Ingeniería en Sistemas de Información

Delibird

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Documento de pruebas

Cuidando el TP desde casa #QuedateEnCasa

Cátedra de Sistemas Operativos
Trabajo práctico Cuatrimestral
- 1C2020 -
Versión 1.0

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

Evaluación

Para la aprobación, un Trabajo Práctico deberá contar con todos los ítems incluidos en el listado de "***Contenidos Mínimos***" al final de este documento. 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.

Repositorio de Pruebas

Para las distintas pruebas se requerirá la ejecución de distintos Scripts. Los mismos se encontraran en el siguiente repositorio: https://github.com/sisoputnfrba/delibird-pruebas

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
```

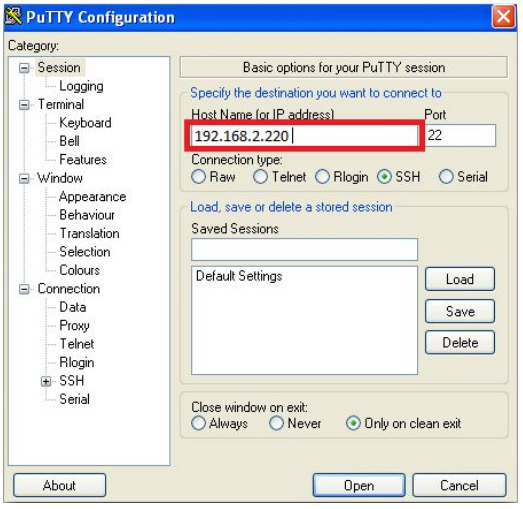
Descargar bibliotecas en un repositorio (como las commons)

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 Team

Disclaimer:

Esta prueba tiene como objetivo verificar el funcionamiento de la planificación y el proceso Team en particular, sin la intervención de los demás módulos. La misma se encargará de verificar los distintos algoritmos de planificación, la detección y resolución de deadlocks.

Actividades:

- Iniciar el proceso con el algoritmo FIFO, ejecutar el script prueba_final_team.sh y verificar que el proceso finalice
- Iniciar el proceso con el algoritmo RR con Quantum 2, ejecutar el script prueba_final_team.sh y verificar que el proceso finalice
- Iniciar el proceso con el algoritmo SJF, ejecutar el script prueba_final_team.sh y verificar que el proceso finalice
- Iniciar el proceso con el algoritmo SJF con Desalojo, ejecutar el script prueba_final_team.sh y verificar que el proceso finalice

Resultados esperados:

- El funcionamiento del algoritmo FIFO debería detectar un deadlock entre el proceso A y C
- El funcionamiento del algoritmo RR debería detectar un deadlock triple entre los procesos
- Como C finaliza antes que la primer captura de A, el mismo se dirige a atrapar a Gengar generando así un deadlock triple
- El funcionamiento del algoritmo SJF debería ser similar al algoritmo FIFO pero deberían dar distintas métricas.
- El funcionamiento del algoritmo SJF con Desalojo debería ser similar al SJF pero con distintas métricas
- Comparar métricas obtenidas de los distintos algoritmos

Configuración del sistema:

```
POSICIONES_ENTRENADORES=[1|3,2|3,3|2]
POKEMON_ENTRENADORES=[Pikachu]
OBJETIVOS_ENTRENADORES=[Pikachu|Squirtle,Pikachu|Gengar,Squirtle|Onix]
TIEMPO_RECONEXION=30
RETARDO_CICLO_CPU=5
ALGORITMO_PLANIFICACION=FIFO
QUANTUM=2
ALPHA=0.5
ESTIMACION_INICIAL=5
IP_BROKER=[A definir por el grupo]
PUERTO_BROKER=[A definir por el grupo]
LOG_FILE=[A definir por el grupo]
```

Prueba Consolidación Broker - Particiones Dinámicas

Disclaimer:

Esta prueba tiene como objetivo verificar el funcionamiento de la consolidación de particiones dinámicas del proceso broker, sin la intervención de los demás módulos. Para esto, será necesario la ejecución del proceso broker y de scripts proporcionados por la cátedra.

Actividades:

1. Ejecutar Script ***consolidacion_basico.sh***
2. Se debe llenar la memoria y luego realizar una suscripción. Finalizada la misma se realiza un nuevo mensaje CATCH
3. Bajo FIFO se eliminan las tres primeras particiones consolidándose y se guarda el último mensaje en la posición 0
4. Bajar el sistema, cambiar algoritmo de reemplazo a LRU y volverlo a levantar ejecutando el script del **paso 1**
5. Al ejecutar la suscripción se actualizan los tiempos de LRU de los primeros dos mensajes por lo que se deben borrar los dos siguientes (Catch Pikachu y Catch Squirtle), guardándose posteriormente el nuevo mensaje en la posición 8

Resultados Esperados:

1. Validar el funcionamiento de la consolidación para el algoritmo de reemplazo FIFO
2. Validar el funcionamiento de la consolidación para el algoritmo de reemplazo LRU

Configuración del sistema:

```
TAMANO_MEMORIA=64
TAMANO_MINIMO_PARTICION=4
ALGORITMO_MEMORIA=PARTICIONES
ALGORITMO_REEMPLAZO=FIFO
ALGORITMO_PARTICION_LIBRE=FF
IP_BROKER=[A Definir por el grupo]
PUERTO_BROKER=[A Definir por el grupo]
FRECUENCIA_COMPACTACION=10
LOG_FILE=[A Definir por el grupo]
```

Prueba Compactación Broker - Particiones Dinámicas

Disclaimer:

Esta prueba tiene como objetivo verificar el funcionamiento de la compactación de particiones dinámicas del proceso broker independiente de los demás. Para esto, será necesario la ejecución del proceso broker y de scripts proporcionados por la cátedra.

Actividades:

1. Ejecutar Script ***compactacion_basico.sh***
2. Se debe llenar la memoria y luego realizar una suscripción. Finalizada la misma se realiza un nuevo mensaje NEW
3. Bajo FIFO se eliminan las primera partición, se ejecuta la compactación, y se repite el mismo proceso 2 vece más. Al finalizar, se almacena el mensaje en la posición 36
4. Bajar sistema, cambiar algoritmo de reemplazo a LRU y volverlo a levantar ejecutando el script
5. Al ejecutar la suscripción se actualizan los tiempos de LRU de los primeros dos mensajes por lo que se deben borrar el mensaje Catch Pikachu. Al compactar se guarda el mensaje nuevo en la posición 44

Resultados Esperados:

1. Validar el funcionamiento de la compactación para el algoritmo de reemplazo FIFO
2. Validar el funcionamiento de la compactación para el algoritmo de reemplazo LRU

Configuración del sistema:

```
TAMANO_MEMORIA=64
TAMANO_MINIMO_PARTICION=4
ALGORITMO_MEMORIA=PARTICIONES
ALGORITMO_REEMPLAZO=FIFO
ALGORITMO_PARTICION_LIBRE=FF
IP_BROKER=[A Definir por el grupo]
PUERTO_BROKER=[A Definir por el grupo]
FRECUENCIA_COMPACTACION=1
LOG_FILE=[A Definir por el grupo]
```

Prueba Memoria Broker - Buddy System

Disclaimer:

Esta prueba tiene como objetivo verificar el funcionamiento del esquema de memoria Buddy System. Para esto, será necesario la ejecución del proceso broker y de los scripts proporcionados por la cátedra.

Actividades:

1. Ejecutar Script ***buddy_basico.sh***
2. Se deben cargar los mensajes en las siguientes posiciones de memoria:
    a. Caught 0
    b. Caught 4
    c. New pikachu 32
    d. Catch onyx 16
3. Bajo FIFO, se eliminan las siguientes particiones:
    a. Caught 0
    b. Caught 4
    c. New pikachu 32

    Se guarda el Catch Charmander en la posición 32
4. Bajar el sistema, cambiar algoritmo de reemplazo a LRU y volverlo a levantar ejecutando el script del **paso 1**
5. Se deben cargar los mensajes en las siguientes posiciones de memoria:
    a. Caught 0
    b. Caught 4
    c. New pikachu 32
    d. Catch onyx 16
6. Bajo LRU, se eliminaran la siguientes particiones:
    a. Caught 0
    b. Caught 4
    c. Catch Onyx 16

    Se guarda Catch Charmander en la posición 0

Resultados Esperados:

1. Validar el funcionamiento del sistema con el esquema de Buddy System para el algoritmo de reemplazo FIFO
2. Validar el funcionamiento del sistema con el esquema de Buddy System para el algoritmo de reemplazo LRU

Configuración del sistema:

```
TAMANO_MEMORIA=64
TAMANO_MINIMO_PARTICION=4
ALGORITMO_MEMORIA=BS
ALGORITMO_REEMPLAZO=FIFO
ALGORITMO_PARTICION_LIBRE=FF
IP_BROKER=[A Definir por el grupo]
PUERTO_BROKER=[A Definir por el grupo]
FRECUENCIA_COMPACTACION=1
LOG_FILE=[A Definir por el grupo]
```

Prueba Game Card

Disclaimer:

El objetivo de la prueba es verificar el correcto funcionamiento del File System individualmente. Para esto, se requerirá la ejecución de una instancia del Proceso GameCard y de los scripts que provee la cátedra.

Actividades:

1. Iniciar el File System
2. Verificar que no existan archivos dentro del mismo
3. Ejecutar el script ***new_pikachu.sh***
4. Se creo la carpeta Pikachu y su metadata indica que el tamaño es 7 bytes
5. Ejecutar el script ***new_pokemons_varios.sh***
6. El tamaño del archivo Pikachu se haya actualizado a 13 bytes
7. Se creo la carpeta Charmander y su metadata indique que posee dos bloques y su tamaño es 70 bytes
8. Ejecutar el script ***catch_charmander.sh***
9. Verificar que el archivo Charmander ahora indique que posee solo un bloque y su tamaño es 61 bytes

Resultados Esperados:

Dentro del file system de linux y el punto de montaje, validar:

1. Verificar que ejecutando el script pikachu (una única sentencia), el file system cree el archivo y asigne correctamente los bloques y bytes
2. Verificar que al realizar múltiples NEW sobre un mismo pokémon, el archivo crezca en tamaño de bloques
3. La asignación de bloques se realiza correctamente
4. Verificar que al realizar catch se libere espacio en el archivo
5. Verificar que al liberar espacio suficiente, se liberen los bloques innecesarios

Configuración del sistema:

```
BLOCK_SIZE=64
BLOCKS=1024
MAGIC_NUMBER=TALL_GRASS
```

Prueba Completa

Disclaimer:

El objetivo de la prueba es verificar el correcto funcionamiento del sistema global incluyendo mensajes Localized en el sistema.

Actividades:

1. Iniciar el proceso Broker
2. Iniciar el proceso FileSystem
3. Ejecutar el script ***new_pokemon_antes_team.sh***
4. Ejecutar el proceso Team 1 y Team 2
5. Una vez que ambos procesos Teams queden sin moverse, ejecutar el script ***new_pokemon_post_team.sh***
6. Esperar a que ambos procesos Team finalicen

Resultados Esperados:

1. Verificar el estado del file system previa a la ejecución de los procesos Team
2. Verificar el estado del file system post ejecución de ambos proceso Team y ejecución de catch
3. Verificar que ambos procesos Team finalicen cumpliendo su objetivo
4. Verificar que la memoria vaya realizando los reemplazos necesarios para el correcto funcionamiento
5. Bajar aplicacion y reemplazar:
    a. El algoritmo FIFO por SJF y RR por SJF con Desalojo
    b. El algoritmo Particiones dinámicas por Buddy System

    Volver a levantar aplicación y verificar el funcionamiento del mismo

Configuración del sistema:

Broker:

```
TAMANO_MEMORIA=64
TAMANO_MINIMO_PARTICION=4
ALGORITMO_MEMORIA=BS
ALGORITMO_REEMPLAZO=FIFO
ALGORITMO_PARTICION_LIBRE=FF
IP_BROKER=[A Definir por el grupo]
PUERTO_BROKER=[A Definir por el grupo]
FRECUENCIA_COMPACTACION=1
LOG_FILE=[A Definir por el grupo]
```

File System:

```
BLOCK_SIZE=64
BLOCKS=1024
MAGIC_NUMBER=TALL_GRASS
```

Team 1:

```
POSICIONES_ENTRENADORES=[1|3,2|3,2|2]
POKEMON_ENTRENADORES=[Pikachu]
OBJETIVOS_ENTRENADORES=[Pikachu|Squirtle,Pikachu|Gengar,Squirtle|Onix]
TIEMPO_RECONEXION=30
RETARDO_CICLO_CPU=5
ALGORITMO_PLANIFICACION=FIFO
QUANTUM=0
ESTIMACION_INICIAL=5
IP_BROKER=[A definir por el grupo]
PUERTO_BROKER=[A definir por el grupo]
LOG_FILE=[A definir por el grupo]
```

Team 2:

```
POSICIONES_ENTRENADORES=[2|3,6|5,9|9,9|2,2|9]
POKEMON_ENTRENADORES=[]
OBJETIVOS_ENTRENADORES=[Vaporeon,Jolteon,Flareon,Umbreon,Espeon]
TIEMPO_RECONEXION=30
RETARDO_CICLO_CPU=5
ALGORITMO_PLANIFICACION=RR
QUANTUM=1
ESTIMACION_INICIAL=5
IP_BROKER=[A definir por el grupo]
PUERTO_BROKER=[A definir por el grupo]
LOG_FILE=[A definir por el grupo]
```

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 propio	
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 (informando al usuario el error)	
El sistema no requiere permisos de superuser (sudo/root) ni Valgrind para ejecutar correctamente	

Módulo Team	
Se respeta el manejo y administración de entrenadores con las necesidades de cada uno.	
Se respeta el grado de multiprocesamiento para la entrada de entrenadores a EXEC	
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 corto plazo del tipo FIFO	
Mantiene un algoritmo de planificación a corto plazo del tipo RR	
Mantiene un algoritmo de planificación a corto plazo del tipo SJF Con Desalojo	
Imprime las métricas correspondientes al finalizar el proceso.	
Se detectan y resuelven correctamente los distintos deadlocks.	
Mantiene un log estable en el cual se ve el estado en todo momento del módulo	

Módulo Broker	
Permite realizar escrituras en memoria	
Permite realizar lecturas de memoria	
Permite liberar correctamente la memoria	
El sistema respeta las especificaciones de Particiones dinámicas	
El sistema respeta las especificaciones de Buddy System	
El sistema respeta el funcionamiento de la compactación	
El sistema respeta el funcionamiento de suscriptores y distribución de mensajes	
Se respeta el algoritmo Best Fit para la asignación de espacios en memoria	
Se respeta el algoritmo First Fit para la asignación de espacios en memoria	
Se respeta el algoritmo LRU para el reemplazo de particiones de memoria	
Se respeta el algoritmo FIFO para el reemplazo de particiones de memoria	
El sistema maneja un multihilos. La cantidad de hilos utilizada es adecuada.	
Mantiene un log estable en el cual se ve el estado en todo momento del módulo	

Módulo Game Card	
Permite crear, escribir y leer archivos	
Las estructuras del FileSystem se actualizan correctamente ante cambios en los archivos y directorios (Bitmap, 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 peticiones de lecto-escritura respetan las condiciones de Bernstein y permiten paralelismo de pedidos	
El sistema maneja un multihilos. La cantidad de hilos utilizada es adecuada.	