

IE2050



Theseus OS

**An Experiment in Operating System
Structure and State Management**

Group 8
15 May 2024

Workload Matrix

IT22196088

Q Joyal

IT22360496

D R Wickrama Arachchi

- **Process Control**
- **Memory Management**
- **Deadlock Management**

- **A Brief Introduction**
- **System Hardware Requirements**
- **Installation Process**

IT22231628

Sharvajen S

IT22169730

D M T N Dissanayaka

- **User Interfaces**
- **Secondary Disk Scheduling Management**
- **References**

- **State Management**
- **Standard Support**
- **Comparative Analysis of Theseus Operating System**
- **Limitations and Extensions to the Case Study**

A Brief Introduction

Experimental operating system for modularity and state management in modern systems software.

A safe-language OS running in a single address space and privilege level.

Implemented as a collection of small cells inspired by biological cells.

Cell abstraction is present in various forms: a crate at implementation time, a single *.o object file after compile time, and a cell at runtime.

Distinct from monolithic, microkernel, and multikernel designs, requiring no hardware reliance.



Theseus OS

Key Findings

of the Theseus OS research paper

Q Joyal
IT22196088

D R Wickrama Arachchi
IT22360496

Sharvajen S
IT22231628

D M T N Dissanayaka
IT22169730

System Hardware Requirements

Tested On

Intel NUC devices, ThinkPad laptops, and Supermicro servers

Main Requirement

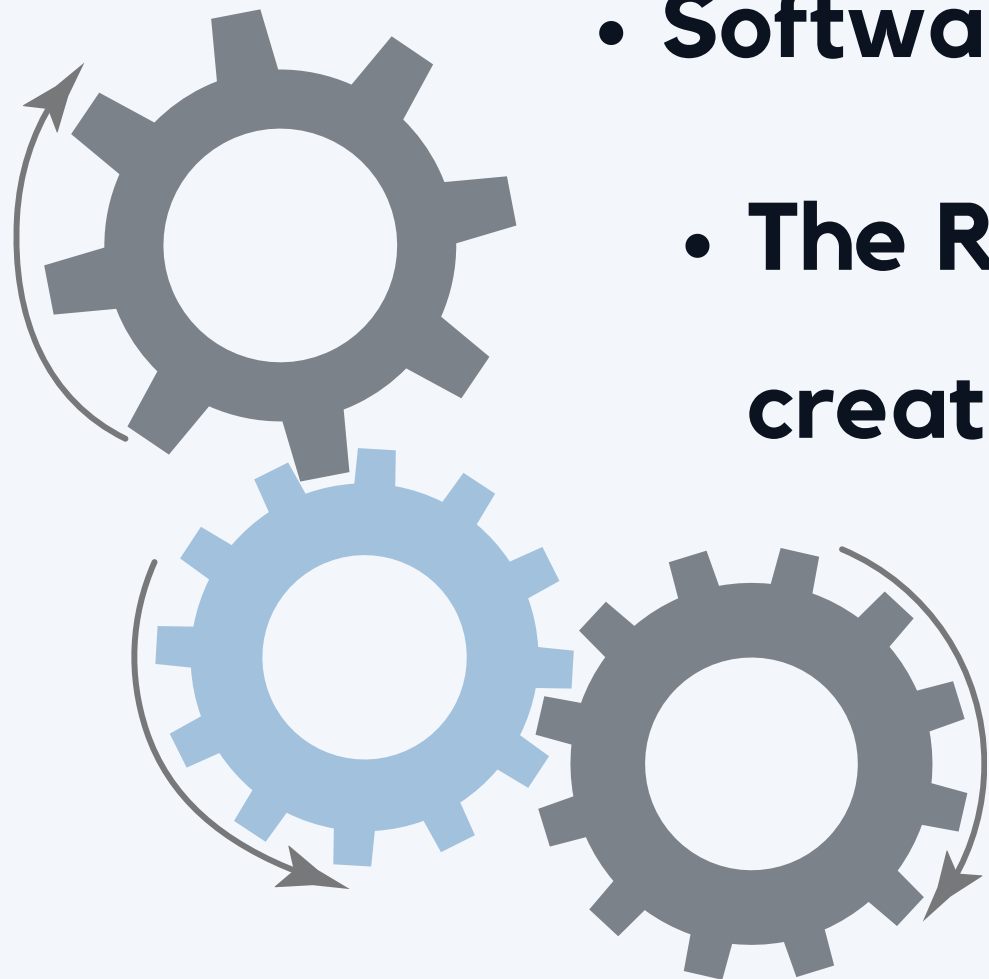
Boot via USB or PXE using traditional BIOS

Designed For

x86_64 architecture

Installation Process

- No standard installation procedures are needed.
- The GitHub repository offers detailed instructions for software building and running.
- Software can be run on Linux, Windows, MacOS, and Docker.
- The README file provides instructions for OS.iso image creation.
- Experiments implemented within the source code.
- Pre-built OS images are available for each setup.



```
Theseus : tmux client — Konsole
[T] kernel/task_struct/src/lib.rs:556: [CPU 2] Task::drop(): bootstrap_task_cpu_0(1)
[I] kernel/spawn/src/lib.rs:188: Cleaned up all 4 bootstrap tasks.
[D] kernel/spawn/src/lib.rs:889: task_cleanup_success: "bootstrap_task_cleanup" successfully exited with return value 0
[T] kernel/task_struct/src/lib.rs:556: [CPU 2] Task::drop(): bootstrap_task_cleanup(14)
[D] kernel/mod_mgmt/src/lib.rs:818: load_crate_as_application(): trying to load application crate at "/namespaces/_applications/ls-1ac6fa56248369a8.o"
[I] kernel/mod_mgmt/src/lib.rs:2866: Symbol "getopts::Options::optflag::he32ebd501bd9d78f" not initially found in namespace "_applications", attempting to load crate "getopts-" into namespace "_applications" that may contain it.
[D] kernel/mod_mgmt/src/lib.rs:855: load_crate: trying to load crate at "/namespaces/_applications/getopts-f60f6f56dd6382cb.o"
[I] kernel/mod_mgmt/src/lib.rs:865: loaded new crate "getopts-f60f6f56dd6382cb", num sections: 83, added 34 new symbols.
[I] kernel/mod_mgmt/src/lib.rs:826: loaded new application crate: "ls-1ac6fa56248369a8", num sections: 37, added 1 new symbols
[D] kernel/spawn/src/lib.rs:721: task_wrapper [1]: "ls-1ac6fa56248369a8(16)" about to call task entry func 0x3f308e0 {fn(alloc::Vec::Vec<alloc::string::String>) -> isize} with arg []
[D] kernel/spawn/src/lib.rs:889: task_cleanup_success: "ls-1ac6fa56248369a8" successfully exited with return value 0
[I] applications/shell/src/lib.rs:1830: terminal: task [16] returned exit value: Some(0)
[T] kernel/task_struct/src/lib.rs:556: [CPU 0] Task::drop(): ls-1ac6fa56248369a8(16)
[T] kernel/crate_metadata/src/lib.rs:288: ### Dropped LoadedCrate: ls-1ac6fa56248369a8
[D] kernel/mod_mgmt/src/lib.rs:818: load_crate_as_application(): trying to load application crate at "/namespaces/_applications/ls-1ac6fa56248369a8.o"
[I] kernel/mod_mgmt/src/lib.rs:826: loaded new application crate: "ls-1ac6fa56248369a8", num sections: 37, added 1 new symbols
[D] kernel/spawn/src/lib.rs:721: task_wrapper [1]: "ls-1ac6fa56248369a8(17)" about to call task entry func 0x3f308e0 {fn(alloc::Vec::Vec<alloc::string::String>) -> isize} with arg ["extra_files/test_files/text/"]
[D] kernel/spawn/src/lib.rs:889: task_cleanup_success: "ls-1ac6fa56248369a8" successfully exited with return value 0
[I] applications/shell/src/lib.rs:1830: terminal: task [17] returned exit value: Some(0)
[T] kernel/task_struct/src/lib.rs:556: [CPU 0] Task::drop(): ls-1ac6fa56248369a8(17)
[T] kernel/crate_metadata/src/lib.rs:288: ### Dropped LoadedCrate: ls-1ac6fa56248369a8
[D] kernel/mod_mgmt/src/lib.rs:818: load_crate_as_application(): trying to load application crate at "/namespaces/_applications/cat-d5c6adc40052f05e.o"
[I] kernel/mod_mgmt/src/lib.rs:2866: Symbol "<io::ioError as core::fmt::Debug>::fmt::h9ede19d9c213e051" not initially found in namespace "_applications", attempting to load crate "io-" into namespace "_kernel" that may contain it.
[D] kernel/mod_mgmt/src/lib.rs:855: load_crate: trying to load crate at "/namespaces/_kernel/io-3ca3499fbd35d589.o"
[I] kernel/mod_mgmt/src/lib.rs:865: loaded new crate "io-3ca3499fbd35d589", num sections: 12, added 4 new symbols
[I] kernel/mod_mgmt/src/lib.rs:826: loaded new application crate: "cat-d5c6adc40052f05e", num sections: 32, added 1 new symbols
[D] kernel/spawn/src/lib.rs:721: task_wrapper [1]: "cat-d5c6adc40052f05e(18)" about to call task entry func 0x3f68660 {fn(alloc::Vec::Vec<alloc::string::String>) -> isize} with arg ["extra_files/test_files/text/the_empire_strikes_back.txt"]
[D] kernel/spawn/src/lib.rs:889: task_cleanup_success: "cat-d5c6adc40052f05e" successfully exited with return value 0
[I] applications/shell/src/lib.rs:1830: terminal: task [18] returned exit value: Some(0)
[T] kernel/task_struct/src/lib.rs:556: [CPU 0] Task::drop(): cat-d5c6adc40052f05e(18)
[T] kernel/crate_metadata/src/lib.rs:288: ### Dropped LoadedCrate: cat-d5c6adc40052f05e
```

Host

Emulating with qemu-system-x86_64

```
Machine View
attack_of_the_clones.txt
a_new_hope.txt

task [17] exited with code 0 (0x0)
#!/ cat extra_files/test_files/text/the_empire_strikes_back.txt
It is a dark time for the Rebellion. Although the Death Star has been destroyed, Imperial troops have driven the Rebel forces from their hidden base and pursued them across the galaxy.

Evading the dreaded Imperial Starfleet, a group of freedom fighters led by Luke Skywalker has established a new secret base on the remote ice world of Hoth.

The evil lord Darth Vader, obsessed with finding young Skywalker, has dispatched thousands of remote probes into the far reaches of space....

task [10] exited with code 0 (0x0)
#!/
fg bg jobs
bn cat cd
date deps examp
heap_eval hello hell
less loadc ls
mkdir ns ping
pnu_sample_stop print_fault_log ps
qemu_test raw_node rn
rq_eval scheduler_eval secon
shell swap test_
test_backtrace test_block_io test_
test_identity_mapping test_lxgbe test_
test_panic test_preemption_counter test_
test_std_fs test_sync_block test_
test_wait_queue test_wastime unico
upd wasm

fg bg jobs
bn cat cd
date deps examp
heap_eval hello hell
less loadc ls
```

Theseus

Process Control

The tasking subsystem in Theseus implements full support for multitasking,

Theseus is a single address space (SAS) OS.

Does not follow the classic POSIX/Unix-like "process" abstraction.

The terms "task" and "thread" can be used interchangeably.


```
pub fn task_switch(  
    next: TaskRef,  
    cpu_id: CpuId,  
    preemption_guard: PreemptionGuard  
) -> (bool, PreemptionGuard)
```

Context switching from one thread to another thread in the same address space is done by via task_switch().

Theseus follows the Rust standard library's model for threading,

You can spawn a new task with a function or a closure as the entry point.

You can customize a new task using a convenient builder pattern.

You can wait for a task to exit by joining it.

You can use any standard synchronization types for inter-task communication.

You can catch the action of stack unwinding after a panic or exception occurs in a task.

```
pub struct Task {  
    pub id: usize,  
    pub name: String,  
    pub mmi: Arc<Mutex<MemoryManagementInfo, DisableIrq>, Global>,  
    pub is_an_idle_task: bool,  
    pub app_crate: Option<Arc<AppCrateRef, Global>>,  
    pub namespace: Arc<CrateNamespace, Global>,  
    /* private fields */  
}
```

A structure that contains contextual information for a thread of execution.

Invariants Upheld in Task Management

Spawning a new task must not violate memory safety.

All task states must be released in all possible execution paths.

All memory transitively reachable from a task's entry function must outlive that task.

Memory Management

All kernel entities, libraries, and applications are loaded into and executed within a single address space.

Theseus's single address space is a virtual address space, not a physical address space.

Virtual and physical addresses are given dedicated, separate types that are not interoperable.

Terminologies

Description of Type	Virtual Memory Type	Physical Memory Type
A memory address	VirtualAddress	PhysicalAddress
A chunk of memory	Page	Frame
A range of contiguous chunks	PageRange	FrameRange
Allocator for memory chunks	page_allocator	frame_allocator

Mapping virtual memory to physical memory

Mapper

Provides functions to map virtual memory to physical memory,

PageTable

A top-level page table.

MappedPages

Range of virtually contiguous pages that are mapped to physical frames and have a single exclusive owner.



Invariants and Safety Guarantees at Compile-time

The mapping from virtual pages to physical frames must be one-to-one, or bijective.

A memory region must be unmapped exactly once,
only after no outstanding references to it remain.

A memory region must not be accessible beyond its bounds.

A memory region can only be referenced as mutable or executable if mapped as such.



Deadlock Management

Utilizes resource cleanup via unwinding within drop handlers.

Tasks own resource objects directly, with ownership tracked by the Rust compiler.

Lock guards are automatically released during unwinding for efficiency.

Custom-built unwinding process is independent and triggered only during exceptions or task termination.

Enables intralingual resource revocation, reducing deadlock risks and enhancing fault isolation.

User Interface

By default, Theseus,

Uses the keyboard as its primary input and optionally a mouse.

Uses the graphical display as its primary output.

In headless mode through serial communication, it will spawn a terminal emulator.

```
QEMU - Press Ctrl+Alt+G to release grab
Machine View
Theseus Terminal Emulator
Press Ctrl+C to quit a task
/:
fg          bg          jobs          clear
bn          cat          cd          channel_eval
date        deps        example        getopts
heap_eval  hello        hull        kill
ls          loadc       ls          lspci
mkdir       ns          ping        pmu_sample_start
pmu_sample_stop  print_fault_log  ps          pud
qemu_test  rau_node    rn          rq
rq_eval    scheduler_eval  seconds_counter  serial_echo
shell      swap        test_aligned_page_allocation  test_async
test_backtrace  test_block_io    test_channel    test_filerw
test_identity_mapping  test_lxgbe    test_libc      test_nix5
test_panic    test_preemption_counter  test_restartable  test_scheduler
test_std_fs    test_sync_block  test_task_cancel  test_tls
test_wait_queue  test_wasntime   unicode_width    unwind_test
upd          uasm

/: cat extra_files/test_files/text/
the_rise_of_skywalker.txt    the_phantom_menace.txt    the_last_jedi.txt    the_force_awakens.txt
the_empire_strikes_back.txt  revenge_of_the_sith.txt    return_of_the_jedi.txt  attack_of_the_clones.txt
a_new_hope.txt

/: cat extra_files/test_files/text/the_empire_strikes_back.txt
It is a dark time for the
Rebellion. Although the Death
Star has been destroyed,
Imperial troops have driven the
Rebel forces from their hidden
base and pursued them across
the galaxy.

Evading the dreaded Imperial
Starfleet, a group of freedom
fighters led by Luke Skywalker
has established a new secret
base on the remote ice world
of Hoth.

The evil lord Darth Vader,
obsessed with finding young
Skywalker, has dispatched
thousands of remote probes into
the far reaches of space...

task [16] exited with code 0 (0x0)
/: █
```

Theseus Terminal Emulator

This is the default VGA graphical interface
that we will be presented while booting in
default mode.

```
QEMU - Press Ctrl+Alt+G to release grab
Machine View
Theseus Terminal Emulator
Press Ctrl+C to quit a task
/:
fg          bg          jobs          clear
bn          cat          cd          channel_eval
date        deps         example        getopt
heap_eval  hello         hull          kill
ls          loadc        ls            lspci
pnu_sample_stop  print_fault_log  ps            pnu_sample_start
qemu_test   rau_node     rm            pud
rq_eval     scheduler_eval  seconds_counter  rq
shell       swap         test_aligned_page_allocation  serial_echo
test_backtrace  test_block_io    test_channel     test_async
test_identity_mapping  test_lxgbe       test_libc        test_filerw
test_panic    test_preemption_counter  test_restartable  test_nix5
test_std_fs   test_sync_block    test_task_cancel  test_scheduler
test_wait_queue  test_wasntime     unicode_width     test_tis
upd          uasm         

/: cat extra_files/test_files/text/
the_rise_of_skywalker.txt    the_phantom_menace.txt    the_last_jedi.txt    the_force_awakens.txt
the_empire_strikes_back.txt  revenge_of_the_sith.txt    return_of_the_jedi.txt  attack_of_the_clones.txt
a_new_hope.txt

/: cat extra_files/test_files/text/the_empire_strikes_back.txt
It is a dark time for the
Rebellion. Although the Death
Star has been destroyed,
Imperial troops have driven the
Rebel forces from their hidden
base and pursued them across
the galaxy.

Evading the dreaded Imperial
Starfleet, a group of freedom
fighters led by Luke Skywalker
has established a new secret
base on the remote ice world
of Hoth.

The evil lord Darth Vader,
obsessed with finding young
Skywalker, has dispatched
thousands of remote probes into
the far reaches of space...

task [16] exited with code 0 (0x0)
/: █
```

List of available utilities

Simple cat utility usage

Theseus Terminal Emulator

This is the default VGA graphical interface that we will be presented while booting in default mode.

Secondary Disk Scheduling Management



- Optimizes disk operations by reducing costly calls to storage medium.
- Enhances system efficiency with increased memory usage.
- Limited by hard-coded references to specific storage device type.
- May produce inconsistent results if other system crates write to the device directly.
- Calls for a more flexible caching solution for secondary disk scheduling management.

State Management

State management refers to the mechanisms used to track and control the state of various system components, such as processes, memory, and devices in Theseus OS

Prioritize minimizing state spill within its cells

- Ensures that no unnecessary state changes occur due to interactions between cells

Minimize

Opaque exploration in client-server interactions

- Clients own progress states independently
- Reduce overhead and eliminate handle-based abstractions

Opaque Explore

State Management

Special States such as soft states, unavoidable hardware related states, managed with "state_db" to ensure persistence.

Persistence

Standard Support

Theseus OS is a conceptual Operating system, therefore the standard support provided is comparatively limited

Official GitHub Page

By visiting the official github repository of the Theseus OS, one can refer to the github README file, use the discussion forum, read or open issues.

Theseus Blog

The Theseus OS blog contains an extensive documentation of crates and the operating system. Additionally, contact details can also be found in the following blog.



Theseus OS

Comparisons

between Theseus OS and
traditional operating systems

D M T N Dissanayaka

IT22169730

User Interface

Theseus OS

- Focus- System Level Interactions
- Redesign needed for improved efficiency

Conventional OS

- Established GUI frameworks and libraries
- Extensive Support for application GUI

Process Management

Theseus OS

- Tasks as threads, with same address space
- Lifecycle Management of tasks
- Preemptive
 - OS interrupts a currently running task to give resources to another task.
- Cooperative multitasking
 - Tasks voluntarily free memory for other tasks

Conventional OS

- Traditional POSIX process model
- Separate process management mechanisms
- Context switching for task management

Memory Management

Theseus OS

- Single Address Space (SAS) architecture
- Utilizes Rust's memory safety features
- Dedicated memory types for clarity

Conventional OS

- Hardware-based memory protection
- Reliance on hardware mechanisms
- Less precise terminology

Deadlock Management

Theseus OS

- Resource cleanup via unwinding
- Drop handlers for timely resource release

Conventional OS

- Manual deadlock detection and resolution
- Lock-based deadlock prevention

Secondary Disk Scheduling

Theseus OS

- Implementation of caching layer for block-based storage devices
- Reduction of disk access calls for improved efficiency
- Limitations include hard-coded references and inefficiencies

Conventional OS

- Utilization of traditional disk scheduling algorithms
- Reliance on disk scheduling policies for disk access optimization
- Established disk scheduling algorithms with optimizations

State Management

Theseus OS

- Minimization of state spill in cells
- Opaque exportation for client-server interactions
- Management of soft states for convenience and performance

Conventional OS

- Standardized state management models
- Emphasis on encapsulation and state preservation
- Focus on critical state preservation

Limitations

Key Findings

Research limited to summarizing key findings of the original research.

System hardware requirements, installation process, user interfaces, process control, memory management, deadlock management, secondary disk scheduling management and standard support.

Word Limit

Strict word limit of around two thousand words were employed during the creation of the report.

Extensions to Research

The following extensions were added which were not explicitly stated in the original research paper

- Secondary disk scheduling management
- State management

References

1.

K. Boos, N. Liyanage, R. Ijaz, and L. Zhong, "Theseus: an Experiment in Operating System Structure and State Management," in Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation, USENIX.
Accessed: Apr. 01, 2024. [Online]. Available: <https://www.usenix.org/system/files/osdi20-boos.pdf>

2.

"___Theseus_Crates___ - Rust," www.theseus-os.com.
https://www.theseusos.com/Theseus/doc/___Theseus_Crates___/index.html
(accessed Apr. 05, 2024).

3.

"Memory Management in Theseus," in The Theseus OS Book, Theseus OS.
Accessed: Apr. 07, 2024. [Online].
Available: <https://www.theseusos.com/Theseus/book/subsystems/memory.html>

4.

"Tasking Subsystem in Theseus," in The Theseus OS Book,
Accessed: Apr. 07, 2024. [Online].
Available: <https://www.theseus-os.com/Theseus/book/subsystems/task.html>

Sharvajan S
IT22231628



References



"Display Subsystem," in The Theseus OS Book, Accessed: Apr. 08, 2024. [Online].

Available: <https://www.theseus-os.com/Theseus/book/subsystems/display/display.html>



"block_cache - Rust," www.theseus-os.com.

https://www.theseusos.com/Theseus/doc/block_cache/index.html

(accessed Apr. 08, 2024)

Sharvajan S
IT22231628



Q Joyal
IT22196088

D R Wickrama Arachchi
IT22360496

Sharvajen S
IT22231628

D M T N Dissanayaka
IT22169730



Theseus OS

Q/A



Theseus OS

Thank You

For Listening