

Ingeniería en Sistemas de Información

# Cache 13

## Pruebas - Evaluación



Cátedra de Sistemas Operativos

Trabajo práctico Cuatrimestral

- 2C2015 -  
Versión [1.0.1]

# Requisitos y notas de la evaluación

## Deploy y Setup

- Es condición necesaria para la evaluación que **el Deploy & Setup del trabajo se realice en menos de 10 minutos**. Pasado este tiempo el grupo perderá el derecho a la evaluación.
- Los archivos de configuración requeridos para los diversos escenarios de pruebas deberán ser preparados por el grupo con anticipación dejando sólo los parámetros desconocidos (ej: IP) incompletos.
- En la fecha de entrega la conexión a Internet podría estar congestionada para clonar el repositorio desde GitHub. Debido a eso **el grupo debe traer una copia del trabajo en un medio extraíble**, subirlo a una máquina virtual y luego copiar dicho repositorio por red entre las VMs. Ver [Anexo - Comandos Útiles](#)

## Compilación y ejecución

- La compilación debe hacerse en la máquina virtual de la cátedra en su edición Server (no se pueden usar binarios subidos al repositorio).
- Para facilitar la visualización de varias terminales de manera simultánea se utilizará la herramienta **PuTTY** para acceder a las consolas de las Máquinas Virtuales.
- Es responsabilidad del grupo verificar que los parámetros de compilación sean portables y conocer y manejar las herramientas de compilación desde la línea de comandos. Ver [Anexo - Comandos Útiles](#)
- Debido a la complejidad y la concurrencia de los eventos que se van a evaluar es imprescindible que el alumno verifique que **su registro (log) permita determinar en todo momento el estado actual y anterior del sistema** y sus cambios significativos.

## Evaluación Final

- Cada grupo deberá llevar **dos** copias impresas de la [planilla de evaluación](#)<sup>1</sup> con los datos de los integrantes completos (dejando los campos “Nota” y “Coloquio” en blanco) y una copia de los presentes tests.
- Las pruebas pueden ser alteradas o modificadas entre instancias de entrega y recuperatorios, y podrán ser adaptadas durante el transcurso de la corrección a criterio del ayudante para lograr validar el correcto funcionamiento y desempeño del sistema desarrollado.
- En los casos en que las modificaciones se vuelvan permanentes, el documento será actualizado y re-publicado para reflejar estos cambios.

---

<sup>1</sup> Al final de este documento

# Pruebas - Evaluación Final - Laboratorio

Las siguientes pruebas son orientativas. Cada evaluador realizará las modificaciones que considere necesarias, para probar los Trabajos Prácticos.

```
git clone http://faq.utn.so/cache13-scripts
```

## Configuración inicial default

### Planificador

```
PUERTO_ESCUCHA: 4000
ALGORITMO_PLANIFICACION: FIFO
QUANTUM: N/A
```

### Administrador de Memoria

```
PUERTO_ESCUCHA: 5000
IP_SWAP: 127.0.0.1
PUERTO_SWAP: 6000
MAXIMO_MARCOS_POR_PROCESO: 10
CANTIDAD_MARCOS: 10
TAMANIO_MARCO : 10
ENTRADAS_TLB: 10
TLB_HABILITADA: SI
ALGORITMO_TLB: FIFO
RETARDO_MEMORIA: 1
ALGORITMO_REEMPLAZO: FIFO
```

### CPU

```
IP_PLANIFICADOR: 127.0.0.1
PUERTO_PLANIFICADOR: 4000
IP_MEMORIA: 127.0.0.1
PUERTO_MEMORIA: 5000
CANTIDAD_HILOS: 1
RETARDO: 5
```

### Administrador de Swap

```
PUERTO_ESCUCHA: 6000
NOMBRE_SWAP: swap.data
CANTIDAD_PAGINAS: 20
TAMANIO_PAGINA: 10
RETARDO_SWAP: 1
RETARDO_COMPACTACION: 60
```

## mCods a correr

<p>corto.cod</p> <pre>iniciar 3; escribir 0 "Corto"; escribir 1 "Corto"; escribir 2 "Corto"; entrada-salida 30; leer 0; leer 1; leer 2; finalizar;</pre>	<p>nosoy.cod</p> <pre>iniciar 5; escribir 0 "NoSoy"; escribir 1 "NoSoy"; escribir 2 "NoSoy"; escribir 3 "NoSoy"; escribir 4 "NoSoy"; entrada-salida 60; finalizar;</pre>	<p>localidad.cod</p> <pre>iniciar 5; escribir 0 "0"; escribir 1 "1"; escribir 2 "2"; escribir 3 "3"; escribir 4 "4"; escribir 0 "00"; escribir 1 "11"; escribir 2 "22"; escribir 0 "000"; escribir 1 "111"; escribir 2 "222"; escribir 0 "0000"; escribir 1 "1111"; escribir 2 "2222"; entrada-salida 60; finalizar;</pre>
<p>io.cod</p> <pre>iniciar 10; escribir 0 "IO B"; entrada-salida 2; leer 0; entrada-salida 5; escribir 1 "IO B"; leer 1; entrada-salida 5; escribir 0 "IO B"; entrada-salida 2; leer 0; entrada-salida 5; escribir 1 "IO B"; leer 1; entrada-salida 5; finalizar;</pre>	<p>mem.cod</p> <pre>iniciar 6; escribir 2 "dos"; leer 2; escribir 3 "tres"; leer 2; escribir 1 "uno" leer 1; escribir 5 "cinco"; leer 2; escribir 4 "cuatro"; leer 5; leer 3; escribir 2 "dos"; leer 5; leer 2; entrada-salida 20; finalizar;</pre>	<p>cpu.cod</p> <pre>iniciar 1; escribir 0 "0"; leer 0; escribir 0 "1"; leer 0; escribir 0 "2"; leer 0; escribir 0 "3"; leer 0; escribir 0 "4"; leer 0; escribir 0 "5"; leer 0; escribir 0 "6"; leer 0; escribir 0 "7"; leer 0; finalizar;</pre>

## Prueba 1 - Condición mínima

Configuración inicial: la default

Esta prueba comprueba el estado determinado como mínimo para que un trabajo práctico sea evaluado.

Se deberá correr `corto.cod` revisando el % uso de la CPU. El mismo debería rondar el 50%. De no ser así, explicar por qué, teniendo en cuenta que hay diferentes formas de implementar este criterio.

## Prueba 2.a) - CPU - FIFO

Configuración inicial: la default (se puede bajar el retardo de CPU a 2). Setear el máximo de marcos por proceso en 3.

Se deberá correr de forma concurrente y en orden, el código de `nosoy.cod`, `nosoy.cod`, `corto.cod` y `nosoy.cod`. Revisar el archivo de swap (por ejemplo, con el comando `tail -f`) viendo que el texto "Corto" no pise el texto "NoSoy". El orden de finalización debe ser el mismo que el de ejecución.

## Prueba 2.b) - CPU - Round Robin

Configuración inicial: la default, cambiando la CPU para que ejecute RR con  $Q=4$  (se puede bajar el retardo de CPU a 2). Setear el máximo de marcos por proceso en 3.

Se deberá correr de forma concurrente y en orden, el código de `nosoy.cod`, `nosoy.cod`, `corto.cod` y `nosoy.cod`. Revisar el archivo de swap (por ejemplo, con el comando `tail -f`) viendo que el texto "Corto" no pise el texto "NoSoy". El orden de finalización debe ser "Corto" y luego los tres procesos "NoSoy".

## Prueba 2.c) - CPU - Multiprocesamiento

Configuración inicial: la default, cambiando la CPU para que ejecute RR con  $Q=3$ . Usar 5 hilos CPU. Setear el máximo de marcos por proceso en 3.

Ejecutar los procesos en el siguiente orden: `corto.cod`, `nosoy.cod`, `corto.cod`, `nosoy.cod`, `corto.cod` y `cpu.cod`. Revisar el % de utilización de la CPU. El mismo debería ser de alrededor de 50% en 4 hilos y 100% en uno, o bien rondar el 70% en los 5. Explicar los valores en cada caso.

Al finalizar los dos procesos `corto.cod`, ejecutar `nosoy.cod` revisando la partición de swap. Debería compactarse. Analizar el tiempo de respuesta.

## Prueba 3.a) - Memoria - Marcos libres

Configuración inicial: la default

Ejecutar el proceso `localidad.cod`. Durante la ejecución revisar el archivo de swap. No debería ser escrito nunca. Realizar un `dump` de memoria y comprobar que se está escribiendo correctamente.

## Prueba 3.b) - Memoria - Thrashing

Configuración inicial: la default con CPU planificando RR  $Q=1$ , y SWAP con un retardo de 10.

Ejecutar 4 veces el proceso `localidad.cod`. Durante la ejecución, el sistema debe verse lento. Chequear el estado de los procesos con el comando `ps`, y matar a uno de ellos. El sistema debe funcionar a velocidad normal ahora. Explicar por qué.

Se recomienda realizar esta misma prueba con 4 hilos CPU. Se debe llegar a resultados similares.

## Prueba 3.c) - Memoria - TLB

Correr la prueba 3.b, pero con 3 procesos `localidad.cod` y un retardo de memoria de 5.

Tomar nota de los tiempos de ejecución. Volver a realizar la prueba, pero limpiando la TLB veces. Los tiempos deberían ser bastante mayores (entre 2 y 5 veces aproximadamente).

## Prueba 4 - Algoritmos

Configuración inicial: la default, con un máximo de 3 marcos por proceso.

Correr `mem.cod`. Revisar el resultado en los logs. Deberían realizarse 9 fallos de página y 13 accesos a swap. Los frames finales deberían tener las páginas 3, 5 y 2.

Cambiar el algoritmo por Clock Modificado. Correr `mem.cod`. Revisar el resultado en los logs. Deberían realizarse 8 fallos de página y 12 accesos a swap. Los frames finales deberían tener las páginas 4, 3 y 2.

Cambiar el algoritmo por LRU. Correr mem. cod. Revisar el resultado en los logs. Deberían realizarse 7 fallos de página y 9 accesos a swap. Los frames finales deberían tener las páginas 3, 5 y 2.

Correr la prueba nuevamente, con un tiempo de acceso a swap de 5 segundos. Limpiar la memoria algunas veces durante la ejecución, mirando que los datos se guarden correctamente en swap. Analizar los fallos de página y los accesos a swap. Los frames finales podrían variar.

## Prueba 5 - Stress

Configuración inicial: la default con la CPU usando 4 hilos y RR con  $Q=1$ , la memoria usando Clock Modificado, una partición de swap de 100 entradas y con todos los retardos muy bajos (cerca de cero).

Correr en el siguiente orden:

- cpu.cod 5 veces
- io.cod 2 veces
- cpu.cod 2 veces
- io.cod 5 veces

Ver que todo se ejecute correctamente. El tiempo aproximado debería rondar los dos minutos y medio. Tomar nota de todas las métricas, para ser charladas en el coloquio.

# Planilla de Evaluación - TP2C2015

Grupo:

Legajo	Nombre y Apellido	Nota

Evaluator:

Coloquio:

---





### Prueba 1 - Condiciones minimas

Observaciones:

### Prueba 2.a - CPU - FIFO

Observaciones:

### Prueba 2.b - CPU - Round Robin

Observaciones:

### Prueba 2.c - CPU - Multiprocesamiento

Observaciones:

### Prueba 3.a - Memoria - Marcos libres

Observaciones:

### Prueba 3.b - Memoria - Thrashing

Observaciones:

### Prueba 3.c - Memoria - TLB

Observaciones:

### Prueba 4 - Algoritmos

Observaciones:

### Prueba 5 - Stress

Observaciones:

# Anexo - Comandos Útiles

Copiar un directorio completo por red

```
scp -rpC [directorio] [ip]:[directorio]
```

Ejemplo:

```
scp -rpC tp-1c2015-repo 192.168.3.129:/home/utnso
```

Descargar **solo** la última versión del código (en vez de todo el repositorio)

```
curl -u '[usuario]' -L -o [archivo] [url_repo]
```

Ejemplo:

```
curl -u 'gastonprieto' -L -o commons.tar
```

<https://api.github.com/repos/sisoputnfrba/so-commons-library/tarball/master>

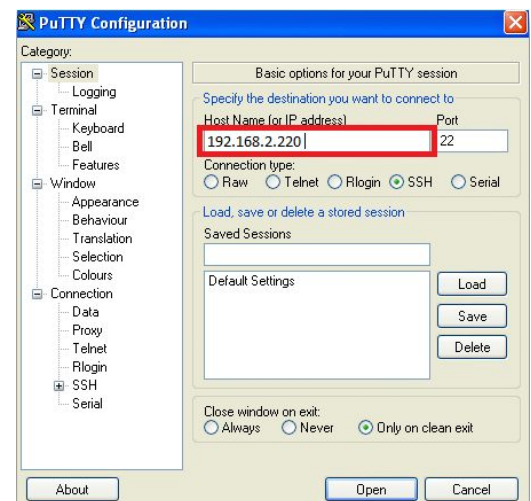
Este comando debe ejecutarse sin salto de línea. Luego **descomprimir** con: `tar -xvf commons.tar`

## PuTTY

Este famoso utilitario nos permite desde Windows acceder de manera simultánea a varias terminales de la Máquina Virtual, similar a abrir varias terminales en el entorno gráfico de Ubuntu.

Ya se encuentra en las computadoras del laboratorio y se puede descargar desde [aquí](#)

Al iniciar debemos ingresar la IP de nuestra máquina virtual en el campo **Host Name (or IP address)** y luego presionar el botón **Open** y loguearnos como **utnso**



**Se recomienda investigar:**

- Directorios y archivos: `cd`, `ls`, `mv`, `rm`, `ln` (creación de symlinks)
- Entorno: `export`, variable de entorno `LD_LIBRARY_PATH`
- Compilación: `make`, `gcc`, `makefile`