Поддержка multiple personalities в strace или как обеспечить корректную трассировку 32-битных программ на 64-битных архитектурах

17 октября 2015 г.

intro

`strace - trace system calls and signals`

— `man 1 strace`

intro

`strace - trace system calls and signals`

— `man 1 strace`

- `-c` Count time, calls, and errors for each system call and report a summary on program exit.
- `-p pid` Attach to the process with the process ID pid and begin tracing.
- `-e trace=set` Trace only the specified set of system calls.
- `-e signal=set` Trace only the specified subset of signals.

```
$ strace cat
execve("/bin/cat", ["cat"], [/* 40 vars */]) = 0
brk(0)                                  = 0x155e000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691ab4000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=135526, ...}) = 0
mmap(NULL, 135526, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8691a92000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1729984, ...}) = 0
mmap(NULL, 3836448, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f86914ed000
mprotect(0x7f869168c000, 2097152, PROT_NONE) = 0
mmap(0x7f869188c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19f000) = 0x7f
mmap(0x7f8691892000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f869189
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691a91000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691a90000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691a8f000
arch_prctl(ARCH_SET_FS, 0x7f8691a90700) = 0
mprotect(0x7f869188c000, 16384, PROT_READ) = 0
mprotect(0x60b000, 4096, PROT_READ)     = 0
mprotect(0x7f8691ab6000, 4096, PROT_READ) = 0
munmap(0x7f8691a92000, 135526)          = 0
brk(0)                                  = 0x155e000
brk(0x157f000)                          = 0x157f000
open("/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=2855152, ...}) = 0
mmap(NULL, 2855152, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8691233000
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
fadvise64(0, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
mmap(NULL, 139264, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691a92000
read(0, "", 131072)                     = 0
```

ptrace()

- `ptrace(PTRACE_foo, pid, ...)`
- `PTRACE_TRACEME`, `PTRACE_SYSCALL`, `PTRACE_PEEKUSER`, `PTRACE_PEEKDATA`

strace

1. `fork()`, `PTRACE_TRACEME` or `waitpid()`
2. `while(true)`
   - `PTRACE_SYSCALL` + `waitpid()`
   - `PTRACE_PEEKUSER` + `PTRACE_PEEKDATA`

strace

1. `fork()`, `PTRACE_TRACEME` or `waitpid()`
2. `while(true)`
   - `PTRACE_SYSCALL` + `waitpid()`
   - `PTRACE_PEEKUSER` + `PTRACE_PEEKDATA`

Аргументы и возвращаемые значения системных вызовов `strace` достает из адресного пространства tracee.

data models

Personality – одно из поддерживаемых ядром ABI.

data models

Personality – одно из поддерживаемых ядром ABI.
У разных personality могут быть разные модели данных.

	LP32	ILP32	LP64	ILP64
char	8	8	8	8
int	16	32	32	64
long	32	32	64	64
pointer	32	32	64	64

x86 и x32: ILP32. x86_64: LP64.

Проблема

Что происходит, когда strace трассирует программу, использующую другую модель данных?

Проблема

Что происходит, когда strace трассирует программу, использующую другую модель данных?

В аргументах системных вызовов попадаются аргументы типов с неявно определенной длиной, например, long и pointer-типы.
А также структуры с такими полями.
У них еще и правила alignment'а могут быть разные.

Аргументы и возвращаемые значения системных вызовов `strace` достает из адресного пространства tracee.

— несколько слайдов назад

В итоге – зависимость их вида от personality tracee.
Более того, некоторые типы зависимосят от data model tracee иначе.

```
int sigaltstack(const stack_t *ss, stack_t *oss);

typedef struct {
    void  *ss_sp;     /* Base address of stack */
    int    ss_flags;  /* Flags */
    size_t ss_size;   /* Number of bytes in stack */
} stack_t;
```

size_t, glibc: long. x86, x32 – 32, x86_64 - 64 бита.

```
int utime(const char *filename,
    const struct utimbuf *times);

struct utimbuf {
    time_t actime;       /* access time */
    time_t modtime;      /* modification time */
};
```

time_t: для x86 и x86_64 – long (32 и 64), для x32 – long long (64).

```
/*
 * Most 64-bit platforms use 'long', while most 32-bit platforms use '__u32'.
 * Yes, they differ in signedness as well as size.
 * Special cases can override it for themselves -- except for S390x, which
 * is just a little too special for us. And MIPS, which I'm not touching
 * with a 10' pole.
 */


#ifndef __statfs_word
#if __BITS_PER_LONG == 64
#define __statfs_word __kernel_long_t
#else
#define __statfs_word __u32
#endif
#endif

struct statfs {
     __statfs_word f_type;
     __statfs_word f_bsize;
     __statfs_word f_blocks;
     <...>
}
```

Проблема

strace должен знать, что ему доставать и как печатать. Но он не знает, как в данных программы с другой personality выглядят те или иные типы.
Можно прописывать руками.

Проблема

strace должен знать, что ему доставать и как печатать. Но он не знает, как в данных программы с другой personality выглядят те или иные типы.
Можно прописывать руками.

There has to be a better way! Цель – полуавтоматически строить вид составных типов для каждой поддерживаемой personality.

DWARF

DWARF is a debugging format used to describe programs in C and other similar programming languages.

— `wiki.dwarfstd.org`

Никто не знает, как выглядит тип в программе с какой-либо personality лучше, чем программа с такой personality.

```
fig7.c:
        1:  int a;
        2:  void foo()
        3:  {
        4:      register int b;
        5:      int c;
        6:  }

<1>:    DW_TAG_subprogram
          DW_AT_name = foo
<2>:    DW_TAG_variable
          DW_AT_name = b
          DW_AT_type = <4>
          DW_AT_location = (DW_OP_reg0)
<3>:    DW_TAG_variable
          DW_AT_name = c
          DW_AT_type = <4>
          DW_AT_location =
                (DW_OP_fbreg: -12)
<4>:    DW_TAG_base_type
          DW_AT_name = int
          DW_AT_byte_size = 4
          DW_AT_encoding = signed
<5>:    DW_TAG_variable
          DW_AT_name = a
          DW_AT_type = <4>
          DW_AT_external = 1
          DW_AT_location = (DW_OP_addr: 0)
```

Figure 7. DWARF description of variables *a*, *b*, and *c*.

```
$ cat utimbuf.c
#include <utime.h>
typedef struct utimbuf utimbuf_t;
utimbuf_t mpers_target_var;

$ gcc -c -gdwarf-4 -m32 utimbuf.c
```

```
-m32 -m64 -mx32 -m16
Generate code for a 16-bit, 32-bit or 64-bit environment.
The -m32 option sets "int", "long", and pointer types
to 32 bits, and generates code that runs on any i386 system.
The -mx32 option sets "int", "long", and pointer types to 32 bits,
and generates code for the x86-64 architecture.
```

— man 1 gcc

```
$ readelf --debug-dump=info utimbuf.o
        <...>
 <1><ba>: Abbrev Number: 5 (DW_TAG_structure_type)
    <bb>   DW_AT_name         : (indirect string,
        offset: 0x95): utimbuf
    <bf>   DW_AT_byte_size    : 8
    <c0>   DW_AT_decl_file    : 2
    <c1>   DW_AT_decl_line    : 37
    <c2>   DW_AT_sibling      : <0xdf>
 <2><c6>: Abbrev Number: 6 (DW_TAG_member)
    <c7>   DW_AT_name         : (indirect string,
offset: 0x154): actime
    <cb>   DW_AT_decl_file    : 2
    <cc>   DW_AT_decl_line    : 39
    <cd>   DW_AT_type         : <0x6a>
    <d1>   DW_AT_data_member_location: 0
 <2><d2>: Abbrev Number: 6 (DW_TAG_member)
    <d3>   DW_AT_name         : (indirect string,
offset: 0x19): modtime
    <d7>   DW_AT_decl_file    : 2
    <d8>   DW_AT_decl_line    : 40
    <d9>   DW_AT_type         : <0x6a>
```

```
 <1><df>: Abbrev Number: 4 (DW_TAG_typedef)
    <e0>   DW_AT_name        : (indirect string,
offset: 0x117): utimbuf_t
    <e4>   DW_AT_decl_file   : 4
    <e5>   DW_AT_decl_line   : 6
    <e6>   DW_AT_type        : <0xba>
 <1><ea>: Abbrev Number: 9 (DW_TAG_variable)
    <eb>   DW_AT_name        : (indirect string,
offset: 0xde): mpers_target_var
    <ef>   DW_AT_decl_file   : 4
    <f0>   DW_AT_decl_line   : 8
    <f1>   DW_AT_type        : <0xdf>
    <f5>   DW_AT_external    : 1
```

DWARF!

utimbuf.c →utimbuf.o →utimbuf.d →utimbuf.h

```
#include <inttypes.h>
typedef
struct {
int32_t actime;
int32_t modtime;
} m32_utimbuf_t;
```

Типичный diff

```
--- a/sysinfo.c
+++ b/sysinfo.c
@@ -1,9 +1,12 @@
 #include "defs.h"
+#include DEF_MPERS_TYPE(sysinfo_t)
 #include <sys/sysinfo.h>
+typedef struct sysinfo sysinfo_t;
+#include MPERS_DEFS

 SYS_FUNC(sysinfo)
 {
-   struct sysinfo si;
+   sysinfo_t si;
```