

## Test de Evaluación Trabajo Práctico Fuselage – Entrega Final

### Mecánica de Evaluación:

Las evaluación de trabajo práctico constará de una serie de pruebas **obligatorias** y **no obligatorias**. El cumplimiento de las pruebas de carácter obligatorio hace corresponder un **4 (cuatro)** como nota grupal. Asimismo, cada prueba no obligatoria que sea cumplida sumará puntos los cuales estarán especificados en las mismas.

Es posible también que el ayudante asignado al grupo pueda sumar calificación en concepto de *calidad de trabajo práctico*. Esto involucra, uso apropiado de las herramientas que proporciona la cátedra, calidad de código, modularización, etc.

Una vez finalizada la evaluación de trabajo práctico, la nota asignada al grupo corresponderá a la nota individual de cada uno de los miembros. Durante el coloquio, cuando cada uno exponga sus conocimientos, la nota individual podrá ser modificada de acuerdo a lo que el evaluador disponga.

### Consideraciones a la hora de las evaluación:

- No esta permitido editar código.
- No esta permitido editar makefiles, por lo que el código debe compilar correctamente.
- La instalación no debería tardar más de 15 minutos.
- La evaluación durará aproximadamente 45 minutos.
- El coloquio durará aproximadamente 60 minutos.
- El desempeño y participación de todos los miembros del grupo durante las pruebas formará parte de la evaluación.
- Cualquier característica del sistema no cubierta por esta pruebas que el alumno desee verificar, debe ser informado al ayudante, el cual decidirá en base a su criterio la evaluación de dicha característica.

## Requerimientos de Evaluación

El objetivo es controlar la correcta aplicación de las restricciones impuestas para el código y diseño del trabajo práctico.

- Controlar que en la compilación no existan bibliotecas externas, que no hayan sido desarrolladas por el grupo y que no hayan sido permitas o especificadas en el Trabajo Práctico ( *libfuse* y *libicu* ).
- Corroborar el uso de ***select***, ***poll*** o ***epoll***.
- Corroborar el uso de ***Mapping File Into Memory*** o ***Unlocked Stream Operations***.
- Corroborar el uso ***posix\_madvise*** o ***posix\_fadvise***.
- Corroborar el uso de ***sockets unix*** en la consola del Proceso Planificador de Disco.
- Corroborar el uso de ***fork+exec*** para ejecutar la consola del Proceso Planificador de Disco.
- Corroborar el uso de bloques en el Proceso File System

## Información del Sistema:

Complete con los datos relacionados con su implementación:

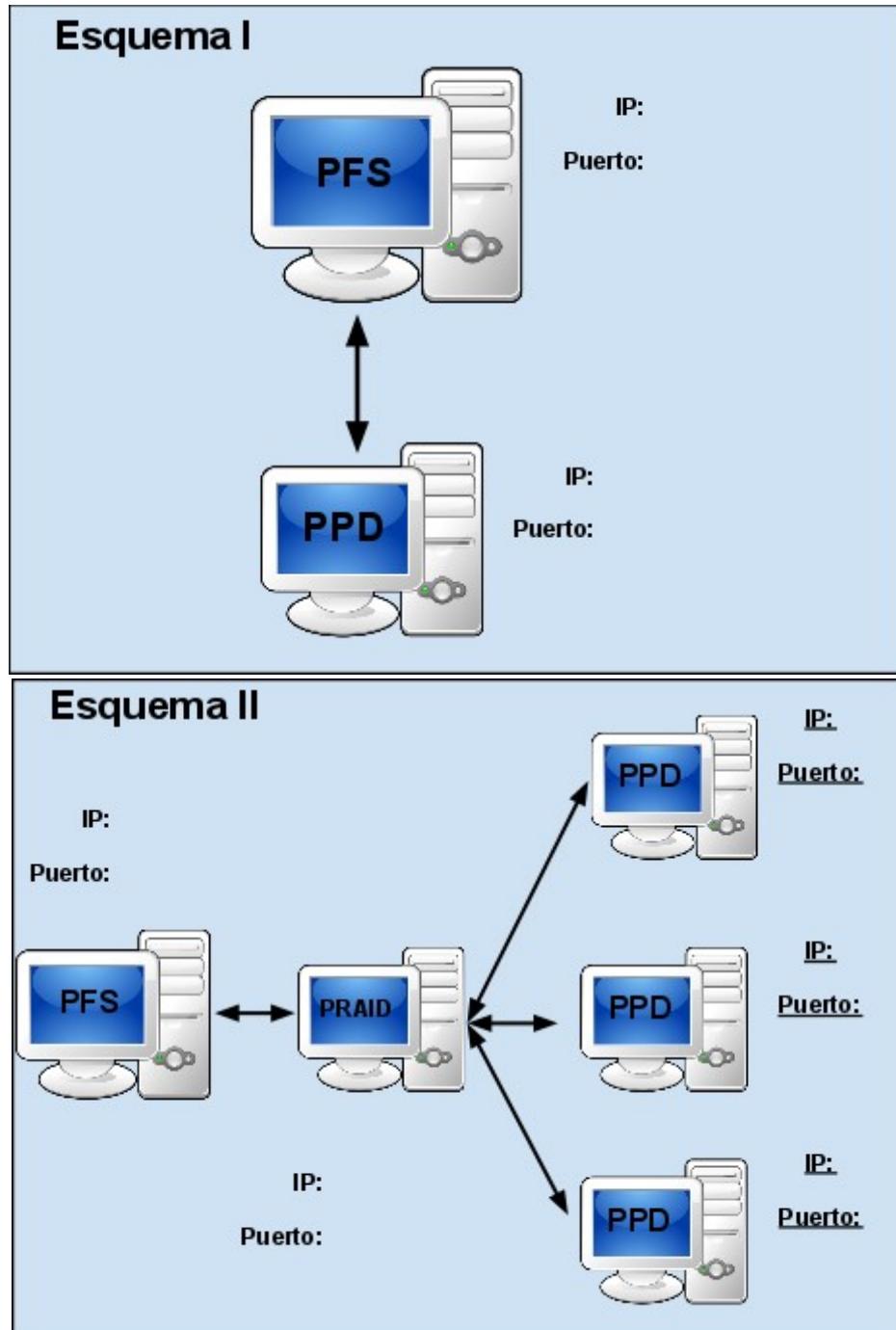
- *Tamaño de Bloque:*
- *Algoritmo de Planificación con Inanición:*
- *Algoritmo de Planificación sin Inanición:*

## Cuestionario Técnico

- Preguntar cómo implementaron el algoritmo de distribución de pedidos en el Proceso RAID.
- Preguntar sobre la FAT si esta en memoria.
- Preguntar que algoritmos eligieron para el PPD.

## Configuración del Sistema:

Para las evaluaciones se manejarán 2 esquemas de prueba, cada prueba indicará que esquema necesita para ser desarrollada. Complete los esquemas con los datos necesarios:



# Pruebas

En todas las pruebas la condiciones de prueba son las siguientes, a menos que la prueba especifique lo contrario:

- Tiempo de Lectura del PPD: 0
- Tiempo de Escritura del PPD: 0
- Cantidad máxima de conexiones del PFS: 1

Formato de *Archivo de Volumen* a utilizar:

- ***mkfs.msdos -F 32 -S 512 -s 8 -C big\_fat32.disk 2097152***
  - ***CHS(8192, 1, 512)***
- ***mkfs.msdos -F 32 -S 512 -s 8 -C small\_fat32.disk 524288***
  - ***CHS(1024, 1, 1024)***

Prueba 1	Lectura de Archivos y Directorios
Desarrollo	<p><b>Esquema:</b> [1] - Sin Cache</p> <p><b>Archivo de Volumen:</b> <i>big_fat32.disk</i></p> <p><b><u>Pasos:</u></b></p> <ol style="list-style-type: none"> <li>1. Listar el punto de montaje.</li> <li>2. Listar los subdirectorios del punto de montaje.</li> <li>3. Intentar listar un subdirectorio inexistente.</li> <li>4. Intentar acceder a un subdirectorio inexistente, usando el comando <b>cd</b> <i>&lt;path&gt;</i>.</li> <li>5. Verificar el estado del archivo: <b>file1_1c.txt</b> <b>md5:</b> 623ca40ea676755bcf47ab2d0c362a97</li> <li>6. Verificar el estado del archivo: <b>file2_1c.txt</b> <b>md5:</b> 9771ad153724f26907f5ba9143f07977</li> <li>7. Verificar el estado del archivo: <b>file3_2c.txt</b> <b>md5:</b> 1268b7e47e0ff5ec60796f207a448cc5</li> <li>8. Verificar el estado del archivo: <b>file4_5c.txt</b> <b>md5:</b> d47bad0ad6e52be8bc7023b1fca2b6e5</li> <li>9. Verificar el estado del archivo: <b>f5_25601c.txt</b> <b>md5:</b> 21c1bbca9c109a45a351ba20a395f29b</li> <li>10. Verificar el estado del archivo: <b>empty.txt</b> <b>md5:</b> d41d8cd98f00b204e9800998ecf8427e</li> </ol> <p><b><u>Nota:</u></b> Cuando dice “<i>verificar el estado</i>” se deberá: verificar la cantidad de clusters asociados, el md5sum del archivo.</p>

Prueba 2	Escritura en Archivos
Desarrollo	<p><b>Esquema:</b> [1] - Sin Cache</p> <p><b>Archivo de Volumen:</b> <i>big_fat32.disk</i></p> <p><b>Requisitos:</b> <u>Prueba 1</u></p> <p><b><u>Pasos:</u></b></p> <ol style="list-style-type: none"> <li>1. Ejecutar el programa <b><i>./write_file "m" file1_1c.txt 500</i></b></li> <li>2. Verificar que haya pisado los primeros 128 bytes con 500 letras 'm'.</li> <li>3. Verificar el <b>md5</b> del archivo luego de la escritura. <ol style="list-style-type: none"> <li>a. <b>md5: 366a337af056cd6d6e356f153246fa10</b></li> </ol> </li> <li>4. Validar el tamaño del archivo usando el comando <b>stat</b>.</li> <li>5. Verificar los clusters asociados al archivo y los clusters libres en el FileSystem.</li> <li>6. Ejecutar el programa <b><i>./write_file "n" file1_1c.txt 6000</i></b></li> <li>7. Verificar que haya pisado los primeros 500 bytes con 6000 letras 'n'.</li> <li>8. Validar el <b>md5</b> del archivo luego de la escritura. <ol style="list-style-type: none"> <li>a. <b>md5: 9801f4c50f352bd2be3f57626088356f</b></li> </ol> </li> <li>9. Validar el tamaño del archivo usando el comando <b>stat</b>.</li> <li>10. Verificar los clusters asociados al archivo y los clusters libres en el FileSystem.</li> <li>11. Ejecutar el comando <b><i>echo "mm" &gt;&gt; file1_1c.txt</i></b> y luego comprobar el md5 <ol style="list-style-type: none"> <li>a. <b>md5: 64807ac73efcb54d317fedd4e4a87df8</b></li> </ol> </li> </ol>

Prueba 3	Truncar Archivos
Desarrollo	<p>Esquema: [1] - Sin Cache</p> <p>Archivo de Volumen: <i>big_fat32.disk</i></p> <p>Requisitos: <u>Prueba 1</u></p> <p><u>Pasos:</u></p> <ol style="list-style-type: none"> <li>1. Verificar los clusters libres.</li> <li>2. Ejecutar el comando <b><i>truncate file3_2c.txt -s 0</i></b> <ul style="list-style-type: none"> <li>○ <b>md5sum: d41d8cd98f00b204e9800998ecf8427e</b></li> <li>○ Libera clusters asociados, esto se validará con el comando <b><i>finfo file3_2c.txt</i></b> del proceso PFS. Por ende tiene que aumentar la cantidad de clusters libres</li> <li>○ Modifica el tamaño del archivo, esto se validará con el comando <b><i>stat file3_2c.txt</i></b></li> </ul> </li> <li>2. Ejecutar el comando <b><i>truncate file3_2c.txt -s 12000</i></b> <ul style="list-style-type: none"> <li>○ Actualiza el tamaño del archivo.</li> <li>○ Agrega nuevos clusters.</li> <li>○ <b>md5: 21d9938f335c6bfab0eeaed58673b073</b></li> </ul> </li> <li>3. Ejecutar el comando <b><i>truncate file3_2c.txt -s 3000</i></b> <ul style="list-style-type: none"> <li>○ Actualiza el tamaño del archivo.</li> <li>○ Remueve los clusters no utilizados.</li> <li>○ <b>md5: 0efa007088f326bbc072c34315f3edb8</b></li> </ul> </li> </ol>

<b>Prueba 4</b>	<b>Rename/Move de Archivos y Carpetas [ Opcional 1pts ]</b>
<b>Desarrollo</b>	<p><b>Esquema:</b> [1] - Sin Cache</p> <p><b>Archivo de Volumen:</b> <i>big_fat32.disk</i></p> <p><b>Requisitos:</b> <u>Prueba 1</u></p> <p><b><u>Pasos:</u></b></p> <ol style="list-style-type: none"> <li>1. Ejecutar el comando <b><i>mv file4_5c.txt file4.txt</i></b> <ul style="list-style-type: none"> <li>○ ls en la carpeta</li> <li>○ stat</li> <li>○ <b>md5: d47bad0ad6e52be8bc7023b1fca2b6e5</b></li> </ul> </li> <li>2. Ejecutar el comando <b><i>mv file4.txt ./Carpeta1/file4.txt</i></b> <ul style="list-style-type: none"> <li>○ ls de la carpeta origen</li> <li>○ ls de la carpeta destino</li> <li>○ stat</li> <li>○ <b>md5: d47bad0ad6e52be8bc7023b1fca2b6e5</b></li> </ul> </li> </ol>

<b>Prueba 5</b>	<b>Crear y Borrar, Archivos y Directorios [ Opcional 1pts ]</b>
<b>Desarrollo</b>	<p><b>Esquema:</b> [1] - Sin Cache</p> <p><b>Archivo de Volumen:</b> <i>big_fat32.disk</i></p> <p><b>Requisitos:</b> <u>Prueba 1</u></p> <p><b><u>Pasos:</u></b></p> <ol style="list-style-type: none"> <li>1. Ejecutar el comando <b><i>mkdir pruebas</i></b> <ul style="list-style-type: none"> <li>○ cd pruebas</li> <li>○ ls .</li> <li>○ ls ..</li> </ul> </li> <li>2. Ejecutar dentro de la carpeta “pruebas” el comando <b><i>echo “hola” &gt; lala.txt</i></b> <ul style="list-style-type: none"> <li>○ <b>md5: 916f4c31aaa35d6b867dae9a7f54270d</b></li> <li>○ ls</li> <li>○ stat</li> </ul> </li> <li>3. Ejecutar el comando <b><i>rm -rf pruebas</i></b> <ul style="list-style-type: none"> <li>○ fsinfo (cluster liberados = 2, 1 archivo y 1 directorio)</li> </ul> </li> </ol>

Prueba 6	Integridad de la FAT [ Opcional 1,5pt ]
Desarrollo	<p><b>Esquema:</b> [1] - Sin Cache</p> <p><b>Archivo de Volumen:</b> small_fat32.disk</p> <p><b>Cantidad máxima de conexiones:</b> 8</p> <p><b>Requisitos:</b> <u>Prueba 5</u></p> <p><b>Pasos:</b></p> <ul style="list-style-type: none"> <li>• Ejecutar el programa <b>./concurrent_write 8 "abcdefgh" 33554432</b>  <i>( Este lanza 8 threads que escriben 32 mb de información en paralelo, el primer archivo contiene todas 'a', el segundo todas 'b', etc ... )</i> <ul style="list-style-type: none"> <li>○ md5 concurrent_file0.txt: bc3d7c2ff64219e33239f2e13c2d21db</li> <li>○ md5 concurrent_file1.txt: 168fe375f6f1fc00911c6130ad3bc6ec</li> <li>○ md5 concurrent_file2.txt: c5bfce7cde37774ab5fc1f20ed846ab5</li> <li>○ md5 concurrent_file3.txt: cdfc11991c34bbfc55ecc7c5d7d8c875</li> <li>○ md5 concurrent_file4.txt: d58b260c5cda8e0265d478e16675687c</li> <li>○ md5 concurrent_file5.txt: 6d19c359472a367db42ee3dcf34eb9fd</li> <li>○ md5 concurrent_file6.txt: 4a74202cbad97e754a708a0eb55abf38</li> <li>○ md5 concurrent_file7.txt: 2d7bd0deecd1a641cd4f621ccf85fe16</li> </ul> </li> </ul>

Prueba 7	Sincronizar un disco en el PRAID
Desarrollo	<p><b>Esquema:</b> [2] - Sin Cache</p> <p><b>Archivo de Volumen:</b> <i>small_fat32.disk</i></p> <p><b>Requisitos:</b> <u>Prueba 1</u>, <u>Prueba 2</u> y <u>Prueba 3</u></p> <p><b><u>Pasos:</u></b></p> <ol style="list-style-type: none"> <li>1. Utilice el comando <code>mkfs.msdos</code> que uso para crear <b><i>small_fat32.disk</i></b> y genere 2 discos vacios; <b><i>small_fat32.disk2</i></b> y <b><i>small_fat32.disk3</i></b></li> <li>2. Ejecute el comando <b><i>md5sum</i></b> sobre <b><i>small_fat32.disk</i></b> y guarde el hash obtenido.</li> <li>3. Levante el equema con solo 1 PPD, el cual apunte a <b><i>small_fat32.disk</i></b>.</li> <li>4. Levante un PPD que apunte a <b><i>small_fat32.disk2</i></b> y haga que se conecte al PRAID.</li> <li>5. Una vez sincronizado, ejecute el comando <b><i>md5sum</i></b> sobre <b><i>small_fat32.disk2</i></b> y compare el hash obtenido con el obtenido con <b><i>small_fat32.disk</i></b> ( <i>Ambos deben ser iguales</i> ).</li> <li>6. Repita los 2 ultimos pasos pera para el <b><i>small_fat32.disk3</i></b>.</li> </ol>

Prueba 8	Probar Distribución de Carga del PRAID
Desarrollo	<p><b>Esquema:</b> [2] - Sin Cache</p> <p><b>Archivo de Volumen:</b> <i>small_fat32.disk</i></p> <p><b>Requisitos:</b> <u>Prueba 7</u></p> <p><b><u>Pasos:</u></b></p> <ol style="list-style-type: none"> <li>1. Ejecutar el comando <b><i>md5sum</i></b> sobre el archivo <b><i>f5_25601c.txt</i></b> y espere a obtener el hash.</li> <li>2. Verificar que los pedidos de lectura generados anteriormente, sean distribuidos entre los PPD de acuerdo al algoritmo especificado.</li> <li>3. Ejecute nuevamente el comando <b><i>md5sum</i></b> sobre el mismo archivo, pero mientras este sigue generando pedidos de lectura finalice la ejecucion de uno de los PPD y luego de otro. Quedando asi solo un PPD funcionando.</li> <li>4. El hash obtenido debe ser el mismo.</li> </ol>

Prueba 9	Test de Stress del PRAID [ Opcional 1,5pt ]
Desarrollo	<p><b>Esquema:</b> [2] - Sin Cache</p> <p><b>Archivo de Volumen:</b> <i>small_fat32.disk</i></p> <p><b>Requisitos:</b> <u>Prueba 8</u></p> <p><b>Pasos:</b></p> <ol style="list-style-type: none"> <li>1. Ejecutar los comandos: <ul style="list-style-type: none"> <li>○ <b><i>md5sum f5_25601c.txt</i></b></li> <li>○ <b><i>dd if=/dev/zero of=./wfile.txt bs=150MB count=1</i></b></li> </ul> </li> <li>2. Durante la ejecución de ambos, finalizar la ejecución de 2 PPD</li> <li>3. Una vez finalizados, ejecutelos nuevamente, haciendolos que se conecten nuevamente al PRAID</li> <li>4. Tras unos pocos segundos, finalice nuevamente 1 de los PPD recientemente conectados.</li> <li>5. Tras finalizar ambos y verificar el md5 ( <b><i>21c1bbca9c109a45a351ba20a395f29b</i></b> ) dio correcto, monte uno de los <b><i>small_fat32.disk</i></b> utilizando el comando <b><i>mount -o loop</i></b> y verifique que el tamaño del archivo <b><i>wfile.txt</i></b> sea el correcto.</li> </ol>

Prueba 10	Probar Algoritmo con Inanición
Desarrollo	<p><b>Esquema:</b> [1] - Sin Cache</p> <p><b>Archivo de Volumen:</b> <i>small_fat32.disk</i> - con CHS(256, 1, 4096)</p> <p><b>Requisitos:</b> <u>Prueba 1</u></p> <p><b>Pasos:</b></p> <ol style="list-style-type: none"> <li>1. Ejecutar el proceso: <ul style="list-style-type: none"> <li>○ <b><i>make_starvation 4 file1.txt</i></b></li> </ul> </li> <li>2. Desde la consola del PPD ejecute el comando: <ul style="list-style-type: none"> <li>○ <b><i>trace 1 900000</i></b></li> </ul> </li> <li>3. Mientras el comando <b><i>make_starvation</i></b> se encuentre en ejecución, el comando <b><i>trace</i></b> no debe retornar.</li> </ol> <p><b>Nota:</b> Esta prueba puede verse sujeta a mas pasos dependiendo el algoritmo implementado.</p>

<b>Prueba 11</b>	<b>Probar Algoritmo sin Inanición</b>
<b>Desarrollo</b>	<p><b>Esquema:</b> [1] - Sin Cache</p> <p><b>Archivo de Volumen:</b> <i>small_fat32.disk</i> - con CHS(256, 1, 4096)</p> <p><b>Requisitos:</b> <u>Prueba 1</u></p> <p><b><u>Pasos:</u></b></p> <ol style="list-style-type: none"> <li>1. Ejecutar el proceso: <ul style="list-style-type: none"> <li>○ <b><i>make_starvation 4 file1.txt</i></b></li> </ul> </li> <li>2. Desde la consola del PPD ejecute el comando: <ul style="list-style-type: none"> <li>○ <b><i>trace 1 900000</i></b></li> </ul> </li> <li>3. El cabezal debe poder alcanzar el sector 1 y 900000 sin problemas.</li> <li>4. Verifique esto, repitiendo los pasos anteriores 2 veces mas.</li> </ol> <p><b>Nota:</b> Esta prueba puede verse sujeta a mas pasos dependiendo el algoritmo implementado.</p>

<b>Prueba 12</b>	<b>Uso de Cache del PFS</b>
<b>Desarrollo</b>	<p><b>Esquema:</b> [1] - Con Cache ( Tamaño 16Kb )</p> <p><b>Archivo de Volumen:</b> <i>small_fat32.disk</i></p> <p><b><u>Pasos:</u></b></p> <ul style="list-style-type: none"> <li>● Correr el proceso: <ul style="list-style-type: none"> <li>○ <b><i>test_cache {archivo} {size_cache}</i></b></li> </ul> </li> </ul> <p><b>Nota:</b> Esta prueba puede involucrar el uso de mas pasos, para verificar en mayor detalle el uso de la cache.</p>

## Herramientas para verificar el estado del FileSystem:

- **Consola PFS**
  - *fsinfo: Clusters libres - Clusters utilizados*
  - *finfo: Clusters asignados a un archivo*
- **Terminal Linux:**
  - *stat <file>: tamaño del archivo*
  - *ls: listado de directorios*
  - *md5sum <file>: genera un código md5 para el archivo indicado*
  - *md5sum <file1> <file2> ... <file\_n> -c <file\_md5>: compara el checksum de los archivos pasados con los que contiene el archivo file\_md5*