Metathesis:
A LaTeX template to Typeset Your Thesis for Submission to the School of Graduate Studies

(Changed the title by modifying the file `thesis.tex`*)*

by

© *my-name* (change this in `thesis.tex`)

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of *faculty* **or** Doctor of Philosophy (change this in `thesis.tex`)

Department of *dept-name* (change this in `thesis.tex`)

Memorial University of Newfoundland

Month Year (change this in `thesis.tex`, too)

St. John's Newfoundland

Abstract

This document provides information on how to write your thesis using the LaTeX document preparation system. You can use these files as a template for your own thesis, just replace the content, as necessary. You should put your real abstract here, of course.

"The purpose of the abstract, which should not exceed 150 words for a Masters' thesis or 350 words for a Doctoral thesis, is to provide sufficient information to allow potential readers to decide on relevance of the thesis. Abstracts listed in Dissertation Abstracts International or Masters' Abstracts International should contain appropriate key words and phrases designed to assist electronic searches."

— MUN School of Graduate Studies

Acknowledgements

Put your acknowledgements here...

"Intellectual and practical assistance, advice, encouragement and sources of monetary support should be acknowledged. It is appropriate to acknowledge the prior publication of any material included in the thesis either in this section or in the introductory chapter of the thesis."

— MUN School of Graduate Studies

Contents

List of Tables

List of Figures

Chapter 1

Marco teórico

1.1 ¿Que es un sistema operativo?

Según Stallings, el sistema operativo es el software encargado de controlar la ejecución de los programas y de administrar los recursos del procesador; además actúa como intermediario entre el usuario y el hardware, proporcionando los servicios necesarios para que las aplicaciones puedan ejecutarse correctamente (Stallings, 2023)

Añadiendo esta idea, Tanenbaum y Bos explican que la finalidad del sistema operativo es convertir las interfaces de hardware, que suelen ser complejas e inconsistentes, en abstracciones practicas y sencillas de usar para los programadores y aplicaciones. Ademas de gestionar recursos como CPU, memoria y dispositivos de E/S, permitiendo así que las aplicaciones trabajen sin problemas con el hardware (Tanenbaum and Bos, 2023).

Según el artículo de la Revista Ogma, que es un científica y multidisciplinaria, un sistema operativo es el software principal que actúa como intermediario entre el usuario y el hardware, cuyo propósito es favorecer el uso eficiente de la computadora. Transforma las interfaces de hardware, complejas e inconsistentes, en abstracciones útiles y manejables para y aplicaciones y programadores, administra y mejora recursos (CPU, memoria, dispositivos de E/S y almacenamiento); organiza la ejecución concurrente de tareas, y ofrece espacios accesibles que permite el acceso de las computadoras, permitiendo que usuarios sin preparación instalen programas, administren archivos. (Cusme Vera et al., 2022)

Un sistema operativo es como el sistema nervioso del computador, coordinando componentes hardware y software para que todo funcione de manera ordenada. Se encarga de organizar y administrar recursos como procesador, memoria, dispositivos

de E/S y almacenamiento.

1.2 Arquitecturas de Sistemas Operativos

1.2.1 Arquitectura Monolítica

El sistema operativo monolítico es un sistema operativo muy simple donde el núcleo controla directamente la gestión de dispositivos, memoria, archivos y procesos. Todos los recursos del sistema son accesibles al núcleo. En los sistemas monolíticos, cada componente del sistema operativo está contenido dentro del núcleo.

En una arquitectura monolítica, el núcleo del sistema operativo está diseñado para proporcionar todos los servicios del sistema operativo, incluyendo la gestión de memoria, la programación de procesos, los controladores de dispositivos y los sistemas de archivos, en un único y gran binario. Esto significa que todo el código se ejecuta en el espacio del núcleo, sin separación entre los procesos del núcleo y los del usuario (GeeksforGeeks, 2024b).

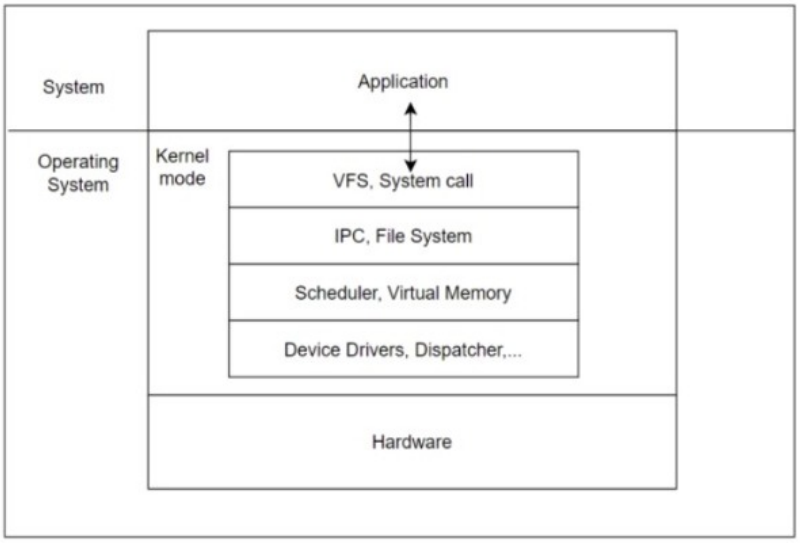

Figure 1.1: Monolithic kernel based operating system (Harshvardhan and Irabatti, 2023)

1.2.2 Arquitectura en Microkernel

Un microkernel es un enfoque para diseñar un sistema operativo (SO). El microkernel proporciona servicios fundamentales para su funcionamiento, como la gestión básica de memoria, la programación de tareas.El microkernel es un tipo de sistema operativo que proporciona servicios básicos.

como los controladores de dispositivos y los sistemas de archivos, son gestionados por procesos a nivel de usuario. El proceso a nivel de usuario se comunica con el microkernel mediante el paso de mensajes. Esta forma de gestionar el proceso hace que los microkernels sean más modulares y flexibles que los kernels monolíticos tradicionales (GeeksforGeeks, 2024a).

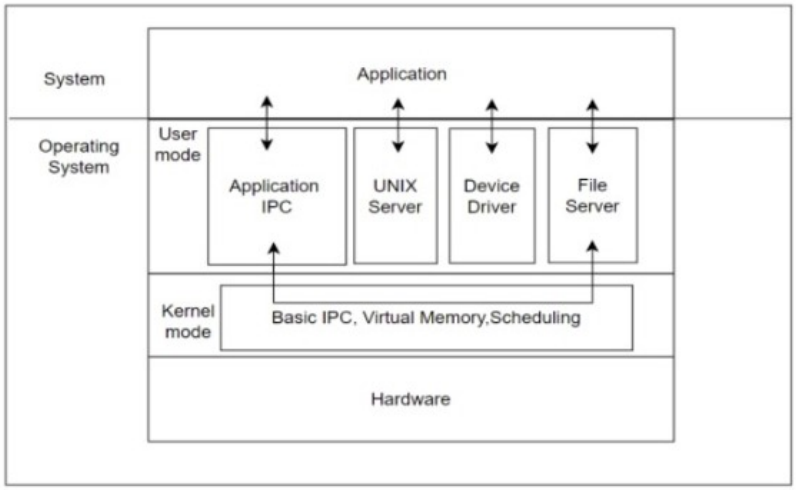

Figure 1.2: Microkernel based operating system (Harshvardhan and Irabatti, 2023)

1.2.3 Arquitectura Hibrida

Un núcleo híbrido es una arquitectura de núcleo basada en la combinación de aspectos de las arquitecturas de micronúcleo y núcleo monolítico utilizadas en sistemas operativos .

La idea de esta categoría es tener una estructura de kernel similar a la de un microkernel, pero implementada en términos de un kernel monolítico. A diferencia de un microkernel, todos (o casi todos) los servicios del sistema operativo se encuentran en el espacio del kernel . Si bien no hay sobrecarga de rendimiento para el paso de mensajes ni el cambio de contexto entre el kernel y el modo de usuario, como en los kernels monolíticos , no hay ventajas de rendimiento al tener servicios en el espacio de usuario , como en los microkernels (Microsoft Wiki / Fandom, 2024).

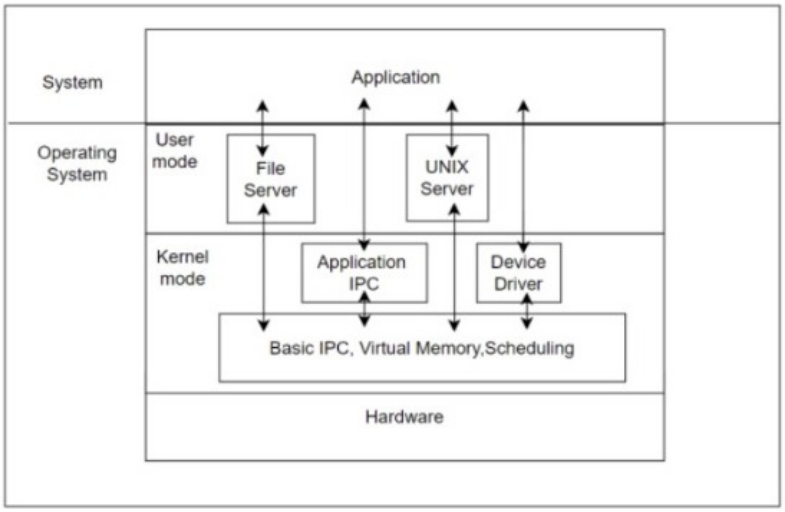

Figure 1.3: Hybridkernel based operating system (Harshvardhan and Irabatti, 2023)

1.2.4 Arquitectura en Exokernel

Exokernel es un sistema operativo desarrollado por el Instituto Tecnológico de Massachusetts (MIT) con el concepto de poner la aplicación bajo control. Los sistemas operativos Exokernel buscan proporcionar gestión de recursos de hardware a nivel de aplicación. La arquitectura de este sistema operativo está diseñada para separar la protección de recursos de la gestión, facilitando así la personalización específica de cada aplicación(Keetmalin, 2017).

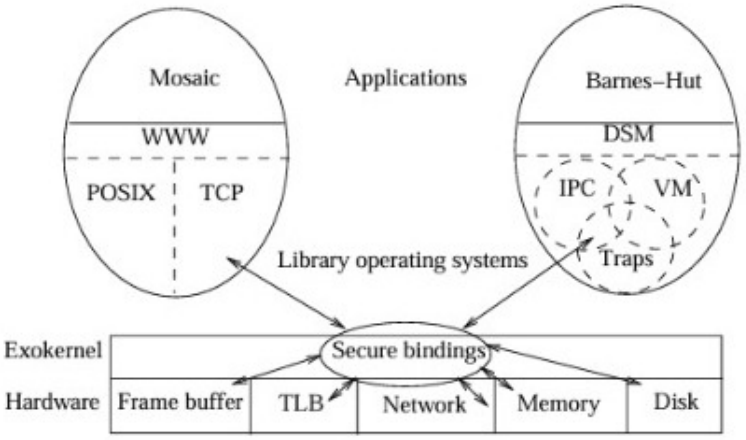

Figure 1.4: Exokernel based operating system (Engler et al., 1995)

Chapter 2

Revisión de sistemas operativos existentes

2.1 RedoxOs

Figure 2.1: Logo del sistema operativo Redox OS (Wikipedia contributors, 2025)

Redox es un sistema operativo tipo Unix que está escrito en el lenguaje de programación Rust, con el objetivo de implementar un microkernel y un sistema de aplicaciones. Rust es un lenguaje enfocado en la seguridad, rendimiento, gratuito y fácil de navegar. Redox se enfocó en la mejora de varios sistemas anteriores que presentaban errores. Uno de sus proyectos es Redox OS. Es compatible con POSIX, y la comunidad está trabajando para escribir la biblioteca libc en Rust, llamada relibc (Saini, 2018).

Dado que para los sistemas operativos es muy importante la seguridad, porque los sistemas operativos tienen un alto nivel de abstracción sobre los recursos del sistema, más aún en el caso de que Linux tuviera muchos errores en diferentes bibliotecas ocasionados por la seguridad de memoria. Rust en cambio evita todos esos problemas al tener seguridad de memoria en tiempo de compilación. El kernel de Redox contiene más de 20 mil líneas de código, y su diseño es de alto nivel por eso aún puede tener problemas (Ellmann and Emmerich, 2019).

El sistema operativo Redox OS se apoya en un microkernel y está inspirado en el sistema operativo MINIX. Las funciones tradicionales se implementan en kernels monolíticos, como controladores de dispositivo, pilas y el sistema de archivos. Los

microkernels son más seguros y ofrecen mayor estabilidad, pero una de las principales desventajas es que el rendimiento que ofrecen no está optimizado para la mayoría del hardware actual (Ritter, 2019).

En cuanto a sus componentes clave, Redox OS integra un microkernel responsable de los procesos del sistema, un **sistema de archivos** implementado en espacio de usuario, **drivers** que también se ejecutan en espacio de usuario para incrementar la estabilidad, y compatibilidad POSIX que permite correr aplicaciones de otros sistemas Unix. También, añade **la biblioteca relibc**, escrita en Rust para mejorar la seguridad y la compatibilidad, ademas de **gestión de memoria** segura que evita vulnerabilidades comunes como los desbordamientos(The Redox OS Community, 2025).

2.2 Mac OS

Figure 2.2: Logo del sistema operativo mac OS (GQ Informática, 2023)

El proyecto Mach dio comienzo en 1985 en la universidad de Carnegie Mellon con la finalidad de crear un microkernel que reemplace al BSD. Su diseño buscaba más seguridad para los usuarios, nose genero lo resultados esperados y se cancelo en 1994. Entonces el proyecto fue adoptado pro OSF/1 Y NeXTSTEP por lo que completo al combinar los componente de BSD, dando origen al kernel XNU de tipo monolítico, pero presento limitaciones con el System 7 y fracaso el proyecto, por ultimo este último avance fue adquirida por Steve Jobs, lo que permitió integrar NeXTSTEP y su kernel XNU como nuevo sistema operativo a Mac OS X, lanzado en 24 marzo del 2001 con el kernel de XNU, que integraba Mach 3.0 y el marco I/O kit, además de ser integrado en Intel Y ARM (Keuper, 2012). El kernel de XNU prosee una arquitectura hibrida que tiene 4 componentes clave **Mach** que se encarga de administrar las tares, que contine hilos. Su caracteristica principal es la mensajería, mediante puntos de comunicación (IPC), usa el Mach Interface Generator (MIG) para acortar lo procesos, tenemos tambien **BDS** que implementa procesos y señales UNIX, adema de sistema de archivos y redes, **I/O Kit** es el marco de controladores orientado a objetos y por ultimo **KEXTs** que encargar de enlazar no solo con el I/O Kit, sino tambien con

otras partes del kernel, extensión de red, sistema de archivos (Levin, 2007). Varios de los primeros trabajos de software abierto de Apple son Darwin y WebKit. Darwin es un sistema operativo público que Apple introdujo en el año 2000. Este sistema fue liberado bajo la Licencia Pública de Apple (APSL), que ha sido aceptada por la OSI y la FSF (Levin, 2007)

2.3 FreeRTOS

Figure 2.3: Logo del sistema operativo FreeRTOS (Llamas, 2025)

FreeRTOS es un sistema operativo en tiempo real con un kernel planificador, desarrollado para ejecutarse con microcontroladores. Está compuesto funcionalidades de planificación en tiempo real, comunicación entre tareas, análisis temporal y primitivas de sincronización con el propósito de cumplir con las tareas con el plazos estrictos de ejecución (He and Huang, 2020). En cuanto a su arquitectura hace uso de un microkernel para administrar tareas en tiempo real, casi similar al RTLinux en que admiten núcleo dual, lo que permita separar tareas critica y no criticas (Serino and Cheng, 2019) Esta escrito en lenguaje C estándar y con algunas lineas de código en ensamblador para que se acople a diferentes arquitecturas. Dentro de su componentes claves tenemos el **planificador**, **mecanismos de comunicación**, **sincronización** gracias al uso de colas y semáforos. Incluyendo la estructura de **Task Control Block (TCB)** para la administración de procesos y tareas (Barry, 2018).

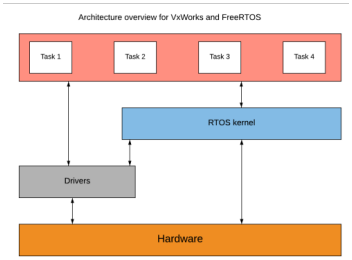

Figure 2.4: Diseño de la arquitectura de FreeRTOS (Llamas, 2025)

FreeRTOS es código abierto con una licencia GPL, lo que permite su uso en aplicaciones comerciales. El código está disponible públicamente y cuenta con documentación extensa en el código fuente en su sitio oficial (The FreeRTOS Project, 2025).

Bibliography

Barry, R. (2018). *Mastering the FreeRTOS Real Time Kernel - A Hands-On Tutorial Guide*. Real Time Engineers Ltd. For FreeRTOS V10.0.0. Accedido el 20 de septiembre de 2025.

Cusme Vera, R. J., Fuentes Prado, S. N., Bravo Vega, C. D., and Gualotuña Quinga, G. B. (2022). Sistemas operativos libres y sistemas operativos privativos: aplicaciones en el campo educativo. *Revista Científica Multidisciplinaria Ogma*, 1(3):85–97.

Ellmann, S. and Emmerich, P. (2019). Porting ixy.rs to redox. Technical report, Technical University of Munich. Accedido el 20 de septiembre de 2025.

Engler, D. R., Kaashoek, M. F., and O'Toole, James, J. (1995). Exokernel: An operating system architecture for application-level resource management. In *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles (SOSP '95)*, pages 251–266, Cambridge, MA, USA. ACM.

GeeksforGeeks (2024a). Microkernel in operating systems. Accedido: 20 septiembre 2025.

GeeksforGeeks (2024b). Monolithic architecture in operating systems. Accedido: 20 septiembre 2025.

GQ Informática (2023). macos, información completa. Accedido el 20 de septiembre de 2025.

Harshvardhan and Irabatti, S. (2023). Study of kernels in different operating systems in mobile devices. *Journal of Online Engineering Education*, 14(1s). Article received: 29 January 2023; Revised: 24 March 2023; Accepted: 20 April 2023.

He, N. and Huang, H.-W. (2020). USE OF FreeRTOS IN TEACHING A REAL-TIME EMBEDDED SYSTEMS DESIGN COURSE. *Computers in Education Journal*, XI(2). Accedido el 20 de septiembre de 2025.

Keetmalin (2017). An introduction on exokernel operating systems. Blog post. Accedido: 20 septiembre 2025.

Keuper, D. (2012). XNU: a security evaluation. Master's thesis, University of Twente. Accedido el 20 de septiembre de 2025.

Levin, J. (2007). Inside the Mac OS X Kernel. In *Proceedings of the 24th Chaos Communication Congress (24C3)*. Accedido el 20 de septiembre de 2025.

Llamas, L. (2025). Freertos cheatsheet, la chuleta con lo imprescindible. Accedido el 20 de septiembre de 2025.

Microsoft Wiki / Fandom (2024). Hybrid kernel. Accedido: 20 septiembre 2025.

Ritter, T. (2019). Disk encryption in redox os. Master's thesis, Masaryk University, Faculty of Informatics. Accedido el 20 de septiembre de 2025.

Saini, R. (2018). Priority based scheduler in rust based redox operating system. Master's thesis, Indian Institute of Technology Madras. Accedido el 20 de septiembre de 2025.

Serino, A. and Cheng, L. (2019). A Survey of Real-Time Operating Systems. Technical Report LU-CSE-19-003, Lehigh University, Department of Computer Science and Engineering, Bethlehem, PA. Accedido el 20 de septiembre de 2025.

Stallings, W. (2023). *Computer Organization and Architecture: Designing for Performance*. Pearson, 12th edition.

Tanenbaum, A. S. and Bos, H. (2023). *Modern Operating Systems*. Pearson, 5th edition.

The FreeRTOS Project (2025). FreeRTOS - Market leading RTOS for embedded systems. Accedido el 20 de septiembre de 2025.

The Redox OS Community (2025). The Redox OS Book. https://doc.redox-os.org/book/. Accedido el 20 de septiembre de 2025.

Wikipedia contributors (2025). Redox (operating system) — Wikipedia, The Free Encyclopedia. [Online; accessed 20-September-2025].

Appendix A

Appendix title

This is Appendix A.

You can have additional appendices too (*e.g.*, `apdxb.tex`, `apdxc.tex`, *etc.*). If you don't need any appendices, delete the appendix related lines from `thesis.tex` and the file names from `Makefile`.