Structure

UnsafeCurrentTask

An unsafe reference to the current task.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```
struct UnsafeCurrentTask
```

Overview

To get an instance of `UnsafeCurrentTask` for the current task, call the `withUnsafeCurrentTask(body:)` method. Don't store an unsafe task reference for use outside that method's closure. Storing an unsafe reference doesn't affect the task's actual life cycle, and the behavior of accessing an unsafe task reference outside of the `withUnsafeCurrentTask(body:)` method's closure isn't defined.

Only APIs on `UnsafeCurrentTask` that are also part of `Task` are safe to invoke from a task other than the task that this `UnsafeCurrentTask` instance refers to. Calling other APIs from another task is undefined behavior, breaks invariants in other parts of the program running on this task, and may lead to crashes or data loss.

For information about the language-level concurrency model that `UnsafeCurrentTask` is part of, see Concurrency in The Swift Programming Language.

Topics

Getting an Unsafe Reference to the Current Task

```
func withUnsafeCurrentTask<T>(body: (UnsafeCurrentTask?) throws -> T)
rethrows -> T
```

Calls a closure with an unsafe reference to the current task.

```
func withUnsafeCurrentTask<T>(body: (UnsafeCurrentTask?) async throws -
> T) async rethrows -> T
```

Instance Properties

```
var basePriority: TaskPriority
```

The current task's base priority.

```
var isCancelled: Bool
```

A Boolean value that indicates whether the current task was canceled.

```
var priority: TaskPriority
```

The current task's priority.

```
var unownedTaskExecutor: UnownedTaskExecutor?
```

The current `TaskExecutor` preference, if this task has one configured.

Instance Methods

```
func cancel()
```

Cancel the current task.

```
func escalatePriority(to: TaskPriority)
```

Escalate the task `priority` of the passed in task to the `newPriority`.

Default Implementations

- Equatable Implementations
- Hashable Implementations

Relationships

Conforms To

`Copyable`, `Equatable`, `Hashable`

See Also

Tasks

`struct Task`

A unit of asynchronous work.

`struct TaskGroup`

A group that contains dynamically created child tasks.

`func withTaskGroup<ChildTaskResult, GroupResult>(of: ChildTaskResult.Type, returning: GroupResult.Type, isolation: isolated (any Actor)?, body: (inout TaskGroup<ChildTaskResult>) async -> GroupResult) async -> GroupResult`

Starts a new scope that can contain a dynamic number of child tasks.

`struct ThrowingTaskGroup`

A group that contains throwing, dynamically created child tasks.

`func withThrowingTaskGroup<ChildTaskResult, GroupResult>(of: ChildTaskResult.Type, returning: GroupResult.Type, isolation: isolated (any Actor)?, body: (inout ThrowingTaskGroup<ChildTaskResult, any Error>) async throws -> GroupResult) async rethrows -> GroupResult`

Starts a new scope that can contain a dynamic number of throwing child tasks.

`struct TaskPriority`

The priority of a task.

`struct DiscardingTaskGroup`

A discarding group that contains dynamically created child tasks.

`func withDiscardingTaskGroup<GroupResult>(returning: GroupResult.Type, isolation: isolated (any Actor)?, body: (inout DiscardingTaskGroup) async -> GroupResult) async -> GroupResult`

Starts a new scope that can contain a dynamic number of child tasks.

`struct ThrowingDiscardingTaskGroup`

A throwing discarding group that contains dynamically created child tasks.

```
func withThrowingDiscardingTaskGroup<GroupResult>(returning: GroupResult.Type, isolation: isolated (any Actor)?, body: (inout ThrowingDiscardingTaskGroup<any Error>) async throws -> GroupResult) async throws -> GroupResult
```

Starts a new scope that can contain a dynamic number of child tasks.