UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería en Sistemas de Información

ESTheR - El Stack The Revenge

Documento de pruebas

Cátedra de Sistemas Operativos

Trabajo práctico Cuatrimestral

- 1C2017 -
Versión [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 con anticipación por el grupo con todos los valores requeridos prefijados dejando los sólo los parámetros desconocidos (ej: IP) incompletos.

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). 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

Evaluación

Cada grupo deberá llevar **dos** copias impresas de la planilla de evaluación[1] con los datos de los integrantes completos (dejar el campo “Nota” y “Coloquio” en blanco) y una copia de los presentes tests.

Para la aprobación un Trabajo Práctico deberá contar con todos los ítems marcados como “**Contenidos Mínimos**”.

Una vez alcanzada la aprobación, los ítems marcados como **“Detalle”** tienen un valor asociado que suman puntaje en caso de ser cumplidos, teniendo como base la nota 6 (seis) y cómo máximo la nota 10 (diez).

Las pruebas **pueden ser alteradas o modificadas entre instancias de entrega** para verificar el correcto funcionamiento y desempeño del sistema desarrollado.
En estos casos el documento será actualizado y re-publicado para reflejar estos cambios.

[1] Al final de este documento

Planilla de Evaluación

Nombre del Grupo	Nota (Grupal)

Legajo	Apellido y Nombres	Nota (Coloquio)

Evaluador/es Práctica	
Evaluador/es Coloquio	

Observaciones:

Planilla de Evaluación

Nombre del Grupo	Nota (Grupal)

Legajo	Apellido y Nombres	Nota (Coloquio)

Evaluador/es Práctica	
Evaluador/es Coloquio	

Observaciones:

Sistema Completo	
Contenidos Mínimos	
Los procesos ejecutan de forma simultánea y la cantidad de hilos en el sistema es la adecuada	
Los procesos establecen conexiones TCP/IP y se comunican mediante un protocolo	
El sistema no registra casos de Espera Activa ni Memory Leaks	
El sistema responde de forma resiliente a la interacción con el entorno	
Se utilizaron de forma criteriosa los métodos estudiados para el manejo de múltiples conexiones *(multiplexado y arquitecturas multi-hilos)*	
El log permite determinar en todo momento el estado actual y anterior de los diversos procesos y del sistema junto con sus cambios significativos	

Proceso Consola	
Contenidos Mínimos	
La consola permite la ejecución de varios hilos programa simultáneamente	
Al finalizar un programa se muestran correctamente las estadísticas de su ejecución	
Detalle	
La consola permite ejecutar varias veces el comando Iniciar programa. (0.25)	
El comando Limpiar Mensajes deja la pantalla en limpio de los mensajes impresos por los diferentes hilos programa. (0.25)	

Proceso Kernel	
Contenidos Mínimos	
Se respeta el algoritmo de planificación	
El proceso informa de forma clara y concisa el estado de las colas de planificación	
El kernel permite que se le conecten de manera independiente Consolas y CPUs	
La desconexión de una Consola solo afecta a los procesos ejecutados desde la misma.	
El sistema libera recursos de forma correcta frente a la desconexión de una Consola	
El sistema contempla las instrucciones del sistema y las ejecuta de forma atómica	
Las estructuras se encuentran correctamente sincronizadas	

El sistema respeta de manera clara el grado de multiprogramación	
La consola del kernel permite operarse de manera clara	
El kernel informa correctamente los Exit Code	
La capa de memoria administra de manera correcta el Heap para los diferentes procesos	
La capa de File System permite operar de manera consistente con los archivos	
La desconexión de una CPU solo afecta al programa proceso que esté ejecutando en esa CPU (si es que hay un proceso en ejecución)	
La finalización de un proceso reduce el grado de multiprogramación.	
Detalle	
El grado de multiprogramación puede modificarse desde la consola y se respeta el cambio. (0.5)	
Se pueden finalizar los procesos desde la consola del kernel y se informa correctamente a las consolas. (0.5)	
Se puede frenar la planificación para que no se muevan los procesos entre colas. (0.5)	
Se respeta el tamaño máximo de Alloc. (0.5)	
Las entradas en las tablas de archivos son correctamente liberadas al terminar un proceso (0.5)	
Si un es archivo abierto como lectura devuelve error si no existe (0.5)	
Si un es archivo abierto como creación NO devuelve error si no existe y se crea. (0.5)	
Un archivo no puede ser eliminado si un programa lo tiene abierto (0.25)	
La escritura sobre el FD 1 se interpreta como impresión por pantalla (0.5)	

Proceso Memoria	
Contenidos Mínimos	
La interfaz de la memoria se respeta.	
La memoria reserva parte de su espacio para sus estructuras administrativas.	
La memoria caché se utiliza y se respetan los tiempos de acceso.	
Las páginas se asignan mediante la función de hash.	
El flush de la caché es correcto y se ve el impacto en la performance.	
La estructura de la memoria se encuentra correctamente sincronizada.	
Las páginas de la caché se reemplazan correctamente según el algoritmo definido.	

Detalle	
Se respetan las estructuras administrativas y su tamaño. (0.5)	
Las primeras páginas de la memoria se encuentran ocupadas. (0.5)	
El retardo puede configurarse de forma correcta y se respetan los cambios. (0.25)	
Los tamaños se informan de manera correcta tanto para la memoria como para los procesos. (0.25)	

Proceso CPU	
Contenidos Mínimos	
Se respetan los Quantums asignados en el algoritmo RR	
Se reportan correctamente los errores en caso de operaciones a memoria incorrecta	
Se respetan las llamadas al sistema, realizandolas de manera sincrónica con el Kernel	
Se respetan los tiempos de sleep entre operaciones	
Las llamadas al sistema wait y signal se envían al kernel para su resolución	
Detalle	
Al recibir una señal de desconexión, la CPU completa la rafaga en ejecución, devuelve el PCB y s desconecta. (0.5)	
La CPU notifica de manera correcta la finalización de un programa. (0.25)	

Proceso FileSystem	
Contenidos Mínimos	
La implementación respeta las estructuras de SADICA	
El sistema responde de forma resiliente al uso del mismo	
El sistema permite ser consultado por la existencia de un directorio	
El sistema permite leer y escribir archivos en posiciones y tamaños aleatorios	
El acceso a los datos en el FileSystem se encuentra correctamente sincronizado	
- El bitmap refleja la liberación de bloques - La estructura de directorios refleja el nuevo nodo disponible	
El sistema falla correctamente al intentar guardar nuevos datos sin que haya bloques disponibles	
El sistema permite guardar un archivo que ocupe todos los bloques de datos en el disco.	

El FS no permite la creación de nuevos archivos si no hay bloques libres.	
Al eliminar un archivo los bloques ocupados son recuperados y se pueden utilizar.	

Prueba Base - Condición Mínima para Presentarse

Mediante esta prueba se evaluará el funcionamiento básico del trabajo práctico. En caso de no cumplir con esta prueba mínima, no se continuará con la evaluación.

Configuración del sistema:

VM1 Kernel Consola-2	VM2 CPU1 Consola-1
VM3 FS	VM4 Memoria CPU2

Configuración del Kernel

```
ALGORITMO=FIFO
GRADO MULTIPROG=5
SEM_IDS=[mutexArch,b]
SEM_INIT=[0,5]
STACK_SIZE=2
SHARED_VARS=[!pasadas]
```

Configuración de la Memoria

```
MARCOS=500
MARCO_SIZE=256
CACHE_X_PROC=0
RETARDO_MEMORIA=100
```

Configuración del FS

```
PUNTO_MONTAJE_= /home/utnso/FS_SADICA
```

Desarrollo

Iniciar el FS; el Kernel; las Consolas 1 y 2 ; las CPUs 1 y 2 y validar las conexiones.
Ejecutar `stackoverflow.ansisop` en Consola-2 y validar que se creen las estructuras correspondientes en la Memoria, validar que finalice el programa por stack overflow.
Ejecutar en la Consola-1 el programa `leerFS.ansisop` y `escribirFS.ansisop`. Validar que se escriba y lea el archivo /archivo.bin una sola vez. Validar la escritura de las variables compartidas.

Prueba Completa

Mediante esta prueba se evaluará por completo el funcionamiento del trabajo práctico repasando las funcionalidades detalladas en el enunciado

Configuración del sistema:

VM1 Kernel Consola-2	VM2 CPU1 Consola-1
VM3 FS CPU3	VM4 Memoria CPU2

Configuración del Kernel

ALGORITMO=RR
QUANTUM=4
QUANTUM_SLEEP=500
GRADO MULTIPROG=3
SEM_IDS=[m,b, c]
SEM_INIT=[1,1,0]
STACK_SIZE=2
SHARED_VARS=[!pasadas, !colas]

Configuración de la Memoria

MARCOS=500
MARCO_SIZE=256
ENTRADAS_CACHE=18
CACHE_X_PROC=3
REEMPLAZO_CACHE=LRU
RETARDO_MEMORIA=100

Configuración del FS

PUNTO_MONTAJE_= /home/utnso/FS_SADICA

Desarrollo

Iniciar el FS; el Kernel; las Consolas 1 y 2 ; las CPUs 1, 2 y 3. y validar las conexiones.
Ejecutar en la Consola 1 un proceso `productor.ansisop` e iniciar un proceso `consumidor.ansisop` en la Consola 2.
Finalizar el proceso `productor.ansisop` que está ejecutando en la Consola 1 y verificar que el proceso `consumidor.ansisop` detiene su ejecución quedando en la cola de bloqueados.
Iniciar un proceso `consumidor.ansisop` en la Consola 1 y verificar que se bloquea.
Iniciar un proceso `productor.ansisop` en la Consola 2 y verificar que se alternan los consumidores de las consolas 1 y 2 en mostrar los resultados.
Ejecutar el comando “Limpiar Mensajes” en una Consola y verificar que la misma se blanquee.
Desconectar la Consola 2 y verificar se finalicen ambos procesos de dicha consola y que el proceso `consumidor.ansisop` de la Consola 1 se bloquee.
Desconectar el proceso `consumidor.ansisop` de la Consola 1 para limpiar los procesos.
Ejecutar en la Consola 1 un proceso `heapbasico.ansisop` y verificar que el proceso comenzará a hacer pedidos y en el último debe aparecer en otra página.

Prueba File System

Mediante esta prueba se evaluará el funcionamiento en detalle del módulo de File System.

Configuración del sistema:

VM1 Kernel	VM2 CPU1 Consola-1
VM3 FS	VM4 Memoria

Configuración del Kernel

ALGORITMO=RR
QUANTUM=4
QUANTUM_SLEEP=500
GRADO MULTIPROG=3
SEM_IDS=[m,b, c]
SEM_INIT=[1,1,0]
STACK_SIZE=2
SHARED_VARS=[!pasadas, !colas]

Configuración de la Memoria

MARCOS=100
MARCO_SIZE=256
ENTRADAS_CACHE=18
CACHE_X_PROC=3
REEMPLAZO_CACHE=LRU
RETARDO_MEMORIA=500

Configuración del FS

PUNTO_MONTAJE_= /home/utnso/FS_SADICA

Desarrollo

Iniciar el FS; el Kernel, la Consola 1, la CPUs 1 y validar las conexiones.
Ejecutar el script `creaArchivo.ansisop` y verificar en el FS que se crea un archivo y se escribe en el mismo.
Ejecutar el script `leeBorra.ansisop` y verificar lo que se imprime por pantalla del contenido del archivo sea correcto y luego se elimine el archivo del FS.
Ejecutar el script `creaGigante.ansisop` y controlar que se crea un archivo que ocupa todos los bloques del FS, para ello ejecutar un `creaArchivo.ansisop` y verificar que el script falla por no poder crear el archivo por falta de espacio.

Prueba Heap y Memoria

Mediante esta prueba se evaluará el funcionamiento en detalle la administración del Heap y la Memoria dentro del trabajo práctico.

Configuración del sistema:

VM1 Kernel Consola-2	VM2 CPU1 Consola-1
VM3 FS	VM4 Memoria CPU2

Configuración del Kernel

ALGORITMO=RR
QUANTUM=4
QUANTUM_SLEEP=500
GRADO MULTIPROG=5
SEM_IDS=[m,b, c]
SEM_INIT=[1,1,0]
STACK_SIZE=2
SHARED_VARS=[!pasadas, !colas]

Configuración de la Memoria

MARCOS=100
MARCO_SIZE=256
ENTRADAS_CACHE=12
CACHE_X_PROC=3
REEMPLAZO_CACHE=LRU
RETARDO_MEMORIA=500

Configuración del FS

PUNTO_MONTAJE_= /home/utnso/FS_SADICA

Desarrollo

Iniciar el FS; el Kernel; las Consolas 1 y 2 ; las CPUs 1 y 2. y validar las conexiones.
Ejecutar en la Consola 1 un proceso `heappasado.ansisop` y verificar que el proceso es desconectado por solicitar excedente del máximo.
Ejecutar en la Consola 1 un proceso `heapcompactar.ansisop` y verificar que el heap se haya compactado y se alojo a la variable “f” en la posición de “b” y “c”.
Verificar que se encuentren todos los marcos libres de la memoria.
Ejecutar en la Consola 1 y 2 un proceso `llenandomemoria.ansisop` y verificar que la memoria vaya asignando marcos a dichos procesos y se expulse primero uno de ellos por falta de memoria y luego el otro ocupe el total de la misma y sea expulsado.
Verificar que cuando se ejecuta `llenandomemoria.ansisop`, el primer llamado a memoria para solicitar una instrucción tarde más que los siguientes.
Ejecutar en la consola 1 un `consumidor.ansisop` y verificar que este ocupa frames de memoria y de cache, una vez verificado esto esperar a que el programa consumidor se bloquee y lanzar en la consola 2 procesos `heapbasico.ansisop` hasta alcanzar el máximo nivel de multiprogramación (si el mismo no se modificó desde el inicio de esta prueba

deberían ser 4), controlar los reemplazos de la memoria cache y verificar el momento en el que el proceso `consumidor.ansisop` es desalojado por falta de espacio.

Prueba Estrés

Mediante esta prueba se evaluará el funcionamiento del trabajo práctico bajo condiciones de estrés. La misma podrá sufrir modificaciones según lo detallado en este documento.

Configuración del sistema:

Configuración del sistema:

VM1 Kernel CPU4 Consola-4	VM2 CPU1 Consola-1
VM3 FS CPU3 Consola-3	VM4 Memoria CPU2 Consola-2

Configuración del Kernel

ALGORITMO=RR
QUANTUM=4
QUANTUM_SLEEP=100
GRADO MULTIPROG=3
SEM_IDS=[m,b, c]
SEM_INIT=[1,1,0]
STACK_SIZE=2
SHARED_VARS=[!pasadas, !colas]

Configuración de la Memoria

MARCOS=500
MARCO_SIZE=256
ENTRADAS_CACHE=18
CACHE_X_PROC=3
REEMPLAZO_CACHE=LRU
RETARDO_MEMORIA=50

Configuración del FS

PUNTO_MONTAJE= /home/utnso/FS_SADICA

Desarrollo

Esta prueba puede variar su naturaleza según el devenir de las pruebas anteriores con los cual no hay un esquema fijo a seguir pero en principio se probaran los siguientes items:

- Desconexión y reconexión de CPUs en tiempo de ejecución.
- Desconexión y reconexión de Consolas en tiempo de ejecución.
- Finalización de programas por medio de señales / finalizaciones abruptas (Ctrl + C).
- Pruebas de cambios en archivos del FS SADICA.
- Resistencia del trabajo práctico al ejecutar con retardos cercanos a 0.
- Gran cantidad de programas ansisop ejecutando de manera simultánea.

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

curl -u '[usuario]' -L -o [nombre_del_archivo_a_generar] [url_repo]

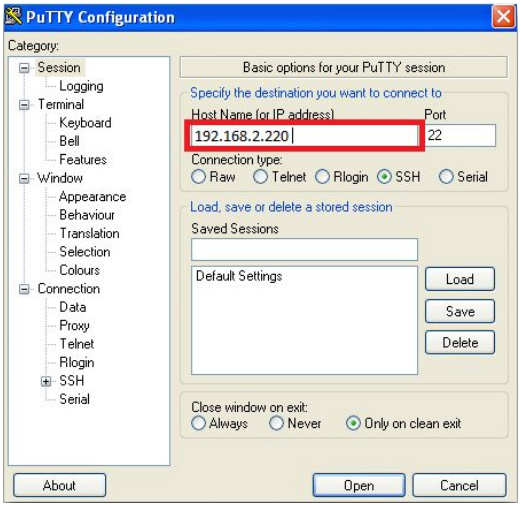
Ejemplo:

```
curl -u 'gastonprieto' -L -o commons_ultimo.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_ultimo.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