



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version


Languages 
Deutsch
日本語  Edit links

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#)

[More](#) ▾



Least slack time scheduling

From Wikipedia, the free encyclopedia



This article **does not cite any references or sources**. Please help [improve this article](#) by [adding citations to reliable sources](#). Unsourced material may be challenged and [removed](#). *(December 2009)*

Least slack time (LST) scheduling is a [scheduling algorithm](#). It assigns priority based on the *slack time* of a process. Slack time is the amount of time left after a job if the job was started now. This algorithm is also known as **least laxity first**. Its most common use is in embedded systems, especially those with multiple processors. It imposes the simple constraint that each process on each available processor possesses the same run time, and that individual processes do not have an affinity to a certain processor. This is what lends it a suitability to embedded systems.

Slack time [\[edit\]](#)

This scheduling algorithm first selects those processes that have the smallest "slack time". Slack time is defined as the temporal difference between the deadline, the ready time and the run time.

More formally, the *slack time* for a process is defined as:

$$(d - t) - c'$$

where d is the process deadline, t is the real time since the cycle start, and c' is the remaining computation time.

Applications [\[edit\]](#)

In realtime scheduling algorithms for periodic jobs, an acceptance test is needed before accepting a sporadic job with a hard deadline. One of the simplest acceptance tests for a sporadic job is calculating the amount of slack time between the release time and deadline of the job.

Suitability [\[edit\]](#)

LST scheduling is most useful in systems comprising mainly aperiodic tasks, because no prior assumptions are made on the events' rate of occurrence. The main weakness of LST is that it does not look ahead, and works only on the current system state. Thus, during a brief overload of system resources, LST can be suboptimal. It will also be suboptimal when used with uninterruptible processes. However, like [earliest deadline first](#), and unlike [rate monotonic scheduling](#), this algorithm can be used for processor utilization up to 100%.

Categories: [Scheduling algorithms](#)

This page was last modified on 14 March 2015, at 13:26.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

