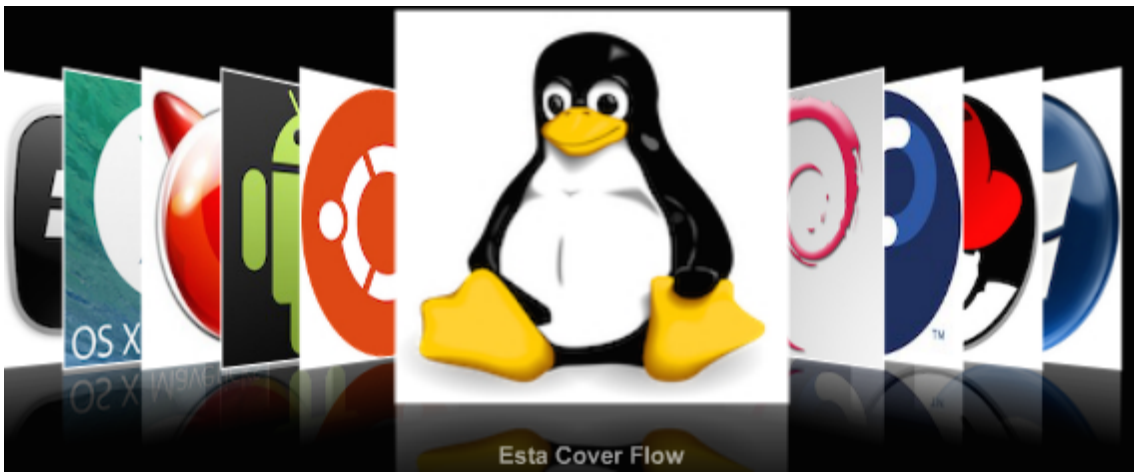
UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería en Sistemas de Información

[Está CoverFlow]

Porque para entender la recursividad primero hay que comprender la recursividad



Documento de pruebas

Cátedra de Sistemas Operativos

Trabajo práctico Cuatrimestral

- 1C2014 -
Versión [0.2]

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 ansisop que se piden se encuentran en el siguiente repositorio:
https://github.com/sisoputnfrba/scripts-ansisop[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 correción 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] Han sufrido 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.

VM1:
Kernel
Programa 5 - completo.ansisop

VM2:
CPU 2
Programa 1 - facil.ansisop
Programa 2 - consumidor.ansisop

VM3:
CPU 1
Programa 3 - productor.ansisop
Programa 6 - segfault.ansisop

VM4:
UMV
CPU 3
Programa 4 - consumidor.ansisop

El Kernel deberá tener los siguientes parámetros de configuración:

```
QUANTUM=3
RETARDO=1000
MULTIPROGRAMACION=3
SEMAFOROS=[b, c]
VALOR_SEMAFOROS=[1, 0]
ID_HIO=[HDD1, LPT1]
HIO=[500, 200]
COMPARTIDAS=[colas,compartida]
TAMANIO_STACK=300
```

La UMV tendrá una memoria de 1024 bytes, usando el algoritmo First Fit.

Desarrollo

Iniciar el Kernel, la UMV y las CPUs 1 y 2. Validar las conexiones, uso de memoria, cantidades de hilos y CPU de los procesos. Desconectar la CPU2 y validar el estado del sistema. Reconectar la CPU2 y validar que efectivamente esté conectada.

Ejecutar el script `facil.ansisop` en la VM2 (Programa 1), observando el tiempo de espera entre quantums. Verificar en la UMV que se creen y modifiquen los segmentos de memoria. Esperar a que finalice el script, y validar que la UMV haya quedado limpia.

Ejecutar el script `consumidor.ansisop` en la VM2 (Programa 2). Verificar que el Programa se bloquea esperando por el semáforo `c`.

Ejecutar el script `productor.ansisop` en la VM 3 (Programa 3). Verificar que los Programas alternan su ejecución y bloqueo, y que `consumidor` muestra números consecutivos.

Ejecutar el script `consumidor.ansisop` en la VM 4 (Programa 4). Verificar que los Programas respetan el quantum y que son planificados según Round Robin.

Ejecutar el script `facil.ansisop` nuevamente en la VM2 (Programa 1). Esperar a que finalice y volver a ejecutarlo. Verificar que se cumpla el algoritmo first-fit para la ubicación de los segmentos en la UMV.

Ejecutar el script `completo.ansisop` en la VM1 (Programa 5), y verificar que los Programas se vayan alternando en las CPUs. Ejecutar la CPU3, y verificar que los Programas comienzan a ejecutarse en ella.

Verificar que el Programa 5 concluye su ejecución, y luego enviar la señal SIGUSR1 a la CPU1. Validar que se cierre al finalizar la ráfaga que está ejecutando, y que todos Programas continúen su ejecución.

Ejecutar el script `segfault.ansisop` (Programa 6) en la VM3 y verificar que la ejecución finaliza con un error. Validar que el resto del sistema siga funcionando.

Otras pruebas

Prueba IO-PLP	**Funcionamiento IO y PLP**
Objetivo	Mediante este test se validará el funcionamiento de entrada y salida, calculo de peso y estrés. Como antes, se verificará los consumos de memoria y CPU.
Configuración	Esquema 1
Resultados esperados	Ambos **forES** entren primero, y ejecuten su correspondiente código, luego el cálculo de **fibo** (Resultado esperado dentro del script), y por ultimo **pesado**.

Prueba UMV	**Consistencia de la UMV**
Objetivo	Mediante este test se validará el funcionamiento del módulo de memoria, su algoritmo de compactación y la habilidad de responder los diferentes pedidos.
Configuración	Esquema 2
Resultados esperados	*(a definir)*

Esquema 1	
Máquinas Virtuales	Se requieren 4 máquinas virtuales para ejecutar este test. VM1, VM2, VM3, VM4 **VM1:** Kernel Programa 1 **VM2:** Programa 2 CPU 2 **VM3:** Programa 3 CPU 1 **VM4:** UMV Programa 4
Kernel	QUANTUM=3 RETARDO=1000 MULTIPROGRAMACION=2 SEMAFOROS=[b, c] VALOR_SEMAFOROS=[1, 0] ID_HIO=[HDD1, LPT1] HIO=[500, 200] COMPARTIDAS=[colas,compartida] TAMANIO_STACK=300
Cantidad de CPUs	2
UMV	MEMORIA=1024 ALGORITMO=Worst Fit
Programa 1	Script: forES.ansisop
Programa 2	Script: forES.ansisop
Programa 3	Script: pesado.ansisop
Programa 4	Script: fibo.ansisop

Esquema 2	
Máquinas Virtuales	Se requieren 4 máquinas virtuales para ejecutar este test. VM1, VM2, VM3, VM4 **VM1:** Kernel Programa 1

	VM2: Programa 2 CPU 2 **VM3:** CPU 1 **VM4:** UMV
Kernel	QUANTUM=3 RETARDO=1000 MULTIPROGRAMACION=2 SEMAFOROS=[b, c] VALOR_SEMAFOROS=[1, 0] ID_HIO=[HDD1, LPT1] HIO=[500, 200] COMPARTIDAS=[colas,compartida] TAMANIO_STACK=300
Cantidad de CPUs	2
UMV	MEMORIA=1024 ALGORITMO=First Fit
Programa 1	Script: vector.ansisop
Programa 2	Script: stackoverflow.ansisop

Planilla de Evaluación - TP1C2014

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 en cualquier momento	
Al enviar SIGUSR1 al CPU, éste se cierra al concluir la ráfaga de ejecución actual, y el Programa puede seguir ejecutando en otras CPUs	
La planificación de RR se respeta	
El Quantum es configurable y se respeta	
Las CPUs ejecutan de forma simultánea e independiente	
Al terminar un programa, la UMV queda limpia	
Se respeta el correcto funcionamiento de System Calls (semáforos y variables compartidas)	
El tiempo de espera de quantums se respeta	
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	
El algoritmo de First Fist se respeta	
La ejecución del script **completo** es independiente de los otros programas	
No hay esperas activas	
Los errores en los procesos no impiden el funcionamiento del resto del sistema (**segfault**)	

Prueba IO-PLP	
El algoritmo de peso del PLP se respeta	
El algoritmo de Worst Fit se respeta	
La ejecución del script **fibo** es independiente de los otros programas	
El estado de las colas de estado (Listos, Bloqueados) del Kernel es válido y comprensible.	
El tiempo de entrada y salida se respeta y se atiende a los pedidos de forma correcta	

Prueba UMV	
La UMV puede atender, en simultáneo, pedidos del sistema y del usuario	
El algoritmo de compactación es correcto	
El retardo se respeta y es configurable desde la consola	
La tabla de segmentos es legible y correcta	

Anexo - Comandos Útiles

- Copiar un directorio[3] 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)

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