UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Super Mario Proc.

Tests

Ingeniería en Sistemas de Información
Cátedra de Sistemas Operativos

- 1C2013 -

Notas previas a la evaluación

Deploy y Setup

Es condición necesaria para la evaluación que **el Deploy & Setup del trabajo se realice en 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.

La fecha de entrega la conexión a Internet podría estar congestionada para clonar el repositorio desde GitHub. *Se recomienda traer una copia del trabajo en un medio extraíble* e investigar métodos para copiar directorios entre máquinas en red (scp).

Compilación y ejecución

La compilación debe hacerse en la máquina virtual de la cátedra en su edición Server. 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 con los datos de los integrantes completos (dejar el campo “Nota” en blanco).

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 de la plataforma** y sus cambios significativos.

Las pruebas pueden ser alteradas o modificadas entre instancias de entrega y recuperatorios. En todos los casos el documento se publicará con anticipación.

Prueba 1	**Funcionamiento básico del sistema**
Objetivo	Mediante este test se validará el funcionamiento mínimo de la plataforma y su reacción ante eventos imprevistos. Se verificará también que los consumos de memoria y CPU estén dentro de rangos razonables. Esta prueba deberá ser aprobada para que el grupo sea considerado en condiciones de ser evaluado.
Configuración	Esquema 1

Prueba 2	**Vidas y Muertes**
Objetivo	Validar los procedimientos para la muerte del personaje y el correcto manejo de señales.
Configuración	Utilizar nuevamente el Esquema 1 desactivando el algoritmo de detección de deadlock.

Prueba 3	**Bloqueo, espera circular e inanición**
Objetivo	Mediante este test se evaluará el correcto uso de las colas de estado del planificador y el funcionamiento del algoritmo de detección de interbloqueo. Dado que el q=1 y que las cajas están a 20 movimientos del origen, todos los personajes deberían tomar su primer recurso cuasi simultáneamente. Luego al intentar tomar el segundo Mario, Luigi y Goomba entrarán en un interbloqueo circular y Tortuga estará en inanición.
Configuración	Esquema 2

Prueba 4	**Modificación de parámetros en tiempo de ejecución**
Objetivo	Validar la correcta implementación de inotify() sobre el archivo de configuración.
Configuración	Esquema 2

Esquema 1	
Máquinas Virtuales	Se requieren 4 máquinas virtuales para ejecutar este test. VM1, VM2, VM3, VM4. **VM1:** Level1-1 Personaje Mario **VM2:** Proceso Plataforma y Koopa Level2-1 Personaje Hongo **VM3:** Level3-1 Personaje Luigi Personaje Goomba **VM4:** Level8-8 Personaje Tortuga

Niveles	La descripción de los Recursos por cada Nivel usa el siguiente formato: Recurso, Identificador, Cantidad de instancias, PosX, PosY.
	
	`Algoritmo de detección de Interbloqueo: Activado`
	`Ejecución del algoritmo de detección de interbloqueo: 20 segundos`
	
	`Level1-1:`
	`Caja1=[Hongos, H, 3, 10, 10]`
	`Caja2=[Monedas, M, 5, 3, 19]`
	`Caja3=[Vidas, V, 2, 17, 5]`
	`Caja4=[Bloques, B, 4, 50, 21]`
	`Caja5=[Flores, F, 1, 3, 3]`
	`Level2-1:`
	`Caja1=[Chocolates, C, 8, 43, 19]`
	`Caja2=[Flores, F, 5, 19, 15]`
	`Level3-1:`
	`Caja1=[Bombas, B, 2, 9, 5]`
	`Caja2=[Telefonos, T, 3, 20, 15]`
	`Caja3=[Relojes, R, 2, 17, 3]`
	`Level8-8:`
	`Caja1=[Calaveras, C, 1, 23, 3]`
	`Caja2=[Huesos, H, 1, 60, 20]`
	`Caja3=[Espinas, E, 1, 40, 10]`

Personajes	`Mario [@]:`
	`Vidas: 5`
	`Plan de Niveles: [Level1-1, Level3-1, Level8-8]`
	`Objetivos[Level1-1]: FHMHB`
	`Objetivos[Level3-1]: BTRB`
	`Objetivos[Level8-8]: HEC`
	
	`Luigi [*]:`
	`Vidas: 5`
	`Plan de Niveles: [Level2-1, Level8-8]`
	`Objetivos[Level2-1]: FCFCFCFCF`
	`Objetivos[Level8-8]: HEC`
	
	`Hongo[#]:`

	Vidas: 5 Plan de Niveles: [Level8-8, Level3-1, Level2-1] Objetivos[Level8-8]: HEC Objetivos[Level3-1]: BTBTRT Objetivos[Level2-1]: CFC Tortuga[&]: Vidas: 5 Plan de Niveles: [Level8-8, Level3-1, Level2-1] Objetivos[Level8-8]: CHE Objetivos[Level3-1]: BTBRT Objetivos[Level2-1]: CFCFCFCFCF Goomba[$]: Vidas: 3 Plan de Niveles: [Level1-1, Level3-1, Level8-8] Objetivos[Level1-1]: HVFHVH Objetivos[Level3-1]: BRB Objetivos[Level8-8]: ECH
Plataforma	Quantum: 3 Tiempo de retardo de quantum: 0.5 seg.
Koopa	Tamaño de partición: 100 bytes GRABAR:A:10:0123456789 GRABAR:B:5:12345 GRABAR:X:30:012345678901234567890123456789 GRABAR:Z:26:ABCDEFGHIJKLMNOPQRSTUVWXYZ BORRAR:Y BORRAR:X BORRAR:A BORRAR:B GRABAR:C:15:012345678912345 GRABAR:Y:6:123456 BORRAR:Z GRABAR:W:28:0123456789012345678ABCDEFGH

Esquema 2	
Máquinas Virtuales	Se requieren 2 máquinas virtuales para ejecutar este test. **VM1:** Nivel DeadLock Proceso Plataforma y Koopa **VM2:** Personaje Tortuga Personaje Luigi Personaje Goomba Personaje Mario
Niveles	Algoritmo de detección de Interbloqueo: Activado Ejecución del algoritmo de detección de interbloqueo: 60 segundos DeadLock

	Caja1=[Hongos, H, 1, 10, 10] Caja2=[Monedas, M, 1, 18, 2] Caja3=[Manzanas, Z, 1, 2, 18] Caja4=[Bananas, B, 1, 8, 12]
Personajes	Mario [@]: Vidas: 5 Plan de Niveles: [DeadLock] Objetivos[DeadLock]: HM Luigi [*]: Vidas: 5 Plan de Niveles: [DeadLock] Objetivos[DeadLock]: MZ Goomba[#]: Vidas: 5 Plan de Niveles: [DeadLock] Objetivos[DeadLock]: ZH Tortuga[&]: Vidas: 5 Plan de Niveles: [DeadLock] Objetivos[DeadLock]: BH
Plataforma	Tiempo de retardo de quantum: 0.5 seg Quantum: 1
Koopa	Tamaño de partición: 10 bytes GRABAR:A:5:01234 GRABAR:B:5:12345 BORRAR:B BORRAR:A GRABAR:C:10:0987654321

Planilla de Evaluación - TP1C2013

Grupo: [NOMBRE DE GRUPO]
Algorítmo: [First Fit, Best Fit, Worst Fit, Next Fit]

Legajo	Nombre y Apellido	Nota

Evaluador:

Coloquio:

Prueba 1	
Existen conexiones TCP entre el personaje, el orquestador, el planificador y el nivel (netstat -nap).	
Los personajes y los niveles pueden ingresar al sistema en cualquier momento.	
Los planes de niveles y los objetivos por nivel se respetan.	
Los personajes de distintos niveles se mueven de manera simultánea mientras que los del mismo nivel se alternan en función del valor de quantum.	
La posición del personaje al momento de solicitar los recursos es la misma que la de la caja del recurso.	
Al solicitar un recurso se decrementa su contador en pantalla y se incrementa al ser devuelto.	
El tiempo de espera de quantums se respeta y es parametrizable.	
El algoritmo de detección de deadlock funciona y se resuelven las situaciones de interbloqueo.	

El uso de CPU y memoria no es excesivo (top).	
La cantidad de hilos en el sistema es la adecuada (un orquestador y un planificador por nivel).	
Concluído todos los planes de niveles se ejecuta el proceso Koopa y este concluye de forma favorable en función del algoritmo asignado al grupo.	

Prueba 2	
El personaje al morir libera los recursos que tenía tomados e inicia nuevamente el nivel desde la posición 0,0.	
Al quedarse sin vidas el personaje inicia nuevamente el plan de niveles.	
Al enviar la señal SIGTERM el personaje pierde una vida.	
Al enviar la señal SIGUSR1 el personaje recibe una vida.	
El personaje puede morir mientras está ejecutando, bloqueado o listo.	
El sistema reacciona ante la desconexión no prevista de un personaje o un nivel.	

Prueba 3	
El estado de las colas de estado (Listos, Bloqueados) del planificador es válido y comprensible.	
El algoritmo de detección de interbloqueo mata al primer personaje que ingresó al sistema.	

Prueba 4	
Todos los planificadores reaccionan a los sucesivos cambios en el valor de quantum del archivo de configuración.	

Anexo - Comandos Útiles

Copiar un directorio 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`

También se puede obtener el script de descarga automática (*link*).

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)