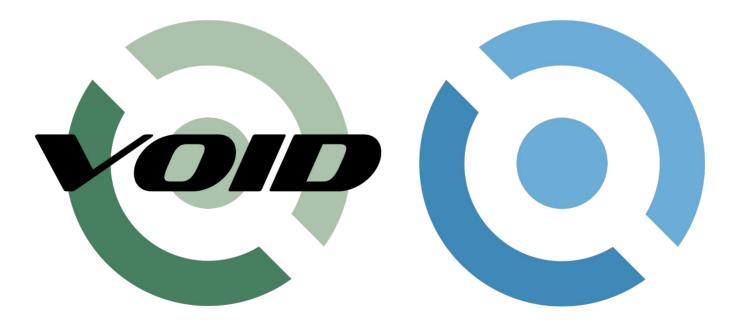
Void Linux

We heard you like little endian



Firstly, who am I?

- A programmer from the Czech Republic
- Involved in open source since 2007
- Background in graphics toolkits (EFL), programming languages, game development
- Currently the primary maintainer for POWER architecture support in Void (and other several things)

What is Void Linux?

- Independent Linux distro with its own package manager
- Focus on simplicity and pragmatism
- Rolling release, stable software, binary based
- Portable (also on x86, ARM, MIPS), choice of glibc and musl, runit, "ports" (xbps-src), cross-compilation
- Low barrier of entry, open and inclusive, informal

Void on POWER vs other distros

- Desktop/workstation focus, not server
- Both little and big endian support
- ELFv2 ABI on big endian glibc
- 4K pages kernel by default
- Wider desktop software support

Quick history of Void on POWER

- Late 2018 initial port on Talos 2 Lite (ppc64le)
- Early 2019 gradual expansion of package coverage
- 64-bit BE musl port, 32-bit ports
- April 2019 64-bit BE glibc port (ELFv2 ABI)
- Late 2019 complete repository coverage
- OpenPOWER Summit EU talk
- November 2019 commit bit in upstream

News since then

	concurrency.hh - q66 - Code - OSS						
t Selection View Go Run Termi	nal Help						
EXPLORER	\cdots \mathbb{G} concurrency.hh \times	65					
> OPEN EDITORS	libostd > ostd > G+ concurrency.hh						
∨ Q66	207 * Every scheduler its supposed to implement its own method `start`, which 208 * will take a function, arguments and will return any value returned from						
> konsole-le	209 * the given function. The given function will be used as the very first						
> letest	210 * task (the main task) which typically replaces your `main` function.						
> libcs							
> libcubescript	* The 'start' method will also set the internal current scheduler pointer						
✓ libostd	• 214 */						
> build	215 struct OSTD_EXPORT scheduler {						
> build test	216 private: 217 struct stack_allocator {						
> corotest							
> data	219 stack_allocator(scheduler &s) noexcept: p_sched(&s) {}						
> doc	220 221 stack_context allocate() {						
> examples	<pre>222 return p_sched->allocate_stack();</pre>						
✓ ostd	223)						
> build	224 225 void deaallocate(stack_context &st) noexcept {						
	226 p_sched->deallocate_stack(st);						
> ext	227)						
€ algorithm.hh	228 229 private:						
€+ argparse.hh	230 scheduler *p_sched;						
C+ channel.hh	231 };						
C concurrency.hh	232 233 protected:						
C context_stack.hh	234 /** Obrief Does nothing, this base class is empty. */						
C coroutine.hh	235 scheduler() {}						
€ environ.hh	236 237 public:						
ۥ event.hh	238 /** @brief Does nothing, this base class is empty. */						
€+ format.hh	239 virtual ~scheduler();						
C+ generic_condvar.hh	240 241 scheduler(scheduler const 6) = delete:						
€+ io.hh	241 scheduler (scheduler const α) = delete; 242 scheduler(scheduler & δ) = delete;						
€+ path.hh	243 scheduler & operator=(scheduler const &) = delete;						
€ platform.hh	244 scheduler &operator=(scheduler &&) = delete; 245	Code - OSS					
€ process.hh	245 246 /** @brief Spawns a task.	Code - OSS					
€ range.hh	247 *	Code - OSS					
ۥ stream.hh	248 * Spawns a task and schedules it for execution. This is a low level 249 * interface function. Typically you will want ostd::spawn().	Version: 1.52.1					
C+ string.hh	 249 * Interface function. Typically you will want ostal:spawn(). 250 * The detailed behavior of the function is completely scheduler dependent. 	Commit: Unknown Date: 2020-12-18T07:40:06.982Z (1 mo ago)					
	251 *	Electron: 9.4.1					
G+ thread pool.hh	252 * @see ostd::spawn() 253 */	Chrome: 83.0.4103.122 Node.is: 12.14.1					
_		V8: 8.3.110.13-electron.0					
G+ thread_pool.hh G+ unit_test.hh G+ vecmath.hh		OS: Linux ppc64 5.10.9_1					
G+ unit_test.hh G+ vecmath.hh	255						
G+ unit_test.hh G+ vecmath.hh > src	● 255 256 /** @brief Tells the scheduler to re-schedule the current task.						
G• unit_test.hh G• vecmath.hh > src > tests	 255 256 257 * * * * 	Сору					
C• unit_test.hh C• vecmath.hh > src > tests ♦ .gitignore	 255 256 /** @brief Tells the scheduler to re-schedule the current task. 257 * 258 * In ostd::thread_scheduler, this is just a hint, as it uses 05 threading 259 * facilities. In coroutine based schedulers, this will typically suspend 						
G• unit_test.hh G• vecmath.hh > src > tests	255 256 /** @brief Tells the scheduler to re-schedule the current task. 257 * z58 * In ostd::thread_scheduler, this is just a hint, as it uses 05 threading						

Java bootstrap

- OpenJDK 8 and 11 in repositories
- Needs itself to build (one version older)
- Re-introduction of GCC6 (last version with GCJ)
- Bootstrap packages for JDK 7, 9, 10
- JITed on ppc64(le), slow on ppc32 (zero backend)

Go bootstrap

- Only on ppc64le
- Using binary bootstrap (official binaries)
- gccgo unreliable (gcompat on musl)
- BE support wrong ABI and needs POWER8
- 32-bit support missing entirely

Haskell bootstrap

- Self-built binary bootstrap snapshots
- Includes musl (figured out cross-builds)
- BE missing for now (ELFv2 patch made though)
- Big endian bugs prevent functional bindist
- More target support later for 8.10

Other languages

- Greater Common Lisp support
- SBCL finally in repos (for now LE only)
- Bootstrapped with ECL
- Preparing a Clisp update
- D, Zig and some others still missing

LibreSSL performance

- Void uses LibreSSL instead of OpenSSL
- Assembly only on x86 and some 32-bit ARM
- Inferior performance and no hardware crypto support
- Solution: import assembly bits from OpenSSL upstream

LibreSSL performance

- https://github.com/q66/libressl-portable-asm
- Now shipping in Void
- Also adds Aarch64 assembly support and improves ARM
- Up to 20x performance increase
- Especially in hw crypto (ghash/AES etc.) on POWER9

Chromium support

- No official upstream support
- Downstream patches available
- Shipping in Void since 84
- Currently only browser engine with JIT support
- Works on musl as well

Electron support

- We have systemwide Electron now
- For now enables VS Code, Element, Rocket.Chat-Desktop
- Only LE, musl works
- Ugly build system workarounds

AMD GPU support

- Problem: AMDGPU DC
- Haphazard hardware floating point usage
- Page size dependency
- Upstream support only in recent kernels
- Need backports provided down to 5.4

Cross-toolchain rework

- Code duplication for cross-toolchain build templates
- Solution: introduce a common build-style
- Each cross-toolchain template is now ~50 lines
- Extra goodies: glibc cross on musl hosts, etc.
- Unified configure args and so on for all
- No more dirty masterdirs

Infrastructure status

Terminal -	^ _ 🗆 X
	100.0%] 100.0%]
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ ⊽Command	
24639 g66 20 056.46 49.26 28128 5 0.6 38.3 1669h gemu-system-ppc64 -drive file=/media/vms/builder_ppc64	be.gcow2.if=v
24647 g66 20 0 56.46 49.26 28128 S 0.0 38.3 49h18:04 gemu-system-ppc64 -drive file=/media/vms/builder_ppc64	_be.qcow2,if=v
24651 q66 20 0 56.46 49.26 28128 S 0.0 38.3 49h12:21 qemu-system-ppc64 -drive file=/media/vms/builder_ppc64	
24653 q66 20 0 56.46 49.26 28128 S 0.0 38.3 48h31:12 gemu-system-ppc64 -drive file=/media/vms/builder_ppc64	
24649 q66 20 0 56.46 49.26 28128 S 0.0 38.3 48h29:22 qemu-system-ppc64 -drive file=/media/vms/builder_ppc64	
24657 q66 20 0 56.46 49.26 28128 S 0.0 38.3 48h0s:40 qemu-system-ppc64 -drive file=/media/vms/builder_ppc64	
24655 g66 20 0 56.46 49.26 28128 S 0.0 38.3 48h07:48 gemu-system-ppc64 -drive file=/media/vms/builder_ppc64	
24659 q66 20 0 56.46 49.26 28128 S 0.0 38.3 48h03:52 qemu-system-ppc64 -drive file=/media/vms/builder_ppc64 24661 q66 20 0 56.46 49.26 28128 S 0.0 38.3 48h01:00 gemu-system-ppc64 -drive file=/media/vms/builder_ppc64	
24661 q66 20 0 56.46 49.26 28128 S 0.0 38.3 48h01:00 qemu-system-ppc64 -drive file=/media/vms/builder_ppc64 24663 q66 20 0 56.46 49.26 28128 S 0.0 38.3 47h44:23 qemu-system-ppc64 -drive file=/media/vms/builder_ppc64	
2003 00 - 25 - 0 - 30-10 - 37-20 20120 - 5 - 0 - 30-3 - 4714 1-23 deal 35:00 - 010 - 010 - 110 -	_bc.qcow2,11-v

Infrastructure status

- Still a Talos 2 Lite (18C, 128G RAM) in my bedroom
- Big endian builds in a VM (KVM-HV)
- Little endian builds on bare metal
- Primary mirror separate server (in Chicago, IL)
- 10G network, 2TB storage

Infrastructure status

- Several other mirrors provided by community
- Could use faster build hardware
- Could use hardware not in my bedroom
- Primary mirror should migrate to build machine
- More build automation necessary WIP

32-bit little endian



32-bit little endian

- Why not?
- Officially does not exist
- In practice, this just works though
- Pretty basic bringup process
- Made easy by xbps-src

32-bit little endian

- No native support for 32-bit LE in Linux
- Works as a chroot on 64-bit kernel though
- But not out of box: fixes are needed
- Fix 32-bit compat in kernel first
- Several issues

32-bit LE kernel compat

- Entering signal handlers clears MSR_LE
- vDSO32 disabled by default
- Certain vDSOs broken on LE disabled those
- Syscalls with 64-bit args: wrong byte order
- All very trivial patches
- Things "run" without them

32-bit LE kernel compat

- But they crash at random points
- After patching, no more known issues
- Fixes are being upstreamed
- VDSO32 will be fixed by moving it to C
- Backports in all Void kernels (4.19+)

Bootstrapping 32-bit LE

- First: create build and cross xbps-src profiles
- Second: create cross-toolchains for 32-bit LE targets
- Minor patching in glibc and musl needed
- Adjustments around xbps-src needed

Bootstrapping 32-bit LE

- Try cross-compiling base-chroot
- Minimal set of packages for the build container
- Fix any errors along the way
- Relatively few and mostly trivial
- Binary-bootstrap a build container after that
- Then rebuild every package (no cross)

Bootstrapping 32-bit LE

- Now we can build the other base metapackages
- Afterwards, other software can be built
- Usual "fix whatever pops up" approach applies
- Most things generally just work
- Could probably function on 32-bit userland

32-bit LE: the Buts

- No official support in glibc
- Dynlinker, symvers etc. shared with 32-bit BE
- Going to propose an official port later
- Same SVR4 ABI as 32-bit BE
- Musl pretty much just works

32-bit LE: the Why

- Because we could, of course
- Easier emulation of 32-bit x86
- Port LE Linux to G3s/G4s? :)
- Or use BE Linux and swizzle syscall data?
- For now, strictly POWER8+ though

32-bit LE: the obstacles

- LLVM support
- Recently upstreamed by the FreeBSD folks
- Backported to Void
- Rust support
- Blocks a fair amount of userland right now
- WIP patches available

32-bit LE: the obstacles

- Proper glibc port needs to be made
- Use the opportunity to improve the ABI?
- Use 64-bit long doubles later (ditch ibm128)
- Choose a unique dynlinker name and fresh symvers
- Fix up our 32-bit BE ports too?

32-bit LE: the results

- Testing repos now available (glibc and musl)
- Can fully compile itself
- A portion of the repo packaged
- Core userland, dev tools, LLVM, Mesa, SDL...
- Initial port of box86

Box86

- Linux x86 emulator developed for ARM devices
- Needs 32-bit little endian host
- Initial PowerPC port now
- Can run glxgears
- Can run Unreal Tournament
- Fairly slow though... no dynarec

The future

Well Linux for PowerPC/Power ISA (unofficial) - packages attaints – Mozella Firedx I Void Linux for PowerPC/Power ISA (unofficial) - packages Documentation CitHub Void Linux for PowerPC/Power ISA (unofficial) - statistics Woid Linux for PowerPC/Power ISA (unofficial) - statistics News Download Packages Documentation CitHub Void Linux for PowerPC/Power ISA (unofficial) - statistics Some interser delises and poblic scenarios should never happen. Some interser delises not always the case for things such as language toolchains though, as well as temporarily broken things.

	void-packages	ppc64le	ppc64le-musl	ppc64	ppc64-musl	ррс	ppc-musl	ppcle	ppcle-musl
# buildable	12050	11567	11521	11109	11081	10609	10562	10562	7869
% buildable		95.99%	95.61%	92.19%	91.96%	88.04%	87.65%	87.65%	65.3%
# built		11567	11521	11109	11081	10609	10562	832	824
% built (total)		95.99%	95.61%	92.19%	91.96%	88.04%	87.65%	6.9%	6.84%
% built (avail)		100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	7.88%	10.47%
# unique	7883	7536	7510	7131	7115	7028	6997	7003	5343
% unique		95.6%	95.27%	90.46%	90.26%	89.15%	88.76%	88.84%	67.78%
# built		7536	7510	7131	7115	7028	6997	389	390
% built (total)		95.6%	95.27%	90.46%	90.26%	89.15%	88.76%	4.93%	4.95%
% built (avail)		100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	5.55%	7.3%

Filter packages...

Colors: up to date, out of date, not built, unavailable, missing dependency, unavailable subpackage.

Package	ppc64le	ppc64le-musl	ppc64	ppc64-musl	ррс	ppc-musl	ppcle	ppcle-musl
	0.0.23b_5	0.0.23b_5	0.0.23b_5		0.0.23b_5	0.0.23b_5		
0ad-data	0.0.23b_2	0.0.23b_2	0.0.23b_2	0.0.23b_2	0.0.23b_2	0.0.23b_2		0.0.23b_2
2048-qt	0.1.6_1	0.1.6_1	0.1.6_1	0.1.6_1	0.1.6_1	0.1.6_1	0.1.6_1	0.1.6_1
2bwm	0.3_1	0.3_1	0.3_1	0.3_1	0.3_1	0.3_1	0.3_1	0.3_1
	1.0.1_1	1.0.1_1	1.0.1_1	1.0.1_1	1.0.1_1	1.0.1_1	1.0.1_1	1.0.1_1
Зргоху	0.8.13_1	0.8.13_1	0.8.13_1	0.8.13_1	0.8.13_1	0.8.13_1	0.8.13_1	0.8.13_1
64tass	1.55.2200_2	1.55.2200_2	1.55.2200_2	1.55.2200_2	1.55.2200_2	1.55.2200_2	1.55.2200_2	1.55.2200_2

Infrastructure

- Official repo at some point?
- 64LE should be easy enough
- 64BE and 32-bit have some issues with server software
- They also need clearing up the ABI situation

ELFv2 BE ABI

- We will need official support in glibc
- Right now things work "by accident"
- Need unique dynlinker name and symvers
- Upstream said they wouldn't be against
- But, need to formalize the ABI

ELFv2 BE ABI

- IBM thinks VSX requirement is a part of the ABI
- Solution: Formalize "ELFv2 Legacy-Compatible" ABI
- To be shared by the new glibc port, musl and FreeBSD
- ELFv2 minus POWER8/VSX requirement

ELFv2 BE ABI

- Extension of the official spec
- No VSX nor VMX requirement
- 64-bit long doubles ditch ibm128, and do not use ieee128 (needs vector registers for passing)
- Coming during 2021

Other things

- Upstream as many patches as possible
- Write a new installer
- Work on enablement of new software
- Work with upstreams on adding support

Conclusion

- Lots of work has been done in a year and half
- There is still plenty more to be done
- Join us on IRC: #voidlinux-ppc @ freenode
- For detailed statistics, visit https://repo.voidlinux-ppc.org/stats.html

Questions time And thanks for listening!



Daniel Kolesa Twitter: @octaforge daniel@octaforge.org https://voidlinux-ppc.org FOSDEM 2021