The StackwalkerAPI

A Cross-Platform API for Collecting Accurate Stackwalks

Dyninst

UNIVERSITY OF MARYLAND 18 56

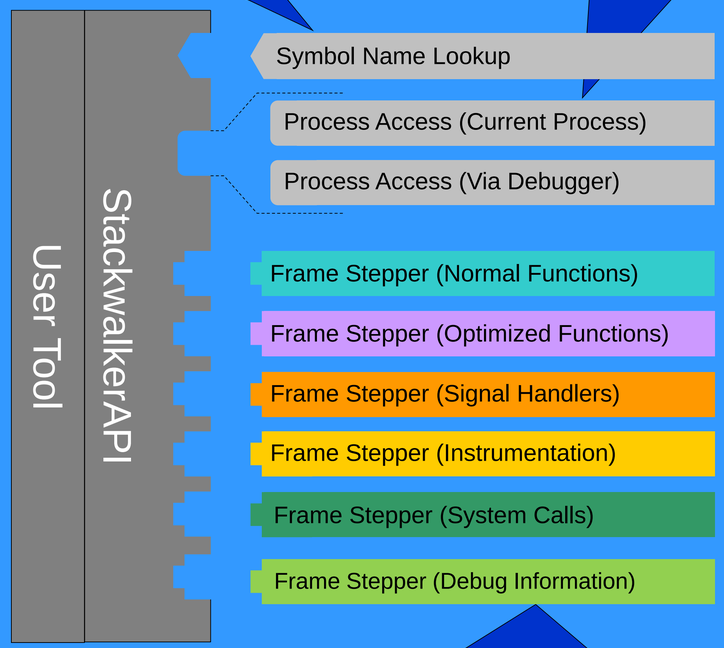
WISCONSIN — UNIVERSITY OF WISCONSIN–MADISON

Customizable Plug-in Interface

- Plug-ins allow StackwalkerAPI to be integrated easily into other tools.
- Customizes how StackwalkerAPI looks up symbol names, accesses a process, or walks through types of stack frames.

- Customizes how symbol names are looked up for each stack frame.
- Default uses SymtabAPI

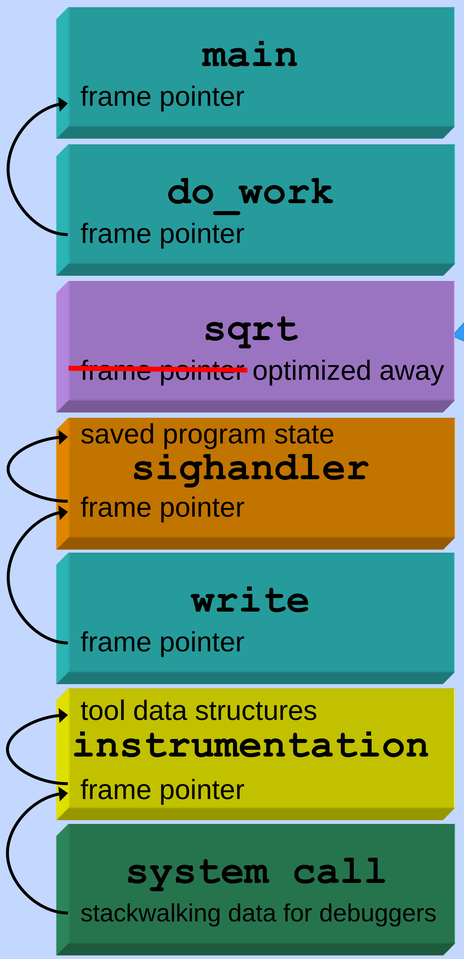
- Customizes how StackwalkerAPI reads from a process.
- Defaults use ProcControlAPI or read from current process.



Frame Steppers:
- Describe how to walk through a type of frame.
- Find address ranges of code that this Frame Stepper can be used over.

The StackwalkerAPI

- Is a cross-platform library for walking call stacks.
- Works through a debugger interface or in its own process.
- Is customizable and extensible, easily integrating into pre-existing tools.



Example User Tool

```
std::vector<Frame> stackwalk;
string s;
Walker *walker = Walker::newWalker();
walker->walkStack(stackwalk);
for (int i=0; i<stackwalk.size(); i++) {
  stackwalk[i].getFuncName(s);
  cout << "Function " << s << endl;
}
```

Handling Optimized Functions

Some Functions may not set up a stack frame.

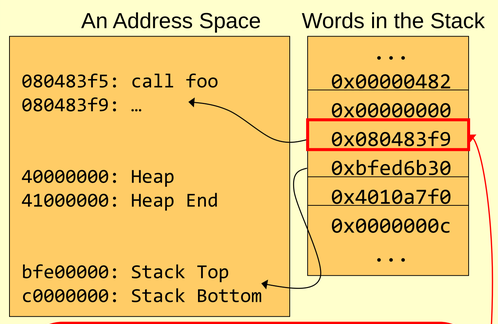
Solution 1: Static analysis

- Analyze the function to understand how the stack changes as it executes.
- For each instruction, determine the distance, Δ, from the top of the current frame to the bottom of the current frame.

```
Δ=0
Δ=4   push %eax
Δ=8   push %edx
Δ=28  sub  $0x20, %esp
Δ=28  cmp  %ebx, $0
Δ=28  je   ...

Δ=28  mov  $8, %ecx
Δ=28  add  %ecx, %eax
Δ=28  inc  *(%eax)
Δ=28  jmp  ...

Δ=32  push $4
Δ=36  push $8
Δ=36  call foo
Δ=28  add  $8,%esp

Δ=24  pop %edx
Δ=20  pop %eax
Δ=0   add $20, %esp
Δ=0   ret
```

Solution 2: Stack Value Inspection

- For each word in the stack, use a heuristic to determine if it could be a return address that was generated by a call instruction.
- Useful if there is no other way to walk through a stack frame, but prone to false positives.



An address is likely the top of a frame if …

- ✓ … it points to an instruction that follows a call, and
- ✓ … the next word in the stack points into the stack.

Version 8.0 available on Linux-x86/x86_64/PPC, BlueGene, Windows, Cray

http://www.paradyn.org