

**BRNO UNIVERSITY OF TECHNOLOGY**

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

Kolejní 4, 616 00 Brno

tel.: +420 5 4114 1113 fax: +420 5 4114 1123

E-mail: [kucera@feec.vutbr.cz](mailto:kucera@feec.vutbr.cz) <http://taceo.eu>



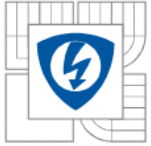
# Real Time Operating Systems

IV.

Pavel Kučera

CAK, E112

[kucera@feec.vutbr.cz](mailto:kucera@feec.vutbr.cz)

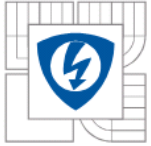


RTOS - IV



# Content

1. Processes in RTX
2. Threads in RTX



# Process

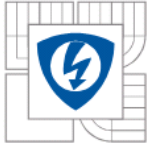
A RTSS process **consists** of:

- an address space,
- object handles,
- one or more paths of execution (threads).

RTSS processes and threads function much like in Win32 environment.



RTSS and Win32 processes and threads can access processes and threads **only** within their own environment.



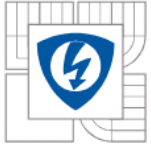
# Process

When a process is **created**, RTSS performs the following tasks:

- loads the executable as a driver,
- allocates process heap from the non-paged pool,
- creates the primary thread.

The maximum number of processes that can exist concurrently in the RTSS environment is equal to the number of RTSS process slots in the registry; the default is 10 slots.

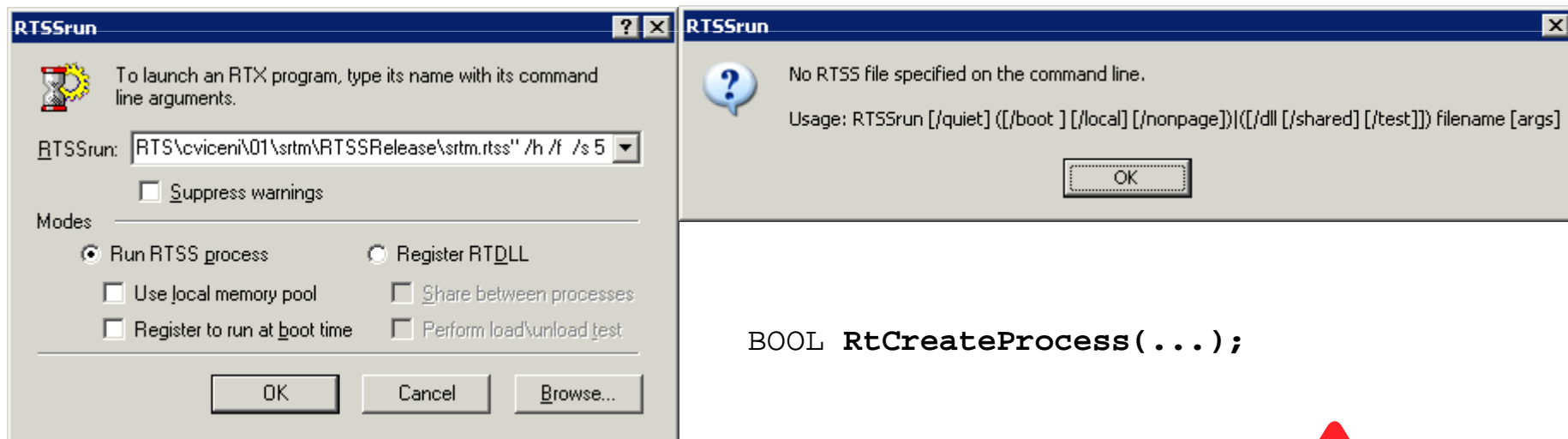




# Process

A process can be **started** by either one of these methods:

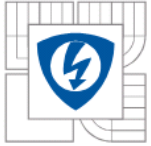
- loading it as a device driver during system boot,
- running the RTSS executable from the command line,
- starting the RTSS process from Win32 applications.



```
BOOL RtCreateProcess(...);
```

RtCreateProcess is supported **only** in Win32 environment.



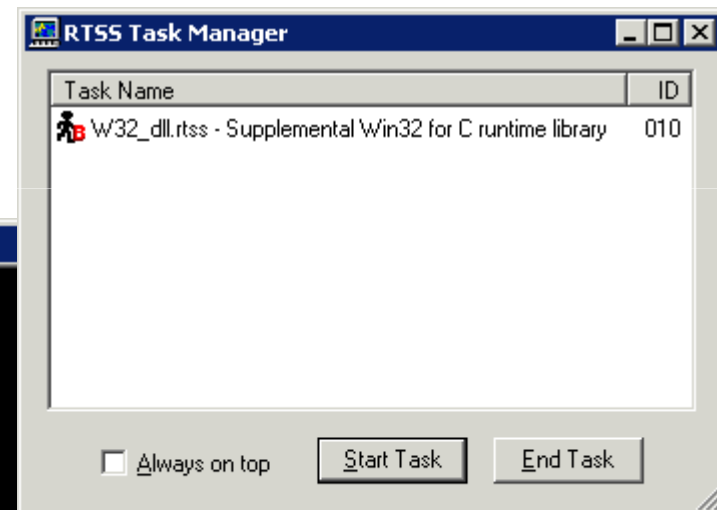


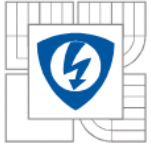
# Process

A process **exits** under one of these conditions:

- the last thread of the process has exited,
- any thread calls `ExitProcess(...)`,
- the process is killed by the:
  - RTSSkill utility
  - RTSS Task Manager

```
C:\WINDOWS\system32\cmd.exe
C:\>rtsskill
RTSSkill - 6.5
RTSS process list:
  PID START STATE COMMAND
  001
  002
  003
  004
  005
  006
  007
  008
  009
  010 /boot R      W32_dll.rtss - Supplemental Win32 for C runtime library
RTSS Registered Dlls:
STATE      NAME
C:\>_
```





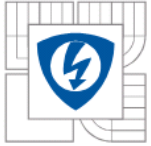
# Threads

A **thread** is created either in RTSS or Win32 environment, depending on the on the current execution environment of the process.

`CreateThread( )` returns handle and thread ID of the created thread.



Handle and thread ID are valid **only** in the `CreateThread( )` caller's environment



# Threads

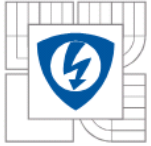
An RTSS thread is the unit of execution in the RTSS environment.

A ready-to-run RTSS thread is scheduled **before** all Windows threads.

The initial thread of a process has an **8 KB** stack.

An RTSS thread runs until it gives up the CPU. A thread gives up the CPU when it:

- waits for a synchronization object,
- lowers its own priority or changes another thread's priority
- suspends itself,
- returns from the timer or interrupt handler (timer and interrupt are threads) routines,
- calls `Sleep( )` with an argument of 0,
- receives a notification that a time quantum is set,
- is interrupted by a higher priority thread.



# Threads

RTSS to Win32 Thread Priority Mapping

RTSS Symbolic Priority Name	RTSS Value	Windows 2000 Symbolic Priority Name for Real-Time Priority Class	Win32 Value
RT_PRIORITY_MIN	0	THREAD_PRIORITY_IDLE	16
RT_PRIORITY_MIN + 1	1	THREAD_PRIORITY_LOWEST	22
RT_PRIORITY_MIN + 2	2	THREAD_PRIORITY_BELOW_NORMAL	23
RT_PRIORITY_MIN + 3	3	THREAD_PRIORITY_NORMAL	24
RT_PRIORITY_MIN + 4	4	THREAD_PRIORITY_ABOVE_NORMAL	25
RT_PRIORITY_MIN + 5...+ 126	5...126	THREAD_PRIORITY_HIGHEST	26
RT_PRIORITY_MAX	127	THREAD_PRIORITY_TIME_CRITICAL	31