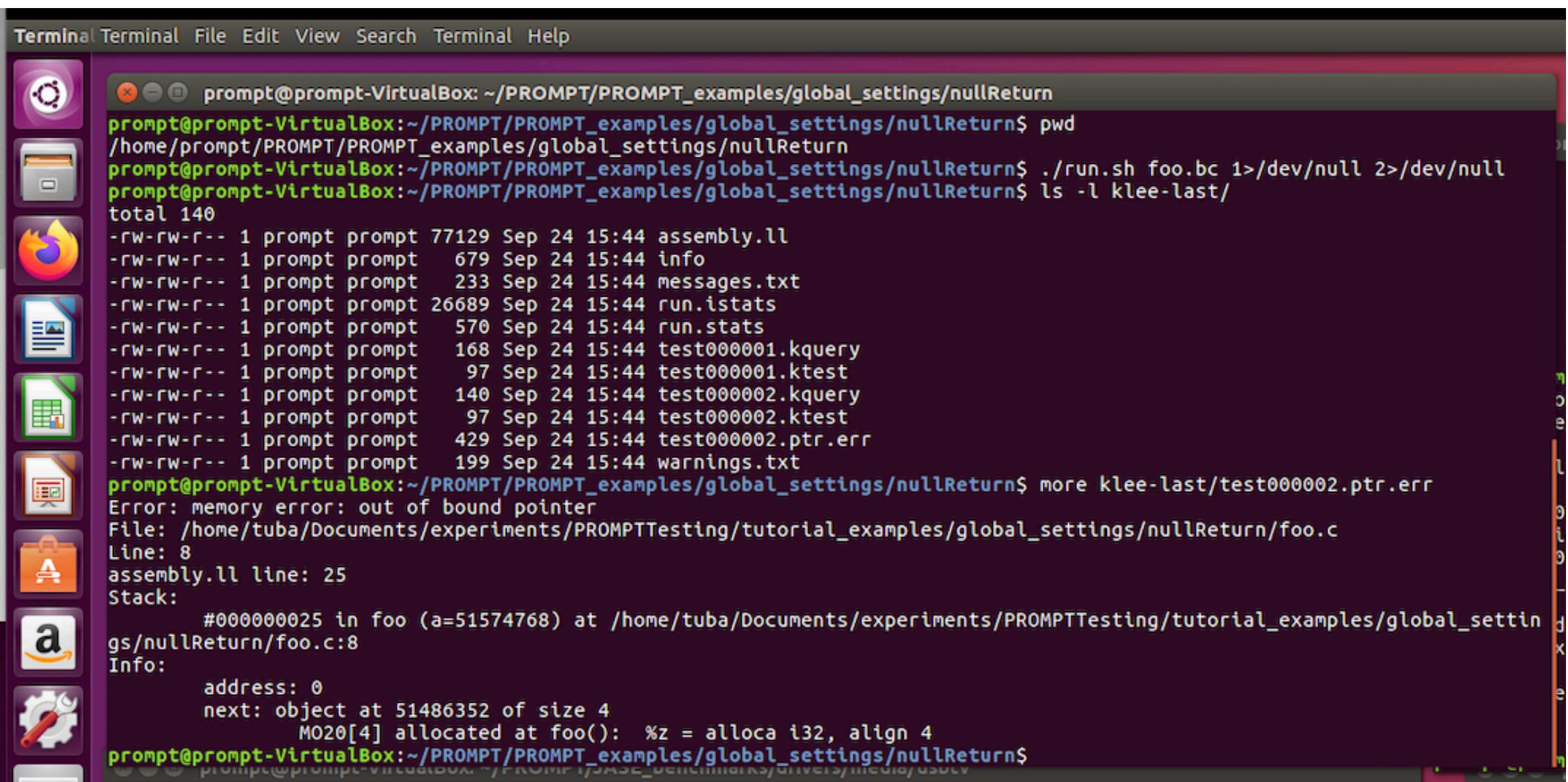
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd global_settings/nullReturn  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 2  
Memory error at L2
```

# Let's try it on the VM!

A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and contains a command-line session. The session starts with the user navigating to a directory, running a script, and then listing files. It ends with the user viewing the contents of a file and seeing a memory error message from KLEE. The desktop interface includes a dock with icons for various applications like the Dash, Home, File Manager, and a browser.

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/global_settings/nullReturn
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/nullReturn$ pwd
/home/prompt/PROMPT/PROMPT_examples/global_settings/nullReturn
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/nullReturn$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/nullReturn$ ls -l klee-last/
total 140
-rw-rw-r-- 1 prompt prompt 77129 Sep 24 15:44 assembly.ll
-rw-rw-r-- 1 prompt prompt 679 Sep 24 15:44 info
-rw-rw-r-- 1 prompt prompt 233 Sep 24 15:44 messages.txt
-rw-rw-r-- 1 prompt prompt 26689 Sep 24 15:44 run.istats
-rw-rw-r-- 1 prompt prompt 570 Sep 24 15:44 run.stats
-rw-rw-r-- 1 prompt prompt 168 Sep 24 15:44 test000001.kquery
-rw-rw-r-- 1 prompt prompt 97 Sep 24 15:44 test000001.ktest
-rw-rw-r-- 1 prompt prompt 140 Sep 24 15:44 test000002.kquery
-rw-rw-r-- 1 prompt prompt 97 Sep 24 15:44 test000002.ktest
-rw-rw-r-- 1 prompt prompt 429 Sep 24 15:44 test000002.ptr.err
-rw-rw-r-- 1 prompt prompt 199 Sep 24 15:44 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/nullReturn$ more klee-last/test000002.ptr.err
Error: memory error: out of bound pointer
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/nullReturn/foo.c
Line: 8
assembly.ll line: 25
Stack:
#000000025 in foo (a=51574768) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/nullReturn/foo.c:8
Info:
address: 0
next: object at 51486352 of size 4
M020[4] allocated at foo(): %z = alloca i32, align 4
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/nullReturn$
```

# Specifying skipping of havocing singleton argument types

## PROSE API MODEL

```
global settings:  
    skip havocing singletons on;  
data models:  
    singleton A;  
function models:  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
struct A {  
    int a;  
};  
struct B {  
    int b;  
};  
void bar(struct A *,  
        struct B *);
```

```
void foo(struct A *a1,  
        struct B *b1) {  
L1: a1->a = 5;  
L2: b1->b = 10;  
L3: bar(a1, b1);  
L4: if (a1->a != 5)  
L5:     assert(0);  
L6: if (b1->b != 10)  
L7:     assert(0);}
```

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd global_settings/skipHavocingSingletons  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 2  
No assertion failure at line L5  
Assertion failure at line L7
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/global_settings/skipHavocingSingletons
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/skipHavocingSingletons$ pwd
/home/prompt/PROMPT/PROMPT_examples/global_settings/skipHavocingSingletons
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/skipHavocingSingletons$ ./run.sh foo.bc 1>/dev/null 2>
/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/skipHavocingSingletons$ ls -l klee-last/
total 144
-rw-rw-r-- 1 prompt prompt 81289 Sep 24 15:47 assembly.ll
-rw-rw-r-- 1 prompt prompt 678 Sep 24 15:47 info
-rw-rw-r-- 1 prompt prompt 229 Sep 24 15:47 messages.txt
-rw-rw-r-- 1 prompt prompt 27346 Sep 24 15:47 run.istats
-rw-rw-r-- 1 prompt prompt 571 Sep 24 15:47 run.stats
-rw-rw-r-- 1 prompt prompt 117 Sep 24 15:47 test000001.kquery
-rw-rw-r-- 1 prompt prompt 233 Sep 24 15:47 test000001.ktest
-rw-rw-r-- 1 prompt prompt 341 Sep 24 15:47 test000002.assert.err
-rw-rw-r-- 1 prompt prompt 145 Sep 24 15:47 test000002.kquery
-rw-rw-r-- 1 prompt prompt 233 Sep 24 15:47 test000002.ktest
-rw-rw-r-- 1 prompt prompt 230 Sep 24 15:47 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/skipHavocingSingletons$ more klee-last/test000002.assert.err
Error: ASSERTION FAIL: 0
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/global_settings/skipHavocingSingletons/foo.c
Line: 21
assembly.ll line: 55
Stack:
#000000055 in foo (a1=37838192, b1=37840928) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples
/global_settings/skipHavocingSingletons/foo.c:21
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/global_settings/skipHavocingSingletons$
```

# Modeling with PROSE

- Global settings
- **Data modeling**
- Function modeling
- Lifecycle modeling

# Specifying initialization of a function pointer field with some function

## PROSE API MODEL

```
global settings:  
data models:  
  (bar = x) where bar is function,  
    x is B field 2;  
  (x > 0) where x is foo arg 0;  
function models:  
lifecycle model:  
  entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
struct B {  
    int b1;  
    int b2;  
    void (*f)(void);  
};  
void bar()  
{L4: assert(0)}
```

```
void foo(int a, struct B *b) {  
    L1: if (a < 0)  
    L2: assert(0);  
    L3: b->f();  
}
```

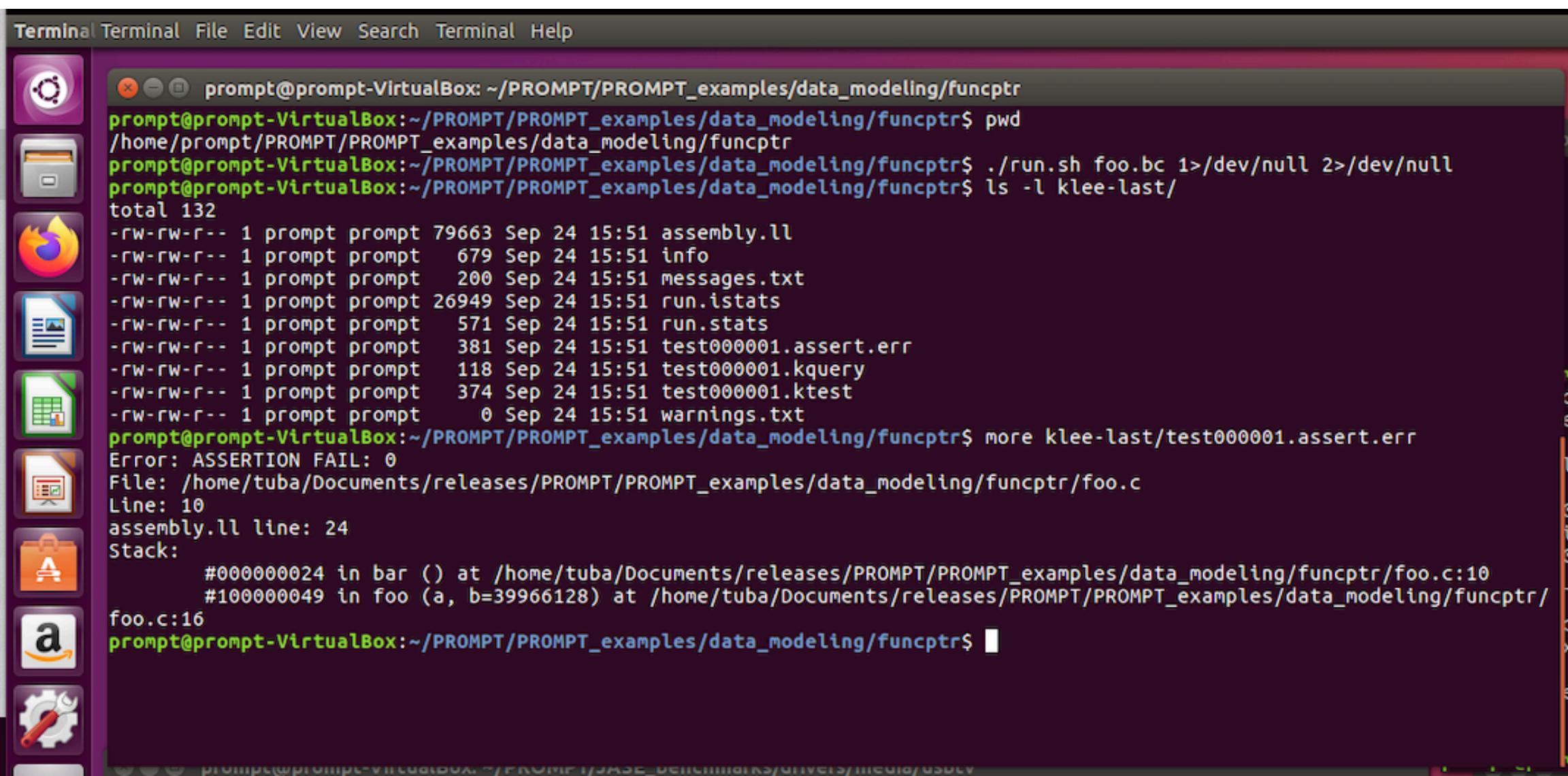
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd data_modeling/funcptr  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 1  
No assertion failure at line L2  
No invalid function pointer at L3  
Assertion failure at line L4
```

# Let's try it on the VM!

A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window has a dark background and light-colored text. It displays a command-line session where a user is testing a program named 'foo'. The session starts with the user navigating to a directory, running a script to build the program, and then running the program itself. The user then lists files in a directory and views the contents of a specific error file using the 'more' command. The terminal window is titled 'Terminal' and is located in the top panel of the desktop environment. The desktop panel also includes icons for various applications like the Dash, Home, and Help.

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/funcptr
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/funcptr$ pwd
/home/prompt/PROMPT/PROMPT_examples/data_modeling/funcptr
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/funcptr$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/funcptr$ ls -l klee-last/
total 132
-rw-rw-r-- 1 prompt prompt 79663 Sep 24 15:51 assembly.ll
-rw-rw-r-- 1 prompt prompt 679 Sep 24 15:51 info
-rw-rw-r-- 1 prompt prompt 200 Sep 24 15:51 messages.txt
-rw-rw-r-- 1 prompt prompt 26949 Sep 24 15:51 run.istats
-rw-rw-r-- 1 prompt prompt 571 Sep 24 15:51 run.stats
-rw-rw-r-- 1 prompt prompt 381 Sep 24 15:51 test000001.assert.err
-rw-rw-r-- 1 prompt prompt 118 Sep 24 15:51 test000001.kquery
-rw-rw-r-- 1 prompt prompt 374 Sep 24 15:51 test000001.ktest
-rw-rw-r-- 1 prompt prompt 0 Sep 24 15:51 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/funcptr$ more klee-last/test000001.assert.err
Error: ASSERTION FAIL: 0
File: /home/tuba/Documents/releases/PROMPT/PROMPT_examples/data_modeling/funcptr/foo.c
Line: 10
assembly.ll line: 24
Stack:
#000000024 in bar () at /home/tuba/Documents/releases/PROMPT/PROMPT_examples/data_modeling/funcptr/foo.c:10
#100000049 in foo (a, b=39966128) at /home/tuba/Documents/releases/PROMPT/PROMPT_examples/data_modeling/funcptr/
foo.c:16
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/funcptr$
```

# Specifying type embedding relationships

## PROSE API MODEL

```
global settings:  
data models:  
    type A embeds type B;  
function models:  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
#define cont_of(ptr, type, member) ({ // container_of  
    void *_mptr = (void *)(ptr);  
    ((type *)(_mptr - offsetof(type, member)));})  
  
struct A {  
    int a;  
    struct B b;  
    char c;  
};  
  
int foo(int x, struct B *b) {struct A *ep;  
L1: ep = cont_of(b, struct A, b);  
L2: if (x > 0)  
L3:     ep->a = x;  
L4: else  
L5:     ep->a = -x;  
L6: return ep->a;  
}
```

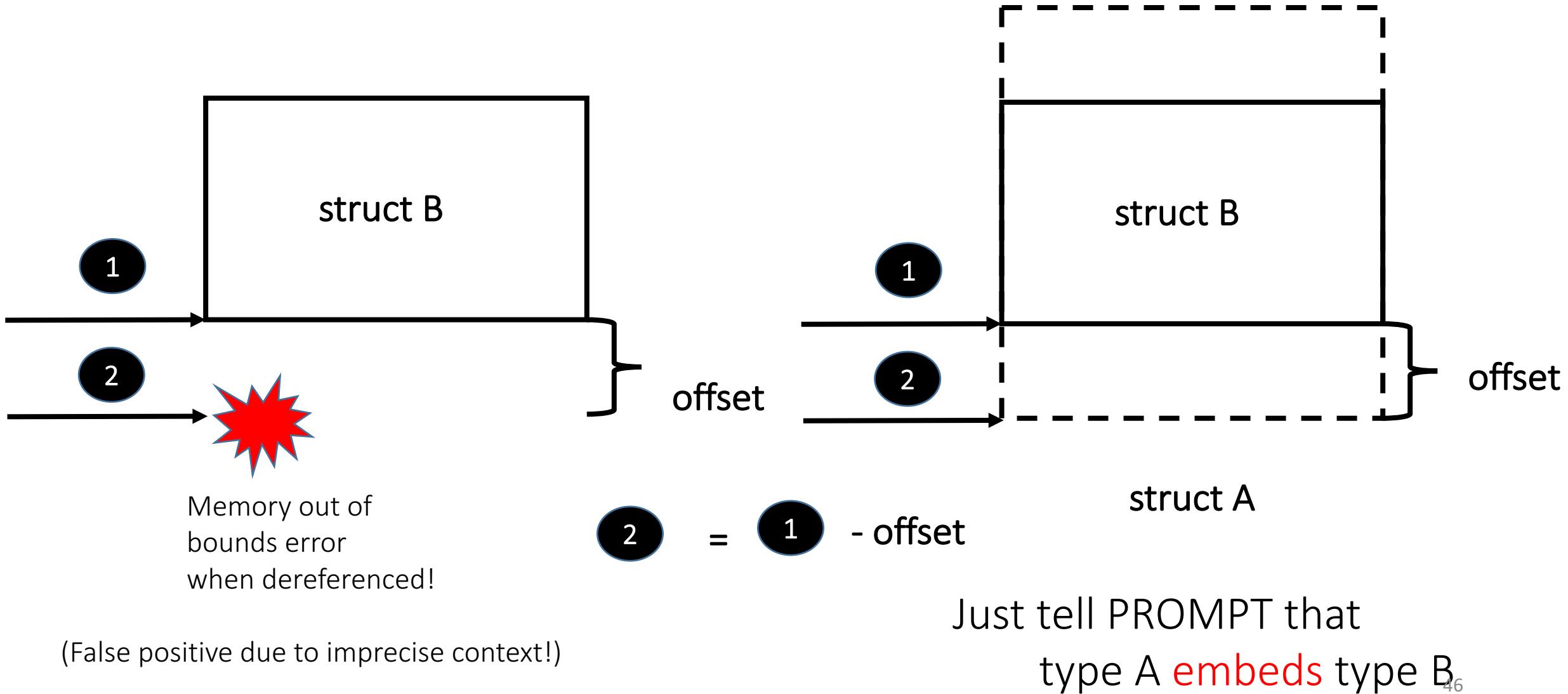
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd data_modeling/embeds  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 2  
No memory errors at lines L3, L5,  
and L6
```

# Pointer arithmetic using a negative offset



# Specifying type embedding relationships

## PROSE API MODEL

```
global settings:  
data models:  
    type A embeds type B;  
function models:  
lifecycle model:  
    entry-point foo
```

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd data_modeling/embeds  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## COMPONENT UNDER ANALYSIS

```
#define cont_of(ptr, type, member) ({ // container_of  
    void *_mptr = (void *)(ptr);  
    ((type *)(_mptr - offsetof(type, member)));})
```

```
struct A {  
    int a;  
    struct B b;  
    char c;  
};
```

```
int foo(int x, struct B *b) {  
    struct A *ep;  
    L1: ep = cont_of(b, struct A,b);  
    L2: if (x > 0)  
    L3:     ep->a = x;  
    L4: else  
    L5:     ep->a = -x;  
    L6: return ep->a; }
```

## PROMPT RESULTS:

```
# of paths: 2  
No memory errors at lines L3, L5,  
and L6
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help

prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/embeds
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/embeds$ pwd
/home/prompt/PROMPT/PROMPT_examples/data_modeling/embeds
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/embeds$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/embeds$ ls -l klee-last/
total 136
-rw-rw-r-- 1 prompt prompt 79997 Sep 24 15:55 assembly.ll
-rw-rw-r-- 1 prompt prompt 678 Sep 24 15:55 info
-rw-rw-r-- 1 prompt prompt 31 Sep 24 15:55 messages.txt
-rw-rw-r-- 1 prompt prompt 27447 Sep 24 15:55 run.istats
-rw-rw-r-- 1 prompt prompt 571 Sep 24 15:55 run.stats
-rw-rw-r-- 1 prompt prompt 146 Sep 24 15:55 test000001.kquery
-rw-rw-r-- 1 prompt prompt 70 Sep 24 15:55 test000001.ktest
-rw-rw-r-- 1 prompt prompt 118 Sep 24 15:55 test000002.kquery
-rw-rw-r-- 1 prompt prompt 70 Sep 24 15:55 test000002.ktest
-rw-rw-r-- 1 prompt prompt 0 Sep 24 15:55 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/embeds$ more klee-last/test000001.kquery
array foo_0_arg_0[4] : w32 -> w8 = symbolic
(query [(Eq false
          (Slt 0
              (ReadLSB w32 0 foo_0_arg_0)))]
      false)
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/embeds$ more klee-last/test000002.kquery
array foo_0_arg_0[4] : w32 -> w8 = symbolic
(query [(Slt 0
          (ReadLSB w32 0 foo_0_arg_0))]
      false)
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/embeds$
```

# Specifying a constant size for a function argument of a pointer type

## PROSE API MODEL

```
global settings:  
data models:  
  (35 = w) where w is  
    argsize foo arg 1;  
function models:  
lifecycle model:  
entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
void foo(int a, int *a2) {  
L1:    a2[34]=4;  
}
```

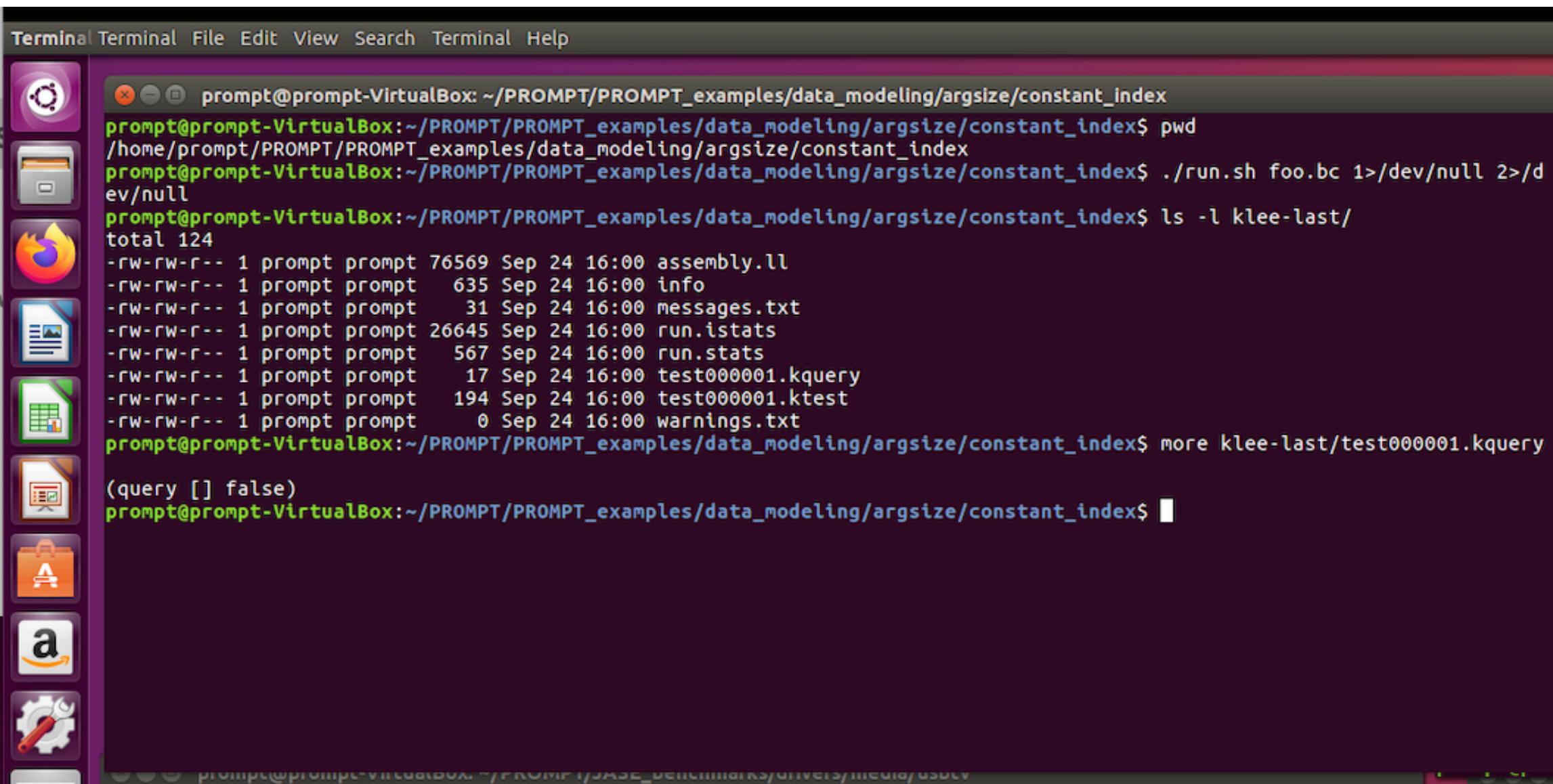
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd data_modeling/arg_size/constant_index  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 1  
No memory error a line L1
```

# Let's try it on the VM!

A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window has a dark background and light-colored text. It displays a series of commands being run in a terminal session. The session starts with the user navigating to a directory, then running a script to produce output files, listing those files, and finally viewing the contents of one of the files.

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/data_modeling/argsize/constant_index
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/constant_index$ pwd
/home/prompt/PROMPT/PROMPT_examples/data_modeling/argsize/constant_index
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/constant_index$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/constant_index$ ls -l klee-last/
total 124
-rw-rw-r-- 1 prompt prompt 76569 Sep 24 16:00 assembly.ll
-rw-rw-r-- 1 prompt prompt 635 Sep 24 16:00 info
-rw-rw-r-- 1 prompt prompt 31 Sep 24 16:00 messages.txt
-rw-rw-r-- 1 prompt prompt 26645 Sep 24 16:00 run.istats
-rw-rw-r-- 1 prompt prompt 567 Sep 24 16:00 run.stats
-rw-rw-r-- 1 prompt prompt 17 Sep 24 16:00 test000001.kquery
-rw-rw-r-- 1 prompt prompt 194 Sep 24 16:00 test000001.ktest
-rw-rw-r-- 1 prompt prompt 0 Sep 24 16:00 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/constant_index$ more klee-last/test000001.kquery
(query [] false)
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/constant_index$
```

# Specifying the size for a function argument of a pointer type with a simple equality

## PROSE API MODEL

```
global settings:  
data models:  
  (x = w) where x is foo arg 0,  
            w is argsize foo arg 1;  
function models:  
lifecycle model:  
entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
void foo(int a, int *a2) {  
L1:      a2[a-1]=4;  
}
```

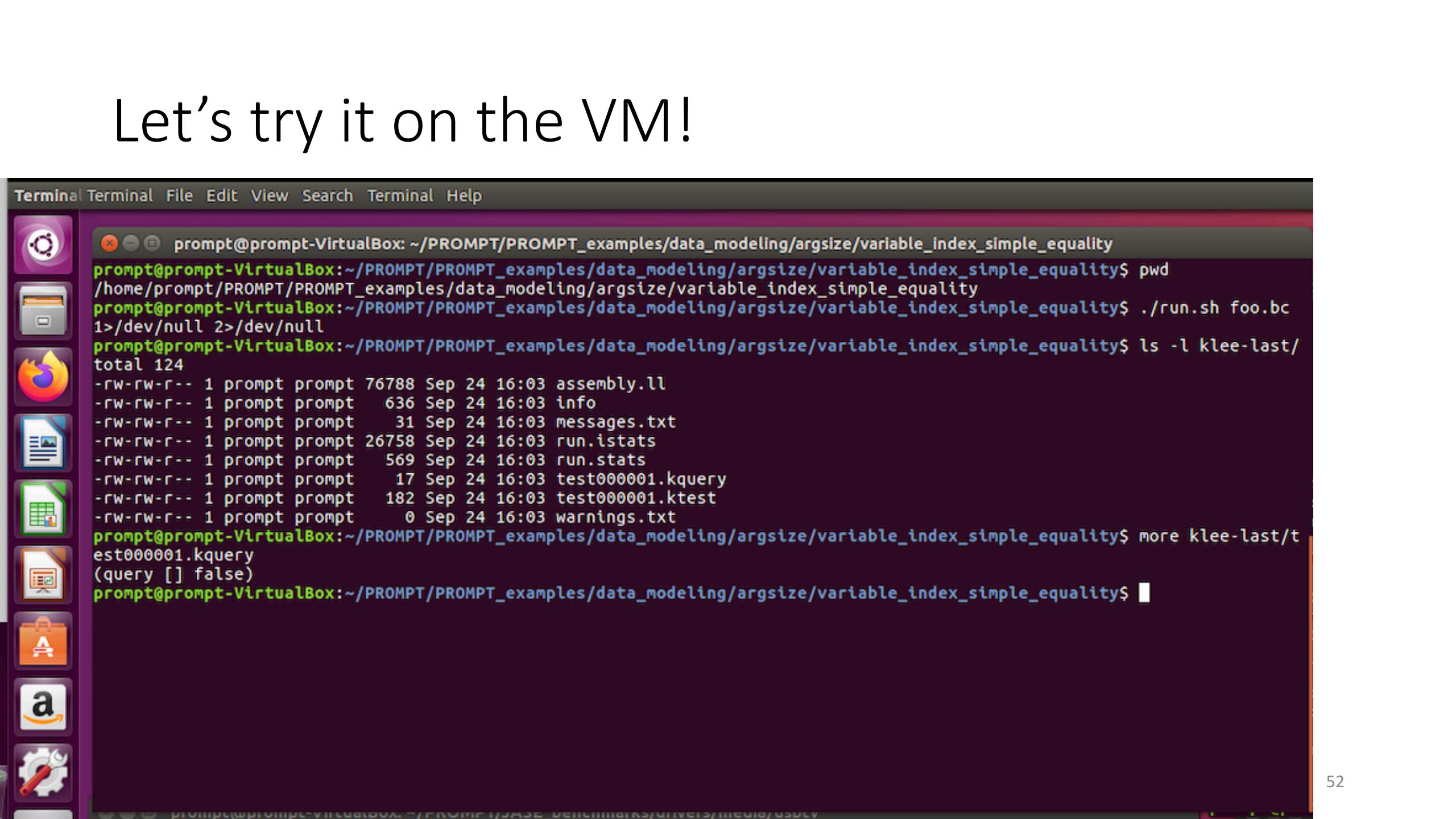
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd data_modeling/arg_size/variable_index_simple_equality  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 1  
No memory error at line L1
```

# Let's try it on the VM!

A screenshot of an Ubuntu desktop environment. On the left, there is a vertical dock with icons for various applications: Dash, Home, File Explorer, Firefox, LibreOffice Writer, LibreOffice Calc, LibreOffice Impress, Amazon Appstore, and System Settings. The main window is a terminal window titled "Terminal". The terminal shows the following command-line session:

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_simple_equality
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_simple_equality$ pwd
/home/prompt/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_simple_equality
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_simple_equality$ ./run.sh foo.bc
1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_simple_equality$ ls -l klee-last/
total 124
-rw-rw-r-- 1 prompt prompt 76788 Sep 24 16:03 assembly.ll
-rw-rw-r-- 1 prompt prompt 636 Sep 24 16:03 info
-rw-rw-r-- 1 prompt prompt 31 Sep 24 16:03 messages.txt
-rw-rw-r-- 1 prompt prompt 26758 Sep 24 16:03 run.istats
-rw-rw-r-- 1 prompt prompt 569 Sep 24 16:03 run.stats
-rw-rw-r-- 1 prompt prompt 17 Sep 24 16:03 test000001.kquery
-rw-rw-r-- 1 prompt prompt 182 Sep 24 16:03 test000001.ktest
-rw-rw-r-- 1 prompt prompt 0 Sep 24 16:03 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_simple_equality$ more klee-last/test000001.kquery
(query [] false)
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_simple_equality$
```

The terminal window has a dark background with light-colored text. The application menu bar at the top includes "Terminal", "File", "Edit", "View", "Search", "Terminal", and "Help".

# Specifying a variable size for a function argument of a pointer type

## PROSE API MODEL

```
global settings:  
data models:  
  ((x + y) < w) where x is foo arg 0,  
                      y is foo arg 1,  
                      w is argsize foo arg 2;  
  ((x + y) > 0) where x is foo arg 0,  
                      y is foo arg 1;  
function models: lifecycle model:  
  entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
void foo(int a, int b, int *a2) {  
L1:    a2[a+b-1]=4;  
}
```

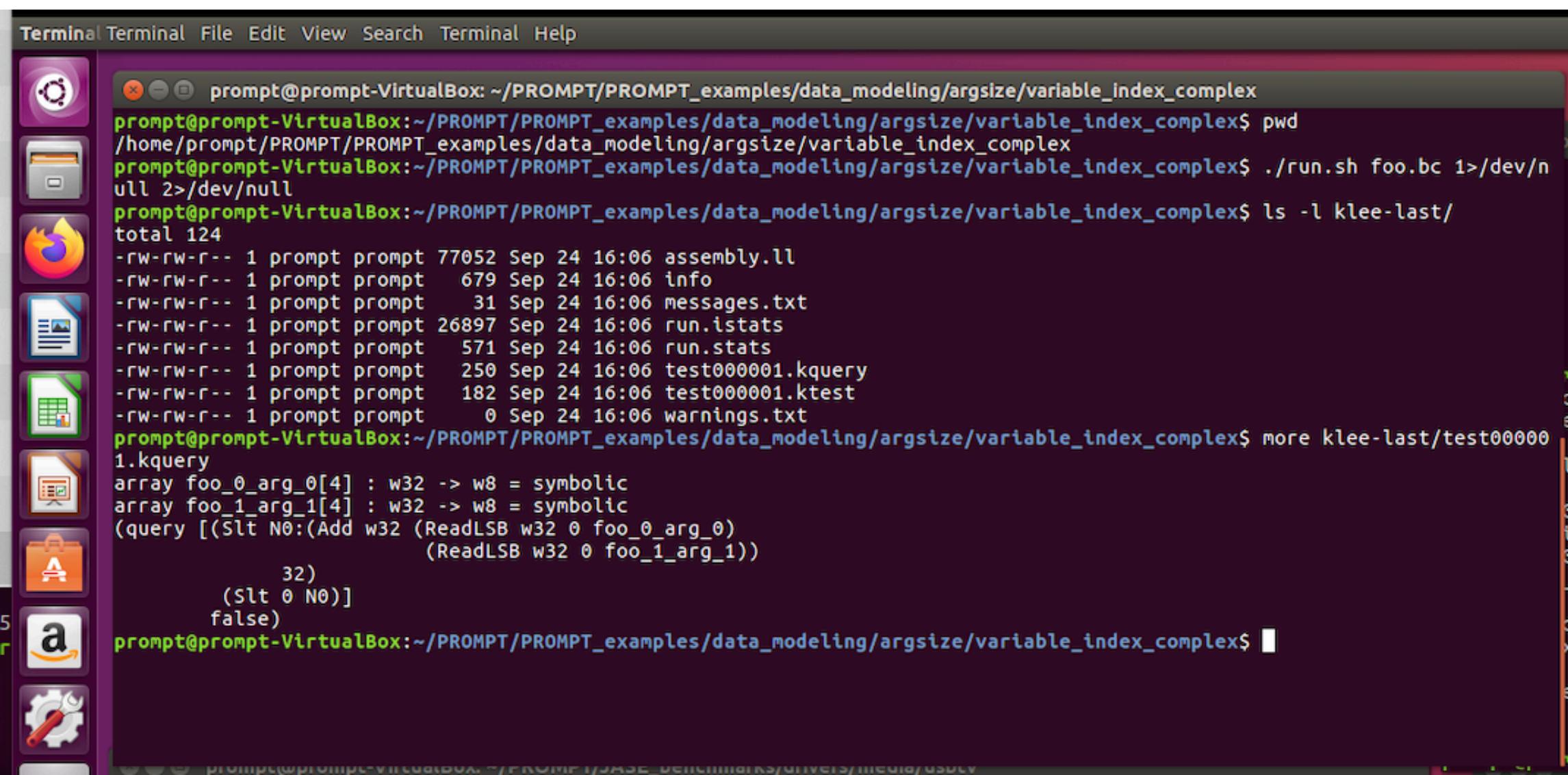
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd data_modeling/arg_size/variable_index_complex  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 1  
No memory error at line L1
```

# Let's try it on the VM!

A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window has a dark background and light-colored text. It displays a command-line session where a user is navigating through a directory of KLEE test cases and viewing assembly code. The desktop interface includes a dock with various icons for applications like the Dash, Home, and Dash to Dock. The terminal window title bar shows the path: ~/PROMPT/PROMPT\_examples/data\_modeling/argsize/variable\_index\_complex. The session starts with the user navigating to the directory, then running a script to generate assembly code, followed by listing files in the klee-last/ directory, and finally viewing the assembly code for a specific test case.

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_complex
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_complex$ pwd
/home/prompt/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_complex
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_complex$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_complex$ ls -l klee-last/
total 124
-rw-rw-r-- 1 prompt prompt 77052 Sep 24 16:06 assembly.ll
-rw-rw-r-- 1 prompt prompt 679 Sep 24 16:06 info
-rw-rw-r-- 1 prompt prompt 31 Sep 24 16:06 messages.txt
-rw-rw-r-- 1 prompt prompt 26897 Sep 24 16:06 run.istats
-rw-rw-r-- 1 prompt prompt 571 Sep 24 16:06 run.stats
-rw-rw-r-- 1 prompt prompt 250 Sep 24 16:06 test000001.kquery
-rw-rw-r-- 1 prompt prompt 182 Sep 24 16:06 test000001.ktest
-rw-rw-r-- 1 prompt prompt 0 Sep 24 16:06 warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_complex$ more klee-last/test000001.kquery
array foo_0_arg_0[4] : w32 -> w8 = symbolic
array foo_1_arg_1[4] : w32 -> w8 = symbolic
(query [(Slt N0:(Add w32 (ReadLSB w32 0 foo_0_arg_0)
                           (ReadLSB w32 0 foo_1_arg_1))
           32)
         (Slt 0 N0)]
      false)
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/argsize/variable_index_complex$
```

# Specifying data constraints on the struct fields

## PROSE API MODEL

```
global settings:  
data models:  
  
(x > 24) where x is foo arg 0;  
  
(x = 24) where x is foo arg 1;  
  
(25 = w) where w is sizeof A field 2;  
  
function models:  
lifecycle model:  
entry-point foo
```

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd data_modeling/sizeof/one_level  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## COMPONENT UNDER ANALYSIS

```
struct B {  
    int b1;  
    int b2;  
    void (*f)(void);  
};  
  
struct A {  
    int a; // used as length  
    int c;  
    struct B *b; // field 2  
};
```

```
void foo(int a, int b, struct A  
*a2) {  
    L1: if (a <= 24)  
    L2: assert(0);  
    L3: if (b != 24)  
    L4: assert(0);  
    L5: a2->b[b].b1=4;  
    L6: a2->b[a].b1=7;  
}
```

## PROMPT RESULTS:

```
# of paths: 4  
No assertion failure at lines L2 and L4.  
Memory error at L6
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/one_level_constant_sizeof
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/one_level_constant_sizeof$ pwd
/home/prompt/PROMPT/PROMPT_examples/data_modeling/sizeof/one_level_constant_sizeof
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/one_level_constant_sizeof$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/one_level_constant_sizeof$ ls klee-last/
assembly.ll    run.istats      test000001.ktest  test000002.ktest  test000004.kquery
info           run.stats       test000001.ptr.err  test000003.kquery  test000004.ktest
messages.txt   test000001.kquery  test000002.kquery  test000003.ktest  warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/one_level_constant_sizeof$ more klee-last/test00001.ptr.err
Error: memory error: out of bound pointer
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/data_modeling/sizeof/one_level_constant_sizeof/foo.c
Line: 21
assembly.ll line: 63
Stack:
#000000063 in foo (a, b=24, a2=56822224) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/data_modeling/sizeof/one_level_constant_sizeof/foo.c:21
Info:
address: (Add w64 56304272
(Mul w64 16
(SExt w64 (ReadLSB w32 0 foo_0_arg_0)))
example: 34416042624
range: [56304672, 34416042624]
next: object at 56304272 of size 400
M034[400] allocated at foo(): %16 = load %struct.B*, %struct.B** %15, align 8, !dbg !145
prev: object at 56299152 of size 4
M024[4] allocated at foo(): %2 = alloca i32, align 4
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/one_level_constant_sizeof$
```

# Specifying data constraints for fields of an embedded struct type

## PROSE API MODEL

```
global settings:  
data models:  
  
(x > 24) where x is foo arg 0;  
  
(x = 24) where x is foo arg 1;  
  
(25 = w) where w is sizeof A field 2;  
  
function models:  
lifecycle model:  
entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
struct A {  
    int a;  
    int c;  
    struct B *b;  
};  
  
struct C {  
    char c1;  
    struct A c2;  
};  
  
void foo(int a, int b, struct C  
*c) {  
    L1: struct A *a2 = &c->c2;  
    L2: if (a <= 24)  
    L3: assert(0);  
    L4: if (b != 24)  
    L5: assert(0);  
    L6: a2->b[b].b1=4;  
    L7: a2->b[a].b1=7;  
}
```

## STEPS:

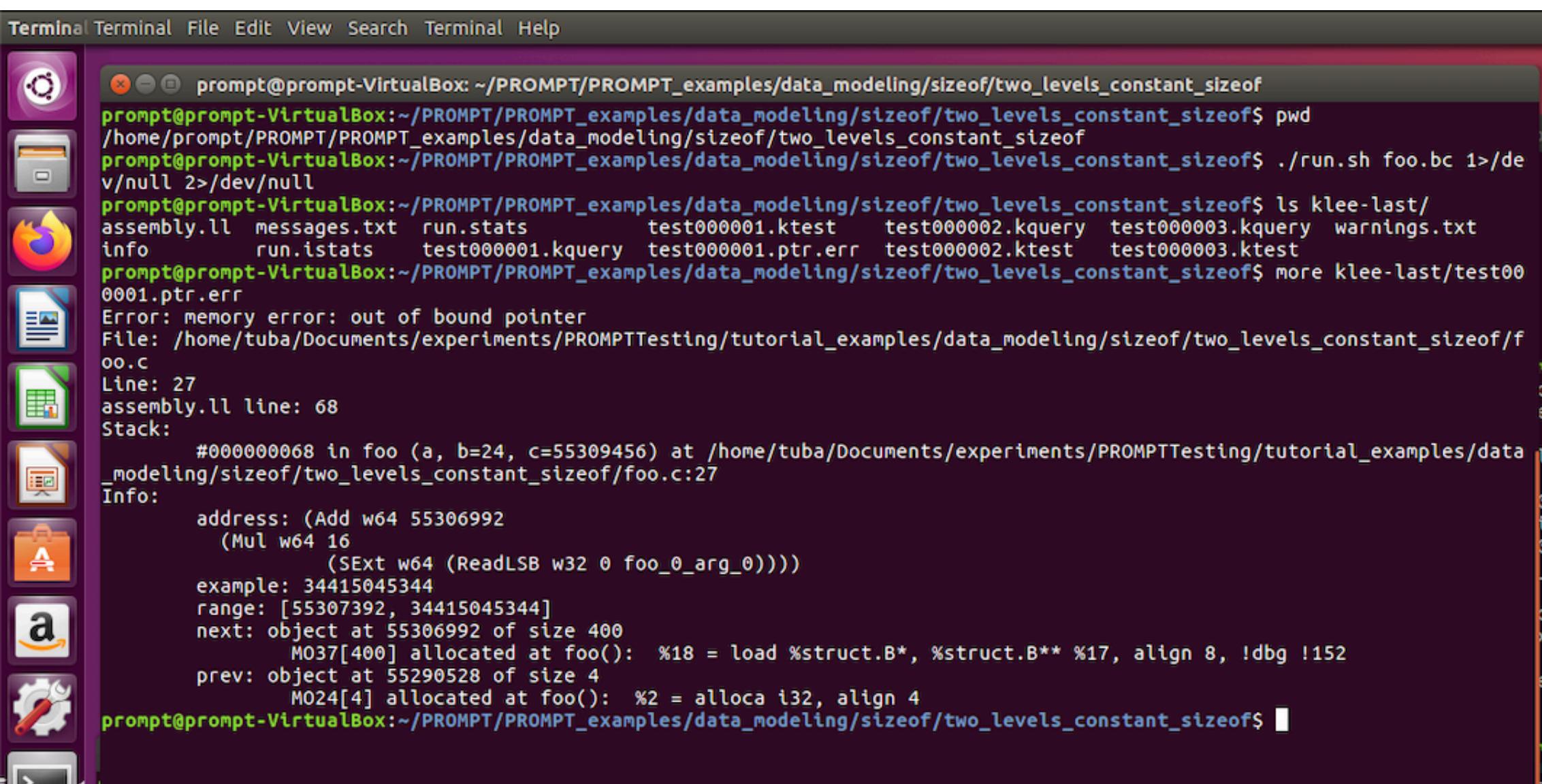
```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd data_modeling/sizeof/two_level  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 3  
No assertion failures at lines L3 and L5.  
Memory error at line L7
```

# Let's try it on the VM!

Terminal Terminal File Edit View Search Terminal Help



```
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/data_modeling/sizeof/two_levels_constant_sizeof
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/two_levels_constant_sizeof$ pwd
/home/prompt/PROMPT/PROMPT_examples/data_modeling/sizeof/two_levels_constant_sizeof
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/two_levels_constant_sizeof$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/two_levels_constant_sizeof$ ls klee-last/
assembly.ll messages.txt run.stats test000001.ktest test000002.kquery test000003.kquery warnings.txt
info run.stats test000001.kquery test000001.ptr.err test000002.ktest test000003.ktest
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/two_levels_constant_sizeof$ more klee-last/test000001.ptr.err
Error: memory error: out of bound pointer
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/data_modeling/sizeof/two_levels_constant_sizeof/foo.c
Line: 27
assembly.ll line: 68
Stack:
#0000000068 in foo (a, b=24, c=55309456) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/data_modeling/sizeof/two_levels_constant_sizeof/foo.c:27
Info:
address: (Add w64 55306992
(Mul w64 16
(SExt w64 (ReadLSB w32 0 foo_0_arg_0))))
example: 34415045344
range: [55307392, 34415045344]
next: object at 55306992 of size 400
M037[400] allocated at foo(): %18 = load %struct.B*, %struct.B** %17, align 8, !dbg !152
prev: object at 55290528 of size 4
M024[4] allocated at foo(): %2 = alloca i32, align 4
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/data_modeling/sizeof/two_levels_constant_sizeof$
```

# Modeling with PROSE

- Global settings
- Data modeling
- **Function modeling**
- Lifecycle modeling

# Specifying functions that perform memory allocations and return the address

## PROSE API MODEL

```
global settings:  
data models:  
function models:  
    alloc bar sizearg 2  
        initzero false symbolize false;  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
char *bar(int a, int b, int c);  
  
void foo(int *a, int b, int c) {  
    L1: c = 10;  
    L2: char *n = bar(a,b,c);  
    L3: n[9] = 'A';  
    L4: n[10] = 'B'; }
```

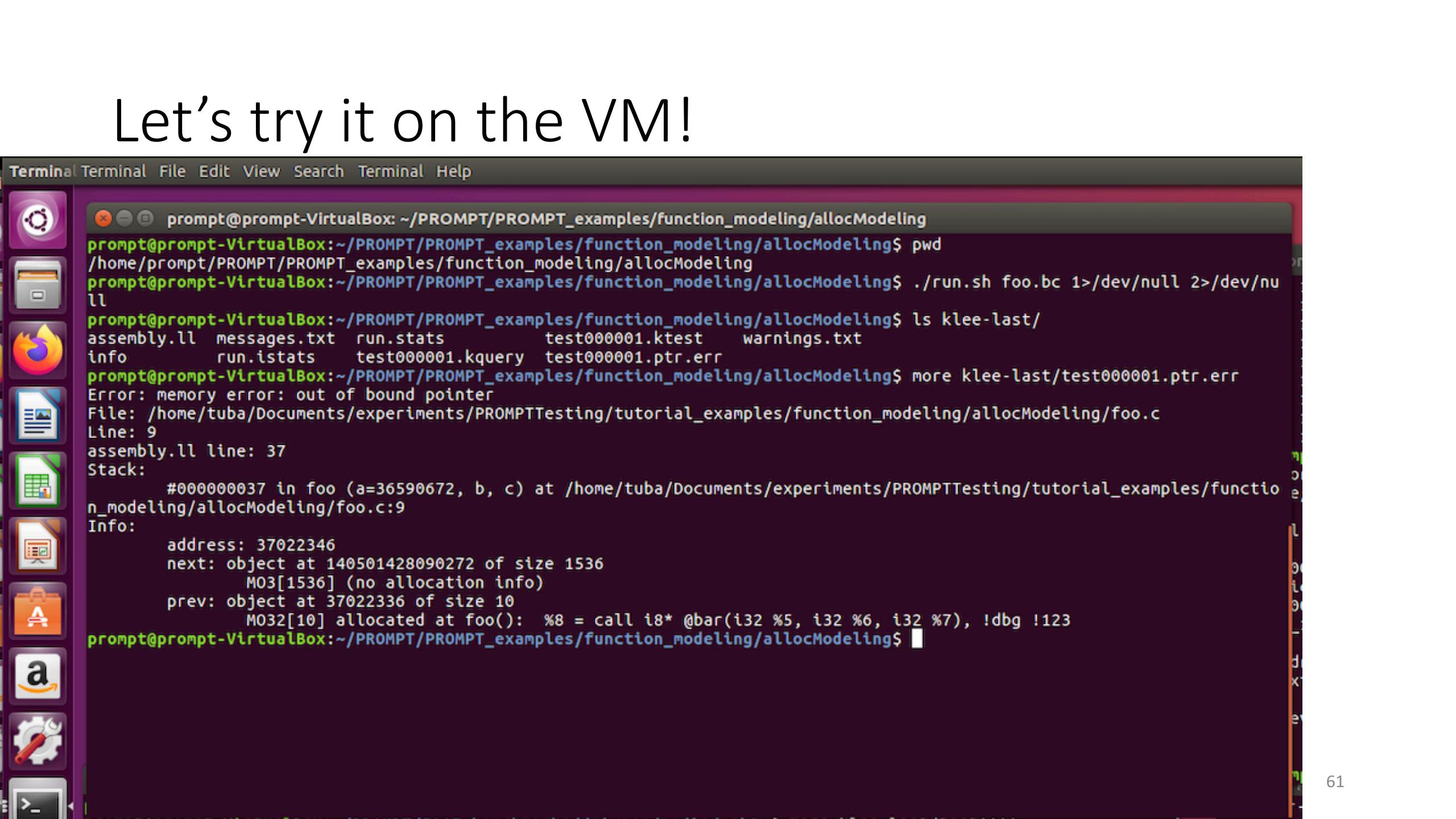
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd function_modeling/allocModeling  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 1  
No memory error at line L3  
Memory error at line L4
```

# Let's try it on the VM!

A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window has a dark background and contains the following text:

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/function_modeling/allocModeling
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocModeling$ pwd
/home/prompt/PROMPT/PROMPT_examples/function_modeling/allocModeling
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocModeling$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocModeling$ ls klee-last/
assembly.ll  messages.txt  run.stats          test000001.ktest  warnings.txt
info         run.istats    test000001.kquery   test000001.ptr.err
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocModeling$ more klee-last/test000001.ptr.err
Error: memory error: out of bound pointer
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/allocModeling/foo.c
Line: 9
assembly.ll line: 37
Stack:
#000000037 in foo (a=36590672, b, c) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/allocModeling/foo.c:9
Info:
address: 37022346
next: object at 140501428090272 of size 1536
M03[1536] (no allocation info)
prev: object at 37022336 of size 10
M032[10] allocated at foo(): %8 = call i8* @bar(i32 %5, i32 %6, i32 %7), !dbg !123
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocModeling$
```

The terminal window is part of a desktop interface with a dock on the left containing icons for various applications like a web browser, file manager, and system tools.

# Specifying functions that perform memory allocations and store the address in an argument

## PROSE API MODEL

```
global settings:  
data models:  
function models:  
    alloc bar initzero false  
    symbolize false  
    memreturn true  
    destarg 3;  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
char *bar(int a, int b, int c, char **d);  
  
void foo(int a, int b, int c) {  
    L1: char *n, *m;  
    L2: c = 10;  
    L3: m = bar(a,b,c,&n);  
    L4: assert(n == m);  
    L5: n[0] = 'A';  
    L6: n[1] = 'B'; }
```

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd function_modeling/allocWithDestModeling  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 1  
No assertion failure at line L4  
No memory error at line L5  
Memory error at line L6
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocWithDestModeling
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocWithDestModeling$ pwd
/home/prompt/PROMPT/PROMPT_examples/function_modeling/allocWithDestModeling
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocWithDestModeling$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocWithDestModeling$ ls klee-last/
assembly.ll messages.txt run.stats test000001.ktest warnings.txt
info run.istats test000001.kquery test000001.ptr.err
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocWithDestModeling$ more klee-last/test000001.ptr.err
Error: memory error: out of bound pointer
File: /home/tuba/Documents/releases/PROMPT/PROMPT_examples/function_modeling/allocWithDestModeling/foo.c
Line: 11
assembly.ll line: 50
Stack:
#000000050 in foo (a, b, c) at /home/tuba/Documents/releases/PROMPT/PROMPT_examples/function_modeling/allocWithDestModeling/foo.c:11
Info:
    address: 38187569
    next: object at 38213728 of size 4
        M020[4] allocated at foo(): %1 = alloca i32, align 4
    prev: object at 38187568 of size 1
        M036[1] allocated at foo(): %7 = call i8* @bar(i32 %4, i32 %5, i32 %6, i8** %n), !dbg !122
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/allocWithDestModeling$
```

# Specifying functions that perform memory deallocation

## PROSE API MODEL

```
global settings:  
data models:  
function models:  
    alloc bar sizearg 2  
        initzero false  
        symbolize false;  
    free zot memarg 1;  
lifecycle model:  
entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
char *bar(int *a, int *b, int c);  
void zot(char *d, char *e);  
  
void foo(int *a, int *b, int c) {  
    L1: char *m,*n;  
    L2: c = 10;  
    L3: n = bar(a,b,c);  
    L4: zot(m,n); // first free  
    L5: zot(m,n); // double-free }
```

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd function_modeling/freeModeling  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 1  
No memory error at line L4  
Memory error at line L5
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/function_modeling/freeModeling
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/freeModeling$ pwd
/home/prompt/PROMPT/PROMPT_examples/function_modeling/freeModeling
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/freeModeling$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/freeModeling$ ls klee-last/
assembly.ll messages.txt run.stats test000001.ktest warnings.txt
info run.istats test000001.kquery test000001.ptr.err
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/freeModeling$ more klee-last/test000001.ptr.err
Error: memory error: invalid pointer: free
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/freeModeling/foo.c
Line: 12
assembly.ll line: 37
Stack:
#000000037 in foo (a=46054736, b=46050496, c) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/freeModeling/foo.c:12
Info:
address: 45966224
next: object at 45966672 of size 8
M024[8] allocated at foo(): %n = alloca i8*, align 8
prev: object at 45799936 of size 16
M012[16] allocated at global:.str.1.4
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/freeModeling$
```

# Specifying functions that are modeled by symbolizing their return value (side-effect free)

## PROSE API MODEL

```
global settings:  
data models:  
function models:  
    returnonly bar;  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
void foo(int *a) {  
    L1: int y = bar(a);  
    L2: if (*a != 5)  
    L3: assert(0);  
    L4: if (y)  
    L5: assert(0);  
  
    int bar(int *a) {  
        *a = 5;  
        return 0;  
    }  
}
```

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd function_modeling/returnOnly  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 3  
Assertion failure at lines L3 and L5
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help

prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/function_modeling/returnOnly
/home/prompt/PROMPT/PROMPT_examples/function_modeling/returnOnly
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/returnOnly$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/returnOnly$ ls klee-last/
assembly.ll    run.istats          test000001.kquery      test000002.kquery      test000003.ktest
info           run.stats           test000001.ktest      test000002.ktest      warnings.txt
messages.txt   test000001.assert.err  test000002.assert.err  test000003.kquery
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/returnOnly$ more klee-last/test000001.assert.err
Error: ASSERTION FAIL: 0
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/returnOnly/foo.c
Line: 13
assembly.ll line: 45
Stack:
#000000045 in foo (a=42712992) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_mode
ling/returnOnly/foo.c:13
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/returnOnly$ more klee-last/test000002.assert.err
Error: ASSERTION FAIL: 0
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/returnOnly/foo.c
Line: 15
assembly.ll line: 54
Stack:
#000000054 in foo (a=42712992) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_mode
ling/returnOnly/foo.c:15
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/returnOnly$ more klee-last/test000002.kquery
array %sym5_bar_0[4] : w32 -> w8 = symbolic
array foo_arg_0[128] : w32 -> w8 = symbolic
(query [(Eq 5
          (ReadLSB w32 0 foo_arg_0))
         (Eq false
             (Eq 0
                 (ReadLSB w32 0 %sym5_bar_0))))]
         false)
prompt@prompt-VirtualBox:~/PROMPT/JASE_benchmarks/drivers/media/usbtv$ more klee-last/test000002.kquery
array %sym5_bar_0[4] : w32 -> w8 = symbolic
array foo_arg_0[128] : w32 -> w8 = symbolic
(query [(Eq 5
          (ReadLSB w32 0 foo_arg_0))
         (Eq false
             (Eq 0
                 (ReadLSB w32 0 %sym5_bar_0))))]
         false)
```

# Specifying returnonly modeling and havocing of certain function arguments

## PROSE API MODEL

```
global settings:  
    skip havocing singletons on;  
data models:  
    singleton A;  
function models:  
    returnonly bar;  
    havoc args 0, 2 of bar;  
    havoc args 0 of zot;  
lifecycle model:  
    entry-point foo
```

## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd function_modeling/havocArgs  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## COMPONENT UNDER ANALYSIS

```
int zot(struct A *a); int bar(int *a, int *b, int *c);  
void foo(int *a, int *b, int *c, struct A *d) {  
    L1: *a = *b = *c = d->a = 5;  
    L2: zot(d);  
    L3: bar(a, b, c);  
    L4: if (*a != 5)  
    L5: assert(0);  
    L6: if (*b != 5)  
    L7: assert(0);  
    L8: if (*c != 5)  
    L9: assert(0);  
    L10: if (d->a != 5)  
    L11: assert(0);  
}
```

## PROMPT RESULTS:

```
# of paths: 4  
Assertions at lines L5, L9, and L11  
reached  
Assertion at line L7 not reached
```

# Let's try it on the VM!

```
Terminal Terminal File Edit View Search Terminal Help
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/function_modeling/havocArgs
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/havocArgs$ pwd
/home/prompt/PROMPT/PROMPT_examples/function_modeling/havocArgs
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/havocArgs$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/havocArgs$ ls klee-last/
assembly.ll      run.istats          test000001.kquery    test000002.kquery    test000003.ktest     test000004.ktest
info             run.stats           test000001.ktest    test000002.ktest    test000004.assert.err  warnings.txt
messages.txt     test000001.assert.err  test000002.assert.err  test000003.kquery   test000004.kquery
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/havocArgs$ more klee-last/test000001.assert.err
Error: ASSERTION FAIL: 0
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/havocArgs/foo.c
Line: 16
assembly.ll line: 52
Stack:
#000000052 in foo (a=29942576, b=29940928, c=29943648, d=29536976) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/havocArgs/foo.c:16
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/havocArgs$ more klee-last/test000001.kquery
array bar_1_arg_0[4] : w32 -> w8 = symbolic
(query [(Eq false
          (Eq 5
              (ReadLSB w32 0 bar_1_arg_0)))]
false)
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/havocArgs$ more klee-last/test000004.assert.err
Error: ASSERTION FAIL: 0
File: /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/havocArgs/foo.c
Line: 22
assembly.ll line: 83
Stack:
#000000083 in foo (a=29942576, b=29940928, c=29943648, d=29536976) at /home/tuba/Documents/experiments/PROMPTTesting/tutorial_examples/function_modeling/havocArgs/foo.c:22
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/havocArgs$
```

# Specifying functions modeled by other functions

## PROSE API MODEL

```
global settings:  
data models:  
function models:  
    bar modeled by zot;  
lifecycle model:  
    entry-point foo
```

## COMPONENT UNDER ANALYSIS

```
void bar(int *a);  
void zot(int *a) { *a = 5; }  
void foo(int *a) {  
    L1: bar(a);  
    L2: if (*a != 5)  
    L3: assert(0);  
}
```

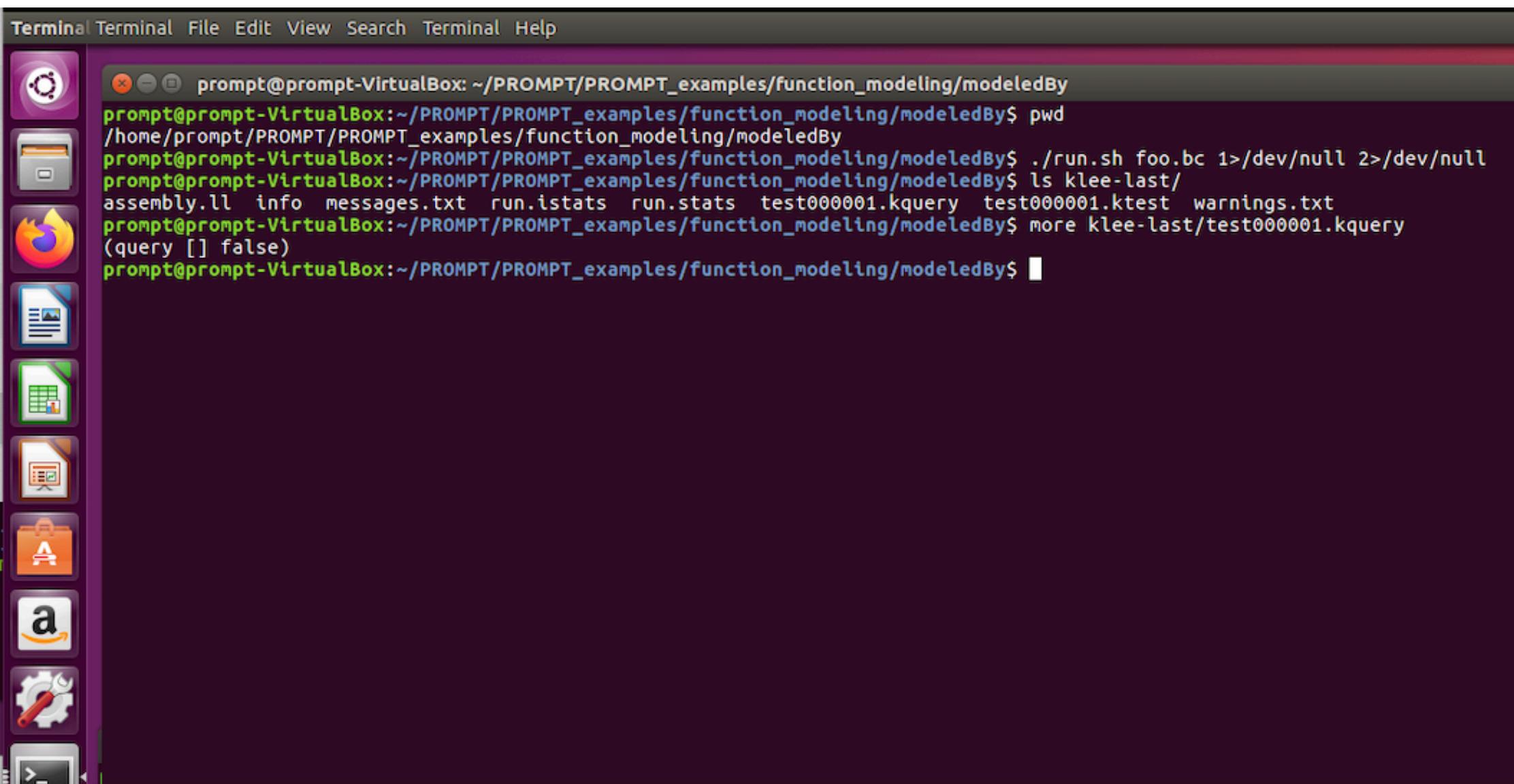
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd function_modeling/modeledBy  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 1  
No assertion failure at line L3
```

# Let's try it on the VM!



A screenshot of an Ubuntu desktop environment. On the left, there is a vertical dock with icons for various applications: Dash, Home, File Explorer, Firefox, Photos, Files, Terminal, Amazon, and Settings. The main window is a terminal window titled "Terminal". The terminal shows the following command-line session:

```
prompt@prompt-VirtualBox: ~/PROMPT/PROMPT_examples/function_modeling/modeledBy
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/modeledBy$ pwd
/home/prompt/PROMPT/PROMPT_examples/function_modeling/modeledBy
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/modeledBy$ ./run.sh foo.bc 1>/dev/null 2>/dev/null
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/modeledBy$ ls klee-last/
assembly.ll  info  messages.txt  run.istats  run.stats  test000001.kquery  test000001.ktest  warnings.txt
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/modeledBy$ more klee-last/test000001.kquery
(query [] false)
prompt@prompt-VirtualBox:~/PROMPT/PROMPT_examples/function_modeling/modeledBy$
```

# Modeling with PROSE

- Global settings
- Data modeling
- Function modeling
- Lifecycle modeling

# Specifying sequential composition with success values

## PROSE API MODEL

```
global settings:  
data models:  
    singleton A;  
function models:  
lifecycle model:  
    foo[1] ; bar
```

## COMPONENT UNDER ANALYSIS

```
struct A {int a;};      int bar(struct A *c) {  
  
int foo(struct A *b){  
    if (b->a > 0)  
        return 0;  
    else return 1;  
}  
L1: if (c->a > 0)  
L2: assert(0);  
L3: if (c->a == -1)  
L4: assert(0);  
L5: return 0;  
}
```

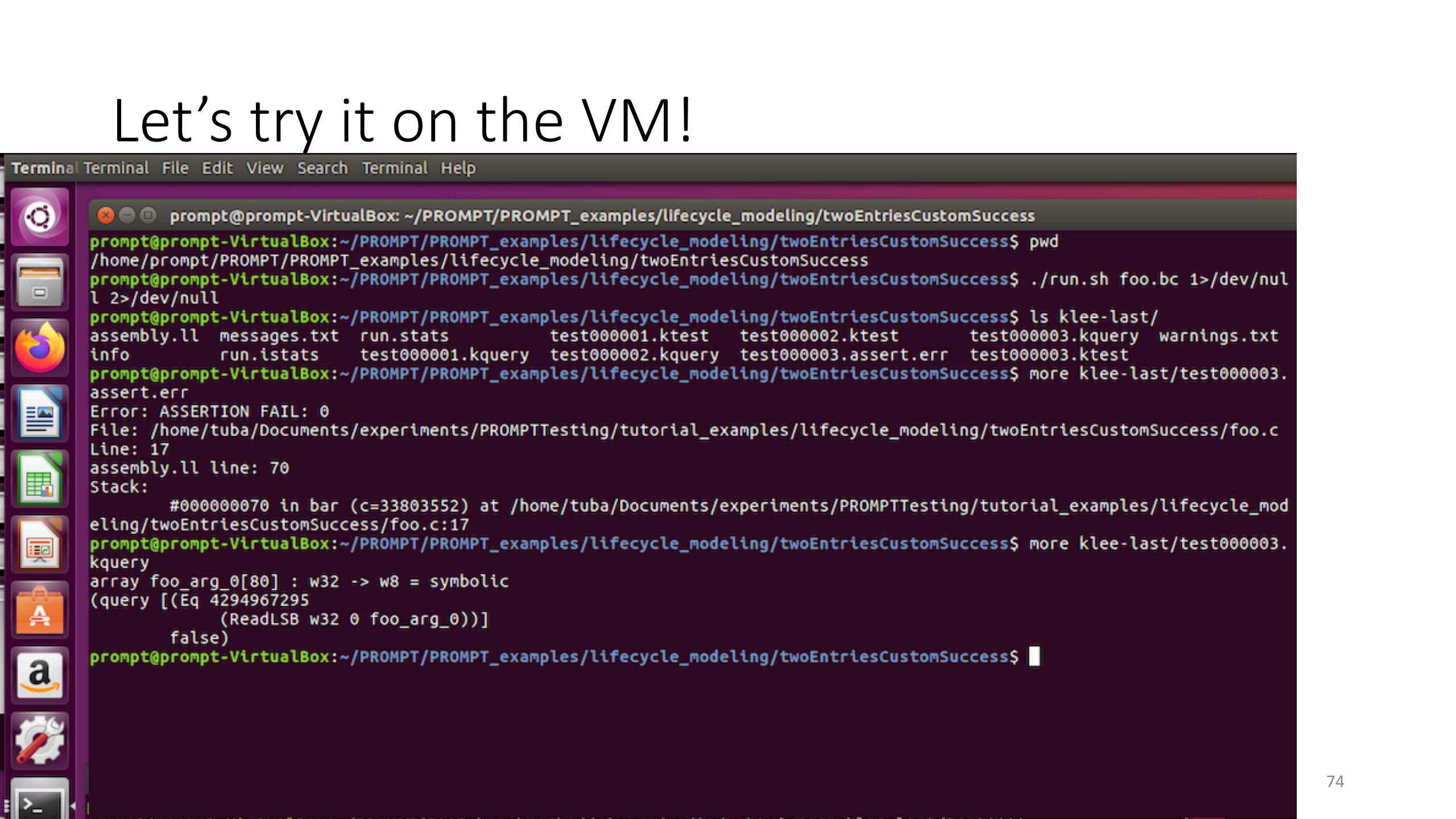
## STEPS:

```
$ export PROMPT=/home/prompt/prompt_build_dir/bin/klee  
$ cd PROMPT_examples  
$ cd lifecycle_modeling/twoEntriesCustomSuccess  
$ ./run.sh foo.bc 2>&1 | tee o.txt  
$ ls -l klee-last/
```

## PROMPT RESULTS:

```
# of paths: 3  
No assertion error at line L2  
Assertion error at line L4
```

# Let's try it on the VM!

A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window has a dark background and light-colored text. It displays a command-line session where a user is testing a program named 'foo'. The session starts with the user navigating to a directory, running a script to build or run the program, and then listing files in a folder. An error message is shown when the user runs 'more' on a specific file, indicating an assertion failure at line 17 of 'foo.c'. The terminal window is titled 'Terminal' and is part of a larger desktop interface with icons for various applications like a web browser, file manager, and terminal.

# PART III: Real-World Case Studies

# Outline for Part III

- openSSL: A Cryptography Library
  - The Heartbleed vulnerability CVE-2014-0160
- BlueZ: A Bluetooth stack
  - A BlueBorne vulnerability CVE-2017-1000251
- Linux kernel
  - A use-after-free in the usbtv driver CVE-2017-17975



# Heartbleed - CVE-2014-0160

- A memory out of bounds read error that can be exploited to read secret data from the server or from the client that is participating in a TLS session
  - High severity, leakage of secret data
  - Does not leave any trace in the logs when exploited
- Can be exploited when a specific feature of the TLS protocol is used
  - heartbeats keep connections alive
  - We have used openSSL 1.0.1c to detect the vulnerability
  - See <https://heartbleed.com/> for more info
- <https://github.com/sysrel/PROMPT/tree/master/caseStudies/openssl/heartbleed>
- Directory on the VM
  - \$ cd /home/prompt/PROMPT/caseStudies/openssl/heartbleed
- How to run
  - \$ export PROMPT=/home/prompt/prompt\_build\_dir/bin/klee
  - \$ ./run.sh 2>/dev/null 1>/dev/null
- Check the result
  - \$ more klee-last/test000005.ptr.error

## COMPONENT UNDER ANALYSIS

```
Int tls1_process_heartbeat(SSL *s) { ...  
    if (s->msg_callback)  
        s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,  
                        &s->s3->rrec.data[0], s->s3->rrec.length,  
                        s, s->msg_callback_arg);  
  
    if (hbtype == TLS1_HB_REQUEST) { ...  
        /* Allocate memory for the response, size is 1 byte  
         * message type, plus 2 bytes payload length, plus  
         * payload, plus padding  
        */  
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);  
        bp = buffer;  
  
        /* Enter response type, length and copy payload */  
        *bp++ = TLS1_HB_RESPONSE;  
        s2n(payload, bp);  
        memcpy(bp, pl, payload);  
        bp += payload;  
        /* Random padding */  
        RAND_pseudo_bytes(bp, padding);  
  
        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 +  
                            payload + padding);
```

## PROSE API MODEL

global settings:

```
array size 2; // size of pl buffer  
init funcptrs to null on;
```

data models:

function models:

lifecycle model:

```
entry-point tls1_process_heartbeat
```

# Let's try it on the VM!

```
prompt@prompt-VirtualBox: ~/PROMPT/caseStudies/openssl/heartbleed
prompt@prompt-VirtualBox:~/PROMPT/caseStudies/openssl/heartbleed$ rm -r klee-out-0/
prompt@prompt-VirtualBox:~/PROMPT/caseStudies/openssl/heartbleed$ ./run.sh 1>/dev/null 2>/dev/null
^C^C[1]+ Exit 1                  ./run.sh > /dev/null 2> /dev/null
prompt@prompt-VirtualBox:~/PROMPT/caseStudies/openssl/heartbleed$ ls -l
total 15392
lrwxrwxrwx 1 prompt prompt      61 Sep 10 23:56 klee-last -> /home/prompt/PROMPT/caseStudies/openssl/
heartbleed/klee-out-0
drwxrwxr-x 2 prompt prompt    4096 Sep 11 00:01 klee-out-0
-rw-rw-r-- 1 prompt prompt     134 Sep 10 19:18 model.txt
-rw-rw-r-- 1 prompt prompt 13532548 Sep 10 19:18 openssl_1_0_1_c_libssl_libcrypto.bc
-rw-rw-r-- 1 prompt prompt 2209071 Sep 10 19:18 o.txt
-rwxrwxr-x 1 prompt prompt     96 Sep 10 23:56 run.sh
prompt@prompt-VirtualBox:~/PROMPT/caseStudies/openssl/heartbleed$ ls -l klee-out-0/
total 77640
f-rw-rw-r-- 1 prompt prompt 68504993 Sep 10 23:56 assembly.ll
-rw-rw-r-- 1 prompt prompt     165 Sep 10 23:57 info
t-rw-rw-r-- 1 prompt prompt     943 Sep 11 00:01 messages.txt
-rw-rw-r-- 1 prompt prompt 10492831 Sep 11 00:04 run.istats
-rw-rw-r-- 1 prompt prompt   21123 Sep 11 00:04 run.stats
-rw-rw-r-- 1 prompt prompt    40561 Sep 10 23:57 test000001.ktest
-rw-rw-r-- 1 prompt prompt    40561 Sep 10 23:57 test000002.ktest
-rw-rw-r-- 1 prompt prompt    40561 Sep 10 23:57 test000003.ktest
-rw-rw-r-- 1 prompt prompt    40561 Sep 10 23:57 test000004.ktest
-rw-rw-r-- 1 prompt prompt     493 Sep 10 23:58 test000005.kquery
-rw-rw-r-- 1 prompt prompt    40561 Sep 10 23:58 test000005.ktest
-rw-rw-r-- 1 prompt prompt     629 Sep 10 23:58 test000005.ptr.err
-rw-rw-r-- 1 prompt prompt     529 Sep 10 23:58 test000006.kquery
```

## PROMPT RESULTS:



```
Error: memory error: out of bound pointer
File: /home/tuba/Documents/releases/PROMPT/runtime/Intrinsic/memcpy.c
Line: 17
assembly.ll line: 383044
Stack:
    #000383044 in memcpy (destaddr=330906419, srcaddr=277348915, len) at
/home/tuba/Documents/releases/PROMPT/runt
ime/Intrinsic/memcpy.c:17
    #100353745 in tls1_process_heartbeat (s=330210224) at
/home/tuba/Documents/tools/OPENSSL/openssl-OpenSSL_1_0_1
c/ssl/t1_lib.c:2469
Info:
    address: 277348915
    next: object at 323117216 of size 13
        M050[13] allocated at global:.str.4.2451
    prev: object at 277348912 of size 16
        M08691[16] allocated at tls1_process_heartbeat(): %4 = load i8*,
i8** %3, align 8, !dbg !250977, !tba
a !250978
```

PROMPT detects the heartbleed vulnerability within a minute!



# BlueBorne CVE-2017-1000251

- Bluetooth is a very popular wireless short range communication protocol
- One of the vulnerabilities dubbed as BlueBorne is in BlueZ
  - BlueZ is the Bluetooth stack used in the Linux kernel
  - The vulnerability can be exploited to perform remote code execution when the extended flow specification (EFS) feature is utilized in L2CAP
- [https://github.com/sysrel/PROMPT/tree/master/JASE\\_benchmarks/bluez/l2cap\\_config\\_rsp](https://github.com/sysrel/PROMPT/tree/master/JASE_benchmarks/bluez/l2cap_config_rsp)
- Directory on the VM
  - \$ cd /home/prompt/PROMPT/JASE\_benchmarks/bluez/l2cap\_config\_rsp
- How to run
  - \$ export PROMPT=/home/prompt/prompt\_build\_dir/bin/klee
  - \$ ./run.sh 2>/dev/null 1>/dev/null
- Check the result
  - \$ more klee-last/test000069.ptr.error

# BlueBorne CVE-2017-1000251 in a nutshell

```
static inline int l2cap_config_rsp(struct l2cap_conn *conn,
struct l2cap cmd_hdr *cmd, u8 *data)
{
    ...
    switch (result) {
        ...
        case L2CAP_CONF_PENDING:
            set_bit(CONF REM CONF PEND, &chan->conf state);
            if (test_bit(CONF LOC CONF PEND, &chan->conf state)) {
                char buf[64]; // the buffer involved in the overflow
                len = l2cap_parse_conf_rsp(chan, rsp->data,
                    len, buf, &result);
            }
    }

    static int l2cap_parse_conf_rsp(struct l2cap chan *chan,
        void *rsp, int len, void *data, u16 *result)
    {
        struct l2cap conf req *req = data;
        void *ptr = req->data; // points into the buffer
        while (len >= L2CAP_CONF_OPT_SIZE) { // len is not limited
            len -= l2cap_get_conf_opt(&rsp, &type ...);
            switch (type) {
                case L2CAP_CONF_MTU:
                    l2cap_add_conf_opt(&ptr, L2CAP_CONF_MTU, 2, chan->imtu);
                    break;
            }
        }
    }
}
```

// No checks on the buffer size!

```
static void l2cap_add_conf_opt(void **ptr,
    u8 type, u8 len, unsigned long val)
{
    struct l2cap conf opt *opt = *ptr;
    ...// write to the buffer
    *ptr += L2CAP_CONF_OPT_SIZE + len;
}
```

global settings:

array size 128;

model funcs with asm off;

symbolize inline asm on;

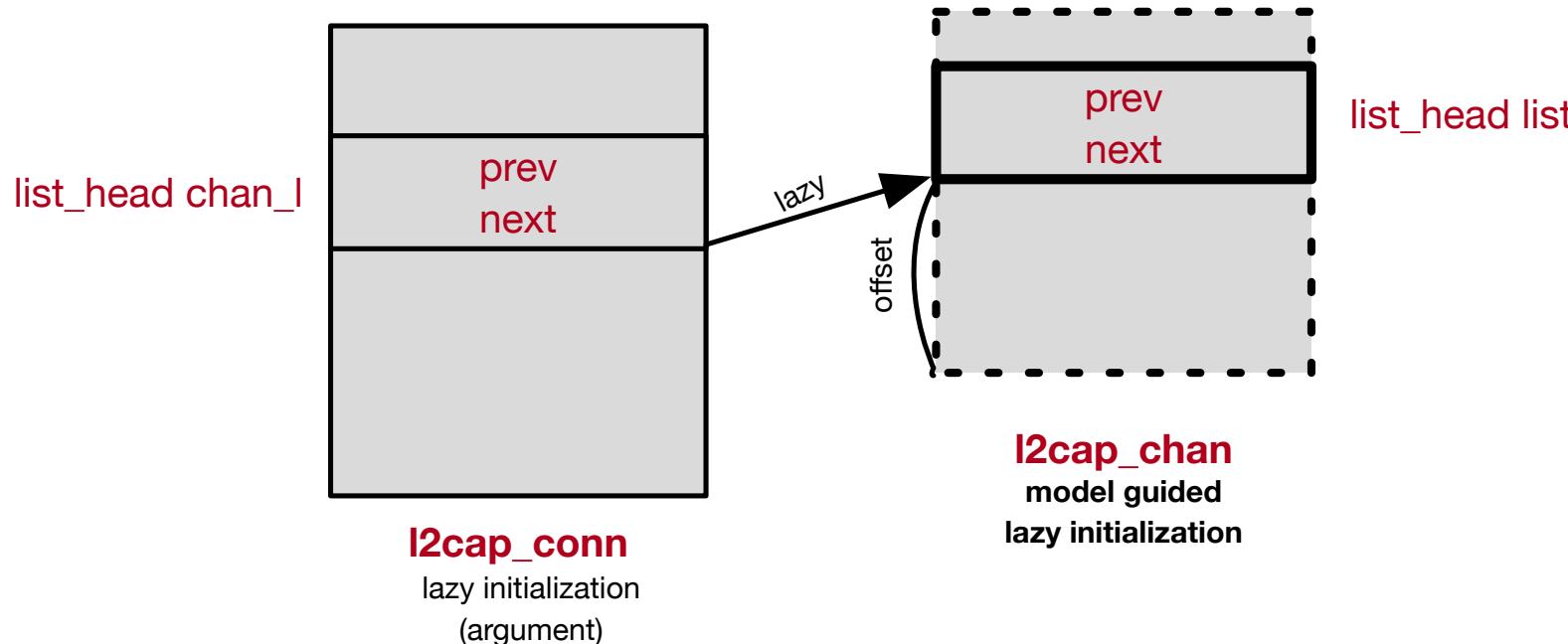
data models: ...

function models: ...

lifecycle model: ...

```
static inline int l2cap_config_rsp(  
    struct l2cap_conn *conn,  
    struct l2cap_cmd_hdr *cmd,  
    u8 *data)  
{  
    ...  
    switch (result) { ...  
        case L2CAP_CONF_PENDING:  
            char buf[64]; // the buffer  
            involved in the overflow  
            len =  
l2cap_parse_conf_rsp(chan, rsp->data,  
                len, buf, &result);  
    ...  
}
```

# Pointer arithmetic using a negative offset



Just tell PROMPT that  
type I2cap\_chan embeds type list\_head

## COMPONENT UNDER ANALYSIS

```
static inline int l2cap_config_rsp(  
    struct l2cap_conn *conn,  
    struct l2cap_cmd_hdr *cmd,  
    u8 *data)  
{  
    ...  
    chan = l2cap_get_chan_by_scid(conn, scid);  
    if (!chan) return 0;  
    switch (result) { ...  
        case L2CAP_CONF_PENDING:  
            char buf[64]; // the buffer  
involved in the overflow  
            len =  
l2cap_parse_conf_rsp(chan, rsp-  
>data,  
len, buf, &result);  
    ...  
}
```

data models:

type l2cap\_chan embeds type list\_head;

(x = l2cap\_sock\_state\_change\_cb) where  
l2cap\_sock\_state\_change\_cb is function,  
x is l2cap\_ops field 5;

(x = l2cap\_sock\_ready\_cb) where  
l2cap\_sock\_ready\_cb is function,  
x is l2cap\_ops field 6;

(x = l2cap\_sock\_suspend\_cb) where  
l2cap\_sock\_suspend\_cb is function,  
x is l2cap\_ops field 9;

```
static inline int l2cap_config_rsp(
    struct l2cap_conn *conn,
    struct l2cap_cmd_hdr *cmd,
    u8 *data)
{
    ...
    chan = l2cap_get_chan_by_scid(conn, scid);
    if (!chan) return 0;
    switch (result) { ...
        case L2CAP_CONF_PENDING:
            char buf[64]; // the buffer
    involved in the overflow
    len =
l2cap_parse_conf_rsp(chan, rsp-
>data,
len, buf, &result);
    ...
}
```

function models:

```

    alloc usb_alloc_coherent sizearg 1 initzero true
symbolize false;
    alloc __kmalloc sizearg 0 initzero true
symbolize false;
    alloc vzalloc sizearg 0 initzero true symbolize
false;
    free kfree memarg 0;
    free vfree memarg 0;
```

```

returnonly skb_clone;
returnonly kfree_skb;
returnonly l2cap_send_cmd;
returnonly hci_send_cmd;
returnonly l2cap_clear_timer;
```

...

lifecycle model:

```
entry-point l2cap_config_rsp
```

```

static inline int l2cap_config_rsp
    struct l2cap_conn *conn,
    struct l2cap_cmd_hdr *cmd,
    u8 *data)
{
    ...
    chan = l2cap_get_chan_by_scid(conn, scid);
    if (!chan) return 0;
    switch (result) {
        case L2CAP_CONF_PENDING:
            char buf[64];// the buffer
involved in the overflow
            len =
l2cap_parse_conf_rsp(chan, rsp-
>data,
len, buf, &result);
            ...
    }
}
```

```

int l2cap_get_conf_opt_PROSE(void **ptr, int *type, int
*olen,
                                unsigned long *val)
{
    struct l2cap_conf_opt *opt = *ptr;
    int len;

    opt->type = 1;      // L2CAP_CONF_MTU
    opt->len = 2;

    len = L2CAP_CONF_OPT_SIZE + opt->len;
    *ptr += len;

    *type = opt->type;
    *olen = opt->len;
    return len;
}

```

```

static inline int l2cap_config_rsp(
    struct l2cap_conn *conn,
    struct l2cap_cmd_hdr *cmd,
    u8 *data)
{
    ...
    chan = l2cap_get_chan_by_scid(conn, scid);
    if (!chan) return 0;
    switch (result) {
        case L2CAP_CONF_PENDING:
            char buf[64];// the buffer
involved in the overflow
            len =
l2cap_parse_conf_rsp(chan, rsp-
>data,
                                len, buf, &result);
    ...
}

```

# Let's try it on the VM!

```
prompt@prompt-VirtualBox: ~/PROMPT/JASE_benchmarks/bluez/l2cap_config_rsp
51
#200047685 in l2cap_config_rsp (conn=130669200, cmd=130635200, cmd_len, data=130652544) at /home/tuba/Documents/tools/clang-kernel-build/linux-stable/net/bluetooth/l2cap_core.c:4186
Info:
    address: 130859232
    next: object at 130860592 of size 8
        M0662[8] allocated at l2cap_parse_conf_rsp(): %1 = alloca i8*, align 8
    prev: object at 130859168 of size 64
        M0615[64] allocated at l2cap_config_rsp(): %buf = alloca [64 x i8], align 16
prompt@prompt-VirtualBox:~/PROMPT/JASE_benchmarks/bluez/l2cap_config_rsp$ ls klee-last/
assembly.ll      test000013.ktest  test000030.ktest  test000047.ktest  test000064.ktest
info              test000014.ktest  test000031.ktest  test000048.ktest  test000065.ktest
messages.txt      test000015.ktest  test000032.ktest  test000049.ktest  test000066.ktest
run.istats        test000016.ktest  test000033.ktest  test000050.ktest  test000067.ktest
run.stats         test000017.ktest  test000034.ktest  test000051.ktest  test000068.ktest
test000001.ktest  test000018.ktest  test000035.ktest  test000052.ktest  test000069.kquery
test000002.ktest  test000019.ktest  test000036.ktest  test000053.ktest  test000069.ktest
test000003.ktest  test000020.ktest  test000037.ktest  test000054.ktest  test000069.ptr.err
test000004.ktest  test000021.ktest  test000038.ktest  test000055.ktest  test000070.ktest
test000005.ktest  test000022.ktest  test000039.ktest  test000056.ktest  test000071.ktest
test000006.ktest  test000023.ktest  test000040.ktest  test000057.ktest  test000072.ktest
test000007.ktest  test000024.ktest  test000041.ktest  test000058.ktest  test000073.ktest
test000008.ktest  test000025.ktest  test000042.ktest  test000059.ktest  test000074.ktest
test000009.ktest  test000026.ktest  test000043.ktest  test000060.ktest  test000075.ktest
test000010.ktest  test000027.ktest  test000044.ktest  test000061.ktest  warnings.txt
test000011.ktest  test000028.ktest  test000045.ktest  test000062.ktest
test000012.ktest  test000029.ktest  test000046.ktest  test000063.ktest
prompt@prompt-VirtualBox:~/PROMPT/JASE_benchmarks/bluez/l2cap_config_rsp$
```

## PROMPT RESULTS:



BlueBorne

```
Error: memory error: out of bound pointer
      File: /home/tuba/Documents/tools/clang-kernel-build/linux-
stable/net/bluetooth/l2cap_core.c
      Line: 2996
      assembly.ll line: 43986
      Stack:
          #000043986 in l2cap_add_conf_opt (ptr=142165200, type=1, len=2,
val=43947) at /home/
              tuba/Documents/tools/clang-kernel-build/linux-
stable/net/bluetooth/l2cap_core.c:2996
          #100048020 in l2cap_parse_conf_rsp (chan=92475136,
rsp=141640710, len, data=14184776
          0, result=141548608) at /home/tuba/Documents/tools/clang-kernel-
build/linux-stable/net/bluet
ooth/l2cap_core.c:3551
          #200047685 in l2cap_config_rsp (conn=141657120, cmd=141631360,
cmd_len, data=1416407
          04) at /home/tuba/Documents/tools/clang-kernel-build/linux-
stable/net/bluetooth/l2cap_core.c
          :4186
```

PROMPT detects  
the BlueBorne  
vulnerability  
within 5  
minutes!

# A use-after-free in the usbtv driver CVE-2017-17975

- Implicit updates to the reference counts cause a use-after free and a double free
- Was detected by the MOXCAFE tool
  - Tuba Yavuz. "Detecting Callback Related Deep Vulnerabilities in Linux Device Drivers". SecDev 2019 .
- Required modeling of the registration and deregistration API for the video subsystem
  - a precise modeling of the reference counts and the clean-up callbacks
- [https://github.com/sysrel/PROMPT/tree/master/JASE\\_benchmarks/drivers/media/usbtv](https://github.com/sysrel/PROMPT/tree/master/JASE_benchmarks/drivers/media/usbtv)
- Directory on the VM
  - \$ cd /home/prompt/PROMPT/JASE\_benchmarks/drivers/media/usbtv
- How to run
  - \$ export PROMPT=/home/prompt/prompt\_build\_dir/bin/klee
  - \$ ./run.sh 2>/dev/null 1>/dev/null
- Check the result
  - \$ more klee-last/test000009.ptr.error

# A use-after-free CVE-2017-17975 in a nutshell

```
static int usbtv_probe(struct usb_interface *intf,
                      const struct usb_device_id *id)
{
    ...
    ret = usbtv_video_init(usbtv);
    if (ret < 0)
        goto usbtv_video_fail;

    ret = usbtv_audio_init(usbtv);
    if (ret < 0)
        goto usbtv_audio_fail;

usbtv_audio_fail:
    usbtv_video_free(usbtv); // first free

usbtv_video_fail:
    usb_set_intfdata(intf, NULL);
    usb_put_dev(usbtv->udev); // use-after-free
    kfree(usbtv); // double-free

    return ret;
}
```

global settings:

model funcs with asm off;

data models:

singleton usb\_interface;  
 singleton vb2\_queue;  
 singleton v4l2\_device;  
 singleton usb\_device;  
 singleton usbtv;

function models:

alloc usb\_alloc\_coherent sizearg 1 initzero true symbolize  
 false;

alloc snd\_card\_new initzero true symbolize true memreturn  
 false destarg 5;

alloc snd\_pcm\_new initzero true symbolize true memreturn  
 false destarg 5;

alloc \_\_kmalloc sizearg 0 initzero true symbolize false;  
 free kfree memarg 0;

returnonly dev\_err;  
 returnonly \_dev\_info ...

```
static int usbtv_probe(struct
usb_interface *intf,
const struct usb_device_id *id)
{
...
ret = usbtv_video_init(usbtv);
if (ret < 0)
    goto usbtv_video_fail;
ret = usbtv_audio_init(usbtv);
if (ret < 0)
    goto usbtv_audio_fail;
usbtv_audio_fail:
    usbtv_video_free(usbtv);
usbtv_video_fail:
    usb_set_intfdata(intf, NULL);
    usb_put_dev(usbtv->udev);
    kfree(usbtv);
    return ret;
}
```

## PROSE API MODEL

```
function models: ..  
    __video_register_device modeled by __video_register_device_PROSE;  
  
    video_unregister_device modeled by video_unregister_device_PROSE;  
  
    v4l2_device_register modeled by v4l2_device_register_PROSE;  
  
    v4l2_device_unregister modeled by v4l2_device_unregister_PROSE;  
  
    v4l2_device_disconnect modeled by v4l2_device_disconnect_PROSE;  
  
    v4l2_device_put modeled by v4l2_device_put_PROSE;  
    v4l2_device_get modeled by v4l2_device_get_PROSE;  
    atomic_add_return modeled by atomic_add_return_PROSE;  
    atomic_dec modeled by atomic_dec_PROSE;  
    atomic_inc modeled by atomic_inc_PROSE;  
    v4l2_i2c_subdev_init modeled by v4l2_i2c_subdev_init_PROSE;  
  
lifecycle model:  
  
usbtv_probe[0] ; usbtv_disconnect
```

## COMPONENT UNDER ANALYSIS

```
static int usbtv_probe(struct  
usb_interface *intf,  
const struct usb_device_id *id)  
{ ...  
    ret = usbtv_video_init(usbtv);  
    if (ret < 0)  
        goto usbtv_video_fail;  
    ret = usbtv_audio_init(usbtv);  
    if (ret < 0)  
        goto usbtv_audio_fail;  
    usbtv_audio_fail:  
        usbtv_video_free(usbtv);  
    usbtv_video_fail:  
        usb_set_intfdata(intf, NULL);  
        usb_put_dev(usbtv->udev);  
        kfree(usbtv);  
    return ret;  
}
```

# Let's try it on the VM!

```
prompt@prompt-VirtualBox: ~/PROMPT/JASE_benchmarks/drivers/media/usbtv
prompt@prompt-VirtualBox:~/PROMPT/JASE_benchmarks/drivers/media/usbtv$ more klee-last/test0000
test00001.ktest  test00008.ktest  test00013.ktest  test00020.ktest  test00027.ktest
test00002.ktest  test00009.kquery  test00014.ktest  test00021.ktest  test00028.ktest
test00003.ktest  test00009.ktest  test00015.ktest  test00022.ktest  test00029.ktest
test00004.ktest  test00009.ptr.err  test00016.ktest  test00023.ktest  test00030.ktest
test00005.ktest  test00010.ktest  test00017.ktest  test00024.ktest  test00031.ktest
test00006.ktest  test00011.ktest  test00018.ktest  test00025.ktest
test00007.ktest  test00012.ktest  test00019.ktest  test00026.ktest
prompt@prompt-VirtualBox:~/PROMPT/JASE_benchmarks/drivers/media/usbtv$ more klee-last/test00009.ptr.err
Error: memory error: out of bound pointer
Stack:
#000000866 in usbtv_probe (intf=50332320, id=50305648)
Info:
address: 50389400
next: object at 50393088 of size 800
    M086[800] allocated at usbtv_probe(): %20 = load %struct.usb_host_interface*, %struct.usb_host_interface** %19, align 8
prev: object at 50362832 of size 4
    M0282[4] allocated at usbtv_video_init(): %57 = call i32 @vb2_queue_init(%struct.vb2_queue*
%56) #13
prompt@prompt-VirtualBox:~/PROMPT/JASE_benchmarks/drivers/media/usbtv$
```

Error: memory error: out of bound pointer

Stack:

#000000866 in usbtv\_probe (intf=43920800, id=43943728)

Info:

address: 44033848

```
; <label>:79 ; preds = %50, %78
%80 = load %struct.usb_interface*, %struct.usb_interface** %2, align 8
call void @usb_set_intfdata(%struct.usb_interface* %80, i8* null) #13
%81 = load %struct.usbtv*, %struct.usbtv** %usbtv, align 8
%82 = getelementptr inbounds %struct.usbtv, %struct.usbtv* %81, i32 0, i32
%83 = load %struct.usb_device*, %struct.usb_device** %82, align 8
call void @usb_put_dev(%struct.usb_device* %83) #13
```

klee-last/assembly.ll

PROMPT detects  
the use-after  
vulnerability  
within 1 minute!

usbtv\_video\_fail:

```
usb_set_intfdata(intf, NULL);
usb_put_dev(usbtv->udev); // use-after-free
kfree(usbtv); // double-free
```

return ret;

}

# Applying PROMPT to your component

- Install PROMPT on your high-end computer
  - The virtual machine instance is just for getting started!
- Generate the bitcode for your project
  - Check out the wllvm tool at <https://github.com/travitch/whole-program-llvm>
  - You may need to link the bitcode generated for individual libraries using llvm-link
- Start with a simple model
  - Entry-point function\_to\_analyze
- Expand the model until you achieve your goal
  - Coverage
  - Reaching certain program locations
  - Finding bugs!
- Happy modeling and bug hunting!

# Acknowledgements

- This work has been partially funded by
  - National Science Foundation (NSF) awards CNS-1815883 and CNS-1942235
  - Semiconductor Research Corporation (SRC)
- Thanks to Joshua Nelson for working on the parser as an undergraduate researcher
- If you use PROMPT for your work, please cite our paper
  - Tuba Yavuz and Ken (Yihang) Bai. *Analyzing System Software Components using API Model Guided Symbolic Execution*. To appear in Automated Software Engineering (27:6).DOI: 10.1007/s10515-020-00276-5

Thank you - Questions?