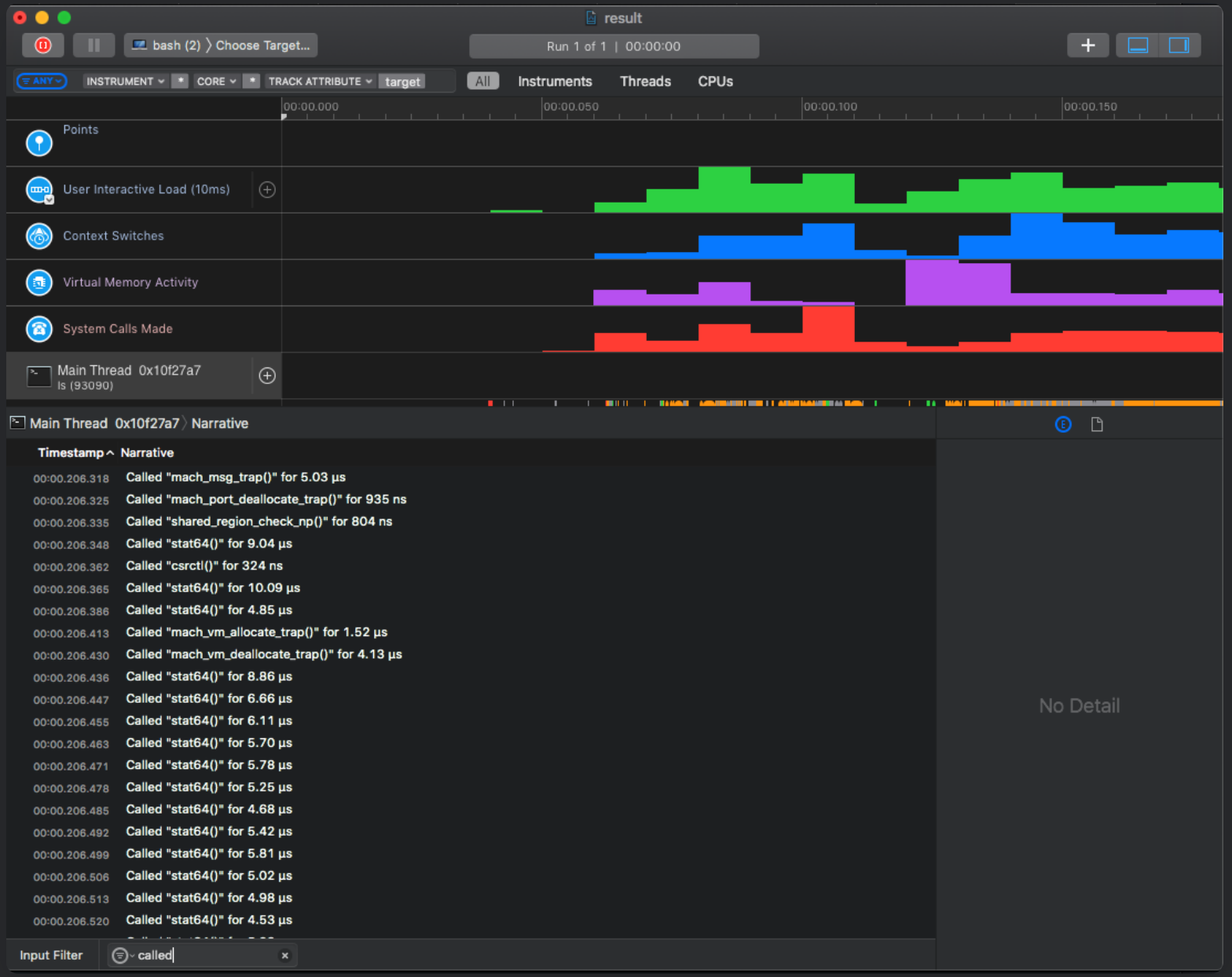
How To Trace Your System Calls On Mac OS

2019-01-10


result
bash (2) 〉Choose Target...
Run 1 of 1 | 00:00:00
INSTRUMENT
CORE
TRACK ATTRIBUTE
target
All
Instruments
Threads
CPUs
00:00.000
00:00.050
00:00.100
00:00.150
Points
User Interactive Load (10ms)
Context Switches
Virtual Memory Activity
System Calls Made
Main Thread 0x10f27a7
ls (93090)
Main Thread 0x10f27a7 〉Narrative
Timestamp
Narrative
00:00.206.318 Called "mach_msg_trap()" for 5.03 µs
00:00.206.325 Called "mach_port_deallocate_trap()" for 935 ns
00:00.206.335 Called "shared_region_check_np()" for 804 ns
00:00.206.348 Called "stat64()" for 9.04 µs
00:00.206.362 Called "csrctl()" for 324 ns
00:00.206.365 Called "stat64()" for 10.09 µs
00:00.206.386 Called "stat64()" for 4.85 µs
00:00.206.413 Called "mach_vm_allocate_trap()" for 1.52 µs
00:00.206.430 Called "mach_vm_deallocate_trap()" for 4.13 µs
00:00.206.436 Called "stat64()" for 8.86 µs
00:00.206.447 Called "stat64()" for 6.66 µs
00:00.206.455 Called "stat64()" for 6.11 µs
00:00.206.463 Called "stat64()" for 5.70 µs
00:00.206.471 Called "stat64()" for 5.78 µs
00:00.206.478 Called "stat64()" for 5.25 µs
00:00.206.485 Called "stat64()" for 4.68 µs
00:00.206.492 Called "stat64()" for 5.42 µs
00:00.206.499 Called "stat64()" for 5.81 µs
00:00.206.506 Called "stat64()" for 5.02 µs
00:00.206.513 Called "stat64()" for 4.98 µs
00:00.206.520 Called "stat64()" for 4.53 µs
Input Filter
called
No Detail

DTruss

DTruss is analog of strace in linux and uses DTrase for it. It allows you to trace system calls from a running process or run a process with tracing.

Note

But from past versions of Mac OS, some paths are protected by SIP(System Integrity Protection) (i.e. /usr/bin) and cannot be traced. But there is a workaround, to change settings in recovery mode but I would not recommend it to you. Then you can trace only applications which are not under SIP protection.

Examples

run and examine the “df -h” command

```
dtruss df -h
```

examine PID 1871

```
dtruss -p 1871
```

examine all processes called “tar”

```
dtruss -n tar
```

run test.sh and follow children

```
dtruss -f test.sh
```

run the “date” command and print elapsed and on cpu times,

```
dtruss -eo date
```

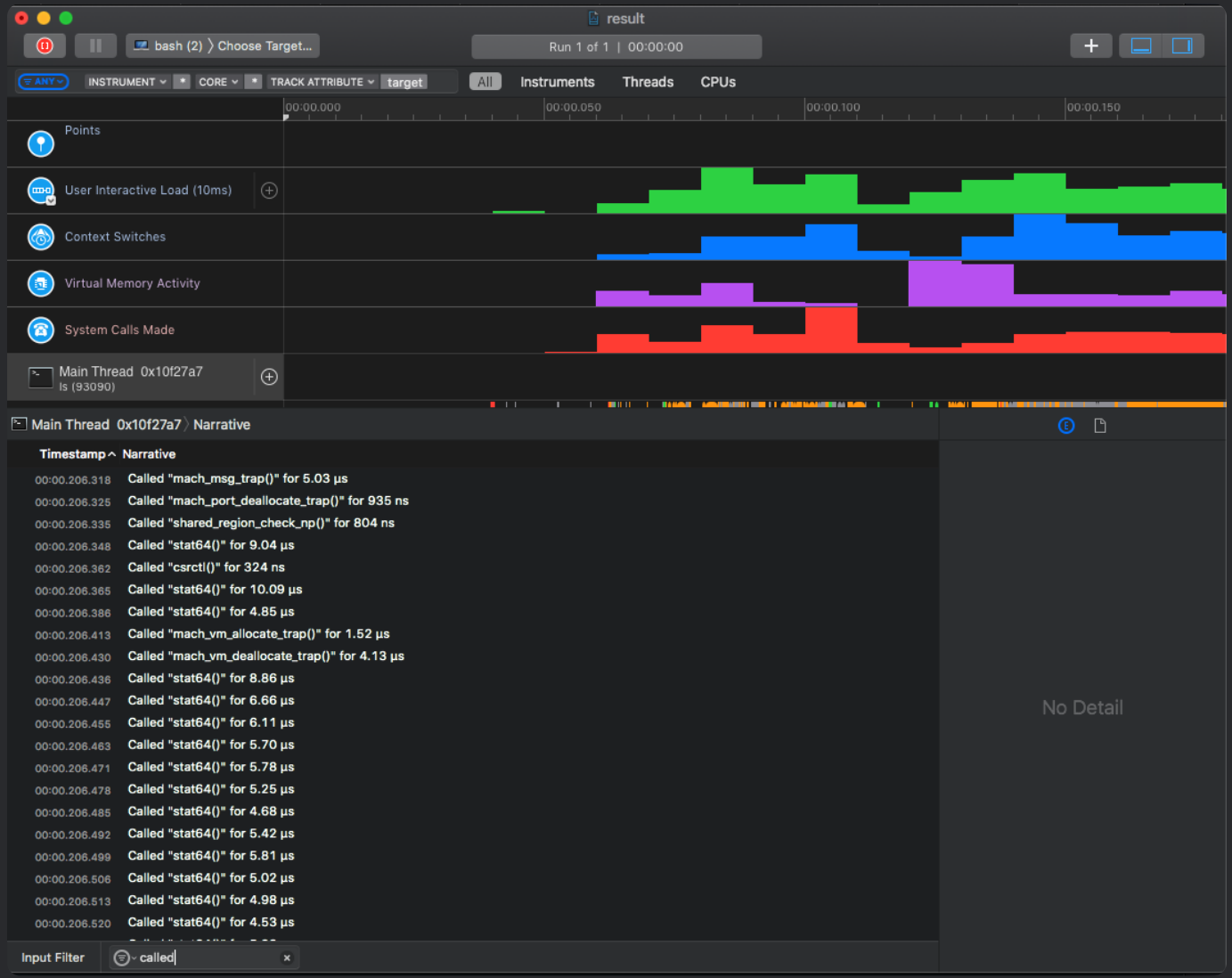
If you need to dive deeper you can check DTrace directly. DTrace is a powerful tool for real-time profiling applications and kernel. You can check it here

Instruments

As an alternative, you can try to use Instruments which is an XCode application (Manual).

Trace system calls and saves it in `result.trace`

```
instruments  -d result.trace -t 'System Trace' /bin/ls
open result.trace
```


result
bash (2) 〉Choose Target...
Run 1 of 1 | 00:00:00
INSTRUMENT
CORE
TRACK ATTRIBUTE
target
All
Instruments
Threads
CPUs
00:00.000
00:00.050
00:00.100
00:00.150
Points
User Interactive Load (10ms)
Context Switches
Virtual Memory Activity
System Calls Made
Main Thread 0x10f27a7
ls (93090)
Main Thread 0x10f27a7 〉Narrative
Timestamp
Narrative
00:00.206.318 Called "mach_msg_trap()" for 5.03 µs
00:00.206.325 Called "mach_port_deallocate_trap()" for 935 ns
00:00.206.335 Called "shared_region_check_np()" for 804 ns
00:00.206.348 Called "stat64()" for 9.04 µs
00:00.206.362 Called "csrctl()" for 324 ns
00:00.206.365 Called "stat64()" for 10.09 µs
00:00.206.386 Called "stat64()" for 4.85 µs
00:00.206.413 Called "mach_vm_allocate_trap()" for 1.52 µs
00:00.206.430 Called "mach_vm_deallocate_trap()" for 4.13 µs
00:00.206.436 Called "stat64()" for 8.86 µs
00:00.206.447 Called "stat64()" for 6.66 µs
00:00.206.455 Called "stat64()" for 6.11 µs
00:00.206.463 Called "stat64()" for 5.70 µs
00:00.206.471 Called "stat64()" for 5.78 µs
00:00.206.478 Called "stat64()" for 5.25 µs
00:00.206.485 Called "stat64()" for 4.68 µs
00:00.206.492 Called "stat64()" for 5.42 µs
00:00.206.499 Called "stat64()" for 5.81 µs
00:00.206.506 Called "stat64()" for 5.02 µs
00:00.206.513 Called "stat64()" for 4.98 µs
00:00.206.520 Called "stat64()" for 4.53 µs
No Detail
Input Filter
called