UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería en Sistemas de Información

[Ráfaga]

La herramienta más eficaz de depuración sigue siendo una cuidadosa reflexión, junto con las declaraciones de impresión juiciosamente colocadas. — Brian W. Kernighan (1979)

Documento de pruebas

Cátedra de Sistemas Operativos

Trabajo Práctico Cuatrimestral

- 2C2014 -
Versión [1.3.aa.82]

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 los sólo los parámetros desconocidos (ej: IP) incompletos.
Los scripts ESO que se piden se encuentran en el siguiente repositorio: https://github.com/sisoputnfrba/scripts-eso[1]
En la fecha de entrega la conexión a Internet podría estar congestionada para clonar el repositorio desde GitHub. Debido a eso *se recomienda traer una copia del trabajo en un medio extraíble* e investigar métodos para copiar directorios entre máquinas en red (`scp`/WinSCP).

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[2] con los datos de los integrantes completos (dejar el campo “Nota” y “Coloquio” en blanco) y una copia de los presentes tests.
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.
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. En otras palabras, este documento es de caracter orientativo: el ayudante podrá realizar o ignorar las pruebas que considere apropiadas para lograr el objetivo de la corrección - verificar 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.

[1] Propenso a sufrir cambios, revisar para tener la última versión.
[2] Al final de este documento

Pruebas

Prueba 1 - Condición mínima

Mediante este test se validará el funcionamiento mínimo del sistema, verificando además el correcto uso de memoria y CPU por parte de los procesos.

Esta prueba deberá ser aprobada para que el grupo sea considerado en condiciones de ser evaluado.

Configuración inicial

Se requieren 4 máquinas virtuales para ejecutar este test.

VM 1:
Kernel
VM 2:
CPU 2
Programa 1 - `arithmetics.bc`
Programa 2 - `bigStack.bc`
VM 3:
CPU 1
Programa 3 - `segFault.bc`
Programa 5 - `STDIN.bc`
VM 4:
MSP
Programa 4 - `STDOUT.bc`

El Kernel deberá tener los siguientes parámetros de configuración:

```
QUANTUM=2
SYSCALLS=/home/utnso/kernel/syscalls.bc
TAMANIO_STACK=400
```

Las CPUs deberán tener los siguientes parámetros de configuración:

```
RETARDO=1000
```

La MSP deberá tener los siguientes parámetros de configuración:

```
CANTIDAD_MEMORIA[3]=3
CANTIDAD_SWAP=1
SUST_PAGS[4]=ALGORITMO1
```

[3] El tamaño de memoria principal (sin espacio de intercambio) esta dado en Kilobytes, mientras que el de espacio de intercambio en Megabytes.
[4] El algoritmo varía entre cada grupo, elijan cualquiera.

Desarrollo

Iniciar el Kernel, la MSP y las CPUs 1 y 2. Validar las conexiones, uso de memoria, cantidades de hilos y CPU de los procesos.

Ejecutar el script `arithmetics.bc` en la VM 2 (Programa 1), observando el tiempo de espera entre quantums. Verificar en la MSP que se creen y modifiquen los segmentos de memoria. Esperar a que finalice el script, y validar que la MSP haya quedado limpia.

Ejecutar el script `bigStack.bc` en la VM 2 (Programa 2). Verificar que el Programa altera su stack.

Ejecutar el script `segFault.bc` en la VM 3 (Programa 3). Verificar que el sistema admite fallos programados.

Ejecutar nuevamente los script `bigStack.bc` y `segFault.bc` en la VM 2 y 3 (Programa 2 y 3) respectivamente, de forma simultánea. Verificar que los Programas respetan el quantum y que son planificados haciendo uso de las dos CPUs y que el fallo de segmento programado no altera el resto del sistema.

Ejecutar nuevamente los script `bigStack.bc` y `STDOUT.bc` en la VM 2 y 4 (Programa 2 y 4) respectivamente, de forma simultánea. Verificar que los Programas respetan el algoritmo de planificación.

Desconectar la CPU 2 y validar el estado del sistema.

Ejecutar el script `STDOUT.bc` en la VM 4 (Programa 4), y verificar que las llamadas al sistema actúen de forma planeada.

Reconectar la CPU 2 y validar que efectivamente esté conectada.

Ejecutar los script `arithmetics.bc`, STDOUT.bc y STDIN.bc en la VM 4 y 3 (Programa 4 y 5) respectivamente; primero ejecutando `STDIN.bc`, esperando a que la consola del programa reconozca la petición de entrada. Luego ejecutar `STDOUT.bc` y verificar la unicidad de atención de las llamadas al sistema. Luego arithmetics.bc y corroborar que la otra CPU ociosa opere. Por último ingresar una palabra de 5 caracteres en la consola de `STDIN.bc` y comprobar la finalización del sistema.

Prueba ProdCons	**Productor Consumidor**
Objetivo	Mediante este test se validará el funcionamiento de llamadas al sistema, semáforos y eventual desconexión de CPU. Ademas se validará el correcto funcionamiento de la MSP con su capacidad de almacenamiento, y la capacidad de cambio de la misma para obtener todos los resultados esperados. Como antes, se verificará los consumos de memoria y CPU.
Configuración	Esquema 1
Resultados esperados	1. Falta de recursos para correr en su totalidad el esquema 2. El ultimo `prueba_consumidor.bc` no ejecutará por falta de memoria. 3. Los scripts `prueba_consumidor.bc` terminaran de forma secuencial, respetando las impresiones del script `prueba_forES.bc`.

Esquema 1	

Máquinas Virtuales	Se requieren 4 máquinas virtuales para ejecutar este test. VM 1, VM 2, VM 3, VM 4.
	VM 1:
	Kernel
	`prueba_consumidor.bc`
	VM 2:
	`prueba_forES.bc`
	`prueba_productor.bc`
	CPU 2
	VM 3:
	`prueba_consumidor.bc`
	CPU 1
	VM 4:
	MSP
	`prueba_consumidor.bc`

Kernel	`QUANTUM=2`
	`SYSCALLS=/home/utnso/kernel/syscalls.bc`
	`TAMANIO_STACK=400`

CPU 1	`RETARDO=1000`
CPU 2	`RETARDO=1000`

MSP	`CANTIDAD_MEMORIA=2`
	`CANTIDAD_SWAP=0`
	`SUST_PAGS=ALGORITMO1`

Prueba MSP	Consistencia de la MSP
Objetivo	Mediante este test se validará el funcionamiento del módulo de memoria y su interacción con hilos de un mismo programa ESO y su algoritmo de swap.
Configuración	Esquema 2
Resultados esperados	Correcto estado de la memoria en cada instancia, donde se comparta el segmento de código y repliquen los segmentos de stack.

Esquema 2	

Máquinas Virtuales	Se requieren 4 máquinas virtuales para ejecutar este test. VM 1, VM 2, VM 3, VM 4.
	
	VM 1:
	Kernel
	`prueba_hilos.bc`
	VM 2:
	`prueba_suma.bc`
	CPU 2
	VM 3:
	CPU 1
	VM 4:
	MSP

Kernel	`QUANTUM=4`
	`SYSCALLS=/home/utnso/kernel/syscalls.bc`
	`TAMANIO_STACK=400`

CPU 1	`RETARDO=1000`
CPU 2	`RETARDO=1000`

MSP	`CANTIDAD_MEMORIA=1`
	`CANTIDAD_SWAP=1`
	`SUST_PAGS=ALGORITMO1`

Planilla de Evaluación - TP2C2014

Grupo:

Legajo	Nombre y Apellido	Nota

Evaluador:

Coloquio:

Condiciones Mínimas	
Existen conexiones TCP entre los procesos (netstat -nap).	
Los CPUs pueden ingresar y salir del sistema	
El Quantum y retardo es configurable y se respeta	
Las CPUs ejecutan de forma simultánea e independiente	
Al terminar un programa, la MSP queda limpia	
El uso de CPU y memoria no es excesivo (top). La sincronización no depende de usleep()	
La cantidad de hilos en el sistema es la adecuada	
La ejecución de los script son independiente	

No hay esperas activas	
El funcionamiento de la instrucción INTE es adecuada, y se respeta el ingreso de palabras de 5 caracteres en STDIN.bc	
Las llamadas al sistema se atienden de forma secuencial	
El uso del TCB KM no frena el resto del sistema	
Los errores en los scripts no impiden el funcionamiento del resto del sistema (**segfault**)	

Prueba ProdCons	
La MSP responde correctamente ante la falta de memoria	
La MSP respeta su configuración	
Los consumidores terminan de forma secuencial	
La desconeccion de una CPU afecta únicamente al script corriendo en la misma	
La disposición del TCB KM alterna entre los scripts que lo soliciten	

Prueba MSP	
La MSP puede atender, en simultáneo, pedidos del sistema y del usuario	
Los algoritmos de swap son correctos	
La tabla de segmentos es legible y correcta	
BONUS: Existe una configuración para que el resultado de hilos.bc sea: 7,5,15	

Anexo - Comandos Útiles

- Copiar un directorio[5] completo por red

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

Ejemplo:

```
scp -rpC so-commons-library 192.168.3.129:/home/utnso
```

- Descargar la última versión del código en vez de todo el repositorio

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

Ejemplo (el comando debe ejecutarse sin salto de línea):

```
curl -u 'gastonprieto' -L -o commons.tar
https://api.github.com/repos/sisoputnfrba/so-commons-library/tarball/master
```

Luego descomprimir con: `tar -xvf commons.tar`

Se recomienda investigar:

- Directorios y archivos: cd, ls, mv, rm, ln (creación de symlinks)
- Acceso a consola por red: scp (copia por red de archivos/directorios), ssh
- Entorno: export, variable de entorno LD_LIBRARY_PATH
- Compilación: make, gcc, makefile
- Herramientas: winscp (scp desde windows), PuTTY (cliente de ssh para Windows)
- Manejo de consolas virtuales (Ctrl+Alt+F1, F2, F3, F4, etc)

[5] Considerar el uso de dispositivos de almacenamiento, evitando tener que bajarlo en el momento.