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 1.3

Versión de Cambios

v1.1 (15/06/2020) Primera Revisión

- *Se arreglan las posiciones de memoria para la Prueba Base del Broker.*
- *Se arreglan los ID's al script new_pikachu.sh y new_pokemons_varios.sh*
- Se agregan test de Prueba Consolidación Broker Particiones Dinámicas, Prueba Compactación Broker Particiones Dinámicas, Prueba Team Espera Circular.

v1.2 (21/06/2020) Segunda Revisión

- *Se arreglan los scripts de compactación y consolidación.*
- Se agrega test de Prueba de Buddy System.

v1.3 (21/06/2020) Segunda Revisión

- *Se arreglan las posiciones de los entrenadores D y E en la prueba de Espera Circular*

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.

Dadas las condiciones del trabajo práctico, el objetivo de este documento es orientar a los alumnos a pruebas que permitirán la evaluación final del mismo. Dado esto, puede aparecer algún contenido que se encuentre próximo a implementar y ayude al grupo a verificar el correcto funcionamiento del trabajo práctico.

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

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

Prueba Base File System

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 la ejecución de 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 realice 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:

Metadata FileSystem

```
BLOCK_SIZE=64
BLOCKS=4096
MAGIC_NUMBER=TALL_GRASS
```

Prueba Base Broker

Disclaimer:

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

Actividades:

1. Ejecutar script ***cargar_memoria_simple.sh***
2. Verificar a partir del log que se haya creado los mensaje dentro de la memoria en la posiciones 0 y 11.
3. Ejecutar script ***cargar_memoria_complejo.sh***
4. Verificar que la memoria quede en el estado adecuado.
5. Reiniciar Broker cambiando el tamaño de partición mínima a 16 y ejecutar el script ***cargar_memoria_complejo.sh***

Resultados Esperados:

1. Se verifica que se creen las particiones adecuadamente administrando los datos como indica el Anexo. Verificar por medio del log las posiciones deben quedar de la siguiente manera:

Tamaño mínimo de partición 4: Posiciones 0, 11, 25, 44, 64, 68, 72, 93, 115, 124
Tamaño mínimo de partición 16: Posiciones 0, 19, 39, 55, 71, 92, 114, 130

Configuración del sistema:

```
TAMANO_MEMORIA=4096
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 Base Team

Disclaimer:

Esta prueba tiene como objetivo verificar el funcionamiento de la planificación bajo el algoritmo FIFO. Para la misma, se requiere la ejecución de un proceso Team.

Actividades:

1. Ejecutar el Módulo Team con la configuración especificada.
2. Ejecutar el script ***catch_pokemon_escenario_1_s1.sh***
3. Esperar a que los tres entrenadores se muevan y atrapen los respectivo pokemon.
4. Ejecutar el script ***catch_pokemon_escenario_1_s2.sh***
5. Esperar a que el entrenador B y entrenador C se muevan a atrapar los pokemon.
6. Verificar que se detecten dos deadlocks diferentes y que se resuelvan.

Resultados Esperados:

1. El entrenador A se mueve a atrapar a Pikachu en posición 1-1.
2. El entrenador C se mueve a atrapar a Squirtle en posición 9-7.
3. El entrenador B se mueve a atrapar a Onix en posición 2-2.
4. El entrenador B se mueve a atrapar a Squirtle en posición 3-5.
5. El entrenador C se mueve a atrapar a Gengar en posición 7-5.
6. Se detecta deadlock entre A-B.
7. Se detecta deadlock entre B-C.
8. Se mueve A o B a la posición del otro e intercambian Pikachu-Squirtle. Finaliza el Entrenador A.
9. Se mueve B o C a la posición del otro e intercambian Onix-Gengar. Finaliza el Entrenador C.
10. Una vez resuelto ambos deadlock, finaliza el entrenador B.
11. Finaliza el proceso Team.

Todos los resultados se validan por medio del archivo log.

Configuración del sistema:

```
POSICIONES_ENTRENADORES=[2|3,6|5,9|9]
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]
```

Prueba Consolidación Broker - Particiones Dinámicas

Disclaimer:

Esta prueba tiene como objetivo verificar el funcionamiento de la consolidación 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 ***consolidacion_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 tres primeras particiones consolidándose y se guarda el último mensaje en la posición 0.
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 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 Team - Espera Circular

Disclaimer:

Esta prueba tiene como objetivo verificar el funcionamiento de la espera circular bajo el algoritmo RR. Para la misma, se requiere la ejecución de un proceso Team.

Actividades:

1. Ejecutar el Módulo Team con la configuración especificada.
2. Ejecutar el script ***espera_circular_basico.sh***
3. Esperar que los 5 entrenadores se muevan y atrapen un pokemon.
4. Se debe detectar un deadlock con espera circular entre los 5 y resolverse.
5. Reiniciar el aplicativo cambiando el algoritmo a Quantum 2 y verificar que funcione igual.
6. Reiniciar el aplicativo cambiando el algoritmo a FIFO y verificar que funcione igual.

Resultados Esperados:

1. El entrenador A se mueve a atrapar a Jolteon en posición 2-2.
2. El entrenador B se mueve a atrapar a Flareon en posición 4-6.
3. El entrenador C se mueve a atrapar a Umbreon en posición 10-6.
4. El entrenador D se mueve a atrapar a Espeon en posición 7-1.
5. El entrenador E se mueve a atrapar a Vaporeon en posición 4-10.
6. Se detecta un deadlock con espera circular entre los 5 entrenadores.
7. Se resuelve el deadlock y finalizan los distintos entrenadores.
8. Finaliza el proceso Team.

Todos los resultados se validan por medio del archivo log.

Configuración del sistema:

```
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]
```

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 scripts proporcionados por la cátedra.

Actividades:

1. Ejecutar Script ***buddy_basico.sh***
2. Se debe 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 sistema, cambiar algoritmo de reemplazo a LRU y volverlo a levantar ejecutando el script.
5. Se debe 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]
```

Anexo - Cómo conectarse por SSH a una Máquina Virtual

Copiar un directorio completo por red

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

Ejemplo:

```
scp -rpC tp-1c2015-repo 192.168.3.129:/home/utnso
```

Descargar bibliotecas en un repositorio (como las commons)

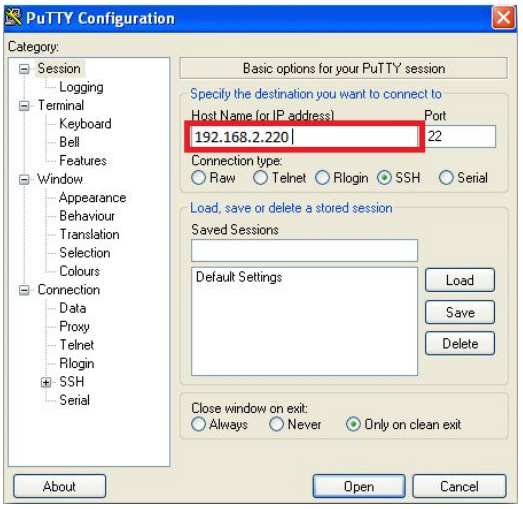
git clone [url_repo]

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.