

Universidad Católica Andrés Bello.
Facultad de Ingeniería.
Escuela de Informática.
Departamento de Telemática.
Sistemas Operativos.



Enunciado del Proyecto

1. TÍTULO Y ALCANCE DEL PROYECTO

Título Específico:

"Simulador de Algoritmos de Planificación de CPU con Análisis de Rendimiento"

Alcance Delimitado:

Proyecto enfocado en la **Unidad 4: Administración del Procesador** con implementación práctica y análisis comparativo básico.

2. DESCRIPCIÓN GENERAL DEL PROYECTO

Objetivo Principal

Desarrollar un simulador que implemente y compare 4 algoritmos principales de planificación de CPU, permitiendo analizar su rendimiento bajo diferentes cargas de trabajo.

Objetivos Específicos

1. **Implementar** 4 algoritmos básicos de planificación de CPU
2. **Simular** procesos con diferentes características
3. **Calcular** métricas de rendimiento básicas
4. **Comparar** resultados entre algoritmos
5. **Presentar** resultados de forma clara

Modalidad de Trabajo

- **Equipo:** 3-4 estudiantes
- **Duración:** 10 días
- **Entrega:** Código fuente, manual de usuario y presentación

3. COMPONENTES TÉCNICOS OBLIGATORIOS

3.1 ALGORITMOS DE PLANIFICACIÓN (OBLIGATORIO - 4 ALGORITMOS)

Algoritmos Básicos (Implementar TODOS) :

1. **FCFS** (First Come, First Served)
2. **SJF** (Shortest Job First) - No apropiativo
3. **Round Robin** con quantum configurable
4. **Prioridades** (con o sin desalojo)

3.2 SIMULACIÓN DE PROCESOS

Características de Procesos a Simular:

Estructura del Proceso:

- └ ID único del proceso
- └ Tiempo de llegada (arrival time)
- └ Tiempo de ráfaga de CPU (burst time)
- └ Prioridad (para algoritmos que la requieran)
- └ Estado actual (Nuevo, Listo, Ejecutando, Terminado)

Generación de Procesos:

- **Manual:** Permitir al usuario crear procesos específicos
- **Por archivo:** Cargar conjuntos de procesos desde archivos de configuración

3.3 MÉTRICAS DE RENDIMIENTO (BÁSICAS OBLIGATORIAS)

Métricas por Proceso:

- **Tiempo de Retorno** (Turnaround Time)
- **Tiempo de Espera** (Waiting Time)
- **Tiempo de Respuesta** (Response Time)

Métricas del Sistema:

- **Tiempo Promedio de Retorno**
- **Tiempo Promedio de Espera**
- **Utilización de CPU**

3.4 INTERFAZ DE USUARIO Y VISUALIZACIÓN (SIMPLIFICADA)

Componentes Mínimos de la Interfaz:

A. Panel de Control:

- Selección de algoritmo de planificación
- Configuración de parámetros básicos (quantum, prioridades)
- Creación/edición de procesos
- Botón de ejecutar simulación

B. Visualización Básica:

- **Métricas calculadas** mostradas en tabla
- **Gráficos básicos** de comparación entre algoritmos
- **Diagrama de Gantt textual** (opcional: gráfico simple)

C. Panel de Resultados:

- Tabla comparativa de métricas por algoritmo
- Conclusión automática de mejor algoritmo para el caso

4. ESPECIFICACIONES TÉCNICAS SIMPLIFICADAS

4.1 Arquitectura del Simulador

SIMULADOR DE PLANIFICACIÓN

```
└── Core/
    ├── scheduler.cpp          # Motor principal de
    planificación
    ├── process.cpp            # Clase/estructura de proceso
    └── algorithms/
        ├── fcfs.cpp           # First Come First Served
        ├── sjf.cpp             # Shortest Job First
        ├── round_robin.cpp     # Round Robin
        └── priority.cpp        # Planificación por prioridades
    └── metrics.cpp            # Cálculo de métricas
└── UI/
    ├── interface.cpp          # Interfaz principal
    └── results_display.cpp    # Mostrar resultados
└── Utils/
    ├── file_handler.cpp       # Manejo de archivos
    └── process_generator.cpp  # Generador de procesos
```

4.2 Plataforma de Desarrollo

Plataforma Única:

- **Eligen UNA plataforma:** Windows, GNU/Linux, o macOS
- **No se requiere portabilidad** entre plataformas
- **Lenguajes recomendados:** C++, Java, o Python

5. CASOS DE PRUEBA ESPECÍFICOS

5.1 Conjuntos de Prueba Obligatorios (REDUCIDOS)

Conjunto 1: Procesos Básicos

Copiar

Proceso	Llegada	Ráfaga	Prioridad
P1	0	8	3
P2	1	4	1
P3	2	9	4
P4	3	5	2

Conjunto 2: Procesos Variados

Copiar

Proceso	Llegada	Ráfaga	Prioridad
P1	0	10	2
P2	2	3	1
P3	4	6	3
P4	6	1	1
P5	8	4	2

Conjunto 3: Caso Personal

- Los estudiantes deben crear un conjunto propio de al menos 4 procesos

6. CRONOGRAMA DETALLADO (10 DÍAS)

Días 1-2: Fundamentos

- Configuración del ambiente de desarrollo
- Implementación de estructuras básicas (Process, Scheduler base)
- Implementación de FCFS

Días 3-4: Algoritmos Principales

- Implementación de SJF
- Implementación de Round Robin

Días 5-6: Algoritmos y Métricas

- Implementación de Priority Scheduling
- Sistema de cálculo de métricas

Días 7-8: Interfaz y Pruebas

- Interfaz básica de usuario
- Casos de prueba y validación

Días 9-10: Finalización

- Documentación (manual de usuario)
- Preparación de presentación
- Entrega final

7. DOCUMENTACIÓN ESPECÍFICA

7.1 Manual de Usuario (5-8 páginas)

Estructura Obligatoria:

1. **Introducción** (1 página)
2. **Instalación** (1-2 páginas)
3. **Guía de Uso** (2-3 páginas)
4. **Ejemplos Prácticos** (1-2 páginas)

7.2 Comentarios en Código

- Comentarios en funciones principales
- Documentación de algoritmos implementados
- Explicación de estructuras de datos básicas

10. CRITERIOS DE EVALUACIÓN ESPECÍFICOS

COMPONENTE 1: IMPLEMENTACIÓN DE ALGORITMOS DE PLANIFICACIÓN

Puntaje: 10 puntos (50% del total)

Criterio	Excelente	Satisfactorio	Insuficiente	Deficiente
Número de Algoritmos Implementados	<ul style="list-style-type: none"> • Implementa los 4 algoritmos básicos (FCFS, SJF, Round Robin, Priority) • Todos los algoritmos están completamente funcionales • Implementación adicional opcional (SRTF o algoritmo avanzado) 	<ul style="list-style-type: none"> • Implementa los 4 algoritmos básicos completos • Algoritmos funcionan correctamente en todos los casos de prueba • Implementación sólida y estable 	<ul style="list-style-type: none"> • Implementa 3 de los 4 algoritmos básicos • Los algoritmos implementados funcionan correctamente • Falta 1 algoritmo básico pero los demás son sólidos 	<ul style="list-style-type: none"> • Implementa 2 o menos algoritmos • Los algoritmos presentan errores de funcionamiento • Implementación incompleta o incorrecta
Correctitud de la Lógica	<ul style="list-style-type: none"> • Lógica de planificación perfectamente implementada • Manejo correcto de estados de proceso • Transiciones de estado apropiadas • Implementación teóricamente sólida 	<ul style="list-style-type: none"> • Lógica correcta en la mayoría de los casos • Estados de proceso bien manejados • Transiciones correctas • Implementación teóricamente sólida • Manejo 	<ul style="list-style-type: none"> • Lógica correcta en los casos básicos • Algunos errores en los casos complejos • Estados de proceso parcialmente correctos • Implementación con desviaciones menores 	<ul style="list-style-type: none"> • Lógica incorrecta o incompleta • Errores en manejo de estados • Transiciones incorrectas • Implementación no sigue la teoría • Errores

	<p>ión fiel a la teoría</p> <ul style="list-style-type: none"> • Casos extremos manejados correctamente 	básico de casos especiales	<ul style="list-style-type: none"> • Casos especiales no considerados 	sistemáticos
Manejo de Parámetros	<ul style="list-style-type: none"> • Round Robin con quantum configurable dinámicamente • Priority con diferentes niveles configurables • Parámetros validados correctamente • Configuración intuitiva para el usuario 	<ul style="list-style-type: none"> • Quantum configurable para RR • Prioridades configurables • Validación básica de parámetros • Configuración funcional 	<ul style="list-style-type: none"> • Quantum fijo pero funcional • Prioridades básicas • Validación mínima • Configuración limitada 	<ul style="list-style-type: none"> • Parámetros fijos o incorrectos • Sin validación • Configuración inexistente • Errores en manejo de parámetros
Robustez del Código	<ul style="list-style-type: none"> • Manejo completo de casos extremos • Validación exhaustiva de entrada • Código no se rompe con entradas inválidas • Mensajes 	<ul style="list-style-type: none"> • Manejo de casos comunes • Validación básica • Recuperación de errores principales • Mensajes de error claros 	<ul style="list-style-type: none"> • Manejo limitado de errores • Validación mínima • Recuperación parcial • Mensajes básicos 	<ul style="list-style-type: none"> • Sin manejo de errores • Sin validación • Aplicación se cuelga con errores • Sin mensajes informativos

	de error informativo s			
--	------------------------	--	--	--

COMPONENTE 2: SISTEMA DE MÉTRICAS Y CÁLCULOS

Puntaje: 4 puntos (20% del total)

Criterio	Excelente	Satisfactorio	Insuficiente	Deficiente
Métricas Implementadas	<ul style="list-style-type: none"> Todas las 6 métricas: Turnaround Time, Waiting Time, Response Time, promedios, CPU Utilization Cálculos matemáticamente correctos Precisión decimal apropiada Métricas adicionales opcionales 	<ul style="list-style-type: none"> 5 de las 6 métricas implementadas Cálculos correctos Precisión adecuada Métricas principales funcionando o perfectamente 	<ul style="list-style-type: none"> 4 de las 6 métricas implementadas Cálculos mayormente correctos Precisión básica Métricas esenciales funcionando 	<ul style="list-style-type: none"> 3 o menos métricas Cálculos incorrectos Precisión inadecuada Métricas no funcionan correctamente
Precisión de Cálculos	<ul style="list-style-type: none"> Resultados 100% precisos Validados con casos conocidos Fórmulas correctamente implementadas Sin errores de redondeo significativos 	<ul style="list-style-type: none"> Resultados precisos en 90%+ casos Fórmulas correctas Errores de redondeo mínimos Validación parcial exitosa 	<ul style="list-style-type: none"> Resultados precisos en 70%+ casos Fórmulas mayormente correctas Algunos errores de cálculo menores Validación limitada 	<ul style="list-style-type: none"> Resultados incorrectos Fórmulas mal implementadas Errores sistemáticos en cálculos Sin validación

	Verificación manual exitosa			
Presentación de Resultados	<ul style="list-style-type: none"> • Tablas comparativas completas y claras • Gráficos básicos de comparación • Identificación automática del mejor algoritmo • Formato profesional y organizado 	<ul style="list-style-type: none"> • Tablas comparativas básicas • Presentación clara de datos • Formato organizado • Comparación entre algoritmos visible 	<ul style="list-style-type: none"> • Presentación básica de métricas • Datos mostrados correctamente • Formato simple pero funcional • Comparación limitada 	<ul style="list-style-type: none"> • Presentación confusa • Datos mal organizados • Sin formato claro • Sin comparación útil

COMPONENTE 3: INTERFAZ DE USUARIO Y CASOS DE PRUEBA

Puntaje: 3 puntos (15% del total)

Criterio	Excelente	Satisfactorio	Insuficiente	Deficiente
Funcionalidad de la Interfaz	<ul style="list-style-type: none"> • Interfaz gráfica intuitiva (GUI) • Controles claros para todos los parámetros • Creación/edición fácil de procesos • Ejecución de simulación simple • Navegación fluida y lógica 	<ul style="list-style-type: none"> • Interfaz funcional (GUI o consola avanzada) • Controles básicos disponibles • Creación de procesos funcional • Ejecución de simulación clara • Navegación 	<ul style="list-style-type: none"> • Interfaz de consola básica • Controles limitados pero funcionales • Funcionalidad mínima presente • Navegación confusa pero usable • Opciones reducidas 	<ul style="list-style-type: none"> • Interfaz muy básica o confusa • Sin controles claros • Difícil de usar • Sin navegación lógica • Funcionalidad muy limitada

		comprendible		
Casos de Prueba	<ul style="list-style-type: none"> • Los 3 conjuntos obligatorios implementados • Conjunto propio bien diseñado • Casos adicionales incluidos • Documentación de resultados esperados • Variedad en tipos de procesos 	<ul style="list-style-type: none"> • Los 3 conjuntos obligatorios • Conjunto propio básico • Resultados documentados • Casos bien ejecutados 	<ul style="list-style-type: none"> • 2 de los 3 conjuntos obligatorios • Conjunto propio simple • Documentación básica • Ejecución correcta 	<ul style="list-style-type: none"> • 1 o ningún conjunto completo • Sin conjunto propio • Sin documentación • Ejecución incorrecta
Visualización de Resultados	<ul style="list-style-type: none"> • Métricas mostradas claramente • Gráficos básicos de comparación • Diagrama de Gantt (textual o gráfico) • Colores y organización clara • Resultados fáciles de interpretar 	<ul style="list-style-type: none"> • Métricas mostradas correctamente • Visualización básica de resultados • Presentación organizada • Resultados comprensibles 	<ul style="list-style-type: none"> • Visualización básica de métricas • Presentación simple • Información mínima mostrada • Resultados interpretables con esfuerzo 	<ul style="list-style-type: none"> • Sin visualización clara • Resultados mal presentados • Información confusa • Sin organización visual

COMPONENTE 4: DOCUMENTACIÓN

Puntaje: 2 puntos (10% del total)

Criterio	Excelente	Satisfactorio	Insuficiente	Deficiente
Manual de Usuario (PDF)	<ul style="list-style-type: none"> • Manual completo 6-8 páginas 	<ul style="list-style-type: none"> • Manual de 5-6 páginas • 	<ul style="list-style-type: none"> • Manual de 4-5 páginas • 	<ul style="list-style-type: none"> • Manual menor a 4 páginas

	<ul style="list-style-type: none"> • Instalación paso a paso con capturas • Guía detallada de uso con ejemplos • Casos prácticos bien explicados • Formato profesional con organización clara • Sección de troubleshooting básico 	<p>Instrucciones de instalación claras</p> <ul style="list-style-type: none"> • Guía de uso básica pero completa • Algunos ejemplos incluidos • Formato organizado • Secciones bien definidas 	<p>Instalación básica explicada</p> <ul style="list-style-type: none"> • Guía de uso mínima • Pocos ejemplos • Formato simple • Organización básica 	<ul style="list-style-type: none"> • Instrucciones incompletas • Guía confusa o inexistente • Sin ejemplos útiles • Formato pobre • Sin organización clara
Comentarios en Código	<ul style="list-style-type: none"> • Comentarios exhaustivos en funciones principales • Documentación de algoritmos implementados • Explicación de estructuras de datos • Comentarios de cabecera en archivos • Estilo consistente y profesional 	<ul style="list-style-type: none"> • Comentarios en funciones principales • Documentación de partes importantes • Explicaciones claras donde necesario • Comentarios de cabecera básicos • Estilo mayormente consistente 	<ul style="list-style-type: none"> • Comentarios básicos en algunas funciones • Documentación mínima • Explicaciones superficiales • Pocos comentarios de cabecera • Estilo inconsistente 	<ul style="list-style-type: none"> • Pocos o ningún comentario útil • Sin documentación interna • Código difícil de entender • Sin comentarios de cabecera • Sin estilo definido

COMPONENTE 5: CALIDAD DE CÓDIGO Y ARQUITECTURA

Puntaje: 1 punto (5% del total)

Criterio	Excelente	Satisfactorio	Insuficiente	Deficiente
Estructura y Organización	<ul style="list-style-type: none"> • Arquitectura modular clara • Separación lógica de responsabilidades • Archivos organizados por funcionalidad • Clases/funciones bien definidas • Código fácil de mantener y extender 	<ul style="list-style-type: none"> • Estructura básica organizada • Separación básica de funciones • Archivos bien organizados • Funciones claramente definidas • Organización lógica comprensible 	<ul style="list-style-type: none"> • Estructura básica presente • Separación mínima de responsabilidades • Organización simple • Funciones identificables • Lógica básica seguable 	<ul style="list-style-type: none"> • Estructura confusa • Sin separación clara de responsabilidades • Archivos desorganizados • Funciones mal definidas • Sin lógica organizacional aparente
Calidad del Código	<ul style="list-style-type: none"> • Código limpio y muy legible • Nombres descriptivos y consistentes • Convenciones de estilo consistentes • Sin código duplicado significativo • Manejo 	<ul style="list-style-type: none"> • Código mayormente limpio • Nombres apropiados y comprensibles • Convenciones básicas seguidas • Duplicación mínima 	<ul style="list-style-type: none"> • Código funcional y legible • Nombres básicos pero comprensibles • Algunas convenciones seguidas • Algo de duplicación presente • Manejo básico de recursos 	<ul style="list-style-type: none"> • Código difícil de leer • Nombres confusos o no descriptivos • Sin convenciones aparentes • Mucha duplicación de código • Problemas en manejo de recursos

	eficiente de recursos	• Manejo adecuado de recursos		
--	-----------------------	-------------------------------	--	--

RESUMEN DE PUNTAJES AJUSTADOS

Componente	Puntaje Máximo	Descripción
Implementación de Algoritmos	10 puntos	Funcionalidad core del simulador (4 algoritmos)
Sistema de Métricas	4 puntos	Cálculos y análisis de rendimiento básico
Interfaz y Casos de Prueba	3 puntos	Experiencia de usuario y validación
Documentación	2 puntos	Manual de usuario y comentarios en código
Calidad de Código	1 punto	Estructura y organización
TOTAL	20 puntos	Puntaje mínimo para aprobar: 10 puntos

11. ENTREGABLES ESPECÍFICOS

11.1 Código Fuente

- **Código compilable** con instrucciones claras
- **Makefile/script de compilación**
- **Archivos de configuración** para casos de prueba

11.2 Ejecutable

- **Aplicación funcional** lista para usar
- **Archivos de ejemplo** con casos de prueba
- **Datos de demostración**

11.3 Documentación

- **Manual de usuario** (PDF)
- **Comentarios en código**

11.4 Presentación

- **Demo en vivo** (15 minutos)
 - **Slides** (máximo 15 diapositivas)
 - **Comparación de resultados**
 - **Conclusiones y lecciones aprendidas**
-

PREGUNTAS FRECUENTES (FAQ) - PROYECTO SIMULADOR DE PLANIFICACIÓN DE CPU

1. ¿Cuántos algoritmos debo implementar exactamente?

Respuesta: Debes implementar **4 algoritmos básicos obligatorios:**

- FCFS (First Come, First Served)
- SJF (Shortest Job First)
- Round Robin
- Priority Scheduling

Si implementas los 4 correctamente, obtienes la puntuación completa. Un 5to algoritmo (como SRTF) puede darte puntos de bonificación.

2. ¿En qué lenguaje de programación puedo desarrollar el proyecto?

Respuesta: Puedes usar cualquiera de estos lenguajes:

- **C++** (recomendado para sistemas operativos)
- **Java** (buena opción para interfaces gráficas)
- **Python** (más fácil para principiantes)

Elige el lenguaje con el que tu equipo se sienta más cómodo y tenga más experiencia.

3. ¿Debo hacer interfaz gráfica obligatoriamente?

Respuesta: No es obligatorio. Puedes elegir entre:

- **Interfaz gráfica (GUI):** Más puntos en visualización
- **Interfaz de consola avanzada:** Aceptable si es clara y funcional
- **Interfaz de consola básica:** Mínimo aceptable

Lo importante es que sea **fácil de usar** y permita configurar parámetros y ver resultados claramente.

4. ¿Qué significa exactamente "tiempo de ráfaga" en los casos de prueba?

Respuesta: Tiempo de ráfaga (Burst Time) es **cuánto tiempo necesita el proceso para ejecutarse completamente** en la CPU.

Ejemplo: Si un proceso tiene ráfaga = 8, significa que necesita 8 unidades de tiempo de CPU para terminar.

Es diferente del tiempo de llegada (cuándo llega) y del tiempo de espera (cuánto espera en cola).

5. ¿Cómo calculo las métricas de rendimiento?

Respuesta: Las fórmulas básicas son:

Tiempo de Retorno = Tiempo de Finalización - Tiempo de Llegada

Tiempo de Espera = Tiempo de Retorno - Tiempo de Ráfaga

Tiempo de Respuesta = Tiempo de Primera Ejecución - Tiempo de Llegada

Promedios = Suma de todos los tiempos / Número de procesos

CPU Utilization = (Tiempo total de CPU ocupada / Tiempo total) × 100

6. ¿Qué debe incluir exactamente el manual de usuario?

Respuesta: El manual debe tener **5-8 páginas** con:

1. **Introducción:** Qué hace el simulador
2. **Instalación:** Cómo instalar y ejecutar (paso a paso)
3. **Guía de uso:** Cómo usar cada función
4. **Ejemplos:** Casos prácticos con capturas de pantalla
5. **Troubleshooting:** Problemas comunes y soluciones

Debe estar en **formato PDF** y tener capturas de pantalla.

7. ¿Puedo usar bibliotecas externas o frameworks?

Respuesta: Sí, pero con moderación:

- **Para interfaz gráfica:** Qt, Tkinter, Swing, JavaFX (recomendado)
- **Para gráficos:** matplotlib (Python), JFreeChart (Java)
- **Evita:** Bibliotecas muy complejas que hagan el trabajo por ti

El **algoritmo de planificación debe ser implementado por ustedes**, no usar bibliotecas que ya lo hagan.

8. ¿Cómo manejo el quantum en Round Robin?

Respuesta: El quantum debe ser **configurable por el usuario**:

```
// Ejemplo conceptual
if (tiempo_ejecutado >= quantum) {
    // Interrumpir proceso actual
    // Mover al final de la cola
    // Seleccionar siguiente proceso
}
```

Valores típicos: quantum = 2, 4, 6 unidades de tiempo. **Debe permitir:** Cambiar el quantum antes de ejecutar la simulación.

9. ¿Qué pasa si mi código no compila el día de la entrega?

Respuesta: La respuesta parece obvia para este apartado, pero veamos qué hacer.

Para evitarlo:

- Prueba tu código en la máquina donde lo vas a presentar
 - Incluye instrucciones de compilación claras
 - Haz pruebas finales 1-2 días antes de la entrega
 - Ten un respaldo de una versión que funcione
-

CONSEJOS ADICIONALES:



Errores Comunes a Evitar:

- No probar con los casos de prueba obligatorios
- Interfaz muy complicada que consume mucho tiempo
- No validar entradas del usuario
- Documentación incompleta o sin capturas

Para Obtener Buena Calificación:

- Enfócate primero en que los 4 algoritmos funcionen correctamente
- Prueba exhaustivamente con los casos obligatorios
- Haz una interfaz simple pero clara
- Documenta bien el código y el manual de usuario
- Prepara la presentación con todos los integrantes



Gestión de Tiempo:

- **Días 1-4:** Algoritmos básicos funcionando
 - **Días 5-7:** Métricas e interfaz
 - **Días 8-9:** Pruebas y documentación
 - **Día 10:** Preparación de presentación y entrega
-

13. RECURSOS DE APOYO

13.1 Bibliografía Específica

- Silberschatz, Galvin, Gagne: "Operating System Concepts"
- Capítulo 5
- Tanenbaum: "Modern Operating Systems" - Capítulo 2
- Stallings: "Operating Systems" - Capítulo 9

13.2 Recursos en Línea

- Simuladores de referencia para comparar resultados
- Calculadoras de métricas de planificación
- Ejemplos de diagramas de Gantt

13.3 Herramientas Recomendadas

- **IDEs:** Visual Studio Code, CLion, IntelliJ IDEA
- **Control de versiones:** Git + GitHub
- **Documentación:** LaTeX, Markdown, Word
- **Diagramas:** Draw.io, Lucidchart

Fechas Importantes

- **Anteproyecto:** Semana 11
- **Revisión de avance:** Semana 13
- **Entrega final:** Semana 14
- **Presentaciones:** Semana 14 y 15