SISTEMAS OPERATIVOS

PRÁCTICA 4 - PROYECTO FINAL: MINER RUSH

Pablo Pollo, Christian Grosso - Grupo 2262

Dato a tener en cuenta:

Debido a que el comando para el primer ejemplo proporcionado es:
./monitor & ./miner 2 1 & ./miner 3 1

Se lanza el monitor y el primer miner en background debido a los "&" pero al no tenerlo el segundo miner, este último se lanza en foreground. Por esto mismo es posible que al ejecutar el comando "./monitor & ./miner 2 1 & ./miner 3 1" la shell no considere ese proceso como un job en background y los mensajes al terminar de dicho miner no aparezcan en la salida.
Puede ser que se necesite pulsar la tecla "Enter" para ver toda la salida correspondiente.

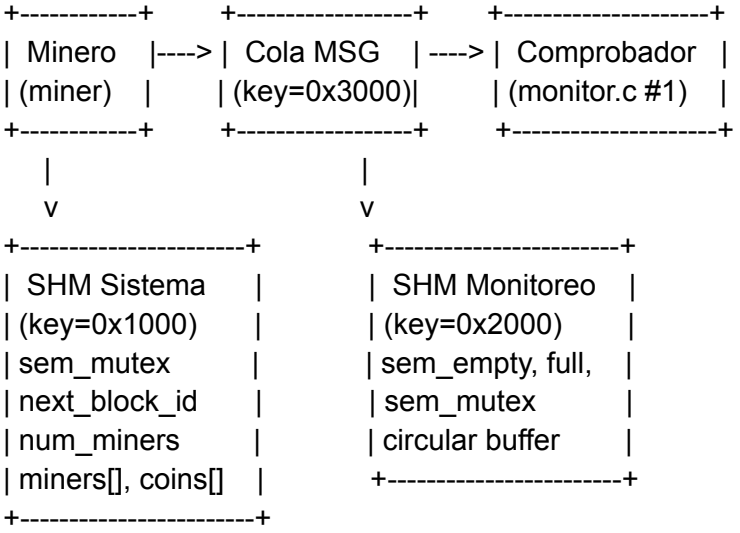
Además, intuimos que por esto mismo, cuando ejecutas el comando no sale visible el PID del segundo miner y muy posiblemente en algunas rondas ese PID sea el ganador. Por lo que puede resultar extraño ver un PID ganador cuando no lo vemos por consola pero se ha de tener en cuenta que ocurre que por lo mismo explicado más arriba.

También se podría ejecutar el mismo comando pero con un & al final, quedando así:
./monitor & ./miner 2 1 & ./miner 3 1 &

Esperamos que nos hayamos explicado de forma correcta.

ENTREGABLE 1 - MEMORIA

a) Diagrama del sistema



b) Descripción del diseño y principales problemas

1. Minero (miner.c)

-Conecta o crea SHM Sistema (clave 0x1000) y sem_mutex.

-Bajo mutex: registra su PID, inicializa coins[], añade a num_miners.

-Para cada ronda:
1.Resuelve POW (pow_hash).
2.Bajo mutex:
Toma y aumenta next_block_id → ID único global.
Incrementa su propia coins[idx].
Hace snapshot de miners[] y coins[] en el bloque.
3.Envía el bloque por msgsnd a la cola.

-Al terminar:
Bajo mutex, decrementa num_miners.
Si es el último, envía TERMINATION_ID, elimina cola y SHM Sistema; si no, solo se desconecta.

2. Comprobador (monitor.c, proceso padre)

-Conecta o crea SHM Monitoreo (clave 0x2000) con tres semáforos anónimos.

-Bucle de msgrcv de la cola:
Down(empty); Down(mutex); buffer[in]=blk; in=(in+1)%N; Up(mutex); Up(full);

-Sale al recibir bloque con ID = TERMINATION_ID, limpia cola, semáforos y SHM Monitoreo.

3. Monitor de impresión (monitor.c, proceso hijo)

-Extrae bloques con el esquema inverso:
Down(full); Down(mutex); blk=buffer[out]; out=(out+1)%N; Up(mutex); Up(empty);

-Imprime con formato exactamente:
Id : %04d
Winner : %d
Target : %08ld
Solution : %08ld (validated | rejected)
Votes : %d/%d
Wallets : pid1:coins1 pid2:coins2 …

-Al bloque de terminación, libera recursos y termina.

* Problemas abordados *

IDs duplicados → contador global next_block_id en SHM Sistema.

Carteras inconsistentes → snapshot de coins[] bajo mutex justo tras incrementar.

Terminación prematura → solo el último minero envía el bloque de terminación.

Recursos huérfanos → limpieza controlada (último en cada recurso).

c) Limitaciones y pruebas

* Limitaciones

-Límite estático de 100 mineros (MAX_MINERS).

-No hay recuperación si un minero muere inesperadamente durante una ronda.

-Uso de espera activa (usleep), que podría no escalar en sistemas muy lentos.

* Pruebas realizadas

1. Desfase de velocidades: mineros y monitor con delays distintos, confirmando que ningún bloque se pierde ni se imprime fuera de orden.

2. Múltiples rondas: ./miner 10 1 junto a ./monitor, verificando IDs secuenciales sin huecos ni duplicados.

3. Concurrencia: lanzar 3–5 mineros simultáneos, comprobar carteras finales y conteo de votos.

4. Limpieza: después de la ejecución, uso de ipcs/ipcrm para confirmar que no quedan colas ni segmentos compartidos.

* Errores conocidos

-Si un minero muere antes de desregistrarse, num_miners no se decrementa y no llega la terminación.

-El sistema no soporta reinicio de un minero con el mismo PID.