Caution: this is a mere introduction to this vast subject

MSE | MASTER OF SCIENCE IN ENGINEERING

MA_EmbReal

Robust Patterns for Reliable Systems (II)

Version: 1.3

Serge Ayer – Luca Haab | 24.04.2023 | Cours MSE

Recall: Our Mission

Recall: Our mission

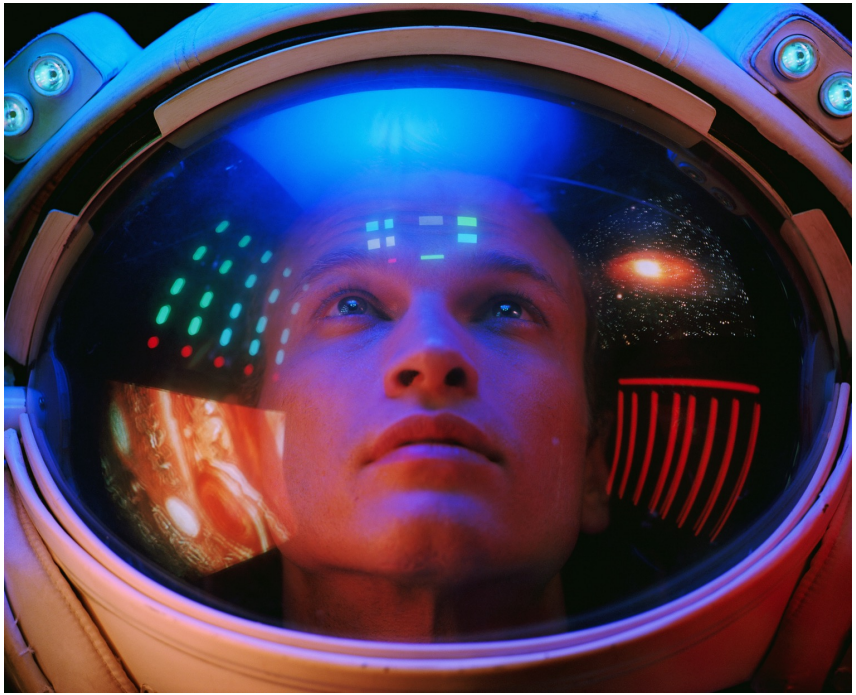
- Program with a mix of periodic / aperiodic tasks
  - Address first scheduling of periodic tasks
  - Add aperiodic tasks
  - Add dependencies among tasks
- Demonstrate that a schedule is feasible given a set of tasks with their constraints and dependencies
  - Use known bounds and elaborate a feasible schedule
  - Compute bounds for blocking times
- Use the appropriate scheduling algorithm in simulation and practice
- Implement a system that meets timing constraints
  - **With functional safety concepts**
  - With timing constraints watchdogs

Gobblers

- A gobbler is an application that consumes too many resources
- A safe system needs to guard against gobblers

Pacman: https://commons.wikimedia.org/wiki/File:Pacman_HD.png

Isolation – Spatial / Temporal



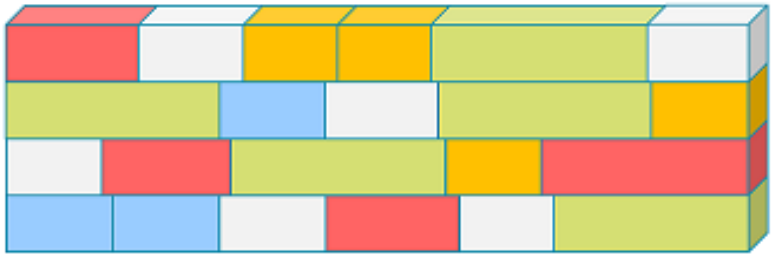
Memory (+ related Spatial Isolation)

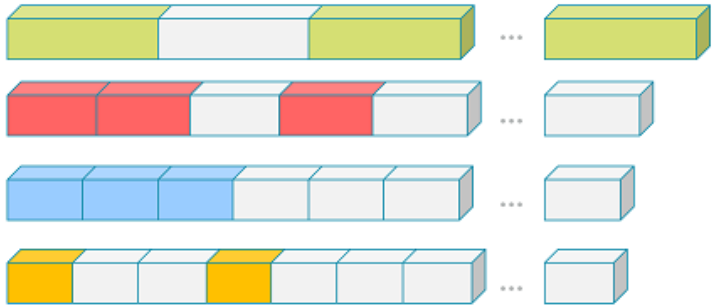
Memory - Challenges

- Fragmentation
- Memory usage
- Overflows

Memory Fragmentation - Pools

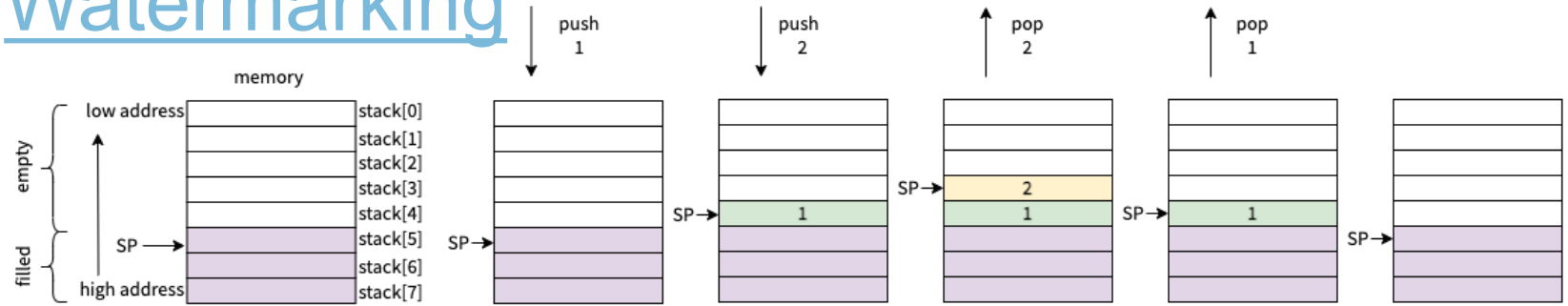
- Global
  - Statically allocated block of memory
  - No distinction what is stored in the pool
- Specific
  - Statically allocated block of memory
  - Each type has its dedicated memory pool





Stack – Memory usage

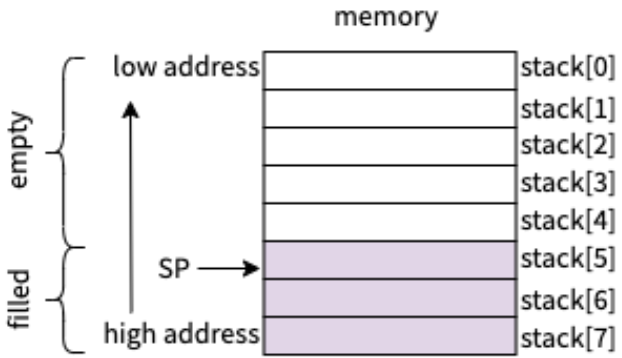
- Watermarking



Implementation example: svcRtxThreadNew@https://github.com/ARM-software/CMSIS_5/blob/master/CMSIS/RTOS2/RTX/Source/rtx_thread.c

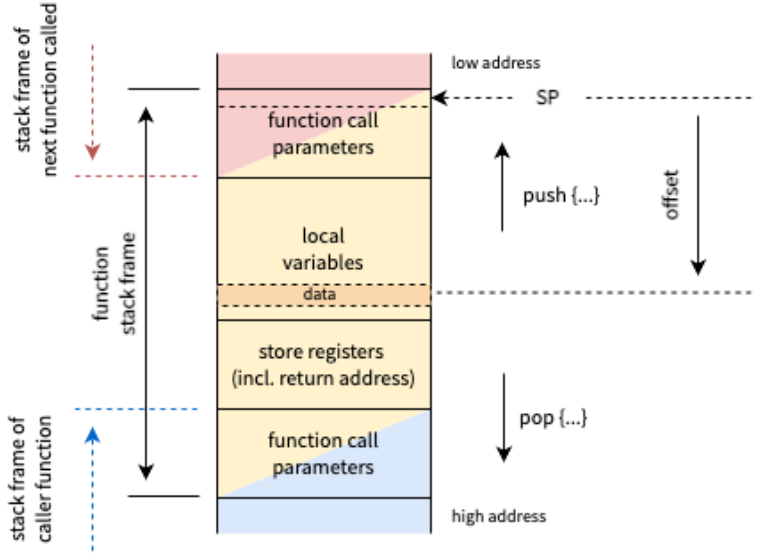
Stack – Buffer overflow

- Overflow Checking




```
void aFancyFunction(){
    int aVarIndex = 0;
    int aVar[3];
}
```

Stack – Buffer overflow



- Application crashes
- Crashing neighbouring buffers
- Changing return address
- Modifying buffer contents
- …

Stack – Buffer overflow

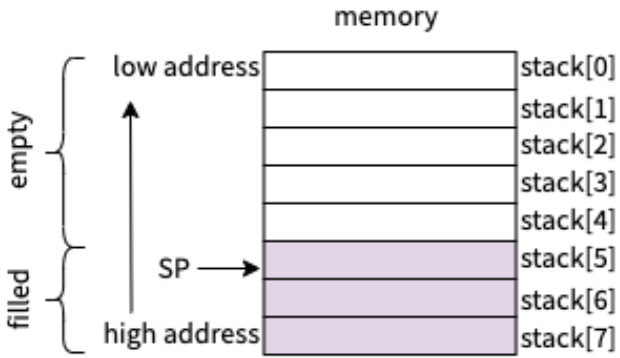
- Let's put this into practice: Leveraging-stack-overflow (20')

Stack – Buffer Overflow

- Overflow Checking

Proactive

Reactive



- Mark memory as non-executable
- Use Address Space Layout Randomization (ASLR)

Tasks (+ related Temporal Isolation)

Tasks - Monitoring

- First thing first:
  - Detect a fault happening
- Apply pattern(s)
  - Watchdog – the simplest

Monitoring Tasks - Watchdog

- A task needs to refresh a watchdog
- A consequent action is triggered if not

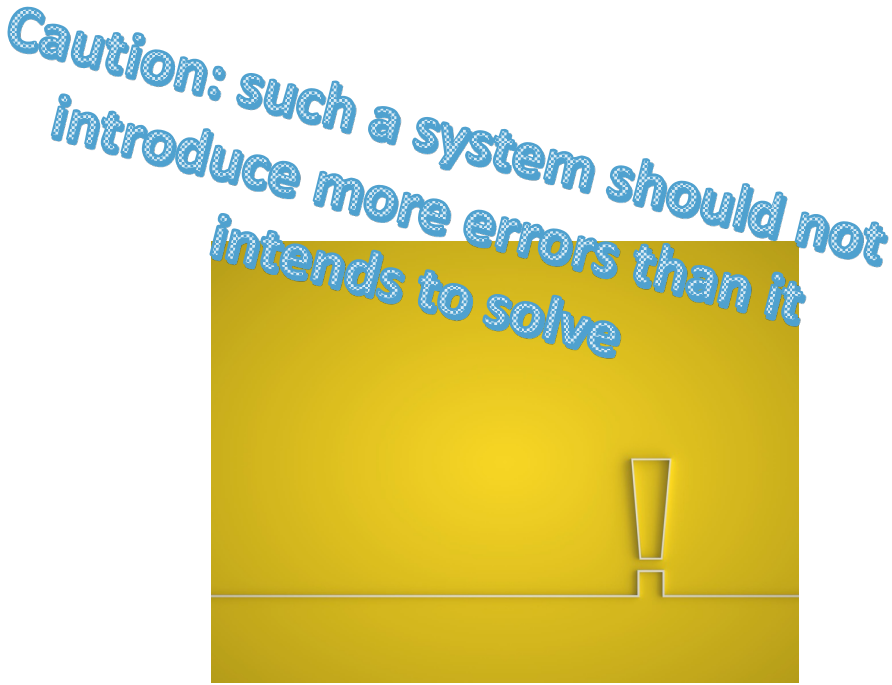
- Note: multiple tasks with different cadences may undergo watchdog scrutiny

Managing Capacity

- A feasible system, has a defined spare capacity
- Baseline to be set
- At every version, baseline is checked
- Profiler

Auditing

- Errors may go unnoticed…
- “Correctable” errors such as
  - orphaned resources
  - indices in data
  - inconsistent states
  - …
- Likely lead to faults eventually
- Audit corrects errors and helps to make a system self-healing



Flight Recorder

It

- Ensures that important logs survive restarts
- Implemented in a non-volatile memory segment

It contains

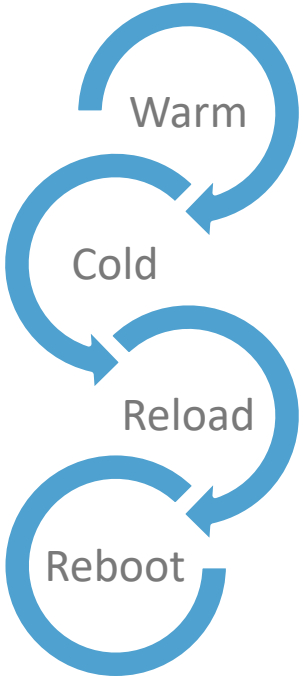
- Error and warning logs (on & off)
- Logs of serious software problems
- Service-affecting actions performed by support technicians

Putting all together

Escalating Restarts

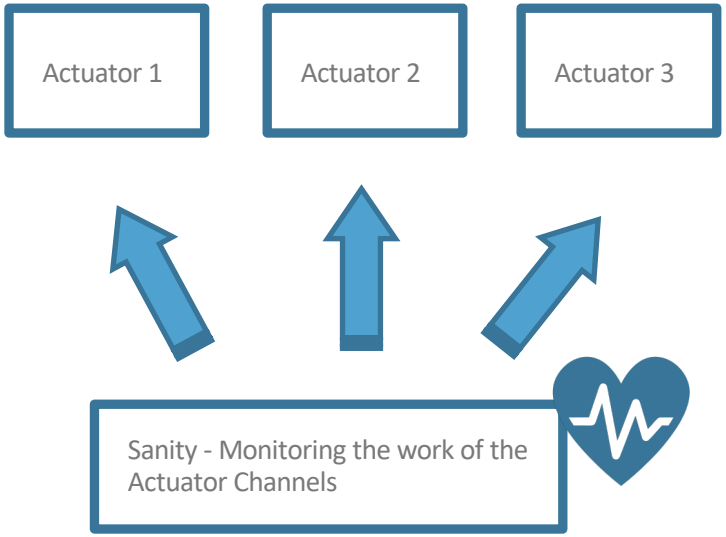
- A failure does occur, so?
  - What is the appropriate consequent action?
  - *Restart* means reinitializing memory

Escalating Restarts



- First: (try to) do no harm
- Then: escalate until the issue is fixed

Escalating Restarts



- Classify importance & dependencies
- Monitor execution
- Apply appropriate escalation

Examples: reincarnation server of Minix, Linux's SystemD

References

- Robust Communications Software - Extreme Availability, Reliability and Scalability for Carrier-Grade Systems, Greg Utas (ISBN 0-470-85434-0)
- Patterns for Fault Tolerant Software, Robert S. Hanmer (ISBN: 978-1-118-35154-3)
- The Architecture of a Reliable Operating System (https://www.cs.vu.nl/~ast/Publications/Papers/asci-2006.pdf)
- On Spatial Isolation for Mixed Criticality, Embedded Systems (https://www-users.york.ac.uk/~rd17/wmc2014/3.pdf)
- Introduction to memory protection unit on STM32 MCUs (https://www.st.com/resource/en/application_note/an4838-introduction-to-memory-protection-unit-management-on-stm32-mcus-stmicroelectronics.pdf)