Sistemas Operativos 2/2017
Laboratorio 2

Profesores:
Cristóbal Acosta (cristobal.acosta@usach.cl)
Miguel Cárcamo (miguel.carcamo@usach.cl)
Fernando Rannou (fernando.rannou@usach.cl)
Ayudantes:
Marcela Rivera (marcela.rivera.c@usach.cl)
Néstor Mora (nestor.mora@usach.cl)

I. Objetivos Generales

Este laboratorio tiene como objetivo aplicar los conceptos de procesos y hebras, mediante el soporte de un sistema operativo basado en el núcleo Linux y el lenguaje de programación C.

II. Objetivos Específicos

- Recibir como parámetros de entradas: nombre del archivo de entrada y de salida, número de hebras a crear, cantidad de palabras, largo y ancho de matriz y finalmente una bandera que indica si se deben mostrar los resultados por pantalla.
- Hacer un programa el cual permita crear una sopa de letras, el cual debe contener palabras insertadas en forma horizontal. Las palabras deben ser válidas, es decir, se debe asegurar la sincronización de las hebras para que éstas puedan insertar las palabras sin corrupciones.
- Cada hebra será encargada de insertar en la sopa de letras un subconjunto de palabras.
- Escribir en un archivo de texto la respuesta final. Es decir, una sopa de letras.

III. Conceptos

III.A. Hebras

Los hilos POSIX, usualmente denominados pthreads, son un modelo de ejecución que existe independientemente de un lenguaje, además es un modelo de ejecución en paralelo. Éstos permiten que un programa controle múltiples flujos de trabajo que se superponen en el tiempo.

Para poder utilizar una hebra, es necesario incluir la librería **pthread.h**. Por otro lado, dentro de la función main, se debe instanciar la variable de referencia a la hebra, para esto se utiliza el tipo de dato **pthread_t** acompañado de la hebra.
Luego, se debe tener una función de la forma:

void * function (void * params)

La cual recibe parámetros del tipo void, por lo cual es necesario castear el o los parámetros de entrada para así poder utilizarlos sin problemas.

Algunas funciones para manejar las hebras son:

- **pthread_create**: Función que crea una hebra. Recibe como parámetros de entrada:
  - La variable de referencia a la hebra que desea crear.
  - Los atributos de éste, los cuales no es obligación de modificar, por lo que en caso de no querer hacerlo, simplemente se deja NULL.
  - El nombre de la función que la hebra ejecutará (la cual debe cumplir con la descripción antes mencionada)
  - Por último, los parámetros de entrada (de la función que se ejecutará) previamente casteados.

  Un ejemplo sería:

  while(i < numeroHebras)
  {
  pthread_create(&hilo[i], NULL, ubicar, (void *) palabras[]);
  i++;
  }

- **pthread_join:** Función donde la hebra que la ejecuta, espera por la hebra que se ingresa por parámetro de entrada.

  Un ejemplo sería:

  while(i < numeroHebras)
  {
  pthread_join(hilo[i],NULL);
  i++;
  }

- **pthread mutex init:** Función que inicializa un mutex, pasando por parámetros la referencia al mutex, y los atributos con que se inicializa.

  Para inicializar la estructura es:

  while(i < numeroHebras)
  {
  pthread_mutex_init(hilo[i].mutexHilo, NULL);
  i++;
  }

  Donde mutexHilo, es un atributo de la estructura de la hebra.

  Para implementar una solución a la sección crítica, se utiliza la función **pthread_mutex_lock()**. Ésta recibe como parámetros la variable que se desea bloquear para el resto de hebras. Un ejemplo de uso sería:

  pthread_mutex_lock(lista[x].mutexPos);

  Donde lista[x] es la posición de una lista. Por otro lado, mutexPos es una variable de tipo **pthread_mutex_t**, la cual permite hacer uso de mutex en una variable.

  Finalmente, para liberar una sección crítica se utiliza la función **pthread_mutex_unlock()**. Ésta recibe como parámetro de entrada, la variable que se desea desbloquear. Su implementación es la siguiente:

  pthread_mutex_unlock(lista[x].mutexPos);

IV. Enunciado

Se pide crear un programa el cual permita construir una sopa de letras, de tamaño NxM. Sin embargo, a través del uso de hebras, se pide insertar las palabras (en la matriz) de forma horizontal.

Para cumplir con estos objetivos, es necesario seguir los siguientes pasos:

- Recibir parámetros de entrada a través del uso de getopt. Los parámetros de entrada son:

- -i: nombre del archivo de entrada
- -h: numero de hebras
- -c: cantidad de palabras
- -n: ancho de matriz
- -m: largo de matriz
- -s: nombre del archivo de salida
- -d: bandera que indica si se deben mostrar los resultados por pantalla

Un ejemplo de como se deben recibir los parámetros de entrada:
>> make
>> ./laboratorio1 -i archivo_entrada.txt -h 7 -c 15 -n 10 -m 10 -s archivo_salida.txt

- Crear una matriz, la cual debe tener sus posiciones inicializadas con el caracter espacio. Esto es hecho dentro del hilo main.

  PISTA: para crear la matriz utilizar memoria dinámica, a través del uso de la función malloc.

- Crear una estructura de datos para la hebras, ésta debe contener: un id, las coordenadas x,y, una lista de palabras (el subconjunto de palabras que debe insertar) y un mutex para proveer un método de sincronización. Para usar mutex, se debe declarar una variable de tipo **pthread_mutex_t**.

- Leer un archivo de texto, el cual contiene las palabras a insertar. El formato de archivo sería:

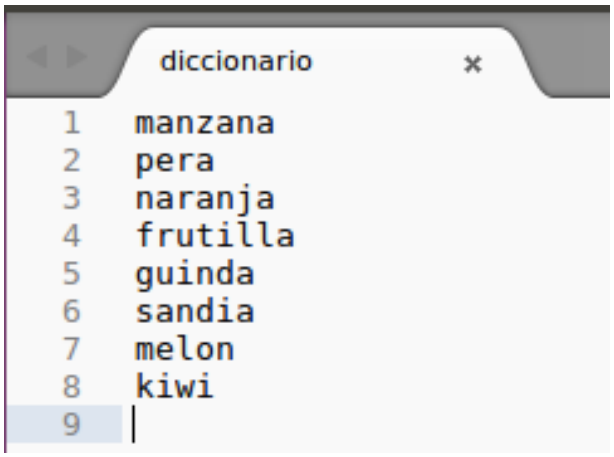

Figure 1. Archivo de entrada

- A partir del archivo de texto, asignar el subconjunto de palabras válidas correspondientes a cada hebra.

- Crear una función que permita insertar palabras horizontales en la matriz. Ésta función la llamaremos ubicar y su cabecera es la siguiente:

void ubicar((void *) id);

  Donde **id** es el identificador de la hebra.

  Cabe destacar que esta función asigna valores aleatorios para las coordenadas x,y. Es decir, se asignará una posición inicial de manera random.

  Esto implica que antes de insertar una palabra, se debe validar que la palabra cabe en la sopa de letras, es decir, para validar se debe comprobar si la posición inicial más el largo de la palabra a insertar, es menor a la cantidad total de columnas (valor M).

- Inicializar las hebras utilizando la función pthread_create, dirigiendo a la hebra a la función **ubicar**.

- Una vez insertadas todas las palabras válidas, se procede por rellenar los espacios vacíos con caracteres. Esto lo hace la hebra main.

  PISTA: Con la función join se esperan a todas las hebras.

- Finalmente se debe escribir en un archivo de texto la matriz resultante, es decir, la sopa de letras final. Éste último paso es hecho por la hebra principal.

  Un ejemplo de archivo de salida es:

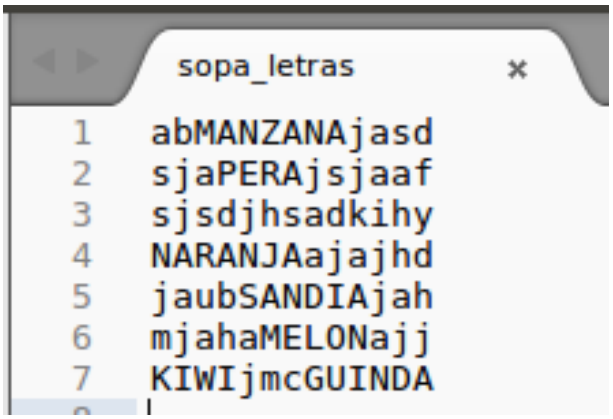

Figure 2. Archivo de salida

V. Entregables

Debe entregarse un archivo comprimido que contenga al menos los siguientes archivos:

1. *Makefile*: archivo make para que compile los programas.
2. uno o mas archivos con el proyecto (*.c, *.h)
3. Clara documentación, es decir, entregar códigos debidamente comentados.

El archivo comprimido debe llamarse: RUTESTUDIANTE1_RUTESTUDIANTE2.zip
Ejemplo: 19689333k_189628735.zip

VI. Fecha de entrega

Sábado 14 de Octubre hasta las 23:55 hrs a través de UsachVirtual.