Project title

Overview

Binaries ↩

old: an_old_alias

new: a_new_alias

Statistics ↩

Program Storage

	Old/ bytes	New/ bytes	Delta/ bytes
overall	329	331	+2
text	321	323	+2
data	8	8	0

Static RAM

	Old/ bytes	New/ bytes	Delta/ bytes
overall	8	8	0
data	8	8	0
bss	0	0	0

Legend

text	instructions
data	initialized global or static variables
bss	uninitialized global or static variables

Symbols

7 symbols found in an_old_alias

7 symbols found in a_new_alias

Symbols ↩

Persisting Symbols ↩

Symbol ▲▼	Type ▲▼	Old Size/ bytes ▲▼	New Size/ bytes ▲▼	Delta/ bytes ▲▼
persisting1(int)	T	18	18	0
persisting2(int)	T	18	18	0
var	D	4	4	0

Columns

Symbol	The symbol name (possibly mangled)
Type	The symbol type (see the documentation of binutils tool nm for more information)
Old Size	The old symbol size either in RAM or program memory
New Size	The new symbol size either in RAM or program memory
Delta	The change to symbol size

Disappeared Symbols ↩

Symbol ▲▼	Type ▲▼	Size/ bytes ▲▼
Test::g(float, float)	T	29
Test::f(int, int)	T	21
func(int)	T	18
Test::m_	D	4

Columns

Symbol	The symbol name (possibly mangled)
Type	The symbol type (see the documentation of binutils tool nm for more information)
Size	The symbol size either in RAM or program memory

New Symbols ↩

Symbol ▲▼	Type ▲▼	Size/ bytes ▲▼
Test1::g(float, float)	T	29
Test1::f(int, int)	T	21
func(double)	T	20
Test1::m_	D	4

Columns

Symbol	The symbol name (possibly mangled)
Type	The symbol type (see the documentation of binutils tool nm for more information)
Size	The symbol size either in RAM or program memory

Similar Symbols ↩

Id ▲▼	Symbols ▲▼	Types ▲▼	Size/ bytes ▲▼	Delta/ bytes ▲▼	Sig. Sim./ % ▲▼	Instr. Sim./ % ▲▼
0	Test::g(float, float) Test1::g(float, float)	T T	29 29	+0	97.7	100.0
1	Test::f(int, int) Test1::f(int, int)	T T	21 21	+0	97.1	100.0
2	Test::m_ Test1::m_	D D	4 4	+0	94.1	100.0
3	Test::f(int, int) Test1::g(float, float)	T T	21 29	+8	61.5	65.7
4	Test::g(float, float) Test1::f(int, int)	T T	29 21	-8	61.5	65.7
5	func(int) func(double)	T T	18 20	+2	57.1	92.2

Columns

ID	Integer id assigned to each symbol pair
Symbols	The two similar symbol names (possibly mangled)
Types	The symbol types (see the documentation of binutils tool nm for more information)
Size	The sizes of the symbols either in RAM or program memory
Delta	The difference in symbol size
Sig. Sim.	Lexicographic symbol signature similarity
Instr. Sim.	Instruction similarity of the symbols' assembly code

Binary Details ↩

an_old_alias ↩

```
Info about the old binary
```

a_new_alias ↩

```
Info about the new binary
```

Build Info

```
Build info. More build info.
```

Symbol Details ↩

Persisting Symbols ↩

Persisting symbol persisting1(int) : old size: 18 bytes, new size: 18 bytes, delta: 0 bytes

	Old		New
f	1 endbr64	f	1 endbr64
	2 push %rbp		2 push %rbp

	Old			New	
	3	mov %rsp,%rbp		3	mov %rsp,%rbp
	4	mov %edi,-0x4(%rbp)		4	mov %edi,-0x4(%rbp)
t	5	mov $0x2b,%eax	t	5	mov $0x2a,%eax
	6	pop %rbp		6	pop %rbp
	7	retq		7	retq

Persisting symbol persisting2(int) : old size: 18 bytes, new size: 18 bytes, delta: 0 bytes

	Old			New	
f	1	endbr64	f	1	endbr64
	2	push %rbp		2	push %rbp
	3	mov %rsp,%rbp		3	mov %rsp,%rbp
	4	mov %edi,-0x4(%rbp)		4	mov %edi,-0x4(%rbp)
t	5	mov $0x2b,%eax	t	5	mov $0x2a,%eax
	6	pop %rbp		6	pop %rbp
	7	retq		7	retq

Disappeared Symbols ↩

Disappeared symbol Test::g(float, float) : size: 29 bytes

```
endbr64
push   %rbp
mov    %rsp,%rbp
mov    %rdi,-0x8(%rbp)
movss  %xmm0,-0xc(%rbp)
movss  %xmm1,-0x10(%rbp)
mov    $0x1,%eax
pop    %rbp
retq
```

Disappeared symbol Test::f(int, int) : size: 21 bytes

```
endbr64
push   %rbp
mov    %rsp,%rbp
mov    %edi,-0x4(%rbp)
mov    %esi,-0x8(%rbp)
mov    $0x2a,%eax
```

```
pop    %rbp
retq
nop
```

Disappeared symbol func(int) : size: 18 bytes

```
endbr64
push   %rbp
mov    %rsp,%rbp
mov    %edi,-0x4(%rbp)
mov    $0x2a,%eax
pop    %rbp
retq
```

New Symbols ↩

New symbol Test1::g(float, float) : size: 29 bytes

```
endbr64
push   %rbp
mov    %rsp,%rbp
mov    %rdi,-0x8(%rbp)
movss  %xmm0,-0xc(%rbp)
movss  %xmm1,-0x10(%rbp)
mov    $0x1,%eax
pop    %rbp
retq
```

New symbol Test1::f(int, int) : size: 21 bytes

```
endbr64
push   %rbp
mov    %rsp,%rbp
mov    %edi,-0x4(%rbp)
mov    %esi,-0x8(%rbp)
mov    $0x2a,%eax
pop    %rbp
retq
nop
```

New symbol func(double) : size: 20 bytes

```
endbr64
push   %rbp
mov    %rsp,%rbp
movsd  %xmm0,-0x8(%rbp)
mov    $0x2a,%eax
pop    %rbp
retq
```

Similar Symbols ↩

Similar pair 0 : old size: 29 bytes, new size: 29 bytes, delta: +0 bytes, sig. sim.: 97.7 %, instr. sim.: 100.0 %

Old: Test::g(float, float)
New: Test1::g(float, float)

	Old		New
t	No Differences Found	t	No Differences Found

Similar pair 1 : old size: 21 bytes, new size: 21 bytes, delta: +0 bytes, sig. sim.: 97.1 %, instr. sim.: 100.0 %

Old: Test::f(int, int)
New: Test1::f(int, int)

	Old		New
t	No Differences Found	t	No Differences Found

Similar pair 3 : old size: 21 bytes, new size: 29 bytes, delta: +8 bytes, sig. sim.: 61.5 %, instr. sim.: 65.7 %

Old: Test::f(int, int)
New: Test1::g(float, float)

	Old		New
f	1 endbr64	f	1 endbr64
	2 push %rbp		2 push %rbp
	3 mov %rsp,%rbp		3 mov %rsp,%rbp
n	4 mov %edi,-0x4(%rbp)	n	4 mov %rdi,-0x8(%rbp)
	5 mov %esi,-0x8(%rbp)		5 movss %xmm0,-0xc(%rbp)
			6 movss %xmm1,-0x10(%rbp)
	6 mov $0x2a,%eax		7 mov $0x1,%eax
	7 pop %rbp		8 pop %rbp
	8 retq		9 retq
t	9 nop	t	

Similar pair 4 : old size: 29 bytes, new size: 21 bytes, delta: -8 bytes, sig. sim.: 61.5 %, instr. sim.: 65.7 %

Old: Test::g(float, float)
New: Test1::f(int, int)

	Old				New		
f	1	endbr64		f	1	endbr64	
	2	push	%rbp		2	push	%rbp
	3	mov	%rsp,%rbp		3	mov	%rsp,%rbp
n	4	mov	%rdi,-0x8(%rbp)	n	4	mov	%edi,-0x4(%rbp)
	5	movss	%xmm0,-0xc(%rbp)		5	mov	%esi,-0x8(%rbp)
	6	movss	%xmm1,-0x10(%rbp)				
	7	mov	$0x1,%eax		6	mov	$0x2a,%eax
	8	pop	%rbp		7	pop	%rbp
	9	retq			8	retq	
t				t	9	nop	

Similar pair 5 : old size: 18 bytes, new size: 20 bytes, delta: +2 bytes, sig. sim.: 57.1 %, instr. sim.: 92.2 %

Old: func(int)
New: func(double)

	Old				New		
f	1	endbr64		f	1	endbr64	
	2	push	%rbp		2	push	%rbp
	3	mov	%rsp,%rbp		3	mov	%rsp,%rbp
t	4	mov	%edi,-0x4(%rbp)	t	4	movsd	%xmm0,-0x8(%rbp)
	5	mov	$0x2a,%eax		5	mov	$0x2a,%eax
	6	pop	%rbp		6	pop	%rbp
	7	retq			7	retq	

Using sortable tables from kryogenix.org