The Intricacies of Sizecoding on Linux

Pushing the interest and boundaries of a very interesting platform

Shiz/WRD







Pick one or more

revisian

- Linux
- MacOS
- Amigaaaaaaa
- C64
- A CPU and some RAM
- Something else!



Pick one or more

revision

- Windows
- Linux ← You can go here
- MacOS
- Amigaaaaaaa
- C64
- A CPU and some RAM
- Something else!





Ew, Linux?!

People are interested!

What makes Linux interesting?

→ Libraries!

FFT? Sure! GLU, Opus, all there! Rendering? ;-)





Ew, Linux?!

People are interested!

What makes Linux interesting?

→ Libraries!

FFT? Sure! GLU, Opus, all there! Rendering? ;-)

Open!

See what your dependencies *really* are doing!

	mark@nene: ~	
Registers		
rax 0x0000000000000000	rbx 0x0000000000000000	<pre>rcx 0x00007ffff7fc0f7</pre>
rdx 0x0000000000000000	rst 0x0000000000000001	rdi 0x00007ffff7fae78
rbp 0x00007ffff7fc0f70	rsp 0x00007fffffffe040	r8 0x00007ffff7fc0fd
r9 0x00007ffff7fe3b50	r10 0x00000000000000007	r11 0x0000000000000000
r12 0x00007ffff7fb0190	<pre>r13 0x00007fffffffe108</pre>	r14 0x0000000000000000
r15 0x0000000000000000	rip 0x00007ffff7ba3fed	eflags [PF ZF IF]
CS 0x0000033	ss 0x000002b	ds 0x00000000
es 0x00000000	fs 0x0000000	gs 0x0000000
Source		
# endif		
<pre>#endif /* !SHARED */</pre>		
/* Register the destru	ctor of the dynamic linker if	there is any. */
/* Register the destru- if (glibc likely (rt	ctor of the dynamic linker if ld fini != NULL))	there is any. */
/* Register the destru- if (glibc_likely (rt cxa atexit ((void	ctor of the dynamic linker if ld_fini != NULL)) (*) (void *)) rtld fini. NULL	there is any. */ . NULL):
/* Register the destru- if (glibc_likely (rt cxa_atexit ((void	ctor of the dynamic linker if ld_fini != NULL)) (*) (void *)) rtld_fini, NULL	there is any. */ , NULL);
<pre>/* Register the destru- if (glibc_likely (rt' cxa_atexit ((void #ifndef SHARED</pre>	ctor of the dynamic linker if ld_fini != NULL)) (*) (void *)) rtld_fini, NULL	there is any. */ , NULL);
<pre>/* Register the destruction if (glibc_likely (rt cxa_atexit ((void #ifndef SHARED</pre>	ctor of the dynamic linker if ld_fini != NULL)) (*) (void *)) rtld_fini, NULL r of the linc — This is only (there is any. */ , NULL);
<pre>/* Register the destru- if (glibc_likely (rt cxa_atexit ((void #ifndef SHARED /* Call the initialize</pre>	ctor of the dynamic linker if Id_fini != NULL)) (*) (void *)) rtId_fini, NULL r of the libc. This is only r	there is any. */ , NULL); needed here if we
<pre>/* Register the destru- if (glibc_likely (rt cxa_atexit ((void #ifndef SHARED /* Call the initialize are compiling for the Stack</pre>	ctor of the dynamic linker if ld_fini != NULL)) (*) (void *)) rtld_fini, NULL r of the libc. This is only r he static library in which ca:	there is any. */ , NULL); needed here if we se we haven't
/* Register the destru- if (glibc_likely (rt' cxa_atexit ((void #ifndef SHARED /* Call the initialize are compiling for ti Stack	<pre>ctor of the dynamic linker if td_ftint != NULL)) (*) (void *)) rtld_ftin, NULL r of the libc. This is only i he static library in which cas to libe core malogia at</pre>	there is any. */ , NULL); needed here if we se we haven't
<pre>/* Register the destru- if (glibc_likely (rt cxa_atexit ((void #ifndef SHARED /* Call the initialize are compiling for ti Stack from 0x00007ffff7ba3fed pile 0x0fffffforbasfed</pre>	ctor of the dynamic linker if td_fint != NULL)) (*) (vold *)) rtld_fini, NULL (*) of the libc. This is only in he static library in which cass inlibc_start_main+61 at	there is any. */ , NULL); needed here if we se we haven't /csu/libc-start.c:237
<pre>/* Register the destru- if (glibc_likely (rt cxa_atexit ((void #ifndef SHARED /* Call the initialize are compiling for the Stack from 0x00007fff7ha3fed main = 0x7fff7fae780 rorg = 1</pre>	<pre>ctor of the dynamic linker if td_ftint != NULL)) (*) (void *)) rtld_ftin, NULL r of the libc. This is only n he static library in which cas inlibc_start_nain+61 at</pre>	there is any. */ , NULL); needed here if we se we haven't /csu/libc-start.c:237
/* Register the destru- if (glubc_likely (rt cxa_atexit ((Void #ifndef SHARED /* Call the initialize are compiling for ti Stack Tom 0x00007fff7ba3fed main = 0x7fff7fae780 argc = 1	ctor of the dynamic linker if td_fint != NULL)) (*) (vold *)) rtld_fini, NULL (*) of the libc. This is only r he static library in which cas inlibc_start_main+61 at	there is any. */ , NULL); heeded here if we se we haven't /csu/libc-start.c:237
<pre>/* Register the destru- if (glibc_likely (rt cxa_atexit ((void #ifndef SHARED /* Call the initialize are compiling for ti Stack from 0x00007fff7ba3fed main = 0x7ffff7fae780 argw = 0x7ffff7fei10 argw = 0x7ffffffei10</pre>	ctor of the dynamic linker if td_ftint != NULL)) (*) (void *)) rtld_fini, NULL r of the libc. This is only n he static library in which ca: inlibc_start_nain+61 at	there is any. */ , NULL); needed here if we se we haven't /csu/libc-start.c:237
/* Register the destru- if (glubc_likely (rt cxa_atexit ((Void #ifndef SHARED /* Call the initialize are compiling for ti Stack from 0x00007fff7hasf8d argc = 1 argc = 0 argc = 0 argc = 0 argc = 0 argc = 0 argc = 1	ctor of the dynamic linker if td_fint != NULL)) (*) (vold *)) rtld_fini, NULL (*) of the libc. This is only n he static library in which cas inlibc_start_main+61 at	there is any. */ , NULL); needed here if we se we haven't /csu/libc-start.c:237
<pre>/* Register the destru- if (_gltbc_ltkely (rt cxa_atexit ((void #ifndef SHARED /* Call the initialize are compiling for ti Stack from 0x00007fff7ha3fed main = 0x7ffff7fae780 argv = 0x7fffffffe110 init = 0x7ffffffce170 finit = 0x7ffffffce170</pre>	<pre>ctor of the dynamic linker if td_ftint != NULL)) (*) (void *)) rtld_ftint, NULL r of the libc. This is only n he static library in which ca: inlibc_start_main+61 at,</pre>	there is any. */ , NULL); needed here if we se we haven't /csu/libc-start.c:237
<pre>/* Register the destru- if (glubc_likely (rt cxa_atexit ((Void #ifndef SHARED /* Call the initialize are compiling for ti Stack from 0x00007fff7has780 argc = 1 argv = 0x7fff7fae780 init = 0x7fff7fcef70 fint = 0x7fff7fcef70 fint = 0x7fff7fcef70</pre>	ctor of the dynamic linker if td_fint != NULL)) (*) (vold *)) rtld_fini, NULL r of the libc. This is only n he static library in which can inlibc_start_main+61 at 50 <_dl_fini>	there is any. */ , NULL); heeded here if we se we haven't fcsu/libc-start.c:237
<pre>/* Register the destru- if (_gltbc_ltkely (rt cxa_atexit ((void #ifndef SHARED /* Call the initialize are compliing for ti Stack from 0x00007fff7ha3fed main = 0x7fff7fed70 init = 0x7fff7fr6070 finit = 0x7ffff7fe30 rtid_finit = 0x7ffff7fe30 stack_end = 0x7ffff7fe30</pre>	<pre>ctor of the dynamic linker if td_ftint != NULL)) (*) (void *)) rtld_fini, NULL r of the libc. This is only n he static library in which cas inlibc_start_main+61 at, 50 <_dl_fini> 08</pre>	there is any. */ , NULL); needed here if we se we haven't /csu/libc-start.c:237
<pre>/* Register the destru- if (glubc_lukely (rt cxa_atexit ((Void #ifndef SHARED /* Call the initialize are compiling for ti Stack from 0x00007fff7/basfed 3 main = 0x7fff7/fae780 argc = 1 argv = 0x7fff7/fae780 init = 0x7fff7/fcef70 finit = 0x7fff7/fcef70</pre>	ctor of the dynamic linker if td_fint != NULL)) (*) (vold *)) rtld_fini, NULL r of the libc. This is only n he static library in which can inlibc_start_main+61 at 50 <_dl_fini> 80	there is any. */ , NULL); heeded here if we se we haven't fcsu/libc-start.c:237
<pre>/* Register the destru- if (_gltbc_ltkely (rt cxa_atexit ((void #ifndef SHARED /* Call the initialize are compliing for the stack from 0x00007fff7hasTed main = 0x7fff7faeT80 argc = 1 argv = 0x7fff7fre070 init = 0x7fff7fre070 rtid_ftnl = 0x7ffff7fe0760 rtid_ftnl = 0x7fff7fre01ba stack_end = 0x7ffff7fe01ba arguments)</pre>	<pre>ctor of the dynamic linker if td_ftint != NULL)) (*) (void *)) rtld_fini, NULL r of the libc. This is only n he static library in which ca: inlibc_start_main+61 at, 50 <_dl_fini> 08</pre>	there is any. */ , NULL); needed here if we se we haven't /csu/libc-start.c:237

>>>



Ew, Linux?!

People are interested!

What makes Linux interesting?

→ Libraries!

FFT? Sure! GLU, Opus, all there! Rendering? ;-)

→ Open!

See what your dependencies *really* are doing!

→ Universal!

Secretly powers a lot of devices, lots of opportunities!

Linux prod count:



Linux prod count:

WHAT'S WRONG?

(as per Pouet)

OFFICIAL[™] DEMOPARTY SURVEY



1 LACK OF TOOLING

2 LACK OF INFO

PRECONCEPTIONS?

Revision party quotes

"Linux sucks! But every OS sucks." "I wouldn't even know where to start..."

"I'm giving up on this piece of shit VST, I'll just do Famitracker instead"

Quotes for illustrative purposes only, possibly slightly altered to protect the guilty

systemctl enable linux-prods # rc-update add linux-prods # update-rc.d add linux-prods # chkconfig --add linux-prods



. TOC .

- 1. Target platform
- 2. Executable basics
- 3. Dynamic linking
- 4. Console to GL
- 5. Crunching it all down

6. The **FUTURE** »

\$ stat /target

- → Latest Ubuntu 64-bit currently: 18.10
- → Proprietary NVIDIA driver
- → Default packages only!

(still a lot) libopus, imagemagick, fftw, espeak...



Consequences

- → glibc ld.so (dynamic linker)
- → no SDL, GLFW, GLUT
- → 64-bit libraries only ... no x32 either ... :-(
- → Static linking: forget about it





	anding	- 1							
	Jualing			typede	ef sti	ruct	{	dont[ET N	
\rightarrow	ELF headers ca	nbe	big!	E1	1519116 1532_F	Half Nord		ype, e_mac	chine;
\rightarrow	\rightarrow W/ho checks these?			F1	f32	Addr	e ei	ntry;	
Static: kernel		typede El El	f struc f32_Wor f32_0f1 f32_Add	ct { rd f	p_ty p_o1	/pe; ffset; addr:	off, e_sh ags; size; entsize.	hoff; e phnum:	
	Dynamic: ld.so	typede E E E ur ur E } Elf3	ef stru lf32_Wo lf32_Ad lf32_Wo nsigned nsigned lf32_Ha 32_Sym;	ct { rd s dr s rd s char s char s lf s	st_nar st_val st_siz st_in st_oth st_shr	ne; lue; ze; fo; ner; ndx;	ddr; lesz; nsz; ags; ign;	entsize, strndx;	e_shnum;

Does it need all of this?

\$ readelf -a ./test
Class:
Data:
OS/ABI:

 \rightarrow

 \rightarrow

- ^{",} Start of section headers: <u>h</u> Flags:
 - Size of section headers: Number of section headers:

unknown: 31 unknown: db unknown: b0

23378022 (bytes into file) 0x6ebe389

32973 (bytes) 60248

\$./test
hello world

Process loading examined

(how do I shot ELF?)



System calls!

Kernel decides what happens:

- → Static binary: kernel parses directly
- → Dynamic binary: dynamic loader loads binary instead

Check parsing code of either!

- \rightarrow Unparsed fields \rightarrow code (or zeroes)
- → Parsed but controllable fields (e_entry)

Stack layout:

```
argv[0] = "./test" <----.
rsp-n-m: AT_NULL, 0 |
rsp-n : AT_FOO, bar
NULL
...
environ[1]
rsp-48: environ[0]
&argv[3] = NULL
&argv[2]
&argv[2]
&argv[1]
&argv[0] ------'
rsp: argc=3</pre>
```

- → Manual system calls (int 0x80/syscall/swi #0)
- → Graphics: /dev/console, /dev/vcsa, fbdev https://www.pouet.net/prod.php?which=3696
- → Sound: device files, ioctl(), write() nah actually, just pipe to /usr/bin/aplay

Entry to sizecoding

- → Default entrypoint:
 C runtime library (crt0/crt1.0)
- → Roll our own! -nostartfiles
- → Info passed by kernel on the stack

Stack not aligned by default!

Required by SSE or segfault 1-byte trick: **push rax**

Meet Marcos.

- → Serious programmer
- → Runs a Windows shop
- → Doesn't know dynamic linking
- → Wants to learn Linux sizecoding



Marcos just wants to use GL. What he needs is...

Dynamic Linking

...but he's heard that's hard.



What makes dynamic linking so heavy?

typedef struct {
 Elf64_Word st_name;
 unsigned char st_info;
 unsigned char st_other;
 Elf64_Section st_shndx;
 Elf64_Addr st_value;
 Elf64_Xword st_size;
} Elf64_Sym;

- → Dynamic linker (PT_INTERP)
- → Needed libraries (DT_NEEDED)
- → All symbols, verbatim
- → PIC and relocation headers
- → Support headers

\$ cc -0s -o hello hello.c && strip hello \$ ls -l hello -rwxr-xr-x 1 mark mark 14312 Apr 20 09:58 hello

What makes dynamic linking so heavy?

Relocation: dynamic linker instruction to patch an executable in-memory

Everytime an instruction accesses a symbol. This scales absolutely terribly.

GOT: global offset table, structure at the start where all resolved symbols are put once **PLT:** procedure linkage table, small code stubs that jump to addresses in the GOT

```
typedef struct {
    Elf32_Addr r_offset;
    Elf32_Word r_info;
} Elf32_Rel;
```

```
typedef struct {
    Elf32_Addr r_offset;
    Elf32_Word r_info;
    Elf32_Sword r_addend;
} Elf32_Rela;
```

ld.so cares more :(

Need dynamic linker to do anything at all \rightarrow Extremely minimal headers become hard \rightarrow non-standard



return 1;

}

- Nobody said we had to use **ld**. so all the time! \rightarrow
- \rightarrow Minimal but *barely-satisfying* headers to just have ld.so load our binary and dependencies
- Do everything else ourselves, no more \rightarrow Elf_Sym/Elf_Rel headers and extra sections!

```
$ 1d ./test
Dynamic section at offset 0xe0 contains 6 entries:
 Taa
        Type
Name/Value
0x10164
Shared
library: [libGL.so.1]
Shared
library: [libgtk-3.so.0]
Shared
library: [libgobject-2.0.so.0]
Shared
library: [libc.so.6]
0x000000000000006 (SYMTAB)
                           0x0
```

Prior art

→ bold

-

http://www.alrj.org/pages/bold.html

Paved the way for this approach! Processes relocations manually Not portable Broken due compiler change :-(

dnload https://github.com/faemivah/dnload

Portable! Includes GLSL minifier Own full linking step Very intrusive Hardcoded list of known symbols Input is (C/C++) source, not object files Was until recently broken on Linux

introducing smol

simple/shoddy/smart minsize-oriented linker

simple simply replace your 1d step with smold
portable wraps GNU ld, loader code portable to any ELF platform
generic feed it .o files from any language, integrate it how you want
uses custom hash-based import system, no STRTAB/SYMTAB

scan input *.o files for
unresolved symbols

readelf & scanelf to the rescue!

resolve symbols create hash symbol table

we have all the required flags!



create minimal new header attach loader, do final link

custom linker script for efficiency

[light side]

smold\$	readelf	-sW obj	/hello	o.start.o grep UND
NOTYPE	LOCAL	DEFAULT	UND	
NOTYPE	GLOBAL	DEFAULT	UND	printf_chk
NOTYPE	GLOBAL	DEFAULT	UND	_GLOBAL_OFFSET_TABLE
NOTYPE	GLOBAL	DEFAULT	UND	exit
NOTYPE smold\$	GLOBAL	DEFAULT	UND	libc_start_main

```
[section .rodata.neededlibs]
_strtab:
_symbols.libc: db "libc.so.6",0
[section .data.smolgot]
_symbols:
global __printf_chk
__printf_chk:
__symbols.libc.__printf_chk: dq 0xb4ebe20b
global __libc_start_main
__libc_start_main:
__symbols.libc.__libc_start_main: dq 0xf63d4e2e
global exit
exit:
___symbols.libc.exit: dq 0x7c967e3f
db 0
__symbols.end:
```

[dark side]

idea: obtain dynamic linker's own state and abuse it for symbol resolution

- → DT_DEBUG section, offset 0x4 meh, extra sections
- → leak from _dl_start_user()

poke **1d.so** state, obtain library symbol hash arrays **for free**!

load



```
/* Symbol hash table. */
Elf Symndx 1 nbuckets;
Elf32 Word 1 gnu bitmask idxbits;
Elf32 Word 1 gnu shift;
const ElfW(Addr) *1 gnu bitmask;
union
  const Elf32 Word *1 gnu buckets;
  const Elf_Symndx *l_chain;
};
union
  const Elf32_Word *l_gnu_chain_zero;
  const Elf Symndx *1 buckets;
};
```

x86_64: own mini-PLT+GOT

x86: E9 D00DBEEF

jump! hash replaced by relative address **PLT** *is* **GOT**

Other small tricks

- → PT_INTERP: do we actually need it? \$ /lib64/ld-*.so ./x
- → Sections? Never heard of 'em! one giant RWX übersection
- → argc/argv/envp not passing them saves bytes! bring your own environment

Speaking of *small* tricks...

COMPRESSION.xz

Everyone uses shell droppers

00000000	<pre> cp \$0 /tmp/M;(se </pre>
00000010	d 1d \$0 lzcat)>\$
00000020	_;\$_;exit.]
00000030	?.E.pr
00000040	tv:~JT.^.
00000050	J;1!I.:N
00000060	" c.5>.K
00000070	8.6~.h.>9,.
080000080	i;"?

~42 bytes overhead

- \rightarrow Touch the file system
- → *Ewww*, sh?!
- \rightarrow LZ compression sucks (for x86):

But they suck.

8d <mark>3d</mark> 1c060001	lea edi, [0x100061c]
8b2db8000001	mov ebp, [0x10000b8]
8b6d04	mov ebp , $[ebp + 4]$
8b <mark>6d0c</mark>	mov ebp, $[ebp + 0xc]$
8b7508	mov esi, [ebp + 8]
ad	lodsd eax, [esi]
83 <mark>f805</mark>	cmp <mark>eax, 5</mark>
75 <mark>fa</mark>	jne 0x1000102

Attempt to fix #1: Fishypack, vondehi

- → Really small decompression stub (asm)
- → Does everything in-memory (memfd_create)
- → A bit larger: ~160 bytes
- → Still LZ (uses gzip/xz)

https://gitlab.com/PoroCYon/vondehi

Attempt to fix #2: XLINK

See also: LCA2019, "Executable Code Golf"

→ PAQ1-based Arithmetic Range Coder

- → Sparse *n*-gram model \rightarrow good for x86!
- → No kkrunchy "instruction stream splitting"; use general approach

Currently DOS-only... Expect more soon!

https://github.com/negge/xlink

Compiler voodoo

Telling gcc/clang to behave when you don't want to go 100% ASM

Compile

-02/-0s -march=haswell -ffast-math

- -flto -fuse-linker-plugin
- -fno-unwind-tables
- -fno-asynchronous-unwind-tables
 -ffunction-sections -fdata-sections

-no-pie -fno-pic -fno-pie -fno-plt
-fno-stack-check -fno-stack-protector
-fomit-frame-pointer

Link

-march=haswell -Wl,-i

```
-flto -fuse-linker-plugin
-ffunction-sections -fdata-sections
-Wl,--gc-sections
```

-no-pie -fno-pic -fno-pie -fno-plt
-nostdlib -nostartfiles

Output: another object file

Now we have all the ingredients Road to GL

Linking to GL: GetProcAddr ... or use libglvnd, link as usual

Attempt #1 X11+GLX

https://github.com/blackle/Linux-OpenGL-Examples: xlib-opengl.c

xlib sucks

Verbose API, brittle, GLX stuff

Attempt #1.1 X11+EGL

More compatible than GLX! Less strict about the window state!

This still sucks.

EGL can sometimes be worse!

Attempt #2

• • •

libgstreamer1.0-0:amd641.14.4-1libgtk-3-0:amd643.24.1-1ubuntu2libgtk-3-bin3.24.1-1ubuntu2

ubuntu-18.10-desktop-amd64.manifest

gtk_gl_area_make_current(gtk_gl_area_new());

We have a GL context!

https://github.com/blackle/Linux-OpenGL-Examples: gtk-opengl.c

SYS_pipe(syncpip2); "manname" In file included from src/snd.c:4: inc/sys.h:433:33: note: expected 'int *' but argument is of type 'volatile int *' SYSCALL FUNC ssize t SYS pipe(int fd[2]) {

cc -o "obj/frag_glsl.bin.o" -c "obj/frag_glsl.bin.c" -flto -fuse-linker-plugin -I inc -I obj -g -02 -fno-plt -fno-sta ck-protector -fno-stack-check -fno-unwind-tables -fno-asynchronous-unwind-tables -fomit-frame-pointer -ffast-math -Wa IL -Mno-attributes -Mno-unused-function -Mno-unknown-pragmas -no-ple -fno-plt -fno-PIE -fno-PIE -m64 -march-core2 -ff unction-sections -fdata-sections -DWEED_ENVIRON -DCUSTOM_CRT0 -DBACKEND_X11_EGL -DCANVAS_RESIZABLE -DVSYNC -DUSE_LIB (C_START_MAIN -OMDOE REL -DCANVAS_WIDTH-8800 -DCANVAS_HEGNT-640

cc -o "obj/snd.bin.o" -c "obj/snd.bin.c" -flto -fluse-linker-plugin -I inc. -I obj -g -02 -fno-plt -fno-stack-protector -fno-stack-check -fno-unwind-tables -fno-asynchronous-unwind-tables -fomit-frame-pointer -ffast-math -Wall -Wno-attr ibutes -Wno-unused-function -Wno-unknown-pragmas -no-pie -fno-pic -fno-PIE -fno-PIC -m64 -march=core2 -ffunction-sect ions -fdata-sections -ONEED_ENVIRON -DCUSTOM_CRT0 -DBACKEND_X11_EGL -DCANVAS_RESIZABLE -DVSYNC -DUSE_LIBC_START_MAIN -OMODE_REL -DCANVAS_WIDTH=800 -OCANVAS_HEIGHT=640

lcc -o "obj/vert_glsl.bin.o" -c "obj/vert_glsl.bin.c" -flto -fuse-linker-plugin -I inc -I obj -g -02 -fno-plt -fno-sta ck-protector -fno-stack-check -fno-unwind-tables -fno-asynchronous-unwind-tables -fomit-frame-pointer -ffast-math -Wa ll -Wno-attributes -Wno-unused-function -Wno-unknown-pragmas -no-pie -fno-pic -fno-PIE -fno-PIE -m64 -march=core2 -ff unction-sections -fdata-sections -ONEED_ENVIRON -DCUSTOM_CRT0 -DBACKEND_X11_EGL -DCANVAS_RESIZABLE -DVSYNC -DUSE_LIB [C_START_MAIN -DMODE_RL -DCANVAS_NIDTH=800 -DCANVAS_RESIGHT=640

cc -Wl,-i -o "bin/main.inc.o" obj/main_x11_glx.c.o obj/crt0.c.o obj/main_default.c.o obj/shdr.c.o obj/main_x11_egl.c. o obj/file.c.o obj/loop.c.o obj/snd.c.o obj/frag_glsl.bin.o obj/snd.bin.o obj/vert_glsl.bin.o \

-Wl,--whole-archive demo/bin/demo_s.a -Wl,--no-whole-archive 🔪

-flto -fluse-linker-plugin -I inc -I obj -g -02 -fno-plt -fno-stack-protector -fno-stack-check -fno-unwind-tabb les -fno-asynchronous-unwind-tabbes -fnoit-frame-pointer -ffast-math. Wall -Wno-attributes -Mno-unused-function. -Wnounknown-pragmas -no-ple -fno-plc -fno-PIE -fno-PIC -m64 -march=core2 -ffunction-sections -fdata-sections -DNEED_ENVI ROW -DCUSTOM_CRT0 -DBACKEND_X11_EGL =DCANVAS_RESIZABLE -DVSYNC -DUSE_LIBC_START_MAIN -DNODE_REL -DCANVAS_WIDTH=800 -D (CANVAS_HEIGHT=640 -nostlib -Wl.-entry -Hl_start -Hl_-rgc-sections -Wl.-print-gc-sections

/usr/bin/ld: removing unused section '.comment' in file '/tmp/cc9EWNwndebugobi

/usr/bin/ld: removing unused section '.text.effect_kill' in file '/tmp/cclEachi.ltrans0.ltrans.o' /usr/bin/ld: removing unused section '.text.effect_tick' in file '/tmp/cclEachi.ltrans0.ltrans.o' /usr/bin/ld: removing unused section '.text.sffect_init' in file '/tmp/cclEachi.ltrans0.ltrans.o' /usr/bin/ld: removing unused section '.text.smd_wait_sync' in file '/tmp/cclEachi.ltrans0.ltrans.o' /usr/bin/ld: removing unused section '.text.smd_dwait_sync' in file '/tmp/cclEachi.ltrans0.ltrans.o' /usr/bin/ld: removing unused section '.text.smd_fork_addr' in file '/tmp/cclEachi.ltrans0.ltrans.o' /usr/bin/ld: removing unused section '.text.smd_fork_file' in file '/tmp/cclEachi.ltrans0.ltrans.o' /usr/bin/ld: removing unused section '.text.loop tick' in file '/tmp/cclEachi.ltrans0.ltrans.o'

cc -o "bin/main.id" "bin/main.inc.o" -flto -fuse-linker-plugin -I inc -I obj -g -O2 -fno-plt -fno-stack-protector -fn o-stack-check -fno-unwind-tables -fno-asynchronous-unwind-tables -fomit-frame-pointer -ffast-math -Wall -Wno-attribut es -Wno-unused-function -Wno-unknown-pragmas -no-pie -fno-pic -fno-PIC -Mno-PIC -M64 -march-wall -Winction-sections -fdata-sections -ONEED_ENVIRON -DUCSTOM Corf0 -DBACKEND X11 EGL -DCAUNAS.RESIZABLE -DVSYUC -DUSE_LIBC START.MAIN -DM ODE.REL -DCAUNAS_WIDTH=800 -DCANNAS_HEIGHT=640 -Wl,--as-needed -LEGL -lGL -lm -LX11 -LXCursor -LXrandr -Ml,--build-tab =none -z noreiro -WL,-z,noseparate-code -WL,--no-ed-frame-hdr -WL,--no-td-generated-unwind-info -T Id/smaller64, ld ve_--

objcopy -R .gnu.version -R .gnu.version r "bin/main.ld" "obj/main.ld.nover"

/home/poro/src/arco/k2/glengine/ext/noelfver/noelfver "obj/main.ld.nover" > "bin/main.ld.stripped"
strip -s. "bin/main.ld.stripped"

/home/poro/src/arco/k2/glengine/ext/ELFkickers/sstrip/sstrip -z "bin/main.ld.stripped"

chmod +x "bin/main.ld.stripped"

/home/poro/src/arco/k2/glengine/tool/autovndh.py -g -l -v --vndh /home/poro/src/arco/k2/glengine/ext/vondehi "bin/mai n.ld.stripped" > "bin/main.ld.vndh"

3015 xz

final: 3181

chmod +x "bin/main.ld.vndh

python3 /home/poro/src/arco/k2/glengine/ext/smol/smol.py -lEGL -lGL -lm -lX11 -lXcursor -lXrandr -lc bin/ma "obi/main_smol_syms.asm"

nasm -f elf64 -g -F dwarf -DUSE_INTERP -DUSE_DNLOAD_LOADER -I "/home/poro/src/arco/k2/glengine/ext/smol/src/" -o "obj /main.smol.stub.o" "obi/main.smol.svms.asm"

ld -nostartfiles --oformat=binary -T /home/poro/src/arco/k2/glengine/ext/smol/ld/link.ld -Map=bin/main.smol.map --cre f -o "bin/main.smol" obi/main.smol.stub.o bin/main.inc.o

/home/poro/src/arco/k2/glengine/tool/autovndh.py -g -l -v --vndh /home/poro/src/arco/k2/glengine/ext/vondehi "bin/mai n.smol" > "bin/main.smol.vndh"

2545 xz

final: 27

chmod +x "bin/main.smol.vndh"

poro@alphard:glengine\$ bin/main.smol.vnd



sound, (unoptimized) shader, no cursor, 64-bit

smol + vondehi + lzma

UI

Go make an intro!

Lots of optimizations are still possible, this is only the beginning

Shiz/WRD
/dev/by-row/15
hi@shiz.me
twitter.com/dev_console

PoroCYon/K2
/dev/by-row/20
no social media
just IRC

We greet

auld / alrj / blackle / breadbox / Calodox faemiyah / gib3 & tix0 / las / leblane parcelshit / PWP / Team210 / unlord / yx

Questions?

github.com/Shizmob/liner

#lsc on IRCNet