Real-Time Programming Languages

Labtorial 10: Ada Ravenscar Profile

Martin Becker

TU München
Institute for Real-Time Computer Systems (RCS)

December 15, 2015

Lehrstuhl für
Realzeit-Computersysteme

Technische Universität München

The Ada *Ravenscar* profile

- turns off features of the Ada language
  - e.g., `select`, `abort`, `delay`
- makes certain implementations invalid, e.g., calling of potentially blocking operations from protected entries
- reduces the size of the program
- makes it analyzable and thus certifiable

Using *Ravenscar*

`pragmas` can be added at the top of each file, but that's tedious and there is a better way:

1. create or open your `gps` project
2. Click on *File* > *New*
   - this file will hold all pragmas that we want to use
3. To activate Ravenscar, write `pragma Profile(Ravenscar);`
4. save the file as `config.adc` in your project directory
5. Go to *Project* > *Properties...* > *General* and select your file as *Global pragmas*
6. Save your project.
7. All pragmas in `config.adc` are now activated for all files.

Ravenscar Templates

To write a valid Ravenscar program, follow the code templates from the lecture slides.

- tasks with single release point
- tasks only at package level
- number of tasks must be fixed
- All data shared between tasks must be put in protected objects
  - the *guards* for entry functions must be simple Boolean variables, i.e.
    - `entry Remove(msg: out Item)when not Empty ...` is forbidden, because the guard `not Empty` is rejected, because the variable must be negated
    - instead do this:

      ```
      entry Remove(msg: out Item)when Have_Items is ...
      ```
- the package `Ada.Calendar` is forbidden $\Rightarrow$ all timing must come from `Ada.Real_Time`

Running *Ravenscar* Programs

- Ravenscar requires fixed-priority preemptive scheduling with immediate priority ceiling, i.e., an RTOS
- this means a standard Linux or Windows does not satisfy the requirement
- Pitfall: A Ravenscar program *will* run on a standard Linux, but it behaves incorrectly
- Workaround:
  1. develop your program with the Ravenscar pragma
  2. turn off the pragma and re-compile the program
  3. it now runs correctly on a standard operating system, but the run-time environment is much larger
- ...or install the RT patchset if you are on Linux (see first Ada lecture)

Exercises for Today

1. explore and understand the *blinker example* (code on Moodle)
   - identify those parts, which prevent applying the *Ravenscar* profile
2. re-write your stopwatch program, such that it is Ravenscar compliant