

TECNOLÓGICO DE COSTA RICA

Escuela de Computación

Ingeniería en Computación - Principios de Sistemas Operativos (IC-6600)

Proyecto #1

Procesos colaborativos en C: Búsqueda

Estudiantes:

Sebastián Bermúdez Acuña - 2021110666

Felipe de Jesús Obando Arrieta - 2021035489

Profesor: MSc. Armando Arce Orozco

24 de septiembre del 2023

II Semestre, 2023

Índice

1.	Introducción	3
2.	Descripción del problema	4
3.	Definición de estructuras de datos	5
4.	Descripción y explicación de los componentes del programa	6
4.1.	Mecanismo de creación de procesos	6
4.2.	Comunicación de procesos.....	6
4.3.	Sincronización de procesos	7
4.4.	Lectura de archivo.....	9
4.5.	Búsqueda de coincidencias.....	10
5.	Pruebas de rendimiento	12
5.1.	Prueba I	12
5.2.	Prueba II	14
5.3.	Prueba III.....	16
6.	Conclusiones	19

1. Introducción

El presente proyecto tiene como objetivo principal evaluar el rendimiento de la realización de tareas colaborativas mediante el uso de múltiples procesos en un entorno Unix. Se aborda la creación de una versión multiprocesos de la utilidad *grep*, que permite buscar múltiples palabras o patrones en un archivo y mostrar las líneas de texto en las que aparecen dichos patrones. Este proyecto se enfoca en encontrar la cantidad óptima de procesos para ejecutar esta tarea en un archivo de gran tamaño.

El programa *grep* se utiliza para buscar patrones definidos por expresiones regulares en uno o varios archivos. Para el manejo de expresiones regulares, se emplea la biblioteca "regex.h", que proporciona las funciones *regcomp* y *regexexec*. Además, se implementa un conjunto compartido de procesos (pool de procesos) para inspeccionar el archivo de manera eficiente. Todos los procesos se crean al inicio del programa y se asignan tareas específicas sobre porciones del archivo. La sincronización entre procesos se logra mediante el uso exclusivo de cola de mensajes.

Este proyecto se desarrolla en lenguaje C. La documentación que acompaña a este proyecto proporcionará una descripción detallada del problema, definición de estructuras de datos, descripción de los componentes principales del programa, mecanismo de creación y administración de procesos, pruebas de rendimiento y conclusiones del proyecto.

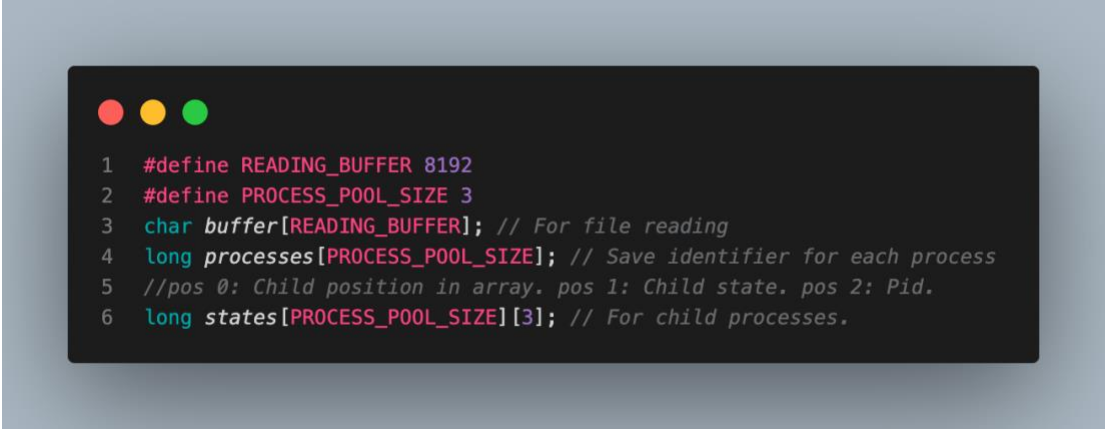
2. Descripción del problema

El proyecto tiene como objetivo principal evaluar el rendimiento de la ejecución colaborativa de tareas mediante múltiples procesos en un entorno Unix. Para ello, se propone desarrollar una versión multiprocesos del programa *grep*, que es una herramienta ampliamente utilizada para buscar patrones en archivos. La tarea consiste en buscar múltiples palabras o patrones en un archivo de gran tamaño y mostrar la línea de texto en la que aparecen dichos patrones.

Dado que las búsquedas se llevarán a cabo en archivos extensos, se plantea la necesidad de utilizar múltiples procesos para aumentar la eficiencia. Se creará un conjunto compartido de procesos (pool de procesos) limitado en cantidad, donde cada proceso trabajará en un buffer propio que contendrá una porción del archivo. El tamaño óptimo del pool de procesos se determinará mediante pruebas de rendimiento que utilizan el tiempo de ejecución como métrica principal. Este proyecto se desarrolla en lenguaje C y se adhiere a restricciones específicas, como el uso exclusivo de la biblioteca "regex.h" para la manipulación de expresiones regulares, y se enfatiza el uso de procesos hijos en lugar de hilos de ejecución.

3. Definición de estructuras de datos

Para el desarrollo de la solución se utilizó una estructura de mensaje que se utiliza para comunicar los procesos, un buffer para almacenar los datos leídos del archivo, así como un arreglo para guardar un identificador de proceso, además, otro array con la información del proceso como la ubicación del hijo en el array, estado del hijo y el PID para que el padre pueda terminar los procesos hijos.

A screenshot of a code editor with a dark background and light-colored text. The code is in C and defines constants and arrays for process management. It includes a reading buffer, a process pool size, and an array of states for child processes. The code is numbered from 1 to 6.

```
1 #define READING_BUFFER 8192
2 #define PROCESS_POOL_SIZE 3
3 char buffer[READING_BUFFER]; // For file reading
4 long processes[PROCESS_POOL_SIZE]; // Save identifier for each process
5 //pos 0: Child position in array. pos 1: Child state. pos 2: Pid.
6 long states[PROCESS_POOL_SIZE][3]; // For child processes.
```

Figura #1: Definición de estructuras de datos

4. Descripción y explicación de los componentes del programa

El programa está compuesto por múltiples componentes, a continuación encontrará un apartado por cada uno de estos, con su respectiva descripción y explicación detallada de su funcionamiento.

4.1. Mecanismo de creación de procesos

El mecanismo de creación de procesos consiste en un ciclo en donde el padre crea un hijo por iteración, al crear el proceso, si se trata del hijo, este se debe salir del bucle, si es el padre debe continuar con la ejecución del ciclo para crear los hijos restantes.



```
1  int createProcesses() {
2      pid_t pid;
3      for (int i = 0; i < PROCESS_POOL_SIZE; i++) {
4          pid = fork();
5          if (pid != 0) {
6              processes[i] = (long)i + 1;
7              states[i][0]=i; //Child position in array.
8              states[i][1]=0; //States -> 0: Available, 1: Reading, 2: Processing
9              states[i][2]=pid; //Child pid
10         }
11         else
12             return i + 1;
13     }
14     return 0;
15 }
```

Figura #2: Mecanismo de creación de procesos.

4.2. Comunicación de procesos

Para comunicar los procesos se utiliza una cola de mensajes, tanto el proceso padre como los procesos hijos envían información a través de esta.

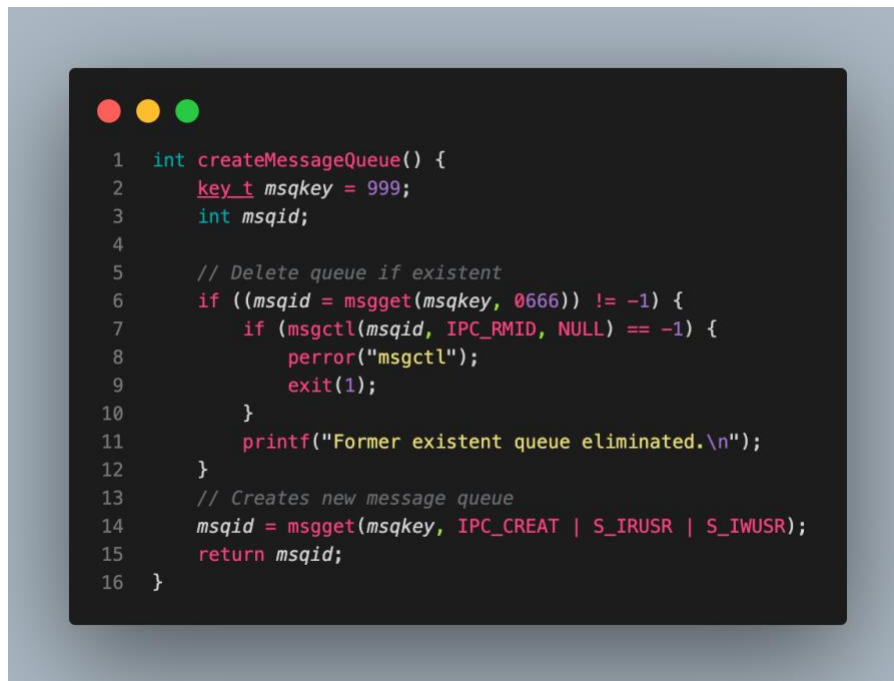


Figura #3: Comunicación de procesos.

4.3. Sincronización de procesos

Después de crear los procesos, los hijos se mantienen a la espera de su turno de lectura. Cuando el hijo recibe un mensaje es porque el padre le está indicando que continúe con la lectura del archivo, en ese mensaje, el padre le indica la posición en donde el hijo deberá continuar con la lectura. Después de leer debe indicarle al padre que concluyó la lectura, notificando la última posición que leyó (-1 si no hay nada más que leer). Además, debe notificarle la posición del arreglo en donde se encuentra, esto para que el padre pueda estar al pendiente del estado de los hijos.

```

1 // Only children loop here
2 long childArrayPosition;
3 while (childID != 0) {
4     // Waits for the father message to start reading
5     msgrcv(msqid, &msg, sizeof(msg.matchesFound), childID, 0); // receives, message type = their id
6
7     childArrayPosition=msg.arrayPosition;
8     msg.type = childID;
9     // Starts reading
10    msg.filePosition = readFile(childID, msg.filePosition, fileName); // Saves last jump position
11    msg.arrayPosition=childArrayPosition;
12
13    msgsnd(msqid, (void *)&msg, sizeof(msg.matchesFound), 0); // Sends message to father. Finished reading
14    searchPattern(msqid, childID, regex, childArrayPosition); // Process data. Looks for regex match.
15 }

```

Figura #4: Sincronización de procesos. Procesos hijos.

Por otra parte, el padre envía un mensaje al primer hijo creado para que empiece la lectura del archivo. Después, entra a un ciclo en donde lo primero que realiza es esperar a que el proceso que se encuentra leyendo envíe el mensaje notificando la terminación de lectura, en este mensaje, el proceso hijo indica la última posición que leyó del archivo para que los siguientes procesos continúen la lectura a partir de ese punto (-1 significa que no hay más que información para leer en el archivo, entonces el padre debe terminar el ciclo). Además, debe indicarle al padre la posición del arreglo, pues, el padre debe actualizar el estado de este hijo. Después de recibir el mensaje por parte del hijo, el padre debe elegir el proceso que continúa con la lectura y debe enviarle el mensaje para que este sepa que es su turno de leer. Posterior a esto, el padre debe revisar si hay mensajes con el type 100, este type es utilizado para notificar al padre que se han encontrado una o varias coincidencias de la expresión regular en el bloque de 8000 (8K) bytes leído, así, el padre empieza a imprimir las coincidencias.


```
1 // Only father enters here.
2 int childCounter = 0;
3 int activeChild = (long)processes[childCounter];
4
5 msg.type = (long)processes[childCounter];
6 msg.arrayPosition = states[0][0]; // To know which children sends the message (array position of states[i][0])
7 msg.filePosition = 0;
8 msgsnd(msqid, (void *)&msg, sizeof(msg.matchesFound), IPC_NOWAIT); // Tells first child to start reading.
9 states[0][1]=1; // Sets first child state in 1: Reading.
10 const char delimiter[] = "|"; // To separate coincidences sent by child
11 char * token ; // To print the coincidences of the regular expression.
12 long communicatedChildPosition = 0; // Position in "states" array
13 int coincidencesFound = 1;
```

Figura #5: Sincronización de procesos. Proceso Padre. Parte 1.

```
1 while(1){
2     msgrcv(msqid, &msg, sizeof(msg.matchesFound), activeChild,0); // Waits for a child to end reading
3     communicatedChildPosition = msg.arrayPosition; // Saves the child that just read
4
5     if (msg.filePosition == -1) // Nothing else to read. End of file.
6         break;
7
8     // Change the state of the child that just read
9     states[communicatedChildPosition][1]=2; // Sets child state 2: Processing
10
11     childCounter = (childCounter + 1 >= PROCESS_POOL_SIZE) ? 0 : childCounter + 1; // Resets child processes pool.
12     activeChild = processes[childCounter];
13     msg.type = processes[childCounter]; // Set next children.
14     msg.arrayPosition = childCounter;
15     msgsnd(msqid, (void *)&msg, sizeof(msg.matchesFound), 0); // Sends next process to read
16     states[childCounter][1]=1; // Sets the state of the child that just sent to read in 1: Reading.
17
18     // Print coincidences from children that finished.
19     msgrcv(msqid, &msg, sizeof(msg.matchesFound), 100,0);
20     communicatedChildPosition=msg.arrayPosition;
21     states[communicatedChildPosition][1]=0; // Sets child state in 0: Available.
22     token = strtok(msg.matchesFound, delimiter);
23     while (token != NULL) {
24         printf("Coincidence #d: %s\n", coincidencesFound, token);
25         token = strtok(NULL, delimiter);
26         coincidencesFound++;
27     }
28 }
```

Figura #6: Sincronización de procesos. Proceso Padre. Parte 2.

4.4. Lectura de archivo

Para la lectura del archivo se utiliza una función que recibe el identificador del proceso que se utiliza para imprimir cuando un proceso no tiene nada más que leer, la última

posición que se leyó del archivo para saber en donde debe continuar la lectura, y el nombre del archivo que debe leer.

La lectura del archivo consiste en leer en bloques de 8K, una vez se lee el buffer, se busca cual es el último salto de línea en el buffer, esto se debe a que si no se tiene en cuenta que el buffer puede cortar las líneas, los procesos pueden repetir coincidencias, entonces, mientras se busca ese último salto de línea, se debe ir borrando los caracteres que están después del salto, pues, estos van a ser leídos por el siguiente proceso que lea el archivo. Una vez se encuentra el último salto, se retorna para que el siguiente proceso continúe la lectura a partir de ese punto.

Si la última posición es -1 significa que no hay nada más que leer, entonces debe terminar.

A screenshot of a code editor showing a C function named `readFile`. The function takes three arguments: `processID` (type `pid_t`), `lastPosition` (type `long`), and `fileName` (type `char *`). It implements logic to read a file in 8KB blocks, find the last newline character in the buffer, and return the position for the next read. If no more data is available (`lastPosition == -1`), it sleeps for 1 second and exits. The code includes error handling for file opening and seeking. The buffer is of type `char` and the reading buffer size is defined as `READING_BUFFER`. The function returns `lastJumpPosition`.

```
1 int readFile(pid_t processID, long lastPosition, char *fileName) {
2     if (lastPosition == -1) { // Nothing more to read.
3         sleep(1);
4         exit(1);
5     }
6
7     FILE *file = fopen(fileName, "rb"); // Open file
8     if (file == NULL) {
9         perror("Error while opening file.");
10        return 1;
11    }
12
13    long lastJumpPosition = -1; //Default -1 if end of file is reached
14
15    // Places reading pointer where the last process ended reading
16    if (fseek(file, lastPosition == 0 ? 0 : lastPosition + 1, SEEK_SET) != 0) {
17        perror("Error while establishing reading position.");
18        fclose(file);
19        return 1;
20    }
21    // Reads 8K from the position given by fseek
22    long bytesRead = fread(buffer, 1, READING_BUFFER, file);
23    if (bytesRead > 0)
24        // From back to front
25        for (long i = bytesRead - 1; i >= 0; i--) {
26            if (buffer[i] == '\n') {
27                lastJumpPosition = ftell(file) - (bytesRead - i);
28                break;
29            } else {
30                // Removes characters after the last jump ('\n')
31                buffer[i] = '\0';
32            }
33        }
34    fclose(file);
35    return lastJumpPosition;
36 }
```

Figura #7: Lectura del archivo.

4.5. Búsqueda de coincidencias

Para la búsqueda de coincidencias de la expresión regular en el archivo, se utiliza el buffer con la información leída en la función de lectura del archivo.

En esta función se divide el buffer en líneas, las líneas se obtiene buscando los saltos de línea en el buffer. Una vez se tiene la línea, se busca la expresión regular en esta, si se encuentra, se guarda en un arreglo de coincidencias (separadas por un | para enviar todas las coincidencias de ese bloque como una sola cadena de caracteres). Al finalizar la búsqueda se envía el arreglo de coincidencias al padre para su posterior impresión.

```

1 void searchPattern(int msqid, pid_t processID, const char *regexStr, long arrayPosition) {
2     strcpy(msg.matchesFound, "");
3     char * ptr = buffer;
4     regex_t regex;
5     if (regcomp(&regex, regexStr, REG_EXTENDED) != 0) {
6         fprintf(stderr, "Error processing regular expression.\n");
7         exit(1);
8     }
9
10    char coincidences[8000] = ""; // To save coincidences found at buffer
11    while (ptr < buffer + READING_BUFFER) {
12        char *newline = strchr(ptr, '\n');
13        if (newline != NULL) {
14            size_t lineLength = newline - ptr;
15            char line[lineLength + 1];
16            strncpy(line, ptr, lineLength);
17            line[lineLength] = '\0';
18            if (regexexec(&regex, line, 0, NULL, 0) == 0) {
19                // Save in array to send it to father later
20                strcat(line, "|"); // Delimiter
21                strcat(coincidences, line);
22            }
23            ptr = newline + 1;
24        } else {
25            char line[READING_BUFFER];
26            strcpy(line, ptr);
27            if (regexexec(&regex, line, 0, NULL, 0) == 0)
28                strcat(line, "|");
29            strcat(coincidences, line);
30            break;
31        }
32    }
33    regfree(&regex);
34    memset(buffer, 0, READING_BUFFER); // Free memory
35
36    // Send coincidences to father
37    msg.type=100;
38    msg.arrayPosition=arrayPosition;
39    coincidences[sizeof(coincidences) - 1] = '\0';
40    // Copies content in coincidences at msg.matchesFound
41    strcpy(msg.matchesFound, coincidences);
42    msgsnd(msqid, (void *)&msg, sizeof(msg.matchesFound), 0);
43    strcpy(buffer, ""); // Clean buffer
44 }

```

Figura #8: Búsqueda de coincidencias

5. Pruebas de rendimiento

Para formalizar un criterio acerca de la cantidad óptima de procesos para realizar la búsqueda de los patrones dentro del texto, se realizaron tres tipos de pruebas distintos. Dos haciendo la búsqueda en el libro de “Don Quijote”, y otra utilizando el texto del libro de “La Divina Comedia”. Cada prueba se enfocará en realizar la ejecución de la misma, utilizando cantidades de procesos diferentes, por lo que cada prueba tendrá una ejecución para cada cantidad de procesos elegidos.

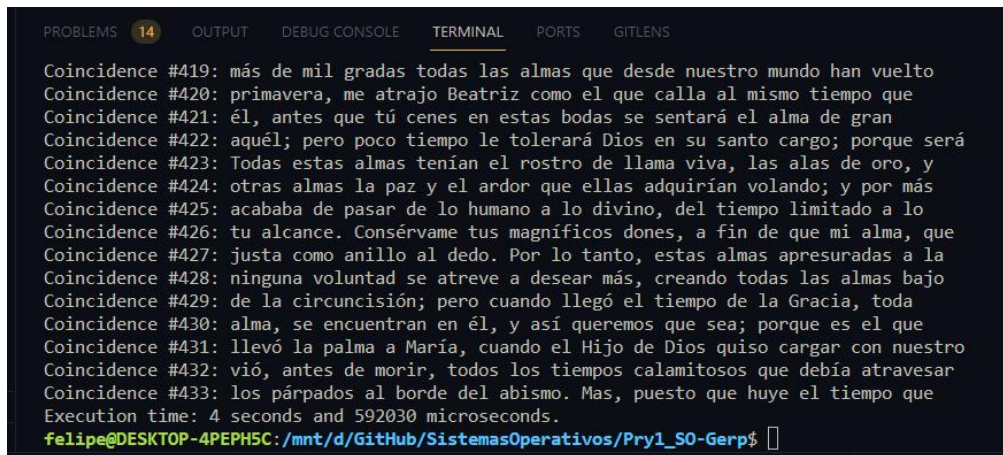
Las pruebas se realizaron en un sistema Linux con distribución Unix. Asimismo, el compilador de C utilizado fue GCC. Después de haber sido compilado, el programa funciona al escribir `./run "NombreArchivo" "(expresion a buscar | expresión a buscar)"`, siendo “run” el nombre del archivo de salida dado al compilador, y “expresión a buscar” cualquier palabra, frase, o expresión regular a encontrar dentro del archivo. De igual forma, se aceptan más parámetros de texto a buscar, mientras estos sean separados por el carácter “|”.

5.1. Prueba I

Se buscan las palabras “tiempo” y “alma” dentro del libro de La Divina Comedia.

5.1.1. 5 procesos

4.592 segundos.

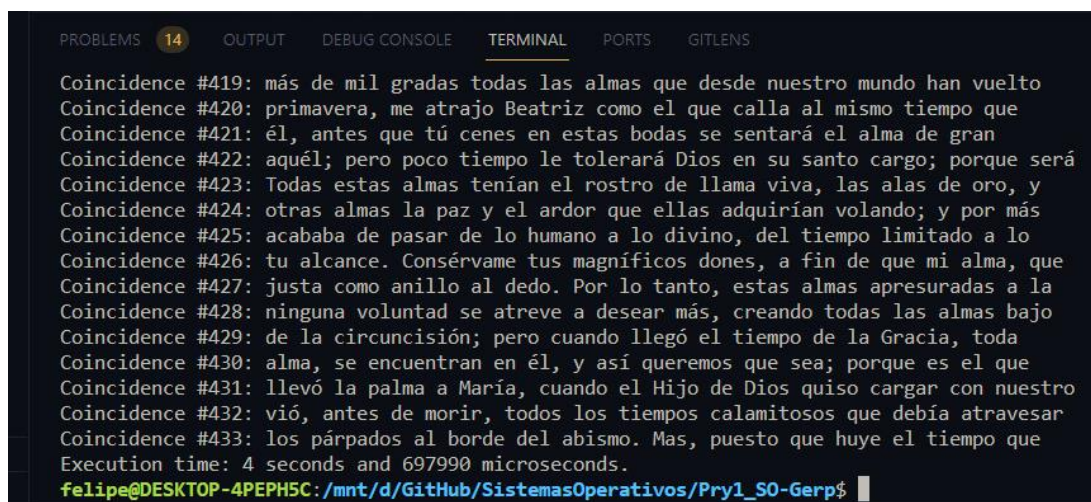


```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Coincidence #419: más de mil gradas todas las almas que desde nuestro mundo han vuelto
Coincidence #420: primavera, me atrajo Beatriz como el que calla al mismo tiempo que
Coincidence #421: él, antes que tú cenes en estas bodas se sentará el alma de gran
Coincidence #422: aquél; pero poco tiempo le tolerará Dios en su santo cargo; porque será
Coincidence #423: Todas estas almas tenían el rostro de llama viva, las alas de oro, y
Coincidence #424: otras almas la paz y el ardor que ellas adquirirían volando; y por más
Coincidence #425: acababa de pasar de lo humano a lo divino, del tiempo limitado a lo
Coincidence #426: tu alcance. Consérvame tus magníficos dones, a fin de que mi alma, que
Coincidence #427: justa como anillo al dedo. Por lo tanto, estas almas apresuradas a la
Coincidence #428: ninguna voluntad se atreve a desear más, creando todas las almas bajo
Coincidence #429: de la circuncisión; pero cuando llegó el tiempo de la Gracia, toda
Coincidence #430: alma, se encuentran en él, y así queremos que sea; porque es el que
Coincidence #431: llevó la palma a María, cuando el Hijo de Dios quiso cargar con nuestro
Coincidence #432: vió, antes de morir, todos los tiempos calamitosos que debía atravesar
Coincidence #433: los párpados al borde del abismo. Mas, puesto que huye el tiempo que
Execution time: 4 seconds and 592030 microseconds.
felipe@DESKTOP-4PEPH5C: /mnt/d/GitHub/SistemasOperativos/Pry1_S0-Gerp$
```

Figura 9: Resultado de la Prueba I, con 5 procesos.

5.1.2. 10 procesos

4.697 segundos.

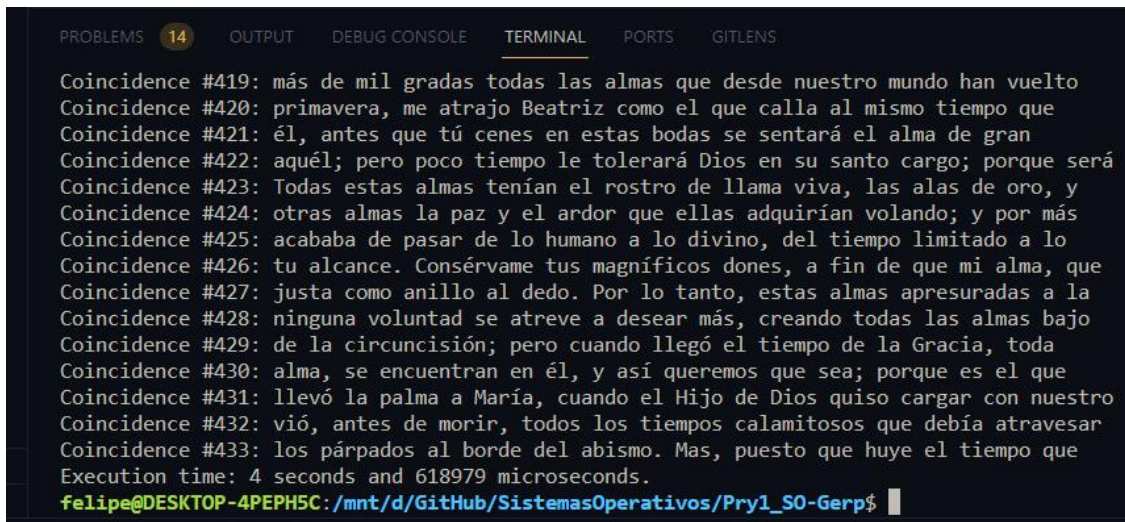


```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Coincidence #419: más de mil gradas todas las almas que desde nuestro mundo han vuelto
Coincidence #420: primavera, me atrajo Beatriz como el que calla al mismo tiempo que
Coincidence #421: él, antes que tú cenes en estas bodas se sentará el alma de gran
Coincidence #422: aquél; pero poco tiempo le tolerará Dios en su santo cargo; porque será
Coincidence #423: Todas estas almas tenían el rostro de llama viva, las alas de oro, y
Coincidence #424: otras almas la paz y el ardor que ellas adquirirían volando; y por más
Coincidence #425: acababa de pasar de lo humano a lo divino, del tiempo limitado a lo
Coincidence #426: tu alcance. Consérvame tus magníficos dones, a fin de que mi alma, que
Coincidence #427: justa como anillo al dedo. Por lo tanto, estas almas apresuradas a la
Coincidence #428: ninguna voluntad se atreve a desear más, creando todas las almas bajo
Coincidence #429: de la circuncisión; pero cuando llegó el tiempo de la Gracia, toda
Coincidence #430: alma, se encuentran en él, y así queremos que sea; porque es el que
Coincidence #431: llevó la palma a María, cuando el Hijo de Dios quiso cargar con nuestro
Coincidence #432: vió, antes de morir, todos los tiempos calamitosos que debía atravesar
Coincidence #433: los párpados al borde del abismo. Mas, puesto que huye el tiempo que
Execution time: 4 seconds and 697990 microseconds.
felipe@DESKTOP-4PEPH5C: /mnt/d/GitHub/SistemasOperativos/Pry1_S0-Gerp$
```

Figura 10: Resultado de la Prueba I, con 10 procesos.

5.1.3. 15 procesos

4.618 segundos.



```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Coincidence #419: más de mil gradas todas las almas que desde nuestro mundo han vuelto
Coincidence #420: primavera, me atrajo Beatriz como el que calla al mismo tiempo que
Coincidence #421: él, antes que tú cenes en estas bodas se sentará el alma de gran
Coincidence #422: aquél; pero poco tiempo le tolerará Dios en su santo cargo; porque será
Coincidence #423: Todas estas almas tenían el rostro de llama viva, las alas de oro, y
Coincidence #424: otras almas la paz y el ardor que ellas adquirirían volando; y por más
Coincidence #425: acababa de pasar de lo humano a lo divino, del tiempo limitado a lo
Coincidence #426: tu alcance. Consérvame tus magníficos dones, a fin de que mi alma, que
Coincidence #427: justa como anillo al dedo. Por lo tanto, estas almas apresuradas a la
Coincidence #428: ninguna voluntad se atreve a desear más, creando todas las almas bajo
Coincidence #429: de la circuncisión; pero cuando llegó el tiempo de la Gracia, toda
Coincidence #430: alma, se encuentran en él, y así queremos que sea; porque es el que
Coincidence #431: llevó la palma a María, cuando el Hijo de Dios quiso cargar con nuestro
Coincidence #432: vió, antes de morir, todos los tiempos calamitosos que debía atravesar
Coincidence #433: los párpados al borde del abismo. Mas, puesto que huye el tiempo que
Execution time: 4 seconds and 618979 microseconds.
felipe@DESKTOP-4PEPH5C:/mnt/d/GitHub/SistemasOperativos/Pry1_S0-Gerp$
```

Figura 11: Resultado de la Prueba I, con 15 procesos.

5.1.4. Resultados

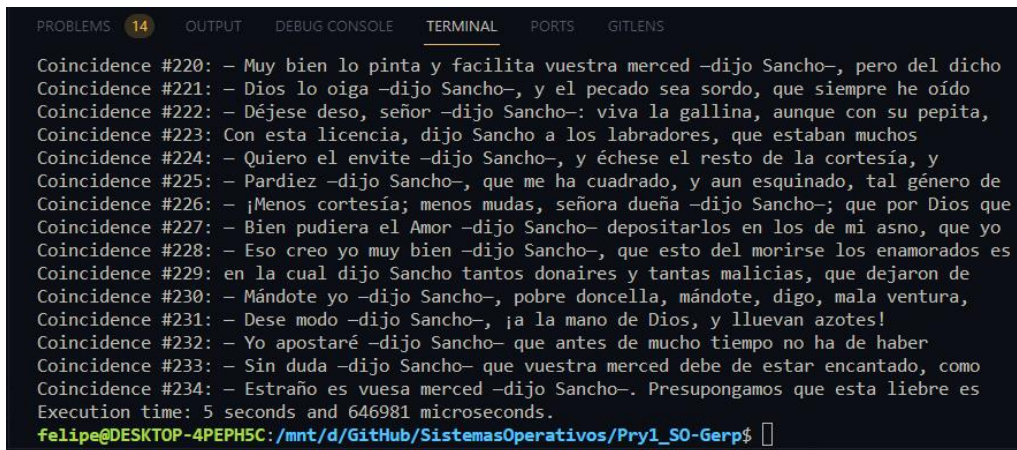
El tiempo de ejecución de las pruebas sugiere una cantidad de procesos óptima cercana a 5. Cabe destacar como con 10 procesos el tiempo de ejecución es mayor al de la ejecución con 5 procesos, sin embargo, al utilizar 15 de estos, el tiempo vuelve a ser menor a la segunda ejecución.

5.2. Prueba II

Se busca la frase “dijo Sancho” dentro del libro de Don Quijote.

5.2.1. 5 procesos

5.646 segundos.

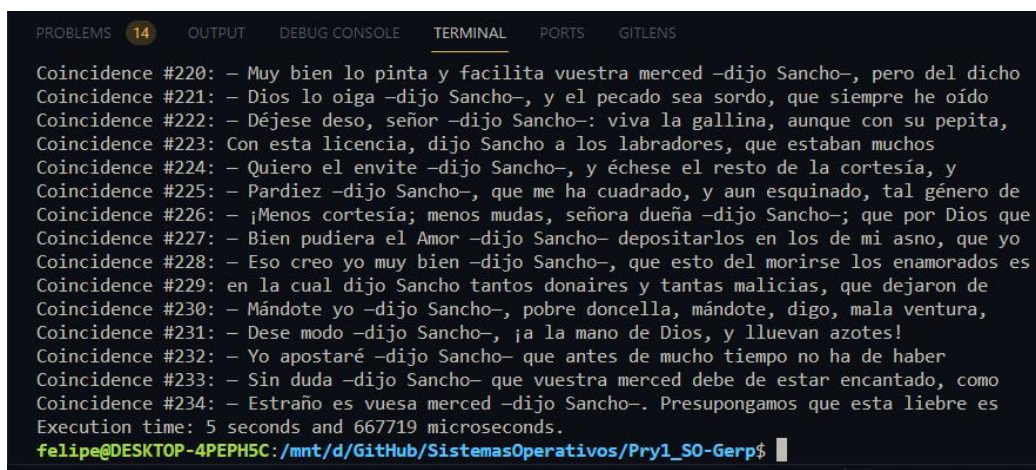


```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Coincidence #220: - Muy bien lo pinta y facilita vuestra merced -dijo Sancho-, pero del dicho
Coincidence #221: - Dios lo oiga -dijo Sancho-, y el pecado sea sordo, que siempre he oído
Coincidence #222: - Déjese deso, señor -dijo Sancho-: viva la gallina, aunque con su pepita,
Coincidence #223: Con esta licencia, dijo Sancho a los labradores, que estaban muchos
Coincidence #224: - Quiero el envite -dijo Sancho-, y échese el resto de la cortesía, y
Coincidence #225: - Pardiez -dijo Sancho-, que me ha cuadrado, y aun esquinado, tal género de
Coincidence #226: - ¡Menos cortesía; menos mudas, señora dueña -dijo Sancho-; que por Dios que
Coincidence #227: - Bien pudiera el Amor -dijo Sancho- depositarlos en los de mi asno, que yo
Coincidence #228: - Eso creo yo muy bien -dijo Sancho-, que esto del morirse los enamorados es
Coincidence #229: en la cual dijo Sancho tantos donaires y tantas malicias, que dejaron de
Coincidence #230: - Mándote yo -dijo Sancho-, pobre doncella, mándote, digo, mala ventura,
Coincidence #231: - Dese modo -dijo Sancho-, ¡a la mano de Dios, y lluevan azotes!
Coincidence #232: - Yo apostaré -dijo Sancho- que antes de mucho tiempo no ha de haber
Coincidence #233: - Sin duda -dijo Sancho- que vuestra merced debe de estar encantado, como
Coincidence #234: - Estraño es vuesa merced -dijo Sancho-. Presupongamos que esta liebre es
Execution time: 5 seconds and 646981 microseconds.
felipe@DESKTOP-4PEPH5C:/mnt/d/GitHub/SistemasOperativos/Pry1_S0-Gerp$
```

Figura 12: Resultado de la Prueba II, con 5 procesos.

5.2.2. 10 procesos

5.667 segundos.

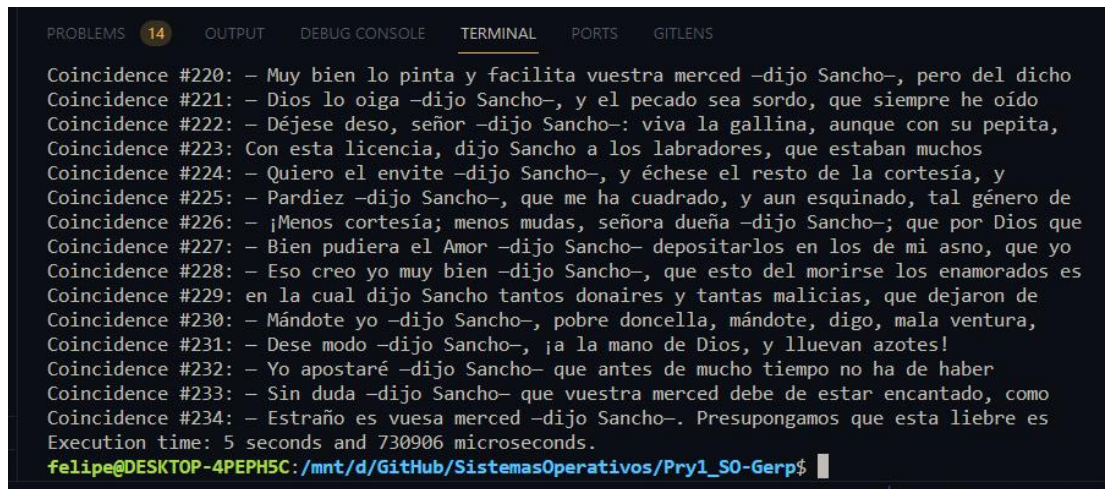


```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Coincidence #220: - Muy bien lo pinta y facilita vuestra merced -dijo Sancho-, pero del dicho
Coincidence #221: - Dios lo oiga -dijo Sancho-, y el pecado sea sordo, que siempre he oído
Coincidence #222: - Déjese deso, señor -dijo Sancho-: viva la gallina, aunque con su pepita,
Coincidence #223: Con esta licencia, dijo Sancho a los labradores, que estaban muchos
Coincidence #224: - Quiero el envite -dijo Sancho-, y échese el resto de la cortesía, y
Coincidence #225: - Pardiez -dijo Sancho-, que me ha cuadrado, y aun esquinado, tal género de
Coincidence #226: - ¡Menos cortesía; menos mudas, señora dueña -dijo Sancho-; que por Dios que
Coincidence #227: - Bien pudiera el Amor -dijo Sancho- depositarlos en los de mi asno, que yo
Coincidence #228: - Eso creo yo muy bien -dijo Sancho-, que esto del morirse los enamorados es
Coincidence #229: en la cual dijo Sancho tantos donaires y tantas malicias, que dejaron de
Coincidence #230: - Mándote yo -dijo Sancho-, pobre doncella, mándote, digo, mala ventura,
Coincidence #231: - Dese modo -dijo Sancho-, ¡a la mano de Dios, y lluevan azotes!
Coincidence #232: - Yo apostaré -dijo Sancho- que antes de mucho tiempo no ha de haber
Coincidence #233: - Sin duda -dijo Sancho- que vuestra merced debe de estar encantado, como
Coincidence #234: - Estraño es vuesa merced -dijo Sancho-. Presupongamos que esta liebre es
Execution time: 5 seconds and 667719 microseconds.
felipe@DESKTOP-4PEPH5C:/mnt/d/GitHub/SistemasOperativos/Pry1_S0-Gerp$
```

Figura 13: Resultado de la Prueba II, con 10 procesos.

5.2.3. 15 procesos

5.7309 segundos.



```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Coincidence #220: - Muy bien lo pinta y facilita vuestra merced -dijo Sancho-, pero del dicho
Coincidence #221: - Dios lo oiga -dijo Sancho-, y el pecado sea sordo, que siempre he oído
Coincidence #222: - Déjese deso, señor -dijo Sancho-: viva la gallina, aunque con su pepita,
Coincidence #223: Con esta licencia, dijo Sancho a los labradores, que estaban muchos
Coincidence #224: - Quiero el envite -dijo Sancho-, y échese el resto de la cortesía, y
Coincidence #225: - Pardiez -dijo Sancho-, que me ha cuadrado, y aun esquinado, tal género de
Coincidence #226: - ¡Menos cortesía; menos mudas, señora dueña -dijo Sancho-; que por Dios que
Coincidence #227: - Bien pudiera el Amor -dijo Sancho- depositarlos en los de mi asno, que yo
Coincidence #228: - Eso creo yo muy bien -dijo Sancho-, que esto del morirse los enamorados es
Coincidence #229: en la cual dijo Sancho tantos donaires y tantas malicias, que dejaron de
Coincidence #230: - Mándote yo -dijo Sancho-, pobre doncella, mándote, digo, mala ventura,
Coincidence #231: - Dese modo -dijo Sancho-, ¡a la mano de Dios, y lluevan azotes!
Coincidence #232: - Yo apostaré -dijo Sancho- que antes de mucho tiempo no ha de haber
Coincidence #233: - Sin duda -dijo Sancho- que vuestra merced debe de estar encantado, como
Coincidence #234: - Estraño es vuesa merced -dijo Sancho-. Presupongamos que esta liebre es
Execution time: 5 seconds and 730906 microseconds.
felipe@DESKTOP-4PEPH5C: /mnt/d/GitHub/SistemasOperativos/Pry1_SO-Gerp$
```

Figura 14: Resultado de la Prueba II, con 15 procesos.

5.2.4. Resultados

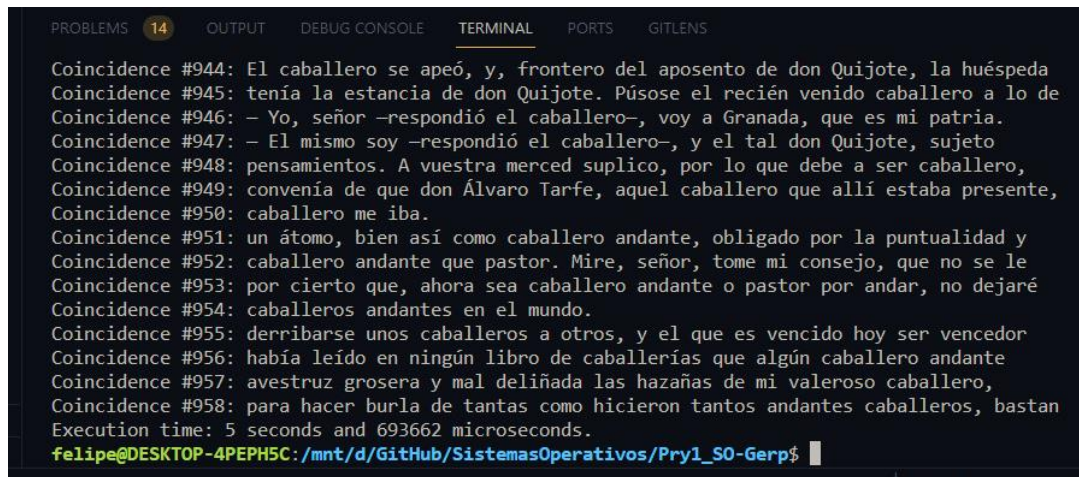
Se sugiere una cantidad óptima de procesos cercana a 5. En este caso, el tiempo de ejecución del programa, aumenta según la cantidad de procesos utilizada.

5.3. Prueba III

Se buscan las palabras “caballero”, “caballeros”, “Caballero” y “Caballeros” en el libro de Don Quijote.

5.3.1. 5 procesos

5.693 segundos.

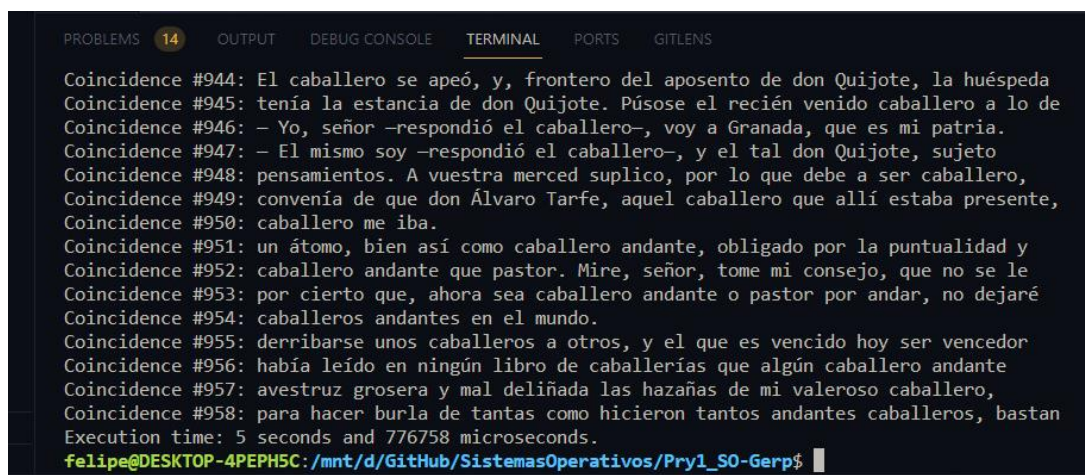


```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Coincidence #944: El caballero se apeó, y, frontero del aposento de don Quijote, la huésped
Coincidence #945: tenía la estancia de don Quijote. Púsose el recién venido caballero a lo de
Coincidence #946: – Yo, señor –respondió el caballero–, voy a Granada, que es mi patria.
Coincidence #947: – El mismo soy –respondió el caballero–, y el tal don Quijote, sujeto
Coincidence #948: pensamientos. A vuestra merced suplico, por lo que debe a ser caballero,
Coincidence #949: convenía de que don Álvaro Tarfe, aquel caballero que allí estaba presente,
Coincidence #950: caballero me iba.
Coincidence #951: un átomo, bien así como caballero andante, obligado por la puntualidad y
Coincidence #952: caballero andante que pastor. Mire, señor, tome mi consejo, que no se le
Coincidence #953: por cierto que, ahora sea caballero andante o pastor por andar, no dejaré
Coincidence #954: caballeros andantes en el mundo.
Coincidence #955: derribarse unos caballeros a otros, y el que es vencido hoy ser vencedor
Coincidence #956: había leído en ningún libro de caballerías que algún caballero andante
Coincidence #957: avestruz grosera y mal deliñada las hazañas de mi valeroso caballero,
Coincidence #958: para hacer burla de tantas como hicieron tantos andantes caballeros, bastan
Execution time: 5 seconds and 693662 microseconds.
felipe@DESKTOP-4PEPH5C: /mnt/d/GitHub/SistemasOperativos/Pry1_S0-Gerp$
```

Figura 15: Resultado de la Prueba III, con 5 procesos.

5.3.2. 10 procesos

5.776 segundos.

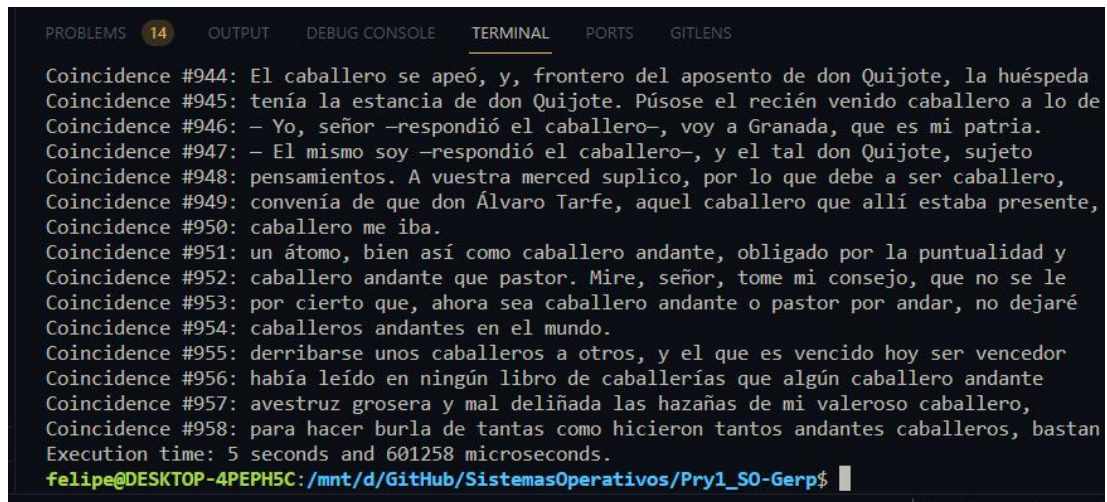


```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Coincidence #944: El caballero se apeó, y, frontero del aposento de don Quijote, la huésped
Coincidence #945: tenía la estancia de don Quijote. Púsose el recién venido caballero a lo de
Coincidence #946: – Yo, señor –respondió el caballero–, voy a Granada, que es mi patria.
Coincidence #947: – El mismo soy –respondió el caballero–, y el tal don Quijote, sujeto
Coincidence #948: pensamientos. A vuestra merced suplico, por lo que debe a ser caballero,
Coincidence #949: convenía de que don Álvaro Tarfe, aquel caballero que allí estaba presente,
Coincidence #950: caballero me iba.
Coincidence #951: un átomo, bien así como caballero andante, obligado por la puntualidad y
Coincidence #952: caballero andante que pastor. Mire, señor, tome mi consejo, que no se le
Coincidence #953: por cierto que, ahora sea caballero andante o pastor por andar, no dejaré
Coincidence #954: caballeros andantes en el mundo.
Coincidence #955: derribarse unos caballeros a otros, y el que es vencido hoy ser vencedor
Coincidence #956: había leído en ningún libro de caballerías que algún caballero andante
Coincidence #957: avestruz grosera y mal deliñada las hazañas de mi valeroso caballero,
Coincidence #958: para hacer burla de tantas como hicieron tantos andantes caballeros, bastan
Execution time: 5 seconds and 776758 microseconds.
felipe@DESKTOP-4PEPH5C: /mnt/d/GitHub/SistemasOperativos/Pry1_S0-Gerp$
```

Figura 16: Resultado de la Prueba III, con 10 procesos.

5.3.3. 15 procesos

5.601 segundos.



```
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Coincidence #944: El caballero se apeó, y, frontero del aposento de don Quijote, la huésped
Coincidence #945: tenía la estancia de don Quijote. Púsose el recién venido caballero a lo de
Coincidence #946: – Yo, señor –respondió el caballero–, voy a Granada, que es mi patria.
Coincidence #947: – El mismo soy –respondió el caballero–, y el tal don Quijote, sujeto
Coincidence #948: pensamientos. A vuestra merced suplico, por lo que debe a ser caballero,
Coincidence #949: convenía de que don Álvaro Tarfe, aquel caballero que allí estaba presente,
Coincidence #950: caballero me iba.
Coincidence #951: un átomo, bien así como caballero andante, obligado por la puntualidad y
Coincidence #952: caballero andante que pastor. Mire, señor, tome mi consejo, que no se le
Coincidence #953: por cierto que, ahora sea caballero andante o pastor por andar, no dejaré
Coincidence #954: caballeros andantes en el mundo.
Coincidence #955: derribarse unos caballeros a otros, y el que es vencido hoy ser vencedor
Coincidence #956: había leído en ningún libro de caballerías que algún caballero andante
Coincidence #957: avestruz grosera y mal deliñada las hazañas de mi valeroso caballero,
Coincidence #958: para hacer burla de tantas como hicieron tantos andantes caballeros, bastan
Execution time: 5 seconds and 601258 microseconds.
felipe@DESKTOP-4PEPH5C: /mnt/d/GitHub/SistemasOperativos/Pry1_S0-Gerp$
```

Figura 17: Resultado de la Prueba III, con 15 procesos.

5.3.4. Resultados

Con la tercera prueba, se sugiere una cantidad óptima de procesos cercana (o aún mayor) a 15. Para esta prueba, entre mayor era la cantidad de procesos utilizados, el tiempo de ejecución era menor.

6. Conclusiones

El proyecto ha demostrado de manera concluyente la eficacia de la ejecución colaborativa de tareas utilizando múltiples procesos para mejorar el rendimiento de la búsqueda de patrones en archivos grandes. A través de una serie de pruebas de rendimiento, se ha evaluado el impacto de la cantidad de procesos en el tiempo de ejecución del programa. Los resultados sugieren que la cantidad óptima de procesos puede variar según el escenario de búsqueda y el tamaño del archivo.

En las pruebas se observó cómo el tiempo de ejecución no aumentó ni disminuyó de manera lineal con el número de procesos. En algunos casos, el aumento en la cantidad de procesos condujo a un mejor rendimiento, mientras que en otros casos, un número excesivo de procesos resultó en una degradación del rendimiento.

Además de los resultados cuantitativos, el proyecto proporcionó valiosas lecciones sobre la sincronización y comunicación efectiva entre procesos en un entorno Unix. La implementación de un pool de procesos estático y las técnicas utilizadas para coordinar y sincronizar la búsqueda en el archivo, considerando las condiciones iniciales, representaron el mayor desafío de la creación del programa. En general, el proyecto se completó con éxito y cumpliendo con los objetivos, finalizando con satisfacción su desarrollo por ambos autores. La realización del presente proyecto significa una comprensión sólida acerca de la forma en la que funcionan y se manipulan los procesos de forma colaborativa.