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

Índice general

Índice de cuadros

Índice de figuras

0.1. Arquitecturas de Sistemas Operativos

0.1.1. Arquitectura Monolítica

0.1.2. Arquitectura en Microkernel

0.1.3. Arquitectura Hibrida

0.1.4. Arquitectura en Exokernel

Capítulo 1

Componentes principales de un sistema operativo

Un sistema operativo integra varios componentes básicos que gestionan los recursos de hardware y proveen servicios a los procesos.

1.1. Gestión de procesos

Este diagrama ilustra los estados básicos por los que pasa un proceso durante su ciclo de vida. Por ejemplo, al crearse un proceso entra en el estado Nuevo; luego pasa a Preparado esperando CPU, y cuando la CPU lo atiende entra en Ejecutando. Durante la ejecución, puede quedar Esperando si necesita realizar E/S; finalmente alcanza el estado Terminado al completar su tarea, Como se muestra en la Figura

La gestión de procesos abarca funciones clave en el sistema operativo, por ejemplo la creación/destrucción de procesos, su planificación y los mecanismos de sincronización/comunicación entre ellos **?**. Entre las funciones principales se incluyen:

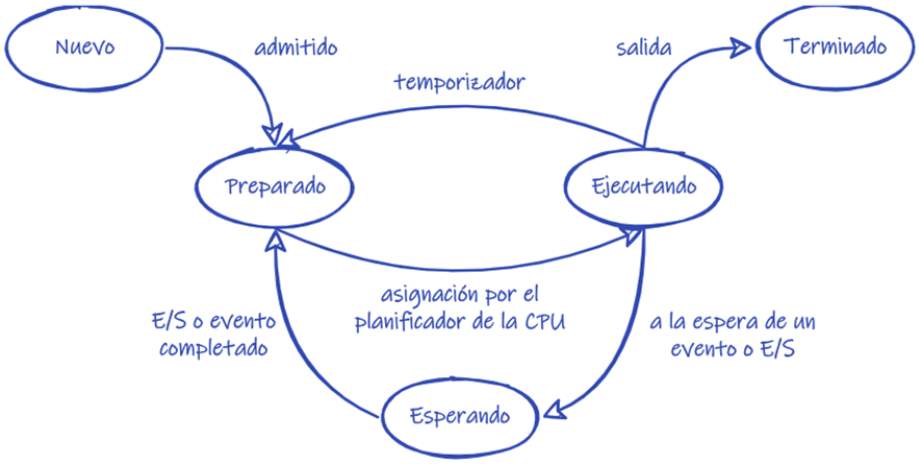

Figura 1.1: diagrama de estados de un proceso. Fuente: **?**

- **Creación y finalización**: El SO inicia nuevos procesos (por ejemplo con llamadas como `fork` en Unix) y elimina procesos completados **?**.
- **Planificación (scheduling)**: El sistema operativo decide el orden en que los procesos usan la CPU (algoritmos FIFO, round robin, prioridades, etc.) **?**.
- **Sincronización y comunicación**: Se emplean mecanismos de comunicación entre procesos (IPC) para que los procesos se comuniquen y sincronicen entre sí **?**, evitando conflictos al acceder a recursos compartidos.

1.2. Gestión de memoria

La gestión de memoria es fundamental para que el sistema operativo asigne y libere espacio de la memoria principal (RAM) a los procesos, aislando cada espacio de direcciones y permitiendo la ejecución concurrente de múltiples programas. En esencia, el administrador de memoria asegura que la CPU pueda cargar en RAM

las instrucciones y datos necesarios de cada proceso. Así, la administración de memoria organiza los procesos de modo que se obtenga la máxima utilidad del espacio disponible, trasladando la información que debe ejecutarse a memoria principal **?**,

Actualmente esta gestión se basa en técnicas de memoria virtual, que permiten al sistema disponer de una memoria lógica más amplia que la física, usando esquemas como la paginación (cada proceso ve un espacio de direcciones independiente). En la figura siguiente se ilustra conceptualmente cómo cada proceso tiene su propia tabla de páginas que mapea sus páginas virtuales (columnas) en marcos de la memoria física (rectángulos coloreados):

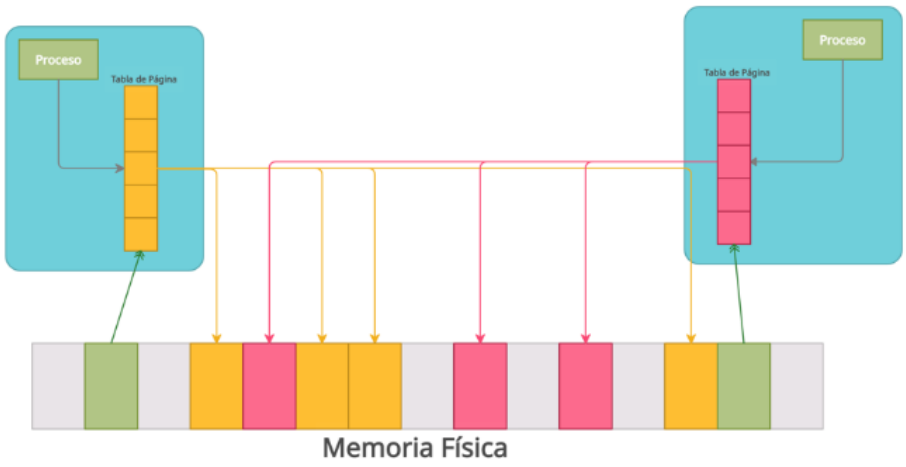

Figura 1.2: Traducción por paginación. Cada proceso (cajas turquesa) posee su propia tabla de páginas (columnas amarilla/rosada) que asigna páginas lógicas a marcos de la memoria física (fila de bloques coloreados). Fuente: **?**

Entre las tareas clave del gestor de memoria se incluyen las siguientes responsabilidades:

- **Control de uso de memoria**
- **Asignación y liberación de memoria**
- **Reubicación (relocation)**
- **Fragmentación y compactación**
- **Memoria virtual**
- **Protección de memoria**

En conjunto, estas funciones garantizan un uso eficiente y seguro del espacio de memoria A modo de ejemplo, la figura anterior muestra cómo la traducción por paginación permite mapear páginas virtuales en marcos físicos, de modo que cada proceso "cree" tener su propio espacio contiguo, mientras que en realidad comparten la RAM de forma ordenada. El gestor de memoria actualiza las tablas de páginas y puede intercambiar páginas con el disco según sea necesario para mantener esta abstracción de memoria virtual.**?**

1.3. Sistema de archivos

1.3.1. Estructura jerárquica

El sistema de archivos organiza los archivos y directorios en forma de árbol invertido. A nivel lógico existe un único directorio raíz (/), nodo principal que contiene todos los demás. Cada nodo del árbol corresponde a un directorio que puede contener subdirectorios, archivos normales o especiales Este diseño jerárquico con nombres úni-

cos por directorio facilita la navegación y la gestión (por ejemplo, evitando colisiones de nombres).

1.3.2. Gestión de datos persistentes

El sistema de archivos provee los medios para almacenar y recuperar datos de forma ordenada en la memoria secundaria (discos, SSD, etc.). La memoria se divide en bloques de tamaño fijo, y el sistema de archivos asigna a cada archivo los bloques necesarios.Además, controla la consistencia de los datos (por ejemplo, mediante journaling u otros esquemas) de modo que la información persista aunque finalicen los procesos o falle el sistema. También gestiona la integridad ante fallos, permitiendo recuperar estructuras de disco sin pérdida de datos. **?**(pag 1)

1.3.3. Operaciones de archivo

El sistema de archivos ofrece llamadas básicas para crear, abrir, leer, escribir, renombrar y eliminar archivos y directorios. Por ejemplo, al invocar open el kernel devuelve un descriptor de archivo, que apunta a una entrada en la tabla de archivos abiertos y al inodo correspondiente. Según Bach (1986) esta tabla relaciona descriptores, entradas de acceso y estructuras de inodos internos. En otras palabras, cada proceso ve un descriptor (un entero), mientras que internamente el sistema mantiene una tabla de archivos que referencia los inodos de los archivos abiertos.**?**(pag. 1)

1.3.4. Asignación de espacio

Los datos de cada archivo se almacenan en bloques de disco. El sistema de archivos controla cuáles bloques están libres y asigna los necesarios a cada archivo. Por ejemplo, en el esquema FAT (File Allocation Table) cada disco tiene una tabla con una entrada por bloque; dicha entrada indica el siguiente bloque del archivo o un marcador especial (libre, defectuoso o fin de archivo)

1.3.5. Seguridad y atributos

El sistema de archivos gestiona los atributos (metadatos) de cada archivo y aplicación de permisos. Por ejemplo, en sistemas Unix cada archivo se describe con un inodo que almacena los permisos de acceso, propietario (UID/GID), fechas de modificación y tipo de archivo. **?**(pag.82)

1.3.6. Journaling

El journaling es un sistema por el cual se pueden implementar transacciones en los sistemas informáticos. También se le conoce como «registro por diario». Se basa en llevar un journal o registro de diario en el que se almacena la información necesaria para restablecer los datos afectados por la transacción en caso de que esta falle; (por ejemplo, corte de energía) al reiniciar simplemente se «reproduce» el journal para completar o deshacer operaciones y restablecer la consistencia. de esta manera, estos sistemas de archivos se pueden restaurar a producción de forma veloz y con menos probabilidad de corrupción. **?**

1.4. Gestión de dispositivos (E/S)

El sistema de E/S del SO actúa como interfaz entre el hardware de los dispositivos (discos, teclados, impresoras, etc.) y el software, ocultando las peculiaridades de cada dispositivo al resto del sistema. **?**(seccion 4.3) Sus componentes principales son:

- **Controladores de dispositivo (drivers)**: Software específico, generalmente provisto por el fabricante, que conoce los detalles del hardware y traduce peticiones genéricas del SO en operaciones concretas sobre el dispositivo. **?**(seccion 4.3)

- **Interfaz genérica de E/S**: El SO ofrece llamadas estándar (open, read, write, close, ioctl, etc.) que los programas usan para interactuar con cualquier dispositivo sin necesitar conocer sus características físicas.(Por ejemplo, los sistemas UNIX es que todos los dispositivos de E/S se representan como un archivo en el sistema de archivos. Esto se puede comprobar rápidamente visitando el directorio /dev, permitiendo operaciones de E/S uniformes) **?**(seccion 4.3)

- **Buffering y caching**: Se enfoca en la sincronización durante una transferencia de datos en curso. Su objetivo es compensar la diferencia de velocidad durante esa operación específica (Buffering).y el caching Se enfoca en la reutilización de datos accedidos recientemente. Su objetivo es anticiparse a futuras solicitudes, almacenando copias de datos para que los accesos subsiguientes sean ultra rápidos.

- **Spooling**: En dispositivos secuenciales no compartibles (como impresoras), el SO encola (spool) los trabajos de varios procesos en almacenamiento intermedio,

gestionándolos en orden sin bloquear a los procesos remitentes.?(seccion 4.3)

La gestión de dispositivos garantiza así que los procesos puedan leer/escribir en hardware diverso usando una interfaz única, mientras el SO optimiza el flujo de datos y protege el acceso concurrente

1.5. Interfaz de usuario

La interfaz de usuario (IU) es el medio por el cual el usuario interactúa con el sistema.

Figura 1.3: Evolución de las interfaces de usuario - de CLI (línea de comandos) a GUI (gráfica) y a NUI (natural). Fuente: **?**

- **CLI (Command Line Interface)**: El usuario escribe comandos en una consola o terminal.
- **GUI (Graphical User Interface)**: Interfaz gráfica con ventanas, iconos y menús (más intuitiva y exploratoria)
- **NUI (Natural User Interface)**: Interacción más directa e intuitiva (por ejemplo táctil, voz o gestos).

Un buen diseño de IU busca usabilidad e intuición, permitiendo al usuario dar órdenes al SO y visualizar información de estado (p.ejemplo. terminales, escritorios, menús de configuración, iconos de carpeta, etc.). El diseño de estas interfaces puede variar ampliamente entre sistemas operativos (texto puro en ciertos sistemas Unix muy básicos, o entornos gráficos complejos en sistemas modernos)

1.6. Seguridad y protección

La seguridad en un sistema operativo garantiza que sólo usuarios autorizados utilicen los recursos del sistema y de la manera correcta Para ello se implementa el principio de menor privilegio, donde cada proceso sólo recibe los permisos estrictamente necesarios. Así, la autentificación identifica al usuario (por ejemplo, pidiendo contraseña) y le asigna permisos adecuados. Por ejemplo, en sistemas como Multics un servicio de autenticación asigna cada proceso a un usuario; si este servicio falla, un proceso podría obtener permisos no autorizados al recibir un usuario equivocado. ?(pag. 50)

1.6.1. Control de acceso

El sistema operativo impone un modelo de control de acceso que asocia permisos con cada recurso (archivos, dispositivos, regiones de memoria, etc.) Cada solicitud (llamada al sistema) de un proceso se verifica contra estos permisos, a través de un monitor de referencia que medía y autoriza cada operación Conceptualmente, esto se puede abstraer como una matriz de acceso: una tabla donde las filas son procesos (o dominios) y las columnas objetos del sistema, y cada celda indica los permisos (leer,

escribir, ejecutar, etc.) que tiene ese sujeto sobre ese objeto Por ejemplo, en Unix cada archivo tiene bits de permiso y un propietario; sólo el propietario puede cambiar los bits de permiso del archivo. ?(pag. 27)

1.6.2. Aislamiento de procesos y protección de memoria

El SO protege la memoria y el contexto de cada proceso para impedir interferencias entre ellos. Cada proceso corre en modo usuario (no privilegiado) en su propio espacio virtual de direcciones; ningún proceso puede leer o escribir directamente la memoria de otro ni del núcleo. En particular, la protección de memoria impide que un usuario acceda a los datos de otros o modifique código/datos críticos del kernel Por ejemplo, la arquitectura de privilegios (rings) restringe instrucciones sensibles a modo kernel (nivel 0); sólo desde ese nivel privilegiado se pueden ejecutar operaciones críticas sobre memoria y dispositivos. ?(pag. 3)

1.6.3. Prevención de intrusiones

Además de la protección interna, el SO incluye mecanismos activos contra amenazas externas o internas. Un sistema de prevención de intrusiones (IPS) supervisa el tráfico y la actividad del sistema para detectar comportamientos anómalos (por firmas conocidas o por desviaciones) y los bloquea antes de que causen daño Por ejemplo, un IPS puede terminar conexiones peligrosas o eliminar contenido maligno automáticamente Estos filtros y cortafuegos integrados en el SO actúan como barrera, rechazando accesos no autorizados (por red o por intentos locales) según políticas definidas. En conjunto, el sistema previene ataques (virus, intrusiones de red, escala-

da de privilegios, etc.) al observar cada petición de acceso y negarla si coincide con patrones o reglas maliciosas. **?**

1.6.4. Auditoría y registro

Para detectar incidentes y comportamientos sospechosos, el SO mantiene logs de auditoría de eventos relevantes (inicio/cierre de sesión, errores de autenticación o de acceso, cambios en permisos, etc.). Cada evento de seguridad se registra con su fecha, usuario y acción intentada. De este modo, es posible revisar los registros para identificar accesos indebidos o intrusiones tras su ocurrencia. Los sistemas de auditoría alertan automáticamente al equipo de seguridad cuando se detecta una anomalía. **?**

1.6.5. Modelos de seguridad clásicos

En los SO con seguridad reforzada también se implementan modelos formales clásicos. El modelo Bell–LaPadula es un modelo de control de acceso obligatorio (MAC) enfocado en confidencialidad: los sujetos y objetos tienen niveles (publico, secreto, top-secret), y se aplica "no leer hacia arriba, no escribir hacia abajo" ("write up, read down") Es decir, un sujeto sólo puede leer datos de igual o menor nivel que el suyo y escribir datos de igual o mayor nivel, evitando filtración de información sensible. En contraste, el modelo Biba protege integridad: invierte las reglas ("read up, write down") Biba asegura que un sujeto no se contamine leyendo información de baja integridad, ni corrompa objetos de alta integridad. En la práctica estos modelos (y variantes como Clark-Wilson) definen estrictamente las relaciones de permiso para garantizar confidencialidad e integridad en sistemas multigrado. **?**

Apéndice A

Appendix title

This is Appendix `??`.

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`.