GeckOS – a Unix-like 6502 operating system

Glenn Holmer

VCFMW, 2019-09-14

Speaker Bio

- ✔ a.k.a. "Cenbe"
- ✔ retired Java programmer/Linux sysadmin
- ✔ collector of programming languages and operating systems for the Commodore 64:

  `https://www.lyonlabs.org/commodore/`

Happy 50th, Unix!

wait... “Unix” on a 6502?

ARE YOU CRAZY?

wait... “Unix” on a 6502?

Multi-tasking on a 6502 faces significant obstacles:

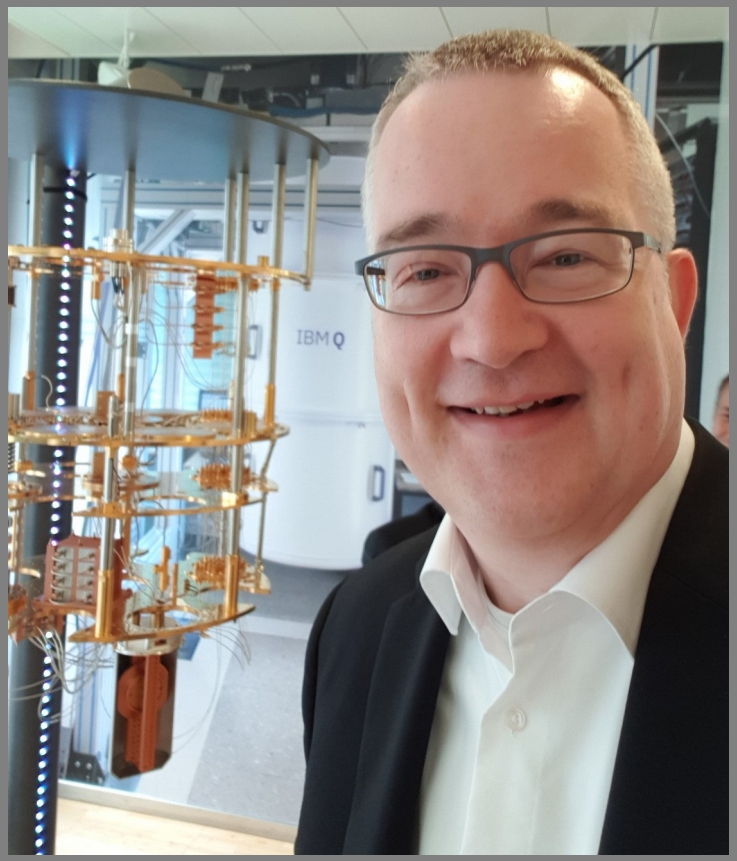
- ✔ typically no hardware memory management
- ✔ no hardware process protection (“ring 0”)
- ✔ limited number of registers
- ✔ single, fixed-location, 256-byte stack

“Unix” on a Commodore 64?

- ✔ It’s been tried with varying degrees of fidelity, e.g. GeckOS, LUnix, Asterix, ACE…
- ✔ None of these are still being developed; most developers are no longer active in the Commodore community.
- ✔ GeckOS seems the most complete, most Unix-like and easiest to work with.

GeckOS history

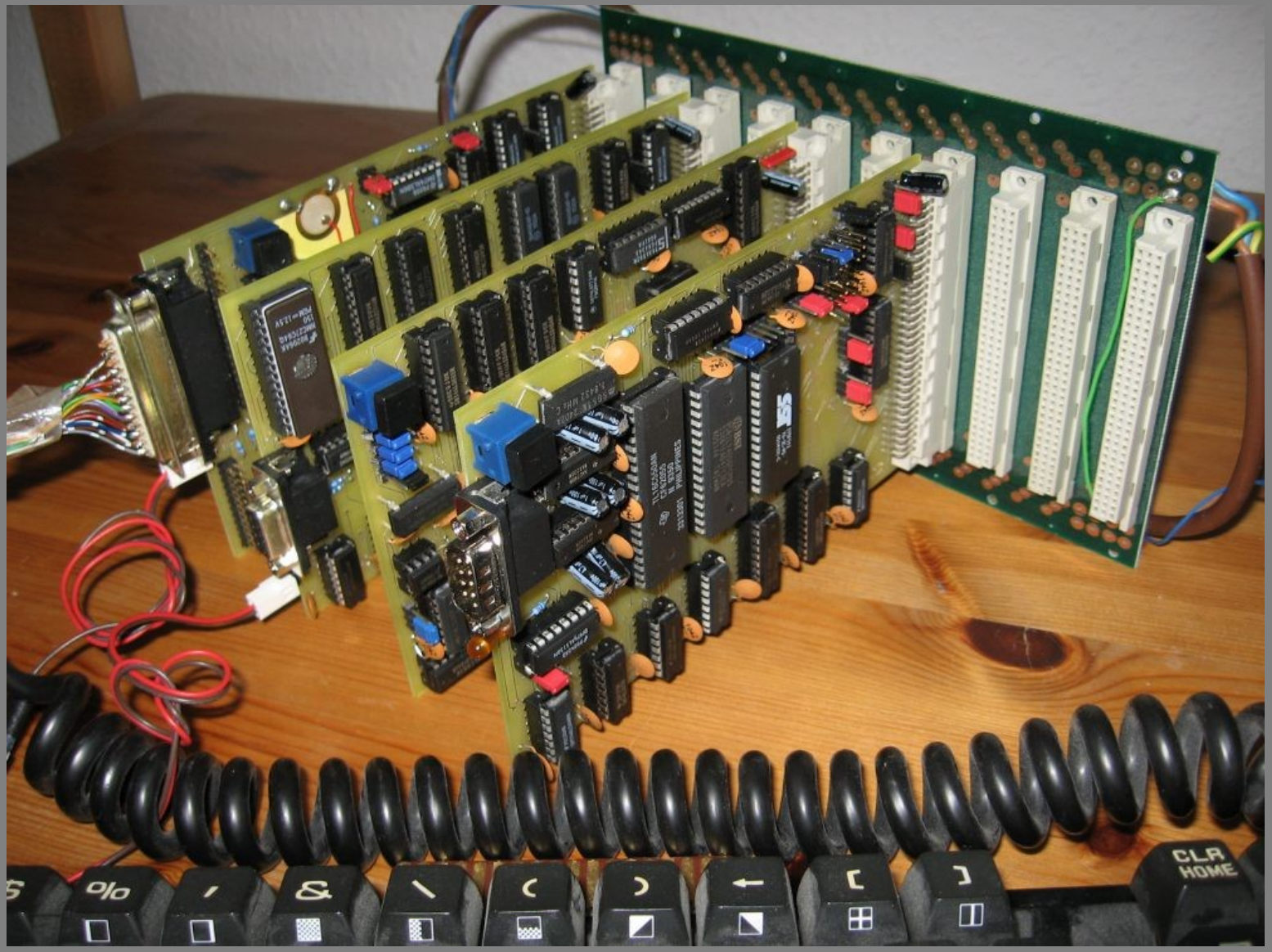
- ✔ Written by André Fachat, originally for the CS/A65 (a 6502 computer with a MMU that he built in 1989).
- ✔ Later expanded to run on other architectures (PET, Commodore 64).
- ✔ 2.0.9 released in 2013.
- ✔ Source available (GPL V2).



André Fachat

CS/A65

http://www.6502.org/users/andre/csa/index.html



GeckOS features

- ✔ preemptive multi-tasking with priorities, multi-threading (max. 12 tasks, 12 threads)
- ✔ signals, semaphores
- ✔ redirection, piping, environment variables
- ✔ a standard library (lib6502)
- ✔ cross-assembler “xa”
  - ✗ use 2.1.4h with GeckOS (not newer versions)
  - ✗ output is o65 relocatable file format
  - ✗ can produce label xref with addresses

Cenbe's Commentary on GeckOS

I've been working on an analysis of GeckOS for those who would like to follow along at home:

`https://www.lyonlabs.org/commodore/onrequest/geckos-analysis.html`

- ✔ source layout
- ✔ system initialization
- ✔ IRQ service routine
- ✔ forking new processes
- ✔ scheduler, task switching
- ✔ running programs from the shell

So how does GeckOS do a task switch?

- ✔ An interrupt is generated every ~20ms by CIA 1 timer B to run the scheduler.
- ✔ The stack is split into two parts: 192 bytes for the kernel, and 64 bytes for user threads. There is a save buffer for each thread's stack.
- ✔ To switch between user space and kernel space, the user and system stack pointers are swapped.
- ✔ During a context switch, the current thread's stack is saved and the new thread's is swapped in.

DEMO

- ✔ shell (both), monitor
- ✔ forking (one program loads and runs another)
- ✔ backgrounding a program (“the Schema demo”)
- ✔ signals (sending messages between programs)
- ✔ semaphores (blocking on available resource)

forking

```
lda #<forkstrc
ldy #>forkstrc
jsr forkto  ;returns child PID in .X

forkstrc
.byt STDIN,STDOUT,STDERR,"forked",0,0
```

signals

sending:

```
lda #SIG_USR1
ldx childpid
sec
jsr SENDSIG
```

receiving:

```
lda #<sigresp
ldx #>sigresp
sec
jsr SETSIG
lda #SIG_USR1
clc
jsr SETSIG
```

semaphores

locking:

```
ldx #SEM_CENBE
sec                 ;clc blocks until free
jsr PSEM            ;returns E_OK or E_SEMSET
```

freeing:

```
ldx #SEM_CENBE
jsr VSEM
```

What can I do with GeckOS?

- ✔ Hack on it! Big fun!
- ✔ Learn about operating systems
- ✔ Write a killer app!
- ✔ but first...

possible extensions/improvements

- ✔ ctrl-C in shell to end wayward program
- ✔ store program names in process table
- ✔ find a way to retrieve program exec address
- ✔ write a ps command for `lsh`
- ✔ Grand Unification of the Shells
- ✔ add devices: CMD HD, REU (filesystem?), μIEC and 1541 Ultimate support
- ✔ 1541 Ultimate networking
- ✔ native speeder in the filesystem?

resources

- GeckOS (source, tools, docs, disk images):
  `http://www.6502.org/users/andre/osa/index.html`
- online HTML documentation:
  `https://www.lyonlabs.org/commodore/onrequest/GeckOS-docs/index.html`
- Cenbe's Commentary on GeckOS:
  `https://www.lyonlabs.org/commodore/onrequest/geckos-analysis.html`

QUESTIONS