**Doc No:**

**Date:** 2016-06-21

**Audience:** LEWG (and SG1)

**Authors:** Pablo Halpern, Intel Corp.  
 [phalpern@halpernwightsoftware.com](mailto:phalpern@halpernwightsoftware.com)

Better Names for Parallel Execution Policies in C++17

**Contents**

1 Abstract 1

2 Changes from R0 1

3 Motivation 2

4 Proposal Overview 3

5 Alternatives Considered 3

5.1 Paint the bike shed a different color 3

5.2 Keep the old names but change the namespace 4

5.3 Put some policies into the std::this\_thread namespace 4

6 Formal Wording 5

7 References 6

8 Acknowledgements 6

# Abstract

The parallel algorithms from the first parallelism technical specification were voted into the C++ working draft in Jacksonville during the March 2016 WG21 meeting. The type of parallelism implemented by each algorithm is specified using an *execution policy* argument of type sequential\_execution\_policy, parallel\_execution\_policy, or parallel\_vector\_execution\_policy. Singleton constants of these types are named sequential (abbreviated seq in the TS but spelled out in the C++ WD), par, and par\_vec, all in the std namespace.

This paper explores the possibility of choosing better names for these execution policies as well as those defined in the *Vector and Wavefront Policies* paper, [P0076](http://www.open-std.org/JTC1/SC22/WG21/docs/papers/2016/p0076r2.pdf), targeted for the next revision of the TS. The goal is to choose appropriate names that do not conflict with other uses of similar names and which are consistent between current and future execution policies.

The changes proposed here are targeted for C++17.

# Changes from R0

The following changes were made during SG1 review Monday morning in Oulu

* Renamed namespace std::execution\_policy to std::execution.
* Changed serial/ser to sequenced/seq.
* Removed the word “execution” from execution policy names and move them into namespace std::execution. Thus we now propose std::execution::parallel\_policy instead of std::parallel\_execution\_policy.

# Motivation

The current WD for C++17 defines three execution policy types with corresponding singleton constants using abbreviated names. In P0076, which was forwarded from SG1 to LEWG in Jacksonville, we propose two more policies, for a total of five policies, each with its corresponding singleton object having an abbreviated name:

|  |  |
| --- | --- |
| **Policy type name** | **Singleton object of that type** |
| sequential\_execution\_policy | seq (TS) sequential (current WD) |
| parallel\_execution\_policy | par |
| vector\_execution\_policy (P0076 only) | vec (P0076 only) |
| unsequenced\_execution\_policy (P0076 only) | unseq (P0076 only) |
| parallel\_unsequenced\_execution\_policy (P0076) parallel\_vector\_execution\_policy (current WD) | par\_unseq (P0076) par\_vec (current WD) |

There are several issues, here. The seq object from the parallelism TS was renamed to sequential in the C++17 working draft. Presumably this is because seq, as an abbreviation, can be confused with “sequence” as in “sequence containers”, especially if it is directly within the std namespace (i.e., there is no std::parallel namespace to disambiguate it). However, it was an initial goal that the singleton objects be given short names, to avoid overly-verbose-calls to parallel algorithms. sequential, while not especially long, is nonetheless more than three times as long as seq.

Another problem with sequential is that it clashes with the proposed unseq object. The seq in unseq does not refer to “sequential” but to “sequenced”. Having sequential and unseq in the same set of names is a recipe for confusion.

The current WD defines the parallel\_vector\_execution\_policy and par\_vec, which are renamed in P0076 to parallel\_unseq\_execution\_policy and par\_unseq. That is because the term vector (and vec) implies certain ordering guarantees that are not implied by parallel\_vector (par\_vec) term. We should not put a term into the next standard that is already known to be misleading.

For all of the above reasons, we need to improve the naming of our execution policies. Since the policies form P0076 are already in the pipeline, we should be considering a naming scheme that encompasses all five policies and, ideally, will continue to make sense as future policies are added.

# Proposal Overview

There are many different potential naming schemes for execution policies, and some decisions can be made independently of others (such as whether we should have a std::parallel namespace). This paper proposes a single, simple set of names. However, see the Alternatives Considered section, below for an in depth discussion of other possibilities.

I propose that the five known execution policies be named as shown in the following table. All singleton token names in the C++17 WD would be put within the std::execution namespace. The execution namespace prevents naming conflicts and ambiguity of meaning for short names like seq and par, but also allows the user to bring them into the current scope with a using directive.

|  |  |
| --- | --- |
| **Policy type name (in namespace std::execution)** | **Singleton token of that type (in namespace std::execution)** |
| sequenced\_policy | seq |
| parallel\_policy | par |
| vector\_policy (P0076 only) | vec (P0076 only) |
| unsequenced\_policy (P0076 only) | unseq (P0076 only) |
| parallel\_unsequenced\_policy | par\_unseq |

# Alternatives Considered

## Paint the bike shed a different color

There is nothing particularly special about the terms I chose for this proposal. I chose a minimalist approach and addressed the issues by changing sequence/seq to sequenced/seq. Here is a (far from exhaustive) list of other possible terms

* For sequential ordering within a single thread:
  + serial/ser
  + ordered/ord
  + strict/strict (“str” would be a bad abbreviation, as it would imply “string”)
* For unsequenced ordering within a single thread:
  + interleaved/intlv (or inter)
  + unordered/unord
* For the execution\_policy namespace
  + parallel
  + exec\_policy
  + parallel\_policy

If anybody wants to advocate strongly for one of these terms, it might be worth having a *short* bike shed discussion and simple vote. (I personally like “interleaved”, but hate the abbreviations.)

## Keep the old names but change the namespace

The conflicting meanings for seq (“sequential” vs. “sequence”) would probably not be an issue if all of the execution policies were within an execution\_policy or parallel namespace. A using directive could be used to avoid being too verbose. This alternative would not address the “sequential” vs. “sequenced” conflict for execution\_policy::seq and execution\_policy::unseq, however.

## Put some policies into the std::this\_thread namespace

Jared Hoberock proposed using std::this\_thread as a way of indicating that an execution policy applies to the current thread only. (See reflector message <http://lists.isocpp.org/parallel/2016/04/0153.php>. Note that the archive discarded all of the newlines from this message, so it is probably better to read my reply, <http://lists.isocpp.org/parallel/2016/04/0210.php>, which includes Jared’s message at the bottom.) According to Jared’s proposal, the five execution policy tokens would be:

std::this\_thread::seq  
std::this\_thread::vec  
std::this\_thread::unordered  
std::par  
std::unordered

It is not clear to me whether Jared would put the execution policy names themselves in the this\_thread namespace (e.g., std::this\_thread::sequential\_execution\_policy) or just the short token names. This proposal has some charm; I like the information communicated by the use of this\_thread. However, I don’t think it adds enough benefit to overcome some of its deficiencies.

My main objection to Jared’s naming proposal is that it implies a certain orthogonality that isn’t really there. Only unordered appears in both namespaces.  The "ordered within a single thread” primitive policy is named seq in the std::this\_thread namespace but par in the other namespace. It is far from obvious that std::unordered expresses any parallelism at all, whereas renaming it to std::par\_unordered would solve that problem and result in five unique names, obviating the need to spread them across two namespaces. There is no single using directive that would make all of the execution policies available without a name conflict.

This use of namespaces would also be redundant if we accept some version of [N4406](http://www.open-std.org/JTC1/SC22/WG21/docs/papers/2015/n4406.pdf), which proposes a mechanism for specifying executors as part of execution policies. Once you have a single-thread executor, there is no need to put a single-thread execution policy into a separate namespace. There are more than two dimensions to the execution policy (concurrency, ordering guarantees, and type of threading, to name three), and they are not orthogonal, so any attempt to express them using a naming convention will necessarily be incomplete. For a handful of execution policies, it seems like overkill. I go into much more detail about these objections in [my response](http://lists.isocpp.org/parallel/2016/04/0210.php) to Jared’s reflector post.

# Formal Wording

All section names and numbers are relative to the March 2016 working draft, [N4582](http://www.open-std.org/JTC1/SC22/WG21/docs/papers/2016/n4582.pdf).

In section 20.18.2 [execpol.syn], rename the <execution\_policy> header, add an execution namespace, and rename the execution policies:

**20.18.2 Header** <execution\_policy> **synopsis [exec**ecution**pol.syn]**

namespace std {

// 20.18.3, *execution policy type trait*:

template<class T> struct is\_execution\_policy;

template<class T>

constexpr bool is\_execution\_policy\_v = is\_execution\_policy<T>::value;

namespace execution {

// 20.18.4, *sequentialsequenced execution policy*:

class sequential\_~~execution~~sequenced\_policy;

// 20.18.5, *parallel execution policy*:

class parallel\_~~execution\_~~policy;

// 20.18.6, *parallel+vectorunsequenced execution policy*:

class parallel\_vector\_~~execution~~unsequenced\_policy;

// 20.18.7*, execution policy objects*:

constexpr sequential\_~~execution~~sequenced\_policy sequentialseq{ *unspecified* };

constexpr parallel\_~~execution\_~~policy par{ *unspecified* };

constexpr parallel\_vector\_~~execution~~unsequenced\_policy par\_vecunseq{ *unspecified* };

}

}

Throughout the remainder of the WD, perform the following replacements:

|  |  |
| --- | --- |
| **Replace occurrences of:** | **with:** |
| sequential\_execution\_policy | execution::sequenced\_policy |
| parallel\_execution\_policy | execution::parallel\_policy |
| parallel\_vector\_execution\_policy | execution::parallel\_unsequenced\_policy |
| sequential (when referring to the policy token) | execution::seq |
| par | execution::par |
| par\_vec | execution::par\_unseq |

# References

[P0076R2](http://www.open-std.org/JTC1/SC22/WG21/docs/papers/2016/p0076r2.pdf) *Vector and Wavefront Policies*, Arch Robison, Pablo Halpern, Robert Geva, Clark Nelson, Jens Maurer, 2016-05.

<http://lists.isocpp.org/parallel/2016/04/0210.php>, *A naming proposal by Jared Hoberock and detailed response by Pablo Halpern*, ISO C++ parallelism reflector, 2016-04

[N4406](http://www.open-std.org/JTC1/SC22/WG21/docs/papers/2015/n4406.pdf) *Integrating Executors with Parallel Algorithm Execution*, Jared Hoberock, Michael Garland, Olivier Giroux, 2015-04-10

# Acknowledgements

Thanks to Jared Hoberock for contributing his ideas on this subject.

SG1 and LEWG did a great job in refining this proposal to come up with the best names.