UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería en Sistemas de Información

Re Distinto

Cátedra de Sistemas Operativos

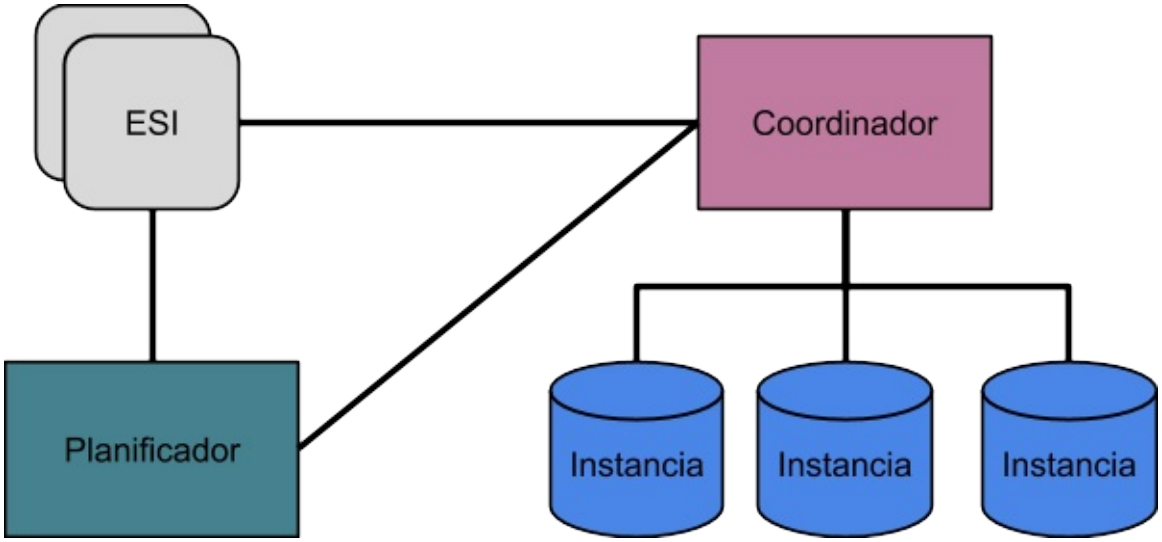
Trabajo práctico Cuatrimestral

- 1C2018 -

Table of Contents

Re Distinto

Arquitectura



La siguiente es la arquitectura del trabajo práctico. Las líneas representan conexiones, que serán realizadas utilizando sockets TCP/IP. La conexión se llevará a cabo utilizando ips y puertos definidos en un archivo de configuración.

Ejecutor Sentencias Interactivas (ESI)

Los Procesos ESI son el punto de entrada al Sistema. Estos correrán un script con instrucciones a ser ejecutadas(^1). Sú única función es la de interpretar de a una las líneas de un programa *(que se le pasará como argumento)* a medida que el Planificador se lo indique. Para resolver las distintas sentencias del programa a ser interpretado; deberá colaborar con el coordinador para bloquear/liberar recursos o datos.

Conexión

Al iniciarse un Proceso ESI, éste se conectará tanto al Planificador como al Coordinador de Re Distinto e intercambiará los necesarios mensajes iniciales para confirmar la conexión con ambos. Una vez establecida la conexión, y a medida que así lo dicte el Planificador, irá parseando las sentencias del script y transmitiéndolas al Coordinador de Re Distinto, quién retornará un mensaje informando el resultado.

Parser

Por cada una de las sentencias interpretadas, es necesario realizar un proceso de parseo, con el fin de traducir la cadena de caracteres en una operación entendible por el Sistema Re Distinto. Para esto, la cátedra proveerá a los alumnos del Parser de Re Distinto (ParSI), ya que la implementación del mismo no concierne a los temas de la materia.

Interacción con Coordinador y Planificador

1. Se recibe una solicitud de ejecución desde el Planificador de Re Distinto a través de la conexión.
2. Se obtiene la próxima sentencia a ser ejecutada según el programa a interpretar.
3. En caso de ser necesario, se envía una solicitud al Coordinador de Re Distinto.
4. El Coordinador retorna un resultado, ya sea por éxito de la operación o por un fallo de la misma.
5. Se le envía el resultado de este al planificador.

Configuración

Campo	Tipo	Ejemplo
IP de Conexión al Coordinador	[cadena]	`"127.0.0.1"`
Puerto de Conexión al Coordinador	[numérico]	`8000`
IP de Conexión al Planificador	[cadena]	`"127.0.0.2"`
Puerto de Conexión al Planificador	[numérico]	`8001`

^1: Para conocer en profundidad las instrucciones ver el Anexo I.

Planificador

Es el proceso encargado de orquestar la ejecución de los procesos ESI. Los procesos ESI llegarán a una cola de listos, donde se los planificará para su próxima ejecución según un algoritmo predeterminado por archivo de configuración.

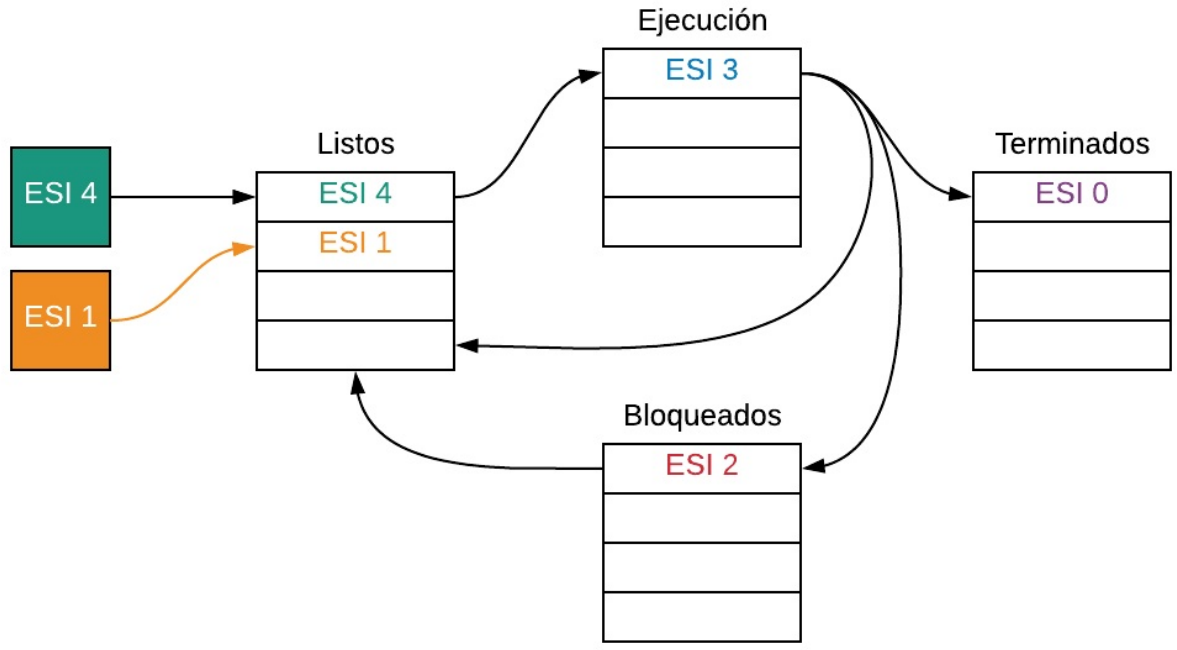
Arazón de simplificar el desarrollo, la ejecución de cada línea de código *del script a ser interpretado por el ESI* **será atómica** y cada vez que una instrucción sea ejecutada se le informará al planificador. En este sentido el Planificador actúa como un Clock del sistema. No se deberá re-planificar los ESI a seguir ejecutando al recibir la confirmación de sentencia ejecutada.

En caso de que un proceso ESI no pueda seguir ejecutando debido a un bloqueo, deberá ser enviado a la cola de proceso bloqueados, esperando a que quien bloqueó la key utilice un *store* y libere la misma.

Por otro lado, en el caso de que el ESI haga una operación de GET sobre un recurso libre, no deberá moverse del estado de ejecución ni ser re-planificado (a menos que el algoritmo del planificador así lo indique).

Una vez finalizado el proceso ESI, se lo enviará a una cola de *finalizados*.

Es importante destacar que cada proceso ESI deberá tener un identificador dentro del sistema.



Algoritmos de Planificación ^6

La ejecución de las instrucciones de cada ESI estará dividida en ráfagas de CPU y ráfagas de bloqueo. Dado que no podemos predecir con certeza la duración de cada ráfaga, el planificador será responsable de realizar las estimaciones correspondientes, cuando sea necesario. La unidad a utilizar será "sentencias" en lugar de tiempo, por cuestiones de simplicidad.

Apartir de estas estimaciones, el planificador podrá utilizar los siguientes algoritmos, los cuales deberán poder ser modificados, junto con sus parámetros, mediante el archivo de configuración al momento de inicializar el planificador:

1. *Shortest Job First:* se dará prioridad al ESI cuya próxima ráfaga sea la más corta. Deberá soportar los modos con y sin desalojo.

2. *Highest Response Ratio Next:* se dará prioridad al ESI cuyo response ratio sea el más alto.

En ambos algoritmos se desconoce la próxima ráfaga, por lo que será estimada utilizando la fórmula de la media exponencial. La formula para esta estimación será la siguiente:

$$\tau_{n+1} = \frac{\alpha}{100} t_n + (1 - \frac{\alpha}{100}) \tau_n$$

Siendo

t_n la duración de la rafaga n

$\alpha \in \mathbb{N}, 0 \leq \alpha \leq 100$

τ_n la estimación de la rafaga n

La estimación inicial de todos los ESI será la misma, y deberá poder ser modificable por archivo de configuración. Ante un empate en la estimación se podrá optar por utilizar el algoritmo de FCFS (First Come First Served)

Consola del planificador

Mediante una consola, el planificador deberá facilitar al usuario las siguientes operaciones:

- Pausar/Continuar planificación(^2): El Planificador no le dará nuevas órdenes de ejecución a ningún ESI mientras se encuentre pausado.
- bloquear *clave ID*: Se bloqueará el proceso ESI hasta ser desbloqueado *(ver más adelante)*, especificado por dicho *ID*(^3) en la cola del recurso *clave*. *Vale recordar que cada línea del script a ejecutar es atómica, y no podrá ser interrumpida; si no que se bloqueará en la próxima oportunidad posible. Solo se podrán bloquear de esta manera ESIs que estén en el estado de listo o ejecutando.*
- desbloquear *clave*: Se desbloqueara el primer proceso ESI bloquedo por la *clave* especificada.
- listar *recurso*: Lista los procesos encolados esperando al recurso.
- kill *ID*: finaliza el proceso. Recordando la atomicidad mencionada en “bloquear”. Al momento de eliminar el ESI, se debloquearan las claves que tenga tomadas.
- status *clave*: Con el objetivo de conocer el estado de una clave y de probar la correcta distribución de las mismas se deberan obtener los siguientes valores: (Este comando se utilizara para probar el sistema)
  - Valor, en caso de no poseer valor un mensaje que lo indique.
  - Instancia actual en la cual se encuentra la clave. (En caso de que la clave no exista, la Instancia actual debería)
  - Instancia en la cual se guardaría actualmente la clave (Calcular este valor mediante el algoritmo de distribución(^4), pero sin afectar la distribución actual de las claves).
  - ESIs bloqueados a la espera de dicha clave.
- deadlock: Esta consola también permitirá analizar los deadlocks que existan en el sistema y a que ESI están asociados. Pudiendo resolverlos manualmente con la sentencia de kill previamente descrita.

Configuración

Campo	Tipo	Ejemplo
Puerto de Escucha de conexiones	[numérico]	`8000`
Algoritmo de planificación	`SJF-CD` / `SJF-SD` / `HRRN`	`HRRN`
Alfa planificación	[numérico entre 0 y 100]	`30` (^5)
Estimación inicial	[numérico]	`5`
IP de Conexión al Coordinador	[cadena]	`"127.0.0.1"`
Puerto de Conexión al Coordinador	[numérico]	`8001`
Claves inicialmente bloqueadas	[Lista de claves]	`materias:K3002, materias:K3001`

^2: Esto se puede lograr ejecutando una sycall bloqueante que espere la entrada de un humano.

^3: El Planificador empezará con una serie de claves bloqueadas de esta manera.

^4: Estos algoritmos se detallarán más adelante.

^5: Para parsear este valor de forma más sencilla, recomendamos cargarlo como un entero de 0 a 100 y dividirlo por 100 antes de usarlo, con el fin de que sea un coeficiente entre 0 y 1

^6: Para mayor información sobre los algoritmos pueden recurrir a: Sistemas Operativos, Silberschatz, Galvin - Capítulo 5: Planificación de la CPU Sistemas Operativos, William Stallings 5ta Ed - Capítulo 9: Planificación uniprocesador (En otras ediciones varía el capitulo)

Coordinador

El Coordinador de Re Distinto es el proceso encargado de distribuir los pedidos de información del Sistema.

Por un lado, tiene como obligaciones recibir las operaciones desde los ESIs, ejecutarlas en una de las Instancias de Re Distinto en base a cómo define el algoritmo de Distribución de su configuración y retornar un mensaje informando el resultado de ejecución. Por otro lado, es el encargado de gestionar las Instancias que están dentro del Sistema, permitiendo la conexión y desconexión de las mismas.

Para poder simular el paso del tiempo en la ejecución, cada vez que un ESI le pida ejecutar algo, la acción estará supeditada a un retardo ficticio dado por archivo de configuración.

Inicialización

El Coordinador deberá ser el primero en iniciarse, y le proveerá a las distintas instancias la configuración de tamaños de la cantidad y el tamaño de las entradas. En primera instancia, leerá su archivo de configuración e inicializará todas las estructuras administrativas que requerirá para la gestión del sistema. Una vez realizado el proceso de inicialización, quedará a la espera de solicitudes de conexión ya sea de Instancias de Re Distinto o de procesos ESI(^7). Por cada una de las conexiones que sean aceptadas, el Coordinador lanzará un Hilo encargado de atender la conexión.

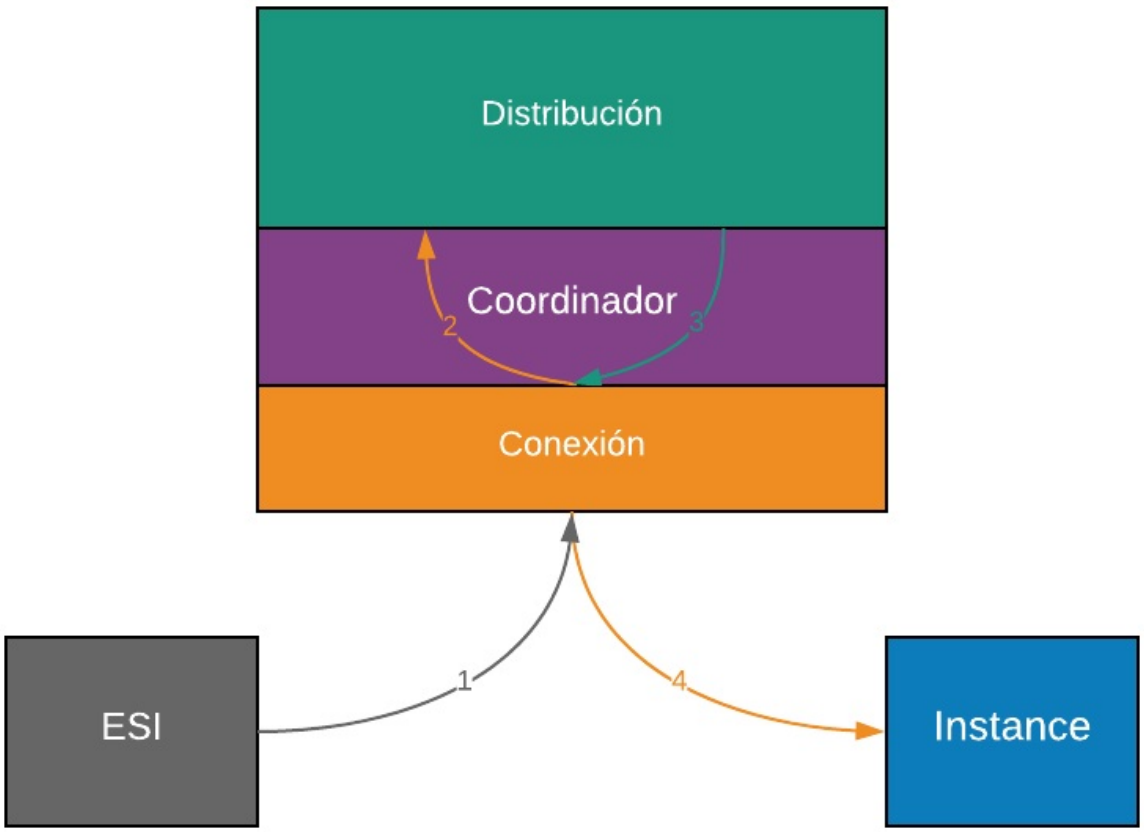
Ejecución

El Coordinador recibirá desde los procesos ESI solicitudes. Además, por cada solicitudes recibida, se almacenará información que permite registrar la ejecución que se dio en el sistema. Dicha información se guardará en un archivo conocido como log de operaciones.

Log de Operaciones

ESI	Operación
`ESI 1`	`SET materias:K3001 Fisica 2`
`ESI 1`	`STORE materias:K3001`
`ESI 2`	`SET materias:K3002 Economia`

Interacción con ESI e Instancia



1. El Coordinador recibe una solicitud proveniente de un proceso ESI.
2. El Coordinador procesa la solicitud con el fin de determinar la Instancia a la que se le asignará la solicitud(^8).
3. Se elige la Instancia asociada y se le envía la solicitud.
4. La instancia retorna al Coordinador.
5. El Coordinador logea la respuesta y envía al ESI.

En el caso que el coordinador decida en el paso 2 que la operación no puede ser ejecutada porque la instancia no existe más en el sistema(^9), le avisará al Planificador, el Planificador abortará al ESI en cuestión. No se deberán tomar acciones sobre los ESIs bloqueados para dicha clave ya que la instancia puede reincorporarse en el sistema a futuro.

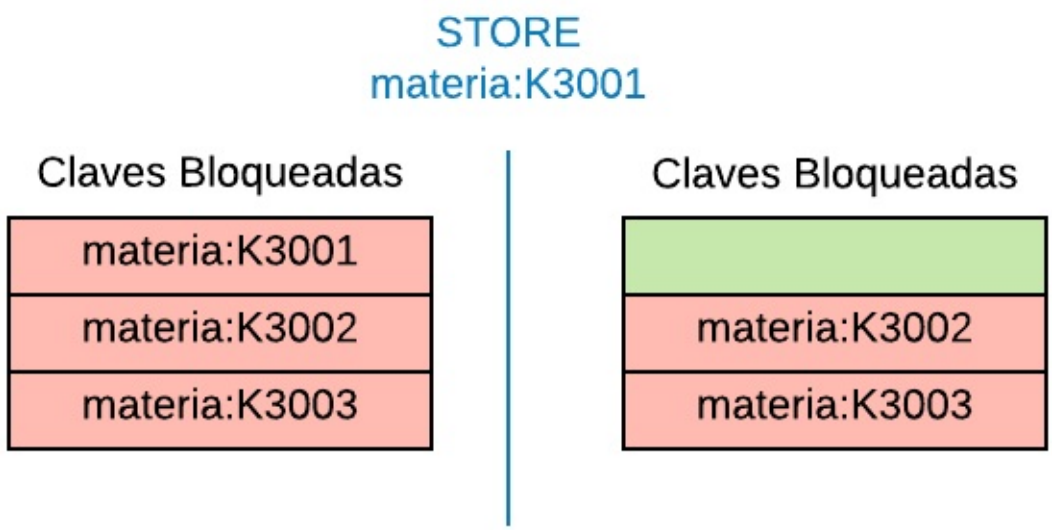
Es importante destacar que la operación GET generará una clave sin valor y ademas modifica el estado de bloqueos y desbloqueos en el planificador. Es el SET el encargado de alterar el valor.

Bloqueos

El sistema de Re Distinto aplica bloqueos sobre claves utilizadas. Es decir, a medida que un ESI solicita una operación de GET sobre una clave específica, esta pasa a estar “tomada” y no puede ser tomada (GET) por ningún ESI hasta que se libere (STORE).

Para lograr este comportamiento, el **Planificador** lleva un registro de qué claves fueron bloqueadas por cada `ESI` en particular, las cuales deberá liberar en cuanto reciba una operación `STORE` con dicha clave por parte de la `ESI` bloqueadora(^10). Esta liberación será de manera **FIFO**; el primer ESI que se encontraba bloqueado esperando esta clave será liberada *(Esto no quiere decir que será inmediatamente tomado por este ESI; sino que estará disponible para ser planificado; y deberá re ejecutar la operación de* `GET` *al ser ejecutado)*(^11).

Cabe aclarar que la finalización de un ESI libera los recursos que este tenía tomados.



Configuración

Campo	Tipo	Ejemplo
Puerto de Escucha de conexiones	[numérico]	`8001`
Algoritmo de Distribución	`LSU` / `EL` / `KE`	`EL`
Cantidad de Entradas	[numérico]	`20`
Tamaño de Entrada (en bytes)	[numérico]	`100`
Retardo (en milisegundos)	[numérico]	`300`

Ejemplos

GET de recurso disponible:

1. El planificador envía la señal de ejecutar al ESI1
2. El ESI1 envía al coordinador la orden de bloquear la clave XX (GET XX)
3. El Coordinador colabora con el Planificador avisando de este recurso
4. El Planificador lleva cuenta que la clave XX está tomada por ESI1

GET de un recurso tomado:

1. *(partiendo de la situación anterior)*
2. El planificador envía la señal de ejecutar al ESI2
3. El ESI2 envía al coordinador la orden de bloquear la clave XX (GET XX)
4. El Coordinador colabora con el Planificador avisando de este recurso
5. El Planificador responde que esta clave se encuentra tomada, y toma nota que el ESI2 se encuentra bloqueado esperándola.

---

^7: Para poder distinguir qué proceso se está conectando, intercambiará mensajes con el proceso (este procedimiento se conoce como handshaking).

^8: Eso varia segun si es una nueva entrada (GET) o si tiene que acceder a una entrada existente (SET o STORE)

^9: Esta información la sabe el coordinador ya que recibe las desconexiones de las entidades o puede consultarle a las entidades al momento de hacer la comprobación.

^10: Es importante tener en cuenta esto a la hora de implementar el comando "deadlock" de la consola del planificador

^11: Si un ESI realiza un GET y esta operación lo bloquea, esa sentencia debe ser considerada al calcular la ráfaga ejecutada. Al desbloquearse y ejecutar nuevamente la operación, volverá a ser considerada para totalizar la ráfaga.

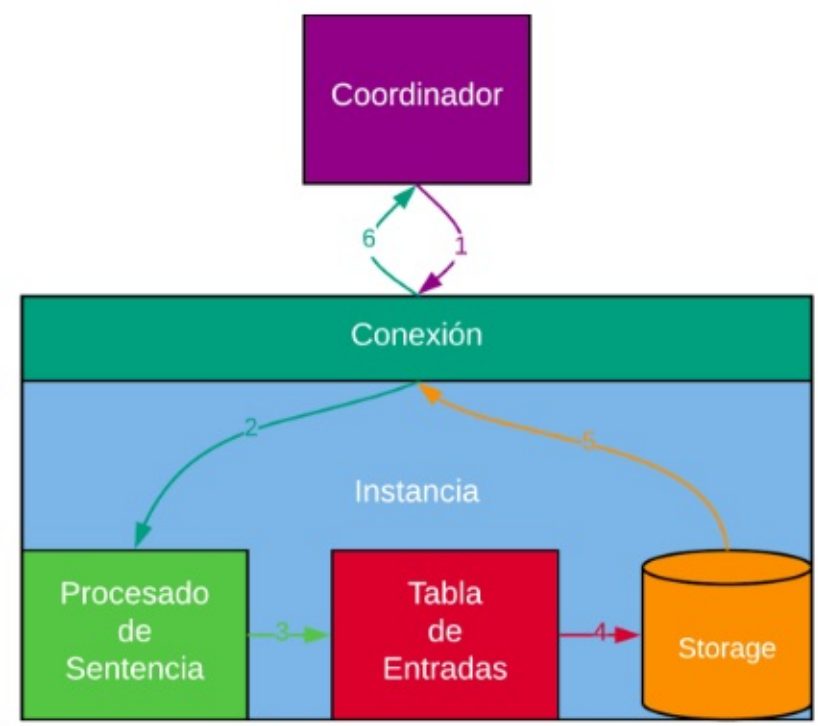
Instancia

Las Instancias de Re Distinto son los procesos encargados del almacenamiento de los datos. Cada Instancia secciona su espacio de almacenamiento en lo que se denominan Entradas y al iniciar la conexión con el coordinador, se definirá el tamaño de las mismas así como la cantidad total disponible dentro de la Instancia. En la sección de almacenamiento se entrará más en detalle respecto a las Entradas.(^12)

Tabla de Entradas

La Tabla de Entradas es una estructura administrativa de la Instancia de Re Distinto que permite gestionar el almacenamiento y facilitar el acceso a los valores almacenados. Para esto, en cada elemento de la tabla se almacenará una clave, el número de entrada asociada y el tamaño del valor almacenado.(^13)

Interacción con Coordinador



1. El Coordinador envía una sentencia de Re Distinto a la Instancia.
2. La Instancia procesa la sentencia con el fin de ejecutar la operación correctamente.
3. Al procesar la sentencia, obtiene la clave asociada que le permitirá acceder a la Tabla de Entradas.
4. De la Tabla de Entradas, se obtendrá la información necesaria para obtener el valor en el almacenamiento.
5. Se accede al valor y se prepara una respuesta para el Coordinador.
6. La Instancia envía el resultado al Coordinador.

Configuración

Campo	Tipo	Ejemplo
IP de Conexión al Coordinador	[cadena]	`"127.0.0.1"`
Puerto de Conexión al Coordinador	[numérico]	`8000`
Algoritmo de Reemplazo	CIRC / LRU / BSU	`BSU`
Punto de montaje	[path absoluto]	`"/home/utnso/instancia1/"`
Nombre de la Instancia	[cadena]	`"Instancia1"`
Intervalo de dump (en segundos)	[numérico]	`10`

^12: Para más información, investigar sobre el comando malloc().

^13: Dada la naturaleza de almacenamiento contiguo de los valores, en caso de que el tamaño del valor exceda el tamaño de la entrada, éste continuará en la siguiente entrada.

Distribución

Para generar alta disponibilidad del sistema, el Coordinador de Re Distinto se encargará de distribuir la ejecución de las operaciones a lo largo de las Instancias de Re Distinto. Para esto, dispondrá de 3 algoritmos de distribución.

Algoritmos de Distribución

Equitative Load(^14)

El algoritmo Equitative Load es el más básico de todos pero, a su vez, es el menos ambiguo de todos. Consiste en despachar una solicitud a una Instancia distinta cada vez. De esta forma, la ejecución de solicitudes se distribuye de forma equitativa entre todas las Instancias.

Least Space Used

El algoritmo LSU se basa en distribuir la ejecución de solicitudes de almacenamiento de acuerdo al espacio utilizado por cada Instancia. Para esto, el Coordinador deberá llevar cuenta del espacio en uso en cada una de las Instancias, para así poder despachar las solicitudes a aquella instancia con el mayor espacio libre *(medido en cantidad de entradas libres)*.

Key Explicit

Por último, el algoritmo Key Explicit, define la Instancia que ejecutará cada solicitud de almacenamiento en la Clave misma. Cuando este algoritmo está en uso, se tomará el primer carácter de la clave en minúscula como indicador de Instancia a la que le corresponde la ejecución de la solicitud.

Consideremos un instante donde existan 4 instancias, la primera instancia será la encargada de almacenar claves que comienzan con las letras "a" a la "g". La segunda instancia de la "h" a la "m". La tercera instancia de la letra "n" hasta la "t", y por último la cuarta instancia de la letra "u" a la "z"(^15). De agregarse nuevas instancias, las claves previamente almacenadas no se moverán de lugar; pero nuevas claves usarán el nuevo número de instancias para calcular la distribución.

Desconexión

Durante la ejecución del Sistema, existe la posibilidad de que una o más Instancias dejen de estar disponibles para el Coordinador de Re Distinto. Ante éstas situaciones, el Coordinador deberá ajustar su distribución de forma acorde a la cantidad de Instancias disponibles. No se deberá eliminar una clave de las tablas del coordinador si la instancia de desconectó; solo se elimina cuando un ESI intenta acceder a ella; de tal forma, si la instancia se reincorpora previo al uso de la clave; la desconexión sera transparente para el/los ESI que deseen operar con dicha clave.

Reincorporación

Así como las Instancias pueden dejar de estar disponibles, también pueden reincorporarse al Sistema y nuevas Instancias también pueden ser agregadas al mismo.

---

^14: Llegado el caso de que un algoritmo produzca un empate, se utilizará el algoritmo Equitative Load, el cual será el desempatador para todos los algoritmos de distribución.

^15: La letra "a" se codifica como el carácter 97 y la "z" como el 122, por lo que habrá 25 letras a ser divididas en 4 instancias, cada una con 7 letras y la última con solo 4. Cada instancia redondeará para arriba la cantidad de claves a ser usadas; y la última posiblemente tenga menos rango que aceptaría

Almacenamiento

El almacenamiento en cada Instancia está definido por una cantidad determinada de Entradas de tamaño configurable. En estas Entradas se almacenarán los datos, con la particularidad de que cada dato que se almacene será siempre en Entradas contiguas para facilitar el acceso.

Dump

Cada determinado intervalo (por archivo de configuración) la instancia almacenará las claves y sus valores en el punto de montaje. La forma de guardarlo será archivo de texto simple, cuyo nombre será el nombre de la clave, y su contenido el valor (independientemente de si ocupan 1 o más entradas). Esta información deberá ser recuperada al momento de iniciar una instancia.

Entrada

Las Entradas son espacio delimitados dentro del almacenamiento en los cuales se pueden almacenar datos. Estas Entradas tienen un tamaño fijo con el fin de facilitar la gestión del almacenamiento.

Clave

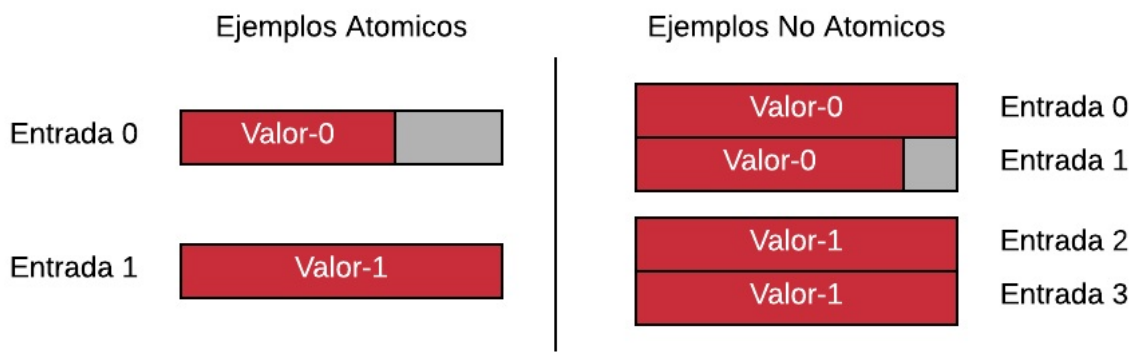
Las claves se conformarán por cadenas de hasta 40 caracteres.

Valor

El valor almacenado para cada key será de tamaño variable, dando lugar a la posibilidad de que un valor ocupe más de una sola entrada en la Instancia de Re Distinto. Otros casos posibles que surgen por la naturaleza variable del valor son aquellos casos en los que el valor tiene menor tamaño que una Entrada; en este caso, se produce fragmentación interna(^16) en esta Entrada.

Valor Atómico

Son aquellos valores que no exceden el tamaño de una entrada de almacenamiento. El proceso de reemplazo de entradas solo se aplica a aquellos valores que cumplan con este criterio.

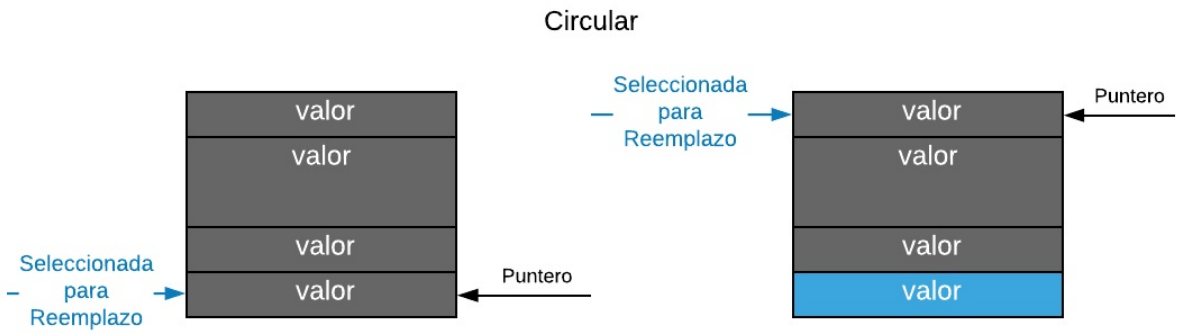


Algoritmos de Reemplazo

Dado que nuestro sistema se enfoca en aspectos de disponibilidad y no tanto en aspectos de consistencia, siempre se permitirá el almacenamiento de un nuevo. Para lograr esto cuando no existe más espacio, es necesario reemplazar entradas ya cargadas en la Instancia. El valor previo se perderá, y si un ESI intenta acceder a este, se abortará por Clave no encontrada.

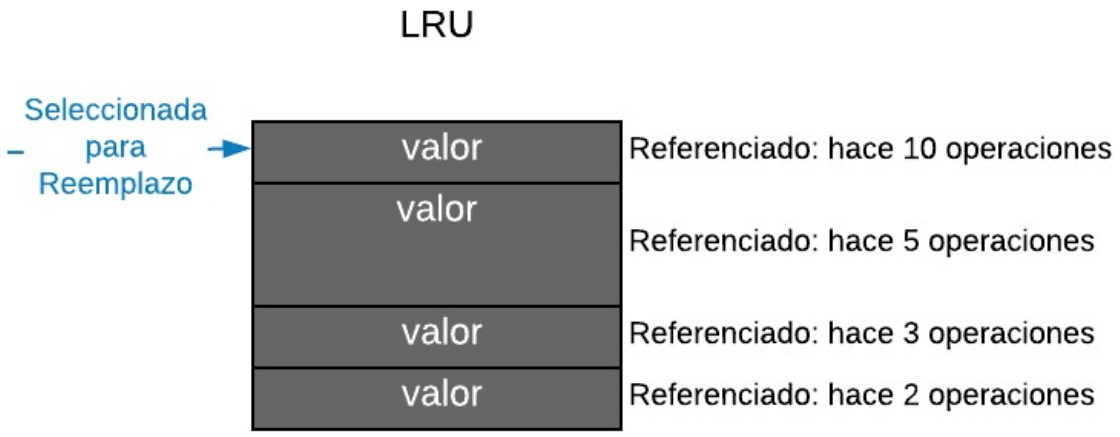
Circular

El algoritmo Circular consiste en tener un puntero sobre las entradas de almacenamiento de la Instancia, el cual establece qué entrada debe ser reemplazada. Una vez que se ha reemplazado la entrada, el puntero se mueve a la siguiente iterando hasta llegar al final. En cuanto llega a la última entrada, el puntero retorna a la primer posición. Así como en Distribución *Equitative Load* es el algoritmo desempatador, en Reemplazo el algoritmo Circular será el desempatador.



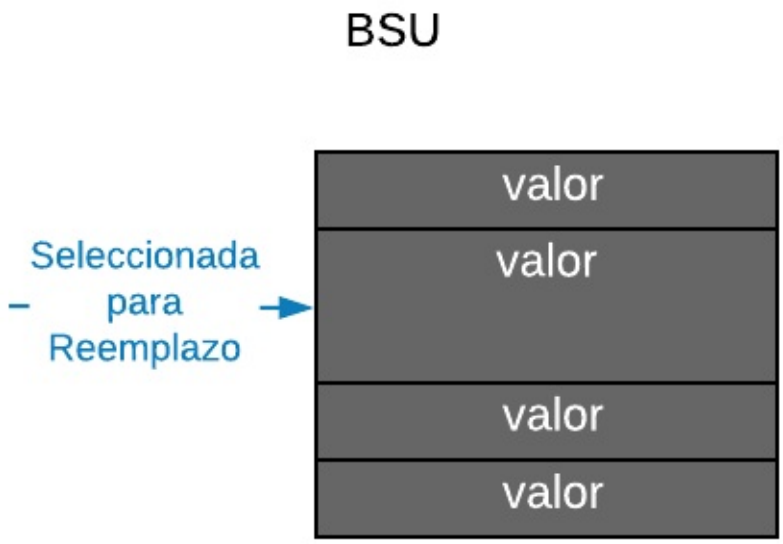
Least Recently Used

El algoritmo LRU se basa en llevar registro de hace cuánto fue referenciada cada entrada. Llegado el momento de reemplazar una entrada, se selecciona aquella entrada que ha sido referenciada hace mayor tiempo(^17).



Biggest Space Used

Por último, el algoritmo BSU lleva registro del tamaño dentro de la entrada atomica que está siendo ocupado, y en el momento de un reemplazo, escoge aquél que ocupa más espacio dentro de una entrada.



Compactación

Dada la naturaleza del almacenamiento en Re Distinto, existe la posibilidad de que se produzca fragmentación externa(^18), impidiendo almacenar más datos en casos en los que en realidad sí existe espacio para guardarlos. Ante éstas situaciones, Re Distinto activará automáticamente el proceso de Compactación ante dicho caso. Este proceso se ejecuta internamente en cada una de las Instancias de forma simultánea; es decir, que cuando llegue el momento de realizar una Compactación, todas las Instancias realizarán dicho proceso.

El proceso de Compactación consiste en el reordenamiento de los datos almacenados en las Entradas, de forma de eliminar los huecos de espacio libre que se generan debido a la fragmentación externa.

Además de realizar el reordenamiento físico de los datos, es de suma importancia actualizar la información asociada de todas las estructuras administrativas de la Instancia, de manera de que éstas sean consistentes con la actualización del almacenamiento.

Mientras está en ejecución el proceso de Compactación y Dump, no se permitirá ejecutar operaciones en el Sistema de Re Distinto y quedarán en espera a que el proceso se complete correctamente.

---

^16: Se profundizará en la cursada.

^17: Para simplificar el manejo de referencias sólo se llevará cuenta de hace cuantas operaciones fue referenciada cada entrada, almacenando el número de ultima referencia y reemplazando el menor. Esto solo sera efectivo si se accedió a la instancia (SET y STORE), caso contrario no afectara el calculo del algoritmo.

^18: Se profundizará en la cursada.

Anexo I: Lenguaje - Operaciones

- `GET clave` : Intenta bloquear el valor de la clave asociada. Si la clave no existe la crea y la bloquea.
- `SET clave valor` : Almacena el valor con la clave asociada. La clave debera estar previamente tomada por el ESI que intenta hacer el SET.
- `STORE clave` : Persiste el valor de la clave asociada en un archivo dentro de la Instancia. Por cada clave persistida se generará un archivo particular conteniendo el valor en texto plano. Esta operación también debe liberar el bloqueo sobre la clave.

Blocking

En el sistema de Re Distinto, el bloqueo se realiza sobre las claves que son solicitadas a lo largo de la ejecución de un script. Específicamente, los bloqueos se aplicarán a medida que se trabaja con una clave(^19) y solo se liberaran dichas claves cuando se realice una operación `STORE` o cuando termine la ejecución del script que bloqueo la clave.

Script 1	Script 2
`GET materias:K3001`	`GET materias:K2001`
`GET materias:K2001`	`GET materias:K3001`

En este caso, el script 1 bloquea la clave materias:K3001 en su primer operación, impidiendo que el script 2 pueda utilizar dicha clave hasta que el script 1 la libere. De la misma manera, el script 2 hace lo mismo con la clave materias:K2001 en su primera operación. Cuando estos scripts llegan a la segunda operación, se da una particularidad y es que ambos están a la espera de que el otro script libere la clave que están usando. Este fenómeno se conoce como deadlock(^20).

Ejemplo de Script

```
GET deportes:futbol:messi
SET deportes:futbol:messi Lionel Messi
GET deportes:basquet:ginobili
SET deportes:basquet:ginobili Emanuel Ginobili
# Persiste el dato de la instancia y libera esta clave
STORE    deportes:basquet:ginobili
```

Manejo de Errores

Error de Tamaño de Clave

De exceder el tamaño máximo de 40 caracteres para la Clave, la operación fallará y se informará al Usuario. Abortando el ESI culpable.

Error de Clave no Identificada

Cuando un Usuario ejecute una operación de STORE o SET y la clave no exista, se deberá generar un error informando de dicha inexistencia al Usuario. Abortando el ESI culpable.

Error de Comunicación

En caso de que ocurra un error en la comunicación con algún proceso, se deberá generar un error informando al Usuario de dicho problema. Si la desconexión ocurre entre el Planificador y el Coordinador; el sistema se considera en un estado inválido. La desconección de Instancias y de ESIs deberá estar contemplada.

Error de Clave Innacesible

Si la clave que intenta acceder existe en el sistema pero se encuentra en una instancia que se encuentra desconectada se debera informar al Usuario de dicho problema y se deberá abortar el ESI.

Error de Clave no Bloqueada

Si la clave que intenta acceder no se encuentra tomada por el ESI en cuestión, se debera informar al Usuario y se debera abortar el ESI.

---

^19: Solo la operación GET bloquea una clave.

^20: Se verá durante la cursada con mayor detalle.

Anexo II: Lenguaje - Secuencias de Ejecución

En este anexo veremos casos de ejecución de scripts y los resultados que deberian tener en cada caso a fin de ser mas explicitos en cuanto a la forma que se desea que se comporte el sistema.
En todos los casos se considera que el script es el único que ejecuta en el sistema.
Los scripts que se proporcionaran al momento de las pruebas no contentran errores que no se encuentren tipificados en el listado

Ejemplo 1 - Script que finaliza OK:

Este es el mismo ejemplo que se encuentra en el Anexo I, es por esto que vamos a tomarlo como primer ejemplo. Dado que el orden de las sentencias siempre es el correcto, el script finaliza sin inconvenientes.

```
# Se crea la clave deportes:futbol:messi en la tabla de claves del coordinador
GET deportes:futbol:messi

# Se crea finalmente la clave deportes:futbol:messi en la instancia y se graba el valor Lionel Messi
SET deportes:futbol:messi Lionel Messi

# Se crea la clave deportes:basquet:ginobili en la tabla de claves del coordinador
GET deportes:basquet:ginobili

# Se crea finalmente la clave deportes:basquet:ginobili en la instancia y se graba el valor Emanuel Ginobili
SET deportes:basquet:ginobili Emanuel Ginobili

# Persiste el dato de la instancia y libera esta clave, no da error ya que la misma existe.
STORE    deportes:basquet:ginobili
```

La clave `deportes:futbol:messi` que se encontraba tomada previamente se libera al finalizar el ESI.

Ejemplo 2 - Script que Aborta por error - Error de Clave no Identificada:

En este ejemplo se intenta hacer STORE de una variable no creada previamente con lo cual aborta el ESI y devuelve el error de `Error de Clave no Identificada`

```
GET deportes:futbol:messi
SET deportes:futbol:messi Lionel Messi
# Intento hacer store de una variable que no fue creada y por lo tanto aborta el ESI.
STORE   deportes:basquet:ginobili
# Estas instrucciones no se llegan a ejecutar.
GET deportes:basquet:ginobili
SET deportes:basquet:ginobili Emanuel Ginobili
```

Al igual que en el Ejemplo 1, al finalizar el ESI la clave `deportes:futbol:messi` se libera.

Ejemplo 3 - Script que Aborta por error - Error de Clave no Bloqueada:

En este ejemplo a pesar de existir la clave en el sistema se intenta hacer SET de la variable `deportes:futbol:messi` que por el STORE previo no se encuentra bloqueada con lo cual aborta el ESI y devuelve el error de `Error de Clave no Bloqueada`

```
GET deportes:futbol:messi
SET deportes:futbol:messi Lionel Messi
STORE   deportes:futbol:messi
# Intento hacer set de una variable que no se encuentra bloqueada y por lo tanto aborta el ESI.
SET deportes:futbol:messi El Mejor del Mundo
# Estas instrucciones no se llegan a ejecutar.
GET deportes:futbol:messi
STORE deportes:futbol:messi
```

En este ejemplo queda de manifiesto que el orden de ejecución correcto deberia ser GET - SET (1 o varios) - STORE, y que cualquier otra variación genera un error ya sea de clave no bloqueada o de clave no identificada.

Ejemplo 4 - Script que Aborta por desconexión de la instancia:

En este caso habra una desconexion en el medio de la ejecución y al no haber reconexion antes de que se intente ejecutar la sentencia el ESI aborta. Afin de simplificar la comprension del ejemplo tendremos una instancia que contendra inicialmente las claves de deportes:futbol:messi y otra que contendra inicialmente las claves de deportes:basquet:ginobili

```
GET deportes:futbol:messi
SET deportes:futbol:messi Lionel Messi
GET deportes:basquet:ginobili

# En este momento se desconecta la instancia que tiene la clave deportes:futbol:messi
# Esta sentencia ejecuta correctamente porque la clave deportes:basquet:ginobili se encuentra en una instancia que no se desco
necta
SET deportes:basquet:ginobili Emanuel Ginobili

# Esta sentencia se ejecuta correctamente porque el algoritmo de distribución envia la nueva clave a otra instancia
GET deportes:futbol:cristiano
SET deportes:futbol:cristiano CR7

# Al momento de ejecutar la siguiente instrucción el coordinador encuentra que la clave existia en el sistema pero se encuentr
a en una instancia que esta desconectada, por lo tanto el ESI aborta.
STORE deportes:futbol:messi
```

Descripción de las entregas

Los checkpoints se dividen entre "Presenciales" y "No presenciales". Este trabajo práctico contará con un único checkpoint presencial, a realizarse en el laboratorio de Medrano. Los checkpoints no presenciales son orientativos, y ayudan al grupo a medir el avance. De forma opcional, pueden ser validados con un ayudante, durante los sábados de soporte.

Checkpoint 1 - No presencial

Fecha: 28 de Abril

Tiempo estimado: 2 semanas

Distribución recomendada: 5 integrantes conectando los procesos)

Objetivos:

- Familiarizarse con Linux y su consola, el entorno de desarrollo y el repositorio
- Aplicar las Commons Libraries, principalmente las funciones para listas, archivos de conf y logs
- Definir el Protocolo de Comunicación
- Familiarizarse con el desarrollo de procesos servidor multihilo (proceso Coordinador, Planificador)

Implementación mínima:

- Creación de todos los procesos que intervienen
- Desarrollar una comunicación simple entre los procesos que permita propagar un mensaje por cada conexión necesaria
- Implementar la consola del Planificador sin funcionalidades

Lectura recomendada:

- http://faq.utnso.com/arrancar
- Beej Guide to Network Programming - link
- Linux POSIX Threads - link
- SO UTN FRBA Commons Libraries - link
- Sistemas Operativos, Silberschatz, Galvin - Capítulo 3: Procesos
- Sistemas Operativos, Silberschatz, Galvin - Capítulo 4: Hilos

---

Checkpoint 2 - No presencial

Fecha: 19 de Mayo

Tiempo estimado: 3 semanas

Distribución recomendada: 1 ESI - 2 Planificador - 1 Coordinador - 1 Instancia

Objetivos:

- Implementación de la base del Protocolo de Comunicación
- Comprender y aplicar mmap()
- Entender el concepto de Shared Library

- Comprender con algo de profundidad cómo funcionan algunos algoritmos sencillos

Implementación mínima:

- Lectura de scripts y utilización del Parser del proceso ESI
- El Planificador debe ser capaz de elegir a un ESI utilizando un algoritmo sencillo (FIFO por ej)
- El Coordinador debe ser capaz de distribuir por Equitative Load.
- Desarrollo de lectura y escritura de Entradas en el Instancia (Operaciones GET/SET).

Lectura recomendada:

- Sistemas Operativos, Silberschatz, Galvin - Capítulo 4: Hilos
- Sistemas Operativos, Silberschatz, Galvin - Capítulo 5: Planificación

---

Checkpoint 3 - Presencial - Laboratorio

Fecha: 9 de Junio

Tiempo estimado: 3 semanas

Distribución recomendada: 1 ESI - 2 Planificador - 1 Coordinador - 1 Instancia

Objetivos:

- Entender las implicancias de un algoritmo de planificación real
- Entender el concepto de productor-consumidor y sus problemas de concurrencia
- Implementar algoritmos similares a los usados en Memoria Virtual

Implementación mínima:

- ESI completo
- Planificador utilizando SJF con y sin desalojo, con todas sus colas.
- La consola del Planificador deberá poder ejecutar los comandos "Pausar/Continuar", "Bloquear", "Desbloquear" y "Listar"
- El Coordinador deberá tener el "Log de Operaciones" funcionando. También deberá ser capaz de comunicar bloqueos.
- La Instancia deberá implementar todas las instrucciones. Ala hora de reemplazar claves, deberá implementar el algoritmo Circular

Lectura recomendada:

- Sistemas Operativos, Silberschatz, Galvin - Capítulo 5: Planificación
- Sistemas Operativos, Silberschatz, Galvin - Capítulo 6: Sincronización
- Sistemas Operativos, Silberschatz, Galvin - Capítulo 8 y 9: Memoria

---

Checkpoint 4 - No Presencial

Fecha: 30 de Junio

Tiempo estimado: 3 semanas

Distribución recomendada: 2 Planificador - 2 Coordinador - 1 Instancia

Objetivos:

- Entender las implicancias de un algoritmo de planificación real
- Utilizar el concepto de "interfase" para que cada proceso planifique de forma similar, soportando diferentes algoritmos.

Implementación mínima:

- Planificador utilizando HRRN
- La consola del Planificador deberá poder ejecutar los comandos "kill" y "status"
- El Coordinador deberá ser capaz de distribuir utilizando "LSU" y "KE". Implementar retardos
- La Instancia deberá ser capaz de soportar desconexiones y reincorporaciones. Además se deberá implementar el algoritmo LRU

Lectura recomendada:

- Sistemas Operativos, Silberschatz, Galvin - Capítulo 5: Planificación
- Sistemas Operativos, Silberschatz, Galvin - Capítulo 6: Sincronización
- Sistemas Operativos, Silberschatz, Galvin - Capítulo 8 y 9: Memoria

---

Entrega final - Presencial - Laboratorio

Fecha: 14 de Julio

Tiempo estimado: 2 semanas

Distribución recomendada: 1 Planificador - 1 Coordinador - 1 Instancia - 2 Testing general y arreglos

Objetivos:

- Implementar un algoritmo de detección de deadlocks
- Probar el TP en un entorno distribuido
- Realizar pruebas intensivas
- Finalizar el desarrollo de todos los procesos

Implementación mínima:

- La consola del Planificador deberá poder ejecutar el comando "deadlock"
- La Instancia deberá ser capaz de soportar dumps y compactación. Se deberá implementar el algoritmo BSU

Lectura recomendada:

- Sistemas Operativos, Silberschatz, Galvin - Capítulo 6: Deadlock
- Sistemas Operativos, Silberschatz, Galvin - Capítulo 10 y 11: File System

---

Segunda fecha de entrega: 28 de Julio

Última fecha de entrega: 4 de Agosto