

Universidad Autónoma de Madrid

Departamento de Informática

Sistemas Operativos

Memorias Práctica - 4

Hecho por: Diego Rodríguez Ortiz

y Alejandro García Hernando

Introducción

En esta práctica hemos combinado todo el conocimiento adquirido a lo largo del curso, para crear un software que cree una red de mineros de bloques inspirada en la tecnología Blockchain. Para ello, hemos tenido que manejar la ejecución paralela de un proceso mediante hilos, generación de nuevos procesos mediante forks, memoria compartida y distintas herramientas para gestionar programas que se ejecutan paralelamente.

El proyecto se divide en 4 tipos de procesos que se dividen el trabajo de minado de tal forma que todo se haga de la forma más eficaz posible. Los Mineros se encargan de buscar las soluciones a los problemas hash planteados, y aquel que obtenga la solución correcta más rápidamente obtiene una moneda a cambio; además, después de cada ciclo de minado le mandan a su proceso hijo, Registrador, un bloque de información que indica el resultado de la ronda de minería. Por otro lado, el proceso Comprobador se encarga de asegurarse de que las soluciones ofrecidas por los Mineros son correctas y mandarlas al proceso Monitor para que las imprima por pantalla.

Por otro lado, la comunicación entre Comprobador y Monitor se debe hacer mediante un sistema productor consumidor, y entre Minero y Registrador a través de una pipeline. El Minero que encuentre la solución al problema más rápido, debe coordinar una votación en la que el resto de procesos Minero deben legitimar su solución propuesta, y posteriormente mandarla al proceso Comprobador. La comunicación entre Mineros se hace mediante memoria compartida, y el proceso ganador envía los resultados a través de una cola de mensajes.

Para evitar problemas de concurrencia y condiciones de carrera indeseadas hay que coordinar los procesos mediante semáforos. Además, cabe comentar que en caso de error en algún proceso se liberan todos los recursos empleados y en caso de ser necesario se notifica a los procesos que dependen de él. Por último, se incluyen diversos comandos en un fichero Makefile que pueden ser empleados para liberar recursos en caso de error fatal.

Estructura de código

El proyecto contiene 11 ficheros: 6 de código y 5 auxiliares

-Makefile: contiene una serie de comandos para la compilación y ejecución del código. Además de un comando que permite comprobar que se han cerrado correctamente los semáforos y memoria compartida y un segundo que los elimina en caso de que no se haya cerrado correctamente.

– minero.c: la función principal del proceso Minero se encuentra aquí. Esta función recoge los parámetros introducidos por el usuario y crea la Pipeline que utilizará para comunicarse con su proceso Registrador. Posteriormente se divide en un proceso Registrador y otro Minero, quién esperará a que su hijo termine para poder finalizar correctamente.

– monitor.c: la función principal de Comprobador y Monitor se encuentra en este archivo. Esta función recoge los parámetros introducidos por el usuario y se divide en un proceso Comprobador y otro Monitor, quién esperará a que su hijo termine para poder finalizar correctamente.

– funciones_minero.c y funciones_minero.h: son archivos creados para albergar las distintas funciones que ejecutarán tanto los procesos Minero como Registrador. Aquí se encuentra todas las funciones asociadas a la minería, votación y compartición de información asociada a estos procesos. También se encuentran las distintas rutinas encargadas de manejar las señales de estos procesos y de la finalización de los mismos.

– funciones_monitor.c y funciones_monitor.h: son archivos creados para albergar las distintas funciones que ejecutarán tanto los procesos Monitor como Comprobador. Aquí se encuentran todas las funciones de comprobación de resultados, impresión de los datos y compartición de información asociada a estos procesos. También se encuentran las distintas rutinas encargadas de la finalización de los mismos.

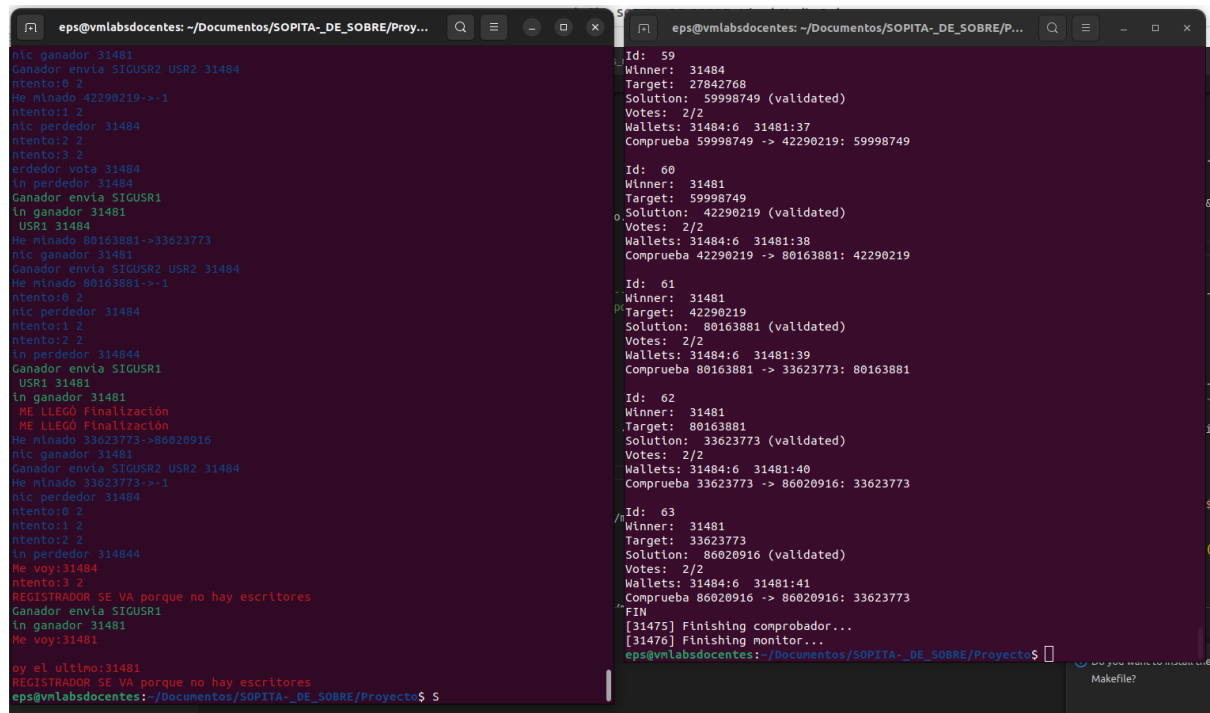
– utils.c y utils.h: estos archivos cuentan con las funciones auxiliares que son llamadas desde los archivos ya mencionados. Las funciones de estos archivos son las empleadas para la sincronización de procesos y para facilitar que los procesos se desincronicen, y comprobar si el programa falla o por el contrario funciona correctamente.

-pow.c y pow.h: contienen únicamente la función hash facilitada.

Pruebas

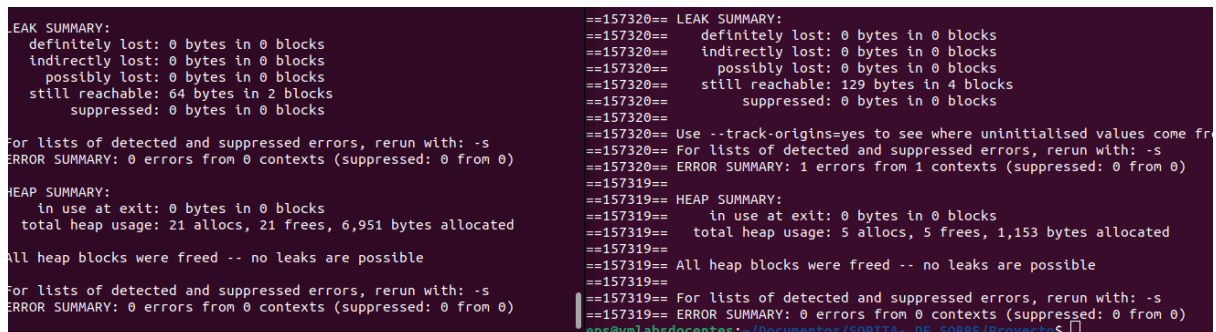
Finalización correcta de todos los procesos con DEBUG

Para asegurarnos de que nuestro programa funciona correctamente definimos la macro DEBUG que imprime por pantalla numerosos comentarios que nos permiten ver la evolución de los procesos. En la siguiente imagen se aprecia la traza de ejecución de los procesos, como se liberan correctamente todos los recursos y un ejemplo de una cartera de Minero.



```
eps@vmlabsdocentes: ~/Documentos/SOPITA_DE_SOBRE/Proy...  
mic ganador 31481  
Ganador envia SIGUSR2 USR2 31484  
Intento:0 2  
He minado 42290219->-1  
Intento:1 2  
mic perdedor 31484  
Intento:1 2  
Intento:2 2  
Intento:3 2  
perdedor vota 31484  
In perdedor 31484  
Ganador envia SIGUSR1  
In ganador 31481  
USR1 31484  
He minado 80163881->33623773  
mic ganador 31481  
Ganador envia SIGUSR2 USR2 31484  
He minado 80163881->-1  
Intento:0 2  
mic perdedor 31484  
Intento:1 2  
Intento:2 2  
In perdedor 31484  
Ganador envia SIGUSR1  
USR1 31481  
In ganador 31481  
ME LLEGÓ Finalización  
ME LLEGÓ Finalización  
He minado 33623773->86020916  
mic ganador 31481  
Ganador envia SIGUSR2 USR2 31484  
He minado 33623773->-1  
mic perdedor 31484  
Intento:0 2  
Intento:1 2  
Intento:2 2  
In perdedor 31484  
He voy:31484  
Intento:3 2  
REGISTRADOR SE VA porque no hay escritores  
Ganador envia SIGUSR1  
In ganador 31481  
He voy:31481  
ay el ultimo:31481  
REGISTRADOR SE VA porque no hay escritores  
eps@vmlabsdocentes:~/Documentos/SOPITA_DE_SOBRE/Proyecto$ S  
eps@vmlabsdocentes:~/Documentos/SOPITA_DE_SOBRE/Proyecto$  
Id: 59  
Winner: 31484  
Target: 27842768  
Solution: 59998749 (validated)  
Votes: 2/2  
Wallets: 31484:6 31481:37  
Comprueba 59998749 -> 42290219: 59998749  
Id: 60  
Winner: 31481  
Target: 59998749  
Solution: 42290219 (validated)  
Votes: 2/2  
Wallets: 31484:6 31481:38  
Comprueba 42290219 -> 80163881: 42290219  
Id: 61  
Winner: 31481  
Target: 42290219  
Solution: 80163881 (validated)  
Votes: 2/2  
Wallets: 31484:6 31481:39  
Comprueba 80163881 -> 33623773: 80163881  
Id: 62  
Winner: 31481  
Target: 80163881  
Solution: 33623773 (validated)  
Votes: 2/2  
Wallets: 31484:6 31481:40  
Comprueba 33623773 -> 86020916: 33623773  
Id: 63  
Winner: 31481  
Target: 33623773  
Solution: 86020916 (validated)  
Votes: 2/2  
Wallets: 31484:6 31481:41  
Comprueba 86020916 -> 86020916: 33623773  
FIN  
[31475] Finishing comprobador...  
[31476] Finishing monitor...  
eps@vmlabsdocentes:~/Documentos/SOPITA_DE_SOBRE/Proyecto$  
Makefile?
```

Ejecución del minero y del monitor



```
LEAK SUMMARY:  
definitely lost: 0 bytes in 0 blocks  
indirectly lost: 0 bytes in 0 blocks  
possibly lost: 0 bytes in 0 blocks  
still reachable: 64 bytes in 2 blocks  
suppressed: 0 bytes in 0 blocks  
For lists of detected and suppressed errors, rerun with: -s  
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)  
HEAP SUMMARY:  
in use at exit: 0 bytes in 0 blocks  
total heap usage: 21 allocs, 21 frees, 6,951 bytes allocated  
All heap blocks were freed -- no leaks are possible  
For lists of detected and suppressed errors, rerun with: -s  
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)  
eps@vmlabsdocentes:~/Documentos/SOPITA_DE_SOBRE/Proyecto$  
==157320== LEAK SUMMARY:  
==157320== definitely lost: 0 bytes in 0 blocks  
==157320== indirectly lost: 0 bytes in 0 blocks  
==157320== possibly lost: 0 bytes in 0 blocks  
==157320== still reachable: 129 bytes in 4 blocks  
==157320== suppressed: 0 bytes in 0 blocks  
==157320== Use --track-origins=yes to see where uninitialised values come from  
==157320== For lists of detected and suppressed errors, rerun with: -s  
==157320== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)  
==157319== HEAP SUMMARY:  
==157319== in use at exit: 0 bytes in 0 blocks  
==157319== total heap usage: 5 allocs, 5 frees, 1,153 bytes allocated  
==157319== All heap blocks were freed -- no leaks are possible  
==157319== For lists of detected and suppressed errors, rerun with: -s  
==157319== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)  
eps@vmlabsdocentes:~/Documentos/SOPITA_DE_SOBRE/Proyecto$
```

Resumen de Valgrind

```

410 Target: 97092092
411 Solution: 27842768
412 Votes: 2/2
413 Wallets: 31484:5 31481:37
414
415 Id: 59
416 Winner: 31484
417 Target: 27842768
418 Solution: 59998749
419 Votes: 2/2
420 Wallets: 31484:6 31481:37
421
422 Id: 60
423 Winner: 31481
424 Target: 59998749
425 Solution: 42290219
426 Votes: 2/2
427 Wallets: 31484:6 31481:38
428
429 Id: 61
430 Winner: 31481
431 Target: 42290219
432 Solution: 80163881
433 Votes: 2/2
434 Wallets: 31484:6 31481:39
435
436 Id: 62
437 Winner: 31481
438 Target: 80163881
439 Solution: 33623773
440 Votes: 2/2
441 Wallets: 31484:6 31481:40
442

```

Ejemplo de una cartera de un minero (el minero con pid: 31484)

Control de parámetros de entrada

Esta simple prueba simplemente nos asegura que los parámetros introducidos al programa son correctos y no pueden generar problemas durante la ejecución del mismo.

```

eps@vmlabsdocentes:~/Documentos/SOPITA-_DE_SOBRE/Proyecto$ ./minero -1 10
1 < <N_SECONDS> < 100 and 1 < <N_TRHEADS> < 100
eps@vmlabsdocentes:~/Documentos/SOPITA-_DE_SOBRE/Proyecto$ ./minero 10 -1
1 < <N_SECONDS> < 100 and 1 < <N_TRHEADS> < 100
eps@vmlabsdocentes:~/Documentos/SOPITA-_DE_SOBRE/Proyecto$ ./minero 10 101
1 < <N_SECONDS> < 100 and 1 < <N_TRHEADS> < 100
eps@vmlabsdocentes:~/Documentos/SOPITA-_DE_SOBRE/Proyecto$ ./minero 101 10
1 < <N_SECONDS> < 100 and 1 < <N_TRHEADS> < 100
eps@vmlabsdocentes:~/Documentos/SOPITA-_DE_SOBRE/Proyecto$ █

```

Control de errores

En esta prueba hemos forzado que uno de los procesos Minero dé error como se aprecia en las siguientes imágenes. En este caso vemos como los errores se controlan y el sistema continúa a pesar de haber perdido a un minero. Las siguientes fotos muestran algunos ejemplos de las comprobaciones realizadas.

```
if (sigprocmask(SIG_UNBLOCK, &MaskWithSignals, NULL) == ERROR || proc_index == 8)
    error_minero("Error desbloqueando las señales", NO_PARAMETER, PipeEscr, &mq, &s_info, proc_index);
```

Simulación de error en el sigprocmask solo para el octavo proceso del sistema

```
Id: 334
Winner: 250279
Target: 73720681
Solution: 48267350 (validated)
Votes: 1/1
Wallets: 250279:126

Id: 335
Winner: 250279
Target: 48267350
Solution: 24365840 (validated)
Votes: 1/1
Wallets: 250279:127
[250269] Finishing...

Id: 336
Winner: 250279
Target: 24365840
Solution: 70740591 (validated)
Votes: 1/1
Wallets: 250279:128
[250270] Finishing...

eps@vmlabsdocentes:~/Documentos/SOPITA-DE_SOBRE/Proyecto$
MAX_TRY
TIMELINE
eps@vmlabsdocentes:~/Documentos/SOPITA-DE_SOBRE/Proyecto$ make run_mins10
./minero 20 40 & ./minero 24 50 & ./minero 35 80 & ./minero 20 4 & ./minero 24 5
& ./minero 35 8 & ./minero 45 30 & ./minero 44 5 & ./minero 60 3 & ./minero 34
10
Error desbloqueando las señales: Interrupted system call
eps@vmlabsdocentes:~/Documentos/SOPITA-DE_SOBRE/Proyecto$ make
gcc -g -Wall -ggdb -c funciones_minero.c
gcc -o minero minero.o funciones_minero.o pow.o utils.o -lm -lpthread -lrt
eps@vmlabsdocentes:~/Documentos/SOPITA-DE_SOBRE/Proyecto$ make run_mins10
./minero 20 40 & ./minero 24 50 & ./minero 35 80 & ./minero 20 4 & ./minero 24 5
& ./minero 35 8 & ./minero 45 30 & ./minero 44 5 & ./minero 60 3 & ./minero 34
10
Error desbloqueando las señales: Resource temporarily unavailable
make: *** [Makefile:50: run_mins10] Error 1
eps@vmlabsdocentes:~/Documentos/SOPITA-DE_SOBRE/Proyecto$ make run_mins10
./minero 20 40 & ./minero 24 50 & ./minero 35 80 & ./minero 20 4 & ./minero 24 5
& ./minero 35 8 & ./minero 45 30 & ./minero 44 5 & ./minero 60 3 & ./minero 34
10
Error desbloqueando las señales: Interrupted system call
eps@vmlabsdocentes:~/Documentos/SOPITA-DE_SOBRE/Proyecto$
```

Ejecución controlada los otros nueve procesos continúan con su ejecución y además los recursos se liberan correctamente

```
if ([proc_index == MAX_MINERS || proc_index == 8]) /*En caso de que ya halla el máximo de mineros*/
    error_minero(NULL, "Number of miners reached the maximum", handler_info_sist, NO_PARAMETER, NULL, s_in
```

Simulación de error de máximos procesos

Ejecución controlada y comprobación de que los recursos están liberados correctamente

```
otes: 1/2
allets: 430459:108 430462:24 430463:55 430464:13

d: 271
inner: 430459
arget: 27097820
olution: 27097820 (rejected)
otes: 1/2
allets: 430459:109 430462:24 430463:55 430464:13

d: 272
inner: 430464
arget: 27097820
olution: 82331938 (validated)
otes: 2/2
allets: 430459:109 430462:24 430463:55 430464:14

d: 273
inner: 430463
arget: 82331938
olution: 4901021 (validated)
otes: 2/2
allets: 430459:109 430462:24 430463:56 430464:14

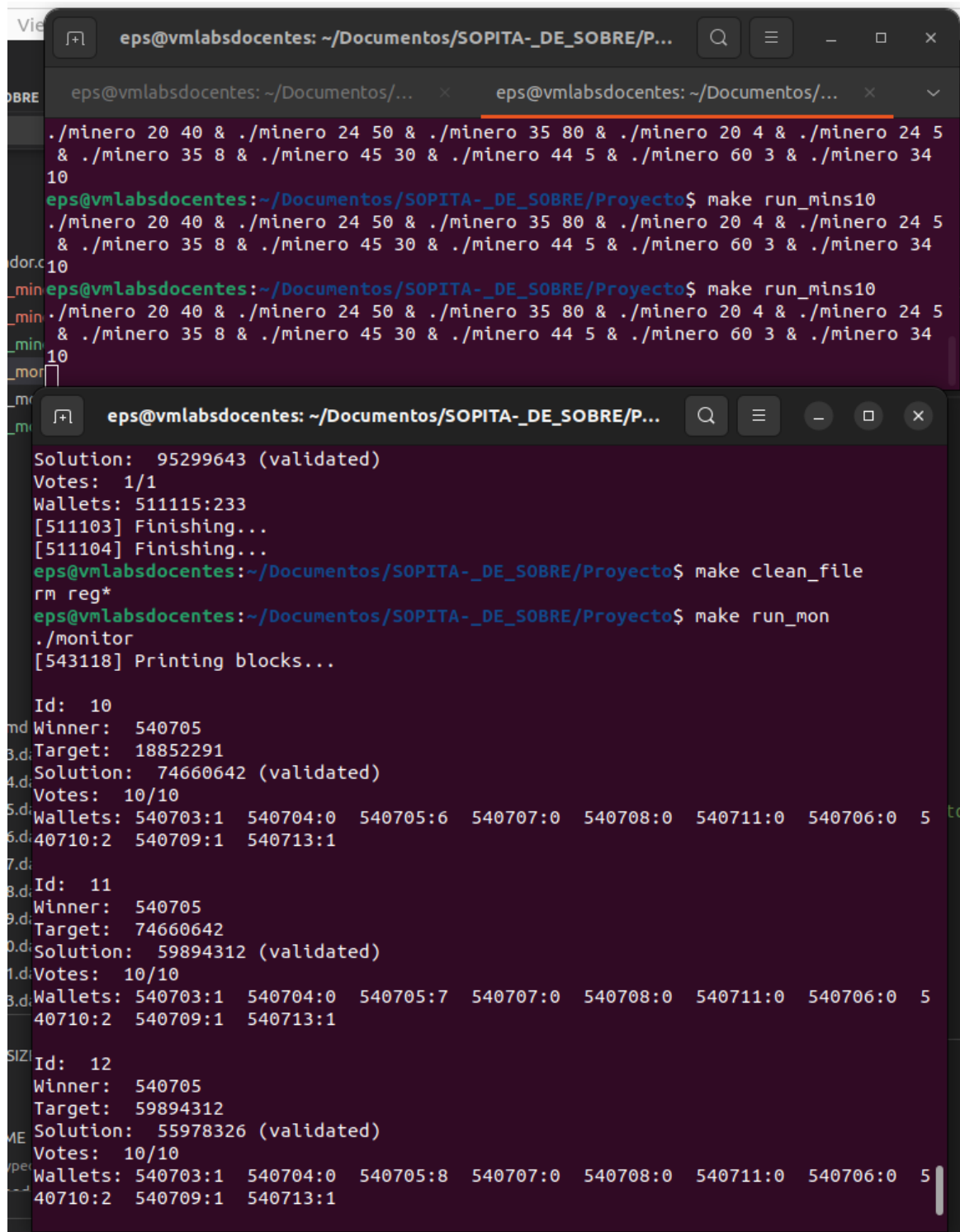
d: 274
inner: 430463
arget: 4901021
olution: 45373123 (validated)
otes: 2/2
allets: 430459:109 430462:24 430463:57 430464:14
430453] Finishing...

d: 275
inner: 430459
arget: 45373123
olution: 29264491 (validated)
otes: 4/4
allets: 430459:110 430462:24 430463:57 430464:14
430454] Finishing...

eps@vmlabsdocentes:~/Documentos/SOPITA-DE_SOBRE/Proyecto$
./minero 20 40 & ./minero 24 50 & ./minero 35 80 & ./minero 20 4 & ./minero 24 5
& ./minero 35 8 & ./minero 45 30 & ./minero 44 5 & ./minero 60 3 & ./minero 34
10
Number of miners reached the maximum: File exists
Number of miners reached the maximum: File exists
make: *** [Makefile:50: run_mins10] Error 1
eps@vmlabsdocentes:~/Documentos/SOPITA-DE_SOBRE/Proyecto$ make cs
ls /dev/shm
libpod_lock libpod_rootless_lock_1000
ls /dev/mqueue
eps@vmlabsdocentes:~/Documentos/SOPITA-DE_SOBRE/Proyecto$
```

Comprobación de que si no hay monitor no se envían mensajes:

Para realizar esta prueba lanzamos los mineros con cierto margen de tiempo y observamos que el primer bloque impreso tiene id mayor que 0 en el caso de la imagen 10. Lo que nos indica que los diez primeros bloques no han sido enviados al monitor.



```
eps@vmlabsdocentes: ~/Documentos/SOPITA-_DE_SOBRE/P...  
eps@vmlabsdocentes: ~/Documentos/... x eps@vmlabsdocentes: ~/Documentos/... x  
./minero 20 40 & ./minero 24 50 & ./minero 35 80 & ./minero 20 4 & ./minero 24 5  
& ./minero 35 8 & ./minero 45 30 & ./minero 44 5 & ./minero 60 3 & ./minero 34  
10  
eps@vmlabsdocentes:~/Documentos/SOPITA-_DE_SOBRE/Proyecto$ make run_mins10  
./minero 20 40 & ./minero 24 50 & ./minero 35 80 & ./minero 20 4 & ./minero 24 5  
& ./minero 35 8 & ./minero 45 30 & ./minero 44 5 & ./minero 60 3 & ./minero 34  
10  
eps@vmlabsdocentes:~/Documentos/SOPITA-_DE_SOBRE/Proyecto$ make run_mins10  
./minero 20 40 & ./minero 24 50 & ./minero 35 80 & ./minero 20 4 & ./minero 24 5  
& ./minero 35 8 & ./minero 45 30 & ./minero 44 5 & ./minero 60 3 & ./minero 34  
10  
Solution: 95299643 (validated)  
Votes: 1/1  
Wallets: 511115:233  
[511103] Finishing...  
[511104] Finishing...  
eps@vmlabsdocentes:~/Documentos/SOPITA-_DE_SOBRE/Proyecto$ make clean_file  
rm reg*  
eps@vmlabsdocentes:~/Documentos/SOPITA-_DE_SOBRE/Proyecto$ make run_mon  
./monitor  
[543118] Printing blocks...  
  
Id: 10  
Winner: 540705  
Target: 18852291  
Solution: 74660642 (validated)  
Votes: 10/10  
Wallets: 540703:1 540704:0 540705:6 540707:0 540708:0 540711:0 540706:0 5  
40710:2 540709:1 540713:1  
  
Id: 11  
Winner: 540705  
Target: 74660642  
Solution: 59894312 (validated)  
Votes: 10/10  
Wallets: 540703:1 540704:0 540705:7 540707:0 540708:0 540711:0 540706:0 5  
40710:2 540709:1 540713:1  
  
Id: 12  
Winner: 540705  
Target: 59894312  
Solution: 55978326 (validated)  
Votes: 10/10  
Wallets: 540703:1 540704:0 540705:8 540707:0 540708:0 540711:0 540706:0 5  
40710:2 540709:1 540713:1
```

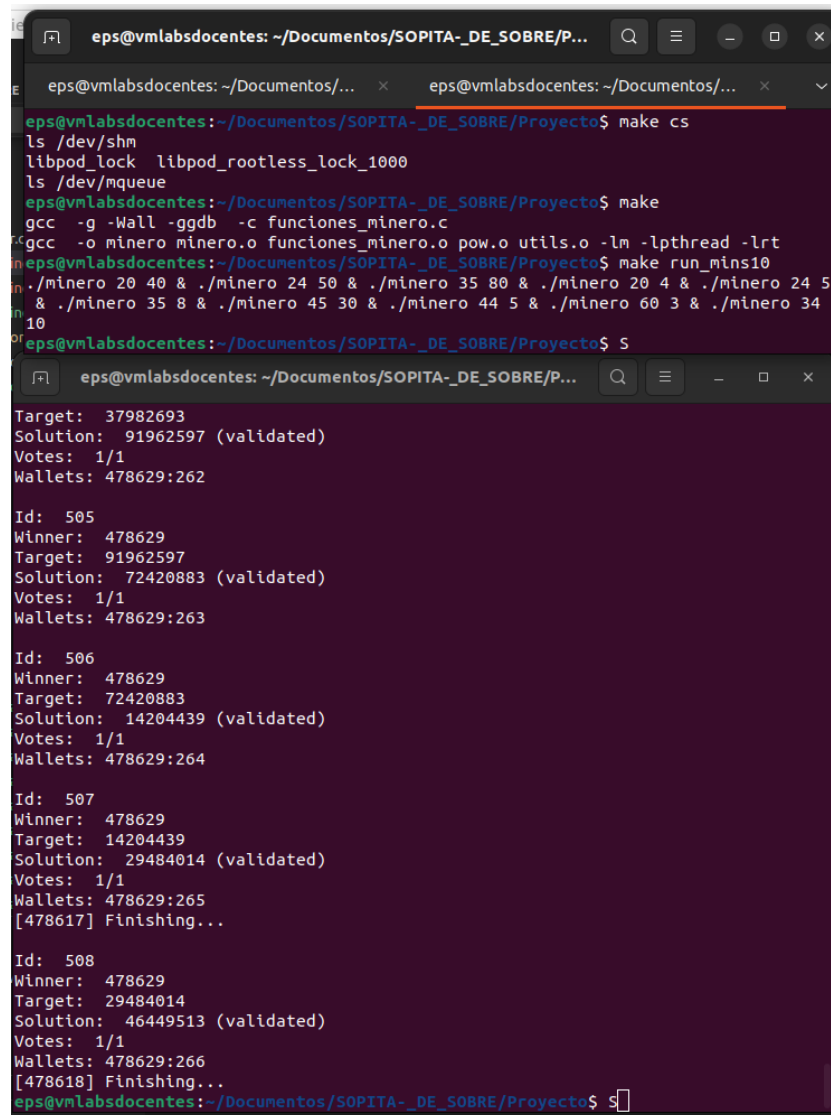
Se inicia la impresión por el bloque 10

Sleeps aleatorios

Finalmente para evitar suposiciones de tiempo añadimos una macro TEST que si está definida añade sleeps aleatorios durante la ejecución de los distintos procesos, de esta forma comprobamos que no hayamos realizado ninguna suposición de tiempos. Esto es un ejemplo de la ejecución con la opción TEST activada. Todos los tests anteriores y los que quedan han sido realizados tanto con esta opción como sin ella.

```
#ifdef TEST
    nanorandsleep();
#endif
```

```
void nanorandsleep()
{
    struct timespec time = {0, rand() % BIG_PRIME};
    nanosleep(&time, NULL);
}
```



```
eps@vmlabsdocentes: ~/Documentos/SOPITA-DE_SOBRE/P...
eps@vmlabsdocentes: ~/Documentos/SOPITA-DE_SOBRE/Proyecto$ make cs
ls /dev/shm
libpod_lock libpod_rootless_lock_1000
ls /dev/mqueue
eps@vmlabsdocentes: ~/Documentos/SOPITA-DE_SOBRE/Proyecto$ make
gcc -g -Wall -ggdb -c funciones_minero.c
gcc -o minero minero.o funciones_minero.o pow.o utils.o -lm -lpthread -lrt
eps@vmlabsdocentes: ~/Documentos/SOPITA-DE_SOBRE/Proyecto$ make run_mins10
./minero 20 40 & ./minero 24 50 & ./minero 35 80 & ./minero 20 4 & ./minero 24 5
& ./minero 35 8 & ./minero 45 30 & ./minero 44 5 & ./minero 60 3 & ./minero 34
10
eps@vmlabsdocentes: ~/Documentos/SOPITA-DE_SOBRE/Proyecto$ S

Target: 37982693
Solution: 91962597 (validated)
Votes: 1/1
Wallets: 478629:262

Id: 505
Winner: 478629
Target: 91962597
Solution: 72420883 (validated)
Votes: 1/1
Wallets: 478629:263

Id: 506
Winner: 478629
Target: 72420883
Solution: 14204439 (validated)
Votes: 1/1
Wallets: 478629:264

Id: 507
Winner: 478629
Target: 14204439
Solution: 29484014 (validated)
Votes: 1/1
Wallets: 478629:265
[478617] Finishing...

Id: 508
Winner: 478629
Target: 29484014
Solution: 46449513 (validated)
Votes: 1/1
Wallets: 478629:266
[478618] Finishing...
eps@vmlabsdocentes: ~/Documentos/SOPITA-DE_SOBRE/Proyecto$ S
```

Ejemplo de ejecución con 10 mineros

Control de señales por parte de Monitor y Minero

Esta prueba se centra en la correcta gestión de la señal SigINT por parte de ambos procesos. Por un lado observamos con la opción DEBUG activada vemos como tras introducir un SigINT a un proceso este sale del sistema de forma controlada.

```
He minado 72657
USR1 72657
USR2 72657
He minado 59894312->-1
n/c perdedor 72657
ln perdedor 726577
USR1 72657
USR2 72657
He minado 55978326->-1
n/c perdedor 72657
ln perdedor 726577
USR1 72657
USR2 72657
He minado 18633092->61031588
n/c perdedor 72657
ln perdedor 726577
USR1 72657
USR2 72657
^C ME LLEGÓ SIGINT
ME LLEGÓ Finalización
USR2 72657
He minado 61031588->-1
n/c perdedor 72657
ln perdedor 726577
Me voy:72657
REGISTRADOR SE VA porque no hay escritores
eps@vmlabsdocentes: ~/Documentos/SOPITA_DE_SOBRE/Proyecto$
```

```
Comprueba 27722120 -> 12375753: 27722120
Id: 42
Winner: 72648
Target: 27722120
Solution: 12375753 (validated)
Votes: 4/4
Wallets: 72649:23 72653:4 72652:1 72648:10
Comprueba 12375753 -> 27468809: 12375753
Id: 43
Winner: 72649
Target: 12375753
Solution: 27468809 (validated)
Votes: 4/4
Wallets: 72649:24 72653:4 72652:1 72648:10
Comprueba 27468809 -> 74299000: 27468809
Id: 44
Winner: 72648
Target: 27468809
Solution: 74299000 (validated)
Votes: 4/4
Wallets: 72649:24 72653:4 72652:1 72648:11
```

Control de SIGINT y ejecución del sistema de forma independiente

Por otro lado observamos con la opción DEBUG activada vemos como tras introducir un SigINT al monitor los mineros siguen ejecutando con normalidad y al relanzarlo se reengancha al sistema. Como se observa al cerrar el monitor nos perdemos los bloques entre el 7 y el 24, pero luego continuamos de forma normal además vemos cómo en ese tiempo han abandonado el sistema varios procesos y no ha conducido a ningún error..

```
He minado 19712626->11970152
n/c ganador 85011
n/c perdedor 85028
Ganador envia SIGUSR2 USR2 85028
Intento:0 2
erdedor vota 85028
ln perdedor 85028
Ganador envia SIGUSR1
ln ganador 85031
USR1 85028
He minado 13976152->13282181
He minado 13976152->13282181
n/c ganador 85028
n/c perdedor 85031
Ganador envia SIGUSR2 USR2 85031
Intento:0 2
ln perdedor 850311
Intento:1 2
Ganador envia SIGUSR1
ln ganador 85028
USR1 85031
ME LLEGÓ Finalización
ME LLEGÓ Finalización
He minado 13282181->34741228
n/c ganador 85028
Ganador envia SIGUSR2 USR2 85031
He minado 13282181->-1
n/c perdedor 85031
Intento:0 2
Intento:1 2
Intento:2 2
erdedor vota 85031
ln perdedor 85031
Me voy:85031
REGISTRADOR SE VA porque no hay escritores
Ganador envia SIGUSR1
ln ganador 85028
Me voy:85028
oy el ultimo:85028
REGISTRADOR SE VA porque no hay escritores
```

```
Comprueba 49574592 -> 16391022: 49574592
Id: 6
Winner: 85027
Target: 49574592
Solution: 16391022 (validated)
Votes: 9/9
Wallets: 85026:1 85028:5 85027:1 85034:0 85032:0 85031:0 85035:0 85029:0 85030:0
Comprueba 16391022 -> 41194344: 16391022
ME LLEGÓ Finalización
Id: 7
Winner: 85028
Target: 16391022
Solution: 41194344 (validated)
Votes: 9/9
Wallets: 85026:1 85028:6 85027:1 85034:0 85032:0 85031:0 85035:0 85029:0 85030:0
Comprueba 41194344 -> 61259182: 41194344
[85023] Finishing monitor...
FIN
[85022] Finishing comprobador...
eps@vmlabsdocentes: ~/Documentos/SOPITA_DE_SOBRE/Proyecto$ ./monitor
[90512] checking blocks...
Pasa mmap 90512
Semáforos inicializados 90512
[90513] Printing blocks...
Comprueba 73337952 -> 55485482: 73337952
Id: 24
Winner: 85026
Target: 73337952
Solution: 55485482 (validated)
Votes: 6/6
Wallets: 85026:4 85028:15 85027:2 85032:3 85031:0 85035:1
Comprueba 55485482 -> 77917774: 55485482
Id: 25
Winner: 85026
Target: 55485482
Solution: 77917774 (validated)
```

Control de SIGINT y ejecución de los mineros de forma independiente

Diagrama de la ejecución

Debido a que se trata de un sistema compuesto por muchos procesos, hemos dividido el esquema del proyecto en tres diagramas:

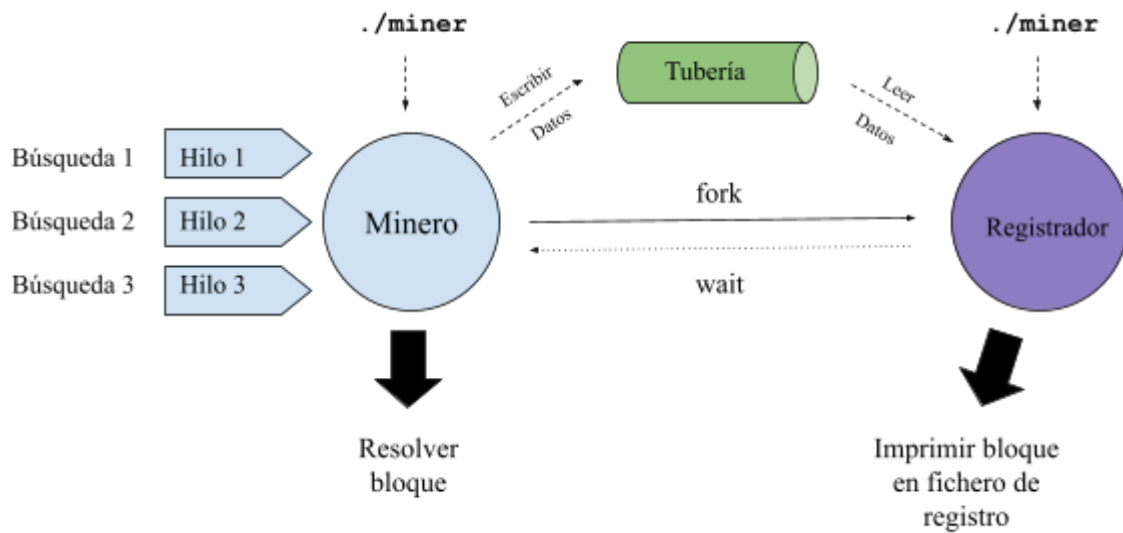


Figura 1: esquema de la comunicación entre un proceso Minero y su proceso hijo Registrador

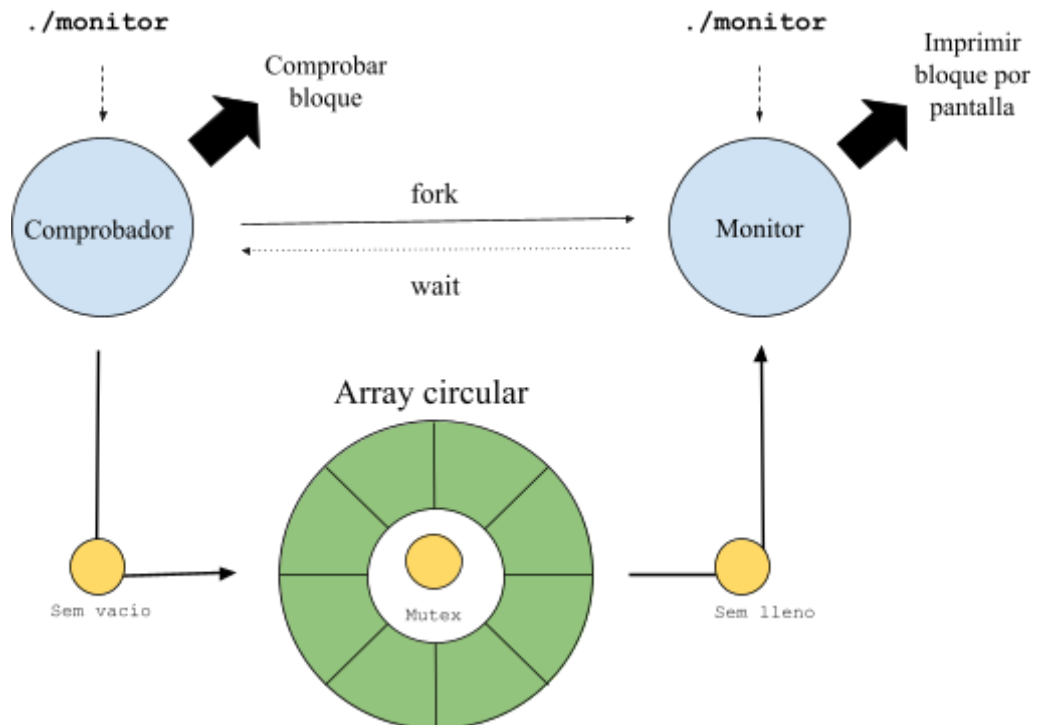


Figura 2: esquema de la comunicación entre un proceso Comprobador y su proceso hijo Monitor.

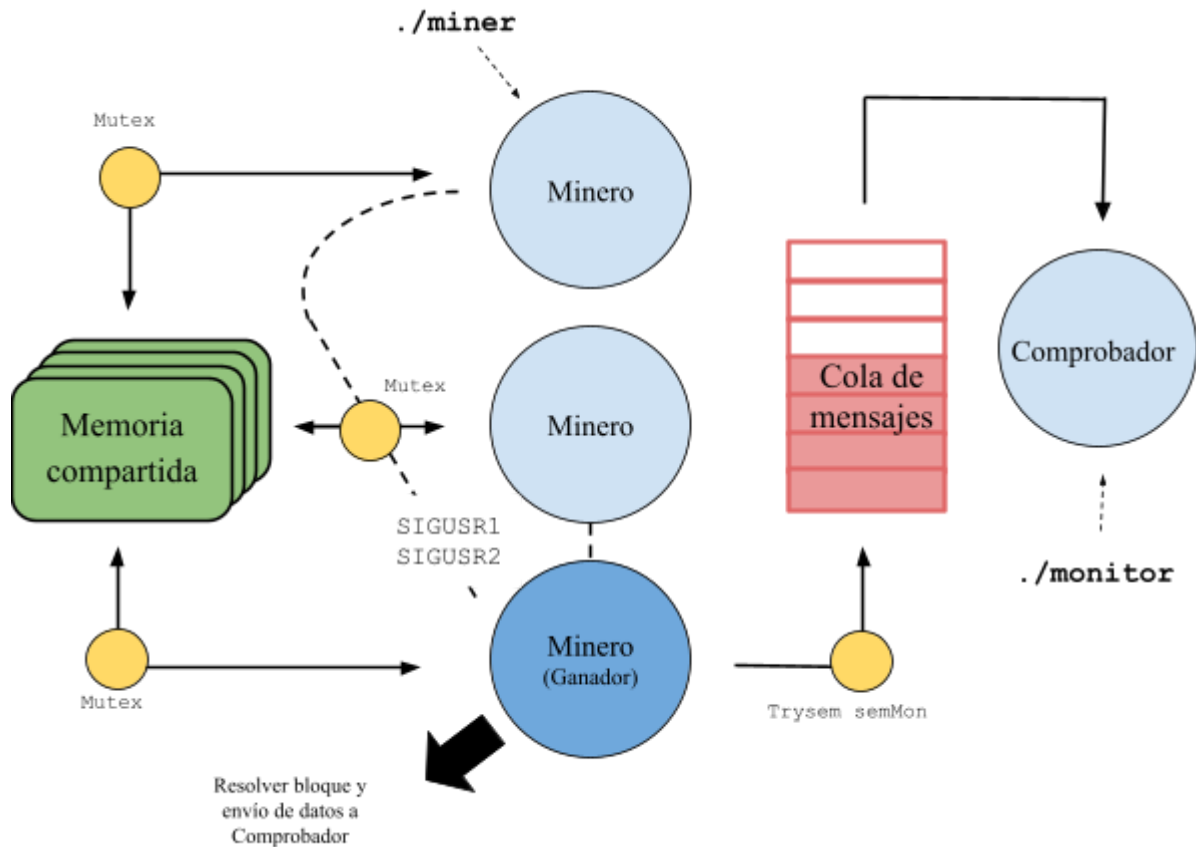


Figura 3: esquema de la comunicación entre los distintos procesos Minero y entre el ganador y el Comprobador.

Descripción del sistema:

Esta descripción se realizará explicando la relación entre los distintos procesos dentro del sistema. Este está formado por 4 tipos de procesos: Mineros, Registradores, Comprobadores y Monitores. Los Registradores y Monitores son procesos hijos de los Mineros y Comprobadores, respectivamente. En el sistema solo se permite la existencia de un Monitor de forma concurrente, por tanto de un Comprobador.

Relación Minero - Minero: (Figura 3)

Los mineros son los encargados de encontrar las preimágenes de una función hash no invertible. Estos compiten por ser los primeros en encontrar dicho resultado. Cada ronda de minado se almacena en una Wallet (Cartera) del sistema. El primero en encontrar la solución se convierte en el ganador de la ronda e inicializa una nueva cartera en un espacio de memoria compartida. En él los mineros perdedores votan en

función de si consideran que el resultado obtenido por el ganador es correcto. Para el acceso a la memoria compartida es necesario el uso de semáforos como se observa en la Figura 3. Durante cada ronda de votación es necesario sincronizar a los votantes para avisarles de que se ha convocado una votación y que ha empezado una nueva ronda de minado. Esto se realiza con el uso de las señales SIGUSR1 y SIGUSR2.

Hemos encontrado diversos problemas en la sincronización de los mineros. Por ejemplo, hemos necesitado añadir un semáforo con nombre para controlar que ningún proceso minero comenzase a usar la memoria compartida antes de que el primer minero la hubiese inicializado.

Relación Minero - Registrador: (Figura 1)

Cada minero durante cada ciclo de minado envía a su hijo, el proceso registrador, un mensaje con el último bloque votado a través de una pipe unidireccional. El Registrador es el encargado de escribir todos esos datos en un fichero *regPID.dat* que refleja lo impreso por pantalla por el Monitor.

Relación Minero - Comprobador: (Figura 3)

Si existe un proceso Comprobador, el ganador de cada ronda de minado le envía el bloque votado por una cola de mensajes. Éste evalúa si la solución es correcta y lo reenvía para impresión a su hijo el proceso Monitor. Para conocer la existencia de un monitor utilizamos un semáforo con nombre que el monitor hace un up cuando accede al sistema y un down cuando abandona indicando que sale del sistema.

Relación Monitor- Comprobador: (Figura 2)

El proceso comprobador crea proceso monitor y espera a recibir bloques por la cola de mensajes que luego reenviará al monitor a través de memoria compartida utilizando un esquema de productor-consumidor gestionado por tres semáforos sin nombre tal y como se aprecia en la figura 2.

Conclusión

En conclusión, este proyecto nos ha permitido combinar diversos conceptos y técnicas aprendidas a lo largo del curso, como la ejecución paralela mediante hilos, forks, memoria compartida y herramientas de gestión de programas paralelos, para desarrollar una red de mineros de bloques inspirada en la tecnología Blockchain. El sistema está compuesto por cuatro tipos de procesos que colaboran entre sí: Mineros, Registrador, Comprobador y Monitor. La comunicación entre estos procesos se lleva a cabo a través de diferentes mecanismos como sistemas productor-consumidor, pipelines, memoria compartida y colas de mensajes.

El proyecto garantiza la integridad y confiabilidad de las soluciones propuestas mediante la validación entre los Mineros. Esto conduce a condiciones de carrera gestionadas mediante el uso de semáforos. Además, se ha implementado un manejo de errores y liberación de recursos eficiente, asegurando la estabilidad y la capacidad de recuperación del sistema en caso de fallos.

En definitiva, este proyecto intenta que los estudiantes demuestren que han adquirido habilidad para aplicar y adaptar el conocimiento estudiado en el curso de una manera práctica y efectiva, creando un software de minería robusto y escalable que aprovecha las ventajas de la tecnología Blockchain y la programación paralela.