

Adaptive Partitioning - Scheduler <#>

Aka APS

What is it?<#>

It's a fair-share thread scheduler which guarantees groups of a user-specified percentage of cpu time when the system is overloaded. When the system is sufficiently underloaded, it chooses to schedule threads based strictly on priority and therefore maintains realtime behavior. Even when overloaded it provides realtime latencies to an engineerable set of critical threads.

Scheduler Partitioning is part of an overall resource partitioning strategy. See [Adaptive Partitioning](#).

Table of Contents<#>

1. Presentation on APS
 - [slide presentation](#): history / use cases / why is it "adaptive" / why is it cool
 - video commentary by the main perp to go with the above slide presentation:
 - slides 1 to 5: [part 1.1](#)
 - slides 6 to 9: [part 1.2](#)
 - slides 10 to 18: [part 1.2](#)
 - slides 19 to end: [part 2.2](#)
2. QNX Designer's Documentation
 - [Requirements Document](#): requirements / APIs / Limitations
 - [How it works FAQ](#): Scheduling / Microbilling/ Algorithms / Overhead / Bankruptcy / Inheritance / Budgets / Security
3. Hacking
 - Where to find the [source](#) code.
 - How to [build APS](#)
 - [APS Buildfile extensions](#)
4. Links to user's documents:
 - [Adaptive Partitioning User's Guide](#)
 - [Installation Notes](#)
 - [Release Notes](#)
 - [APS Command Reference](#)
 - Library reference for [SchedCtl\(\)](#)
 - [APS chapter](#) from the QNX System Architecture guide.
5. Other links
 - A [blog](#) posting about why debugging is better with APS.
 - The wikipedia entry on [Adaptive Partition Schedulers](#)