

Content

- Work Matrix
- Introduction
- Key Findings
 - System Hardware Requirements
 - Installation Process
 - User Interfaces
 - Process Control
 - Memory Management
 - Deadlock Management
 - Secondary Disk Scheduling Management
 - Standard Support
 - State Management
- Comparative Analysis of Theseus Operating System
- Limitations and Extensions to the Case Study

Theseus: An Experiment in Operating system Structure and State Management

Work matrix

Quintus Joyal IT22196088

- Process Control
- Memory Management
- Deadlock Management

DR Wikrama Arachi IT221360496

- Brief Introduction
- System Hardware Requirements
- Installation Process

Sharvajen S IT22231628

- User Interfaces
- Secondary Disc Sheduling Management
- State Management

DM Thimira Niromin Dissanayaka IT22169730

- Standard Support
- Comparative Analysis of Theseus Operating System
- Limitations

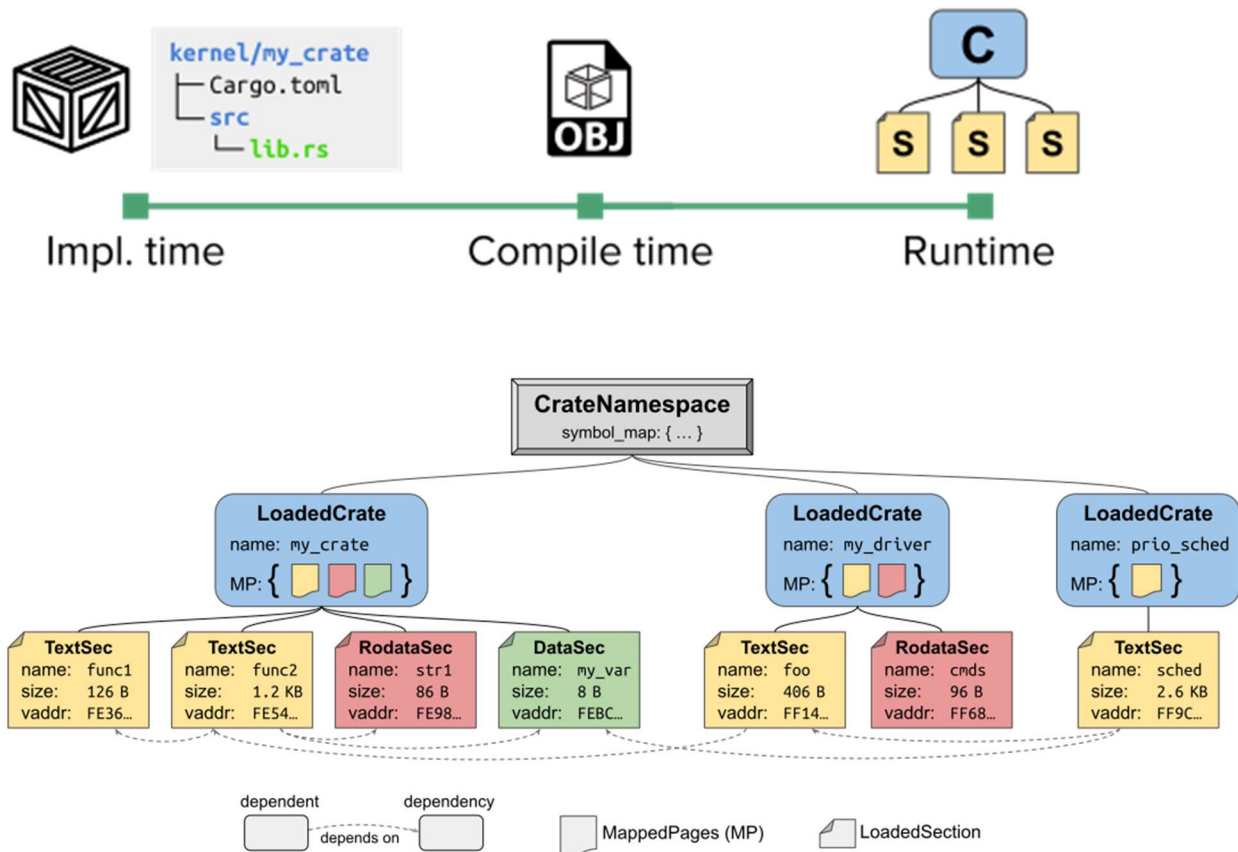
Table of Contents

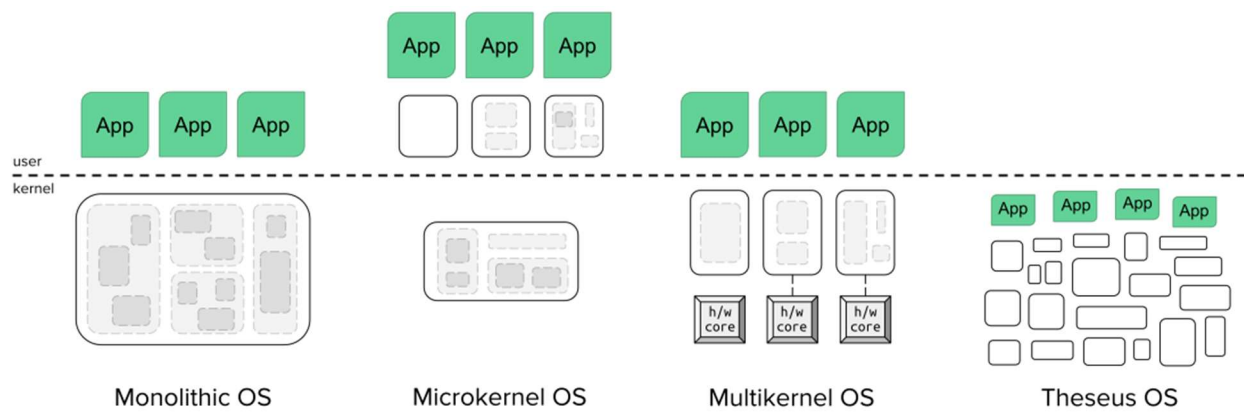
Table of Contents	3
A Brief Introduction	4
Key Findings.....	6
System Hardware Requirements	7
Installation Process	7
User Interfaces	9
Process Control	10
Memory Management	11
Deadlock Management	12
Secondary Disk Scheduling Management	12
Standard Support.....	13
State Management.....	13
Comparison	14
User Interface.....	14
Process Management.....	14
Memory Management	14
Deadlock Management	15
Secondary Disk Scheduling	15
State Management.....	15
Limitations and Extensions to the Case Study	16
Refrences.....	17

A Brief Introduction

DR Wikrama Arachi IT221360496

- 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.





Key Findings

System Hardware Requirements

DR Wikrama Arachi IT221360496

- 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

DR Wikrama Arachi IT221360496

- The GitHub repository offers detailed instructions for software building and running.
- Software can be run on Linux, Windows, MacOS, and Docker.
- No standard installation procedures are needed.
- 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:809: 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:809: task_cleanup_success: "ls-1ac6fa56248369a8" successfully exited with return value 0
[I] applications/shell/src/lib.rs:1030: 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 0x3f588e0 {fn(alloc::vec::Vec<alloc::string::String>) -> isize} with arg ["extra_files/test_files/text/"]
[D] kernel/spawn/src/lib.rs:809: task_cleanup_success: "ls-1ac6fa56248369a8" successfully exited with return value 0
[I] applications/shell/src/lib.rs:1030: 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:809: task_cleanup_success: "cat-d5c6adc40052f05e" successfully exited with return value 0
[I] applications/shell/src/lib.rs:1030: 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

0 1 29s 1 make
```

```
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.

Loading 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 [18] exited with code 0 (0x0)
/:
fg bg jobs
bn cat cd
date deps examp
heap_eval hello hull
less loadc ls
mkdir ns ping
pnu_sample_stop print_fault_log ps
qemu_test raw_mode ra
rq_eval scheduler_eval second
shell swap test
test_backtrace test_block_io test
test_identity_mapping test_lxybe test
test_panic test_preemption_counter test
test_std_fs test_sync_block test
test_wait_queue test_wasntime unico
upd wasm

/:
fg bg jobs
bn cat cd
date deps examp
heap_eval hello hull
less loadc ls
```

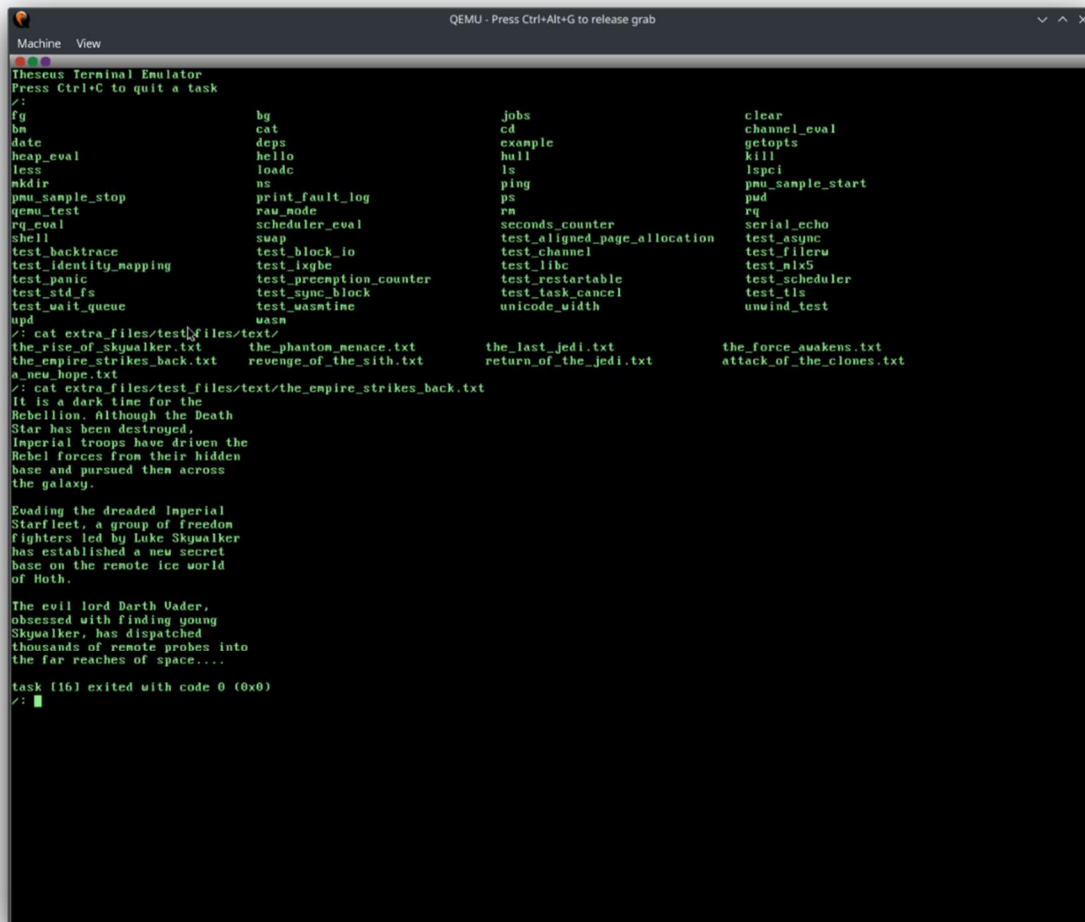
Figure 2.2.3.1: qemu-system-x86_64

User Interfaces

Sharvajen S IT22231628

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.



```
Machine View
QEMU - Press Ctrl+Alt+G to release grab

Theseus Terminal Emulator
Press Ctrl+C to quit a task
/:
fg          bg          jobs          clear
bm          cat          cd          channel_eval
date        deps         example       getopt
heap_eval  hello         hull         kill
less       loadc        ls          lspci
mkdir      nc          ping        pmu_sample_start
pmu_sample_stop print_fault_log raw_mode    pod
qemu_test  qemu_test  scheduler_eval rm          rq
rq_eval    swap        seconds_counter serial_echo
shell      test_block_io test_aligned_page_allocation test_async
test_backtrace test_lxgbc test_channel test_filerw
test_identity_mapping test_preemption_counter test_llbc test_nlx5
test_panic test_sync_block test_restartable test_scheduler
test_stdfs test_wasmtime test_task_cancel test_tls
test_wait_queue wasm         unicode_width unwind_test
upd

/: 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)
/: █
```

Process Control

Quintus Joyal IT22196088

- The tasking subsystem in Theseus implements full support for multitasking,
 - Theseus is a single address space (SAS) OS, it does not have a dedicated address space for each task.
 - Does not follow the classic POSIX/Unix-like "process" abstraction.
 - The terms "task" and "thread" can be used interchangeably.
 - One could also consider the entirety of Theseus to be a single "process" in that all tasks execute within the same address space,
-
- Context switching from one thread to another thread in the same address space is done by via `task_switch()`.

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

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 */
}
```

Invariants Upheld in Task Management

1. Spawning a new task must not violate memory safety.
2. All task states must be released in all possible execution paths.
3. All memory transitively reachable from a task's entry function must outlive that task.

Memory Management

Quintus Joyal IT22196088

- 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.

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

1. The mapping from virtual pages to physical frames must be one-to-one, or bijective.
2. A memory region must be unmapped exactly once, only after no outstanding references to it remain.
3. A memory region must not be accessible beyond its bounds.
4. A memory region can only be referenced as mutable or executable if mapped as such.

Deadlock Management

Quintus Joyal IT22196088

- 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.
-

Secondary Disk Scheduling Management

Sharvajen S IT22231628

- Optimizes disk operations by reducing costly calls to storage medium.
- Enhances system efficiency with increased memory usage.
- Limited by hardcoded 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.

Standard Support

Thimira

Conceptual Operating System, thus limited support.

Official GitHub page

Read or Open Issues

Discussion Forum

Theseus Blog

Documentation

Contacts

State Management

Sharvajen S IT22231628

- Prioritize minimizing state spill within its cells
 - o Ensures that no unnecessary state changes occur due to interactions between cells
- Opaque exploration in client-server interactions
 - o Clients own progress states independently
 - Note: Clients can manage states without constant com with server
 - o Reduce OS overhead
 - o Eliminate handle-based abstractions
 - Note: Handles represents resources/objects. By eliminating abstractions -> Design simplify + Reduce Overhead
- Special States (soft states, unavoidable hardware related states) managed with `state_db` to ensure persistence.

State Spill: program's state, such as variables or registers, needs to be temporarily stored in memory[RAM] (spilled)

Comparison

User Interface

Theseus	Conventional OS
<ul style="list-style-type: none">- Focus- System Level Interactions- Redesign needed for improved efficiency	<ul style="list-style-type: none">- Established GUI frameworks and libraries- Extensive Support for application GUI

Process Management

Theseus	Conventional OS
<ul style="list-style-type: none">- Tasks as threads, with same address space- Lifecycle Management of tasks- Preemptive OS interrupts currently running task to give resources to another task.- cooperative multitasking Tasks voluntarily free memory for other tasks	<ul style="list-style-type: none">- Traditional POSIX process model- Separate process management mechanisms- Context switching for task management

Memory Management

Theseus	Conventional OS
<ul style="list-style-type: none">- Single Address Space (SAS) architecture- Utilizes Rust's memory safety features- Dedicated memory types for clarity	<ul style="list-style-type: none">- Hardware-based memory protection- Reliance on hardware mechanisms- Less precise terminology

Deadlock Management

Theseus	Conventional OS
<ul style="list-style-type: none">- Resource cleanup via unwinding • Drop handlers for timely resource releas	<ul style="list-style-type: none">- Manual deadlock detection and resolution • Lock-based deadlock prevention

Secondary Disk Scheduling

Theseus	Conventional OS
<ul style="list-style-type: none">- Implementation of caching layer for block-based storage devices- Reduction of disk access calls for improved efficiency- Limitations include hardcoded references and inefficiencies	<ul style="list-style-type: none">- Utilization of traditional disk scheduling algorithms- Reliance on disk scheduling policies for disk access optimization- Established disk scheduling algorithms with optimizations

State Management

Theseus	Conventional OS
<ul style="list-style-type: none">- Minimization of state spill in cells- Opaque exportation for client-server interactions- Management of soft states for convenience and performance	<ul style="list-style-type: none">- Standardized state management models- Emphasis on encapsulation and state preservation- Focus on critical state preservation

Limitations and Extensions to the Case Study

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.

Extensions to Research

- Secondary Disk Scheduling management

- State management

Report ~ 2k Words

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>

[5]

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

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

[6]

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

https://www.theseusos.com/Theseus/doc/block_cache/index.html (accessed Apr. 08, 2024)