IE2050

Theseus OS

An Experiment in Operating System Structure and State Management

Group 8

15 May 2024

Workload Matrix

IT22196088 **Q Joyal**

- Process Control
- Memory Management
- Deadlock Management

IT22360496 **D R Wickrama Arachchi**

- A Brief Introduction
- System Hardware Requirements
- Installation Process

IT22231628 **Sharvajen S**

- User Interfaces
- Secondary Disk Scheduling Management
- References

IT22169730 **D M T N Dissanayaka**

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

D R Wickrama Arachchi
IT22360496

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

Q Joyal
IT22196088

D R Wickrama Arachchi
IT22360496

Sharvajen S
IT22231628

D M T N Dissanayaka
IT22169730

Key Findings

of the Theseus OS research paper

D R Wickrama Arachchi
IT22360496

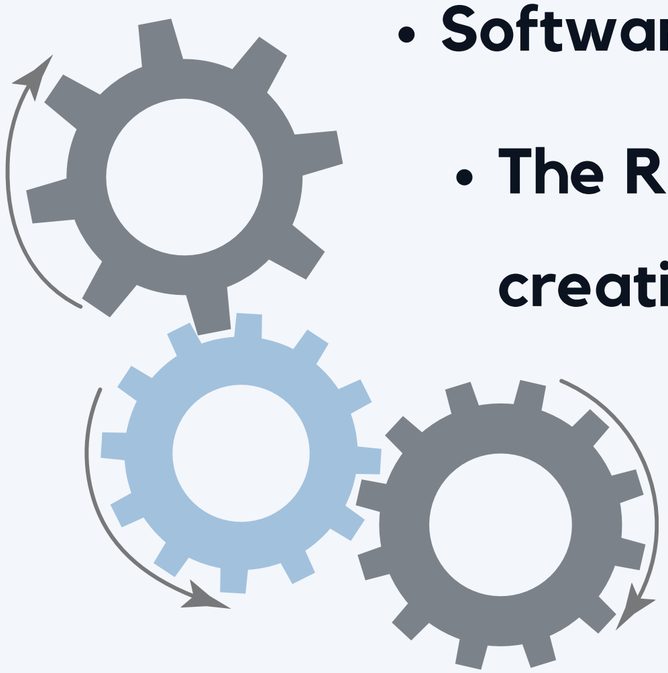
System Hardware Requirements

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

D R Wickrama Arachchi
IT22360496

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.

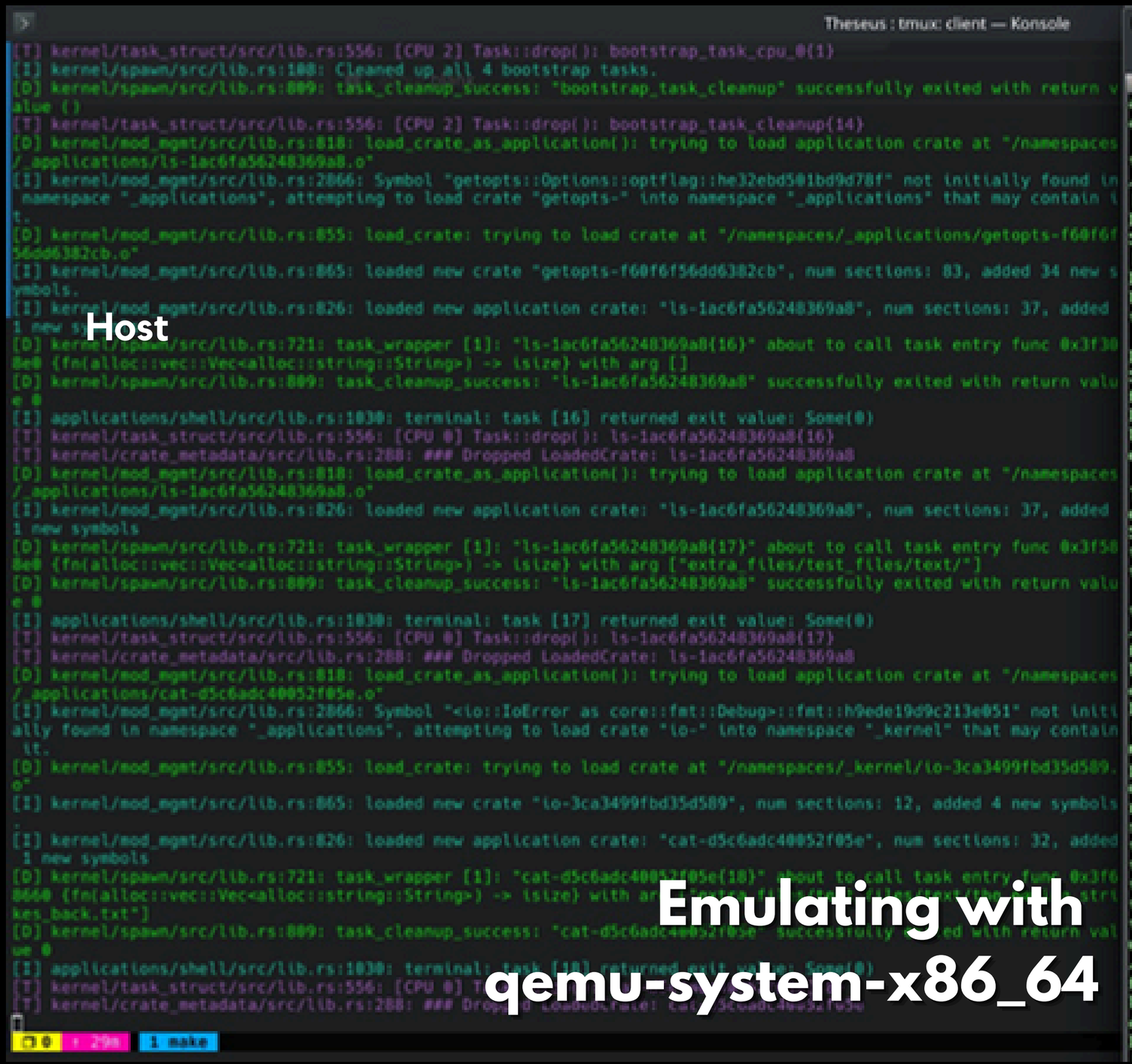


D R Wickrama Arachchi
IT22360496

Host

Theseus

Emulating with qemu-system-x86_64



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.

Q Joyal
IT22196088

```
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().

Q Joyal
IT22196088

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.

Q Joyal
IT22196088

```rust
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.

Q Joyal
IT22196088

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.

Q Joyal
IT22196088

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.

Q Joyal
IT22196088

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

Q Joyal
IT22196088

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.

Q Joyal
IT22196088

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.

Q Joyal
IT22196088

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.

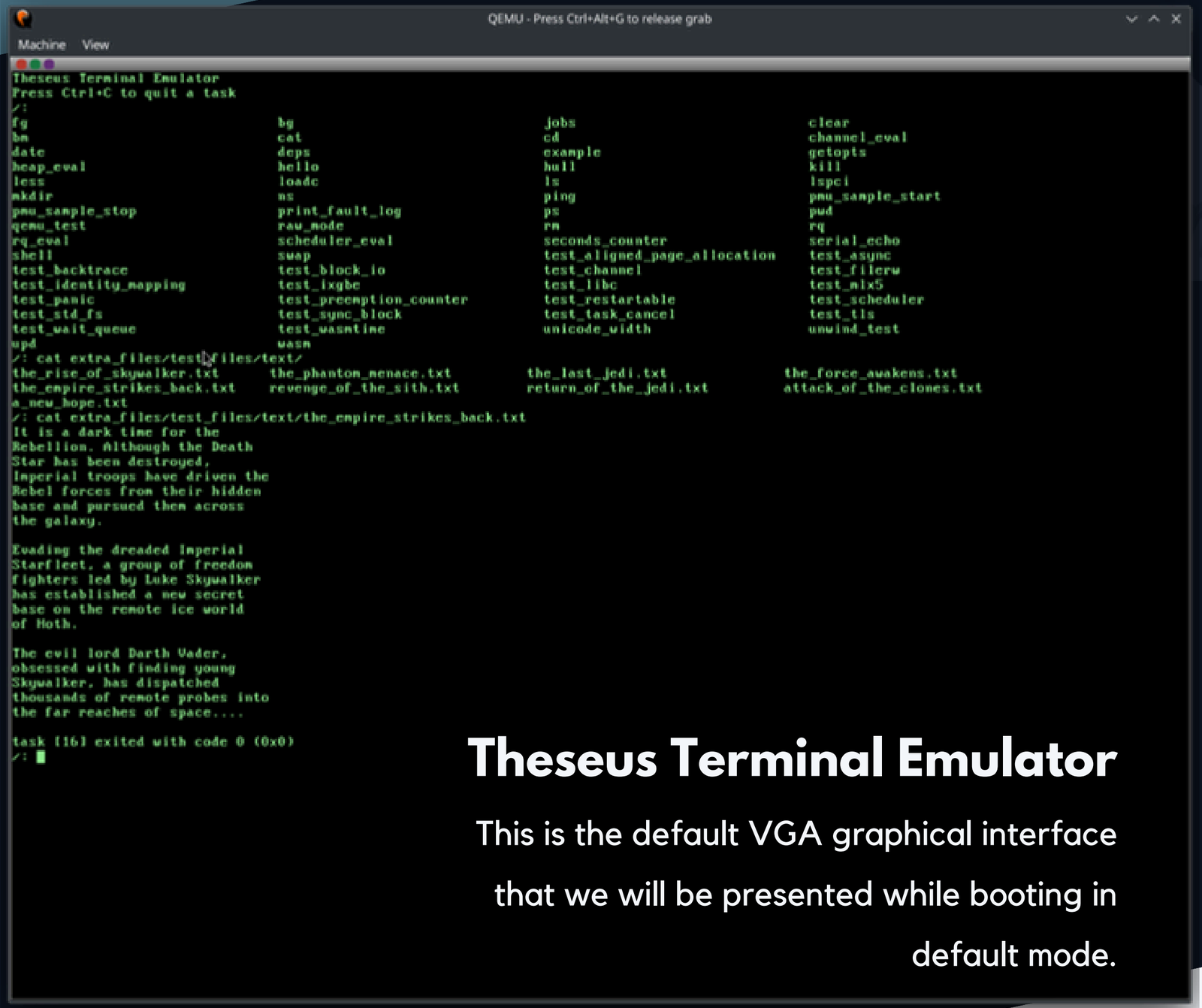
Q Joyal
IT22196088

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.

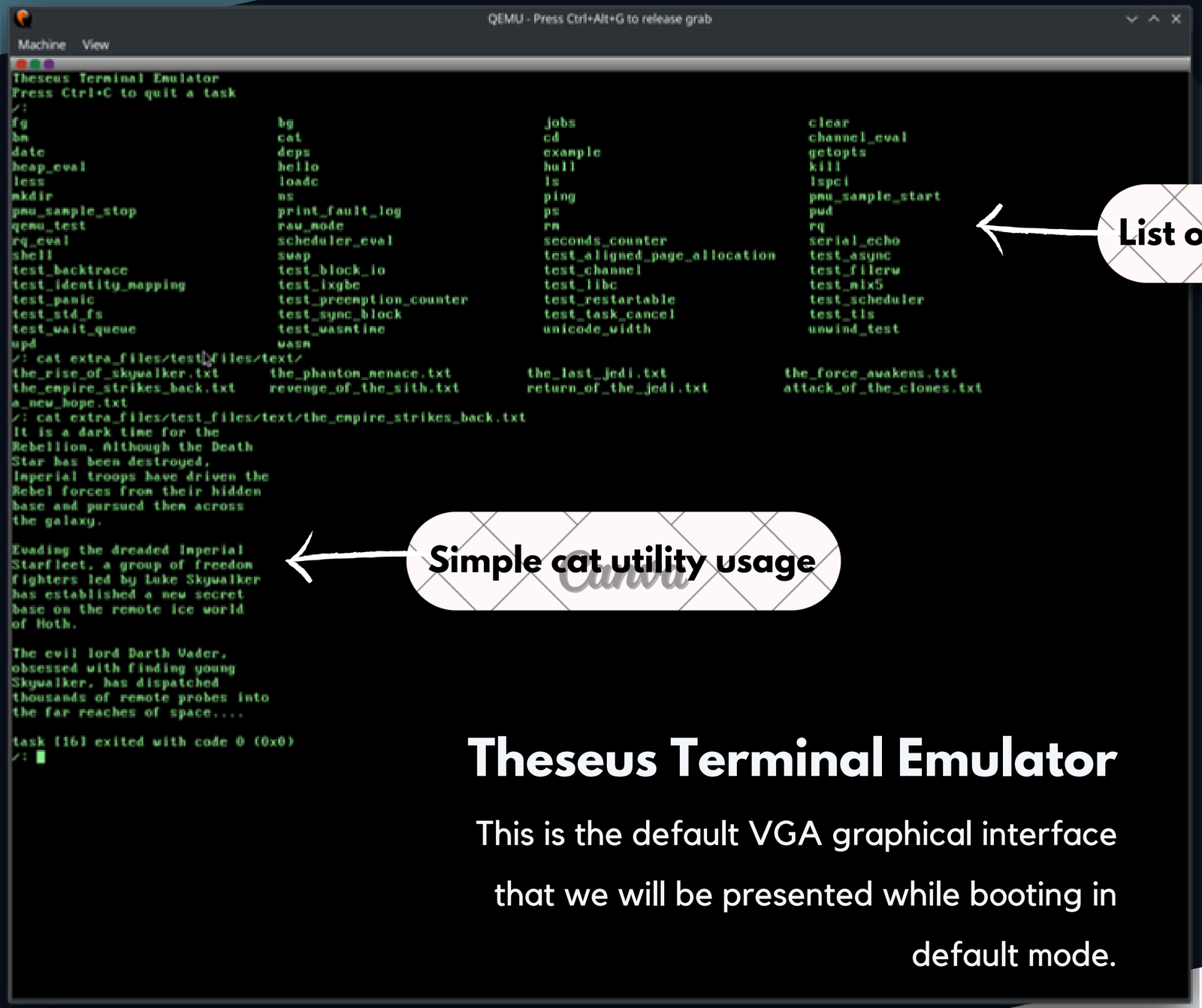
Sharvajen S
IT22231628



Theseus Terminal Emulator

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

Sharvajen S
IT22231628



List of available utilities

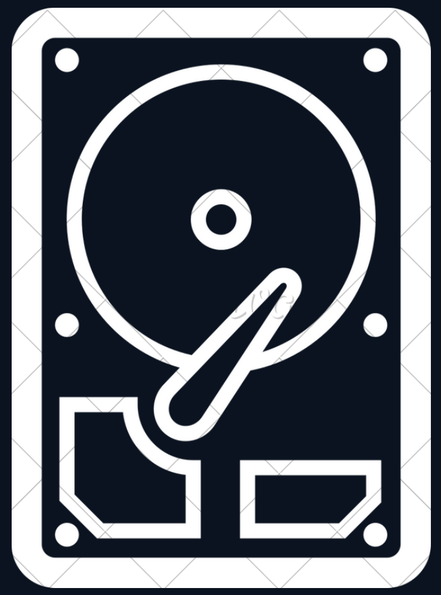
Simple cat utility usage

Theseus Terminal Emulator

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

Sharvajen S
IT22231628

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.

Sharvajen S
IT22231628

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

D M T N Dissanayaka
IT22169730

State Management

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

Persistence

D M T N Dissanayaka
IT22169730

Standard Support

Thesues 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.

D M T N Dissanayaka
IT22169730

Theseus OS

Comparisons

between Theseus OS and traditional operating systems

D M T N Dissanayaka

IT22169730

User Interface

Theseus OS	Conventional OS
• Focus- System Level Interactions • Redesign needed for improved efficiency	• Established GUI frameworks and libraries • Extensive Support for application GUI

D M T N Dissanayaka
IT22169730

Process Management

Theseus OS	Conventional 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	• Traditional POSIX process model • Separate process management mechanisms • Context switching for task management

D M T N Dissanayaka
IT22169730

Memory Management

Theseus OS	Conventional OS
• Single Address Space (SAS) architecture • Utilizes Rust's memory safety features • Dedicated memory types for clarity	• Hardware-based memory protection • Reliance on hardware mechanisms • Less precise terminology

D M T N Dissanayaka
IT22169730

Deadlock Management

Theseus OS	Conventional OS
• Resource cleanup via unwinding • Drop handlers for timely resource release	• Manual deadlock detection and resolution • Lock-based deadlock prevention

D M T N Dissanayaka
IT22169730

Secondary Disk Scheduling

Theseus OS	Conventional 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	• Utilization of traditional disk scheduling algorithms • Reliance on disk scheduling policies for disk access optimization • Established disk scheduling algorithms with optimizations

D M T N Dissanayaka
IT22169730

State Management

Theseus OS	Conventional OS
• Minimization of state spill in cells • Opaque exportation for client-server interactions • Management of soft states for convenience and performance	• Standardized state management models • Emphasis on encapsulation and state preservation • Focus on critical state preservation

D M T N Dissanayaka
IT22169730

Limitations

Word Limit

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

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.

Extensions to Research

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

- Secondary disk scheduling management
- State management

D M T N Dissanayaka
IT22169730

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

Sharvajen S
IT22231628

References

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)

Sharvajen S
IT22231628

Theseus OS

Q/A

Q Joyal
IT22196088

D R Wickrama Arachchi
IT22360496

Sharvajen S
IT22231628

D M T N Dissanayaka
IT22169730

Theseus OS

Thank You

For Listening