LINUSE: Libraries IN USEr space

Documento de pruebas

Si compila está bien... pero si arranca es perfecto.

Cátedra de Sistemas Operativos

Trabajo práctico Cuatrimestral

- 2C2019 -
Versión 1.2

Requisitos y notas de la evaluación

Los requisitos expuestos a continuación se encuentran ampliados en las Normas del Trabajo Práctico, que por practicidad, se han resumido a continuación.

Deploy y Setup

Es condición necesaria para la evaluación que **el Deploy y 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).
Será responsabilidad del grupo verificar las dependencias requeridas para la compilación, y en caso de requerir bibliotecas provistas por la cátedra, descargarlas. También es responsabilidad de los integrantes del grupo conocer y manejar las herramientas de compilación desde la línea de comandos. Ver Anexo - Comandos Útiles

Cada uno de los programas a utilizar en las pruebas se encuentran en el repositorio https://github.com/sisoputnfrba/linuse-tests-programs. Dichos programas serán compilados usando el makefile provisto en el repositorio. Para correr cada programa compilado, será necesaria utilizar el comando `make nombredelaprueba`. Recomendamos leer la información del repositorio, a fin de comprender el código de las pruebas.

Evaluación

Para esta instancia, no requerimos que ningún grupo traiga impresa copia de los presentes test ni copia de la planilla de evaluación.

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

El trabajo práctico será calificado únicamente especificando si el mismo ha sido aprobado o desaprobado.

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.

Anexo - Comandos Útiles

Copiar un directorio completo por red

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

Ejemplo:

```
scp -rpC tp-2c2019-repo 192.168.3.129:/home/utnso
```

Descargar bibliotecas en un repositorio (como las commons)

git clone [url_repo]

Ejemplo:

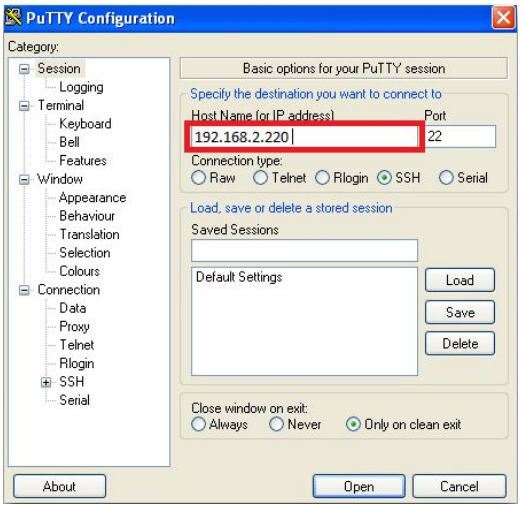
git clone https://github.com/sisoputnfrba/so-commons-library

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
- Criptografía: md5sum
- Visor de procesos del sistema: htop

Prueba Planificación

Configuración del sistema:

Cada VM deberá contar con una consola htop abierta y una consola limpia.
VM1: Proceso SUSE
VM2, VM3: N procesos productor `<Caballeros_de_SisOp_Afinador>`
, M procesos consumidor `<Caballeros de SisOP Solo>`, K procesos `<Audiencia>`.

Archivos de Configuración:

SUSE:

```
LISTEN_PORT=5003
METRICS_TIMER=70
MAX_MULTIPROG=15
SEM_IDS=[solo_hiper_mega_piola, afinado]
SEM_INIT=[1,0]
SEM_MAX=[10, 10]
ALPHA_SJF=0.5
```

Objetivos:

- Verificar el correcto funcionamiento de los semáforos dentro del sistema
- Verificar que en el sistema puedan correr concurrentemente múltiples programas
- Verificar que la planificación de los distintos procesos en SUSE se realice mediante los algoritmos estipulados en el enunciado del trabajo práctico

Actividades:

1) Levantar en una VM el proceso SUSE a la espera de recibir procesos.
2) En otra VM levantar un proceso productor `<Caballeros_de_SisOp_Afinador>`
3) Correr en múltiples máquinas varios procesos `<Audiencia>`
4) Correr en múltiples máquinas varios procesos consumidor `<Caballeros de SisOP Solo>`

Resultados Esperados:

- Se debe verificar que los hilos "Campo" de `<Audiencia>` tengan prioridad a la hora de ser elegidos para ejecutarse por sobre el hilo "Palco".
- Se debe verificar que la cantidad de ejecuciones para los consumidores iniciales sea equitativa, es decir, si el proceso productor inicial finaliza dejando el semáforo en 420 instancias y levantamos 4 consumidores, cada uno debe realizar 105 wait.

Prueba Memoria

Configuración del sistema:

Cada VM deberá contar con una consola htop abierta y una consola limpia.
VM1: Proceso MUSE
VM2: Proceso SUSE
VM1, VM2, VM3: N procesos recursivos

Archivos de Configuración:

SUSE:

```
LISTEN_PORT=5010
METRICS_TIMER=70
MAX_MULTIPROG=15
SEM_IDS=[]
SEM_INIT=[]
SEM_MAX=[]
ALPHA_SJF=0.5
```

MUSE:

```
LISTEN_PORT=5050
MEMORY_SIZE=140
PAGE_SIZE=4
SWAP_SIZE=2048
```

Objetivos:

- Verificar el uso de una memoria secundaria (swap) en caso de no contar con espacio disponible en la memoria principal
- Verificar la correcta interacción entre el módulo de Memoria y el módulo de File system
- Verificar que dentro de la memoria provista por MUSE los distintos programas puedan realizar las operaciones principales especificadas (pedir memoria, escribir, leer, liberar memoria, etc)

Actividades:

1) Levantar en una VM el proceso SUSE a la espera de recibir procesos.
2) Levantar en otra VM el programa `<Recursiva>` pasándole como parámetro de dirección de memoria el valor 10.
3) Una vez finalizado el programa anterior, levantar en múltiples máquinas varios programas `<Archivo de swap supermasivo>`

Resultados Esperados:

- Se debe verificar que inicialmente el programa recursivo imprima los valores del 10..1 y luego imprima los valores del 1..10.
- Al correr los múltiples programas recursivos se debe verificar que la memoria principal realice el proceso de swap entre memoria principal y secundaria. Se debe verificar el funcionamiento del algoritmo de reemplazo y que se esté swapeando.
  Una forma de verificar este comportamiento es analizando los cambios en el md5 del archivo de swap mediante `watch -n5 md5sum file.bin`. Para esto se puede levantar sac-server en una cuarta VM o bien revisar el archivo de swap creado por el grupo a tal fin.

Prueba File System

Configuración del sistema:

Cada VM deberá contar con una consola htop abierta y una consola limpia.
VM1: Proceso Sac-Server
VM2: Proceso Sac-Cli

Archivos de Configuración:

Sac-Sever:

```
LISTEN_PORT=8002
```

Objetivos:

- Verificar el correcto funcionamiento de la especificación de file system provista
- Verificar la integración con FUSE para poder realizar las operaciones especificadas (leer, crear y eliminar directorios y crear, leer, escribir y eliminar archivos)

Actividades:

1) Abrir una consola de Linux en la VM en donde este corriendo Sac-Cli
2) Crear un archivo con contenido aleatorio
   ```
   head -c 10M </dev/urandom >archivo.random
   ```
3) Copiar el archivo dentro del punto de montaje de SAC
4) Copiar en otra máquina el archivo desde SAC al filesystem local
5) Comparar el md5 de los 3 archivos.
6) Correr el script `<poco de todo>`

Resultados Esperados:

- Se debe verificar que el script ejecute sin errores
- Los md5s de los 3 archivos deben coincidir.
- Revisar que el script se haya ejecutado correctamente creando los directorios y archivos correctos.

Prueba Completa

Configuración del sistema:

Cada VM deberá contar con una consola htop abierta y una consola limpia.
VM1: Proceso Sac-Server
VM2: Proceso Sac-Cli
VM3: Proceso MUSE
VM4: Proceso SUSE

Archivos de Configuración:

SUSE:

```
LISTEN_PORT=8004
METRICS_TIMER=50
MAX_MULTIPROG=20
SEM_IDS=[presion_emitida,presion_recibida,
revolucion_emitida,revolucion_recibida]
SEM_INIT=[0,1,1,0]
SEM_MAX=[1,1,1,1]
ALPHA_SJF=0.4
```

MUSE:

```
LISTEN_PORT=8007
MEMORY_SIZE=4096
PAGE_SIZE=16
SWAP_SIZE=2048
```

Sac-Sever:

```
LISTEN_PORT=8002
```

Objetivos:

- Verificar el funcionamiento correcto del sistema integral, con todos los componentes interactuando entre sí
- Verificar que el módulo de Memoria tenga la capacidad de permitir a los programas compartir memoria
- Verificar que la memoria informe en caso de suceder un SEGFAULT

Actividades:

1) Levantar los cuatro procesos iniciales a la espera de peticiones por parte de programas
2) Levantar en simultáneo el script `<Estres compartido>` y el script `<Revolucion compartida>`

Aclaración: aquellos grupos que como recorte del trabajo práctico no deban realizar memoria compartida, deberán correr esta prueba primero levantando al `<Estres`

`Privado>` y una vez que el mismo termine levantar el `<Revolucion Privado>`. Los resultados esperados serán los mismos que se especifican abajo.

3) Una vez finalizados los anteriores, levantar múltiples `<Caballeros de SisOp>` con distinta cantidad de hilos cada uno

Resultados Esperados:

- Debido a que ambos procesos realizarán un muse_map(...) sobre el mismo archivo, se debe verificar que el proceso lector vaya tomando los cambios realizados por el proceso escritor
- El script `<Revolucion compartida>` o `<Revolucion Privado>` debe finalizar debido a un SEGFAULT

Planilla de Evaluación - TP2C2019

Nombre del Grupo	Nota (Grupal)

Legajo	Apellido y Nombres	Nota (Coloquio)

Evaluador/es Práctica	
Evaluador/es Coloquio	

Observaciones:

Sistema Completo	
El deploy se hace de forma automatizada y en un tiempo límite de 10 a 15 minutos	
Los procesos ejecutan de forma simultánea y la cantidad de hilos y subprocesos 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	
El sistema continúa su funcionamiento ante comandos erróneos o paths inexistentes (informándole al usuario el error)	
El sistema no requiere permisos de superuser (sudo/root) para ejecutar correctamente	
El sistema no requiere de Valgrind o algún proceso similar para ejecutar correctamente	

Módulo Planificación	
Permite sincronizar los hilos de los distintos programas mediante la administración de semáforos	
Se respeta el grado de multiprogramación para la entrada de hilos a READY	
Permite la ejecución concurrente de múltiples hilos y programas	
Mantiene un algoritmo de planificación a corto plazo del tipo SJF	
Mantiene un algoritmo de planificación a largo plazo del tipo FIFO	
Imprime las métricas correspondientes a la ejecución de los diferentes hilos y programas	
Las responsabilidades del módulo se encuentran divididas entre la biblioteca local Hilolay y el planificador central SUSE	
Mantiene los estados de ejecución indicados, e informa por consola la transición de los hilos/programas entre los mismos	

Módulo Memoria

Permite realizar pedidos de memoria por parte de los programas	
Permite realizar escrituras en memoria	
Permite realizar lecturas de memoria	
Permite realizar liberaciones de memoria	
Indica SEGFAULT en caso de sobrepasar los límites de memoria alocada por un programa	
Los distintos programas tienen la posibilidad de compartir memoria	
Se utiliza una memoria secundaria para virtualizar la memoria principal	
Al realizar el intercambio de páginas entre memoria principal y secundaria, el sistema reemplaza las mismas siguiendo el algoritmo Clock Modificado	
Funciona utilizando la implementación indicada de Segmentación Paginada	
Las responsabilidades del módulo se encuentran divididas entre la biblioteca local libMUSE y la memoria central MUSE	
Las peticiones de lecto-escritura respetan las condiciones de Bernstein y permiten paralelismo de pedidos	

Módulo File System	
Permite crear, escribir, leer y borrar archivos	
Permite crear, escribir, listar y borrar directorios	
Las estructuras del FileSystem se actualizan correctamente ante cambios en los archivos y directorios (Bitmap, tabla de bloques, …)	
Ante cambios en las estructuras del FileSystem el sistema informa correctamente cuáles son las operaciones realizadas (bloques que cambiaron, metadatas, archivos)	
Utiliza correctamente la especificación de FileSystem indicada	
Las responsabilidades del módulo se encuentran divididas entre las operaciones indicadas a FUSE y el FileSystem central Sac-Server	
Las peticiones de lecto-escritura respetan las condiciones de Bernstein y permiten paralelismo de pedidos	