



Pyromaniac: Evolution

Adapt, Evolve or Die

Gerph, November 2025





0. Introduction



Introduction



What I'll talk about

I'll be talking about ...

1. Recap on RISC OS Pyromaniac.
2. Towards 64-bit RISC OS.
3. Changing my approach.
4. Conclusion.

Hopefully I won't be *too* technical, but there will be examples of running code.



Introduction



Who am I?

- I'm Charles, but known as Gerph in most things online.
- I worked at RISCOS Ltd and produced RISC OS Select.
- I ported almost all of RISC OS to be 32 bit.
- I've written the only other implementation of RISC OS - RISC OS Pyromaniac - from scratch.
- I'm a strong advocate of taking RISC OS forward, rather than letting it stay stagnant.





1. Recap on RISC OS Pyromaniac





Recap on RISC OS Pyromaniac

What is RISC OS Pyromaniac? (1)

- An implementation of RISC OS.
- Intended for testing, development, debugging and experimentation.
- A system that helps you to try things out.
- Runs on modern systems.
- Written in high level language.





Recap on RISC OS Pyromaniac

What is RISC OS Pyromaniac? (2)

- Easy to run things without hardware or system emulators.
- Provides many debug options to explain what is happening in the system.
- Allows a full disassembly of every instruction.
- High level language means changing behaviour is easier.
- Cloud systems mean that automated testing is possible.
- Use the advanced editors, IDEs and AI to develop your code.



Recap on RISC OS Pyromaniac



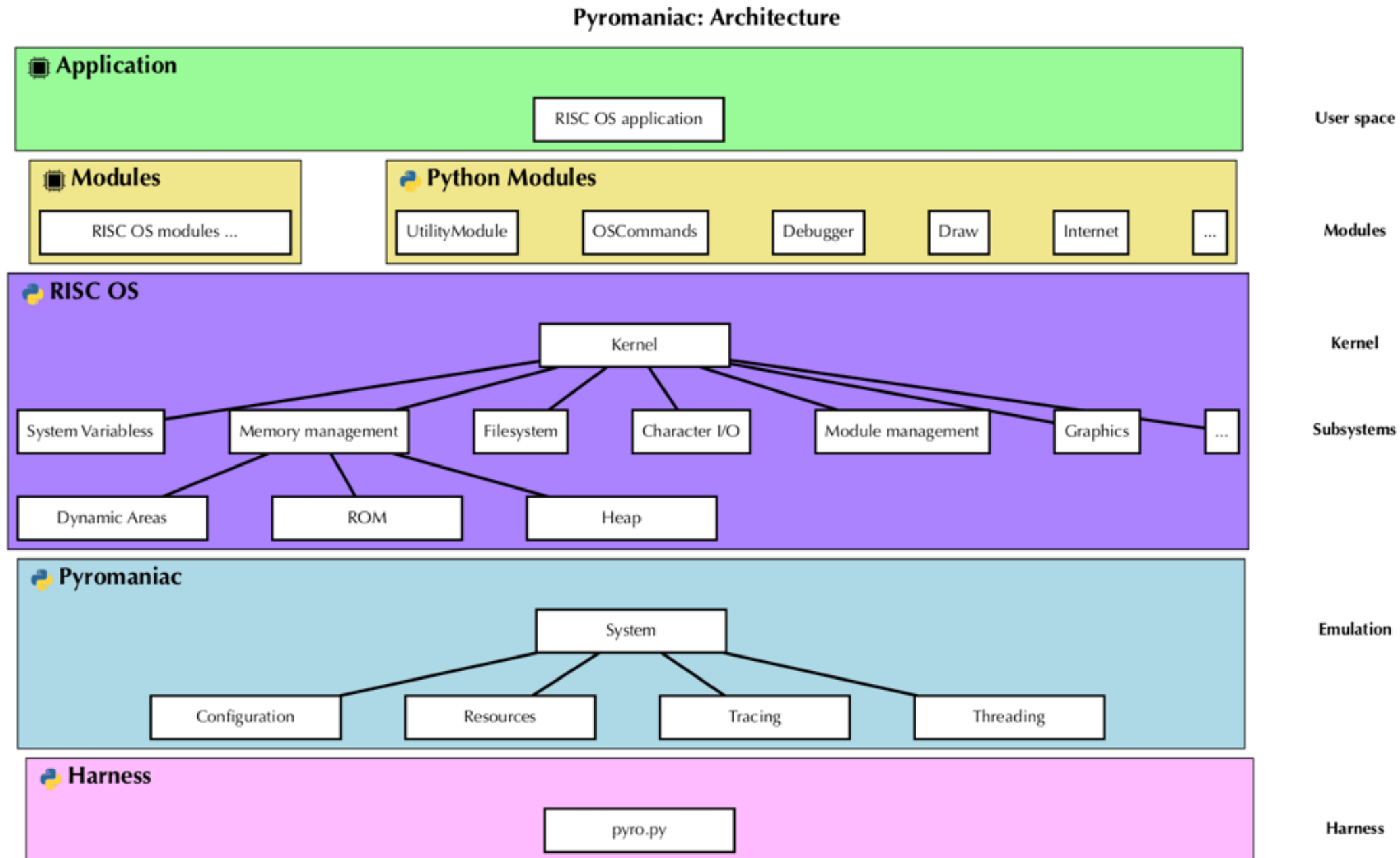
What's under the hood? (1)

- RISC OS Pyromaniac implements documented RISC OS interfaces.
- Doesn't implement everything - depends on what you're testing.
- Doesn't have hardware - no memory mapped devices.
- Does implement RISC OS interfaces to drivers.
- OS is written in Python.
- Emulation provided by Unicorn - a QEmu-derived system.



Recap on RISC OS Pyromaniac

What's under the hood? (2)



Recap on RISC OS Pyromaniac



Why do this?

- I enjoy it.
- I want to make it easier to develop software.
- I want to enable others to do things.
- I want to be able to use modern techniques for development.

... and from that, it's grown far beyond my original goals.





RISC OS Pyromaniac demo





2. Towards 64-bit RISC OS





Towards 64-bit RISC OS

What's changed? (1)

Lots of things!

The detail is in the Pyromaniac changelog on the website.

Most of it relates to 64-bit work.



Towards 64-bit RISC OS

What's changed? (2)





Towards 64-bit RISC OS

What driven the changes?

- Testing with a broad selection of applications found bugs.
- Porting old games made improvements to the system.
- Trying out different ways of doing things suggested better debug and implementations.
- Experimenting with architectures produced 64-bit RISC OS.





Towards 64-bit RISC OS

6502 emulation

- I wanted to try out a 6502 emulator - running a system like the BBC Micro.
- Updated an existing open source emulator to use Unicorn-like interfaces.
- Restructured interfaces so that trapping system calls was easier.

I wanted to make it possible to run 6502 emulation inside RISC OS Pyromaniac.

- If you can run 6502 with RISC OS Pyromaniac, anything is possible.
- It's an intermediate step towards AArch64 or x64.

But actually it's a bit too alien to make work.





Towards 64-bit RISC OS

Replacing ARM with AArch64

- Update the disassembler system to support Thumb.
- Then update to support AArch64.
- Introduce DebuggerPlus features - widely used, and we don't want to clash.
- Debugger also supports the exception dump.
- Making the exception dump descriptive allows for other architectures.





Towards 64-bit RISC OS

Describing CPU registers

`OS_PlatformFeatures_64` describes the exception dump layout, to address the general problem for the future. The dump layout describes the architecture identifier, the length of instructions and the size of the dump block itself. And then for each register, we describe:

- The register's name.
- How offset the data is stored at.
- How wide the register is in bits.
- The alignment (eg the stack in AArch64 must be aligned to 4 bits).

For flags, we also describe where the flag is in a register, and its meaning.



Towards 64-bit RISC OS

Christmas calendars 2023



```
DecAOF (Completed)
Norcroft RISC OS Arm64 C vsn 5.91 [14 Nov 2023]
** Header (file ADFS::HardDisc4.$DDE-Examples.C/C++CHello.c.HelloW
AOF file type: Little-endian, Relocatable object
AOF Version: 312
No of areas: 1
No of symbols: 5

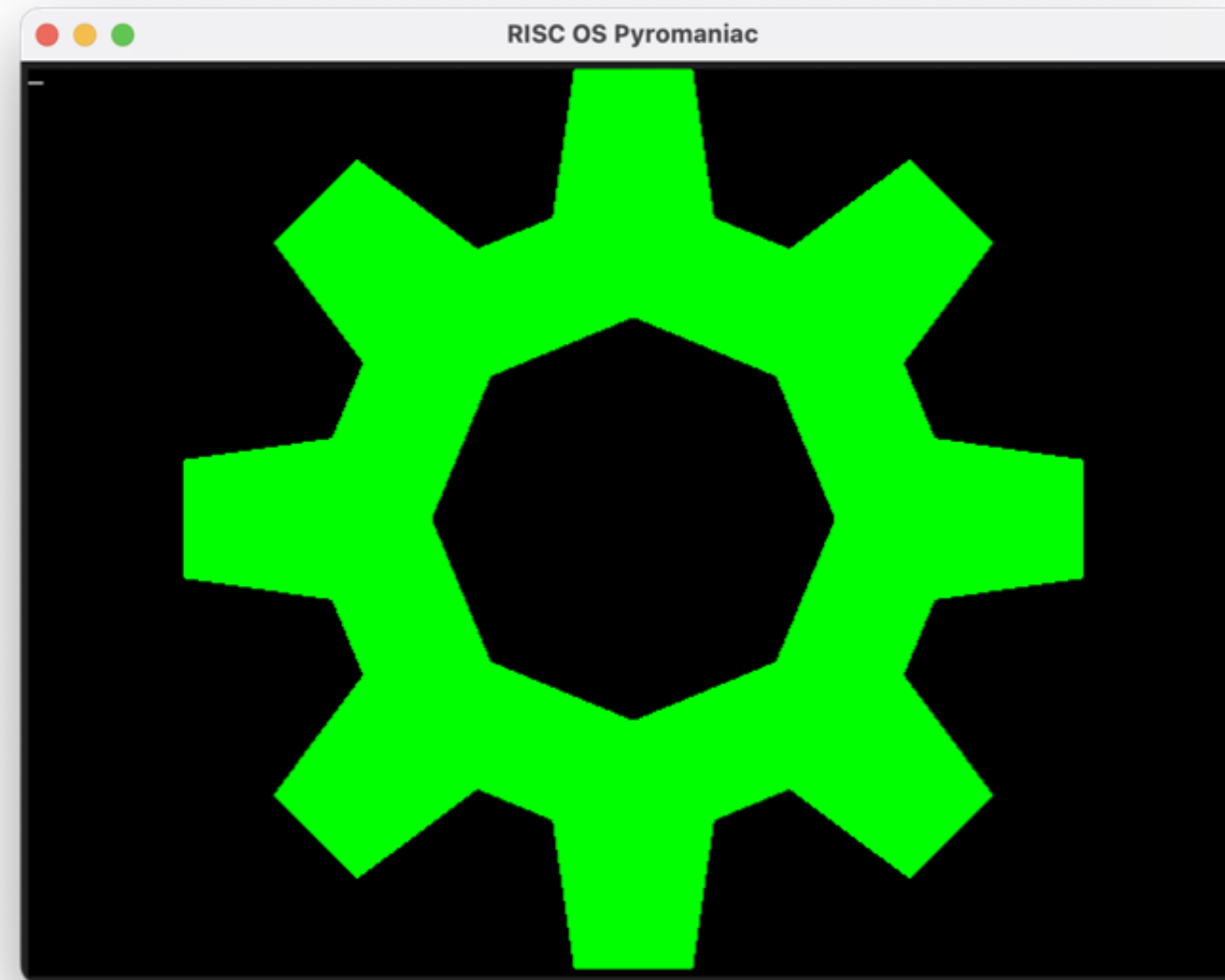
** Area C$$code, Alignment 4, Size 60 (0x003c), 1 relocations
Attributes: Code{AArch64,NoSWStackCheck}: Read only: VFP
0x000000: a9bf7bfd .{.. : STP x29,lr,[sp,#-16]!
0x000004: 910003fd .... : MOV x29,sp
0x000008: 100000a0 .... : ADR x0,0x1c
0x00000c: 97ffffffd .... : BL _printf
0x000010: 52800000 ...R : MOV w0,#0
0x000014: a9c17bfd .{.. : LDP x29,lr,[sp,#16]!
0x000018: d65f03c0 ... : RET
0x00001c: 73616553 Seas : DCD 0x73616553
0x000020: 73276e6f on's : DCD 0x73276e6f
0x000024: 65726720 gre : DCD 0x65726720
0x000028: 6e697465 etin : DCD 0x6e697465
0x00002c: 66207367 gs f : DCD 0x66207367
0x000030: 206d6f72 rom : DCD 0x206d6f72
0x000034: 4c4f4f52 R00L : DCD 0x4c4f4f52
0x000038: 0000000a .... : DCD 0xa

** Relocation table
At 00000c: Instr[97ffffffd] PC-relative to symbol _printf
```



Towards 64-bit RISC OS

The first AArch64 RISC OS program





Towards 64-bit RISC OS

Presenting 64-bit RISC OS

- Those programs were made open source, on GitHub.
- A C library for 64-bit RISC OS was created on GitHub.
- I gave a presentation on what I had done and decisions I had made at ROUGOL in August.
- I encouraged people that this was something we could do.





Presentation resources



Towards 64-bit RISC OS



Creating a build environment

- All the 32-bit tools and libraries were already built into a Docker image for easier use.
- Extending this to 64-bit tools made the system much more useful.
- Took many, many iterations!



Towards 64-bit RISC OS



Live coding (1)

- Coding on YouTube live streams.
- Partly to improve my skills.
- Showing people how they can do things.
- To give the community more resources.
- Suggested ages ago by someone in a meeting.
- And because Matt Godbolt did it!



Towards 64-bit RISC OS

Live coding (2)



```
rodix.c
28
29
30
31 /***** Gerph *****/
32 Function: lookup_swi
33 Description: Convert SWI number to a string
34 Parameters: swinum = the number to convert
35 Returns: pointer to static string, or NULL if not decodable
36 *****/
37 const char *lookup_swi(uint32_t swinum)
38 {
39     static char buffer[128];
40     _kernel_oserror *err;
41     err = _swix(OS_SWINumberToString, _INR(0, 2), swinum, buffer, sizeof(buffer));
42     if (err)
```

```
good
mine
newdebug.txt
output-darm
output-pyro
pyro
pyrodebug.txt
reloccode.txt
trace.txt
trace2.txt
../darm-internal.h
../old-armv7-tbl.note
../tests/expand
../tests/tests
../utils/elfdarm

nothing added to commit but untracked files present (use "git add" to track)
charles@mooncake ~/external/ports/darm/RISCOS
riscos-cdir h
riscos-cc -c -Wc -fa -IC:
Norcroft RISC OS ARM C vsn 5.18
"c/rodix", line 107: Warning: a
"c/rodix", line 118: Warning: a
"c/rodix", line 132: Warning: a
c/rodix: 3 warnings, 0 errors,
riscos-link -rmf -rescan -C++ -o rm32/Debugger,ffa oz32.modhead oz32.rodix oz32.tpa oz32.veneer oz32.cli-parser oz32.memor
.io oz32.os oz32.introspection oz32.defaultcpuregs oz32.breakpoints oz32.prefetch oz32.ifsrdfsr oz32.faultinfo C:o.stubsG
Debugger: Module built {RAM}
charles@mooncake ~/external/ports/darm/RISCOS (add-fpa-disassembly+4)> cp rm32/Debugger.ffa ~/Library/ApplicationSupport/R
charles@mooncake ~/external/ports/darm/RISCOS (add-fpa-disassembly+4)>
mv ~/Library/ApplicationSupport/RPCEmu/hostfs/debugger.ffc ~/Library/ApplicationSupport/RPCEmu/hostfs/debuggertest,ffc
charles@mooncake ~/external/ports/darm/RISCOS (add-fpa-disassembly+4)>
```

```
HostFS:HostFS.$ debuggertest (Code u)
00000000: 00000000: E92D4000: PUSH (lr)
00000004: 173: E1000001: MOV r0, r1
00000008: 00000000: 00000000: ADD lr, 00000000 ; -> code: at 00000000
0000000C: 00000000: E4000001: LDRB r0, [r0], #1
00000010: 00000000: E3300064: TEQ r0, #100
00000014: 00000000: 00000070: BEQ 000001fc ; call: disassemble
00000018: 00000000: E3300066: TEQ r0, #102
0000001C: 00000000: 00000190: BEQ 00000190 ; call: disassemble_fpa
00000020: 00000000: E3300074: TEQ r0, #116
00000024: 00000000: 0000020c: BEQ 0000020c ; call: disassemble_thumb
00000028: 00000000: E3300070: TEQ r0, #112
0000002C: 00000000: 00000247: BEQ 00000247 ; call: debugger_plus
00000030: 00000000: E3300061: TEQ r0, #97
00000034: 00000000: 00000250: BEQ 00000250 ; call: disassemble_arch
00000038: 00000000: E3500020: CMP r0, #32
0000003C: 00000000: 00000069: BEQ 00000069
00000040: 00000000: EF000001: SWI OS_WriteS
00000044: Synt: 746E7953: STRBTVC r7, [lr], #2307
00000048: ex: 20307061: E0RSCS r7, r10, r1, ror #16
0000004C: debu: 75626564: STRBVC r6, [r2, #1300]
00000050: gger: 72656767: RSBVC r6, r5, #19c0000
00000054: fd1: 7C647020: Undefined instruction
00000058: fit1: 7C747C66: Undefined instruction
0000005C: pf(b: 623C6670: EORSVS r6, r12, #17000000
00000060: ic), 2C3E6369: Undefined instruction
00000064: eor: 726F653C: RSBVC r6, pc, #1f000000
00000068: >.lp: 707C7C3E: RSBVC r2, r12, lr, lsr r12
0000006C: d(f1: 6C663C64: Undefined instruction
00000070: ags): 3E736761: Undefined instruction
00000074: 1c7: 70c13e3e: STRBVC r012, [r012], #1000 ; -17c
```

```
HostFS 4 Net Discs Apps
charles@mooncake ~/external/ports/darm/RISCOS (add-fpa-disassembly+4)> cp rm32/Debugger.ffa ~/Library/ApplicationSupport/R
charles@mooncake ~/external/ports/darm/RISCOS (add-fpa-disassembly+4)>
mv ~/Library/ApplicationSupport/RPCEmu/hostfs/debugger.ffc ~/Library/ApplicationSupport/RPCEmu/hostfs/debuggertest,ffc
charles@mooncake ~/external/ports/darm/RISCOS (add-fpa-disassembly+4)>
```



Towards 64-bit RISC OS

Christmas calendars 2024



```
RISC OS 7.66 (08 Dec 2024) 8MB
RISC OS Pyromaniac (AArch64)

None of the 32bit binaries will work here.
64bit binaries are in $.Library64.

Use ctrl-c for Escape, and ctrl-d to exit.

Supervisor

*simple_hello_world
Seasons greetings from RISC OS 64
*memoryi 81e4+20
000081E4 : .{.. : A9BF7BFD : STP      x29, x30, [sp, #-410]!    ; Function: main
000081E8 : .... : 910003FD : MOV      x29, sp
000081EC : .... : F0000000 : ADRP     x0, 40000b000    ; -> [494000153, 4aa1503e1, 4aa1403e0, 497fffd19]
000081F0 : ..(. : 9128E000 : ADD     x0, x0, #4a38     ; #2616, (long)-> 40000ba38 = "Seasons greetings from RISC OS 64"
000081F4 : m... : 9400006D : BL      4000083a8         ; -> Function: printf
000081F8 : ...R : 52800000 : MOVZ    w0, #0
000081FC : .{.. : A8C17BFD : LDP     x29, x30, [sp], #410
00008200 : .._. : D65F03C0 : RET
*█
```



Towards 64-bit RISC OS



Moonshots

| *"RISC OS is at risk of being left behind unless we act decisively."*

-- Thus spake ROOL, as they asked for £2.5 million.



Towards 64-bit RISC OS



Moonshots

“RISC OS is at risk of being left behind unless we act decisively.”

-- Thus spake ROOL, as they asked for £2.5 million.

- The Moonshots are intended to be larger blocks of work building towards a version of RISC OS that can be moved to 64-bit.
- They were looking for large donators to help with this.

I had done a lot of the basis for developing a 64-bit system, so I made a claim on their bounty.



Towards 64-bit RISC OS



My bounty claim (1)

Why ?

- Partly to raise awareness that I've been doing work towards this for years.
- Partly that if there's some money available, it'd be nice.
- Partly to highlight the amount of benefit it would be to work with others.

And of course, there was the fact that this would be done openly so that the community would see the co-operation.



Towards 64-bit RISC OS



My bounty claim (2)

What was it?

- Software and tools:
 - A 64-bit OS for testing and development.
 - Automation for modern CI development.
 - Tooling to build for 64-bit.
 - Documentation tailored towards documenting variants of the OS.
 - Test suites that exercise the current OS.
- Rationale:
 - Details of how this fits into the development process.
 - Evidence of what's been claimed.
- A claimed value.





3. Changing my approach





Changing my approach

What's going wrong? (1)

- Things hadn't worked out with ROOL (still waiting for them to provide guidance, or... anything really).
- The community hadn't moved to start towards 64-bit.
- My technical things aren't making any impact...

So let's work out how to change the approach...



Changing my approach

What's going wrong? (2)

I can't change ROOL, but why is the community not on board?

- Apathy.
- Lack of direction.
- Waiting for something to happen. Or just for ROOL.
- Scepticism.
- Resignation to their fate.
- No need to change.
- Just wanting to use the system.
- Longing for things from 30 years ago.
- Lack of time and inclination.



Changing my approach

What's going wrong? (3)

What have I been doing?

- Communicating.
- Open sourcing software.
- Demonstrating.
- Presenting and live coding.
- Providing encouragement.



Changing my approach

Make a plan. Execute it. (1)

- There's no written plan for RISC OS Pyromaniac.
- There's only me working on it.
- Having a plan for RISC OS 64 work would hit some of the issues I just mentioned.

Of course, just having a plan doesn't magically make people get on board... but it contributes to the viability of the project.

Changing my approach

Make a plan. Execute it. (2)

A plan will...

- Define what needs to be done.
- Allow problems to be ironed out early.
- Show that things have been thought about.
- Identify deficiencies.
- Provide encouragement to help out.
- Offers guidance for what's needed.
- Give confidence that progress can be made.

... and avoids the current situation where people just hope that something will magically appear.



Changing my approach

Work out what we mean

There are different types of components...

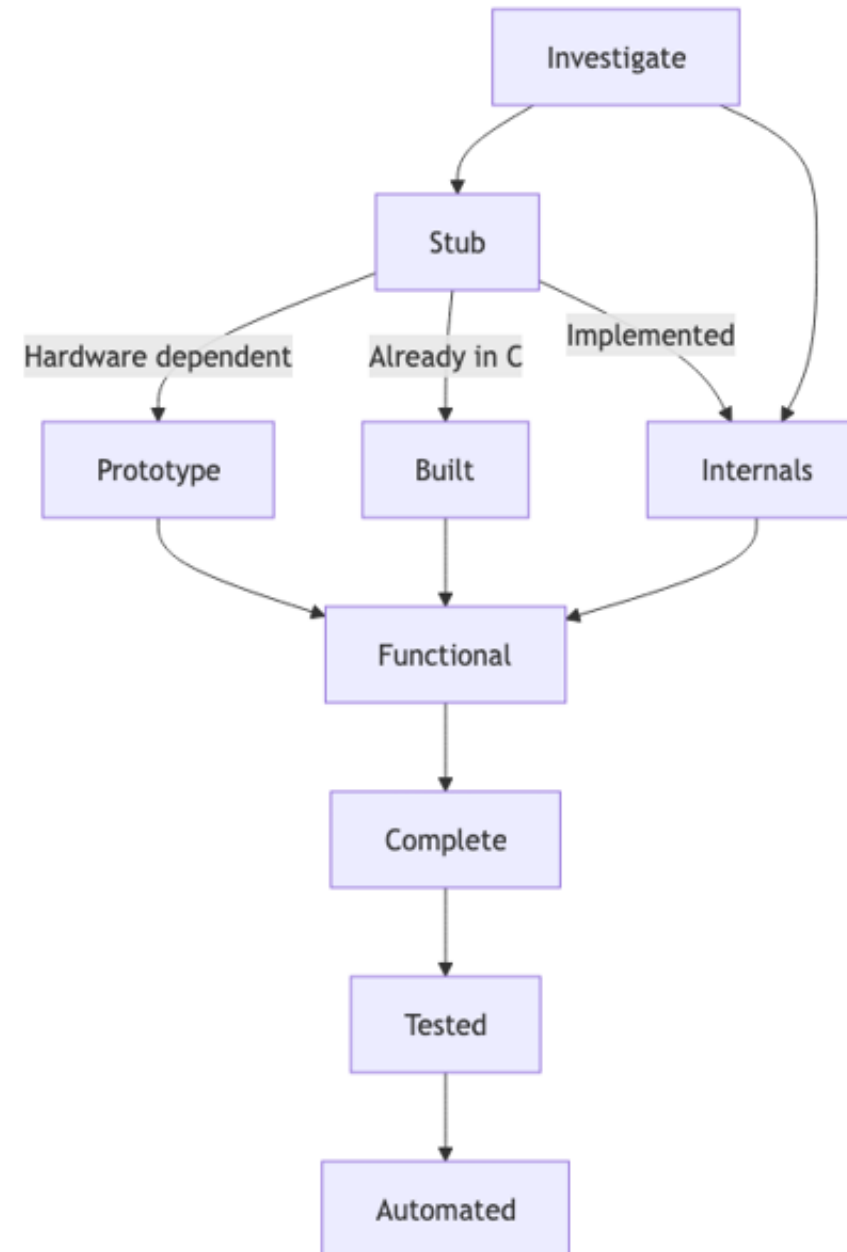
- Original assembler components - which need a complete re-write.
- Libraries and prototypes - a prototype that shows how to build a filesystem is more valuable to developers.
- Original C components - which will need updating to be aware of 64-bit.

Each of these types of components will have a different development path, but we can still track them.



Changing my approach

Software Development Lifecycle





RISC OS 64 status tour



Changing my approach

Communicating the status

- The repository is updated regularly.
- Components and phases have more documentation added as I find I need it.
- Status is visible to all.
- Anyone wanting to comment can open issues, or ask questions.

If it's not sufficient, more detail can always be added.



Changing my approach

Testing new components (1)



riscos64-status / rm64 /

Add file ▾ ...

gerph Add the implementation of a module with dummy SWIs. 99420ff · 2 weeks ago History

Name	Last commit message	Last commit date
..		
ARM,ffa	Added the stub implementation of the ARM module.	3 months ago
BlendTable,ffa	Updated many modules with 64bit versions that work.	4 months ago
BootCmds,ffa	Added BootCmds to the modules we can currently build and work.	3 months ago
BootLog,ffa	Updated with modules now that generic veneers work.	4 months ago
BufferManager,ffa	Updated with modules now that generic veneers work.	4 months ago
Conversions,ffa	Added Conversions module; updated state of OSLib and libs.	3 months ago
ConvertBMP,ffa	Add the Convert* modules which have been built.	3 months ago
ConvertDoom,ffa	Add the Convert* modules which have been built.	3 months ago
ConvertGIF,ffa	Updated status to reflect building of ModGen.	2 months ago
ConvertICO,ffa	Add the Convert* modules which have been built.	3 months ago
ConvertPCX,ffa	Add the Convert* modules which have been built.	3 months ago
ConvertPNM,ffa	Add the Convert* modules which have been built.	3 months ago
ConvertXBM,ffa	Updated status to reflect building of ModGen.	2 months ago
Debugger,ffa	Updated the Debugger module with the v0.08 release version.	2 months ago
DrawFile,ffa	Updated the status of DrawFile; it's built but not working.	3 months ago
ErrorLog,ffa	Updated with modules now that generic veneers work.	4 months ago
FileTypes,ffa	Updated many modules with 64bit versions that work.	4 months ago
IconBorderBeveled,ffa	Updated with modules now that generic veneers work.	4 months ago
IconBorderCrossy,ffa	Updated with modules now that generic veneers work.	4 months ago
IconBorderFob,ffa	Updated with modules now that generic veneers work.	4 months ago
IconBorderLoop,ffa	Update IconBorders and HWScan.	3 months ago



Changing my approach

Testing new components (2)



riscos-test summary				
	Tests	Passed	Skipped	Failed
JUnit Test Report	136 ran	128 passed	8 skipped	0 failed

Test	Result
JUnit Test Report	
<i>absolutes (aarch32)</i>	
BandLimit	passed
BootMenu	skipped
CMunge	passed
CheckMouse	passed
ClrMonitor	passed
FileCoreCk	passed
ForHooks	passed
ImgConvert	passed
Kitten	passed
MD5Hash	passed
MiniGrep	passed
MiniJTran	passed
MiniUnzip	passed
MiniZip	passed
ModGen	passed
ModServices	passed
VTranslate	passed
bison	passed
flex	passed
perl	passed
sed	passed

VideoTTX	passed
WimpTemplates	passed
ZLib	passed
<i>modules (aarch64)</i>	
ARM	passed
BlendTable	passed
BootCmds	passed
BootLog	passed
BufferManager	passed
Conversions	passed
ConvertBMP	passed
ConvertDoom	passed
ConvertGIF	passed
ConvertICO	passed
ConvertPCX	passed
ConvertPNM	passed
ConvertXBM	passed
Debugger	passed
DrawFile	passed
ErrorLog	passed
FileTypes	passed
IconBorderBeveled	passed
IconBorderCrossy	passed
IconBorderFob	passed
IconBorderLoop	passed
IconBorderPlain	passed
IconBorderPlainPopping	passed
IconBorderSkins	passed
KeyInput	passed



Changing my approach

Testing without an OS?

- We have an OS - we have two.
- We have RISC OS Classic - for legacy.
- We have RISC OS Pyromaniac - for 32-bit and 64-bit.
- We have the RISC OS Build service to test things.

This gives us a means to test manually and automatically relatively easily.



Changing my approach

Developing for a new OS?



Developing for a new OS needs tools...

- Norcroft C compiler only targets ARM.
- Better compilers exist which have mass usage and development.
- GCC can be made to work for RISC OS.
- If we remove the RISC OS-specific needs, it's even easier.

The Docker image that I've created allows building of RISC OS components in both 32-bit and 64-bit very simply.





Build environment demonstration



Changing my approach

Native languages



Cross-compiling is good, but we also want languages running in RISC OS...

- **SDL BASIC** - rudimentary port.
- **MyBASIC** - lightweight interpreter port.
- **SED** - it's a language, honest.
- **PocketPython** - Python for small systems.
- **Perl** - yeah, I like Perl.
- **HUProlog** - might as well have a functional language.
- **Lua** - rudimentary port, but surprisingly easy.
- **PicoC** - C interpreter. I've even added `_swix`.
- **Berry** - a small scripting language.
- **TCC** - C compiler, targetting AArch64.





RISC OS 64 language demonstration





4. Conclusion





Conclusion

I've talked about many how the RISC OS Pyromaniac project has evolved.
I've talked about what I'm trying to do to address the issues I see.

- Built up a plan.
- Communicated what's being done.
- Explaining the details.
- Videos to show development and build a community.
- Tools and testing environments for people to use.

And I'm doing this because I hope to encourage others.



Conclusion



Are *you* interested in being involved?





Questions

I'll take any questions that people have.

Slides and Info: <https://presentation.riscos.online/pyromaniac4/>

