Upstreaming the FreeBSD Port

FOSDEM 2022

Paul FLOYD, 6 Feb 2022
Agenda

About me
About FreeBSD
About Valgrind
About Valgrind on FreeBSD
About the future
About me
Grew up here
Fast Forward 20 years
Work Here
Siemens EDA

OVERVIEW

Eldo Platform

Industry-proven platform for analog-centric circuits, offering a differentiated solution for reliability verification and comprehensive circuit analysis and diagnostics for analog, RF, and mixed-signal circuits.

- Read Fact Sheet
- Get in touch with our sales team 1-800-547-3000
About FreeBSD
FreeBSD version support matrix

- FreeBSD supports 3 major releases
- -CURRENT is the unstable development branch, now at 14.0-CURRENT
- -STABLE is the stable development branch, now at 12.3-STABLE and 13.0-STABLE
- -RELEASE are tags on the -STABLE branches for packaged releases
Platforms

- FreeBSD runs on amd64, i386, PowerPC, ARM and RISC-V
- MIPS and SPARC being phased out
- amd64 and ARM aarch64 are Tier 1
- Others Tier 2
- i386 was Tier 1 until FreeBSD 13 was released in 2021
Core and Ports

- FreeBSD core includes the kernel and also a minimal userland: shell, compiler, vi etc.

- For the rest there ports, which can install pre-build binaries or build from source. Maintainers look after ports.

- Committers are people with write access to the FreeBSD git repos (core, ports and doc)
<table>
<thead>
<tr>
<th>Number</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Created by</td>
<td>Simon Barner <a href="mailto:barner@gmx.de">barner@gmx.de</a></td>
</tr>
<tr>
<td>2</td>
<td>PORTNAME</td>
<td>valgrind</td>
</tr>
<tr>
<td>3</td>
<td>PORTVERSION</td>
<td>3.18.1</td>
</tr>
<tr>
<td>4</td>
<td>PORTREPO</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>CATEGORIES</td>
<td>devel</td>
</tr>
<tr>
<td>6</td>
<td>MASTER_SITES</td>
<td>SOURCEWARE/valgrind</td>
</tr>
<tr>
<td>7</td>
<td>MAINTAINER</td>
<td><a href="mailto:pjfloyd@wanadoo.fr">pjfloyd@wanadoo.fr</a></td>
</tr>
<tr>
<td>8</td>
<td>COMMENT</td>
<td>Memory debugging and profiling tool</td>
</tr>
<tr>
<td>9</td>
<td>LICENSE</td>
<td>GPLv2</td>
</tr>
</tbody>
</table>

FreeBSD ports

---

**100 most recent commits**

FreshPorts has everything you want to know about FreeBSD software, ports, packages, applications, whatever term you want to use.

**Mon, 17 Jan 2022**

- **wine-devel 7.0-r6.1 emulators**
  - Microsoft Windows compatibility environment
  - emulators/wine-devel: update to 7.0-r6

- **rust-analyzer 2022.01.17 devel**
  - Experimental Rust compiler front-end for IDEs
  - devel/rust-analyzer: update to 2022-01-17

- **sctd 0.2.0.10 accessibility**
  - Set color temperature (set) daemon

- **gnome-podcasts 0.5.1.1 audio**
  - Podcast app for GNOME

- **lepton 0.10.2.5 audio**
  - Rust vlc codec decoder

- **myrnx 1.2.1.7 audio**
  - Modern volume mixer for PulseAudio

- **ncspot 0.9.3.1 audio**
  - N curses Spotify client

- **shoutwave 2.0.1.7 audio**
  - Listens to internet radio
Other web sites

• Bugzilla - bug reports and also maintainer patches for committers
• Phabricator - for discussion and approval of patches (mainly core)
• FreeBSD wiki
Some differences compared to Linux
(With impact on Valgrind)

• Userland tools - sed etc.
• System compiler - clang (and lld)
• FreeBSD libc
• No /proc by default
• UFS/ZFS filesystem
About Valgrind
Tool startup, 1
A different story

- _start inline assembler in the executable
- Creates own stack
- Calls _start_in_C_freebsd
- Calls valgrind_main - 380 lines of text
- Process early command line arguments
- Start debug logging
Tool startup, 2

- Start address space manager - memory maps, segments
- Process remaining command line arguments
- Load guest image
- Sets LD_PRELOAD for core and tool
- Setup client stack - allocation and also argv/env/auxv
- Read debuginfo
- Start executing guest
Basic Guest execution

- Guest code is running in VEX
- Signals are handled by the host
- Syscalls and ioctl are all intercepted
  - Argument checking
  - Virtualisation (e.g., process management)
- Tool-specific functions also intercepted (malloc for memcheck, pthread_* for DRD and Helgrind)
Lines of Code
With regression tests

(About 750kloc in C files)
Lines of Code
Without regression tests

VEX 57 %
Coregrind 30 %
Memcheck 4 %

(About 500kloc in C files)
About Valgrind on FreeBSD
History of the port
A long and windy road

• Initially ported by Doug Rabson
• First added to FreeBSD ports on 23 May 2004, version 2.1.1?
• Maintenance taken over by Stan Sedov, August 2009
• Numerous other contributors
• My work started in earnest in January 2020
• Took over maintainership April 2021
• FreeBSD code merged in October 2021 for Valgrind 3.18.1
Port status, January 2020

broken

- On amd64, lld had added read-only non-fixed ELF segment mapping that broke Valgrind’s guest loading
- On x86 many many things broken
- DRD and Helgrind unuseable
- Zero FreeBSD specific regression tests
Development Env

- Fairly old HP z400, FreeBSD 13.0-RELEASE
- VirtualBox with images for FreeBSD 12, 13 and i386
- Triple boot FreeBSD, Fedora 34, Windows 10
Development tools

ktrace

- FreeBSD kernel tracing tool, equivalent of strace
- Good for a quick view of syscalls
- Can compare with strace
- Can compare guest application alone and under Valgrind

```
32122 slp  CALL  open(0x80022b008,0x300000<O_RDONLY ... 
32122 slp  NAMI  "/lib/libc.so.7"
32122 slp  RET  open 3
32122 slp  CALL  fstat(0x3,0x7ffffffc0)
32122 slp  STRU  struct stat {dev=18314720584186972 ... 
32122 slp  RET  fstat 0
32122 slp  CALL  mmap(0,0x1000,0x1<PROT_READ> ... 
32122 slp  CALL  nanosleep(0x7ffffff2d0,0x7fffffff2c0)
32122 slp  RET  nanosleep 0
32122 slp  CALL  exit(0)
```
Development tools

dtrace

• More general tracing tool, from OpenSolaris
• More powerful, can be scripted

```c
#!/usr/sbin/dtrace -s

/* int sysctl(const int *name, u_int namelen, void *oldp, size_t *oldlenp,
   const void *newp, size_t newlen) */

/* 202 AUE_SYSCTL STD { int freebsd32___sysctl(int *name,
   u_int namelen, void *old, uint32_t *oldlenp,
   const void *new, uint32_t newlen); */

syscall:freebsd:__sysctl:entry,
syscall:freebsd32:freebsd32___sysctl:entry {

  this->namelen = (unsigned int)arg1;

  this->name   = (int*)copyin(arg0, sizeof(int) * this->namelen);

  printf("name = {%d, %d, %d, %d, %d, %d}, namelen = %d [%s],
         this->name[0],
        this->namelen >= 2 ? this->name[1] : 0,
        this->namelen >= 3 ? this->name[2] : 0,
        this->namelen >= 4 ? this->name[3] : 0,
        this->namelen >= 5 ? this->name[4] : 0,
        this->namelen >= 6 ? this->name[5] : 0,
        this->namelen,
        execname);

  usstack();
}
```
Development tools
Binary analysis

- elfdump, objdump
- Need to look at ELF segments
- Very often need to look at guest assembler
Development tools

Valgrind traces

- Valgrind has many options and a lot of code that outputs debug info
- Verbose (-v, repeated for more verbosity)
- Debug (-d, repeated for more verbosity)
- --trace options for syscalls, redirs, signals
Memory layout
Valgrind address space manager

```
--21643:1: aspacem (0,4,7) memcheck-amd64-freebsd
--21643:1: aspacem (1,73,7) slp
--21643:1: aspacem (2,117,7) ld-elf.so.1
--21643:1: aspacem (3,142,1) vgd-b-pipe-shared-mem-vgdb-21643-by-paulf-on-euler
--21643:1: aspacem (4,201,8) vgpreload_core-amd64-freebsd.so
--21643:1: aspacem (5,280,8) libc.so.7
--21643:1: aspacem vgpreload_memcheck-amd64-freebsd.so
--21643:1: aspacem (7,402,0) [free slot: size=36 next=0]
--21643:1: aspacem 1: file 0000200000-0000200fff 4096 r---- d=0x696e301b i=1762811 o=0 (1,73)
--21643:1: aspacem 6: file 0004000000-0004005fff 24576 r---- d=0x4190bc5c i=1081502 o=0 (2,117)
--21643:1: aspacem 10: anon 000401f000-000401ffff 4096 rw--
--21643:1: aspacem 13: anon 0004820000-0004840fff 135168 rw--
--21643:1: aspacem 15: file 0004842000-0004842fff 4096 r---- d=0x696e301b i=1779577 o=0 (4,201)
--21643:1: aspacem 19: file 0004846000-000484bfff 24576 r---- d=0x696e301b i=1780364 o=0 (6,320)
--21643:1: aspacem 23: file 0004859000-00048dcfff 540672 r---- d=0x4190bc5c i=1081388 o=0 (5,280)
--21643:1: aspacem 27: anon 0004a40000-00005469ff 10m rw--
--21643:1: aspacem 29: FILE 0038000000-00380befff 782336 r---- d=0x696e301b i=1780171 o=0 (0,4)
--21643:1: aspacem 34: ANON 0038286000-0038a862ff 37m rw--
--21643:1: aspacem 36: RSVN 0402000000-0402000fff 4096 ----- SmFixed
--21643:1: aspacem 37: ANON 0402001000-04029a7fff 9m rwx--
--21643:1: aspacem 38: ANON 04029a8000-04029a9fff 8192 ----- SmUpper
--21643:1: aspacem 39: ANON 04029aa000-0402aa9fff 1048576 rwx--
--21643:1: aspacem 40: ANON 0402aaa000-0402aabff 8192 ----- SmUpper
--21643:1: aspacem 41: FILE 0402aac000-0402aacfff 4096 rw--- d=0x884e7a5b i=240 o=0 (3,142)
--21643:1: aspacem 42: ANON 0402aad000-0402a44fff 32m rwx--
--21643:1: aspacem 44: ANON 0405121000-0405220fff 1048576 rwx--
--21643:1: aspacem 46: RSVN 07fb001000-07fffbffff 15m ----- SmUpper
--21643:1: aspacem 47: anon 07ffbc8000-07fc000fff 20480 rwx--
--21643:1: aspacem 48: 07fc001000-07fffffffff 63m
--21643:1: aspacem 50: ANON 7fffffff8000-7fffffffdefff 511m ----- SmUpper
--21643:1: aspacem 51: ANON 7fffffffdefff-7fffffffefef 131072 rwx--
```
## Memory layout

### Procstat

```
paulf> procstat -v 32430
```

<table>
<thead>
<tr>
<th>PID</th>
<th>START</th>
<th>END</th>
<th>PRT</th>
<th>RES</th>
<th>PRES</th>
<th>REF</th>
<th>SHD</th>
<th>FLAG</th>
<th>TP</th>
<th>PATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>32430</td>
<td>0x200000</td>
<td>0x201000 r--</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>CN</td>
<td>slp</td>
<td>vn</td>
<td>/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x400000</td>
<td>0x400600 r--</td>
<td>6</td>
<td>28</td>
<td>377</td>
<td>0</td>
<td>CN</td>
<td></td>
<td>vn</td>
<td>/usr/local/bin//binexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x401f00</td>
<td>0x40200000 rw-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x402000</td>
<td>0x40210000 rwx</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x482000</td>
<td>0x48410000 rw-</td>
<td>27</td>
<td>27</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x484300</td>
<td>0x48430000 r--</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>CN</td>
<td></td>
<td>vn</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x484600</td>
<td>0x484c0000 r--</td>
<td>6</td>
<td>16</td>
<td>4</td>
<td>0</td>
<td>CN</td>
<td></td>
<td>vn</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x485900</td>
<td>0x488d0000 r--</td>
<td>84</td>
<td>328</td>
<td>500</td>
<td>0</td>
<td>CN</td>
<td></td>
<td>vn</td>
<td>/usr/local/bin/relpath/freebsd/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x4a4000</td>
<td>0x546a0000 rw-</td>
<td>27</td>
<td>27</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x38000000</td>
<td>0x3a8f0000 r--</td>
<td>161</td>
<td>588</td>
<td>3</td>
<td>0</td>
<td>CN</td>
<td></td>
<td>vn</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x38286000</td>
<td>0x3a830000 rw-</td>
<td>4598</td>
<td>4598</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x40201000</td>
<td>0x4029a0000 rwx</td>
<td>1220</td>
<td>1476</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x4029a000</td>
<td>0x4029aa0000</td>
<td>0</td>
<td>1476</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x4029aa00</td>
<td>0x402aaa0000 rwx</td>
<td>256</td>
<td>1476</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x402aac000</td>
<td>0x402ac0000</td>
<td>0</td>
<td>1476</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x402aad000</td>
<td>0x402aad0000 rw-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>vn</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x402ad0000</td>
<td>0x402a450000 rwx</td>
<td>1867</td>
<td>2123</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x40521000</td>
<td>0x405210000 rwx</td>
<td>256</td>
<td>2123</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x7fbbff0000</td>
<td>0x7fbbff0000 rwx</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x7fbbfd0000</td>
<td>0x7fbbfe0000 rwx</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x7fbbfe0000</td>
<td>0x7fbbff0000 rwx</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x7fbbff0000</td>
<td>0x7fc0010000 rwx</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x7fbbff0000</td>
<td>0x7fbbff0000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>df</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
<tr>
<td>32430</td>
<td>0x7fbbff0000</td>
<td>0x8000000000000 r-x</td>
<td>1</td>
<td>1</td>
<td>129</td>
<td>0</td>
<td></td>
<td></td>
<td>ph</td>
<td>/usr/local/bin/relpath/freebsd/libexec/ld-elf.so.1</td>
</tr>
</tbody>
</table>
```
FreeBSD source
Kernel, libc, libthr

• Sometimes needed to build a custom kernel

• Built debug versions of libc and libthr

• Heavy use of syscalls.master which describes all syscalls in the kernel (also i386 version)
Fixing the port
The early days

• Cloned the official sourceware.git repo
• Merged repos containing work done by Phil Longstaff, Ed Maste and Mark Johnston
• Started with GCC builds and unoptimised debug builds
• Initially did a lot of code cleaning
• Added version checking for the build OS version
• Fixed the initial creation of the stack on x86
• Added handling of non-fixed RO segments
Regression tests

Common problems

• Many testcases deliberately use Undefined Behaviour which can produce different results on different platforms

• Clang optimisation. Quite often test case code was simply optimised away. Usually fixed with either `volatile` or `__attribute__((no inline))`

• Clang code generation. For code using the ternary operator `?:`, clang emits `cmov` opcodes whilst GCC emits `cmp` and a conditional jump.
Failure at the very beginning of guest startup on x86

- The auxv that Valgrind synthesises for the guest does not contain any pointers to dynamic memory
- This includes the PAGESIZES
- rtld (the link loader) normally uses auxv PAGESIZES
- If it is missing it will use sysctls to get MAXPAGESLEN and then PAGESIZE
Failure at the very beginning of guest startup on x86

- When an x86 application is running on amd64, MAXPAGESLEN returns as 3 when it should be 2.
- The `sysctl` to read PAGESIZES then sees a request for 3 elements for something that the x86 subsystem thinks has a size of 2.
- `sysctl` returns ENOMEM 😞
- Fix in Valgrind - bodge PAGESIZES and suppress warnings 😞
- Fixed in FreeBSD Sat May 16 2020, [https://svnweb.freebsd.org/base/?view=revision&revision=361109](https://svnweb.freebsd.org/base/?view=revision&revision=361109) 😊
LD_PRELOAD

Memcheck not detecting any dynamic memory errors on x86

• Valgrind needs to set LD_PRELOAD for the core/tool shared libraries

• amd64 on amd64 and i386 on i386 use LD_PRELOAD

• i386 on amd64 uses LD_32_PRELOAD

• Detecting that an i386 process is on amd64 was not working and not setting LD_32_PRELOAD (was using sysctl HW MACHINE)

• Fix: look to see if /libexec/ld-elf32.so.1 exists on the machine
Debug printing
crash whenever -d used on x86

- Valgrind -d printing uses inline asm to make a `write` syscall to stderr
- i386 syscalls take arguments on the stack
- The FreeBSD code looked like a copy/paste from Linux, passing arguments in registers
  - (It is possible to take arguments in register like Linux but this requires both Linux emulation to be enabled and for a bit to be set in the executable file ELF headers)
- Fix: rewrite `local_sys_write_stderr` to pass arguments on the stack
DRD threadid

Assert in DRD

- DRD needs to track the threadid for each thread
- It does this for the main thread very early in process execution
- On FreeBSD, this is before libthr.so has been loaded
- So `pthread_self()` calls the stub function in libc.so which returns a junk value
- Later checks that the tracked threadid match `pthread_self()` cause an assert
- Fix: in `DRD_(init)(void)`
  `dlclose(dlopen("/lib/libthr.so.3", RTLD_NOW|RTLD_GLOBAL|RTLD_NODELETE));`
sigreturn kernel messages,1
Not fixed

User code

Kernel signal handler

Signal handler

Trampoline and Sigreturn
sigreturn kernel messages, 2
When linked with -pthread
sigreturn kernel messages,3
Working by accident under Valgrind

pid 1722 (none-amd64-freebsd): sigreturn rflags = 0x4
Some stats

How much effort?

- Around 700 pushes to the GitHub repo I used - but I do push early and push often
- 145 GitHub issues closed - enhancements as well as bugs
- Estimate 6 person months work full time
- About 9kloc added to coregrind
- 45 new regression tests
Future work

- There are a few missing features
  1. No vgdb invoker
  2. TLS handling for Helgrind
  3. Clean handling for split RW PT_LOAD (lld >= 9.0.0)
  4. llvm OMP support
- Numerous small issues
And so what?

Who is using Valgrind on FreeBSD

- Don’t have any download/usage stats
- Bug reports - good!
  - Rust issue with stack
  - Google perftools detecting Valgrind (used by SWI Prolog)
  - glusterfs crash due to bad `swapcontext`
  - HardenedBSD COMPAT10 dependency, `_realpathat` syscall
Thanks!
A long windy road, but not alone

- Thanks to all those that worked on the port before me
- Thanks to the folks on the freebsd-hackers and freebsd-toolchain mailing lists
- Thanks to the Valgrind developers for putting up with me
- Thanks to Nick Briggs, a big help for the i386 version
Contact

- Libera.Chat IRC #valgrind-dev #freebsd #freebsd-dev #freebsd-ports

- pjfloyd@wanadoo.fr

- Bug reports to https://bugs.kde.org (preferred) or https://github.com/paulfloyd/freebsd_valgrind (minor issues)