ODYSSEY SPACE RESEARCH

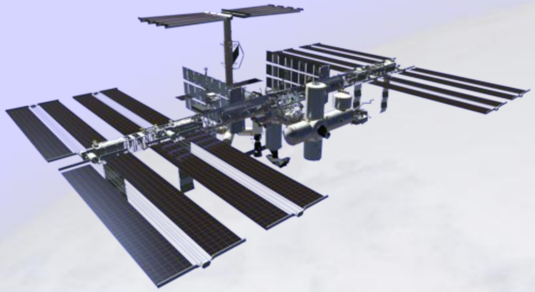
Lessons Learned: cFS on Linux and RTEMS

Allen Brown, Thadeus Fleming
Odyssey Space Research, LLC

2018 Workshop on Spacecraft Flight Software
Southwest Research Institute
San Antonio, Texas

Company Background

- Odyssey Space Research LLC
  - Established 2003
  - Houston TX & Denver CO
- Core Areas
  - GN&C algorithms, design, analysis, integration, evaluation, test
  - Flight software development, integration, test
  - Simulation development, integration
  - Trajectory / Mission design, analysis
- Current Principal Projects
  - Orion Multi-Purpose Crew Vehicle
  - Commercial Crew
  - Commercial Resupply Services 1 & 2
  - ISS Visiting Vehicle Integration
  - Exploration Mission
    - Analysis and design
    - Flight software
  - Flight dynamics for mission operations
  - Satellites: LEO and beyond



Company cFS Areas

- NASA Support
  - Integrated FSW Simulations
    - Project Gateway
    - Moon Mission
    - ...and others
  - New cFS platforms (Xenomai, ARINC 653)
  - Orion BFS
  - cFS Voting Architecture
  - Distributed cFS Integration
  - New cFS Apps/Libs

- Commercial Applications
  - Integrated FSW Simulations
  - DoD test satellite
  - Science Satellite
  - ...and others
  - CFDP cFS Ground Node

- Internal Research & Development
  - Integrated FSW Simulations
  - Human-in-the-Loop Flight Mockup (displays, vehicle & environment sim, cFS FSW)

Company cFS Areas

- Full-stack development
  - BSP, custom drivers, PSP, OSAL
  - Custom cFS applications, libraries
  - C&DH, GNC, and more
- Ground dev/test and operations support
- cFS Training & Consulting
  - Internal and for commercial customers upon request
    - Training classes and materials
    - Templates, guidelines, HOWTO's

FSW development opportunities growing
Government and Commercial applications
cFS and custom solutions

Linux Lessons Learned

Real-time Threads without `root`

- CFS on Linux often run as `root` to enable real-time threads and thread priorities
- Sufficient, but not necessary: Linux allows real-time scheduling via other means
  - Capabilities (`CAP_SYS_NICE`)
  - Resource limits (`RLIMIT_RTPRIO`)
- Principle of least privilege suggests using one of these methods instead of running as root

Real-time Threads without `root`: Which Method?

- `CAP_SYS_NICE`
  - Pro: less change to CFS code
  - Con: not as easy to use in development
    - Set per file, cleared if file is replaced (e.g. recompiled)
    - Setting effectively requires `root`
- `RLIMIT_RTPRIO`
  - Pro: easier for development use
    - Can be set per-user with common `pam_limits` tool
    - Recompiling doesn't affect it
  - Con: requires small PSP patch
    - Must request raise to limit for running process
- OSR is testing `RLIMIT_RTPRIO`

Real-time Threads without `root`: A Hitch

- CFS (POSIX OSAL) assumes `root` is needed
- Will not try to set priorities if `geteuid() != 0`
- POSIX doesn't specify what, if any, permissions are needed
- Cannot portably check for permissions
- More portable to try it and see
- POSIX OSAL needs some patches
- `posix-ng` OSAL does the right thing

Multicore Scheduling on Linux

- Linux, like many other OSes, implements a separate run queue per CPU
- Realtime priorities only used to order processes per queue, **not** globally
- Strict task ordering by descending priority NOT guaranteed when tasks are scheduled on different CPUs
- Restricting CFS to one CPU will provide the expected behavior

Running CFS alone on a CPU

- Improved real-time performance vs. scheduled with other processes
- e.g. run on 4th CPU of 4 (index 3)
- Kernel command line parameters
  - `isolcpus=3`
    - Exclude the CPU from normal load balancing
    - Deprecated in recent kernels in favor of cpusets, but easier to use
  - `irq_affinity=0-2`
    - Send interrupts to other CPUs
    - Not supported by all IRQ controller hardware
- Start CFS on CPU 3 with `taskset`
  - `taskset -c 3 ./core-linux`

Smaller Linux Patches

- `pthread_setname_np` integration
  - Linux API to set a thread name, similar API on BSDs
  - Visible e.g. in debugger
  - OSAL tasks already have names
  - Add to `OS_TaskCreate` to associate task name with thread
- Protect ES PerfLog with semaphore
  - Symptom: data corruption in performance logs
  - Multi-thread issue: ES tries to lock interrupts; impossible on Linux
  - Protect with an OSAL semaphore instead

Smaller Linux Patches

- Sub-microsecond timestamp resolution in PSP TimeBase API
  - Used in CFE ES PerfLog
  - Linux exposes nanosecond-resolution timestamps
  - PSP uses an OSAL function which rounds to 1 μs
  - Fix: use the `clock_gettime` function directly instead, tweak resolution parameters appropriately
- Fix for message queue leak
  - Call `mq_unlink` immediately after `mq_open`
  - Implemented in `posix-ng`

APIs for Potential Future Use

- `procfs`
  - Linux virtual file system
  - Exposes many system statistics
  - e.g. `/proc/stat` has both per-core and aggregate CPU usage info; may be useful in Health & Status app
- `dl_iterate_phdr`
  - Linux extension to inspect dynamic libraries
  - Could be used to implement missing POSIX OSAL features
    - `OS_SymbolTableDump`
    - `OS_ModuleInfo`
  - Also implemented by some BSDs

cFS on RTEMS

Lessons Learned and Software Updates

cFS & RTEMS Deployment

- Mission:
  - Cobham UT700 LEON3FT 32-bit SPARC v8 processor
  - cFS: 6.5.0a (released cFS applications, etc.)
  - Objective: Minimal changes to cFE, existing cFS apps
  - RTEMS pre-5.1, goal: RTEMS 5.1 official release
  - RTEMS OSAL
  - Custom: BSP, PSP, cFS custom mission-specific apps
- Dev env:
  - Linux on x86-64 (pc-linux PSP & POSIX OSAL)
  - Smoke tests with full stack on QEMU/LEON3
    - RTEMS, BSP, PSP, RTEMS OSAL, cFE, cFS apps (limited I/O, storage)
  - LEON3 dev hardware running full stack

cFE Updates

- cFE 6.5.0a open-source release
    - Bugfix: CFE_ES_ShellOutputCommand()
        - was limited to 4 chars, fixed to support CFE_ES_MAX_SHELL_CMD
    - ccsds.h command secondary header assumed uint16
        - Alignment-sensitive platform: tweaked to be uint8[2] and updated macros
    - Bugfix in cFE SB unit test & minor tweak to unit test #includes

cFS Application Updates: CF

- CF (CFDP File Transfer), starting from v2.2.1
  - Made configurable: incoming PDU message limit
  - Fixed HK throttling semaphore count, supports all OSALs
    - OS_CountSemGetInfo() doesn't always return count (POSIX, RTEMS tested)
  - Added wrappers to call OSAL/PSP, not direct POSIX calls
    - printf ->OS_printf, fopen -> OS_open, fread, fwrite, stat, etc.
    - time() -> CFE_PSP_GetTime()
  - Bugfixes and cleanup, added unit tests
    - Fixed endian assumptions and data alignment issues (Babelfish 11?)
      - Programming assumptions vs processor restrictions (x86 vs. LEON3)
    - Fixed PDU Checksum length error (Babelfish 101?)
    - Removed default behavior "assert calls exit()"
    - Fixed many build warnings

cFS Application Updates: HS

- Health & Safety (HS), starting from v2.3.1
  - Added RTEMS HS custom layer, no core app changes
  - CPU Utilization for app HK & CPU hogging detection
  - Commands: report per-thread CPU utilization via events
    - Single thread or all threads
  - Created an RTEMS API for thread CPU utilization

cFS Application Updates: MD, MM

- Memory Dwell (MD), v2.3.1
  - MD_AppData is in header, not source
  - Caused multiple-defined symbols error on LEON3 linker
    - But not on Linux linker

- Memory Manager (MM), starting from 2.4.1
  - Bugfix in MM_DumpMem16ToFile() & MM_DumpMem32ToFile() had incorrect stride when dumping memory
  - Noted assumptions on 2-byte and 4-byte sized arguments and config values

RTEMS OSAL Update

- RTEMS 5.x, single processor
  - Moved from 4.11.x
- cFE/cFS loading support with RTL
  - With RTEMS OSAL actively preventing missing symbols
  - Ops rule: No unloading/reloading cFS apps
- Supporting cFE 6.5.0a interrupt locks
  - cFE: ES performance monitor, TIME, etc.
  - Tested with OSAL INT locks & task preemption
- Closed out development

New cFS Support Tools

- MMTool
  - Creates MM load files from binary blobs
  - Useful for loading/patching
- FileCRCTool
  - Generates CRCs on files, or sections of files
  - CRC matches cFE ES CRC
  - Useful for comparing to FM file CRC and CS one-shot CRC for memory regions
- pc-linux PSP that syncs with the Linux clock
  - Useful for non-RTOS cFS deployments that need to be in sync with Linux system time

RTEMS+LEON3+cFS Lessons Learned

- Data alignment: critical to get right
  - LEON will error with incorrect alignment
    - x86 is less restrictive, (too) easy to develop unportable code
  - Developed data alignment guidelines for cFS app devs
    - Make alignment explicit with OS_ALIGN(n), make padding explicit
    - Use compile-time check for assumptions, used fixed-width types
  - Used compiler warnings on alignment & implicit padding
  - RTEMS OSAL, PSP, BSP - under our control
  - cFE: SB messages assumed to be 32-bit aligned
    - cFS apps must ensure
    - Some cFE messages have 16-bit natural alignment (cast-align warnings)
  - Beware 64-bit types in messages, tables on 32-bit cFE
  - MM app: The MM_MEM32 is operationally critical

RTEMS+LEON3+cFS Lessons Learned

- Dev env: VMs under configuration control
  - Build bit-identical binaries
- Use the same compiler version on all platforms
  - Kept Linux host GCC same version as RTEMS GCC
- Use all the compiler warnings you can, early
- RTEMS vs cFS conventions: task names
  - 4-char names vs longer cFE/cFS names (RTEMS OSAL map cFE)
- Coordinate your task priorities system-wide
  - RTEMS tasks, OSAL shell task, cFE tasks
  - cFS app main tasks and child tasks
- Optimization (-O2)
  - Affects in-memory tar FS (rtems_bin2c)
  - cFS tables need OS_USED for elf2cfetbl

RTEMS+LEON3+cFS Lessons Learned

- cFS+RTEMS RTL needs embedded symbol table
  - Used two-step link process to embed
- Some linker “help” still required
  - A few additional symbols must be given to linker: libm support, strcat, etc. (cFS app support)
  - Optimization: may have to include entire lib (tar FS)
- Don’t leave Earth without your map file
- Console writes: system performance impact
  - Weaning off all that debug goodness on a short schedule?
  - In-memory log solutions: printk(), OS_printf()

RTEMS+LEON3+cFS Lessons Learned

- Always have a SIL with command/telemetry
  - cFS on Linux handy for development
  - But need full-stack SIL for dev testing
- Need engineering UI early
  - Support all dev/test platforms
  - Full cFE/cFS command/telemetry set before custom apps
  - Full-fledged scripting capabilities: test automation, checkout support
- Using CFDP?
  - Have a CFDP peer to support dev & test - early
  - We used pc-linux cFS with CF and a cmd/tlm bridge