UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Super Mario Proc. RELOADED

Tests

Ingeniería en Sistemas de Información
Cátedra de Sistemas Operativos

- 2C2013 -

Requisitos y notas de 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.

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[1] con los datos de los integrantes completos (dejar el campo “Nota” 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. En todos los casos el documento se publicará con la debida anticipación.

[1] Al final de este documento

Condición Mínima	**Funcionamiento básico del sistema**
Objetivo	Mediante este test se validará el funcionamiento mínimo del sistema. 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
Resultados esperados	Koopa validará la correcta creación de directorios, subdirectorios y la lectura y escritura de archivos en porciones de diversos tamaños. La información deberá ser mostrada por el grupo mediante el grasa-dump. El Nivel 1 debería representar claramente el uso del SRDF En el Nivel 2 los personajes deberían inter-bloquearse y se puede mostrar la reacción del sistema al matar un personaje manualmente. También se utilizará para validar el cambio de Quantum del algoritmo RR Se debe observar un interbloqueo en el Nivel 3 y validar su resolución por el algoritmo de detección de interbloqueo. El Nivel 8 debería mostrar el correcto funcionamiento de los enemigos tanto en presencia de personajes como en su ausencia.

Koopa	`#!/bin/bash`
	
	`for i in {1..5}`
	`do`
	`touch $i.txt`
	`for j in {1..10}`
	`do`
	`mkdir -p dir$i/dir$j`
	`done`
	`done`
	
	`dd if=/dev/zero of=dir1/archivo.zero bs=1 count=102400`
	`dd if=/dev/zero of=dir3/archivo.zero bs=4095 count=3`
	`echo "HOLA MUNDO" > dir2/hola.txt`
	`cp /bin/bash archivo.bin`
	`dd if=archivo.bin of=dir5/dir5/b.bin bs=1 count=5000`

Prueba 1	**Vidas y Muertes**

Objetivo	Validar los procedimientos para la muerte del personaje y el correcto manejo de señales.
Configuración	Esquema 1
Resultados esperados	Enviar la señal SIGUSR1 y SIGTERM a un personaje y observar su comportamiento en el sistema. Validar que luego de perder todas sus vidas el personaje solicite un Continue

Prueba 2	**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.
Configuración	Esquema 2
Resultados esperados	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. Cambiar el algoritmo y el valor de quantum y validar su correcto funcionamiento en tiempo de ejecución.

Prueba FS	**Consistencia del FileSystem GRASA**
Objetivo	Corroborar los límites y capacidades del proceso FileSystem y de la implementación de GRASA. Validar todos los tests con el programa grasa-dump
Script 1	Ejecutar desde consola los siguientes comandos para intentar generar 600 archivos vacíos y 600 directorios. Debería soportar un máximo de 1024 nodos. for i in {1..600}; do truncate -s 0 $i; done for i in {601..1200}; do mkdir $i; done
Script 2	Crear un disco de 100 MB y llenarlo con un archivo grande # dd if=/dev/urandom of=archivo.bin bs=1024 count=102400

Massive File Creator	1) Instalar libssl-dev: # apt-get install libssl-dev 2) Descargar Massive-File-Creator # curl -L https://github.com/sisoputnfrba/massive-file-creator/tarball/master -o mfc.tar.gz && tar xvfz mfc.tar.gz 3) Compilar # gcc massive-file-creator.c -o mfc -lcrypto -lpthread 4) Ejecutar en el path del montaje del proceso FileSystem # ./mfc 10 1024 path/al/fs/GRASA prefix_ Esto lanzará 10 hilos y cada uno generará un archivo con nombre prefix_[x] de 1 MB en path/al/fs/GRASA de manera concurrente y luego corroborará que se hayan generado correctamente.

Esquema 1	
Máquinas Virtuales	Se requieren 3 máquinas virtuales para ejecutar este test. VM1, VM2, VM3 **VM1:** Nivel1 Personaje Mario Nivel8 **VM2:** Proceso Plataforma y Koopa Personaje Koopa **VM3:** Nivel2 Personaje Luigi Nivel3

Niveles	La descripción de los Recursos por cada Nivel usa el siguiente formato: Recurso, Identificador, Cantidad de instancias, PosX, PosY.
	
	`Nivel1:`
	`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]`
	`Enemigos=2`
	`Sleep_Enemigos=500`
	`Algoritmo=SRDF`
	`Retardo=500`
	`TiempoChequeoDeadlock=60000`

	Recovery=0 Nivel2: Caja1=[Chocolates, C, 8, 43, 19] Caja2=[Flores, F, 5, 19, 15] Enemigos=0 Sleep_Enemigos=500 Algoritmo=RR Quantum=3 Recovery=0 TiempoChequeoDeadlock=10000 Nivel3: Caja1=[Bombas, B, 2, 9, 5] Caja2=[Telefonos, T, 3, 20, 15] Caja3=[Relojes, R, 2, 17, 3] Enemigos=2 Sleep_Enemigos=2000 Algoritmo=SRDF Recovery=1 TiempoChequeoDeadlock=20000 Nivel8: Caja1=[Calaveras, C, 1, 75, 22] Caja2=[Diablitos, D, 1, 10, 2] Enemigos=9 Sleep_Enemigos=500 Algoritmo=RR Quantum=1 Retardo=500 Recovery=1 TiempoChequeoDeadlock=10000
Personajes	Mario [@]: Vidas: 5 Plan de Niveles: [Nivel1, Nivel3, Nivel8] Objetivos[Nivel1]: FHMHB Objetivos[Nivel3]: BTRB Objetivos[Nivel8]: DC Luigi [&]: Vidas: 5 Plan de Niveles: [Nivel2, Nivel8] Objetivos[Nivel2]: FCFCFCFCF Objetivos[Nivel8]: CD Koopa[#]: Vidas: 10 Plan de Niveles: [Nivel1, Nivel2, Nivel3, Nivel8] Objetivos[Nivel8]: DC Objetivos[Nivel3]: BTBTRT Objetivos[Nivel2]: CFCFCF Objetivos[Nivel1]: HMVBF

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] Enemigos=0 Sleep_Enemigos=500 Algoritmo=RR Quantum=1 Retardo=500 TiempoChequeoDeadlock=10000
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

Planilla de Evaluación - TP2C2013

Grupo: [NOMBRE DE GRUPO]

Legajo	Nombre y Apellido	Nota

Evaluador:

Coloquio:

Condiciones Mínimas	
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 en los distintos niveles se mueven de manera simultánea mientras que los del mismo nivel se alternan en función del algoritmo respetando el 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 sincronización no depende de usleep()	
La cantidad de hilos en el sistema es la adecuada (un orquestador y un planificador por nivel).	
Los enemigos se mueven en busca del personaje no bloqueado más cercano y en forma de "L" si no hay personajes	
Verificar que no haya esperas activas	

Prueba 1	
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 solicita un continue e inicia nuevamente el plan de niveles.	
Al enviar la señal SIGTERM el personaje pierde una vida, sin afectar el desempeño en el nivel.	
Al enviar la señal SIGUSR1 el personaje recibe una vida, sin afectar el desempeño en el nivel.	
El personaje puede morir mientras está ejecutando, bloqueado o listo.	
La plataforma sigue funcionando 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.	
Todos los planificadores reaccionan a los sucesivos cambios del valor de quantum, del retardo entre turnos y del algoritmo de planificación desde el archivo de configuración.	

Prueba FS	
El disco se muestra lleno al intentar crear el nodo 1025	
El disco se muestra lleno si se supera su espacio disponible	
El Filesystem implementa FUSE de forma Multihilo	

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*):

```
wget http://faqoperativos.com.ar/get-repo.sh
chmod +x get-repo.sh
./get-repo.sh diegocapusotto tp-20131c-esta_hablando_del_sisop
```

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)